The Anatomy of USB-Based Keystroke Injection Attacks

Maksymilian Miketa
Nottingham Trent University
Nottingham United Kingdom

***Abstract*— USB-based keystroke injection attacks pose a formidable threat to computer security due to their stealth and efficacy. This paper presents a systematic review of the anatomy of such attacks, providing a comprehensive background on USB technology and its inherent vulnerabilities that attackers exploit. The USB protocol's lack of authentication mechanisms permits the execution of keystroke injection via devices such as a BadUSB and Rubber Ducky. These devices, disguised as ordinary peripherals, leverage the protocol's trust to perform unauthorised actions on host machines. This paper categorises the types of payloads that can be deployed through various attacks, ranging from exfiltrating data to taking control over the system and installing malware. Each payload type is analysed to understand the subtleties of its execution. Furthermore, currently available protection methods against these attacks are analysed for their effectiveness and limitations. Finally, the need for continued research and development of more robust defence strategies is discussed to improve security measures and defend against these evolving attacks The findings aim to inform and equip stakeholders with the knowledge to anticipate and mitigate the risks associated with USB-based keystroke injection attacks, contributing to the advancement of secure computing practices.**

Keywords – USB, BadUSB, HID, Rubber Ducky, USBlock, Keyblock, ProvUSB

I. Introduction

Introduced in 1996, the Universal Serial Bus (USB) quickly became a staple for connecting peripherals to computers [16]. Its plug-and-play design is ubiquitous, but this simplicity also brings significant security risks.

The USB gained dominance over other protocols and ports as new generations are released every few years providing more features and increased data transfer speeds with its popularity raising concerns about the protocol's security [21]. Most computers require USB devices such as mice and keyboards as well as other additional peripherals connected to a computer to use for input and/or output which can lead to an attack vector if one of the devices is malicious. The USB protocol has weaknesses for attackers to exploit them in several ways. This paper focuses on USB-based keystroke injection attacks. In these attacks, malicious firmware is embedded within a USB device to mimic a Human Interface Device (HID), such as a keyboard, and execute unauthorised commands on the host system [3]. Since these devices appear harmless, they can bypass standard security and cause serious problems.

A thorough literature review serves as the groundwork for this analysis, examining the techniques used by USB devices to exploit the USB protocol, the challenges in detecting these covert operations, and the limitations faced by current defensive strategies. The discussion extends to potential countermeasures, including disabling autorun features, deploying specialised software, and educating users about the risks of unfamiliar USB devices. The objective is to underscore the persistent threat posed by these attacks and to encourage further exploration into enhancing USB security measures.

In exploring these attacks, the analysis also considers the broader implications for cybersecurity. It reflects on how USB technology could evolve to prioritise security without compromising user convenience. The review suggests that future research should focus on creating USB protocols resilient to such exploits, possibly through advanced authentication methods or hardware-based security solutions.

The widespread adoption of USB devices revolutionised how users connect peripherals to computers. However, this convenience comes with a hidden cost as USB's inherent vulnerabilities make it susceptible to keystroke injection attacks. These attacks allow malicious actors to steal information, take control of systems, or deploy malware simply by inserting a compromised USB device. This paper delves into the technical aspects of these attacks by exploring existing defence strategies and investigating potential future advancements in USB security. Through examining these issues, the research aims to raise awareness of this critical cybersecurity threat and pave the way for the development of more secure USB protocols and user practices.

II. SYSTEMATIC LITERATURE REVIEW

The Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) technique was employed to choose appropriate research papers for the analysis [1]. PRISMA consists of three phases: identification, screening phase and inclusion phase. The identification stage involved finding the research papers. The search strategy was based on keywords, synonyms, and abbreviations related to USB attacks such as "USB attacks", "USB Keystroke Injection", "Rubber Ducky" etc. which were searched on Google Scholar. The publishing date was set from 2018-2023 to eliminate older papers within the search as many papers would have contained outdated research and data due to advancements in technology. The initial search resulted in 384 results. All the identified papers underwent the screening phase which was reduced to 57 papers. Within this phase, duplicates, foreign papers, papers with only an abstract, irrelevant papers to the topic and inaccessible papers were removed. Lastly, 14 papers were chosen in the inclusion phase which were the most relevant and suitable papers for this research.

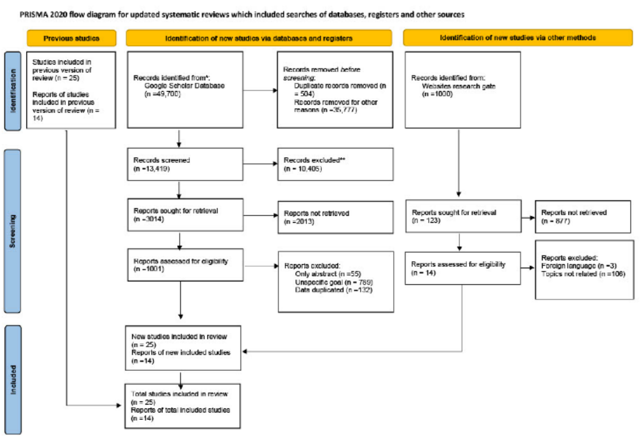
PRISMA 2020 flow diagram for updated systematic reviews which included searches of databases, registers and other sources
Previous studies
Identification of new studies via databases and registers
Identification of new studies via other methods
Identification
Screening
Included
Studies included in previous version of review (n = 25)
Reports of studies included in previous version of review (n = 14)
Records identified from*: Google Scholar Database (n =49,700)
Records removed before screening: Duplicate records removed (n = 504) Records removed for other reasons (n =35,777)
Records identified from: Websites research gate (n =1000)
Records screened (n =13,419)
Records excluded** (n = 10,405)
Reports sought for retrieval (n =3014)
Reports not retrieved (n =2013)
Reports sought for retrieval (n = 123)
Reports not retrieved (n = 877)
Reports assessed for eligibility (n =1001)
Reports excluded: Only abstract (n =55) Unspecific goal (n = 789) Data duplicated (n =132)
Reports assessed for eligibility (n = 14)
Reports excluded: Foreign language (n =3) Topics not related (n =106)
New studies included in review (n = 25) Reports of new included studies (n =14)
Total studies included in review (n = 25) Reports of total included studies (n =14)

Fig 1. PRISMA 2020 Flow Diagram

III. Background

Understanding the USB protocol is paramount before delving into any USB-based attacks. This section provides an insightful overview of USB components and protocol operations, essential for comprehending the intricacies of subsequent attacks.

The USB standard is the primary method for linking peripherals to a host. The communication between the host and connected USB device is based on a tiered-star topology with a dedicated master controller. A root hub is master controller acting as a hub and provides multiple USB ports to connect USB accessories and peripherals.

USB devices fall into two main categories: Input/Output (I/O) devices, which enhance host capabilities, and hub devices, solely facilitating additional device connections to the host.

In addition to facilitating communication between a USB device and the host, USB cables also supply power to devices that are bus-powered. For self-powered hubs and I/O devices to supply enough electrical power, an extra power cable might be needed. The USB cable connections design prioritises power supply by ensuring power pins are longer than signal pins. A microcontroller chip, which consists of a CPU and sometimes a bootloader, controls USB devices. While the bootloader makes it easier to upgrade firmware after production, the CPU runs software, controlling how the device responds to requests from the host. USB devices contain multiple logical sub-devices referred to as device functions. As an example, a webcam with a built-in microphone has two functions that are video and audio [5].

Every function is tied to a specific logical address on the bus, known as an endpoint, which creates a conduit for communication called a pipe. These pipes are split into two groups: message pipes, which are used for quick exchanges of commands and status information, and stream pipes, which accommodate various data transfer processes [2]. For instance:

- **Interrupt transfers** are designed to offer immediate feedback with minimal delay, ideal for peripherals like mice and keyboards.
- **Isochronous transfers** are utilised for tasks that need time-synchronised operations at a fixed data rate, with the risk of data loss, particularly in real-time audio or video scenarios.
- **Bulk transfers** handle substantial, intermittent data movements, making use of spare bandwidth without assurance of immediate delivery or dedicated bandwidth, typically employed for transferring files.

Upon connecting a USB device, the enumeration process unfolds through four key steps:

1. **Detection of device connection:** The host identifies a connected device by monitoring changes in the data lines.
2. **Determination of device speed:** Variations in data lines assist in identifying the device's speed.
3. **Determination of device descriptors:** Devices are recognised through descriptors dispatched to the host, involving a series of queries and responses.
4. **Loading of drivers:** After complete device identification, the host loads a driver to manage the USB device. While standard USB devices typically rely on drivers integrated into the host's operating system, custom drivers may be necessary for non-standard specifications.

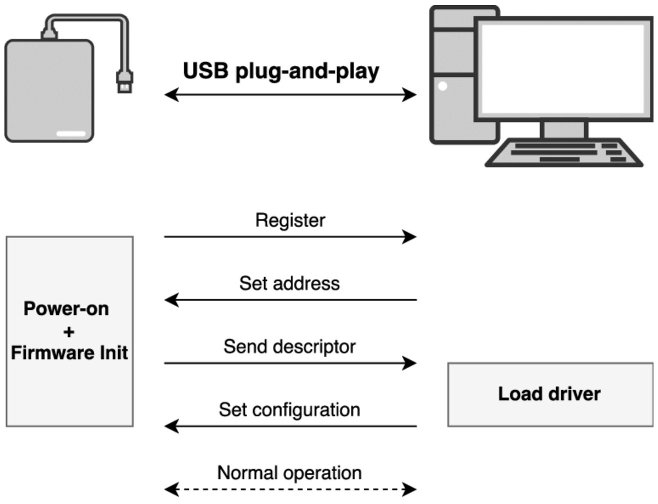

Fig 2. USB Plug and Play Protocol

Today’s operating systems, such as Microsoft Windows, Linux, and Apple Mac OS, use data from the hub about connected USB devices to seamlessly install the drivers they need. By examining the device’s declared interface descriptors and its class, along with the unique vendor and product IDs (VID and PID), the system figures out what the device does and sets up the right drivers for it.

The upcoming section will explore the vulnerabilities inherent in the USB protocol, with a particular emphasis on how malicious USB devices bypass security measures and employ keystroke injection attack by exploiting various attack vectors.

IV. USB ATTACK VECTOR

The USB protocol lacks a mechanism for authenticating devices. Instead, USB hubs accept any declared capabilities from connected devices without verification. Modern USB devices often have multiple functions for valid reasons, such as a mouse that also functions as a display device. These multifunctional devices can pose risks that are difficult for users to assess, making USB a prime target for exploitation.

Historically, the complexity of exploiting USB protocol vulnerabilities was considerable. However, this changed when USB chips capable of being reprogrammed became widely available. Firmware updates are frequently necessary for consumer products due to accelerated market release schedules and limited testing, which can be preferable to the logistical challenges of a product recall. These updates also reduce the technical barriers to initiating USB-based attacks.

Figure 3 illustrates how a USB device announces its capabilities during the setup phase of a control transfer. For example, a device may declare it supports both mass storage and keyboard functions. While the mass storage function might be expected if a user connects a USB drive, without specific device knowledge, such declarations could be used for malicious purposes [7].

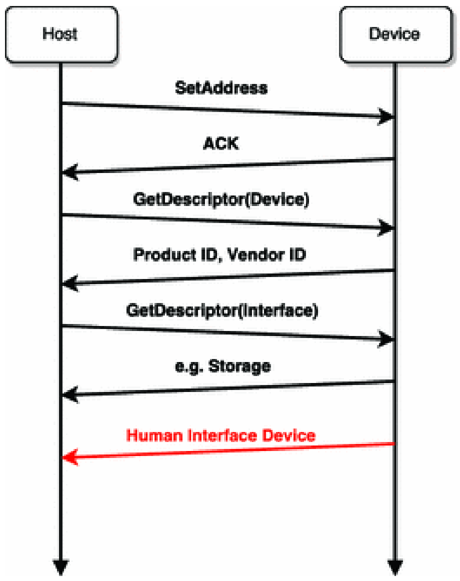

Fig 3. USB Packet Sequence Diagram

The declaration of keyboard functionality requires the operating system to activate a keyboard driver. If this declaration originates from tampered firmware, it marks the initial success of an attack. The compromised firmware can then execute the second phase, which involves simulating keystrokes from a non-existent keyboard. These “keystroke injection attacks” rely on the user not noticing the system interactions caused by these simulated keystrokes, allowing the attack to proceed with its harmful objectives, such as downloading and executing malware.

Various methods have been developed to exploit the USB protocol’s inherent vulnerabilities. The attack vector consists of a user connecting a malicious USB device usually after an attacker performed reconnaissance through social engineering on a user, computer, or organisation to gain information about the user and the system. The attacker can gain information on when a machine is free to connect the USB device and when to disconnect it to avoid detection, the USB can be placed in a non-suspicious location where it is picked up by the user and connected or, it can be physically given to the user. The estimated success rate of a user picking up and connecting an unknown USB device is from 48-98% [10] resulting in a highly attack success rate.

A. Rubber Ducky

A USB Rubber Ducky is a device that resembles a regular USB flash drive but acts as a keyboard when plugged into a computer. It leverages the trust computers have in HIDs to execute pre-programmed keystroke payloads at remarkable speeds. The Rubber Ducky was first introduced in 2010 and was designed to use for security testing but the programmable payloads can range from benign tasks like opening programs to malicious activities such as installing malware, exfiltrating data, or creating backdoors [9]. The attack vector of a Rubber Ducky is the method it uses to deliver these payloads, typically by simulating keystrokes that a user would type. Because of its appearance and the speed at which it operates, it can bypass traditional security measures and execute its payloads before any suspicious activity is detected [6].

B. BadUSB

A BadUSB is a programmable microcontroller that manipulates the firmware of USB devices by programming a seemingly benign USB device, such as a flash drive, to act as a HID like a keyboard or mouse. BadUSB is related to Rubber Ducky but includes a more diverse range of attacks. Once connected to a device, the BadUSB device is automatically trusted by the system and can execute a series of preloaded commands or keystrokes. These actions can range from opening a command prompt to downloading and running malware, all without the user’s knowledge [11]. Additionally, BadUSBs can be disguised as innocuous-looking USB cables that facilitate data transfer and charging while executing malicious scripts. Some advanced versions, like the OMG Cable, contain a hidden Wi-Fi microcontroller that can send payloads wirelessly through the device [17].

V. USB-ATTACK PAYLOADS

Keystroke injection attacks via USB devices, such as BadUSBs and Rubber Duckies, present a significant security threat due to their ability to execute a wide array of payloads. These payloads are scripts or sets of instructions that the device ‘types’ into the host machine, exploiting the inherent trust computers have in keyboard inputs.

Data Exfiltration and Destruction Payloads: The first category of payloads is designed for data exfiltration and destruction. These can be programmed to search for and transmit sensitive information, such as system logs, browser history, or specific files, to an external server controlled by the attacker. Alternatively, they can issue commands to delete critical files or directories, rendering the device inoperable or compromising its integrity [18].

System Control and Access Payloads: Another type of payload aims to establish control over the system. This can be achieved by executing a series of commands that open backdoors for remote access, such as reverse shells. These shells allow attackers to control the system remotely, potentially leading to data theft, surveillance, or further exploitation [20].

PHUKD/URFUKED Platforms: Like Rubber Duckies, these platforms allow attackers to time the injection of malicious keystrokes, providing greater control over when the attack occurs to reduce detection and better chance of the attack succeeding.

Credential Theft Payloads: These payloads are designed to retrieve user credentials from the host machine, often leveraging tools like Mimikatz to extract passwords from system memory [19].

Malware Installation Payloads: Some payloads focus on installing malware, ransomware, or other malicious software without the user's knowledge, often by automating the download and execution process.

The upcoming section will present concise defence strategies against USB attack vectors, focusing on essential preventative measures.

VI. DEFENCE AGAINST USB KEYSTROKE INJECTION ATTACKS

Protecting against USB keystroke injection attacks is essential for mitigating risks and safeguarding computer systems from dangerous payloads, such as those previously mentioned from compromised USB devices. This section will explore various defence strategies aimed at thwarting the sophisticated methods employed by attackers. Understanding the mechanics of these attacks enables organisations to implement robust measures that enhance their cybersecurity defences and avert potential security breaches.

Traffic Analysis and Anomaly Detection: The deployment of systems capable of analysing USB packet traffic is pivotal for identifying unusual patterns that may signal an attack. Solutions such as USBlock are instrumental in this process. USBlock is a security tool that employs temporal analysis to detect and flag devices exhibiting anomalous behaviour, effectively serving as an early warning system against potential threats. By monitoring deviations from normal USB device operations, USBlock helps to pre-emptively address security risks [6].

Software Intermediary Layer: Introducing a software intermediary layer, such as Keyblock, enhances system security by serving as a protective barrier. Keyblock operates between the USB hardware input and the computer's operational processes. It meticulously examines every input, ensuring only legitimate commands pass through. This critical scrutiny helps filter out suspicious or potentially harmful commands, thereby preventing any unauthorised or malicious execution. By acting as a vigilant gatekeeper, Keyblock plays a crucial role in maintaining the integrity of the system's operations [14].

Keystroke Dynamics Monitoring: The implementation of a keystroke dynamics monitoring framework is essential for distinguishing between human and automated typing behaviours [15]. This system works by creating a reference model of normal human keystroke rhythms. Once established, any deviation from this baseline—particularly automated keystroke sequences that could signal an injection attack—is promptly detected and intercepted. This monitoring is vital for identifying and blocking non-human typing patterns that pose a security threat as there is a significant difference in human and automated typing patterns.

Policy Enforcement and User Training: It's imperative to train users about the risks associated with connecting uncertified USB devices and to implement stringent organisational policies on USB device usage. Raising awareness is a key defensive strategy, as well-informed users are less prone to inadvertently introduce compromised devices into secure systems. Restricting access controls for certain users may prevent certain payloads to perform as intended such as gaining access to shell or confidential data. Disabling Auto-Run features can mitigate the risk of inadvertent execution of dangerous payloads helps to block the automatic initiation of software that could be embedded in USB devices upon connection.

Device Authentication Protocols: Employ USB devices that incorporate secure authentication protocols wherever possible. This ensures that only recognised devices are authorised as legitimate input sources, thereby preventing the acceptance of potentially harmful unauthorised devices. Using an architecture such as ProvUSB to only allow pre-approved USB drives within a system [12] or a USBWall for enumerting USB devices and notifying the user if the device musst be removed or can be used [13].

Regular Security Audits: Performing consistent security audits is vital to confirm that all defensive mechanisms are up-to-date, effective, and correctly enforced. These audits are instrumental in uncovering new security gaps and affirming the reliability of established security measures.

USB Plug-and-Play Protocol Analysis: Conducting a comprehensive security analysis of the USB plug-and-play protocol is a critical step in fortifying defences against USB cyber threats. By gaining a deep understanding of the protocol's vulnerabilities, it becomes possible to devise specific security patches. These patches are crucial in minimising the protocol's exposure to exploitation by malicious actors, effectively shrinking the potential avenues for attack. This proactive approach to protocol analysis is a key strategy in reducing the risk of cyber incursions.

By adopting these proactive defensive measures, organisations can establish a formidable shield against USB keystroke injection attacks. Maintaining an adaptive defense stance that evolves alongside new threats and technological advancements is essential for ongoing protection.

VII. CONCLUSIONS

USB devices pose a serious threat due to the ease of keystroke injection attacks. Almost any computer can be compromised simply by plugging in a malicious USB device. This is especially concerning because users rely so heavily on USB devices and often trust them to be safe. Fortunately, there are methods to defend against these attacks. Users can strengthen their defences by adopting stricter USB device usage policies. This includes being selective about what USB devices are plugged in, only using trusted sources for files, and keeping software up to date. Additionally, advancements in the USB protocol itself are crucial to prevent these attacks from happening in the first place. By addressing both user behaviour and technical vulnerabilities, significant progress can be made in stopping USB keystroke attacks to keep users and devices safe.

While this research provides valuable insights into current threats, it also serves as preparation for future security challenges. Although the study was conducted using the Google Scholar database, which provides extensive coverage on this topic, future work could enhance validity by incorporating a broader range of databases. As technology evolves, new attack methods will likely emerge. By understanding how USB keystroke injection attacks work, we can develop better defences against these future threats.

References

[1] PRISMA, "PRISMA," Prisma-statement.org, 2023. http://prisma-statement.org/ (accessed Feb. 21, 2024).

[2] N. Nissim, R. Yahalom, and Y. Elovici, "USB-based attacks," *Compters & Security*, vol. 70, pp. 675–688, 2017, Accessed: Mar. 14, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404817301578

[3] Cimpanu, "Here's a List of 29 Different Types of USB Attacks," BleepingComputer, 2018.https://www.bleepingcomputer.com/news/security/heres-a-list-of-29-different-types-of-usb-attacks/(accessed Mar. 14, 2024).

[4] J. Clark, S. Leblanc, and S. Knight, "Compromise through USB-based Hardware Trojan Horse device," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 555–563, 2010, doi: https://doi.org/10.1016/j.future.2010.04.008..

[5] N. Farhi, N. Nissim, and Y. Elovici, "Malboard: A novel user keystroke impersonation attack and trusted detection framework based on side-channel analysis," Computers & Security, vol. 85, pp. 240–269, Aug. 2019, doi: https://doi.org/10.1016/j.cose.2019.05.0

[6] S. Neuner, A. G. Voyiatzis, S. Fotopoulos, C. Mulliner, and E. R. Weippl, "USBlock: Blocking USB-Based Keypress Injection Attacks," Data and Applications Security and Privacy XXXII, pp. 278–295, 2018, doi: https://doi.org/10.1007/978-3-319-95729-6_18.

[7] B. Anderson, Seven deadliest USB attacks. Burlington, Ma: Syngress, 2010.

[8] S. Angel *et al.*, "Defending against Malicious Peripherals with Cinch," *www.usenix.org*, 2016. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/angel

[9] Hak5, "USB Rubber Ducky," Hak5. https://shop.hak5.org/products/usb-rubber-ducky (accessed Mar. 16, 2024).

[10] M. Tischer *et al.*, "Users Really Do Plug in USB Drives They Find," *IEEE Xplore*, May 01, 2016. https://ieeexplore.ieee.org/abstract/document/7546509/ (accessed Mar. 08, 2020).

[11] S. Han et al., "IRON-HID: Create Your Own Bad USB." Accessed: Mar. 17, 2024. [Online]. Available: https://archive.conference.hitb.org/hitbsecconf2016ams/wp-content/uploads/2015/11/Seunghun-Han-IRON-HID-Create-Your-Own-Bad-USB-Device.pdf

[12] D. (Jing) Tian, A. Bates, K. R. B. Butler, and R. Rangaswami, "ProvUSB," *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 242–253, Oct. 2016, doi: https://doi.org/10.1145/2976749.2978398.

[13] M. Kang and H. Saiedian, "USBWall: A novel security mechanism to protect against maliciously reprogrammed USB devices," Information Security Journal: A Global Perspective, vol. 26, no. 4, pp. 166–185, Jul. 2017, doi: https://doi.org/10.1080/19393555.2017.1329461.

[14] C. D. B. Borges, J. R. B. de Araujo, R. L. de Couto, and A. M. A. Almeida, "Keyblock: a software architecture to prevent keystroke injection attacks," *sol.sbc.org.br*, Nov. 06, 2017. https://sol.sbc.org.br/index.php/sbseg/article/view/19526 (accessed Jun. 16, 2023).

[15] D. Umphress and G. Williams, "Identity verification through keyboard characteristics," International Journal of Man-Machine Studies, vol. 23, no. 3, pp. 263–273, Sep. 1985, doi: https://doi.org/10.1016/s0020-7373(85)80036-5.

[16] G. W. den Besten, "The USB 2.0 Physical Layer: Standard and Implementation," *Analog Circuit Design*, pp. 359–378, 2003, doi: https://doi.org/10.1007/0-306-48707-1_17.

[17] Hak5, "O.MG Cable," *Hak5*. https://shop.hak5.org/products/omg-cable

[18] S. Blanchet, "BadUSB, the Threat hidden in ordinary objects.," 2018, Accessed: Mar. 17, 2024. [Online]. Available: https://www.researchgate.net/publication/331876425_BadUSB_the_threat_hidden_in_ordinary_objects

[19] B. Cannols and A. Ghafarian, "Hacking Experiment by Using USB Rubber Ducky Scripting," 2017. Available: https://www.iiisci.org/journal/PDV/sci/pdfs/ZA340MX17.pdf

[20] S. Vouteva, R. Verbij, and J. Roos, "Feasibility and Deployment of Bad USB," 2015. Available: https://rp.os3.nl/2014-2015/p49/report.pdf

[21] H. Liu, R. Spolaor, F. Turrin, R. Bonafede, and M. Conti, "USB Powered Devices: A Survey of Side-Channel Threats and Countermeasures," High-Confidence Computing, p. 100007, Mar. 2021, doi: https://doi.org/10.1016/j.hcc.2021.100007.

[22] B. Borges *et al.*, "Time analysis of attacks to USB plug-and-play by human interface device emulation and keystroke injection," 2017.