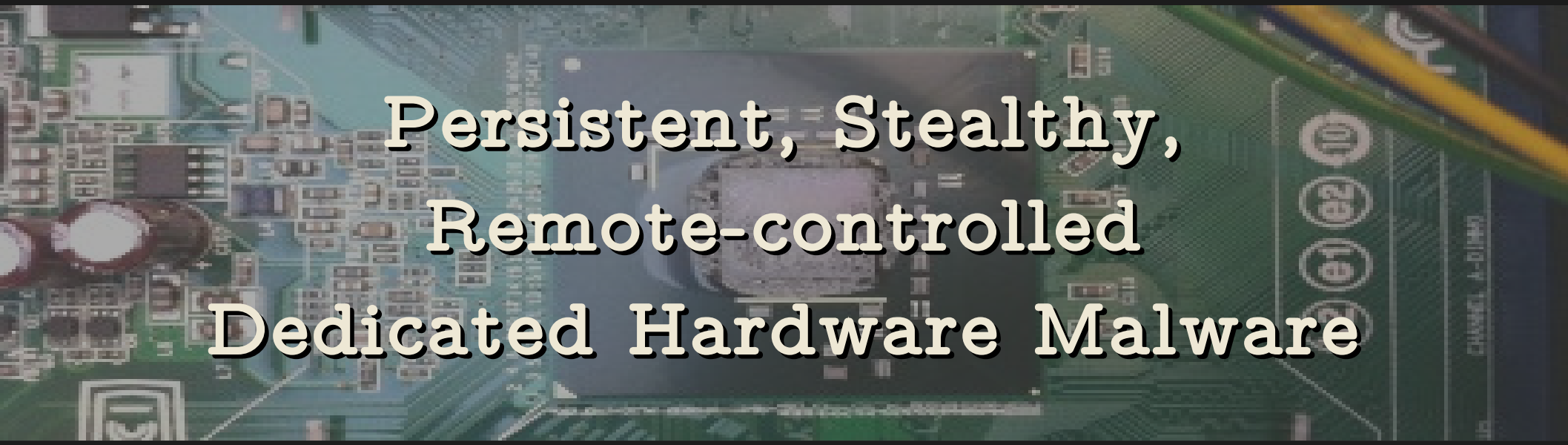


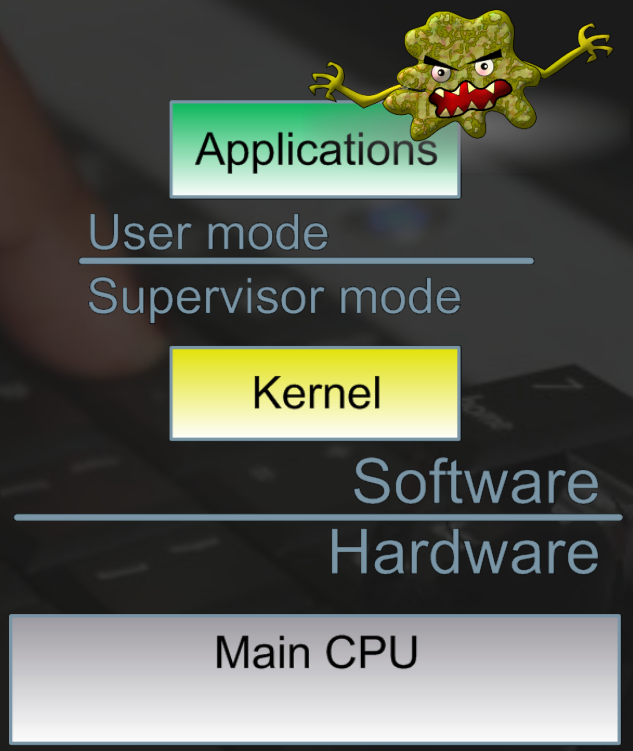
# Persistent, Stealthy, Remote-controlled Dedicated Hardware Malware

**Patrick Stewin and Iurii Bystrov**
Security in Telecommunications (SecT)
TU Berlin
patrickx@sec.t-labs.tu-berlin.de


**44CON 2013, London, UK**

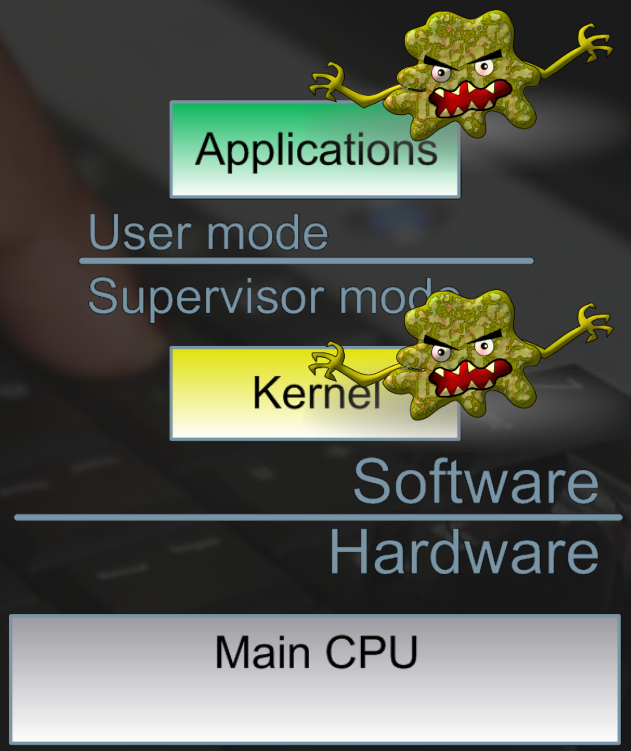
# ARMS RACE

## Malware developers ↔ anti-malware community



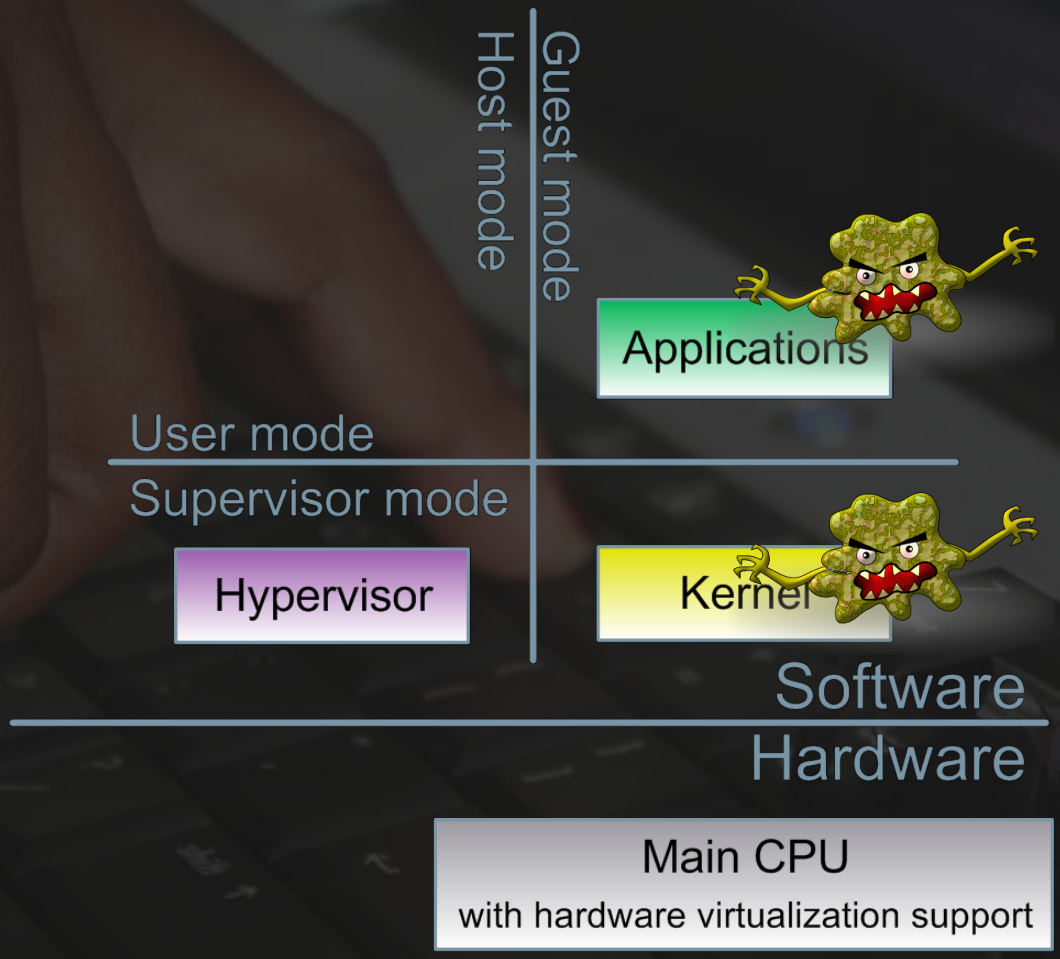
# ARMS RACE

## Malware developers ↔ anti-malware community



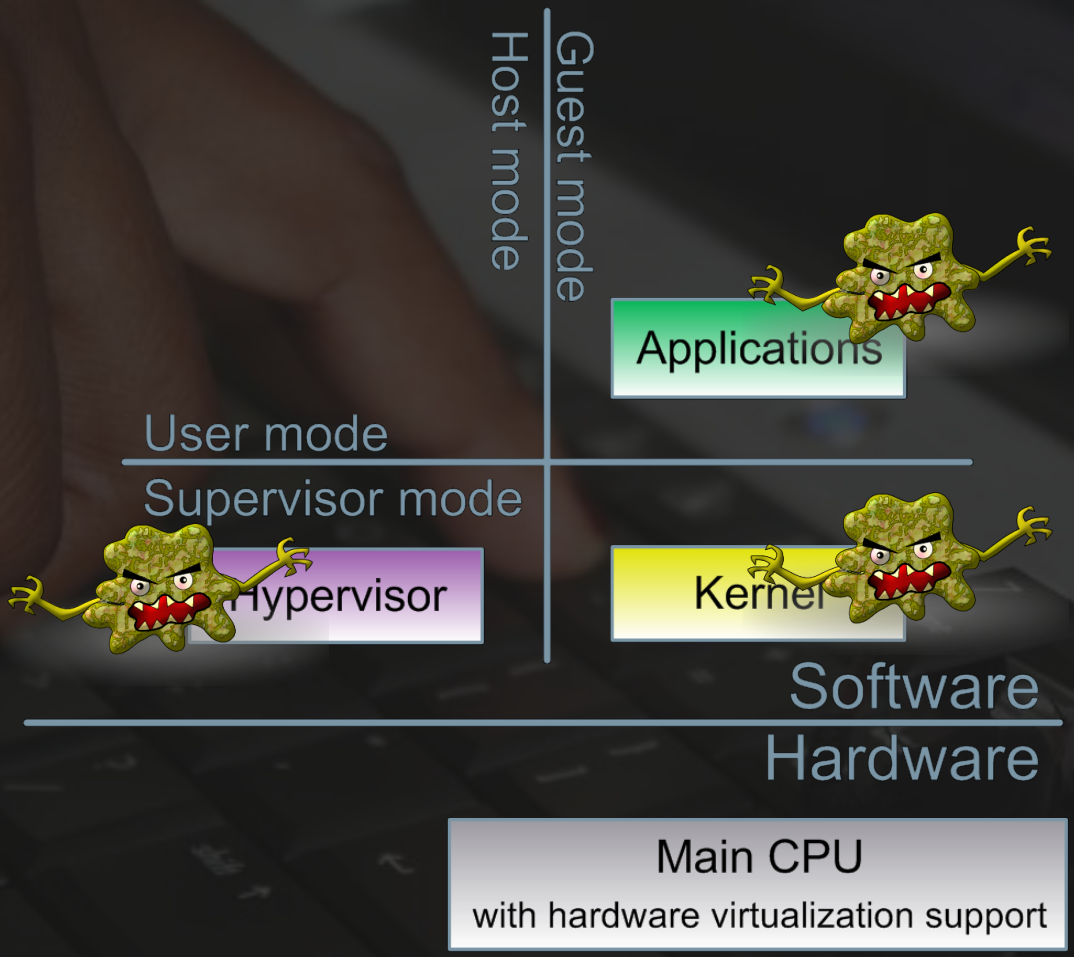
# ARMS RACE

## Malware developers ↔ anti-malware community



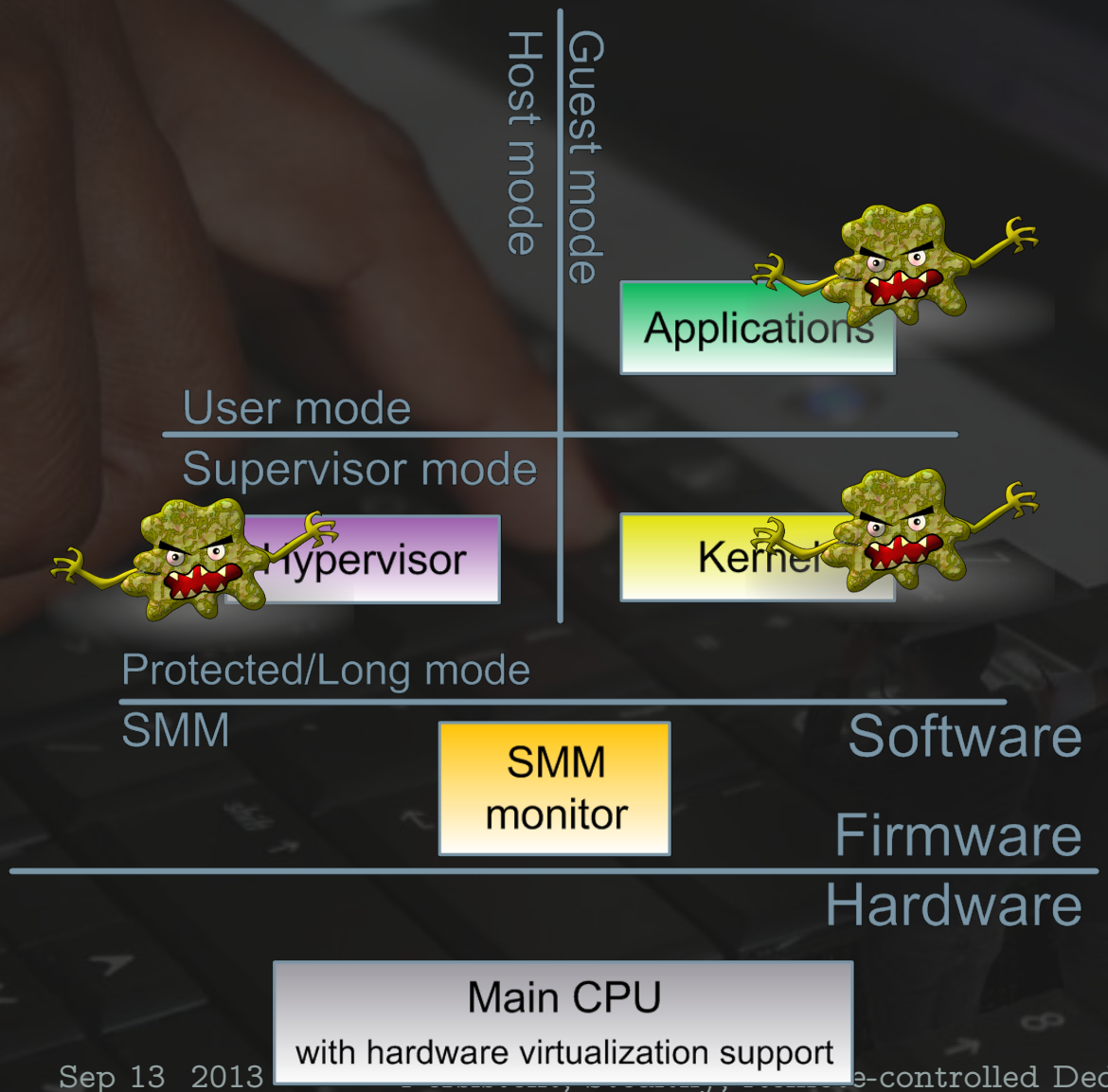
# ARMS RACE

## Malware developers ↔ anti-malware community



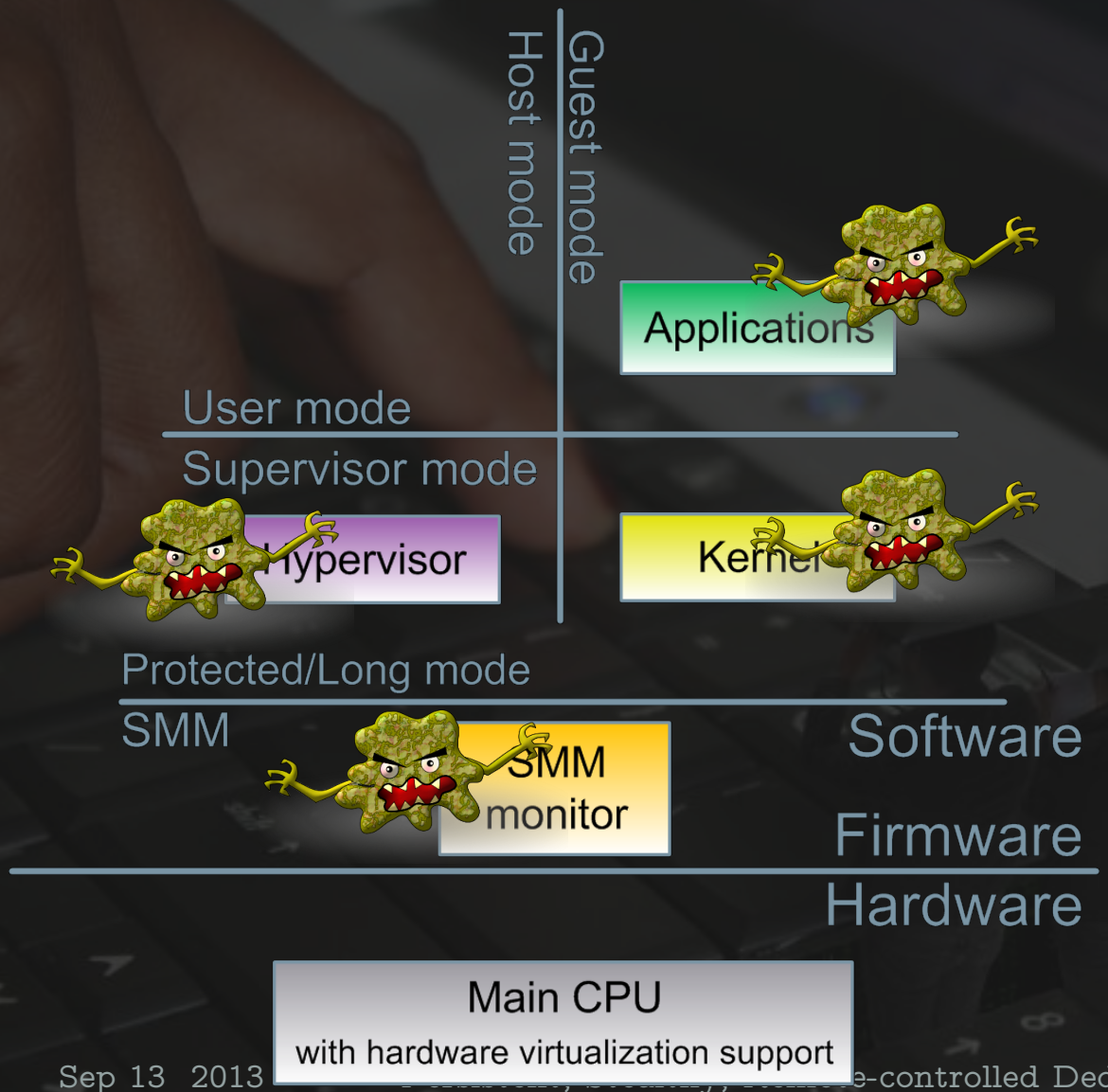
# ARMS RACE

## Malware developers ↔ anti-malware community



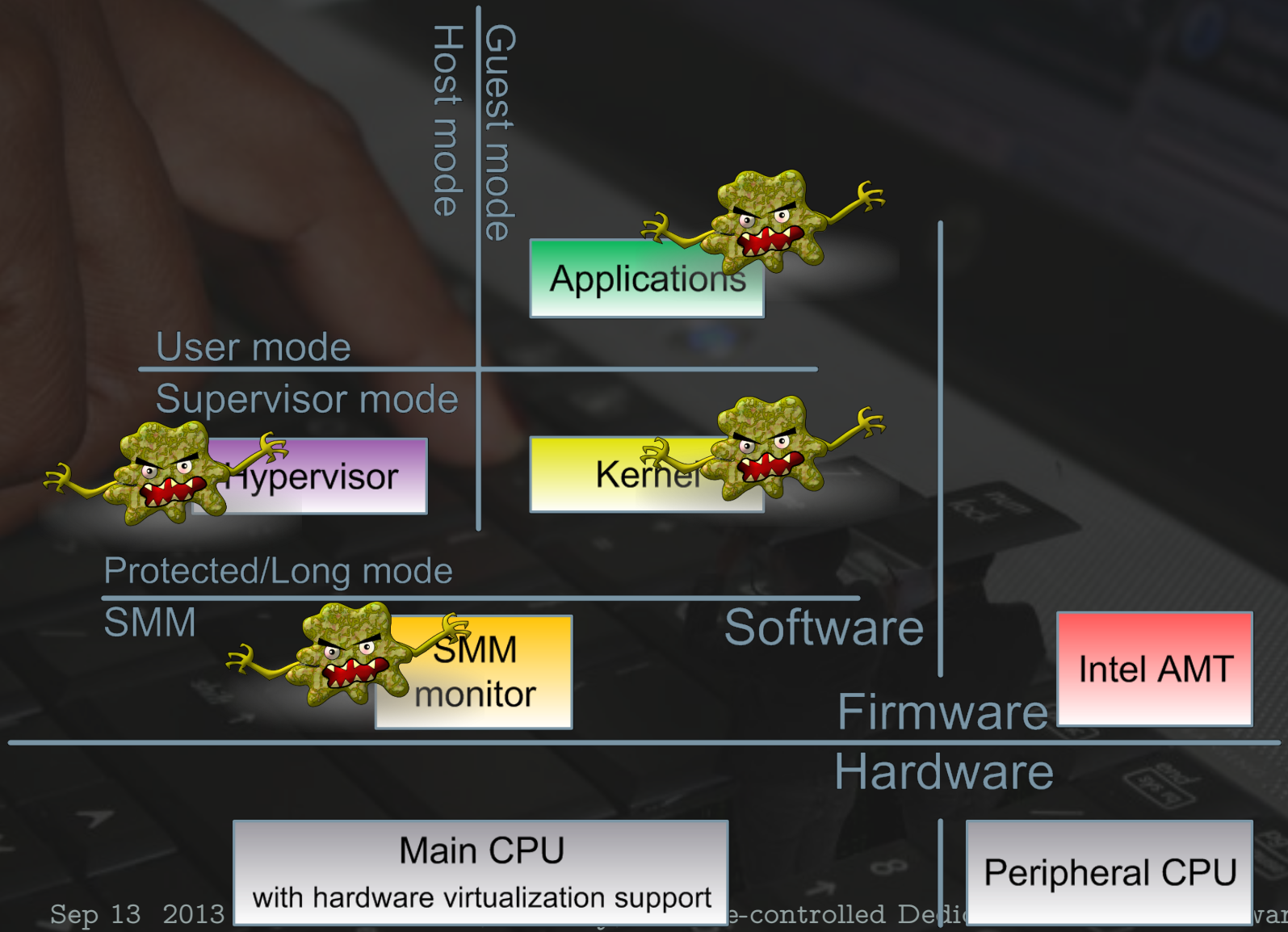
# ARMS RACE

## Malware developers ↔ anti-malware community



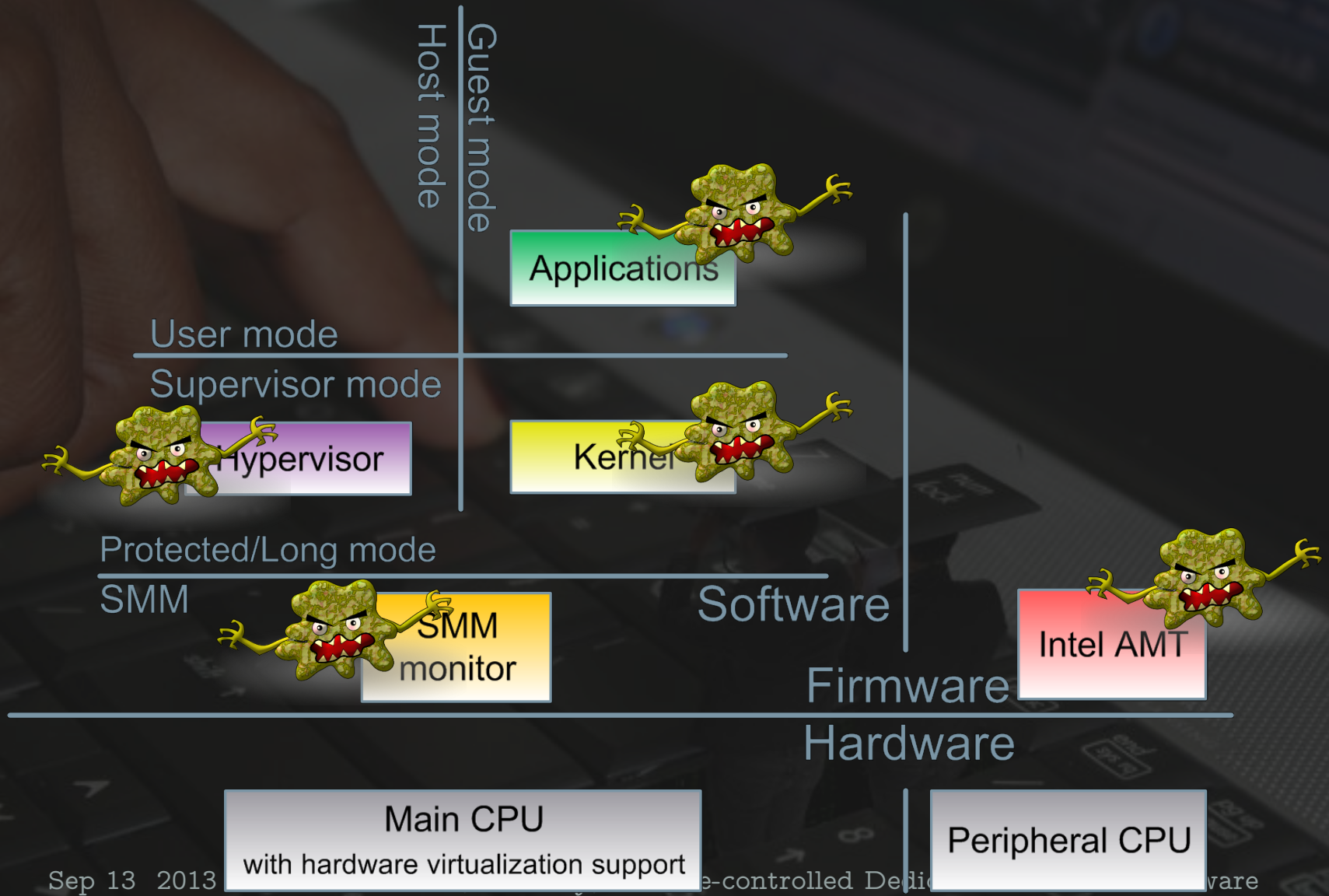
# ARMS RACE

## Malware developers ↔ anti-malware community



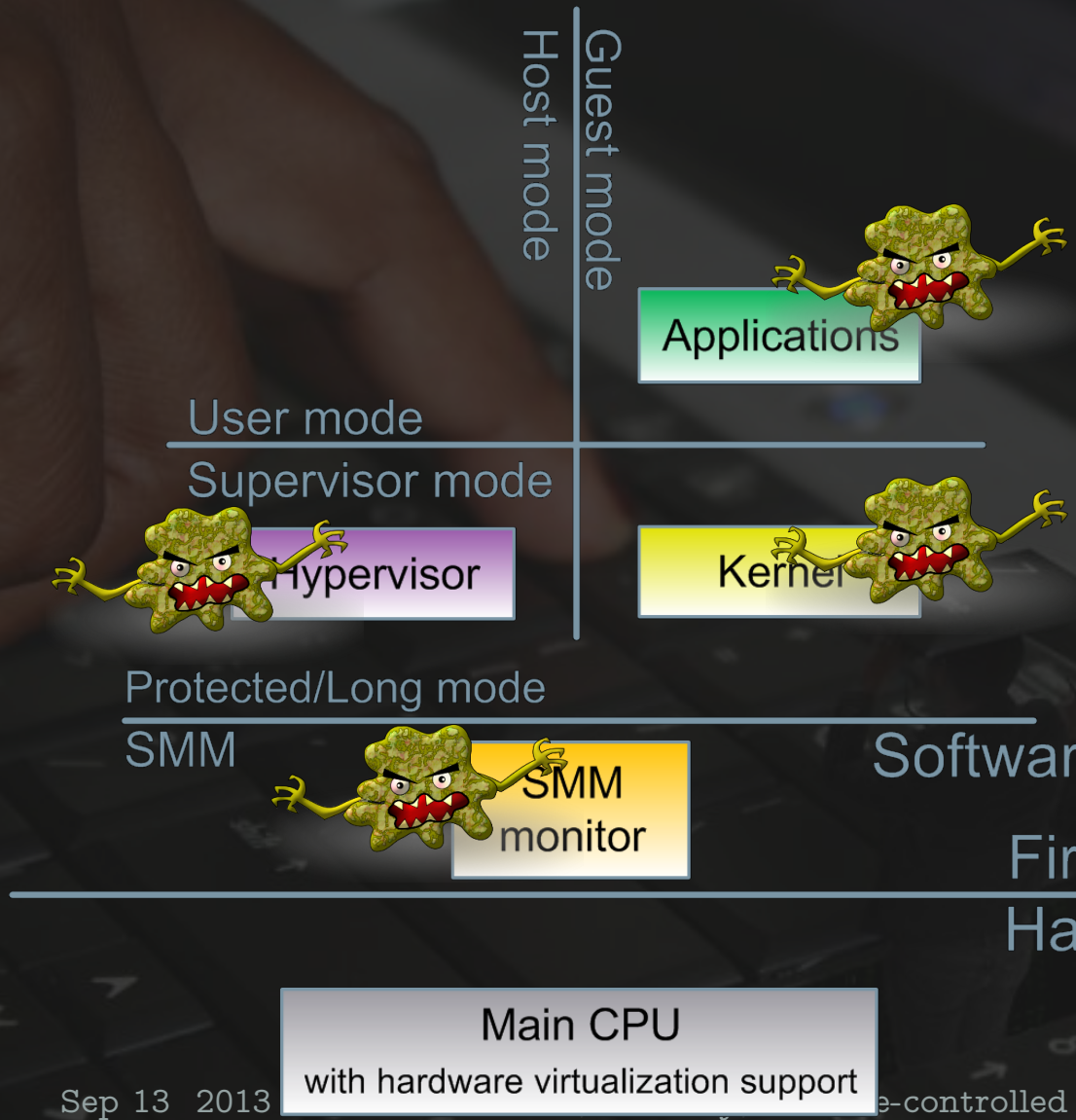
# ARMS RACE

## Malware developers ↔ anti-malware community

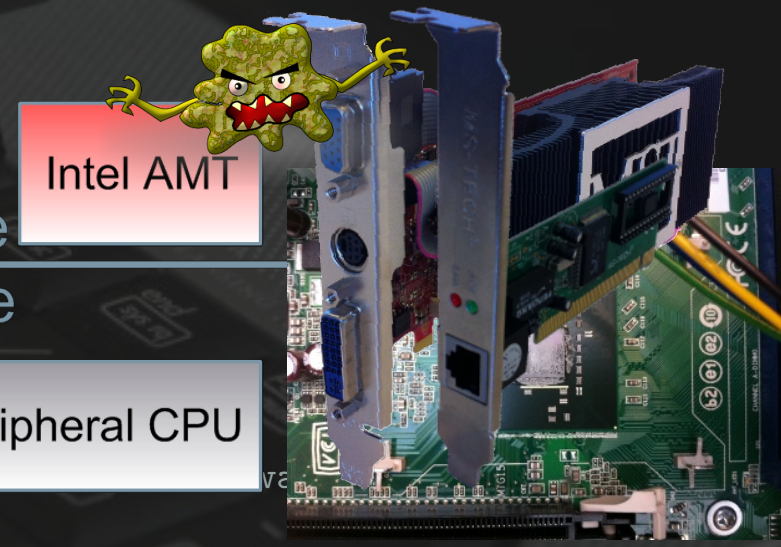


# ARMS RACE

## Malware developers ↔ anti-malware community




- $ Processor/RAM
- $ Direct Memory Access (DMA)
- $ Unconsidered by AV-Software/ Host Firewall



# [patrickx@44con:~$] cat 'Overview'

1. DmA based keystroke loGGER
2. Out-of-Band network channel
3. Covert network channel

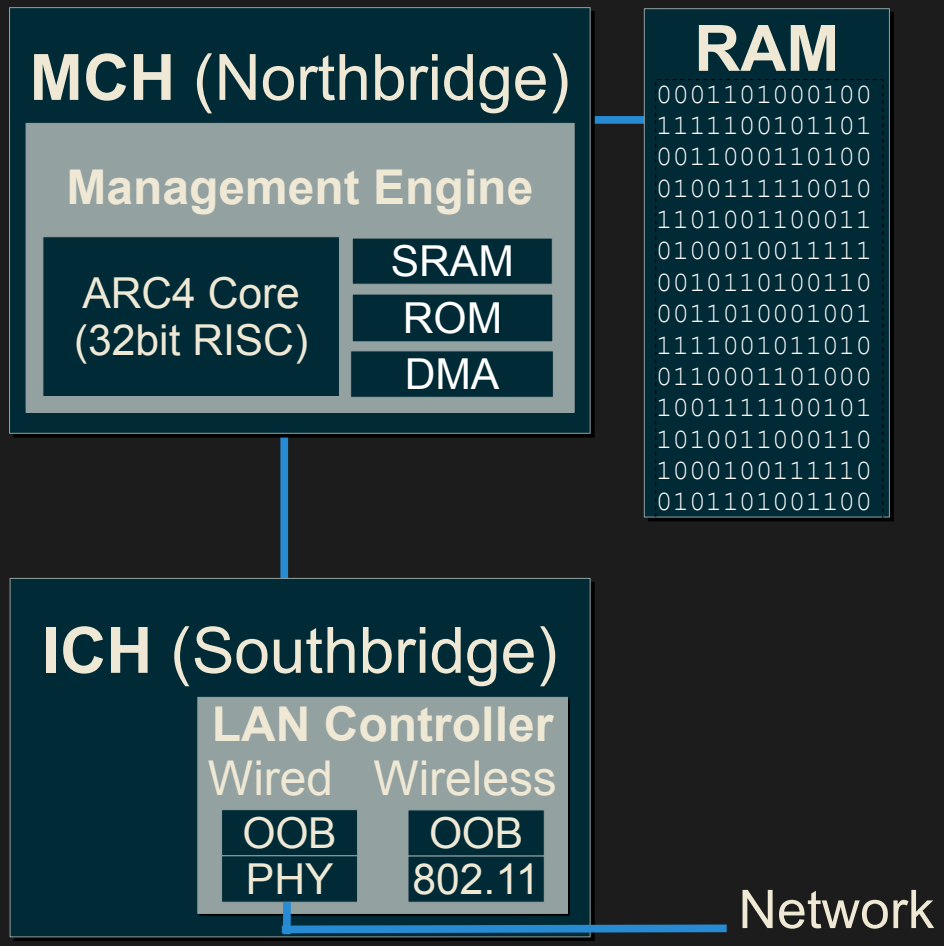

DMA BASED KEYSTROKE LOGGER

# [patrickx@44con:~$] cat 'What is DAGGER?'

- $ Written in C / ARC4 assembly
- $ Part of academic research project
- $ Not only a keylogger anymore
- $ Access to host memory (DMA read/write)
- $ Isolated network channel
- $ 32bit/64bit based attack targets
- $ ...

# [patrickx@44con:~$] cat 'Our Attack Environment'

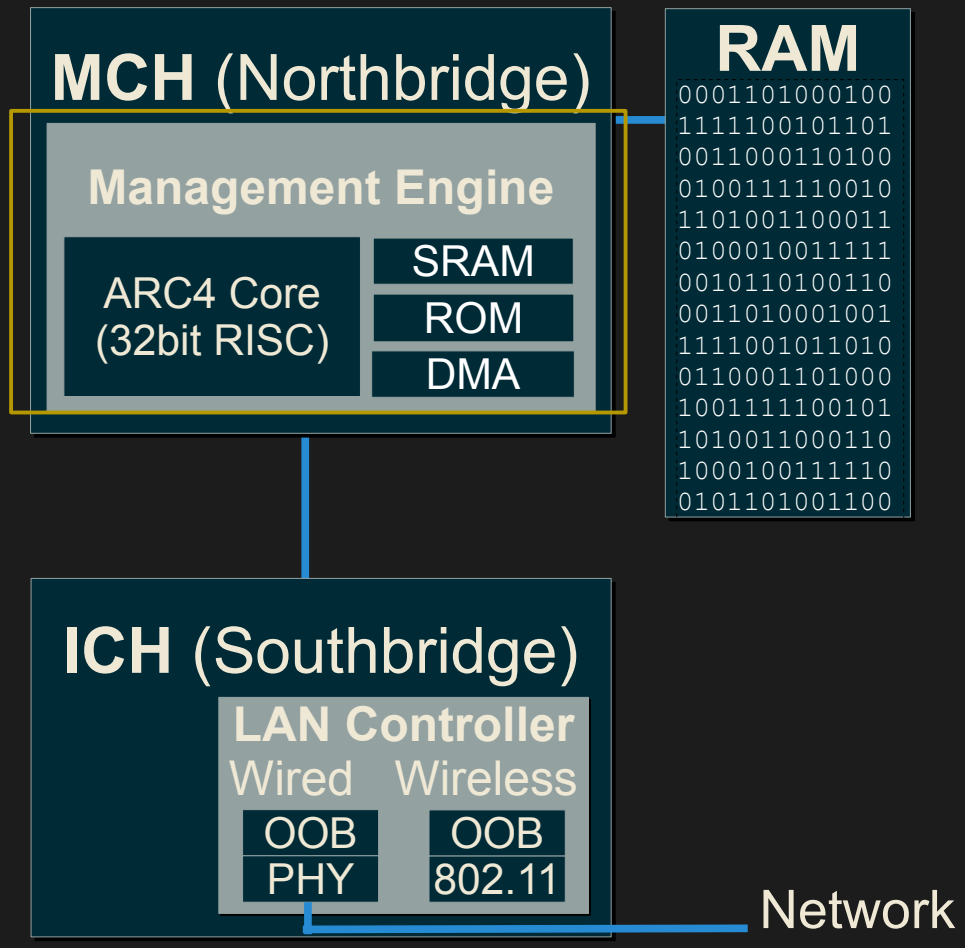
## $ Manageability Engine




*(Q35 Chipset)*

# [patrickx@44con:~$] cat 'Our Attack Environment'

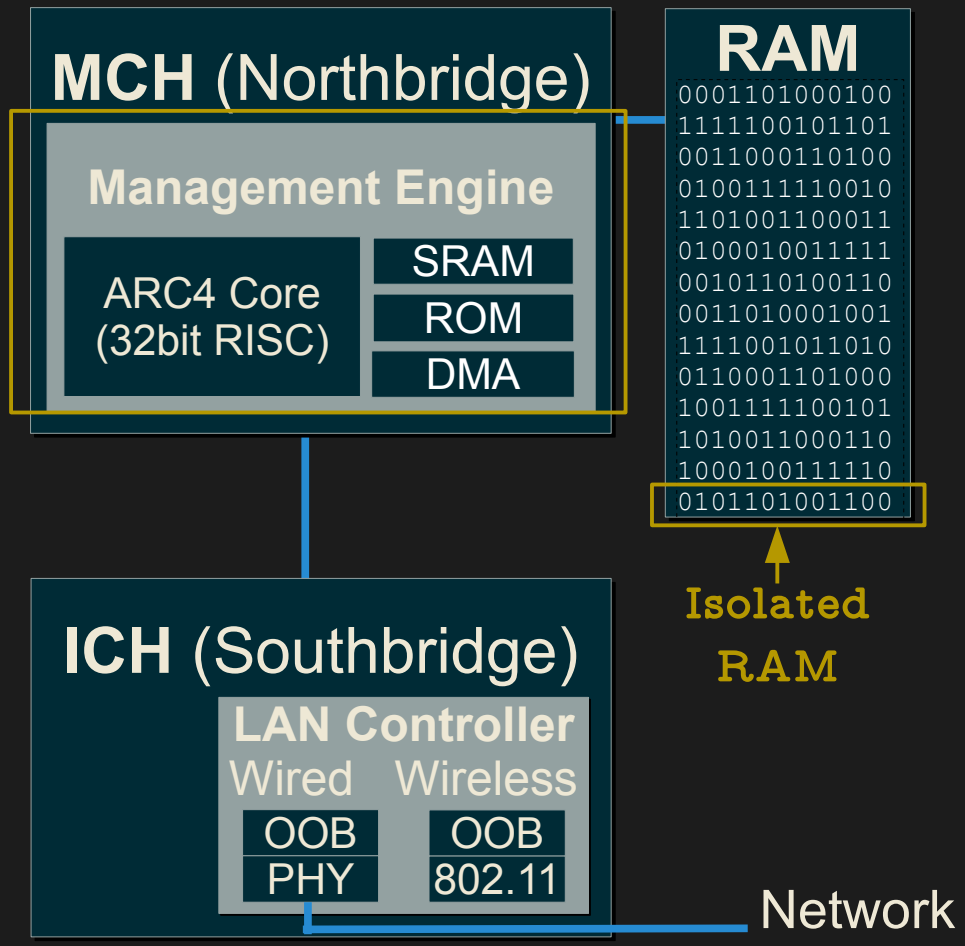
## $ Manageability Engine




*(Q35 Chipset)*

# [patrickx@44con:~$] cat 'Our Attack Environment'

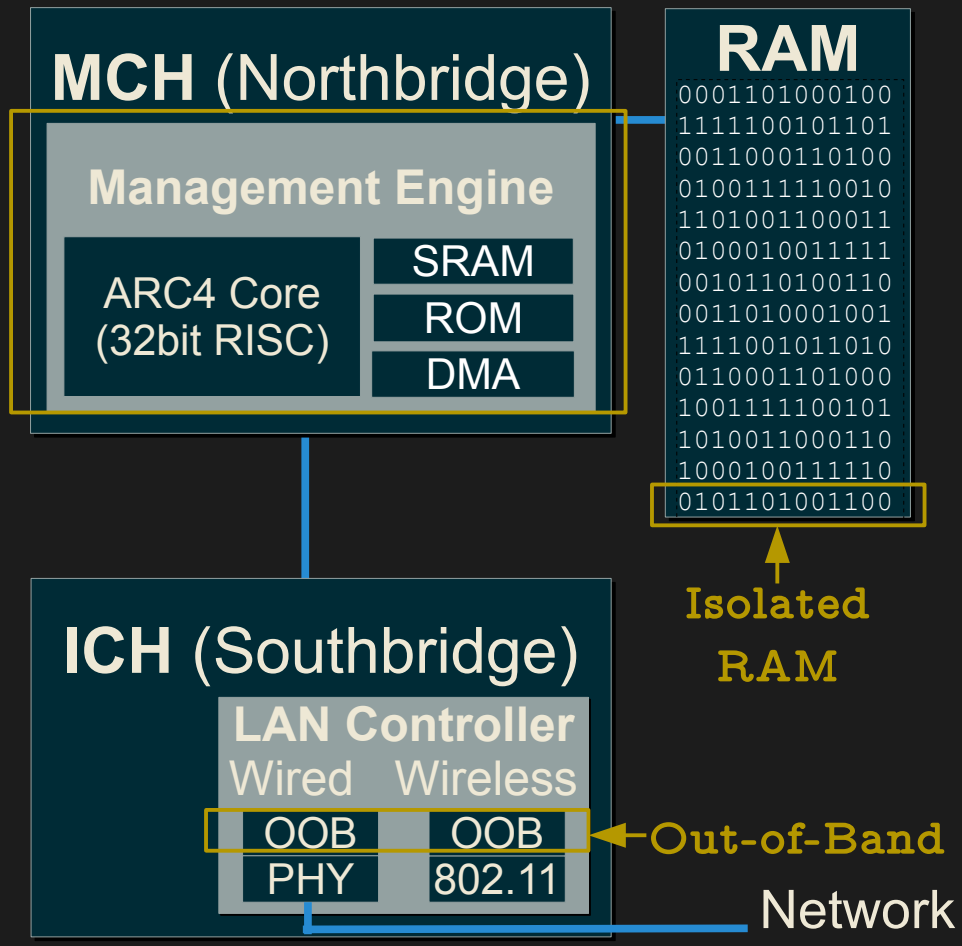
## $ Manageability Engine




*(Q35 Chipset)*

# [patrickx@44con:~$] cat 'Our Attack Environment'

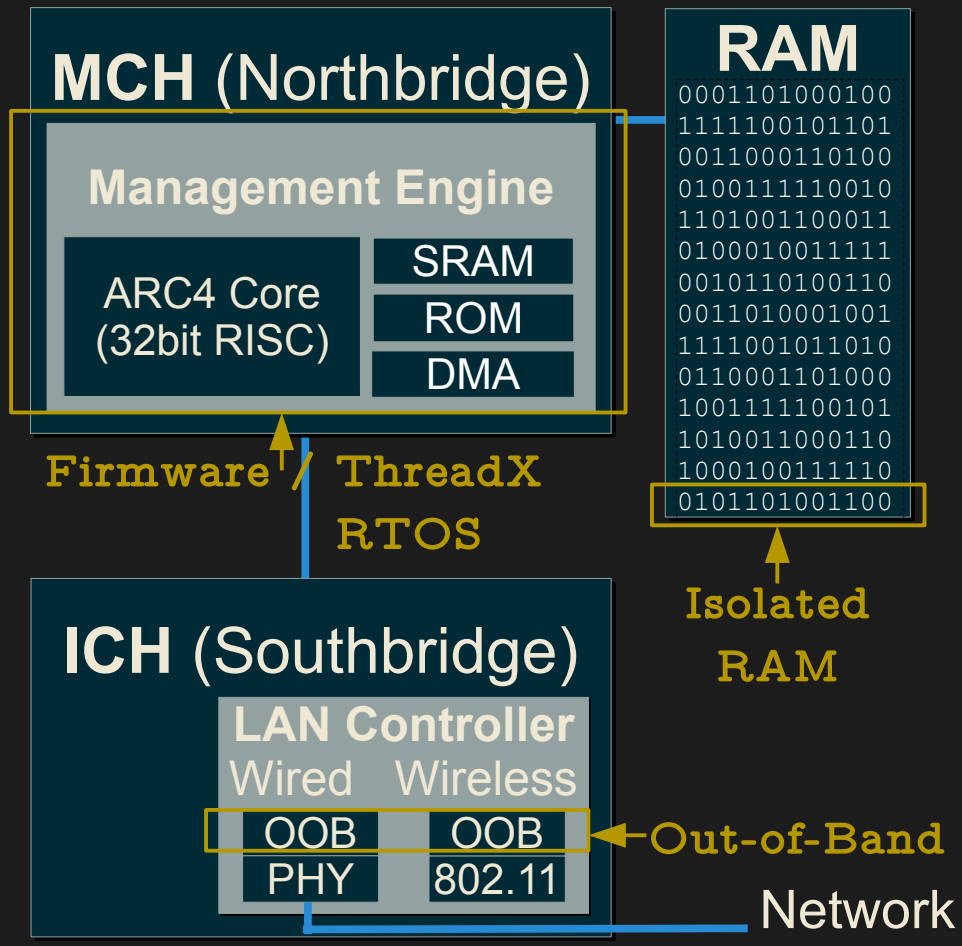
## $ Manageability Engine




*(Q35 Chipset)*

# [patrickx@44con:~$] cat 'Our Attack Environment'

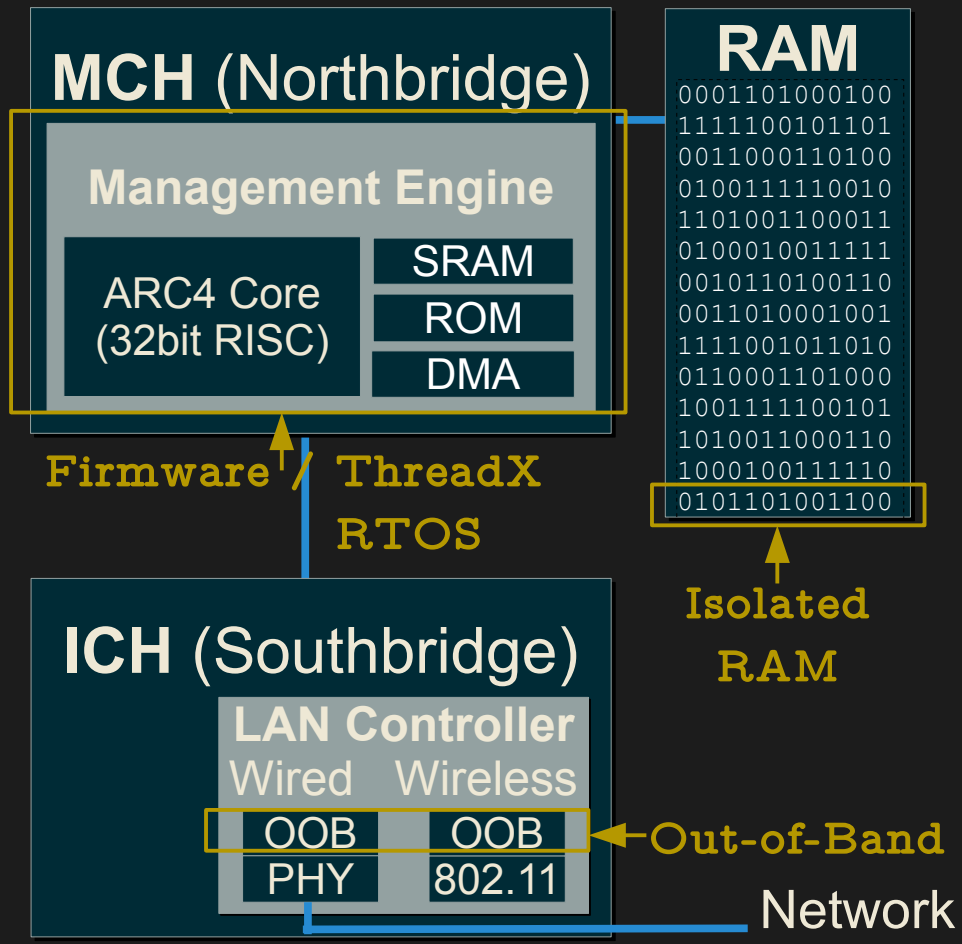
## $ Manageability Engine

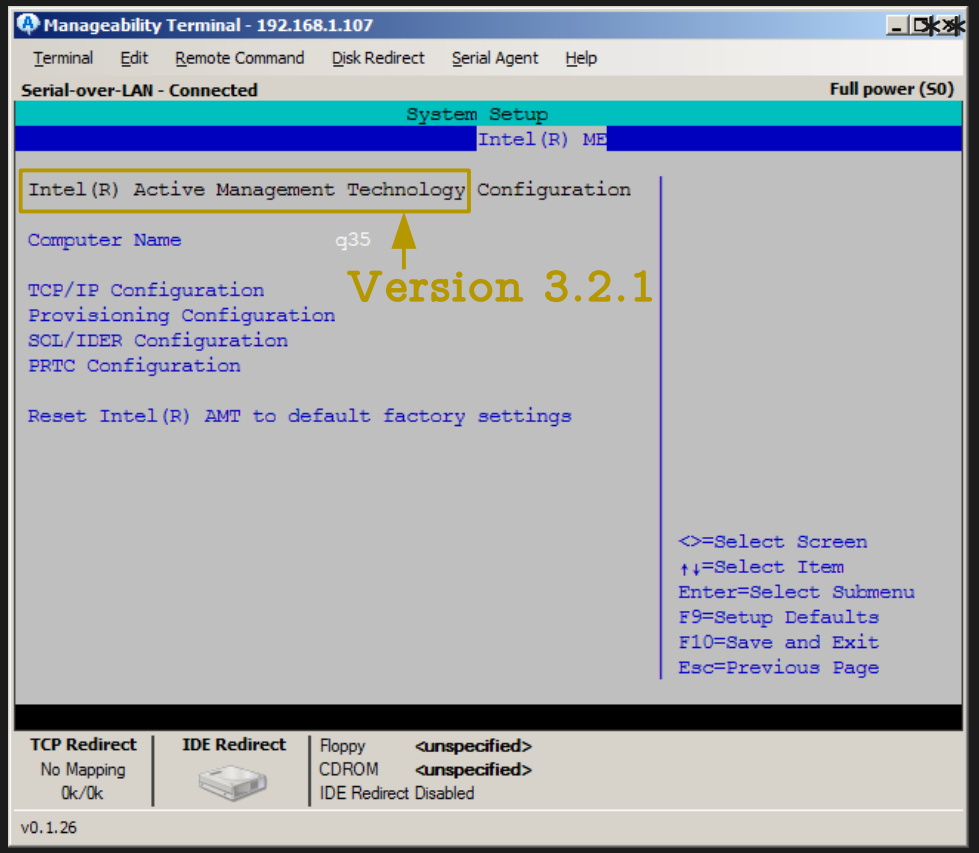



*(Q35 Chipset)*

# [patrickx@44con:~$] cat 'Our Attack Environment'

## $ Manageability Engine

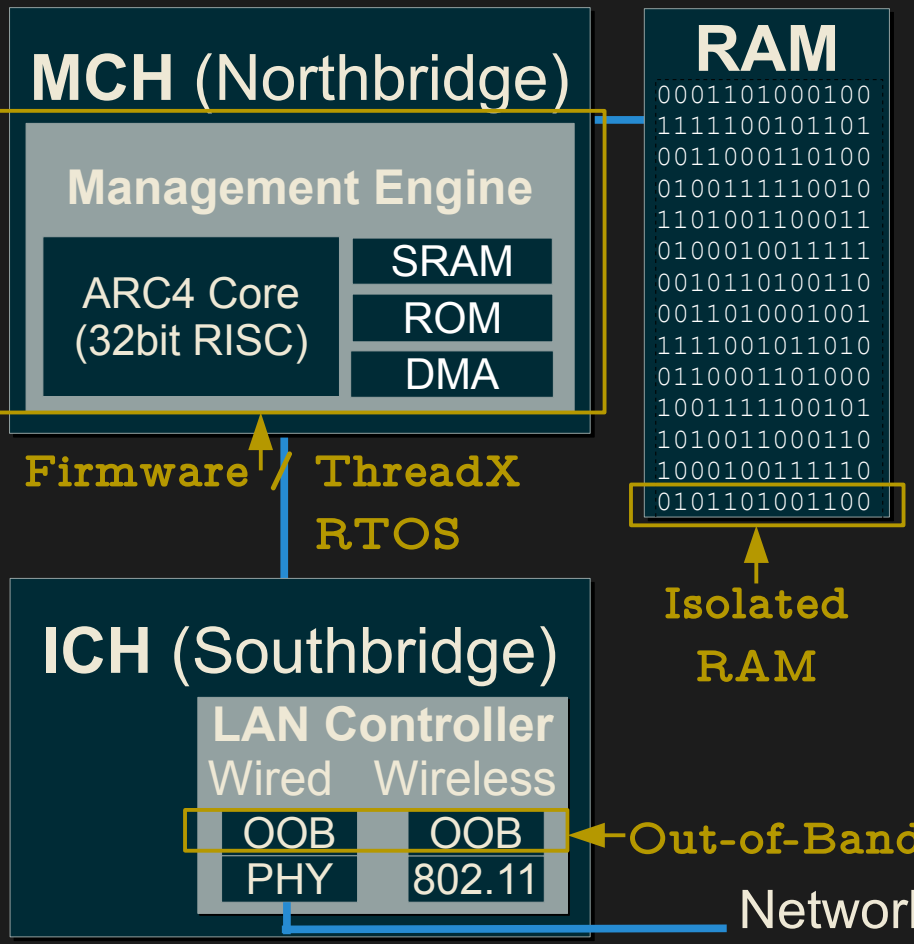


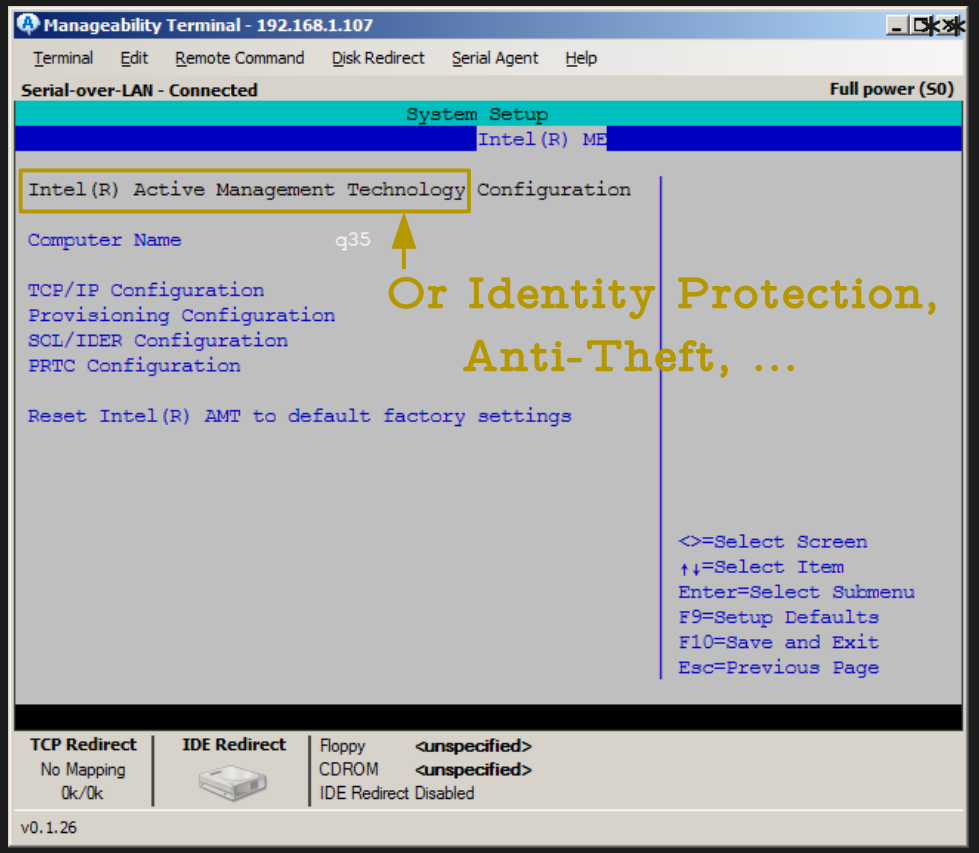




*(Q35 Chipset)*

# [patrickx@44con:~$] cat 'Our Attack Environment'

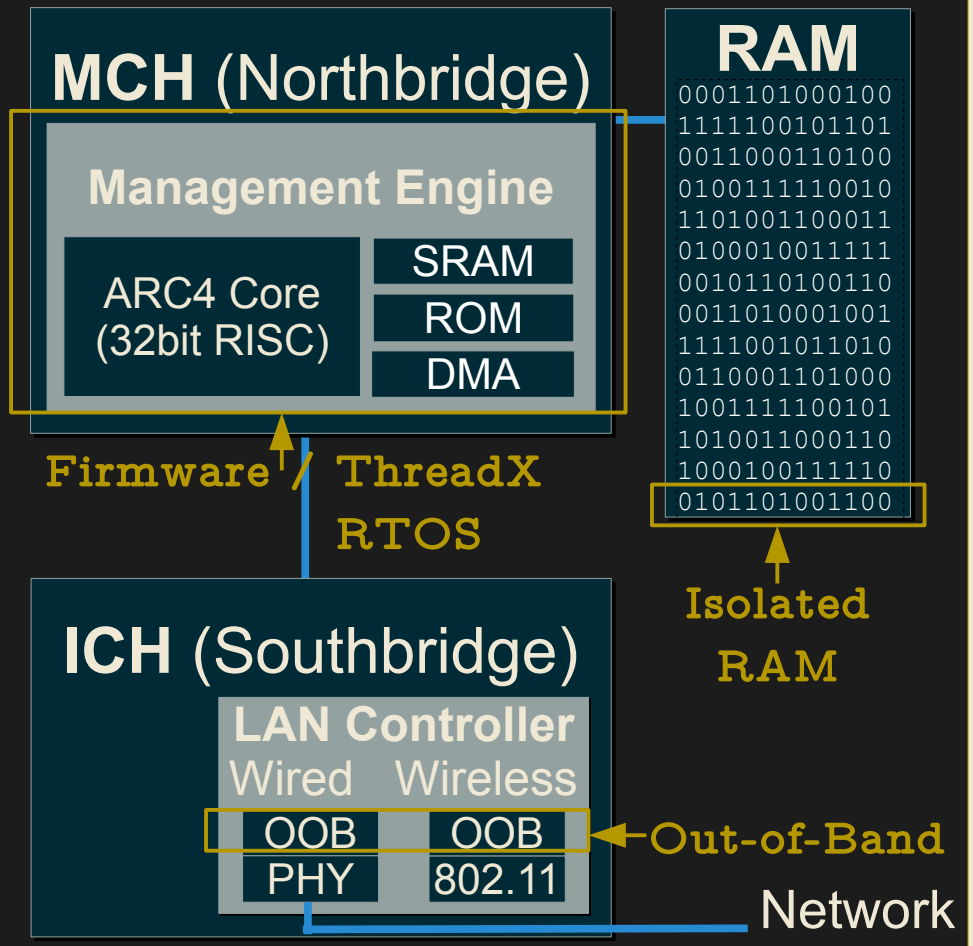
## $ Manageability Engine




*(Q35 Chipset)*

## $ Manageability Engine




*(Q35 Chipset)*

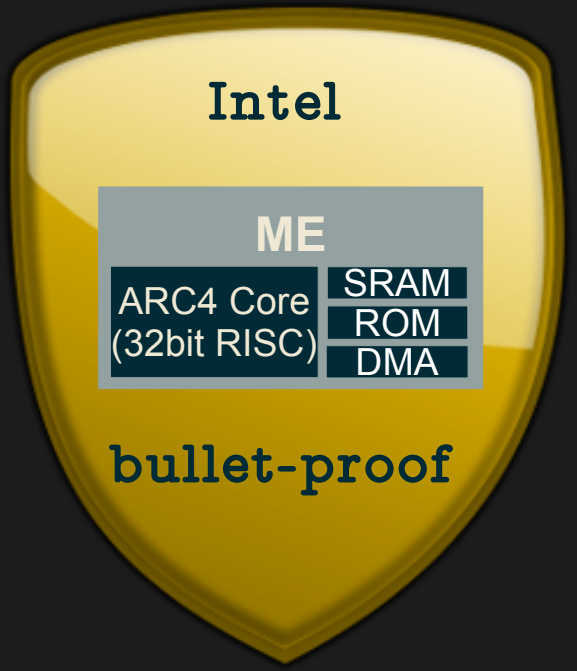
### ARC Historical Overview*

#Mathematical, Argonaut, Rotation & I/O:
MARIO chip :)
#SuperFX
#ARC
#1st ME generation:
ARCTangent-A4/
ARC4
#2nd ME generation:
ARCTangent-A5/
ARCompact
→ see [Sko12] !



*Details: http://en.wikipedia.org/wiki/ARC_International
**http://en.wikipedia.org/wiki/File:MARIO_CHIP_1_Starwing.jpg (Artikbot, CC BY-SA 3.0)
***http://www.youtube.com/watch?v=k8dxLr_xVv4 [0:21:44]

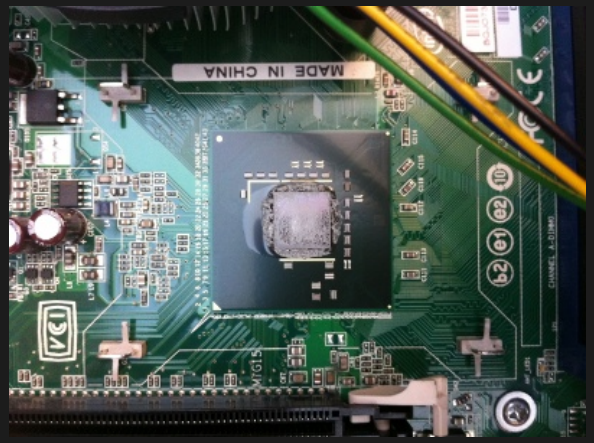
# [patrickx@44con:~$] cat 'Our Attack Environment'




$ Nonvolatile storage isolation
$ Signed firmware
$ Measured launch
$ Access control
$ …

→ DAGGER infiltration via memory remapping trick described in [Ter09] → Very good starting point!

# [patrickx@44con:~$] cat 'ME vs NIC'


- $ NIC could host DMA based keyloggers
- $ Unclear if NICs are just as well isolated from host (see [Duf11])

# [patrickx@44con:~$] cat 'Search for Valuable Data'

## $ Challenges

- $ Huge amount of memory
- $ Virtual vs. physical memory addresses
- $ No constant addresses for target structures

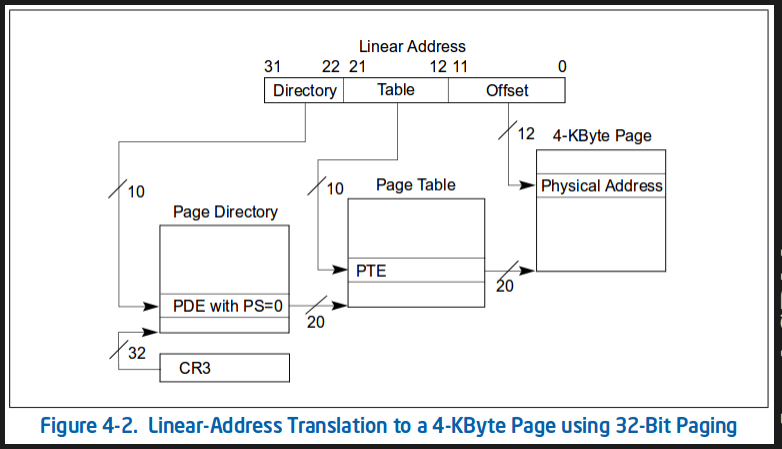
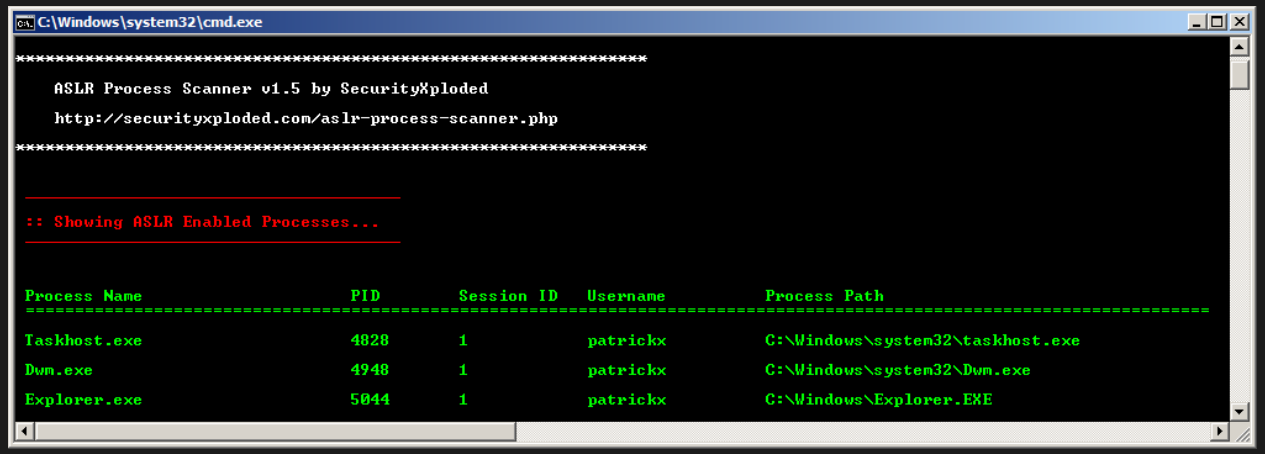



Figure 4-2. Linear-Address Translation to a 4-KByte Page using 32-Bit Paging

Intel SDM



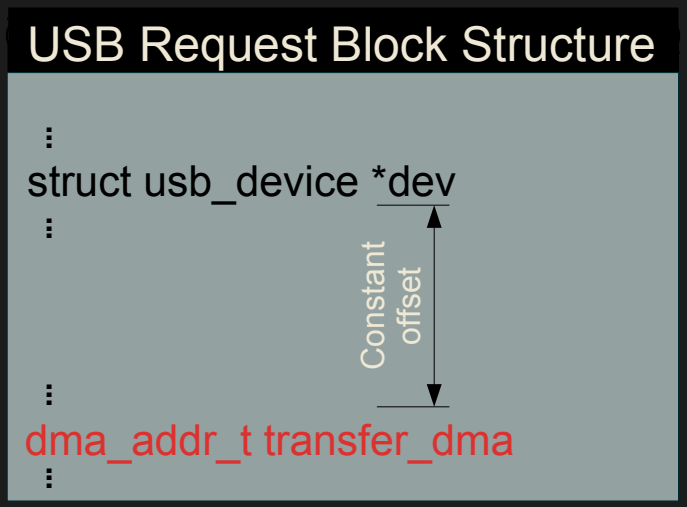
# [patrickx@44con:~$] cat 'Searching for Keystrokes'

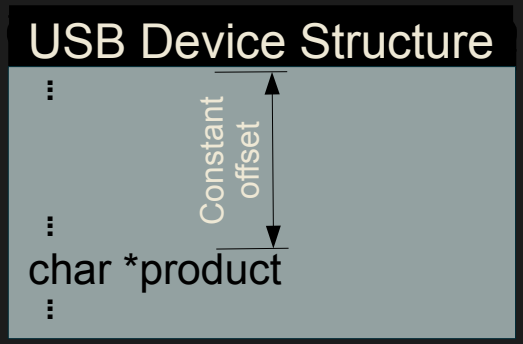
# [patrickx@44con:~$] cat 'Searching for Keystrokes'

# [patrickx@44con:~$] cat 'Linux Target'

$ Kernels tested: 2.6.32/3.0.9(32bit) / 3.5.0(64bit)

$ Signature scan:

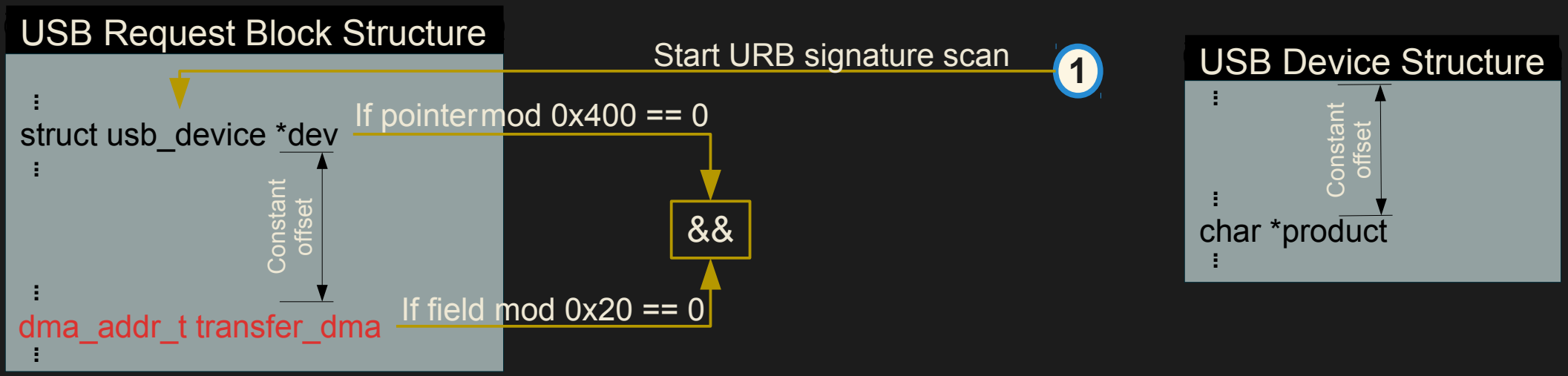


# [patrickx@44con:~$] cat 'Linux Target'

$ Kernels tested: 2.6.32/3.0.9(32bit) / 3.5.0(64bit)

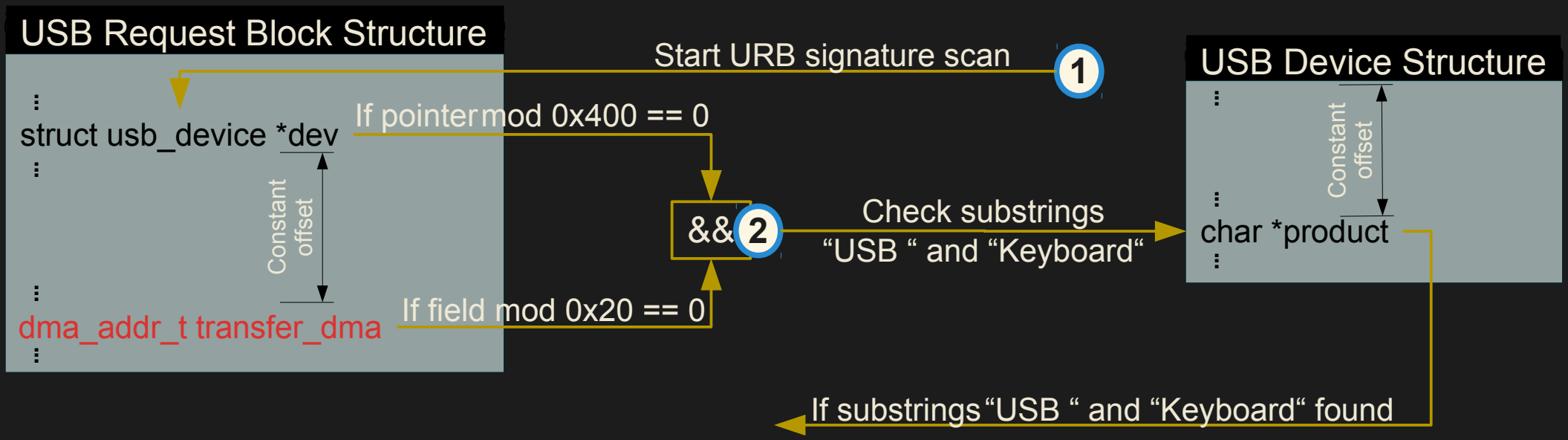
$ Signature scan:



# [patrickx@44con:~$] cat 'Linux Target'

$ Kernels tested: 2.6.32/3.0.9(32bit) / 3.5.0(64bit)

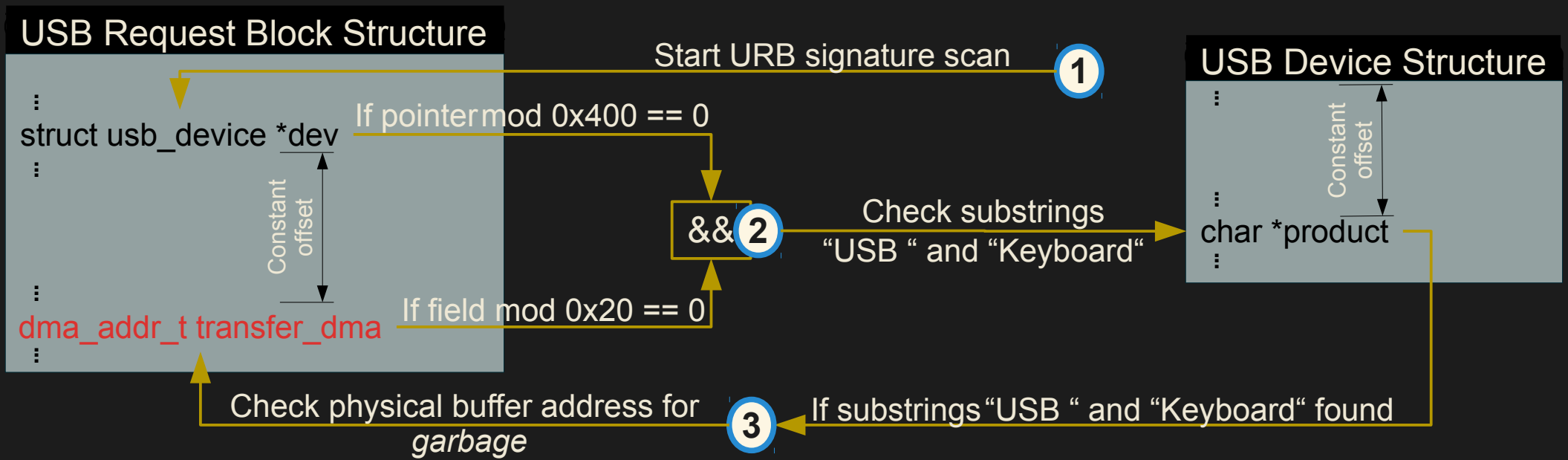
$ Signature scan:



# [patrickx@44con:~$] cat 'Linux Target'

$ Kernels tested: 2.6.32/3.0.9(32bit) / 3.5.0(64bit)

$ Signature scan:



# [patrickx@44con:~$] cat 'Linux Target'

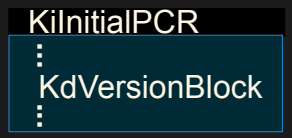
- $ **Mapping virtual to physical memory addresses**
  - $ 32bit: subtract constant offset → 0xc0000000
  - $ 64bit: see Documentation/x86/x86_64/mm.txt

# [patrickx@44con:~$] cat 'Windows Target'

- **$ Kernels tested: Vista / 7**
- **$ CR3 value required**
  (Verified within DAGGER/DAGGER traverses page tables)
- **$ No source code: IDA Pro, WinDbg, debug symbols**
- **$ Search path via Object Manager Namespace Directory:**

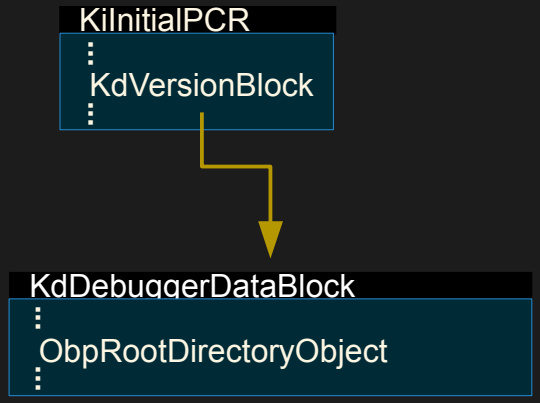
# [patrickx@44con:~$] cat 'Windows Target'

- **$ Kernels tested: Vista / 7**
- **$ CR3 value required**
  (Verified within DAGGER/DAGGER traverses page tables)
- **$ No source code: IDA Pro, WinDbg, debug symbols**
- **$ Search path via Object Manager Namespace Directory:**



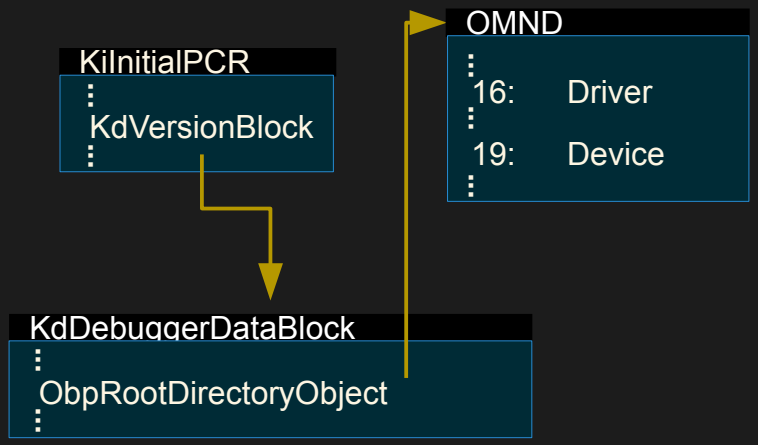
# [patrickx@44con:~$] cat 'Windows Target'

- $ Kernels tested: Vista / 7
- $ CR3 value required
  (Verified within DAGGER/DAGGER traverses page tables)
- $ No source code: IDA Pro, WinDbg, debug symbols
- $ Search path via Object Manager Namespace Directory:



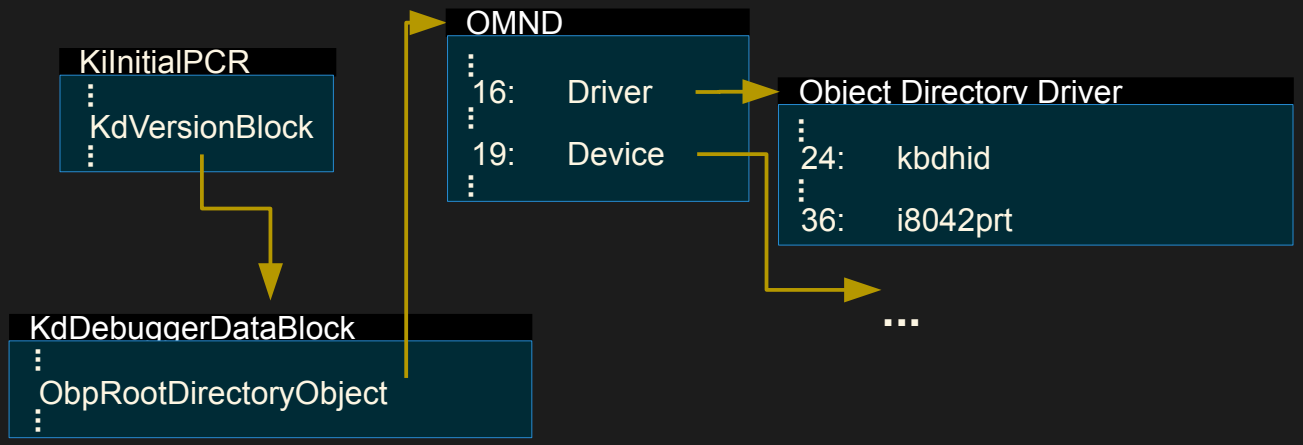
# [patrickx@44con:~$] cat 'Windows Target'

- $ Kernels tested: Vista / 7
- $ CR3 value required
  (Verified within DAGGER/DAGGER traverses page tables)
- $ No source code: IDA Pro, WinDbg, debug symbols
- $ Search path via Object Manager Namespace Directory:



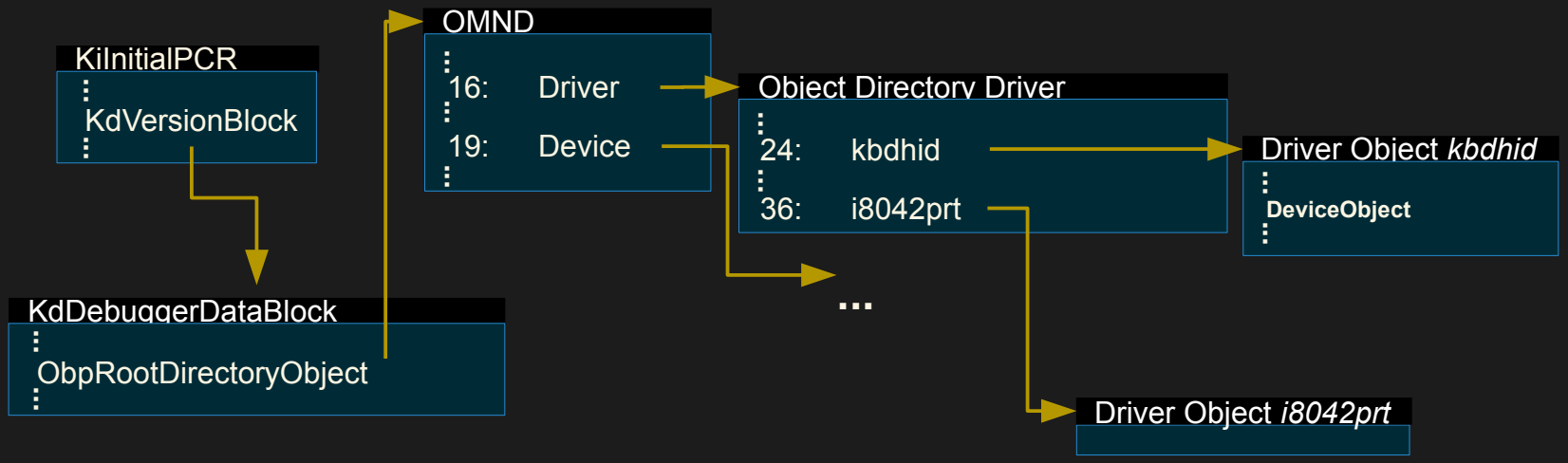
# [patrickx@44con:~$] cat 'Windows Target'

- $ Kernels tested: Vista / 7
- $ CR3 value required
  (Verified within DAGGER/DAGGER traverses page tables)
- $ No source code: IDA Pro, WinDbg, debug symbols
- $ Search path via Object Manager Namespace Directory:



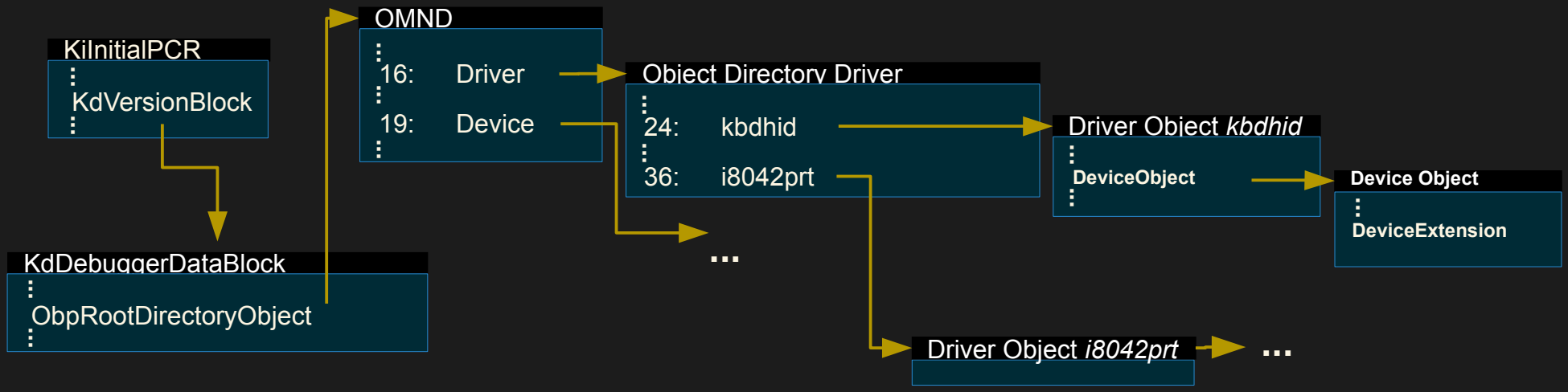
# [patrickx@44con:~$] cat 'Windows Target'

- $ Kernels tested: Vista / 7
- $ CR3 value required
  (Verified within DAGGER/DAGGER traverses page tables)
- $ No source code: IDA Pro, WinDbg, debug symbols
- $ Search path via Object Manager Namespace Directory:



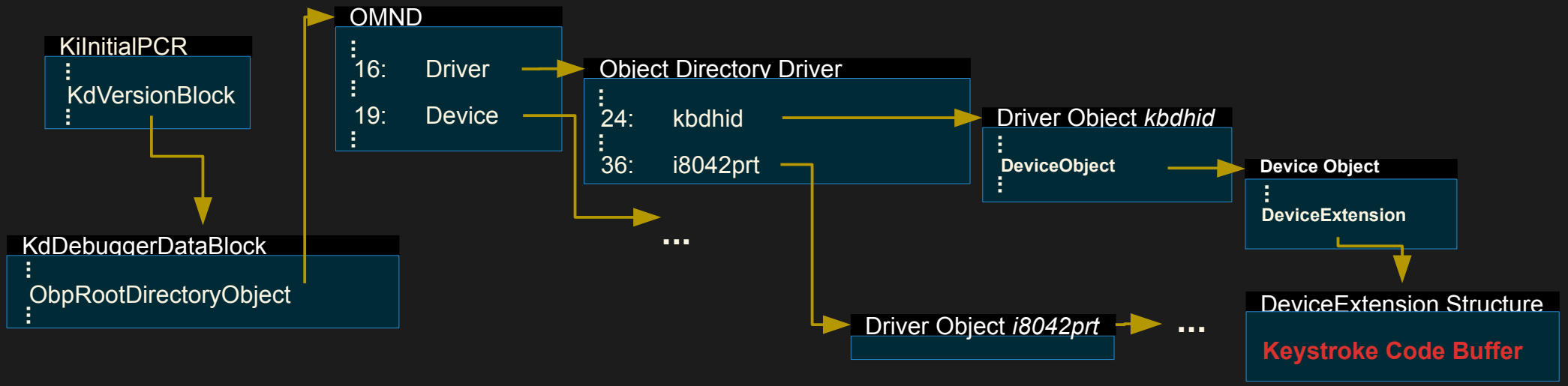
# [patrickx@44con:~$] cat 'Windows Target'

- $ Kernels tested: Vista / 7
- $ CR3 value required
  (Verified within DAGGER/DAGGER traverses page tables)
- $ No source code: IDA Pro, WinDbg, debug symbols
- $ Search path via Object Manager Namespace Directory:

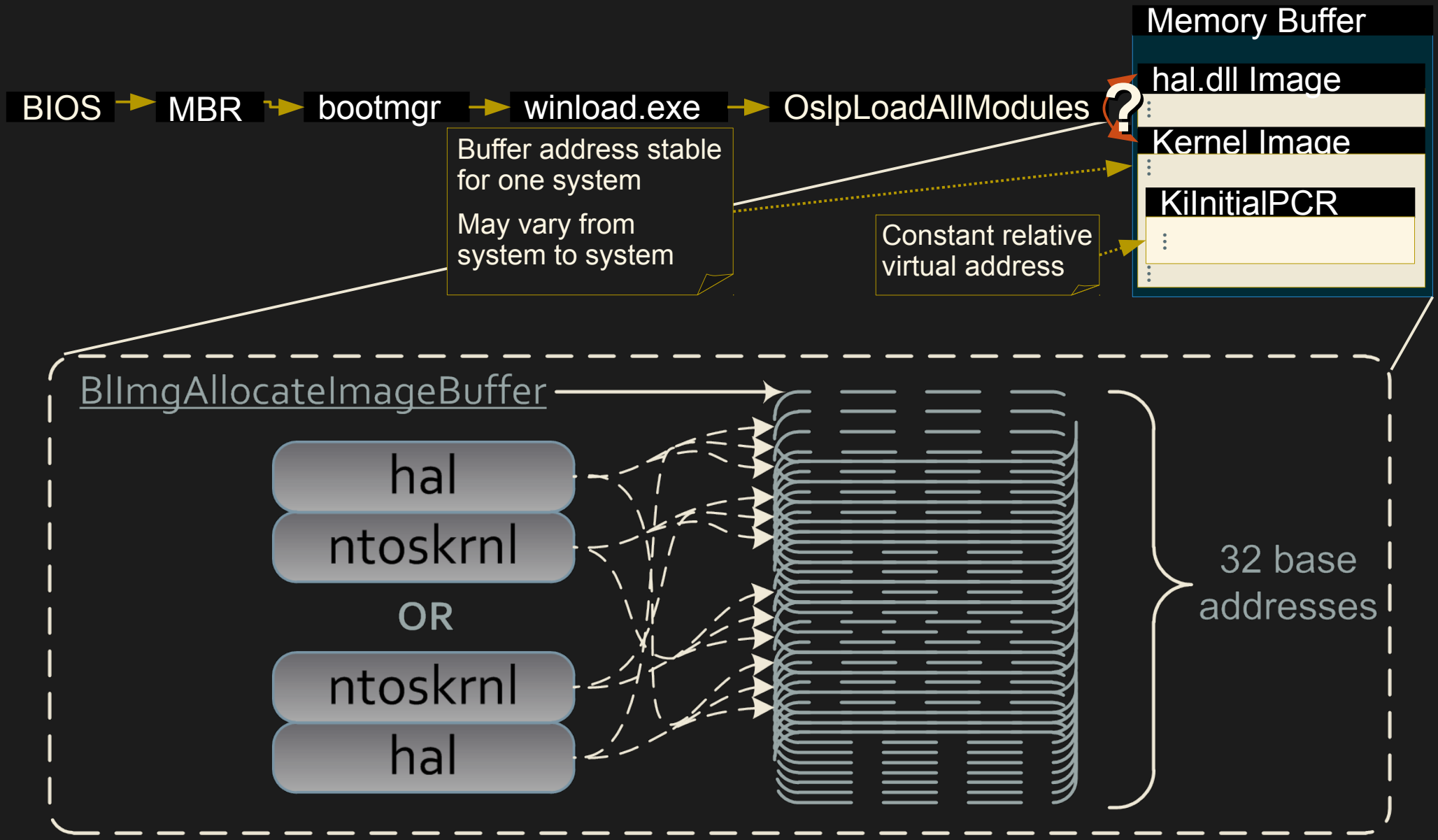


# [patrickx@44con:~$] cat 'Windows Target'

- $ Kernels tested: Vista / 7
- $ CR3 value required
  (Verified within DAGGER/DAGGER traverses page tables)
- $ No source code: IDA Pro, WinDbg, debug symbols
- $ Search path via Object Manager Namespace Directory:



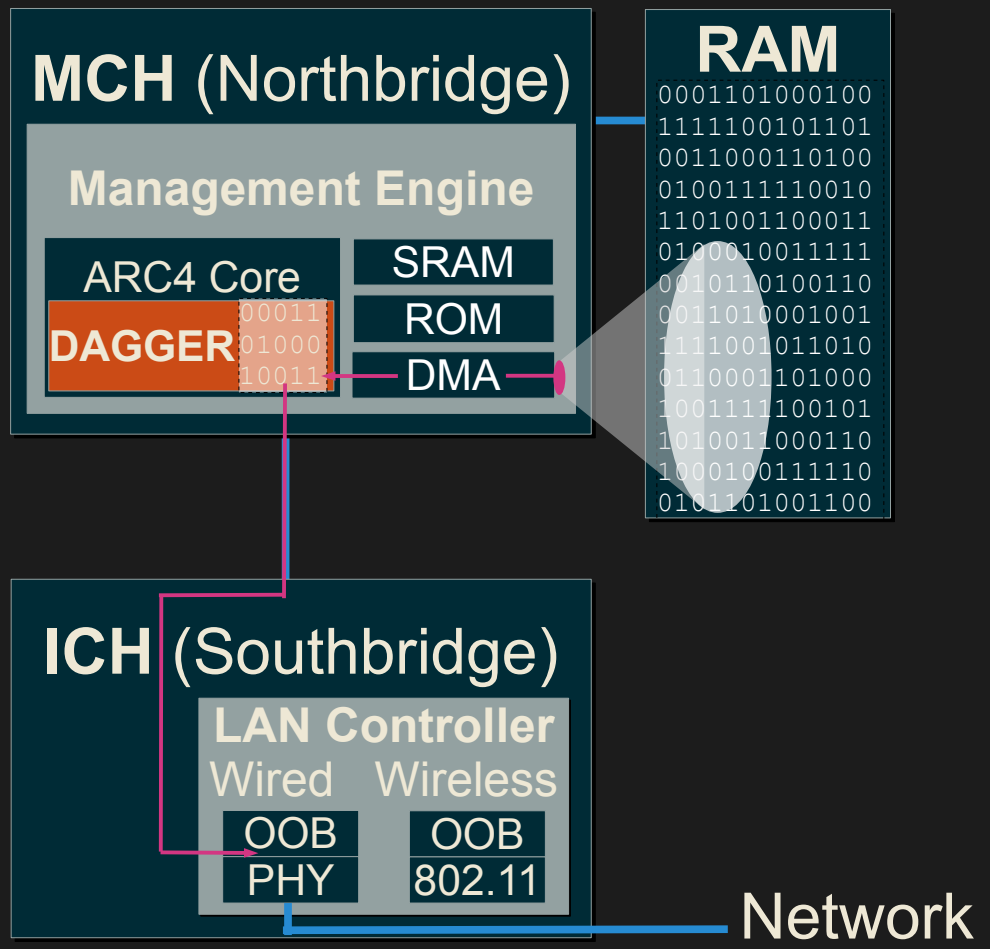
# [patrickx@44con:~$] cat 'Address Randomization'

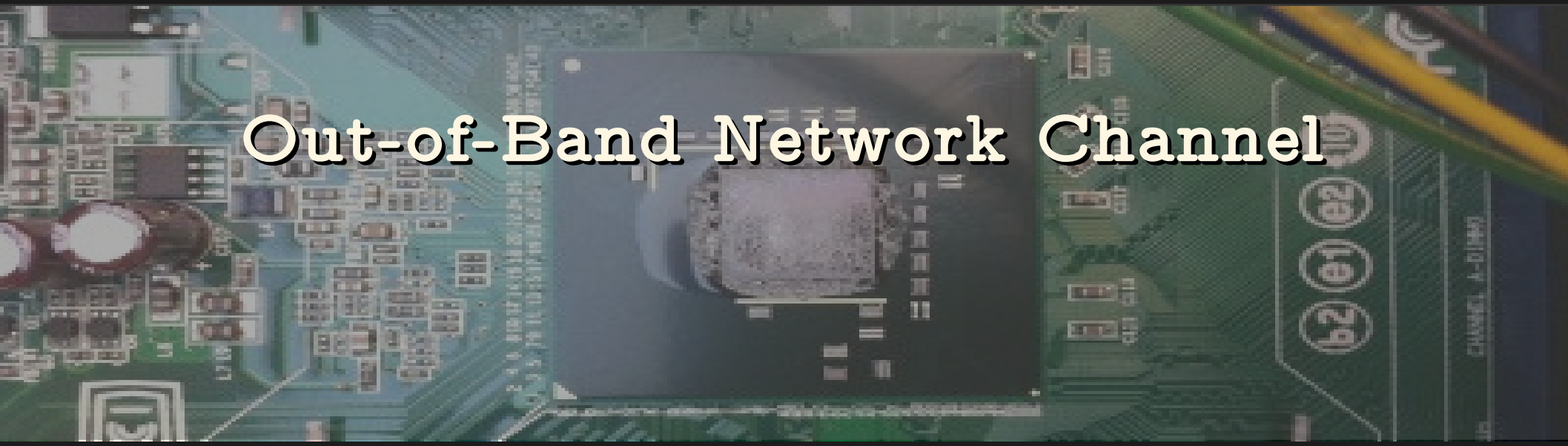


# [patrickx@44con:~$] cat 'Required ME Features'

$ **DMA read access**
→ easy
(we just changed
two bits)

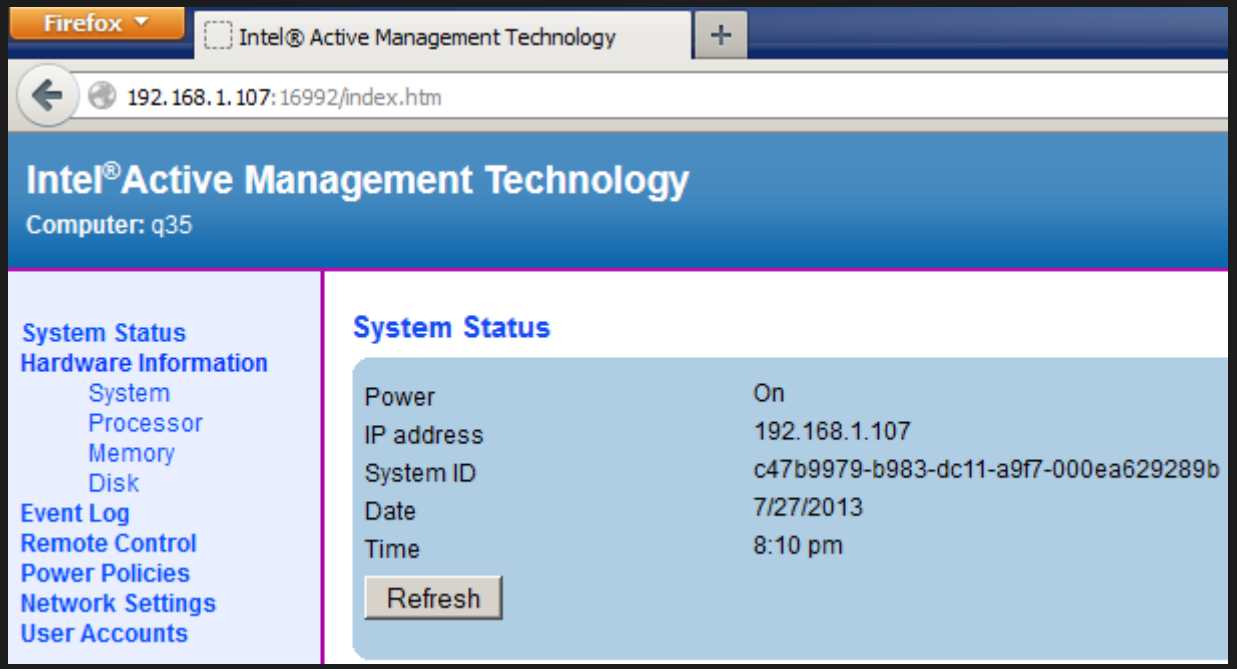
$ **Stealthy network channel**
→ challenging
(more than
two bits :) )




Out-of-Band Network Channel

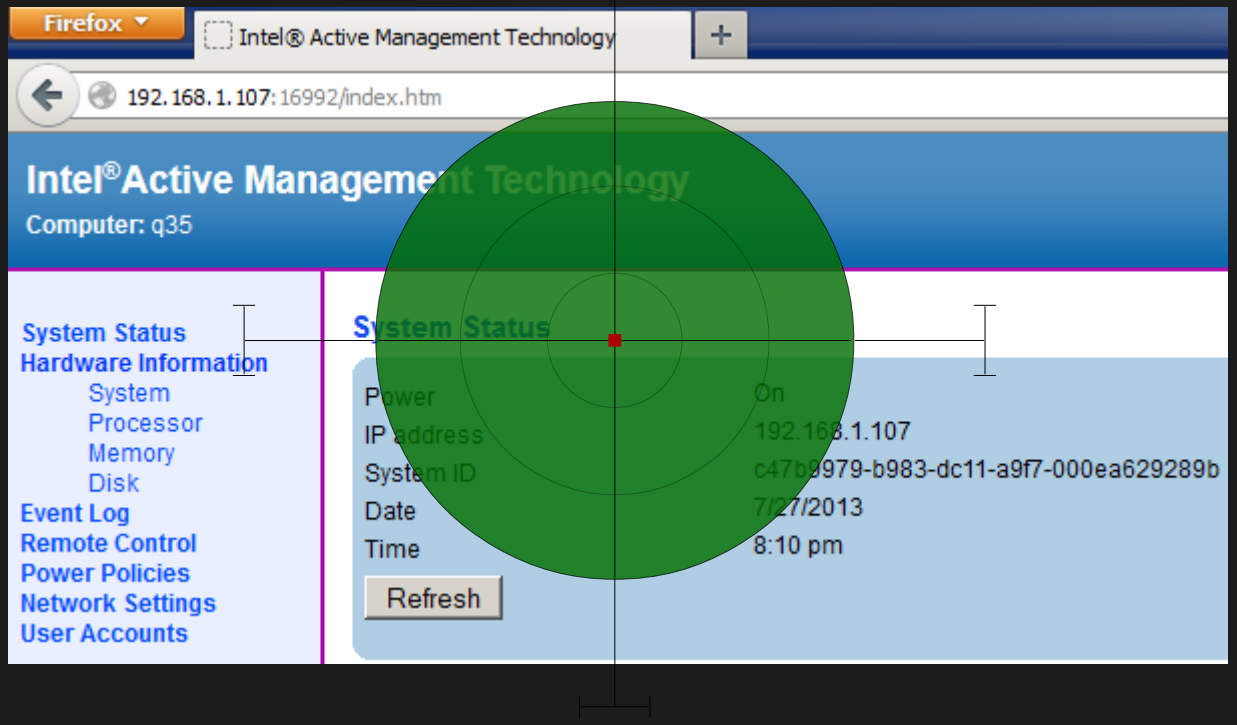
# [patrickx@44con:~$] cat 'Target: ME OOB'

$ Needed not only to exfiltrate captured keystroke codes, but also to download new attack code!



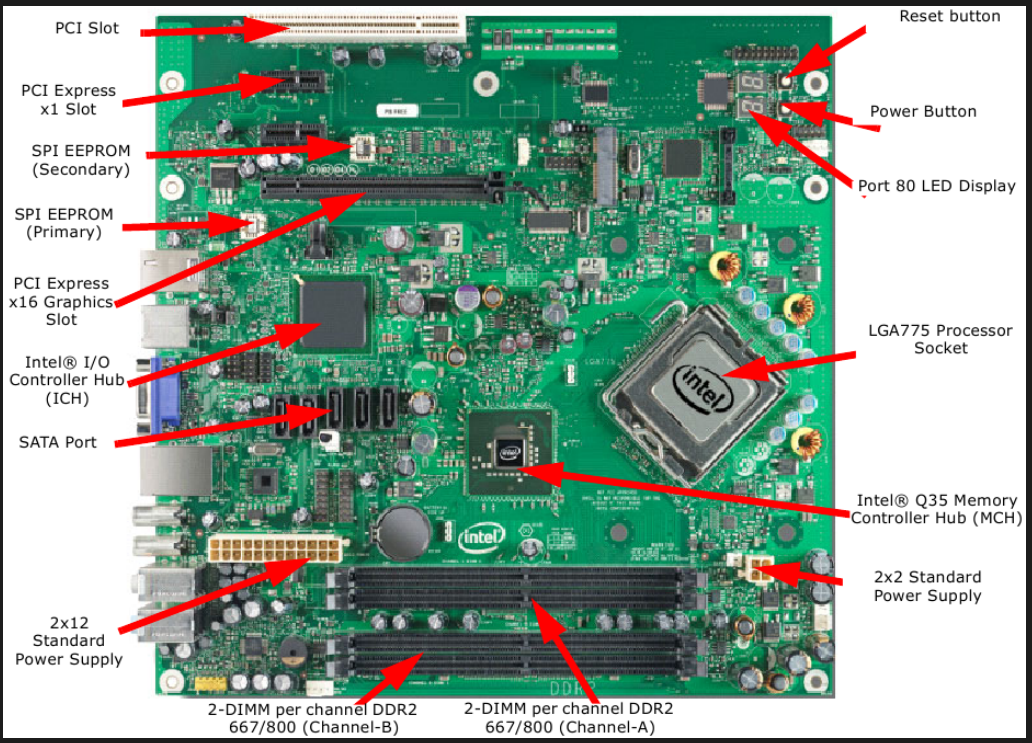
# [patrickx@44con:~$] cat 'Target: ME OOB'

$ Needed not only to exfiltrate captured keystroke codes, but also to download new attack code!

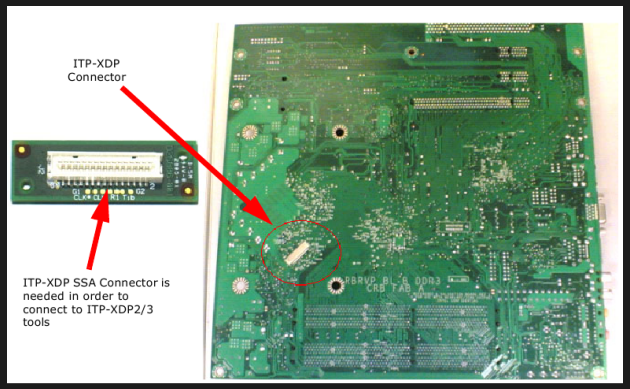



**How to find firmware code responsible for webserver replies?**

# [patrickx@44con:~$] cat 'Some Tools Required'




**Board Features ([Int07], p.11)**




**ITP-XDP Connector location (J2BC) ([Int07], p.20)**

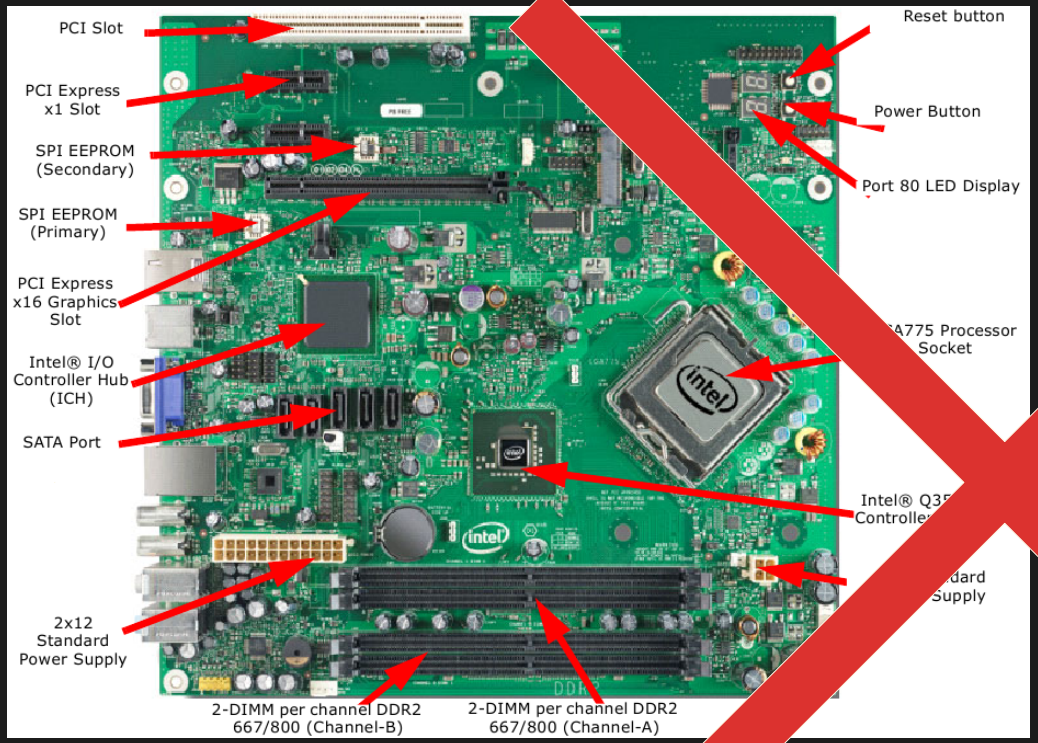
- Programming DMA hardware over JTAG port in debugger
- DMA-ing 64 bytes from system memory containing malicious VMExit handler code to internal chipset memory

```
arc> dump 0x2000000
02000000: 00000000 00000000 00000000 00000000 | ................
02000010: 00000000 00000000 00000000 00000000 | ................
02000020: 00000000 00000000 00000000 00000000 | ................
02000030: 00000000 00000000 00000000 00000000 | ................
02000040: 00000000 00000000 00000000 00000000 | ................
02000050: 00000000 00000000 00000000 00000000 | ................
02000060: 00000000 00000000 00000000 00000000 | ................
02000070: 00000000 00000000 00000000 00000000 | ................
arc> arc:dma(1,0x73000,0,0x2000000,64)
Transferred 64 B of data from Host 0x00073000
General Status = 1
02000000: fa 55 8b ec 81 ec 84 00 00 00 89 45 b4 89 5d b8 | .U.........E..].
02000010: 89 4d bc 89 55 c0 89 75 cc 89 7d d0 0f 20 d0 89 | .M..U..u..}.. ..
02000020: 45 c4 8d 45 f8 50 68 02 44 00 00 e8 b8 08 00 00 | E..E.Ph.D.......
02000030: 8d 45 d4 50 68 1e 68 00 00 e8 aa 00 00 00 8d 45 | .E.Ph.h........E
02000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
02000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
02000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
02000070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
arc> dump 0x2000000
02000000: ec8b55fa 0084ec81 45890000 b85d89b4 | .U.........E..].
02000010: 89bc4d89 7589c055 d07d89cc 89d0200f | .M..U..u..}.. ..
```

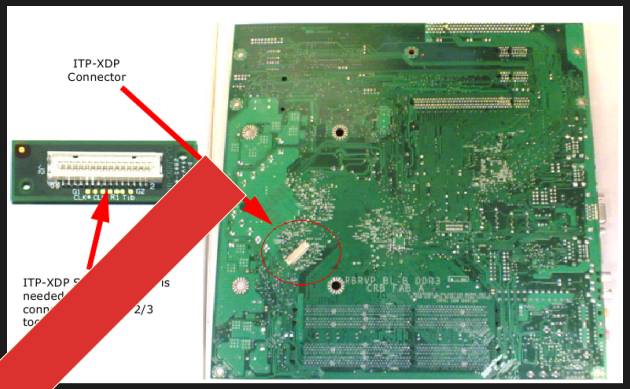
13 7/6/2008 

**Let's Program DMA Manually ([Bul08], p.13)**

# [patrickx@44con:~$] cat 'Some Tools Required'




**Board Features ([Int07] ...11)**




**ITP-XDP Connector location (J2BC) ([Int07], p.20)**

- Programming DMA hardware over JTAG port in debugger
- DMA-ing 64 bytes from system memory containing malicious VMExit handler code to internal chipset memory

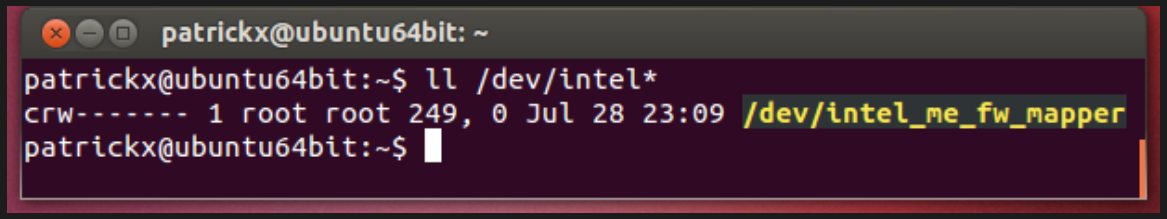
```
arc> dump 0x2000000
02000000: 00000000 00000000 00000000 00000000 | ................
02000010: 00000000 00000000 00000000 00000000 | ................
02000020: 00000000 00000000 00000000 00000000 | ................
02000030: 00000000 00000000 00000000 00000000 | ................
02000040: 00000000 00000000 00000000 00000000 | ................
02000050: 00000000 00000000 00000000 00000000 | ................
02000060: 00000000 00000000 00000000 00000000 | ................
02000070: 00000000 00000000 00000000 00000000 | ................
arc:dma(1,0x73000,0,0x2000000,64)
arred 64 B of data from Host 0x00073000
Status = 1
02000000: fa 55 8b ec 81 ec 84 00 00 00 89 45 b4 89 5d b8 | .U.........E..].
02000010: 89 4d bc 89 55 c0 89 75 cc 89 7d d0 0f 20 d0 89 | .M..U..u..}.. ..
02000020: c4 8d 45 f8 50 68 02 44 00 00 e8 b8 08 00 00 | E..E.Ph.D.......
02000030: d4 50 68 1e 68 00 00 e8 aa 00 00 00 8d 45 | .E.Ph.h.........E
02000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
02000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
02000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
02000070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
arc> dump 0x20
02000000: ec8b55 ec81 45890000 b85d89b4 | .U.........E..].
02000010: 89bc4d8 55 d07d89cc 89d0200f | .M..U..u..}.. ..
```

13 7/6/2008 Copyright © Int... 2006. All rights reserved. Third-party marks and brands are the pro... respective owners. All products, dates, and figures are prelimina... to change without notice.

**Let's Program DMA Manually ([Bul08], p.13)**

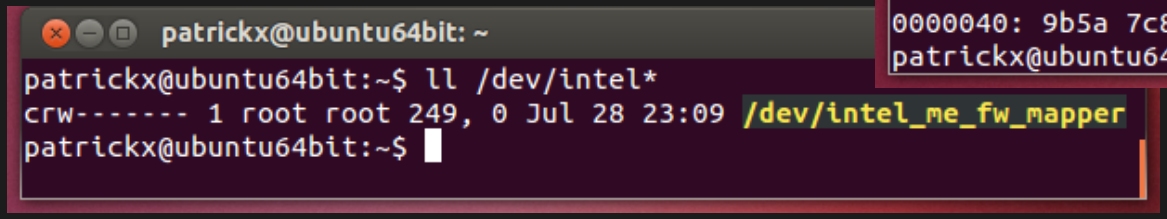
# [patrickx@44con:~$] cat 'Our Research Tools'

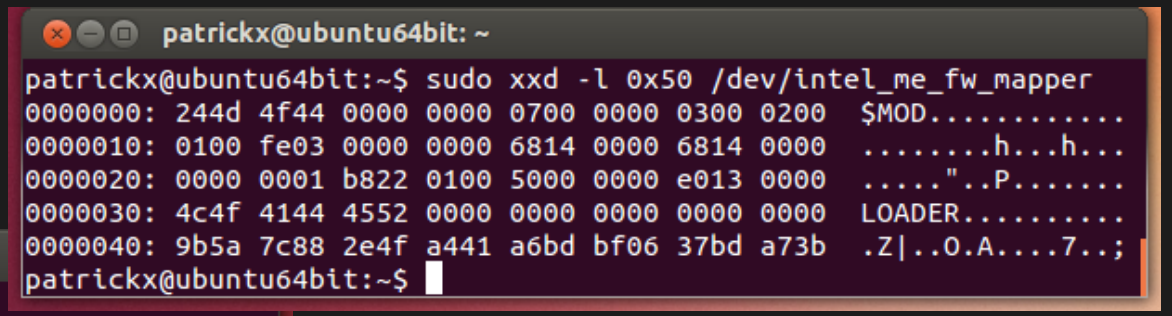
## $ Linux:



# [patrickx@44con:~$] cat 'Our Research Tools'

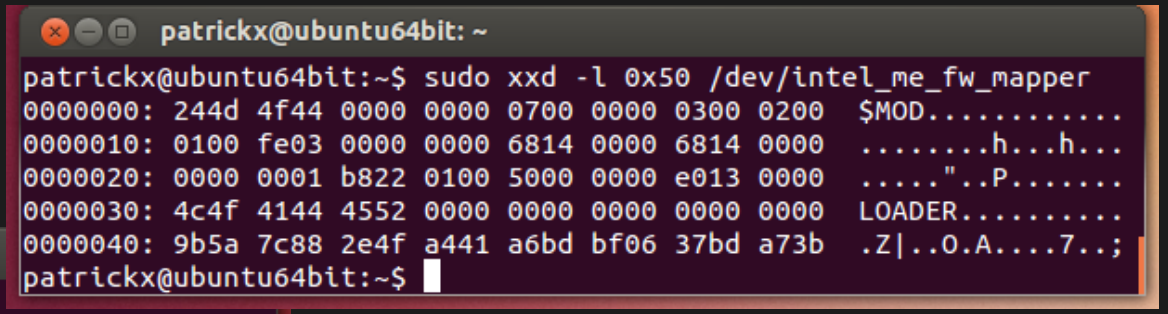
## $ Linux:

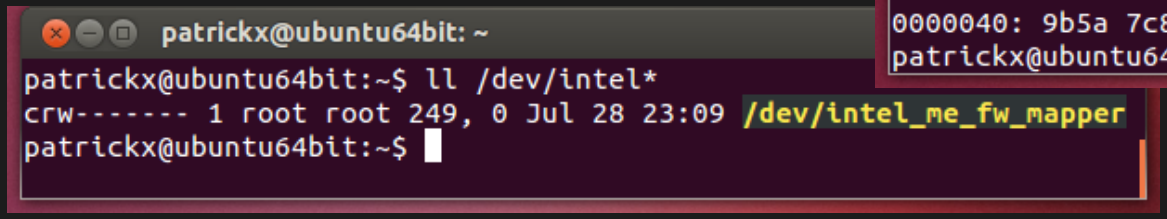


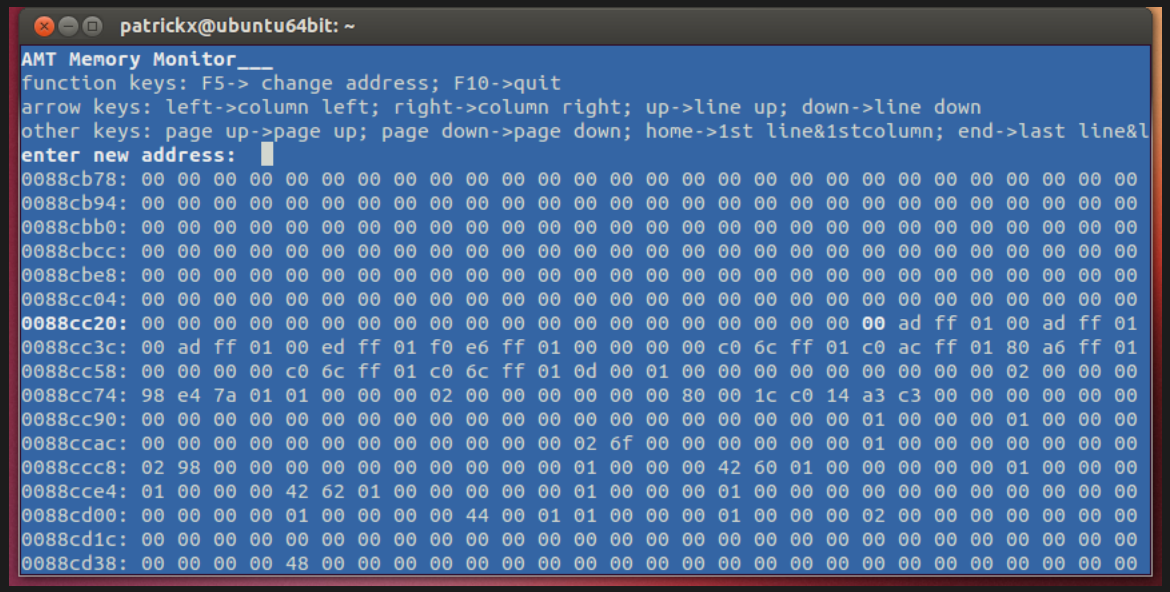



# [patrickx@44con:~$] cat 'Our Research Tools'

## $ Linux:



# [patrickx@44con:~$] cat 'Our Research Tools'

## $ Linux:

```
patrickx@ubuntu64bit: ~
patrickx@ubuntu64bit:~$ sudo xxd -l 0x50 /dev/intel_me_fw_mapper
0000000: 244d 4f44 0000 0000 0700 0000 0300 0200  $MOD............
0000010: 0100 fe03 0000 0000 6814 0000 6814 0000  ........h...h...
0000020: 0000 0001 b822 0100 5000 0000 e013 0000  ....."..P.......
0000030: 4c4f 4144 4552 0000 0000 0000 0000 0000  LOADER..........
0000040: 9b5a 7c88 2e4f a441 a6bd bf06 37bd a73b  .Z|..O.A....7..;
patrickx@ubuntu64bit:~$
```

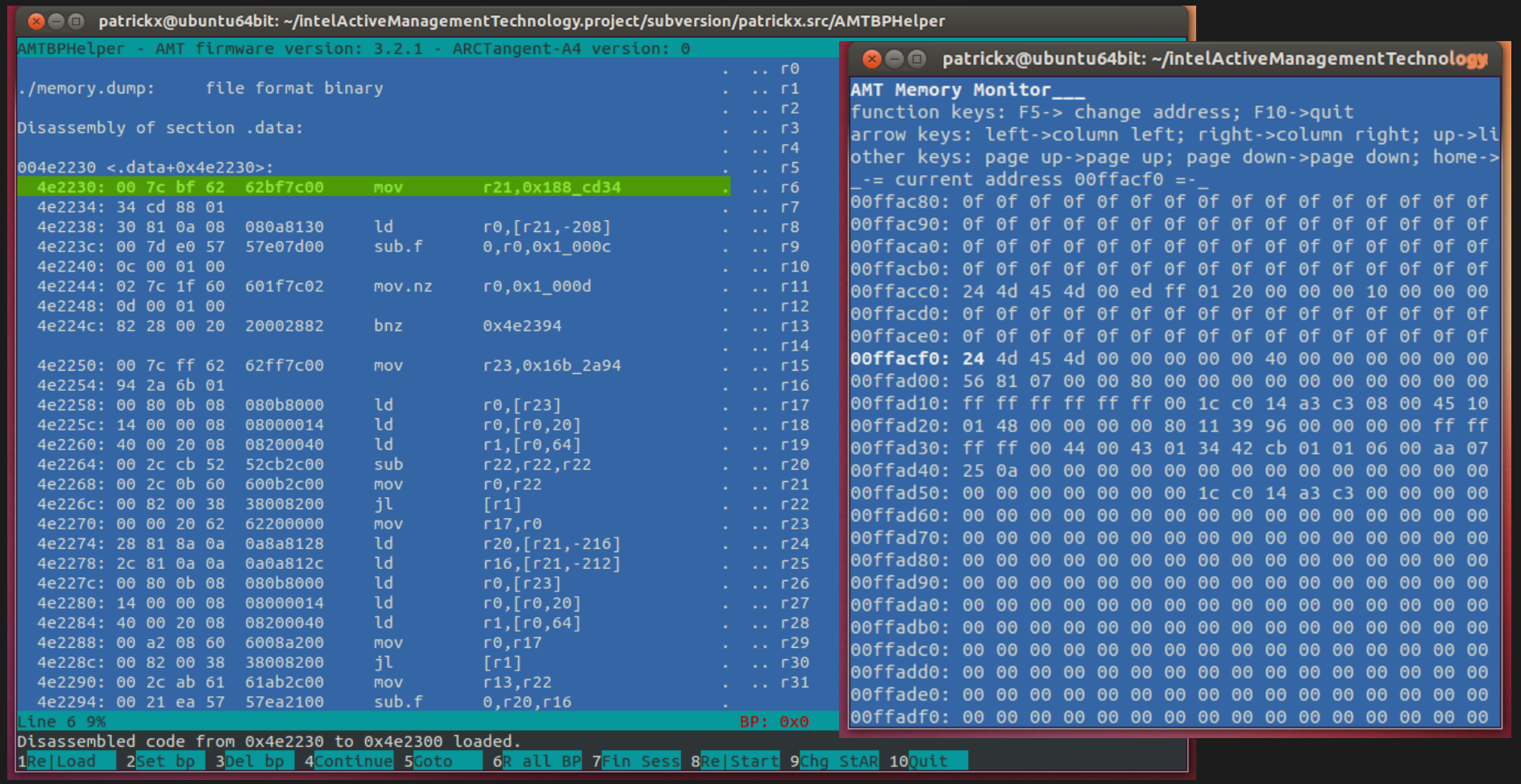
```
patrickx@ubuntu64bit: ~
patrickx@ubuntu64bit:~$ ll /dev/intel*
crw------- 1 root root 249, 0 Jul 28 23:09 /dev/intel_me_fw_mapper
patrickx@ubuntu64bit:~$
```

```
patrickx@ubuntu64bit: ~
AMT Memory Monitor___
function keys: F5-> change address; F10->quit
arrow keys: left->column left; right->column right; up
other keys: page up->page up; page down->page down; hor
enter new address:
0088cb78: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0088cb94: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0088cbb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0088cbcc: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0088cbe8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0088cc04: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0088cc20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0088cc3c: 00 ad ff 01 00 ed ff 01 f0 e6 ff 01 00 00 00 00
0088cc58: 00 00 00 00 c0 6c ff 01 c0 6c ff 01 0d 00 01 00
0088cc74: 98 e4 7a 01 01 00 00 00 02 00 00 00 00 00 00 00
0088cc90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0088ccac: 00 00 00 00 00 00 00 00 00 00 00 00 02 6f 00 00
0088ccc8: 02 98 00 00 00 00 00 00 00 00 00 00 01 00 00 00
0088cce4: 01 00 00 00 42 62 01 00 00 00 00 00 01 00 00 00
0088cd00: 00 00 00 00 01 00 00 00 00 44 00 01 01 00 00 00
0088cd1c: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0088cd38: 00 00 00 00 48 00 00 00 00 00 00 00 00 00 00 00
```

```
patrickx@ubuntu64bit: ~/intelActiveManagementTechnology.project/subversion/patrickx.src/AMTBPHelper
 =================
 -* AMTBPHelper *-
=================

Press F1 and enter start and stop address of code to be disassembled.

*LOADER   : 0x000000..0x0122b8, code: 0x000050..0x0013e0, entry: 0x000050
*KERNEL   : 0x0122d0..0x28979c, code: 0x012320..0x05f068, entry: 0x031a10
*PMHWSEQ  : 0x2897b0..0x28ddf0, code: 0x289800..0x28cad8, entry: 0x28a170
*QST      : 0x28de00..0x2a79e8, code: 0x28de50..0x29b3f4, entry: 0x291b48
*OS       : 0x2a7a00..0x88ee28, code: 0x2a7a50..0x5ada48, entry: 0x4ecc58
*ADMIN_CM: 0x88ee40..0x98ccf8, code: 0x88ee90..0x91a810, entry: 0x8b2994
*AMT_CM   : 0x98cd10..0xaa35fc, code: 0x98cd60..0xa2089c, entry: 0x9bb964
*ASF_CM   : 0xaa3610..0xab4dec, code: 0xaa3660..0xaad59c, entry: 0xaabc58
*INJ?CODE: 0x180000..0x250000, code: 0x??????..0x??????, entry: 0x180074

.. r0   00000000   .. ar0   00000000
.. r1   00000000   .. ar1   00000000
.. r2   00000000   .. ar2   00000000
.. r3   00000000   .. ar3   00000000
.. r4   00000000   .. ar4   00000000
.. r5   00000000   .. ar5   00000000
.. r6   00000000   .. ar6   00000000
.. r7   00000000   .. ar7   00000000
.. r8   00000000   .. ar8   00000000
.. r9   00000000   .. ar9   00000000
.. r10  00000000   .. ara   00000000
.. r11  00000000   .. arb   00000000
.. r12  00000000   .. arc   00000000
.. r13  00000000   .. ard   00000000
.. r14  00000000   .. are   00000000
.. r15  00000000   .. arf   00000000
.. r16  00000000   .. ar10  00000000
.. r17  00000000   .. ar11  00000000
.. r18  00000000   .. ar12  00000000
.. r19  00000000   .. ar13  00000000
.. r20  00000000   .. ar14  00000000
.. r21  00000000   .. ar15  00000000
.. r22  00000000   .. ar16  00000000
.. r23  00000000   .. ar17  00000000
.. r24  00000000   .. ar18  00000000
.. r25  00000000   .. ar19  00000000
.. r26  00000000   .. ar1a  00000000
.. r27  00000000   .. ar1b  00000000
.. r28  00000000   .. ar1c  00000000
.. r29  00000000   .. ar1d  00000000
.. r30  00000000   .. ar1e  00000000
.. r31  00000000   .. ar1f  00000000

BP: 0x0                StartAR: 0
AMTBPHelper started.
1Re|Load  2Set bp  3Del bp  4Continue  5Goto  6R all BP  7Fin Sess  8Re|Start  9Chg StAR  10Quit
```

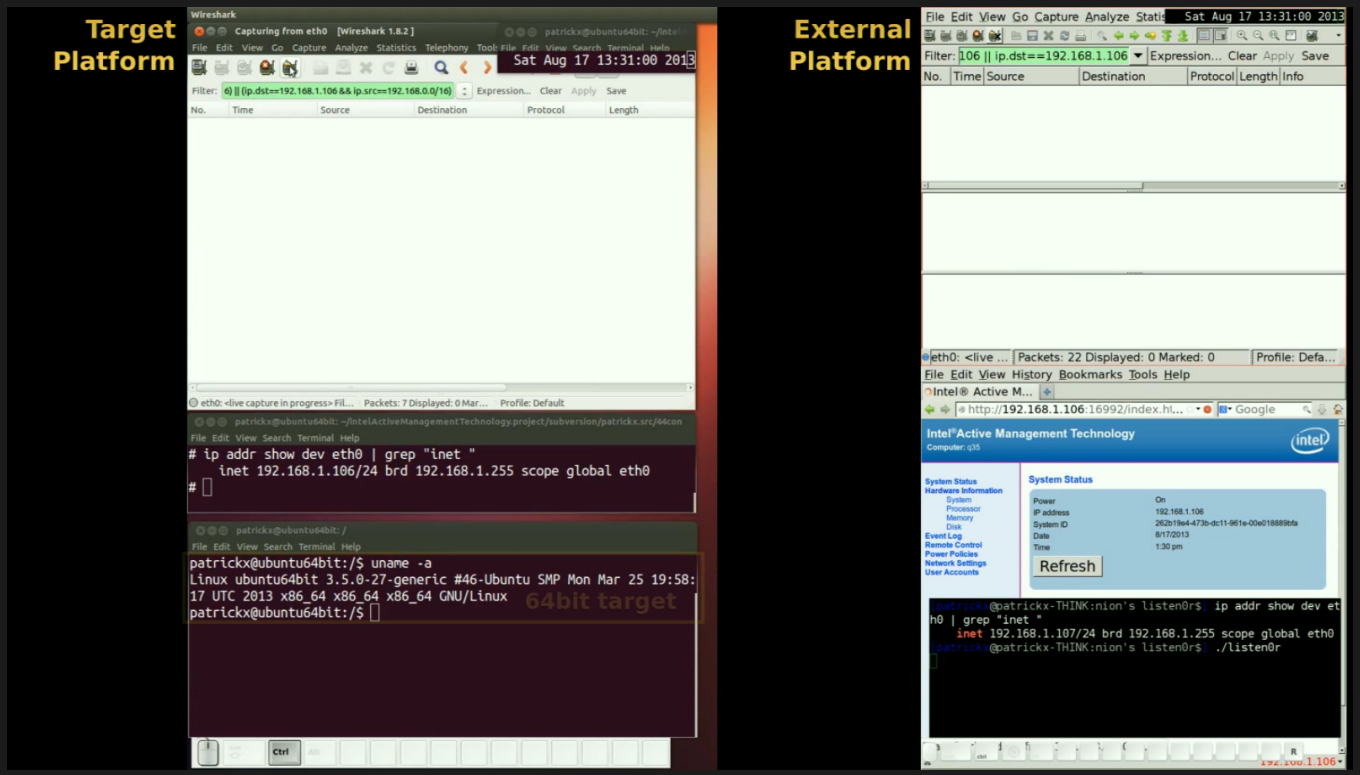
# [patrickx@44con:~$] cat 'Code for Sending Packets'

## $ (un)plug network cable → one DHCP packet

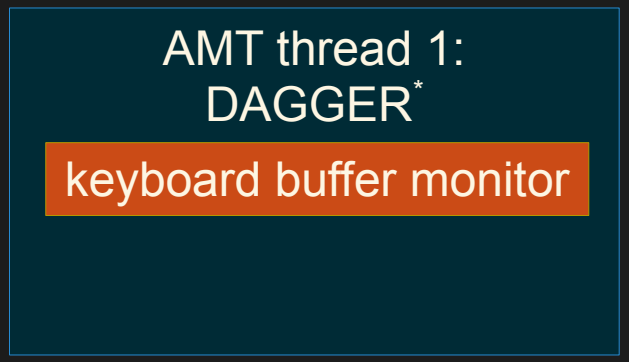


# Demo Video 1

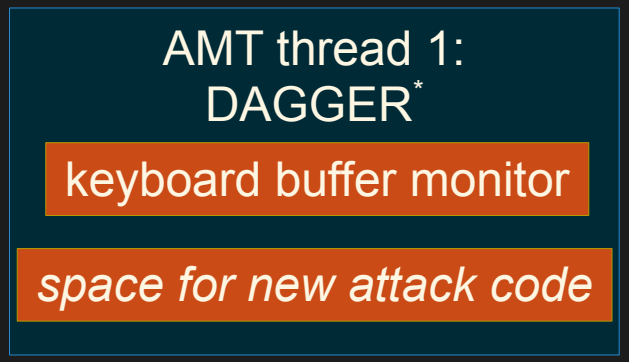
## *Exfiltrating Password via OOB*



# [patrickx@44con:~$] cat 'DAGGER Updates'



# [patrickx@44con:~$] cat 'DAGGER Updates'



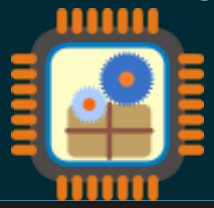
# [patrickx@44con:~$] cat 'DAGGER Updates'

AMT thread 1:
DAGGER*

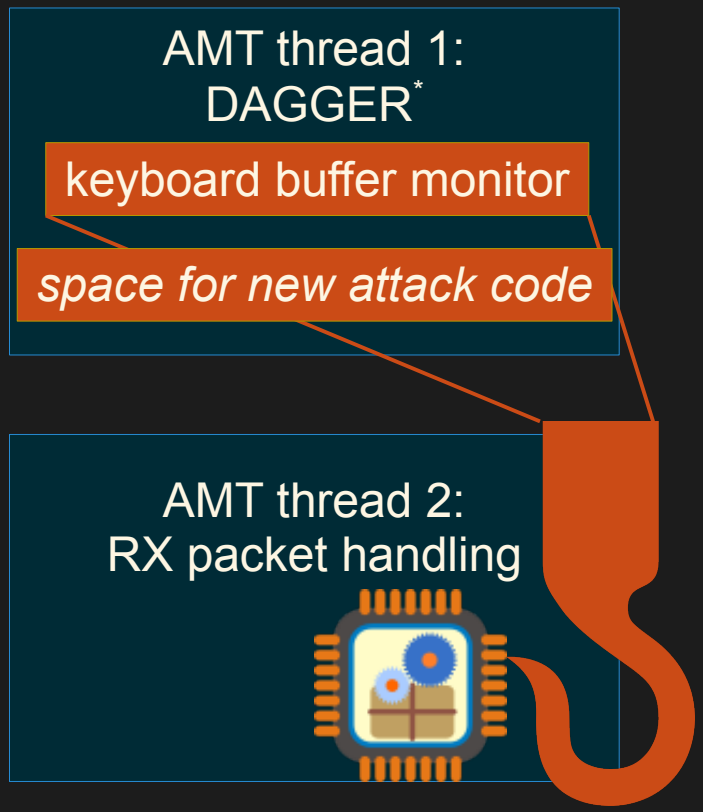
keyboard buffer monitor

*space for new attack code*

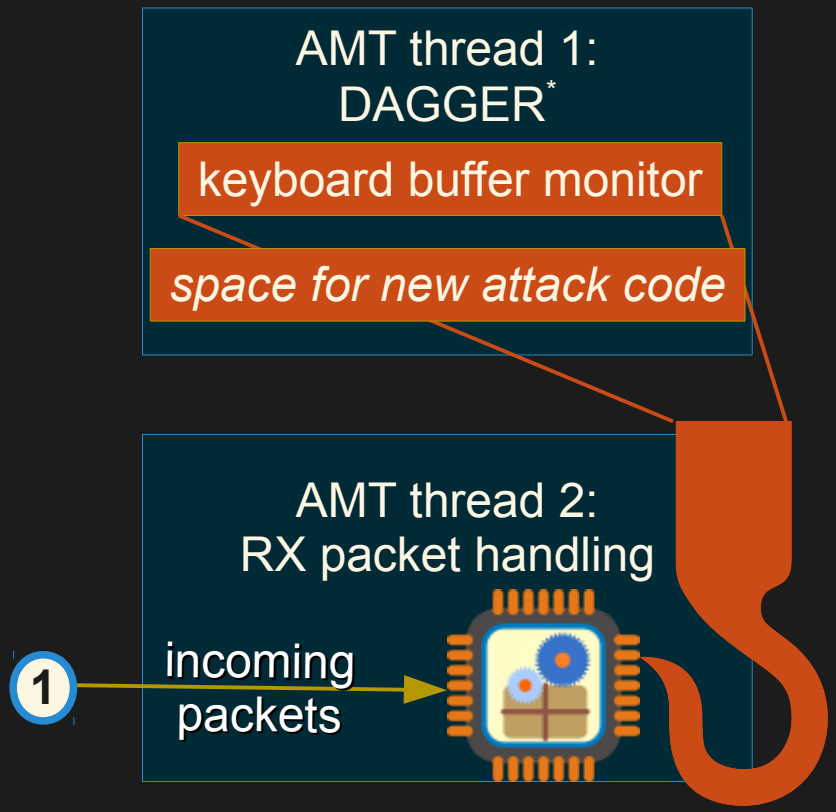
AMT thread 2:
RX packet handling



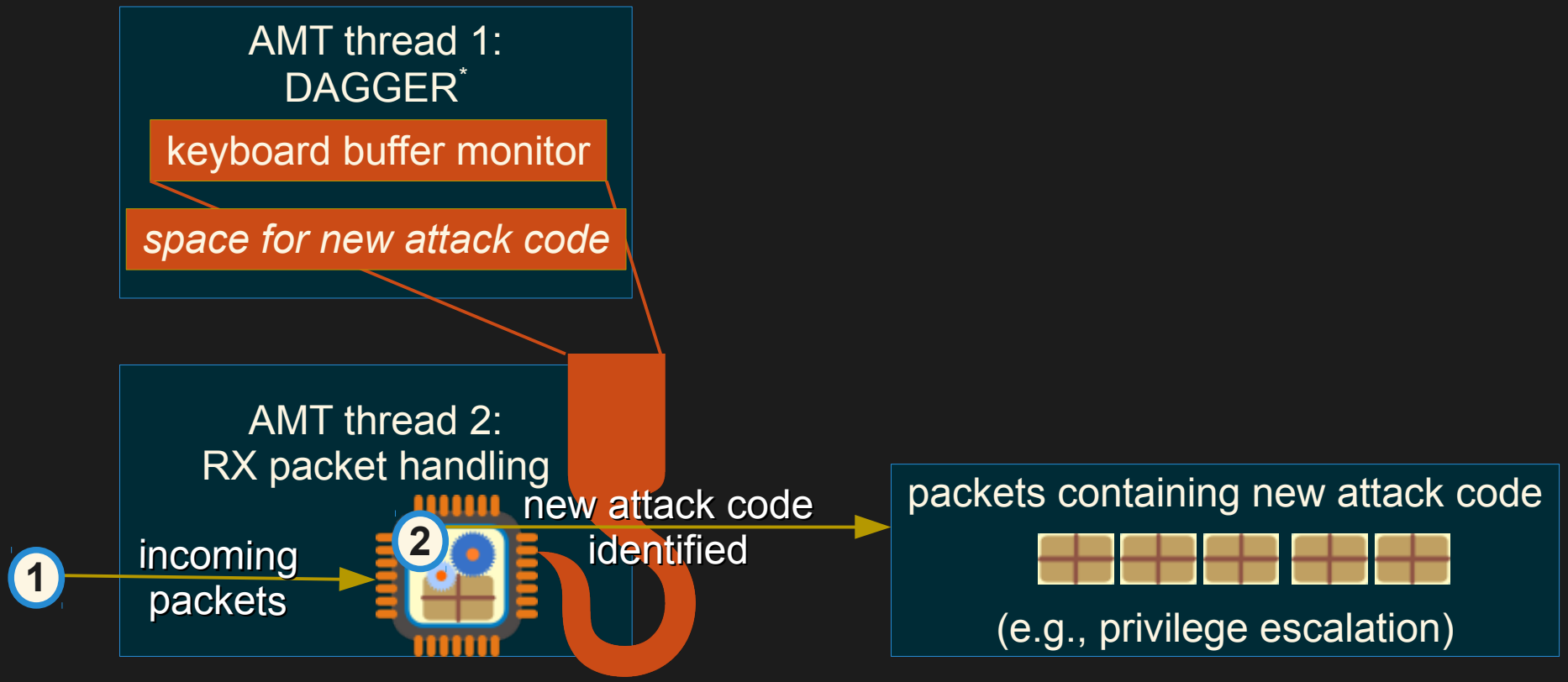
# [patrickx@44con:~$] cat 'DAGGER Updates'



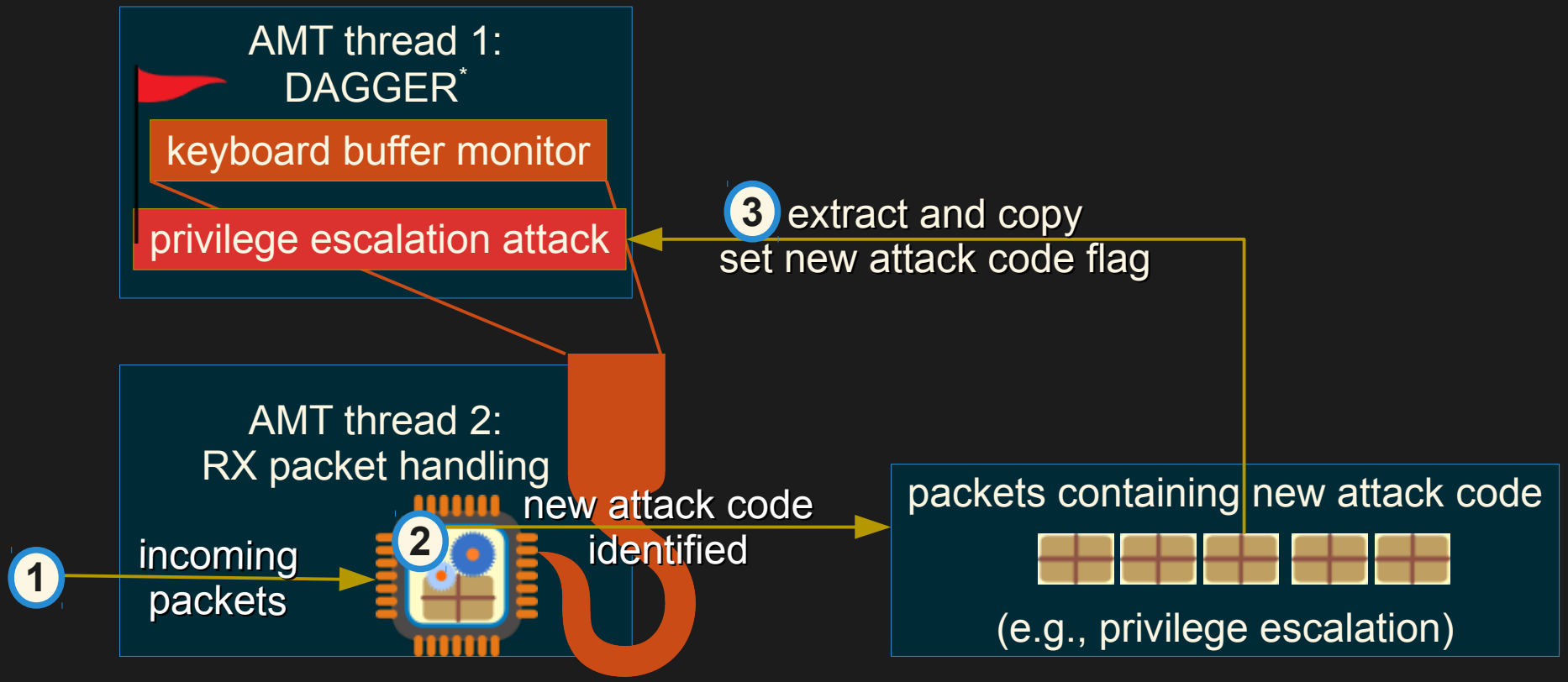
# [patrickx@44con:~$] cat 'DAGGER Updates'



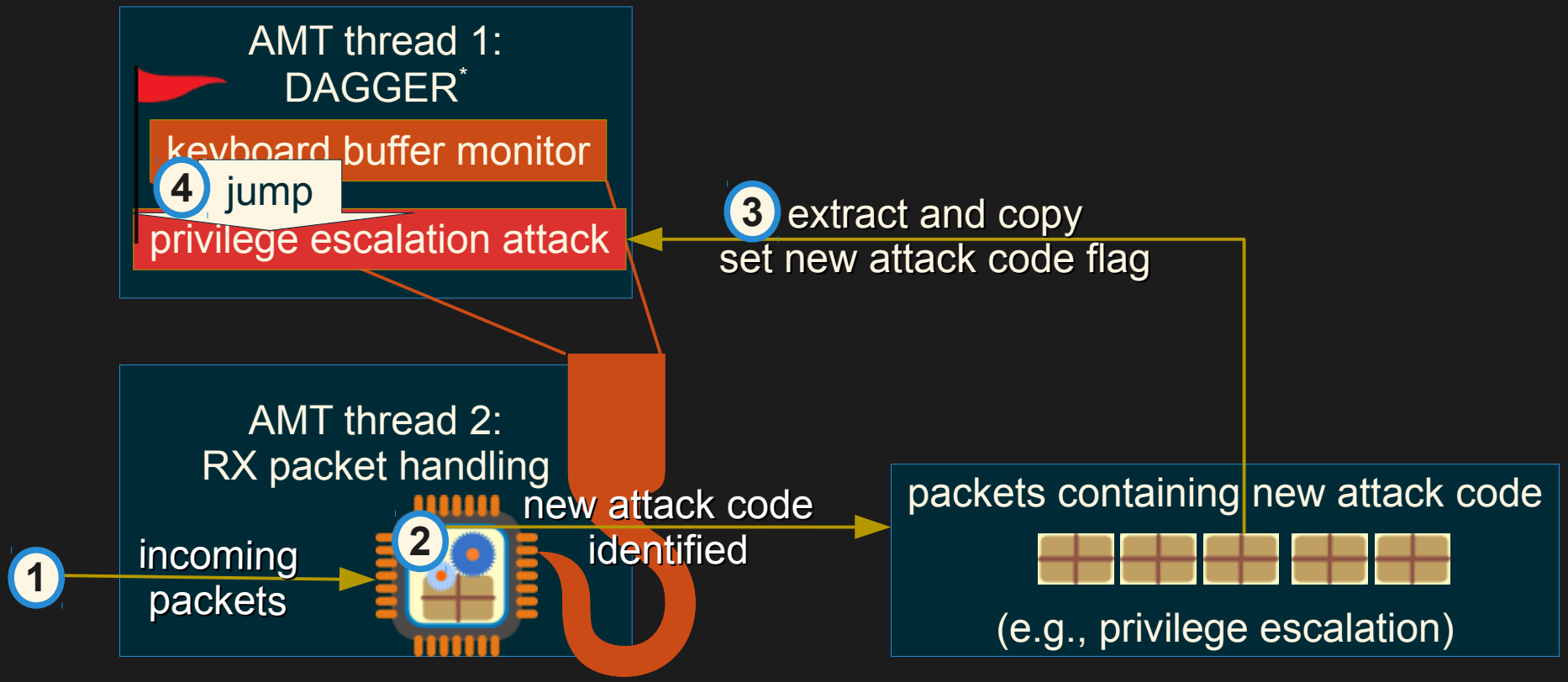
# [patrickx@44con:~$] cat 'DAGGER Updates'



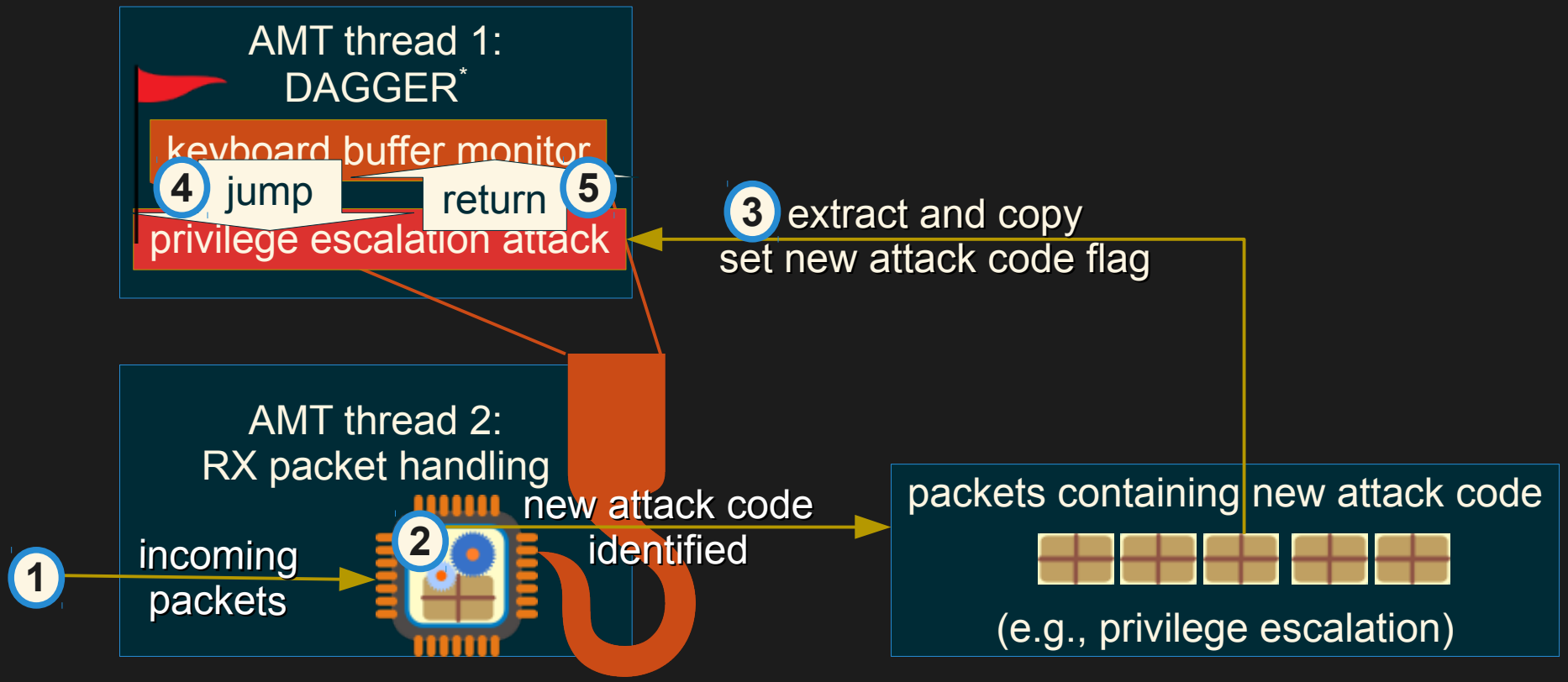
# [patrickx@44con:~$] cat 'DAGGER Updates'



# [patrickx@44con:~$] cat 'DAGGER Updates'

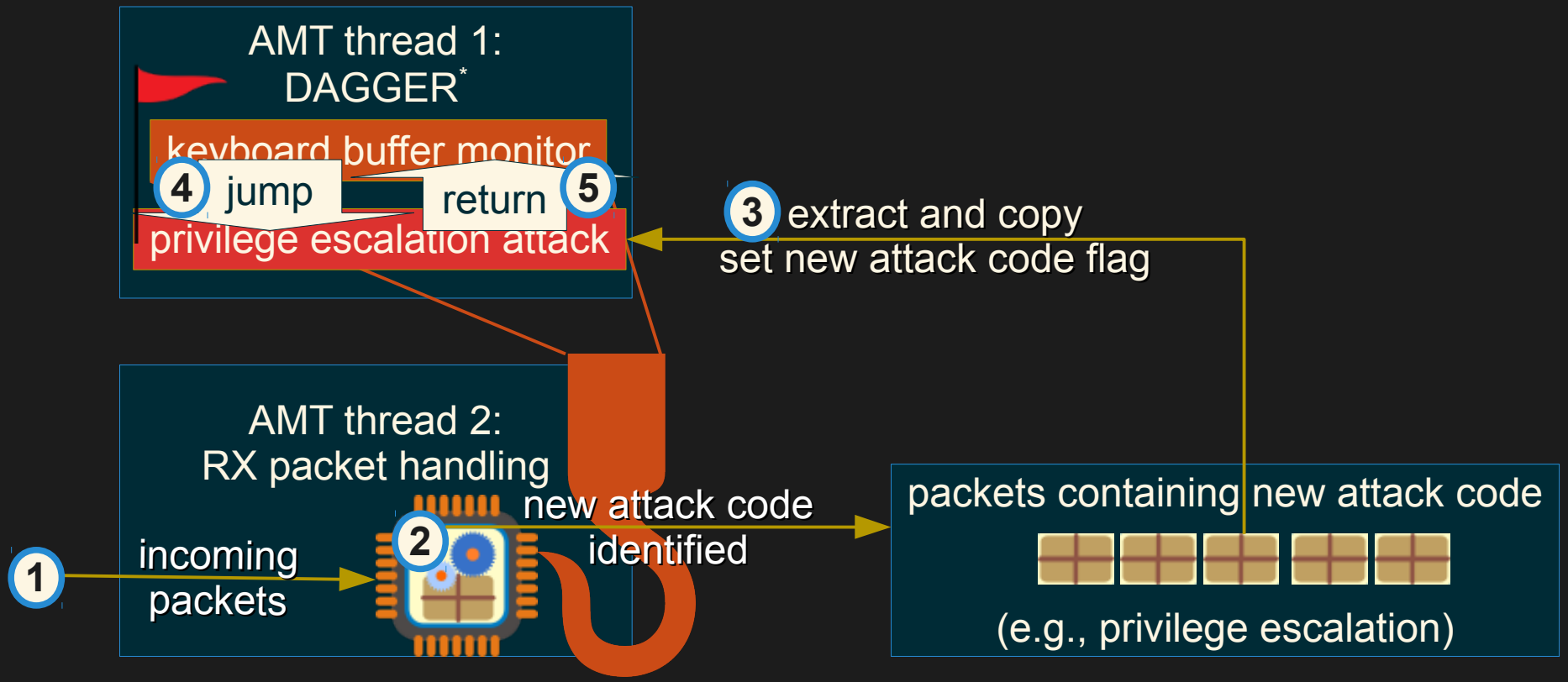


# [patrickx@44con:~$] cat 'DAGGER Updates'

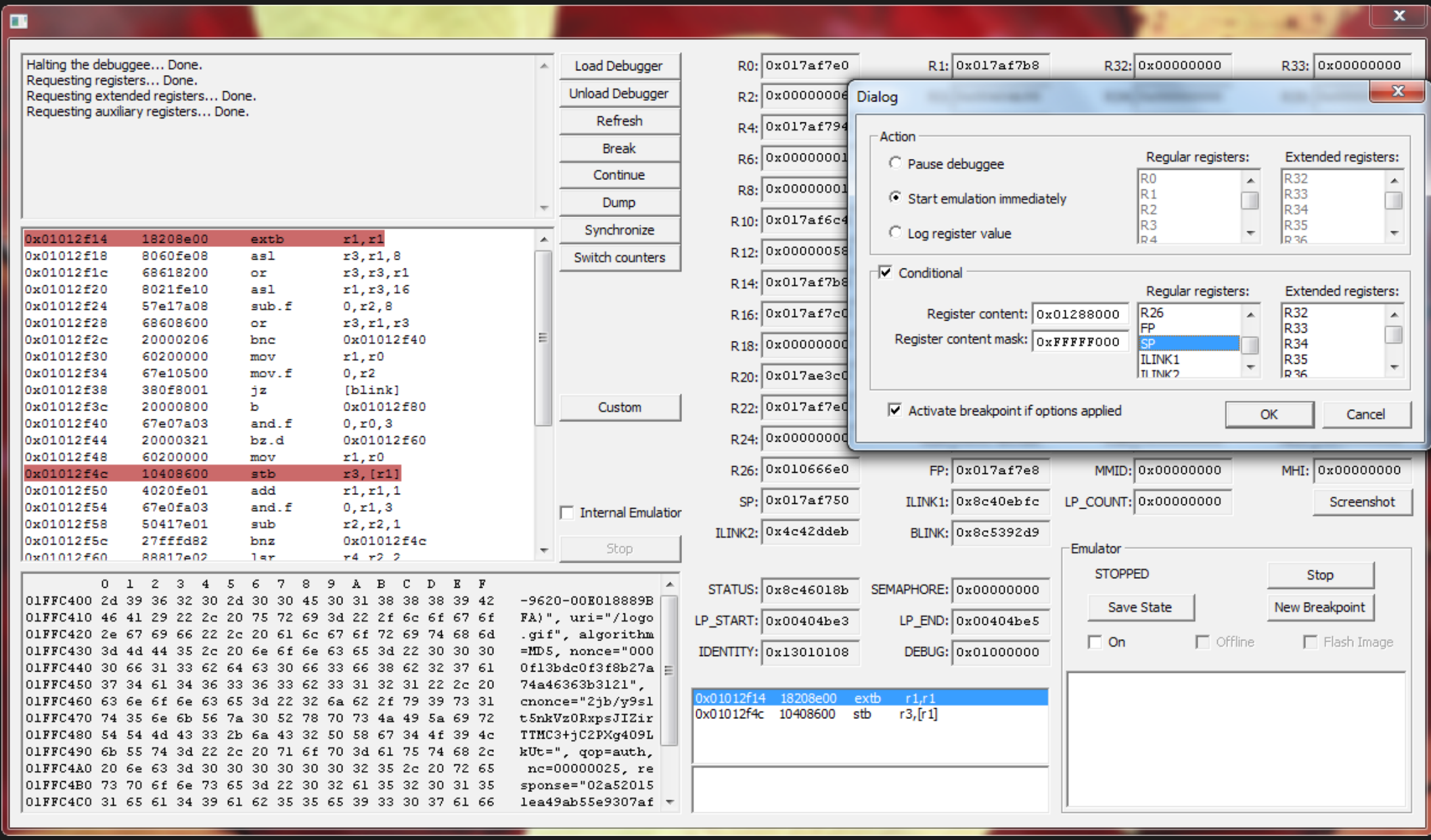


# [patrickx@44con:~$] cat 'DAGGER Updates'

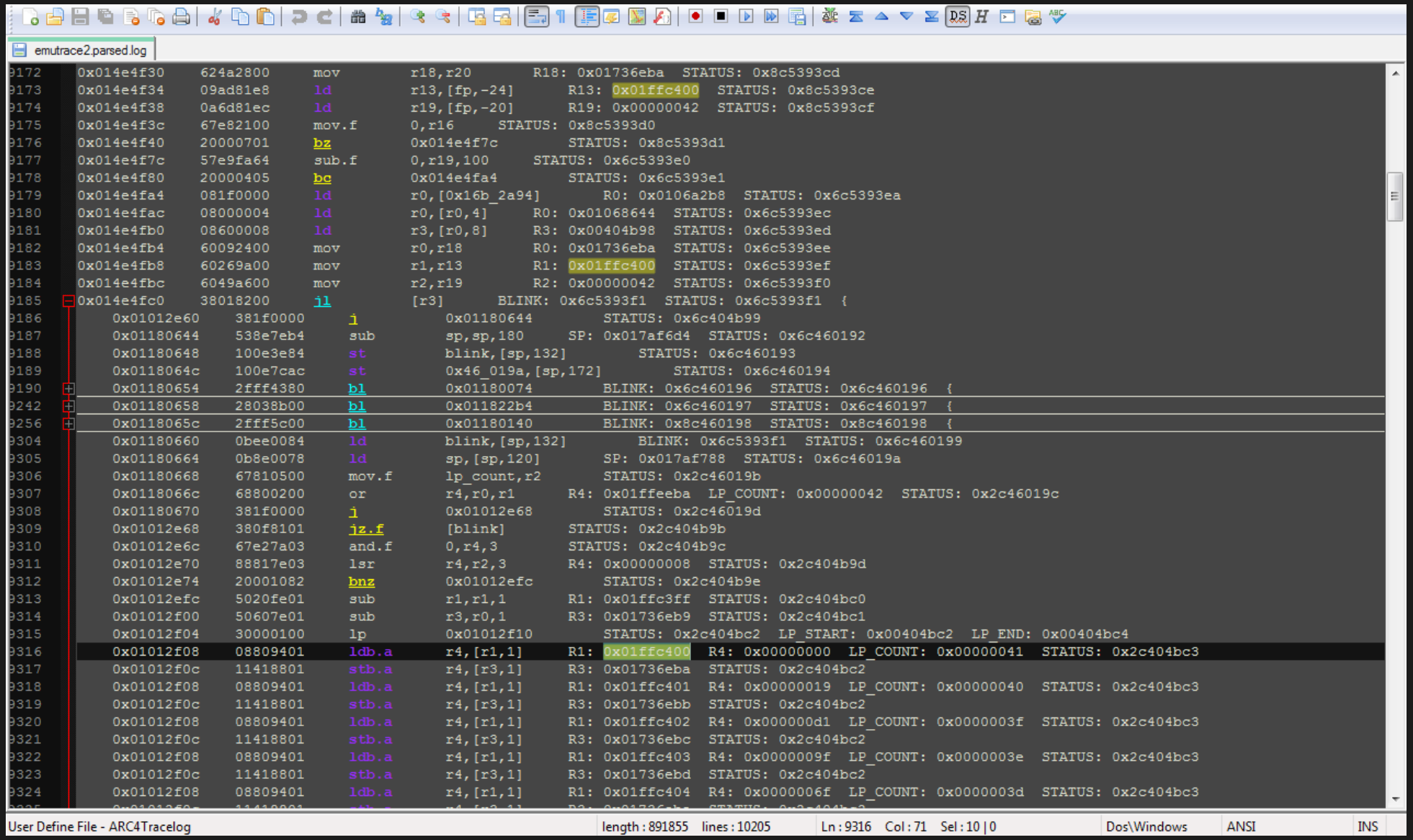
→ **How to find code responsible for handling incoming network packets?**



# [patrickx@44con:~$] cat 'Our Research Tools'



# [patrickx@44con:~$] cat 'Trace Log'



# [patrickx@44con:~$] cat 'Trace Log'

```
mov       r0,r18          R0: 0x01736eba  STATUS: 0x6c5393ee
mov       r1,r13          R1: 0x01ffc400  STATUS: 0x6c5393ef
mov       r2,r19          R2: 0x00000042  STATUS: 0x6c5393f0
jl        [r3]       BLINK: 0x6c5393f1  STATUS: 0x6c5393f1  {
    j         0x01180644          STATUS: 0x6c404b99
    sub       sp,sp,180       SP: 0x017af6d4  STATUS: 0x6c460192
    st        blink,[sp,132]          STATUS: 0x6c460193
    st        0x46_019a,[sp,172]          STATUS: 0x6c460194
    bl        0x01180074          BLINK: 0x6c460196  STATUS: 0x6c
    bl        0x011822b4          BLINK: 0x6c460197  STATUS: 0x6c
    bl        0x01180140          BLINK: 0x8c460198  STATUS: 0x8c
    ld        blink,[sp,132]          BLINK: 0x6c5393f1  STATUS:
    ld        sp,[sp,120]         SP: 0x017af788  STATUS: 0x6c460
    mov.f     lp_count,r2         STATUS: 0x2c46019b
    or        r4,r0,r1        R4: 0x01ffeeba  LP_COUNT: 0x0000004
    j         0x01012e68          STATUS: 0x2c46019d
    jz.f      [blink]         STATUS: 0x2c404b9b
    and.f     0,r4,3          STATUS: 0x2c404b9c
    lsr       r4,r2,3         R4: 0x00000008  STATUS: 0x2c404b9d
    bnz       0x01012efc          STATUS: 0x2c404b9e
    sub       r1,r1,1         R1: 0x01ffc3ff  STATUS: 0x2c404bc0
    sub       r3,r0,1         R3: 0x01736eb9  STATUS: 0x2c404bc1
    lp        0x01012f10          STATUS: 0x2c404bc2  LP_START: 0
    ldb.a     r4,[r1,1]       R1: 0x01ffc400  R4: 0x00000000  LP_
    stb.a     r4,[r3,1]       R3: 0x01736eba  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc401  R4: 0x00000019  LP_
    stb.a     r4,[r3,1]       R3: 0x01736ebb  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc402  R4: 0x000000d1  LP_
    stb.a     r4,[r3,1]       R3: 0x01736ebc  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc403  R4: 0x0000009f  LP_
    stb.a     r4,[r3,1]       R3: 0x01736ebd  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc404  R4: 0x0000006f  LP_
```

# [patrickx@44con:~$] cat 'Trace Log'

```
mov       r0,r18          R0: 0x01736eba  STATUS: 0x6c5393ee
mov       r1,r13          R1: 0x01ffc400  STATUS: 0x6c5393ef
mov       r2,r19          R2: 0x00000042  STATUS: 0x6c5393f0
jl        [r3]       BLINK: 0x6c5393f1  STATUS: 0x6c5393f1  {
    j         0x01180644          STATUS: 0x6c404b99
    sub       sp,sp,180       SP: 0x017af6d4  STATUS: 0x6c460192
    st        blink,[sp,132]          STATUS: 0x6c460193
    st        0x46_019a,[sp,172]          STATUS: 0x6c460194
    bl        0x01180074          BLINK: 0x6c460196  STATUS: 0x6c
    bl        0x011822b4          BLINK: 0x6c460197  STATUS: 0x6c
    bl        0x01180140          BLINK: 0x8c460198  STATUS: 0x8c
    ld        blink,[sp,132]          BLINK: 0x6c5393f1  STATUS:
    ld        sp,[sp,120]         SP: 0x017af788  STATUS: 0x6c460
    mov.f     lp_count,r2         STATUS: 0x2c46019b
    or        r4,r0,r1        R4: 0x01ffeeba  LP_COUNT: 0x0000004
    j         0x01012e68          STATUS: 0x2c46019d
    jz.f      [blink]         STATUS: 0x2c404b9b
    and.f     0,r4,3          STATUS: 0x2c404b9c
    lsr       r4,r2,3         R4: 0x00000008  STATUS: 0x2c404b9d
    bnz       0x01012efc          STATUS: 0x2c404b9e
    sub       r1,r1,1         R1: 0x01ffc3ff  STATUS: 0x2c404bc0
    sub       r3,r0,1         R3: 0x01736eb9  STATUS: 0x2c404bc1
    lp        0x01012f10          STATUS: 0x2c404bc2  LP_START: 0
    ldb.a     r4,[r1,1]       R1: 0x01ffc400  R4: 0x00000000  LP_
    stb.a     r4,[r3,1]       R3: 0x01736eba  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc401  R4: 0x00000019  LP_
    stb.a     r4,[r3,1]       R3: 0x01736ebb  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc402  R4: 0x000000d1  LP_
    stb.a     r4,[r3,1]       R3: 0x01736ebc  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc403  R4: 0x0000009f  LP_
    stb.a     r4,[r3,1]       R3: 0x01736ebd  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc404  R4: 0x0000006f  LP_
```

} memcpy call

# [patrickx@44con:~$] cat 'Trace Log'

```
mov       r0,r18          R0: 0x01736eba  STATUS: 0x6c5393ee
mov       r1,r13          R1: 0x01ffc400  STATUS: 0x6c5393ef
mov       r2,r19          R2: 0x00000042  STATUS: 0x6c5393f0
jl        [r3]       BLINK: 0x6c5393f1  STATUS: 0x6c5393f1  {
    j         0x01180644          STATUS: 0x6c404b99
    sub       sp,sp,180       SP: 0x017af6d4  STATUS: 0x6c460192
    st        blink,[sp,132]          STATUS: 0x6c460193
    st        0x46_019a,[sp,172]          STATUS: 0x6c460194
    bl        0x01180074          BLINK: 0x6c460196  STATUS: 0x6c
    bl        0x011822b4          BLINK: 0x6c460197  STATUS: 0x6c
    bl        0x01180140          BLINK: 0x8c460198  STATUS: 0x8c
    ld        blink,[sp,132]          BLINK: 0x6c5393f1  STATUS:
    ld        sp,[sp,120]         SP: 0x017af788  STATUS: 0x6c460
    mov.f     lp_count,r2         STATUS: 0x2c46019b
    or        r4,r0,r1        R4: 0x01ffeeba  LP_COUNT: 0x0000004
    j         0x01012e68          STATUS: 0x2c46019d
    jz.f      [blink]         STATUS: 0x2c404b9b
    and.f     0,r4,3          STATUS: 0x2c404b9c
    lsr       r4,r2,3         R4: 0x00000008  STATUS: 0x2c404b9d
    bnz       0x01012efc          STATUS: 0x2c404b9e
    sub       r1,r1,1         R1: 0x01ffc3ff  STATUS: 0x2c404bc0
    sub       r3,r0,1         R3: 0x01736eb9  STATUS: 0x2c404bc1
    lp        0x01012f10          STATUS: 0x2c404bc2  LP_START: 0
    ldb.a     r4,[r1,1]       R1: 0x01ffc400  R4: 0x00000000  LP_
    stb.a     r4,[r3,1]       R3: 0x01736eba  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc401  R4: 0x00000019  LP_
    stb.a     r4,[r3,1]       R3: 0x01736ebb  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc402  R4: 0x000000d1  LP_
    stb.a     r4,[r3,1]       R3: 0x01736ebc  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc403  R4: 0x0000009f  LP_
    stb.a     r4,[r3,1]       R3: 0x01736ebd  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc404  R4: 0x0000006f  LP_
```

memcpy parameter

memcpy call

# [patrickx@44con:~$] cat 'Trace Log'

```
mov       r0,r18          R0: 0x01736eba  STATUS: 0x6c5393ee
mov       r1,r13          R1: 0x01ffc400  STATUS: 0x6c5393ef
mov       r2,r19          R2: 0x00000042  STATUS: 0x6c5393f0
jl        [r3]       BLINK: 0x6c5393f1  STATUS: 0x6c5393f1  {
    j         0x01180644          STATUS: 0x6c404b99
    sub       sp,sp,180       SP: 0x017af6d4  STATUS: 0x6c460192
    st        blink,[sp,132]          STATUS: 0x6c460193
    st        0x46_019a,[sp,172]          STATUS: 0x6c460194
    bl        0x01180074          BLINK: 0x6c460196  STATUS: 0x6c
    bl        0x011822b4          BLINK: 0x6c460197  STATUS: 0x6c
    bl        0x01180140          BLINK: 0x8c460198  STATUS: 0x8c
    ld        blink,[sp,132]          BLINK: 0x6c5393f1  STATUS:
    ld        sp,[sp,120]         SP: 0x017af788  STATUS: 0x6c460
    mov.f     lp_count,r2         STATUS: 0x2c46019b
    or        r4,r0,r1        R4: 0x01ffeeba  LP_COUNT: 0x0000004
    j         0x01012e68          STATUS: 0x2c46019d
    jz.f      [blink]         STATUS: 0x2c404b9b
    and.f     0,r4,3          STATUS: 0x2c404b9c
    lsr       r4,r2,3         R4: 0x00000008  STATUS: 0x2c404b9d
    bnz       0x01012efc          STATUS: 0x2c404b9e
    sub       r1,r1,1         R1: 0x01ffc3ff  STATUS: 0x2c404bc0
    sub       r3,r0,1         R3: 0x01736eb9  STATUS: 0x2c404bc1
    lp        0x01012f10          STATUS: 0x2c404bc2  LP_START: 0
    ldb.a     r4,[r1,1]       R1: 0x01ffc400  R4: 0x00000000  LP_
    stb.a     r4,[r3,1]       R3: 0x01736eba  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc401  R4: 0x00000019  LP_
    stb.a     r4,[r3,1]       R3: 0x01736ebb  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc402  R4: 0x000000d1  LP_
    stb.a     r4,[r3,1]       R3: 0x01736ebc  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc403  R4: 0x0000009f  LP_
    stb.a     r4,[r3,1]       R3: 0x01736ebd  STATUS: 0x2c404bc2
    ldb.a     r4,[r1,1]       R1: 0x01ffc404  R4: 0x0000006f  LP_
```

memcpy parameter

memcpy call

*our main hook is also traced into*

# [patrickx@44con:~$] cat 'Trace Log'

```
mov      r0,r18          R0: 0x01736eba  STATUS: 0x6c5393ee
mov      r1,r13          R1: 0x01ffc400  STATUS: 0x6c5393ef
mov      r2,r19          R2: 0x00000042  STATUS: 0x6c5393f0
jl       [r3]       BLINK: 0x6c5393f1  STATUS: 0x6c5393f1  {
    j        0x01180644          STATUS: 0x6c404b99
    sub      sp,sp,180       SP: 0x017af6d4  STATUS: 0x6c460192
    st       blink,[sp,132]          STATUS: 0x6c460193
    st       0x46_019a,[sp,172]      STATUS: 0x6c460194
    bl       0x01180074          BLINK: 0x6c460196  STATUS: 0x6c
    bl       0x011822b4          BLINK: 0x6c460197  STATUS: 0x6c
    bl       0x01180140          BLINK: 0x8c460198  STATUS: 0x8c
    ld       blink,[sp,132]          BLINK: 0x6c5393f1  STATUS:
    ld       sp,[sp,120]         SP: 0x017af788  STATUS: 0x6c460
    mov.f    lp_count,r2         STATUS: 0x2c46019b
    or       r4,r0,r1        R4: 0x01ffeeba  LP_COUNT: 0x0000004
    j        0x01012e68          STATUS: 0x2c46019d
    jz.f     [blink]         STATUS: 0x2c404b9b
    and.f    0,r4,3          STATUS: 0x2c404b9c
    lsr      r4,r2,3         R4: 0x00000008  STATUS: 0x2c404b9d
    bnz      0x01012efc          STATUS: 0x2c404b9e
    sub      r1,r1,1         R1: 0x01ffc3ff  STATUS: 0x2c404bc0
    sub      r3,r0,1         R3: 0x01736eb9  STATUS: 0x2c404bc1
    lp       0x01012f10          STATUS: 0x2c404bc2  LP_START: 0
    ldb.a    r4,[r1,1]       R1: 0x01ffc400  R4: 0x00000000  LP_
    stb.a    r4,[r3,1]       R3: 0x01736eba  STATUS: 0x2c404bc2
    ldb.a    r4,[r1,1]       R1: 0x01ffc401  R4: 0x00000019  LP_
    stb.a    r4,[r3,1]       R3: 0x01736ebb  STATUS: 0x2c404bc2
    ldb.a    r4,[r1,1]       R1: 0x01ffc402  R4: 0x000000d1  LP_
    stb.a    r4,[r3,1]       R3: 0x01736ebc  STATUS: 0x2c404bc2
    ldb.a    r4,[r1,1]       R1: 0x01ffc403  R4: 0x0000009f  LP_
    stb.a    r4,[r3,1]       R3: 0x01736ebd  STATUS: 0x2c404bc2
    ldb.a    r4,[r1,1]       R1: 0x01ffc404  R4: 0x0000006f  LP_
```

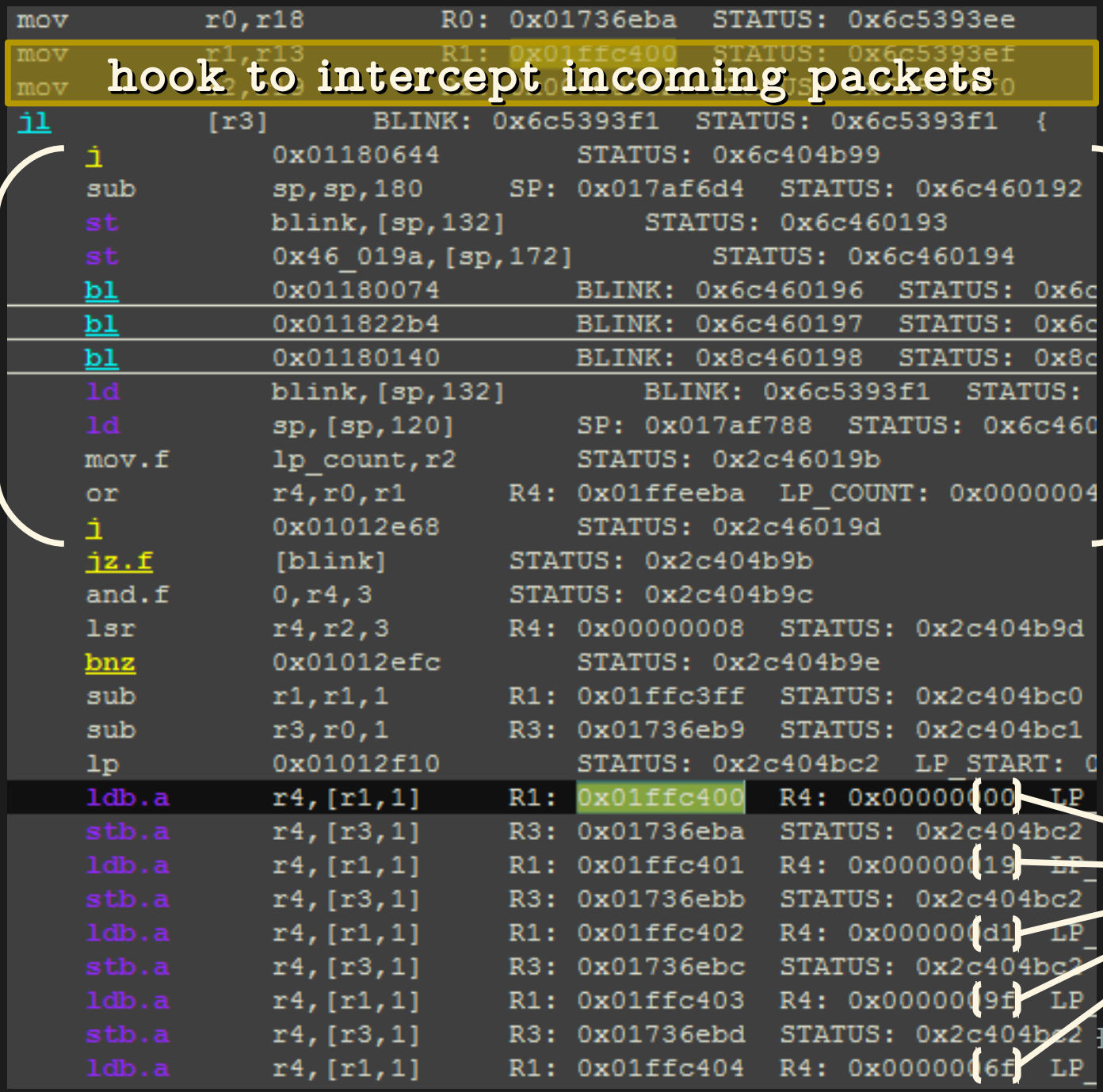
memcpy parameter

memcpy call

*our main hook is also traced into*

first bytes of an incoming packet

# [patrickx@44con:~$] cat 'Trace Log'

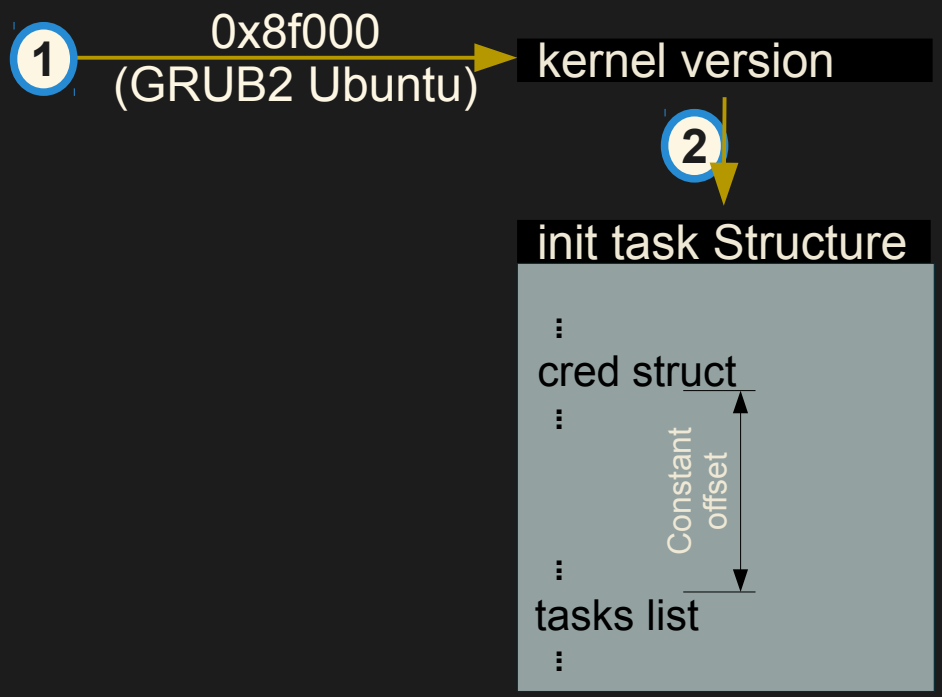


[patrickx@44con:~$] cat 'Privilege Escalation'

# [patrickx@44con:~$] cat 'Privilege Escalation'


(derive addresses, offsets, size of structures)

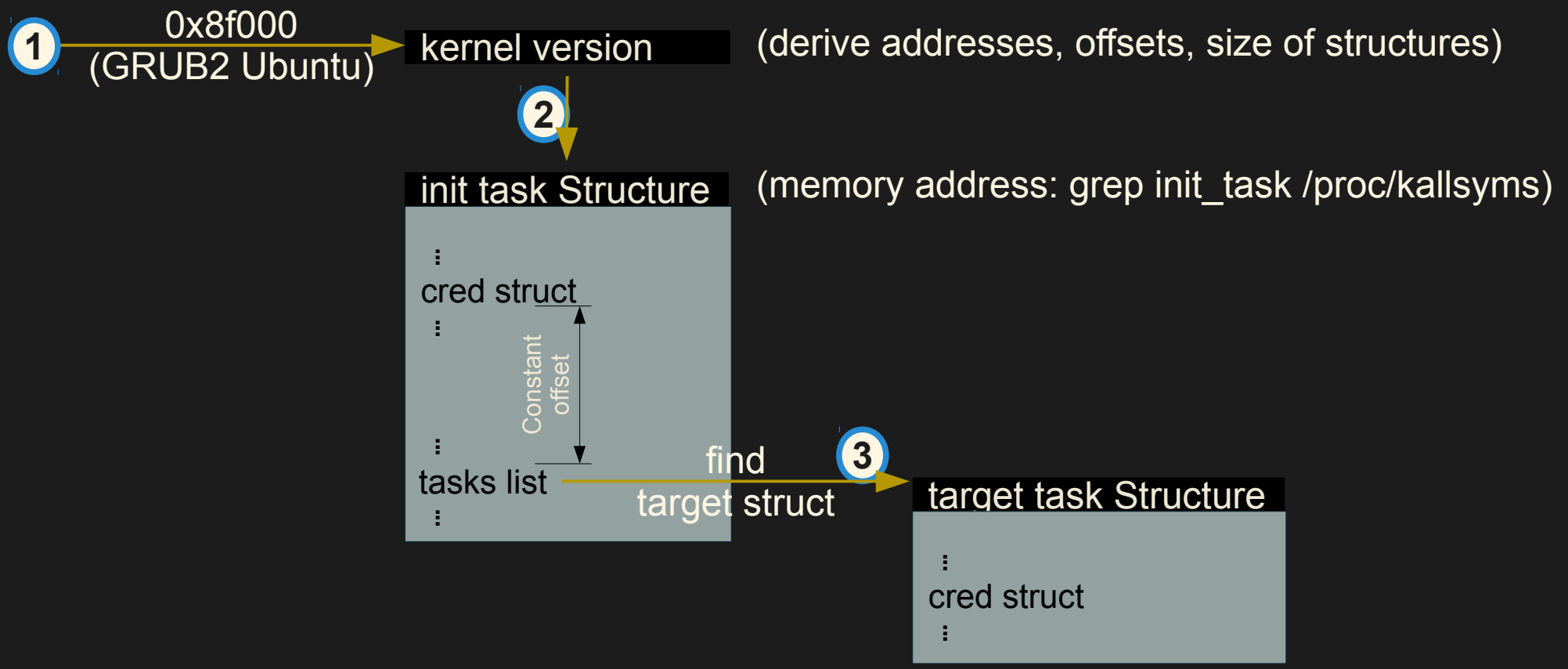
# [patrickx@44con:~$] cat 'Privilege Escalation'



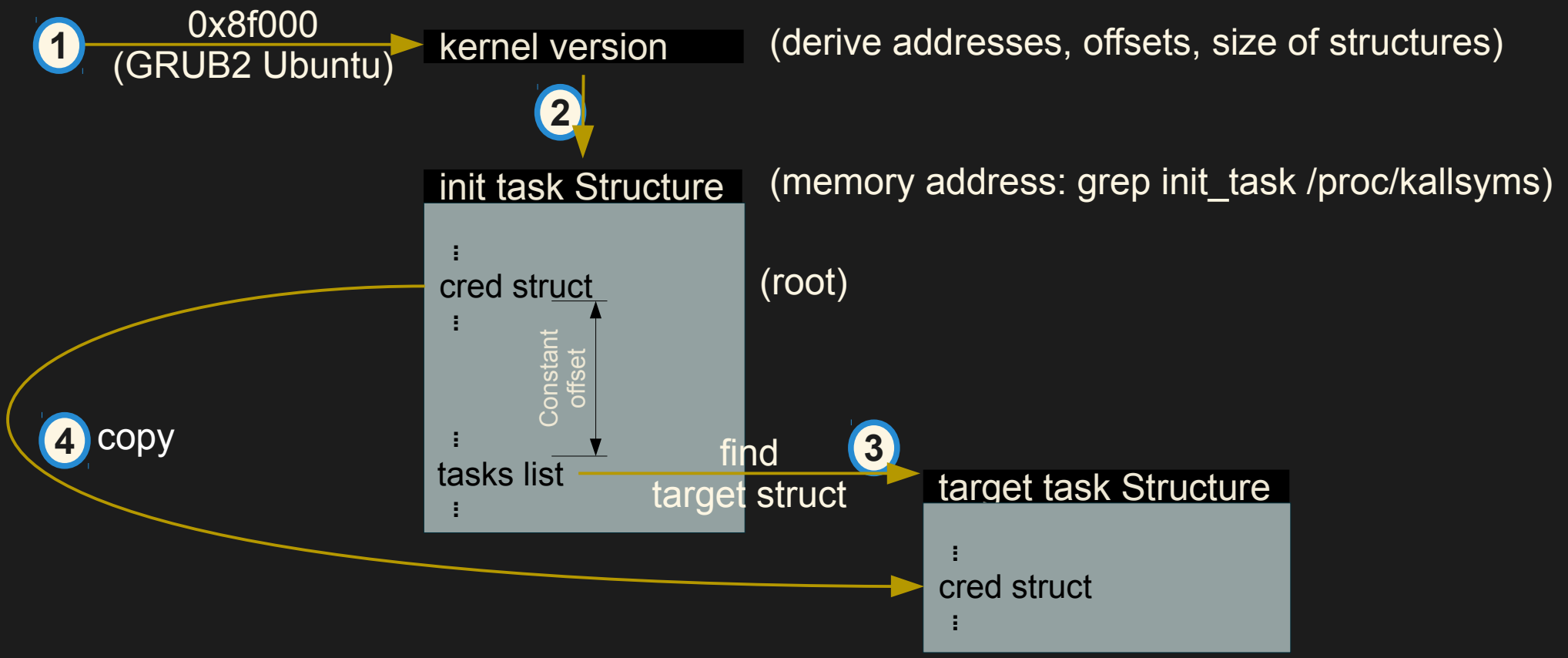

(derive addresses, offsets, size of structures)

(memory address: grep init_task /proc/kallsyms)

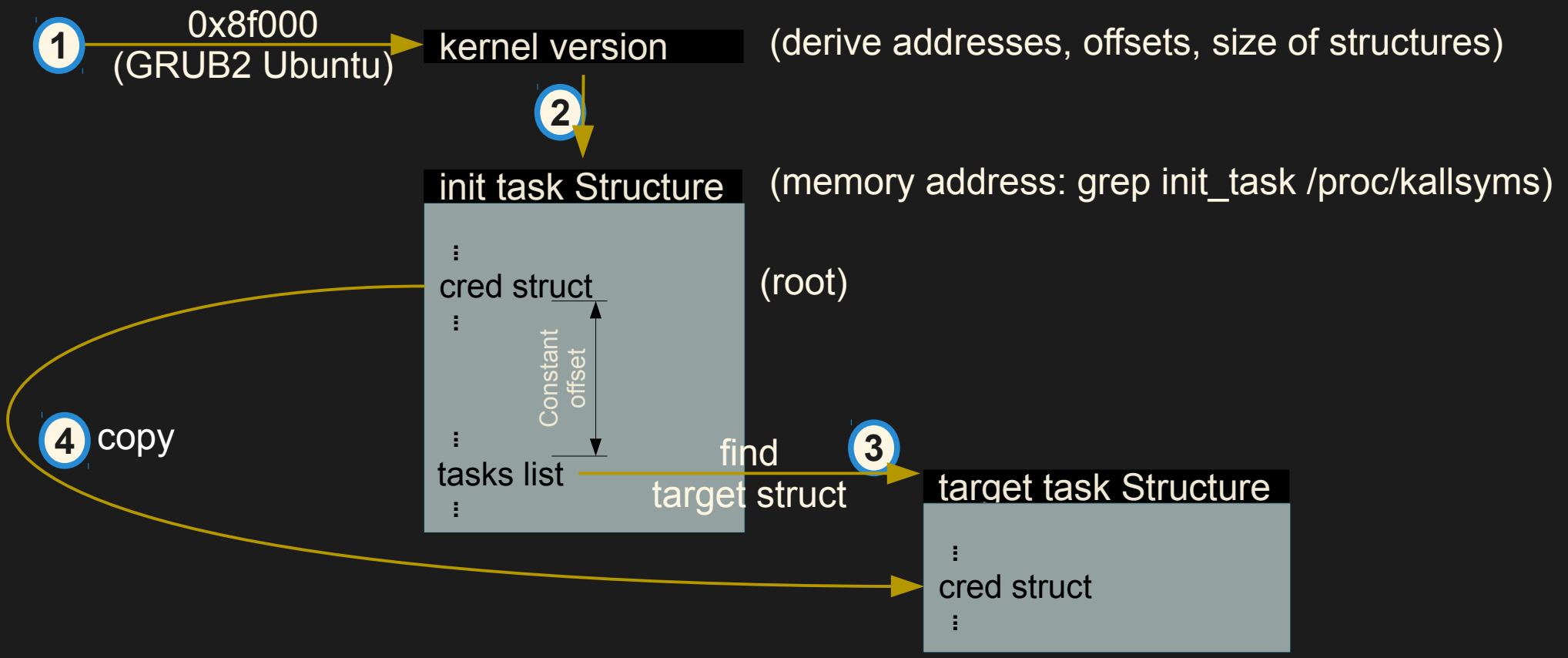
# [patrickx@44con:~$] cat 'Privilege Escalation'



# [patrickx@44con:~$] cat 'Privilege Escalation'



# [patrickx@44con:~$] cat 'Privilege Escalation'

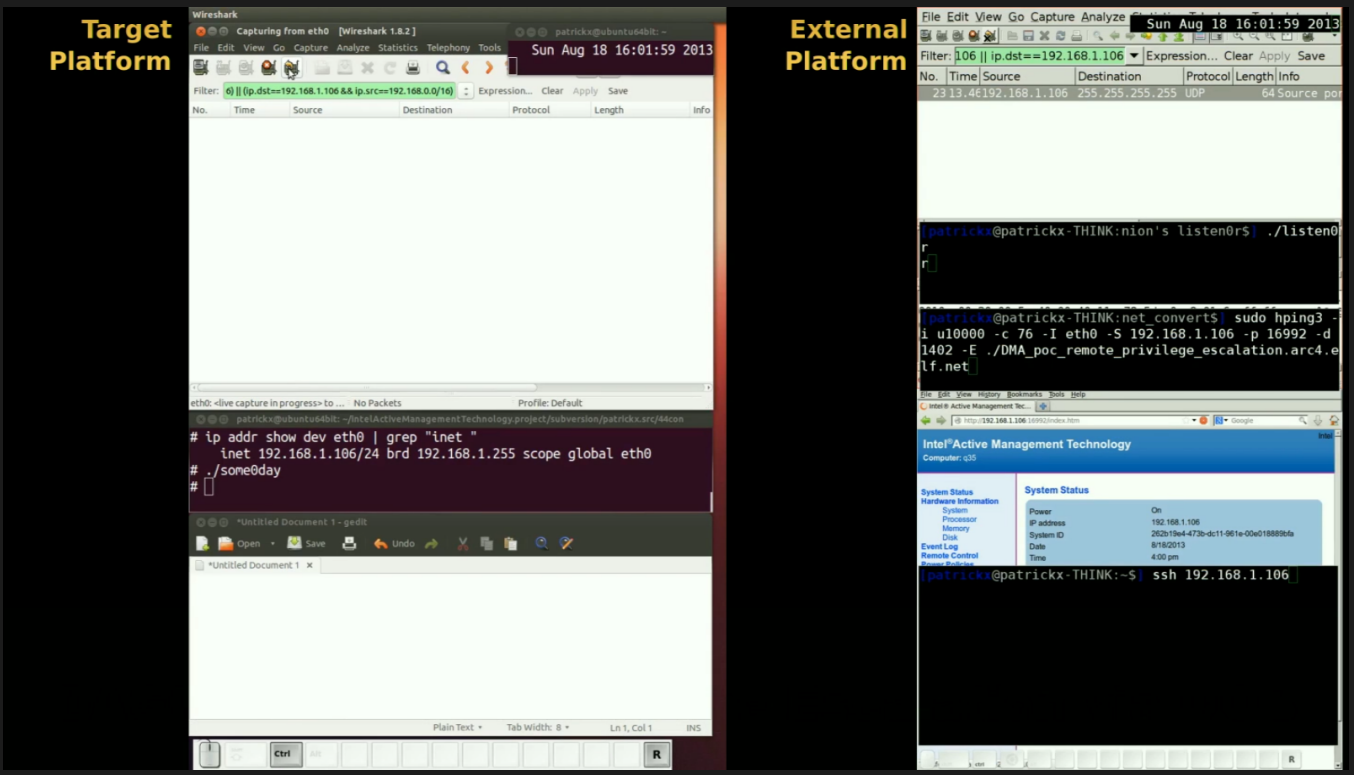



$ **Binary: DMA_poc_remote_privilege_escalation.arc4.elf**

$ **Sent via hping3**

man hping3 “[...] send (almost) arbitrary TCP/IP packets to network hosts [...]”

# Demo Video 2
# Privilege *Escalation*
# *via OOB*




Covert Network Channel

# [patrickx@44con:~$] cat "Trick Non-host Monitors"

$ JitterBug based, see "Keyboards and Covert Channels" [Sha06]:

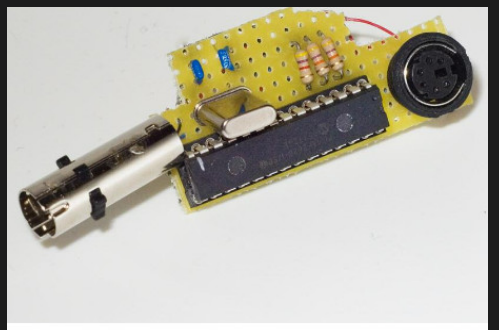
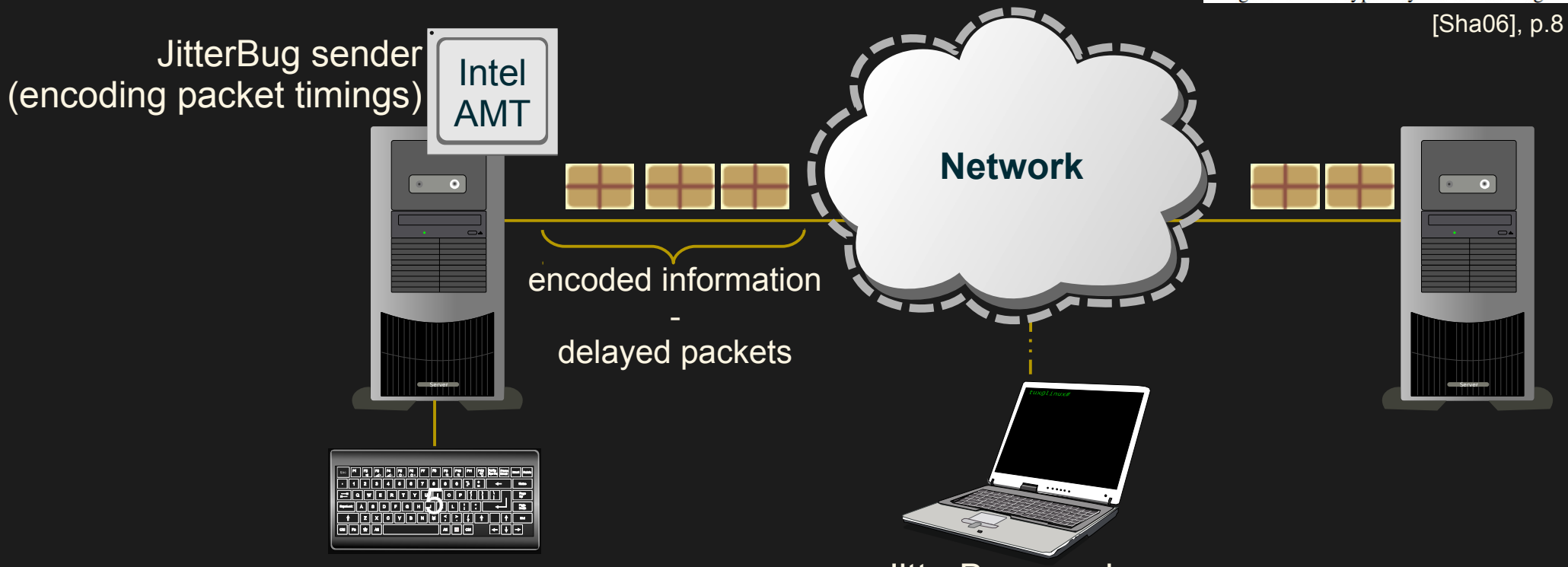


Figure 4: Prototype Keyboard JitterBug

[Sha06], p.8



# [patrickx@44con:~$] cat 'More ME Features'

- $ Outgoing packet interception ✓
- $ Measure time!

  AMT peripheral (timer) access:

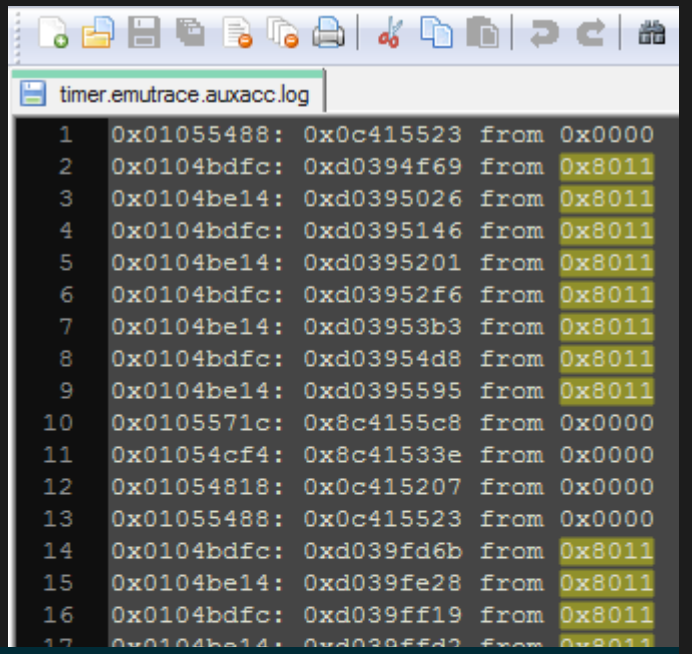
  `lr r0,[0x8011]`

  → Read timer register:

  resolution

  ~ 996500 Hz

- $ Packets to delay



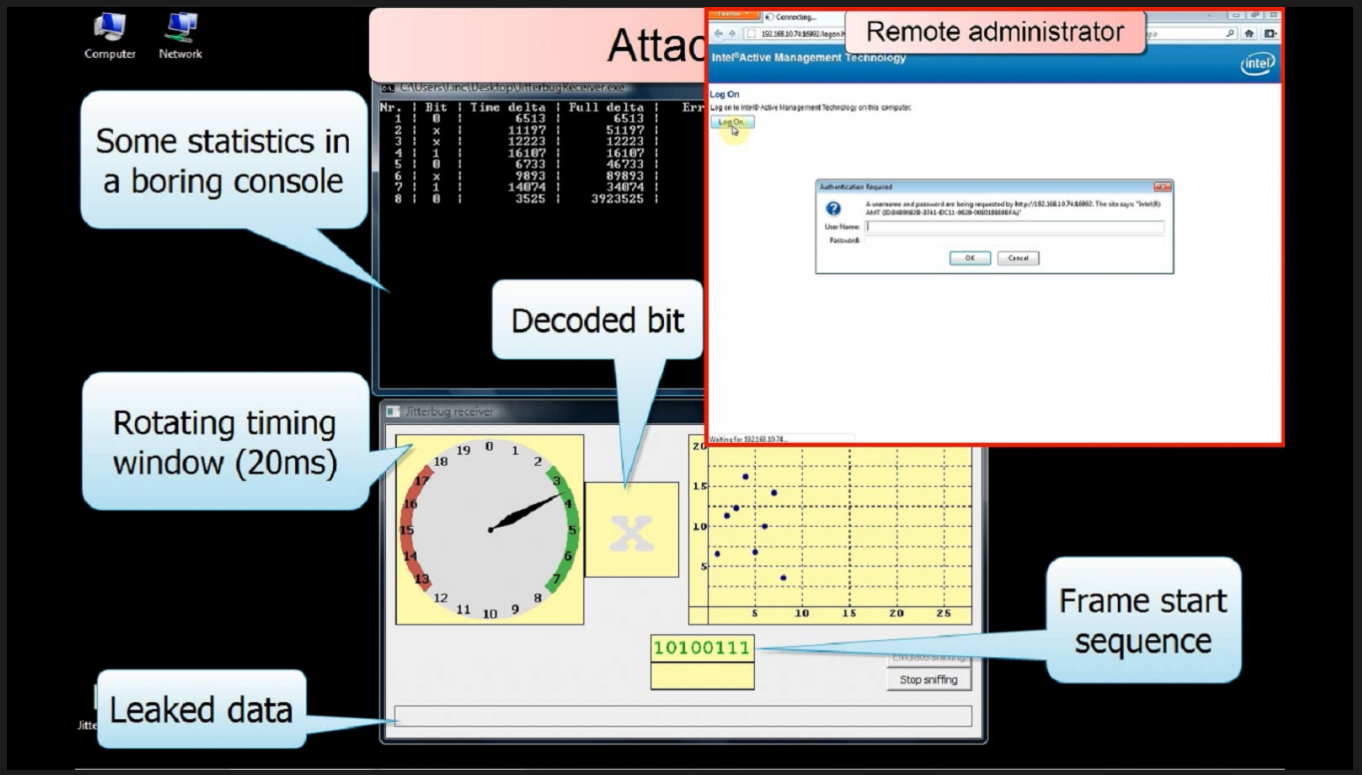

| No. | Time | Protocol Info |
|---|---|---|
| 1 | 0.000000 | TCP amt-soap-http > 7512 [SYN, ACK] |
| 2 | 0.001725 | TCP amt-soap-http > 7512 [ACK] |
| 3 | 0.002169 | TCP amt-soap-http > 7512 [ACK] |
| 4 | 0.207100 | TCP amt-soap-http > 7512 [PSH,ACK] |
| 5 | 0.209416 | TCP amt-soap-http > 7512 [PSH,ACK] |
| 6 | 0.214836 | TCP amt-soap-http > 7512 [PSH,ACK] |
| 7 | 13.125414 | TCP amt-soap-http > 7512 [FIN,PSH,ACK] |

**Wireshark log of an AMT TCP session**

# [patrickx@44con:~$] cat 'Execution Stages'

| No. | Description | Duration | Overhead |
|---|---|---|---|
| 1. | Find keyboard buffer | 100-110 ms | AMT irresponsive |
| 2. | Log sensitive information (e.g., detect keystrokes following a login name) | determined by user input | insignificant |
| 3. | Leak sensitive information (encode into legitimate packet delays) | unlimited, continuous replay | low, but detectable |

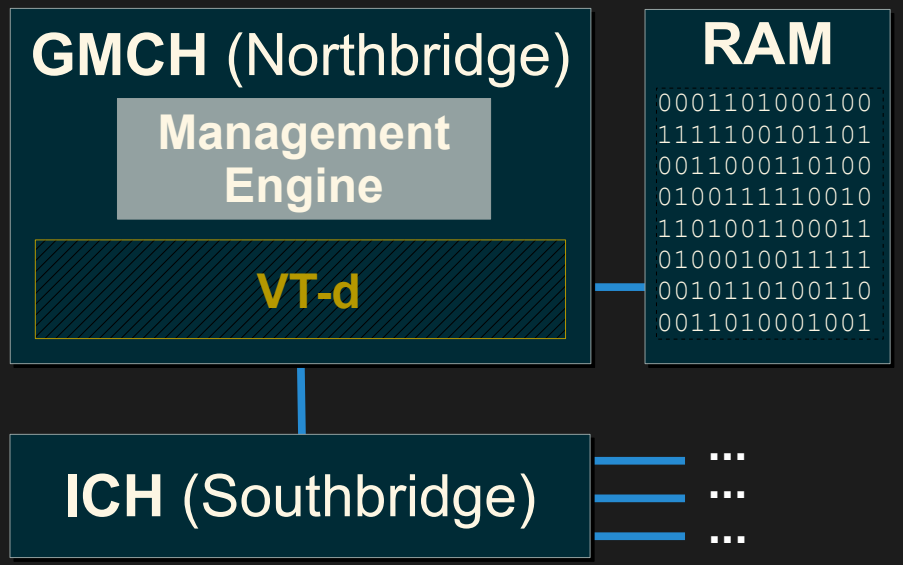
# Demo Video 3
## *JitterBug*




Final Remarks

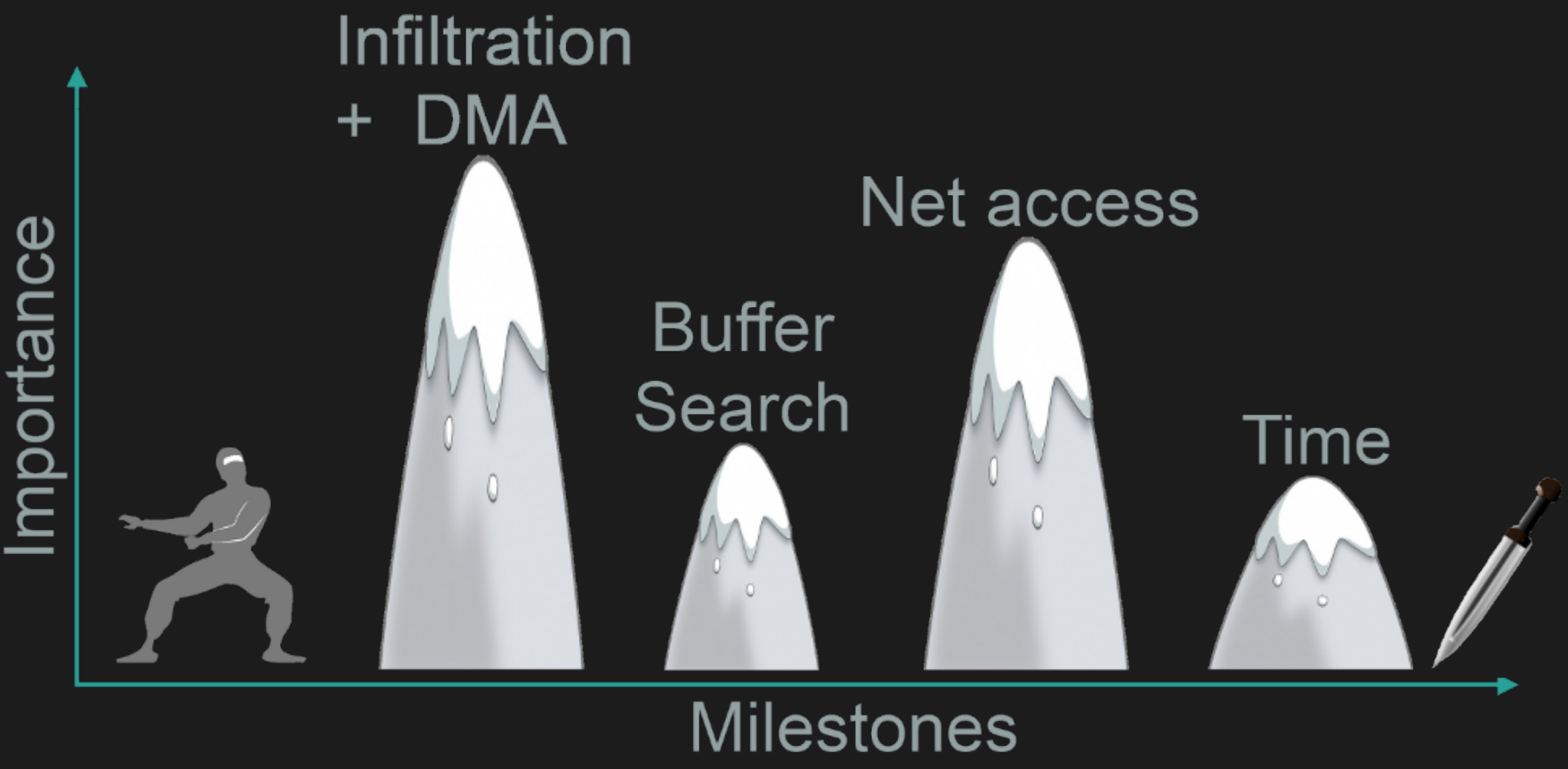
# [patrickx@44con:~$] cat 'Countermeasures'

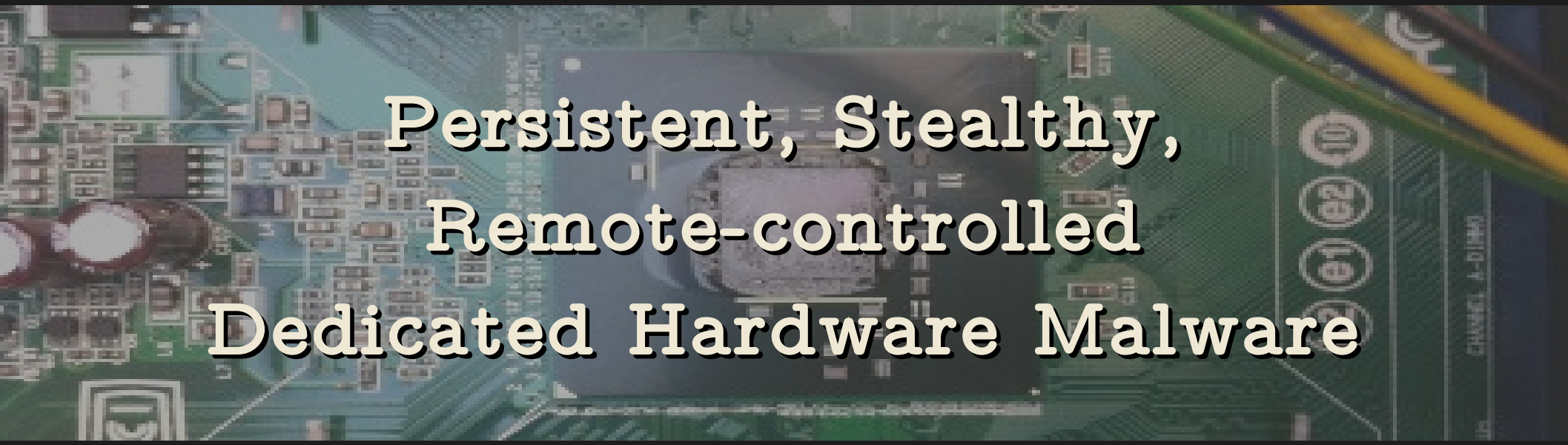
- $ **Virtualiztion Technology for Directed I/O (I/OMMU, [Abr06])**
  - $ Attacks: [San10], [Woj09], [Woj11a], [Woj11b]
  - $ No driver for Windows (including 8)
- $ **Academic:**
  - $ VIPER - Verifying the integrity of peripherals' firmware [Li11]
  - $ NAVIS - Network Adapter Verification and Integrity checking Solution [Duf11]
  - $ BARM - Bus Agent Runtime Monitor [Ste13]




Conclusion

# [patrickx@44con:~$] cat 'Conclusion'





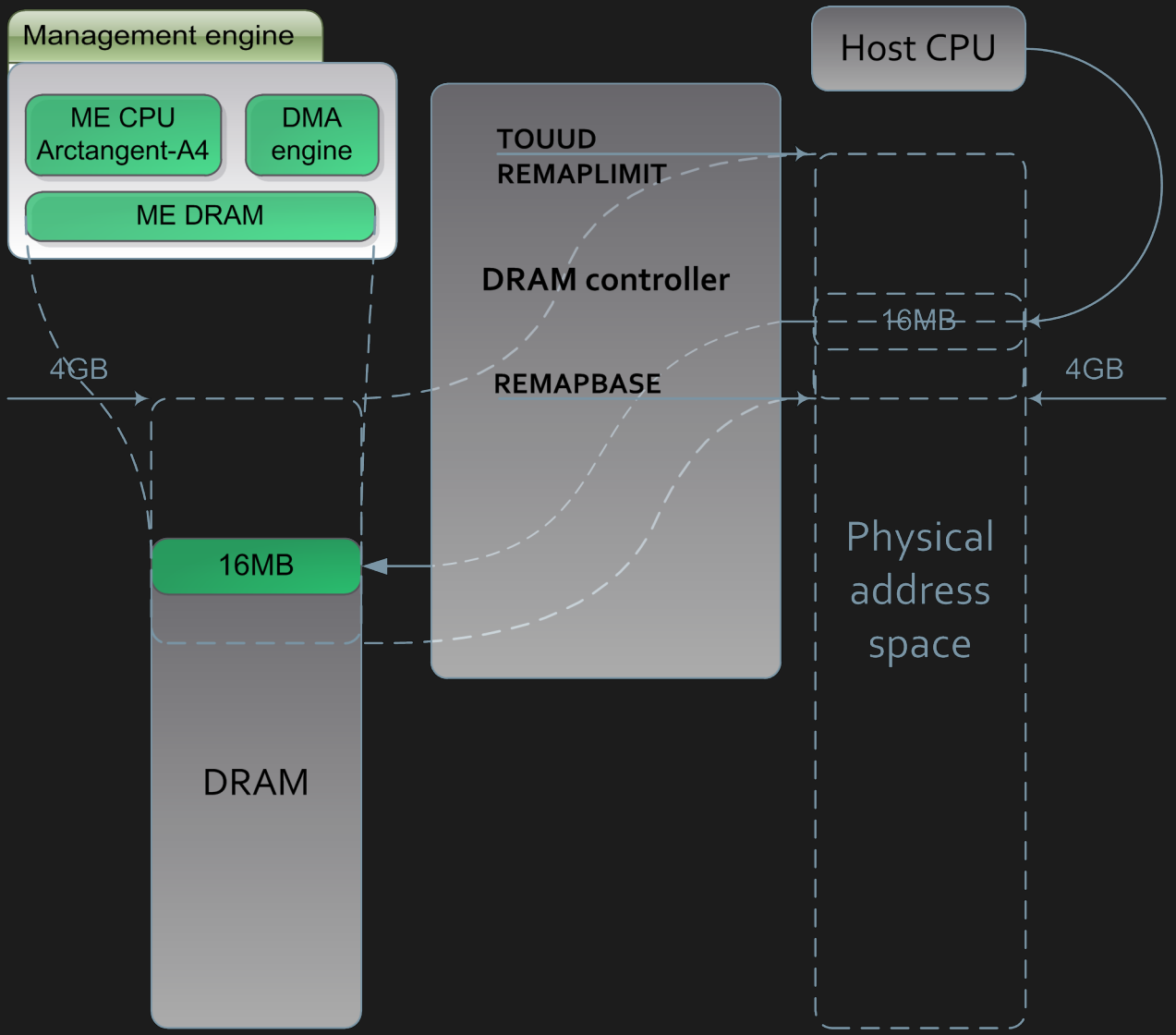
# Persistent, Stealthy, Remote-controlled Dedicated Hardware Malware

**Patrick Stewin and Iurii Bystrov**
Security in Telecommunications (SecT)
TU Berlin
patrickx@sec.t-labs.tu-berlin.de


**44CON 2013, London, UK**


BACKUP

# [patrickx@44con:~$] cat 'Memory Reclaiming'



# [patrickx@44con:~$] cat 'References/Related Work'


[Abr06] D. Abramson, J. Jackson, S. Muthrasanallur, G. Neiger, G. Regnier, R. Sankaran, I. Schoinas, R. Uhlig, B. Vembu, and J. Wiegert: Intel Virtualization Technology for Directed I/O

[Aum10] D. Aumaitre and C. Devine: Subverting Windows 7 x64 Kernel with DMA attacks

[Boi06] A. Boileau: Hit by a Bus: Physical Access Attacks with Firewire.

[Bul08] Y. Bulygin: Chipset based Approach to detect Virtualization Malware.

[Del10] G. Delugre: Closer to metal: Reverse engineering the Broadcom NetExtreme's firmware

[Del11] G. Delugre. How to develop a rootkit for Broadcom NetExtreme network cards

[Dor04] M. Dornseif: 0wned by an iPod - hacking by Firewire.

[Dor05] M. Dornseif, M. Becher, and C. N. Klein: FireWire - all your memory are belong to us

[Duf10] L. Duflot, Y.-A. Perez, G. Valadon, and O. Levillain: Can you still trust your network card?

# [patrickx@44con:~$] cat 'References/Related Work'


[Duf11] L. Duflot, Y.-A. Perez, and B. Morin: What if you can't trust your network card?

[Int07] Intel Corporation: Intel Core 2 Duo Processor and Intel Q35 Express Chipset Development Kit

[Kum09] A. Kumar, P. Goel and Y. Saint-Hilaire: Active Platform Management Demystified – Unleashing the power of Intel vPro Technology

[Li11] Y. Li, J. M. McCune, and A. Perrig: VIPER: Verifying the integrity of peripherals' firmware

[May05] D. Maynor: DMA: Skeleton key of computing && selected soap box rants

[San10] F. Sang, E. Lacombe, V. Nicomette, and Y. Deswarte: Exploiting an I/OMMU vulnerability

[Sha06] G. Shah, A. Molina and M. Blaze: Keyboards and Covert Channels

[Sko12] I. Skochinsky: Rootkit in your laptop: Hidden code in your chipset and how to discover what exactly it does

# [patrickx@44con:~$] cat 'References/Related Work'


[Ste12] P. Stewin and I. Bystrov. Understanding DMA Malware

[Ste13] P. Stewin: A Primitive for Revealing Stealthy Peripheral-Based Attacks on the Computing Platform's Main Memory

[Ter09] A. Tereshkin and R. Wojtczuk: Introducing Ring -3 Rootkits

[Tri08] A. Triulzi: Project Maux Mk.II.

[Tri10] A. Triulzi: The Jedi Packet Trick takes over the Deathstar

[Woj09] R. Wojtczuk, J. Rutkowska, and A. Tereshkin: Another Way to Circumvent Intel Trusted Execution Technology

[Woj11a] R. Wojtczuk,, and J. Rutkowska: Attacking Intel TXT via SINIT code execution hijacking

[Woj11b] R. Wojtczuk, and J. Rutkowska: Following the White Rabbit: Software attacks against Intel VT-d technology