PRINTSHOP: SERIAL PRINTER ENVIRONMENTS AND SECURITY

Research Proposal

Doctor of Philosophy

in

Cyber Operations

January 23, 2024

By

Micah Flack

Dissertation Chair:

Dr. Vaidyan Varghese

Dissertation Committee:

Dr. Yong Wang

Dr. Michael Ham

Beacom College of Computer and Cyber Sciences

ABSTRACT

Securing supply chains for critical infrastructure and any production environment is a growing concern. Third-parties or nation state level attackers have been shown to target employees or infrastructure indirectly to gain access to their target's network. One of the devices being examined to aid this research is the SNBC BTP-S80, a USB/serial connected thermal printer. These devices are made with foreign software and hardware, and they are used off the shelf without any security review. In some instances, the devices implement an MPU/MCU and an FPGA for I/O processing, which creates a potential gap for data to be modified. The proposed research aims to assess these devices for any risks and demonstrate that they could be used as a part of supply chain attacks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Introduction

1.1 Background

Serial printers are devices commonly used for instant reporting of system data for industrial control systems (ICS) and receipts for point-of-sale (POS) systems. These devices are connected to their host using Wi-Fi, bluetooth, ethernet, or USB; in some cases, serial RS232 is an option as well. The goal of this research is to assess what software and hardware protections are enabled, as well as, how configurable the serial printers are for further exploit research.

(a) Square POS (b) SurePOS

Figure 1.1: Comparison of common POS systems

Figure 1.1 shows us two similar looking point-of-sale systems. However, the operating system and required hardware used by both is different. Typically, unless you have the Square provided terminal, their software/client is installed onto an Android or iOS device and connected to a Square compatible card reader [1]. Whereas, the SurePoS, NCR, or other common EFTPoS system will run a proprietary OS based on Windows or Linux [2]. Furthermore, these PoS require some form of printing receipts as record keeping for the business owner and customer. And these devices also vary in terms of processing capabilities and operating system.

For instance, a common thermal printer seen with PoS systems, integrated with fuel pumps, or other industrial control equipment, is the SNBC BTP-S80 thermal printer [3], [4]. There are multiple versions of the device with support for Bluetooth, USB only, or combination of USB/Serial/Ethernet. The bluetooth hardware is provided over an accessory 25-pin serial connection, with more I/O as a serial connection via RS232C connector and USB Type-B. It has driver support for various platforms: Android, iOS, Windows, Linux, and MacOS. The most interesting aspects are the processor, an Arm Cortex M4 clocked at 3.54MHz, and the operating system, a proprietary version of FreeRTOS. The system architecture is Armv7E-M with JTAG/SWD hardware debugging support [5], [6].

By default, the printer has enough headroom to process ESC/POS commands for printing paper and a webserver for debugging or general diagnostics. In theory, the uncompromised device could be flashed with modified firmware to act as a decoy and human-input-device (HID) against the host PoS. The viability of any vulnerabilities would likely be dependent upon supply chain attacks or physical bait-and-switch tactics [7].

1.2 Significance

According to the Federal Trade Commission (FTC), there were 37,932 reports of credit card fraud in 2012 and 87,451 reports in 2022 [8], [9]. This marks an increase of credit card payment fraud by an estimated, 30.5%. By comparison, since 2020, there has been a 14.6% increase in credit card related fraud. Which does not include the millions of other fraud reports the FTC receives every year. In 2022 alone, there were around 5.1 million fraud, identity theft, and miscellaneous reports in total [8], [9]. The statistics for these reports stresses how crucial the security of payment systems are, both physical and online. And, the need to secure them grows every year.

Spyduino is [10] a working example of a programmable BadUSB device using an Arduino to mimic a Human Interface Device (HID). Arduinos are typically more accessible and easily developed compared to an embedded device whose design is more single purpose [11], [12]. Especially if the goal is to not modify hardware or require hands-on access for exploitation. However, the research shows us that is it possible create HID clones from scratch if the hardware is compatible.

The Arduino used in their research is powered by an ATmega328P microcontroller with 32KB flash memory, 2KB SRAM, and 1KB EEPROM. Compared to the most likely target device of our proposed research, the SNBC BTP-S80, it features an ARM Cortex M4 microcontroller with 512KB flash memory, 96KB SRAM, 4KB of EEPROM. This is relevant to the proposed research, because it shows that a device with similar hardware specifications was feasible; meaning, it is likely that our own research will be successful. BadUSBs are a known and tested area of research. The novelty of this proposal comes from the assessment of the printer devices and showing whether one could be used maliciously within their environments (e.g., PoS systems, or ICS).

1.3 Research Goals and Objectives

This research primarily focuses on physical POS systems or terminals and their hardware (serial accessories), rather than online solutions. For instance, not mobile payment apps like Venmo, CashApp, Zelle, or Paypal [13] since their environments typically do not use serial print devices. It is also likely that more research would be needed for emulating touch inputs for mobile environments versus the traditional keyboard attacks that will be implemented. Presumably, the host-to-guest communication will not differ greatly between other environments (e.g., ICS). If the printers have demonstrable weaknesses with an Ubuntu host, that will fulfill the testing requirements.

The goal of this research is to further establish academic works in regards to embedded printer devices testing and security. This area is loosely documented within academia and only mentioned vaguely in relation to statistical reports or applied research using entirely different environments. For instance, most researchers limit their analysis of the environment to smartphones and the corresponding payment app, or detection systems for card skimmers [7].

Through this research we hope to identify supply chain risks using side channel attacks from auxiliary devices. Some examples of how the research could be applied in the future vary: BadUSB/BashBunny [14], JuiceShop [15], DVWA [16], or Webgoat [17]. Works within the PoS system context or embedded systems discussing supply chain attacks through third-party hardware are limited.

1.4 Research Questions

The research questions that this proposal seeks to answer are as follows:

- Q1: Can the hardware be reflashed with a modified firmware image (e.g., FreeRTOS, ReconOS, VxWorks)? Testing a version of the original firmware with additional libraries, or an alternative OS, allows us to see if supply chain attacks are a concern. Either by the manufacturer, supplier, or other party. Reflashing is not novel by itself, however, the device might have protections in place to prevent it.

- Q2: Does the hardware and firmware have enough resources to support HID functionality on-top of printing? In other words, can we maintain operation of standard printer command interpretation and side-channel input attacks without causing crashes or delays? The viability of the attack depends on it going unnoticed by operators or technicians.

- Q3: Besides HID cloning, what other threat areas are exposed (e.g., network stack, web management portal, memory protections)? Are there any identifiable or known exploits when accessing the configuration panel (e.g., HTTP/2)? These provide a non-invasive method for bootstrapping the device.

Each of these goals will be approached individually as prescribed by the methodology.

Related Works

2.1 RTOS: Software and Security

[18] introduces several embedded kernels and discusses their differences in regard to developing a secure mass storage device. For this research, we are primarily interested in RTOS-like kernels because of existing support for a sample device like the SNBC BTP-S80 printer. However, the paper criticizes such operating systems because their "real-time driven design is barely compatible with the overhead produced by security mechanisms." For many applications, there is a trade off with RTOS where performance is the main criteria and security is not a priority. [19] introduces several common RTOS and discusses their security issues. Notably, most RTOS are susceptible to code injection, cryptography inefficiency, unprotected shared memory, priority inversion, denial of service attacks, privilege escalation, and inter-process communication vulnerabilities. Depending on the MPU (microprocessor unit), the vendor has hardware protections like Intel SGX or Arm Trust Zone. These are all areas that can be used for pivoting onto the device, especially shared memory and privilege escalation. If the target device firmware is outdated (or, even libraries used by the firmware) and there are known CVEs that can be repeatedly exploited, persistence mechanisms are not a requirement to gain routine access.

2.2 PoS Attack Patterns

Typically, when discussing attack patterns for PoS systems they are limited to card skimming, fake payment processor requests, or EMV cloning. In rarer cases, they might deliver malware to perform memory scraping within the PoS system or attempt swapping hardware while employees are distracted. None of these attacks include thermal printers at any point during their attack chain or delivery.

Easily the most common and well known type of attack is card skimming. Attackers will place these devices directly on top of the existing equipment to skim, or gather, credit card information at the time of purchases. They can be incredibly difficult to identify because of the sleek and stealthy designs that fraudsters use. But there is plenty of research being presented on how to quickly detect these devices [7], [20].

Without going into too much technical detail, card skimming attacks are accomplished by reading the signals emitted when swiping a magnetic card or by using an NFC reader in proximity to the payment terminal. When the customer goes to pay and uses their card, the nearby skimmer will record the transaction data being transferred. NFC skimmers, however, are not limited to being used near the terminals. Skimmer capabilities vary, and in some cases they have cameras as well or keypads for capturing PIN and zip code data.

In response to the susceptibility of magnetic cards, EMV cards were created. They are able to avoid the issues that magnetic cards and NFC share by using a chip to securely exchange transaction data with the payment terminal using secret authentication codes. The idea is that these codes cannot be tampered with or easily cloned. Despite these security advancements, EMV cards are susceptible to pre-play attacks targeting the "unpredictable number" (UN) algorithm used by ATMs [21].

[13]

Social engineers use payment processor mobile applications to directly target their victims instead of using elaborate and technical attacks against servers or user equipment [22]. The attackers simply send payment requests disguised as payments using their preferred platform. Unwittingly, the victim will accept the request thinking they were receiving money instead.

In some cases, the fraudster sends the victim money but requests a refund shortly after. As a result, the victim is either charged fees for processing the transactions or they have already spent the refunded money. These attacks are much simpler in-terms of delivery compared to the others and the intended outcomes are different. There are instances where the user device is compromised by malware specifically for exfiltrating banking data or similar PCI, but further discourse is outside the scope of the proposed research [23].

Researchers at Stony Brook University [24], demonstrated a successful introspection-based memory scraping attack against nine commercial PoS applications. Within their environment, it is assumed that the given VM (i.e., Dom0) within the shared virtualization platform (i.e., Xen) is compromised and it has escaped the guest environment. Because the privileges associated with the first VM, it has read access to the others and can perform out-of-VM memory scraping. This exact attack is likely limited to the platform used for the experiment, Xen Hypervisor; attempting something similarly against VMWare, Virtualbox, or QEMU would require further experimentation due to architectural differences. Also, PCI-DSS and PA-DSS requirements were not an obstacle for this attack since the data is not stored to disk and it is read from memory instead.

2.3 BadUSB-like Devices

BadUSB is a well-known and documented attack vector. One of the most popular hacker tools is built-on the concept [14]. However, there are some limitations:

- Precision of attacks is limited since scripts or effects are typically deployed blind. There is no knowledge of the user environment nor ability to interact with functional user interface mechanisms (e.g., a mouse clicking a button).
- Limited to the USB 2.0 standard. Meaning, no support for video adapters like HDMI, DisplayPort, or PowerDelivery like with USB 3.0.
- There are existing methods for limiting USB access from the host, such as GoodUSB [25].

GoodUSB supports the Linux USB stack, so another solution would be required for

Windows systems or RTOS. This all depends on the environment of the connected host, the PoS system. It is entirely possible that the PoS could have software like Crowdstrike Falcon deployed, which would monitor system behavior and mass storage device access [26]. Although the experiment environment will not use such software, it is an important distinction to make.

In [27], they describe several attacks at each of the applicable layers to USB attacks: the human, application, transport, and physical layers. These attacks would typically require some human element for deployment, but that is not the focus of the research (e.g., social engineering versus hardware hacking). Whereas the physical layer could allow signal eavesdropping or injection. This could enable a modified printer to overvolt the host (USBKiller [28]) to cause physical damage or perform other side-channel attacks [29]. Either of those methods would require investigating the device hardware to determine what level of control the bootloader or operating system has over power delivery.

2.4 Summary

As demonstrated by the previous works, vulnerability assessment of an embedded device is a well documented process. However, the extent that a serial thermal printer (e.g., Figure 1.1) can be maliciously expanded through a modified FreeRTOS image, while supporting original functionality, has not. And, given success in the assessment, it could suggest room for continual and improved research.

Proposed Research

3.1 Methodology

For this research, the quantitative approach and case study research will be used [30], [31] to create a design artifact. The goal being to gather and examine, point-in-time, data from a serial printer device as a common sample representative of the affected population. By using quantitative survey research, it is possible to evaluate potential vulnerabilities for the attacks hypothesized, as well as, prototype a modified firmware image to use them against the host environment.

Each step of the process shown in Figure 3.1, is described as follows:

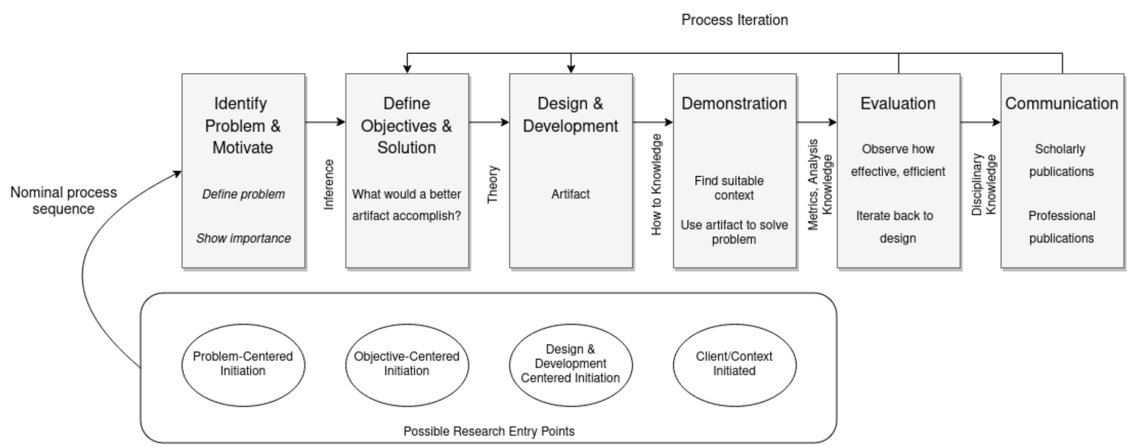

Figure 3.1: Research methodology diagram [32]

- **Problem Identification:** A research gap exists when identifying the risks that off-the-shelf hardware exposes to host environments (i.e., PoS systems, HMI, ICS). Auxiliary devices, like serial printers, are black box pieces of equipment with questionable supply chain and development life cycles.

- **Objectives and Solutions:** This research aims to thoroughly examine such a device and demonstrate that it can be used as a part of a hypothetical supply chain attack. Using the information gathered, a modified firmware image with HID cloning will be created. This solution should prove that such an attack is a viable method and there are existing issues with sourcing third-party hardware.

- **Research Design:** The research design process begins by tearing down the target device and identifying components. That information is then used to recover the firmware for analysis, as well as, analyze the network and USB communications with a host. After reviewing, a modified firmware image will be created and then flashed to the device. All information is documented and collected. Refer to Section 4.1 for further detail about the exact process.

- **Demonstration:** Using the artifact, the researcher will demonstrate that scripted HID attacks against the host are possible. Should the host filter devices using restricted vendor IDs, this attack will still work irregardless of the host operating system. It is also important that original functionality is maintained with the modified firmware.

- **Evaluation:** Whether or not the research has been successful depends on several factors: firmware recovery, firmware modification (using recovered firmware or third-party), attack development, and sustaining the original ESC/POS print functions. Assuming each step has been completed, the final demonstration should be able to show working print operations and scripted attacks against the host. These attacks should also work irrespective of target environment.

- **Communication:** The research will present the identified gaps and issues to the embedded security field. By demonstrating the research and artifacts, the researchers hope to greater increase the scrutiny for auxiliary, third-party hardware and the associated risks. In addition, the artifacts could potentially be used for future research into embedded security and exploit development.

Research Design

4.1 Design Process

The design process is divided into two parts, information collection and exploitation. Beginning with teardown of the equipment, identification of components, and then analysis of the firmware, networking, and lastly, the USB communications between the host and device. Using the information gathered within the first stage, modifications will be made to the firmware installed on the serial printer. Each step of the process shown in Figure 4.1, is described as follows:

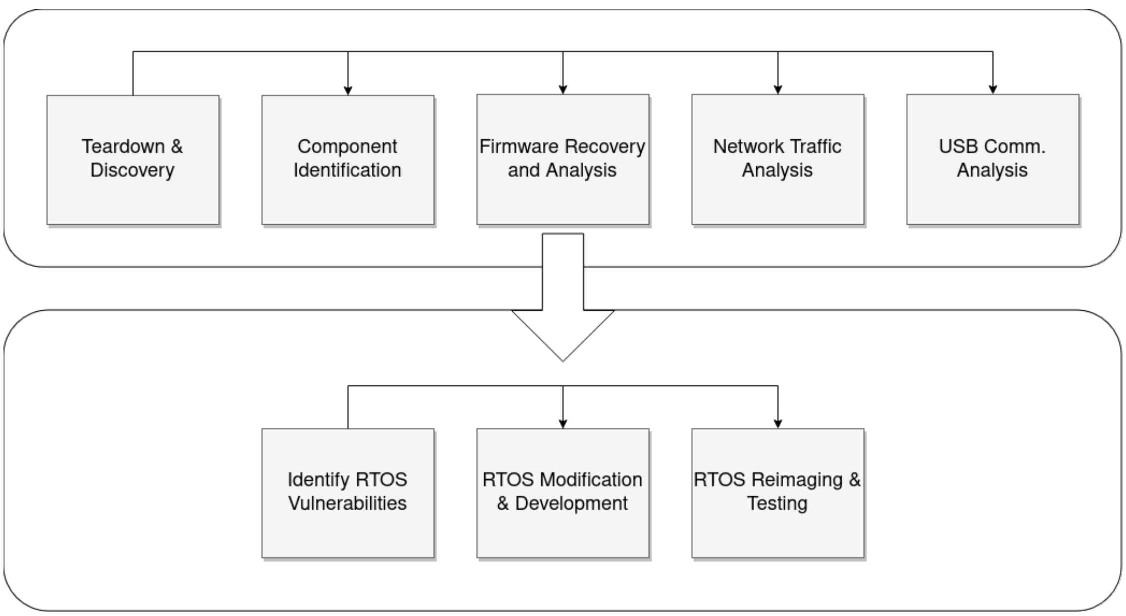

Figure 4.1: Design process diagram

- **Teardown and Discovery:** The device is disassembled and documented at each step. Pictures are taken of each component, part numbers are identified, and technical datasheets are collected.

- **Component Identification:** Using information from the prior step, component function is identified and any information needed to interface with the component is documented.

- **Firmware Recovery and Analysis:** Using the technical datasheets, firmware is recovered and analyzed in a disassembler (e.g., Ghidra). Libraries used by the operating system and their opensource repositories are documented. Using the opensource information, potential vulnerabilities are identified within the firmware.

- **Network Traffic Analysis:** Any network traffic created during use is captured and analysed using software tools (e.g., WireShark). In conjuction with prior firmware analysis and identification of operating system libariries, communications are reviewed for potential vulnerabilities. This is a potential area for remote code execution (RCE) if the management software is poorly implemented.

- **USB Communication Analysis:** Communications between the host and device are captured for future development of human-interface-device (HID) cloning. The information is needed for accurately cloning identifiers assuming hosts restrict devices using vendor IDs.

- **Identify RTOS Vulnerabilities:** Should analyses identify any exploitable vulnerabilities, further research is conducted to discover any public releases or proof of concepts (PoCs).

- **RTOS Modification and Development:** Opensource RTOS firmware is modified to allow HID cloning while maintaining original print functionality. The attack vector is crucial step towards proving viability of supply chain attacks using the print devices.

- **RTOS Reimaging and Testing:** Firmware is built and reimaged/reflashed onto the target device. Testing includes the verification of ESC/POS commands interpreter operation and HID attack vector.

4.2 Data Collection Process

The data collection process begins with gathering technical specifications from device manufacturers. Typically, these contain information about the capabilities of the intended device functions. For a printer, this could contain information ranging from hardware specifications (e.g., CPU, architecture, memory) to things like printed pages per minute. This information forms the baseline for the device survey. Afterwards, further specifications will be gathered for components as each device is disassembled and examined.

The next step in the data collection process would be identifying the SoC. In the event that there is no beforehand knowledge, the SoC can be identified by comparing gathered datasheets during the components discovery. This is easily accomplished using an online service like FindChips [33]. The expected type and format for SoCs is described by Figure 4.1.

The process for gathering flash/memory chip specifications is similar; identify serial number and manufacturer, then find the component datasheet. Gathering the pin layouts and format is useful for later stages, should manual flash recovery be needed. The expected format for memory chips can be seen at Figure 4.2.

Specifications	
Architecture	32-bit ARM
Platform	ARM Cortex-M3
Frequency	80-MHz, 100DMIPS performance
Memory	128KB single-cycle Flash memory 64KB single-cycle SRAM
Firmware	Internal ROM loaded with StellarisWare
Advanced Comm. Interfaces	UART, SSI, I2C, I2S, CAN
Debug Interfaces	JTAG, SWD
Package format	100-pin LQFP 108-ball pin BGA

Table 4.1: SoC technical specs example using Stellaris LM3S2793 Microcontroller

The final report will contain each of these tables for the device and their identified core components. Operating system features and protections will be loosely summarized for the device, and there is no set reporting format or requirements. The identified information will aid the final step of the process, creating a design artifact.

Specifications	
Single power supply operation	2.7 to 3.6V
Software Features	SPI Bus Compatible Serial Interface
Memory architecture	Uniform 64KB sectors 256 byte page size
Programming	Page programming (up to 256 bytes) Operations are page-by-page basis Accelerated mode via 9V W#/ACC pin Quad page programming
Erase commands	Bulk erase function Sector erase for 64KB sectors Sub-sector erase for 4KB and 8KB sectors
Protections	W#/ACC pin used with Status Register Bits to protect specified memory regions andconfigure parts as read-only One time programmable area for permanent and secure identification
Package format	16-pin SO 8-contact WSON 24-ball BGA, 5x5 pin config 24 ball BGA, 6x6 pin config

Table 4.2: Memory specifications example using Infineon Technologies S25FL064P [34]

4.3 Hardware Assessment

NIST SP 800-115 [35] provides general guidelines for performing information security testing and assessment, however, there is little information regarding hardware reverse engineering and firmware analysis. Their guidelines are aimed more towards single/multi-tasking operating systems like Windows or Unix-like, those where network logging and listener agents is feasible. For the targeted devices in this research proposal, a different approach is needed that evaluates hardware protections of the SoC and flash memory.

Analysis of device components, once disassembled, requires using a hardware debugger tool with the correct interface. The majority of the targeted devices are expected to use joint test action group (JTAG) or single wire debugging (SWD). By referring to the manufacturer datasheet for a given SoC, it is possible to identify the pin layout for serial debugging access.

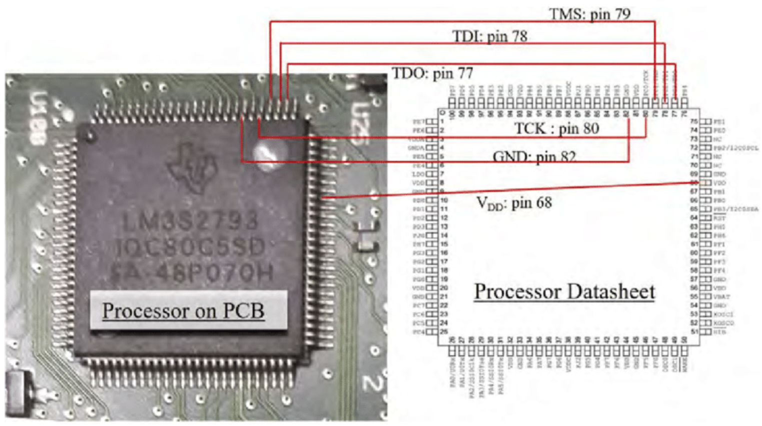

Figure 4.2: JTAG pin out example for Texas Instruments LM3S2793

Figure 4.2 is an example showing what the physical SoC looks like on a PCB compared to the pin layout described in the datasheet. The dot in the top left of the SoC denotes the beginning of the pin layout. Counting in a counter-clockwise method indicates the pin number and the associated functions. For instance, to access the JTAG debug interface on the LM3S2793:

Function	**Pin #**	**Function**	**Pin #**
TDO	77	TDI	78
TMS	79	TCK	80
GND	82	V_{DD}	68

Table 4.3: example JTAG pin-out for the LM3S2793

Using this information, a device like the JTAGULATOR [36] can be connected and enumerate or verify pin layouts as described. Ball joint SoCs require a different process and are much harder to debug if there is no visible header available on the board. Once an interface is connected, if debugger access is not disabled, the researcher can interact with the bootloader to further investigate enabled protections and recover flash storage.

If the JTAG is disabled, the researcher will then attempt to recover flash manually using a device like the Segger J-Link [37]. The Segger has pre-defined and existing support for working with flash memory and flash breakpoints, whereas using OpenOCD with the JTAGULATOR would require time crafting custom configurations. Assuming there are no access protections to the flash memory, the researcher can begin performing firmware analysis to identify the operating system or potential vulnerabilities. Documenting the size and address range of memory regions is a key part of the process.

4.4 Networking Traffic Analysis

Work in progress.

4.5 USB Communication Analysis

Work in progress.

4.6 RTOS Modification

Work in progress.

Timeline

The timeline for the research proposal is divided into several parts: review, disassembly, identification, analysis, firmware modification, testing, and writing. The dates provided are rough estimates and will vary as the project progresses. Each part of the research process is described as follows (refer to Table 5.1):

Task	Duration	Start Date	End Date
Review	7 days	24 Mar 25	30 Mar 25
Review proposal comments/feedback	2 days	24 Mar 25	25 Mar 25
Amend any changes	3 days	26 Mar 25	28 Mar 25
Prepare work environment	2 days	29 Mar 25	30 Mar 25
Disassembly	7 days	31 Mar 25	06 Apr 25
Disassemble printer	2 days	31 Mar 25	01 Apr 25
Document interior/exterior of device	5 days	02 Apr 25	06 Apr 25
Identification	7 days	07 Apr 25	13 Apr 25
Identify device components	3 days	07 Apr 25	09 Apr 25
Attempt hardware debug	1 days	10 Apr 25	10 Apr 25
Identify software/hardware protections	2 days	11 Apr 25	12 Apr 25
Document	1 days	13 Apr 25	13 Apr 25
Analysis	76 days	14 Apr 25	29 Jun 25
Capture network/USB traffic	7 days	14 Apr 25	20 Apr 25
Analyze network/USB traffic	31 days	21 Apr 25	21 May 25
Analyze recovered firmware	31 days	22 May 25	22 Jun 25
Document previous tasks	7 days	23 Jun 25	29 Jun 25
Firmware Modification	62 days	30 Jun 25	31 Aug 25
Modify existing firmware	31 days	30 Jun 25	31 Jul 25
Modify third-party firmware	31 days	01 Aug 25	31 Aug 25
Testing	60 days	01 Sep 25	31 Oct 25
Setup testing environment	7 days	01 Sep 25	07 Sep 25
Reflash device w/ modified firmware	14 days	08 Sep 25	21 Sep 25
Test print operations	7 days	22 Sep 25	28 Sep 25
Test HID attacks	7 days	29 Sep 25	05 Oct 25
Make changes and retest	25 days	06 Oct 25	31 Oct 25
Extra	30 days	01 Nov 25	30 Nov 25
Writing	62 days	01 Dec 25	31 Jan 25

Table 5.1: Research lifecycle

References

[1] J. Ondrus and K. Lyytinen, "Mobile Payments Market: Towards Another Clash of the Titans?" In *2011 10th International Conference on Mobile Business*, Jun. 2011, pp. 166–172. DOI: `10.1109/ICMB.2011.41`. (visited on 10/23/2023).

[2] S. T. Ebimobowei, Z. Enebraye Peter, and Y. Pual, "THE ROLE OF SOFTWARE IN A CASHLESS ECONOMY (CASE STUDY NIGERIA)," *International Journal of Research -GRANTHAALAYAH*, vol. 6, no. 1, pp. 177–186, Jan. 2018, ISSN: 2350-0530, 2394-3629. DOI: `10.29121/granthaalayah.v6.i1.2018.1607`. (visited on 10/23/2023).

[3] *SNBC BTP-S80 Thermal Printer - Black Cabinet (USB/Serial/Ethernet)*, https://www.crs-usa.com/products/snbc-btp-s80-thermal-printer-black-(usb-serial-ethernet). (visited on 10/23/2023).

[4] *SNBC New Beiyang-Intelligent Micro-Super, Smart Express Cabinet, Barcode Label Printer, Ticket Printer_Electronics_Receipt/Log Printer, Barcode/Label Printer, Special Scanning Products, Mixed Print Scanning Products, Smart Express Cabinet, Smart Micro-Super_New Beiyang specializes in the development, production, sales and service of intelligent print identification and system integration products. It provides leading products and complete, one-stop application solutions for various industries around the world. It is the only core design in the industry in the country through independent innovation. Manufacturing technology and forming a large-scale production enterprise.* https://www.snbc.com.cn/. (visited on 10/23/2023).

[5] *Cortex-M4*, https://developer.arm.com/Processors/Cortex-M4. (visited on 10/23/2023).

[6] *FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions*, https://www.freertos.org/index.html. (visited on 10/23/2023).

[7] N. Scaife, C. Peeters, and P. Traynor, "Fear the Reaper: Characterization and Fast Detection of Card Skimmers," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1–14, ISBN: 978-1-939133-04-5. (visited on 10/23/2023).

[8] *Consumer Sentinel Network Data Book for January - December 2011*, https://www.ftc.gov/reports sentinel-network-data-book-january-december-2011, Oct. 2023. (visited on 10/23/2023).

[9] C. FortheSentinel, "Consumer Sentinel Network Data Book 2022," 2022.

[10] E. Karystinos, A. Andreatos, and C. Douligeris, "Spyduino: Arduino as a HID Exploiting the BadUSB Vulnerability," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, May 2019, pp. 279–283. DOI: 10.1109/DCOSS.2019.00066.

[11] *A Comparative Study of Arduino, Raspberry Pi and ESP8266 as IoT Development Board. — International Journal of Advanced Research in Computer Science — EBSCOhost*, https://openurl.ebsco.com/EPDB%3Agcd%3A15%3A5556095/detailv2?sid=ebsco%3Ap (visited on 03/05/2024).

[12] D. P. Griffiths and D. M. N. Kabir, *ECIAIR 2019 European Conference on the Impact of Artificial Intelligence and Robotics*. Academic Conferences and publishing limited, Oct. 2019, ISBN: 978-1-912764-44-0.

[13] Y. Wang, C. Hahn, and K. Sutrave, "Mobile payment security, threats, and challenges," in *2016 Second International Conference on Mobile and Secure Services (MobiSecServ)*, Feb. 2016, pp. 1–5. DOI: 10.1109/MOBISECSERV.2016.7440226.

[14] Hak5, *Bash Bunny*, https://shop.hak5.org/products/bash-bunny. (visited on 10/23/2023).

[15] *OWASP Juice Shop — OWASP Foundation*, https://owasp.org/www-project-juice-shop/. (visited on 10/23/2023).

[16] R. Wood, *DAMN VULNERABLE WEB APPLICATION*, Oct. 2023. (visited on 10/23/2023).

[17] *OWASP WebGoat — OWASP Foundation*, https://owasp.org/www-project-webgoat/. (visited on 10/23/2023).

[18] R. Benadjila, M. Renard, P. Trebuchet, P. Thierry, and A. Michelizza, "Wookey : Usb devices strike back," 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:199552896.

[19] W. D. Yu, D. Baheti, and J. Wai, "Real-Time Operating System Security,"

[20] N. Scaife, J. Bowers, C. Peeters, *et al.*, "Kiss from a Rogue: Evaluating Detectability of Pay-at-the-Pump Card Skimmers," in *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019, pp. 1000–1014. DOI: 10.1109/SP.2019.00077. (visited on 02/13/2024).

[21] M. Bond, O. Choudary, S. J. Murdoch, S. Skorobogatov, and R. Anderson, "Chip and Skim: Cloning EMV Cards with the Pre-play Attack," in *2014 IEEE Symposium on Security and Privacy*, San Jose, CA: IEEE, May 2014, pp. 49–64, ISBN: 978-1-4799-4686-0. DOI: 10.1109/SP.2014.11. (visited on 02/13/2024).

[22] D.-G. Beju and C.-M. Făt, "Frauds in Banking System: Frauds with Cards and Their Associated Services," in *Economic and Financial Crime, Sustainability and Good Governance*, ser. Contributions to Finance and Accounting, M. V. Achim, Ed., Cham: Springer International Publishing, 2023, pp. 31–52, ISBN: 978-3-031-34082-6. DOI: 10.1007/978-3-031-34082-6_2. (visited on 02/13/2024).

[23] H. Darvish and M. Husain, "Security Analysis of Mobile Money Applications on Android," in *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA: IEEE, Dec. 2018, pp. 3072–3078, ISBN: 978-1-5386-5035-6. DOI: 10.1109/BigData.2018.8622115. (visited on 02/13/2024).

[24] J. Hizver and T.-c. Chiueh, "An Introspection-Based Memory Scraper Attack against Virtualized Point of Sale Systems," in *Financial Cryptography and Data Security*, D. Hutchison, T. Kanade, J. Kittler, *et al.*, Eds., vol. 7126, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 55–69, ISBN: 978-3-642-29888-2 978-3-642-29889-9. DOI: 10.1007/978-3-642-29889-9_6. (visited on 02/13/2024).

[25] D. J. Tian, A. Bates, and K. Butler, "Defending Against Malicious USB Firmware with GoodUSB," in *Proceedings of the 31st Annual Computer Security Applications Conference*, ser. ACSAC '15, New York, NY, USA: Association for Computing Machinery, Dec. 2015, pp. 261–270, ISBN: 978-1-4503-3682-6. DOI: 10.1145/2818000.2818040. (visited on 10/24/2023).

[26] J. Backer, "Sdn-controlled isolation orchestration to support end-user autonomy," Ph.D. dissertation, WORCESTER POLYTECHNIC INSTITUTE, 2021.

[27] J. Tian, N. Scaife, D. Kumar, M. Bailey, A. Bates, and K. Butler, "SoK: "Plug & Pray" Today – Understanding USB Insecurity in Versions 1 Through C," in *2018 IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 1032–1047. DOI: 10.1109/SP.2018.00037. (visited on 11/05/2023).

[28] *USB Kill devices for pentesting & law-enforcement*, https://usbkill.com/. (visited on 11/05/2023).

[29] K. Sridhar, S. Prasad, L. Punitha, and S. Karunakaran, "EMI issues of universal serial bus and solutions," in *8th International Conference on Electromagnetic Interference and Compatibility*, Dec. 2003, pp. 97–100. DOI: 10.1109/ICEMIC.2003.237887. (visited on 11/05/2023).

[30] R. Babbie, *The Basics of Social Research*. Cengage Learning, 2017, ISBN: 978-1-305-58586-7.

[31] J. Creswell and J. Creswell, *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE Publications, 2017, ISBN: 978-1-5063-8671-3.

[32] K. Peffers, T. Tuunanen, M. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, Dec. 2007, ISSN: 0742-1222. DOI: 10.2753/MIS0742-1222240302. (visited on 03/04/2024).

[33] *Findchips: Electronic Part Search*, https://www.findchips.com/. (visited on 11/06/2023).

[34] *S25FL064P Series NOR Flash Datasheets – Mouser*, https://www.mouser.com/c/ds/semiconducto ics/nor-flash/?series=S25FL064P. (visited on 11/06/2023).

[35] "NIST SP 800-115," *NIST*, Jan. 2020. (visited on 11/06/2023).

[36] *JTAGulator*, Grand Idea Studio, Nov. 2023. (visited on 11/06/2023).

[37] *SEGGER J-Link debug probes*, https://www.segger.com/products/debug-probes/j-link/. (visited on 11/06/2023).