BadUSB — On accessories that turn evil

Karsten Nohl <nohl@srlabs.de>
Sascha Krißler <sascha@srlabs.de>
Jakob Lell <jakob@srlabs.de>
Demo 1 – **USB stick takes over Windows machine**
Agenda

- USB background
  - Reprogramming peripherals
  - BadUSB attack scenarios
  - BadUSB exposure
  - Defenses and next steps
USB devices are recognized using several identifiers

**Examples**

<table>
<thead>
<tr>
<th>Identifier</th>
<th>USB thumb drive</th>
<th>Webcam</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface class</td>
<td>8 – Mass Storage</td>
<td>a. 1 – Audio</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b. 14 – Video</td>
</tr>
<tr>
<td>End points</td>
<td>0 – Control</td>
<td>0 – Control</td>
</tr>
<tr>
<td></td>
<td>1 – Data transfers</td>
<td>1 – Video transfers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 – Audio transfers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 – Video interrupts</td>
</tr>
<tr>
<td>Serial number (optional)</td>
<td>AA627090820000000702</td>
<td>0258A350</td>
</tr>
</tbody>
</table>
USB devices are initialized in several steps

USB device

Power-on + Firmware init

USB plug-and-play

Register
Set address
Send descriptor
Set configuration
Normal operation
Optional: deregister
Register again ...

Load driver

Load another driver

Devices can have several identities

- A device indicates its capabilities through a descriptor
- A device can have several descriptors if it supports multiple device classes; like webcam + microphone
- Device can deregister and register again as a different device
USB devices include a micro-controller, hidden from the user

The only part visible to the user
Agenda

- USB background
- Reprogramming peripherals
  - BadUSB attack scenarios
  - BadUSB exposure
- Defenses and next steps
A) Document firmware update process

1. Find leaked firmware and flash tool on the net
2. Sniff update communication using Wireshark
3. Replay custom SCSI commands used for updates
4. (Reset bricked devices through short-circuiting Flash pins)

B) Reverse-engineer firmware

1. Load into disassembler (complication: MMU-like memory banking)
2. Apply heuristics:
   - Count how often function starts match up with function calls for different memory location guesses; the most matches indicate that you guessed right
   - Find known USB bit fields such as descriptors
3. Apply standard software reversing to find hooking points

C) Patch firmware

1. Add hooks to firmware to add/change functionality
2. Custom linker script compiles C and assembly code and injects it into unused areas of original firmware

Other possible targets
We focused on USB sticks, but the same approach should work for:
- External HDDs
- Webcams, keyboards
- Probably many more ...
Agenda

- USB background
- Reprogramming peripherals
- **BadUSB attack scenarios**
- BadUSB exposure
- Defenses and next steps
Demo 2 – Windows infects USB stick which then takes over Linux machine
Keyboard emulation is enough for infection and privilege escalation (w/o need for software vulnerability)

<table>
<thead>
<tr>
<th>Challenge – Linux malware runs with limited user privileges, but needs root privileges to infect further sticks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Approach – Steal sudo password in screensaver</strong></td>
</tr>
<tr>
<td>Restart screensaver (or policykit) with password stealer added via an LD_PRELOAD library</td>
</tr>
<tr>
<td>▪ User enters password to unlock screen</td>
</tr>
<tr>
<td>▪ Malware intercepts password and gains root privileges using sudo</td>
</tr>
</tbody>
</table>
Demo 3 – Android phone changes DNS settings in Windows
Network traffic can also be diverted by “DHCP on USB”

DNS assignment in DHCP over spoofed USB-Ethernet adapter

All DNS queries go to attacker’s DNS server

---

**Attack steps**

1. USB stick spoofs Ethernet adapter
2. Replies to DHCP query with DNS server on the Internet, but without default gateway

**Result**

3. Internet traffic is still routed through the normal Wi-Fi connection
4. However, DNS queries are sent to the USB-supplied server, enabling redirection attacks
“Can I charge my phone on your laptop?” – Android phones are the simplest USB attack platform

**Preparation** – Android comes with an Ethernet-over-USB emulation needing little configuration

**Attack** – Phone supplies default route over USB, effectively intercepting all Internet traffic

DHCP overrides default gateway over **USB-Ethernet**

Computer sends all Internet traffic through phone

Proof-of-concept released at: [srlabs.de/badusb](srlabs.de/badusb)

**Hacked by the second factor?**

Using keyboard emulation, a virus-infected smartphone could hack into the USB-connected computer.

This compromises the “second factor” security model of online banking.
Bonus: Virtual Machine break-out

1. VM tenant reprograms USB device (e.g., using SCSI commands)

2. USB peripherals spawns a second device that gets connected to the VM host

3. USB device spoofs key strokes, changes DNS, ...
Boot-sector virus, USB style

Fingerprint OS/BIOS.
Patched USB stick firmware can distinguish Win, Mac, Linux, and the BIOS based on their USB behavior

Hide rootkit from OS/AV.
When an OS accesses the stick, only the USB content is shown

Infect machine when booting.
When the BIOS accesses the stick, a secret Linux is shown, booting a root kit, infecting the machine, and then booting from hard disk

USB content, for example Linux install image

Secret Linux image
Demo 4 – USB thumb drive emulates keyboard and second drive to infect computer during boot
Family of possible USB attacks is large

<table>
<thead>
<tr>
<th>Attacks shown</th>
<th>More attack ideas</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emulate keyboard</td>
<td>Hide data on stick or HDD</td>
<td>- External storage can choose to hide files instead of deleting them</td>
</tr>
<tr>
<td>Spoof network card</td>
<td>Rewrite data in-flight</td>
<td>- Viruses can be added to files added to storage</td>
</tr>
<tr>
<td>“USB boot-sector” virus</td>
<td>Update PC BIOS</td>
<td>- First access by virus scanner sees original file, later access sees virus</td>
</tr>
<tr>
<td></td>
<td>Spoof display</td>
<td>- Emulate a keyboard during boot and install a new BIOS from a file in a secret storage area on a USB stick</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Emulate a USB display to access security information such as Captchas and randomly arranged PIN pads</td>
</tr>
</tbody>
</table>
Agenda

- USB background
- Reprogramming peripherals
- BadUSB attack scenarios
- BadUSB exposure
- Defenses and next steps
We analyzed the possible reach of BadUSB from two perspectives

<table>
<thead>
<tr>
<th>Top-down analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ Start from largest USB controller vendors</td>
</tr>
<tr>
<td>▪ Find their chip families for popular use cases</td>
</tr>
<tr>
<td>▪ Analyze datasheets and web sites for whether chips can be reprogrammed</td>
</tr>
<tr>
<td>▪ 5 device classes: Host, Hub, Charger, Storage, Peripheral</td>
</tr>
<tr>
<td>▪ From top 8 chip vendors</td>
</tr>
<tr>
<td>▪ Totaling 52 chip families (not every vendor serves each class)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bottom-up analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ Start from actual hardware</td>
</tr>
<tr>
<td>▪ Open device to find which chips are used</td>
</tr>
<tr>
<td>▪ Determine whether bootloader and firmware storage (e.g. SPI flash) are available</td>
</tr>
<tr>
<td>▪ Try to find firmware update tools for their chips</td>
</tr>
<tr>
<td>▪ Analyzed 33 devices from six device classes: Hub, Input/HID, Webcam, SD adapter, SATA adapter</td>
</tr>
<tr>
<td>▪ Results released at opensource.srlabs.de</td>
</tr>
</tbody>
</table>
Both analyses suggest that up to half of USB chips are BadUSB-vulnerable
Small hardware design differences can determine BadUSB-vulnerability

These USB hubs both contain the same controller chip

Only one of them also contains an SPI flash that can store BadUSB modifications
Recent trends suggest that BabUSB-exposure is further growing

**Insight**

- Some device types appear more reprogrammable / BadUSB-vulnerable:
  - The early devices of a new standard (e.g. the first available USB 3 devices)
  - Peripherals with special functionality (e.g. SATA adapter that can copy disks)
  - High-end peripherals

- Custom-tailored chips in high-volume devices were traditionally less likely to be reprogrammable; probably because mask ROMs are cheaper than Flash
  - Many such use cases are increasingly served with *reprogrammable* multi-purpose chips, that realize economies of scale by combining applications

- USB controllers found not to be reprogrammable were missing an essential component for upgrades, such as bootloader or Flash to store the update
  - All those controllers that bring the essentials seem to be upgradable
  - Protection from malicious updates is very rare: Only one (large) chip family brings fuse bits; none implement firmware signing
Agenda

- USB background
- Reprogramming peripherals
- BadUSB attack scenarios
- BadUSB exposure

Defenses and next steps
# No effective defenses from USB attacks exist

<table>
<thead>
<tr>
<th>Protection idea</th>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Whitelist USB devices</strong></td>
<td>- USB devices do not always have a unique serial number</td>
</tr>
<tr>
<td></td>
<td>- OS’s don’t (yet) have whitelist mechanisms</td>
</tr>
<tr>
<td><strong>Block critical device classes, block USB completely</strong></td>
<td>- Obvious usability impact</td>
</tr>
<tr>
<td></td>
<td>- Very basic device classes can be used for abuse; not much is left of USB when these are blocked</td>
</tr>
<tr>
<td><strong>Scan peripheral firmware for malware</strong></td>
<td>- The firmware of a USB device can typically only be read back with the help of that firmware (if at all): A malicious firmware can spoof a legitimate one</td>
</tr>
<tr>
<td><strong>Use code signing for firmware updates</strong></td>
<td>- Implementation errors may still allow installing unauthorized firmware upgrades</td>
</tr>
<tr>
<td></td>
<td>- Secure cryptography is hard to implement on small microcontrollers</td>
</tr>
<tr>
<td></td>
<td>- Billions of existing devices stay vulnerable</td>
</tr>
<tr>
<td><strong>Disable firmware updates in hardware</strong></td>
<td>- <strong>Simple and effective</strong> (but mostly limited to new devices)</td>
</tr>
</tbody>
</table>
Responsibility for BadUSB mitigation is unclear

<table>
<thead>
<tr>
<th>No response from chip vendors</th>
<th>Fixes are not yet in sight</th>
<th>vs. BadUSB malware becomes more realistic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>▪ Phison, the mostly discussed vendor, notes that they are already offering better chips. Their customers don’t seem to chose them often</td>
<td>▪ Sample exploit code for Phison USB 3 controllers was released by Adam Caudill and Brandon Wilson at Derbycon in September</td>
</tr>
<tr>
<td></td>
<td>▪ Other affected vendors have stayed quiet</td>
<td>▪ Only mitigation attempts right now are quick fixes such as GData’s Keyboard Guard</td>
</tr>
<tr>
<td></td>
<td>▪ No affected vendor offers patches or a threat advisory</td>
<td></td>
</tr>
<tr>
<td>No response from peripheral vendors</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>▪ No OS implementers do not appear to work on solution; with one exception: FreeBSD adds an option to switch off USB enumeration</td>
<td></td>
</tr>
</tbody>
</table>

SECURITY RESEARCH LABS
USB peripherals can also be re-programmed for constructive purposes

Idea 1 – Speed up database queries

- Data can be parsed on the stick before (or instead of) sending it back to the host
- Our original motivation was to speed up of A5/1 rainbow table lookups

Idea 2 – Repurpose cheap controller chips

- Use the reprogrammable chips for other applications than USB storage
- The flowswitch / phison project, for example, aims for a low-cost USB 3 interface for FPGAs
Take aways

- **USB peripherals provide for a versatile infection path**
- Once infected – through USB or otherwise – malware can use peripherals as a *hiding place*, hindering system clean-up
- As long as USB controllers are re-programmable, USB peripherals should *not be shared* with others

Questions?

usb@srlabs.de
The USB microcontroller market is split among many vendors.

**Wired USB Market Share**
(2012 Cypress Shareholders Meeting)

Source: goo.gl/NtN0cf