# Marshall Plan Scholarship
# Final Report:
# Security of the Universal Serial Bus

Sebastian Neuner
sneuner@sba-research.org
Friedhofstrasse 50/5
2103 Langenzersdorf

Boston, on 20th December 2014

# Contents

**1 General Information** **3**

**2 Acknowledgements** **4**

**3 Introduction** **5**
3.1 Report Purpose . . . . . 5
3.2 Northeastern University Boston . . . . . 5
3.2.1 Possibilities . . . . . 5

**4 USB Project** **6**
4.1 Universal Serial Bus . . . . . 6
4.2 General USB Fundamentals . . . . . 6
4.2.1 USB Internal Structure . . . . . 8
4.2.2 USB Protocol . . . . . 9
4.2.3 Mechanical and Electrical Fundamentals . . . . . 13
4.2.4 Wireshark Fundamentals . . . . . 13
4.2.5 USB attacks . . . . . 15
4.2.6 Current Defense Mechanisms . . . . . 17
4.2.7 Researched Defense Mechanism . . . . . 18

**5 Events** **19**
5.1 Academic Meetings . . . . . 19
5.2 Capture-the-Flag-Tournaments . . . . . 20

# 1 General Information

**Scholarship recipient**
Sebastian Neuner
Friedhofstrasse 50/5
2103 Langenzersdorf
http://www.sba-research.org/team/researchers/sebastian-neuner/

Duration of stay: 5 months
Field of study: Computer Science
Research Topic: Security of the Universal Serial Bus

**Home Institution**
Vienna University of Technology
Karlsplatz 13
1040 Vienna
Thesis Advisor: Dr. Edgar Weippl
https://www.sba-research.org/team/management/edgar-weippl/
http://www.ifs.tuwien.ac.at/user/31

**Host Institution**
Northeastern University Boston
College of Computer and Information Science
440 Huntington Avenue
202 West Village H
Boston, Massachusetts 02115
Advisor: Prof. Engin Kirda
http://www.ccs.neu.edu/home/ek/

# 2 Acknowledgements

At this point I want to thank the Marshall Plan Foundation for their generous support as well as the International Office of the Vienna University of Technology that provides information and help for outgoing students. I furthermore want to thank SBA Research and especially my diploma thesis advisor Edgar Weippl. Finally I want to thank the lab in Boston: Especially Engin Kirda who invited me to work with him, William Robertson and Collin Mulliner.

# 3 Introduction

This section describes the purpose of the stay in Boston and introduces into the carried out research project at Northeastern University.

## 3.1 Report Purpose

This report should summarize the stipend recipients experiences at the host institution Northeastern University Boston (NEU). It is important to mention, that most of the current research results achieved in the lab are confidential. These results are written down in an academic paper and are work in progress. To reduce the chance of being "scooped", this report has only rough details and includes only an overview of the work, but gathered fundamentals.

## 3.2 Northeastern University Boston

The reason why the scholarship recipient has chosen NEU as the host institution are fairly simple:
The research lab led by Prof. Engin Kirda is specialized on the focus of the recipients Masters' thesis, his upcoming Phd studies as well as a lot of his interests. This specific area is information security in operating systems with regards to attack and defense mechanisms of those systems. Furthermore the team led by Prof. Kirda at the college of computer and information science (CCIS) consists of experts in various other field that open the possibilities to enhance knowledge in a lot of aspects among others: Android security, web security and social media security.

### 3.2.1 Possibilities

Besides the possibilities in the cyber security lab, other research labs at CCIS are carrying out interesting research in topics like: Computer theory (e.g. complexity theory), robotics, algorithmic research, etc. Attending seminars in special topics is possible if clarified with the corresponding professor in advance.

# 4 USB Project

This section provides insights into the research carried out at NEU. To understand why this research is important and emerging the threat model is explained in Section 3.

## 4.1 Universal Serial Bus

The Universal Serial Bus (USB) has become more and more popular in the past decade. In 2009 there were 6 billion devices used with a raising shipment of 2 billion devices per year more [21]. Since machines get thinner and more portable they lack external interfaces like USB. On the opposite people want to connect more and more external devices like USB drives, keyboards as well as other necessary or unnecessary gadgets [23]. Attackers followed this trend and developed sophisticated attacks by using USB devices. This will be described in Section 4.2.5.

## 4.2 General USB Fundamentals

USB is as the name describes a serial bus. This kind of communication sends each bit after another. On the contrary parallel buses send data at the same time over different channels on the same hardware. Due to several reasons, like synchronisation of bits sent at the same time serial buses almost replaced parallel buses.
USB itself was developed in the 1990s as a tiered star topology and is illustrated in Figure 1.

This topology specifies one master controller including 127 connected and managed USB devices at the same time. The limitation of 127 is caused by the limited 8 bit addressing in each USB packet.
Furthermore USB is standardized in three standards: USB 1.x, USB 2.0 and USB 3.0. USB 1.1 provides a speed up to 12 Mbit/s, USB 2.0 provides speed up to 480 Mbit/s and USB 3.0 provides up to 5000 Mbit/s.

The USB controller also known as root hubs manage the devices and are specified. USB 1.1 has two of those specifications: On the one hand the Universal Host Controller Interface (UHCI) [13] that was developed by Intel and on the other hand the Open Host Controller Interface (OHCI) [4] developed by Compaq, Microsoft and National Semiconductor. USB 2.0 is single specified in the Enhanced Host Controller Interface (EHCI) [12] by different companies such as Intel and NEC. With the appearance of USB

[1] http://www.usblyzer.com/img/articles/physical-usb-bus-topology-host-view.png

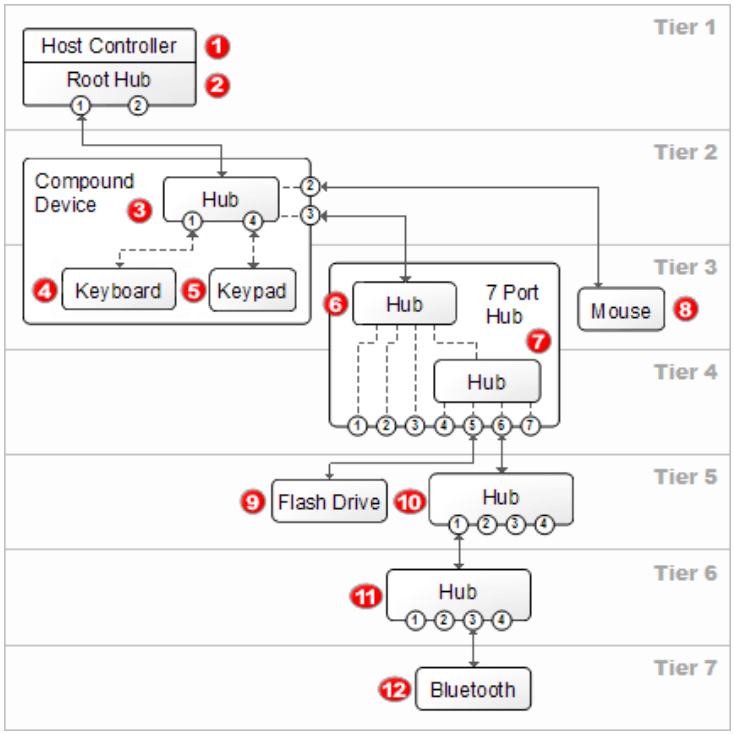


Figure 1: Tiered star topology of USB[1]

3.0 the specification eXtensible Host Controller Interface (xHCI) [14] was introduced to manage every USB standard from 1.0 to 3.0 and all future standards.

USB devices provide the functionality of being hotplugged into the operating system (plug and play). This means a user plugs a USB device into a USB port on a computer running a modern operating system which automatically detects the device and loads the appropriate driver for it. The decision which driver has to be loaded is determined by the USB bus the device is plugged into. This bus detects the product ID and vendor ID every USB device has integrated. After detection of those numbers the bus looks up the appropriate driver in a hardcoded list. As soon as the driver is bound to the USB device the whole root hub is scanned again for devices that are not already bound to see whether the currently loaded driver is applicable to one of them [28].

Hotplugging makes USB very convenient to use: As soon as the device is plugged into a port, it will be ready to use in a short time (if the appropriate driver is found on the operating system). This convenience on the other hand provides an attacker a large surface to attack the user through USB. Recent

attacks related to this specific topic are addressed in more depth in Section 4.2.5.

### 4.2.1 USB Internal Structure

Regarding the USB 2.0 standard each device provides among others: pipes, endpoints and interfaces as illustrated in Figure 2 [5].

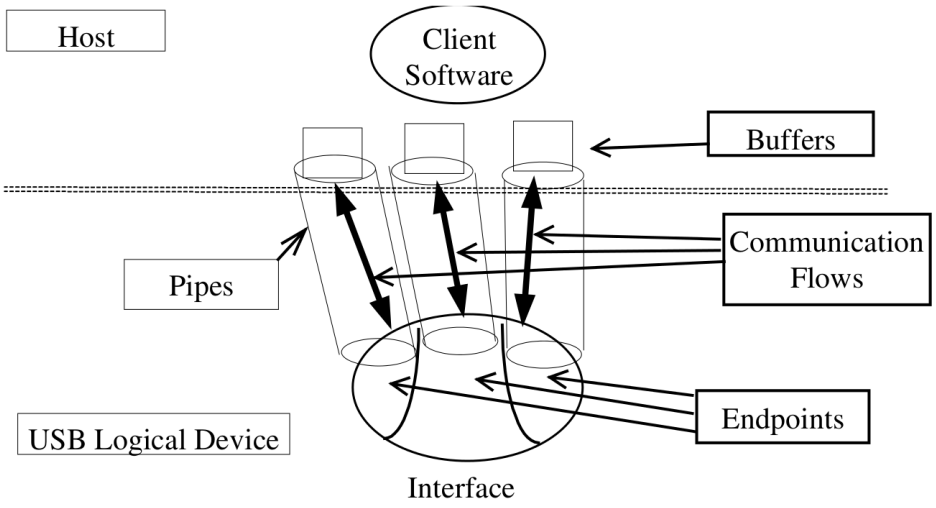


Figure 2: Internal structure of USB[2]

Regarding Craig Peacock [19] endpoints are sources and sinks of data in USB. Every of those endpoints has a number and a direction described as IN and OUT. Data streams in IN direction flows from the device to the host and data streams with OUT direction flow from the host to the device. The connection of the host with the devices over endpoints is called pipe. Pipes carry information in their parameters like direction of the data flow within the endpoints and the transfer types as described in Section 4.2.2. Furthermore endpoints are logically grouped together as USB interfaces. As mentioned above a driver is loaded for every USB device. To be more precise a driver is bound automatically to every interface the USB device provides to the operating system. As an example a webcam that provides two interfaces: One interface that needs a webcam driver to provide the camera functionality and one interface that needs a microphone driver for voice recognition.

[2]Taken from Figure 5-10 of the USB 2.0 standard [5]

| Group | PID Value | Packet Identifier |
|---|---|---|
| Token | 0001 | OUT Token |
| | 1001 | IN Token |
| | 0101 | SOF Token |
| | 1101 | SETUP Token |
| Data | 0011 | DATA0 |
| | 1011 | DATA1 |
| | 0111 | DATA2 |
| | 1111 | MDATA |
| Handshake | 0010 | ACK Handshake |
| | 1010 | NAK Handshake |
| | 1110 | STALL Handshake |
| | 0110 | NYET (No Response Yet) |
| Special | 1100 | PREample |
| | 1100 | ERR |
| | 1000 | Split |
| | 0100 | Ping |

Table 1: PID Values[3]

### 4.2.2 USB Protocol

The USB protocol is initiated by the host towards the device. The first packet sent by the host is a Token packet. This packet includes important information such as the devices' address and the endpoint on that specific device. The packet that follows is an optional Data packet. This packet carries the payload of the USB message. Finally the Status packet signalizes the successful receiving of the data or if the endpoint supposed to accept data denies the data or is not able to receive data [20].

**Packet Fields** Typically USB packets consist of the fields Synchronisation (Sync), Packet ID (PID), Address (ADDR), Endpoint (ENDP), Checksum (CRC) and End of Packet (EOP). Sync packets synchronise the clock of the receiver and the transmitter. PID packets identify the type of packet. Possible PID values are outlined in Table 1.

The Address field clearly specifies the target device the packet is supposed to be sent to. The Endpoint field specifies the endpoint on the target device that is supposed to handle the sent data. The CRC field holds a checksum for the payload. The EOP packet signalizes the end of the current packet.

[3]Taken from the USB 2.0 specification [20]

| Sync | PID | ADDR | ENDP | CRC5 | EOP |
|---|---|---|---|---|---|

Table 2: Token packet format

| Sync | PID | DATA | CRC16 | EOP |
|---|---|---|---|---|

Table 3: Data packet format

**Packet Types** The four different packet types of USB are described below [20]:

- **Token Packets** describe the transaction type that follows after the packet. These packets are divided into three types:
  - **IN** packets notify the USB device that the host is about to request information.
  - **OUT** packets notify the USB device that the host is about to send information.
  - **Setup** packets begin control transfers.

  The format of Token packets is illustrated in Table 2.
- **Data Packtes** contain the actual payload and are divided into Data0, Data1 and for high speed transmissions another two PIDs, a DATA2 field and MDATA. The format for Data packets is shown in Table 3.
- **Handshake Packets** acknowledge received data to the sender or report errors to the sender. Furthermore Handshake packets are divided into three types, basically only consisting the PID as shown in Table 4 surrounded by Sync and EOP.
- **Start of Frame (SOP) Packets** indicate the start of a new frame and consist of a 11 bit frame number in addition to the Sync, PID, CRC and EOP fields as shown in Table 5.

| Sync | PID | EOP |
|---|---|---|

Table 4: Handshake packet format

| Sync | PID | Frame Number | CRC5 | EOP |
|---|---|---|---|---|

Table 5: Start of Frame packet format

**Transfer Types** As explained above, endpoints accept or send data. Therefore endpoints have a certain type known as transfer type [18] handled by the built-in firmware. These types specify different modes of operation as described in this section.

**Control Transfer**:
This type is important for setting up the initial communication between the device and the host computer. E.g. the enumeration of interfaces on the device is done with Control Transfer packets. A typical control transfer has three stages: In the Setup Stage Setup packets are used to set up the initial state with the USB device. The Data Stage is not mandatory and specifies the endpoint direction IN and OUT and the data that is sent. And finally the Status Stage which is used by the specified function for status reporting.

**Interrupt Transfer**:
Interrupts are typically generated by the USB device. USB interrupts are requested by the host that is furthermore responsible to poll for new events generated by the device. An interrupt transfer has a guaranteed latency and is unidirectional. A well-known application for interrupts is a USB keyboard or a USB mouse. Every time a key is pressed or released by the user an interrupt is sent from the USB device to the host for interpretation by the driver. A sample interrupt transfer captured from a USB keyboard is described in Section 4.2.4.

**Isochronous Transfer**:
These transfer types are used for continuous data deliveries. This includes for example audio or video streaming over USB. Isochronous transfer types include among others the following important features: Full USB bandwidth, error detection and predefined latency.

**Bulk Transfer**:
Bulk transfer is used for large data transfer e.g. between an USB mass storage device and a host system. An important part of Bulk transfers is the CRC error detection on the payload to prevent the acceptance and further processing of altered data and the guaranteed delivery to the recipient over USB.

**Device States** To conclude the Section USB Protocols the last mentioned USB characteristics are USB device states [5] and are illustrated in Figure 3 as well as Table 6. These states specify the whole process for USB devices on a higher level from being attached to a physical USB port until fully configured or suspended.

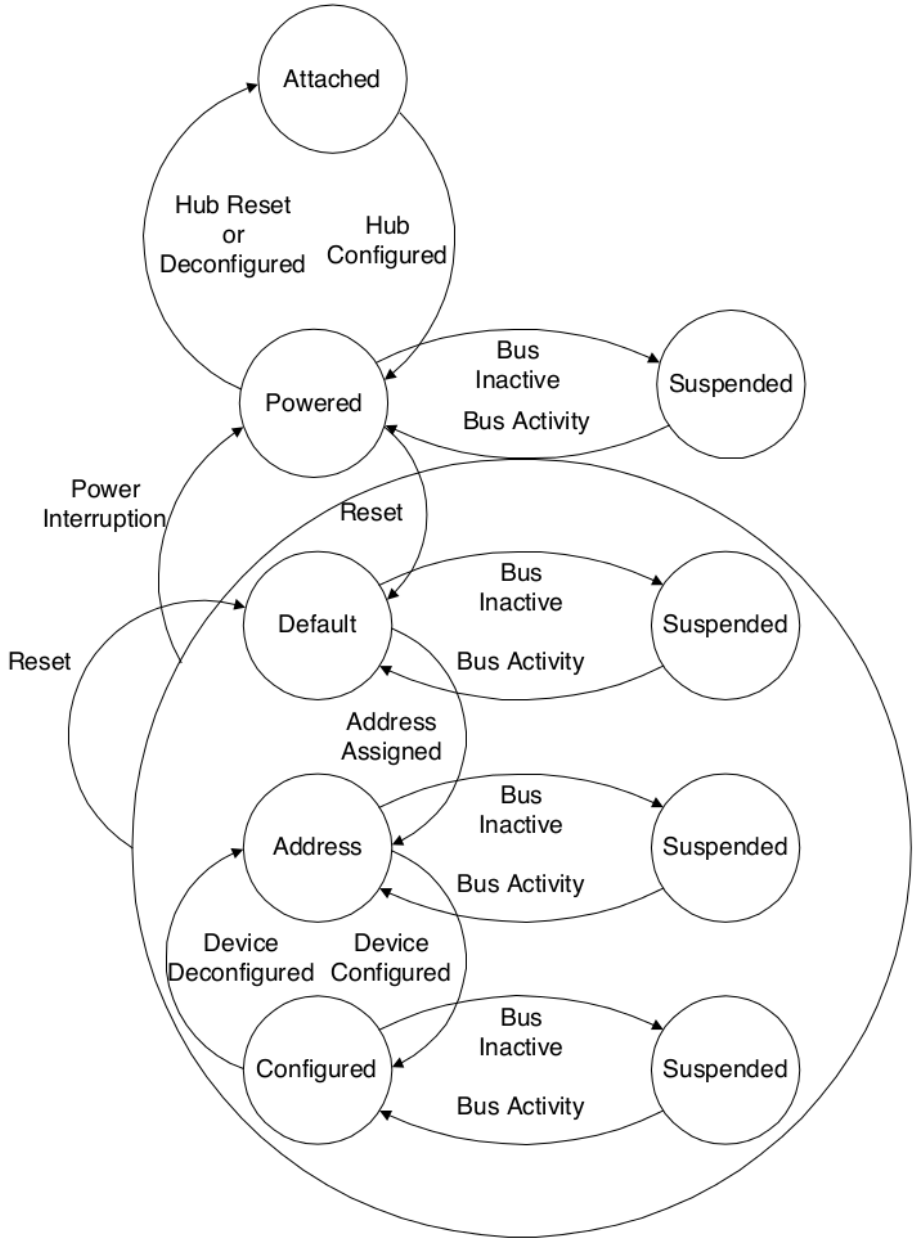


Figure 3: USB device states[4]

### 4.2.3 Mechanical and Electrical Fundamentals

There are two standard types of connectors for USB: Type A and type B [17]. These two types are shown in Figure 4.

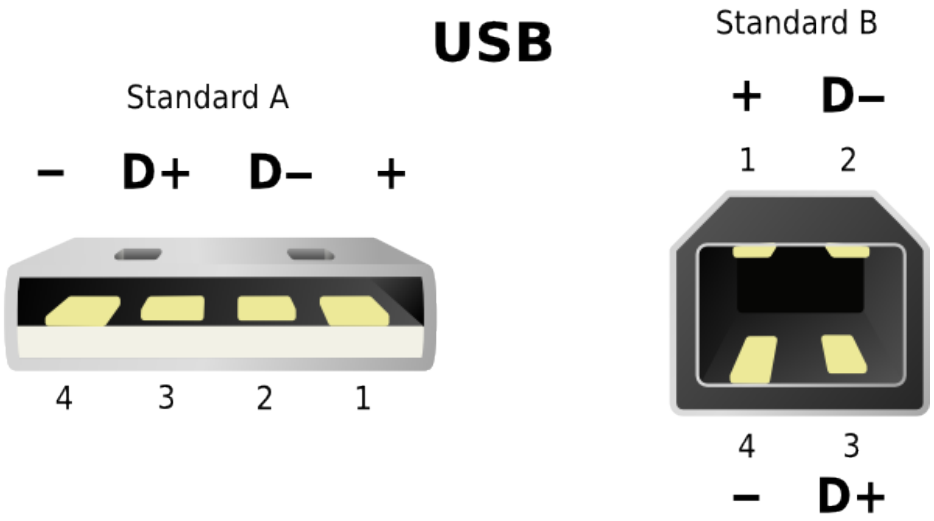


Figure 4: USB standard connector types[6]

Furthermore there exist several other subtypes of connectors like the micro versions of type A and type B or the corresponding mini-types of A and B. However these connectors do not only differ by size but also by pin layout. These different pin layouts are illustrated in Table 7 for standard USB connectors and Table 8 for micro USB connectors.

### 4.2.4 Wireshark Fundamentals

Wireshark is a packet capturing and analyzing tool for different operating systems [8]. The Wireshark capture engine is called Dumpcap [7] relying on different libraries. This library is called WinPcap [26] on Windows and libpcap [16] on Linux flavored operating systems. At least for Microsoft Windows and Linux it is possible to not just capture traffic over networks such as LAN, but also traffic that is sent through USB ports. Linux is capable of capturing USB packets with the usbmon interface [29] that is integrated

[4]Taken from Figure 9-1 of the USB 2.0 standard [5]
[5]Taken from Table 9-1 of the USB 2.0 standard [5]
[6]https://upload.wikimedia.org/wikipedia/commons/6/67/USB.svg
[7]Taken from Beyond Logic USB in a Nutshell [17]
[8]Taken from Micro-USB connector pinout [22]

| Attached | Powered | Default | Address | Configured | Suspended | State |
|---|---|---|---|---|---|---|
| No | — | — | — | — | — | Device is not attached to the USB. Other attributes are not significant. |
| Yes | No | — | — | — | — | Device is attached to the USB, but is not powered. Other attributes are not significant. |
| Yes | Yes | No | — | — | — | Device is attached to the USB and powered, but has not been reset. |
| Yes | Yes | Yes | No | — | — | Device is attached to the USB and powered and has been reset, but has not been assigned a unique address. Device responds at the default address. |
| Yes | Yes | Yes | Yes | No | — | Device is attached to the USB, powered, has been reset, and a unique device address has been assigned. Device is not configured. |
| Yes | Yes | Yes | Yes | Yes | No | Device is attached to the USB, powered, has been reset, has a unique address, is configured, and is not suspended. The host may now use the function provided by the device. |
| Yes | Yes | — | — | — | Yes | Device is, at minimum, attached to the USB and is powered and has not seen bus activity for 3 ms. It may also have a unique address and be configured for use. However, because the device is suspended, the host may not use the device's function. |

Table 6: USB device states[5]

| PIN Number | Cable Color | Function |
|---|---|---|
| 1 | Red | $V_{BUS}$ (5 Volts) |
| 2 | White | Data- |
| 3 | Green | Data+ |
| 4 | Black | Ground |

Table 7: Standard USB pin layout[7]

| PIN Number | Cable Color | Function |
|---|---|---|
| 1 | Red | VCC (5 Volts) |
| 2 | White | Data- |
| 3 | Green | Data+ |
| 4 | — | Mode Detect |
| 5 | Black | Ground |

Table 8: Micro USB pin layout[8]

in kernels after 2.6.11 as a loadable module. Wireshark is able to make use of the usbmon interface. It displays USB packets and makes them dissectible by using Wireshark filters. Figure 5 shows a packet dump of a HID device (USB keyboard) captured with Wireshark and usbmon. The capture shows interrupt packets requested by the host as explained in Section 4.2.2. As an example packet 311 is the answer in form of an interrupt to the request in packet 310. It carries data that is labeled by Wireshark as "Leftover Capture Data". In this specific case of an USB keyboard capture it carries the data of key interactions e.g. which key on the keyboard is pressed or released.

### 4.2.5 USB attacks

Due to badUSB USB attacks are more emerging than ever. No rogue USB device can be inserted without a possible infection of malware or the injection of malicious commands. Former well-known attacks were carried out by malware that was hidden on USB drives waiting for their execution on connection with a operating system. Modern attacks such as reflashing attacks and the badUSB attack are described in this section.

**Reflashing Attacks** Running on every USB devices hardware is a software that provides the USB devives main functionality to the user. This software is flashed directly onto the chip of the USB device. As an example a software on

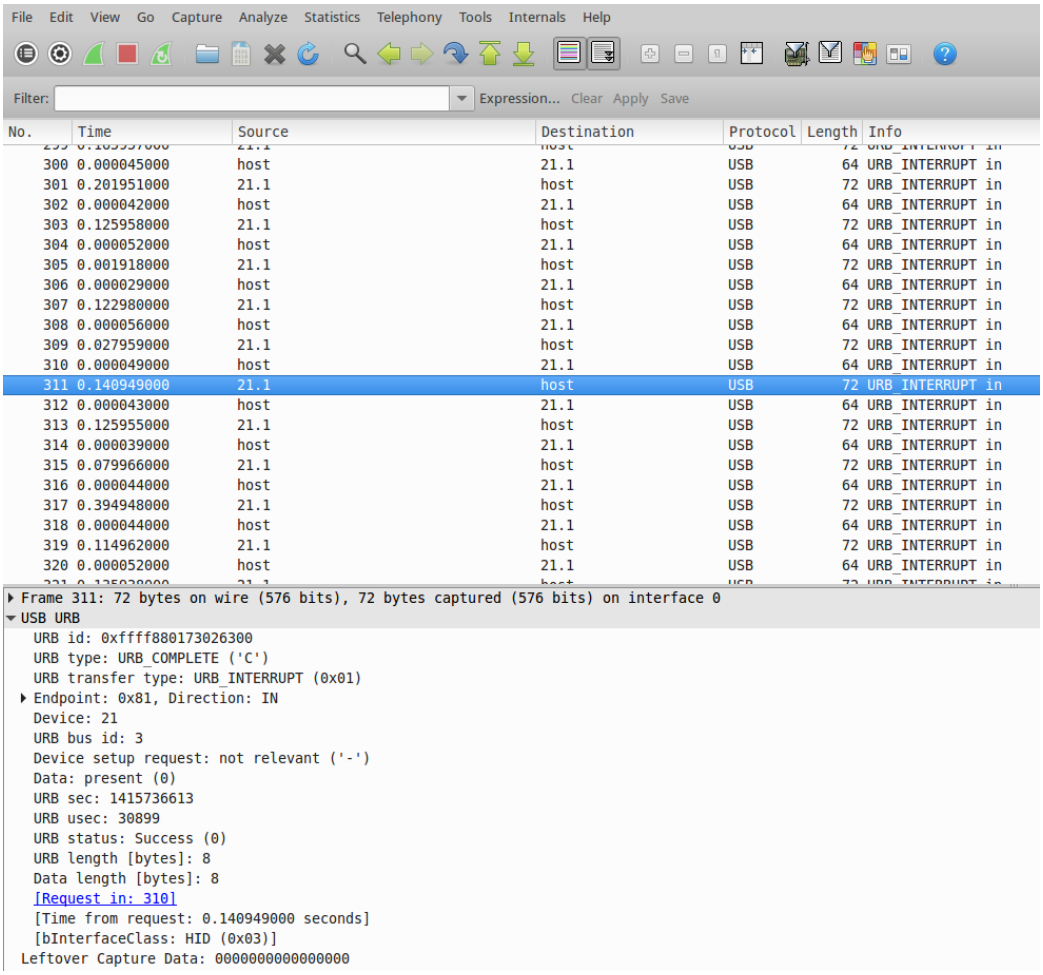


Figure 5: Wireshark packet capture of USB keyboard

the chip that handles SCSI commands for the USB flash drive that provides mass storage functionality to the user. In case of a reflashing attack the attacker has two options: Delete the original software and write the malicious software onto the chip or write the malicious software as addition to the original one onto the chip. The second choice is much stealthier e.g. on a USB thumb drive that acts as a mass storage but furthermore acts as a HID. In practice an attack could look like the following example: The user finds a USB drive on a parking lot of a company and plugs it into the companies network. At first glance the mass storage is empty. In the background the HID driver in the operating system is additionally loaded and bound to the drive. While the drive is connected to the computer it emulates keystrokes to compromise not only the local computer but the whole network.

Although this seems like a great attack vector there exist some restrictions. One of those restrictions is the need of a specific USB chip with a specific pre-flashed firmware version like the Phison 2251-03 [11].

**badUSB Attack** The badUSB USB attack is a specific "media-hyped" [9] [27] [10] version of the USB reflashing attack [25]. In addition to its actual attack payload the attack varies within the following scenarios [6]:

- **Windows takeover:** A USB drive that emulates keystrokes to inject commands into the Windows operating system as soon as the driver is bound to the device as described in Section 4.2.5. Those commands lead to e.g. open ports that act as backdoors potential attackers.
- **Windows infects USB device:** A USB device is plugged into a maliciously modified computer that runs Windows. That system rewrites the software on chip (reflashing). As soon as the device is plugged into a machine that runs Linux the sudo password will be sniffed on input. This sniffing is done while the victim user authenticates its session e.g. during unlock of a locked screen.
- **Attack via Android Phone:** This attack makes use of the users trust into ordinary Android devices. The attacker pretends to charge his phone while the phone actually changes the DNS settings of the computer. If this attack is successful this results in e.g. queries going to "mybankaccount.com" that is actually resolved to 192.168.0.1 (benign server) is then resolved to 172.16.0.1 (server controlled by attacker).
- **USB Boot Sector Virus:** If keystrokes are emulated at boot time the USB drive is able to install a malicious BIOS out of a hidden part of the USB drives mass storage. Every boot of the operating system would be carried out by a malicious BIOS.

### 4.2.6 Current Defense Mechanisms

Several defense mechanisms are provided by different authors. The first approach discussed here is the block udev approach: By creating a certain udev rule on Linux operating systems all HID USB devices like keyboards will be blocked when they are plugged in [2]. Udev [15] messages in Linux are sent by the kernel to inform the userland part of the operating system of new events on the PCI (USB) bus. Udev is furthermore used to dynamically manage files/devices in the "/dev" folder structure.

Another approach that tackles the threat caused by malicious USB devices is the GData USB Keyboard Guard [3]. When a device is plugged into a Windows machine for the first time a notification window appears asking the user whether he wants to allow the keyboard or block it. In case the user blocks the device it is added to a blacklist, in case the user allows the device it is added to a whitelist of benign devices. Saved in this whitelist/blacklist is the product-ID and vendor-ID value pair, that should identify the keyboard uniquely. A German online magazine for IT and IT security related topics explained why this system is easy to circumvent and no real protection against those attacks [24]. An USB device could e.g. emulate a whitelistet ID to circumvent the filter.

### 4.2.7 Researched Defense Mechanism

During the research stay at the Northeastern University in Boston a novel approach regarding defense of USB reflashing attacks was researched. This approach for Linux based operating systems is more sophisticated than the defense mechanisms discussed in Section 4.2.6. The approach designed and implemented by the scholarship recipient acts as a secure layer between the USB device and the user. Since neither the idea nor the implementation is published yet but in the process of being published, this report will not contain any more details regarding the project.

# 5 Events

This section describes events happened during the stay of the scholarship recipient at the Northeastern University Boston.

## 5.1 Academic Meetings

Due to the large amount of meetings the communication within the research lab is great. Furthermore meetings that are not held by the lab itself but by the university are good opportunities to get in touch with interesting people from other universities and hear talks about interesting topics.

**Weekly Research Jour Fixes** Those weekly meetings include the whole security lab (professors and students). The main topic is the progress made in the past week but also upcoming tasks and conference deadlines. Furthermore these meetings keep the students together and reduce the chance of forgetting important dates. If any of the students is going to present a paper or a poster at an upcoming conference this meeting is used to practice the talk and discuss possible questions of the audience at the actual conference. This meetings lasts roughly an hour.

**Research Meetings** This irregular meeting happens every two or three weeks depending of the students' needs. Within the meeting the current project progress is discussed mainly with the advisor and the project members. Since problems are discussed until no further input is brought up by neither the advisor nor the students this meeting lasts between 30 minutes and two hours.

**PhD Proposals** To show PhD students what the studies include in higher semesters it is possible to attend PhD proposals. In this event, typically happening in the last part of the PhD studies the student presents the actual content of his/her work to the audience including professors which evaluate this work. If the progress in the studies and the predictable output is sufficient the student is going to finish his studies typically within one year. The scholarship recipient attended the PhD proposal of Travis Mayberry "Oblivious RAM".

**Hiring Talks** Whenever a faculty position is offered a hiring talk is held by the applicant. It is possible to visit these hiring talks by the PhD students and even encouraged to attend. The applicant presents the work he has done

so far and is planning to do to the audience. The attending students are not involved in the final hiring decision. The scholar attended the hiring talk of Willard Rafnsson.

**Distinguished Lecture Series** For these talks interesting speakers from all over the world are invited. These speakers talk about their past projects or interesting projects in the future respectively. During the time of the scholar one distinguished lecture was held by Milind Tambe talking about science of security games.

**Invited Talks** In contrary to the distinguished lectures these invited talks are held for students of the secure systems lab only. During the stay of the scholar in Boston Christian Platzer from TU Wien held a talk about his current publication on breaking integrated circuit device security.

**PhD Social Hour** The last event mentioned in this section is the weekly PhD social hour. This is a get together of all PhD students of the department of computer science at NEU. While enjoying coffee and cookies the students are encouraged to discuss their projects and get valuable input from other research labs. Although the name would suggest one hour duration the event lasts until the coffee is gone.

## 5.2 Capture-the-Flag-Tournaments

Besides all the academic events and meetings so-called capture-the-flag-tournaments (CTF) [1] were participated by the scholar within the joint team of the university of Santa Barbara and the Northeastern University Boston: Team Shellphish. Those hacking events which typically last 8 hours to two days heavily train the offensive security skills of the participants. The scholarship recipient participated in his spare time in the following tournaments:

**CSAW Qualifier:** 19.09.2014 until 21.09.2014

**hack.lu CTF:** 21.10.2014 until 23.10.2014

**SECCON CTF Qualifier:** 06.12.2014 until 07.12.2014

**RuCTFe:** 20.12.2014

# References

[1] CTF? WTF? https://ctftime.org/ctf-wtf/.

[2] How to prevent BadUSB attacks on linux desktop. https://security.stackexchange.com/questions/64524/how-to-prevent-badusb-attacks-on-linux-desktop.

[3] GData Software AG. How to be SICHER from USB attacks. https://www.gdata.at/at-usb-keyboard-guard.

[4] National Semiconductor Compaq, Microsoft. OpenHCI Open Host Controller Interface Specification for USB. ftp://ftp.compaq.com/pub/supportinformation/papers/hcir1_0a.pdf, September 1999.

[5] Compaq et al. Universal Serial Bus Specification Revision 2.0. http://www.usb.org/developers/docs/usb20_docs/usb_20_112614.zip, April 2000.

[6] Karsten Nohl et al. This thumbdrive hacks computers. https://srlabs.de/blog/wp-content/uploads/2014/11/SRLabs-BadUSB-Pacsec-v2.pdf, November 2014.

[7] Ulf Lamping et al. How Wireshark Works Overview. https://www.wireshark.org/docs/wsdg_html_chunked/ChWorksOverview.html.

[8] Wireshark Foundation. Wireshark. https://www.wireshark.org/.

[9] Dan Goodin. This thumbdrive hacks computers. http://arstechnica.com/security/2014/07/this-thumbdrive-hacks-computers-badusb-exploit-makes-devices-turn-evil/, July 2014.

[10] Andy Greenberg. Why the Security of USB Is Fundamentally Broken. http://www.wired.com/2014/07/usb-security/, July 2014.

[11] Ionut Ilascu. BadUSB Code is Out. http://news.softpedia.com/news/BadUSB-Code-Is-Out-USB-Makers-Need-to-Improve-Security-460979.shtml, October 2014.

[12] Intel. Enhanced Host Controller Interface Specification. http://www.intel.com/content/www/us/en/io/universal-serial-bus/ehci-specification.html.

[13] Intel. Universal Host Controller Interface (UHCI) Design Guide. ftp://ftp.netbsd.org/pub/NetBSD/misc/blymn/uhci11d.pdf, March 1996.

[14] Intel. eXtensible Host Controller Interface for Universal Serial Bus (xHCI). http://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/extensible-host-controler-interface-usb-xhci.pdf, May 2009.

[15] Greg Kroah-Hartman. udev–a userspace implementation of devfs. In *Proc. Linux Symposium*, pages 263–271. Citeseer, 2003.

[16] Luis MartinGarcia. tcpdump and libpcap. http://www.tcpdump.org/, 2014.

[17] Craig Peacock. Connectors. http://www.beyondlogic.org/usbnutshell/usb2.shtml.

[18] Craig Peacock. Endpoint Types. http://www.beyondlogic.org/usbnutshell/usb4.shtml.

[19] Craig Peacock. USB in a NutShell. http://www.beyondlogic.org/usbnutshell/usb1.shtml.

[20] Craig Peacock. USB Protocols. http://www.beyondlogic.org/usbnutshell/usb3.shtml.

[21] Melissa J. Perenson. SuperSpeed USB 3.0: More Details Emerge. http://www.pcworld.com/article/156494/superspeed_usb.html, January 2009.

[22] Pinouts.ru. Micro-USB connector pinout. http://pinouts.ru/PortableDevices/micro_usb_pinout.shtml, May 2014.

[23] Sharon Profis. 10 awesome ways to use a usb flash drive. http://www.cnet.com/how-to/10-awesome-ways-to-use-a-usb-flash-drive/, 2012.

[24] Juergen Schmidt. Kostenloses G-Data-Tool schuetzt vor BadUSB-Angriffen. http://www.heise.de/newsticker/meldung/Kostenloses-G-Data-Tool-schuetzt-vor-BadUSB-Angriffen-2329545.html, September 2014.

[25] SrLabs. Turning USB peripherals into BadUSB. https://srlabs.de/badusb/.

[26] Riverbed Technology. WinPcap. http://www.winpcap.org/, 2013.

[27] Steven Vaughan-Nichols. BadUSB: Big, bad USB security problems ahead. http://www.zdnet.com/article/badusb-big-bad-usb-security-problems-ahead/, July 2014.

[28] Ian Wienand. What actually happens when you plug in a USB device? https://www.technovelty.org/linux/what-actually-happens-when-you-plug-in-a-usb-device.html, July 2007.

[29] Pete Zaitcev. The usbmon: Usb monitoring framework. In *Linux Symposium*, page 291, 2005.