POSTER: Dynamically Detecting USB Attacks in Hardware (Extended Abstract)

Kyle Denney, Enes Erdin, Leonardo Babun, A. Selcuk Uluagac
 kdenn016@fiu.edu, eerdi001@fiu.edu, lbabu002@fiu.edu, suluagac@fiu.edu
 Cyber-Physical Systems Security Lab (CSL)
 Florida International University
 Miami, Florida

ABSTRACT

Malicious USB devices can disguise themselves as benign devices (e.g., keyboard, mouse, etc.) to insert malicious commands on end devices. Advanced software-based detection schemes (deeper operating system level) are used to identify the malicious nature of such mimic devices. However, a powerful adversary (e.g., as rootkits or advanced persistent threats) can subvert those software-based detection schemes. To address these concerns, we present our ongoing work to dynamically detect these threats in hardware. Specifically, we utilize a novel hardware-assistance mechanism to collect unaltered USB data at the physical layer which is fed into a machine learning-based classifier to determine the true nature of the USB device.

ACM Reference Format:

Kyle Denney, Enes Erdin, Leonardo Babun, A. Selcuk Uluagac. 2019. POSTER: Dynamically Detecting USB Attacks in Hardware (Extended Abstract). In *WiSec '19: ACM Conference on Security and Privacy in Wireless and Mobile Networks*, May 15–17, 2019, Miami, FL, USA. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/3317549.3326315>

1 INTRODUCTION

The plug-and-play nature of USB allows for a wide attack vector in which malicious actors can steal or modify sensitive data in computing systems [6]. For this reason, sensitive computing environments restrict USB devices to remove the threat of malicious USB devices [2].

Work is ongoing in removing the threat of USB-based attacks, specifically in mitigating *rapid-keystroke injection attacks*, where a USB device mimics the keystrokes of a user and types an attack directly into the user's computer. We note that current software-based approaches [4, 5, 7, 8] can be tampered with through rootkits injected in the kernel which obfuscate or remove evidence that a rogue USB device even exists.

In this poster, we present our ongoing work to establish a USB detection methodology enacted at the hardware level to dynamically detect keystroke injection attacks. Through the use of a hardware-based mechanism, we can analyze USB data in a *tamper-proof* way

Figure 1: USB Human Interface Device (HID) report

```
#Delay initial attack by 3 seconds.
DELAY 3000

#Open terminal
ALT F2
DELAY 500
STRING xterm
DELAY 500
ENTER

DELAY 1000
STRING cd $HOME
STRING rm -r ./Documents
STRING Your documents were deleted!
```

Figure 2: Rubber Ducky attack example.

such that a malicious actor may not subvert it. Initial results indicate a valid USB detection methodology which can dynamically detect advanced keystroke injection attacks through successfully distinguishing human and mimicked typing dynamics.

2 BACKGROUND

This work attempts to distinguish differences in malicious and benign typing behaviors to detect malicious USB devices. To do so, we extract the human interface device reports from our captured USB data. Human Interface Devices (HIDs) are a subclass of USB devices which are designed for human input (e.g., keyboards, mice, gaming controllers, etc.) [3]. The report format for a standard keyboard is shown in Figure 1. As shown, there are fields in a keyboard report for a total of six concurrent key presses plus one field for pressed modifier keys (i.e., SHIFT, CTRL, ALT, etc.). Note that the reserved bit is ignored in the reports. When a user presses a key, the HID report will continuously include the key being pressed until the user subsequently releases the key.

To conduct a keystroke-injection attack, the attacker has to (1) create a malicious payload and (2) have the malicious USB device be inserted into the target computer. First, the malicious actor designs the attack s/he wants performed (Figure 2). The attacker writes out the keyboard commands to do such an attack with delays introduced to ensure the commands are typed correctly (e.g., wait for a terminal to open). With the text created, the malicious actor must compile the malicious keystrokes ("payload") into the malicious USB device [1]. With the payload created, the attacker then has to insert the USB device into the target computer. Once inserted, the device injects

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
 WiSec '19, May 15–17, 2019, Miami, FL, USA
 © 2019 Copyright held by the owner/author(s).
 ACM ISBN 978-1-4503-6726-4/19/05.
<https://doi.org/10.1145/3317549.3326315>

Figure 3: Architecture to classify malicious USB attacks.

the malicious commands by sending an HID report with the current character in the payload. To the computer, this HID report simply looks like keystroke presses from a benign keyboard.

3 ARCHITECTURE AND INITIAL RESULTS

The overall architecture for the detection framework is described in Figure 3. First, a USB device is plugged into the hardware-based data collector where the USB signals are sent to the target computer and also copied for processing. The data pre-processor recreates the USB packets and extracts keyboard reports and timestamps. Typing dynamics are extracted from the HID reports to feed into the classifier which determines if the typing is truly human behavior. Each module of the architecture is described in detail below.

Hardware-based Data Collector – We utilize a hardware-based mechanism between USB devices and the host computer to collect USB data. USB traffic flows from the USB device through the data collector and into the host. Utilizing a hardware mechanism such as this provides two distinct advantages. First, it is possible to subvert or tamper data collected by software sniffers. However, we collect data from the physical layer signals created by the USB device, which is not susceptible to OS-level obfuscation means. Additionally, by using a separated piece of hardware for our methodology, an entire segment of computation is removed that would be normally performed on the host computer – saving resources for normal operation.

Data Pre-Processor – The data pre-processor takes incoming binary from the data collector and recreates the USB data packets. With the recreated USB packets, the framework extracts information about the device (e.g., device ID, frequency of packets sent, the data of each packet).

Feature Extraction – We analyze time-based features classify benign and malicious USB keyboard devices. The features used are key-transition time (i.e., the time delay between keystrokes) and keystroke duration (i.e., the duration a key is held down); each used as raw data, and standardized vectors.

Classification Results – The extracted features are then placed into a classifier to determine if the device is malicious or benign. If the device is considered malicious, the FPGA mechanism turns off communication between the USB device and the host computer. Figure 4 shows our initial results of constructing classification models from our training samples. As shown, initial results indicate that the proposed methodology is capable of distinguishing benign and malicious typing behaviors. However, further work must be

Figure 4: Initial classification results using a variety of classification models.

done (e.g., precision-recall analysis, scoring times, etc.) to determine which classification model is most optimal for our approach.

4 CONCLUSION

In this ongoing work, we demonstrate the feasibility of dynamically detecting malicious USB devices at the physical layer. Our approach uses a hardware mechanism placed between the USB-device and the OS to collect unalterable USB traffic. Initial results show that a machine learning classifier can successfully detect advanced keystroke-injection attacks by distinguishing human and machine typing dynamics.

5 ACKNOWLEDGEMENTS

This work is partially supported by the US National Science Foundation (Awards: NSF-CAREER-CNS-1453647, NSF-1663051) and Florida Center for Cybersecurity’s Capacity Building Program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

- [1] [n. d.]. Looks like a flash drive. Types like a keyboard. <https://www.hak5.org/gear/usb-rubber-ducky>
- [2] 2018. IBM bans USB drives - but will it work? <https://nakedsecurity.sophos.com/2018/05/11/ibm-bans-usb-drives-but-will-it-work/>
- [3] Admin. 2018. Tutorial about USB HID Report Descriptors. <https://eleccelerator.com/tutorial-about-usb-hid-report-descriptors/>
- [4] Brandon L Daley. 2016. *USB Safe: Applying One Class SVM for Effective USB Event Anomaly Detection*. Technical Report. Northeastern University, College of Computer and Information Systems Boston United States.
- [5] Collin Mulliner and Edgar R Weippl. 2018. USBBlock: 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*, Vol. 10980. Springer, 278.
- [6] Karsten Nohl and Jakob Lell. 2014. BadUSB—On accessories that turn evil. *Black Hat USA* (2014).
- [7] Sunil Sikka, Utpal Srivastva, and Rashika Sharma. 2017. A Review of Detection of USB Malware. *International Journal of Engineering Science* 14283 (2017).
- [8] Dave Jing Tian, Adam Bates, and Kevin Butler. 2015. Defending Against Malicious USB Firmware with GoodUSB. In *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC 2015)*. ACM, New York, NY, USA, 261–270. <https://doi.org/10.1145/2818000.2818040>