

DPTTrace: Dual Purpose Trace for Exploitability Analysis of Program Crashes

Rohit Mothe (@rohitwas)

Senior Security Researcher

rohit.mothe *noSPAM* intel.com

Rodrigo Rubira Branco (@BSDDaemon)

Principal Security Researcher

rodrigo.branco *noSPAM* intel.com

Disclaimer

- ▶ We don't speak for our employer(duh!). All the opinions and information presented here is our responsibility (actually no one has seen this talk before today)
- ▶ **IMPORTANT: No, we are *not* part of the Intel Security Group (McAfee)**

Agenda

- Objectives
- Current state of Affairs or Security Today
- Taint Analysis Introduction
- Our approach – Dual Tracing
- Comparison with other ideas
- Demos
- Limitations
- Future

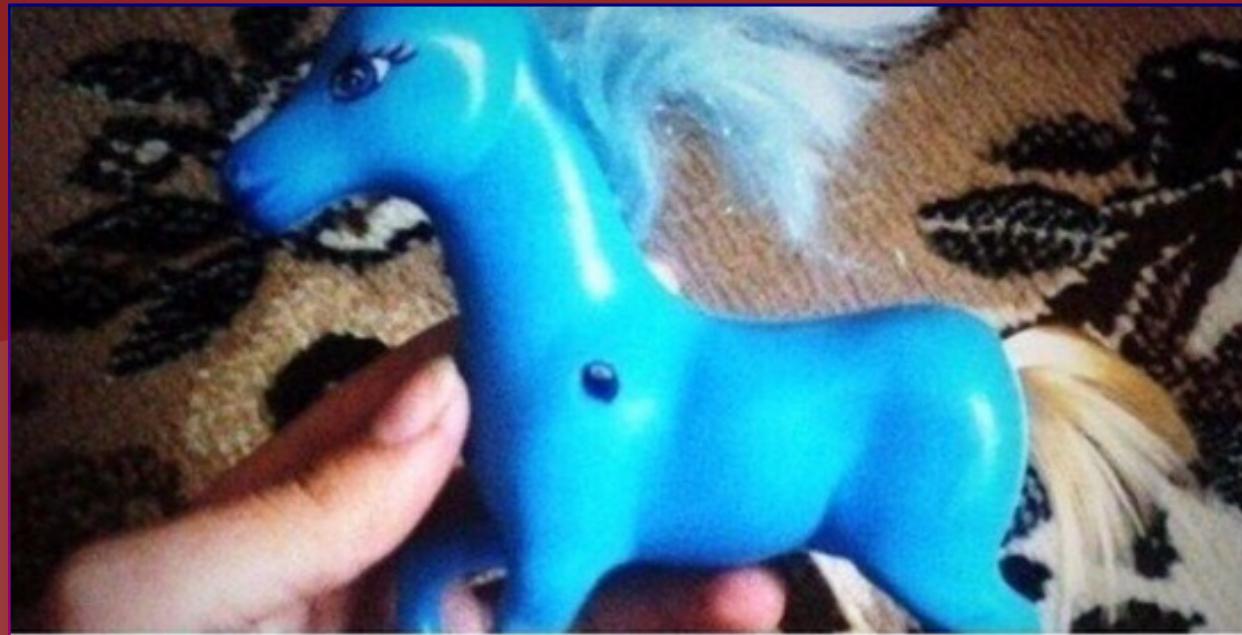
Objectives

- Contribute towards improving the state of the art in crash analysis
- Automate laborious/repetitive parts, but still requiring skilled exploit writer/analyst
- Discuss hybrid usage of techniques and the mixture of automation with manual analysis

Current State of Affairs

- ▶ Buggy programs deployed on critical servers
- ▶ Rapidly-evolving threats, attackers and tools (exploitation frameworks)
- ▶ Lack of developers training, resources and people to fix problems and create safe code
- ▶ **That's why we are here today, right?**

tl; dr



thnx Marcio !

Taint Analysis for Program Crashes

- ▶ Through our work we try to answer two fundamental questions:
 - ▶ Are the input operands in the attacker's control?
 - ▶ And if so, is the forward execution providing a primitive that is good for an attacker?
- ▶ Taint Analysis is one specific kind of program flow analysis and we use it to define the influence of external data (attacker's controlled data) over the analyzed application
- ▶ Since the information flows, or is copied to, or influences other data there is a need to follow this influence in order to determine the control over specific areas (registers, memory locations). This is a requirement in order to determine exploitability

History and Lore- Backward-Taint

- ▶ Original Motivation: Complex client-side vulnerability in a closed (at the time) file format
- ▶ Extended Motivation: Trying to better analyse hundreds of thousands of bugs in Microsoft Word (search for Ben Nagy, Coseinc)
- ▶ Initial version integrated with a fuzzer, only for Linux (showed in 2011 at Troopers)
- ▶ Ported version for Solaris to analyze a vulnerability released by Secunia in the same software RISE Security released a vulnerability a month before (also circa 2011)
- ▶ Thanks to Julio Auto's parallel research in the same field, a Windows version was created (extended in this research)

History and Lore-Forward-Taint

- ▶ Original Motivation: Triaging submissions in a vulnerability purchase program is hard. Many submissions lack a complete exploit but still might have real value
- ▶ Extended Motivation: Categorizing fuzzing crashes is a pain (NOT bang(!) exploitable categorizing)
- ▶ Manual process includes lots of repetitive steps
- ▶ Automation is key. Certain classes repeat themselves (such as UAF)
- ▶ ‘Prototyping Exploitation’ in such cases is both cost and time effective. Also a more reasonable and simpler ‘automatable’ problem than automating exploit writing for all classes of bugs. Prototype or GTFO!

Existing Solutions - What we aren't

- ▶ !exploitable
 - ▶ Tries to classify unique issues (crashes appearing through different code paths, machines involved in testing, and in multiple test cases). Group the crashes for analysis
 - ▶ Quickly prioritizes issues (since crashes appear in thousands, while analysis capabilities are VERY limited)
 - ▶ Classic, timeless!
- ▶ Spider Pig
 - ▶ Created by Piotr Bania
 - ▶ Not available for testing, but from the paper: It is much more advanced than the provided tool (but well, it is not available?)
 - ▶ Virtual Code Integration (or Dynamic Binary Rewriting)
 - ▶ Disputable Objects: Partially controlled data is analyzed using the parent data
- ▶ Taint Bochs
 - ▶ Used for tracking sensitive data lifecycle in memory

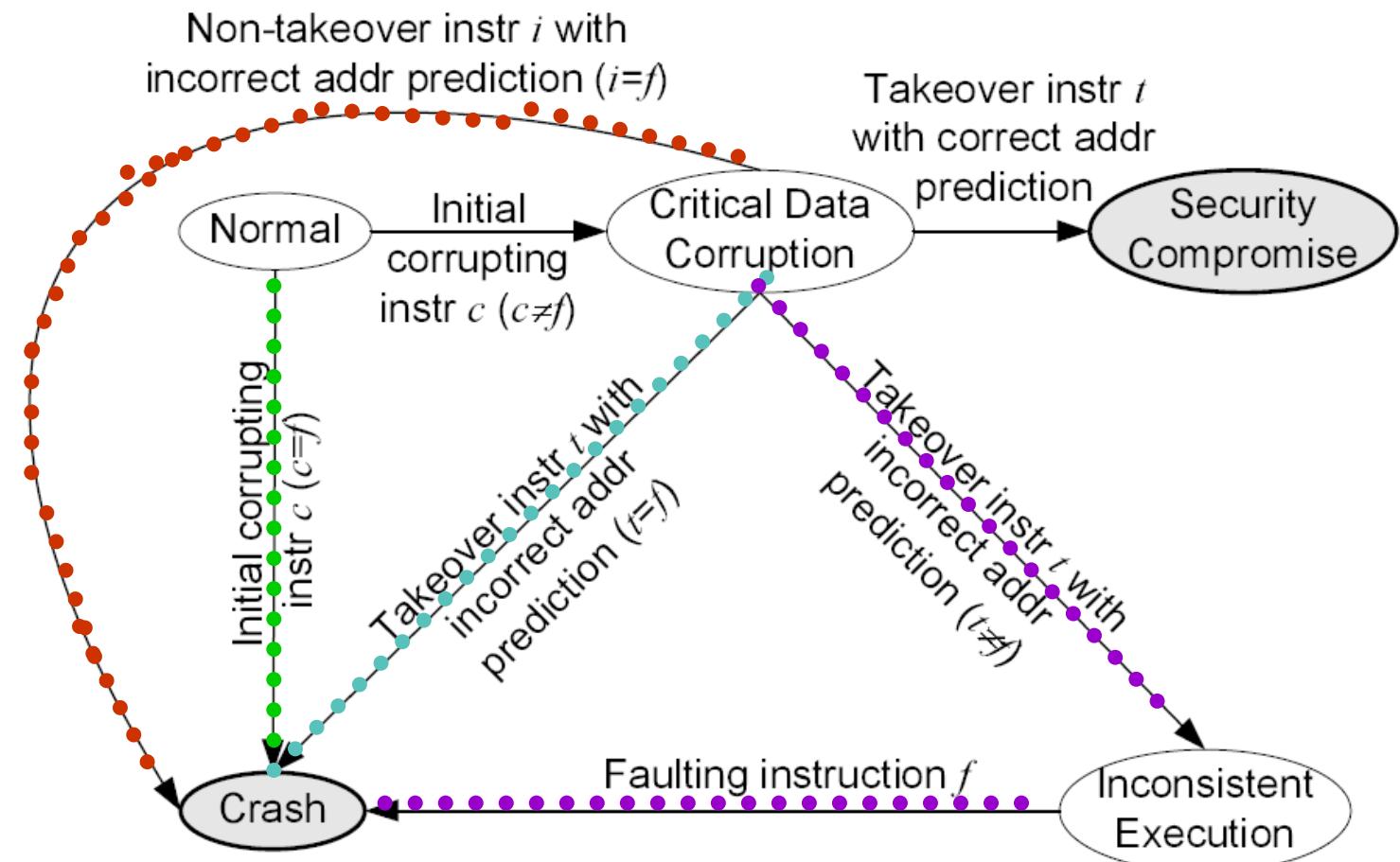
Existing Solutions - What we aren't

contd..

- ▶ Taint Check
 - ▶ Uses DynamicRIO or Valgrind
 - ▶ Taint Seed: Defining the tainted values (data coming from the network for example)
 - ▶ Taint Tracker: Tracks the propagation, Taint Assert: Alert about security violations
 - ▶ Used while testing software to detect overflow conditions, does not really help in the exploit creation
- ▶ Bitblaze
 - ▶ An amazing platform for binary analysis
 - ▶ Provides better classification of exploitability (Charlie Miller talk in BH)
 - ▶ Can be used as base platform for the provided solution (VINE)
- ▶ Moflow Framework
 - ▶ Cisco Talos. Tools built on CMU's BAP framework.
 - ▶ sliceflow- post-crash graph back taint slicer
 - ▶ Post-crash forward symbolic emulator looking for more exploitable conditions
 - ▶ Pretty neat and advanced!

State Transition for Memory Corruption

- Case 1 (green): Format String
- Case 2 and 3 (red and blue): buffer overflow
- Case 4 (purple): unpredictable



Source:

Automatic Diagnosis and Response to Memory Corruption Vulnerabilities

c: corrupting instruction
t: takeover instruction
f: faulting instruction

Moving Backward

- ▶ Legitimate assumption:
 - ▶ To change the execution of a program illegitimately we need to have a value being derived from the attacker's input (which we call: controlled by the attacker)
- ▶ String sizes and format strings should usually be supplied by the code itself, not from external, un-trusted inputs
- ▶ Any data originated from or arithmetically derived from un-trusted source must be inspected

Analyzing Taint

- ▶ Tainted data: Data from un-trusted source
- ▶ Keeps track of tainted data (from un-trusted source)
- ▶ Monitors program execution to track how tainted attribute propagates
- ▶ Detect when tainted data is used in sensitive way

Taint Propagation

- ▶ When a tainted location is used in such a way that a value of other data is derived from the tainted data (like in mathematical operations, move instructions and others) we mark the other location as tainted as well
- ▶ The transitive relation is:
 - ▶ If information A is used to derive information B:
 - ▶ $A \rightarrow t(B)$ -> Direct flow
 - ▶ If B is used to derive information C:
 - ▶ $B \rightarrow t(C)$ -> Direct flow
 - ▶ Thus: $A \rightarrow t(C)$ -> Indirect flow
- ▶ Due to the transitive nature, you can analyze individual transitions or the whole block ($A \rightarrow t(C)$)

Location

- ▶ A location is defined as:
 - ▶ Memory address and size
 - ▶ Register name (we use the register entirely, not partially -> thus %al and %eax are the same)
 - ▶ When setting a register, we set it higher (setting %al as tainted will also taint %eax)
 - ▶ When clearing a register, we clear it lower
- ▶ To keep track over bit operations in a register it is important to taint the code-block level of a control flow graph
 - ▶ This creates extra complexity due to the existence of the flow graph and data flow dependencies graph
 - ▶ The dependencies graph represents the influence of a source data in the operation been performed

Flows

- ▶ Explicit flow:
 - ▶ `mov %eax, A`
- ▶ Implicit flow:
 - ▶ `If (x == 1) y=0;`
- ▶ Conditional statements require a special analysis approach:
 - ▶ In our case, we are analyzing the trace of a program (not the program itself, but only what was executed during the debugging section)
 - ▶ We have two different analysis step: tracing and analysis

Special Considerations

- ▶ Partial Tainting: When the untrusted source does not completely control the tainted data
- ▶ Tainting Merge: When there are two different untrusted sources being used to derive some data
- ▶ Data
 - ▶ In Use: when it is referenced by an operation
 - ▶ Defined: when the data is modified

Inheritance problems

Problem: state explosion for binary operations !

Application

```
mov %eax <- A  
mov B <- %eax
```

```
add %ebx <- D
```

Propagation Tracking

$$\begin{aligned} \text{taint}(\%eax) &= \text{taint}(A) \\ \text{taint}(B) &= \text{taint}(\%eax) \end{aligned}$$
$$\text{taint}(\%ebx) |= \text{taint}(D)$$

Inheritance Tracking

%eax inherits from A
B inherits from %eax

insert D into %ebx's
inherit-from list

Events

Rare

e.g., malloc/free, system calls

Frequent

e.g., memory access,
data movement

Tracking Instructions

- ▶ Pure assignments: Easy to track
 - ▶ If a tainted location is used to define another location, this new location will be tainted
- ▶ Operations over strings are tainted when:
 - ▶ They are used to calculate string sizes using a tainted location
 - ▶ `a = strlen(tainted(string));`
 - ▶ Since the 'string' is tainted, we assume the attacker controls 'a'
 - ▶ Search for some specific char using a tainted location, defining a flag if found or not found
 - ▶ `pointer = strchr(tainted(string), some_char);`
 - ▶ If (pointer) flag=1;
 - ▶ 'flag' is tainted if the attacker controls 'string' or 'some_char'

Tracking Instructions

contd...

- ▶ Arithmetic instructions with at least one tainted data usually define tainted results
- ▶ Those arithmetic instructions can be simplified to map to boolean operations and then the following rules applies

Eflags and Flow Information

- ▶ The eflags register can also be tainted to monitor flags conditions influencing in operations (and flow)
- ▶ In the presented approach, conditional branches are taken care due to the trace generated by the WinDBG plugin (single-stepping)

Backward Taint Analysis

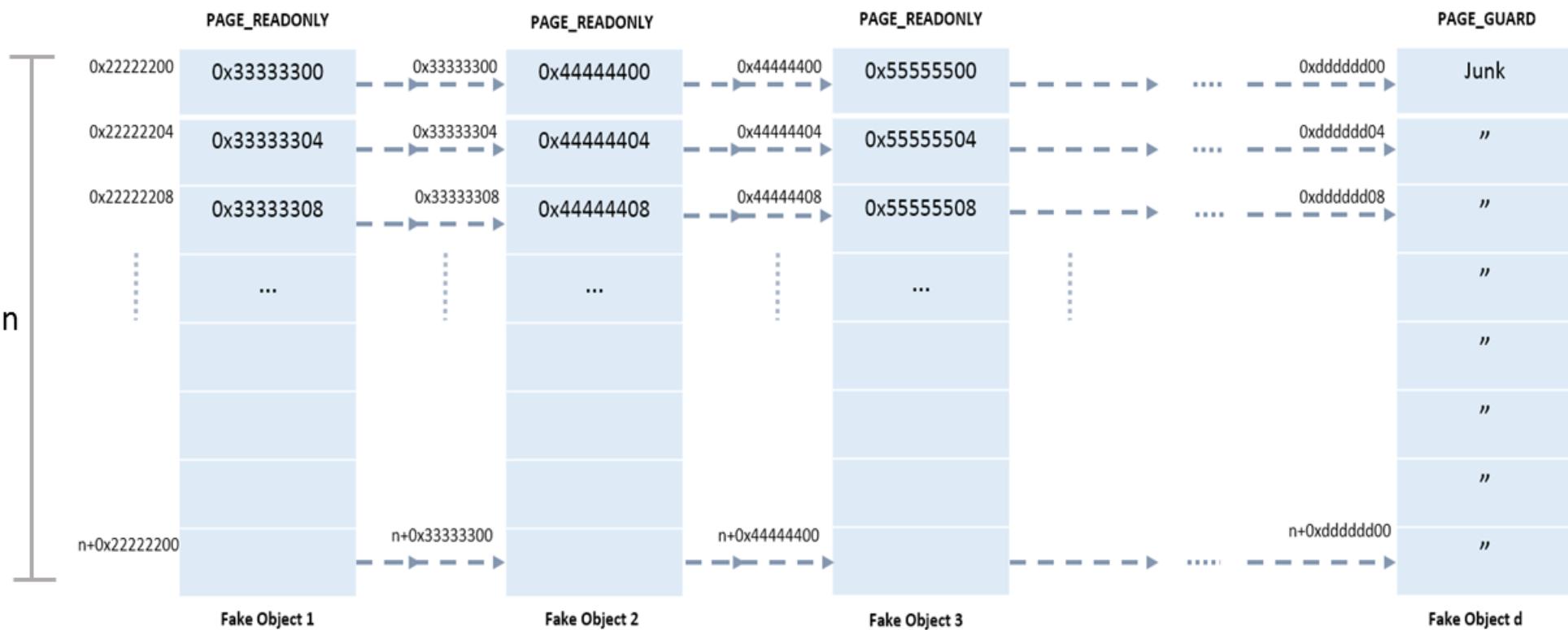
- ▶ Divide the analysis process in two parts:
 - ▶ A trace from a good state to the crash (incrementally dumped to a file) -> Gather substantial information about the target application when it receives the input data, which is formally named 'analysis'
 - ▶ Analysis of the trace file -> Formally defined as 'verification' step, where the conclusive analysis is done

Forward Taint Analysis

24

- To see what kind of primitives (read/write/calls) are available we ‘prototype’ input control and allocate a fake object structure in memory such that the program can continue from the point of the crash to other code paths.
- The property of such fake memory structure should guarantee to a reasonable extent that any memory references (like virtual function tables or other object pointers) will be resolved including memory address references that are additive or subtractive to the faulting address(which is already assumed controllable).
- In essence one could imagine it as simulating the reallocation of a fake object ‘within’ the debugger in a use-after-free situation and continuing the exception. Or allocating an adjacent object in an out of bounds access violation, etc.

Fake Memory Structure Sample



n - size of each object in bytes

d - depth/number of fake objects in the linked list chain

Forward Logic

- ▶ In the debugger you see a seemingly non exploitable read AV (access violation).
 - ▶ Example: `mov eax ,[ecx]` ; (ecx is supposed here to be a pointer to attacker controlled memory.)
 - ▶ You allocate a chunk of memory within the process (preferably the size of the memory pointed to by ecx to mimic an accurate freed block control using heap massaging,feng-shui)
- ▶ The permissions of all memory blocks in a linked list chain are read-only. So any attempt to write/execute on any of the values within the memory blocks would cause an exception later and that shows evidence of exploitability
- ▶ Now manually change the ecx value in the crash to point to the address of the root of this linked list which is the root of the chain of memory blocks pointing to one another
- ▶ Continue the program execution and it will continue from the point of crash with the modified value of ecx.

Need for Intermediate Languages

- ▶ Assembly instructions have explicit operands, which are easy to deal with, and sometimes implicit operands:
 - ▶ Instruction: push eax
 - ▶ Explicit operand: eax
 - ▶ What it really does?
 - ▶ $ESP = ESP - 4$ (a subtraction)
 - ▶ $SS:[ESP] = EAX$ (a move)
 - ▶ Here we have ESP and SS as implicit operands
 - ▶ Tks to Edgar Barbose for this great example!

Implementing the Tracer

- ▶ Instead of using an intermediate language, we play straight with the debugger interfaces (WinDBG). Windbg or GTFO!
- ▶ The tracer stores some useful information, like effective addresses and data values and also simplifies the instructions for easy parsing:
 - ▶ CMPXCHG r/m32, r32 -> 'Compare EAX with r/m32. If equal, ZF is set and r32 is loaded into r/m32. Else, clear ZF and load r/m32 into AL'
 - ▶ Such an instruction creates the need for conditional taints, since by controlling %eax and r32 the attacker controls r/m32 too.

Implementation Details

- ▶ Instead of using an intermediate language, we play straight with the debugger interfaces (WinDBG). Windbg or GTFO!
- ▶ Trace File Contains:
 - ▶ Mnemonic of the instruction and operands
 - ▶ Dependences for the source operand
 - ▶ Eg: Elements of an indirectly addressed memory
 - ▶ This creates a tree of the dataflow, with a root in the crash instruction
- ▶ The verification (GUI and cmdline program) step reads this file and:
 - ▶ Search this tree using a BFS algorithm
- ▶ Forward step uses the debugger interfaces for the memory allocation and forward execution



30

Program Execution Timeline

31

Trace! Check Taint! Forward Execution

Program start

Trace! Check Taint! Forward Execution

Initial Crash, AV

Trace! Check Taint! Forward Execution

Exception → Constraint 1

...
...
...
...
...
...
...
...
...
...
...
...

Trace! Check Taint!

Exception → Constraint n

Exploitable Primitive?! Profit!

OR march on, down to hell ☹

Theoretical Example

- ▶ 1-) mov edi, 0x1234 ; dst=edi, src=0x1234
- ▶ 2-) mov eax, [0xABCD] ; dst=eax, src=ptr 0xABCD ; Note 0xABCD is evil addr
- ▶ 3-) lea ebx, [eax+ecx*8] ; dst=ebx, src=eax, srcdep1=ecx
- ▶ 4-) mov [edi], ebx ; dst=ptr 0x1234, src=ebx
- ▶ 5-) mov esi, [edi] ; dst=esi, src=ptr 0x1234, srcdep1=edi
- ▶ 6-) mov edx, [esi] ; Crash!!!

Theoretical Example

contd..

- ▶ 6-) Where does [esi] come from?
- ▶ 5-) [edi] is moved to esi, where edi comes from and what does exist in [edi]?
- ▶ 4-) [edi] receives ebx and edi is defined in 1-) from a fixed value
- ▶ 3-) ebx comes from a lea instruction that uses eax and ecx
- ▶ 2-) eax receives a value controlled by the attacker
- ▶ ... ecx is out of the scope here :)

Assumptions & Challenges

- ▶ Since we only use the trace information, if the crash input data does not force a flow, we can't see the influence of the input over this specific flow data
- ▶ To solve that:
 - ▶ If a jmp is dependent of a flag, the attacker controls branch decision
 - ▶ Control over a branch means tainted EIP
 - ▶ To define the value of EIP, consider:
 - ▶ The address if the jump is taken
 - ▶ The address of the next instruction (if the jump is not taken)
 - ▶ The value of the interesting flag register (0 or 1)
 - ▶ Then: $\%eip \leftarrow (\text{address of the next instruction}) + \text{value of the register flag} * (|\text{address if jump is taken} - \text{address of the next instruction}|)$

Forward Analysis

- ▶ The method here was conceived originally to help determine whether crashes for potential UAF (Use-After-Free) bugs in browsers are exploitable or not
- ▶ UAFs in browsers or any significantly large programs for that matter are often hard to analyze for exploitability and typically involve following varied code paths in the control flow to find a write access violation/potential code redirection using indirect calls
- ▶ The idea is not just limited to UAFs though
- ▶ After input control(first part of the problem) has been determined, the next logical step is to gauge what can be done with it.

Command-line options

```
0:018> .load dptracer
0:018> !dptrace_help
Dual Purpose Tracer v1.0 Alpha - Copyright (C) 2008-2016
License: This software was created as companion to a Black Hat Presentation.
Developed by Rodrigo Rubira Branco (BSDaemon) <rodrigo@kernelhacking.com> and Rohit Mothe <rohitwas@gmail.com> (alphabetical order of names)
Heavily based on VDT-Tracer by Julio Auto and Rodrigo Branco

!dptrace_trace <filename> - trace the program until a breakpoint or exception and save the trace
    in a file to be later consumed by the Visual Data Tracer GUI.
!dptrace_forward <n(required)> -s<(required)> -p<(OPTIONAL)> - forward analysis, either no arguments or all mandatory
!dptrace_analyzer <analyzer_filepath> <trace_filepath> <close_gui> <controlled_ranges> <instr_index>
!dptrace_analyzer_help - help to the !dptrace_run_analyzer command
!dptrace_forward_help - help to the !dptrace_forward command
!dptrace_help - this help screen
```

```
*** ERROR: Symbol file could not be found. Defaulted to export symbols for C:\Windows\system32\ntdll.dll -
ntdll!DbgBreakPoint:
77c140f0 cc          int     3
0:016> .load dptracer
```

```
0:016> !dptrace_trace C:\Desktop\LogFiles\Log.vdt |
```

Analyzer

```
"C:\Program Files (x86)\Microsoft Office\Office15\EXCEL.EXE" - WinDbg:10.0.10586.567 X86
File Edit View Debug Window Help
Command - "C:\Program Files (x86)\Microsoft Office\Office15\EXCEL.EXE" - WinDbg:10.0.10586.567 X86

0:000> !dptrace_analyzer """C:\\\\Users\\\\rrbranco\\\\Desktop\\\\Black Hat 2016\\\\DPTrace-BlackHat 2
Args: ""C:\\Users\\rrbranco\\Desktop\\Black Hat 2016\\DPTrace-BlackHat 2016\\Debug\\DPTRACE-GUI.exe"
0:000> .unload dptrace-tracer
Unloading dptrace-tracer extension DLL
0:000> .load dptrace-tracer
0:000> !dptrace_analyzer """C:\\\\Users\\\\rrbranco\\\\Desktop\\\\Black Hat 2016\\\\DPTrace-BlackHat 2
Args: ""C:\\Users\\rrbranco\\Desktop\\Black Hat 2016\\DPTrace-BlackHat 2016\\Debug\\DPTRACE-GUI.exe"

Opening file: C:\\Users\\rrbranco\\Desktop\\Black Hat 2016\\DPTrace-BlackHat 2016\\Sample_output\\dpt
Processing file...
Number of instrs (and instruction to check taint of): 124116 124115
Range Start: 0x80000 Range End: 0x81000
Range Start: 0x10000 Range End: 0x11000
Instruction: 775bb5b7 8b9264040000      mov      edx,dword ptr [edx+464h] ds:002b:00381464=000000

Dumping instruction taint information:

instr->Src tainted: *00381464
instr->SrcDep1 tainted: edx

Closing GUI
```

Forward

```
0:009> !dptrace_forward help
!dptrace_forward:

[*] Two options of running-

    a) Simple Run with no arguments
        !dptrace_forward
        = > default number of objects n = 2, size s= 40, permissions are RW for first object, READONLY for all other

    b) If running with arguments to configure the run following rules apply -
        !dptrace_forward - n(required) - s(required) - p(OPTIONAL)
        = > number of fake objects(required), size of each object in bytes(required), page permissions(OPTIONAL)(0x0)

        Parameters n and s are parsed in decimal(base 10). however, if passing the 3rd parameter p, specify the const

<Optional>For passing 'p' use the following map -
    PAGE_EXECUTE      0x10
    PAGE_EXECUTE_READ 0x20
    PAGE_EXECUTE_READWRITE 0x40
    PAGE_EXECUTE_WRITECOPY 0x80
    PAGE_NOACCESS     0x01
    PAGE_READONLY     0x02 // defualt protection if run with no arguments.
    PAGE_READWRITE    0x04
    PAGE_WRITECOPY    0x08

</Optional>

0:009> !dptrace_forward 3 100 0x2

Allocated range is
3da0000-3da1000,3db0000-3db1000,3dc0000-3dc1000
```

Forward

```
0:012> dd 3eb0000
03eb0000 03ec0000 03ec0004 03ec0008 03ec000c
03eb0010 03ec0010 03ec0014 03ec0018 03ec001c
03eb0020 03ec0020 03ec0024 03ec0028 03ec002c
03eb0030 03ec0030 03ec0034 03ec0038 03ec003c
03eb0040 03ec0040 03ec0044 03ec0048 03ec004c
03eb0050 03ec0050 03ec0054 03ec0058 03ec005c
03eb0060 03ec0060 41414141 41414141 41414141
03eb0070 41414141 41414141 41414141 41414141
0:012> dd 03ec0000
03ec0000 03ed0000 03ed0004 03ed0008 03ed000c
03ec0010 03ed0010 03ed0014 03ed0018 03ed001c
03ec0020 03ed0020 03ed0024 03ed0028 03ed002c
03ec0030 03ed0030 03ed0034 03ed0038 03ed003c
03ec0040 03ed0040 03ed0044 03ed0048 03ed004c
03ec0050 03ed0050 03ed0054 03ed0058 03ed005c
03ec0060 03ed0060 41414141 41414141 41414141
03ec0070 41414141 41414141 41414141 41414141
0:012> dd 03ed0000
03ed0000 cccccccc cccccccc cccccccc cccccccc
03ed0010 cccccccc cccccccc cccccccc cccccccc
03ed0020 cccccccc cccccccc cccccccc cccccccc
03ed0030 cccccccc cccccccc cccccccc cccccccc
03ed0040 cccccccc cccccccc cccccccc cccccccc
03ed0050 cccccccc cccccccc cccccccc cccccccc
03ed0060 cccccccc cccccccc cccccccc cccccccc
03ed0070 cccccccc cccccccc cccccccc cccccccc
0:012> !vprot 3eb0000
BaseAddress: 03eb0000
AllocationBase: 03eb0000
AllocationProtect: 00000004 PAGE_READWRITE
RegionSize: 00001000
State: 00001000 MEM_COMMIT
Protect: 00000004 PAGE_READWRITE
Type: 00020000 MEM_PRIVATE
0:012>
BaseAddress: 03eb0000
AllocationBase: 03eb0000
AllocationProtect: 00000004 PAGE_READWRITE
RegionSize: 00001000
State: 00001000 MEM_COMMIT
Protect: 00000004 PAGE_READWRITE
Type: 00020000 MEM_PRIVATE
0:012> !vprot 03ec0000
BaseAddress: 03ec0000
AllocationBase: 03ec0000
AllocationProtect: 00000004 PAGE_READWRITE
RegionSize: 00001000
State: 00001000 MEM_COMMIT
Protect: 00000002 PAGE_READONLY
Type: 00020000 MEM_PRIVATE
0:012> !vprot 03ed0000
BaseAddress: 03ed0000
AllocationBase: 03ed0000
AllocationProtect: 00000004 PAGE_READWRITE
RegionSize: 00001000
State: 00001000 MEM_COMMIT
Protect: 00000002 PAGE_READONLY
Type: 00020000 MEM_PRIVATE
```



**WHAT IS
DEAD
MAY
NEVER
DIE**

Sample Analysis 1

```
Breakpoint 0 hit
eax=05bf3c38 ebx=00000400 ecx=05db7260 edx=05bf3c33 esi=002be28c edi=00000000
eip=638038d5 esp=002be174 ebp=002be1a4 iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00200202
AcroForm!D11UnregisterServer+0x1bd752:
638038d5 8b01          mov     eax,dword ptr [ecx]  ds:0023:05db7260=63bd8d68
```

We did a bit of cheating to avoid huge traces (from that point on til the crash, we would have traced more than 10 million instructions)

- CVE-2010-0188 – Adobe Reader Libtiff TIFFFetchShortPair Stack-based Buffer Overflow
- TIFF file embedded in a PDF where the IFD Entry has Tag ID (0x0129, 0x0141, 0x0212 or **0x0150**) and Tag Type 3 (short)
- The field data count of the TIFF file will be used as size (dc*2) to copy to a fixed buffer in stack

```
0:000> !dptrace_analyzer "\""C:\\Users\\rrbranco\\Desktop\\Black Hat 2016\\DPTrace-BlackHat 2016\\Debug\\DPTRACE-GUI.exe\" \"C:\\Users\\rrbranco\\Desktop\\Blac
Args: "\"C:\\Users\\rrbranco\\Desktop\\Black Hat 2016\\DPTrace-BlackHat 2016\\Debug\\DPTRACE-GUI.exe\" \"C:\\Users\\rrbranco\\Desktop\\Black Hat 2016\\DPTrace-BlackHat 2016\\S\\
```

```
Opening file: C:\\Users\\rrbranco\\Desktop\\Black Hat 2016\\DPTrace-BlackHat 2016\\Sample_output\\dptrace-test2.vdt
Processing file...
```

```
Instruction: 651c35ed 8b01      mov     eax,dword ptr [ecx]  ds:0023:062d9300=65591260
```

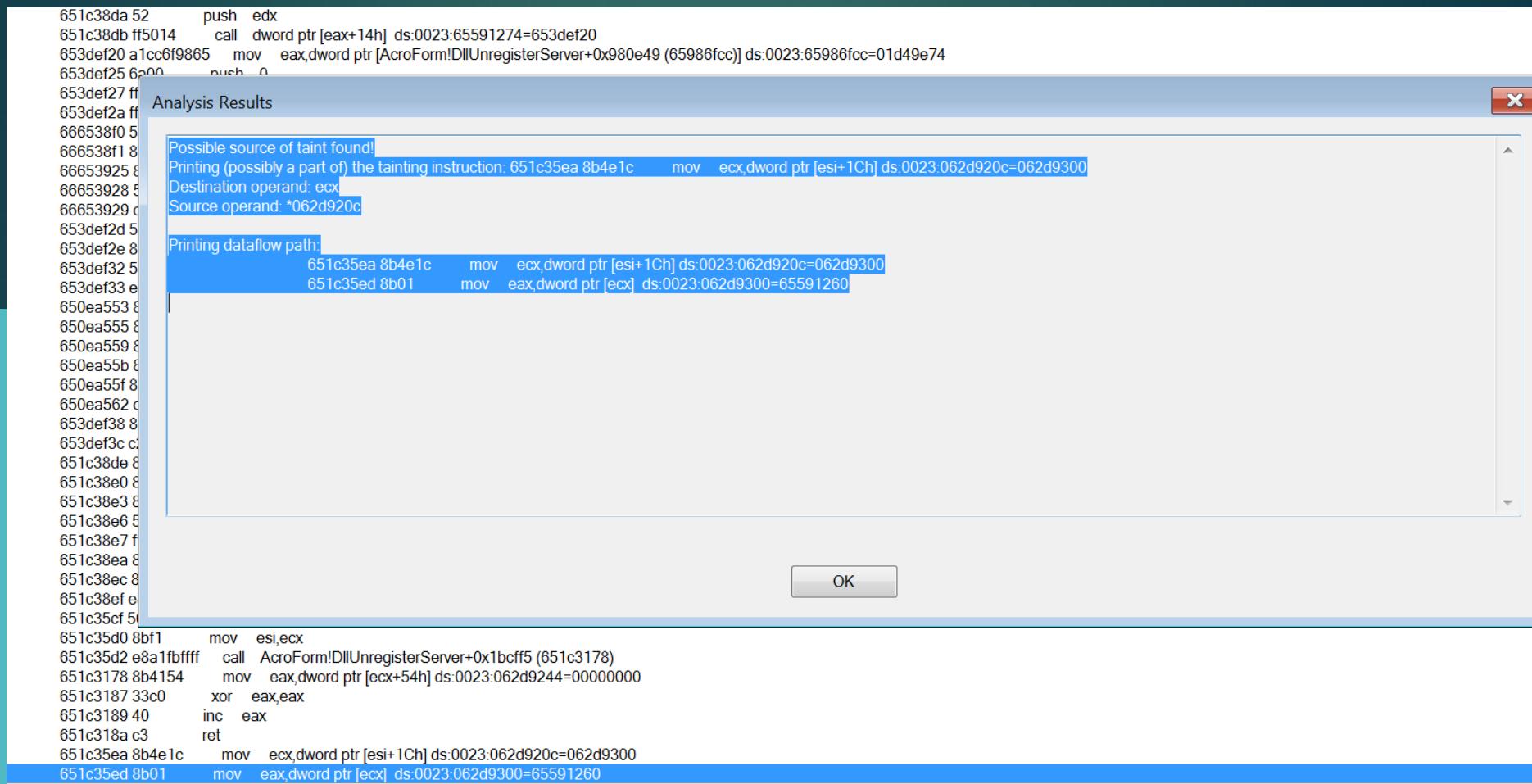
```
Dumping instruction taint information:
```

```
instr->Src tainted: *062d9300
instr->SrcDep1 tainted: ecx
```

At the crash point, we check the trace to see if the pointer is indeed controlled

Contd...

43



Dataflow information can be visualized in the GUI

Contd...

```

0:000> g
(6c.180): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=424144b7 ebx=00000400 ecx=42414241 edx=00000002 esi=002be28c edi=00000276
eip=638038d5 esp=002be044 ebp=002be074 iopl=0 nv up ei pl nz na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00210206
AcroForm!D11UnregisterServer+0x1bd752:
638038d5 8b01          mov     eax,dword ptr [ecx]  ds:0023:42414241=?????????

```

"C:\Program Files\Adobe\Reader 9.0\Reader\AcroRd32.exe" - WinDbg:6.11.0001.404 X86

File Edit View Debug Window Help

Command - "C:\Program Files\Adobe\Reader 9.0\Reader\AcroRd32.exe" - WinDbg:6.11.0001.404 X86

```

ModLoad: 10000000 10095000 C:\Program Files\Adobe\Reader 9.0\Reader\cryptocme2.dll
ModLoad: 03760000 037d6000 C:\Program Files\Adobe\Reader 9.0\Reader\ccme_base.dll
ModLoad: 733d0000 733d7000 C:\Program Files\Adobe\Reader 9.0\Reader\viewerps.dll
(274.7b4): C++ EH exception - code e06d7363 (first chance)
ModLoad: 65730000 65dd7000 C:\Program Files\Adobe\Reader 9.0\Reader\plug_ins\PPKLite.api
ModLoad: 6faa0000 6faa7000 C:\Windows\system32\WSOCK32.dll
ModLoad: 768d0000 76905000 C:\Windows\system32\WS2_32.dll
ModLoad: 77350000 77356000 C:\Windows\system32\NSI.dll
ModLoad: 4a800000 4a8a7000 C:\Program Files\Adobe\Reader 9.0\Reader\icucnv36.dll
ModLoad: 4ad00000 4ad17000 C:\Program Files\Adobe\Reader 9.0\Reader\icudt36.dll
ModLoad: 72890000 7289c000 C:\Windows\system32\ATMLIB.dll
ModLoad: 6bea0000 6bf11000 C:\Program Files\Adobe\Reader 9.0\Reader\plug_ins\Accessibility.api
ModLoad: 69660000 696cc000 C:\Program Files\Adobe\Reader 9.0\Reader\AdobeXMP.dll
ModLoad: 686a0000 68707000 C:\Program Files\Adobe\Reader 9.0\Reader\plug_ins\PDDom.api
(274.7b4): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=00000001 ebx=03beb658 ecx=00000001 edx=00000000 esi=004eff60 edi=03beb658
eip=45454443 esp=0030dfec ebp=42414241 iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00210202
45454443 ?? ???

```

We indeed control the values (coming from our input file)

Sample Analysis 2

MS14-035 Internet Explorer CI... | Pid 2256 - WinDbg:6.11.0001.404 X86
File Edit View Debug Window Help

Command - Pid 2256 - WinDbg:6.11.0001.404 X86

```
0:005> p
eax=0414c9f0 ebx=00000000 ecx=00000002 edx=0414ca90 esi=0871cf88 edi=0414ca38
eip=6a7f38f2 esp=0414c94c ebp=0414ca4c iopl=0 nv up ei pl nz na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00000206
mshtml!CBase::InvokeEvent+0xf1:
6a7f38f2 899c248c000000 mov    dword ptr [esp+8Ch],ebx ss:0023:0414c9d8=00000000
0:005> pt
eax=00000000 ebx=00000000 ecx=9ad2cbeb edx=08b01000 esi=085f5f30 edi=00000000
eip=6a7f3a6d esp=0414ca50 ebp=0414cbac iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00000246
mshtml!CBase::InvokeEvent+0x62d:
6a7f3a6d c22400      ret     24h
0:005> !dptrace_trace C:\Users\MacbookRo\Desktop\PoCs\midnight_log4.vdt
(8d0.504): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.

A total of 22293 instructions were traced and 15557 were dumped to C:\Users\MacbookRo\Desktop\PoCs\midnight_log4.vdt
Duration of this command in seconds: 7.000000
0:005> !dptrace_forward 2 68

Allocated range is
3ce0000-3ce1000.3cf0000-3cf1000

0:005> r
eax=00000004 ebx=085f7fb0 ecx=00000002 edx=00000004 esi=08588fa0 edi=00000002
eip=6a7eb792 esp=0414cf6c ebp=0414cf8c iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010202
mshtml!CElement::GetLookasidePtr+0x7:
6a7eb792 23461c      and    eax,dword ptr [esi+1Ch] ds:0023:08588fbc=????????
```

0:005> r esi =3ce0000

CVE-2014-0282 IE8/9/10/11 ‘Cinput’ Use-After-Free (MS14-035)

Contd...

MS14-035 Internet Explorer CI...

Pid 2256 - WinDbg:6.11.0001.404 X86

File Edit View Debug Window Help

Command - Pid 2256 - WinDbg:6.11.0001.404 X86

```
eip=6a7f3a6d esp=0414ca50 ebp=0414cbac iopl=0          nv up ei pl xr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000          efl=000000246
mshtml!CBase::InvokeEvent+0x62d:
6a7f3a6d c22400      ret     24h
0:005> !dptrace_trace C:\Users\MacbookRo\Desktop\PoCs\midnight_log4.vdt
(8d0.504): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.

A total of 22293 instructions were traced and 15557 were dumped to C:\Users\MacbookRo\Desktop\PoCs\midnight_log4.vdt
Duration of this command in seconds: 7.000000

0:005> !dptrace_forward 2 68

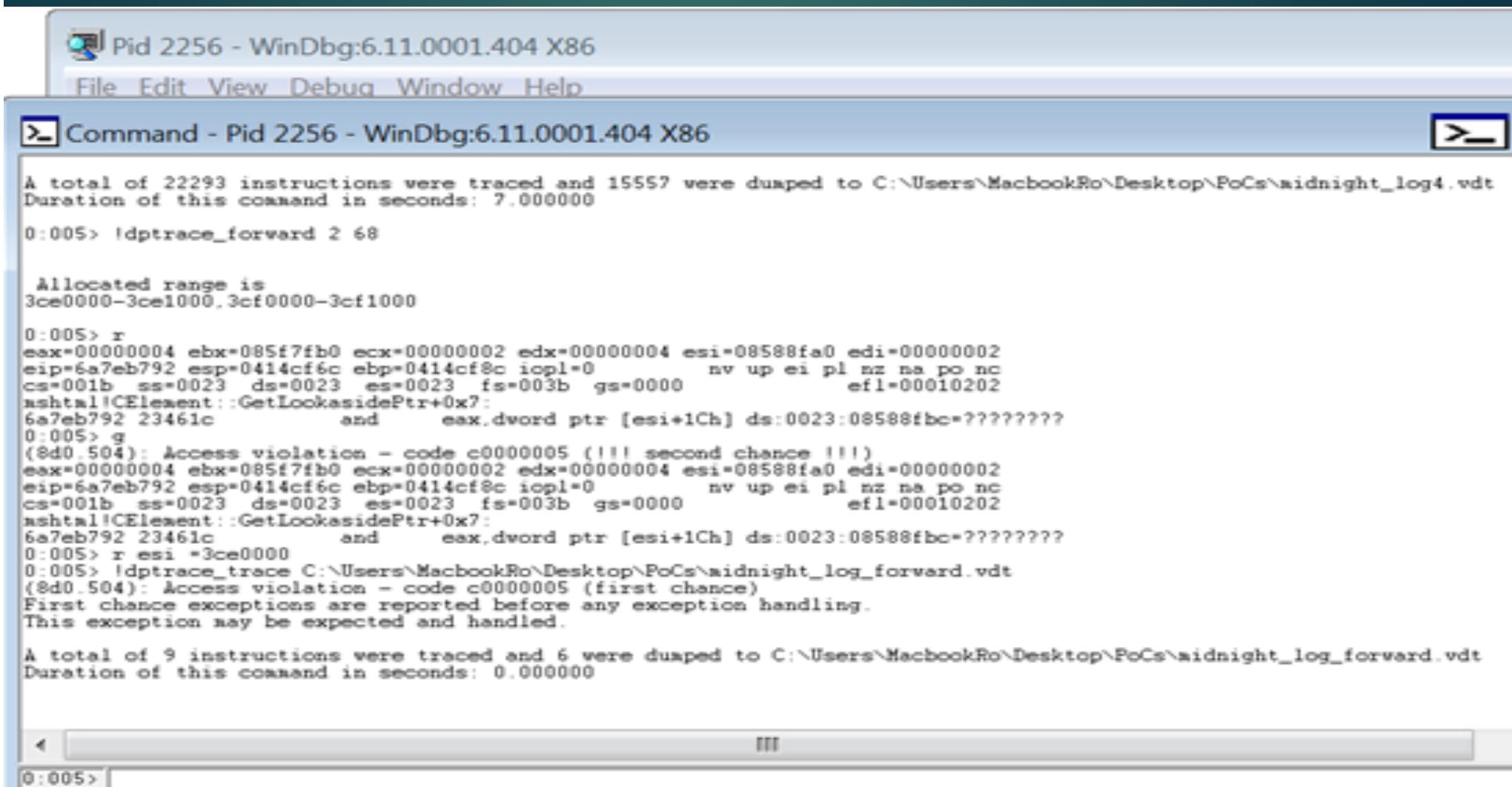
Allocated range is
3ce0000-3ce1000,3cf0000-3cf1000

0:005> r
eax=00000004 ebx=085f7fb0 ecx=00000002 edx=00000004 esi=08588fa0 edi=00000002
eip=6a7eb792 esp=0414cf6c ebp=0414cf8c iopl=0          nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000          efl=00010202
mshtml!CElement::GetLookasidePtr+0x7:
6a7eb792 23461c      and     eax,dword ptr [esi+1Ch] ds:0023:08588fbc=???????
0:005> g
(8d0.504): Access violation - code c0000005 (!!! second chance !!!)
eax=00000004 ebx=085f7fb0 ecx=00000002 edx=00000004 esi=08588fa0 edi=00000002
eip=6a7eb792 esp=0414cf6c ebp=0414cf8c iopl=0          nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000          efl=00010202
mshtml!CElement::GetLookasidePtr+0x7:
6a7eb792 23461c      and     eax,dword ptr [esi+1Ch] ds:0023:08588fbc=???????
0:005> r esi =3ce0000

0:005> !dptrace_trace C:\Users\MacbookRo\Desktop\PoCs\midnight_log_forward.vdt
```

Replace Freed object with the root of the fake object chain

Contd...



Pid 2256 - WinDbg:6.11.0001.404 X86

File Edit View Debug Window Help

Command - Pid 2256 - WinDbg:6.11.0001.404 X86

```
A total of 22293 instructions were traced and 15557 were dumped to C:\Users\MacbookRo\Desktop\PoCs\midnight_log4.vdt
Duration of this command in seconds: 7.000000

0:005> !dptrace_forward 2 68

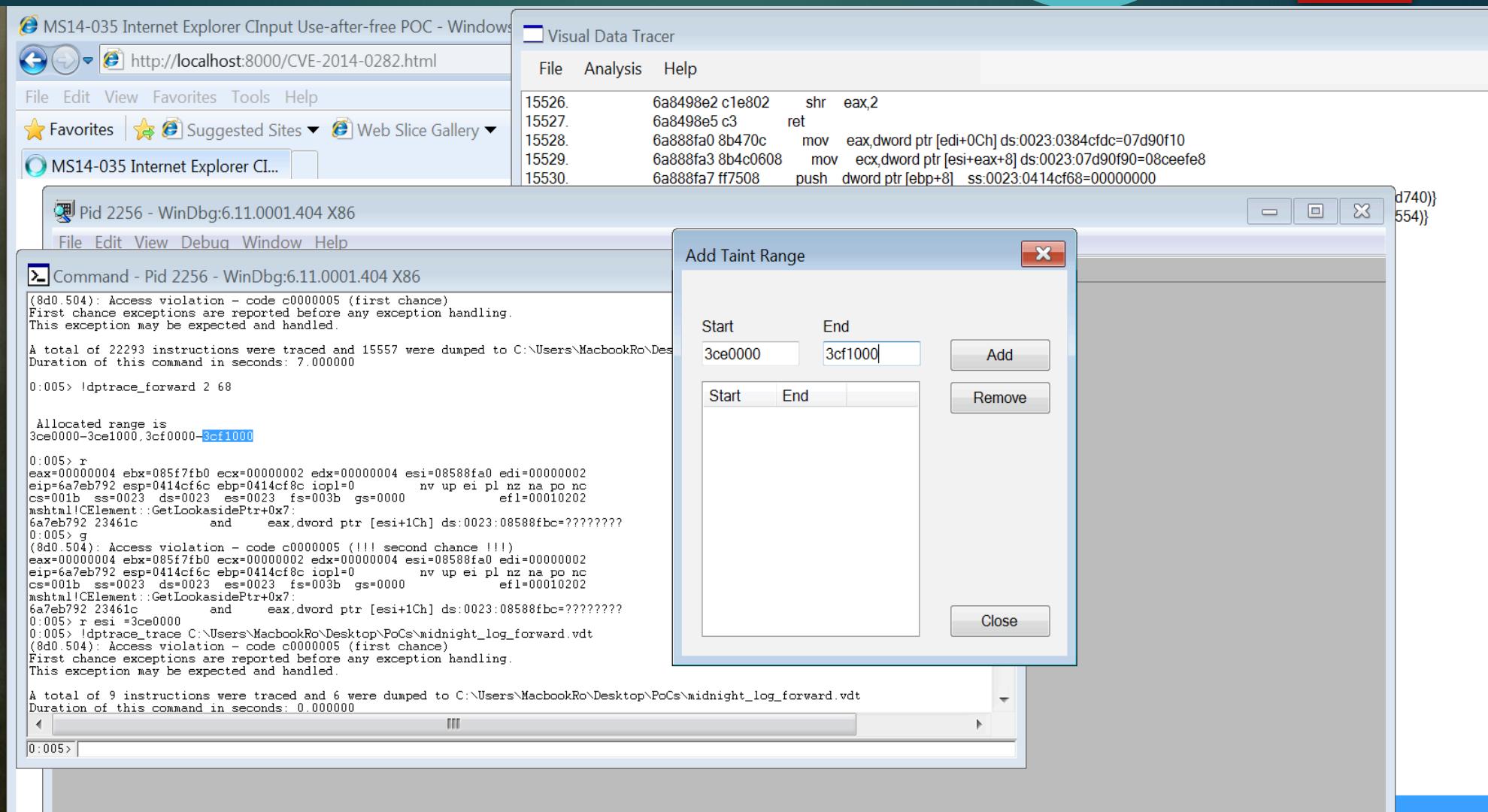
Allocated range is
3ce0000-3ce1000,3cf0000-3cf1000

0:005> r
eax=00000004 ebx=085f7fb0 ecx=00000002 edx=00000004 esi=08588fa0 edi=00000002
eip=6a7eb792 esp=0414cf6c ebp=0414cf8c iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010202
nshnt!CElement::GetLookasidePtr+0x7:
6a7eb792 23461c and eax,dword ptr [esi+1Ch] ds:0023:08588fbc=?????????
0:005> g
(8d0.504): Access violation - code c0000005 (!!! second chance !!!)
eax=00000004 ebx=085f7fb0 ecx=00000002 edx=00000004 esi=08588fa0 edi=00000002
eip=6a7eb792 esp=0414cf6c ebp=0414cf8c iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010202
nshnt!CElement::GetLookasidePtr+0x7:
6a7eb792 23461c and eax,dword ptr [esi+1Ch] ds:0023:08588fbc=?????????
0:005> r esi =3ce0000
0:005> !dptrace_trace C:\Users\MacbookRo\Desktop\PoCs\midnight_log_forward.vdt
(8d0.504): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.

A total of 9 instructions were traced and 6 were dumped to C:\Users\MacbookRo\Desktop\PoCs\midnight_log_forward.vdt
Duration of this command in seconds: 0.000000
```

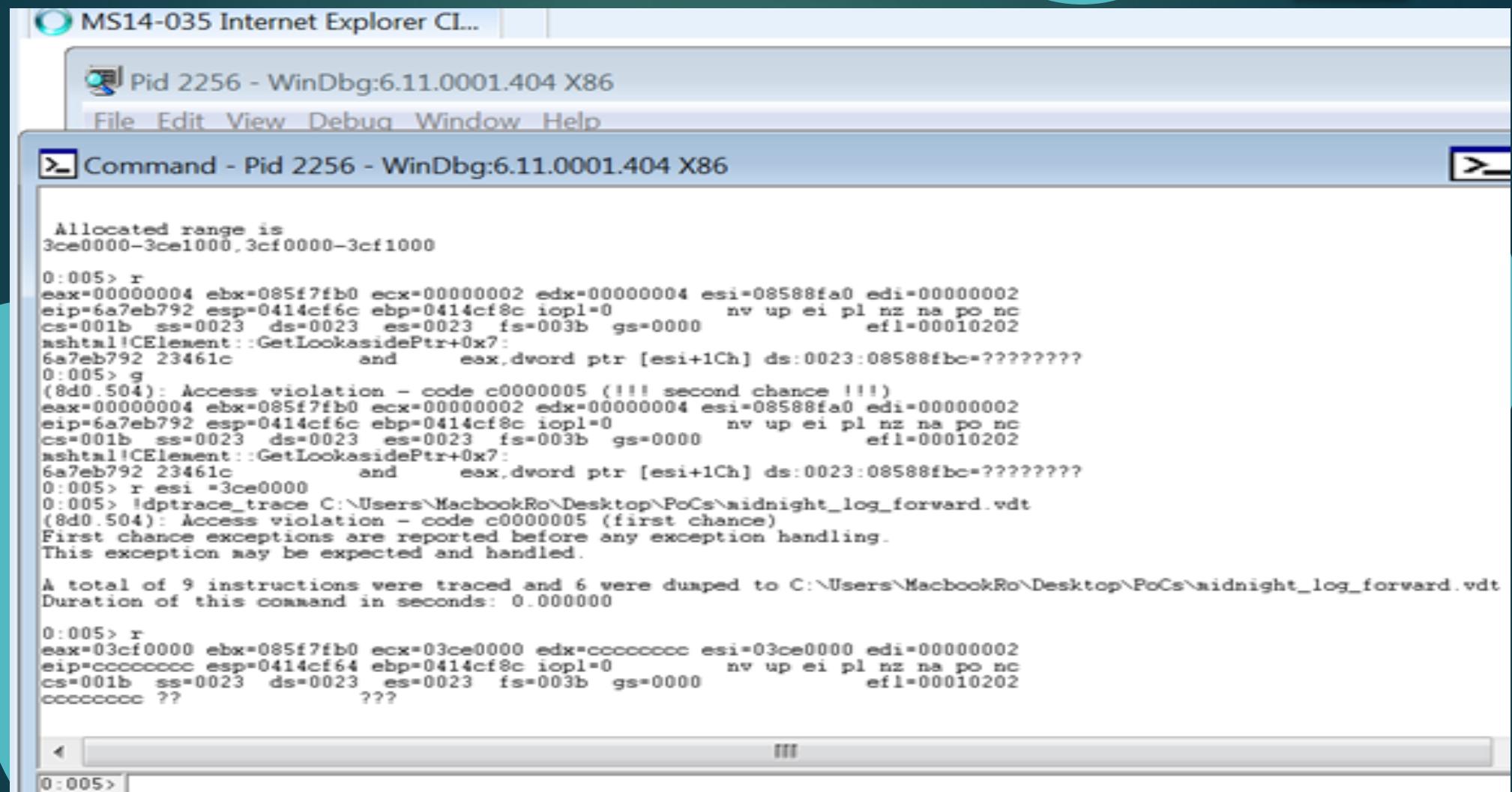
Continue from initial crash and trace each subsequent
breakpoint/access violation

Contd...



Add the range of the fake allocated objects, so when we look for the taint information on the instruction of interest, we can confirm it is mapped to our controlled memory areas

Contd...



MS14-035 Internet Explorer CI... | Pid 2256 - WinDbg:6.11.0001.404 X86

File Edit View Debug Window Help

Command - Pid 2256 - WinDbg:6.11.0001.404 X86

```
Allocated range is  
3ce0000-3ce1000,3cf0000-3cf1000

0:005> r  
eax=00000004 ebx=085f7fb0 ecx=00000002 edx=00000004 esi=08588fa0 edi=00000002  
eip=6a7eb792 esp=0414cf6c ebp=0414cf8c iopl=0 nv up ei pl nz na po nc  
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010202  
nshtal!CElement::GetLookasidePtr+0x7:  
6a7eb792 23461c and eax,dword ptr [esi+1Ch] ds:0023:08588fbc=????????  
0:005> g  
(8d0.504): Access violation - code c0000005 (!!! second chance !!!)  
eax=00000004 ebx=085f7fb0 ecx=00000002 edx=00000004 esi=08588fa0 edi=00000002  
eip=6a7eb792 esp=0414cf6c ebp=0414cf8c iopl=0 nv up ei pl nz na po nc  
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010202  
nshtal!CElement::GetLookasidePtr+0x7:  
6a7eb792 23461c and eax,dword ptr [esi+1Ch] ds:0023:08588fbc=????????  
0:005> r esi =3ce0000  
0:005> !dptrace_trace C:\Users\MacbookRo\Desktop\PoCs\midnight_log_forward.vdt  
(8d0.504): Access violation - code c0000005 (first chance)  
First chance exceptions are reported before any exception handling.  
This exception may be expected and handled.

A total of 9 instructions were traced and 6 were dumped to C:\Users\MacbookRo\Desktop\PoCs\midnight_log_forward.vdt  
Duration of this command in seconds: 0.000000

0:005> r  
eax=03cf0000 ebx=085f7fb0 ecx=03ce0000 edx=cccccccc esi=03ce0000 edi=00000002  
eip=cccccccc esp=0414cf64 ebp=0414cf8c iopl=0 nv up ei pl nz na po nc  
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010202  
???????? ???
```

Program Control Immediately evident. We just need to make sure we can point indeed it to our fake structure

Contd...

50

MS14-035 Internet Explorer CI... Pid 2256 - WinDbg:6.11.0001.404 X86

File Edit View Debug Window Help

Command - Pid 2256 - WinDbg:6.11.0001.404 X86

```
0:005> r esi *3ce0000
0:005> !dptrace_trace C:\Users\MacbookRo\Desktop\PoCs\midnight_log_forward.vdt
(8d0.504): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.

A total of 9 instructions were traced and 6 were dumped to C:\Users\MacbookRo\Desktop\PoCs\midnight_log_forward.vdt
Duration of this command in seconds: 0.000000

0:005> r
eax=03cf0000 ebx=085f7fb0 ecx=03ce0000 edx=cccccccc esi=03ce0000 edi=00000002
eip=cccccccc esp=0414cf64 ebp=0414cf8c iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010202
???????? ??
0:005> dd esi
03ce0000 03cf0000 03cf0004 03cf0008 03cf000c
03ce0010 03cf0010 03cf0014 03cf0018 03cf001c
03ce0020 03cf0020 03cf0024 03cf0028 03cf002c
03ce0030 03cf0030 03cf0034 03cf0038 03cf003c
03ce0040 03cf0040 41414141 41414141 41414141
03ce0050 41414141 41414141 41414141 41414141
03ce0060 41414141 41414141 41414141 41414141
03ce0070 41414141 41414141 41414141 41414141
0:005> dd 03cf0000
03cf0000 cccccccc cccccccc cccccccc cccccccc
03cf0010 cccccccc cccccccc cccccccc cccccccc
03cf0020 cccccccc cccccccc cccccccc cccccccc
03cf0030 cccccccc cccccccc cccccccc cccccccc
03cf0040 cccccccc cccccccc cccccccc cccccccc
03cf0050 cccccccc cccccccc cccccccc cccccccc
03cf0060 cccccccc cccccccc cccccccc cccccccc
03cf0070 cccccccc cccccccc cccccccc cccccccc
```

0:005>

We see that the EIP value at time of crash comes from our fake object allocated at the previous crash

Contd...

MS14-035 Internet Explorer Clinput Use-after-free POC - Windows Internet Exp http://localhost:8000/CVE-2014-0282.html

File Edit View Favorites Tools Help

Favorites Suggested Sites Web Slice Gallery

MS14-035 Internet Explorer Cl... Pid 2256 - WinDbg:6.11.0001.404 X86

File Edit View Debug Window Help

Command - Pid 2256 - WinDbg:6.11.0001.404 X86

```

WARNING: Frame IP not in any known module. Following frames may be wrong.
0x414cf60 6a7eb694 6a85bde8 6a641730 00002fd0 0xcccccccc
0x414cf64 6a85bde8 6a641730 00002fd0 00001200 mshtml!CElement::Doc+0x7
0x414cf68 6a641730 00002fd0 00001200 6a86cb54 mshtml!CElement::GetLookasidePtr+0
0x414cf6c 6a7a3150 005f7fb0 00002fd0 6a7a311d mshtml!CElement::ForaElement::DoReset+0x9c
0x414cf6f 6a85f10b 005f7fb0 00002fd0 00ba7fd0 mshtml!Method_void_void+0x75
0x414cf8c 6a8666c6 005f7fb0 000003f2 00000001 mshtml!CBase::ContextInvokeEx+0x5c
0x414d00c 6a88738e 005f7fb0 000003f2 00000001 mshtml!CElement::ContextInvokeEx+0x5c
0x414d008 6a80b0c0 005f7fb0 000003f2 00000001 mshtml!CFormElement::VersionedInvoke
0x414d0fc 6aef26e 07ee7fd0 000003f2 00000001 mshtml!PlainInvokeEx+0xeb
0x414d138 6aef26e 08856d10 000003f2 000000409 jscript!IDispatchExInvokeEx+0x104
0x414d174 6aef43a 08856d10 00000409 00000001 jscript!IDispatchExInvokeEx+0x6a
0x414d234 6aef43e 000003f2 00000001 00000000 jscript!InvokeDispatchEx+0x98
0x414d268 6af0d9a0 08856d10 0414d29c 00000001 jscript!VAR::InvokeByName+0x139
0x414d2b4 6af0d9a0 08856d10 00000001 00000000 jscript!VAR::InvokeDispName+0x7d
0x414d2e0 6af0d9a0 08856d10 00000000 00000001 jscript!VAR::InvokeByDispID+0xce
0x414d47c 6af0d7d 0414d494 0414d5d0 0073af88 jscript!CScriptRuntime::Run+0x2b80
0x414d564 6af05cd8 0414d5d0 00000000 00000000 jscript!ScrFncObj::CallWithFrameOr
0x414d5ac 6af05ef1 0414d5d0 00000000 00000000 jscript!ScrFncObj::Call+0x8d
0x414d628 6af0620a 0073af88 0414d7e8 00000000 jscript!CSession::Execute+0x15f
0x414d674 6af0c3b9 0884adf0 0414d7e8 0414d7f8 jscript!ColeScript::ExecutePending
0:005> ub 6a7eb694
mshtml!CElement::SecurityContext+0x29:
6a7eb680 90    nop
6a7eb684 90    nop
6a7eb688 90    nop
6a7eb68c 90    nop
6a7eb694 90    nop
mshtml!CElement::Doc:
6a7eb684 8b01  mov    eax,dword ptr [ecx]
6a7eb688 8b01  mov    edx,dword ptr [eax+70h]
6a7eb692 ffd2  call   edx

```

0:005>

Done!

Visual Data Tracer

15526.	6a8498e2 c1e802	shr	eax,2
15527.	6a8498e5 c3	ret	
15528.	6a888fa0 8b470c	mov	eax,dword ptr [edi+0Ch] ds:0023:0384cfcc=07d90f10
15529.	6a888fa3 8b4c0608	mov	ecx,dword ptr [esi+eax+8] ds:0023:07d90f90=08ceeffe8
15530.	6a888fa7 ff7508	push	dword ptr [ebp+8] ss:0023:0414cf68=00000000
15531.	6a888faa 8b01	mov	eax,dword ptr [ecx] ds:0023:08ceeff8=[mshtml!CElementAryCacheItem::'vtable' (6a7ed740)]
15532.	6a888fac ff501c	call	dword ptr [eax+1Ch] ds:0023:6a7ed75c=[mshtml!CElementAryCacheItem::GetAI (6a7c2554)]
15533.	6a7c2554 8bff	mov	edi,edi
15534.	6a7c2556 55	push	ebp
15535.	6a7c2557 8bec	mov	ebp,esp
15536.	6a7c2559 8b4508	mov	eax,dword ptr [ebp+8] ss:0023:0414cf58=00000000
15537.	6a7c2560 8b510c	mov	edx,dword ptr [ecx+0Ch] ds:0023:08ceeff4=00000010
15538.	6a7c2563 c1ea02	shr	edx,2
15539.	6a7c256a 8b4914	mov	ecx,dword ptr [ecx+14h] ds:0023:08ceeffc=08218ff0
15540.	6a7c256d 8b0481	mov	eax,dword ptr [ecx+eax*4] ds:0023:08218ff0=08588fa0
15541.	6a7c2570 5d	pop	ebp
15542.	6a7c2571 c20400	ret	4
15543.	6a888faf 8b4d0c	mov	ecx,dword ptr [ebp+0Ch] ss:0023:0414cf6c=0414cf80
15544.	6a888bf2 8901	mov	dword ptr [ecx],eax ds:0023:0414cf80=00000000
15545.	6a888bf4 33c0	xor	eax, eax
15546.	6a888bf6 5e	pop	esi
15547.	6a888fb7 5d	pop	ebp
15548.	6a888bf8 c20800	ret	8
15549.	6a641720 8b742410	mov	esi,dword ptr [esp+10h] ss:0023:0414cf80=08588fa0
15550.	6a641728 6a02	push	2
15551.	6a64172a 5f	pop	edi
15552.	6a64172b e85ba01a00	call	mshtml!CElement::GetLookasidePtr (6a7eb78b)
15553.	6a7eb78b 33c0	xor	eax, eax
15554.	6a7eb78d 40	inc	eax
15555.	6a7eb78e 8bcf	mov	ecx,edi
15556.	6a7eb790 d3e0	shl	eax,cl
15557.	6a7eb792 23461c	and	eax,dword ptr [esi+1Ch] ds:0023:08588fb8=????????
1.	6a7eb792 23461c	and	eax,dword ptr [esi+1Ch] ds:0023:03ce001c=03cf001c
2.	6a85bde1 8bce	mov	ecx,esi
3.	6a85bde3 e8a5f8f8ff	call	mshtml!CElement::Doc (6a7eb68d)
4.	6a7eb68d 8b01	mov	eax,dword ptr [ecx] ds:0023:03ce0000=03cf0000
5.	6a7eb68f 8b5070	mov	edx,dword ptr [eax+70h] ds:0023:03cf0070=cccccccc
6.	6a7eb692 ffd2	call	edx {cccccccc}

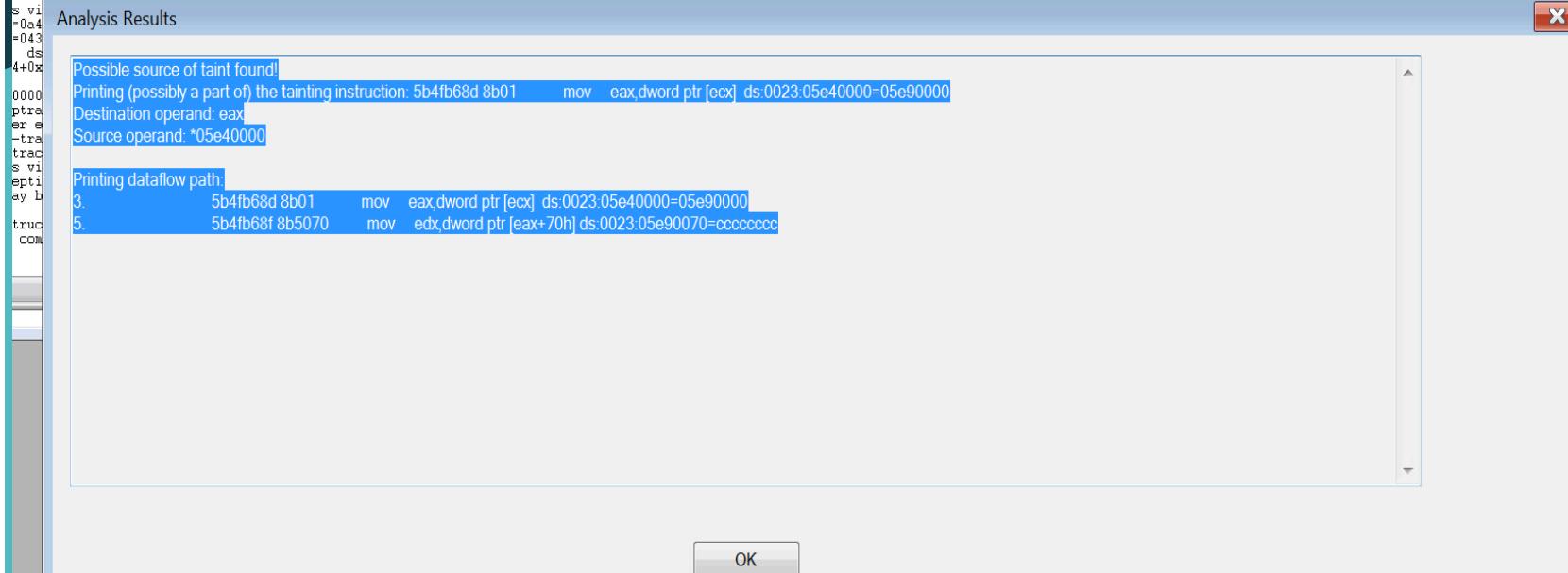
Visualize it in the tracer and trace the program control (or directly in the command line of the debugger, shown later)

Contd...

52

```
0 74207000 C:\Windows\system32\nidimap.dll
s violation - code c0000005 (first chance)
exceptions are reported before any exception handling.
ay be expected and handled.
=0a4ccfb0 ecx=00000002 edx=00000004 esi=0950effa0 edi=00000002
=0433cb94 ebp=0433cbba4 iopl=0 nv up ei pl nz na po nc
 ds=0023 es=0023 fs=003b gs=0000 efl=00010202
l file could not be found. Defaulted to export symbols for C:\Windows\System32\mshtml.dll -
4+0x4b1f3:
    and     eax,dword ptr [esi+1Ch] ds:0023:0950efbc=???????
forward
e_forward found
racer
forward
passed, using default n=2, s=40 (bytes) and p=0x02 <(read only)>
h arguments type !vdt-tracer.vdt_help for help
```

```
is
5e90000-5e91000
```



Taint source is confirmed also in the analyzer (visual here). Same thing can be obtained in the command line by !dptrace_analyzer <analyzer_binary> <trace_file> <keep GUI open> <ranges> <index of instruction to check the taint of>

Contd...

```
Unloading dptracer extension DLL
0:004> !dptrace_trace C:\Users\MacbookRo\Desktop\PoCs\log_again4.vdt
WARNING: This break is not a step/trace completion.
The last command has been cleared to prevent
accidental continuation of this unrelated event.
Check the event, location and thread before resuming.
(f54.e4): Break instruction exception - code 80000003 (first chance)

A total of 677415 instructions were traced and 455209 were dumped to C:\Users\MacbookRo\Desktop\PoCs\log_again4.vdt
Duration of this command in seconds: 205.000000

0:012> r
eax=7ffd000 ebx=00000000 ecx=00000000 edx=77c7f125 esi=00000000 edi=00000000
eip=77c140f0 esp=051dfa5c ebp=051dfa88 iopl=0 nv up ei pl xr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00000246
ntdll!DbgBreakPoint:
77c140f0 cc int 3
0:012> g
eax=00000000 ebx=0c05af20 ecx=00000003 edx=0a85af78 esi=5b50fbec edi=5b505164
eip=5b50fc19 esp=0439c964 ebp=0439c96c iopl=0 ov up ei pl nz ac pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00000a16
mshtml!PlainQueryInterface+0x1f:
5b50fc19 8b4508 mov eax,dword ptr [ebp+8] ss:0023:0439c974=05a2afdf
0:004> g
Breakpoint 0 hit
eax=5b498bb8 ebx=0a2c0fb0 ecx=00000002 edx=00000004 esi=08310f88 edi=00000002
eip=5b35173a esp=0439cae0 ebp=0439caf0 iopl=0 nv up ei pl xr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00000246
mshtml!CFormElement::DoReset+0xe2:
5b35173a 8bce mov ecx,esi
0:004> !dptrace_trace C:\Users\MacbookRo\Desktop\PoCs\log_again5.vdt
(f54.918): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.

A total of 891392 instructions were traced and 618734 were dumped to C:\Users\MacbookRo\Desktop\PoCs\log_again5.vdt
Duration of this command in seconds: 266.000000

0:004> r
eax=00000004 ebx=0a2c0fb0 ecx=00000002 edx=00000004 esi=0b491fa0 edi=00000002
eip=5b4fb792 esp=0439cad0 ebp=0439caf0 iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010202
mshtml!CElement::GetLookasidePtr+0x7:
5b4fb792 23461c and eax,dword ptr [esi+1Ch] ds:0023:0b491fbc=????????
```

Because the backward taint analysis demand tracing the process, so we can later construct the BFS analysis, it is important to use intelligently/diligently. In the case of this issue, we use to analyze a part of the execution, instead of the initial crash.

Contd...

Pid 1896 - WinDbg:6.11.0001.404 X86

File Edit View Debug Window Help

Command

```

ModLoad: 743e0000 743e7000 C:\Windows\system32\AVRT.dll
ModLoad: 73500000 73536000 C:\Windows\system32\AUDIOSES.DLL
ModLoad: 734f0000 734f8000 C:\Windows\system32\msacm32.drv
ModLoad: 734d0000 734e4000 C:\Windows\system32\MSACM32.dll
ModLoad: 734c0000 734c7000 C:\Windows\system32\midimap.dll
ModLoad: 5d920000 5d9d2000 C:\Windows\System32\jscript.dll
(768.988): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=00000004 ebx=0aac5fb0 ecx=00000002 edx=00000004 esi=0a844fa0 edi=00000002
eip=08a3b792 esp=043dcba4 ebp=043dcbb4 iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010202
*** ERROR: Symbol file could not be found. Defaulted to export symbols for C:\Windows\System32\mshtml.dll -
mshtml!Ordinal104+0x4bf3:
08a3b792 23461c and eax,dword ptr [esi+1Ch] ds:0023:0a844fbc=???????
0:005> g
(768.988): Access violation - code c0000005 (!!! second chance !!!)
eax=00000004 ebx=0aac5fb0 ecx=00000002 edx=00000004 esi=0a844fa0 edi=00000002
eip=08a3b792 esp=043dcba4 ebp=043dcbb4 iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010202
mshtml!Ordinal104+0x4bf3:
08a3b792 23461c and eax,dword ptr [esi+1Ch] ds:0023:0a844fbc=???????
0:005> !dptrace_forward 4 200
No export dptrace_forward found
0:005> .load dptracer
0:005> !dptrace_forward 4 200

Allocated range is
5880000-5881000,5890000-5891000,58a0000-58a1000,58b0000-58b1000

0:005> r esi=5880000
0:005> r
eax=00000004 ebx=0aac5fb0 ecx=00000002 edx=00000004 esi=05880000 edi=00000002
eip=08a3b792 esp=043dcba4 ebp=043dcbb4 iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010202
mshtml!Ordinal104+0x4bf3:
08a3b792 23461c and eax,dword ptr [esi+1Ch] ds:0023:0588001c=0589001c
0:005> !dptrace_trace C:\Users\MacbookRo\Desktop\PoCs\log_final2.vdt
(768.988): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.

A total of 9 instructions were traced and 6 were dumped to C:\Users\MacbookRo\Desktop\PoCs\log_final2.vdt
Duration of this command in seconds: 0.000000

```

Contd...

55

Pid 1896 - WinDbg:6.11.0001.404 X86

File Edit View Debug Window Help

Command

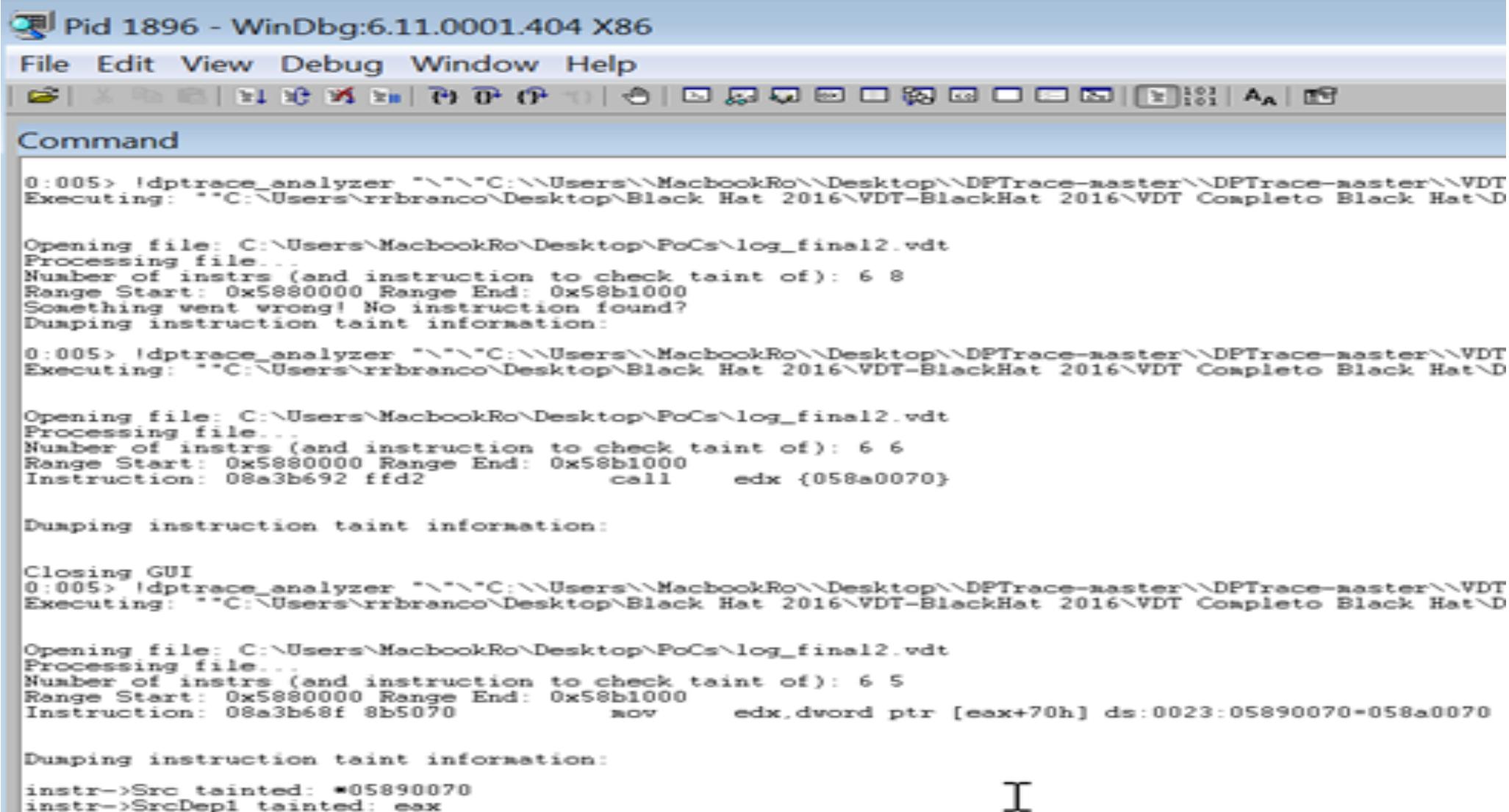
```
ModLoad: 743e0000 743e7000 C:\Windows\system32\AVRT.dll
ModLoad: 73500000 73536000 C:\Windows\system32\AUDIOSES.DLL
ModLoad: 734f0000 734f8000 C:\Windows\system32\msacm32.drv
ModLoad: 734d0000 734e4000 C:\Windows\system32\MSACM32.dll
ModLoad: 734c0000 734c7000 C:\Windows\system32\midimap.dll
ModLoad: 5d920000 5d9d2000 C:\Windows\System32\jscript.dll
(768.988): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=00000004 ebx=0aac5fb0 ecx=00000002 edx=00000004 esi=0a844fa0 edi=00000002
eip=08a3b792 esp=043dcbb4c ebp=043dcbb6c icpl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010202
*** ERROR: Symbol file could not be found. Defaulted to export symbols for C:\Windows\System32\mshtml.dll -
mshtml!Ordinal1104+0x4bf3:
08a3b792 23461c and eax,dword ptr [esi+1Ch] ds:0023:0a844fbc=???????
0:005> g
(768.988): Access violation - code c0000005 (!!! second chance !!!)
eax=00000004 ebx=0aac5fb0 ecx=00000002 edx=00000004 esi=0a844fa0 edi=00000002
eip=08a3b792 esp=043dcbb4c ebp=043dcbb6c icpl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010202
mshtml!Ordinal1104+0x4bf3:
08a3b792 23461c and eax,dword ptr [esi+1Ch] ds:0023:0a844fbc=???????
0:005> !dptrace_forward 4 200
No export dptrace_forward found
0:005> .load dptracer
0:005> !dptrace_forward 4 200

Allocated range is
5880000-5881000,5890000-5891000,58a0000-58a1000,58b0000-58b1000

0:005> r esi=5880000
0:005> r
eax=00000004 ebx=0aac5fb0 ecx=00000002 edx=00000004 esi=05880000 edi=00000002
eip=08a3b792 esp=043dcbb4c ebp=043dcbb6c icpl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010202
mshtml!Ordinal1104+0x4bf3:
08a3b792 23461c and eax,dword ptr [esi+1Ch] ds:0023:0588001c=0589001c
0:005> !dptrace_trace C:\Users\MacbookRo\Desktop\PoCs\log_final2.vdt
(768.988): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.

A total of 9 instructions were traced and 6 were dumped to C:\Users\MacbookRo\Desktop\PoCs\log_final2.vdt
Duration of this command in seconds: 0.000000

0:005> !dptrace_analyzer """C:\\\\Users\\\\MacbookRo\\\\Desktop\\\\DPTTrace-master\\\\DPTTrace-master\\\\VDT Complete Black Hat2\\\\Debug\\\\DPTRACE-GUI.exe\" \"C:\\\\Users\\\\MacbookRo\\\\Desktop\\\\PoCs\\\\log_final2.vdt
Ln 0, Col 0 Sys 0:<Local> Proc 000:768 Thrd 005:988 ASM OVR CAPS NUM
```



Pid 1896 - WinDbg:6.11.0001.404 X86

File Edit View Debug Window Help

Command

```
0:005> !dptrace_analyzer """C:\\\\Users\\\\MacbookRo\\\\Desktop\\\\DPTTrace-master\\\\DPTTrace-master\\\\VDT
Executing: """C:\\Users\\rrbranco\\Desktop\\Black Hat 2016\\VDT-BlackHat 2016\\VDT Completo Black Hat\\D

Opening file: C:\\Users\\MacbookRo\\Desktop\\PoCs\\log_final2.vdt
Processing file...
Number of instrs (and instruction to check taint of): 6 8
Range Start: 0x5880000 Range End: 0x58b1000
Something went wrong! No instruction found?
Dumping instruction taint information:

0:005> !dptrace_analyzer """C:\\\\Users\\\\MacbookRo\\\\Desktop\\\\DPTTrace-master\\\\DPTTrace-master\\\\VDT
Executing: """C:\\Users\\rrbranco\\Desktop\\Black Hat 2016\\VDT-BlackHat 2016\\VDT Completo Black Hat\\D

Opening file: C:\\Users\\MacbookRo\\Desktop\\PoCs\\log_final2.vdt
Processing file...
Number of instrs (and instruction to check taint of): 6 6
Range Start: 0x5880000 Range End: 0x58b1000
Instruction: 08a3b692 ffd2          call    edx {058a0070}

Dumping instruction taint information:

Closing GUI
0:005> !dptrace_analyzer """C:\\\\Users\\\\MacbookRo\\\\Desktop\\\\DPTTrace-master\\\\DPTTrace-master\\\\VDT
Executing: """C:\\Users\\rrbranco\\Desktop\\Black Hat 2016\\VDT-BlackHat 2016\\VDT Completo Black Hat\\D

Opening file: C:\\Users\\MacbookRo\\Desktop\\PoCs\\log_final2.vdt
Processing file...
Number of instrs (and instruction to check taint of): 6 5
Range Start: 0x5880000 Range End: 0x58b1000
Instruction: 08a3b68f 8b5070        mov     edx,dword ptr [eax+70h] ds:0023:05890070=058a0070

Dumping instruction taint information:
instr->Src tainted: *05890070
instr->SrcDep1 tainted: eax
```

I

Everything that was done using the GUI (setting the taint ranges, defining the instruction of interest and analyzing its taint information) is possible to do via the command-line of the debugger, as shown here

Sample Analysis 3

57

CVE-2015-6152 IE 11 CObjectElement Use-After-Free . Initial Crash on IE 11 without patches.

Contd...

58

Pid 2152 - WinDbg:6.2.9200.16384 X86

File Edit View Debug Window Help

Disassembly

Offset: @\$scopeip

Sc7685bf 8b07 mov eax,dword ptr [edi]
Sc7685c1 89442424 mov dword ptr [esp+24h],eax
Sc7685c5 f74024000000300 test dword ptr [eax+24h],30000h ds:0023:0678deb4-????????
Sc7685cc 0f8573010000 jne MSHTML!CTreeNode::ComputeFormatsHelper+0x1fb (Sc768745)
Sc7685d2 f74024000000400 test dword ptr [eax+24h],40000h

Command

Allocated range is
5b20000-5b21000,5b30000-5b31000,5b40000-5b41000,5b50000-5b51000

0:007> dd 5b20000
05b20000 05b30000 05b30004 05b30008 05b3000c
05b20010 05b30010 05b30014 05b30018 05b3001c
05b20020 05b30020 05b30024 05b30028 05b3002c
05b20030 05b30030 05b30034 05b30038 05b3003c
05b20040 05b30040 05b30044 05b30048 05b3004c
05b20050 05b30050 05b30054 05b30058 05b3005c
05b20060 05b30060 05b30064 05b30068 05b3006c
05b20070 05b30070 05b30074 05b30078 05b3007c

0:007> !vprot 5b20000
BaseAddress: 05b20000
AllocationBase: 05b20000
AllocationProtect: 00000004 PAGE_READWRITE
RegionSize: 00001000
State: 00001000 MEM_COMMIT
Protect: 00000004 PAGE_READWRITE
Type: 00020000 MEM_PRIVATE

0:007> !vprot 05b30000
BaseAddress: 05b30000
AllocationBase: 05b30000
AllocationProtect: 00000004 PAGE_READWRITE
RegionSize: 00001000
State: 00001000 MEM_COMMIT
Protect: 00000002 PAGE_READONLY
Type: 00020000 MEM_PRIVATE

Fake object chain of 4 objects of size 200. Precise size can be determined by manual analysis to figure out the freed/alloc'd function and checking the size of the root object.

Contd...

59

Pid 2152 - WinDbg:6.2.9200.16384 X86

File Edit View Debug Window Help

Disassembly

Offset: @\$scopeip

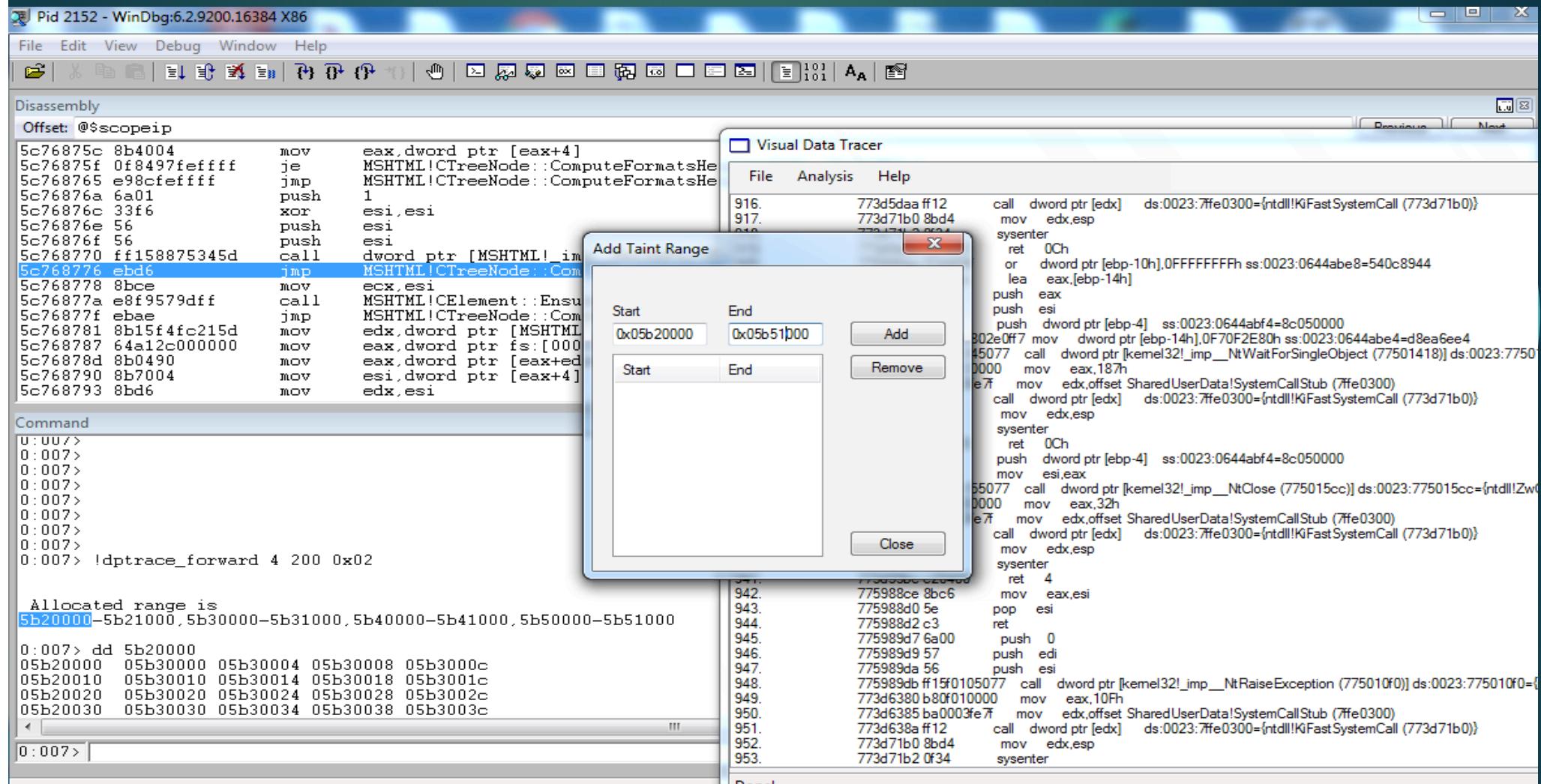
```
Sc768593 cd29    int     29h
Sc768595 e99171a2ff jmp     MSHTML!CFormatInfo::Cleanup+0x1ba (5c18f72b)
Sc76859a b904000000 mov     ecx, 4
Sc76859f cd29    int     29h
Sc7685a1 e99171a2ff jmp     MSHTML!CFormatInfo::Cleanup+0x19e (5c18f70f)
Sc7685a6 f705801f225d00040000 test    dword ptr [MSHTML!Microsoft_IEEnableBits (5d221f80)], 400
Sc7685b0 0f85fed20d00 jne    MSHTML!`CBackgroundInfo::Property<CBackgroundImage>'::`7'
Sc7685b6 f7470800001000 test    dword ptr [edi+8], 100000h
Sc7685bd 7573    jne    MSHTML!CTreeNode::ComputeFormatsHelper+0xe8 (5c768632)
Sc7685bf 8b07    mov     eax, dword ptr [edi]
Sc7685c1 89442424 mov     eax, dword ptr [esp+24h].eax
Sc7685c5 f7402400000300 test    dword ptr [eax+24h], 30000h ds:0023-05b20024=2400b305
Sc7685cc 0f8573010000 jne    MSHTML!CTreeNode::ComputeFormatsHelper+0x1fb (5c768745)
Sc7685d2 f7402400000400 test    dword ptr [eax+24h], 40000h
Sc7685d9 0f840fd30d00 je     MSHTML!`CBackgroundInfo::Property<CBackgroundImage>'::`7'
Sc7685df 8b4030    mov     eax, dword ptr [eax+30h]
Sc7685e2 2403    and    al, 3
Sc7685e4 3c01    cap    al, 1
Sc7685e6 0f85f1d20d00 jne    MSHTML!`CBackgroundInfo::Property<CBackgroundImage>'::`7'
Sc7685ec 8b442424 mov     eax, dword ptr [esp+24h]
```

Command

```
eax=0678de90 ebx=0e3d2fc0 ecx=00000000 edx=5d21edf0 esi=0679dfac edi=0679dfa0
eip=5c7685c5 esp=0644af40 ebp=0644bc18 iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00210246
MSHTML!CTreeNode::ComputeFormatsHelper+0x53:
5c7685c5 f7402400000300 test    dword ptr [eax+24h], 30000h ds:0023:0678deb4=?????????
0:007> r eax=5b20000
0:007> r
eax=05b20000 ebx=0e3d2fc0 ecx=00000000 edx=5d21edf0 esi=0679dfac edi=0679dfa0
eip=5c7685c5 esp=0644af40 ebp=0644bc18 iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00210246
MSHTML!CTreeNode::ComputeFormatsHelper+0x53:
5c7685c5 f7402400000300 test    dword ptr [eax+24h], 30000h ds:0023:05b20024=2400b305
```

'Redefine' the reference to freed reference (eax) with the first fake object .
Continue the execution with !dptrace_trace and monitor the forward trace .

Contd...



Add taint range and check to see if the source of an access violation can be traced back to controlled input

Contd...

```
D:007> ed eax+24 300
D:007> r
eax=05ba0000 ebx=0ea4bf00 ecx=00000000 edx=5d36edf0 esi=0616ffac edi=0616ffa0
eip=5c8b85c5 esp=05e1b320 ebp=05e1bfe8 iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010246
MSHTML!CTreeNode::ComputeFormatsHelper+0x53:
5c8b85c5 f7402400000300 test dword ptr [eax+24h],30000h ds:0023:05ba0024=00030000
```

Following another path by meeting a new constraint

Contd...

62

Pid 2152 - WinDbg:6.2.9200.16384 X86

File Edit View Debug Window Help

Disassembly

Offset: @\$scopeip

```
Sc18f7ed 83e4e0    and    esp, 0FFFFFFE0h
Sc18f7f0 83ec78    sub    esp, 78h
Sc18f7f3 alacf8215d mov    eax, dword ptr [MSHTML!__security_cookie (5d21f8ac)]
Sc18f7f8 33c4    xor    eax, esp
Sc18f7fa 89442474 mov    dword ptr [esp+74h], eax
Sc18f7fe 8bc1    mov    eax, ecx
Sc18f800 56    push   esi
Sc18f801 8b750c    mov    esi, dword ptr [ebp+0Ch]
Sc18f804 57    push   edi
Sc18f805 8b4824    mov    ecx, dword ptr [eax+24h] ds:0023:3f800024=?????????
Sc18f808 8b7d08    mov    edi, dword ptr [ebp+8]
Sc18f80b 89442410    mov    dword ptr [esp+10h], eax
Sc18f80f 897c241c    mov    dword ptr [esp+1Ch], edi
Sc18f813 f7c100000300 test   ecx, 30000h
Sc18f819 0f858b7a5d00 jne    MSHTML!CElement::ComputeFormats+0x55b (5c7672aa)
Sc18f81f f7c100000400 test   ecx, 40000h
Sc18f825 0f84dc215600 je     MSHTML!CElement::ComputeFormats+0x5eb (5c6f1a07)
Sc18f82b 8b4030    mov    eax, dword ptr [eax+30h]
```

Command

```
0:007> r eax=5b20000
0:007> !dptrace_trace C:\Users\rohitwas\Desktop\PoCs\IE11_log3
(868.54c): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.

A total of 26 instructions were traced and 23 were dumped to C:\Users\rohitwas\Desktop\PoCs\IE11_log3
Duration of this command in seconds: 0.000000

0:007> r
eax=3f800000 ebx=0e3d2fc0 ecx=3f800000 edx=00000000 esi=0644ac5c edi=0644ac5c
eip=5c18f805 esp=0644aea0 ebp=0644af24 iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00210202
MSHTML!CElement::ComputeFormats+0x1d:
Sc18f805 8b4824    mov    ecx, dword ptr [eax+24h] ds:0023:3f800024=?????????
```

More constraints

Contd...

63

The screenshot shows two windows side-by-side. On the left is the WinDbg debugger interface for process Pid 2152. The 'Disassembly' tab is selected, showing assembly code starting at offset @\$scopeip. The assembly code includes instructions like AND, SUB, MOV, XOR, PUSH, and MOV, dealing with registers esp, eax, ecx, edx, and memory locations such as [MSHTML!__sec]. The 'Command' tab below shows debugger commands and an access violation report. A stack trace is also present in the command window.

On the right is the 'Visual Data Tracer' tool. It has a 'File' menu and a list of 23 numbered traces. Each trace is a sequence of assembly instructions, such as:

- 1. 5c18f75c 8b4004 mov eax,dword ptr [eax+4] ds:0023:05b20004-0400b305
- 2. 5c7685f6 8b80b8000000 mov eax,dword ptr [eax+0B8h] ds:0023:05b300bc-bc00b405
- 3. 5c7685fc 8b400c mov eax,dword ptr [eax+0Ch] ds:0023:05b400c8=41414141
- 4. 5c7685ff 8b0f mov ecx,dword ptr [edi] ds:0023:0644ac5c=0000803f
- 5. 5c768601 89842494000000 mov dword ptr [esp+94h],eax ss:0023:0644afc8=e85e7406
- 6. 5c768608 8d842490000000 lea eax,[esp+90h]
- 7. 5c76860f 57 push edi
- 8. 5c768610 50 push eax
- 9. 5c768611 89bc2498000000 mov dword ptr [esp+98h],edi ss:0023:0644afc4=906ff005
- 10. 5c768618 e8cb71a2ff call MSHTML!CElement::ComputeFormats (5c18f7e8)
- 11. 5c18f7e8 8bff mov edi,edi
- 12. 5c18f7ea 55 push ebp
- 13. 5c18f7eb 8bec mov ebp,esp
- 14. 5c18f7ed 83e4e0 and esp,0FFFFFFE0h
- 15. 5c18f7f0 83ec78 sub esp,78h
- 16. 5c18f7f3 8b4004 mov eax,dword ptr [MSHTML!__security_cookie (5d21f8ac)] ds:0023:5d21f8ac=f3362f42
- 17. 5c18f7f8 33c4 xor eax,esp
- 18. 5c18f7fa 89442474 mov dword ptr [esp+74h],eax ss:0023:0644af1c=00000000
- 19. 5c18f7fe 8bc1 mov eax,ecx
- 20. 5c18f800 56 push esi
- 21. 5c18f801 8b750c mov esi,dword ptr [ebp+0Ch] ss:0023:0644af30=5cac4406
- 22. 5c18f804 57 push edi
- 23. 5c18f805 8b4824 mov ecx,dword ptr [eax+24h] ds:0023:3f800024=????????

Checking the taint source again. This particular execution run leads us to uncertainty and we aren't sure of an exploitable primitive yet.

Contd...

64

Pid 3204 - WinDbg:6.2.9200.16384 X86

File Edit View Debug Window Help

Disassembly

Offset: @\$scopeip

```
Sc8b85b6 f7470800001000 test dword ptr [edi+8],100000h
Sc8b85bd 7573 jne MSHTML!CTreeNode::ComputeFormatsHelper+0xe8 (5c8b8632)
Sc8b85bf 8b07 mov eax,dword ptr [edi]
Sc8b85c1 89442424 mov dword ptr [esp+24h],eax
Sc8b85c5 f7402400000300 test dword ptr [eax+24h],30000h ds:0023:0e858eb4=?????????
Sc8b85cc 0f8573010000 jne MSHTML!CTreeNode::ComputeFormatsHelper+0x1fb (5c8b8745)
Sc8b85d2 f7402400000400 test dword ptr [eax+24h],40000h
```

Command

```
ModLoad: 73600000 7360a000 C:\Windows\system32\ddrawex.dll
ModLoad: 5e870000 5e957000 C:\Windows\system32\DDRAW.dll
ModLoad: 73470000 73476000 C:\Windows\system32\DCIMAN32.dll
(c84.fc8): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=0e858e90 ebx=0f776fc0 ecx=00000000 edx=5d36edf0 esi=0e85cfac edi=0e85cfa0
eip=5c8b85c5 esp=05d8b500 ebp=05d8c1d0 iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010246
MSHTML!CTreeNode::ComputeFormatsHelper+0x53:
5c8b85c5 f7402400000300 test dword ptr [eax+24h],30000h ds:0023:0e858eb4=?????????
0:008> g
(c84.fc8): Access violation - code c0000005 (!!! second chance !!!)
eax=0e858e90 ebx=0f776fc0 ecx=00000000 edx=5d36edf0 esi=0e85cfac edi=0e85cfa0
eip=5c8b85c5 esp=05d8b500 ebp=05d8c1d0 iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010246
MSHTML!CTreeNode::ComputeFormatsHelper+0x53:
5c8b85c5 f7402400000300 test dword ptr [eax+24h],30000h ds:0023:0e858eb4=?????????
0:008> !dptrace_forward 4 200

Allocated range is
55e0000-55e1000,55f0000-55f1000,65f0000-65f1000,6600000-6601000
```

So we carry on another execution while trying to meet some other constraints and hit an alternate code path this time.

Contd...

65

Pid 3204 - WinDbg:6.2.9200.16384 X86

File Edit View Debug Window Help

Disassembly

Offset: @\$scopeip

```
773d71ac 8d642400 lea esp, [esp]
ntdll!KiFastSystemCall:
773d71b0 8bd4 mov edx, esp
773d71b2 0f34 sysenter
ntdll!KiFastSystemCallRet:
773d71b4 c3 ret
773d71b5 8da42400000000 lea esp, [esp]
773d71bc 8d642400 lea esp, [esp]
ntdll!Vt!+SystemCall+11.
```

Command

```
Allocated range is
55e0000-55e1000,55f0000-55f1000,65f0000-65f1000,6600000-6601000

0:008> r eax=55e0000
0:008> dd edi 11
0e85cfa0 0e858e90
0:008> ed 0e85cfa0 0e85cfa0
0:008> ed 0e85cfa0 55e0000
0:008> ed esp+24 55e0000
0:008> eb eax+30 1
0:008> eb eax+24 40000 1
          ^ Overflow error in 'eb eax+24 40000 1'
0:008> eb eax+24 40000
          ^ Overflow error in 'eb eax+24 40000'
0:008> ed eax+24 40000
0:008> !dptrace_trace C:\Users\rohitwas\Desktop\PoCs\log_final1.vdt
WARNING: Continuing a non-continuable exception

STATUS_STACK_BUFFER_OVERRUN encountered
WARNING: Step/trace thread exited

A total of 3664 instructions were traced and 2507 were dumped to C:\Users\rohi
Duration of this command in seconds: 1.000000
```

0:008>

Visual Data Tracer

File Analysis Help

```
67. 76f597e0 c1e902 shr ecx,2
68. 76f597e9 f3ab rep stos dword ptr es:[edi]
69. 76f597e9 f3ab rep stos dword ptr es:[edi]
70. 76f597e9 f3ab rep stos dword ptr es:[edi]
71. 76f597e9 f3ab rep stos dword ptr es:[edi]
72. 76f597e9 f3ab rep stos dword ptr es:[edi]
73. 76f597e9 f3ab rep stos dword ptr es:[edi]
74. 76f597e9 f3ab rep stos dword ptr es:[edi]
75. 76f597e9 f3ab rep stos dword ptr es:[edi]
76. 76f597e9 f3ab rep stos dword ptr es:[edi]
77. 76f597e9 f3ab rep stos dword ptr es:[edi]
78. 76f597e9 f3ab rep stos dword ptr es:[edi]
79. 76f597e9 f3ab rep stos dword ptr es:[edi]
80. 76f597e9 f3ab rep stos dword ptr es:[edi]
81. 76f597e9 f3ab rep stos dword ptr es:[edi]
82. 76f597e9 f3ab rep stos dword ptr es:[edi]
83. 76f597e9 f3ab rep stos dword ptr es:[edi]
84. 76f597f3 8b442408 mov eax,dword ptr [esp+8] ss:0023:05d8b454=80b4d805
85. 76f597f7 5f pop edi
86. 76f597f8 c3 ret
87. 5c2df8b9 33c0 xor eax,eax
88. 5c2df8b8 83c40c add esp,0Ch
89. 5c2df8c6 89442460 mov dword ptr [esp+60h],eax ss:0023:05d8b4c0=180c0000
90. 5c2df8c8 89442464 mov dword ptr [esp+64h],eax ss:0023:05d8b4c4=c06f770f
91. 5c2df8c8 89442468 mov dword ptr [esp+68h],eax ss:0023:05d8b4c8=e4b4d805
92. 5c2df8d4 c7442460fffff mov dword ptr [esp+60h],0FFFFFFFh ss:0023:05d8b4c0=00000000
93. 5c2df8dc 8d442460 lea eax,[esp+60h]
94. 5c2df8e0 c7442464fffff mov dword ptr [esp+64h],0FFFFFFFh ss:0023:05d8b4c4=00000000
95. 5c2df8e8 c7442468fffff mov dword ptr [esp+68h],0FFFFFFFh ss:0023:05d8b4c8=00000000
96. 5c2df8f0 898788020000 mov dword ptr [edi+28h],eax ds:0023:05d8b818=00000000
97. 5c2df8f6 8d442420 lea eax,[esp+20h]
98. 5c2df8f8 898784020000 mov dword ptr [edi+28h],eax ds:0023:05d8b814=00000000
99. 5c2df900 8b442410 mov eax,dword ptr [esp+10h] ss:0023:05d8e470=00005e05
100. 5c2df904 8b08 mov ecx,dword ptr [eax] ds:0023:055e0000=00005f05
101. 5c841a0e 6a01 push 1
102. 5c841a10 e88bf52e00 call MSHTML!_report_securityfailure (5cb30fa0)
103. 5cb30fa0 8bff mov edi,edi
104. 5cb30fa2 55 push ebp
```

Done!

We try another path this time by crafting some different values within the fake object, notably the value of 0x40000 in the dword @ fake_object+0x24. We also modify references to the same fake object in edi (CTreeNode *) and on the stack (esp+24). Hit a more interesting exception!

Contd...

Preliminary analysis shows us that the MSHTML!!report_securityfailure call was triggered due to a failed VTguard_check (next figure)

Contd...

67

```
; START OF FUNCTION CHUNK FOR ?ComputeFormats@CElement@@QAEJPAUCFormatInfo@@PAUCTreeNode@@Z
loc_63ABE7AB:
    mov     dword ptr [edi+284h], 0
    mov     dword ptr [edi+288h], 0
    mov     ecx, [eax]
    cmp     dword ptr [ecx+328h], offset __vtguard
    jnz     loc_63BE1A0E
```

```
mov     ecx, [ecx+51Ch] ; void *
mov     [esp+80h+var_74], ecx ; save ecx to local on stack which gets called into below
cmp     ecx, offset ?ComputeFormatsVirtual@CElement@@UAEJPAUCFormatInfo@@PAUCTreeNode@@Z ; CEElement::ComputeFormatsVirtual
jnz     loc_63C57240
```

```
loc_63C57240:           ; CTable::ComputeFormatsVirtual
    cmp     ecx, offset ?ComputeFormats@CElement@@QAEJPAUCFormatInfo@@PAUCTreeNode@@Z
    jnz     loc_63AC0FAE
```

```
; START OF FUNCTION CHUNK FOR ?ComputeFormats@CElement@@QAEJPAUCFormatInfo@@PAUCTreeNode@@Z
loc_63AC0FAE:
    mov     edi, esp
    push    esi
    push    [esp+84h+var_64]
    call    ds::__guard_check_icall_fptr
    mov     ecx, [esp+88h+var_70]
    call    [esp+88h+var_74] ; CODE EXEC!!!!
    jmp     loc_6369850F
; END OF FUNCTION CHUNK FOR ?ComputeFormats@CElement@@QAEJPAUCFormatInfo@@PAUCTreeNode@@Z
```

Confirm taint control and we influence the pointer which is dereferenced to do the vtguard check. That there is code execution right after the vtguard_check can either be looked into the debugger or within IDA for more clarity as shown above

Challenges & Limitations

- ▶ Determining the actual range of memory which needs to be traced. Determining this is easier for some cases (like file format bugs) whereas for browser based bugs this can be difficult (and sometimes unnecessary)
- ▶ Explosion and partial tainting (we assume full control when merging taint)
- ▶ Because the tracer outputs instruction information, it needs to understand the semantics of it (for example, source and destination operands):
 - ▶ It only supports the most basic x86 subset (no x87, MMX, XMM, etc) (future versions , also, helping is caring!)

Challenges & Limitations

- Another limitation of the approach is covering conditional code paths that hit only on certain values expected to be in the memory address (checking of reference counters, object type tag or some other metadata that affects the control flow of the program after the crash point)
 - Branch Explosion! Similar problems can arise with symbolic execution approach
- Manual analysis involves knowing where to break , where to start tracing, etc. The closer to the exception the better because of smaller traces and faster processing time by the analyzer
- Not a magic solution that works on its own without a skilled analyst. Not a one size fits all solution either. Meant to augment crash analysis.

Future

- ▶ We aren't soothsayers. More like sooth-slayers _./
- ▶ Please, read TODO.txt in the code trunk and send pull requests :p

Latest version of this presentation, paper, code and demos available at:

- ▶ <https://github.com/rrbranco/blackhat2016>

Acknowledgments

- ▶ Julio Auto for his previous work alongside one of the authors of this work (Rodrigo Branco) in implementing VDT (Vulnerability Data tracer) which was the original implementation of the backward taint tracing plugin
- ▶ David D. Rude(@bannedit) and Kiran Bandla(@kb) for ideas and feedback regarding the initial prototype of the forward trace
- ▶ **All of the other researchers who contributed to this field!**

Thanks!

Rohit Mothe (@rohitwas)

Senior Security Researcher

rohit.mothe *noSPAM* intel.com

Rodrigo Rubira Branco (@BSDaemon)

Principal Security Researcher

rodrigo.branco *noSPAM* intel.com