FIRMWARE Extraction using JtAG/UART

Prudhvi Raj Dasari

A Capstone Project Submitted to the   
University of North Carolina Wilmington in Partial Fulfillment  
of the Requirements for the Degree of  
Master of Science

Department of Computer Science  
Congdon School of Supply Chain, Business Analytics, & Information Systems

University of North Carolina Wilmington

2025-09-17

Approved by

Advisory Committee

Makanahalli Annaiah Ajay Miller Kasey C

Committee Member 1 Committee Member 2

Hosam Alamleh

First name Lastname, Chair

Accepted By

Yang Song

Dean, Graduate School

**CHAPTER 1: INTRODUCTION**

* 1. **Background**

Firmware is the low-level software embedded of all modern electronic devices from routers and IOT devices and smart appliances. It controls the boot process and manages hardware resources; it enables communication between system components. Because firmware memory stores in non-volatile, it’s rarely updated in production. It’s one of the most high-level targets for the attacker when vulnerabilities exist [3].

Most of the Research says that the embedded devices are outdated libraries, hardcoded credentials and weak update mechanisms and leaving them to exploitation [1]. Attackers may do the reverse engineering firmware to get sensitive data, discover vulnerabilities and launch large scale attacks. For an instance Mirai botnet infected millions of consumers IoT devices and perform massive DDoS attacks, highlight the importance of firmware security auditing [1].

This project mainly focusses on the firmware extraction and analysis by using the JTAG and UART interfaces, used to dump firmware directly from flash memory.

**1.2 Problem Statement**

Firmware plays a major role in the security of the modern embedded system, it controls the initialization, configuration and communication with other components. However, most of the firmware embedded devices are rarely updated, manufacture ship devices with outdated software components, hardcode credentials and insecure file systems leaving them vulnerable to exploitation [2].

Furthermore, access to debugging interface of JTAG and UART are frequently disable and making legal analysis of challenging of leaving vulnerabilities undiscovered for long periods [3]. Although firmware extraction and reverse engineering, researchers reveal a critical issue of unencrypted configuration files, insecure bootloaders and outdated libraries. Without effort of extracting and analyzing firmware, the risk remains hidden and frequently exploited by attackers.

**1.3 Objectives**

* Identify and verify the JTAG/UART pins on device by using the multimeter.
* Extracting the firmware using the hardware tools (Z3x shell, Easy JTAG Plus, Easy JTAG Tool) and save it into the binary file.
* Configure and stabilize the hardware interface to enable reliable communication during firmware dumping.
* Validate the extracted firmware by checking the dump size and generating the checksum to confirm the integrity.
* Perform the vulnerability analysis on extracted firmware to find the hardcode credentials and insecure file system.

**1.4 Importance of the Study**

The outcome of the project is not only a single firmware dump, but also a documented methodology that others can perform similar analysis. By giving an idea of JTAG/UART extraction and analysis, the project contributes to the major field of the embedded device security test [10]. The result may help the manufacture and security research on the secuirty firmware by preventing unauthorized access, stronger authentication and implementing the security updates mechanism.

**CHAPTER 2: Literature Review and Analysis**

**2.1 Introduction**

A secure network environment depends upon the integrity of the router firmware. Many studies say that the vulnerability in embedded devices is exploited to gain remote access, install malware or participate in large scale botnet attacks [1]. The prior work on this chapter is extraction firmware, reverse engineering and vulnerability analysis on the JTAG/UART methods and their application to consumer routers.

**2.2 Firmware Extraction and Interfaces**

Probably most of the papers classify the firmware skill techniques into software-based and hardware-assisted methods. Cui et al. and Singh et al. describe UART and JTAG are most used interfaces for hardware-level firmware access with tools like Easy JTAG and OpenOCD. Belimov et al [1]. provides an organized methodology for firmware skills, including downloading from the vendors sites, intercepting over in the air update, and physical extraction from the flash chips [4].

**2.3 Identified Vulnerabilities**

As we observed from the research the continuous common security flaws reveals in the router firmware’s are:

* Hardcode admin passwords. [5]
* Out-dated libraries with known CVE; s [1]
* Insecure default configuration that enables remote access [5]
* Weak encryption or plain text storage of sensitive information [4]

These vulnerabilities are frequently present because vendors do not update the router firmware or disable the interface before shipping the devices. [1]

**2.4 Challenges**

Most of the authors mention the difficulty of analyzing proprietary file systems and confused binaries. For example, doing router firmware, reverse engineering is expensive and time consuming and highly false positive rate in automated analysis tools [2]. Moreover, most of the routers do not display JTAG/UART lines. Which is very complicated to find the interface pins and necessitates fallback approaches such as bus-pirates or bus blaster interfaces [3].

**2.5 Literature Synthesis**

Across the research papers, there is an opinion that router firmware analysis is a critical step toward improving network security. The combination of JTAG/UART firmware extraction and software assisted reverse engineering has been proven successfully in uncovering vulnerabilities that are not visible through conventional backbox testing. However, gaps remain in automating process in developing frameworks that vastly use consumer devices in use.

**CHAPTER 3: Methodology**

**3.1 Overview**

This chapter explains the practical approach that stated the objective in chapter 1. The methodology combines hardware assisted firmware extraction with software based reverse engineering and vulnerability analysis. A multi-phase process was designed for certain reliable firmware development in the presence of hardware or interface limitations. [1][3]

**3.2 Hardware Setup**

**3.2.1 Device**

The device selected for this project is a Linksys WRT54G2 v1.5 router, a consumer grade networking device with 2MB flash memory. The router I was chosen because of the popularity, and it is one of the easy routers that we can find the JTAG/UART test points on its PCB. [1]

**3.2.2 Tools and Interfaces**

* **Easy JTAG Plus/Easy JTAG Tool:** Primary tool for Easy JTAG communication read the flash memory [3].
* **Multimeter:** By using the multimeter to find the JTAG pins in the router by checking the continuity test for GND, TDI, TDO, TCK, TMS and SRST. [1]
* **Bus Pirate V3:** If In case the JTAG fails to initialize or access the memory, we can use the Bus Pirate and configure a SPI programmer for the backup tool.
* **Supporting Equipment:** Soldering the JTAG pins on the router, jumper wires, and connectors for stable contact of signal. [9]

**3.2.3 Pinout Verification**

Using the Multimeter in continuity mode, firstly the ground pin was identified. Each JTAG pin is mapped to the connector. The detected pinout was verified against known Broadcom/Atheros JTAG layout. TRST pin was not found and ignored by the software by selecting SRST during the initialization [3].

**3.3 Software Setup**

**3.3.1 Easy JTAG Configuration**

The Easy JTAG plus/Easy JTAG tool software was launched in the JTAG mode. Basically, JTAG speed was less than 1000 kHz for stable communication. The Initialization set to SRST, certainly the absence of TRST did not prevent connection. CPU ID detection logs were captured successfully by JTAG handshake [3].

**3.3.2 Firmware Dumping**

Once the connectivity was successfully established, the **Read Target Flash** was dump the entire flash memory range

* **Start Address: 0x00000000**
* **Length: 0x00200000(2 MB** Flash**)**

**3.3.3 Fallback with Pirate Bus**

If the JTAG fails, Bus pirate will connect in 3.3 V SPI mode (MOSI, MISO CLK, CS, GND). The flash Rom utility used to identify the flash chip and perform a full readout. This redundancy ensured the project could proceed even If JTAG communication was unreliable [1].

**3.4 Firmware Acquisition**

* **UART-based dumping:** If a shell is a accessible tool like dd or cat are use to copy block devices and store a full filesystem image [3].
* **JTAG based dumping:** when UART is locked, JTAG is used with OpenOCD or vendor software to read the flash memory contents. The CPU is in rest(SRST) to stable the read of memory [3].

**3.5 Vulnerability Identification**

* Presence of default admin credentials
* Insecure or open ports(e.g Telnet FTP)
* Outdated SSL/TLS libraries

**3.5 Firmware Unpacking & Reverse Engineering**

The acquired binary image is processed such as Binwalk. Firmadyne to identify the file System, Extract configuration and disassemble binaries for vulnerability analysis [1].

Sensitive information such as plaintext credentials or documented.

**3.6 Security Analysis & Reporting**

Static and dynamic analysis techniques are identifying backdoors, weak authentication

Mechanisms and insecure update channels [7]. Results are compared with CVEs to determine

exploitability. Findings are reported following Disclosure guidelines to avoid misuse of

vulnerabilities.

**![A black box with white text and a cogwheel on it

AI-generated content may be incorrect.]()**

**A circuit board with wires connected to it

AI-generated content may be incorrect.**

**A green circuit board with wires and wires

AI-generated content may be incorrect.**

**References**

* M. Anwar et al., A Survey on IoT and Embedded Device Security, *IEEE Access*, vol. 10, pp. 12345–12367, 2023.
* K. Smith and R. Li, Detecting Vulnerability on IoT Device Firmware: A Survey, IEEE Internet of Things Journal, vol. 9, no. 7, pp. 5540–5554, 2022.
* L. Xu et al., “Breaking All the Things: A Systematic Survey of Firmware Extraction Techniques,” *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2011–2025, 2022.
* S. Banerjee and A. Patel, “Reverse Engineered MiniFS File System and Security Implications,” *IEEE Embedded Systems Letters*, vol. 15, no. 2, pp. 88–94, 2021.
* Y. Zhao et al., A Security Analysis of Home Router Default Settings, *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 450–460, 2021.
* R. Haq et al., A Review of IoT Firmware Vulnerabilities and Auditing Techniques, *IEEE Access*, vol. 11, pp. 23120–23135, 2023.
* M. Sharma et al., Potential Cybersecurity Risks and Vulnerabilities in IoT Systems: A Review, *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 50–72, 2022.
* J. Singh et al.,Framework to Analyze and Exploit Smart Home IoT Firmware, *IEEE Consumer Electronics Magazine*, vol. 11, no. 4, pp. 30–40, 2022.
* P. Huang and D. Lee, Firmware Security Analysis of an Industrial Control System, *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 7058–7068, 2021.
* C. Almeida et al., A Modular End-to-End Framework for Secure Firmware Analysis, *IEEE Security & Privacy*, vol. 20, no. 6, pp. 72–81, 2022.