**Serial Debugging README**

**General Information**

* Author: Micah Flack
* Date: 11/28/2022
* Description: This is a presentation that covers the basics of using serial debuggers and how to connect with their interfaces on a given device. Reader should gain the necessary knowledge and skills to apply what was learned to interfaces not demonstrated.

**Why You Should Care**

Just about every IoT or embedded device is going to have some type of serial debug interface. And, in many cases the pre-production debug access is still enabled on these devices. Which is only one of the many concerns surrounding physical device security.

For example, during the 19 months between February 2021 and September 2022, two point-of-sale (POS) malware operators were able to process 167,000 stolen payment records. In many cases, these are simply because of compromised card readers or malicious near field readers controlled by the hackers. However, there are some cases where the operators directly compromise the POS itself or accessory hardware it uses to store payment information before it is retrieved.

In this lab, the board we borrowed is from your average thermal printer used by POS systems for creating receipts. It runs on an ARM Cortex-M4 processor with 512kb of memory and 96kb of RAM using the real-time operating system (RTOS) and a serial port or USB for communication. These resources are enough to run a lightweight webserver or operate as malicious HID if the POS does not filter inputs.

To introduce students to this threat landscape, we begin by identifying those components as well as how to communicate with them.

**Main Ideas**

1. 167,000 payment records were stolen last year because of compromised POS systems between February 2021 and September 2022.
2. Over-specced and unsecured embedded devices are often overlooked as a means of compromising systems within an organization.
3. These vulnerabilities allow any attacker with a basic knowledge and cheap tools to create malicious devices for data exfiltration or for pivoting further into a network.

**Future Direction**

A solid understanding of this topic can allow a researcher to further study hardware reverse engineering. There are many attacks that branch off from this knowledge, like memory forensics and recovery directly from flash while the device is running. A researcher could also perform an audit of the target machine to determine what restrictions were enabled during production for the device. For instance, could you halt the device CPU, directly access memory, make any modifications to the system flash, or even change CPU flags that govern memory flash security?

**Stream of Topics**

Additional topics related to this type of hardware attack may include some of the following:

* Bootloader recovery
* Modifying security options for CPU
* Flashing “unauthorized firmware”
* Understanding how to use GDB w/ hardware devices

**Additional Resources**

* [Video Explanation](https://www.youtube.com/watch?v=_FSM_10JXsM)
* [Bus Pirate (serial debugger) explanation of wiring and use](https://learn.sparkfun.com/tutorials/bus-pirate-v36a-hookup-guide/all)
* [JTAGulator – device for automatically scanning traces for debug interfaces](https://github.com/grandideastudio/jtagulator)
* [Embedded Reverse Engineering - mcuoneclipse](https://mcuoneclipse.com/)
* [POS Malware Trends - Malwarebytes](https://www.malwarebytes.com/blog/news/2022/10/point-of-sale-malware-used-stole-the-details-of-over-167000-credit-cards)