intel Security

StackPivotChecker
- Instrumentation Techniques and ROP Exploit Rapid Analysis

TM

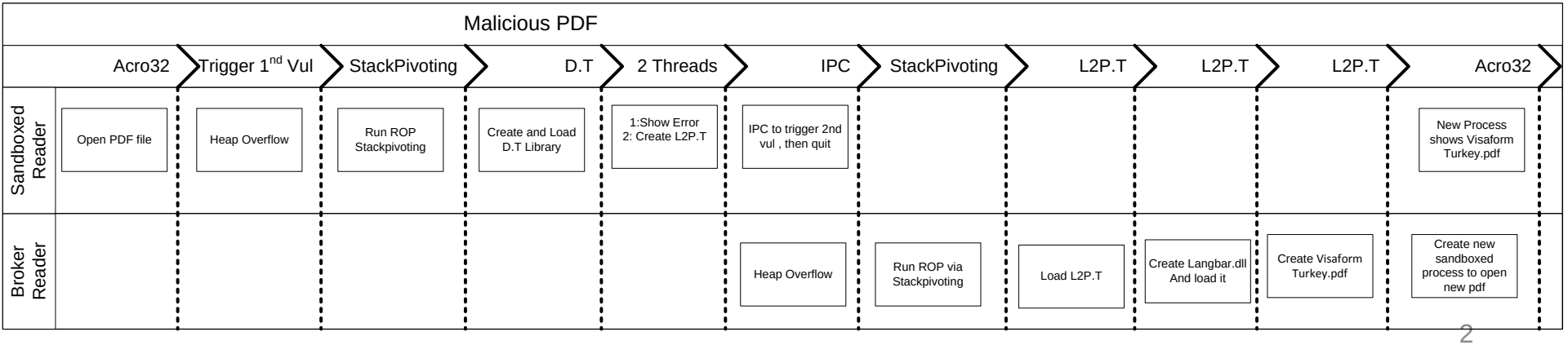
- Xiaoning Li (Intel Labs)
- Haifei Li (McAfee Labs)

Agenda

- APSA13-02 PDF Zero Day
- Challenges in Vulnerability/Exploit Analysis
- ROP and StackPivoting
- Instrumentation and Trace Techniques
- StackPivotingChecker with BTF
- Case Study
- Conclusion
- Q&A

APSA13-02 PDF Zero Day

- Reported by FireEye in February 2013
- Best Client-Side Bug: CVE-2013-0641
- Sophisticated ROP only, without shellcode
- First public in-the-wild exploit of Adobe Sandbox Bypassing
- Very complex exploit and ROP exploits without shellcode

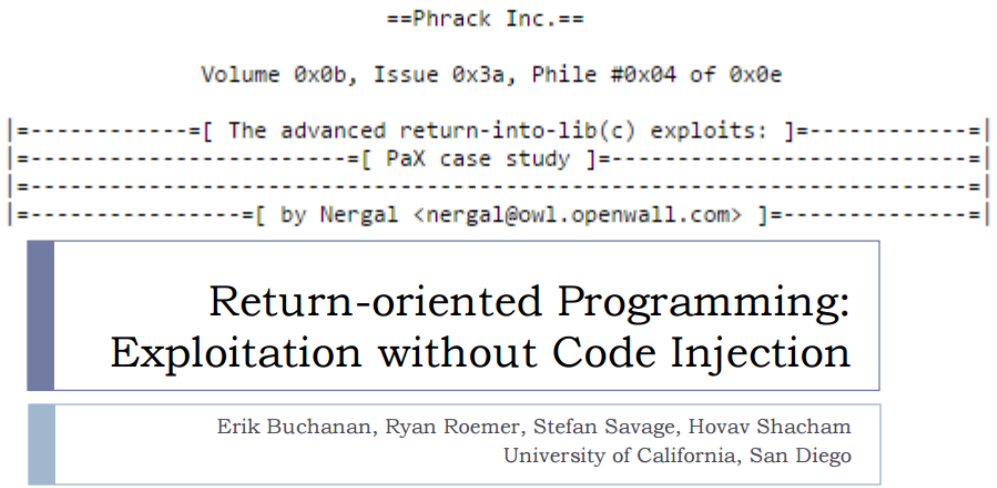


Challenges in APSA 13-02 Analysis

- Vulnerability trigger point is far away from final malicious code
- ROP only and thousands of ROP gadgets!
- Lots of manpower to reverse whole attack chain from vulnerability trigger point to final payload

ROP and StackPivoting

- ROP (return-oriented programming) to reuse oriented code



==Phrack Inc.==

Volume 0x0b, Issue 0x3a, Phile #0x04 of 0x0e

|=------------=[The advanced return-into-lib(c) exploits:]=------------=|
|=-------------------------=[PaX case study]=-------------------------=|
|=--=|
|=-----------------=[by Nergal <nergal@owl.openwall.com>]=--------------=|

Return-oriented Programming: Exploitation without Code Injection

Erik Buchanan, Ryan Roemer, Stefan Savage, Hovav Shacham
University of California, San Diego

- ROP requires control of the stack
- When attacker doesn't control, can pivot ESP to point to controlled memory
- Unintended instructions often have stackpivoting gadgets

```
text:209B9F50
text:209B9F51
text:209B9F52
text:209B9F53
text:209B9F56
```

```
push    eax
pop     esp
pop     ecx
movzx   eax, ax
retn
```

```
.text:108FDC60 ; ---------------------------------
.text:108FDC60                 xchg    eax, esp
.text:108FDC61                 retn
.text:108FDC61 ; ---------------------------------
```

Instrumentation/Trace Techniques

- API Hooking
- Single Step
- Branch Trap Flag
- Performance Monitor Counter
- Processor Trace

API Hooking

- Inline hooking at callee
- Have to modify API code with a JMP-like branch instruction

```
kernel32!VirtualProtectStub:
76692bcd e93ed595f9      jmp     6fff0110
76692bd2 5d              pop     ebp
76692bd3 e900f5fbff      jmp     kernel32!VirtualProtect (766520d8)
```

```
6fff0110 68516c38d4      push    0D4386C51h
6fff0115 60              pushad
6fff0116 9c              pushfd
6fff0117 54              push    esp
6fff0118 e893d399fd      call    emet+0x4d4b0 (6d98d4b0)
6fff011d 9d              popfd
6fff011e 61              popad
6fff011f 83c404          add     esp,4
6fff0122 8bff            mov     edi,edi
6fff0124 55              push    ebp
6fff0125 8bec            mov     ebp,esp
6fff0127 e9a62a6a06      jmp     kernel32!VirtualProtectStub+0x5 (76692bd2)
```

Single Step

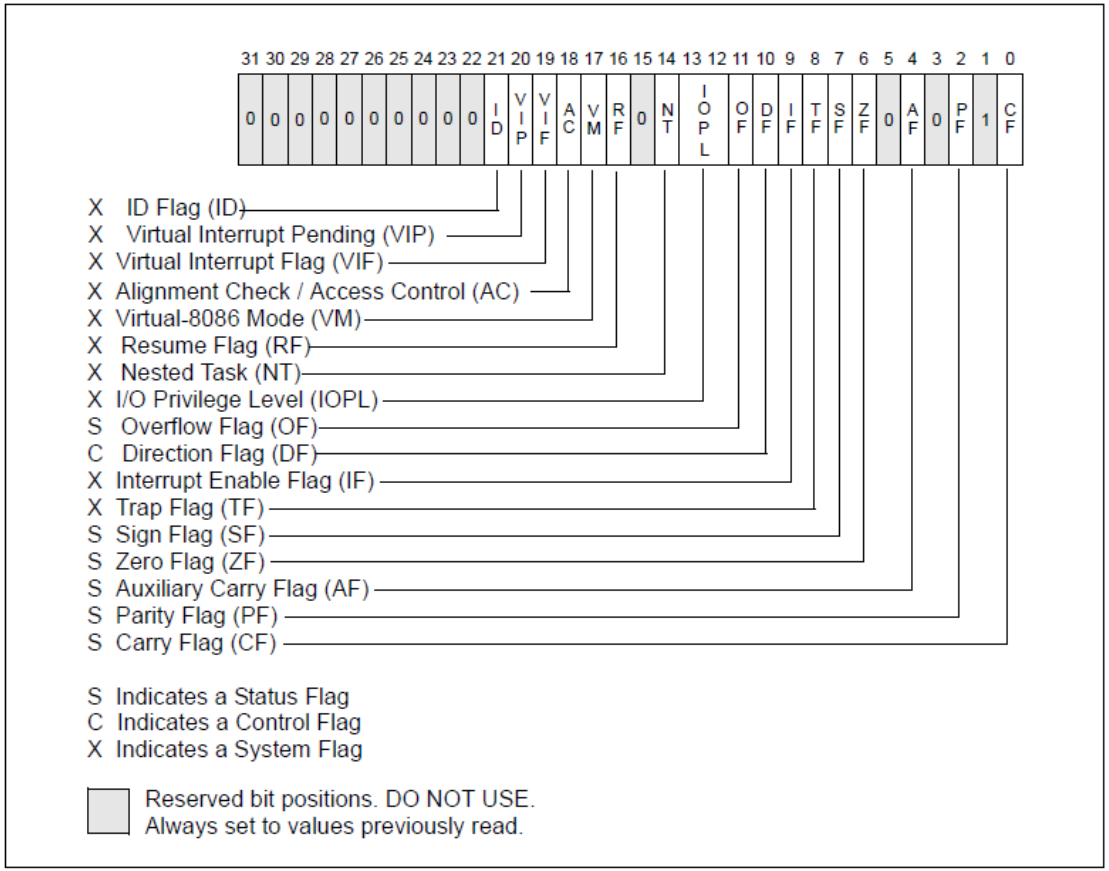


Figure 3-8. EFLAGS Register

TF (bit 8) **Trap flag** — Set to enable single-step mode for debugging; clear to disable single-step mode.

Single Step + BTF

- BTF is the flag in MSR_DEBUGCTLA MSR
- Used to enable single-step on branches

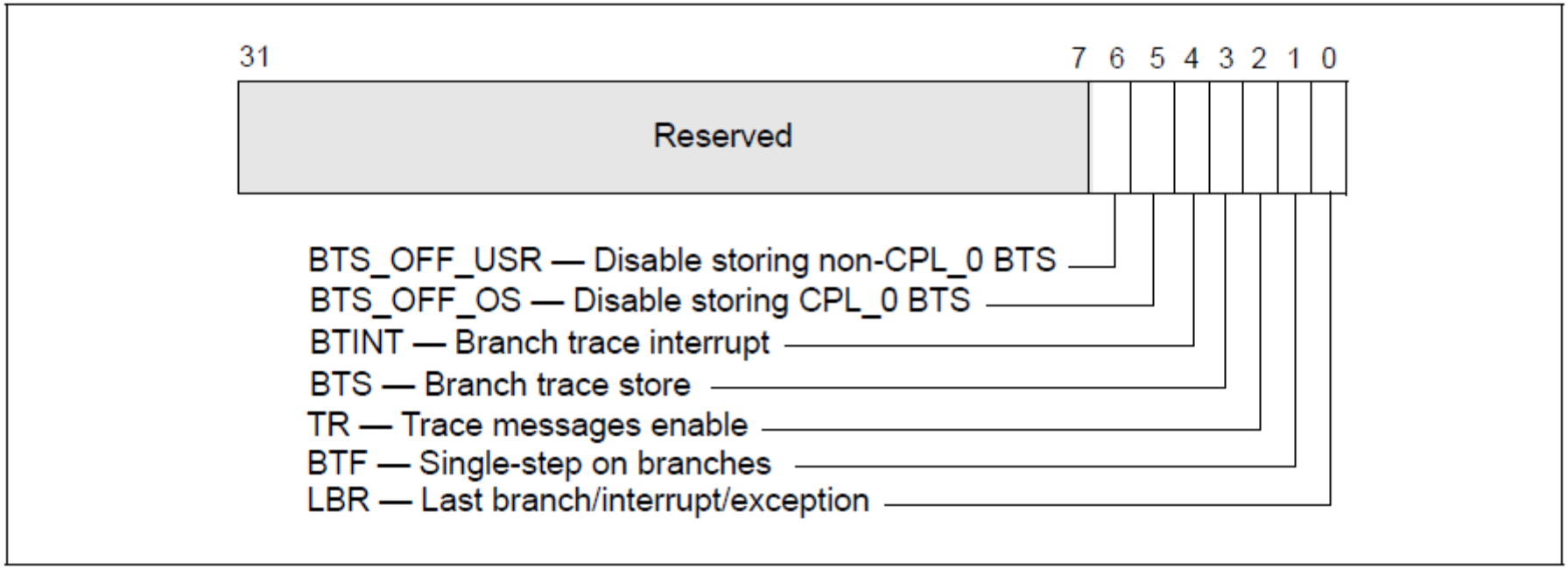


Figure 17-12. MSR_DEBUGCTLA MSR for Pentium 4 and Intel Xeon Processors

Performance Monitor Counter

Transparent ROP Detection using CPU Performance Counters

他山之石，可以攻玉
Stones from other hills may serve to polish jade

Xiaoning Li Intel Labs
Michael Crouse Harvard University

THREADS Conference 2014

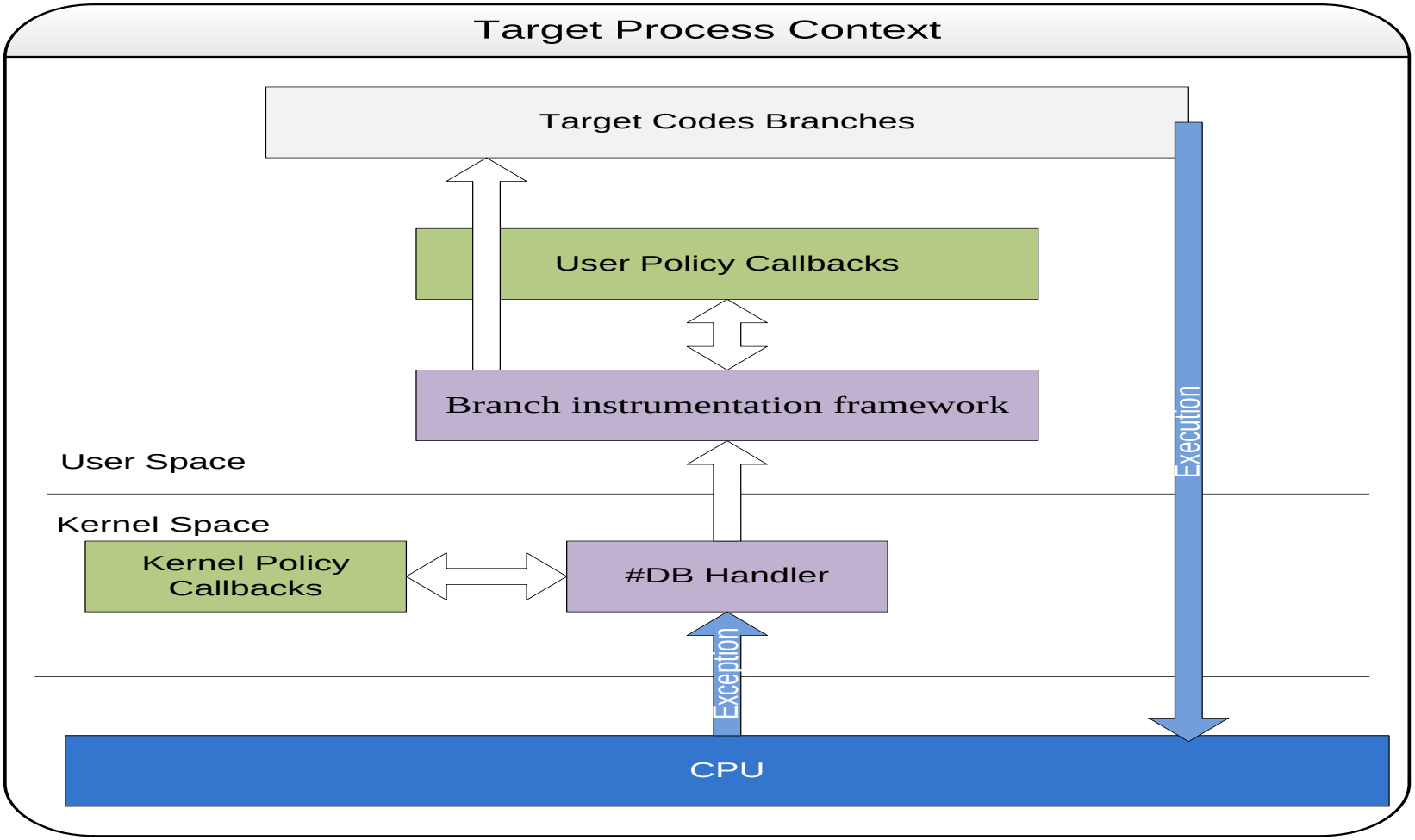
Processor Trace

CHAPTER 36
INTEL® PROCESSOR TRACE

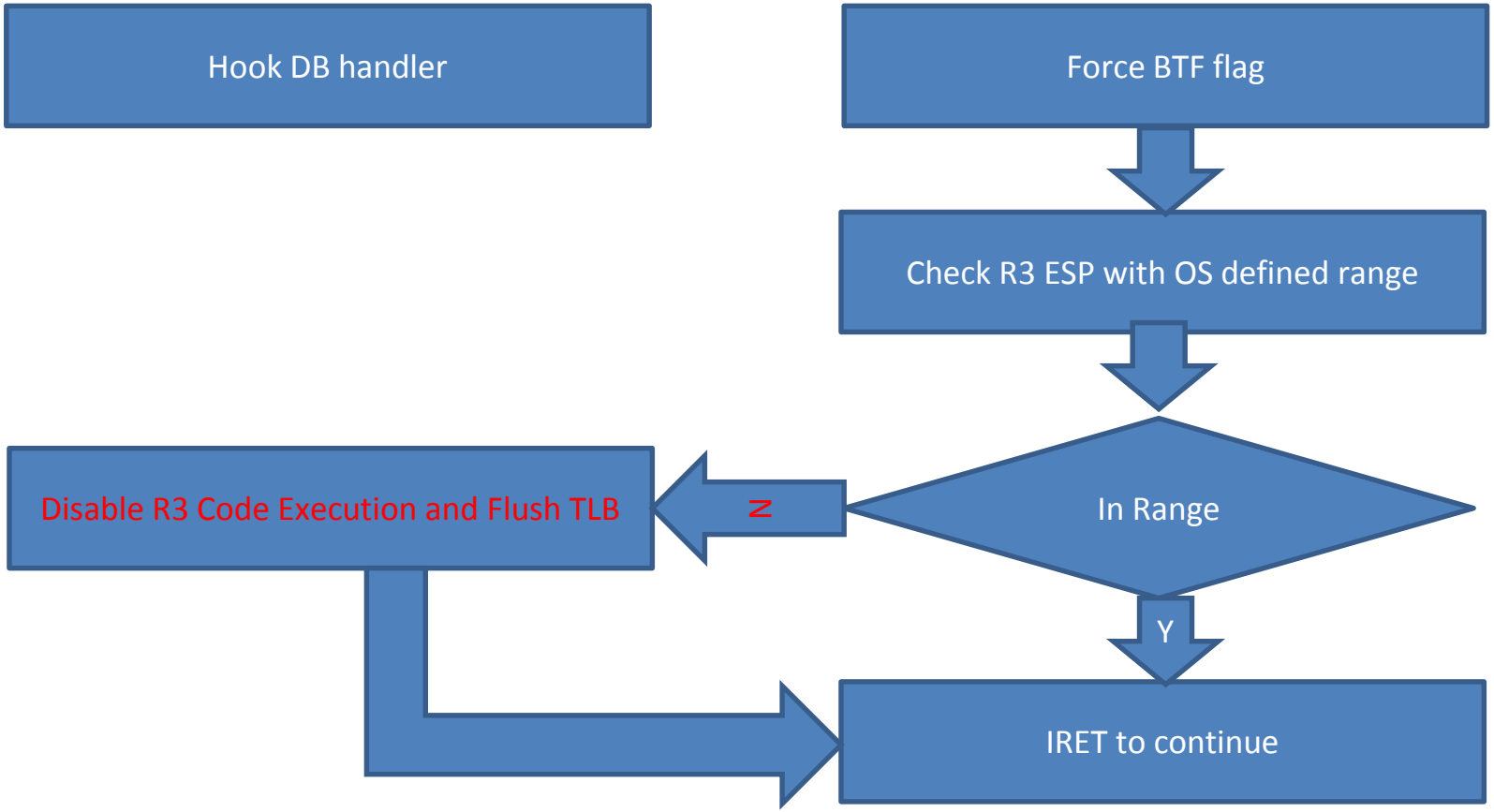
36.1 OVERVIEW

Intel® Processor Trace (**Intel PT**) is an extension of Intel® Architecture that captures information about software execution using dedicated hardware facilities that cause only minimal performance perturbation to the software being traced. This information is collected in **data packets**. The initial implementations of Intel PT offer **control flow tracing**, which generates a variety of packets to be processed by a software decoder. The packets include timing, program flow information (e.g. branch targets, branch taken/not taken indications) and program-induced mode related information (e.g. Intel TSX state transitions, CR3 changes). These packets may be buffered internally before being sent to the memory subsystem or other output mechanism available in the platform. Debug software can process the trace data and reconstruct the program flow.

StackPivotChecker with BTF



StackPivotChecker Demo



Stack Pivoting Detection

- Validate stack limitation
- From FS:18h
- Use _NT_TIB or _TEB to get stack limitation

```
0:000> dt _NT_TIB
ntdll!_NT_TIB
   +0x000 ExceptionList     : Ptr32 _EXCEPTION_REGISTRATION_RECORD
   +0x004 StackBase         : Ptr32 Void
   +0x008 StackLimit        : Ptr32 Void
   +0x00c SubSystemTib      : Ptr32 Void
   +0x010 FiberData         : Ptr32 Void
   +0x010 Version           : Uint4B
   +0x014 ArbitraryUserPointer : Ptr32 Void
   +0x018 Self              : Ptr32 _NT_TIB
```

XD Bit in PAE PTE

Table 4-11. Format of a PAE Page-Table Entry that Maps a 4-KByte Page (Contd.)

Bit Position(s)	Contents
11:9	Ignored
(M–1):12	Physical address of the 4-KByte page referenced by this entry
62:M	Reserved (must be 0)
63 (XD)	If IA32_EFER.NXE = 1, execute-disable (if 1, instruction fetches are not allowed from the 4-KByte page controlled by this entry; see Section 4.6); otherwise, reserved (must be 0)

Disable Execution and Flush TLB

- #define PTE_BASE_X86_PAE 0xc0000000
- #define PDE_BASE_X86_PAE 0xc0600000
- #define MiGetPteAddress_X86_PAE(va) (((ULONG)(((((ULONG)(va)) >> 12) << 3) & 0x7FFFF8) + PTE_BASE_X86_PAE))
- #define MiGetPdeAddress_X86_PAE(va) (((ULONG)(((((ULONG)(va)) >> 21) << 3) & 0x3FF8) + PDE_BASE_X86_PAE))

```
mov eax, CurrentEIP
shr  eax, 0Ch
shl  eax, 3
and  eax, 7FFFF8h
sub  eax, 40000000h
mov  dword ptr [eax+4], 80000000h
mov  eax, cr3
mov  cr3, eax
```

Case Study

- CVE 2010 3654 Exploit
- https://www.exploit-db.com/exploits/17187/
- Download the vulnerable Flash binary at http://fpdownload.macromedia.com/get/flashplayer/installers/archive/fp10.1_archive.zip

CVE2010-3654 Exploit Overview

Title	**: Adobe Flash player Action script type confusion**
Version	**: flash10h.dll**
Discovery	**: Malware writers**
Exploit	**: www.abysssec.com**
Vendor	**: http://www.adobe.com**
Impact	**: Critical**
Contact	**: info [at] abysssec.com**
Twitter	**: @abysssec**
CVE	**: CVE-2010-3654**

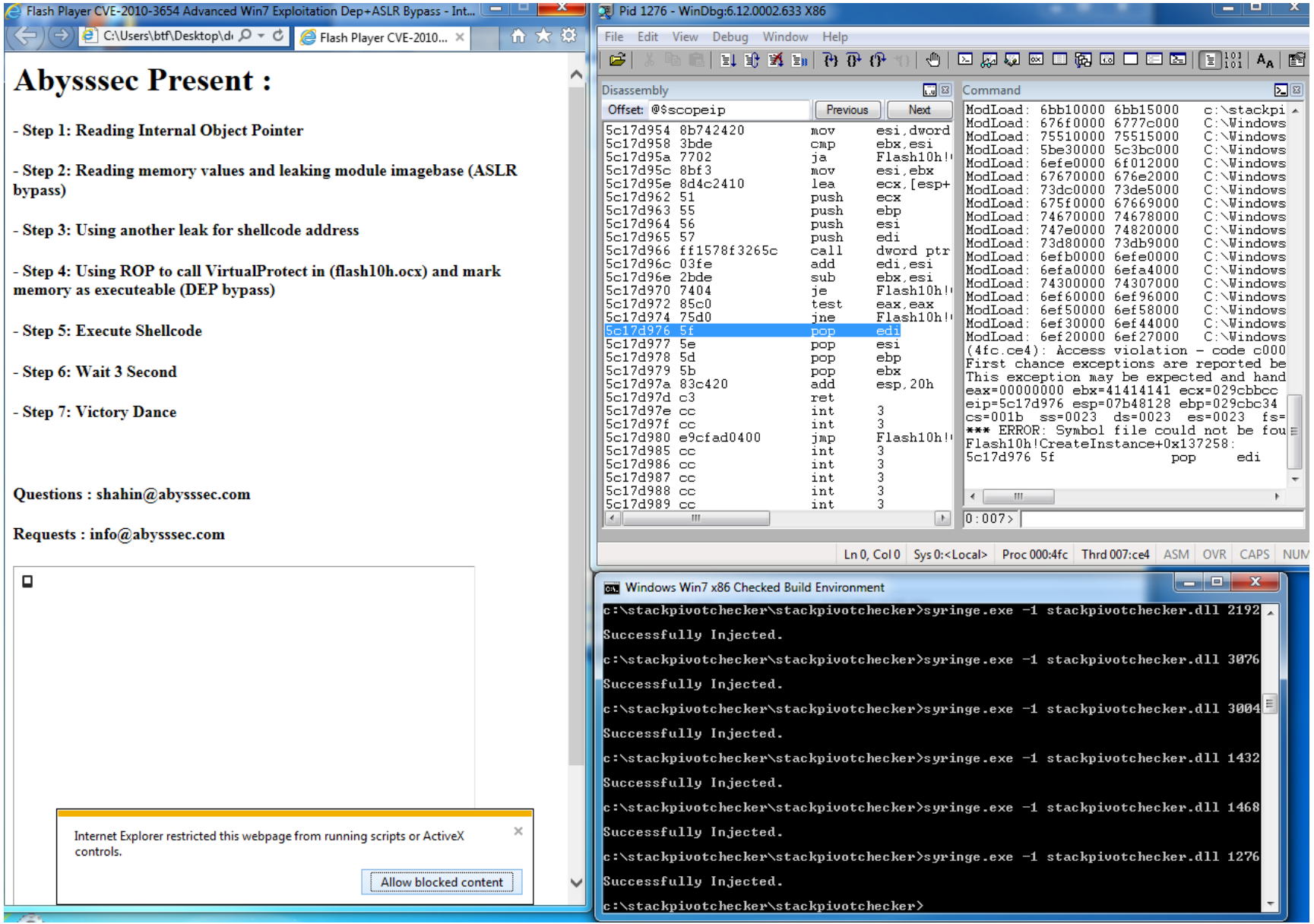
ROP Gadgets

- Start ROP

```
0abaffc9 52              push    edx
0abaffca 6a00            push    0
0abaffcc 51              push    ecx
0abaffcd ffd0            call    eax {Flash10h+0xfa851 (67efa851)}
```

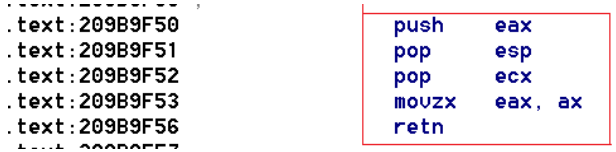
```
67efa851 87cc            xchg    ecx,esp
67efa853 0400            add     al,0
67efa855 5f              pop     edi
67efa856 8bc6            mov     eax,esi
67efa858 5e              pop     esi
67efa859 5b              pop     ebx
67efa85a 83c440          add     esp,40h
67efa85d c20800          ret     8
```

StackPivotChecker Demo



Zero-Day Analysis Process

- Set up breakpoint in common APIs
- Trace back to trigger point
- Reverse engineer more details for root cause analysis
- Our automation tool detected CVE-2013-0640 once stack pivoting done within 10 minutes running real exploit sample



Conclusion

- Stack pivoting is the common technique used by ROP exploit.
- Stack pivoting could be detected automatically via Single Step + BTF.
- The performance penalty for offline exploit analysis is acceptable.

References

[1] Practical Return-Oriented Programming , Dino Dai Zovi, RSA 2010

[2] Return Oriented Exploitation, Dino Dai Zovi, Black Hat 2010

[3] Security Mitigations for Return-Oriented Programming Attacks, Piotr Bania

[4] White Phosphorus Exploit Pack Sayonara ASLR DEP Bypass Technique, http://www.whitephosphorus.org/sayonara.txt

[5] Active Zero-Day Exploit Targets Internet Explorer Flaw, http://blogs.mcafee.com/mcafee-labs/active-zero-day-exploit-targets-internet-explorer-flaw

[6] Pwn2Own 2010 Windows 7 Internet Explorer 8 exploit, Peter Vreugdenhil, 2010

[7] Advanced Exploitation of Internet Explorer Heap Overflow (Pwn2Own 2012 Exploit), Alexandre Pelletier, 2012

[8] Smashing the Heap with Vector: Advanced Exploitation Technique in Recent Flash Zero-day Attack, Haifei Li

[9] Interpreter Exploitation: Pointer Inference and JIT Spraying, Dion Blazakis, Black Hat DC 2010

[10] Attacking the Windows 7/8 Address Space Randomization, http://kingcope.wordpress.com/2013/01/24/attacking-the-windows-78-address-space-randomization

[11] DEP/ASLR bypass without ROP/JIT, Yang Yu, CanSecWest 2013

[12] The BlueHat Prize, http://www.microsoft.com/security/bluehatprize

[13] kBouncer: Efficient and Transparent ROP Mitigation, Vasilis Pappas

[14] On the Effectiveness of Address-Space Randomization, Hovav Shacham et al., 2004.

[15] Adobe XFA exploits for all! First Part: The Info-leak, Nico Waisman, 2013

Thanks!

Xiaoning.Li@intel.com
Haifei_Li@McAfee.com

Thanks to Bing Sun, Chong Xu, and Dan Sommer of McAfee Labs


intel
Security