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**

I. Introduction

II. Background

III. Related Work

IV. BadUSB-C

A. Motivation

B. Implementation

V. Experiment

VI. Discussiom

VII. Conclusion

References

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.

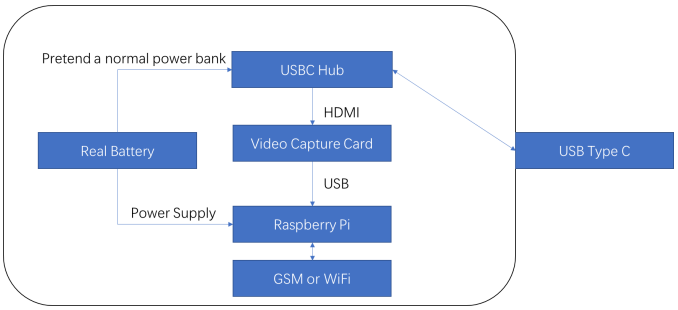

Fig. 1. Workflow of BadUSB-C

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

[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