

# Persistent, Stealthy, Remote-controlled Dedicated Hardware Malware

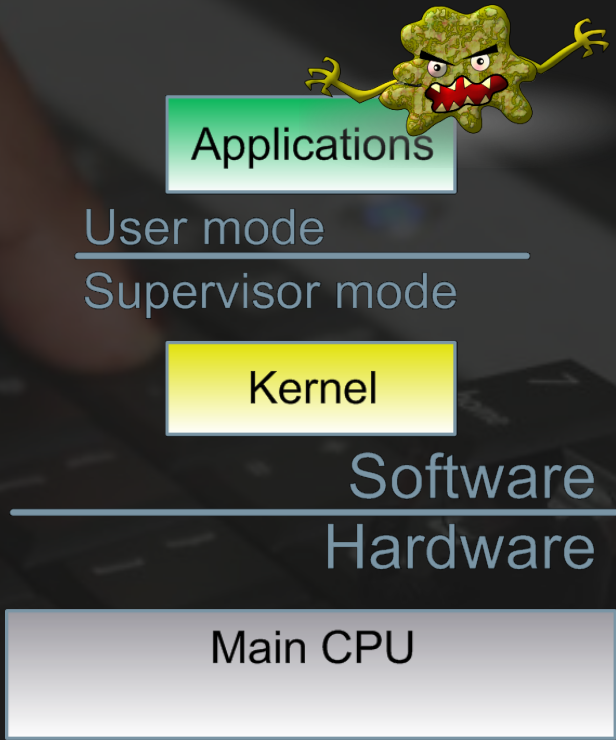
Patrick Stewin and Iurii Bystrov  
Security in Telecommunications (SecT)  
TU Berlin  
[patrickx@sec.t-labs.tu-berlin.de](mailto:patrickx@sec.t-labs.tu-berlin.de)



44CON 2013, London, UK

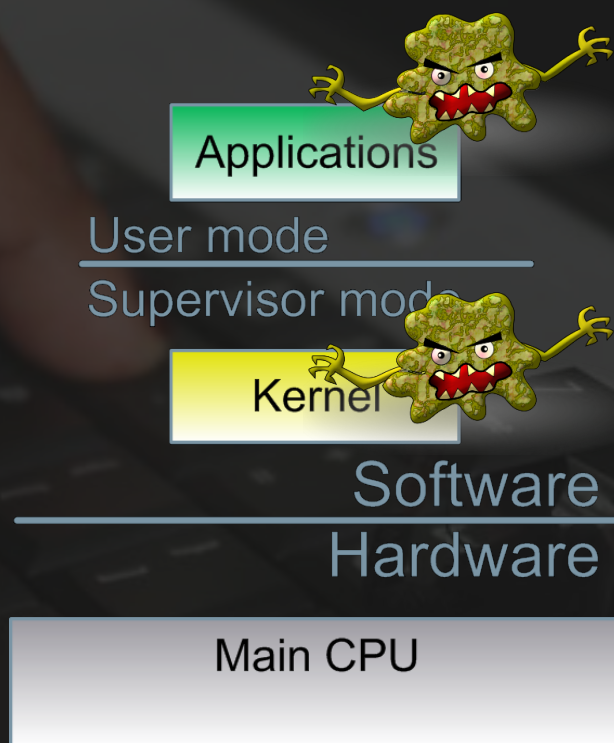
# ARMS RACE

Malware developers ↔ anti-malware community



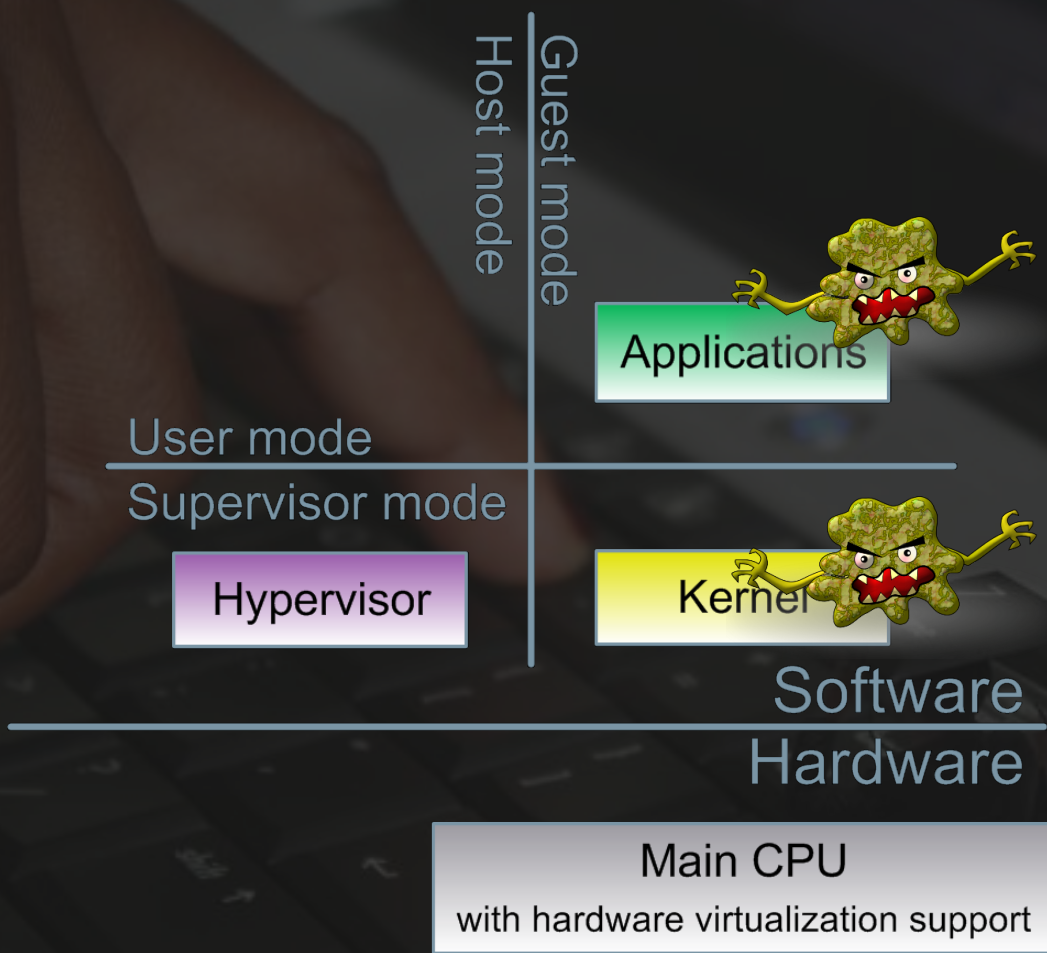
# ARMS RACE

Malware developers ↔ anti-malware community



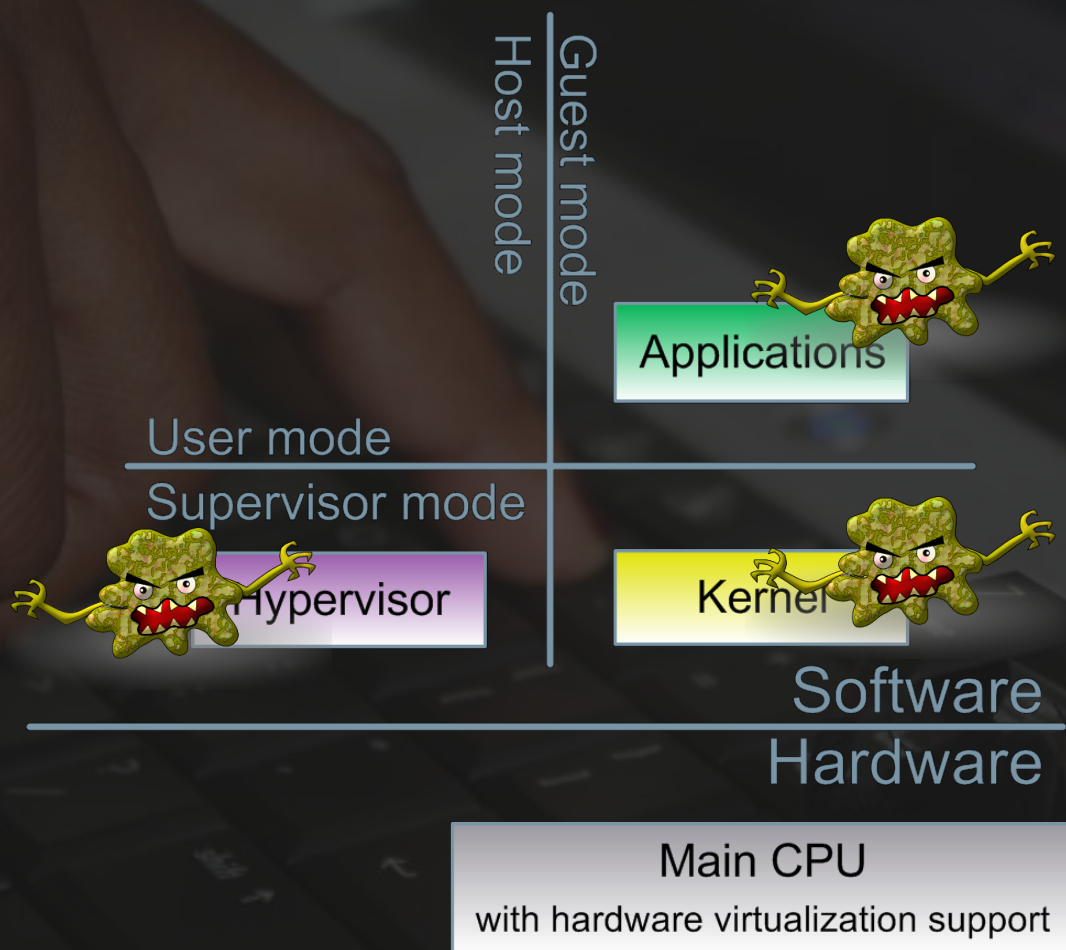
# ARMS RACE

Malware developers ↔ anti-malware community



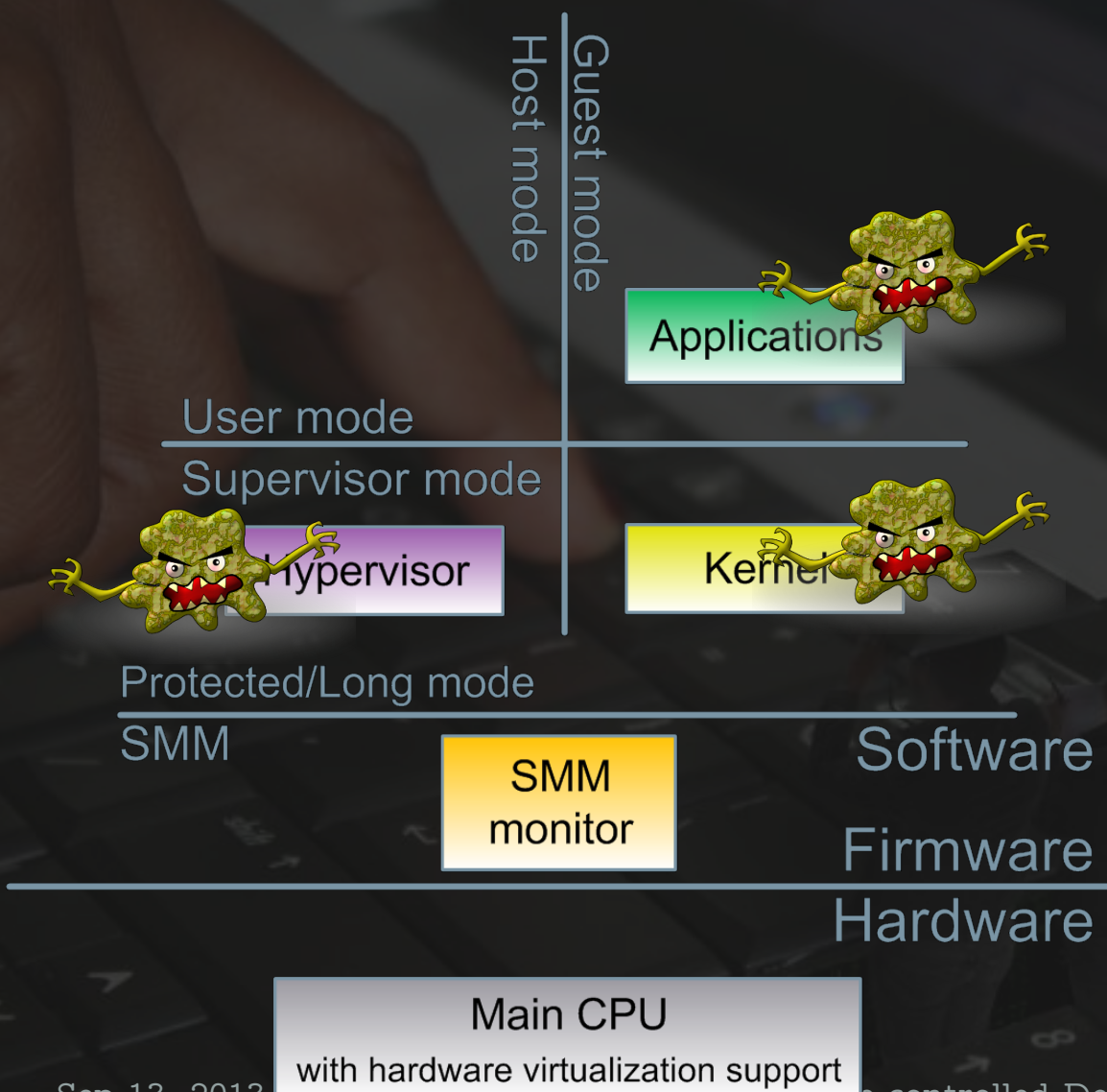
# ARMS RACE

Malware developers ↔ anti-malware community



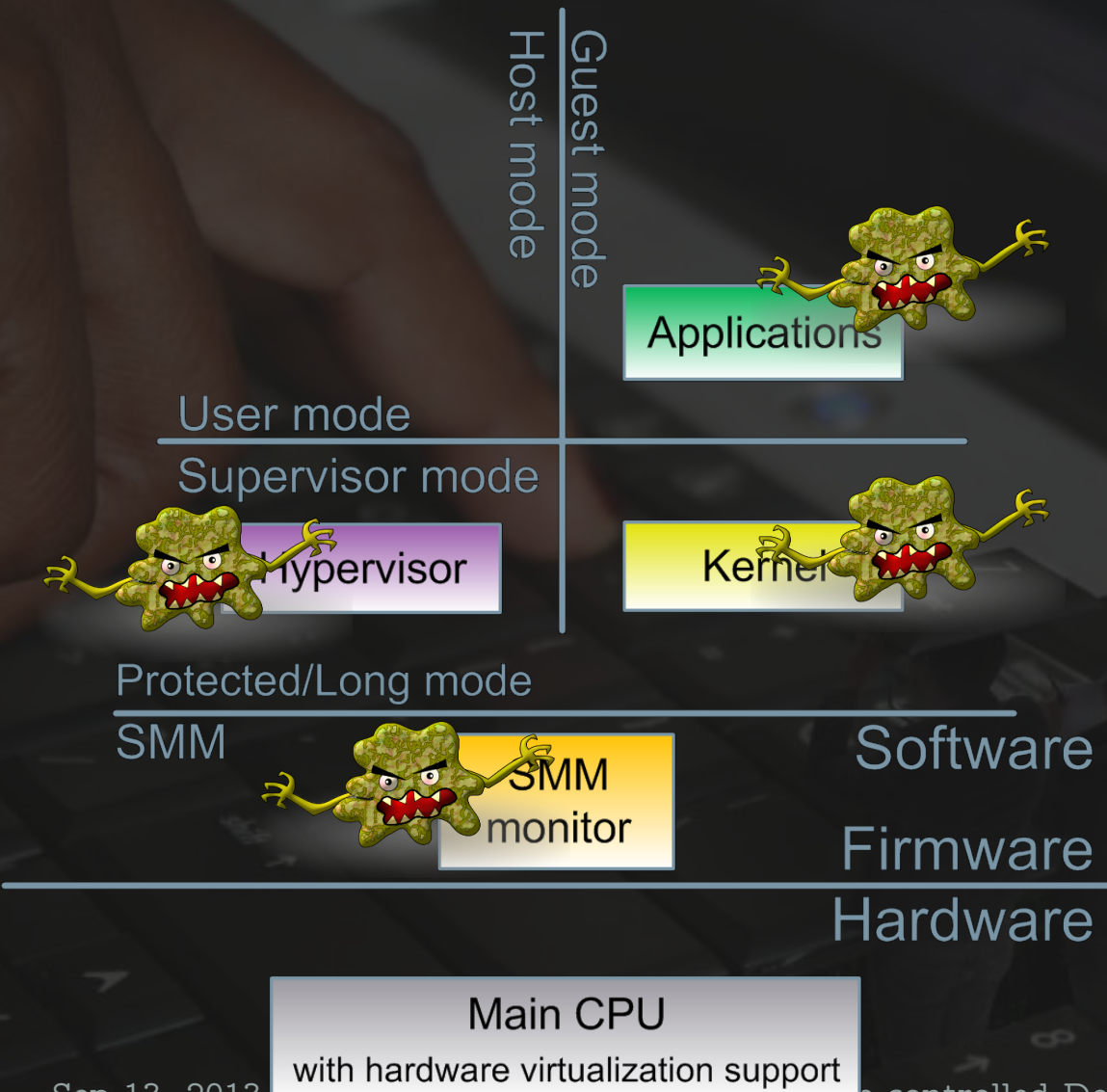
# ARMS RACE

## Malware developers ↔ anti-malware community



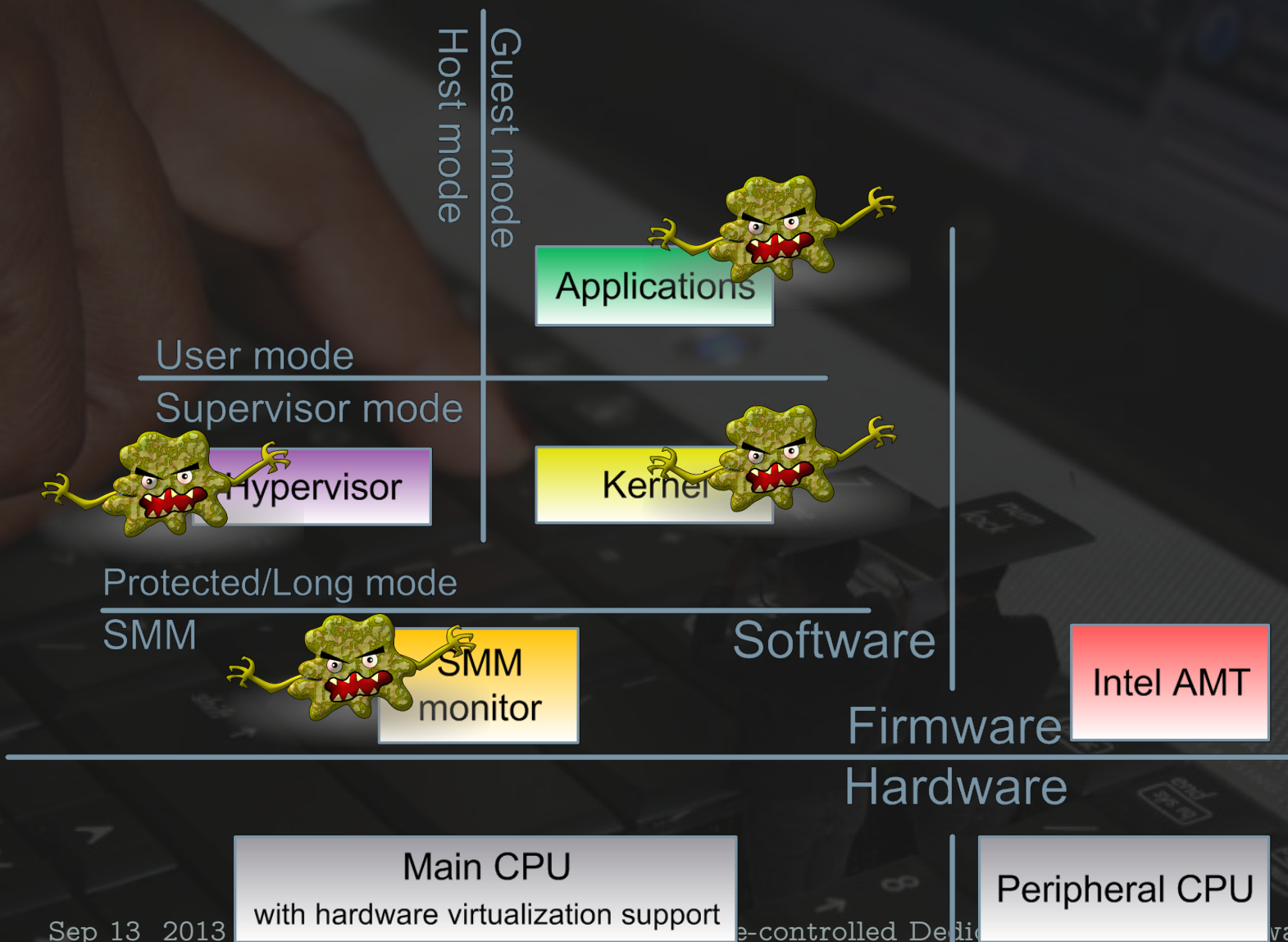
# ARMS RACE

Malware developers ↔ anti-malware community



# ARMS RACE

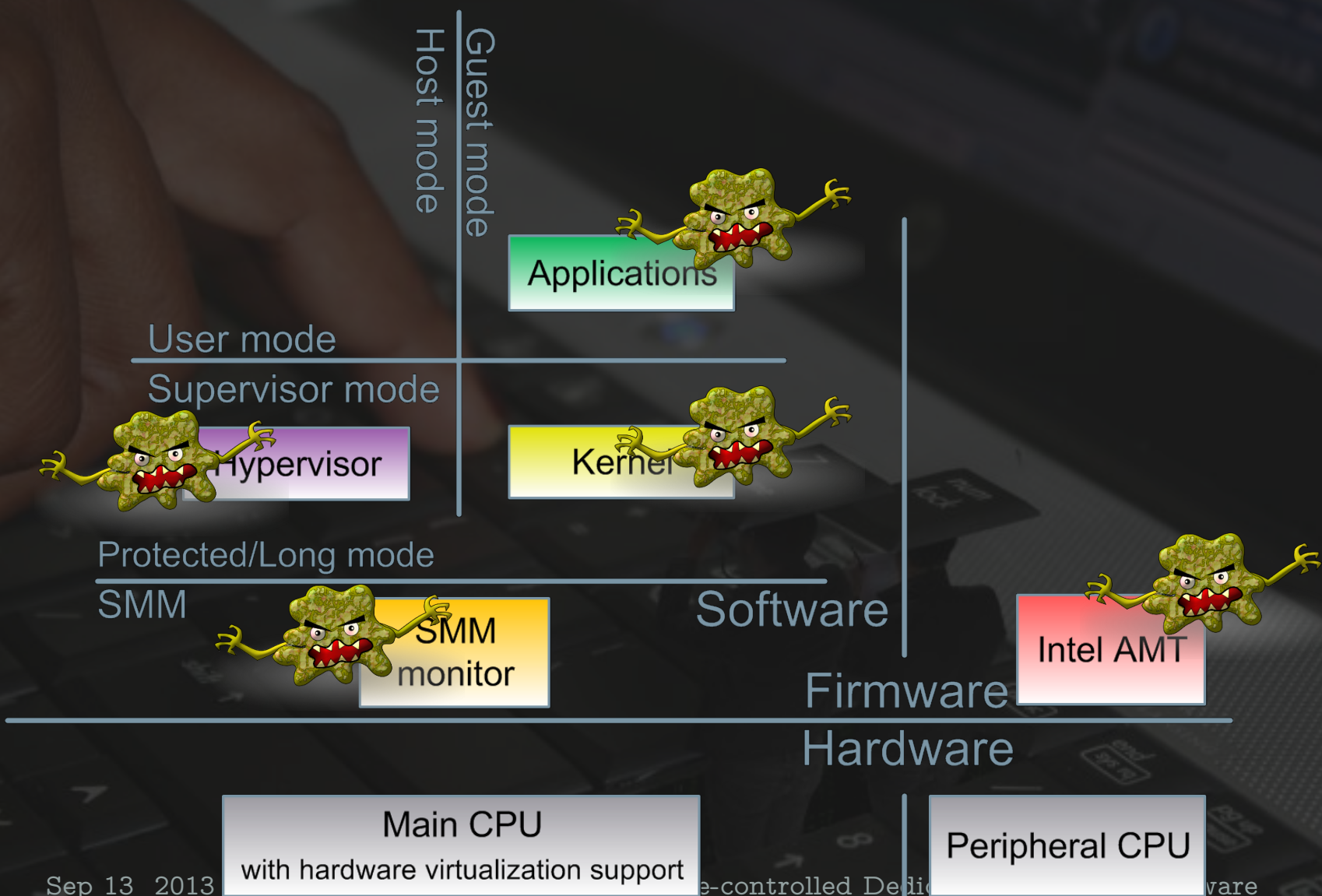
Malware developers ↔ anti-malware community





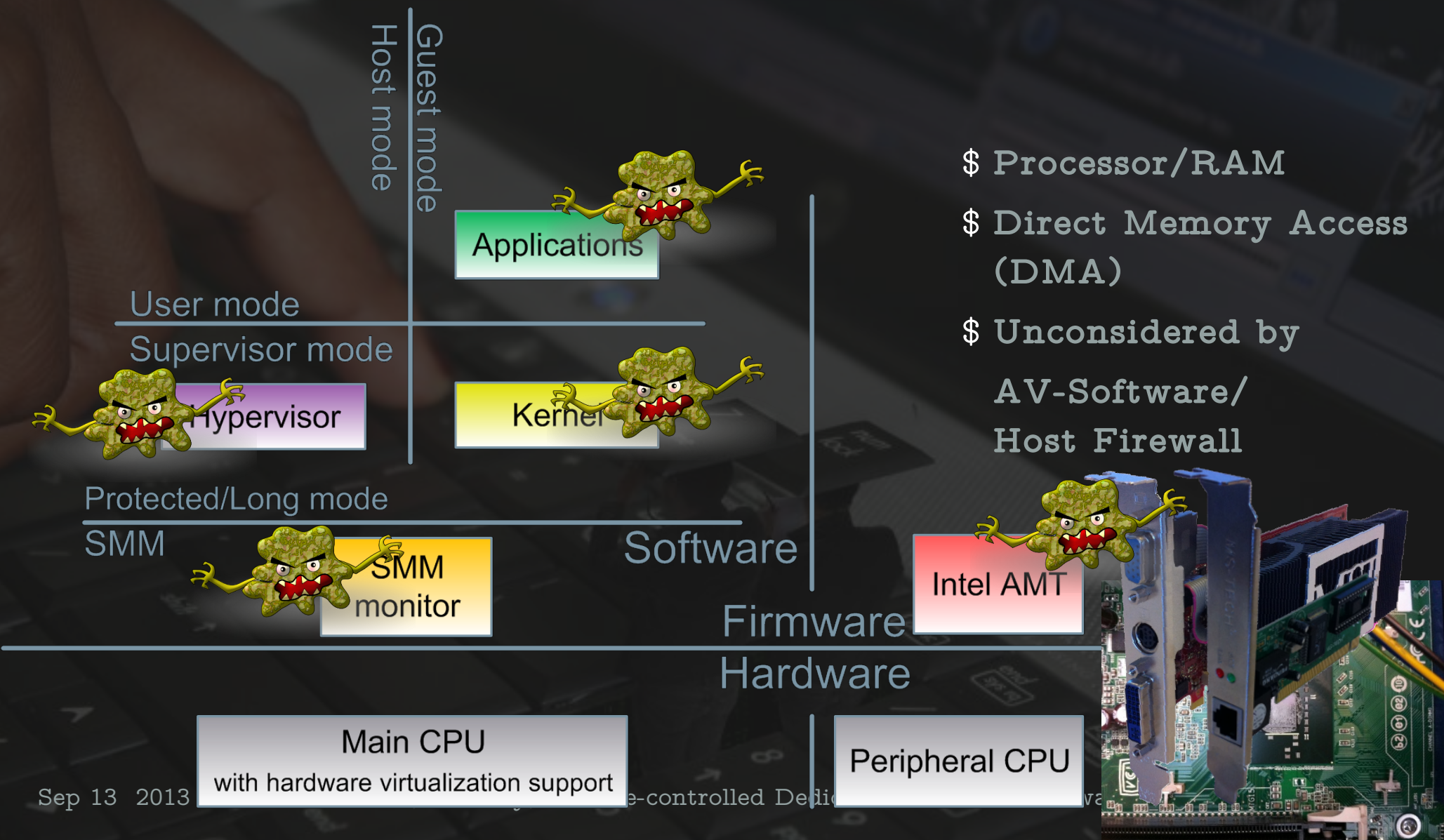
# ARMS RACE

## Malware developers ↔ anti-malware community



# ARMS RACE

Malware developers ↔ anti-malware community




```
[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

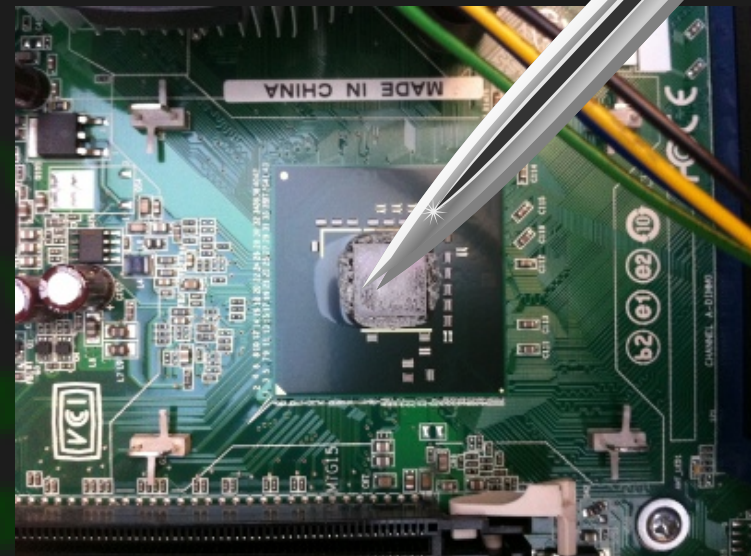
\$ Not only a keylogger anymore

\$ Access to host memory  
(DMA read/write)

\$ Isolated network channel

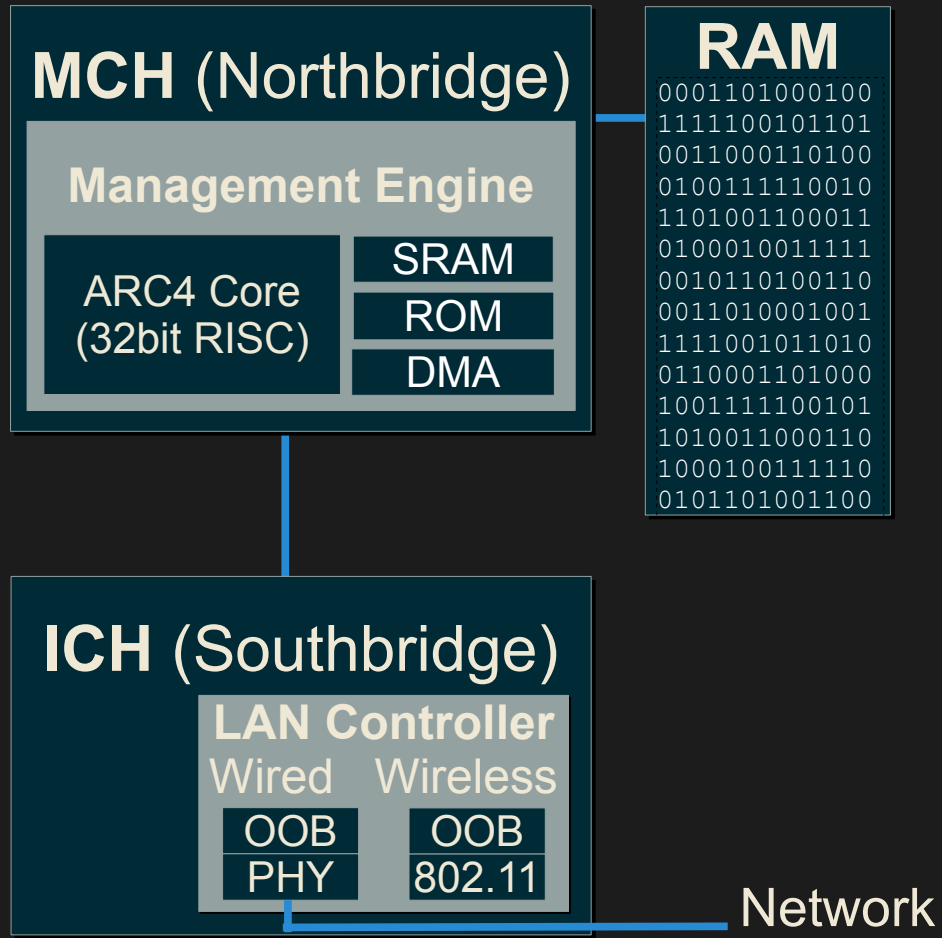
\$ 32bit/64bit based  
attack targets

\$ ...



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

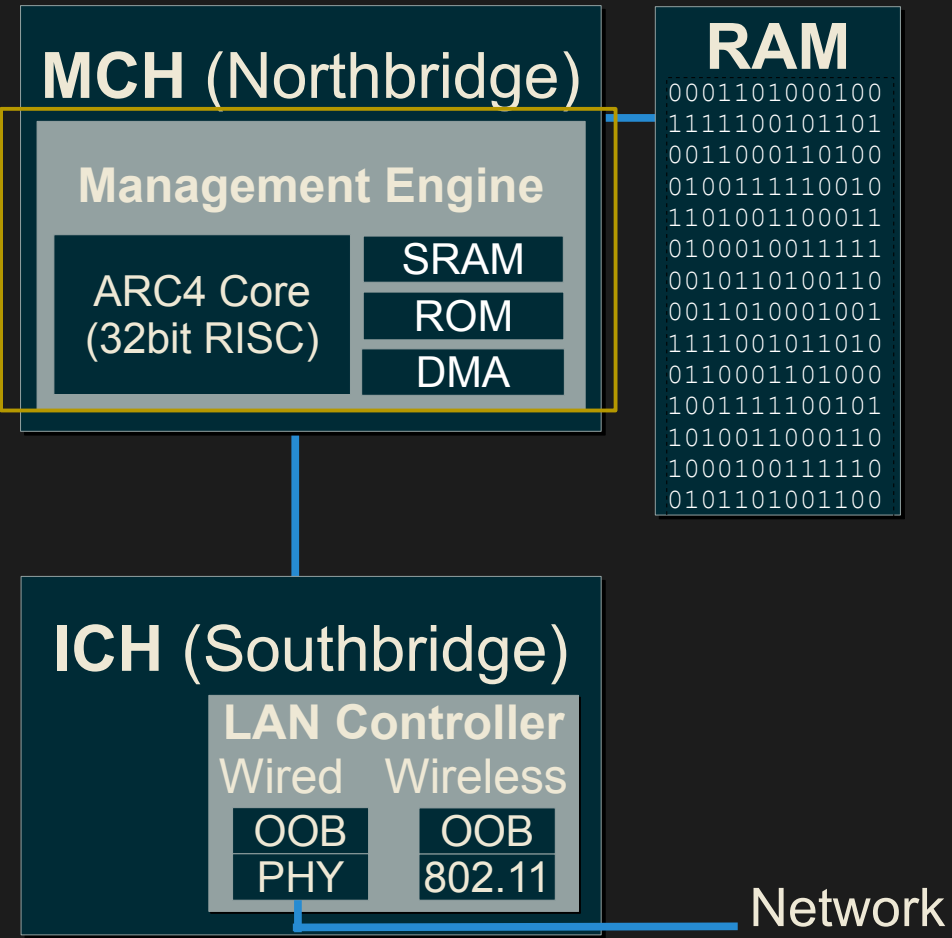
# \$ Manageability Engine



(Q35 Chipset)

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

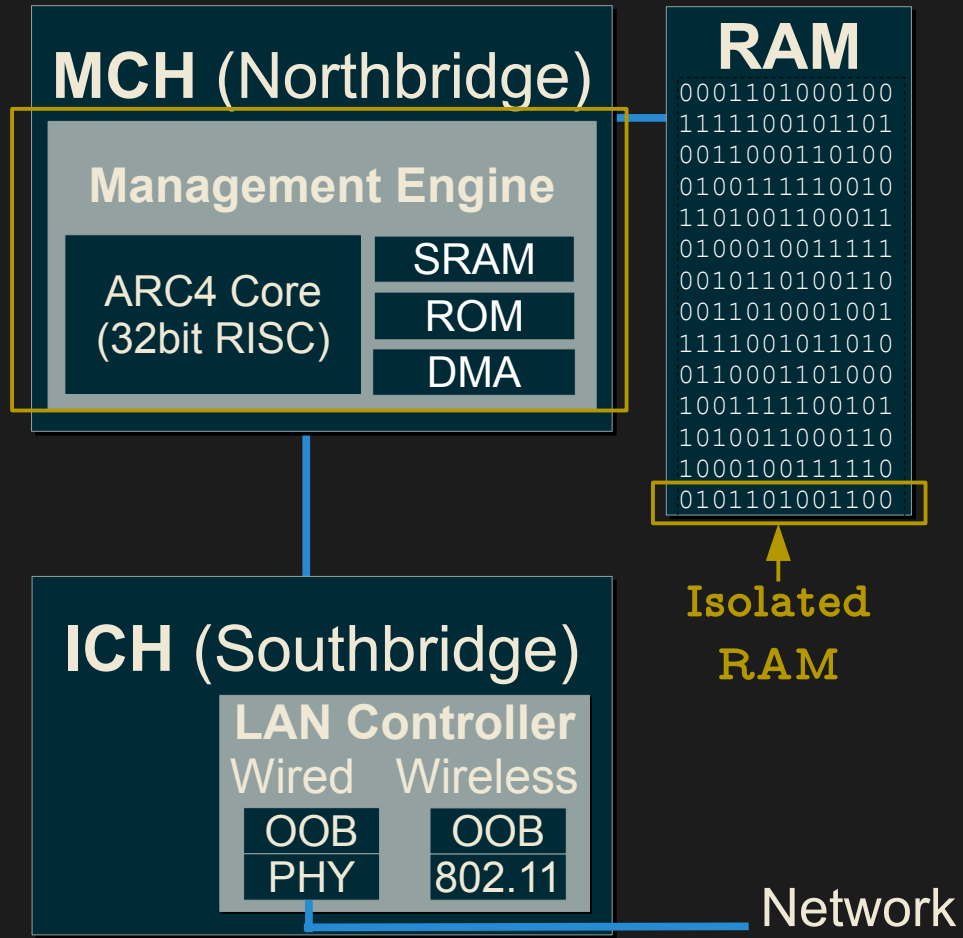
# \$ Manageability Engine



(Q35 Chipset)

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

# \$ Manageability Engine

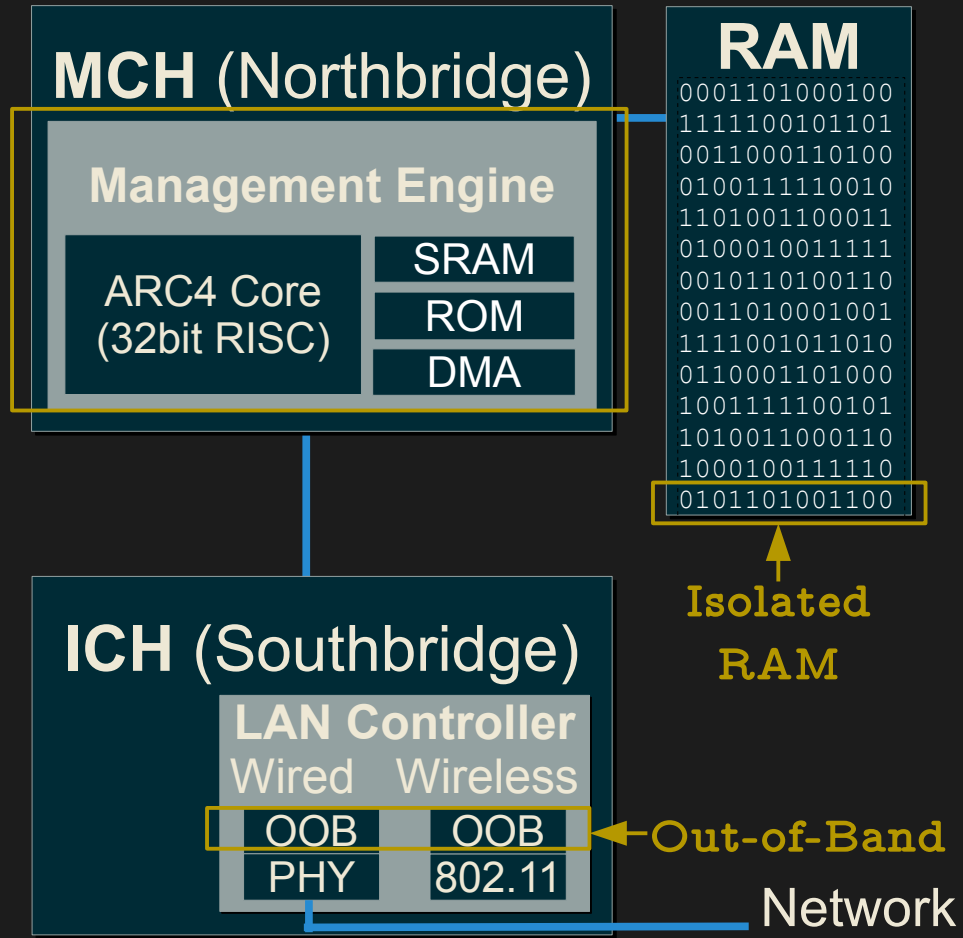


(Q35 Chipset)



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

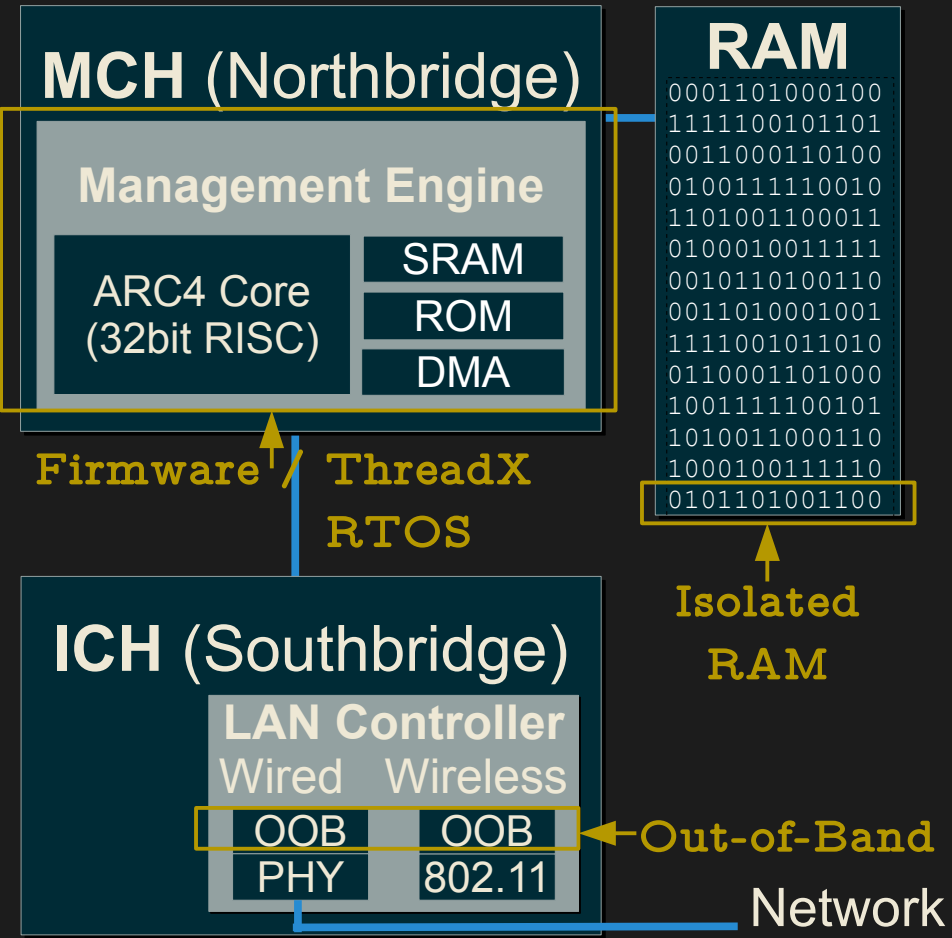
# \$ Manageability Engine



(Q35 Chipset)

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

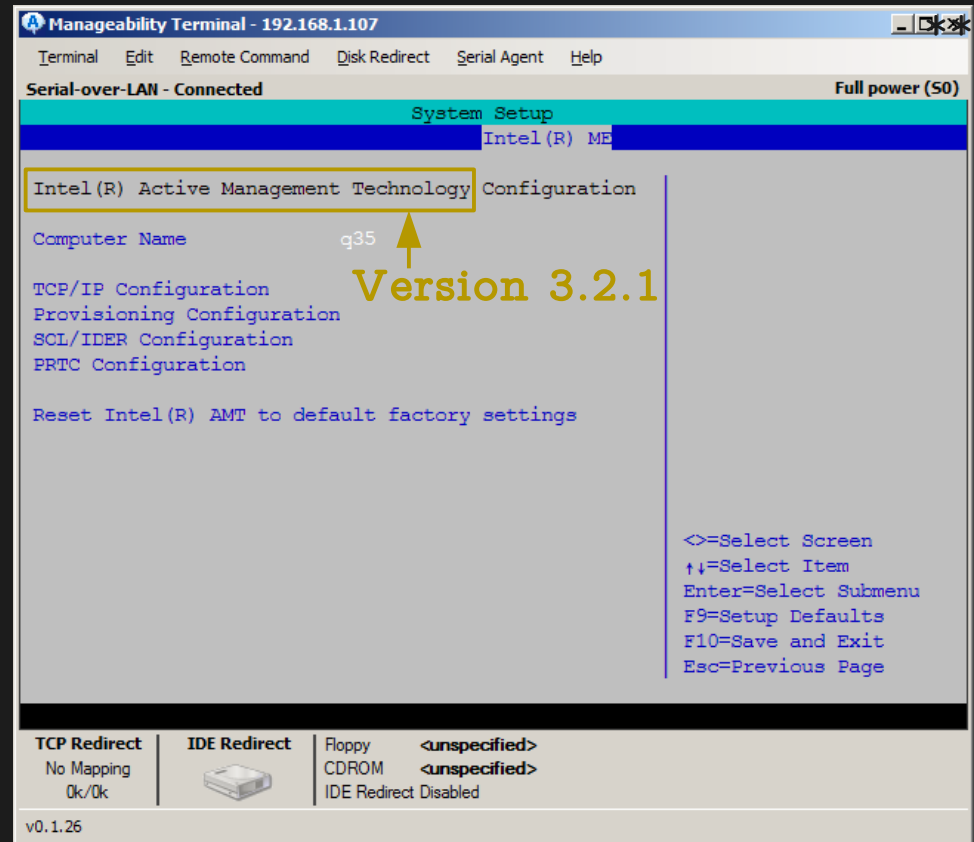
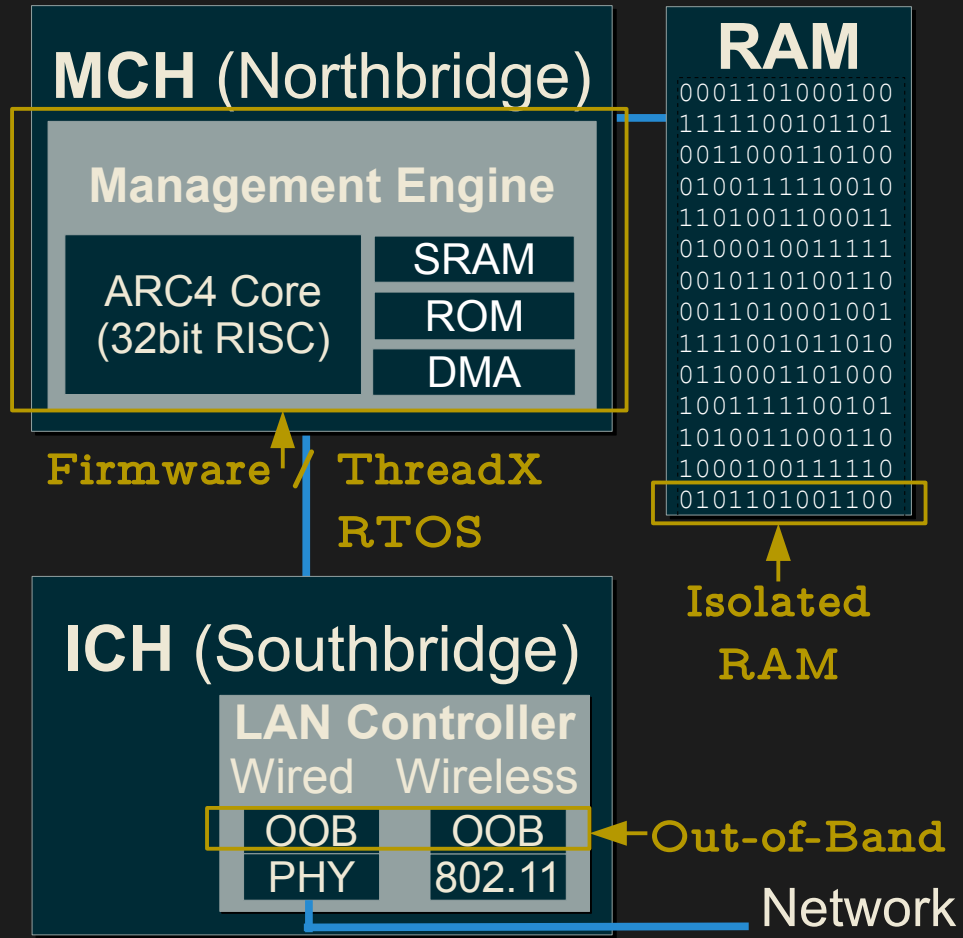
# \$ Manageability Engine



(Q35 Chipset)

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

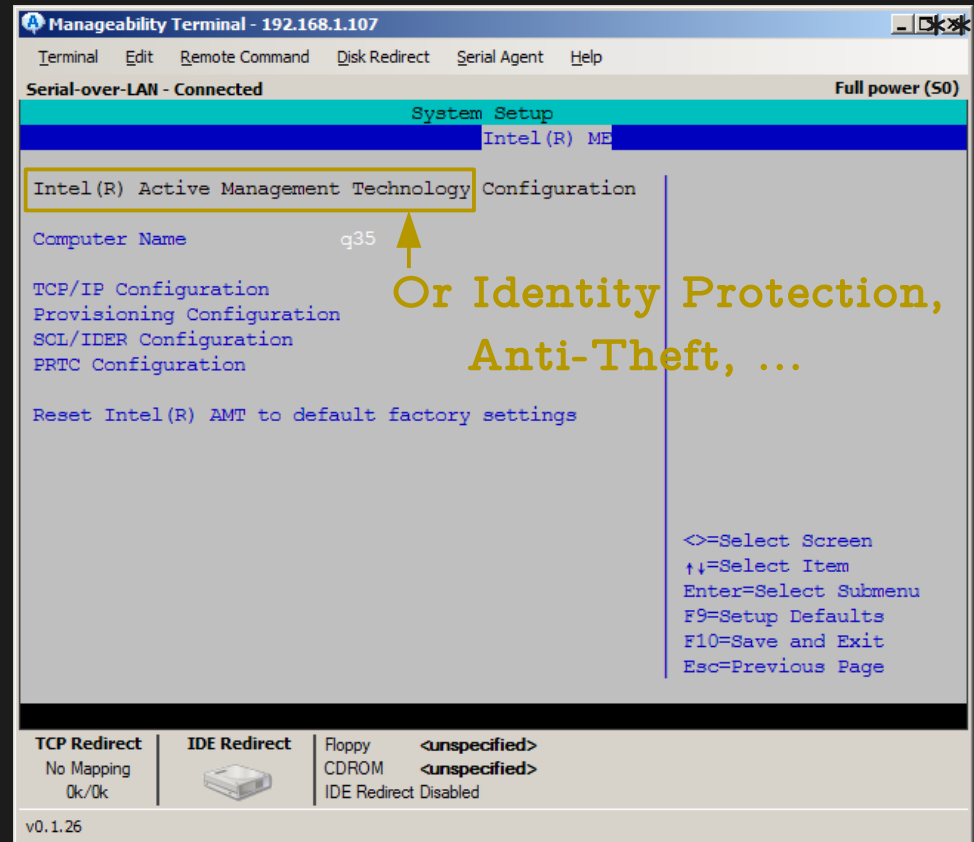
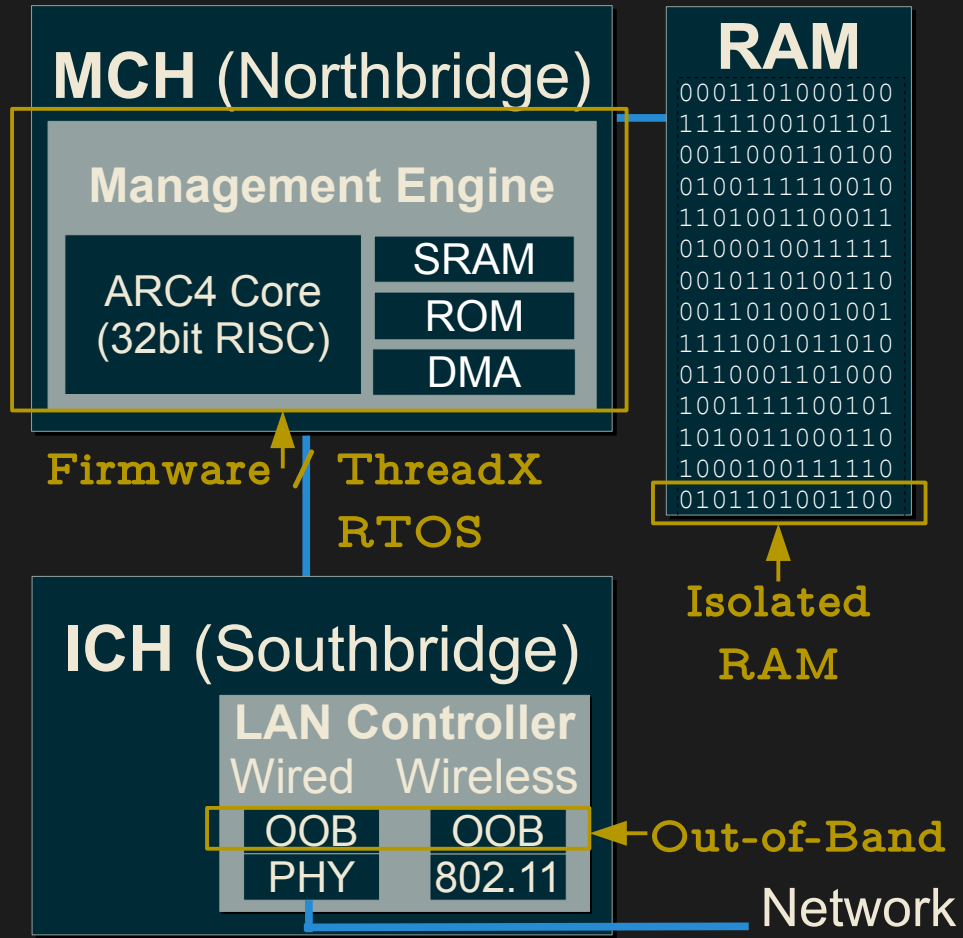
# \$ Manageability Engine



(Q35 Chipset)

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

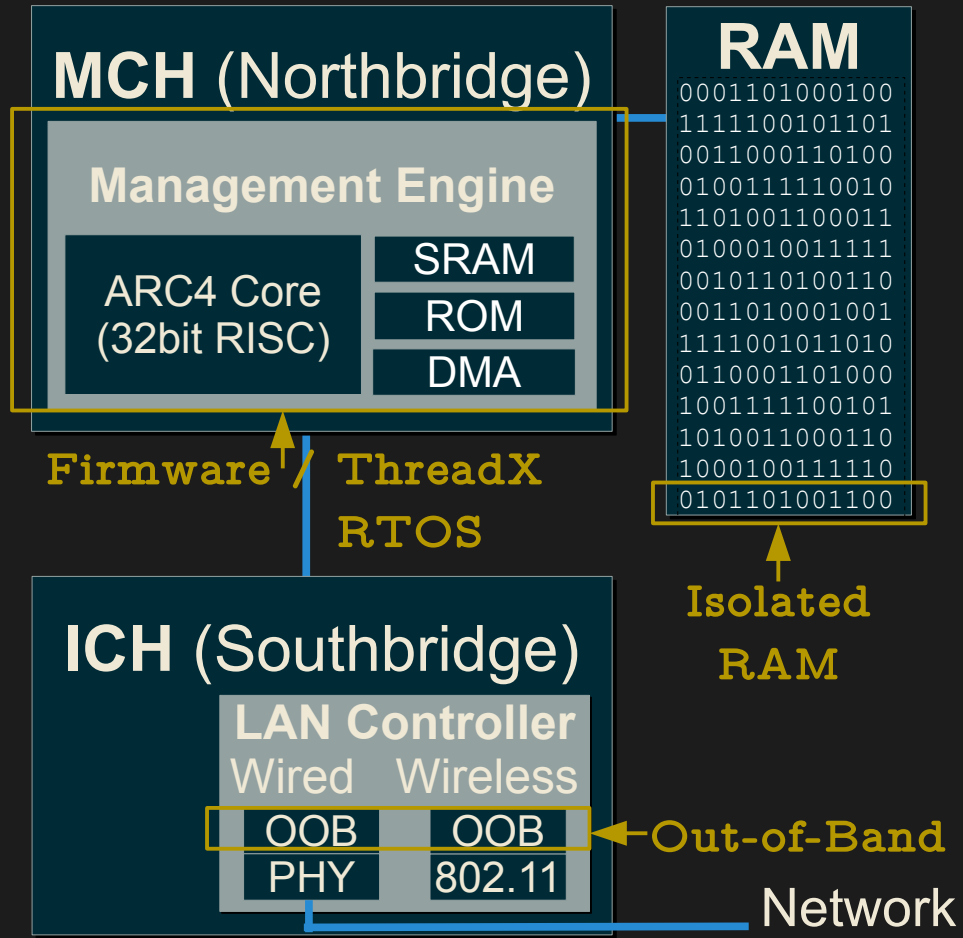
# \$ Manageability Engine



(Q35 Chipset)

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

# \$ Manageability Engine



(Q35 Chipset)

## ARC Historical Overview\*

# Mathematical, Argonaut, Rotation & I/O: MARIO chip :)

# SuperFX

# ARC

# 1<sup>st</sup> ME generation:  
ARCTangent-A4/  
ARC4

# 2<sup>nd</sup> ME generation:  
ARCTangent-A5/  
ARCompact  
→ see [Sko12] !

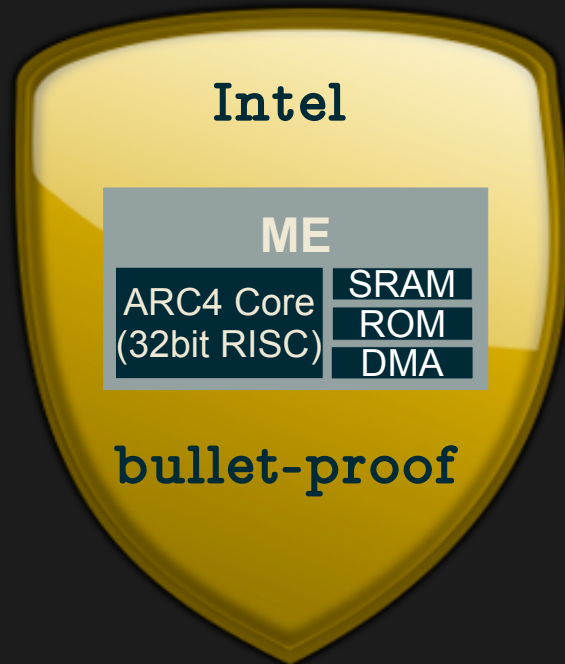


\*Details: [http://en.wikipedia.org/wiki/ARC\\_International](http://en.wikipedia.org/wiki/ARC_International)

\*\*[http://en.wikipedia.org/wiki/File:MARIO\\_CHIP\\_1\\_Starwing.jpg](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](http://www.youtube.com/watch?v=k8dxLr_xVv4) [0:21:44]

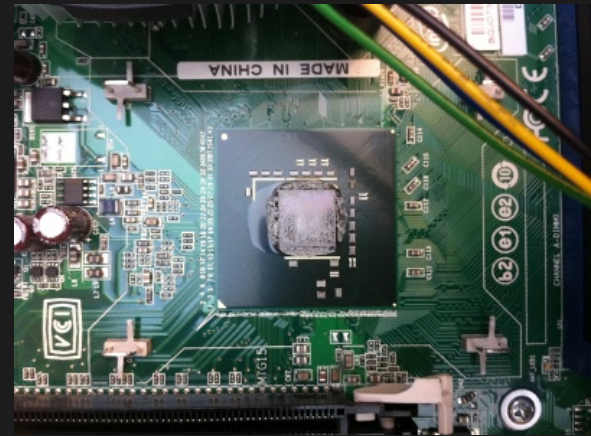
```
[patrickx@44con:~$] cat 'Our Attack Environment'
```



- \$ 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

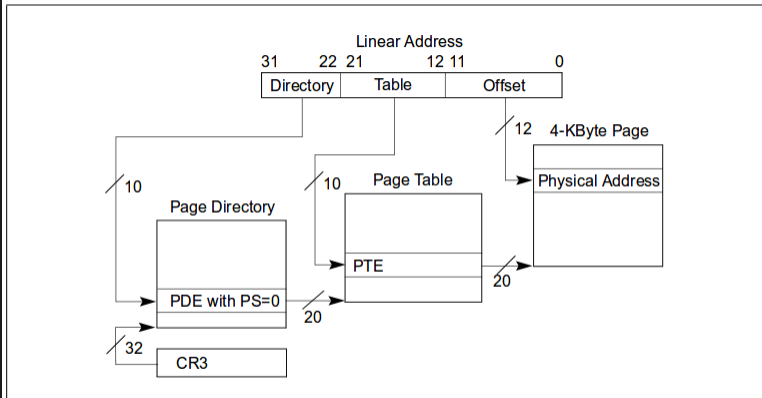


Figure 4-2. Linear-Address Translation to a 4-KByte Page using 32-Bit Paging

Intel SDM

```

C:\Windows\system32\cmd.exe
*****
ASLR Process Scanner v1.5 by SecurityXploded
http://securityxploded.com/aslr-process-scanner.php
*****

:: Showing ASLR Enabled Processes...

Process Name          PID      Session ID  Username  Process Path
-----
Taskhost.exe          4828     1           patrickx  C:\Windows\system32\taskhost.exe
Dwm.exe                4948     1           patrickx  C:\Windows\system32\Dwm.exe
Explorer.exe           5044     1           patrickx  C:\Windows\Explorer.EXE

```



[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:

### USB Request Block Structure

```

:
struct usb_device *dev
:
:
:
dma_addr_t transfer_dma
:

```

A vertical double-headed arrow labeled "Constant offset" spans the distance between the pointer field `struct usb_device *dev` and the field `dma_addr_t transfer_dma`.

### USB Device Structure

```

:
:
:
char *product
:

```

A vertical double-headed arrow labeled "Constant offset" spans the distance between an unknown field (indicated by a colon) and the field `char *product`.

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

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

\$ Signature scan:

### USB Request Block Structure

```

:
: struct usb_device *dev
:
:
:
: dma_addr_t transfer_dma
:

```



If pointer mod 0x400 == 0

If field mod 0x20 == 0

Start URB signature scan

1

&&

### USB Device Structure

```

:
:
: char *product
:

```



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

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

\$ Signature scan:

### USB Request Block Structure

```

:
: struct usb_device *dev
:
:
:
: dma_addr_t transfer_dma
:

```

Constant offset

Start URB signature scan 1

If pointer mod 0x400 == 0

&& 2

Check substrings "USB " and "Keyboard"

If field mod 0x20 == 0

### USB Device Structure

```

:
:
: char *product
:

```

Constant offset

If substrings "USB " and "Keyboard" found

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

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

\$ Signature scan:

### USB Request Block Structure

```

:
: struct usb_device *dev
:
:
:
: dma_addr_t transfer_dma
:
:

```

Constant offset

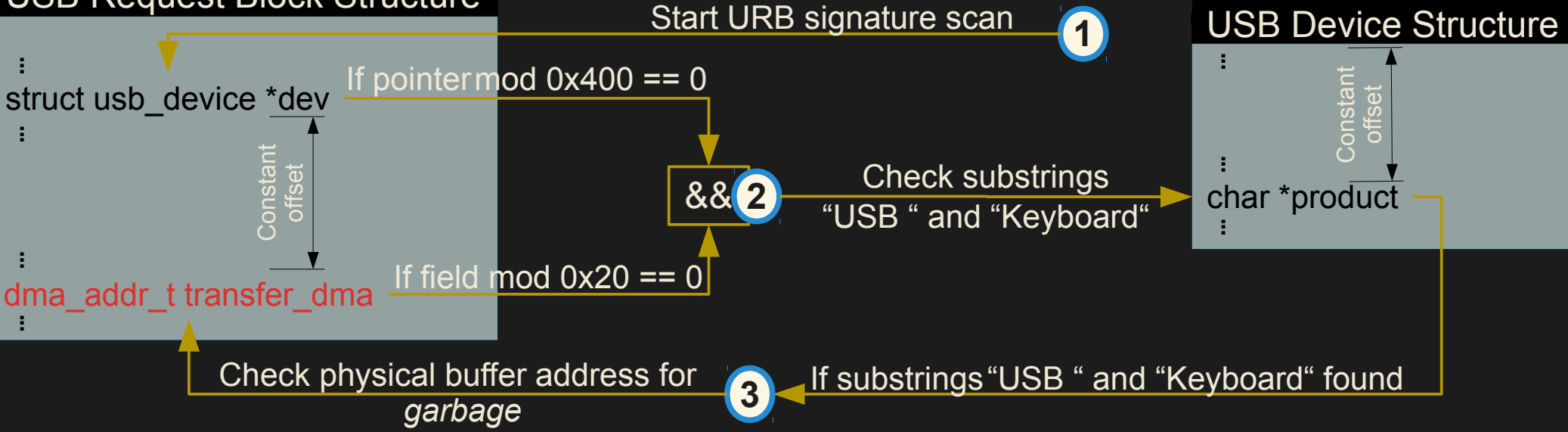
### USB Device Structure

```

:
:
: char *product
:
:

```

Constant offset



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

\$ Mapping virtual to physical memory addresses

\$ 32bit: subtract constant offset → 0xc0000000

\$ 64bit: see Documentation/x86/x86\_64/mm.txt

user space	0x0000000000000000
hole	0x00007fffffffffffff
guard hole	0xffff800000000000
all phys. memory	0xffff880000000000
hole	0xffffc80000000000
vmalloc/ioremap space	0xffffc90000000000
hole	0xffffe8ffffffffffff
virtual memory map	0xffffea0000000000
unused hole	0xffffeaffffffffffffff
kernel text mapping	0xfffffffff8000000
module mapping space	0xfffffffffa000000
	0xfffffffffffff00000

```
[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:
```

```
KiInitialPCR
```

```
⋮  
KdVersionBlock  
⋮
```

```
[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:**

```
KiInitialPCR  
⋮  
KdVersionBlock  
⋮
```



```
KdDebuggerDataBlock  
⋮  
ObpRootDirectoryObject
```

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

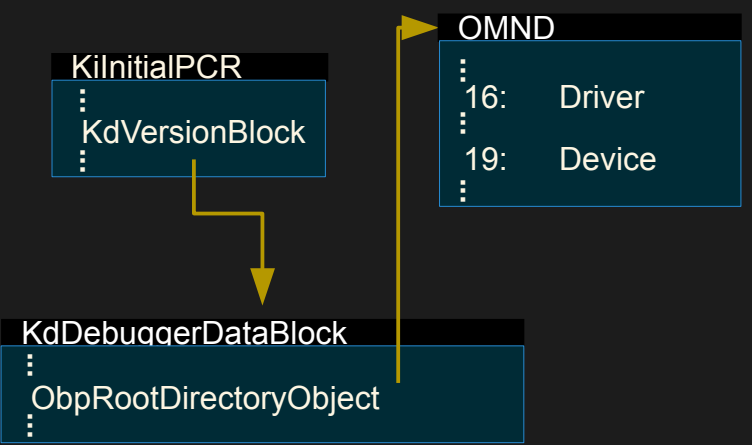
\$ Kernels tested: Vista / 7

\$ 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'

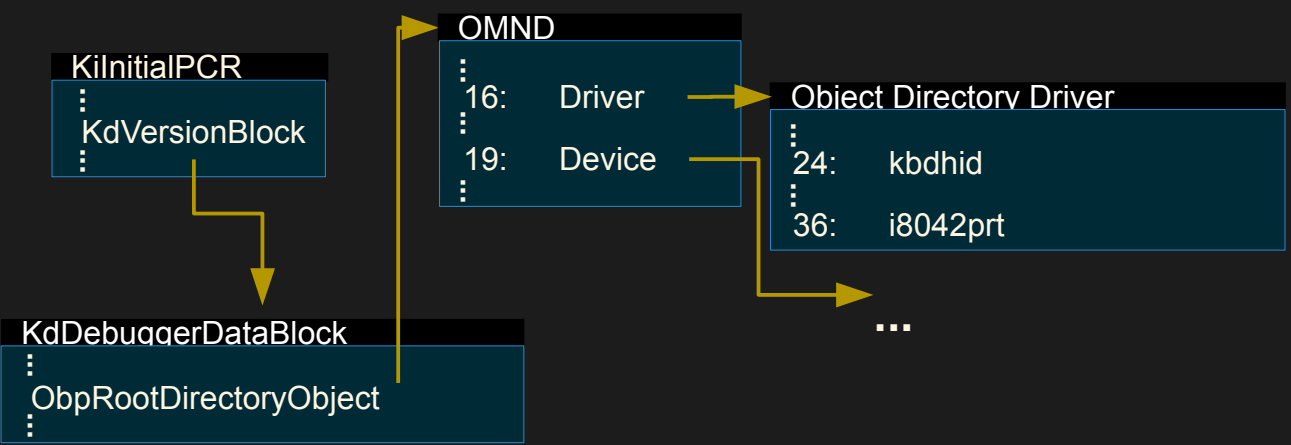
\$ Kernels tested: Vista / 7

\$ 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'
```

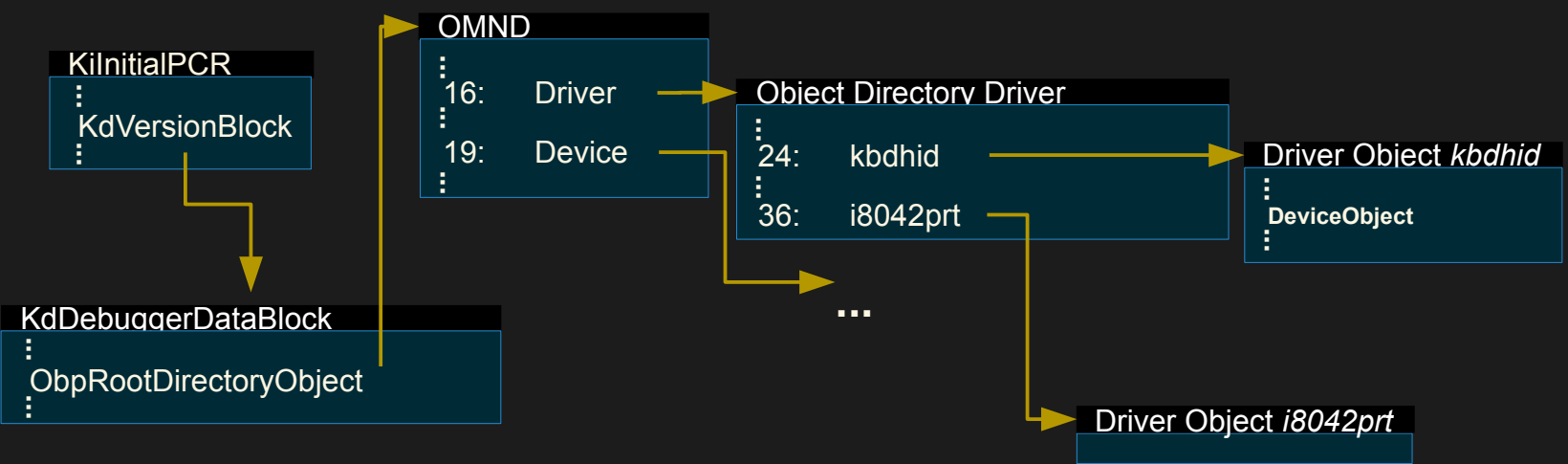
\$ Kernels tested: Vista / 7

\$ 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'

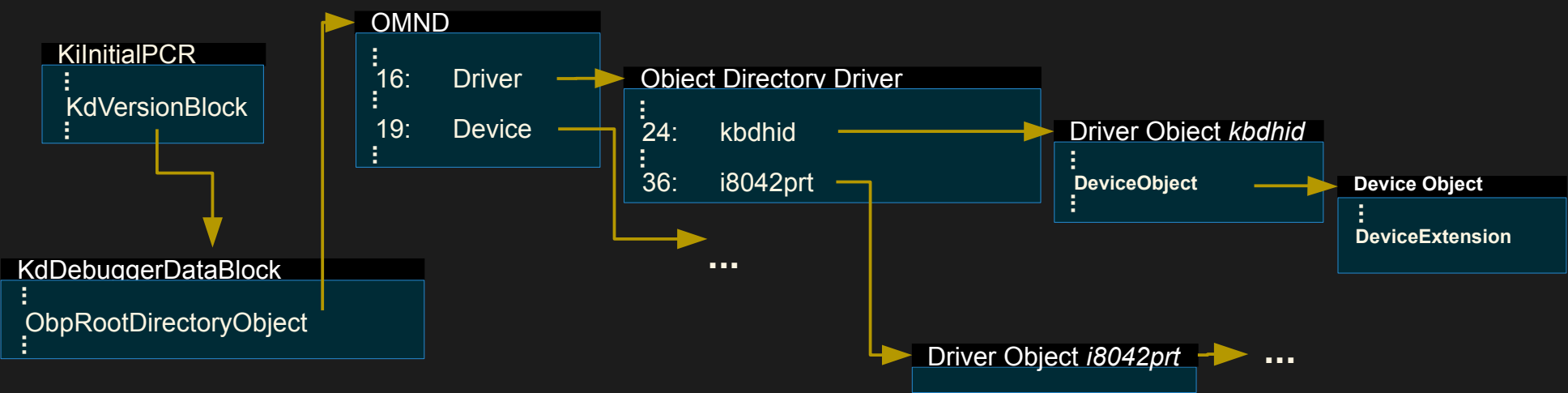
\$ Kernels tested: Vista / 7

\$ 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'

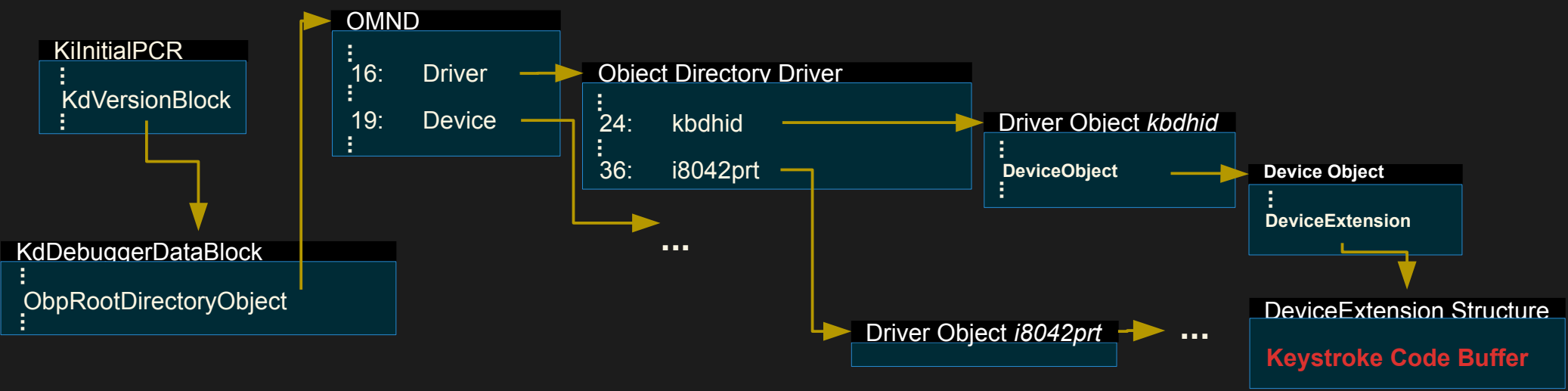
\$ Kernels tested: Vista / 7

\$ CR3 value required

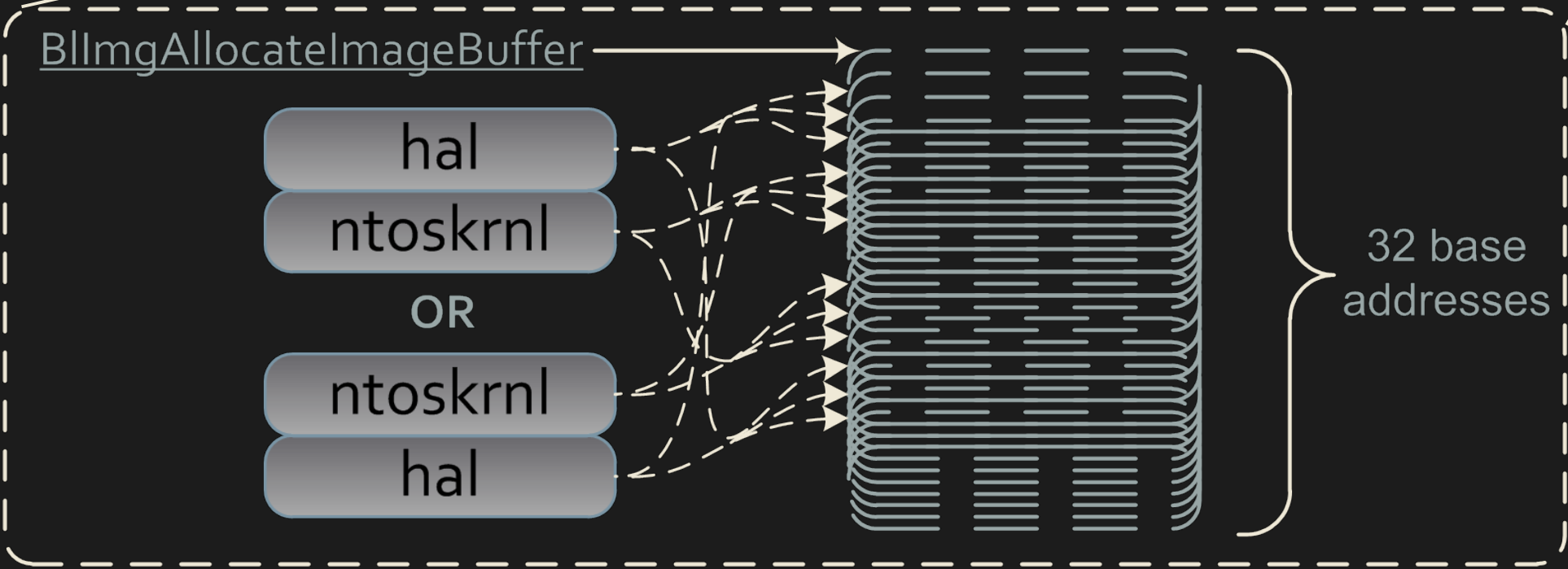
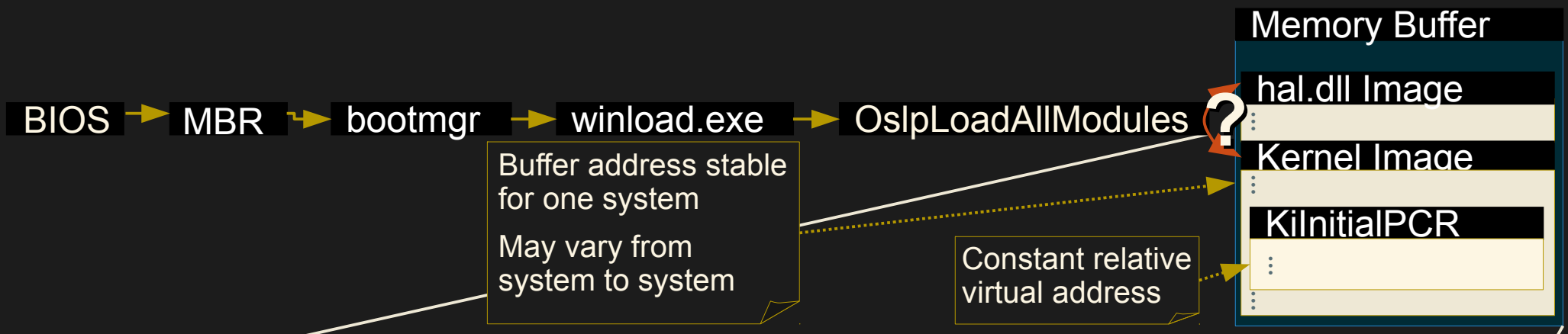
(Verified within DAGGER/DAGGER traverses page tables)

\$ 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

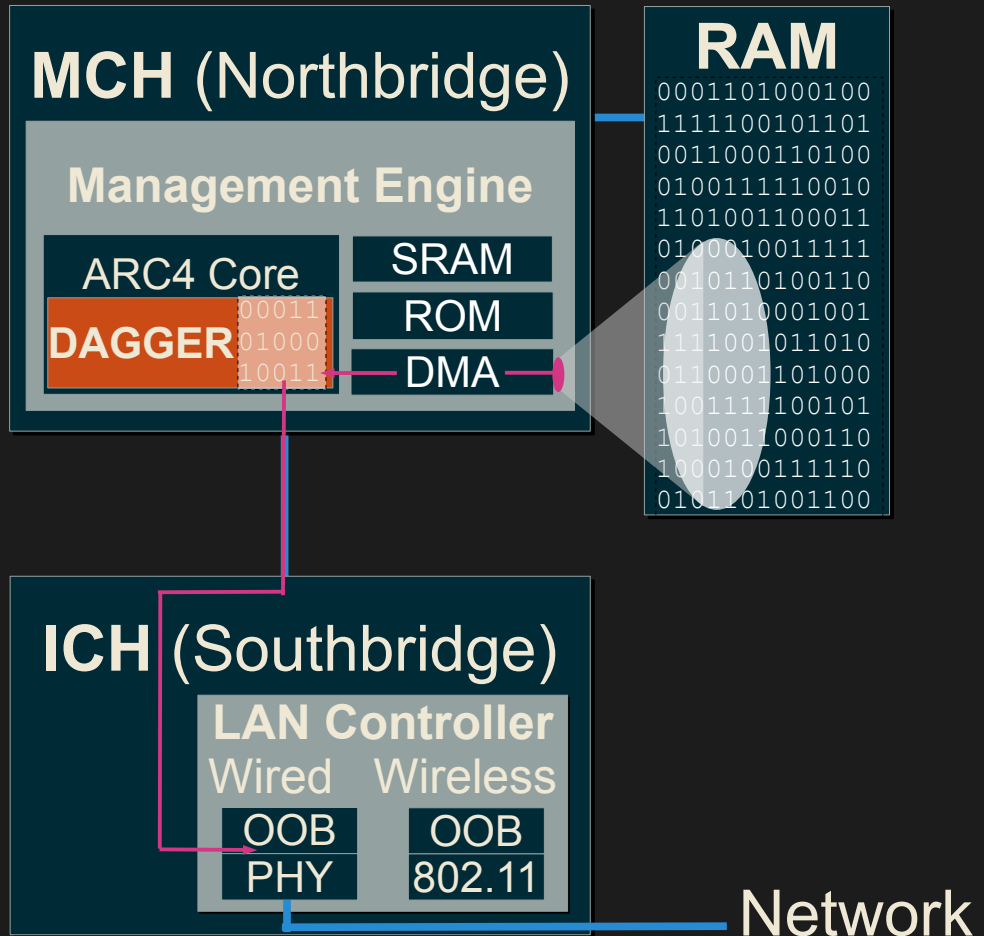
→ easy

(we just changed two bits)

\$ Stealthy network channel

→ challenging

(more than two bits :))

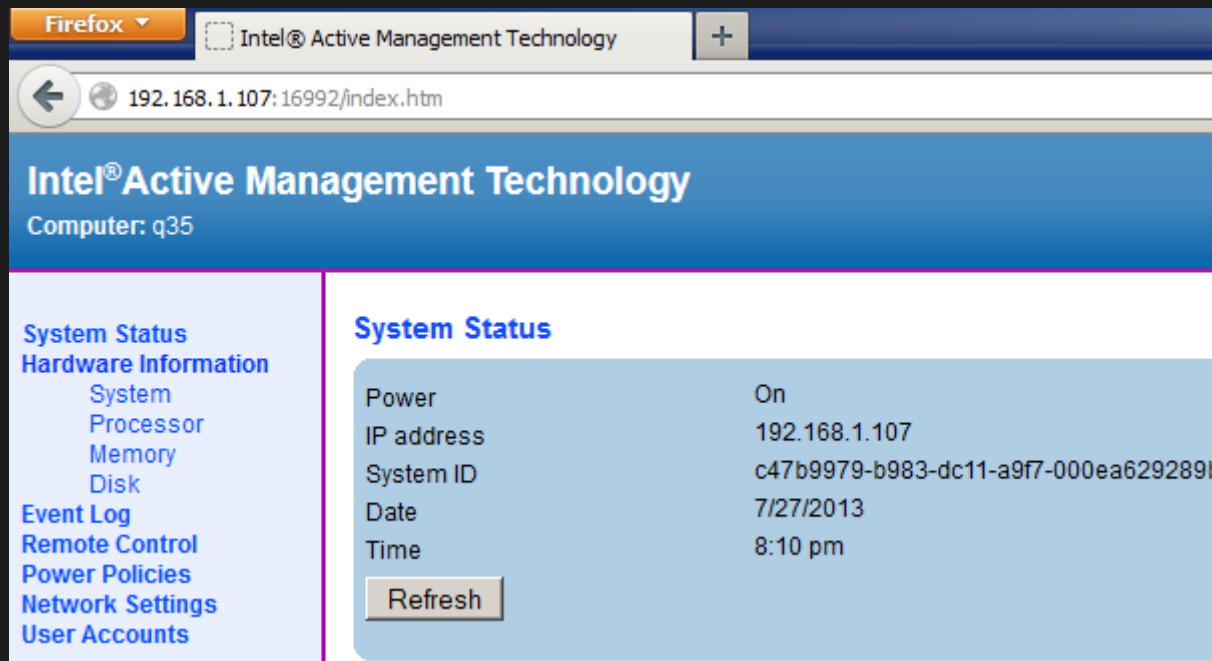




# Out-of-Band Network Channel

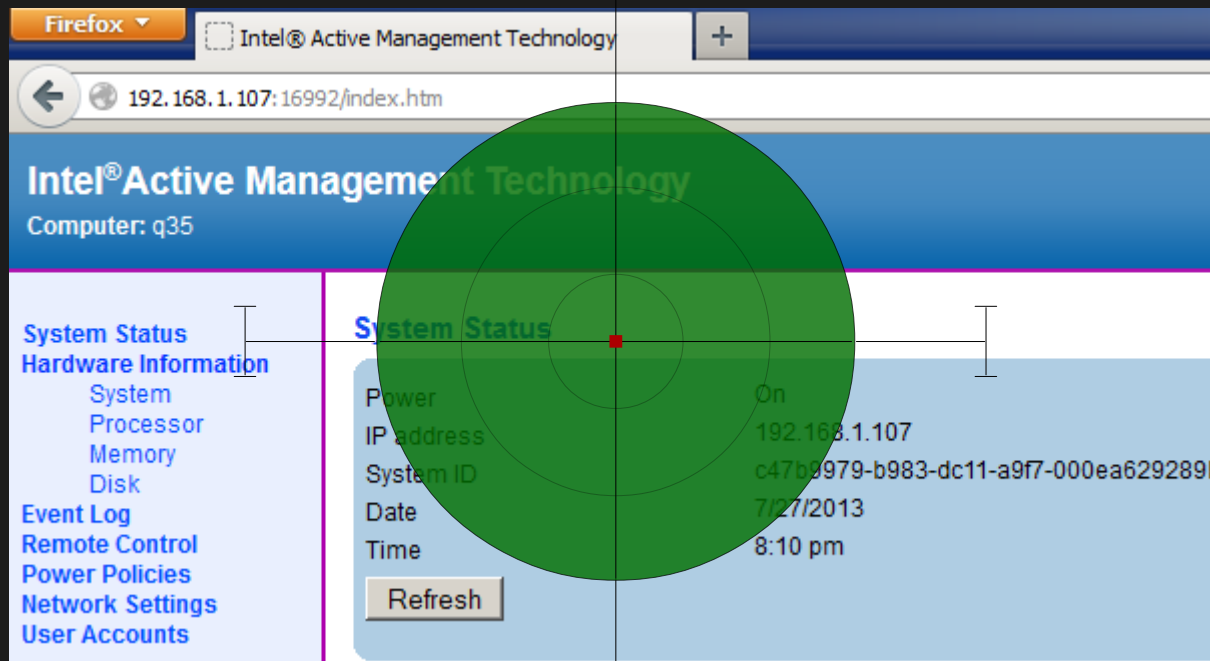
```
[patrickx@44con:~$] cat 'Target: ME OOB'
```

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



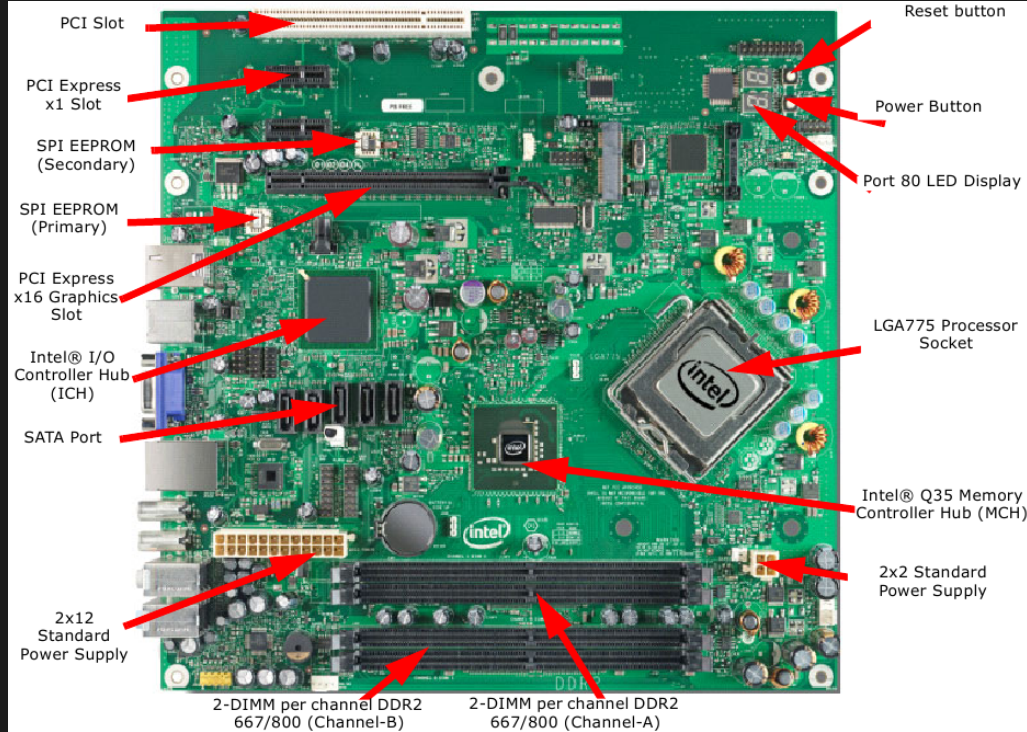
```
[patrickx@44con:~$] cat 'Target: ME OOB'
```

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

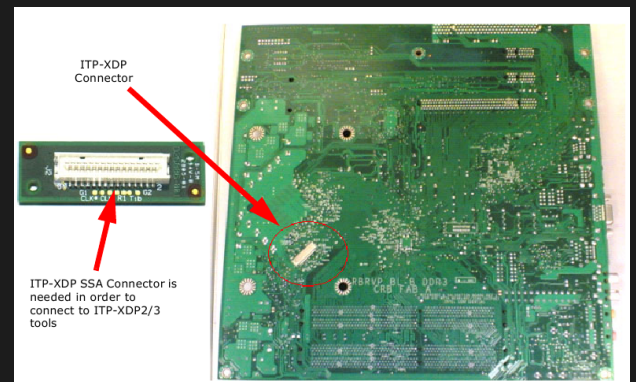


**How to find firmware code responsible for webserver replies?**

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



Board Features ([Int07], p.11)



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

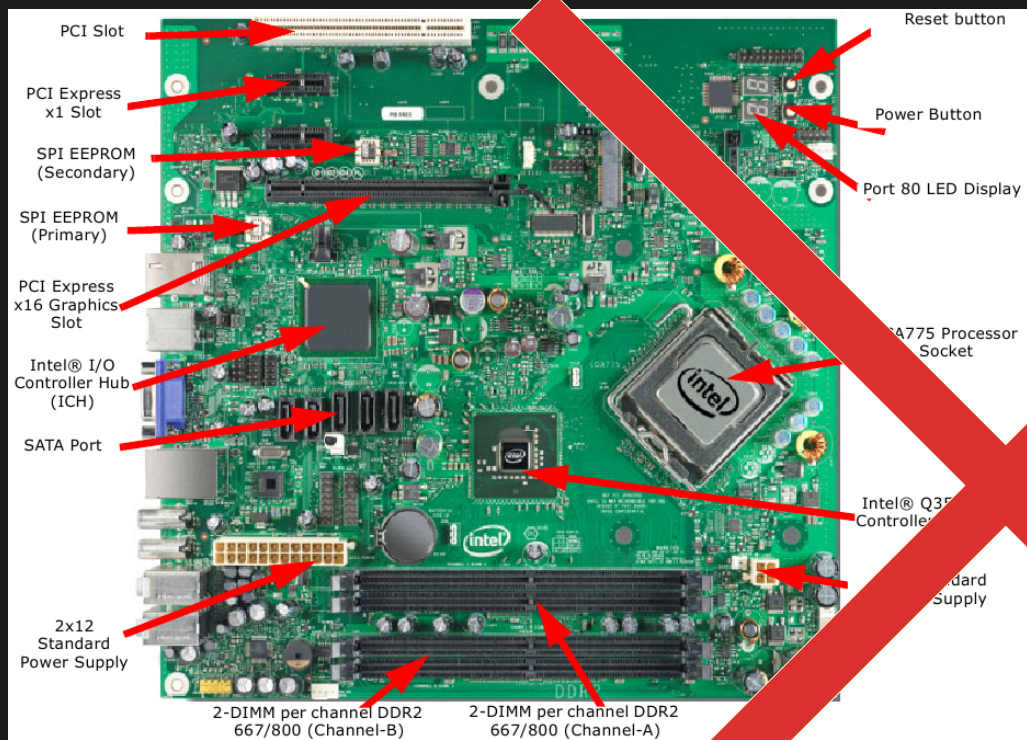
- Programming DMA hardware over JTAG port in debugger
- DMA-ing 64 bytes from system memory containing malicious VMExit handler code to internal chipset memory

```

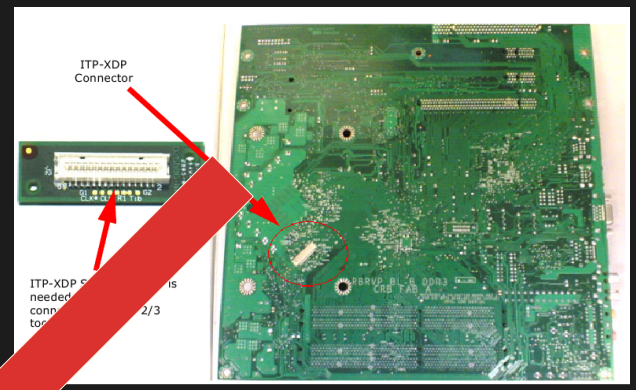
arc> dump 0x20000000
02000000: 00000000 00000000 00000000 00000000 :
02000010: 00000000 00000000 00000000 00000000 :
02000020: 00000000 00000000 00000000 00000000 :
02000030: 00000000 00000000 00000000 00000000 :
02000040: 00000000 00000000 00000000 00000000 :
02000050: 00000000 00000000 00000000 00000000 :
02000060: 00000000 00000000 00000000 00000000 :
02000070: 00000000 00000000 00000000 00000000 :
arc> arc:dma<1,0x73000,0,0x200000,64>
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..l
02000010: 89 4d hc 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 5b 68 1e 68 00 00 e8 aa 08 00 00 8d 45 | .E.Ph.h.....E
02000040: 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 |
02000060: 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 |
arc> dump 0x20000000
02000000: ec8b55fa 0084ec81 45890000 b85d89b4 | .U.....E..l
02000010: 89bc4d89 7589c055 d07d82cc 89d0200f | .M.U.u.)...
    
```

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

[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

```

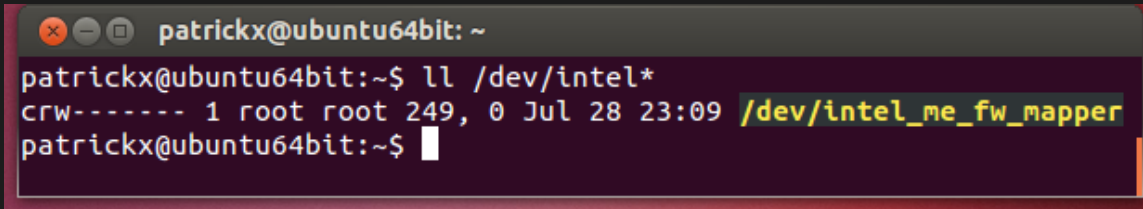
gpc> dump 0x20000000 00000000 00000000 00000000 :
02000000: 00000000 00000000 00000000 00000000 :
02000010: 00000000 00000000 00000000 00000000 :
02000020: 00000000 00000000 00000000 00000000 :
02000030: 00000000 00000000 00000000 00000000 :
02000040: 00000000 00000000 00000000 00000000 :
02000050: 00000000 00000000 00000000 00000000 :
02000060: 00000000 00000000 00000000 00000000 :
02000070: 00000000 00000000 00000000 00000000 :
gpc> arc_dma(1, 0x73000, 0, 0x200000, 64)
Dumped 64 B of data from Host 0x0073000
Status = 1
02000000: fa 55 8b ec 81 ec 84 00 00 00 89 45 b4 89 5d b8 | .U.....E..l
02000010: 00 4d hc 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: 5d d4 5b 68 1e 68 00 00 e8 aa 08 00 00 8d 45 | .E.Ph.h.....E
02000040: 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 |
02000060: 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 |
gpc> dump 0x20000000 00000000 00000000 00000000 :
02000000: ec8b55 00 00 ec81 45890000 b85d89b4 | .U.....E..l
02000010: 89bc4d8 00 00 8955 d07d82cc 89d0200f | .M.U.u.>...

```

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

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

\$ Linux:

A terminal window with a dark background and light text. The title bar shows 'patrickx@ubuntu64bit: ~'. The terminal content shows the command 'll /dev/intel\*' being executed, resulting in the output 'crw----- 1 root root 249, 0 Jul 28 23:09 /dev/intel\_me\_fw\_mapper'. The cursor is on the line following the output.

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

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

\$ Linux:

```
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:~$ sudo xxd -l 0x50 /dev/intel_me_fw_mapper  
00000000: 244d 4f44 0000 0000 0700 0000 0300 0200  $MOD.....  
00000010: 0100 fe03 0000 0000 6814 0000 6814 0000  .....h...h...  
00000020: 0000 0001 b822 0100 5000 0000 e013 0000  .....".P.....  
00000030: 4c4f 4144 4552 0000 0000 0000 0000 0000  LOADER.....  
00000040: 9b5a 7c88 2e4f a441 a6bd bf06 37bd a73b  .Z|..O.A....7..;  
patrickx@ubuntu64bit:~$
```



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

## \$ Linux:

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

```
patrickx@ubuntu64bit:~$ sudo xxd -l 0x50 /dev/intel_me_fw_mapper
00000000: 244d 4f44 0000 0000 0700 0000 0300 0200  $MOD.....
00000010: 0100 fe03 0000 0000 6814 0000 6814 0000  .....h...h...
00000020: 0000 0001 b822 0100 5000 0000 e013 0000  .....".P.....
00000030: 4c4f 4144 4552 0000 0000 0000 0000 0000  LOADER.....
00000040: 9b5a 7c88 2e4f a441 a6bd bf06 37bd a73b  .Z|..O.A...7..;
patrickx@ubuntu64bit:~$
```

```
patrickx@ubuntu64bit:~$
```

```
AMT Memory Monitor___
function keys: F5-> change address; F10->quit
arrow keys: left->column left; right->column right; up->line up; down->line down
other keys: page up->page up; page down->page down; home->1st line&1stcolumn; end->last line&
enter new address: █
```

```
0088cb78: 00 00 00 00 00 00 00 00 00 00 00 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 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 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 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 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 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 00 00 00 00 00 ad ff 01 00 ad ff 01
0088cc3c: 00 ad ff 01 00 ed ff 01 f0 e6 ff 01 00 00 00 00 c0 6c ff 01 c0 ac ff 01 80 a6 ff 01
0088cc58: 00 00 00 00 c0 6c ff 01 c0 6c ff 01 0d 00 01 00 00 00 00 00 00 00 00 00 02 00 00 00
0088cc74: 98 e4 7a 01 01 00 00 00 02 00 00 00 00 00 00 80 00 1c c0 14 a3 c3 00 00 00 00 00 00
0088cc90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 01 00
0088ccac: 00 00 00 00 00 00 00 00 00 00 00 00 02 6f 00 00 00 00 00 00 01 00 00 00 00 00 00
0088ccc8: 02 98 00 00 00 00 00 00 00 00 00 00 01 00 00 00 42 60 01 00 00 00 00 00 01 00 00
0088cce4: 01 00 00 00 42 62 01 00 00 00 00 00 01 00 00 00 01 00 00 00 00 00 00 00 00 00 00
0088cd00: 00 00 00 00 01 00 00 00 00 44 00 01 01 00 00 00 01 00 00 00 02 00 00 00 00 00 00
0088cd1c: 00 00 00 00 00 00 00 00 00 00 00 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 00 00 00 00 00 00 00 00 00 00 00
```

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

## \$ Linux:

```
patrickx@ubuntu64bit:~$ sudo xxd -l 0x50 /dev/intel_me_fw_mapper
00000000: 244d 4f44 0000 0000 0700 0000 0300 0200  $MOD.....
00000010: 0100 fe03 0000 0000 6814 0000 6814 0000  .....h...h...
00000020: 0000 0001 b822 0100 5000 0000 e013 0000  .....".P.....
00000030: 4c4f 4144 4552 0000 0000 0000 0000 0000  LOADER.....
00000040: 9b5a 7c88 2e4f a441 a6bd bf06 37bd a73b  .Z|..O.A...7..;
patrickx@ubuntu64bit:~$
```

```
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
0088cb94: 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
0088cbcc: 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
0088cc04: 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
0088cc3c: 00 ad ff 01 00 ed ff 01 f0 e6 ff 01 00 00 00
0088cc58: 00 00 00 00 c0 6c ff 01 c0 6c ff 01 0d 00 01
0088cc74: 98 e4 7a 01 01 00 00 00 02 00 00 00 00 00 00
0088cc90: 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
0088ccc8: 02 98 00 00 00 00 00 00 00 00 00 00 01 00 00
0088cce4: 01 00 00 00 42 62 01 00 00 00 00 00 01 00 00
0088cd00: 00 00 00 00 01 00 00 00 00 44 00 01 01 00 00
0088cd1c: 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
```

```
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

The image shows two terminal windows side-by-side. The left window is titled 'patrickx@ubuntu64bit: ~/IntelActiveManagementTechnology.project/subversion/patrickx.src/AMTBPHelper'. It displays the output of the 'memory.dump' command, showing the disassembly of a section of code. The instruction '4e2230: 00 7c bf 62 62bf7c00 mov r21,0x188\_cd34' is highlighted in green. The right window is titled 'patrickx@ubuntu64bit: ~/IntelActiveManagementTechnology' and shows the 'AMT Memory Monitor' interface. It displays a memory dump with the address '00ffacf0' highlighted in red. The memory dump shows a sequence of hexadecimal values, with '00ffacf0: 24 4d 45 4d 00 00 00 00 40 00 00 00 00 00 00 00' being the highlighted line.

[patrickx@44con:~\$]

# Demo Video 1 Exfiltrating Password via OOB

**Target Platform**

```

# ip addr show dev eth0 | grep "inet "
inet 192.168.1.106/24 brd 192.168.1.255 scope global eth0
#

patrickx@ubuntu64bit:~$ uname -a
Linux ubuntu64bit 3.5.0-27-generic #46-Ubuntu SMP Mon Mar 25 19:58:17 UTC 2013 x86_64 x86_64 x86_64 GNU/Linux
patrickx@ubuntu64bit:~$
  
```

64bit target

**External Platform**

```

# ip addr show dev eth0 | grep "inet "
inet 192.168.1.107/24 brd 192.168.1.255 scope global eth0
patrickx@patrickx-THINK:nion's listen0r$ ./listen0r
  
```

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

AMT thread 1:  
DAGGER\*

keyboard buffer monitor

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

AMT thread 1:  
DAGGER\*

keyboard buffer monitor

*space for new attack code*

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

AMT thread 1:  
DAGGER\*

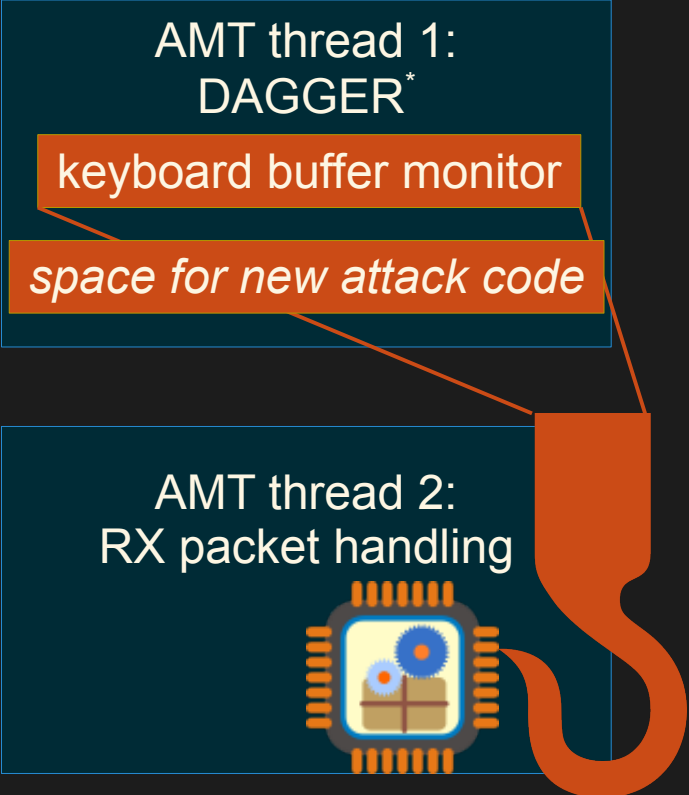
keyboard buffer monitor

*space for new attack code*

AMT thread 2:  
RX packet handling

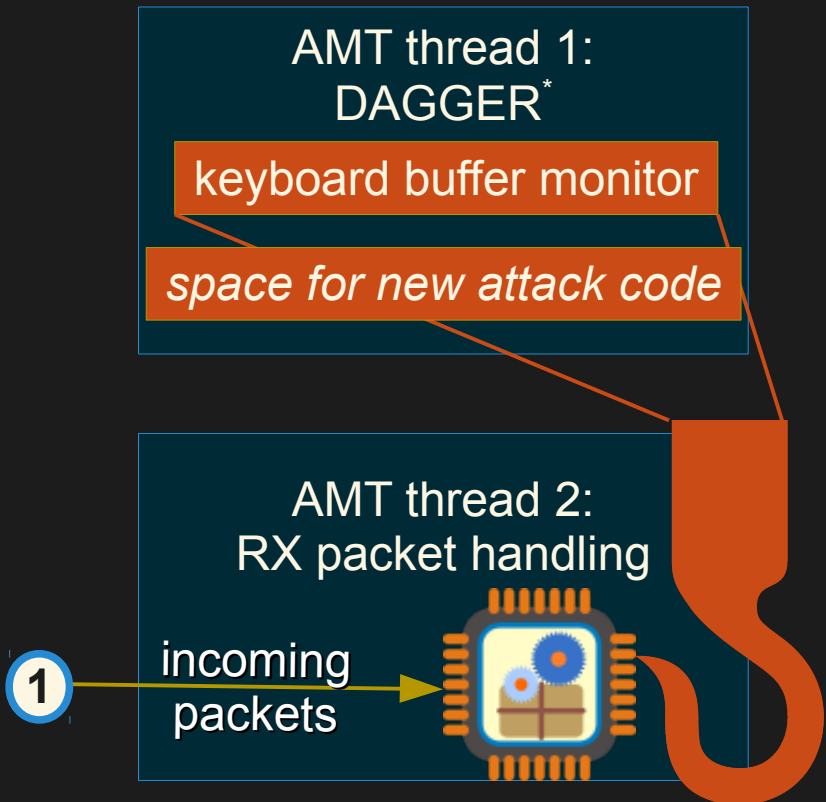


```
[patrickx@44con:~$] cat 'DAGGER Updates'
```

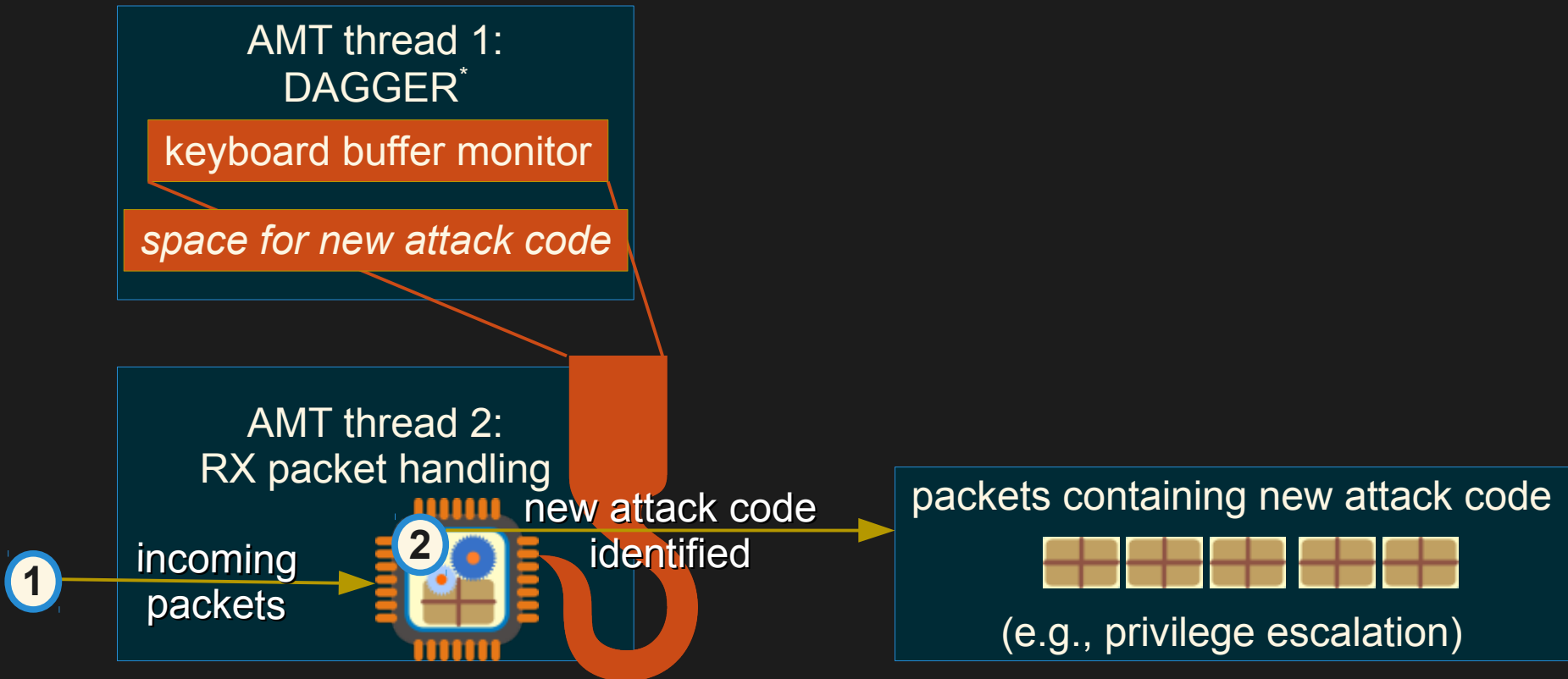




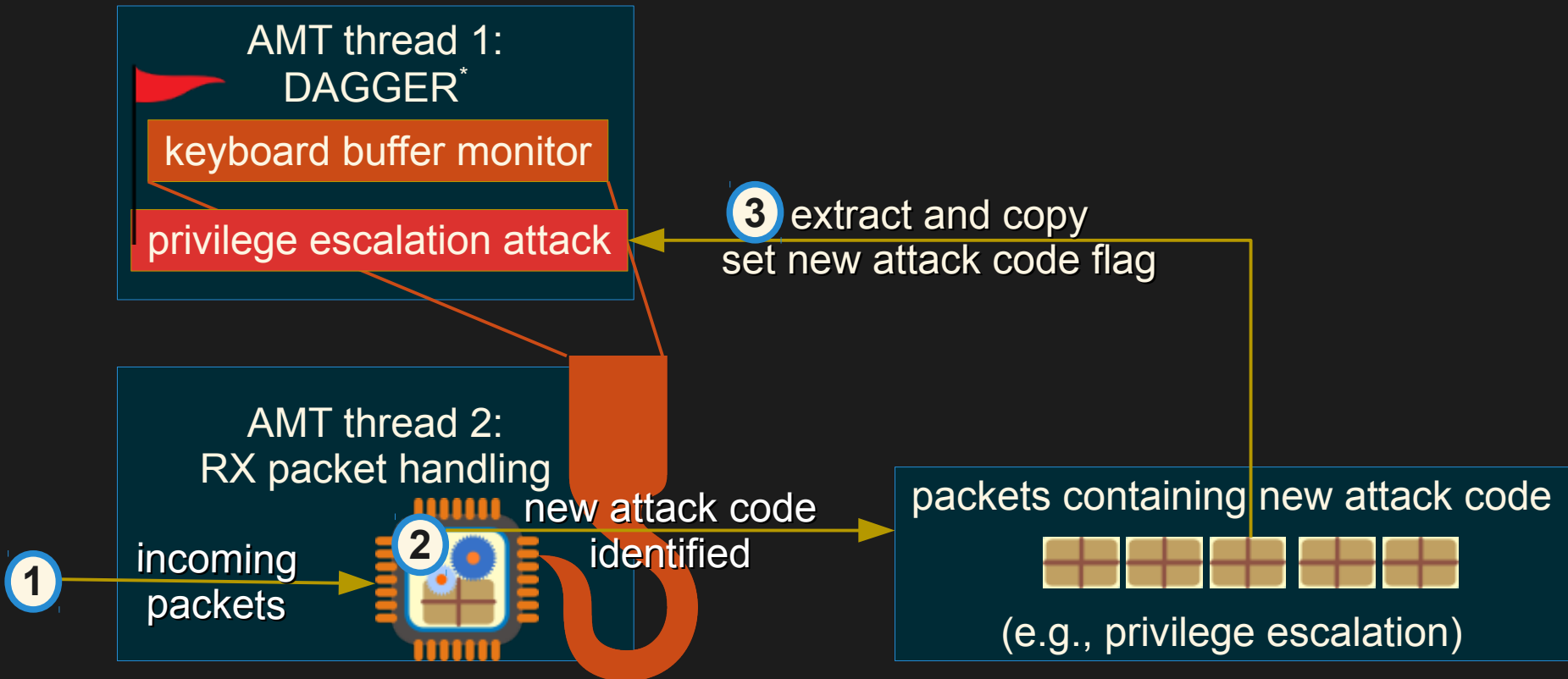
```
[patrickx@44con:~$] cat 'DAGGER Updates'
```



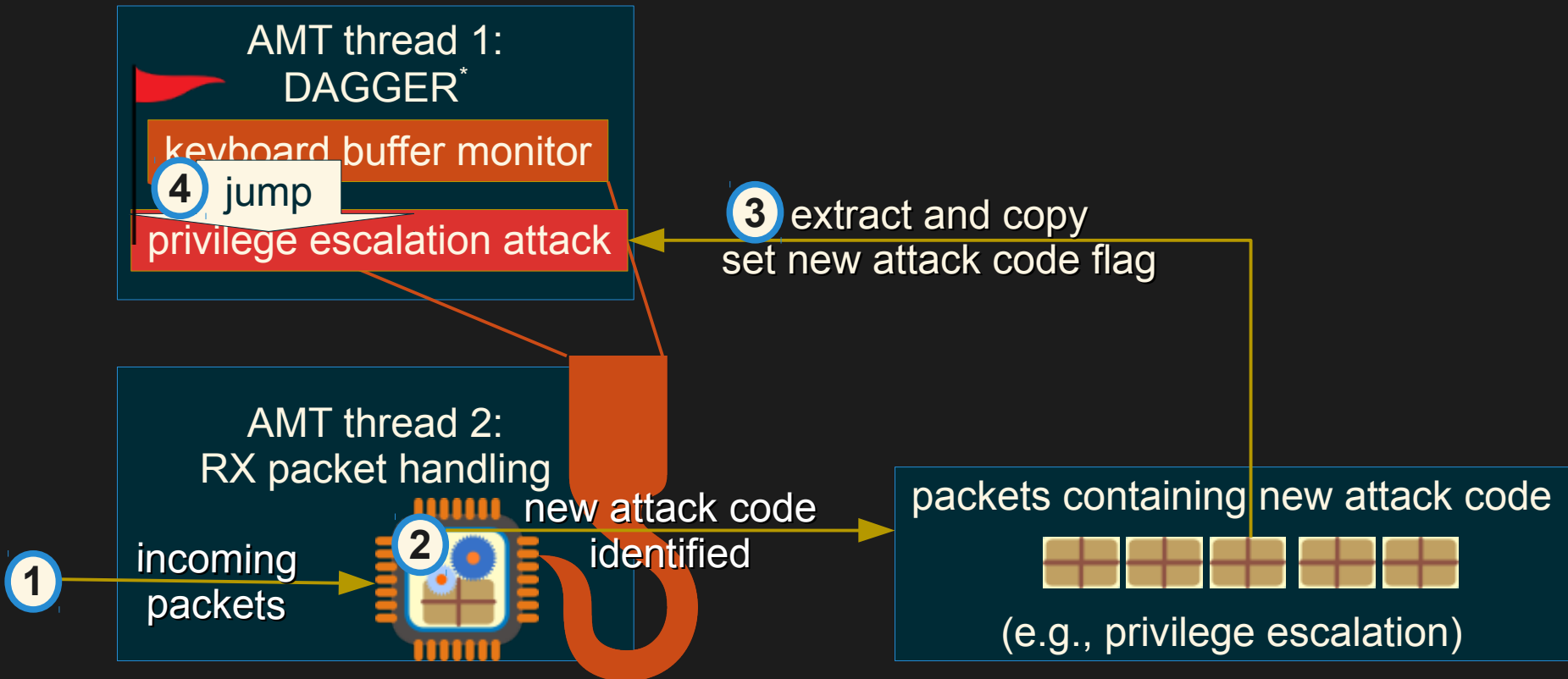
[patrickx@44con:~\$] cat 'DAGGER Updates'



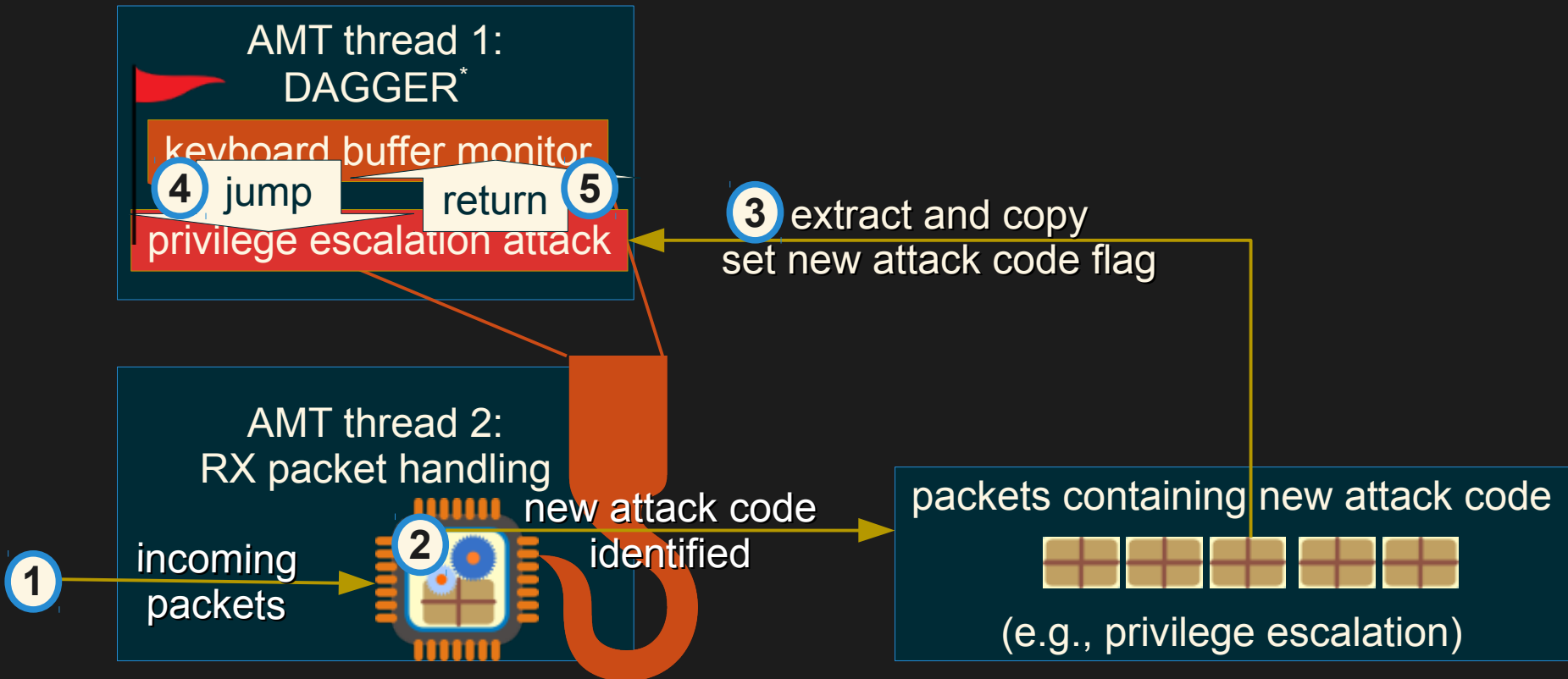
[patrickx@44con:~\$] cat 'DAGGER Updates'



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

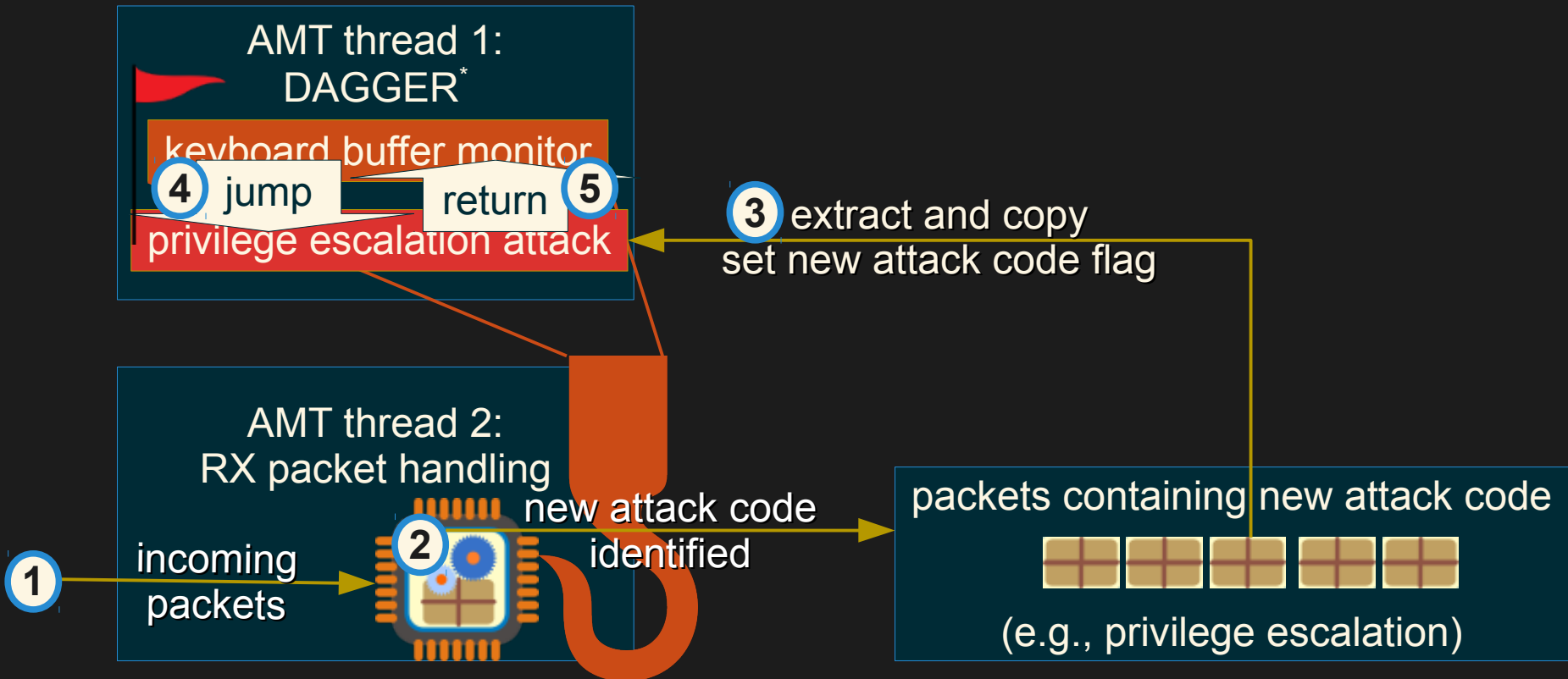


[patrickx@44con:~\$] cat 'DAGGER Updates'



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

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



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

Halting the debuggee... Done.  
 Requesting registers... Done.  
 Requesting extended registers... Done.  
 Requesting auxiliary registers... Done.

Buttons: Load Debugger, Unload Debugger, Refresh, Break, Continue, Dump, Synchronize, Switch counters, Custom, Stop

Registers: R0: 0x017af7e0, R1: 0x017af7b8, R2: 0x00000000, R3: 0x00000000, R4: 0x017af794, R6: 0x00000001, R8: 0x00000001, R10: 0x017af6c4, R12: 0x00000058, R14: 0x017af7b8, R16: 0x017af7c0, R18: 0x00000000, R20: 0x017ae3c0, R22: 0x017af7e0, R24: 0x00000000, R26: 0x010666e0, FP: 0x017af7e8, MMID: 0x00000000, MHI: 0x00000000, SP: 0x017af750, ILINK1: 0x8c40ebfc, LP\_COUNT: 0x00000000, ILINK2: 0x4c42ddeb, BLINK: 0x8c5392d9

STATUS: 0x8c46018b, SEMAPHORE: 0x00000000, LP\_START: 0x00404be3, LP\_END: 0x00404be5, IDENTITY: 0x13010108, DEBUG: 0x01000000

Emulator: STOPPED, Save State, New Breakpoint, On, Offline, Flash Image

Assembly code (hex, decimal, instruction, operands):

```

0x01012f14 18208e00 extb r1,r1
0x01012f18 8060fe08 asl r3,r1,8
0x01012f1c 68618200 or r3,r3,r1
0x01012f20 8021fe10 asl r1,r3,16
0x01012f24 57e17a08 sub.f 0,r2,8
0x01012f28 68608600 or r3,r1,r3
0x01012f2c 20000206 bnc 0x01012f40
0x01012f30 60200000 mov r1,r0
0x01012f34 67e10500 mov.f 0,r2
0x01012f38 380f8001 jz [blink]
0x01012f3c 20000800 b 0x01012f80
0x01012f40 67e07a03 and.f 0,r0,3
0x01012f44 20000321 bz.d 0x01012f60
0x01012f48 60200000 mov r1,r0
0x01012f4c 10408600 stb r3,[r1]
0x01012f50 4020fe01 add r1,r1,1
0x01012f54 67e0fa03 and.f 0,r1,3
0x01012f58 50417e01 sub r2,r2,1
0x01012f5c 27fffd82 bnz 0x01012f4c
0x01012f60 88817e02 lsr r4,r2,2
    
```

Hex dump:

```

0 1 2 3 4 5 6 7 8 9 A B C D E F
01FFC400 2d 39 36 32 30 2d 30 30 45 30 31 38 38 38 39 42 -9620-00E018889B
01FFC410 46 41 29 22 2c 20 75 72 69 3d 22 2f 6c 6f 67 6f FA)", uri="/logo
01FFC420 2e 67 69 66 22 2c 20 61 6c 67 6f 72 69 74 68 6d .gif", algorithm
01FFC430 3d 4d 44 35 2c 20 6e 6f 6e 63 65 3d 22 30 30 30 =MD5, nonce="000
01FFC440 30 66 31 33 62 64 63 30 66 33 66 38 62 32 37 61 0f13bdc0f3f8b27a
01FFC450 37 34 61 34 36 33 36 33 62 33 31 32 31 22 2c 20 74a46363b3121",
01FFC460 63 6e 6f 6e 63 65 3d 22 32 6a 62 2f 79 39 73 31 cnonce="2jb/y9s1
01FFC470 74 35 6e 6b 56 7a 30 52 78 70 73 4a 49 5a 69 72 t5nkVz0RxsJIZir
01FFC480 54 54 4d 43 33 2b 6a 43 32 50 58 67 34 4f 39 4c TTMC3+jC2PXg409L
01FFC490 6b 55 74 3d 22 2c 20 71 6f 70 3d 61 75 74 68 2c kUt=", qop=auth,
01FFC4A0 20 6e 63 3d 30 30 30 30 30 32 35 2c 20 72 65 nc=000000025, re
01FFC4B0 73 70 6f 6e 73 65 3d 22 30 32 61 35 32 30 31 35 sponse="02a52015
01FFC4C0 31 65 61 34 39 61 62 35 35 65 39 33 30 37 61 66 1ea49ab55e9307af
    
```

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

```
emutrace2.parsed.log
9172 0x014e4f30 624a2800 mov r18,r20 R18: 0x01736eba STATUS: 0x8c5393cd
9173 0x014e4f34 09ad81e8 ld r13,[fp,-24] R13: 0x01ffc400 STATUS: 0x8c5393ce
9174 0x014e4f38 0a6d81ec ld r19,[fp,-20] R19: 0x00000042 STATUS: 0x8c5393cf
9175 0x014e4f3c 67e82100 mov.f 0,r16 STATUS: 0x8c5393d0
9176 0x014e4f40 20000701 bz 0x014e4f7c STATUS: 0x8c5393d1
9177 0x014e4f7c 57e9fa64 sub.f 0,r19,100 STATUS: 0x6c5393e0
9178 0x014e4f80 20000405 bc 0x014e4fa4 STATUS: 0x6c5393e1
9179 0x014e4fa4 081f0000 ld r0,[0x16b_2a94] R0: 0x0106a2b8 STATUS: 0x6c5393ea
9180 0x014e4fac 08000004 ld r0,[r0,4] R0: 0x01068644 STATUS: 0x6c5393ec
9181 0x014e4fb0 08600008 ld r3,[r0,8] R3: 0x00404b98 STATUS: 0x6c5393ed
9182 0x014e4fb4 60092400 mov r0,r18 R0: 0x01736eba STATUS: 0x6c5393ee
9183 0x014e4fb8 60269a00 mov r1,r13 R1: 0x01ffc400 STATUS: 0x6c5393ef
9184 0x014e4fbc 6049a600 mov r2,r19 R2: 0x00000042 STATUS: 0x6c5393f0
9185 0x014e4fc0 38018200 jl [r3] BLINK: 0x6c5393f1 STATUS: 0x6c5393f1 {
9186 0x01012e60 381f0000 j 0x01180644 STATUS: 0x6c404b99
9187 0x01180644 538e7eb4 sub sp,sp,180 SP: 0x017af6d4 STATUS: 0x6c460192
9188 0x01180648 100e3e84 st blink,[sp,132] STATUS: 0x6c460193
9189 0x0118064c 100e7cac st 0x46_019a,[sp,172] STATUS: 0x6c460194
9190 0x01180654 2fff4380 bl 0x01180074 BLINK: 0x6c460196 STATUS: 0x6c460196 {
9242 0x01180658 28038b00 bl 0x011822b4 BLINK: 0x6c460197 STATUS: 0x6c460197 {
9256 0x0118065c 2fff5c00 bl 0x01180140 BLINK: 0x8c460198 STATUS: 0x8c460198 {
9304 0x01180660 0bee0084 ld blink,[sp,132] BLINK: 0x6c5393f1 STATUS: 0x6c460199
9305 0x01180664 0b8e0078 ld sp,[sp,120] SP: 0x017af788 STATUS: 0x6c46019a
9306 0x01180668 67810500 mov.f lp_count,r2 STATUS: 0x2c46019b
9307 0x0118066c 68800200 or r4,r0,r1 R4: 0x01ffeeba LP_COUNT: 0x00000042 STATUS: 0x2c46019c
9308 0x01180670 381f0000 j 0x01012e68 STATUS: 0x2c46019d
9309 0x01012e68 380f8101 jz.f [blink] STATUS: 0x2c404b9b
9310 0x01012e6c 67e27a03 and.f 0,r4,3 STATUS: 0x2c404b9c
9311 0x01012e70 88817e03 lsr r4,r2,3 R4: 0x00000008 STATUS: 0x2c404b9d
9312 0x01012e74 20001082 bnz 0x01012efc STATUS: 0x2c404b9e
9313 0x01012efc 5020fe01 sub r1,r1,1 R1: 0x01ffc3ff STATUS: 0x2c404bc0
9314 0x01012f00 50607e01 sub r3,r0,1 R3: 0x01736eb9 STATUS: 0x2c404bc1
9315 0x01012f04 30000100 lp 0x01012f10 STATUS: 0x2c404bc2 LP_START: 0x00404bc2 LP_END: 0x00404bc4
9316 0x01012f08 08809401 ldb.a r4,[r1,1] R1: 0x01ffc400 R4: 0x00000000 LP_COUNT: 0x00000041 STATUS: 0x2c404bc3
9317 0x01012f0c 11418801 stb.a r4,[r3,1] R3: 0x01736eba STATUS: 0x2c404bc2
9318 0x01012f08 08809401 ldb.a r4,[r1,1] R1: 0x01ffc401 R4: 0x00000019 LP_COUNT: 0x00000040 STATUS: 0x2c404bc3
9319 0x01012f0c 11418801 stb.a r4,[r3,1] R3: 0x01736ebb STATUS: 0x2c404bc2
9320 0x01012f08 08809401 ldb.a r4,[r1,1] R1: 0x01ffc402 R4: 0x000000d1 LP_COUNT: 0x0000003f STATUS: 0x2c404bc3
9321 0x01012f0c 11418801 stb.a r4,[r3,1] R3: 0x01736ebc STATUS: 0x2c404bc2
9322 0x01012f08 08809401 ldb.a r4,[r1,1] R1: 0x01ffc403 R4: 0x0000000f LP_COUNT: 0x0000003e STATUS: 0x2c404bc3
9323 0x01012f0c 11418801 stb.a r4,[r3,1] R3: 0x01736ebd STATUS: 0x2c404bc2
9324 0x01012f08 08809401 ldb.a r4,[r1,1] R1: 0x01ffc404 R4: 0x00000006 LP_COUNT: 0x0000003d STATUS: 0x2c404bc3
```



```
[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
il    [r3]         BLINK: 0x6c5393f1  STATUS: 0x6c5393f1  {
  i      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
  i      0x01012e68      STATUS: 0x2c46019d
  iz.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
il      [r3]        BLINK: 0x6c5393f1  STATUS: 0x6c5393f1  {
  i      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
  i      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
il      [r3]        BLINK: 0x6c5393f1  STATUS: 0x6c5393f1  {
  i      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
  i      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
il      [r3]        BLINK: 0x6c5393f1  STATUS: 0x6c5393f1  {
i       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
i       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

```

} memcopy parameter  
 } memcopy 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
il      [r3]        BLINK: 0x6c5393f1  STATUS: 0x6c5393f1  {
i       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
i       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'

```

mov     r0,r18      R0: 0x01736eba  STATUS: 0x6c5393ee
mov     r1,r13      R1: 0x01ffc400  STATUS: 0x6c5393ef
mov     r2,r0       R2: 0x01736eb9  STATUS: 0x6c5393f0
hook to intercept incoming packets
il    [r3]        BLINK: 0x6c5393f1  STATUS: 0x6c5393f1  {
i     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: 0x6c460197
bl    0x011822b4   BLINK: 0x6c460197  STATUS: 0x6c460198
bl    0x01180140   BLINK: 0x8c460198  STATUS: 0x8c460199
ld    blink,[sp,132]  BLINK: 0x6c5393f1  STATUS: 0x6c46019a
ld    sp,[sp,120]   SP: 0x017af788   STATUS: 0x6c46019b
mov.f lp_count,r2   STATUS: 0x2c46019b
or    r4,r0,r1     R4: 0x01fffeba  LP_COUNT: 0x00000004
i     0x01012e68   STATUS: 0x2c46019d
iz.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*

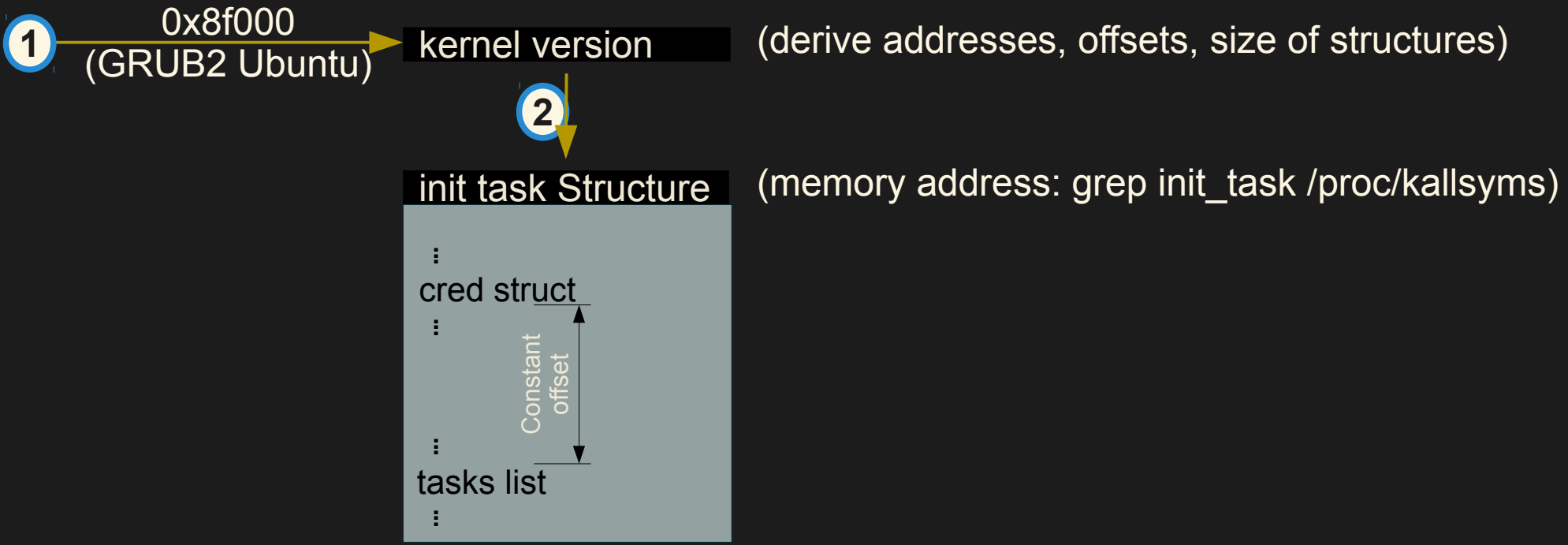
```
[patrickx@44con:~$] cat 'Privilege Escalation'
```

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

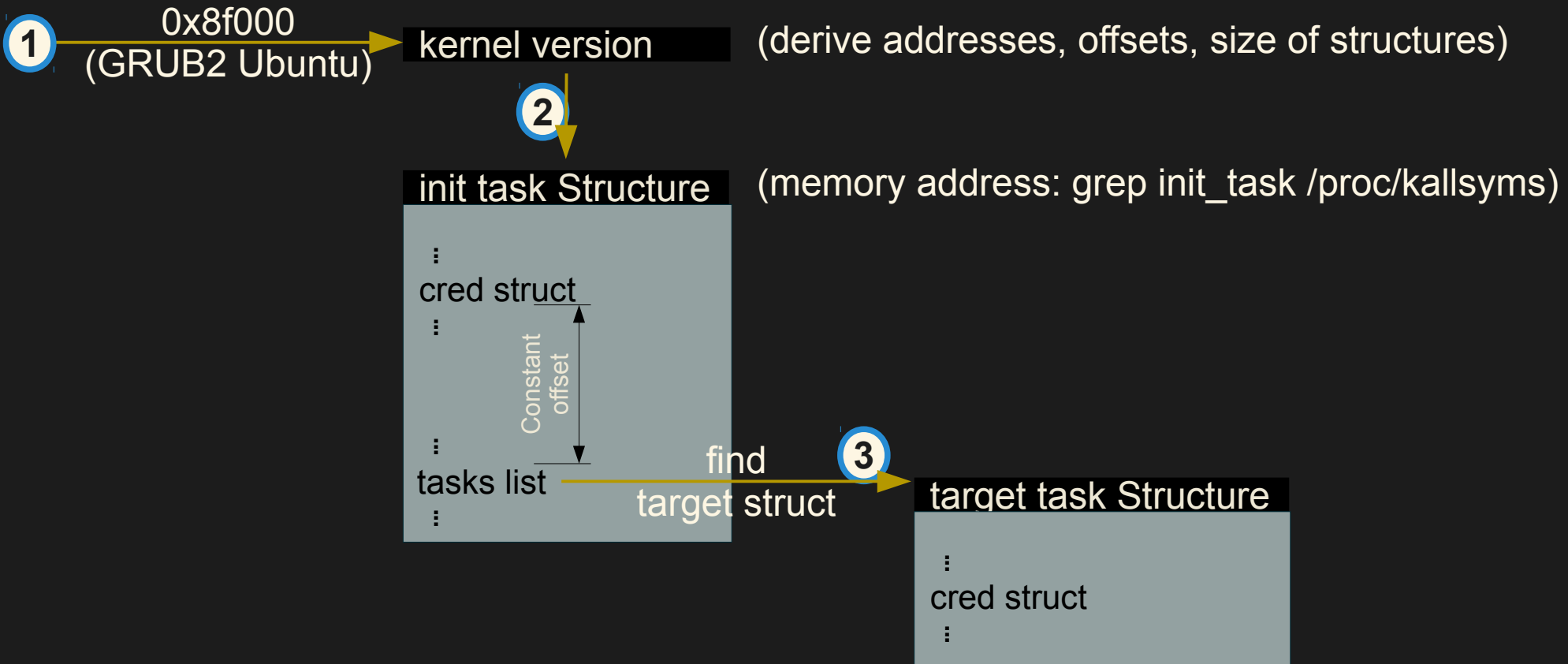
①  $0x8f000$   
(GRUB2 Ubuntu) → kernel version (derive addresses, offsets, size of structures)



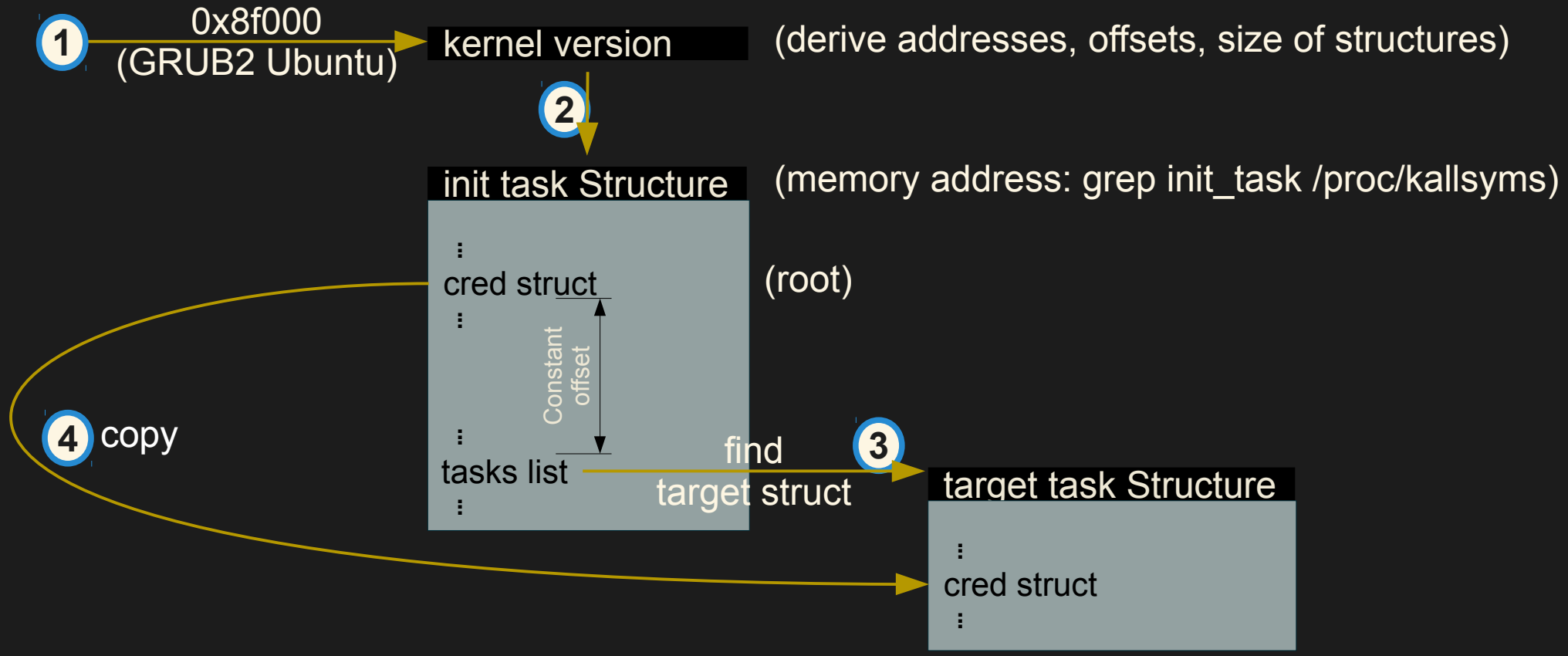
# [patrickx@44con:~\$] cat 'Privilege Escalation'



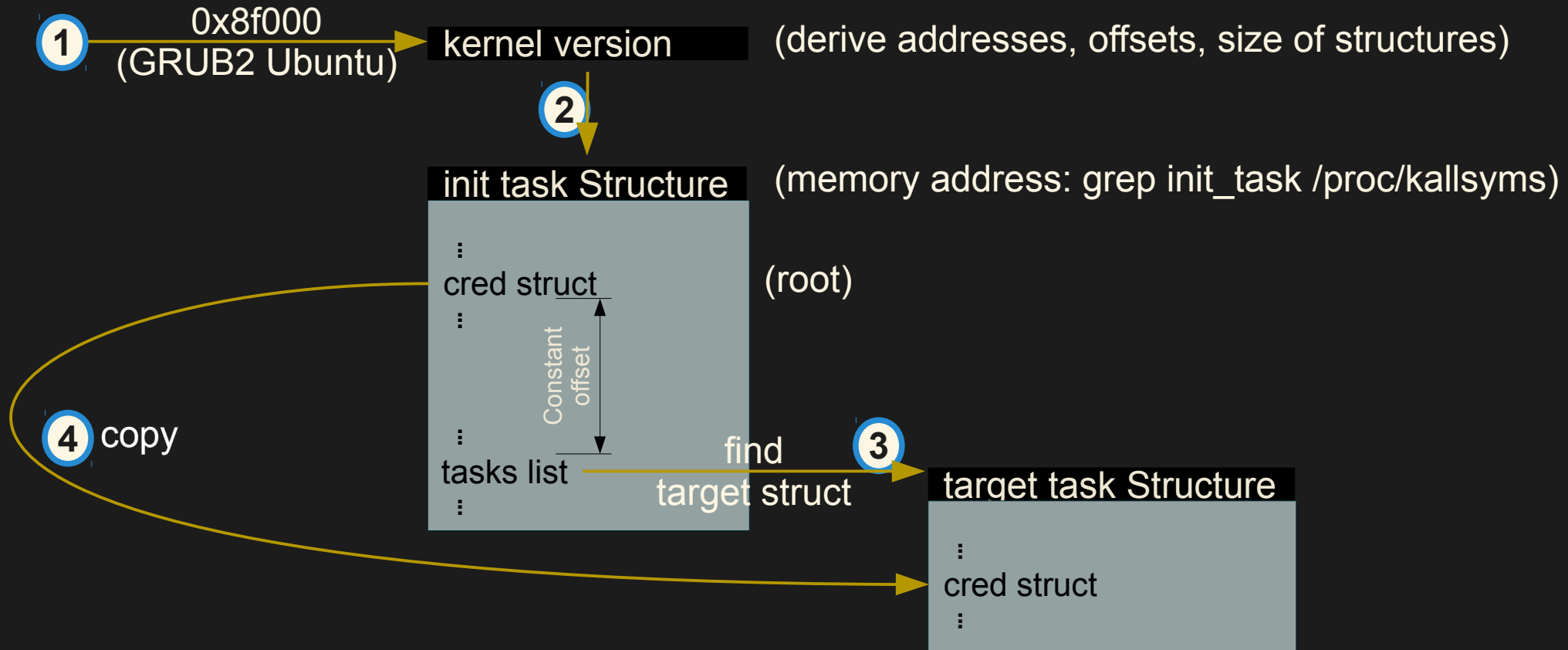
# [patrickx@44con:~\$] cat 'Privilege Escalation'



# [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 [...]”

[patrickx@44con:~\$]

# Demo Video 2 Privilege Escalation via OOB

**Target Platform**

Wireshark - Capturing from eth0 [Wireshark 1.8.2] Sun Aug 18 16:01:59 2013

Filter: `(ip.dst==192.168.1.106 && ip.src==192.168.0.0/16)`

No.	Time	Source	Destination	Protocol	Length	Info
23	13.46	192.168.1.106	255.255.255.255	UDP	64	Source port

```

eth0: <live capture in progress> to ... No Packets Profile: Default
patrickx@ubuntu:~$ ./intelActiveManagementTechnology/progost/jobserver/bin/patrickx.srv/44con
# ip addr show dev eth0 | grep "inet "
inet 192.168.1.106/24 brd 192.168.1.255 scope global eth0
# ./some0day
#
  
```

**External Platform**

File Edit View Go Capture Analyze Sun Aug 18 16:01:59 2013

Filter: `[06] ip.dst==192.168.1.106`

No.	Time	Source	Destination	Protocol	Length	Info
23	13.46	192.168.1.106	255.255.255.255	UDP	64	Source port

```

patrickx@patrickx-THINK:nion's listen0r$ ./listen0r
r
patrickx@patrickx-THINK:net_convert$ sudo hping3 -i u10000 -c 76 -I eth0 -S 192.168.1.106 -p 16992 -d 1402 -E ./DMA_poc_remote_privilege_escalation.arc4.e
lf.net

http://192.168.1.106/16992/index.htm

Intel® Active Management Technology
Computer: c35

System Status
Hardware Information
System: Power On
Processor: IP address 192.168.1.106
Memory: System ID 202019a-473b-611-961a-00d01880904e
Disk: Date 8/18/2013
Event Log: Remote Control Time 4:00 pm
Remote Software

patrickx@patrickx-THINK:~$ ssh 192.168.1.106
  
```



# Covert Network Channel

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

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

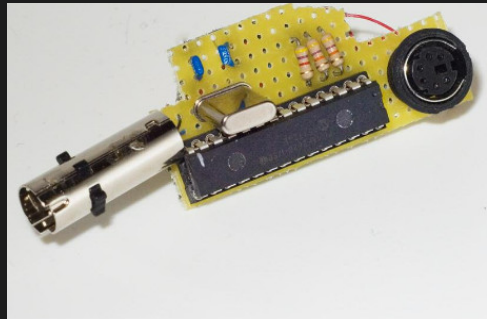
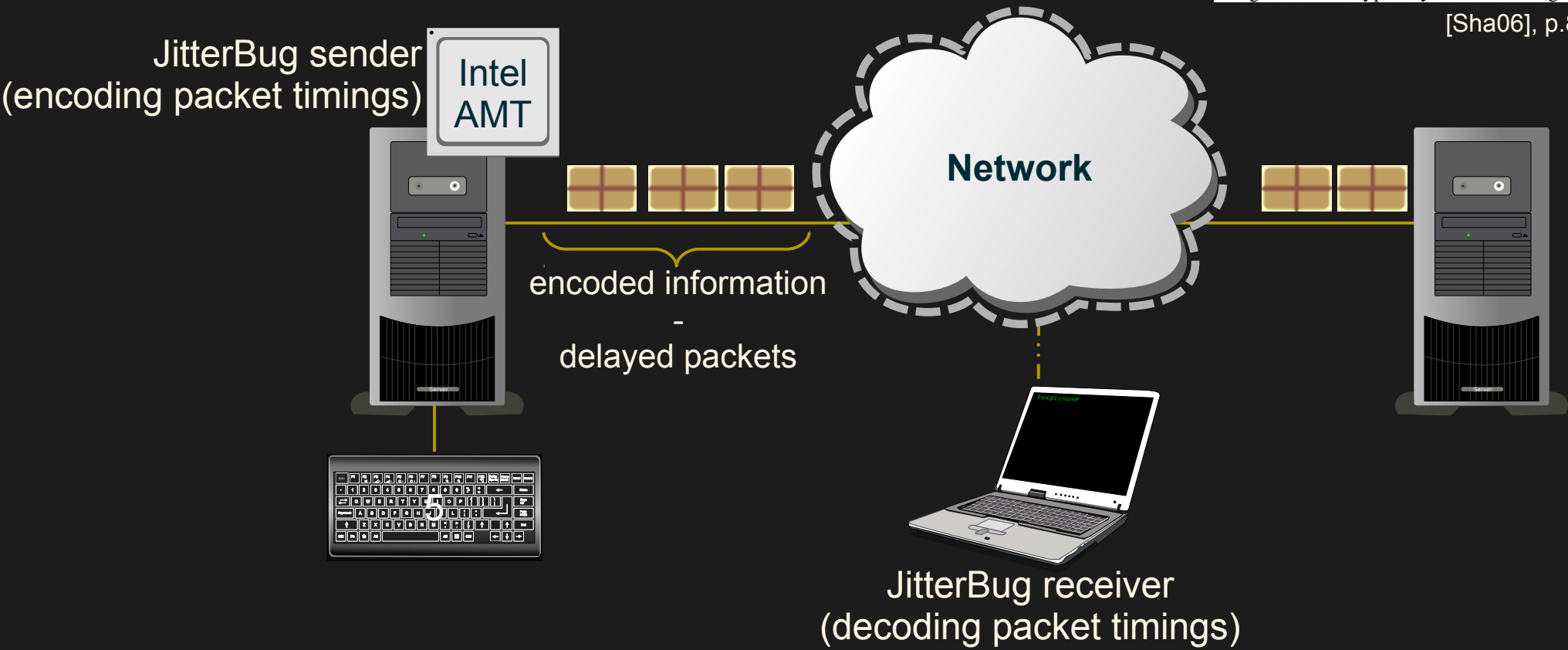


Figure 4: Prototype Keyboard JitterBug [Sha06], p.8



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

\$ Outgoing packet interception ✓

\$ Measure time!

AMT peripheral (timer) access:

```
lr r0,[0x8011]
```

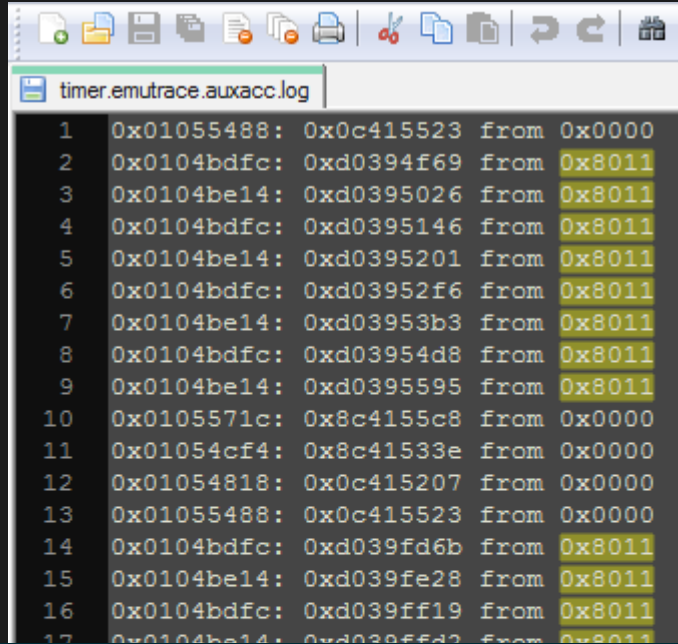
→ Read timer register:

resolution

~ 996500 Hz

\$ Packets  
to delay

Wireshark log of an  
AMT TCP session



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]



```
[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

The screenshot displays a remote administrator window titled "Remote administrator" with the Intel logo. It shows a "Log On" screen for "Intel® Active Management Technology" with fields for "User Name" and "Password". Below this is a console window with a table of statistics:

Nr.	Bit	Time	delta	Full delta	Err
1	0		6513	6513	
2	x		11197	51197	
3	x		12223	12223	
4	1		16107	16107	
5	0		6733	46733	
6	x		9893	89893	
7	1		14074	34074	
8	0		3525	3923525	

Other elements include a "Jitterbug receiver" window with a rotating timing window (20ms) and a jitter plot. A callout points to a "Decoded bit" (X) and a "Frame start sequence" (10100111). A "Leaked data" callout points to a text field at the bottom.



# Final Remarks

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

\$ Virtualization Technology for Directed I/O (I/OMMU, [Abr06])

\$ Attacks: [San10], [Woj09], [Woj11a], [Woj11b]

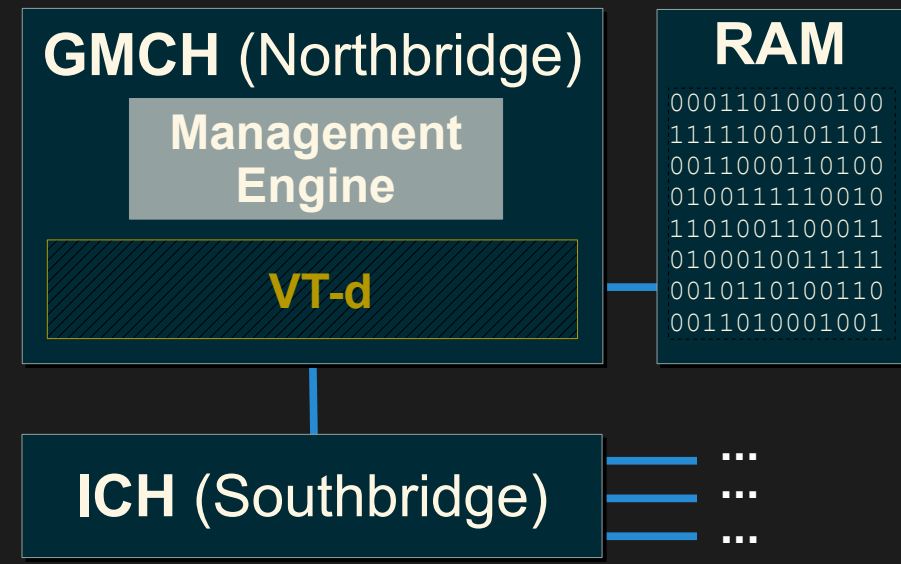
\$ No driver for Windows (including 8)

\$ Academic:

\$ VIPER - Verifying the integrity of peripherals' firmware [Li11]

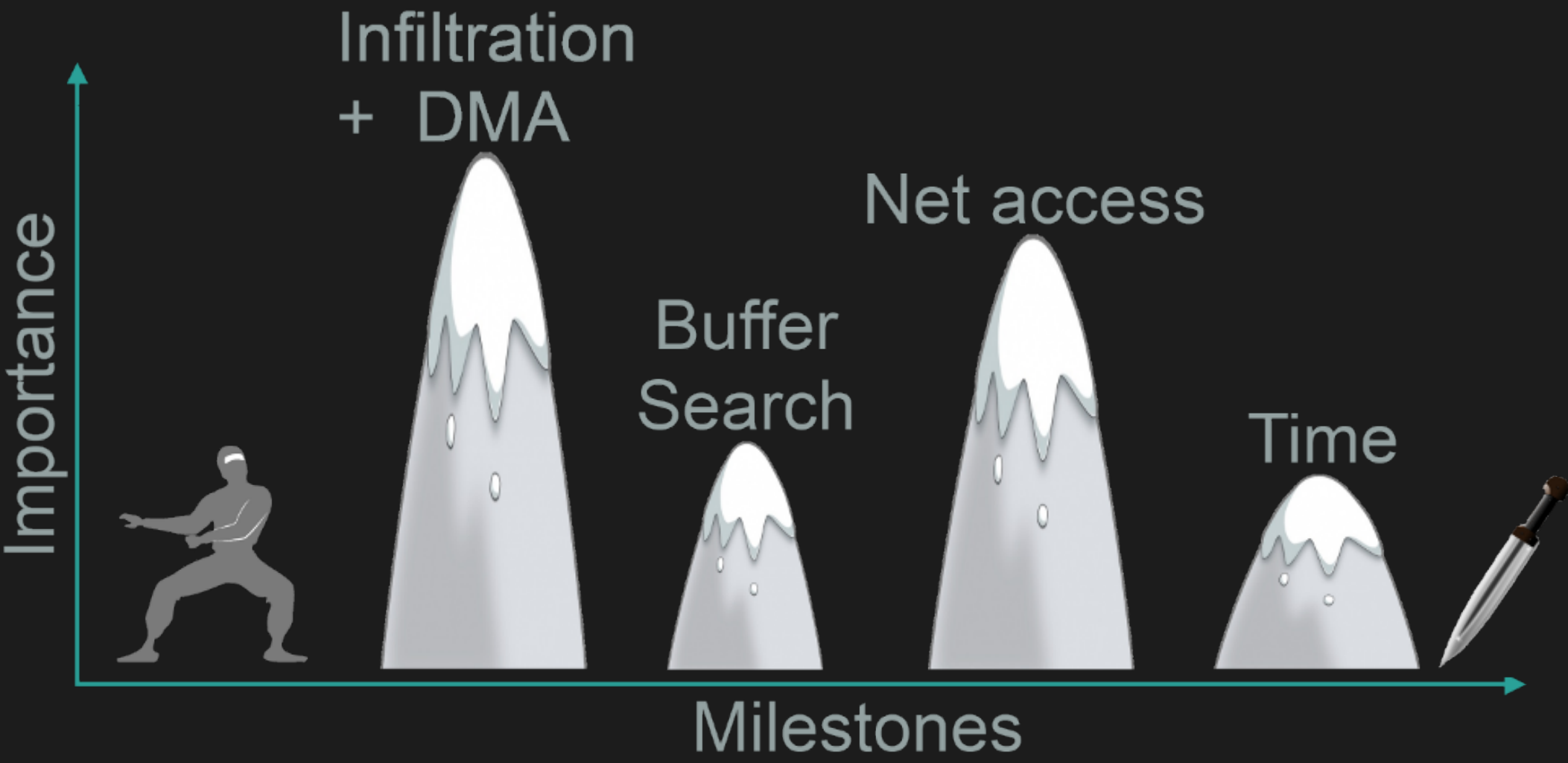
\$ NAVIS - Network Adapter Verification and Integrity checking Solution [Duf11]

\$ BARM - Bus Agent Runtime Monitor [Ste13]





# Conclusion



# 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](mailto:patrickx@sec.t-labs.tu-berlin.de)



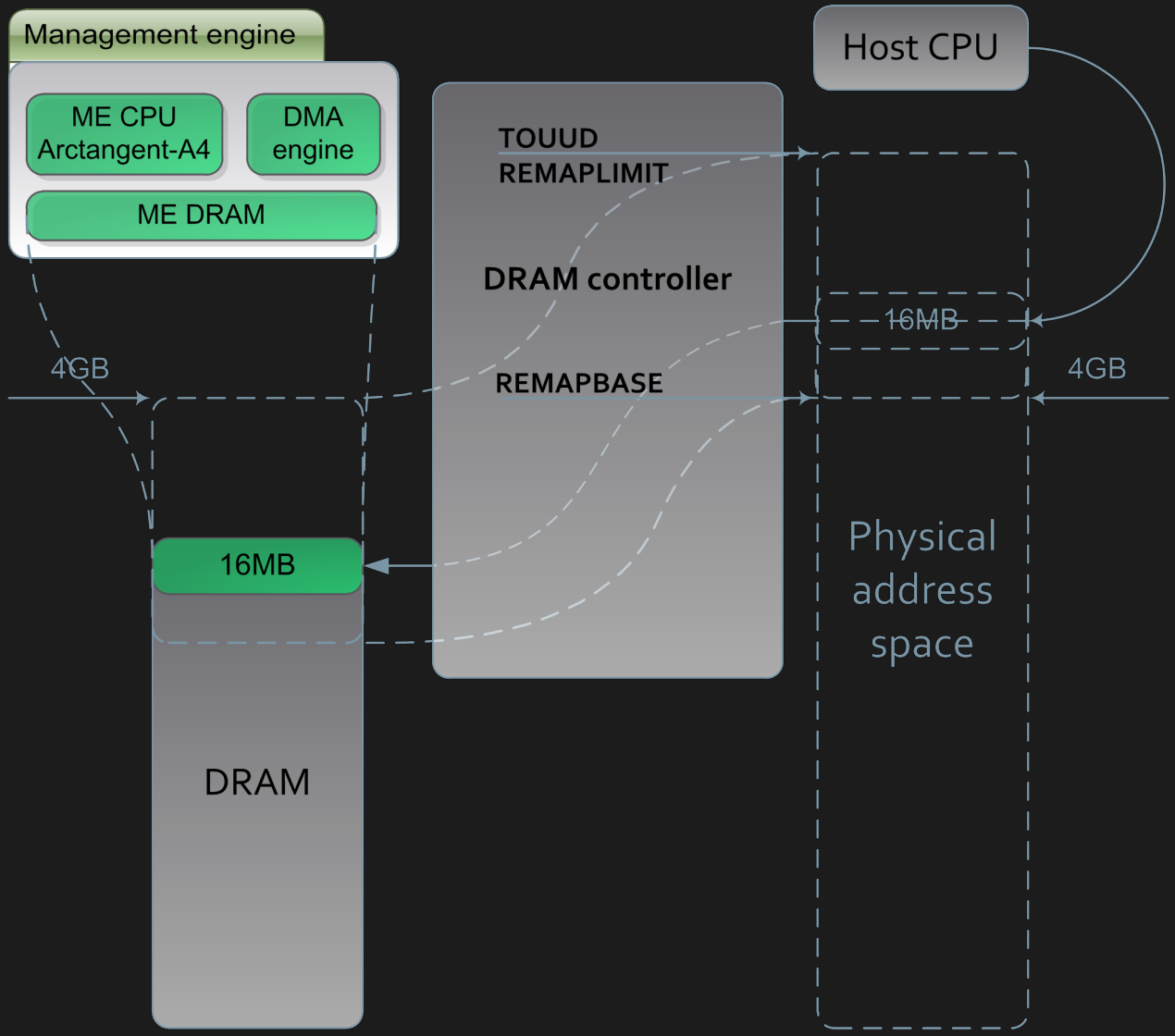
44CON 2013, London, UK



**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: Owned 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. Dufлот, 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