RUHR UNIVERSITÄT BOCHUM RUB

emproof

Reverse Engineering x86 Processor Microcode

CanSecWest 2018

Marc 16, 2018, Vancouver, Canada

Philipp Koppe, Benjamin Kollenda, Marc Fyrbiak, Christian Kison,
Robert Gawlik, Christof Paar, Thorsten Holz

Horst Görtz Institute for IT-Security
Ruhr-Universität Bochum
<firstname.lastname>@rub.de

emproof
www.emproof.de

Outline

RUHR UNIVERSITÄT BOCHUM RUB

- What is microcode?
- Architectural crash course
- Analysis
- Demo

Outline

RUHR UNIVERSITÄT BOCHUM RUB

- What is microcode?
- Architectural crash course
- Analysis
- Demo

What is microcode?

RUHR UNIVERSITÄT BOCHUM RUB

- Firmware for the processor

What is microcode?

- Firmware for the processor
  - Instruction decoding

What is microcode?

- Firmware for the processor
  - Instruction decoding
  - Fix CPU bugs

What is microcode?

- Firmware for the processor
  - Instruction decoding
  - Fix CPU bugs
  - Exception handling

What is microcode?

RUHR UNIVERSITÄT BOCHUM RUB

- Firmware for the processor
  - Instruction decoding
  - Fix CPU bugs
  - Exception handling
  - Power Management

What is microcode?

RUHR UNIVERSITÄT BOCHUM RUB

- Firmware for the processor
  - Instruction decoding
  - Fix CPU bugs
  - Exception handling
  - Power Management
  - Complex features (Intel SGX)

x86 ISA is complex

RUHR UNIVERSITÄT BOCHUM

RUB

C3	**ret**

x86 ISA is complex

RUHR UNIVERSITÄT BOCHUM

RUB

```
C3                          ret

48 b8 88 77 66 55           movabs   rax,0x1122334455667788
44 33 22 11
```

x86 ISA is complex

RUHR UNIVERSITÄT BOCHUM

RUB

```
C3                          ret

48 b8 88 77 66 55           movabs   rax,0x1122334455667788
44 33 22 11

64 ff 03                    inc DWORD PTR fs:[ebx]
```

x86 ISA is complex

RUHR UNIVERSITÄT BOCHUM

RUB

```
C3                          ret

48 b8 88 77 66 55           movabs   rax,0x1122334455667788
44 33 22 11

64 ff 03                    inc DWORD PTR fs:[ebx]

64 67 66 f0 ff 03           lock inc WORD PTR fs:[bx]
```

x86 ISA is complex

RUHR UNIVERSITÄT BOCHUM

RUB

```
C3                          ret

48 b8 88 77 66 55           movabs   rax,0x1122334455667788
44 33 22 11

64 ff 03                    inc DWORD PTR fs:[ebx]

64 67 66 f0 ff 03           lock inc WORD PTR fs:[bx]

2e c4 e2 71 96 84           vfmaddsub132ps xmm0, xmm1,
be 34 23 12 01              xmmword ptr cs:
                            [esi + edi * 4 + 0x11223344]
```

Micro Ops

RUHR UNIVERSITÄT BOCHUM

RUB

```
pop [ebx]
```

Micro Ops

RUHR UNIVERSITÄT BOCHUM

RUB

```
pop [ebx]
```

```
load temp, [esp]
store [ebx], temp
add esp, 4
```

Microcode updates are used to fix CPU bugs

RUHR UNIVERSITÄT BOCHUM

RUB

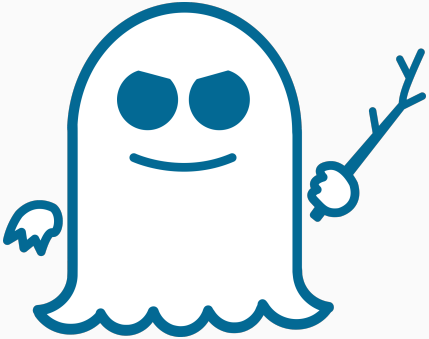
Microcode updates are used to fix CPU bugs

RUHR UNIVERSITÄT BOCHUM RUB

Microcode updates are used to fix CPU bugs

Microcode updates are used to fix CPU bugs

Microcode updates are used to fix CPU bugs

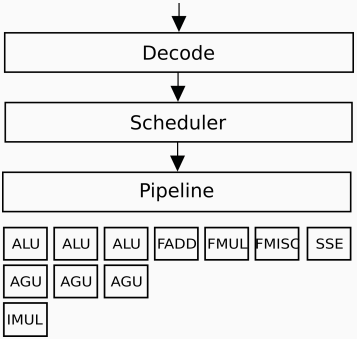


Outline

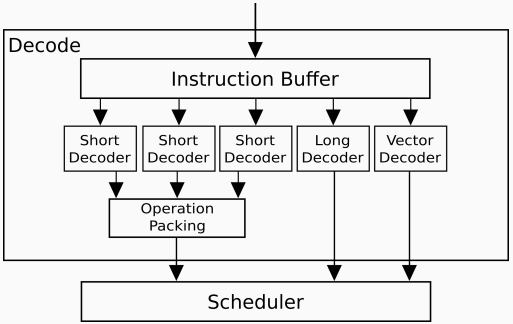
RUHR UNIVERSITÄT BOCHUM RUB

- What is microcode?
- Architectural crash course
- Analysis
- Demo

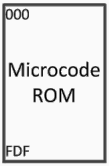
x86 Instruction Decoding



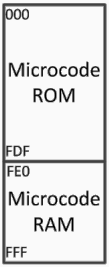
x86 Instruction Decoding



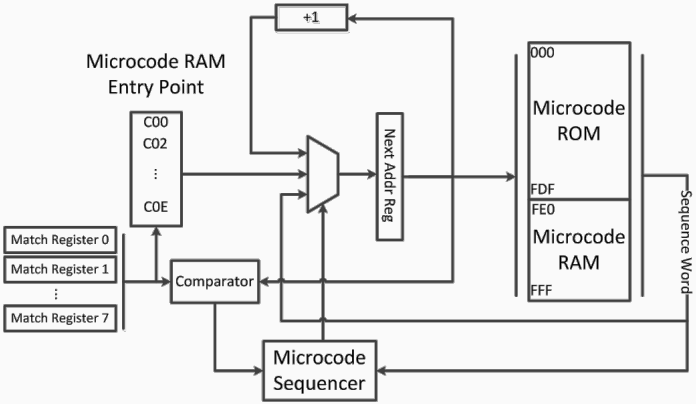
Microcode Engine (Vector Decoder)



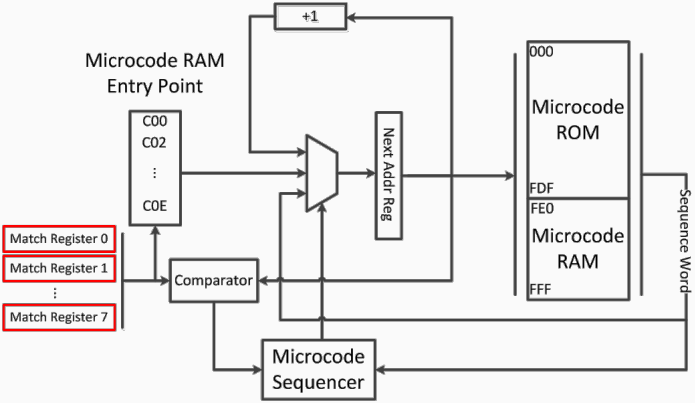
Microcode Engine (Vector Decoder)



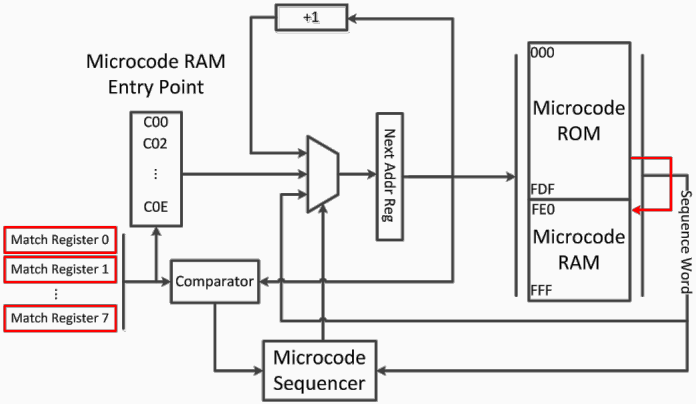
Microcode Engine (Vector Decoder)



Microcode Engine (Vector Decoder)



Microcode Engine (Vector Decoder)



Microcode Update Mechanism

RUHR UNIVERSITÄT BOCHUM RUB

- Kernel mode
- Load microcode update into RAM
- Write virtual address to MSR `0xC0010020`
- Microcode patches not persistent

Microcode Update File Format

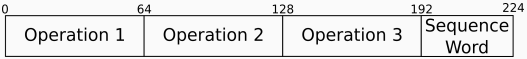
RUHR UNIVERSITÄT BOCHUM RUB

B↓ Bit→	0 31	32 63
0	date	patch ID
8	patch block | len | init	checksum
16	northbridge ID	southbridge ID
24	CPUID	magic value
32	match register 0	match register 1
40	match register 2	match register 3
48	match register 4	match register 5
54	match register 6	match register 7
64	triad 0, microinstruction 0	
72	triad 0, microinstruction 1	
80	triad 0, microinstruction 2	
88	triad 0, sequence word	triad 1 ...

Microcode Update File Format

RUHR UNIVERSITÄT BOCHUM RUB

B↓ Bit→	0 31	32 63
0	date	patch ID
8	patch block | len | init	checksum
16	northbridge ID	southbridge ID
24	CPUID	magic value
32	match register 0	match register 1
40	match register 2	match register 3
48	match register 4	match register 5
54	match register 6	match register 7
64	triad 0, microinstruction 0	
72	triad 0, microinstruction 1	
80	triad 0, microinstruction 2	
88	triad 0, sequence word	triad 1 ...



Outline

- What is microcode?
- Architectural crash course
- Analysis
- Demo

Starting Point

Starting Point

- CPUs updatable

Starting Point

RUHR UNIVERSITÄT BOCHUM RUB

- CPUs updatable
- Update drivers in Linux kernel

Starting Point

- CPUs updatable
- Update drivers in Linux kernel
- Microcode updates

Starting Point

- CPUs updatable
- Update drivers in Linux kernel
- Microcode updates
- Update file format

Starting Point

- CPUs updatable
- Update drivers in Linux kernel
- Microcode updates
- Update file format
- Hints that there is no strong crypto

Update hexdump


```
0000000  02062004 00000039 00208000 3e331cfd 00000000 00000000 00000048 aaaaaa00
0000020  00000644 00000140 ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
0000040  00fffbff fd9035c3 a9fe7bf8 3f00ff0f c7ffddfe 3c03bc1e 00d57a80 ffffe14
0000060  eb1fe1fe ff3cf64c 807f879f f0266027 18fe1d74 6abd2000 e7fffbcf 0ff0f3f0
0000080  9fffee3f 1bc3e440 07fc01fe ff0dfe00 ebe00035 3fe00ff0 f86ff007 ff803fc0
00000a0  e1bfc01f fe00ff03 86ff007f 78001a30 f007f81f 37f803fc c01fe07f dfe00ff0
00000c0  007f81ff 7f803fc3 000f7ffc 03fc0ff8 fc01fe1b 0ff03fe0 f007f86f 3fc0ff80
00000e0  c01fe1bf 07bffe00 fe07fc01 00ff0dfe f81ff007 03fc37f8 e07fc01f 0ff0dfe0
0000100  dfff0003 03fe00ff 7f86ff00 0ff803fc fe1bfc01 3fe00ff0 f86ff007 ff8001ef
0000120  ff007f81 c37f803f fc01fe07 0dfe00ff f007f81f 37f803fc c000f7ff 803fc0ff
0000140  bfc01fe1 00ff03fe ff007f86 03fc0ff8 fc01fe1b 007bffe0 1fe07fc0 e00ff0df
0000160  7f81ff00 803fc37f fe07fc01 00ff0dfe 3dfff000 f03fe00f 07f86ff0 c0ff803f
0000180  1fe1bfc0 03fe00ff 7f86ff00 fff8001e 1ff007f8 fc37f803 7fc01fe0 f0dfe00f
00001a0  ff007f81 c37f803f fc000f7f f803fc0f 1bfc01fe e00ff03f 6ff007f8 803fc0ff
00001c0  bfc01fe1 0007bffe 01fe07fc fe00ff0d 07f81ff0 f803fc37 1fe07fc0 e00ff0df
00001e0  03dfff00 ff03fe00 007f86ff fc0ff803 01fe1bfc f03fe00f 07f86ff0 efff8001
0000200  81ff007f 3fc37f80 07fc01fe ff0dfe00 1ff007f8 fc37f803 ffc000f7 ff803fc0
0000220  e1bfc01f fe00ff03 86ff007f f803fc0f 1bfc01fe e0007bff c01fe07f dfe00ff0
0000240  007f81ff 7f803fc3 01fe07fc fe00ff0d 003dfff0 0ff03fe0 f007f86f 3fc0ff80
0000260  c01fe1bf ff03fe00 007f86ff 1efff800 f81ff007 03fc37f8 e07fc01f 0ff0dfe0
0000280  81ff007f 3fc37f80 7ffc000f 0ff803fc fe1bfc01 3fe00ff0 f86ff007 ff803fc0
00002a0  e1bfc01f fe0007bf fc01fe07 0dfe00ff f007f81f 37f803fc c01fe07f dfe00ff0
00002c0  0003dfff 00ff03fe ff007f86 03fc0ff8 fc01fe1b 0ff03fe0 f007f86f 01efff80
00002e0  7f81ff00 803fc37f fe07fc01 00ff0dfe f81ff007 03fc37f8 f7ffc000 c0ff803f
0000300  1fe1bfc0 03fe00ff 7f86ff00 0ff803fc fe1bfc01 ffe0007b 7fc01fe0 f0dfe00f
0000320  ff007f81 c37f803f fc01fe07 0dfe00ff f0003dff e00ff03f 6ff007f8 803fc0ff
0000340  bfc01fe1 00ff03fe ff007f86 001efff8 07f81ff0 f803fc37 1fe07fc0 e00ff0df
0000360  7f81ff00 803fc37f 0f7ffc00 fc0ff803 01fe1bfc f03fe00f 07f86ff0 c0ff803f
0000380  1fe1bfc0 bffe0007 07fc01fe ff0dfe00 1ff007f8 fc37f803 7fc01fe0 f0dfe00f
00003a0  ff0003df fe00ff03 86ff007f f803fc0f 1bfc01fe e00ff03f 6ff007f8 8001efff
```

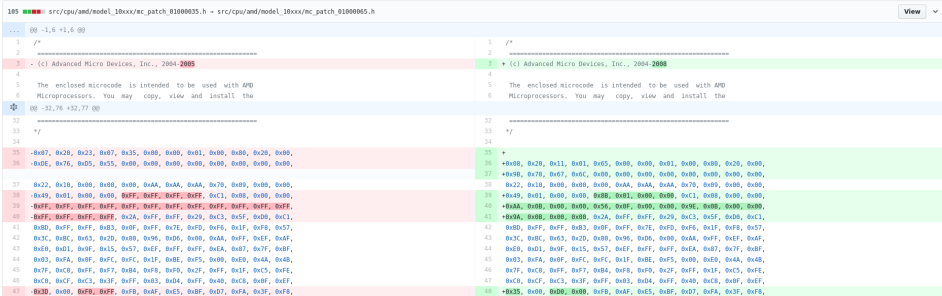
Update hexdump


```
0000000  02062004 00000039 00208000 3e331cfd 00000000 00000000 00000048 aaaaaa00
0000020  00000644 00000140 ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
0000040  00fffbff fd9035c3 a9fe7bf8 3f00ff0f c7ffdffe 3c03bc1e 00d57a00 fffffe14
0000060  eb1fe1fe ff3cf64c 007f8f79 f0266027 18fe1d74 6abd2000 e7fffbcf 0ff0f3f0
0000080  9fffee3f 1bc3e440 07fc01fe ff0dfe00 ebe00035 3fe00ff0 f86ff007 ff003fc0
00000a0  e1bfc01f fe00ff03 86ff007f 7f001a30 f007f81f 37f003fc c01fe07f dfe00ff0
00000c0  007f81ff 7f003fc3 000f7ffc 03fc0ff8 fc01fe1b 0ff03fe0 f007f86f 3fc0ff00
00000e0  c01fe1bf 07bffe00 fe07fc01 00ff0dfe f81ff007 03fc37f8 e07fc01f 0ff0dfe0
0000100  dfff0003 03fe00ff 7f86ff00 0ff003fc fe1bfc01 3fe00ff0 f86ff007 ff001efe
0000120  ff007f81 c37f803f fc01fe07 0dfe00ff f007f81f 37f003fc c000f7ff 803fc0ff
0000140  bfc01fe1 00ff03fe ff007f86 03fc0ff8 fc01fe1b 007bffe0 1fe07fc0 e00ff0df
0000160  7f81ff00 803fc37f fe07fc01 00ff0dfe 3dfff000 f03fe00f 07f86ff0 c0ff003f
0000180  1fe1bfc0 03fe00ff 7f86ff00 fff001e1 1ff007f8 fc37f003 7fc01fe0 f0dfe00f
00001a0  ff007f81 c37f803f fc000f7f f003fc0f 1bfc01fe e00ff03f 6ff007f8 803fc0ff
00001c0  bfc01fe1 0007bffe 01fe07fc fe00ff0d 07f81ff0 f003fc37 1fe07fc0 e00ff0df
00001e0  03dfff00 ff03fe00 007f86ff fc0ff003 01fe1bfc f03fe00f 07f86ff0 efff0001
0000200  81ff007f 3fc37f80 07fc01fe ff0dfe00 1ff007f8 fc37f003 ffc000f7 ff003fc0
0000220  e1bfc01f fe00ff03 86ff007f f003fc0f 1bfc01fe e0007bff c01fe07f dfe00ff0
0000240  007f81ff 7f803fc3 01fe07fc fe00ff0d 003dfff0 0ff03fe0 f007f86f 3fc0ff00
0000260  c01fe1bf ff03fe00 007f86ff 1efff000 f81ff007 03fc37f8 e07fc01f 0ff0dfe0
0000280  81ff007f 3fc37f80 7ffc000f 0ff003fc fe1bfc01 3fe00ff0 f86ff007 ff003fc0
00002a0  e1bfc01f fe0007bf fc01fe07 0dfe00ff f007f81f 37f003fc c01fe07f dfe00ff0
00002c0  0003dfff 00ff03fe ff007f86 03fc0ff8 fc01fe1b 0ff03fe0 f007f86f 01efff00
00002e0  7f81ff00 803fc37f fe07fc01 00ff0dfe f81ff007 03fc37f8 f7ffc000 c0ff003f
0000300  1fe1bfc0 03fe00ff 7f86ff00 0ff003fc fe1bfc01 ffe0007b 7fc01fe0 f0dfe00f
0000320  ff007f81 c37f803f fc01fe07 0dfe00ff f0003dff e00ff03f 6ff007f8 803fc0ff
0000340  bfc01fe1 00ff03fe ff007f86 001efff8 07f81ff0 f003fc37 1fe07fc0 e00ff0df
0000360  7f81ff00 803fc37f 0f7ffc00 fc0ff003 01fe1bfc f03fe00f 07f86ff0 c0ff003f
0000380  1fe1bfc0 bffe0007 07fc01fe ff0dfe00 1ff007f8 fc37f003 7fc01fe0 f0dfe00f
00003a0  ff0003df fe00ff03 86ff007f f003fc0f 1bfc01fe e00ff03f 6ff007f8 8001efff
```

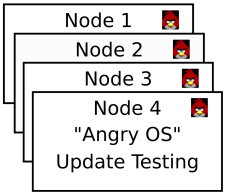
Update hexdump


```
0000000  02062004 00000039 00208000 3e331cfd 00000000 00000000 00000048 aaaaaa00
0000020  00000644 00000140 ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
0000040  00fffbff fd9035c3 a9fe7bf8 3f00ff0f c7ffdffe 3c03bc1e 00d57a80 fffffe14
0000060  eb1fe1fe ff3cf64c 807f879f f0266027 18fe1d74 6abd2000 e7fffbcf 0ff0f3f0
0000080  9fffee3f 1bc3e440 07fc01fe ff0dfe00 ebe00035 3fe00ff0 f86ff007 ff803fc0
00000a0  e1bfc01f fe00ff03 86ff007f 7a001a30 f007f81f 37f803fc c01fe07f dfe00ff0
00000c0  007f81ff 7f803fc3 000f7ffc 03fc0ff8 fc01fe1b 0ff03fe0 f007f86f 3fc0ff80
00000e0  c01fe1bf 07bffe00 fe07fc01 00ff0dfe f81ff007 03fc37f8 e07fc01f 0ff0dfe0
0000100  dfff0003 03fe00ff 7f86ff00 0ff803fc fe1bfc01 3fe00ff0 f86ff007 ff801fef
0000120  f007f81 c37f803f fc01fe07 0dfe00ff f007f81f 37f803fc c000f7ff 803fc0ff
0000140  bfc01fe1 00ff03fe ff007f86 03fc0ff8 fc01fe1b 007bffe0 1fe07fc0 e00ff0df
0000160  7f81ff00 803fc37f fe07fc01 00ff0dfe 3dfff000 f03fe00f 07f86ff0 c0ff803f
0000180  1fe1bfc0 03fe00ff 7f86ff00 fff801e 1ff007f8 fc37f803 7fc01fe0 f0dfe00f
00001a0  ff007f81 c37f803f fc000f7f f803fc0f 1bfc01fe e00ff03f 6ff007f8 803fc0ff
00001c0  bfc01fe1 0007bffe 01fe07fc fe00ff0d 07f81ff0 f803fc37 1fe07fc0 e00ff0df
00001e0  03dfff00 ff03fe00 007f86ff fc0ff803 01fe1bfc f03fe00f 07f86ff0 efff8001
0000200  81ff007f 3fc37f80 07fc01fe ff0dfe00 1ff007f8 fc37f803 ffc000f7 ff803fc0
0000220  e1bfc01f fe00ff03 86ff007f f803fc0f 1bfc01fe e0007bff c01fe07f dfe00ff0
0000240  007f81ff 7f803fc3 01fe07fc fe00ff0d 003dfff0 0ff03fe0 f007f86f 3fc0ff00
0000260  c01fe1bf ff03fe00 007f86ff 1efff800 f81ff007 03fc37f8 e07fc01f 0ff0dfe0
0000280  81ff007f 3fc37f80 7ffc000f 0ff803fc fe1bfc01 3fe00ff0 f86ff007 ff803fc0
00002a0  e1bfc01f fe0007bf fc01fe07 0dfe00ff f007f81f 37f803fc c01fe07f dfe00ff0
00002c0  0003dfff 00ff03fe ff007f86 03fc0ff8 fc01fe1b 0ff03fe0 f007f86f 01efff80
00002e0  7f81ff00 803fc37f fe07fc01 00ff0dfe f81ff007 03fc37f8 f7ffc000 c0ff803f
0000300  1fe1bfc0 03fe00ff 7f86ff00 0ff803fc fe1bfc01 ffe0007b 7fc01fe0 f0dfe00f
0000320  ff007f81 c37f803f fc01fe07 0dfe00ff f0003dff e00ff03f 6ff007f8 803fc0ff
0000340  bfc01fe1 00ff03fe ff007f86 001efff8 07f81ff0 f803fc37 1fe07fc0 e00ff0df
0000360  7f81ff00 803fc37f 0f7ffc00 fc0ff803 01fe1bfc f03fe00f 07f86ff0 c0ff803f
0000380  1fe1bfc0 bffe0007 07fc01fe ff0dfe00 1ff007f8 fc37f803 7fc01fe0 f0dfe00f
00003a0  ff0003df fe00ff03 86ff007f f803fc0f 1bfc01fe e00ff03f 6ff007f8 8001efff
```

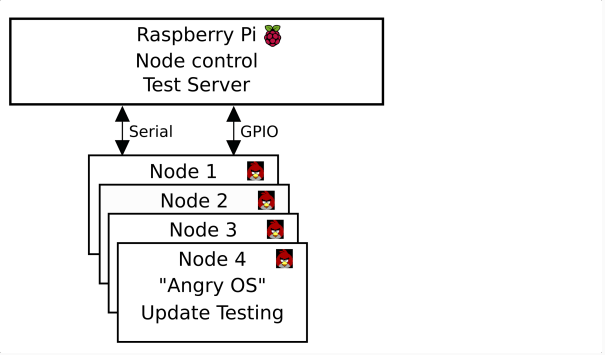
Git Diff of “Crypto”



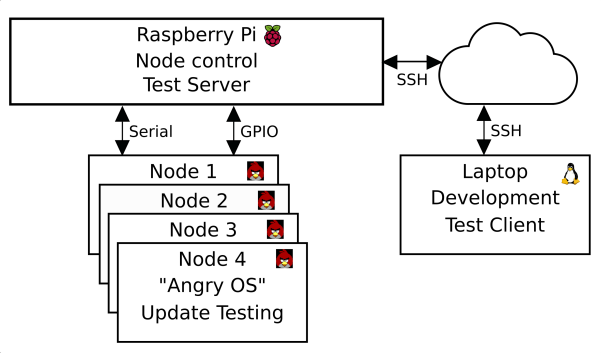
Framework



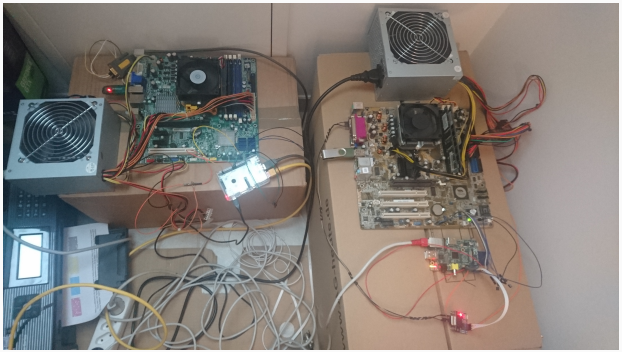
Framework


Raspberry Pi
Node control
Test Server
Serial
GPIO
Node 1
Node 2
Node 3
Node 4
"Angry OS"
Update Testing

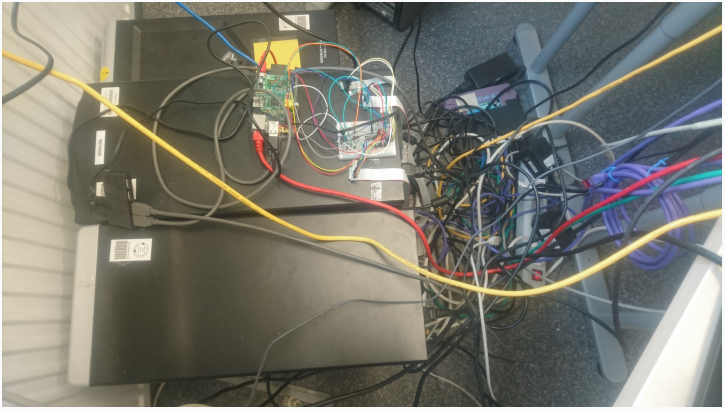
Framework


Raspberry Pi
Node control
Test Server
SSH
Serial
GPIO
SSH
Node 1
Node 2
Node 3
Node 4
"Angry OS"
Update Testing
Laptop
Development
Test Client

Framework



Framework

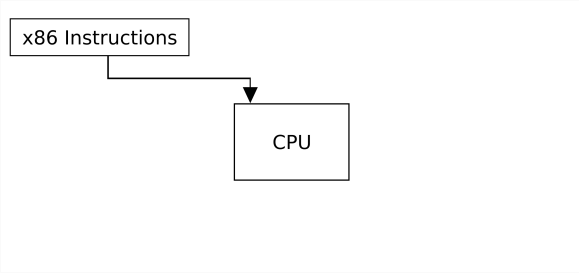


Reverse Engineering Setting

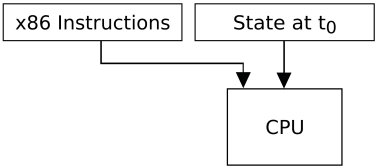
RUHR UNIVERSITÄT BOCHUM RUB

- Unknown instruction set analysis
- Black box model with oracle
- Feed inputs, filter and observe outputs
- Infer structure, encoding, meaning

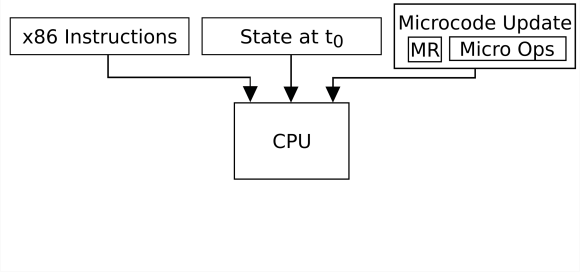
Processor Oracle



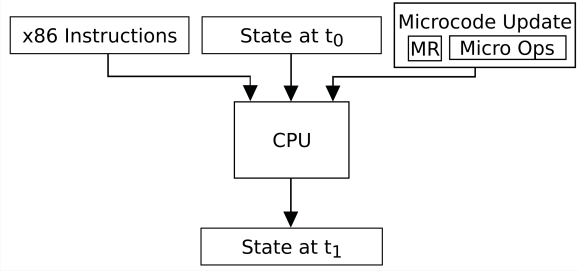
Processor Oracle



Processor Oracle



Processor Oracle



Heatmaps

RUHR UNIVERSITÄT BOCHUM RUB

ROM Address	vector instruction
0x900 - 0x913	-
0x900 - 0x913	-
0x914 - 0x917	rep_cmps_mem8
0x918 - 0x95f	-
0x960	mul_mem16
0x961	idiv
0x962	mul_reg16
0x963	-
0x964	imul_mem16
0x965	bound
0x966	imul_reg16
0x967	-
0x968	bts_imm
0x969 - 0x971	-
0x972 - 0x973	div
0x974 - 0x975	-
0x976 - 0x977	idiv
0x978	-
0x979 - 0x97a	idiv
0x97b - 0x9a7	-
0x9a8	btr_imm
0x9a9 - 0x9ad	-
0x9ae	mfence
0x9af - 09ff	-

Input:

```
eax 00000101 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

```
00000000000001111101000000011110110000000000000000000011010101
```

Input:

```
eax 00000101 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

```
00000000000001111101000000011110110000000000000000000011010101
```

Output:

```
eax 000001d6 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

Encoding Reverse Engineering

RUHR UNIVERSITÄT BOCHUM RUB

Input:

```
eax 00000101 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

```
00000000000001111101000000011110110000000000000000000000001010101
```

Encoding Reverse Engineering

RUHR UNIVERSITÄT BOCHUM RUB

Input:

```
eax 00000101 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

```
00000000000001111101000000011110110000000000000000000000001010101
```

Output:

```
eax 0000015 6 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

Encoding Reverse Engineering

RUHR UNIVERSITÄT BOCHUM RUB

Input:

```
eax 00000101 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

```
00000000000001111101000000011110110000000000000000000000001010101
```

Output:

```
eax 000001 56 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

eax = eax + 0x55

Encoding Reverse Engineering

RUHR UNIVERSITÄT BOCHUM RUB

Input:

```
eax 00000101 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

```
00000000000001111101000000011110110000000000000000000001010101
```

Output:

```
eax 0000015 6 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

eax = eax + 0x55

```
                                                          Imm
00000000000001111101000000011110110000000000000000 0000000001010101
                                                    0x55
```

Input:

```
eax 00000101 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

```
00000001100001111101000000011110110000000000000000000011010101
```

Encoding Reverse Engineering

RUHR UNIVERSITÄT BOCHUM RUB

Input:

```
eax 00000101 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

```
00000001100001111101000000011110110000000000000000000011010101
```

Output:

```
eax 000001d4 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

Encoding Reverse Engineering

RUHR UNIVERSITÄT BOCHUM RUB

Input:

```
eax 00000101 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

```
00000001100001111101000000011110110000000000000000000000001010101
```

Encoding Reverse Engineering

RUHR UNIVERSITÄT BOCHUM RUB

Input:

```
eax 00000101 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

```
00000001100001111101000000011110110000000000000000000000001010101
```

Output:

```
eax 0000015 4 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

Encoding Reverse Engineering

RUHR UNIVERSITÄT BOCHUM RUB

Input:

```
eax 00000101 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

```
00000001100001111101000000011110110000000000000000000000001010101
```

Output:

```
eax 00000154 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

$eax = eax \oplus 0x55$

Encoding Reverse Engineering

RUHR UNIVERSITÄT BOCHUM RUB

Input:

```
eax 00000101 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

```
00000001100001111101000000011110110000000000000000000000001010101
```

Output:

```
eax 000001 54 ebx 00000101 ecx 00000102 edx 00000103
esi 00000104 edi 00000105 ebp 00000106 esp 0013b4??
```

eax = eax ⊕ 0x55

```
Uk1 Operation                                                  Imm
0   000000110 00011111010000000111101100000000000000 0000000001010101
    xor                                                        0x55
```

Micro Op Encoding

RUHR UNIVERSITÄT BOCHUM

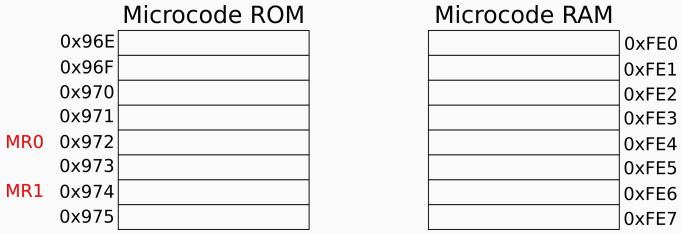
RUB

```
Uk1 Operation SwapOps OpMode Op1    Uk2 PZSFlags CFlag Uk3 OpClass SegReg Size Op2    RegMode Uk4    Uk5Imm Imm
u   ooooooooo x       m      111111 uuu f         f     u   CCC     ssss   zzz  222222 r       uuuuuu u      iiiiiiiiiiiiiiii
0   001111100 0       1      011111 010 0         0     0   000     1111   011  010110 0       000000 0      0000000011010101
    div2                     t24q                             reg     os4    64b  t15q                       0xd5

Uk1 Operation SwapOps OpMode Op1    Uk2 PZSFlags CFlag Uk3 OpClass SegReg Size Op2    RegMode Uk4    Uk5Reg Op3    Uk6Reg
u   ooooooooo x       m      111111 uuu f         f     u   CCC     ssss   zzz  222222 r       uuuuuu uu     333333 uuuuuuuuu
0   001111111 1       0      101001 100 0         0     0   001     0111   010  101010 1       010000 00     010000 000000000
    ld                       regmd5                           ld      rs     32b  t35d                       t9d

Uk1 ShortOprn Condition SwapOps OpMode Op1    Uk2 PZSFlags CFlag Uk3 OpClass SegReg Size Op2    RegMode Uk4    RomAddr
u   oooo      ccccc     x       m      111111 uuu f         f     u   CCC     ssss   zzz  222222 r       uuuuuu aaaaaaaaaaaaaaaa
0   0101      00100     1       1      111001 101 0         0     0   000     1111   011  111011 0       000000 00000000000000011
    jcc       EZF                      t50q                             reg     os4    64b  t52q                  0x3

Uk1              Action Uk2 RomAddr
uuuuuuuuuuuuuuuu ooo    uu  aaaaaaaaaaaa
1111111111111110 010    10  010110100101
                 branch     0x5a5
```

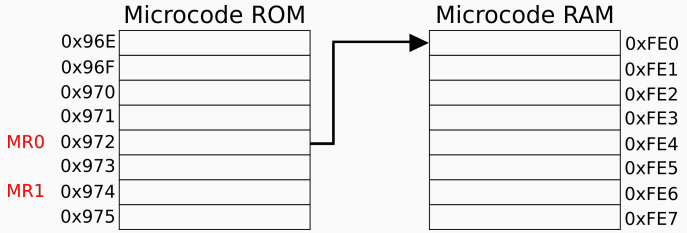
Microcode RTL


```
sub eax, edx
sub.C t56q, rcx, 0x100
jcc ECF, 1
.sw_next // implied sequence word if omitted

ld t1d, [eax]
st [edx], t1d
mov eax, eax
.sw_complete

mov eax, 1
sub.Q rax, rcx
add.EP t56d, eax, ecx
.sw_branch 0xF01
```

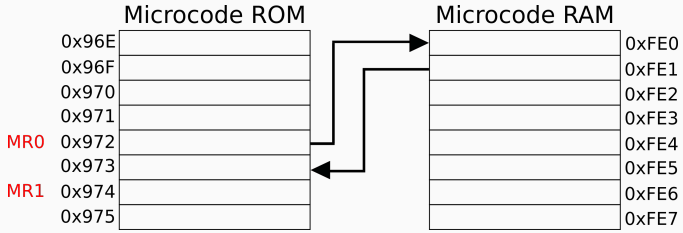
Infer Logic of ROM Triads



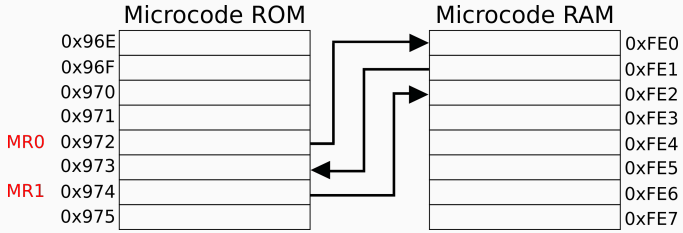
Infer Logic of ROM Triads

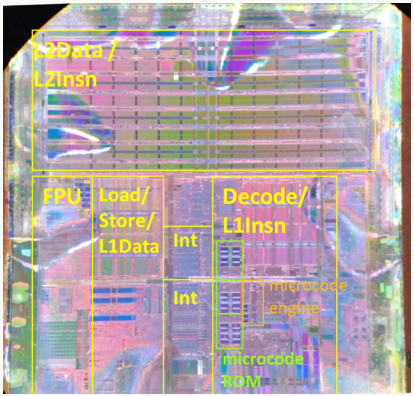


Infer Logic of ROM Triads



Infer Logic of ROM Triads

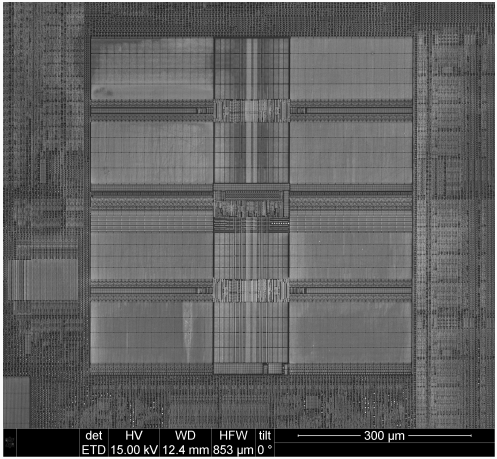



L2Data /
L2Insn
FPU
Load/
Store/
L1Data
Int
Int
Decode/
L1Insn
microcode
engine
microcode
ROM

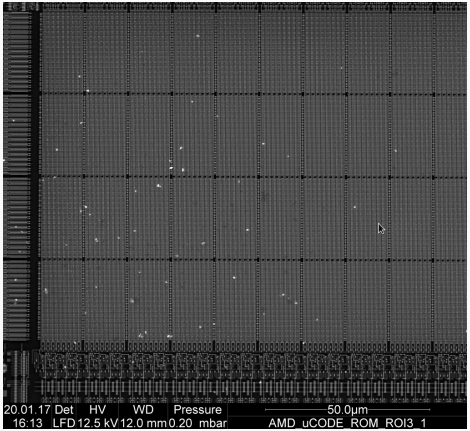
Hardware Analysis

RUHR UNIVERSITÄT BOCHUM

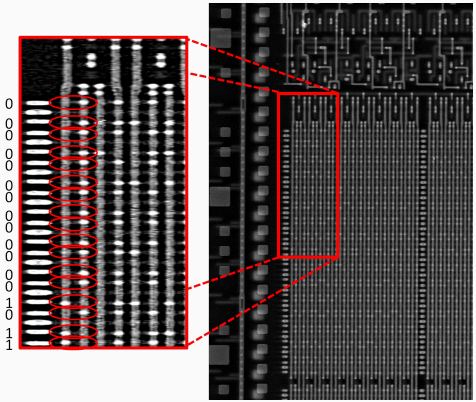
RUB



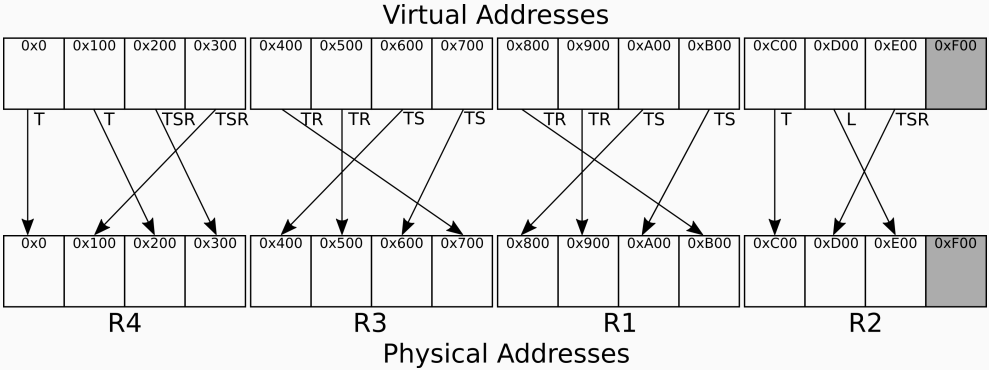
Hardware Analysis



Hardware Analysis



Hardware Analysis - ROM Layout



WRMSR


```
mov t13q, t56q, -0x1
mov t14q, t56q, -0x1
rrl t10q, rcx, 0x10

xor.Z t56w, t10w, -0x4000
jcc EZF, -0x4f7
xor.Z t56w, t10w, -0x3fff

jcc EZF, 0x430
sub.C t12q, rcx, 0x200
jcc nECF, 0x9

mov.Z t11q, t56q, 0x1f
jcc EZF, 0x2a3
sub.C t56q, rcx, 0x17b
```

RE Results

- Heatmaps - location of handlers for x86 instructions in microcode ROM

RE Results

- Heatmaps - location of handlers for x86 instructions in microcode ROM
- 29 Micro Ops
  - Logic, arithmetic, load, store
  - Write x86 program counter
  - Conditional microcode branch
  - Read special internal registers (TSC, CR*, CPL)

RE Results

- Heatmaps - location of handlers for x86 instructions in microcode ROM
- 29 Micro Ops
  - Logic, arithmetic, load, store
  - Write x86 program counter
  - Conditional microcode branch
  - Read special internal registers (TSC, CR*, CPL)
- Sequence word
  - Next triad, sequence complete, unconditional branch

RE Results

- Heatmaps - location of handlers for x86 instructions in microcode ROM
- 29 Micro Ops
  - Logic, arithmetic, load, store
  - Write x86 program counter
  - Conditional microcode branch
  - Read special internal registers (TSC, CR*, CPL)
- Sequence word
  - Next triad, sequence complete, unconditional branch
- Substitution engine - replace bit masks in micro ops with arguments from x86 instruction

RE Results

- Heatmaps - location of handlers for x86 instructions in microcode ROM
- 29 Micro Ops
  - Logic, arithmetic, load, store
  - Write x86 program counter
  - Conditional microcode branch
  - Read special internal registers (TSC, CR*, CPL)
- Sequence word
  - Next triad, sequence complete, unconditional branch
- Substitution engine - replace bit masks in micro ops with arguments from x86 instruction
- ROM dump with disassembly

RE Results

- Heatmaps - location of handlers for x86 instructions in microcode ROM
- 29 Micro Ops
  - Logic, arithmetic, load, store
  - Write x86 program counter
  - Conditional microcode branch
  - Read special internal registers (TSC, CR*, CPL)
- Sequence word
  - Next triad, sequence complete, unconditional branch
- Substitution engine - replace bit masks in micro ops with arguments from x86 instruction
- ROM dump with disassembly
- Augmenting x86 instructions

Offensive Microprograms

RUHR UNIVERSITÄT BOCHUM RUB

- Remote microcode attacks
  - Control flow hijack in browsers induced by microcode
  - Triggered remotely with ASM.JS, WebAssembly

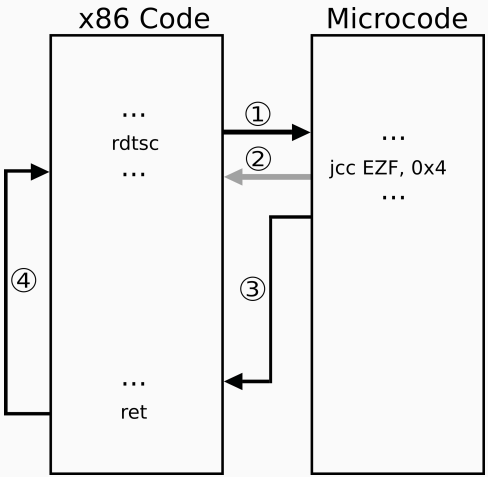
Offensive Microprograms

- Remote microcode attacks
  - Control flow hijack in browsers induced by microcode
  - Triggered remotely with ASM.JS, WebAssembly
- Cryptographic microcode Trojans
  - Introduce timing side-channels in constant-time ECC implementation
  - Inject faults to enable fault attacks

Constructive Microprograms

RUHR UNIVERSITÄT BOCHUM RUB

- Hooking framework
  - Hook selected x86 instruction, jump to trampoline
  - Apply pre-filter in microcode directly
  - No overhead for non-hooked instructions



Constructive Microprograms

- `RDTSC` - Limit resolution in userspace

Constructive Microprograms

RUHR UNIVERSITÄT BOCHUM RUB

- `RDTSC` - Limit resolution in userspace
- `BOUND` - Replace with HWASAN instruction
  - `bound reg, size`
  - Checks the address in reg for an access of given size
  - Follows HWASAN semantics
  - No x86 register used, smaller code size, faster

Constructive Microprograms

- `RDTSC` - Limit resolution in userspace
- `BOUND` - Replace with HWASAN instruction
  - `bound reg, size`
  - Checks the address in reg for an access of given size
  - Follows HWASAN semantics
  - No x86 register used, smaller code size, faster
- `WRMSR` - Authenticate microcode updates
  - Before update compute HMAC of update blob
  - Verify against signature appended to original update
  - Jump to default handler to apply update

Constructive Microprograms

RUHR UNIVERSITÄT BOCHUM RUB

- Microcode enclave
  - Malicious kernel code cannot interfere, remote attestation
  - Limitations wrt code size and memory
  - Enclave program is implemented in microcode, must be well behaving

Constructive Microprograms

- Microcode enclave
  - Malicious kernel code cannot interfere, remote attestation
  - Limitations wrt code size and memory
  - Enclave program is implemented in microcode, must be well behaving
- Instruction Set Randomization
  - Systems defense against code injection, JIT-ROP
  - Shuffle instruction semantics
  - Mask input/output before reading from/writing to registers or memory

Outline

- What is microcode?
- Architectural crash course
- Analysis
- Demo

Demo - Scenario

RUHR UNIVERSITÄT BOCHUM

RUB

- Not an attack, demonstration of capabilities of malicious microcode update
- Unmodified Firefox and Linux
- Malicious microcode update loaded
- Backdoor triggered via Webassembly module

Firefox SHRD Backdoor (simplified)

RUHR UNIVERSITÄT BOCHUM

RUB

```
mov t1d, 0xdead   // load trigger constant
sll t1d, 16
add t1d, 0xc0de
```

Firefox SHRD Backdoor (simplified)

RUHR UNIVERSITÄT BOCHUM RUB

```
mov t1d, 0xdead   // load trigger constant
sll t1d, 16
add t1d, 0xc0de

sub.Z t1d, regmd4 // compare argument 1 to constant
jcc nEZF, 0x3     // jump to implementation of shrd, not shown
mov eax, 11       // syscall number -> eax
```

Firefox SHRD Backdoor (simplified)

RUHR UNIVERSITÄT BOCHUM RUB

```
mov t1d, 0xdead   // load trigger constant
sll t1d, 16
add t1d, 0xc0de

sub.Z t1d, regmd4 // compare argument 1 to constant
jcc nEZF, 0x3     // jump to implementation of shrd, not shown
mov eax, 11       // syscall number -> eax

add ebx, ecx, 20  // prepare buffer offsets and syscall args
...
```

Firefox SHRD Backdoor (simplified)

RUHR UNIVERSITÄT BOCHUM RUB

```
mov t1d, 0xdead   // load trigger constant
sll t1d, 16
add t1d, 0xc0de

sub.Z t1d, regmd4 // compare argument 1 to constant
jcc nEZF, 0x3     // jump to implementation of shrd, not shown
mov eax, 11       // syscall number -> eax

add ebx, ecx, 20  // prepare buffer offsets and syscall args
...

mov t1d, regmd6   // get EIP offset from argument 2
add t1d, pcd      // add offset and next EIP
writePC t1d       // set next EIP to new value
.sw_complete
```

Security Implications

- No signature, any update accepted

Security Implications

- No signature, any update accepted
- Backdoors are possible

Security Implications

RUHR UNIVERSITÄT BOCHUM

RUB

- No signature, any update accepted
- Backdoors are possible
- Trojans very hard to detect

Security Implications

- No signature, any update accepted
- Backdoors are possible
- Trojans very hard to detect
- Not really fixable (hardware recall...)

Security Implications

- No signature, any update accepted
- Backdoors are possible
- Trojans very hard to detect
- Not really fixable (hardware recall...)
- Hacky fix: load update to secure (or brick) update mechanism

Attack Scenarios

RUHR UNIVERSITÄT BOCHUM RUB

- Vendor
  - Full knowledge and (for newer CPUs) signing keys
  - Easy to hide backdoor in encrypted “bugfix” update or directly in silicon

Attack Scenarios

- Vendor
  - Full knowledge and (for newer CPUs) signing keys
  - Easy to hide backdoor in encrypted “bugfix” update or directly in silicon
- State actor
  - Can coerce vendors into disclosing knowledge and keys
  - Ressources for targeted (hardware-)RE and interdiction
  - Insertion into BIOS/UEFI ROM

Attack Scenarios

- Vendor
  - Full knowledge and (for newer CPUs) signing keys
  - Easy to hide backdoor in encrypted “bugfix” update or directly in silicon
- State actor
  - Can coerce vendors into disclosing knowledge and keys
  - Ressources for targeted (hardware-)RE and interdiction
  - Insertion into BIOS/UEFI ROM
- Advanced independent attacker (unlikely)
  - Knowledge via RE or leak
  - Signing keys via weak crypto or leak
  - Gain kernel access, escalate to BIOS write or update BIOS via physical attack
  - Probably not useful for widespread attacks

Detecting Backdoors

- Timing measurements
  - Timing measurements for all instructions and input values
  - Backdoor will introduce slight timing delay and/or change the retired micro op count
  - Especially useful for comparing two microcode patch levels

Detecting Backdoors

- Timing measurements
  - Timing measurements for all instructions and input values
  - Backdoor will introduce slight timing delay and/or change the retired micro op count
  - Especially useful for comparing two microcode patch levels
- Reverse engineering
  - RE the microcode update and ROM
  - For 100% coverage also requires RE of the complete silicon
  - Check implementation of each instruction against spec
  - Open source cores are an advantage (if you trust your fab)

Conclusion

RUHR UNIVERSITÄT BOCHUM RUB

- Microcode can be reverse engineered and changed
- . . . and gives powerful capabilities
- Sample updates and driver available on Github (use at your own peril!)

`https://github.com/RUB-SysSec/Microcode`

Horst Görtz Institute for IT-Security
Ruhr-Universität Bochum

emproof
www.emproof.de