Is It Possible to Steal Money Through USB-C?

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

4th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

5th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

6th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—Abstract [1].**

***Index Terms*—BadUSB**

A. Motivation

As we introduced in the Section III, there are plenty of existing work ✎ [shuqing: Add citation] focusing on BadUSB attack. Many of these take use of the *default trust* given by computers to HIDs, pretend normal USB devices as HIDs and uitilize USB protocol to perform attacks. However, these attacks suffer from various drawbacks: ❶ attackers can only simulate limited types of HIDs like keyboards, mice, and disks, which makes the attacks less effective; ❷ accurate attacks couldn't be performed due to lacking of user interface (UI). Whatever HID the attackers simulate their USB devices to be, they couldn't obtain the UI to check the current situation, which makes it nearly impossible to locate their attacks precisely or confirm the effects after attacks.

In this work, we utilized the new features of USB-C to address the problems above. Benefiting from USB-C, we simulated vedio card and thus obtained the video stream of UI to perform accurate attacks.

B. Implementation

The implementation of BadUSB-C extends several existing offensive BadUSB device including Rubber Ducky (where we implement the co-operation of HID emulation and screen streaming) and Raspberry Pi (on which we implement the capture of video stream and its processing). We implemented the QR-Code extraction agent from scratch. The complete workflow of BadUSB-C, demonstrating the interaction of all the components is presented at Fig. 1.

When the victim plugs in BadUSB-C device, the USB-C hub handles all the communication between each components

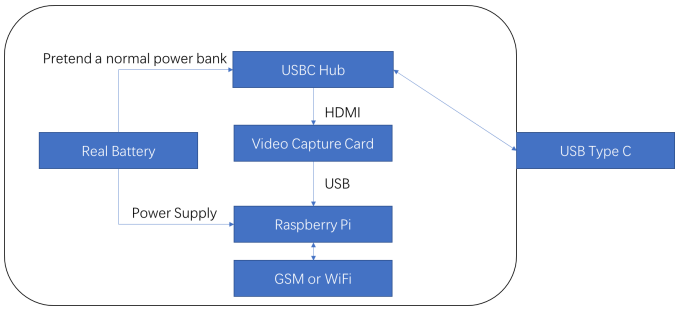

Fig. 1: Workflow of BadUSB-C

and the victim's device. Due to the trust-by-default feature of USB protocol, the victim's video stream will be captured. Later, the stream is either processed directly by Raspberry Pi or transmitted to the attacker through GSM/WiFi components.

With video stream the capability of original Rubbery Ducky attack is considerably expanded. For example, the attacker can directly control the victim's device and view the private data in the stream. More details is presented in the Section V

References

[1] M. Böhme, V. Pham, and A. Roychoudhury, "Coverage-based greybox fuzzing as markov chain," *IEEE Trans. Software Eng.*, vol. 45, no. 5, pp. 489–506, 2019. [Online]. Available: https://doi.org/10.1109/TSE.2017.2785841