COSC 6840
Ethical Hacking Theory & Practice
Final Project

ACME Manufacturing, Inc. Security Analysis

Jonathan Menzel

TABLE OF CONTENTS

1. Introduction

1.1 Client Overview (Fictitious Description)

ACME Manufacturing, Inc., located in Milwaukee, WI, is a manufacturing company specializing in robust IoT hardware. The company has been in business for approximately five years and hired me, the owner of Jon's Ethical Hacking, LLC., to conduct a security assessment for one of their newly designed products, "The Node," before hundreds of them are shipped out to their manufacturing facilities for data collection and analysis.

1.2 Assessment Scope

The scope of this engagement was to perform penetration testing on the supplied devices to determine the likelihood of a nefarious person being able to hack them. That said, the goal was to retrieve stored passwords and flags on the devices and to submit/upload them when discovered.

The assessment included:

- Provided a PowerPoint presentation of all available findings.
- Uploaded all findings to GitHub.
- Used data sheet to aid in hardware connectivity/recon.
- Analyzed devices and executed Linux commands/ran Kali Linux software to decrypt hashed passwords
- Accessed UART shell to submit proof of discovery.
- Provided recommendations on how to strengthen security posture.

2. Methodology

2.1 Tools and Environment

- screen command – Used for UART session to execute shell commands.
- Kali Linux VM – Software used to run John the Ripper.
- John the Ripper – Used to help crack the encrypted password hash
- flashrom – Software tool used to read raw SPI flash contents.
- Linux CLI tools – strings, grep, xxd, hexdump for memory inspection and pattern extraction.
- STM32F103C8T6 Microcontroller (MCU) - "Node" - connected to serial adapter
- ST-Link v2 Debugger
- CH341a connected to "Node" via SPI pins

2.2 Analysis Workflow

I followed a similar workflow for each device that I analyzed. This consisted of accessing the supplied data sheet for the “Node” board. After doing that, I reviewed online resources, such as YouTube to confirm the correct wiring, and then used the UART adapter, ST-Link v2, or CH341A depending on which flag I was trying to retrieve.

3. Findings

3.1 Vulnerability Overview

Vulnerabilities discovered:

- Unauthenticated UART Shell – Anyone with physical access to the device could execute privileged commands. There was no admin credentials required to get into the device.
- Weak Password Hash – The password hash was easily able to be cracked with a standard wordlist using John the Ripper/opensource software.
- Plaintext Secrets – Internal and external flash stored flags that were easily accessible and visible to the user. Again, no admin credentials were required to access these.

3.2 Security Impact Assessment

An unethical attacker could perform the following:

- Access privileged commands without entering admin credentials.
- Extract internal and external secrets that could let them impersonate the device or even clone it.
- Change how the device works by loading their own firmware onto it, which may allow them to hack other devices.
- Pull sensitive operational data from ACME’s manufacturing environment.
- Use Youtube as a guide on how to setup the devices. (I used this for assistance) and the videos were highly detailed and explanations on how to set up the devices.

If ACME Manufacturing, Inc. ships these devices to multiple facilities without first securing them, a hacker could easily develop malware that could spread throughout facilities and may capture network traffic that could further exploitation.

3.3 Evidence and Analysis

• UART Flag – Easily retrieved by running the get_uart_flag command.

• Password Cracking Flag – Obtained easily through the UART shell and taking the password and running John the Ripper wordlist through it. This password was discovered in less than 5 minutes. Huge security flaw!

• Internal Flash Flag – Found plaintext password in internal flash dump

• External Flash Flag – Extracted from W25Q64JV SPI flash dump using CH341A device purchased from Amazon.

4. Scripts

4.1 Purpose

In a different proposal that I conducted for another company two months ago, I developed an automated script to search for vulnerabilities. However, for this assessment, I performed a more targeted approach by executing individual Linux commands that extracted the .bin file and was able to locate the flags.

4.2 Execution Method

For all of the devices, I followed the same execution process, which consisted of a basic recon like researching the hardware online, reading the provided datasheet, and watching YouTube videos on how to correctly connect the pins. After that, I connected the tools (UART adapter, ST-Link v2, CH341A) to the "Node" device and ran individual Linux commands when discovering a new flag.

5. Recommendations

Based on our analysis of the firmware and findings, we recommend the following changes for "quick wins":

5.1 Immediate Security Fixes

• Disable UART Shell – Should only be allowed to access during development by engineers.

• Enable STM32 Readout Protection – Prevents internal flash from being able to be dumped to a file - https://stm32world.com/wiki/STM32_Readout_Protection_(RDP)

• Encrypt External SPI Flash – Store sensitive data with AES or similar encryption. I was able to order the CH341a device from Amazon for $10 (anyone could purchase and do this) and was able to retrieve this flag. Not secure!

• Remove Plaintext Credentials – Secrets should never be stored in readable form.

5.2 Future Enhancements to Strengthen Security Posture

• Integrate GitLab CI/CD – Enterprise version automatically scans for secrets/passwords and notifies if any are found. This will catch vulnerabilities before the code is put into production.

• Conduct Ethical Hacking – Develop an internal cybersecurity team that performs ethical hacking on devices before they are shipped out. There is an upfront cost but could save your reputation/legal fees if devices are hacked.

• Perform Threat Modeling (STRIDE/PASTA) – Develop engineering teams to potential attacks that could occur.

6. Conclusion

Based on my review, I was able to locate several vulnerabilities on the “Node” device. These were found on the microcontroller, ST-Link v2 debugger, and external memory (CH341a). I was able to discover all of the flags except for the one on the Secret Stream Flag. That said, the “Node” device does have vulnerabilities that need to be patched before it is sent out to the manufacturing facilities. The vulnerabilities that were discovered were easy enough for a novice hacker to obtain fairly quickly.

In conclusion, I highly recommend that ACME Manufacturing, Inc. integrates the recommendations that I have outlined in this proposal before shipping the “Node” to any facility. These will help prevent unauthorized access to the device and make it more difficult for someone to retrieve sensitive information. By disabling the UART shell, adding STM32 readout protection, encrypting the external SPI flash, and removing plaintext credentials, ACME can improve the security of the “Node” device. In addition, adding future enhancements like GitLab CI/CD secret scanning, performing routine ethical hacking tests, and using threat modeling will help ensure that issues like this are caught early on. Taking these steps will make the “Node” **much** more secure and better prepared for a production environment.