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

- 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

(G)MCH (Northbridge)

Embedded µController

ARC4 Core (32 bit)

SRAM

ROM

DMA

Manageability Engine

VT-x

Frontside Bus

DDR2

SMRAM

RAM

Flash

BIOS

SMM

MEBx

ME FW

ME Data

GbE FW

3PDS

ICH (Southbridge)

LAN Controller

Filters

Sensors

MAC

Wired

Outdoor OOB

PHY

Wireless

Outdoor OOB

802.11

BIST

SMM

IAMT

USB

PCIe

...
“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

- 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

- Computer forensic (find USB keyboard buffer)

```c
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

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

<table>
<thead>
<tr>
<th>Feature</th>
<th>[Ter09] POC</th>
<th>USB Keylogger Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>infiltration via exploit</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>placed completely in ARC4 environment</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>write access to host environment</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>read access to host environment</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>can find and monitor OS data</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>runs constantly</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>exfiltration via OOB network capabilities</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

[Ter09] POC Rootkit compared to our USB Keylogger Prototype
Target Platform Infiltration

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

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

Logged bytes from keyboard buffer

Captured Network Packet containing Bytes from Keyboard Buffer

Keystroke Logger Demo: Online Banking Sign On

0x02: left shift key

0x04: character 'a'
Important Related Work


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 delays when accessing same resources:
      - Memory ?
      - Network ?
      - Bus Master ?

- Evaluate Windows version of our keystroke logger
- Implementation of covert timing channel (e.g., JitterBug)
Thank you!

Questions?