Computer and Communication Program (SSP)
Faculty of Engineering, Alexandria University

Graduation Project Report

USafeB: USB Sandboxing

Authors:

Abdelrahman Hesham El-Dawy
Ahmad Mahmoud El-Reffaiy
Ahmed Mohamed Ismail
Mahmoud Hassan El-Hendy
Rania Khaled

Supervised by:

Prof. Dr. Mohamed Abougabal
Dr. Ahmed Kosba

Table of Contents

List of Tables

List of Figures

Chapter 1

1.1 Introduction

Peripheral devices, ever since their conception, became an essential part of the computing experience for both consumers and developers alike. Particularly with the emergence of the Universal Standard Bus (USB) standard in 1996 by a group of manufacturers in order to limit the fragmentation of connection methods used with peripheral devices. It was followed closely by IBM selling the first USB flash drive in 2000, where it became a convenient and indispensable tool for moving files from a device to another.

With time, these peripheral devices only became more powerful as technology evolved, from increasing mass storage to eventually allowing manufacturers to create a fully functional computer in the size of a flash drive. These changes, granting more features and offer more value to its users, also significantly increased the security risk.

Since peripherals are more powerful now and have a myriad of options at their possession, it became easier and more dangerous to use them in order to exploit their receiving host.

1.2 Background

1.2.1 CIA triad

Confidentiality, Integrity and Availability is the name of a model known as the CIA-triad that aims to serve as a framework for maintaining information security within an organization. It can also be used to describe what defines a security breach. It is argued that any cybersecurity attack possible breaks at least one CIA principle. These principles, in terms of cybersecurity, are defined as follows:

Confidentiality: Information must be protected from all unauthorized access.
Integrity: Information must be consistent during its lifecycle, protected from unlawful modification as well as compromise during transmission.
Availability: Information must be available for retrieval whenever it is demanded.

These definitions will be used to discuss attack vectors in Section 1.3.

1.2.2 Universal Serial Bus

At the time of writing, there are four generations of USB standards that are backward compatible. A brief overview will be presented for the USB 2.x specification [1] due to its consistent popularity as well as how it serves as a base for future versions.

The USB 2.0 defined a single model for cabling and connectors, where peripherals identify themselves, automatically mapping their functions to driver and configuration. It supports a wide range of packet sizes, allowing a range of device buffering options, with an error-handling mechanism built into its protocol. A USB system is described by three definitional areas, which will serve as the primary notation for the remainder of this document:

- USB devices: These can be hubs, printers, and mass storage devices.
- USB host: The machine in which the USB device is connected to.
- USB interconnect: The manner in which USB devices are connected to and communicate with the host. This includes its bus topology and data flow models.

USB devices are accessed by a USB address assigned to them when they are attached and enumerated. After undergoing this enumeration process, the device sends a set of interface descriptors that denote the protocol of communication that the device expects, as well as the functionality that it provides. The operating system responds by loading the appropriate driver for each interface, configures it appropriately so that the USB device can start its normal operation.

USB attack vectors rely solely on plugging a USB device into the host in order to perform malicious functionality. A recent experiment [2] by CompTIA in 2015 observes consumers' habits when they find USB sticks at random, where about 200 unbranded USB sticks were dropped across high traffic public spaces in business districts, and findings were that 17% of employees plugged the found USB stick into their device. This indicates that the average person is irresponsible with USB devices is not uncommon and it could potentially cause danger in the event that such acts were committed in a critical workplace containing secretive data.

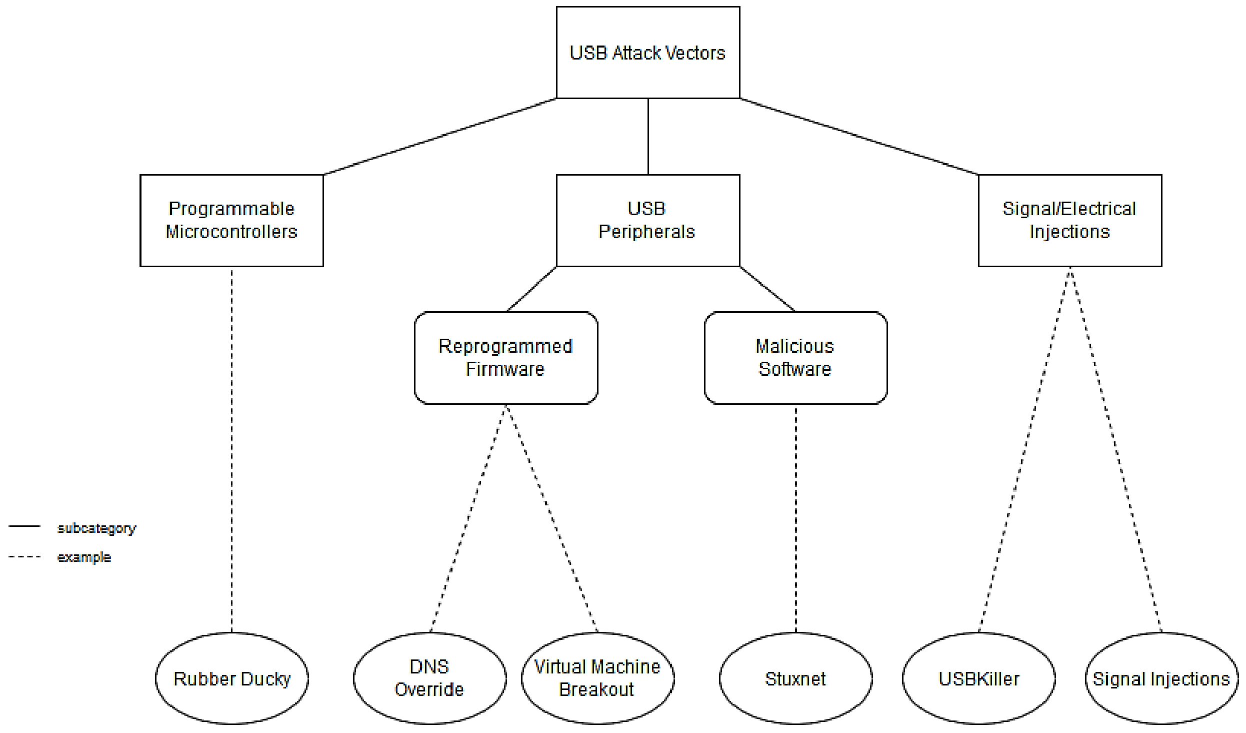
The USB ecosystem was founded on a trust-by-default property, and security was not an initial consideration. A recent survey[3] investigated whether USB Type-C was the answer, as it was the first attempt by USB-IF to address security issues with its dedicated security pipeline, and findings concluded the answer was that while USB Type-C took initial steps towards fixing some of USB's crucial security errors, it was still incomplete and flawed, in addition to being a solution that completely disregards previous USB specifications.

1.2.3 USB Threat Model

A USB device that is to be connected to a host-system might be designed with malicious intent that would lead to the compromising at least one CIA triad principle, as mentioned in Section 1.2.1.

The device can deviate from USB specifications following methods such as protocol spoofing, this is achieved by reprogramming USB device's internal microcontrollers. A USB device can also have a reprogrammed firmware that executes malicious actions such as malware downloading. USB devices that are not reprogrammed on a low level are also included in the threat model, as those devices might have internally malicious software such as viruses or harmful autorun scripts that take advantage of how peripherals interact with host computers and exploit bugs in operating systems.

Electrical shock-based and signal injection-based attacks such as USBKiller [4] are not a focal point, but in this case, it would destroy the middleware and not the host, successfully protecting the host system but losing the device in the process. Figure 1 illustrates this classification of USB attack vectors as well as providing some examples of each kind.



(Figure 1: A classification of USB attack vectors)

1.3 Motivation

Universal Serial Bus (USB) peripherals have undoubtedly become an indispensable part of humanity's lives, as they're widely used by individuals as well as across a plethora of organizations, and its popularity only makes the threat that they pose more severe. With the emergence of BadUSB[5] devices; these commonly used peripherals have become dangerous attack vectors. Currently, there are several security threats and sophisticated attacks posed by USB peripherals, including but not limited to:

- Rubber Ducky: a commercial keystroke injection attack platform released in 2010. Once connected to a host computer, the Rubber Ducky poses as a keyboard and injects a preloaded keystroke sequence.
- Programmable HID USB Keystroke Dongle (PHUKD) attack platforms: similar to Rubber Ducky but allows an attacker to select the time when it injects the malicious keystrokes. It uses Teensy[6] Microcontroller.
- Evilduino: similar to PHUKD but uses Arduino[6] microcontrollers instead of Teensy. Also works by emulating a keyboard/mouse and can send keystrokes/mouse cursor movements to the host according to a preloaded script.
- Default Gateway Override: an attack that uses a microcontroller to spoof a USB Ethernet adapter to override DHCP settings and hijack local traffic.
- Smartphone-based HID attacks: first described in a research paper for which researchers created custom Android gadget drivers to overwrite how Android interacted with USB devices. The malicious driver interacted with the Android USB gadget API to simulate USB keyboard and mouse devices connected to the phone.

- DNS Override by Modified USB Firmware: researchers modified the firmware of a USB flash drive and used it to emulate a USB-ethernet adapter, which then allowed them to hijack local traffic.
- Keyboard Emulation by Modified USB Firmware: several researchers showed how poisoning the firmware of USB flash drives, an attacker could inject keyboard strokes.
- Password Protection Bypass Patch: a small modification of a USB flash drive's firmware allows attackers to bypass password-protected USB flash drives.
- Boot Sector Virus: researchers used a USB flash drive to infect the computer before it boots.
- iSeeYou: Proof of concept program that reprograms the firmware of a class of Apple internal iSight webcams so that an attacker can covertly capture video without the LED indicator warning.

USB attack	**CIA-triad violations**
Rubber Ducky	Confidentiality
PHUKD	Confidentiality
Evilduino	Confidentiality, Integrity, Availability
Default Gateway Override	Confidentiality, Integrity
Smartphone-based HID Attacks	Confidentiality
DNS Override	Confidentiality, Integrity
Password Protection Bypass	Confidentiality, Integrity, Availability

(Table 1: CIA-triad violations of some of the aforementioned threats)

Chapter 2

2.1 Related Work

A survey was conducted for papers addressing the topic of USB security, not all of which were proposed, comprehensive defensive methods, but may include taxonomy reports as well as means to defend against a specific categorization of attack vectors. In addition to those papers, there were numerous projects that were not formally documented as well as additional, generic security research, including one that addresses cyber-security habits of employees in the workplace. Due to the niche nature of the USB defenses topic, research has been sparse. Nonetheless, several unique techniques for defending a computer system against USB threats were taxonomized and evaluated in the following sections.

Methods to protecting USB hosts from malicious USB drives can be categorized into host-based solutions, middleware solutions, hardware modifications to the peripherals themselves as well as hybrid approaches. Firstly, by examining end-point solutions, GoodUSB[8] relies on user identification of the device, and LBM[9] requires kernel-level modifications at the host-system. Hardware modifications such as Kells[10] and ProvUSB[11] might be able to successfully trample the underlying issues of USB peripherals, however, they would necessitate the deployment of new devices.

Out of those publications, there were four attempting a similar approach of isolating the host from USB devices using a middleware. The latest of which was published in 2016, showing that research has shifted from the direction of sandboxes to host-based, platform-centric approaches or towards hardware modifications of the USB devices themselves.

The targeted solution should not interfere with either the host or the USB hardware and is to operate self-sufficiently, without special dependencies from either side. This is to ensure platform-independence, portability as well as maximize ease of deployment on the user's machine. Hence an attempt will be made to revisit the early approaches.

Existing middleware solutions rely on intermediate hardware, such as SandUSB[12] which scans and analyzes USB packets through a Raspberry Pi2 with additional modules installed, and USBWall[13] which uses a BeagleBone Black with USBProxy[21] on board. Another approach is USBCheckin[14], which forces user interaction before allowing a device to interface with the host, an approach that is only functional against a limited range of attacks.

DeviceVeil[7], the most recent relevant publication at the time of writing, requires installing a new integrated circuit into the USB flash drive in order to enable bi-directional authentication. It utilized Trusted Platform Modules (TPM) as well as a customized secure boot to facilitate the bi-directional authentication mechanism, but it places several modifications (albeit of the soft variety) on both the USB hardware as well as the host machine.

Machine-learning approaches are being implemented in newer methods such as USBeSafe[15], which constructs a classification problem from analyzing USB packets in terms of packet interarrival times, payloads, along with other variables. While these methods are finding potential success, they are not employed in this solution as the delicate requirements of host-system protection against malicious peripherals do not tolerate the potentially varying results of a classification problem, as well as how any false positive detection can completely prevent usability of the product for actually harmless USB devices.

Several of the approaches we have examined, such as the use of Context and Provenance[16] in defending against malicious USB drives, cannot be implemented in an isolated middleware as it operates on tracking user activity and associates certain patterns with suspicious behavior. Physical middleware are isolated devices that should not be able to obtain or track user activity of the host device as it is there only to defend the host against external attacks, and as such, those patterns should never be accessed. The same principle extends to approaches that utilize other forms of user data such as the detection of abnormal keyboard activities, although this test can be implemented in the middleware itself. A summary of the surveyed papers and their general approaches is outlined in Table 1.

It must be noted that while the ultimate goal of the approaches mentioned below is to provide reliable defenses against attacks from the USB port, they are not necessarily addressing the same type of attacks or even the same type of problems. Table 2 is a comparison between the key features implemented in some of the surveyed solutions relevant to the proposed middleware approach. It is also important to note that these three classifications have their own unique set of compromises associated with them. For example, by opting for a host-based modification approach, portability is greatly reduced.

Similarly, a middleware approach is likely to perform slower than its host-based counterpart. Hardware-level modifications might achieve the greatest results in terms of portability and performance; however, they are not being considered due to the difficulty of producing new hardware devices and potentially altering the widely adopted USB industry standard.

Attribute	Classification	Technique	Focus Issue
GoodUSB [8]	Host-based	User identification of the connected device	General
LBM [9]	Host-based	Kernel modification of the host system	General
USBeSafe [15]	Host-based	Machine-learning based solution	General
FirmUSB [19]	Host-based	Vetting device firmware	General
Uscramble [20]	Host-based	Lightweight encryption	Eavesdropping
USBGuard [22]	Host-based	Framework for implementing USB authorization policies	Whitelisting
USBProxy [21]	Host-based	Allows filtering of incoming USB packets via user-defined rules	General
Kells [10]	Hardware	Low-level USB architectural changes	Securing USB design
ProvUSB [11]	Hardware	Low-level USB architectural changes	Securing USB design
DeviceVeil [7]	Hybrid (Host-based and Hardware)	A modified PUF IC attached to USB peripheral	Bi-directional authentication
Cinch [17]	Middleware	Attaches peripherals to a logically separate machine with an interposition layer.	General
SandUSB [12]	Middleware	Relaying USB packets from USB to PC	General
USBWall [13]	Middleware	Runs USBProxy as on a physical middleware	General
USBCheckIn [14]	Middleware	Forcing user interaction to verify integrity of device	Spoofing

(Table 2: Classifying some of the related USB defenses as well as a summary of their techniques)

The following table (Table 3) summarizes the existing and desirable features from various related work of different classifications and techniques. Those will be explained in detail in the following chapter.

Feature	SandUSB [12]	USBGuard [22]	Cinch [17]	GoodUSB [8]	USBWall [13]	USBFilter [18]
USB Information Profiling	✓	x	x	✓	✓	x
Keyboard Dynamics	✓	x	x	x	x	x
Device Blacklisting	✓	✓	x	✓	✓	✓
Platform Independent	✓	x	✓	✓	x	x
Payload Matching	✓	x	✓	✓	x	✓
OS Segmentation	✓	x	✓	✓	x	x
USB Policy	✓	✓	✓	✓	✓	✓
Signature-Based Defenses	x	x	✓	x	x	x
Authentication and Encryption	x	x	✓	x	x	x
Driver Vulnera-bilities Defense	x	x	✓	x	x	x
Limiting USB Device Functionalities	x	x	x	x	x	✓
Focus on Ease of Initial Deployment and Use	High	Low	Low	Medium	Medium	Low
Focus on Portability	High	Low	Low	Low	Medium	Low
Deliberate Safe Extraction of Non-Malicious Data	x	x	x	x	x	x
Hidden Partition Attacks	x	x	x	x	x	x
Virtual Environment Break-Out	x	x	x	x	x	x
Middleware Protection	x	-	x	-	x	-

(Table 3: Summary of features in related work)

2.2 Need to Extend Related Work

A gap exists between protection solutions from USB devices and ease of deployment as well as affordability. Expecting a user to behave responsibly and knowingly as to not allow their devices to be compromised or suggesting hardware or kernel-level modifications to their system is rather ambitious in terms of commercial deployment. The goal is to provide an affordable middleware solution against USB-based attacks that could be easily deployed by anyone; one that is fully platform-independent without compromising on being secure. The ideal approach to take, in this case, is the middleware approach.

In addition, papers have only focused on preventing malicious communication, while, in a work environment, it is possible that a user would want to retrieve files from a knowingly malicious storage device without compromising themselves. In order to achieve that, there must be a method in which data can be identified and selectively transferred in a format that does not harm the USB host.

Hence, on close study of papers previously mentioned in the related work (section 2.1), the focal points to be extended are listed below:

- Focus on Ease of Initial Deployment and Use
- Platform Independence
- Portability
- Deliberate Safe Extraction of Non-Malicious Data
- Hidden Partition Attacks
- Virtual Environment Break-Out
- Middleware Protection

2.3 Scope of Work

Implementing signature-based defenses as well as defenses against driver vulnerabilities, assisted by payload matching, device blacklisting or limiting. It is targeted to maximize portability and platform independence. The targeted peripherals are USB flash drives, which means peripherals of other kinds will not be accepted. USB drives are to be completely isolated from user machine and data transmission is allowed only after performing a security check.

In addition, USafeB aims to provide a safe, secure medium for transferring valid data that is located amidst malicious ones to the device to be protected. This functionality should be implemented by providing the user with a mechanism that enables them to safely extract non-malicious data from a flash drive that was deemed malicious, as well as the option to reject all incoming traffic from a given USB device.

Feature	***Proposed Solution***	**Motive**
USB Information Profiling	✓	Collecting useful information for analysis purposes
Keyboard Dynamics	***x***	Irrelevant problem
Device Blacklisting	✓	To increase performance, we will blacklist previously known malicious devices
Platform Independent	✓	To increase the usability of the tool
Payload Matching	✓	To make use of well-known attack signatures to defend against them
OS Segmentation	***x***	Irrelevant problem
USB Policy	✓	Adding some pre-built policies to ensure maximum safety for the user
Signature-Based Defenses	✓	Using +30 online scanners to deeply analyze the suspected binaries
Authentication and Encryption	***x***	Orthogonal problem
Driver Vulnerabilities Defense	✓	Driver vulnerability attacks had a great share of USB attacks recently
Limiting USB Device Functionalities	***x***	Our hardware design prevents us from applying this feature
Focus on Ease of Initial Deployment and Use	**Highest**	Highly prioritized.
Focus on Portability	**Highest**	Highly prioritized.
Deliberate Safe Extraction of Non-Malicious Data	✓	To further increase usability.
Hidden Partition Attacks	***x***	Future work
Virtual Environment Break-Out	***x***	Future work
Middleware Protection	***x***	Future work

(Table 4: Table of features in proposed solution)

2.4 Future Work

Defenses should be implemented against newer attacks, such as ones emerging from hidden partition attacks that allow covert data exfiltration. In addition to that, USB firmware might break out of virtual environments. In a middleware implementation that makes use of virtual containers, it is essential to further investigate these types of attacks.

In addition, effort should be placed into protecting the middleware itself. Now, its design allows it to serve as a honeypot for its host system. There is further work to be done in defending and cleansing the middleware in case of infection. The hardware aspect of the middleware should also be protected, from signal and electrical injections as previously mentioned in the threat model (section 1.2.3).

2.5 Conclusion

A survey was conducted regarding existing publications that were written in the topic of defending against USB-based attacks. It was concluded that there needed to be an improvement on the facet of usability and ease of deployment on a defense mechanism that was not a tradeoff with security. In addition, a method of extracting non-malicious data from a USB device is necessary to develop, in order to remove an incentive from the user to opt for using a risky USB device. In the next chapter, the overall design of USafeB will be presented.

References

[1] "USB 2.0 Documents". www.usb.org. 7 May 2018.
[2] “Cyber Secure: A look at Employee Cybersecurity Habits in the Workplace”. February 2017.
https://www.royalit.nyc/wp-content/uploads/2017/02/Royal-IT-NY-Brochure-Cyber-Secure.pdf
[3] J. Tian, N. Scaife, D. Kumar, M. Bailey, A. Bates and K. Butler, "SoK: "Plug & Pray" Today – Understanding USB Insecurity in Versions 1 Through C," (2018) IEEE Symposium on Security and Privacy (SP), San Francisco, CA, (pp. 1032-1047).
[4] USB Killer. https://usbkill.com/
[5] K. Nohl, S. Krißler and J. Lell, "BadUSB — On accessories that turn evil," (2014) in *Black Hat*.
[6] Nissim, N., Yahalom, R., & Elovici, Y. (2017). USB-based attacks. Computers & Security, 70, (pp. 675-688).
[7] K. Suzaki, Y. Hori, K. Kobara and M. Mannan, "DeviceVeil: Robust Authentication for Individual USB Devices Using Physical Unclonable Functions," *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Portland, OR, USA, 2019, (pp. 302-314).
[8] Tian, Dave & Bates, Adam & Butler, Kevin. (2015). Defending Against Malicious USB Firmware with GoodUSB. (pp. 261-270).
[9] Tian, D. J., Hernandez, G., Choi, J. I., Frost, V., Johnson, P. C., & Butler, K. R. (2019, May). LBM: a security framework for peripherals within the linux kernel. In *2019 IEEE Symposium on Security and Privacy (SP)*, (pp. 967-984).
[10] Butler, K. R., McLaughlin, S. E., & McDaniel, P. D. (2010, December). Kells: a protection framework for portable data. In *Proceedings of the 26th Annual Computer Security Applications Conference*, (pp. 231-240).
[11] Tian, D., Bates, A., Butler, K. R., & Rangaswami, R. (2016, October). Provusb: Block-level provenance-based data protection for usb storage devices. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, (pp. 242-253).
[12] Loe, E. L., Hsiao, H. C., Kim, T. H. J., Lee, S. C., & Cheng, S. M. (2016, December). SandUSB: An installation-free sandbox for USB peripherals. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, (pp. 621-626).
[13] Kang, M., & Saiedian, H. (2017). USBWall: A novel security mechanism to protect against maliciously reprogrammed USB devices. *Information Security Journal: A Global Perspective*, *26*(4), (pp. 166-185).
[14] Griscioli, F., Pizzonia, M., & Sacchetti, M. (2016, December). USBCheckIn: Preventing BadUSB attacks by forcing human-device interaction. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)* (pp. 493-496).
[15] Kharraz, A., Daley, B. L., Baker, G. Z., Robertson, W., & Kirda, E. (2019). {USBESAFE}: An End-Point Solution to Protect Against USB-Based Attacks. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2019)* (pp. 89-103).
[16] B Barker, A. W. (2012). Provenience, provenance, and context (s). *See Adler & Bruning*, *2012*, (pp. 19-30).

[17] Angel, S., Wahby, R. S., Howald, M., Leners, J. B., Spilo, M., Sun, Z., ... & Walfish, M. (2016). Defending against malicious peripherals with Cinch. In *25th {USENIX} Security Symposium ({USENIX} Security 16)* (pp. 397-414).
[18] Tian, D. J., Scaife, N., Bates, A., Butler, K., & Traynor, P. (2016). Making {USB} Great Again with {USBFILTER}. In *25th {USENIX} Security Symposium ({USENIX} Security 16)* (pp. 415-430).
[19] Hernandez, G., Fowze, F., Tian, D., Yavuz, T., & Butler, K. R. (2017, October). Firmusb: Vetting USB device firmware using domain informed symbolic execution. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 2245-2262).
[20] Neugschwandtner, M., Beitler, A., & Kurmus, A. (2016, April). A transparent defense against USB eavesdropping attacks. In *Proceedings of the 9th European Workshop on System Security* (pp. 1-6).
[21] "USBProxy-legacy" https://github.com/usb-tools/USBProxy-legacy
[22] "USBGuard software framework" https://usbguard.github.io/