BadUSB — On accessories that turn evil

Karsten Nohl <nohl@srlabs.de>
Sascha Krißler <sascha@srlabs.de>
Jakob Lell <jakob@srlabs.de>

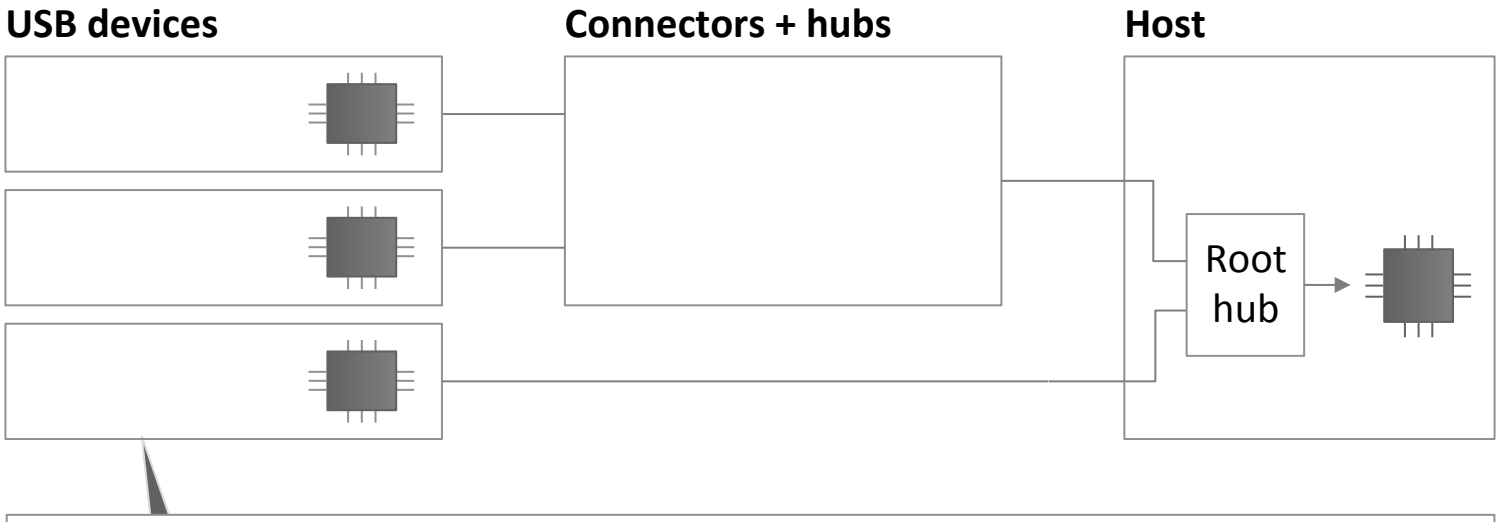
SECURITY
RESEARCH
LABS

Demo 1 – **USB stick takes over Windows machine**

Agenda

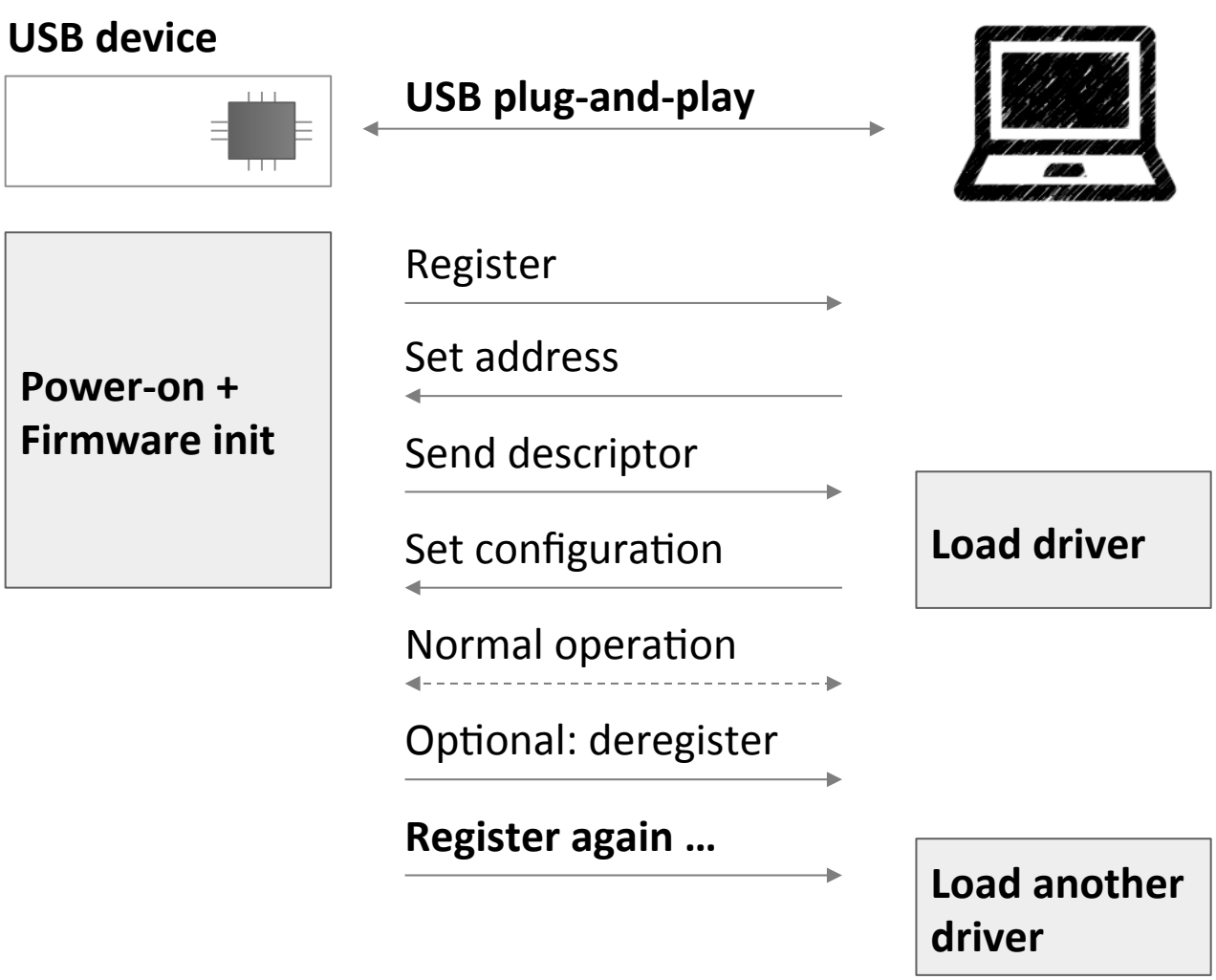
- **USB background**
- Reprogramming peripherals
- BadUSB attack scenarios
- BadUSB exposure
- Defenses and next steps

USB devices are recognized using several identifiers



	Examples	
Identifier	USB thumb drive	Webcam
Interface class	8 – Mass Storage	a. 1 – Audio b. 14 – Video
End points	0 – Control 1 – Data transfers	0 – Control 1 – Video transfers 6 – Audio transfers 7 – Video interrupts
Serial number (optional)	AA6270908200000000702	0258A350

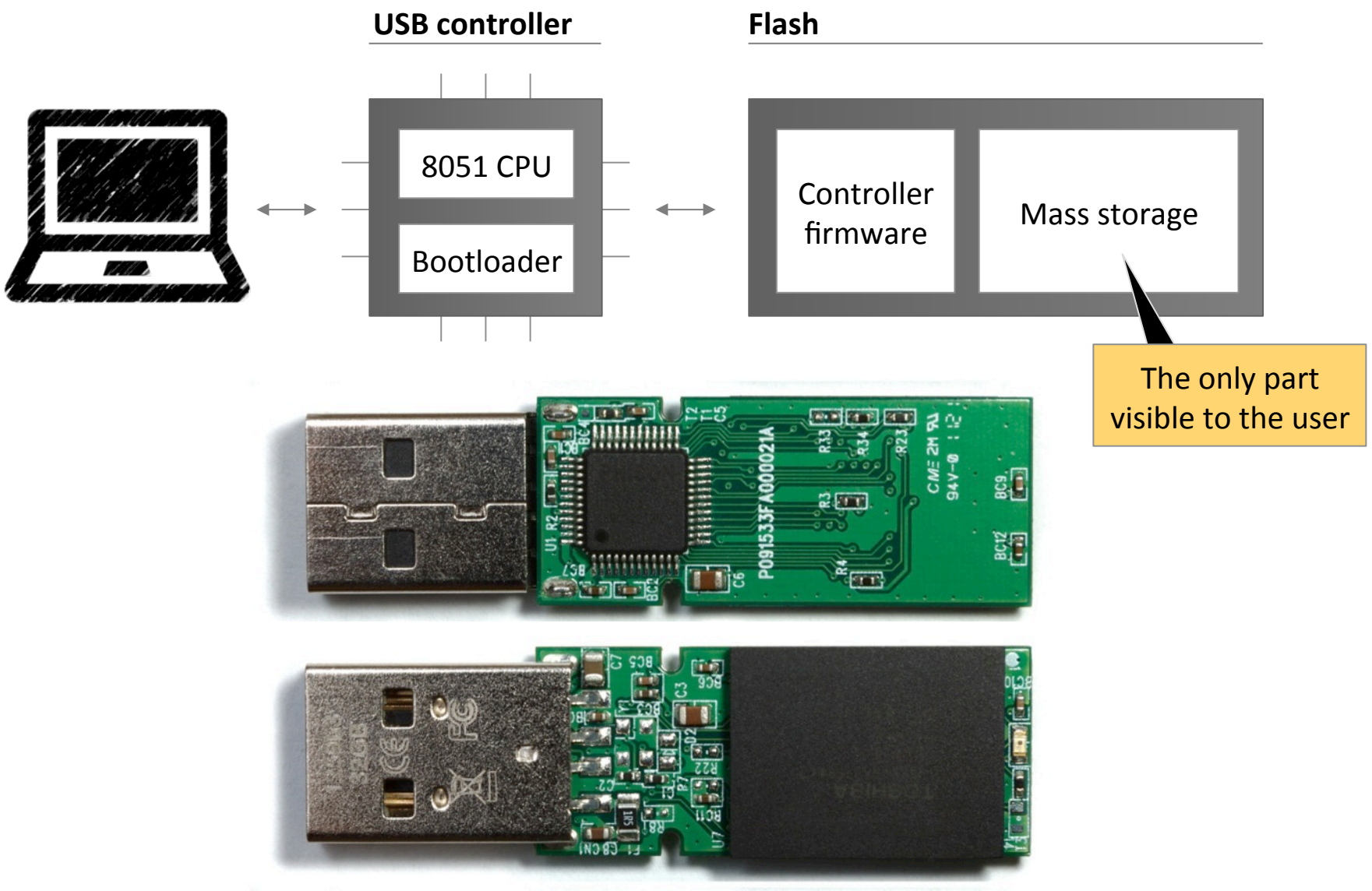
USB devices are initialized in several steps



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



Agenda

- USB background
- **Reprogramming peripherals**
- BadUSB attack scenarios
- BadUSB exposure
- Defenses and next steps

Reversing and patching USB firmware took 2 months

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)

Challenge – Linux malware runs with limited user privileges, but needs *root* privileges to infect further sticks

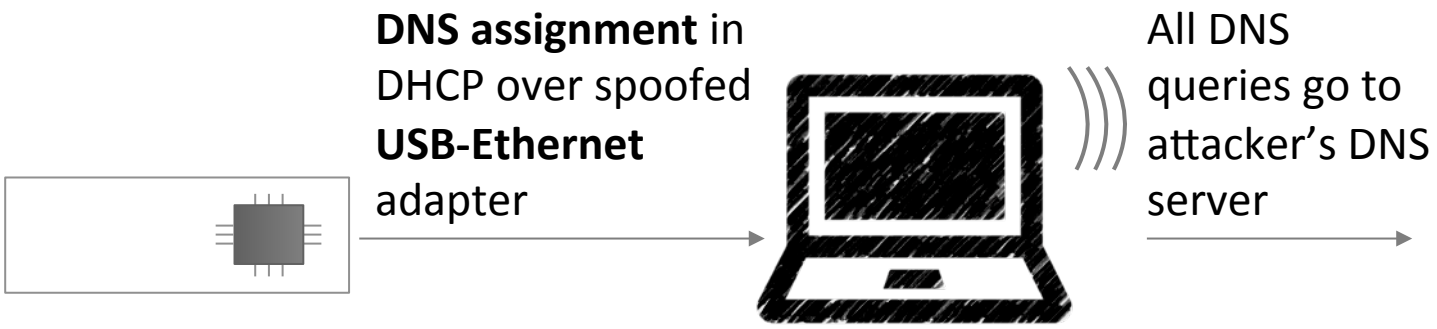
Approach – Steal *sudo* password in screensaver

Restart screensaver (or *policykit*) with password stealer added via an LD_PRELOAD library

- User enters password to unlock screen
- Malware intercepts password and gains root privileges using *sudo*

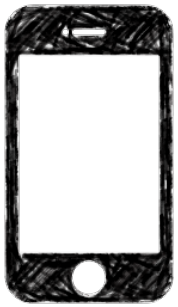
Demo 3 – **Android phone changes DNS settings in Windows**

Network traffic can also be diverted by “DHCP on USB”



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



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

Computer sends all Internet traffic through phone

Proof-of-concept released at: **srlabs.de/badusb**

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

▽

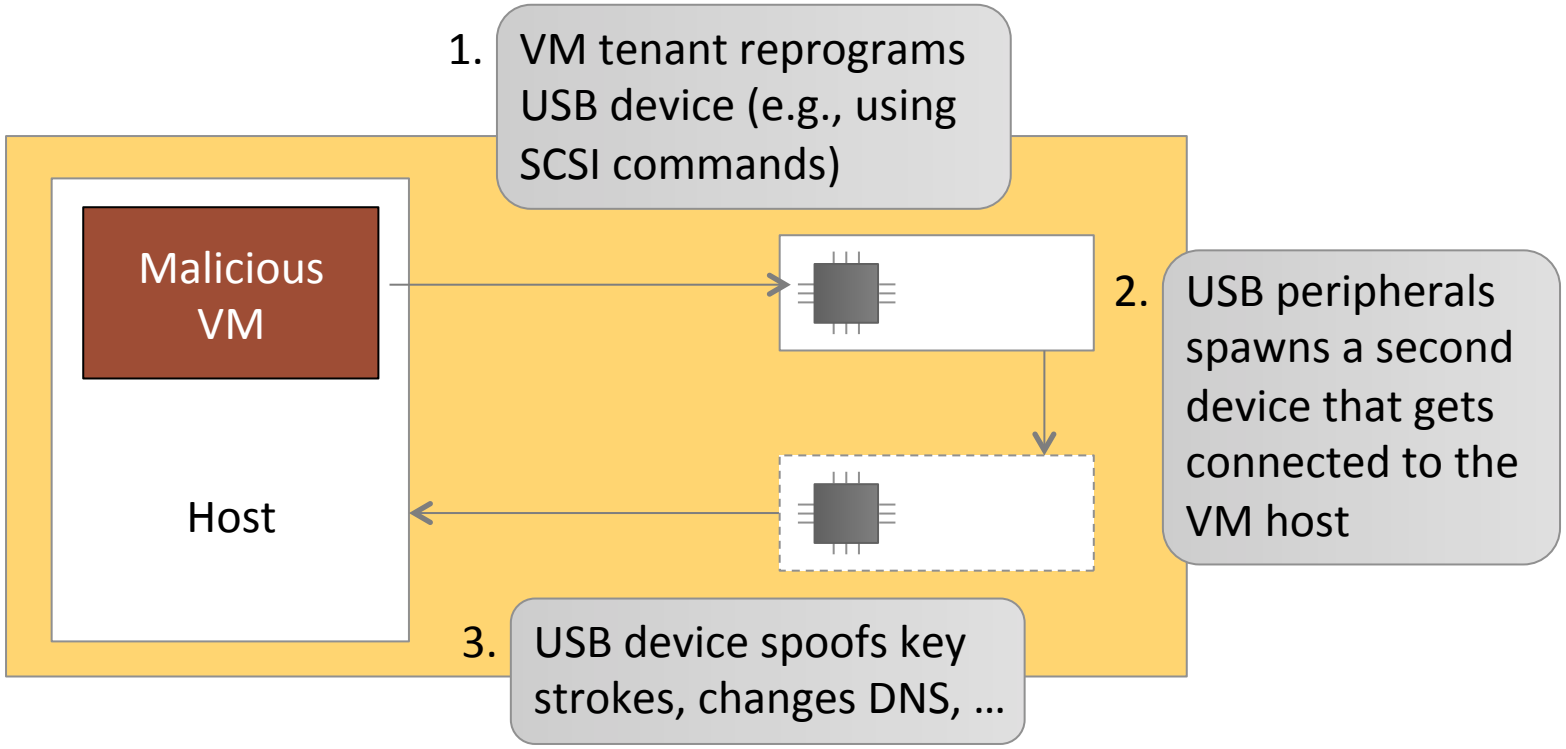
Attack – Phone supplies default route over USB, effectively intercepting all Internet traffic

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

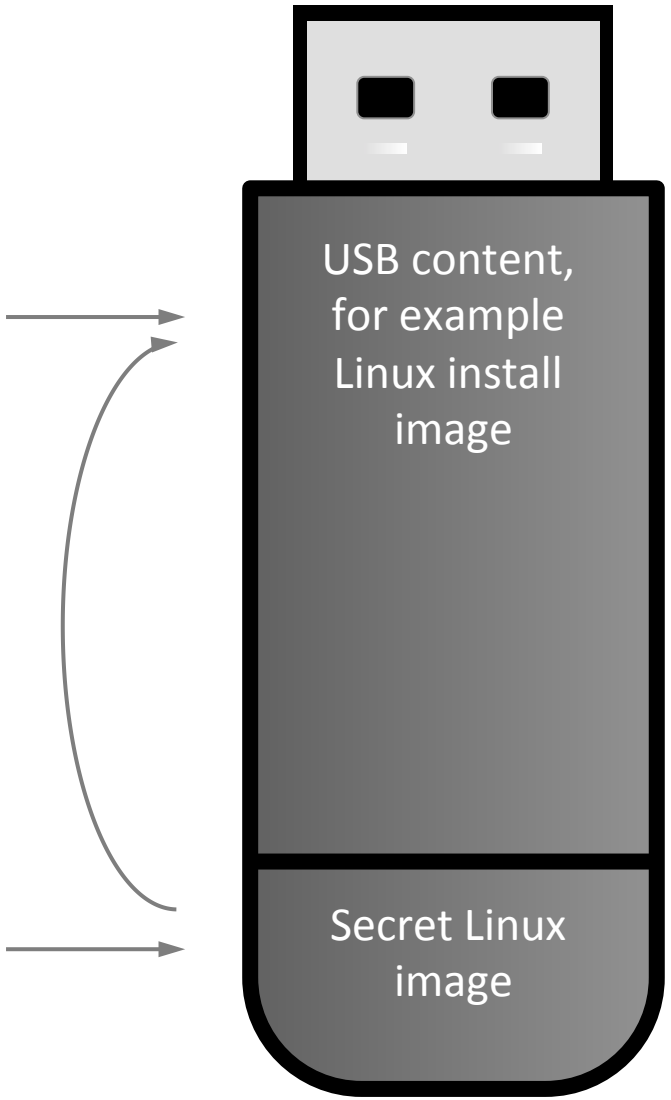


Boot-sector virus, USB style

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



Demo 4 – **USB thumb drive emulates keyboard and second drive to infect computer during boot**

Family of possible USB attacks is large

Attacks shown

- Emulate keyboard
- Spoof network card
- "USB boot-sector" virus

More attack ideas	Effect
Hide data on stick or HDD	▪ External storage can choose to hide files instead of deleting them
Rewrite data in-flight	▪ Viruses can be added to files added to storage ▪ First access by virus scanner sees original file, later access sees virus
Update PC BIOS	▪ Emulate a keyboard during boot and install a new BIOS from a file in a secret storage area on a USB stick
Spoof display	▪ Emulate a USB display to access security information such as Captchas and randomly arranged PIN pads

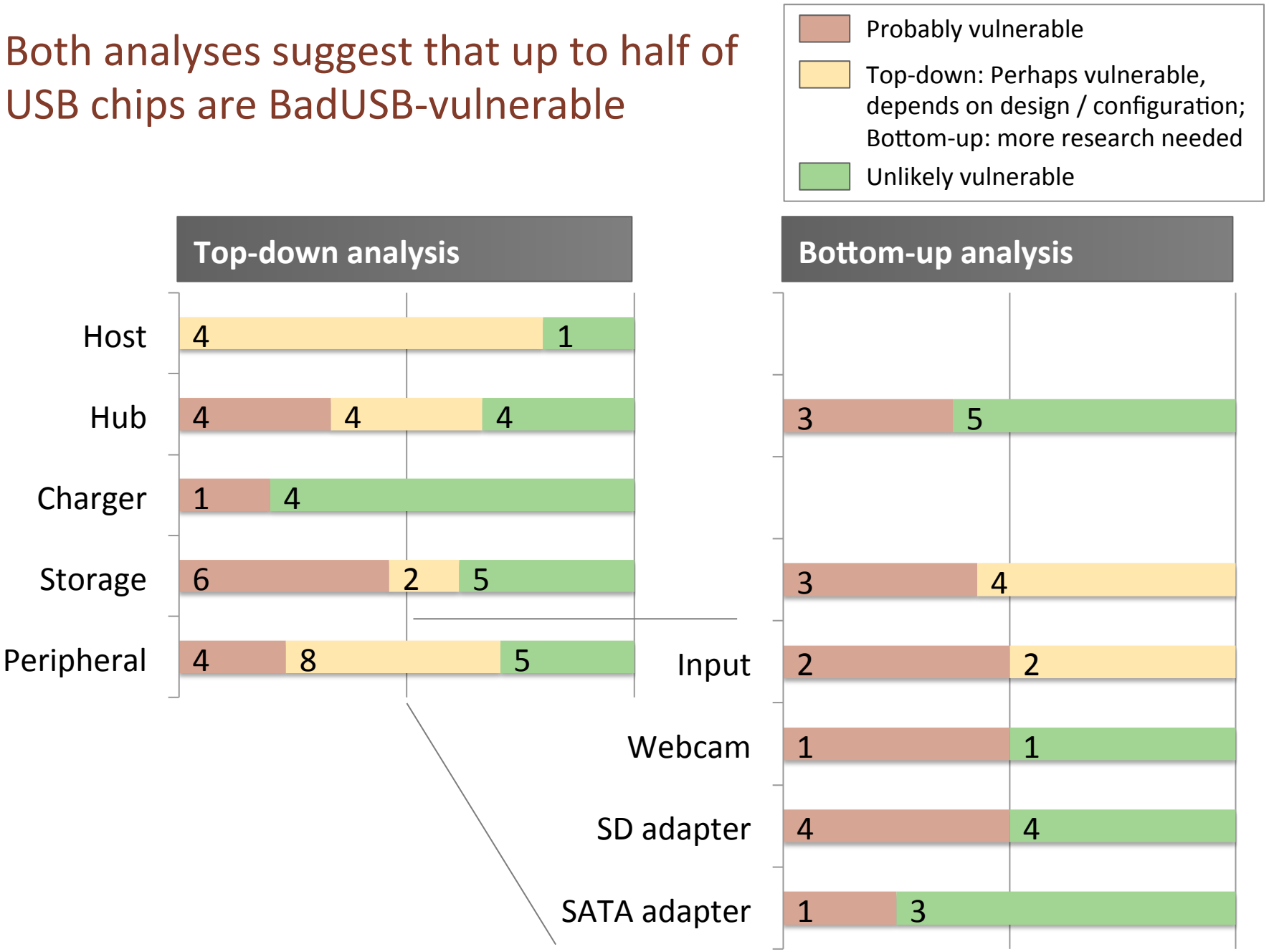
Agenda

- USB background
- Reprogramming peripherals
- BadUSB attack scenarios
- **BadUSB exposure**
- Defenses and next steps

We analyzed the possible reach of BadUSB from two perspectives

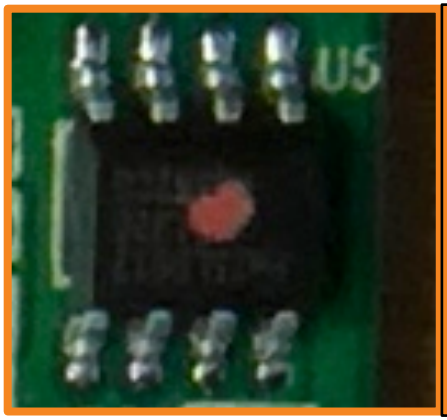
Top-down analysis	Bottom-up analysis
▪ Start from largest USB controller **vendors** ▪ Find their chip families for popular use cases ▪ Analyze **datasheets** and web sites for whether chips can be reprogrammed	▪ Start from actual **hardware** ▪ Open device to find which chips are used ▪ Determine whether bootloader and firmware storage (e.g. SPI flash) are available ▪ Try to find **firmware update tools** for their chips
▪ 5 device classes: Host, Hub, Charger, Storage, Peripheral ▪ From top 8 chip vendors ▪ Totaling **52 chip families** (not every vendor serves each class)	▪ Analyzed **33 devices** from six device classes: Hub, Input/HID, Webcam, SD adapter, SATA adapter ▪ Results released at **`opensource.srlabs.de`**

Both analyses suggest that up to half of USB chips are BadUSB-vulnerable

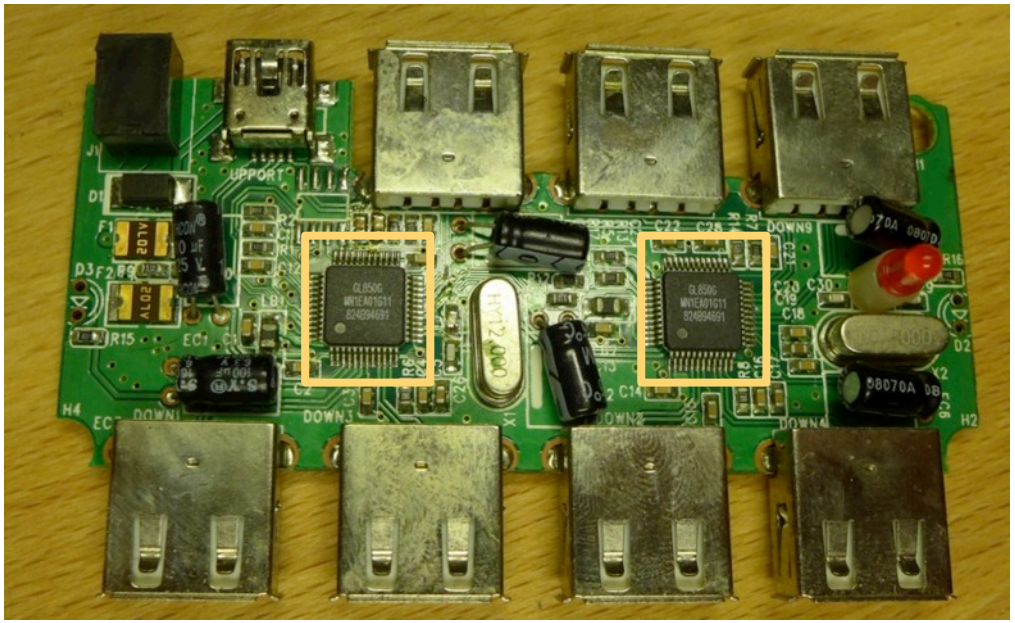


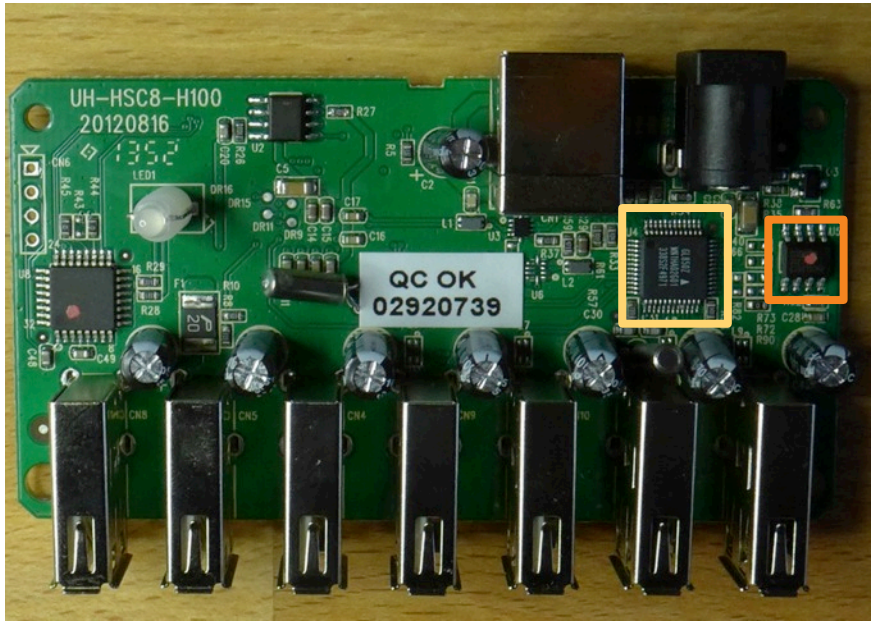
Small hardware design differences can determine BadUSB-vulnerability

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
Trend 1 – **Newer and more complex devices are more vulnerable**	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
Trend 2 – **Chips become more versatile, and thereby more vulnerable**	▪ 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
Trend 3 – **Most controllers that can be programmed are vulnerable**	▪ 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

Protection idea	Limitation
Whitelist USB devices	▪ USB devices do not always have a unique serial number ▪ OS's don't (yet) have whitelist mechanisms
Block critical device classes, block USB completely	▪ Obvious usability impact ▪ Very basic device classes can be used for abuse; not much is left of USB when these are blocked
Scan peripheral firmware for malware	▪ 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
Use code signing for firmware updates	▪ Implementation errors may still allow installing unauthorized firmware upgrades ▪ Secure cryptography is hard to implement on small microcontrollers ▪ Billions of existing devices stay vulnerable
Disable firmware updates in hardware	▪ **Simple and effective** (but mostly limited to new devices)

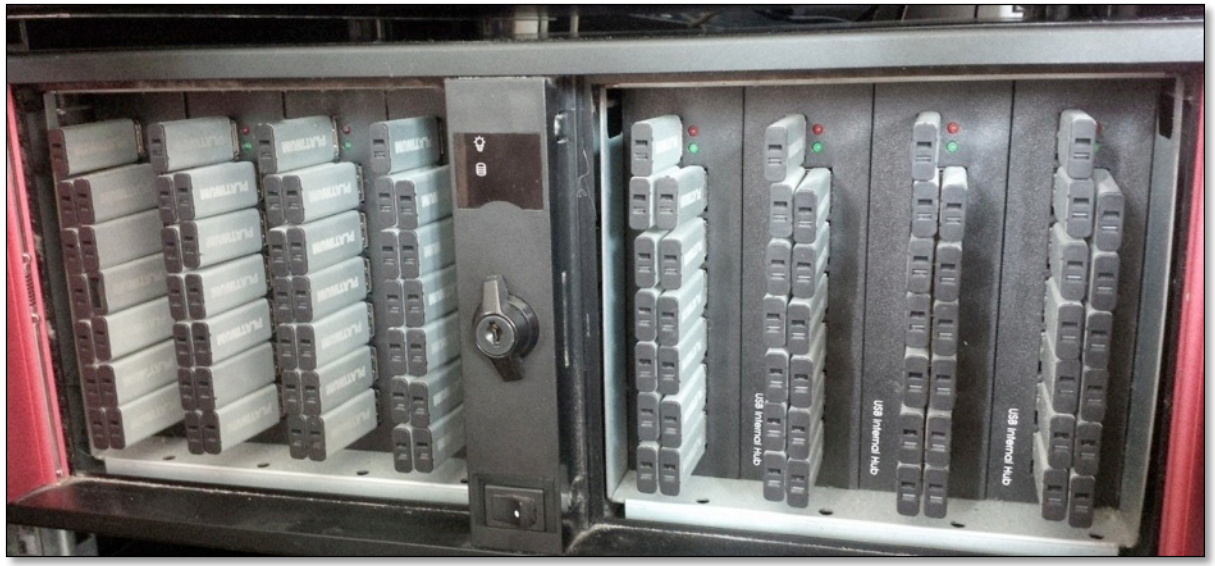
Responsibility for BadUSB mitigation is unclear

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

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

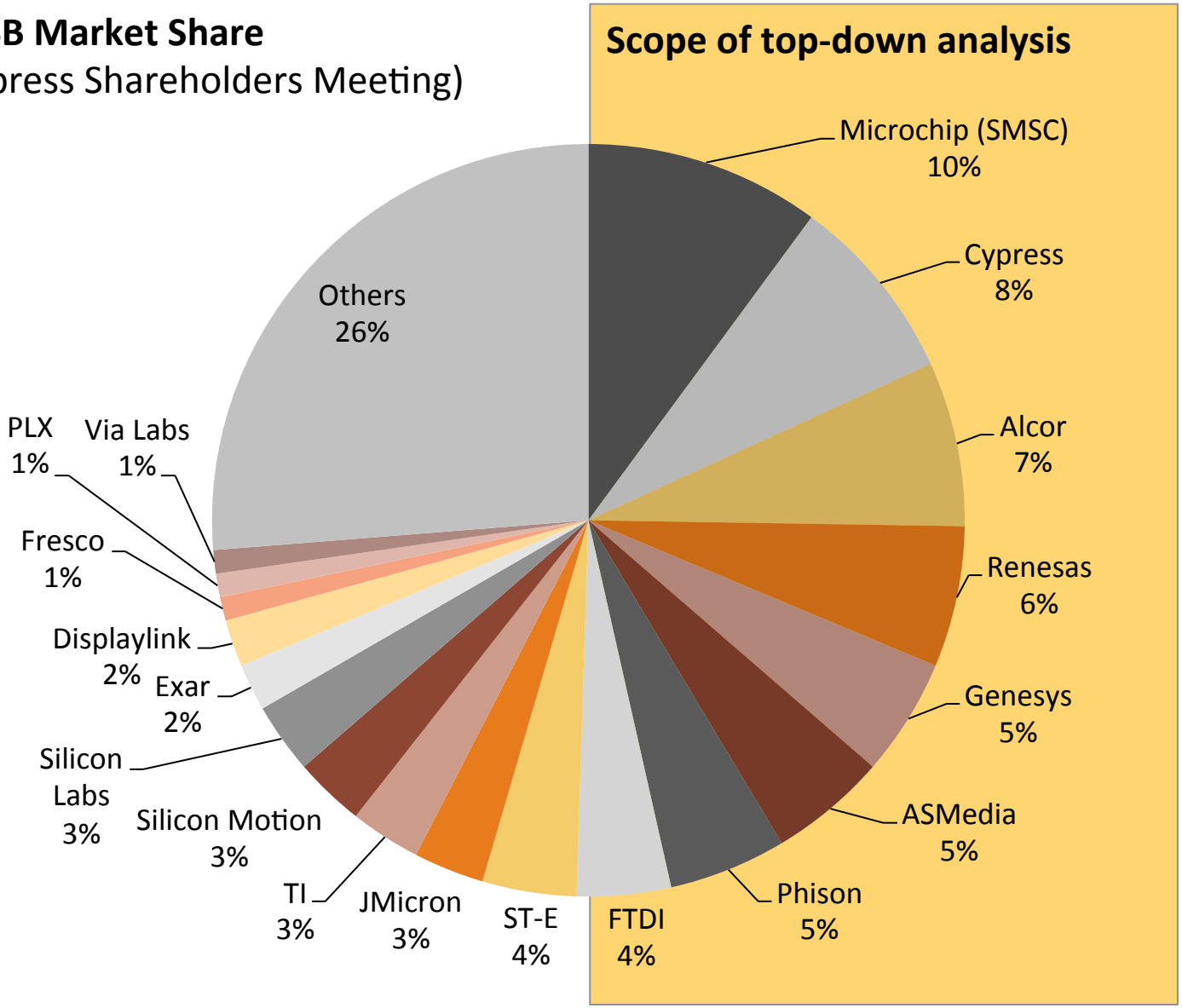
- **USB** peripherals provide for a versatile **infection path**
- Once infected – through USB or otherwise – malware can use peripherals as a **hiding place**, hindering system clean-up
- As long as USB controllers are re-programmable, USB peripherals should **not be shared** with others

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