Exploit Mitigation using Multi-Variant Execution

Stijn Volckaert
University of California, Irvine

OWASP Belgium Chapter Meeting – 18 October 2016
Adobe Patches 10 Critical Vulnerabilities in Flash Player, Shockwave Player, and ColdFusion

Posted on April 9th, 2013 by Derek Erwin

Adobe Issues Emergency Updates For Zero-Day Flaw in Flash Player

Memory corruption flaw is being exploited in the wild to distribute ransomware samples like Locky and Cerber.

Firefox 45 browser update patches 22 (Another) Update To Adobe Flash Addresses Latest 0-Day Vulnerability

April 15, 2011


Adobe Fixes 18 Vulnerabilities in Flash Player

By Eduard Kovacs on November 12, 2014

Zero-Day Vulnerability Bypasses Apple's Security Features

Updat Adobe Flash update closes several critical holes

36 March 2016, 8:15 am EDT By Hoia Ungureanu Tech Times

Update Flash now! Adobe releases patch, fixing critical security holes

BY GRAHAM CLULEY POSTED 22 SEP 2015 - 09:24AM

Critical Adobe Flash bug under active attack currently has no patch

Exploit works against the most recent version; Adobe plans update later this week.

by Dan Goodin — Jun 14, 2016 12:56pm PDT

Adobe Releases Security Update for 19 ‘Critical’ Vulnerabilities in Flash Player

By David Bisson DEC 20, 2015 | LATEST SECURITY NEWS
Possible Solutions

- Type-Safe Languages (e.g. Rust)
- Mitigations:
  - Integrity-Based (e.g. CFI)
  - Randomization-Based (e.g. ASLR)
- Multi-Variant Execution Environments (MVEEs)
Memory Corruption Attacks

```c
void foo() {
    char buf[256];
    gets(buf);
    printf("%s", buf);
}

int main(int argc, char** argv) {
    foo();
    return 0;
}
```
Multi-Variant Execution Environments (MVEEs)

In a nutshell:

• Run multiple program variants in parallel
• Variant system calls executed in lock-step
• Suspend them at every system call
• Compare system call numbers/arguments
• Master/Slave replication for I/O
Performance Considerations

Programs can execute at **native speed** (assuming you have enough idle CPU cores and memory bandwidth)

The total system load does **not** increase by a factor of $n$ (with $n$ the number of variants)

**BUT** there are some problems!
Slow System Call Interception

Split-Monitor Design:

- Handle security-sensitive system calls in Cross-Process Monitor (CP-MON)
- Handle non-sensitive system calls in In-Process Monitor (IP-MON)
Code Reuse

Protects against:
- return-to-libc (RILC)
- return-oriented programming (ROP)
- jump-oriented programming (JOP)
- just-in-time code reuse (JIT-ROP) [*]
- counterfeit object-oriented programming (COOP) [*]
- ...

[*] Requires eXecute-only memory support
Code Injection

xor rdx, rdx
mov qword rbx, '//bin/sh'
shr rbx, 0x8
push rbx
mov rdi, rsp
push rax
push rdi
mov rsi, rsp
mov al, 0x3b
syscall

execve("/bin/sh", ["/bin/sh"], NULL)
Code Injection
Non-Control Data Attacks

1: `void` ProcessConnection(connection* c) {
2:     cred_t user;
3:     char message[1024];
4:     int i = 0;
5:     auth_user(&user, c);
6:     while (!end_of_message(c))
7:         message[i++] = get_next_char(c);
8:     seteuid(user.user_id);
9:     ExecuteRequest(message);
10: }

Return address

i
Non-Control Data Attacks

```c
1: void ProcessConnection(connection* c) {
2:     cred_t user;
3:     char message[1024];
4:     int i = encrypt(0, 0);
5:     auth_user(&user, c);
6:     while (!end_of_message(c)) {
7:         register int tmp = decrypt(i, 0);
8:             int i = encrypt(tmp + 1, 0);
9:         message[i++] = get_next_char(c);
10:     seteuid(user.user_id);
11:     ExecuteRequest(message);
12: }  // decrypt(user.user_id, 0))
13: }
14: }
15: 
16: }
```
Non-Control Data Attacks

**BONUS:** Information Leakage Protection
PROTECT ALL

THE PROGRAMS
Multithreading

Master Thread 1

Master Thread 2

Slave Thread 1

Slave Thread 2

monitor 1

monitor 2

kernel
Multithreading

- Master Thread 1
- Slave Thread 1
- Master Thread 2
- Slave Thread 2

- accept
- nanosleep
- monitor 1
- monitor 2

- STOP

Kernel
Shared Memory

Program

Monitor

Kernel

Display Server
Other problems

• Data races
• vdso/vsyscall pages
• RDTSC/RDTSCP
• Address Dependence
https://github.com/stijn-volckaert/ReMon