Backdoor in the Core

Altering the Intel x86 Instruction Set at Runtime

Who Are We?

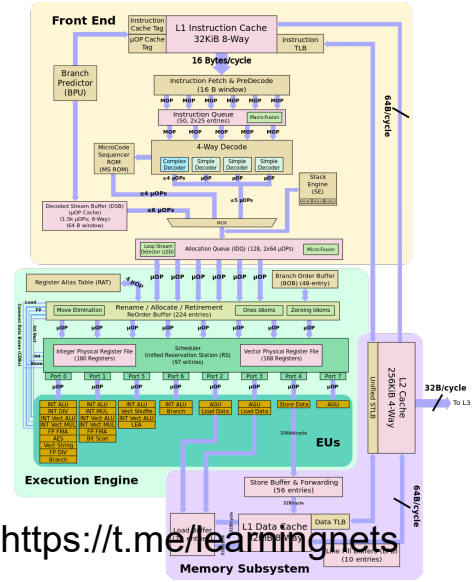
Alexander Krog

- Vulnerability researcher at Vectorize
- CTF player (Kalmarunionen)
- email: alexander.dalsgaard.krog@proton.me
- Discord: @zanderdk

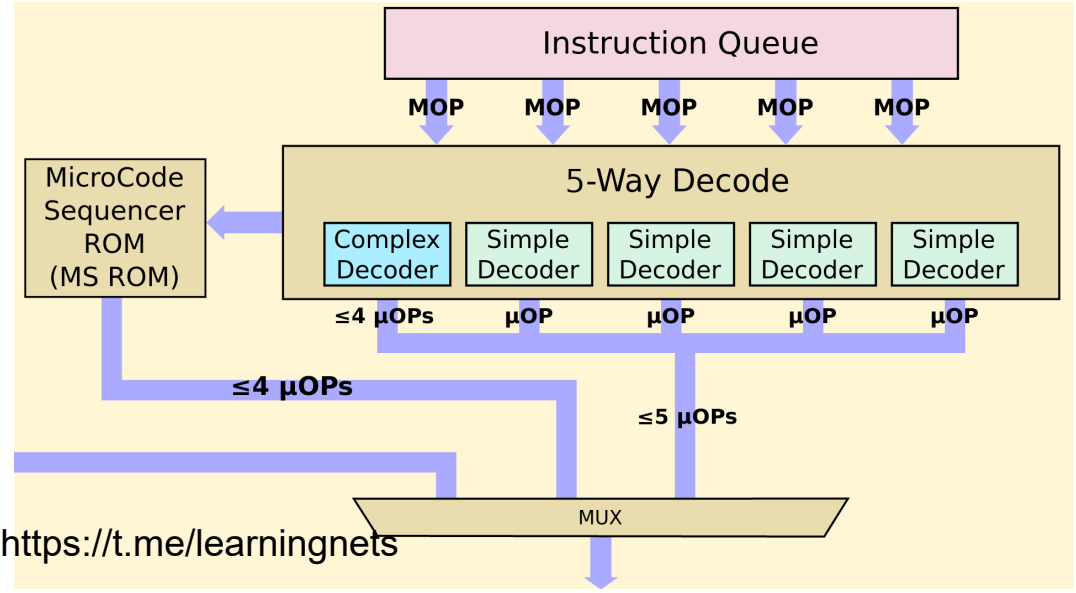
Alexander Skovsende

- Grad student at Danish Technical University
- CTF player (Kalmarunionen)

Architectural Crash Course



https://t.me/learningnets



What/Why is Microcode?

Complex Instructions and CPU features

- Fix CPU bugs
  - Spectre meltdown e.g.

Official Updates

- Applied by kernel/firmware
- Signed & encrypted by Intel

Microcode ROM/RAM

Uaddr	instruction	instruction	instruction	implicit nop	sequence word
U0000 RO	uop 0	uop 1	uop 2	nop	seqw 0
U0004 RO	uop 4	uop 5	uop 6	nop	seqw 1
U0008 RO	uop 8	uop 9	uop A	nop	seqw 2
U000C RO	uop C	uop D	uop E	nop	seqw 3
U0010 RO	uop 10	uop 11	uop 12	nop	seqw 4
...	...	...	...	...	...
U7C00 RW	uop 7C00	uop 7C01	uop 7C02	nop	seqw 1f00
U7C04 RW	uop 7C04	uop 7C05	uop 7C06	nop	seqw 1f01
U7C08 RW	uop 7C08	uop 7C09	uop 7C0A	nop	seqw 1f02
U7C0C RW	uop 7C0C	uop 7C0C	uop 7C0C	nop	seqw 1f03
U7C10 RW	uop 7C10	uop 7C11	uop 7C12	nop	seqw 1f04
...	...	...	...	...	...
U7DFC RW	uop 7DFC	uop 7DFD	uop 7DFE	nop	seqw 1F7F

Example

xadd instruction

- Add and exchange registers

```
xadd rax, rbx          ->    tmp0:= OR_DSZ64(r64src, 0)
                       ->    r64src:= ZEROEXT_DSN(r64dst)
                       ->    r64dst:= ADD_DSZ(tmp0, r64dst)
                       ->    SEQW UEND0
```

Micro Architecture - Registers

Registers

- Normal registers
- 16 tmp registers
- 8 tmp floating point registers
- System register
- Ucode state register

Micro Architecture - Memory

Normal Memory

- Virtual access to RAM
- Physical access to RAM

Ucode memory

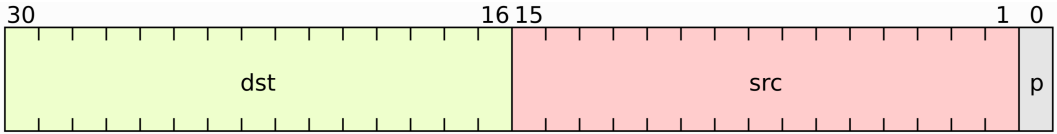
- Data RAM for data
- 0x100 qword

Micro Architecture - Misc

Internal Core Component Communication

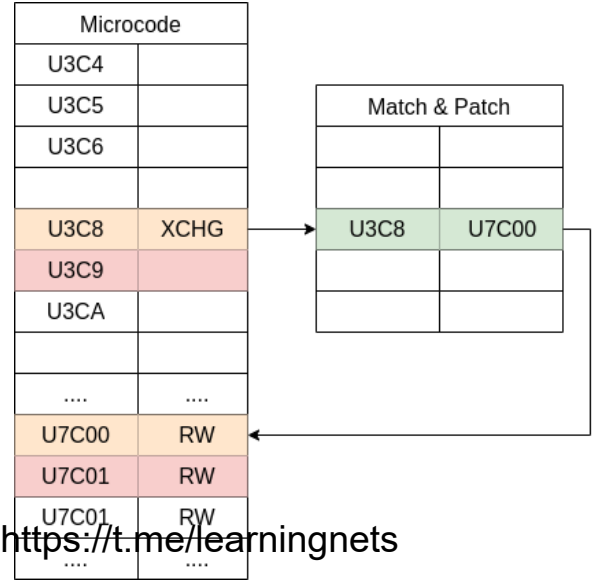
- CRBUS - Control Register Bus
  - Translation look aside buffer
  - Direct access to Caches
  - Microcode Sequencer
- IOSF - Intel On-Chip System Fabric
  - External chip communication

Match & Patch



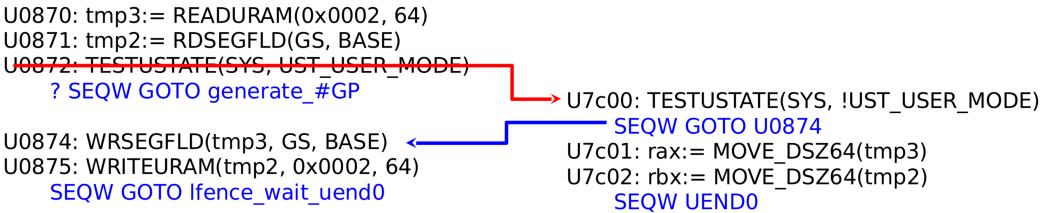
1. p
   - present flag
2. src
   - is a uaddr » 1
3. dst
   - is a uaddr » 1

Match & Patch



Match & Patch

Dynamic Inspection



How can we change micro code?

How can we change micro code?

Microcode Access

- Requires Red Unlock
- Intel debug instructions

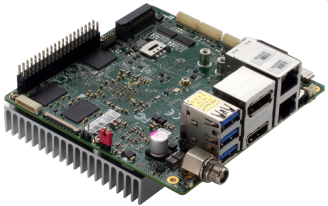
ME-Exploit

- Vulnerability found in Intel Management Engine by Positive Technologies
- Red Unlock
- Intel Debug Instructions

Pre-built flash Image

Pre-built flash Image

- Flash Images: https://libmicro.dev
- Trusted Execution Engine Exploit included



udbgrd & udbgwr

- udbgrd - Micro architecture debug read instruction
- udbgwr - Micro architecture debug write instruction
- access to control register bus

Changing instruction set at run time

- Custom Processing Unit
  - Assembler for microcode
  - works from EFI shell
  - The syntax used so far
  - https://github.com/pietroborrello/CustomProcessingUnit

Lib-Micro

- Lib-Micro
  - Assembler for microcode
  - Dynamic or Statically linked Linux library
  - https://github.com/zanderdk/lib-micro

Lib-Micro

```
void yolo(void) {
    ucode_t ucode_patch[] = {
        {   //0x7da0
            MOVE_DSZ64_DI(TMP0, 0x1337),
            MOVE_DSZ64_DI(RBX, 0xdead),
            MOVE_DSZ64_DR(RAX, TMP0), END_SEQWORD
        }
    };
    patch_ucode(0x7da0, ucode_patch, ARRAY_SZ(ucode_patch));
    hook_match_and_patch(0, 0x0740 /* sysexitq xlat */, 0x7da0);
}
```

Disappearing Microcode Changes

IN AL, DX

- Significantly bigger than other port io instructions
- Re-apply microcode update from Intel

Patch

- Abusing for fuzzing and tracing
- Patched out using rom -> rom match & patch

Finally Stable Microcode Patches!

Demo Time

Backdoor

Syscall Instruction

- Jumps to kernel space
- Kernel handles requested syscall
- rcx contains userspace return address

Backdoor

Pseudocode

```
if (RAX != SYS_write || mem[RSI+0x100] != 0xd00df00d) {
  goto normal_syscall;
}
~
~
~
~
~
~
~
~
~
~
~
```

Backdoor

Pseudocode

```
if (RAX != SYS_write || mem[RSI+0x100] != 0xd00df00d) {
  goto normal_syscall;
}
mem[RSI+0x100] = RIP; // save state
mem[RSI+0x108] = RAX;
mem[RSI+0x110] = RDI;
mem[RSI+0x118] = RSI;
mem[RSI+0x120] = RDX;
~
~
~
~
~
```

Backdoor

Pseudocode

```
if (RAX != SYS_write || mem[RSI+0x100] != 0xd00df00d) {
  goto normal_syscall;
}
mem[RSI+0x100] = RIP; // save state
mem[RSI+0x108] = RAX;
mem[RSI+0x110] = RDI;
mem[RSI+0x118] = RSI;
mem[RSI+0x120] = RDX;
RCX = RSI + 0x130;
~
~
~
~
```

Backdoor

Pseudocode

```
if (RAX != SYS_write || mem[RSI+0x100] != 0xd00df00d) {
  goto normal_syscall;
}
mem[RSI+0x100] = RIP; // save state
mem[RSI+0x108] = RAX;
mem[RSI+0x110] = RDI;
mem[RSI+0x118] = RSI;
mem[RSI+0x120] = RDX;
RCX = RSI + 0x130;
RAX = SYS_mprotect;
RDI = RSI & ~0xfff;
RSI = 0x2000;
RDX = 0x7;
goto normal_syscall;
```

Backdoor

Userspace Code

- fork
- Parent process
  - Restore from saved context
  - Saved from microcode
  - Continue normal execution
- Child process
  - Do the evil e.g. pop calc

Reversing a CPU

Reversing Instruction Set

Partially unknown instructions

- Side by side
- Tracing
- Dynamic testing
  - Playground instructions
  - sysexitq
  - vmwrite etc.

Future work

Future work

- Find bug in CPU
  - Help from intel thanks
  - Red Unlock from software

Thanks and Acknowledgment:

Thanks

- Kalmarunionen

Acknowledgment

- Mark Ermolov et. al.
- Positive Technologies
- CustomProcessingUnit - Pietro Borrello

Questions?