RSAConference2018
San Francisco | April 16 – 20 | Moscone Center

MATTERS NOW

SESSION ID: HTA-F01

ENHANCE VIRTUALIZATION STACK WITH INTEL CET AND MPX

Xiaoning Li

Chief Security Architect
Alibaba Cloud

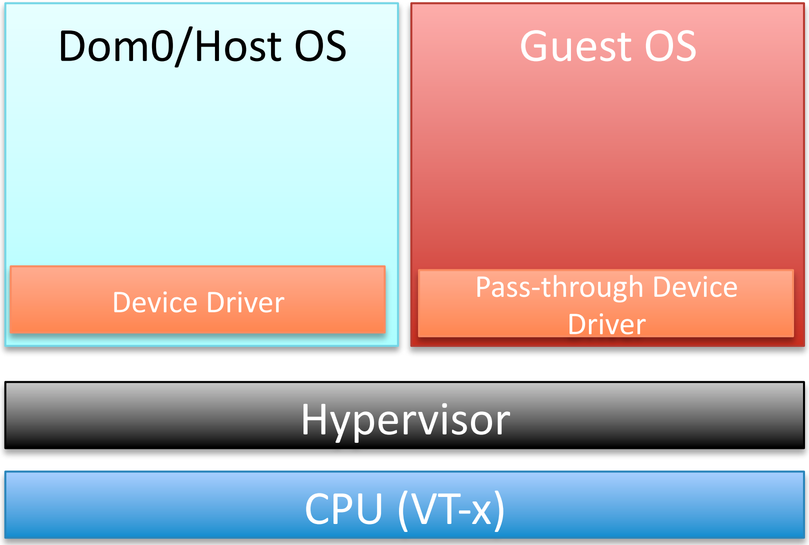
Ravi Sahita

Principal Engineer
Intel Corporation

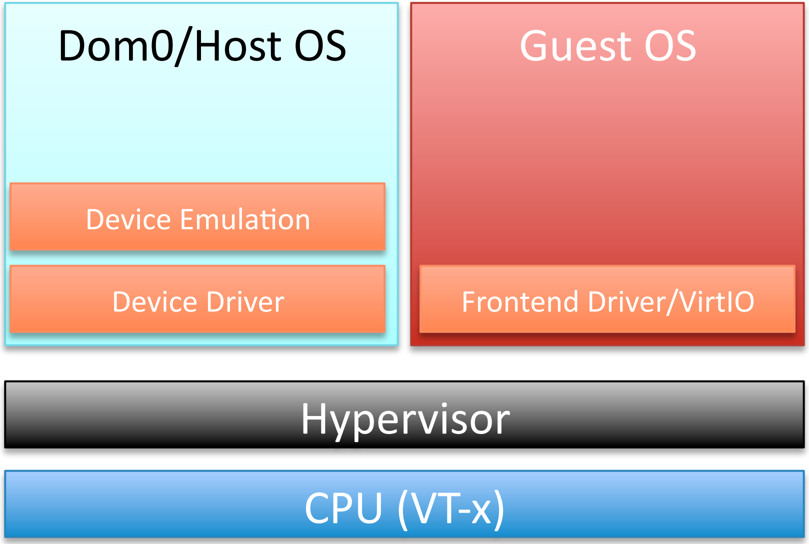
Agenda

- Full Virtualization Stack
- QEMU Vulnerabilities
- Intel CET
- Intel MPX and PKU
- VM Escape Case Study
- Mitigation with CET/MPX
- Other Mitigations

Full Virtualization Stack



HVM Guest OS with Device Pass-through



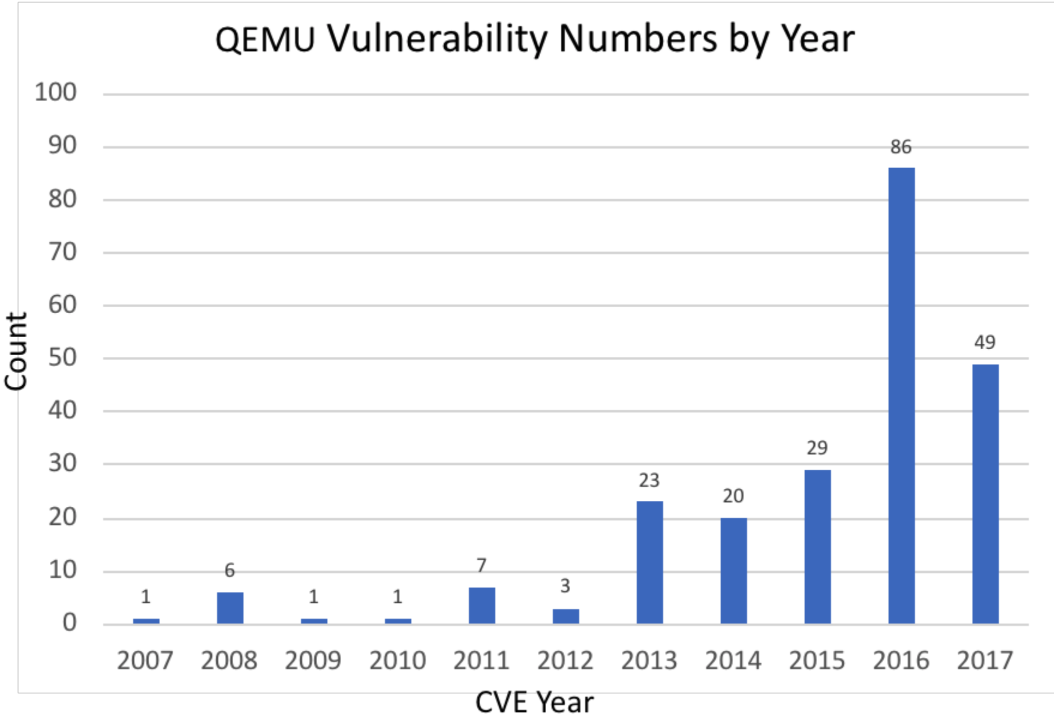
HVM Guest OS Without Device Pass-through

Why Device Emulation?

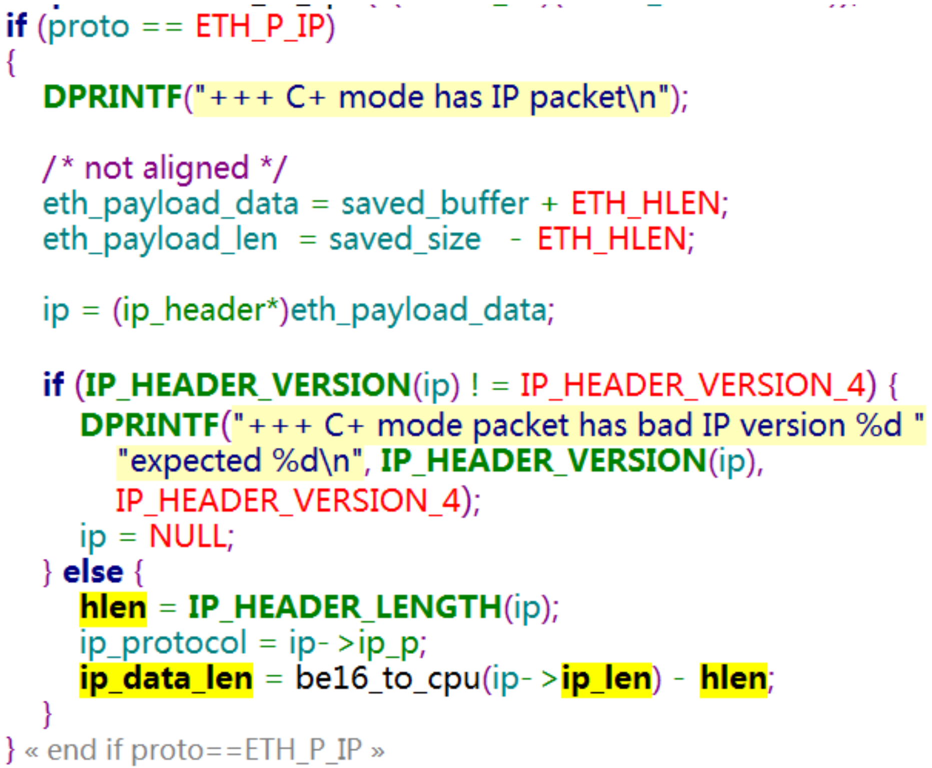
- Supports more guest devices with virtual devices
  - If physical devices number are not enough - limited GPU resource
  - If physical devices don’t support device virtualization – Not every device support SRIOV
  - If physical devices don’t exist – some outdated devices
- Popular usage in cloud environment to support many VMs
- QEMU can provide device emulation for KVM/XEN, but it brings new attack surface on virtualization stack

QEMU VULNERABILITIES

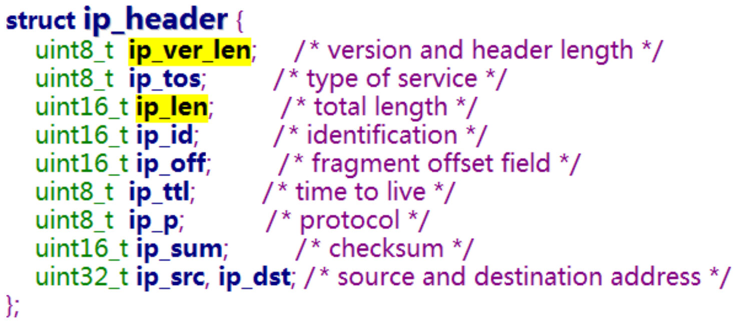
QEMU Vulnerabilities by 2018 Jan



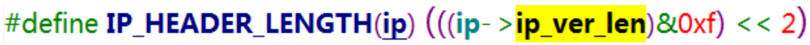
CVE-2015-5165 – Information Leak



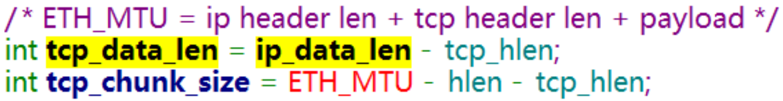
```
if (proto == ETH_P_IP)
{
    DPRINTF("+++ C+ mode has IP packet\n");

    /* not aligned */
    eth_payload_data = saved_buffer + ETH_HLEN;
    eth_payload_len  = saved_size   - ETH_HLEN;

    ip = (ip_header*)eth_payload_data;

    if (IP_HEADER_VERSION(ip) != IP_HEADER_VERSION_4) {
        DPRINTF("+++ C+ mode packet has bad IP version %d "
            "expected %d\n", IP_HEADER_VERSION(ip),
            IP_HEADER_VERSION_4);
        ip = NULL;
    } else {
        hlen = IP_HEADER_LENGTH(ip);
        ip_protocol = ip->ip_p;
        ip_data_len = be16_to_cpu(ip->ip_len) - hlen;
    }
} « end if proto==ETH_P_IP »
```



```
struct ip_header {
    uint8_t  ip_ver_len;    /* version and header length */
    uint8_t  ip_tos;        /* type of service */
    uint16_t ip_len;        /* total length */
    uint16_t ip_id;         /* identification */
    uint16_t ip_off;        /* fragment offset field */
    uint8_t  ip_ttl;        /* time to live */
    uint8_t  ip_p;          /* protocol */
    uint16_t ip_sum;        /* checksum */
    uint32_t ip_src, ip_dst; /* source and destination address */
};
```

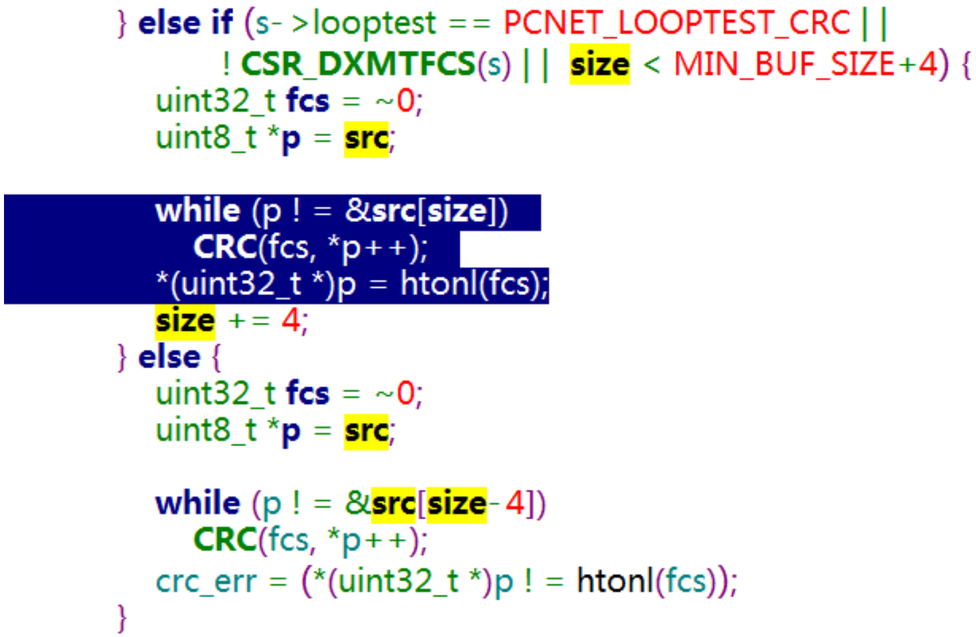


```
#define IP_HEADER_LENGTH(ip) (((ip->ip_ver_len)&0xf) << 2)
```



```
/* ETH_MTU = ip header len + tcp header len + payload */
int tcp_data_len = ip_data_len - tcp_hlen;
int tcp_chunk_size = ETH_MTU - hlen - tcp_hlen;
```

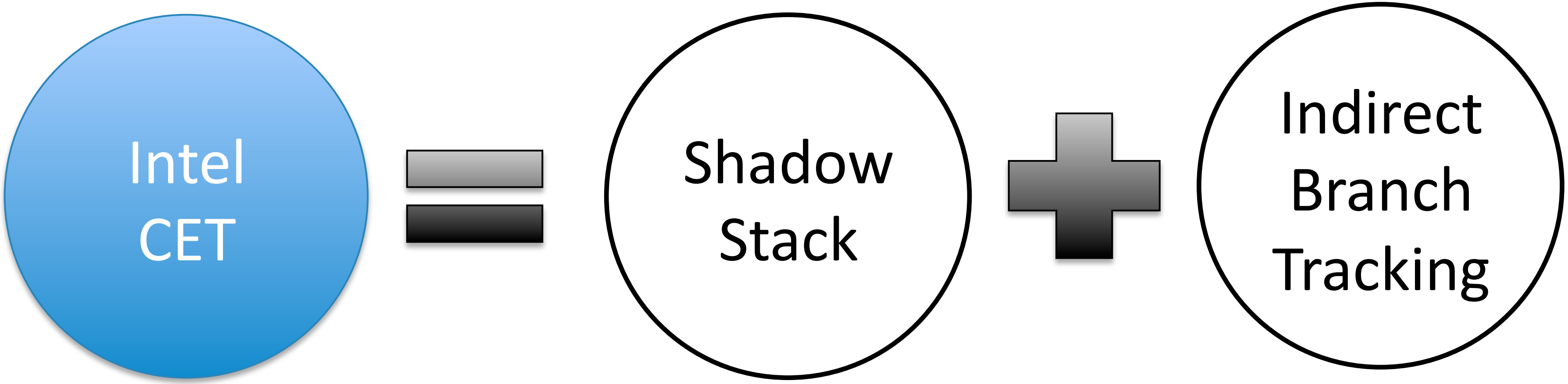
CVE-2015-7504 – Heap Overflow



```
} else if (s->looptest == PCNET_LOOPTEST_CRC ||
           !CSR_DXMTFCS(s) || size < MIN_BUF_SIZE+4) {
    uint32_t fcs = ~0;
    uint8_t *p = src;

    while (p != &src[size])
        CRC(fcs, *p++);
    *(uint32_t *)p = htonl(fcs);
    size += 4;
} else {
    uint32_t fcs = ~0;
    uint8_t *p = src;

    while (p != &src[size-4])
        CRC(fcs, *p++);
    crc_err = (*(uint32_t *)p != htonl(fcs));
}
```

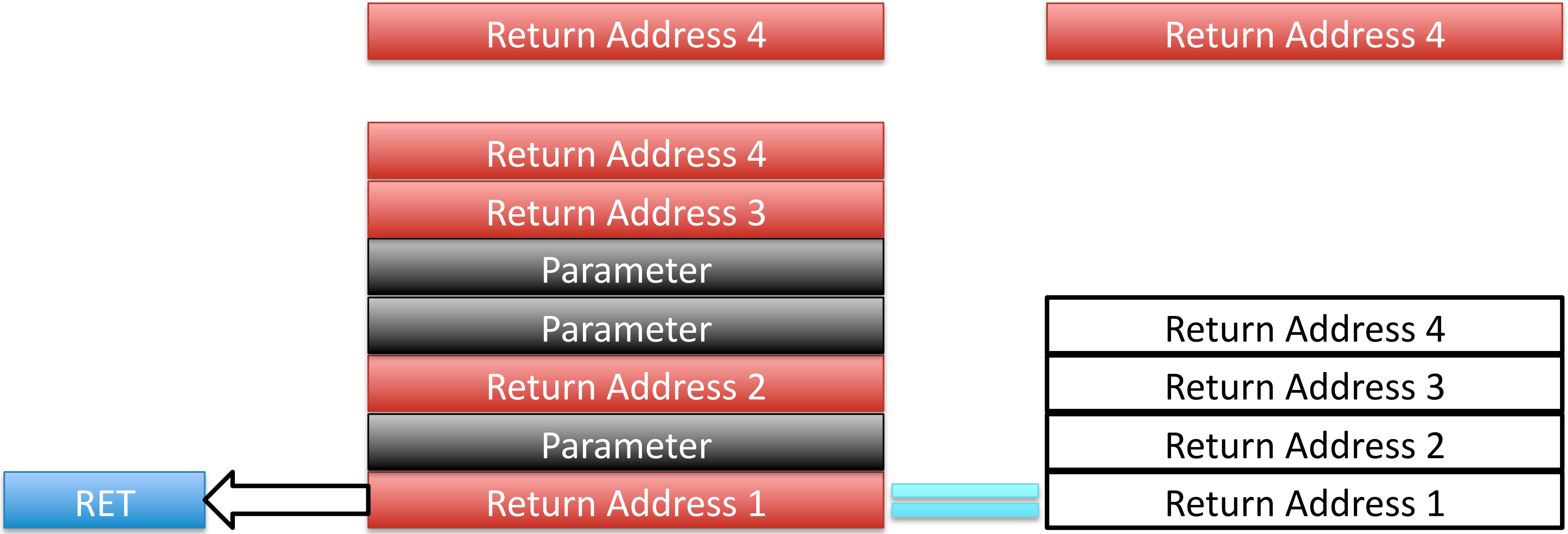
INTEL CET

Intel® Control-flow Enforcement Technology

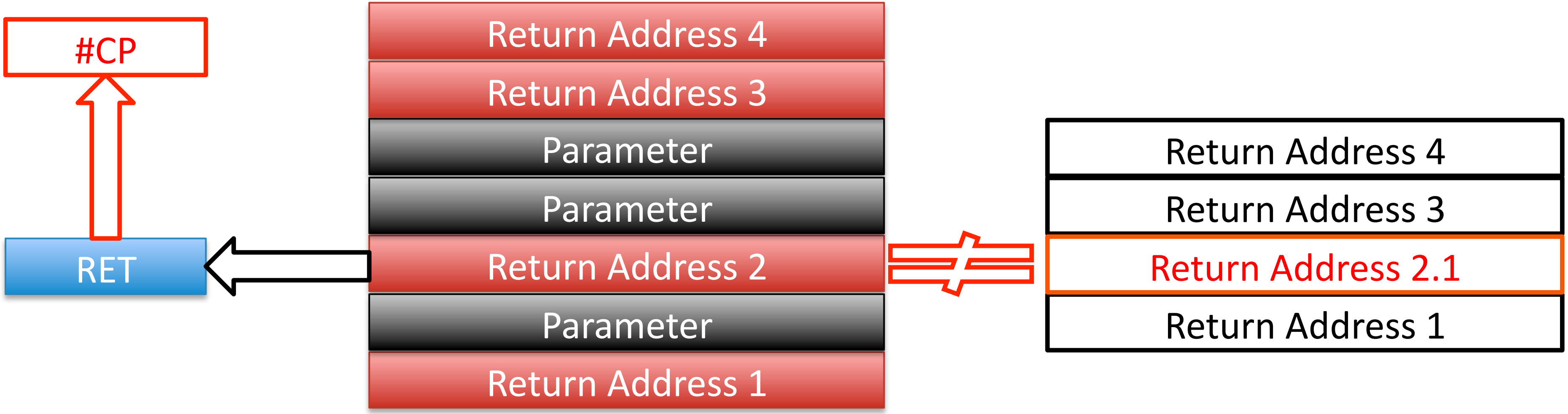


Intel CET = Shadow Stack + Indirect Branch Tracking

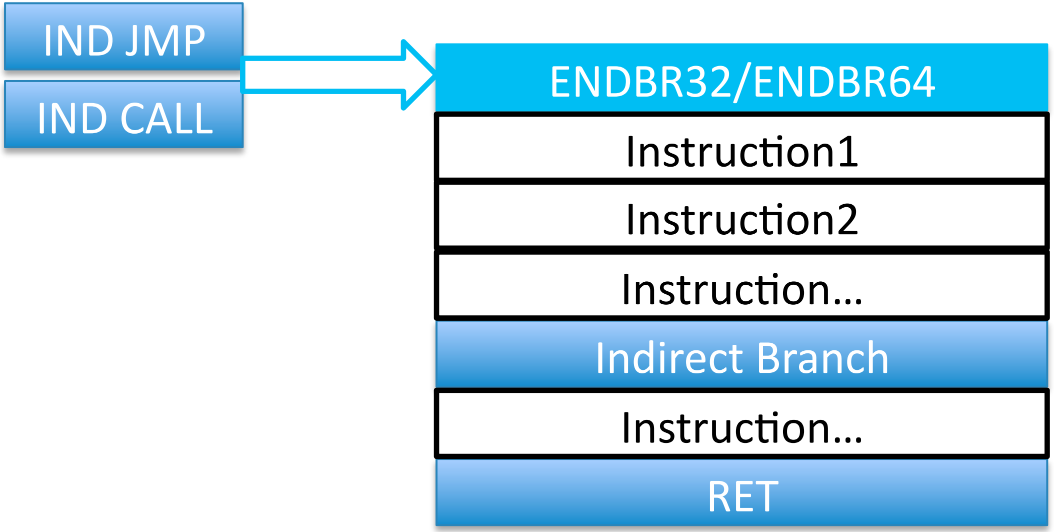
Shadow Stack



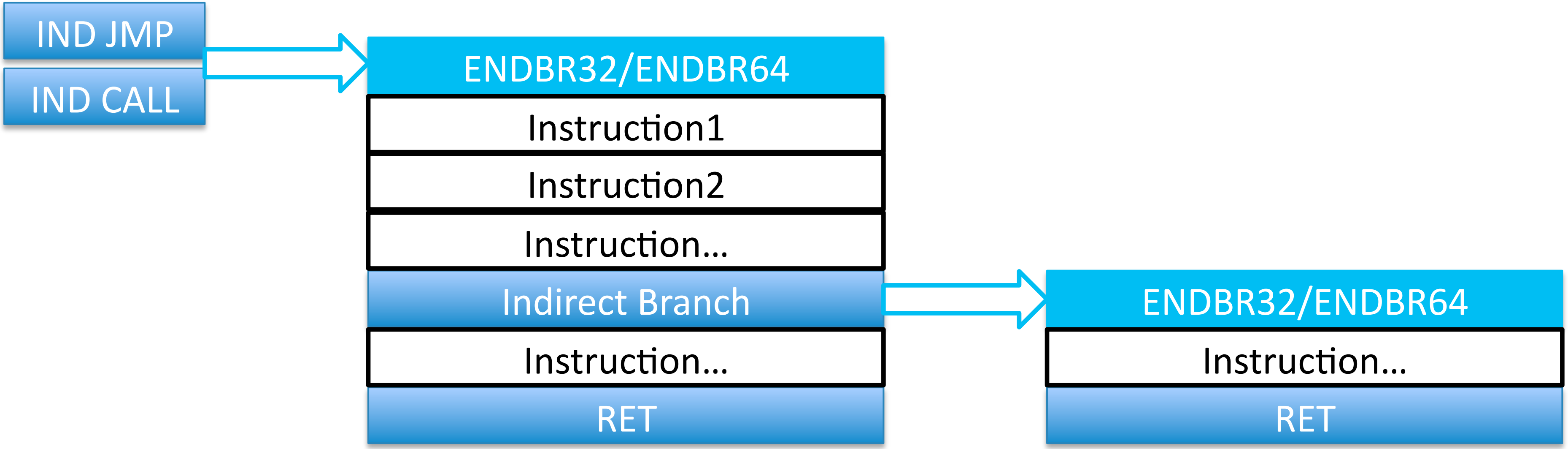
Shadow Stack Control Protection Exception



Indirect Branch Tracking



Indirect Branch Tracking



IND JMP

IND CALL

ENDBR32/ENDBR64

Instruction1

Instruction2

Instruction…

Indirect Branch

Instruction…

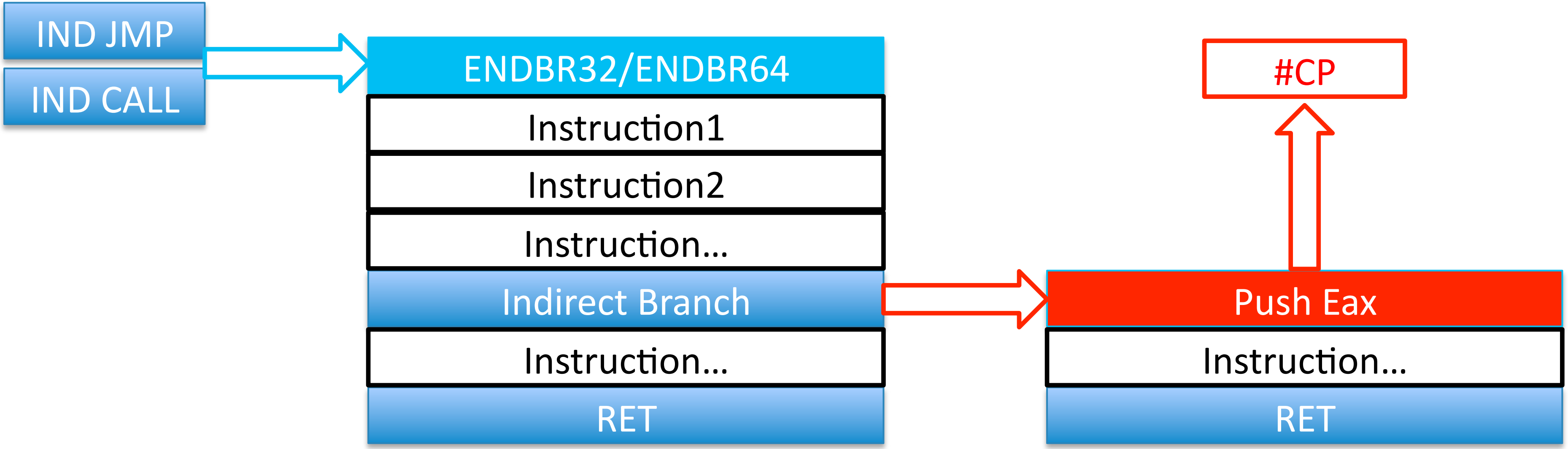
RET

ENDBR32/ENDBR64

Instruction…

RET

Indirect Branch Tracking #CP



IND JMP

IND CALL

ENDBR32/ENDBR64

Instruction1

Instruction2

Instruction…

Indirect Branch

Instruction…

RET

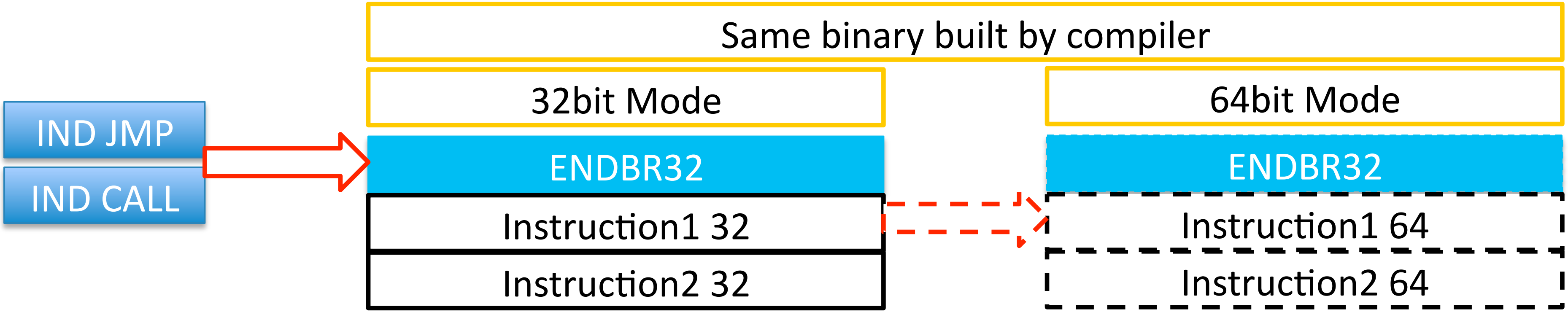
#CP

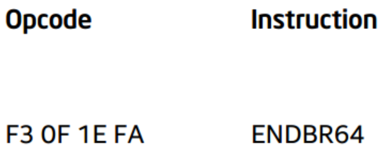
Push Eax

Instruction…

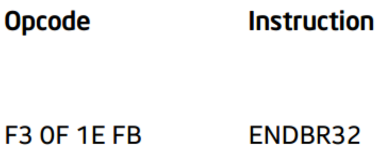
RET

Cross Mode Indirect Branch Tracking





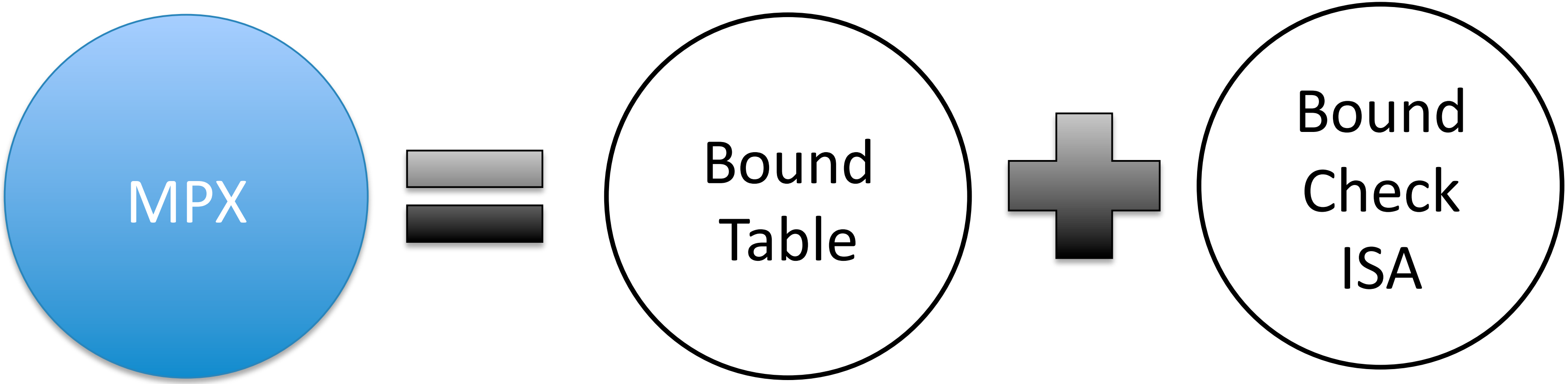
Opcode	Instruction
F3 0F 1E FA	ENDBR64



Opcode	Instruction
F3 0F 1E FB	ENDBR32

INTEL MPX

Intel® Memory Protection Extensions



MPX = Bound Table + Bound Check ISA

Bound Table

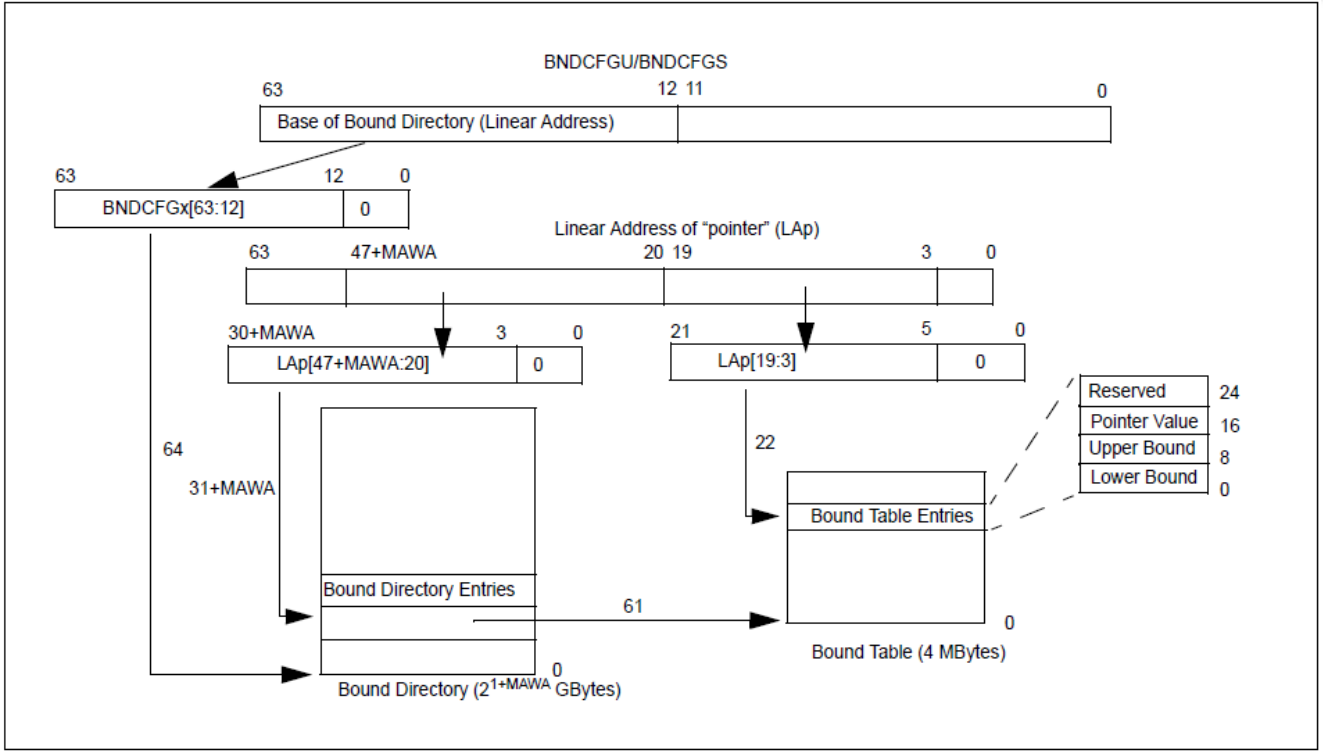
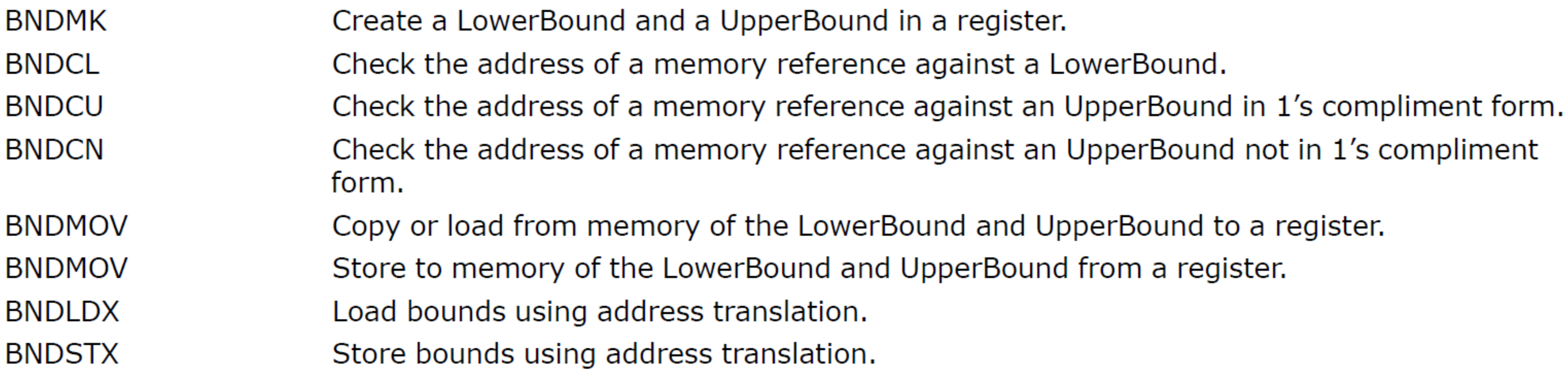


Figure 17-4. Bound Paging Structure and Address Translation in 64-Bit Mode

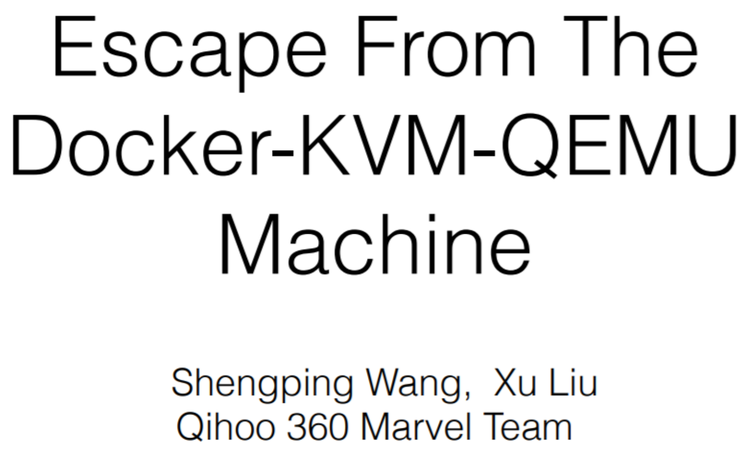
Bound Instructions



BNDMK	Create a LowerBound and a UpperBound in a register.
BNDCL	Check the address of a memory reference against a LowerBound.
BNDCU	Check the address of a memory reference against an UpperBound in 1's compliment form.
BNDCN	Check the address of a memory reference against an UpperBound not in 1's compliment form.
BNDMOV	Copy or load from memory of the LowerBound and UpperBound to a register.
BNDMOV	Store to memory of the LowerBound and UpperBound from a register.
BNDLDX	Load bounds using address translation.
BNDSTX	Store bounds using address translation.

QEMU VM ESCAPE CASE STUDY

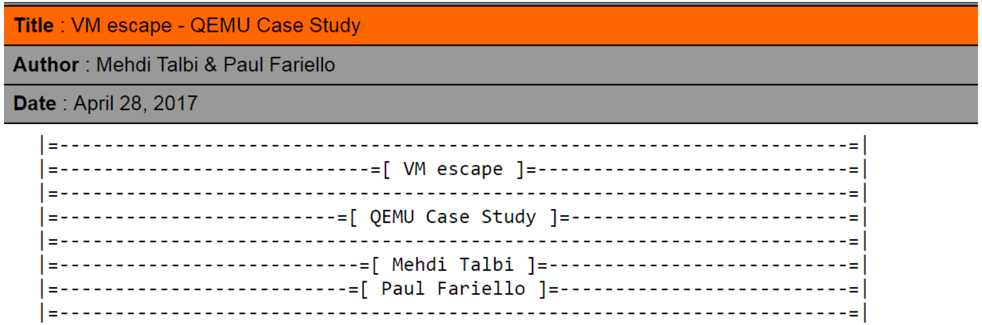
CVE-2015-5165 and CVE-2015-7504 Exploit



Escape From The Docker-KVM-QEMU Machine

Shengping Wang, Xu Liu
Qihoo 360 Marvel Team

#HITB2016AMS D1T1 - Escape From The Docker KVM QEMU Machine - Shengping Wang and Xu Liu



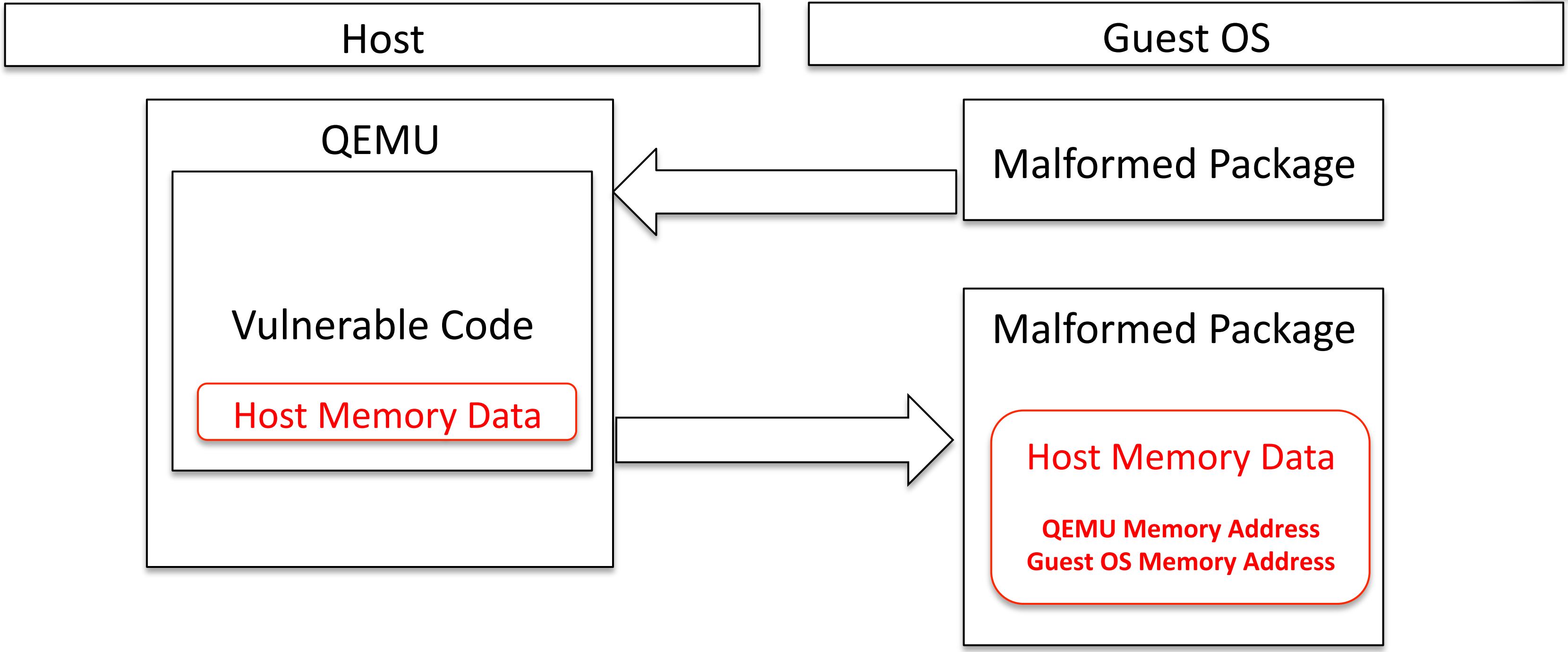
Title : VM escape - QEMU Case Study

Author : Mehdi Talbi & Paul Fariello

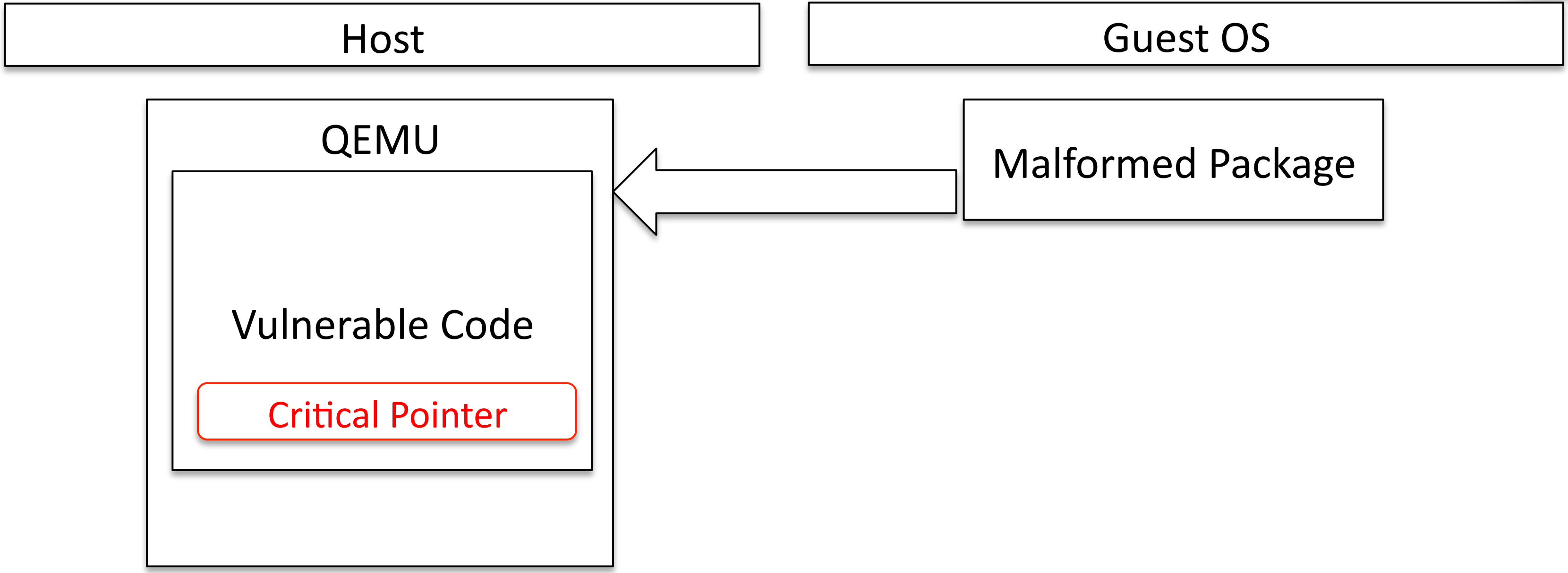
Date : April 28, 2017

```
|=-----------------------------------------------------------------------=|
|=-----------------------------=[ VM escape ]=----------------------------=|
|=-----------------------------------------------------------------------=|
|=-------------------------=[ QEMU Case Study ]=-------------------------=|
|=-----------------------------------------------------------------------=|
|=---------------------------=[ Mehdi Talbi ]=---------------------------=|
|=--------------------------=[ Paul Fariello ]=--------------------------=|
|=-----------------------------------------------------------------------=|
```

CVE-2015-5165 Memory Disclosure



CVE-2015-7504 Code Execution



Host

Guest OS

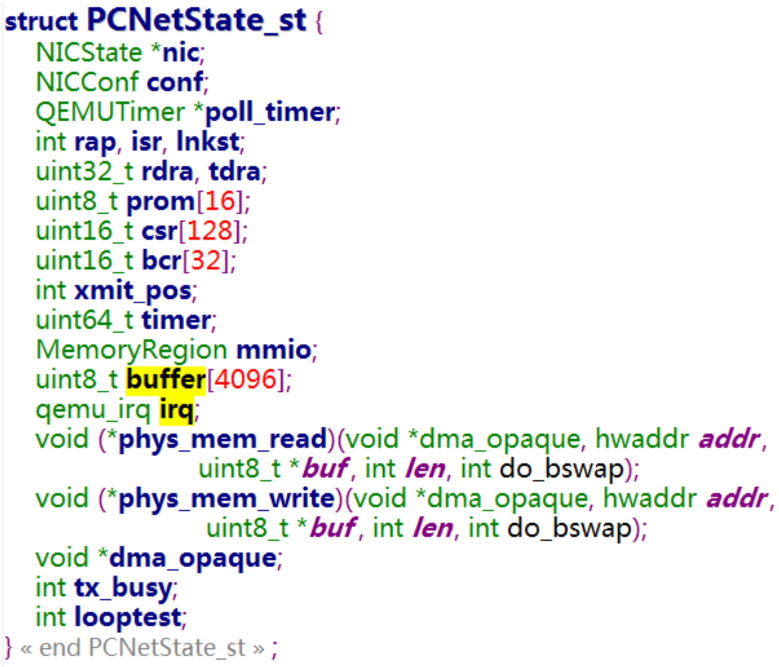
QEMU

Vulnerable Code

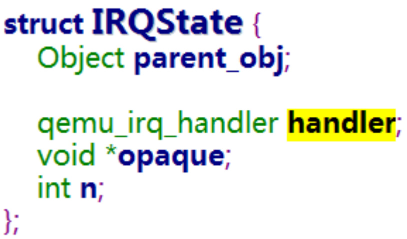
Critical Pointer

Malformed Package

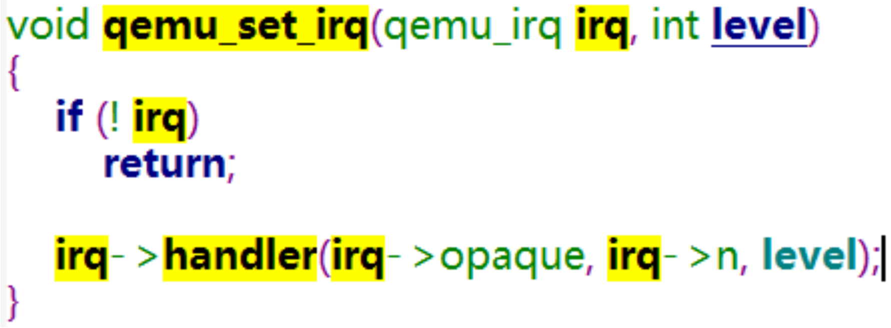
CVE-2015-7504 Code Execution



```
struct PCNetState_st {
    NICState *nic;
    NICConf conf;
    QEMUTimer *poll_timer;
    int rap, isr, lnkst;
    uint32_t rdra, tdra;
    uint8_t prom[16];
    uint16_t csr[128];
    uint16_t bcr[32];
    int xmit_pos;
    uint64_t timer;
    MemoryRegion mmio;
    uint8_t buffer[4096];
    qemu_irq irq;
    void (*phys_mem_read)(void *dma_opaque, hwaddr addr,
                         uint8_t *buf, int len, int do_bswap);
    void (*phys_mem_write)(void *dma_opaque, hwaddr addr,
                          uint8_t *buf, int len, int do_bswap);
    void *dma_opaque;
    int tx_busy;
    int looptest;
} « end PCNetState_st » ;
```

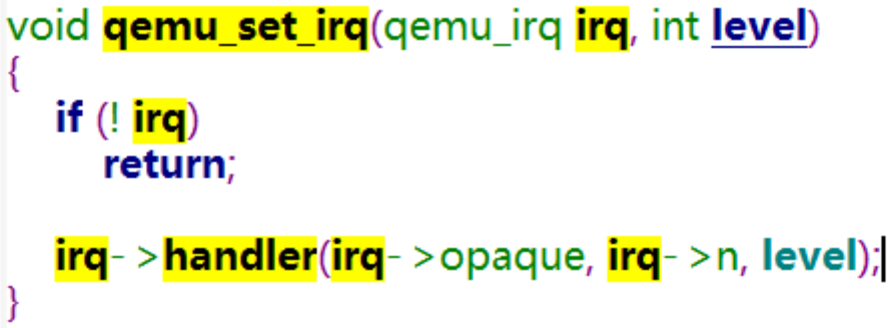


```
struct IRQState {
    Object parent_obj;

    qemu_irq_handler handler;
    void *opaque;
    int n;
};
```



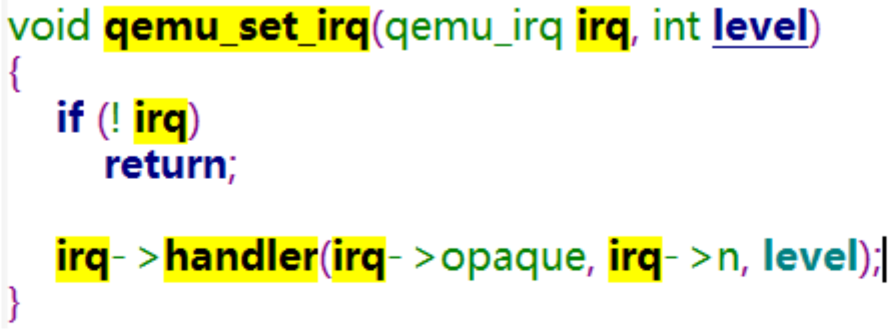
```
void qemu_set_irq(qemu_irq irq, int level)
{
    if (! irq)
        return;

    irq- >handler(irq- >opaque, irq- >n, level);
}
```

CVE-2015-7504 Code Execution



```
void qemu_set_irq(qemu_irq irq, int level)
{
    if (! irq)
        return;

    irq- >handler(irq- >opaque, irq- >n, level);
}
```

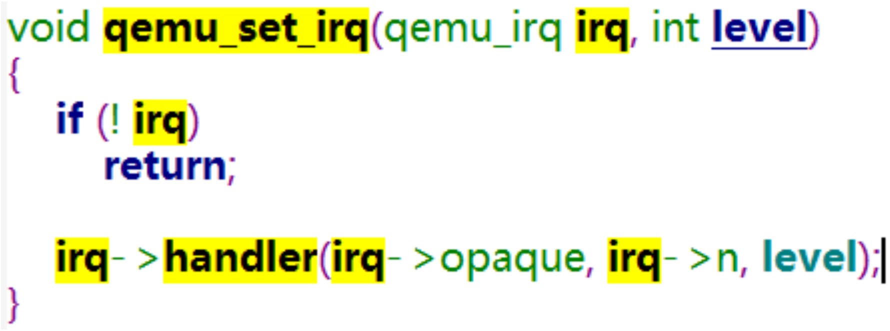
CVE-2015-7504 Exploit in HITB



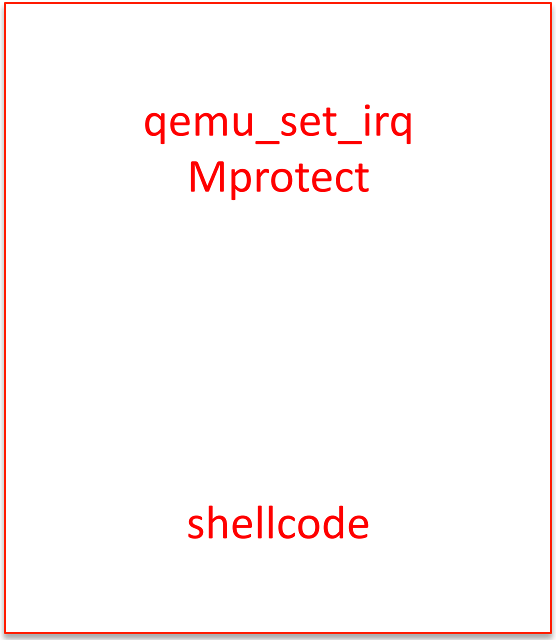
```
void qemu_set_irq(qemu_irq irq, int level)
{
    if (! irq)
        return;

    irq- >handler(irq- >opaque, irq- >n, level);
}
```

```
xchg rax,rsp;
ret
```

CVE-2015-7504 Exploit in Phrack



```
void qemu_set_irq(qemu_irq irq, int level)
{
    if (! irq)
        return;

    irq- >handler(irq- >opaque, irq- >n, level);
}
```

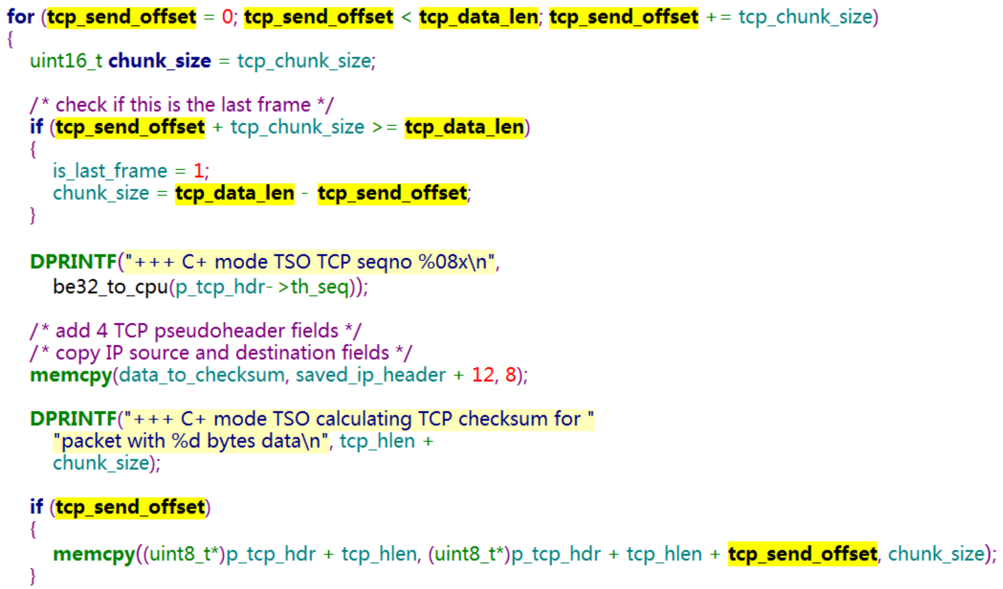


qemu_set_irq
Mprotect

shellcode

MITIGATION WITH CET/MPX

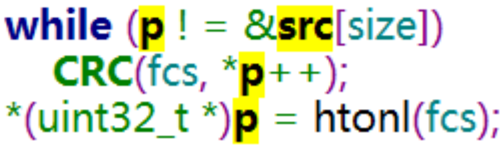
CVE-2015-5165 – Out of Bound Access Defense with MPX



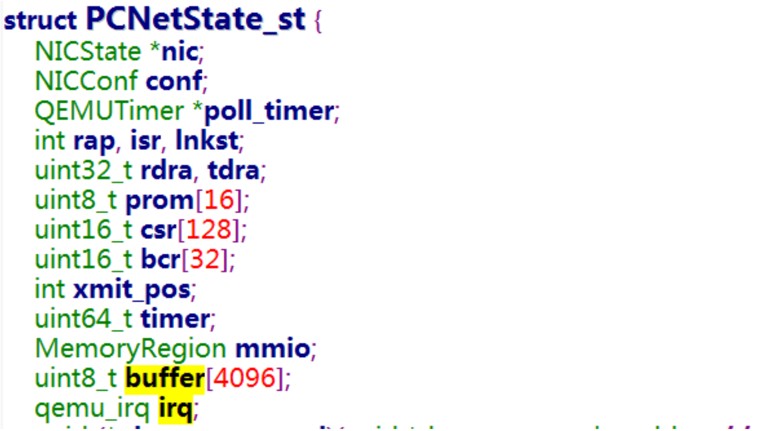
```
for (tcp_send_offset = 0; tcp_send_offset < tcp_data_len; tcp_send_offset += tcp_chunk_size)
{
    uint16_t chunk_size = tcp_chunk_size;

    /* check if this is the last frame */
    if (tcp_send_offset + tcp_chunk_size >= tcp_data_len)
    {
        is_last_frame = 1;
        chunk_size = tcp_data_len - tcp_send_offset;
    }

    DPRINTF("+++ C+ mode TSO TCP seqno %08x\n",
        be32_to_cpu(p_tcp_hdr->th_seq));

    /* add 4 TCP pseudoheader fields */
    /* copy IP source and destination fields */
    memcpy(data_to_checksum, saved_ip_header + 12, 8);

    DPRINTF("+++ C+ mode TSO calculating TCP checksum for "
        "packet with %d bytes data\n", tcp_hlen +
        chunk_size);

    if (tcp_send_offset)
    {
        memcpy((uint8_t*)p_tcp_hdr + tcp_hlen, (uint8_t*)p_tcp_hdr + tcp_hlen + tcp_send_offset, chunk_size);
    }
```

Enable MPX on packet memory access

CVE-2015-7504 – Out of Bound Read Defense with MPX



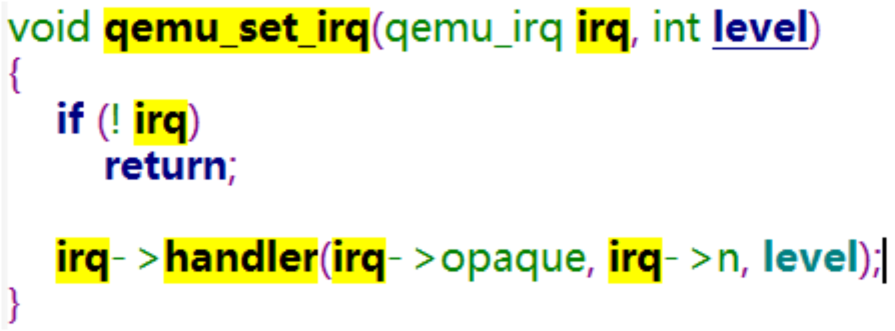
```
while (p ! = &src[size])
    CRC(fcs, *p++);
*(uint32_t *)p = htonl(fcs);
```



```
struct PCNetState_st {
    NICState *nic;
    NICConf conf;
    QEMUTimer *poll_timer;
    int rap, isr, lnkst;
    uint32_t rdra, tdra;
    uint8_t prom[16];
    uint16_t csr[128];
    uint16_t bcr[32];
    int xmit_pos;
    uint64_t timer;
    MemoryRegion mmio;
    uint8_t buffer[4096];
    qemu_irq irq;
```

Enable MPX on buffer[4096] memory access

CVE-2015-7504 Exploit Defense with CET

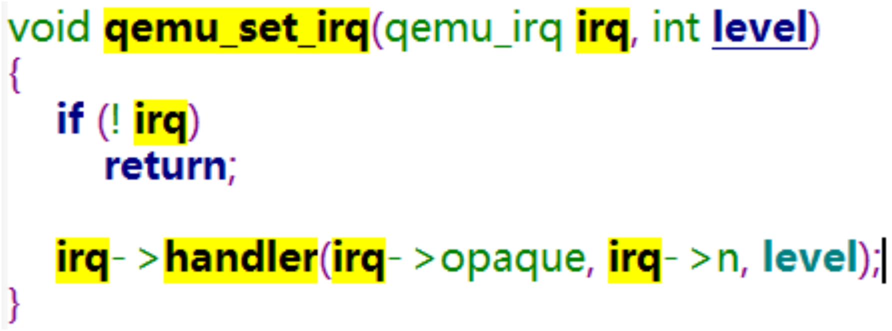


```
void qemu_set_irq(qemu_irq irq, int level)
{
    if (! irq)
        return;

    irq- >handler(irq- >opaque, irq- >n, level);
}
```

xchg rax,rsp;
ret

Shadow Stack can stop “xchg rax,rsp;ret”

CVE-2015-7504 Exploit Defense with CET



```
void qemu_set_irq(qemu_irq irq, int level)
{
    if (! irq)
        return;

    irq- >handler(irq- >opaque, irq- >n, level);
}
```

qemu_set_irq

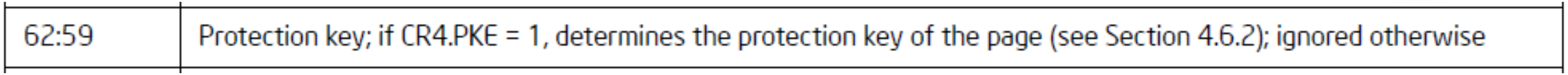
Mprotect

shellcode

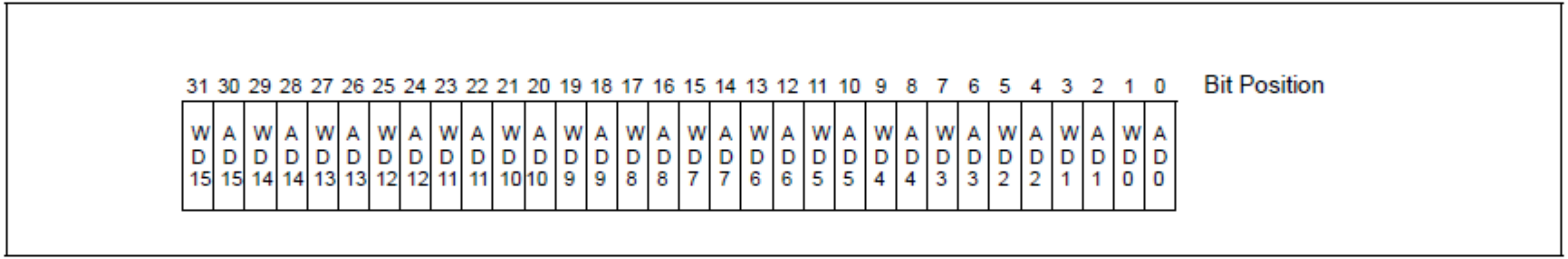
Indirect Branch Tracking can stop irq->handler calling qemu_set_irq without valid tag

OTHER MITIGATIONS

Other Mitigation - PKU

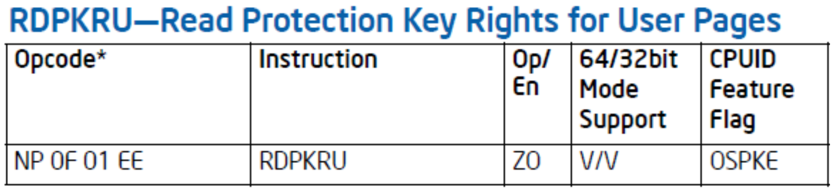


62:59	Protection key; if CR4.PKE = 1, determines the protection key of the page (see Section 4.6.2); ignored otherwise



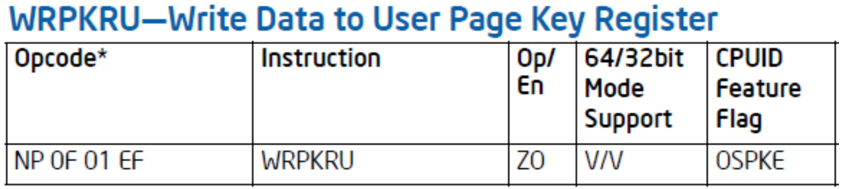
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Bit Position
WD15	AD15	WD14	AD14	WD13	AD13	WD12	AD12	WD11	AD11	WD10	AD10	WD9	AD9	WD8	AD8	WD7	AD7	WD6	AD6	WD5	AD5	WD4	AD4	WD3	AD3	WD2	AD2	WD1	AD1	WD0	AD0	

Figure 2-9. Protection Key Rights Register for User Pages (PKRU)



RDPKRU—Read Protection Key Rights for User Pages

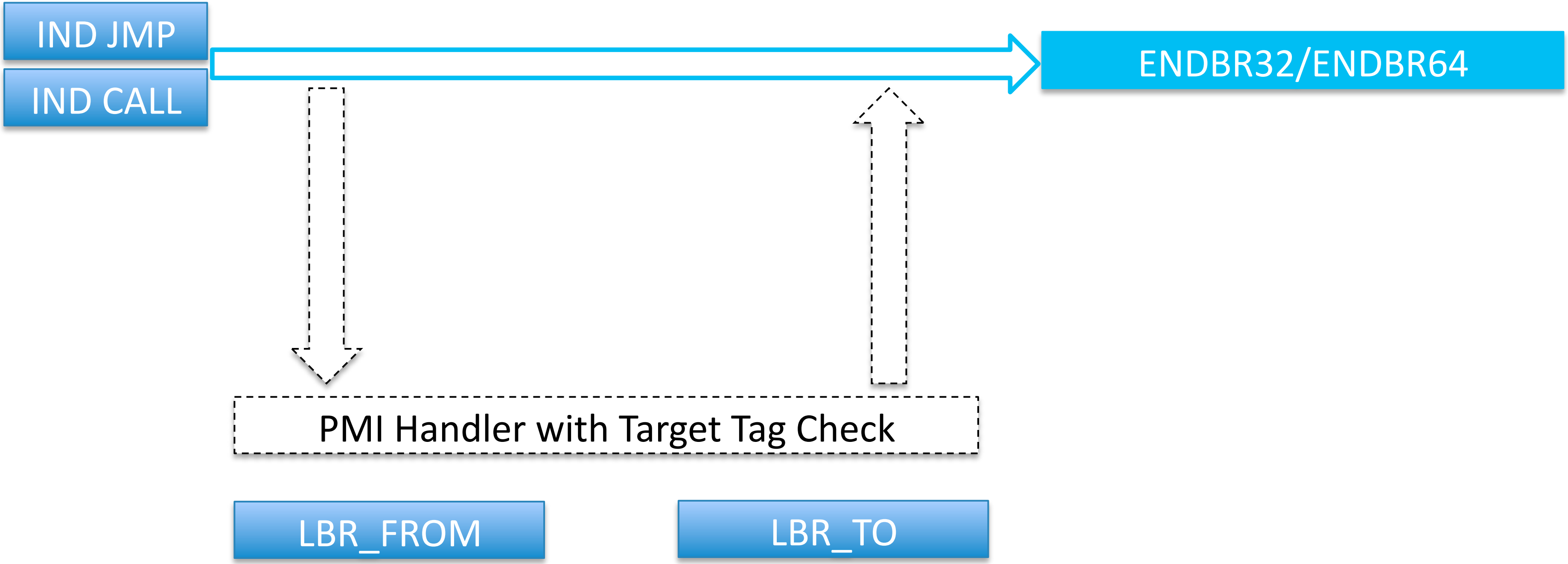
Opcode*	Instruction	Op/ En	64/32bit Mode Support	CPUID Feature Flag
NP 0F 01 EE	RDPKRU	ZO	V/V	OSPKE



WRPKRU—Write Data to User Page Key Register

Opcode*	Instruction	Op/ En	64/32bit Mode Support	CPUID Feature Flag
NP 0F 01 EF	WRPKRU	ZO	V/V	OSPKE

Other Mitigation - PMI



IND JMP

IND CALL

ENDBR32/ENDBR64

PMI Handler with Target Tag Check

LBR_FROM

LBR_TO

Summary

- VM escape is practical and impacts cloud foundation security
- Intel CET/MPX can enhance mitigation on ROP/JOP/COP and buffer overflow, specifically on cloud virtualization stack

Call For Actions

- Apply new CPU mechanisms such as CET/MPX/PMU/PKU on exploit defense

THANK YOU! QUESTIONS?

XIAONING.LI@ALIBABA-INC.COM