BADUSB-C: Revisiting BadUSB with Type-C

Anonymous Authors

***Abstract*—The security of USB protocol has been paid extensive attention to, despite its wide usage. Due to the *trust-by-default* characteristics, USB security has caused serious problems. For example, a well-known firmware attack, BadUSB, performs malicious operations on the victim hosts through disguising normal USB devices as human interface devices like keyboards and mice. However, BadUSB suffers from several limitations. Attackers cannot obtain the status of user interface to conduct precise attacks, and they can be detected by multiple existing countermeasures. In this work, we extended BadUSB to support the new features of USB Type-C and proposed a multi-mode attack model, BADUSB-C. It obtains UI status to make attacks more precise and effective, and also bypasses existing defense methods for BadUSB. To the best of our knowledge, BADUSB-C is the first attack model utilizing USB Type-C. To validate the usability and effectiveness, we conducted extensive experiments and a user study on a real-life attack scenario. 10 volunteers are involved in, and 4,172 of 94,058 records obtained by BADUSB-C in the user study are related to sensitive information. We also discussed the recommended countermeasures for our attack model, including isolated UI rendering, which may be inspiring for future research on defense methods.**

***Index Terms*—USB; BadUSB; Type-C; Attack**

I. INTRODUCTION

The Universal Serial Bus (USB) protocol has become popular around the world since its appearance, as it provides a unified and easy-to-use approach for a large range of devices to communicate with each other. From version 1.0 till now, USB specification has evolved rapidly and offered more and more functionalities. Nowadays, devices with USB support are ubiquitous.

On the other side of the coin, the security of USB has caused serious problems. The designers of the USB protocol did not care much about security issues as they wanted to make an easy-to-use protocol. There are more than four hundred vulnerabilities referencing USB on CVE list [1]. As a result, many attackers exploit these vulnerabilities and the *trust-by-default* characteristics of USB to conduct attacks, which puts the privacy and financial security of USB users in danger [2].

BadUSB is a well-known class of firmware attacks [3]. These attacks are conducted through modification of the device firmware, which will disguise normal USB devices as other types of devices that are *trust-by-default* by the hosts. Typical simulated devices include HID (short for human interface devices, including keyboards, mice, etc.) and disks. Utilizing BadUSB, attackers can pretend themselves as normal users, typing malicious commands to victim computers, downloading and executing malicious scripts, and copying out private data from disks. Such attacks can easily escape from traditional anti-virus software since it is hard to distinguish them from normal USB devices.

Despite the advantageous features of BadUSB, there exist several limitations as follows. (1) Attackers cannot conduct attacks precisely, which decreases the capabilities of BadUSB attacks. When performing attacks on another host, the attackers can not obtain the current user interface (UI) status which limits them from taking subsequent moves. For example, it is hard for attackers to locate specific functional UI patterns like buttons and links on victim computers by disguising USB devices like mice. That explains why typical BadUSB attacks often only stay in the command line, using commands to download malicious scripts for execution. However, these attacks may be intercepted by anti-virus software or firewall due to the usage of the host network. (2) To our best knowledge, existing BadUSB attacks only utilize the features of USB 2.0. The release of USB 3.0 makes USB more powerful, with a higher transmission rate for data and the support towards a larger range of peripherals including DisplayPort, HDMI, PowerDelivery, etc. BadUSB attacks can become more effective with the help of newly supported features in USB 3.0. (3) There have emerged multiple efficacious countermeasures after the appearance of BadUSB. For example, GoodUSB offers a defense method by limiting the functions of USB devices to users' expectations [4]. It provides a graphical interface for users to describe the functionalities or roles of the USB device and reject any usage beyond the description.

In this work, we proposed our approach addressing the limitations mentioned above and implemented a multi-mode attack model of USB, named BADUSB-C. BADUSB-C extends BadUSB to support the features of USB Type-C. Since USB Type-C is capable to transfer video stream data, BADUSB-C could obtain the information of victim's graphical interface during attacks. Combining it with the emulation of traditional HIDs like keyboards and mice, attackers are capable of performing precise attacks. We did experiments to verify that BADUSB-C could also bypass many countermeasures for BadUSB since they often rely on interaction with the graphical interface. Moreover, we implemented multiple attacking modes of USB attacks based on our approach to verify its effectiveness, including HID Emulator Mode, Video Capture Mode, and Full Control Mode. To improve the efficiency and performance of BADUSB-C, we designed a filtering algorithm to preprocess the video data before network transmission. Moreover, we conducted a series of experiments for each attack mode as well as for different types of devices, including smartphones, personal computers, and tablet computers, to validate the usability of BADUSB-C. We also conducted a user study for attacks in sharing power banks, one of the application scenarios of BADUSB-C. During the user study, we obtained a large amount of sensitive data through BADUSB-C, since users are usually unconscious of the necessity to check the USB devices they plugged in for security purposes. 10 volunteers are involved in the user study. At last, we found that 4,796 of 94,058 records obtained by BADUSB-C are related to privacy information of users. After the validation of our attack model, we proposed several defense methods as countermeasures, including external hardware authorization, distrust-by-default, etc. It is worth noting that we designed a method, called isolated UI rendering, to separate the user interface into sensitive and insensitive layers. Only content on the insensitive layer will be passed to the insecure driver and thus rendered on the external display, which protects content on the sensitive layer.

We summarize our key contributions as follows:

- To our best knowledge, this is the first work to utilize new features of USB Type-C. The combination of new support with conventional BadUSB makes attacks more precise and effective.
- Our approach can bypass many previous countermeasures of BadUSB.
- We conducted multiple experiments and a user study to validate the usability and effectiveness of BADUSB-C. We also proposed several countermeasures for our attack model, which are reasonable and insightful.

The rest of this paper is structured as follows. Section II provides the background of USB specification. Section III introduces the existing work of USB security from the aspects of attacking and defense respectively. In Section IV, we present the threat model and the overall implementation of BADUSB-C in three different modes. The experiments and user study we conducted are featured in Section V. We present the possible countermeasures of BADUSB-C in Section VI. The limits and impacts of our approach are discussed in Section VII, and the conclusion lies in Section VIII.

II. BACKGROUND

We first introduce the development of USB specification and emphasize the key points adopted in this work. We also organized a brief timeline for introducing key points of each protocol in Table I.

Proposed in 1996, USB 1.0 [5] was developed to provide a unified interface and thus reducing the cost of reconfiguring the software. It is worth mentioning that as a polled-bus interface, all data transfers are initiated by the host.

Right after one year of the appearance of USB 1.0, a standard named HID (Human Interface Device) [6] was born based on USB. HID is designed with to unify the implementation for devices like mice, keyboards, etc. Before its appearance, the standard is divided between manufacturers, for example, the mouse of Company A may use X-Y coordinates to represent its location while the mouse of Company B uses relative displacement. This means every device needs its own driver to work. After HID, users only need to write one driver for an entire class of HIDs. Furthermore, HID standard also requires all devices to be PnP (*Plug-and-Play*), which is indeed convenient but insecure too.

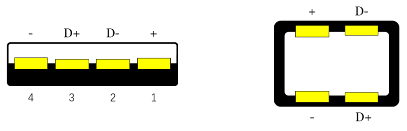


Fig. 1: USB 1.x & 2.x Connector.

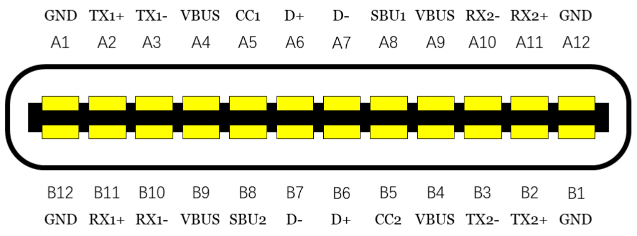

GND TX1+ TX1- VBUS CC1 D+ D- SBU1 VBUS RX2- RX2+ GND
A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12
B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1
GND RX1+ RX1- VBUS SBU2 D- D+ CC2 VBUS TX2- TX2+ GND

Fig. 2: USB Type-C Connector.

In 1998, the first widely supported USB protocol was born. USB1.1 [7] provided two data transfer rates which are low speed (1.5 Mbit/s) and full speed (12 MBit/s). At this point, due to the transfer limit, it only supports limited types of devices like keyboards, mice, etc.

In 2000, USB 2.0 [8] specification was released. With high speed (480 Mbit/s) mode introduced, printers, cameras, CD-ROM drives, and network cards were supported in this revision. Such a high data transfer rate also gave rise to the popularity of “flash drive”, a portable device that allows physically transferring data around [2]. Though various peripherals were supported in USB 2.0, there was no reliable way to identify the type of device. This security flaw allowed attacks like BadUSB [3], [9].

USB 3.0 [10] was announced in 2008, with a super speed (5 Gbit/s) data transfer rate. Like its predecessor, more classes of peripherals were supported in this revision. In 2013, the USB Type-C connector standard was introduced as a part of USB 3.1 [11], providing a unified connector type for PowerDelivery (PD), Thunderbolt, DisplayPort, and HDMI. Yet no improvement of security was introduced in 3.x revisions, meaning any device claiming itself as a monitor can capture the video stream from the host. Exposing such a multi-propose connector unprotected is insecure and allows attacks like this work BADUSB-C. In 2017, USB 3.2 [12] was released, doubling the data transfer rate (20 Gbit/s).

As illustrated in Figure 1, the original USB 1.x & 2.x connector only has two pins for data transferring (D+ & D-), which has significantly limited data transfer rate (5 Gbits/s Max) and cannot support for peripherals like DisplayPort (10.8 Gbit/s Min). Apart from that, support for other peripherals also require dedicated transferring lane as their standards are not compatible with USB in most cases.

Thus, to provide support towards a wider range of peripherals, a 24-pins standard called USB Type-C [13] is introduced in 2013 by USB-IF [14]. As it is designed to be double-sided, the number of actually usable pins is halved. Nevertheless, this standard has largely enhanced the capability of the USB 3.x protocol. As presented in Figure 2, Type-C added two high-speed data lanes (TRX1 & TRX2) and kept the original data lane (D+ & D-). The added lanes are used exclusively to support peripherals like DisplayPort while the kept data lane transfers USB packets.

During the development of USB specification, security was rarely considered. As the USB-IF believes it is the duty of original equipment manufacturers (OEMs) to decide whether security features should be implemented [15]. But the divergent implementations give a chance for attacks like BadUSB [9] and our BADUSB-C.

III. Related Work

We survey related works on USB attacks in Section III-A and USB security defense in Section III-B, respectively.

A. USB Attacks

During the development of the USB protocol, many USB-based attacks were proposed, ranging from DoS (Denial of Service) to protocol masquerading.

From the kernel perspective, its USB software stack generally expects devices to follow the USB standard and may not consider corner cases of malformed USB packets. Based on this, Facedancer [19] and Syzkaller [20] use fuzzing techniques to uncover the bugs lying in the kernel drivers. These bugs can cause kernel crashes and lead to a DoS attack. Though this poses a great challenge to the availability of a system, this attack still requires physical access to the host and is unable to cause more damage other than DoS.

In the field of USB security, protocol masquerading is also a widely used attack scheme. Due to the lack of authentication in the USB protocol, malicious devices can hide their real functionality with re-written firmware [3], [9], [21]–[24]. These works rewrite the firmware of a normal-looking flash drive, which allows it to pretend like other devices. When these modified drives are connected to the host, they could be recognized as a keyboard or mouse. Then the attacker can execute malicious payloads as they were using the victim's devices. Due to the limitation of USB 2.0 [8] protocol, these BadUSB attacks [3] are unable to obtain video feedback from the victim as video stream was not supported until USB 3.0 [10]. Though USB 2.0 does not support video transmission, there exists a protocol called MHL (Mobile High-Definition Link) which extends the USB standard and allows the video signal to be transmitted through the USB interface. JFC [17], short for juicy filming charging is such a work that abuses this standard and exfiltrate video data from the victim without permission. However, in JFC, this data exfiltration is not combined with BadUSB attacks, and MHL is an outdated standard, which limits its capability.

Besides attacking from the protocol perspective, previous works use a USB device as a payload delivery means. Duqu [16] uses a user-mode rootkit to hide malicious files on the USB storage device, [25] uses a zero-day exploit and malicious *autorun.inf* to execute the malware automatically. There are also works like [26]–[28] following the same paradigm and performing code-injection attacks. These attacks are much more damaging and flexible, but they require certain existing flaws like a [29] vulnerability and USB is merely a payload delivery method.

As a data transmission protocol, USB inevitably leaks electromagnetic signals to the environment which may contain sensitive information. Leveraging this physical phenomenon, previous works [24], [30]–[38] eavesdrop on leaked signals and recover the sensitive data. In a similar fashion, [39], [40] emit electromagnetic emissions by data injection on the bus with the connected USB devices as an RF transmitter and [41], [42] inject analog power to cause physical damage to the host machine. Even though the data, including the video data, could be recovered in this way, these attacks for executing malicious code are too difficult to work, and invisibility is a problem that cannot be ignored due to the spatial locality of radio frequency.

Since USB 3.1 was introduced with USB Type-C in 2013, display port and HDMI connectors have been provided by USB Type-C, transferring of video data can be combined with the other attacks, like protocol masquerading, protocol corruption, and code injection. This has paved the path for our BADUSB-C.

B. USB Security Defenses

Many defenses have been proposed to defend against BadUSB attacks [2].

From the hardware perspective, the BadUSB attack requires 'D-' and 'D+' pins which are defined by the protocol to transmitting data. Without these pins, data cannot be transferred via a USB cable. Based on this fact, USB Condom [43] is a hardware solution to block data channels by adding a blocker in the connector. This blocker can cut off the 'D-' and 'D+' connection while leaving power pins intact. However, this method poses a great challenge to the plug-and-play property of USB, as once it is deployed, it stops all USB functions other than charging.

Under the premise of ensuring the full functionality of USB devices, some works improve the security during the connection establishment. Windows Defender ATP [44] maintains a whitelist of USB devices, only devices on the whitelist are allowed to communicate with the host. This prevents all potential attacks from untrusted devices, however, this requires users to have a certain security awareness and technical background to maintain a valid whitelist. For example, a naive user may add the USB device from unknown sources to his/her whitelist without precaution. Fortunately, some designs have proposed to overcome this drawback. For instance, Mohammadmoradi *et al.* [45] propose a strategy to generate such a whitelist automatically. This strategy first generates a unique fingerprint for each device based on its functionality. Then these fingerprints are used to maintain a secure and valid whitelist of USB devices. There is another work mediating USB connectivity for industrial control. TMSUI [46] relies on the rich experience of administrators to build a whitelist. However, some modified USB devices may hide their real functionality from the user.

Year	Protocol Version	Supported Peripherals	Transfer Speed	Attacks
1996	USB 1.x [5], [7]	Keyboard, Mouse...	1.5 Mbit/s or 12 Mbit/s	HID Emulating(BadUSB) [3]
2000	USB 2.0 [8]	Flash Drive, High-Definition Link, CD Driver...	480 Mbit/s	Autorun Attack [16], Juice Filming [17], [18]
2008	USB 3.0 [10]	/	5 Gbit/s	/
2013	USB 3.1 [11]	HDMI, DisplayPort, ThunderBolt...	10 Gbit/s	BADUSB-C
2017	USB 3.2 [12]	/	20 Gbit/s	/

TABLE I: USB Protocol Timeline.

To solve this flaw, GoodUSB [4] reports the functionality claimed by the device to the user and let the user decides whether to authorize. When a device is plugged in, GoodUSB loads its driver but limits its functionality until a series of authorizations are completed. These authorizations are designed to be performed manually. As BadUSB is normally unable to obtain the video stream of the host, an attacker cannot complete this authorization with an automatic script. Thus this defense is sufficient for a normal BadUSB attack. But if the attacker has the access to victim's screen, GoodUSB would be bypassed easily as the attacker can complete this authorization manually and perform subsequent attacks.

After the USB enumeration and driver loading, some related works archive 'defense in depth' against BadUSB attacks. Neuner *et al.* [47] prevent BadUSB attacks from the malicious flash drive by analyzing the temporal characteristics of BadUSB-liked attacks. This defense mechanism is effective because the attacker cannot obtain the screen of the host using BadUSB. In this case, the malicious device can only inject keystrokes in a very short time to reduce the risk of being discovered. This defect causes the typing characteristics of BadUSB to be detectable. Pham *et al.* [48] optimize windows security features, which can block the execution of unsigned files and the installation of unsigned drivers carried on portable media. Moreover, in GoodUSB, a VM is deployed in the host as a honeypot to detect and stop malicious behavior of USB devices.

In addition to injection attacks, data theft attack is also one of the focuses of the academic community. As mentioned in Section III-A, there exists an attack called JFC (Juicy Film Charging) [17] which abuses the MHL standard to steal video stream from the victim. In order to mitigate this issue, Meng et al. [49] proposed a statistical model using status like GPU/CPU usage to detect JFC attacks.

Therefore, there exists a clear trade-off between the effectiveness and the plug-and-play property. Though hardware disabling solutions like USB Condom archives almost absolute security, the functionality of USB is sacrificed. Other solutions like GoodUSB or whitelist are either bypassable or insufficient under certain scenarios. Vendors may sacrifice complete security to improve usability, which allows attackers to take advantage of.

We summarize the former efforts on USB attacks and defenses in Table II. [24], [30]–[42] are not included in this table, because these works are all recovering or injecting signal on the physical layer, we only illustrate the effectiveness of each defense against various attacks, including our BADUSB-C, on the software layer in Table II.

IV. BADUSB-C

A. Threat Model

We build our threat model on the basic assumptions that common users without technical background would not treat normal-looking USB device as malicious and be cautious about them. This assumption is consistent with existing works [18]. Moreover, we neglect the effect of notifications from USB devices, as users may not have the knowledge to fully understand such notifications. In fact, during our experiment, no device had security notifications raised about suspicious USB devices and only mobile devices had raised a charging notification about our BADUSB-C, which may not be a sufficient warning for users.

We also assume the victim's device is equipped with fully functional USB 3.x protocol and USB Type-C connector. As the USB 3.x standard is common nowadays, this assumption can be easily fulfilled especially in recent devices.

B. Implementation

As introduced in Section III, existing works [3], [9], [21]–[24] focus on BadUSB attacks. Many of these take advantage of the *trust-by-default* policy of PC, pretend to be normal HID devices and utilize USB protocol to perform attacks. However, these attacks suffer from various drawbacks: ❶ attackers can only simulate limited types of devices such as HIDs (like keyboards and mice) and disks, which makes the attacks less effective; ❷ accurate attacks could not be performed due to lack of user interface (UI) status. Whatever HID the attackers simulate their USB devices to be, they could not obtain the UI to check the status information, which makes it nearly impossible to carry out their attacks precisely or know the effects after attacks.

Based on the drawbacks introduced above, several defense mechanisms were proposed. GoodUSB [4] implemented a CAPTCHA-like [50] procedure, which requires user to complete certain challenges when an unknown HID device is plugged in. As the BadUSB is only able to emulate a limited kind of devices such as HIDs or disks, by no means the BadUSB can bypass it.

In this work, we utilize the new features of USB 3.x [11], [12] to address the problem above. Benefiting from the latest protocol, we simulate external display and thus obtain the video stream to perform accurate attacks. Thus, our BADUSB-C is capable of bypassing defenses such as GoodUSB. Taking advantage of USB 3.x [11], [12], BADUSB-C can fetch the video stream of the victim and complete the challenge manually by the attacker. Moreover, with video capability

Defense \ Attack	Facedancer [19], Syzkaller [20]	[3], [9], [21]–[24]	JFC [17]	Duqu [16], [25]–[28]	BadUSB-C
USB condom [43]	●	●	●	●	●
Windows Defender ATP [44], Mohammadmoradi *et al.* [45], TMSUI [46]	○	◐	◐	◐	◐
GoodUSB [4]	◐	●	●	●	○
Neuner *et al.* [47]	○	●	○	○	○
Pham *et al.* [48]	○	○	○	●	○
JFCGuard [49]	○	○	◐	○	◐

● means that the defense is effective
◐ means that the defense is partial effective
○ means that the defense is not effective

TABLE II: Effectiveness of Defenses against USB Attacks

combined with traditional BadUSB, our BadUSB-C achieved multiple new attack paradigms and are tested under various real-world scenarios.

As there were various BadUSB implementations available, this work focuses on our new extensions. Next, we first introduce the components we used in BadUSB-C, then we focus on the three different attack modes we implement for various scenarios.

1) Attack Model: Figure 3 shows the architecture of our attack model. ① represents the victim's devices; ② is our BadUSB-C; ③ is the attacker's remote PC. The details of each component in BadUSB-C are as follows.

- USB 3.x Hub exports the USB 3.x connector into various ports, like DisplayPort, USB 2.0 port, etc.
- Video Capture Card converts DisplayPort signal into compatible data, which is later processed by the Single Board Computer (i.e., an embedded computer).
- HID Emulator emulates HID device which can be controlled from the attacker.
- Single Board Computer processes video stream from the victim via the Video Capture Card or sends commands to the victim via HID Emulator.
- Wi-Fi/GSM Module transmits the sensitive data or receives commands to/from the remote attacker PC.

HID Emulation Mode. This mode majorly relies on the "HID Emulator" in the Figure 3 to send constructed HID packet to the victim. These constructed HID packets are interpreted by the victim as valid keystrokes and mouse moves. Thus, the attacker can executes arbitrary scripts on the victim's device. This is the function of the original BadUSB. Based on this, we made the following improvements.

First, BadUSB-C supports defense-bypassing. As the original BadUSB cannot get feedback from the victim, defense mechanisms like GoodUSB [4] are sufficient to prevent these BadUSB attacks. However, these defense mechanisms leverage visual notification to block access for unknown USB devices. This means that when the defense requests authorization from the user, our BadUSB-C is able to capture the content of authorization challenge and bypass it. After the defense is

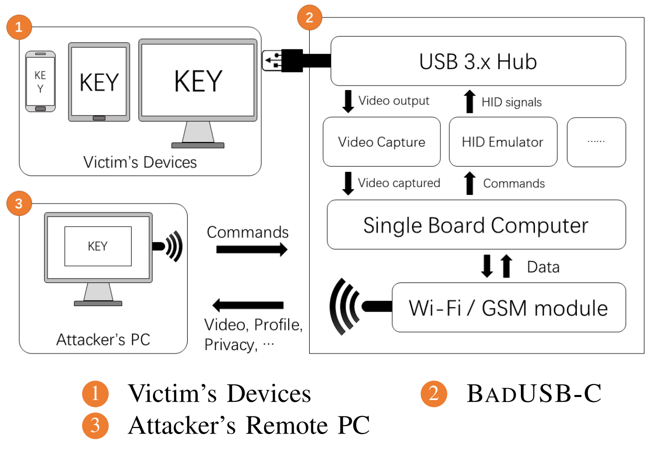


Fig. 3: Attack Model

bypassed, BadUSB-C continues the emulation and script execution, resulting in a successful attack.

Moreover, as the mouse relies on the visual feedback to work properly, its emulation and automation were not supported by the original version of BadUSB. Yet with the video output support from USB 3.x, our BadUSB-C implements a full-functional mouse emulation. This function enables attacks toward pure GUI programs and shows attacks in the mobile attack scenarios. Details can be obtained in Section V.

The advantage of this mode is that it archives defense bypass and attack feedback with video streaming. Moreover, with mouse supported, BadUSB-C extends the original BadUSB attack and results in further attacks in mobile devices.

Video Capture Mode. BadUSB-C under this mode does not emulate other USB device and solely relies on the video stream function of USB 3.x; it uses "Video Capture" component as shown in the Figure 3 to transmit the stream to the embedded "Single Board Computer". The victim's device would mistakenly treat BadUSB-C as an external monitor and output its video stream. This stream is latter processed by the embedded computer to extract sensitive data.

When running in this mode, BadUSB-C passively processes the victim's video stream and detect for "valuable" private data. The data is considered as "valuable" or not is decided by a customized detector, we implemented a simple payment code detector for the poof-of-concept purpose, which demonstrates that we successfully transfer money from a victim to an attacker. More detail can be obtained in Section V.

It is worth mentioning that BadUSB-C under this mode is completely passive, making it hard to be detected. With different detector implemented, BadUSB-C under this mode is capable of serving more purposes.

Full Control Mode. In BadUSB-C, we have implemented all components required to control a computer/mobile device completely, including a video stream and a keyboard/mouse emulation. Thus with all components enabled, not only the victim's device treat BadUSB-C as an external display, but also a valid HID input source. Hence we can archive complete hijack of victim's device.

BadUSB-C under this mode follows a simple logic. BadUSB-C receives video stream from the victim's device and redirect it to the attacker via GSM/WiFi. In the meanwhile, BadUSB-C also receives keystrokes and mouse movement from the attacker through GSM/WiFi and replays them to the victim by the USB emulation.

This mode enable attacker to perform delicate operations that are beyond automation. Moreover, this mode provides a backdoor that does not require host network and thus is undetectable by the firewall running on the host machine.

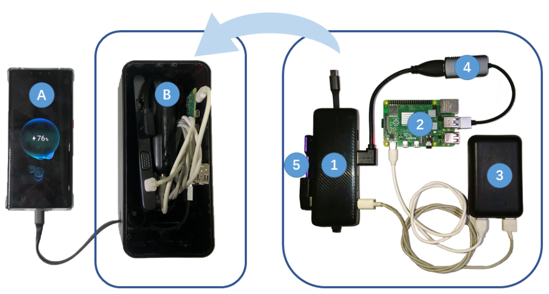
The advantage of this mode is that it can completely hijack the victim device and provide a backdoor beyond detection of firewalls; however, this complete control also comes with the price of high power consumption and risk of being detected by the user.

V. Experiment

To evaluate the effectiveness of BadUSB-C in different modes, we conducted three experiments on BadUSB-C using devices with USB Type-C capabilities from different OEMs, including a mobile phone, a tablet, and a laptop.

Setup. As mentioned in Section IV, our BadUSB-C only requires common components that are easily accessible online or in any electronic store. Here we choose the following parts to build a prototype. To begin with, we choose the Raspberry Pi 4B [51] as the embedded Single Board Computer inside BadUSB-C, which is powerful enough to process video data and has an onboard WiFi chip. As for the HID Emulator, we use an Atmel ATMEGA32U4 board [52] with USB protocol support, which is able to emulate multiple HID devices with our modified firmware. About the USB 3.x Hub, we use one from the UGREEN [53], which supports HDMI, USB 2.0, and many other exported peripherals. Apart from these essential parts, we also use an auxiliary power-bank to provide power for the Raspberry Pi and the mobile devices used by the victim. The image of our BadUSB-C prototype can be found in Figure 4.

Ⓐ is a Huawei mobile phone as the victim's device; Ⓑ is a compact look of BadUSB-C prototype; ① is the USB 3.x Hub; ② is a Raspberry Pi 4B as the Single Board Computer; ③ is an auxiliary power bank; ④ is the Video Capture Card; ⑤ is an Atmel ATMEGA32U4 board as the HID Emulator.



Ⓐ Victim's Device	Ⓑ BadUSB-C
① USB 3.x Hub	② Raspberry Pi 4B
③ Auxiliary Power Bank	④ Video Capture
⑤ ATMEGAA32U4 Board	

Fig. 4: BadUSB-C Prototype

A. *HID Emulator Mode*

In the experiment of HID emulator mode, we used Lenovo Xiaoxin Pro 13 2020, a PC in Windows 10/Ubuntu OS with two USB Type-C interfaces as the target device. During this experiment, BadUSB-C disguised itself as a normal keyboard and a home-brewed version of GoodUSB [4] is deployed to test the defense-bypassing. We have tried to deploy the original GoodUSB on the target device; as the GoodUSB is proposed in 2015, the code is outdated and cannot be executed at target device; so we have to home-brew a similar defense mechanism based on its paper [4]. In the beginning, our home-brewed version of GoodUSB asked the victim to manually complete a CAPTCHA to authorize the new USB device, a.k.a our BadUSB-C. However, our attack can enter keystrokes to complete the CAPTCHA with the video stream by BadUSB-C. Up to this point, the bypass of defense like GoodUSB is completed and our BadUSB-C becomes a trusted device. Next, our BadUSB-C works similar to the original BadUSB, inserting keystrokes to perform arbitrary execution on the laptop. We tested three scripts, ranging from reverse shell backdoor to malware payload execution, all of which resulting in success. With this experiment, we have demonstrated the defense-bypassing is successful using BadUSB-C.

B. *Video Capture Mode*

During the experiment of privacy extraction via our video capture mode, we chose *HUAWEI P30*, a smartphone in EMUI 9.1 (Android 9.0 based) with a USB Type-C interface, as the target device. In the privacy extraction experiment, BadUSB-C passively captured video from the victim's device and used *OpenCV* to identify the sensitive information from the video stream. When the victim viewed text or photos with text, BADUSB-C used the techniques of Optical Character Recognition (OCR) to extract text from corresponding video frames. In this experiment, the attacker had successfully extracted text like name, address, ID number, and other sensitive personal information. We also tested the payment code extractor, which enables an attacker to identify payment code in the video stream and perform transactions without the password. As this is also a part of our case study, more details about the extracted sensitive data can be found in Section V-D and Table IV.

Note that the video capture mode only needs to process the victim's video stream locally, it does not need to transmit the real-time video back to the attacker, which is needed when the network connection between BADUSB-C and the attacker is not stable.

C. *Fully Control Mode*

To test the capability of the full control mode, we chose iPad Pro (3rd generation), a tablet in iOS 14.3 with a USB Type-C interface, as the target device in this experiment. Besides disguising as normal HID devices like a conventional BadUSB [3], BADUSB-C also transmitted real-time video stream from the target device to the attacker via WiFi. After the connection establishment, the attacker performed a series of actions to test the capability of BADUSB-C. In the beginning, the attacker accessed the album application on the iPad and obtained all the photos inside. After that, the attacker sent messages via victim's account. At last, the attacker performed a transaction using the financial application. All of these tests resulted in success.

Through this experiment, we have found that with video transmission and mouse emulation, BADUSB-C extensively expanded the attack capability of BadUSB, especially in mobile devices. In short, we have archived the complete hijack of the victim's device in this experiment.

D. *Case study*

1) Background: We first introduce the technical background of our case study, sharing power bank service and QR code payment.

Sharing Power Bank. Sharing power bank provides users with short-term rental of power banks. The company deploys power bank stations in the city and users can rent a power bank from any of the power bank stations, charge their device on the trip, return the rented power bank to the near stations, and pay the rental fee.

Power bank sharing is a popular service in Asia, power bank stations are deployed in markets, stores, and even newsstands. For example, Figure 5 are photos of two power bank stations in China, which are taken outside of a supermarket. Brick [54] is also such a power bank sharing service provider from Sweden. It provides power bank rental service all over Sweden and is planning on expanding its service to entire Europe.

Not only does sharing power bank provides convenience to users, but it also brings security issues. We noticed that most of the power bank stations do not check the integrity of power banks during the rental process, and users are hardly cautious

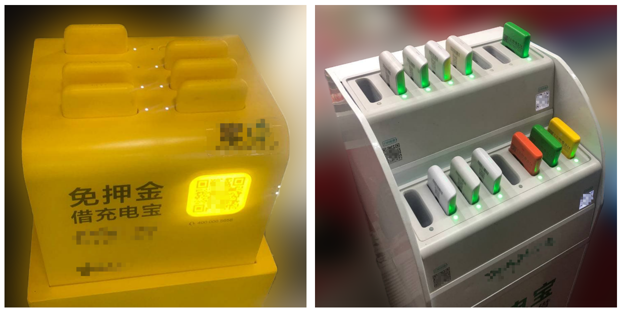


Fig. 5: Two Power Bank Stations.

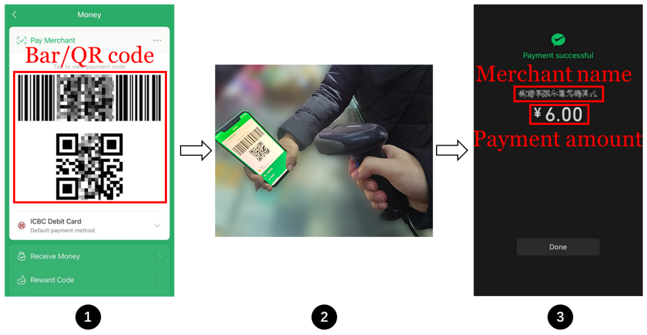


Fig. 6: Bar/QR Code Payment Procedure

to check the power banks when connecting their devices. An attacker is able to tamper rented power banks and return them to a power bank station causing a potential threat to subsequent users.

QR Code Payment. QR code payment is a new type of payment method that is popular in Asia. Its most well-known cases are WeChat Pay [55] and Alipay [56]. QR code payment provides merchant and client a convenient way of offline payment while ensuring equivalent security as the credit card. As illustrated in Figure 6, QR code payment is typically performed in the following steps: ❶ The client presents the payment QR code on the mobile device to the merchant. The QR code is encoded with a globally unique ID to identify the client's account. ❷ The merchant scans the payment QR code and charges the corresponding amount of money. By presenting this QR code, the client authorizes the proceeding transaction. ❸ After confirmation, the payment service provider proceeds with this transaction and returns the payment result to both the merchant and the client.

Next, we explain one type of payments called micropayment. A micropayment is pre-determined by the payment service provider with thresholds in the user agreement. In real-world scenarios, payment service providers use different rules for micropayment purchases. For example, WeChat Pay [55] regards transaction under USD $154 as micropayments. Different from a typical payment procedure, when micropayments are made, confirmations can be applied automatically without clients' permission which aims to provide convenience to both merchant and client. If a victim's payment code is

Keyword	Text	Name	Frame Number
username	X 8B cas.******.edu.cn Username: 11****18 Password:	<user1>	385
username	Login Weibo Login with SMS and verification code ...... +86 151****4587	<user5>	1947
username	QQ 14*****50l Login with phone number New user registration 2345678 9 0	<user3>	4308
username	connect to *** username h*****l Save account information Open VPN.....	<user6>	7925
phone number	Login with phone number ...... +86 186****2483 l	<user1>	313
phone number	Log in with your mobile phone number. ...... mobile phone number 131****9310	<user9>	210
@	contact email 30*******7G@qq.com, contact phone 027-88******	<user7>	5324
@	Account 73*****5@qq.com @qq.com @163.com @gmail.com	<user6>	621

TABLE III: Examples of Searching OCR Results with Keywords.

Keyword	Account	Password	Phone number	Email	Username	Regexp for email
Number of records containing keywords	680	1,596	1,510	164	522	78
Total Number of Records	4,172					

TABLE IV: Privacy Extraction Result.

leaked to the attackers, they can use that code to authorize multiple micropayments without needing permission. In order to prevent such cases, both WeChat Pay [55] and Alipay [56] have designed a refreshing mechanism for the payment code, which is to refresh the QR code every minute. This is sufficient to stop an attack like sneak shots but unable to stop a real-time attack like our BADUSB-C. In summary, the payment QR code is highly sensitive on users' devices and our following case study is about how to obtain this code using an attacker-crafted power bank via BADUSB-C.

2) Attack Scenario: In this part, we will introduce a real-life attack scenario to show that our BADUSB-C is a practical offensive tool. This scenario can be decomposed into the following steps.

I. The attacker rents a power bank from one of the power bank stations and replaces the internal components with BADUSB-C.
II. After the modification, an attacker-crafted power bank is returned to a rental station in a crowded area like an airport or railway station, which increases the probability of success.
III. A user borrows the modified power bank and connects it to her own device, becoming the victim of BADUSB-C.
IV. The attacker now has complete control over the victim and can perform various attacks using different modes.

Next, we summarize the possible threats toward user under different modes. First, under HID emulator mode, the attacker is able to implant malware and backdoor script into victims' devices. Moreover, using video capture mode, once the victims access their sensitive data like QR payment code or album, their sensitive data will be immediately transmitted via BADUSB-C to the attacker. Lastly, with full control mode, the attacker has complete control over the victims' device and can conduct any action on the victim device.

To further validate our attacking scenario, we invited 10 volunteers, who are students in our university, to participate and conduct a user study. Before the user study experiment, we disclose to the volunteers how their data might be used and request permission from the volunteers and the university ethics review boards. During the case study, volunteers took turns to use their phones for half an hour with BADUSB-C connected, which are considered as a normal power bank. To obtain data close to real-life, they are requested to use phones like when they use the normal shared power banks. After the case study experiment, we introduced to them our attack, analyze the screen recordings together to make sure their personal data is not at risk.

3) Result: After collecting the captured videos of the mobile phone, we analyzed the videos both automatically and manually. During the automatic analysis, we used scripts to perform OCR recognition for each frame in the recorded videos and stored all of the OCR results in a database. In total, there are $94,058$ entries of data collected among 10 volunteers. With these results, we could learn what content the victim visited, especially the privacy data. We searched the result database with the keywords *account, username, password, phone number and email*, as well as the regular expression of email. The records searched by such keywords and regular expression are usually related to privacy information of users. For example, when we searched with *account* as a keyword, victim's accounts can be found in the database, as shown in the Table III and Table IV. The frame number is the position of this frame in the recorded video, which indicates a target for manual analysis for further data extraction. At last, we found 4,172 of 94,058 records obtained by BADUSB-C which are related to privacy information, as shown in the Table IV.

In the manual analysis, we replayed the videos and extracted sensitive information. Accounts of internet applications such as Apple, iCloud, Facebook, Twitter can be obtained. Moreover, all of the typing inputs on the virtual keyboard, including the system keyboard and the built-in security keyboards of the financial applications, can be clearly recorded. We can obtain the plain text of passwords such as WiFi passwords. Furthermore, the received SMS verification code (usually used to confirm real-name authentication) can be obtained when it appears in the top notification bar.

In summary, though we cannot directly obtain the user's password on the lock screen, we can still check all of the information presented on the screen, extract sensitive information including but not limited to social accounts, bank accounts, personal financial situation, etc., if the user unconsciously unlocks the screen. It is worth mentioning that secure keyboards built in financial apps show their keyboard typing sequences, which can be easily captured by BADUSB-C.

VI. COUNTERMEASURES

Authorization mechanisms like GoodUSB [4] have been proposed as countermeasures against BadUSB attacks. As mentioned in Section IV and Section V, GoodUSB relies on internal display for authorization, which can be bypassed via our BADUSB-C. Next, we discuss our recommended countermeasures against BADUSB-C.

External Hardware Authorization. One possible countermeasure is to introduce external hardware completing the authorization process. Contrary to the GoodUSB, USBCheckIn [57] adopts a dedicated hardware between the host and device. When a device is plugged-in, the authorization will be conducted on the dedicated hardware instead of the internal display, preventing the host from being hijacked. Though USBCheckIn is an adequate defense against BADUSB-C, the external hardware brings additional cost and inconvenience, especially for mobile devices.

Distrust-by-Default. Most security issues of USB protocol are due to its *trust-by-default* assumption; BADUSB-C also relies on this feature to work. To defend against BADUSB-C and other USB-based attacks, we can simply reject all unauthorized device – applying the *distrust-by-default* policy. It is worth mentioning that *distrust-by-default* policy is not the same as GoodUSB [4]. GoodUSB permits limited functionality of untrusted devices while this strict policy forbids all functions of an untrusted device. Though *distrust-by-default* policy effectively prevents these attacks, it also causes considerable inconvenience for users.

Isolated UI Rendering. During our experiments, we noticed that BADUSB-C is actually unable to redirect out the locking screen keyboard from the iPad OS. Instead, the keyboard is only available on the internal display. However, this defense is only enabled on the locking screen keyboard, other keyboards (e.g., the virtual ones used by the apps) are still vulnerable to our BADUSB-C. This mechanism has inspired us to propose a new defense against our BADUSB-C called *Isolated UI Rendering*. As illustrated in Figure 7, we designed two separated UI render layers and corresponding drivers, one is secure the other is insecure. When an application requires to render, it can pass the content along with a tag identifying whether the content is "sensitive" or not. If the content is tagged with "sensitive", then the OS will only render it on the secure layer, which only shows the sensitive content (e.g., a keyboard) on the trusted screen. For the rest of the rendering, it would render on both the trusted and untrusted display (e.g., insensitive contents). For example, in Figure 7, the 'password' and 'keyboard' are recognized as sensitive while other parts of content are in sensitive. Thus, the 'password' and 'keyboard' are not rendered on untrusted screen.

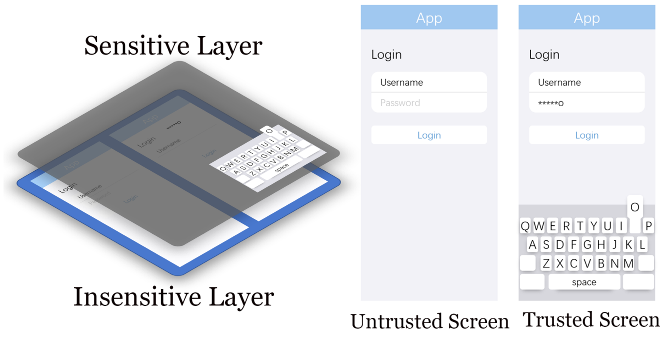


Fig. 7: Isolated UI Rendering

VII. DISCUSSION

A. Limitation

There exist multiple limitations of BADUSB-C. To begin with, BADUSB-C can only gain the information and control access of the host itself instead of external hardware. Consequently, as we introduced in the Section VI, BADUSB-C can hardly bypass the defense approaches that use external hardware for authorization. Moreover, most of the devices will prompt users to give authentication to the USB devices or select one of the functional modes after they are plugged in. Though some of such prompts are not conspicuous for non-experts, especially when BADUSB-C is concealed within other functional hardware such as power banks , the probability of whether users could get aware that something unusual happens will increase with the existence of these prompting messages.

B. Impact

The USB protocol is used widely as introduced in the preceding sections. As the technologies develop rapidly, more and more devices will be equipped with USB-C capabilities, which makes BADUSB-C more influential. Due to the neglect to check the security of plugged-in USB devices of non-professional users, BADUSB-C can be hardly detected by them while attacks are performing. The popularity and universality of public USB devices, including sharing power banks, even increase such risks. Moreover, BADUSB-C provides a better way for traditional BadUSB attacks, since attackers can obtain the screen streaming with ease. Attackers can use such technologies to perform more precise attacks, such as interacting with the user interface, as well as control the consequences of their attacks. In summary, BADUSB-C can be applied in various application scenarios and brings rather huge impacts.

VIII. CONCLUSION

Leveraging the new features of USB 3.x [10]–[12], we explore a new attack scheme named BADUSB-C and three attack modes. Each of these three modes has its own strength and use scenarios. The HID Emulation mode largely extends the original BadUSB and successfully bypasses existing defenses. The video capture mode proved practical and powerful in the user study, as it extracts victim's sensitive information in a stealthy way. The full control mode archives the complete hijack of the target device, allowing us to perform various types of subsequent attacks. By experimenting BADUSB-C with mobile phones, tablets, and PCs, we further test its ability under different modes and shows it can bypass defenses such as GoodUSB [4]. In our user study, we obtain and analyze video stream extracted from 10 participants with BADUSB-C, which demonstrates its capability in a real-life scenario. In the end, we propose a new defense scheme named *isolated UI rendering*, which can effectively stop our BADUSB-C.

As for future work, we will further explore the potential of *isolated UI rendering* and implement it on a customized operating system. Moreover, we also hope to lower the power consumption for network transmission in BADUSB-C to make it more practical.

IX. Responsive Disclosure

Responsive disclosure have already been carried out, we have already contacted the affected vendors through proper channel and are waiting for mitigation being deployed.

References

[1] "Common vulnerabilities and exposures," 2020. [Online]. Available: https://cve.mitre.org/

[2] J. D. Tian, N. Scaife, D. Kumar, M. Bailey, A. Bates, and K. R. B. Butler, "Sok: "plug & pray" today - understanding USB insecurity in versions 1 through C," in *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 2018, pp. 1032–1047. [Online]. Available: https://doi.org/10.1109/SP.2018.00037

[3] K. Nohl and J. Lell, "Badusb-on accessories that turn evil," *Black Hat USA*, vol. 1, no. 9, pp. 1–22, 2014.

[4] J. D. Tian, A. Bates, and K. R. B. Butler, "Defending against malicious USB firmware with goodusb," in *Proceedings of the 31st Annual Computer Security Applications Conference, Los Angeles, CA, USA, December 7-11, 2015*. ACM, 2015, pp. 261–270. [Online]. Available: https://doi.org/10.1145/2818000.2818040

[5] Compaq, D. E. Corporation, I. P. Company, Intel, Microsoft, NEC, and N. Telecom., Universal Serial Bus Specification, Revision 1.0, January 1996.

[6] USB-IF, "Device class definition for human interface devices (HID)," 2001.

[7] Compaq, D. E. Corporation, I. P. Company, Intel, Microsoft, NEC, and N. Telecom., Universal Serial Bus Specification, Revision 1.1, September 1998.

[8] ——, Universal Serial Bus Specification, Revision 2.0, April 2000.

[9] Hak5, "Episode 709: Usb rubber ducky part 1," http://hak5.org/episodes/episode-709, 2013.

[10] I. HP *et al.*, "Universal serial bus 3.0 specification," 2008.

[11] ——, "Universal serial bus 3.1 specification," 2013.

[12] I. M. R. S. Apple, Hewlett-Packard and T. Instruments., "Universal serial bus 3.2 specification," 2017.

[13] G. Chandler *et al.*, "Universal serial bus type-c cable and connector specification," *USB Type-C Cable and Connector Specification, USB*, vol. 3, pp. 1–171.

[14] "USB implementers forum," 1995. [Online]. Available: https://www.usb.org/about/

[15] USB-IF, "USB-IF statement regarding USB security," 2014.

[16] P. Szor, "Duqu–threat research and analysis," *McAfee Labs*, 2011.

[17] W. Meng, L. W. Hao, M. S. Ramanujam, and S. P. T. Krishnan, "Charging me and I know your secrets!: Towards juice filming attacks on smartphones," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security, CPSS 2015, Singapore, Republic of Singapore, April 14 - March 14, 2015*, J. Zhou and D. Jones, Eds. ACM, 2015, pp. 89–98. [Online]. Available: https://doi.org/10.1145/2732198.2732205

[18] W. Meng, L. W. Hao, Z. Liu, C. Su, and Y. Li, "Evaluating the impact of juice filming charging attack in practical environments," in *Information Security and Cryptology - ICISC 2017 - 20th International Conference, Seoul, South Korea, November 29 - December 1, 2017, Revised Selected Papers*, ser. Lecture Notes in Computer Science, H. Kim and D. Kim, Eds., vol. 10779. Springer, 2017, pp. 327–338. [Online]. Available: https://doi.org/10.1007/978-3-319-78556-1_18

[19] GoodFET, "Facedancer21." http://goodfet.sourceforge.net/hardware/facedancer21/, 2016.

[20] Google, "Found linux kernel usb bugs.." https://github.com/google/syzkaller/blob/master/docs/linux/found bugs usb.md, 2017.

[21] A. D. Ramadhanty, A. Budiono, and A. Almaarif, "Implementation and analysis of keyboard injection attack using usb devices in windows operating system," in *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*, 2020, pp. 449–454.

[22] J. Bang, B. Yoo, and S. Lee, "Secure usb bypassing tool," *digital investigation*, vol. 7, pp. S114–S120, 2010.

[23] M. Brocker and S. Checkoway, "iseeyou: Disabling the macbook webcam indicator LED," in *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*, K. Fu and J. Jung, Eds. USENIX Association, 2014, pp. 337–352. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/brocker

[24] S. Kamkar, "Usbdriveby." http://samy.pl/usbdriveby/, 2014.

[25] K. Zetter, "Meet 'flame,'the massive spy malware infiltrating iranian computers," *Wired Magazine*, 2012.

[26] H. J. Highland, "The brain virus: fact and fantasy," *Computers & Security*, vol. 7, no. 4, pp. 367–370, 1988.

[27] "Common vulnerabilities and exposures," 2010. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-2568,

[28] S. Shin and G. Gu, "Conficker and beyond: a large-scale empirical study," in *Twenty-Sixth Annual Computer Security Applications Conference, ACSAC 2010, Austin, Texas, USA, 6-10 December 2010*, C. Gates, M. Franz, and J. P. McDermott, Eds. ACM, 2010, pp. 151–160. [Online]. Available: https://doi.org/10.1145/1920261.1920285

[29] Z. Wang and A. Stavrou, "Exploiting smart-phone USB connectivity for fun and profit," in *Twenty-Sixth Annual Computer Security Applications Conference, ACSAC 2010, Austin, Texas, USA, 6-10 December 2010*, C. Gates, M. Franz, and J. P. McDermott, Eds. ACM, 2010, pp. 357–366. [Online]. Available: https://doi.org/10.1145/1920261.1920314

[30] ——, "Exploiting smart-phone USB connectivity for fun and profit," in *Twenty-Sixth Annual Computer Security Applications Conference, ACSAC 2010, Austin, Texas, USA, 6-10 December 2010*, C. Gates, M. Franz, and J. P. McDermott, Eds. ACM, 2010, pp. 357–366. [Online]. Available: https://doi.org/10.1145/1920261.1920314

[31] K. Sridhar, S. Prasad, L. Punitha, and S. Karunakaran, "Emi issues of universal serial bus and solutions," in *8th International Conference on Electromagnetic Interference and Compatibility*. IEEE, 2003, pp. 97–100.

[32] A. Davis, "Revealing embedded fingerprints: Deriving intelligence from usb stack interactions," *Blackhat USA*, 2013.

[33] Y. Su, D. Genkin, D. C. Ranasinghe, and Y. Yarom, "USB snooping made easy: Crosstalk leakage attacks on USB hubs," in *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, E. Kirda and T. Ristenpart, Eds. USENIX Association, 2017, pp. 1145–1161. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/su

[34] mich., "Inside a low budget consumer hardware espionage implant."

[35] A. Bates, R. Leonard, H. Pruse, D. Lowd, and K. R. B. Butler, "Leveraging USB to establish host identity using commodity devices," in *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014. [Online]. Available: https://www.ndss-symposium.org/ndss2014/leveraging-usb-establish-host-identity-using-commodity-devices

[36] K. Nohl., "Badusb exposure: Hubs." https://opensource.srlabs.de/projects/badusb/wiki/Hubs, November 2014.

[37] L. Letaw, J. Pletcher, and K. R. B. Butler, "Host identification via USB fingerprinting," in *2011 IEEE Sixth International Workshop on Systematic Approaches to Digital Forensic Engineering, SADFE 2011, Oakland, CA, USA, May 26, 2011*, R. F. Erbacher, R. H. Campbell, and Y. Guan, Eds. IEEE Computer Society, 2011, pp. 1–9. [Online]. Available: https://doi.org/10.1109/SADFE.2011.9

[38] D. F. Oswald, B. Richter, and C. Paar, "Side-channel attacks on the yubikey 2 one-time password generator," in *Research in Attacks, Intrusions, and Defenses - 16th International Symposium, RAID 2013, Rodney Bay, St. Lucia, October 23-25, 2013. Proceedings*, ser. Lecture Notes in Computer Science, S. J. Stolfo, A. Stavrou, and C. V. Wright, Eds., vol. 8145. Springer, 2013, pp. 204–222. [Online]. Available: https://doi.org/10.1007/978-3-642-41284-4_11

[39] M. Guri, M. Monitz, and Y. Elovici, "Usbee: Air-gap covert-channel via electromagnetic emission from USB," in *14th Annual Conference on Privacy, Security and Trust, PST 2016, Auckland, New Zealand, December 12-14, 2016*. IEEE, 2016, pp. 264–268. [Online]. Available: https://doi.org/10.1109/PST.2016.7906972

[40] TURNIPSCHOOL, "Nsa playset." http://www.nsaplayset.org/ turnip-school.

[41] D. Wagenaar, D. Pavlov, and S. Yannick, "Usb baiting," *Universite van Amserdam*, 2011.

[42] B. Leung, "Surjtech's 3m usb a-to-c cable completely violates the usb spec. seriously damaged my laptop." https://www.amazon. com/re-view/R2XDBFUD9CTN2R/ref=cm cr rdp perm, 2016.

[43] INT3.CC, "The original usb condom," 2018, https://int3.cc/collections/usb/products/usbcondoms.

[44] Microsoft, "How to control usb devices and other removable media using microsoft defender for endpoint," 2018, https://docs.microsoft.com/en-us/windows/security/threat-protection/device-control/control-usb-devices-using-intune#prevent-installation-of-specifically-prohibited-peripherals.

[45] H. Mohammadmoradi and O. Gnawali, "Making whitelisting-based defense work against badusb," in *Proceedings of the 2nd International Conference on Smart Digital Environment, ICSDE 2018, Rabat, Morocco, October 18-20, 2018*, F. E. Bouanani and A. Habbani, Eds. ACM, 2018, pp. 127–134. [Online]. Available: https://doi.org/10.1145/3289100.3289121

[46] B. Yang, Y. Qin, Y. Zhang, W. Wang, and D. Feng, "TMSUI: A trust management scheme of USB storage devices for industrial control systems," in *Information and Communications Security - 17th International Conference, ICICS 2015, Beijing, China, December 9-11, 2015, Revised Selected Papers*, ser. Lecture Notes in Computer Science, S. Qing, E. Okamoto, K. Kim, and D. Liu, Eds., vol. 9543. Springer, 2015, pp. 152–168. [Online]. Available: https://doi.org/10.1007/978-3-319-29814-6_13

[47] S. Neuner, A. G. Voyiatzis, S. Fotopoulos, C. Mulliner, and E. R. Weippl, "Usblock: Blocking usb-based keypress injection attacks," in *Data and Applications Security and Privacy XXXII - 32nd Annual IFIP WG 11.3 Conference, DBSec 2018, Bergamo, Italy, July 16-18, 2018, Proceedings*, ser. Lecture Notes in Computer Science, F. Kerschbaum and S. Paraboschi, Eds., vol. 10980. Springer, 2018, pp. 278–295. [Online]. Available: https://doi.org/10.1007/978-3-319-95729-6_18

[48] D. V. Pham, M. N. Halgamuge, A. Syed, and P. Mendis, "Optimizing windows security features to block malware and hack tools on usb storage devices," in *Progress in electromagnetics research symposium*, 2010.

[49] W. Meng, L. Jiang, Y. Wang, J. Li, J. Zhang, and Y. Xiang, "Jfcguard: Detecting juice filming charging attack via processor usage analysis on smartphones," *Computers & Security*, vol. 76, pp. 252 – 264, 2018.

[50] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: using hard AI problems for security," in *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, ser. Lecture Notes in Computer Science, E. Biham, Ed., vol. 2656. Springer, 2003, pp. 294–311. [Online]. Available: https://doi.org/10.1007/3-540-39200-9_18

[51] R. P. Foundation, "Raspberry pi 4B," 2019, https://www.raspberrypi.org/products/raspberry-pi-4-model-b.

[52] M. T. Incorporated, "ATmega32u4 chip," 2016, https://www.microchip.com/wwwproducts/en/ATmega32u4.

[53] U. G. Limited, "UGREEN company introduction," 2012, https://www.ugreen.com/pages/about-ugreen.

[54] Brick, "Brick," 2019, https://brickapp.se/about-us/.

[55] "WeChat Pay," 2020. [Online]. Available: https://pay.weixin.qq.com/index.php/public/wechatpay_en

[56] Alipay, "Alipay," 2020, https://global.alipay.com/platform/site/ihome.

[57] F. Griscioli, M. Pizzonia, and M. Sacchetti, "Usbcheckin: Preventing badusb attacks by forcing human-device interaction," in *14th Annual Conference on Privacy, Security and Trust, PST 2016, Auckland, New Zealand, December 12-14, 2016*. IEEE, 2016, pp. 493–496. [Online]. Available: https://doi.org/10.1109/PST.2016.7907004