Berlin Institute of Technology

FG Security in Telecommunications

© Weiss

Evaluating “Ring -3” Rootkits

SPRING 6: SIDAR Graduierten-Workshop über Reaktive Sicherheit

Patrick Stewin, March 21, 2011, Bochum, Germany
patrickx@sec.t-labs.tu-berlin.de

Agenda

- Introduction
- “Ring -3” Execution Environment
- Our “Ring -3” Rootkit
- Target Platform Infiltration
- Exfiltration of Collected Data
- Summary
- Future Work

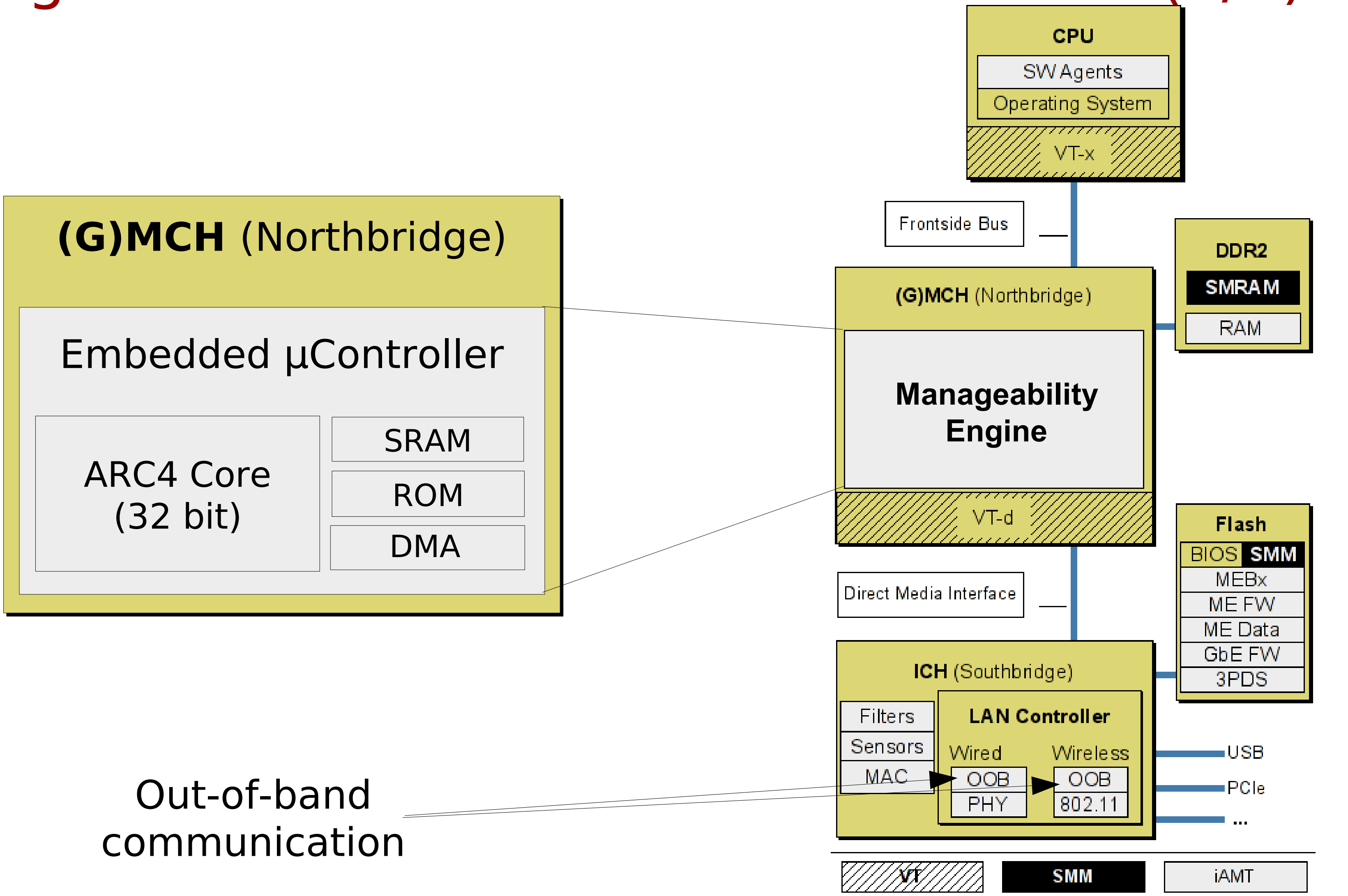
Introduction

Memory Controller Hub, Serial Peripheral Interface Chip,
“A Quest to Ring -3“ (cf. [Ter09])

- Rootkits:
  - Stealth
  - Isolated
  - ring 3 (user space)
    → ring 0 (kernel space)
    → “ring -1” (hypervisor/VMM)
    → “ring -2” (System Management Mode)
    → “ring -3” (Intel Active Management Technology)

- No ring -3 in hardware → “ring -3”
- Illustration: following the x86 ring protection model
- “Ring -3” rootkits related to modern x86 platforms

“Ring -3” Execution Environment (1/2)



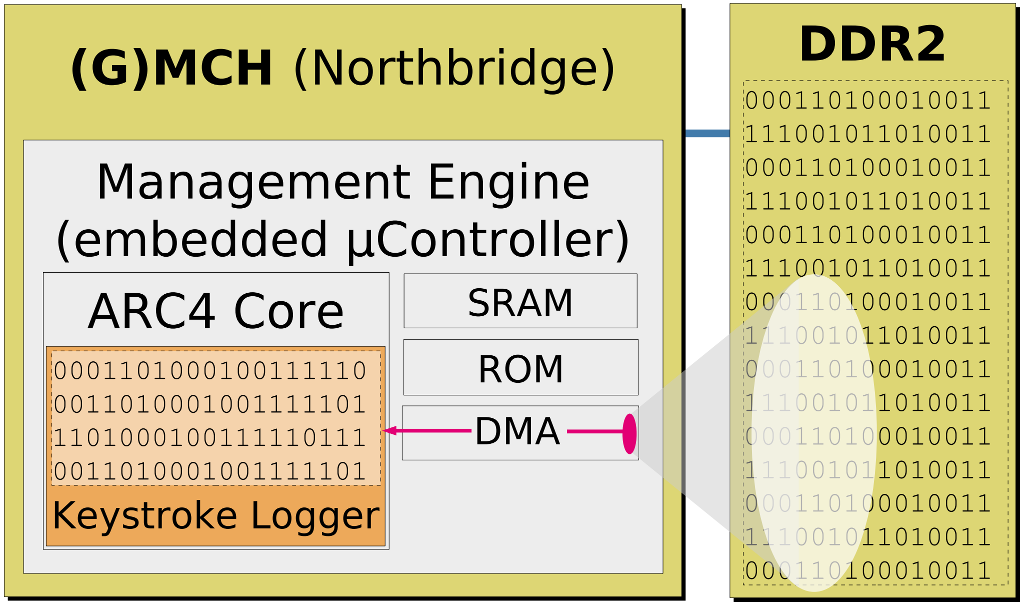
Out-of-band communication

“Ring -3” Execution Environment (2/2)

- Isolated execution environment implemented using an embedded µController
- Own flash memory storing:
  - µKernel
  - Drivers
  - Services
  - Applications
- Still working when platform powered down
- Active when turned off in BIOS
- More powerful than hypervisor or system management mode (SMM) based rootkits
- Actually intended for active platform management (cf. [Kum09])

Our “Ring -3” Rootkit (1/3)

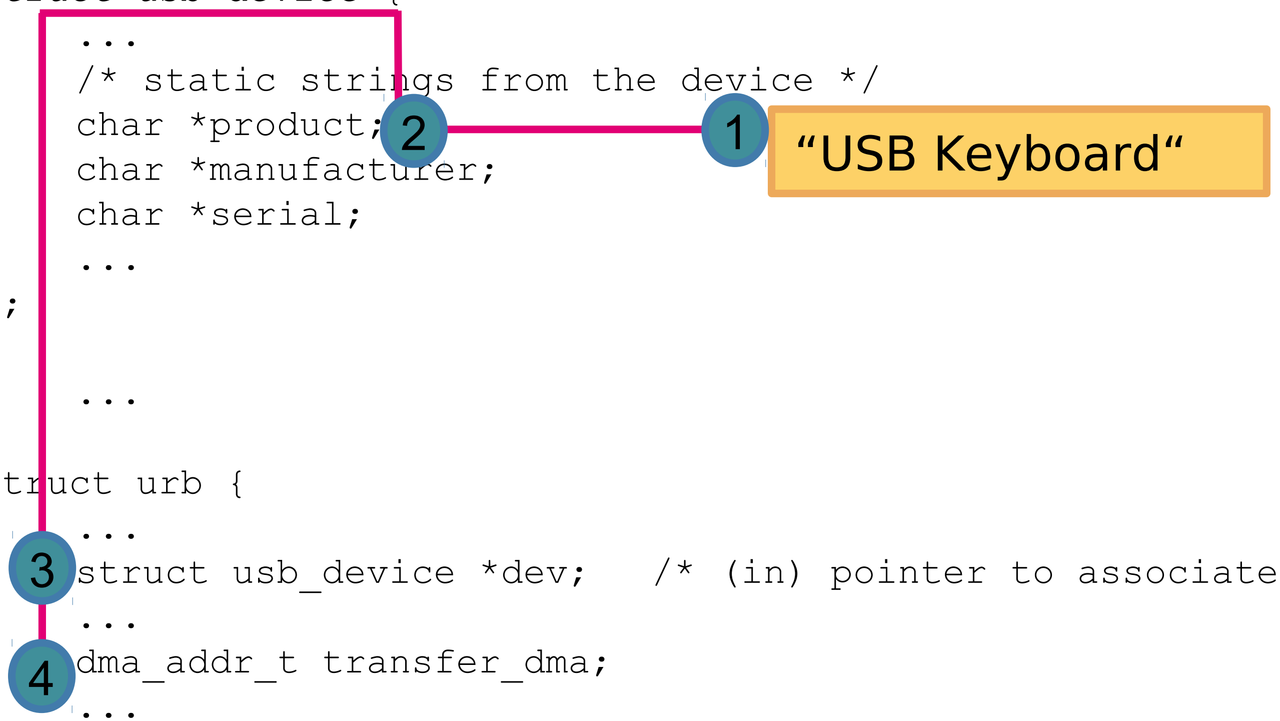
- USB Keystroke Logger for Linux operating system
  - Finds keyboard buffer
  - Monitors keyboard buffer constantly in background
  - Sends logged keystroke codes to external platform



Keystroke Logger executed in Isolated Execution Environment

Our “Ring -3” Rootkit (2/3)

- Computer forensic (find USB keyboard buffer)



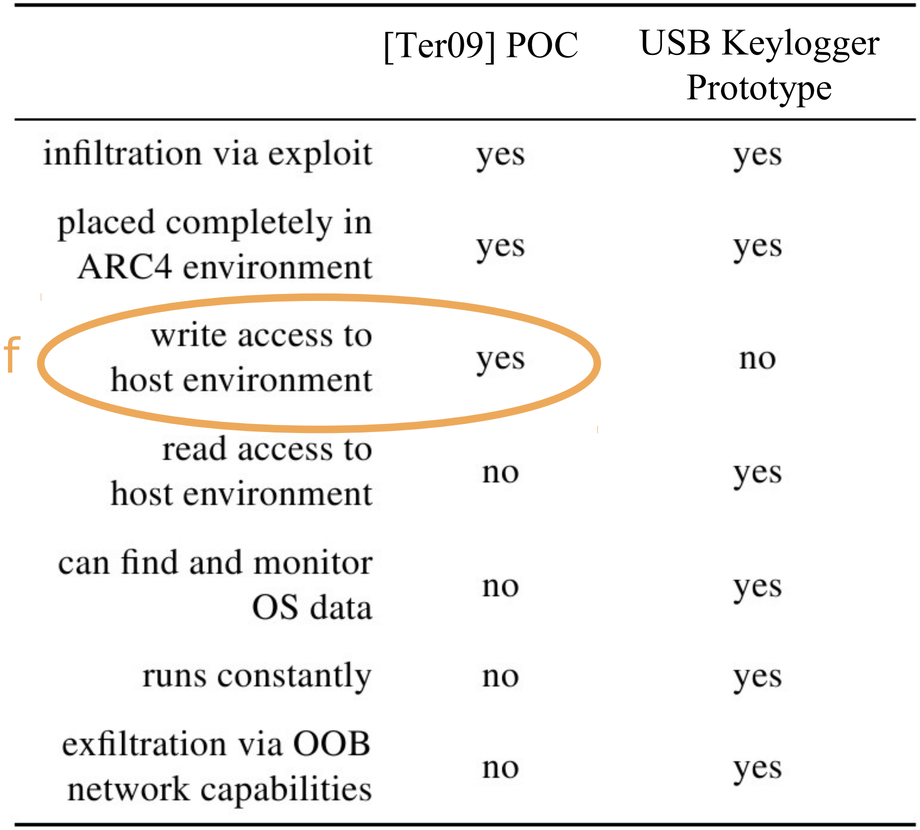
“USB Keyboard“

```
struct usb device {
     ...
     /* static strings from the device */
     char *product;
     char *manufacturer;
     char *serial;
     ...
};


     ...

struct urb {
     ...
     struct usb_device *dev;   /* (in) pointer to associated device */
     ...
     dma_addr_t transfer_dma;
     ...
};
```

USB Human Interface Device Structures in Host Memory
(cf. /usr/src/linux-source-2.6.31/include/linux/usb.h)

Our “Ring -3” Rootkit (3/3)

- Why not evaluating “ring -3” rootkit provided by [Ter09] ?



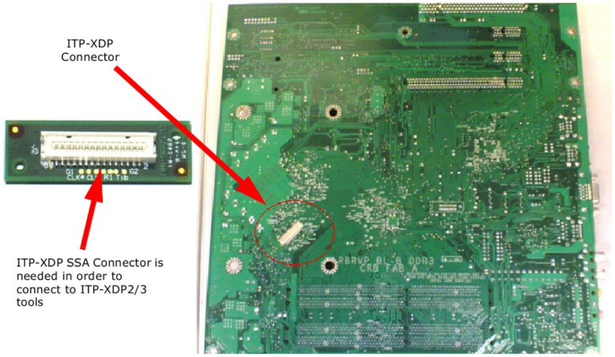
	[Ter09] POC	USB Keylogger Prototype
infiltration via exploit	yes	yes
placed completely in ARC4 environment	yes	yes
write access to host environment	yes	no
read access to host environment	no	yes
can find and monitor OS data	no	yes
runs constantly	no	yes
exfiltration via OOB network capabilities	no	yes

Reveals itself

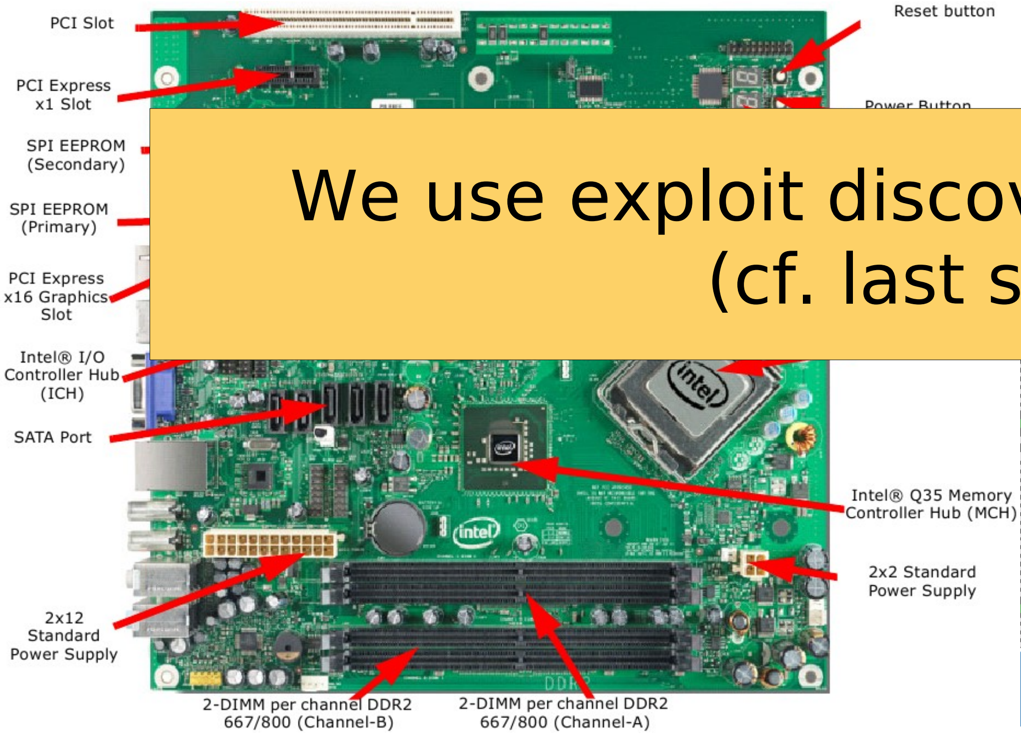
[Ter09] POC Rootkit compared to our USB Keylogger Prototype

Target Platform Infiltration

- No Intel developer board providing “ring -3” (and JTAG support, cf. [Bul08])


ITP-XDP Connector
ITP-XDP SSA Connector is needed in order to connect to ITP-XDP2/3 tools

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

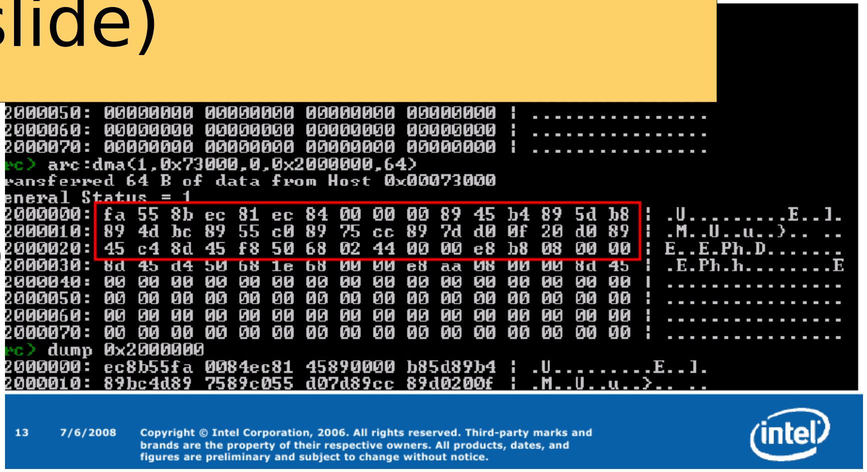

PCI Slot
PCI Express x1 Slot
SPI EEPROM (Secondary)
SPI EEPROM (Primary)
PCI Express x16 Graphics Slot
Intel® I/O Controller Hub (ICH)
SATA Port
2x12 Standard Power Supply
2-DIMM per channel DDR2 667/800 (Channel-B)
2-DIMM per channel DDR2 667/800 (Channel-A)
2x2 Standard Power Supply
Intel® Q35 Memory Controller Hub (MCH)
Power Button
Reset button

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

We use exploit discovered by [Ter09] (cf. last slide)

ger

alicious



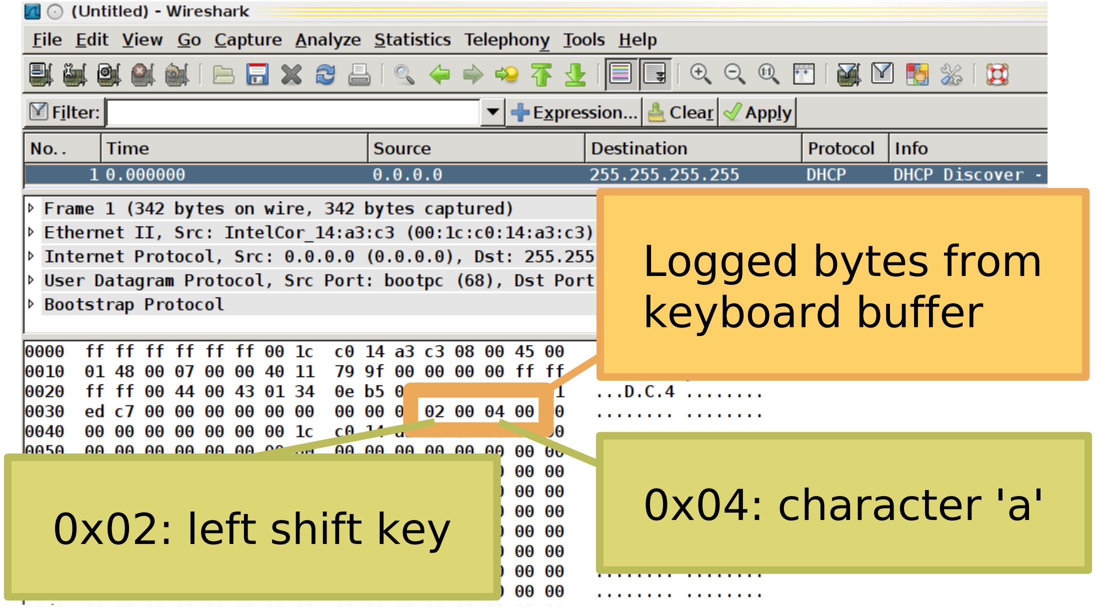
```
2000050: 00000000 00000000 00000000 00000000 ¦ ................
2000060: 00000000 00000000 00000000 00000000 ¦ ................
2000070: 00000000 00000000 00000000 00000000 ¦ ................
rc> arc:dma(1,0x73000,0,0x2000000,64)
ransferred 64 B of data from Host 0x00073000
eneral Status = 1
2000000: fa 55 8b ec 81 ec 84 00 00 00 89 45 b4 89 5d b8 ¦ .U.........E..].
2000010: 89 4d bc 89 55 c0 89 75 cc 89 7d d0 0f 20 d0 89 ¦ .M..U..u..}.. ..
2000020: 45 c4 8d 45 f8 50 68 02 44 00 00 e8 b8 08 00 00 ¦ E..E.Ph.D.......
2000030: 8d 45 d4 50 68 1e 68 00 00 e8 aa 08 00 00 8d 45 ¦ .E.Ph.h........E
2000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ¦ ................
2000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ¦ ................
2000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ¦ ................
2000070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ¦ ................
rc> dump 0x2000000
2000000: ec8b55fa 0084ec81 45890000 b85d89b4 ¦ .U.........E..].
2000010: 89bc4d89 7589c055 d07d89cc 89d0200f ¦ .M..U..u..}.. ..
```

13 7/6/2008 Copyright © Intel Corporation, 2006. All rights reserved. Third-party marks and brands are the property of their respective owners. All products, dates, and figures are preliminary and subject to change without notice.

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

Exfiltration of Collected Data

- Data collected by USB Keystroke Logger placed in a network packet
  - In our case DHCP discover
  - Sent via iAMT's OOB communication (invisible for host)



(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply

No. .	Time	Source	Destination	Protocol	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	DHCP Discover -

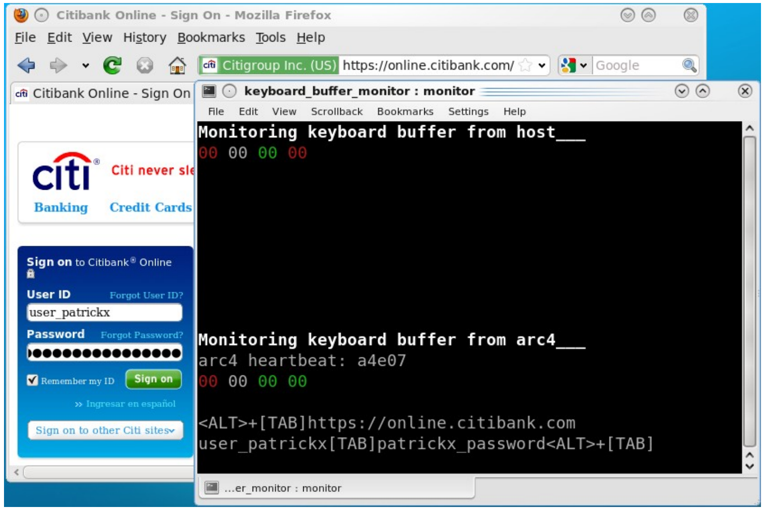
```
▹ Frame 1 (342 bytes on wire, 342 bytes captured)
▹ Ethernet II, Src: IntelCor_14:a3:c3 (00:1c:c0:14:a3:c3)
▹ Internet Protocol, Src: 0.0.0.0 (0.0.0.0), Dst: 255.255
▹ User Datagram Protocol, Src Port: bootpc (68), Dst Port
▹ Bootstrap Protocol

0000  ff ff ff ff ff ff 00 1c  c0 14 a3 c3 08 00 45 00
0010  01 48 00 07 00 00 40 11  79 9f 00 00 00 00 ff ff
0020  ff ff 00 44 00 43 01 34  0e b5 0            1   ...D.C.4 ........
0030  ed c7 00 00 00 00 00 00  00 00 0  02 00 04 00 0   ........ ........
0040  00 00 00 00 00 00 00 1c  c0                 0
0050  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   ........ ........
                                              00 00   ........ ........
```

Logged bytes from keyboard buffer

0x02: left shift key

0x04: character 'a'

Captured Network Packet containing Bytes from Keyboard Buffer



Citibank Online - Sign On - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Citigroup Inc. (US) https://online.citibank.com/
Google
Citibank Online - Sign On
citi
Citi never sle
Banking
Credit Cards
Sign on to Citibank® Online
Forgot User ID?
Forgot Password?
» Ingresar en español
Sign on to other Citi sites
keyboard_buffer_monitor : monitor
File Edit View Scrollback Bookmarks Settings Help
...er_monitor : monitor

User ID: user_patrickx

Password: ●●●●●●●●●●●●●●●●

☑ Remember my ID Sign on

```
Monitoring keyboard buffer from host___
00 00 00 00




Monitoring keyboard buffer from arc4___
arc4 heartbeat: a4e07
00 00 00 00

<ALT>+[TAB]https://online.citibank.com
user_patrickx[TAB]patrickx_password<ALT>+[TAB]
```

Keystroke Logger Demo: Online Banking Sign On

Important Related Work

- [Bul08] Y. Bulygin: Chipset based Approach to detect Virtualization Malware. TuCancUnix, (2008). [Online]. Available: `http://www.tucancunix.net/ceh/bhusa/BHUSA08/speakers/Bulygin_Detection_of_Rootkits/bh-us-08-bulygin_Chip_Based_Approach_to_Detect_Rootkits.pdf`
- [Int07] Intel Corporation, “Intel® CoreTM 2 Duo Processor and Intel ® Q35 Express Chipset Development Kit – User’s Manual,” Oct. 2007. [Online]. Available: `ftp://download. intel.com/design/intarch/MANUALS/318476.pdf`
- [Kum09] A. Kumar, P. Goel and Y. Saint-Hilaire, “Active Platform Management Demystified – Unleashing the power of Intel vPro Technology“, Intel Press, 2009.
- [Ste10] P. Stewin, J.-P. Seifert: “In God We Trust All Others We Monitor” [Extended Abstract]. In: CCS ’10: Proceedings of the 17th ACM Conference on Computer and Communications Security. ACM, p.639-641. (2010). [Online]. Availabe: `http://portal.acm.org/ft_gateway.cfm?id=au1866381&type=pdf&CFID=6743120&CFTOKEN=21999560`
- [Ter09] A. Tereshkin and R. Wojtczuk, *“Introducing Ring -3 Rootkits,”* Black Hat USA, Jul. 2009. [Online]. Available: `http://www.blackhat.com/presentations/bh-usa-09/TERESHKIN/BHUSA09-Tereshkin-Ring3Rootkit-SLIDES.pdf`

Summary

- Stealth USB Keystroke Logger
  - Isolated from host environment → AV software unable to find keystroke logger
- Monitors Linux OS (currently ported to Windows OS)
  - Finds keyboard buffer
  - Collects keystroke codes
  - Exfiltrates keystroke codes via isolated network channel
- Current prototype can be detect using second platform
  - See future work ...

Future Work

- Detection Mechanism for Host Platform
  - [Ter09] discussed countermeasures, but
    - Also provide approaches to defeat countermeasures (cf. virtual CDROM)
  - First detection approaches:
    - Provoke deleays when accessing same resources:
      - Memory ?
      - Network ?
      - Bus Master ?
- Evaluate Windows version of our keystroke logger
- Implementation of covert timing channel (e.g., JitterBug)

Questions?

Thank you!