SDIO: a new peripheral attack vector

Research Project 2

Thom Does
thom.does@os3.nl

Dana Geist
dana.geist@os3.nl
Introduction

- Secure Digital Input Output (SDIO)
  - Adds I/O functions to SD
  - PDAs, tablets, laptops
- SDIO presents similarities with USB
- BadUSB attack (2014)
  - Inject keystrokes
  - Rogue DHCP
- Seemingly no protections to prevent BadUSB-like attacks
Research Question

Could SDIO be used as a new attack vector on SDIO-aware hosts?
State of art

- No previous research on SDIO as an attack vector
- SD/SDIO specifications
  - Only simplified version available without license
- SD card hack (2013)
  - Several microSD cards were tested
  - Reversed engineered firmware
  - Developed novel applications for microcontroller
Attack path: WLAN SDIO card

Step 1
Attack path: WLAN SDIO card

Step 1
Attack path: WLAN SDIO card

Step 1
Attack path: WLAN SDIO card

Step 2
Attack path: WLAN SDIO card

Step 3
SDIO Stack
# Physical Layer: SPI vs. SD

<table>
<thead>
<tr>
<th>SPI</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wide variety of applications</td>
<td>Used by SD cards and readers</td>
</tr>
<tr>
<td>Well-known “open” protocol</td>
<td>License required</td>
</tr>
<tr>
<td>Simple (one data line)</td>
<td>More complex (commands, data lines)</td>
</tr>
<tr>
<td>Supported natively by many MCUs</td>
<td>Special purpose MCUs or bitbanging</td>
</tr>
<tr>
<td>Fallback protocol for SD</td>
<td>Default for SD</td>
</tr>
</tbody>
</table>

![Diagram showing Business Logic, SDIO, SPI, and SD](image)
SDIO Layer

- Maintained by SD association
- Documentation requires licensing
- Defines SDIO commands
  - Formats
  - Initialization
  - Transfer modes
- Master-Slave based protocol
  - The card reader is the Master
  - The SDIO card is the slave
Business Logic Layer

- Multiple manufacturers
- Standardized and manufacturer specific interfaces
  - Firmware
  - Drivers
- Each interface is an attack surface
  - WLAN, bluetooth, GPS
- Manipulate higher-level applications
  - DHCP-client, command injection, navigation system
SDIO Model
SDIO Model: Host’s drivers

- Manufacturer Drivers (wilc100, wilc100-sdio)
- Generic OS Drivers (mmc, sd, sdio)
- Card reader Drivers (rtsx_pci_sdmmc)
How can the host system be exploited?

Target the card's firmware
How can the host system be exploited?

- Target the card’s firmware
  - Build SDIO-capable device from scratch
  - Modify existing firmware
How can the host system be exploited?

1. Target the card's firmware
2. Build SDIO-capable device from scratch
3. Modify existing firmware
4. Reverse engineer firmware
Build SDIO device from scratch (SPI)

If host supports SPI

- Use low cost microcontrollers to implement protocol
- Build low cost sniffers to ease the development
- Use open source software to analyze the protocol
- Not all hosts support SPI
Build SDIO device from scratch (SD)

If host supports SD only

- Most microcontrollers do not natively support the protocol
- Using commodity hardware for bitbanging could be cumbersome
- No open source protocol analyzers tools
- Complex solutions like FPGA + IP core software
  - Expensive
  - Steep learning curve
  - Requires business logic programming
Modify existing firmware

- Get the firmware
- Find hooking points
- Rewrite specific functions
- Two main options:
  - Firmware embedded in SD card
  - Firmware loaded to device by the driver
## SDIO-based vs. USB-based attacks

<table>
<thead>
<tr>
<th></th>
<th>SDIO attack</th>
<th>BadUSB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hosts</strong></td>
<td>Laptops, tablets, PDAs</td>
<td>Desktops, laptops, printers, routers</td>
</tr>
<tr>
<td><strong>Devices</strong></td>
<td>Limited vendors and applications</td>
<td>Many vendors and applications</td>
</tr>
<tr>
<td><strong>Stealthiness</strong></td>
<td>Embedded in port</td>
<td>Protruding from port</td>
</tr>
<tr>
<td><strong>Ease of exploitation</strong></td>
<td>No “off-the-shelf” products</td>
<td>USB Armory, Rubber Ducky, known vulnerable firmware</td>
</tr>
</tbody>
</table>
Discussion

● Licensing required by SD Association
● Attack feasible but:
  ○ Time consuming
  ○ Expensive
  ○ Not possible to create general purpose malicious firmware
● Likelihood
  ○ Affects SDIO aware hosts only
  ○ Kernel module needs to be loaded
● Impact:
  ○ Wide range of attacks possible
● Mitigation: vendors should sign or encrypt their firmware
Conclusions

- SDIO cards are supported by various types of hosts
  - Laptops, phones, tablets, PDAs
- SDIO is an attack vector
  - No protections found
- Firmware might be modified, or developed from scratch
  - SD is more effective than SPI
- Currently, SDIO-based attacks seem less likely than USB-based attacks
  - Ease of exploitation
  - Number of vendors / products supporting SDIO
  - Kernel module needs to be loaded
References

Research material:

- SDIO specifications
- BADUSB - On Accessories that Turn Evil by Karsten Nohl + jakob Lell
  - https://www.youtube.com/watch?v=nuruzFqMglw
- SD card hack
- A Microcontroller-based HF-RFID Reader Implementation for the SD-Slot

Images:

- https://www.parallelula.org/create-sdcard/
- http://www.actel.com/ipdocs/iW-SDIO_Slave_demo_board_DS.pdf
- https://www.sdcard.org/developers/overview/sdio/index.html