BADUSB-C: Revisiting BadUSB with Type-C

Anonymous Authors

***Abstract*—The security of the Universal Serial Bus (USB) protocol has been paid extensive attention to because of its wide usage. Due to the *trust-by-default* characteristics, USB security has caused severe problems. For example, a well-known firmware attack, BadUSB, performs malicious operations on the victim hosts through disguising ordinary USB devices as human interface devices like keyboards and mice. However, BadUSB suffers from several limitations. Attackers cannot obtain the status of User Interface (UI) to conduct precise attacks and get the visual feedback of their attacks. In this work, we extended BadUSB to support the new USB Type-C features and proposed a multi-mode attack model, BADUSB-C. This obtains UI status to make attacks more precise and effective. 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 to simulate daily usage and summarized the private information collected. 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 worldwide since its appearance in 1996, as it provides a unified and easy-to-use approach for ~~a~~ an extensive 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.

Conversely, the security of USB has caused severe problems. Recent research of all USB specifications indicates that security has not been taken into consideration. [1] There are more than 400 vulnerabilities related to USB on CVE list [2]. 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 [1].

BadUSB is a well-known class of firmware attacks [3]. These attacks are conducted by modifying the device firmware, which are disguised as ordinary USB devices as other types of devices that are *trust-by-default* by the hosts. Typically, simulated devices include Human Interface Device (HID) [4] and disks. Utilizing BadUSB, attackers can pretend to be regular users, typing malicious commands to victims' computers, downloading and executing malicious scripts, and copying out private data from disks. Such attacks can easily avoid detection by traditional anti-virus software since it is hard to distinguish them from ordinary 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 cannot obtain the current User Interface (UI) status, limiting them from taking subsequent moves. For example, it is hard for attackers to locate specific functional UI patterns such as buttons and links on victim computers by disguising USB devices, e.g., mice. This 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 host network usage. (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 more extensive range of peripherals such as DisplayPort, HDMI and PowerDelivery. BadUSB attacks can become more effective with the help of newly supported features in USB 3.x. (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 [5]. It provides a Graphical User Interface (GUI) for users to describe the functionalities or roles of the USB device and reject any usage beyond the description.

In this work, we addressed 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. Although many smartphones equipped with USB-C connectors do not support USB 3.x protocol, such as some non-flagship products of Xiaomi, more and more vendors like HUAWEI and Samsung tend to support USB 3.x protocol in their high-end smartphones [**?**]. Since USB Type-C can transfer video stream data, BADUSB-C could obtain the information of the victim's GUI during attacks. Combining it with the emulation of traditional HIDs, e.g., keyboards and mice, attackers are capable of performing precise attacks.

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~~ Fully 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~~ 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~~ case 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 , since users are usually unconscious of the necessity to check~~

~~the devices they plugged in for security purposes.~~ 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 the insensitive layer's content ~~will~~ would be passed to the insecure driver and thus rendered on the external display, protecting the sensitive layer's content.

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.
- We conducted ~~multiple experiments and a user study~~ a case study and multiple experiments 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 works 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 **??**.

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 [6] 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 Human Interface Device (HID) [4] was designed based on USB. HID is designed to unify the implementation for devices like mice, keyboards, etc. Before its appearance, the standard is divided ~~between~~ among 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, the HID standard also requires all devices to be Plug-and-Play (PnP), which is indeed convenient but insecure too.

In 1998, the first widely supported USB protocol was designed. USB 1.1 [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~~limitation, it only supports limited types of devices like keyboards, mice, etc.

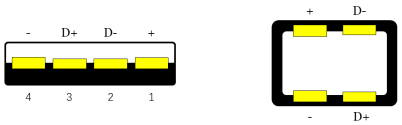


Fig. 1: USB 1.x & 2.x Connector.

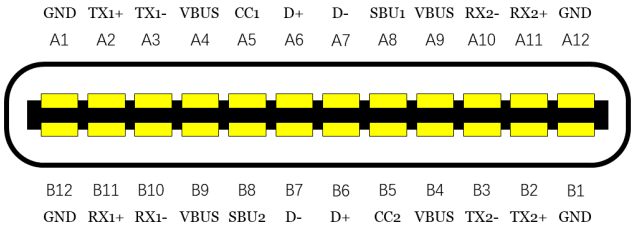


Fig. 2: USB Type-C Connector.

In 2000, the 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 [1]. Although 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 introduced 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, 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-purpose connector unprotected is insecure and allows attacks similar to 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 peripherals like DisplayPort (10.8 Gbit/s Min). Apart from that, support for other peripherals also requires dedicated transferring lanes 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] was 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 the USB specification, security was insufficiently considered [1]. 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 surveyed 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 Denial of Service (DoS) 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 act 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 are using the victim's device. 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].

Mobile High-Definition Link (MHL) extends the USB standard and allows the video signal to be transmitted through the USB interface even for USB 2.0 protocol. Juice Filming Attacks (JFA) [17], abuses this standard and monitoring the victim's device utilizing USB 2.0 protocol. Although JFA [17] could exfiltrates video data from the victim's device without permission, there are two main limitation compared with BADUSB-C: ❶ JFA [17] can only exfiltrate video data passively from the victim's device, while BADUSB-C can control the victim's device at the same time as monitoring the screen. All data can be collected by JFA only after the victim's interaction, while BADUSB-C can gather sensitive data actively through HID injection. For example, if a victim leaves his/her phone charging without locking it, JFA cannot obtain anything in this case, while BADUSB-C can actively control his/her device and gather sensitive information. ❷ The official list of MHL Devices [25] shows that the latest mobile devices supporting MHL were released in 2015, which indicates the lack of support of MHL and limits the attack range of JFA. BADUSB-C attacks over USB Type-C interface, the latest USB connector designed to replace all previous USB connector including MicroUSB [26]. So BADUSB-C has a wider attack range.

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, Flame [27] uses a zero-day exploit and malicious *autorun.inf* to execute the malware automatically. There are also works [**?**], [**?**], [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 zero-day 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] eavesdrop on the leaked signals and recover the sensitive data. In a similar fashion, USBee [**?**] and TURNIPSCHOOL [**?**] emit electromagnetic emissions by data injection on the bus with the connected USB devices as an RF transmitter and USBKiller [**?**] injects 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 [11] was introduced with USB Type-C in 2013, DisplayPort 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 [1].

From the hardware perspective, the BadUSB attack requires (D+ & D-) pins which are defined by the protocol to transmit data. Without these pins, data cannot be transferred via a USB cable. Based on this fact, USB Condom [**?**] is a hardware solution to block data channels by adding a blocker in the connector. This blocker can cut off the (D+ & D-) connection while leaving the power pins intact. However, this method poses a great challenge to the PnP 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 while establishing connection. Windows Defender ATP [**?**] 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

Year	Protocol Version	Supported Peripherals	Transfer Speed	Attacks
1996	USB 1.x [6], [7]	Keyboard, Mouse...	1.5 Mbit/s or 12 Mbit/s	HID Emulation (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.

whitelist without precaution. Fortunately, some designs have been proposed to overcome this drawback. For instance, Mohammadmoradi *et al.* [**?**] 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 [**?**] relies on the rich experience of administrators to build a whitelist. However, some modified USB devices may hide their real functionality from the user.

To solve this flaw, GoodUSB [5] reports the functionality claimed by the device to the user and lets the user decide whether to authorize. When a device is plugged in, GoodUSB blocks its functionality before enumeration until a series of authorizations are completed. These authorizations are designed to be performed manually, thus the malicious devices will be detected if they make a different claim that does not match the authorization results. USBeSafe [**?**] analyses the feature vector constructed with the data collected from USB Request Blocks, detects the novel USB packets based on the machine learning algorithm and sends an alert to the user before enumeration. As BadUSB attacks normally request privileged interfaces during enumerating, these defenses are sufficient for these attacks. Moreover, Mueller *et al.* [**?**] improve the security without changing user experience. It can detect the presence of the user, block malicious devices in the user's absence until the session is unlocked.

After the USB enumeration and driver loading, some related works follow the *defense in depth* approach to defend against BadUSB attacks. Neuner *et al.* [**?**] prevent BadUSB attacks from the malicious flash drive by analyzing the temporal characteristics of BadUSB-like attacks. This defense mechanism is effective because the attacker cannot obtain the screen of the victim 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 property causes the typing characteristics of BadUSB to be detectable. Pham *et al.* [**?**] 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 JFA [17] which abuses the MHL standard to steal the video stream from the victim. In order to mitigate this issue, Meng et al. [**?**] propose a statistical model using status like GPU/CPU usage to detect JFA attacks.

Therefore, there exists a clear trade-off between the effectiveness and the PnP property. Although hardware disabling solutions like USB Condom achieves almost absolute security, the functionality of USB is sacrificed. Other solutions like 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] 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 a basic assumption that common users without technical background would not treat a normal-looking USB device as malicious and be cautious about them. This assumption is consistent with existing works [18]. Moreover, we omit the effect of notifications from USB devices, as users may not have the knowledge to fully understand such notifications.

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 fulfilled 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 the USB protocol to perform attacks. However, these attacks suffer from various drawbacks: ❶ attackers can only simulate limited types of devices such as HIDs (e.g. keyboards and mice) and disks, which makes the attacks less effective; ❷ accurate attacks could not be performed due to a lack of UI status. Whatever HIDs 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. For example, GoodUSB [5] im-

Defense \ Attack	Facedancer [19], Syzkaller [20]	[3], [9], [21]–[24]	JFC [17]	Duqu [16], [?], [?], [27], [28]	BADUSB-C
USB condom [?]	●	●	●	●	●
Windows Defender ATP [?], Mohammadmoradi *et al.* [?], TMSUI [?]	○	◐	◐	◐	◐
GoodUSB [5]	◐	●	●	●	●
USBeSafe [?]	○	●	●	●	●
Mueller *et al.* [?]	○	●	●	◐	●
Neuner *et al.* [?]	○	●	○	○	○
Pham *et al.* [?]	○	○	○	●	○
JFCGuard [?]	○	○	◐	○	◐

● 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

plemented an authorization procedure when a new device is plugged in and blocks other functions except authorized ones.

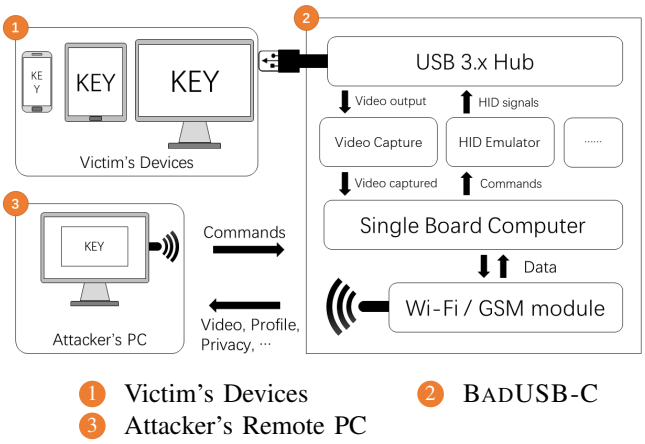
In this work, we utilize the new features of USB 3.x [11], [12] to address the problems above. Benefiting from the latest protocol, we simulate an external display and thus obtain the video stream to perform accurate attacks. However, we are still unable to bypass defenses like GoodUSB [5], which completely blocks unauthorized functionality. This will be further discussed in Section VII

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 by the attacker.
- Single Board Computer processes the 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 mainly relies on the "HID Emulator" in Figure 3 to send constructed HID ~~packet~~ packets to the victim. These constructed HID packets are interpreted by the victim as valid keystrokes and mouse movements. Thus, ~~the attacker~~ attackers can execute arbitrary scripts on the victim's device. This is the function of ~~the~~



① Victim's Devices ② BADUSB-C
③ Attacker's Remote PC

Fig. 3: Attack Model.

~~original~~ BadUSB. Based on this, we made the following improvements.

First, as BADUSB-C has established a feedback channel using USB 3.x protocol, when performing keystroke and mouse ~~movement~~ movements injection, an attacker can obtain real-time feedback from the victim's device. This feedback allows an attacker to tell whether the previous attack succeeded and decide what to perform next. Existence of such a feedback channel largely strengthens the attack ability of BADUSB-C.

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 fully-functional mouse emulation. This function enables attacks toward pure GUI programs and shows potential in the mobile attack scenarios. Details can be obtained in Section V.

The advantage of this mode is that it achieves attack feedback with video streaming. Also, 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 the "Video Capture" component as shown in the Figure 3 to transmit the stream to the embedded "Single Board Computer". The victim's device mistakenly treats BADUSB-C as an external monitor and output its video stream. This stream is later processed by the embedded computer to extract sensitive data.

When running in this mode, BADUSB-C passively processes the victim's video stream and detects "valuable" private data. The data is considered as "valuable" or not by a customized detector. We implemented a simple payment code detector for the proof-of-concept purpose, which demonstrates that we successfully transfered 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 detectors 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 does the victim's device treat BADUSB-C as an external display, but also a valid HID input source. Hence we can achieve complete hijack of the victim's device.

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

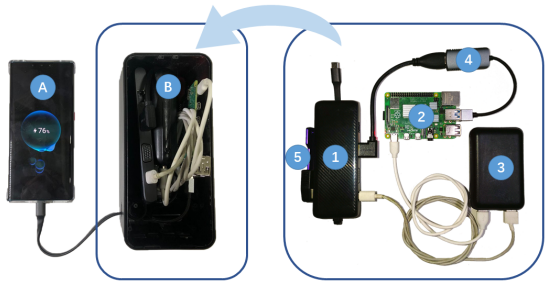
This mode enables attackers to perform delicate operations that are beyond automation. Moreover, this mode provides a backdoor that does not require a host network and thus is undetectable by the firewall running on the host machine.

The advantage of this mode is that it can completely hijack the victim's device and provides 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 chose the following parts to build a prototype. To begin with, we chose the Raspberry Pi 4B [**?**] 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 used an Atmel ATMEGA32U4 board [**?**] with USB protocol



A Victim's Device
B 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.

support, which is able to emulate multiple HIDs with our modified firmware. About the USB 3.x Hub, we used one from UGREEN [**?**], which supports HDMI, USB 2.0, and many other exported peripherals. Apart from these essential parts, we also used 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.

A is a HUAWEI mobile phone, the victim's device; B 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.

A. Attack Initialization

After the BADUSB-C is plugged into the victim's device, there is an initialization process to make sure the BADUSB-C is able to carry out subsequent attacks.

- **Screen Mirror**: During our experiment, we noticed that some devices did not mirror their screens to our BADUSB-C by default. In this case, BADUSB-C would inject a sequence of keystrokes to set the victim's device ~~into mirror their screens~~to mirror the primary screen. In both Windows 10 and Ubuntu (Gnome Desktop), BADUSB-C can inject ~~"~~"Win+P"~~ ("Super" is another name for the "Win" key "~~ (The "Win" key is the "Super" key in Windows and Ubuntu by default) to switch between different modes for external display. Figure 5 illustrates how these keystrokes works on Ubuntu and ~~Windows 10 is similar .~~ it is similar in Windows 10. In MacOS, according to the official manual [**?**], BADUSB-C can inject ~~"~~"Command+F1~~"~~" keystrokes to set the ~~monitor into mirroring~~ victim's device to mirror the primary screen. In EMUI, there is a special mode called ~~"Desktop Mode"~~"Desktop Mode", which allows users to use their smartphones like a desktop computer with an external screen. If this mode is enabled, as victim's ~~default~~

~~setting~~"desktop" can be obtained, BADUSB-C can inject mouse movements to switch to mirror mode~~as can obtain victim's 'desktop' in this mode~~.

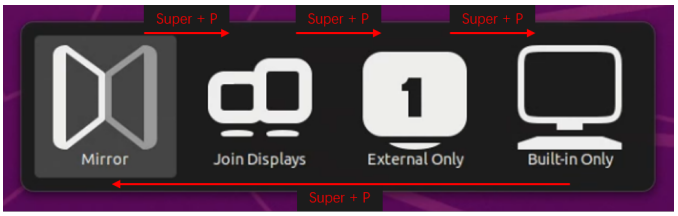


Fig. 5: Switching Screen Mode ~~Switch~~ in Ubuntu.

- **Dismiss Notification**: In our experiment, we found that some devices ~~will~~ would notify the user the presence of an external screen. During our experiment, the following notifications were raised:
  - On Lenovo Xiaoxin Pro 13, it raised a pop-up asking user to select the functionality of the external screen.
  - On HUAWEI P30, it showed a status bar indicator and a persistent notification about the external monitor.
  - On iPad Pro (the third generation), it also showed a blue status bar indicator and a notification about USB accessories if iPad is locked.

  To avoid being discovered by the victim, BADUSB-C can inject keystrokes and mouse movements to dismiss some of these notifications. For example, in the first case, BADUSB-C is able to inject a sequence of mouse movements to set the screen at mirror mode to dismiss the pop-up.

B. HID Emulator Mode

In the experiment of HID emulator mode, we used Lenovo Xiaoxin Pro 13 2020, a PC with Windows 10 (OS Build 18363.1379) and Ubuntu 20.04.2 LTS with two USB Type-C interfaces as the target ~~device~~devices. During this experiment, BADUSB-C disguised itself as a normal keyboard and an external screen as feedback channel. When first plugged in, the victim's device raised a pop-up window asking the victim which mode should the new screen be set to. BADUSB-C immediately injected a sequence of mouse movements and clicks to set itself as a mirror to the primary monitor. Thus we had successfully completed the attack initialization. Then we tested three scripts, ranging from reverse shell backdoor to malware payload execution, all of ~~which~~ them resulted in success. Compared to original BadUSB attacks like Rubber Ducky, our BADUSB-C can provide attackers real-time feedback which allows attackers to perform more accurate attacks.

C. Video Capture Mode

During the experiment of privacy extraction via our video capture mode, we chose *HUAWEI P30 (ELE-AL00)*, a smartphone in EMUI 11.0.0.135(C00E128R2P5) (Android 10.0 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 successfully extracted text such as 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 performs transactions without the password. During our experiment, we noticed that this HUAWEI device supports 'Desktop Mode' for its external screen, which enables user to use their devices like a desktop computer with a external screen. If BADUSB-C is set into this mode, BADUSB-C is able to inject mouse movements to set itself into mirroring the primary screen, as we discussed in Section V-A. As this is also a part of our case study, more details about the extracted sensitive data can be found in Section V-E and Table **??**.

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 useful when the network connection between BADUSB-C and the attacker is not stable.

D. Fully Control Mode

To test the capability of the full control mode, we chose iPad Pro (3rd generation), a tablet running iOS 14.4 (Build 18D52) with a USB Type-C interface, as the target device in this experiment. Besides disguising themself as normal HIDs like a conventional BadUSB [3], BADUSB-C also transmitted real-time video stream from the target device to the attacker via WiFi. After establishing the connection, 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 the 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 achieved the complete hijack of the victim's device in this experiment.

E. Case study

BADUSB-C can be used in various attack scenarios, ranging from mobile devices to PC devices. For example, BADUSB-C can be attached to the power station, which provides USB 3.x hubs, in the airport to perform attacks. Most people charge their laptops or smartphones in the power station in emergency, with negligence of security. In the following paragraphs, we will demonstrate the attack scenarios of BADUSB-C using sharing power bank as an example.

Fig. 6: Two Power Bank Stations.

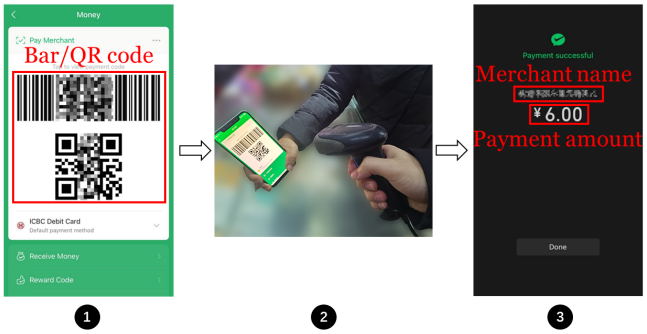


Fig. 7: Bar/QR Code Payment Procedure.

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

Sharing Power Banks. Sharing power banks provide 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 6 are photos of two power bank stations in China, which are taken outside of a supermarket. Brick [**?**] 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.

Sharing power banks not only provides convenience to users, but 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 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 [**?**] and Alipay [**?**]. 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 7, QR code payments are 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 [**?**] regards transactions under ~~USD~~ US$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 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 [**?**] and Alipay [**?**] 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 opportunistic theft of a payment code 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 broken down 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 his/her own device, becoming the victim of BADUSB-C.
IV. The attacker now has complete control over the victim's device and can perform various attacks using different modes.

Next, we summarize the possible threats toward users under different modes. First, under HID emulator mode, the attacker is able to implant malware and backdoor scripts into victims' devices. Moreover, using video capture mode, once the victims access their sensitive data such as QR payment code or photos,

their sensitive data ~~will~~ could 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~~devices and could conduct any actions on the victims' devices.

3) Experiment: ~~To further test~~ We conducted experiments to further validate the capabilities of BADUSB-C in ~~this attack scenario , we conducted a simulation experiment. We selected~~ the attack scenario introduced in Section V-E2. 11 applications were selected and tested on *HUAWEI P30 (ELE-AL00)* ~~, test each application with the following strategy~~ step by step: (1) Login in with a test account. (2) Keep the default settings. (3) Attach attacker's BADUSB-C to the test device. (4) Simulate victim's daily usage of the application.

During this experiment, we tested full control mode which actively obtained sensitive information and video capture mode which passively obtained the sensitive information visited by the victim. We noticed that, most sensitive information could be obtained directly through full control mode without victim's interaction, but there was also certain information that had to be obtained passively through video capture mode. For example, as illustrated in Table **??**, in most applications, information like "Account" can be obtained directly while information like "Payment Password" cannot be obtained until the victim inputs his/her password.

4) Result: In summary, BADUSB-C is able to actively obtain most sensitive information from the screen, such as browsing history, personal account, phone number without victim's interaction. There also exists information such as payment password that is not available immediately, but such a piece of information can be obtained once the victim inputs it. The obtained information can be ~~further~~ used to guess the victim's lock screen password and poses ~~even greater~~ further threat to victim's privacy.

VI. COUNTERMEASURES

Next, we discuss our recommended countermeasures against BADUSB-C.

External Hardware Authorization. One possible countermeasure is to introduce external hardware completing the authorization process. For example, USBCheckIn [**?**] 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. Although 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~~ devices – applying the *distrust-by-default* policy. For example, in GoodUSB [5], all USB devices are enabled with corresponding functionality only after being authorized by the user. Defense like USB Condom distrusts all types of USB devices by blocking data channels and stopping all USB functions other than charging. Such defense methods effectively stop attacks like BADUSB-C.

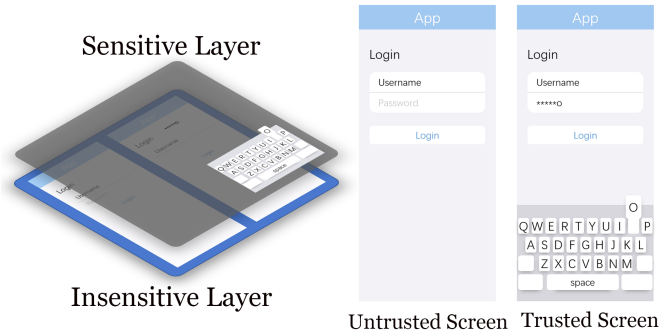


Fig. 8: Isolated UI Rendering

Isolated UI Rendering. During our experiments, we noticed that BADUSB-C is actually unable to mirror the lock screen keyboard from the iPad OS. Instead, the keyboard is only available on the internal display. However, this defense is only enabled on the lock 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 8, we designed two separated UI render layers and corresponding drivers, one is secure the ~~other~~ another 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~~ could 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 8, the "password" and "keyboard" are recognized as sensitive while other parts of content are insensitive. Thus, the "password" and "keyboard" are not rendered on untrusted screen.

VII. DISCUSSION

A. Limitation

There exist multiple limitations of BADUSB-C. To begin with, when BADUSB-C is attached, the victim's device may not be set to mirror the screen. Depending on the victim's settings, the BADUSB-C may be used as an extended monitor or be set to 'Desktop Mode' or not be enabled at all. Though we proposed the attack initialization in Section V-A to inject keystrokes to ensure BADUSB-C is mirroring primary screen. But without knowing the exact operating system of the victim's device, BADUSB-C may have to try multiple injections before successfully mirroring the screen. Moreover, as mentioned in Section V, BADUSB-C cannot directly obtain the information on the lock screen. Apart from that, we also noticed that in most devices, when an external monitor is connected to

Category	Application	Leaked Sensitive Information	
		Without Victim's Interaction	**With Victim's Interaction**
Social & Finance App	WeChat (8.0.1)	Account, Financial Status, Chat History, Payment Code	Payment Password
Social App	WhatsApp (2.21.5.18)	Account, Contacts, Chat History, Phone Number	
	Facebook (309.0.0.47.119)	Account, Posts, Contacts	
Finance App	Alipay (10.2.15.9500)	Account, Financial Status, Payment Code	Payment Password
	Cash App (3.35.1)	Email, Phone Number, Cash Balance	Payment Password
	Paypal (7.38.1)	Account, PayPal Balance	Payment Password
Shopping App	Amazon Shopping (22.6.0.100)	Account, Orders, Shopping Cart	
Tool	Chrome (89.0.4389.72)	Browsing History	
	Health (11.0.5.508)	Personal Health Metrics	
System	Messages (11.0.1.430)	Contacts, Chat History	
	Settings (11.0.0.300) - WiFi	WiFi SSID	WiFi Password
	Settings (11.0.0.300) - VPN	VPN Address, VPN Account, VPN Password	

TABLE III: Sensitive Information Leaked From Applications.

the device, there will be notifications about the monitor. In iPad, there is a small blue icon on the notification bar. In Windows 10, there is a pop-up for user to select the functionality of the external monitor. In latter case of a pop-up, BADUSB-C can dismiss it by injecting keystrokes as described in Section V-A. But they are still noticeable by the victim. Also, it cannot be ignored that DisplayPort over USB is not available on all devices. When selecting devices to test BADUSB-C, we find that many smartphones equipped with USB-C connectors actually do not support USB 3.x protocol. There is a incomplete list of devices that support DisplayPort over USB [**?**]. Most vendors like HUAWEI and Samsung tend to support USB 3.x protocol in their high-end smartphones. But there also exists vendor like Xiaomi who does not support USB 3.x protocol at all.

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 [26], which makes BADUSB-C more influential. Since non-professional users may neglect checking the security of plug-in USB devices, they may unaware of the attacks from 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 and controlling 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. The video capture mode proved practical and powerful, as it extracts victim's sensitive information in a stealthy way. The fully control mode achieves 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. In our ~~user study~~experiment, we obtain and analyze video stream extracted from ~~10 participants~~ 11 applications with BADUSB-C, which demonstrates its capability in a simulated real-life scenario. In the end, we propose a new defense scheme named *isolated UI rendering*, which can effectively stop ~~our~~ attacks with 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. RESPONSIBLE DISCLOSURE

Responsible disclosure have already been carried out, we have contacted HUAWEI and Apple through proper channels. We received response from HUAWEI on March 7th and discussed the mitigation plan on March 10th while Apple has not responded yet. HUAWEI security team has confirmed this issue and been working on mitigation. We are actively facilitating their mitigation process and waiting for a fix to be deployed. After the mitigation is deployed, HUAWEI will assign a CVE ID for this vulnerability.

X. ACKNOWLEDGEMENT

We are very grateful to our shepherd, Michael Schwarz, and the anonymous reviewers for their valuable ~~comments~~ feedback that improved the paper.