



# Malicious File for Exploiting Forensic Software

Takahiro Haruyama / Hiroshi Suzuki  
Internet Initiative Japan Inc.



# Who am I?

- Forensic Investigator & Malware Analyst @ Internet Initiative Japan
- Presentations
  - SANS DFIR Summit, Blackhat EU, RSA Conference Japan, etc..
- Hands-ons
  - CEIC, FIRST TC Kyoto/Lisbon, etc..
- Tools
  - EnCase EnScript, IDAPython script, etc..
  - <http://cci.cocolog-nifty.com/blog/>
- EnCase Certified Examiner since 2009
- twitter: [@cci\\_forensics](https://twitter.com/cci_forensics)

# Overview

- Background
- Fuzzing Oracle Outside In
- Anti-forensics by exploiting bugs
- Countermeasures
- Wrap-up



Background



# File Viewer in Forensic Software

- Forensic software needs a function viewing file content
  - Most commercial tools adopt the same library
    - e.g., EnCase, FTK, X-Ways, etc..
- Oracle Outside In Technology [1]
  - analyze/extract/convert over 500 different file types data
  - also used by enterprise software
    - e.g., Microsoft Exchange, Cisco Security Agent, IBM OmniFind Enterprise Edition, McAfee GroupShield, Symantec Enterprise Vault

# Motivation

- Several bugs of Oracle Outside In were reported last year [2]
- If still exploitable, lots of forensic investigators are exposed to risks when viewing/processing crafted malicious files
  - process hang-up
  - malware infection **with privilege**
  - other anti-forensic techniques
    - e.g., data alternation or hiding, evidence deletion
- Researched about the exploitability

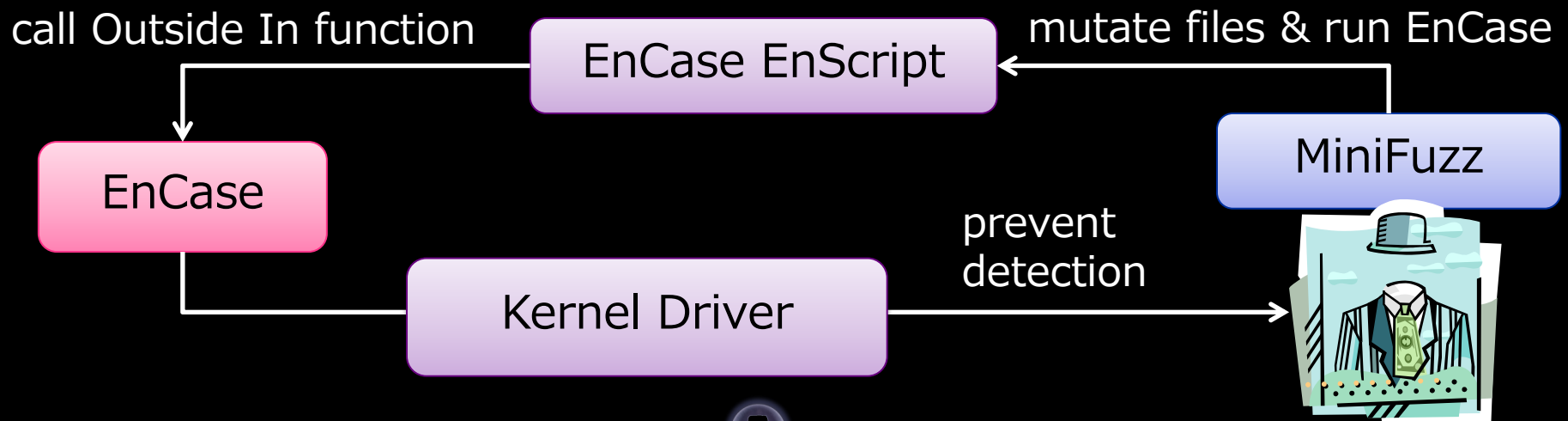


# Fuzzing Oracle Outside In



# Fuzzer Implementation

- The fuzzer using EnCase
  - MiniFuzz [3]
  - EnCase EnScript
  - kernel driver

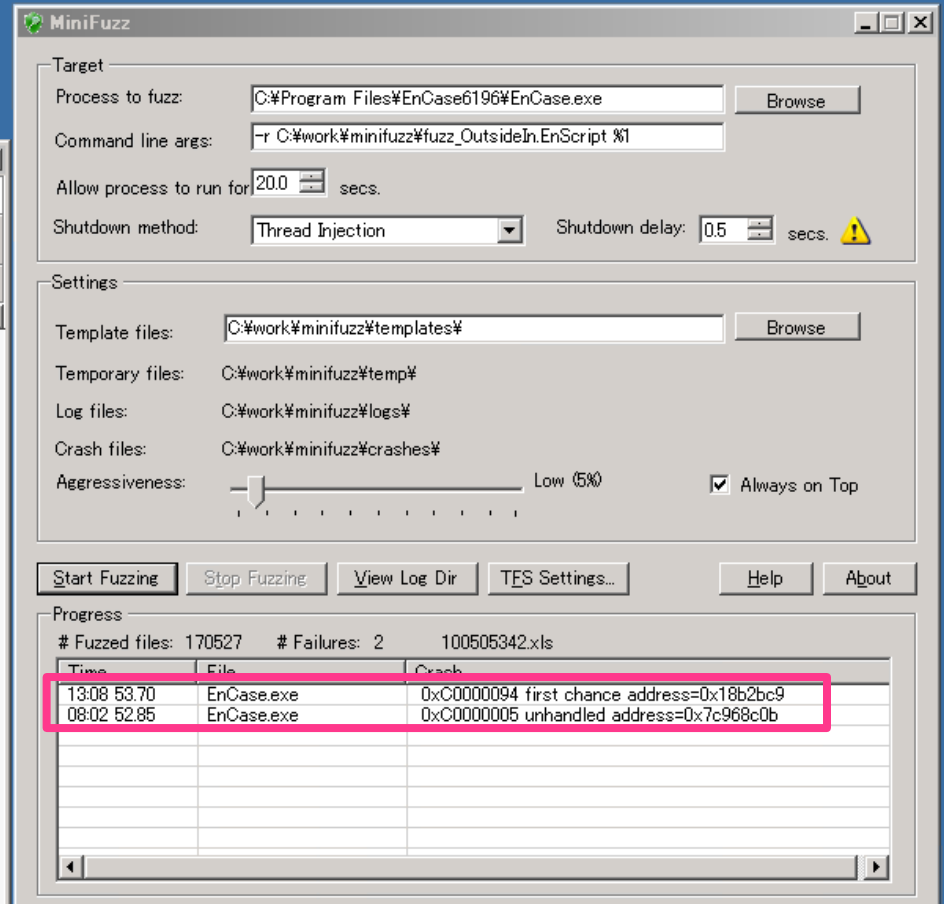
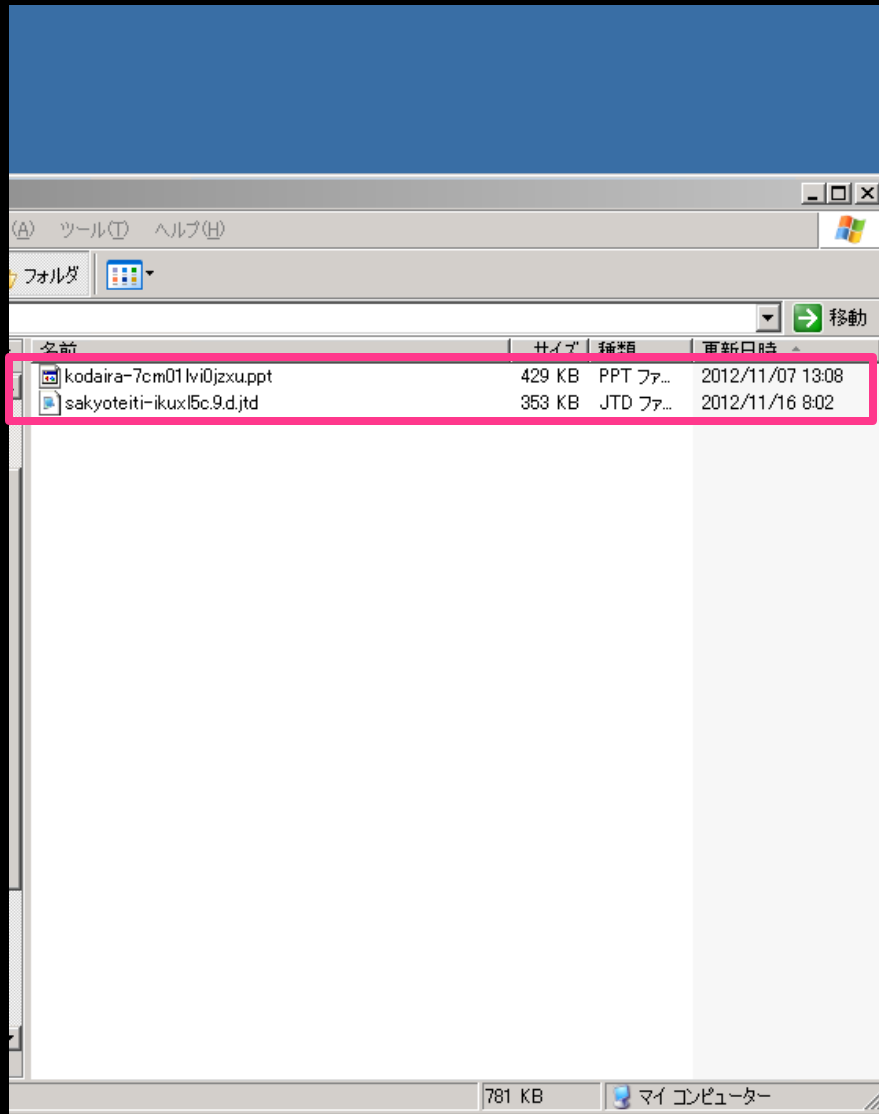




# EnScript for Fuzzer

- EnScript methods calling Outside In
  - DocumentClass
    - GetDocView
      - Returns a list containing one image for each page in the document
    - WriteTranscript
      - Returns a text extracted from the document
- One pitfall
  - EnScript can't receive command-line arguments
    - Use 3rd party tool [4]

# Check the Result





# Anti-forensics by exploiting bugs



# Anti-forensics by exploiting bugs

- Two examples
  - Process hang-up using infinite loop DoS vulnerability
  - Arbitrary code execution using heap overflow vulnerability
- Tested versions
  - 8.3.5 (too old!) on EnCase 6
    - To tell the truth, most EnCase users prefer 6 to 7 ☺
  - 8.3.7 on X-Ways Forensics
    - X-Ways recommends 8.3.7 instead of 8.4.1 for web history examination
  - 8.4.0 on EnCase 7

# Process Hang-up using Infinite Loop DoS Vulnerability

- Included in the function parsing Hangul Word Processor format document
  - A malicious file can cause infinite loop in vshwp2.dll
- JVN#68663052 <sup>[5]</sup> (CVE-2013-3776 <sup>[6]</sup>)
  - Affected version: 8.4.1 and earlier
  - Resolved on July 16th, 2013 <sup>[7]</sup>

```
push    ecx
mov     [esp+2ACh+var_274], ebx
call   sub_7591100
mov     ebx, [esp+2ACh+var_274]
add     esp, 18h
cmp     ebx, [edi+0Ch] ; constant value: ebx = 00000000, [edi+0Ch] = 00000000
jb     never_ending_loop
```

```
pop     esi
pop     ebp
pop     ebx
add     esp, 288h
retn
```

```
sub_7591100:
pop     ebx
add     esp, 288h
retn
```

# Demo

- 8.4.0 on EnCase 7

# Arbitrary Code Execution using Heap Overflow Vulnerability

```
xor    ecx, ecx
add    [esi+heap_struct1.field_0_buffer_len], -1
mov    ch, al          ; converting endian?
movzx  edi, cx
js     short loc_00401000
```

```
mov    ecx, [esi+10h]
movzx  eax, byte ptr [ecx]
add    ecx, 1
mov    [esi+10h], ecx
jmp    short loc_00401000
```

```
loc_00401000:
call   sub_00401000
movsx  eax, ax
```

```
loc_00401000:
mov    edx, [ebp+258h]
movzx  ecx, word ptr [esp+3Ch+wstr_len]
add    [esp+3Ch+arg_0], 0FFFFFFh
or     edi, eax
mov    eax, [esp+3Ch+current_offset]
mov    [edx+eax*2], di ; access violation
add    eax, 1
cmp    eax, ecx
mov    [esp+3Ch+current_offset], eax
```

- Included in the function parsing Ichitaro format document
  - A malicious file can overwrite heap chunks
- JVN#07497769 [8] (CVE-2013-3781 [9])
  - Affected version: 8.3.7 and earlier
  - Resolved on July 16th, 2013 [7]

# Overwriting Function Pointers

- Vista or later Windows OS adopt various mitigation techniques to prevent an exploitation of heap overflow <sup>[10]</sup>
  - Look-aside lists have been replaced by the Low Fragmentation Heap (LFH)
  - heap entry metadata randomization
  - randomized heap base address
  - etc...
- One promising method is overwriting function pointers in heap chunks
  - But the offset values to them are not constant



# Heap Spraying

- Heap spraying is a payload delivery technique
  - It allows us to put our shellcodes at a predictable address (e.g., 0x0c0c0c0c)
- We need to fill chunks of memory in the heap before gaining control over EIP
- How?
  - Javascript or vbscript in web browsers
  - Javascript or ActionScript in Adobe Reader
  - ? in forensic software

# Heap Spraying with Bitmap Images

- Bitmap heap spraying [11] is an effective technique because forensic investigators often examine image files
- Most forensic software supports displaying several images at the same time
  - EnCase
    - “Bookmark Page as Image” in Doc view
    - Make the bookmarked images “set-included”
  - X-Ways Forensics
    - Double-click image files

# Heap Spraying with Bitmap Images: Example on EnCase 6

7122CE26	8BA8 0010000	MOV EBP,DWORD PTR DS:[EAX+100]			
7122CE2C	8B5D 0C	MOV EBX,DWORD PTR SS:[EBP+C]			
7122CE2F	837B 0C 00	CMP DWORD PTR DS:[EBX+C],0			
7122CE33	56	PUSH ESI			
7122CE34	57	PUSH EDI			
7122CE35	74 0B	JE SHORT scocch.7122CE42			
7122CE37	8B43 0C	MOV EAX,DWORD PTR DS:[EBX+C]			
7122CE3A	8B08	MOV ECX,DWORD PTR DS:[EAX]			
7122CE3C	50	PUSH EAX			
7122CE3D	FFD1	CALL ECX			
7122CE3F	83C4 04	ADD ESP,4			
7122CE42	8B43 20	MOV EAX,DWORD PTR DS:[EBX+20]			
7122CE45	85C0	TEST EAX,EAX			
7122CE47	8B35 00E02271	MOV ESI,DWORD PTR DS:[<&KERNEL32.H kernel32.HeapFree			
7122CE4D	8B3D 04E02271	MOV EDI,DWORD PTR DS:[<&KERNEL32.G kernel32.GetProcessHeap			
7122CE53	74 08	JE SHORT scocch.7122CE5D			
7122CE55	50	PUSH EAX			
7122CE56	6A 00	PUSH 0			
7122CE58	FFD7	CALL EDI			
7122CE5A	50	PUSH EAX			
7122CE5B	FFD6	CALL ESI			
7122CE5D	53	PUSH EBX			
7122CE5E	330B	XOR EBX,EBX			
7122CE60	53	PUSH EBX			
7122CE61	FFD7	CALL EDI			
7122CE63	50	PUSH EAX			
7122CE64	FFD6	CALL ESI			
7122CE66	8B55 04	MOV EDX,DWORD PTR SS:[EBP+4]			
7122CE69	52	PUSH EDX			
7122CE6A	53	PUSH EBX			
7122CE6B	FFD7	CALL EDI			

Address	Hex dump	ASCII
0C0C0C0C	0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C	.....
0C0C0C1C	0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C	.....
0C0C0C2C	0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C	.....
0C0C0C3C	0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C	.....
0C0C0C4C	0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C	.....
0C0C0C5C	0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C	.....

Address	Hex dump	ASCII
00120FA4	00 12 0F A4	
7122CE3F	71 22 CE 3F	RETURN to scocch.7122CE3F
0012DFA8	00 12 DF A8	
0C0C0C0C	0C 0C 0C 0C	
0012DFAC	00 12 DF AC	RETURN to kernel32.HeapFree
0012DFB0	00 12 DF B0	
00290688	00 29 06 88	
0012DFB4	00 12 DF B4	
00000014	00 00 00 14	
0012DFB8	00 12 DF B8	
00000000	00 00 00 00	
0012DFBC	00 12 DF BC	RETURN to scocch.712211DD from scoc

Register	Value	Comment
EAX	0C0C0C0C	
ECX	0C0C0C0C	
EDX	00000000	
EBX	0C0C0C0C	
ESP	0012DFA4	
EBP	0C0C0C0C	
ESI	00290688	
EDI	77C1F198	kernel32.HeapFree
EIP	0C0C0C0C	
C 0	ES 0023	32bit 0(FFFFFFFF)
P 1	CS 001B	32bit 0(FFFFFFFF)
A 0	SS 0023	32bit 0(FFFFFFFF)
Z 0	DS 0023	32bit 0(FFFFFFFF)
S 0	FS 003B	32bit 7FFDE000(FFF)
T 0	GS 0000	NULL
D 0		
O 0	LastErr	ERROR_SUCCESS (00000)
EFL	00000206	(NO,NB,NE,A,NS,PE,GE)
ST0	empty	0.0
ST1	empty	0.0
ST2	empty	0.0
ST3	empty	0.0
ST4	empty	0.0
ST5	empty	0.0
ST6	empty	0.0
ST7	empty	149837.35935484952640
		3 2 1 0 E S P
FST	0120	Cond 0 0 0 1 Err 0 0 1
FCW	027F	Prec NEAR,53 Mask 1

# Demo

- 8.3.5 on EnCase 6
- 8.3.7 on X-Ways Forensics

# Success Probability of the Heap Overflow Exploitation

- Not necessarily succeed
  - Function pointers called in a short time should be included in overwritten area
    - e.g., sccch/sccut/sccvw, ole32, etc..
  - current success probability
    - EnCase 6
      - 40-50%
    - X-Ways
      - 10-20%
  - To improve the probability, we need to manipulate heap chunk layout before causing overflow ☹



# Countermeasures



# Prevention of Arbitrary Code Execution using Heap Overflow

- Developers should
  - check buffer boundary to prevent overflow
  - enable “HeapEnableTerminateOnCorruption”
    - Use HeapSetInformation [12] API
    - If a heap corruption detected, the process terminates immediately
      - 0xc0000374 (STATUS\_HEAP\_CORRUPTION)

o\_0 - main thr3ad, module EnCase

```
00A5B140 FF15 C834B400 CALL DWORD PTR DS:[&kernel32.HeapSetInformation] kernel32.HeapSetInformation
00A5B153 895D FC      MOV DWORD PTR SS:[EBP-4],EBX
00A5B156 64:A1 18000000 MOV EAX,DWORD PTR FS:[18]
00A5B15C 8B70 04      MOV ESI,DWORD PTR DS:[EAX+4]
00A5B15F 895D DC      MOV DWORD PTR SS:[EBP-24],EBX
00A5B162 BF 2C92F300 MOV EDI,EnCase.00F3922C
00A5B167 53          PUSH EBX
00A5B168 56          PUSH ESI
00A5B169 57          PUSH EDI
00A5B16A FF15 2834B400 CALL DWORD PTR DS:[&kernel32.InterlockedCompareExchange] kernel32.InterlockedCompareExchange
00A5B170 3BC3        CMP EAX,EBX
00A5B172 74 19      JE SHORT EnCase.00A5B180
00A5B174 3BC6        CMP EAX,ESI
00A5B176 75 08      JNZ SHORT EnCase.00A5B180
00A5B178 33F6       XOR ESI,ESI
00A5B17A 46          INC ESI
00A5B17B 8975 DC      MOV DWORD PTR SS:[EBP-24],ESI
00A5B17E 74 10      JMP SHORT EnCase.00A5B190
```

Registers (FPU)

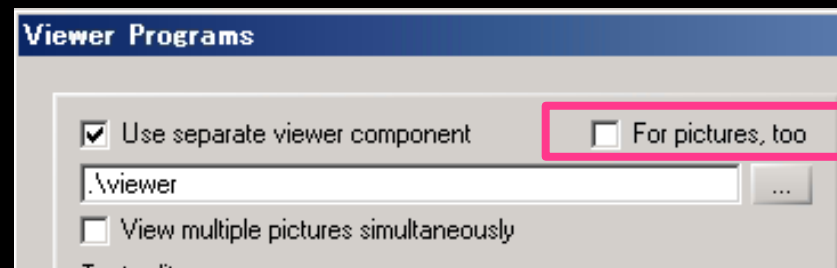
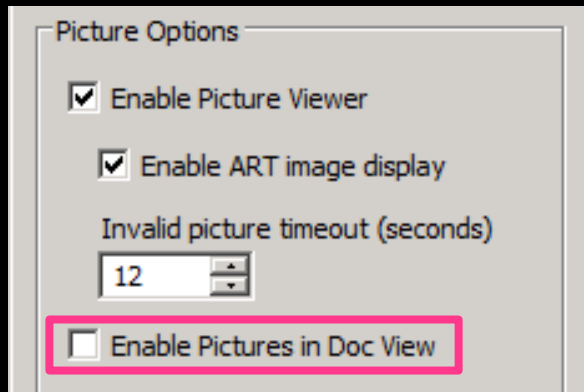
```
EAX 025AFE44
ECX 00020000
EDX 00000000
EBX 00000000
ESP 025AFE24
EBP 025AFE00
ESI 000000B8
EDI 00401000 EnCase.00401000
EIP 00A5B140 EnCase.00A5B140
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDF000(FFF
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (000
EFL 00000246 (NO,NB,E,BE,NS,PE,
STA empty -UNORM D008 01050104
```

DS:[00B43408]=7C839499 (kernel32.HeapSetInformation)

Address	Hex dump	ASCII
00B43000	ED 7E D8 77 51 4D DC 77 2C D5 DB 77 8E 4E DA 77	025AFE20 00000000 Arg1 = 00000000
00B43010	AE 69 DA 77 66 4C DA 77 96 15 DA 77 E5 6C D9 77	025AFE24 00000000 Arg2 = 00000000
00B43020	E5 EC D8 77 C4 54 D9 77 BF 9A DA 77 B6 51 D9 77	025AFE28 00000000 Arg3 = 00000000
00B43030	A0 42 D9 77 E7 EA D8 77 BB 7A D8 77 F4 E9 D8 77	025AFE2C 00000000 Arg4 = 00000000
00B43040	52 78 D8 77 E0 4C DA 77 CA 7F D8 77 11 82 D8 77	025AFE30 CB8E105A
00B43050	59 5B D9 77 F9 1B DC 77 21 1B DC 77 D3 79 DA 77	025AFE34 00401000 EnCase.00401000

# Heap spray Prevention

- Disable the operation for bitmap heap spray
  - EnCase 6
    - [Tools] -> [Options] -> [Global]
    - uncheck "Enable Pictures in Doc View"
  - X-Ways Forensics
    - [Options] -> [Viewer Programs]
    - uncheck "For pictures, too"





# Common Countermeasures

- Use the latest version
  - 8.4.0 or later seems to fix most bugs discovered by my fuzzer
    - Do not use EnCase 6 / X-Ways with 8.3.7
- Do not install file viewer
  - We can install EnCase/X-Ways without Outside In
- Configure for exploit mitigation [13]
  - e.g., EMET, AppLocker, etc...



Wrap-up



# Wrap-up

- Know the risk when examining unknown files acquired in forensic investigation
  - The file viewer component is fragile
    - The two bugs may be just a little bit of the problem
  - Investigators should pay attention to the security settings of their workstations
- Forensic software vendors tend to use an old version of the component
  - They should update their products as soon as the latest version is released

Questions?  
(twitter: @cci\_forensics)

Please complete the Speaker  
Feedback Surveys

# References

- [1] Oracle Outside In Technology  
<<http://www.oracle.com/us/technologies/embedded/025613.htm>>
- [2] Oracle Outside In contains multiple exploitable vulnerabilities  
<<https://www.kb.cert.org/vuls/id/118913>>
- [3] SDL MiniFuzz File Fuzzer  
<<http://www.microsoft.com/en-us/download/details.aspx?id=21769>>
- [4] Command Line DLL.zip <<http://www.swiftforensics.com/p/downloads.html>>
- [5] JVN#68663052 Oracle Outside In vulnerable to denial-of-service (DoS)  
<<http://jvn.jp/en/jp/JVN68663052/index.html>>
- [6] CVE-2013-3776 <<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-3776>>
- [7] Oracle Critical Patch Update Advisory - July 2013  
<<http://www.oracle.com/technetwork/topics/security/cpujuly2013-1899826.html>>
- [8] JVN#07497769 Oracle Outside In vulnerable to buffer overflow  
<<http://jvn.jp/en/jp/JVN07497769/index.html>>
- [9] CVE-2013-3781 <<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-3781>>
- [10] Preventing the exploitation of user mode heap corruption vulnerabilities  
<<http://blogs.technet.com/b/srd/archive/2009/08/04/preventing-the-exploitation-of-user-mode-heap-corruption-vulnerabilities.aspx>>
- [11] Exploit writing tutorial part 11 : Heap spraying Demystified  
<<https://www.corelan.be/index.php/2011/12/31/exploit-writing-tutorial-part-11-heap-spraying-demystified/>>
- [12] HeapSetInformation function  
<[http://msdn.microsoft.com/en-us/library/windows/desktop/aa366705\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/aa366705(v=vs.85).aspx)>
- [13] Forensics Software and Oracle Outside In  
<[https://www.cert.org/blogs/certcc/2013/07/forensics\\_software\\_and\\_oracle.html](https://www.cert.org/blogs/certcc/2013/07/forensics_software_and_oracle.html)>