Control-flow Enforcement Technology

Yu-cheng Yu <yu-cheng.yu@intel.com>
Agenda

Introduction

CET - Shadow Stack

CET - Indirect Branch Tracking

Summary & Status
Introduction

Control-flow Enforcement Technology (CET)

- An upcoming Intel® processor family feature that counters return/jump-oriented programming (ROP) attacks

- Two components:
  - Shadow Stack (SHSTK)
  - Indirect Branch Tracking (IBT)
Control-flow Definition

The code execution path, branched by RET, JMP, or CALL.

<table>
<thead>
<tr>
<th>Op Code</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>RET</td>
<td>On program stack</td>
</tr>
<tr>
<td>JMP *%rax</td>
<td>In memory (%rax as a pointer)</td>
</tr>
<tr>
<td>CALL *%rax</td>
<td>In memory (%rax as a pointer)</td>
</tr>
</tbody>
</table>
Return/Jump Oriented Programming (ROP) Attacks

Stack Buffer Overflow + Gadgets → System Call

No code injection is needed!
The Stack Buffer Overflow

```c
void copy_string(char *input) {
    char buf[4];
    memcpy(buf, input, strlen(input));
}
```
A Code Gadget Example

mov 0xc3084189, %eax

mov %eax, 0x8(%ecx)

ret
The ROP Attack

Program stack

- return address
- saved RBP
- buf[3]
- buf[2]
- buf[1]
- buf[0]

JMP xxxx
JMP xxxx
RET
system call
Agenda

Introduction

CET - Shadow Stack

CET - Indirect Branch Tracking

Summary & Status
Shadow Stack Concept

CALL

Program Stack

Return Address

Shadow Stack

Return Address
Shadow Stack -- Return Address Matching

RET

Program Stack

Return Address

Shadow Stack

Return Address
Shadow Stack Exception

- Bad Address
- Return Address

Program Stack

Shadow Stack
Shadow Stack Management

• Copy-on-write vs. Copy-on-access
  • Sharing & Page Table Entry (PTE)

• Unwinding
  • Handling setjmp/longjmp and ucontext in GLIBC

• Context switch
  • Kernel-mode shadow stack context switch (with TOKENS)
  • User-mode shadow stack context switch (with XSAVES)
Copy-on-Write

Normal Page

PTE: R/W D
Page: Mem Page

Shared Page

PTE: R/W D
Page: Mem Page

Page duplicated

PTE: R/W D
Page: Mem Page
Shadow Stack PTE

- Readable
- A new error code indicates CALL/RET access
Shadow Stack Sharing

- Normal SHSTK PTE: R/O, D → Mem Page
- Shared SHSTK PTEs: R/O, D → Mem Page
- Shared page duplicated: R/O, D → Mem Page
Management of (Read-Only + Dirty) PTEs

Kernel Mode:

User Mode:
Shadow Stack Unwinding

main() → func_1() → func_2() → longjmp

Shadow stack

Return addr #1
Return addr #2

INCSSP <Steps>
Shadow Stack Context Switch

- Ring 3: IA32_PL3_SSP
- Ring 0: IA32_PL0_SSP, IA32_INTERRUPT_SSP_TABLE
Shadow Stack Switching - User Mode

Task A

CET MSRs

XSAVES/XRESTORS

CET MSRs

Task B
Shadow Stack in Exception Handling

Supervisor Token

shadow stack available

Exception

IRET

Supervisor Token

shadow stack in-use

Busy
Shadow Stack in Exception Handling

- Supervisor Token
  - shadow stack available
  - SETSSBSY
  - CLRSSBSY

- Supervisor Token
  - shadow stack in-use

Busy
Shadow Stack Tokens

- Always 64-bit in size
- Stored in the shadow stack itself
Shadow Stack Switching - Kernel Mode

Incoming SHSTK

\[ y \]
\[ y - 8 \]

Restore token

\[ \text{RSTORSSP } y - 8 \]

Outgoing SHSTK

\[ x \]
\[ x - 8 \]

Restore token

\[ \text{SAVEPREVSSP} \]
Agenda

Introduction

CET - Shadow Stack

CET - Indirect Branch Tracking

Summary & Status
Indirect Branch Tracking (IBT)

```c
main() {
    int (*f)();
    f = test;
    f();
}

test() { return }
```
Indirect Branch Tracking

• No additional software development effort

• The compiler does the work:
  • ENDBR64/ENDBR32 Op code
  • No-track prefix
IBT Compatibility

• ENDBR is NOP on legacy processors
• Optional legacy bitmap for working with legacy libraries
IBT Legacy Code

<table>
<thead>
<tr>
<th>Linear Address Width</th>
<th>Bitmap Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 bit</td>
<td>128 KB</td>
</tr>
<tr>
<td>48 bit</td>
<td>8 GB</td>
</tr>
<tr>
<td>57 bit</td>
<td>4 TB</td>
</tr>
</tbody>
</table>

- Virtual memory is only committed when used
- Small percentage of whole address range
Agenda

Introduction
CET - Shadow Stack
CET - Indirect Branch Tracking
Summary & Status
CET Instructions Review

- RDSSP – Read shadow stack pointer
- INCSSP – Shadow stack unwinding

- RSTORSSP, SAVEPREVSSP – Shadow stack context switching
- SETSSBSY, CLRSSBSY – Mark shadow stack in-use

- ENDBR, No-Track – Indirect branch tracking
Other Protection Measures

- Address space layout randomization
- Glibc pointer encryption
- Safe stack
- Compatibility & performance tradeoffs

**CET offers minimal effort and low performance overhead!**
Summary

• CET closes a security hole
  • Specification is available by searching “Intel CET” or at:
  • Minimal effort for application development
    • Compile and use
    • Minor changes in ucontext and jump buffer
• GCC/GLIBC support available soon
  • Some patches published
  • Binutils already available
• Linux* patches available Q1/2018
Disclaimers

Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

Intel, the Intel logo and others are trademarks of Intel Corporation in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.

© 2017 Intel Corporation.