Size Does Matter
Why Using Gadget-Chain Length to Prevent Code-reuse Attacks is Hard

ENES GÖKTAŞ (VRIJE UNIVERSITEIT AMSTERDAM)
ELIAS ATHANASOPOULOS (FORTH-ICS)
MICHALIS POLYCHRONAKIS (COLUMBIA UNIVERSITY)
HERBERT BOS (VRIJE UNIVERSITEIT AMSTERDAM)
GEORGIOS PORTOKALIDIS (STEVENS INSTITUTE OF TECHNOLOGY)
Advanced Exploitation of Mozilla Firefox Use-After-Free Vulnerability (Pwn2Own 2014)

Published on 2014-05-20 17:19:47 UTC by Arno, Sandro, and Vincent Urgoat

Hi everyone,

Pwn2Own 2014 was very exciting as we witnessed mobile devices now getting more secure than ever before. However, it is reported that additional efforts are required to ensure that these devices remain secure.

In this year’s edition of Pwn2Own, we have observed attacks on Internet Explorer 11, Google Chrome, and Firefox. We have reported all the vulnerabilities and our full exploitation guide will protect users.

One of the vulnerabilities we have exploited is a use-after-free vulnerability in Microsoft’s Internet Explorer 11 (MSFA2014-30 / CVE-2014-1512). This flaw was identified by Mike Zhan and has been confirmed by Microsoft. The vulnerability allows attackers to execute arbitrary code on the victim’s machine. The vulnerability has been repaired in Internet Explorer 11.10.880.0, and Microsoft has assigned CVE-2014-1776 to it.

Malware Protection Center

A journey to CVE-2014-0497 exploit

msft-mmpc | 17 Feb 2014 2:50 PM | 1

Last week we published a blog post about a CVE-2013-5330 exploit. We’ve also recently seen a new, similar attack targeting a patched Adobe Flash Player vulnerability (CVE-2014-0497).

The vulnerability related to this malware was addressed with a patch released by Adobe on February 4, 2014. Flash Player versions 12.0.0.43 and earlier are vulnerable. We analyzed how these attacks work and found the following:

New Zero-Day Exploit targeting Internet Explorer Versions 9 through 11 Identified in Targeted Attacks

April 26, 2014 | By Xiaobo Chen, Dan Caselden and Mike Scott | Advanced Malware, Exploits, Targeted Attack, Uncategorized

Summary

FireEye Research Labs identified a new Internet Explorer (IE) zero-day exploit used in targeted attacks. The vulnerability affects IE6 through IE11, but the attack is targeting IE9 through IE11. This zero-day bypasses both ASLR and DEP. Microsoft has assigned CVE-2014-1776 to the vulnerability and released security advisory to track this issue.
Control-Flow Integrity

Promising defense mechanism against ROP

We showed that recent CFI proposals do not stop ROP attacks (see “Out of Control: Overcoming CFI”, Oakland ’14)
Inspecting Branching History

Alternative promising defenses against ROP

State-of-the-art proposals:
- kBouncer (Pappas et al., Usenix Security 2013)
- ROPecker (Cheng et al., NDSS 2014)

Fundamentally based on:
- a Control-Flow Integrity policy, and
- a Heuristic-based policy

Assume to be broken

Focus of this talk

What are the security implications?
Heuristic-based policy

Relies on two threshold parameters

Chain length $C_T \geq 4$

$L_T$ or less number of instructions are considered as gadgets
$= \text{max gadget length}$

$C_T$ or more gadgets in sequence is an attack
$= \text{gadget chain threshold}$

$L_T \leq 5$

BH check
Picking the “best” Thresholds

An attacker could **mix short gadgets with long gadgets** longer than $L_T$.

Preferably: $L_T$ as large as possible & $C_T$ as small as possible

But setting $L_T$ too large and $C_T$ too small can lead to False Positives

Thresholds have to be chosen carefully!
## Chosen thresholds of defenses

<table>
<thead>
<tr>
<th></th>
<th><strong>kBouncer</strong></th>
<th><strong>ROPeecker</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time-of-Check</strong></td>
<td>Entry of Sensitive API</td>
<td>Entry of Sensitive API + Exit of executable code window</td>
</tr>
<tr>
<td><strong>Gadget Length</strong></td>
<td>20 instructions</td>
<td>6 instructions</td>
</tr>
<tr>
<td><strong>Inspect BH instances</strong></td>
<td>Detected max &quot;benign&quot; gadget chain length: 5</td>
<td>Detected max &quot;benign&quot; gadget chain length: 10</td>
</tr>
<tr>
<td><strong>Gadget Chain Length</strong></td>
<td>8 gadgets</td>
<td>11 gadgets</td>
</tr>
</tbody>
</table>
Difficulties with Heuristic Breakers

Heuristic Breakers may easily:

- Use high number of different registers
- Leave used registers dirty at exit
- Require memory preparations
- Have a whacky code sequence

```asm
mov eax, ebx
mov ecx, edx
add esi, edi
mov esi, [0x1234]
cmp esi, 10
jg X
mov ecx, 0x2321
div ecx
mov [eax], edi
mov ecx, 0x5678
and edi, ecx
xor eax, edi
retn
```
Proof-of-Concept Exploit

Real IE8 vulnerability

Bypasses ASLR, DEP, kBouncer

**Idea:** intersperse a Heuristic breaker in ROP chain to prevent reaching $C_T$

**Goal:** execute our injected code
Branching History

- Non-Gadget
- Gadget

Number of instructions:
- 10
- 15
- 9
- 34
- 12
- 11
- 5
- 7
- 2
- 2
- 2
- 2
- 2
- 2
- 3
- 3

BH check

VirtualProtect

SHELLCODE
Branching History

ATTACK DETECTED

# = Non-Gadget
# = Gadget
# = Nr. of instr.

VirtualProtect
BH check
SHELLCODE
**Branching History**

- 10
- 15
- 9
- 34
- 12
- 11
- 5
- 7
- 33
- 2
- 2
- 2
- 3

**NO ATTACK**

- # = Non-Gadget
- # = Gadget
- # = Nr. of instr.

- 7 → 2 → 2 → 2 → 3 → VirtualProtect
- BH check
- SHELLCODE
Implications of Stricter Thresholds

On mixing short gadgets with Heuristic Breakers

Assume:
\[ L_T = 20 \]
\[ C_T = 3 \]

Difficulties for an attacker:
- Not enough space to prepare Heuristic Breaker
- Not enough space to restore state after Heuristic Breaker
- Not enough space to prepare a function call
Per Application Thresholds

![Graph showing the relationship between number of instances and gadget-chain length for different applications such as Acrobat, IE (Google), IE (YouTube), Excel, Word, PowerPoint, and WMPlayer.](image)
Conclusion

Choosing the right thresholds for ROP detection is difficult

The “long gadgets are not usable” assumption is broken

We need better tools to evaluate our defenses