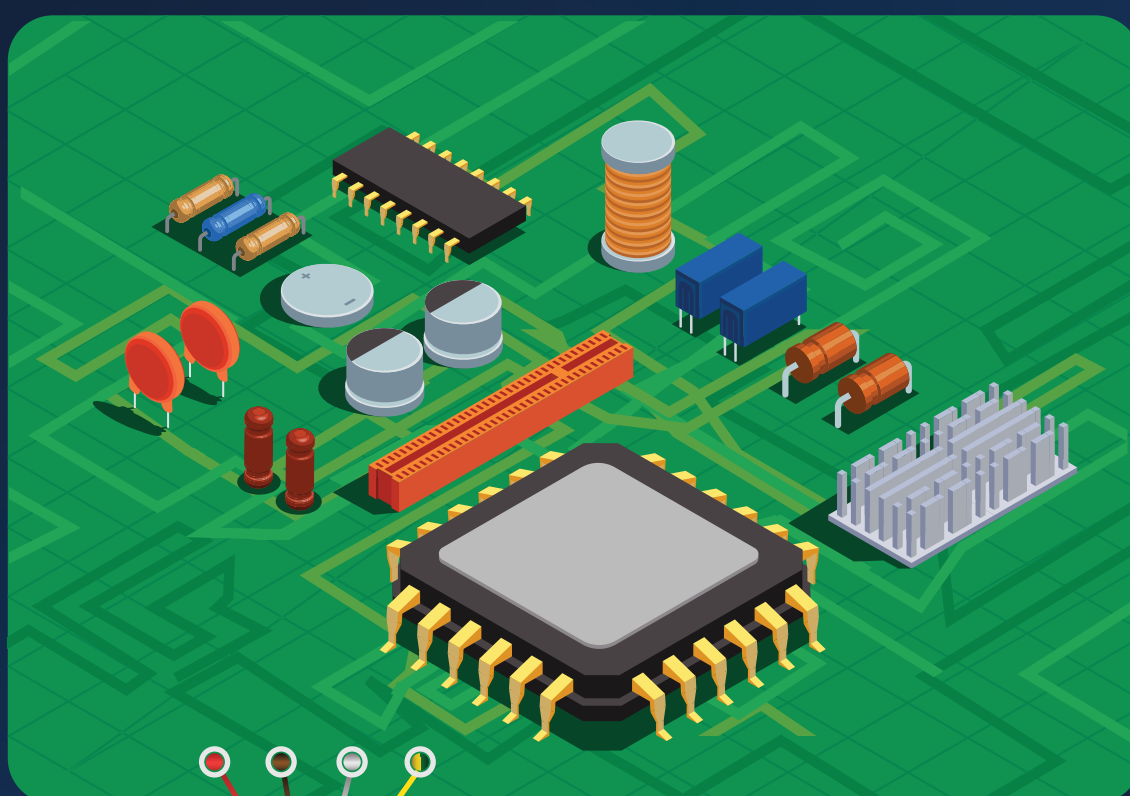




**SECURITYBOAT**  
Frontline Of Your Business

# FIRMWARE EXTRACTION

## HANDBOOK

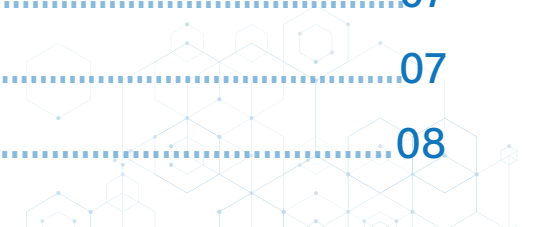


[www.securityboat.net](http://www.securityboat.net)



# Table of Contents

|                                                   |    |
|---------------------------------------------------|----|
| 1. What is firmware.....                          | 01 |
| 2. Types of firmware.....                         | 01 |
| 2.1 Unencrypted Firmware.....                     | 01 |
| 2.2 Encrypted Firmware.....                       | 01 |
| 2.3 Obfuscated Firmware.....                      | 01 |
| 3. What is Firmware Extraction.....               | 02 |
| 3.1 Physical Extraction.....                      | 02 |
| 3.1.1 JTAG Interface.....                         | 02 |
| 3.1.2 UART Interface.....                         | 02 |
| 3.1.3 Chip-Off Technique.....                     | 02 |
| 3.2 Software-Based Extraction.....                | 03 |
| 3.2.1 Firmware Update Mechanisms.....             | 03 |
| 3.2.2 Remote Debugging.....                       | 03 |
| 4.3 Firmware Reverse Engineering.....             | 04 |
| 4.1 Static Analysis.....                          | 04 |
| 4.2 Dynamic Analysis.....                         | 04 |
| 5. Bypass.....                                    | 05 |
| 5.1 Bypass update mechanisms.....                 | 05 |
| 5.2 Bypass serial/debug interface protection..... | 05 |
| 6. Tools.....                                     | 06 |
| 6.1 Binwalk.....                                  | 06 |
| 6.2 Ghidra.....                                   | 06 |
| 6.3 Logic Analyzer.....                           | 06 |
| 7. Impact.....                                    | 06 |
| 8. Prevention.....                                | 06 |
| 8.1 Physical.....                                 | 07 |
| 8.1.1 Removing/Blocking connection.....           | 07 |
| 8.1.2 Chip Encryption.....                        | 07 |
| 8.2 Software.....                                 | 07 |
| 8.2.1 Secure update mechanisms.....               | 07 |
| 9. QR Code.....                                   | 08 |







# 1. What is Firmware

Firmware is a type of software embedded in hardware devices that controls their basic functions and operations, stored in non-volatile memory. It acts as an intermediary between the hardware and higher-level software. For example, firmware in a router manages network connections, while in a printer, it controls printing operations.

## 2. Types of firmware

### 2.1 Unencrypted Firmware

Unencrypted firmware is software code embedded within electronic devices that lacks protection through encryption, meaning its instructions are readable without any cryptographic security.

### 2.2 Encrypted Firmware

Encrypted firmware is a type of software code embedded within electronic devices that is protected through encryption, ensuring its instructions are only accessible to authorized users or processes with the correct decryption key.

### 2.3 Obfuscated Firmware

Obfuscated firmware is a form of software where the code is intentionally obscured or made difficult to understand using various techniques. These might include renaming variables, rearranging code structures, inserting improper instructions, or employing encryption.



Unencrypted Firmware



Encrypted Firmware



Obfuscated Firmware





## 3. What is Firmware Extraction

Firmware extraction involves retrieving the software code embedded in devices EPROM, EEPROM. By examining this code, analysts can uncover how the device operates and discover any potential security weaknesses or vulnerabilities. This process is essential for understanding the device's inner workings and enhancing its security measures.

### 3.1 Physical Extraction

#### 3.1.1 JTAG Interface

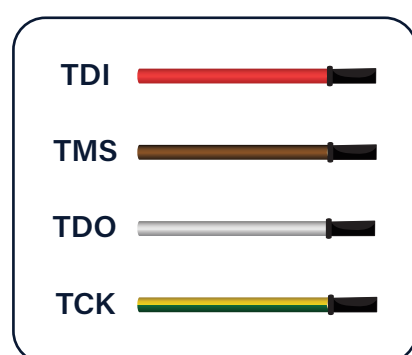
JTAG firmware extraction involves using the Joint Test Action Group (JTAG) interface to access and read the firmware stored on a device's memory. This method requires connecting a JTAG debugger tool to the device's JTAG pins, which facilitates direct communication with the device's processor. Through this connection, the debugger can issue commands to read the firmware data from the device's flash memory, which is then saved and analyzed using specialized software tools.

#### 3.1.2 UART Interface

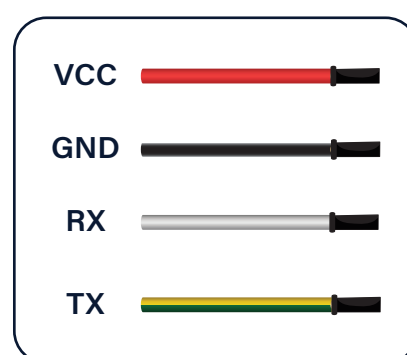
UART firmware extraction involves using the Universal Asynchronous Receiver-Transmitter (UART) interface to access and retrieve firmware from a device. This method requires connecting a UART adapter to the device's UART pins, which are typically used for serial communication. By interfacing with the device through a terminal program, security analysts can send commands to interact with the bootloader or debug console, allowing them to read and save the firmware data for further analysis.

#### 3.1.3 Chip-Off Technique

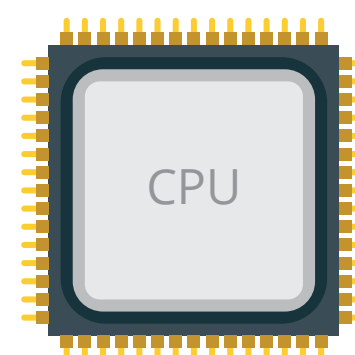
Chip-off firmware extraction involves physically removing the memory chip from a device's circuit board to directly access and read its contents. This process requires desoldering the memory chip from the board and connecting it to a specialized reader or programmer. Once connected, the firmware data can be extracted and analyzed using dedicated software tools. This method bypasses any in-device security measures, providing direct access to the firmware for thorough analysis and vulnerability assessment.



JTAG pinouts



UART pinouts



Chip-Off

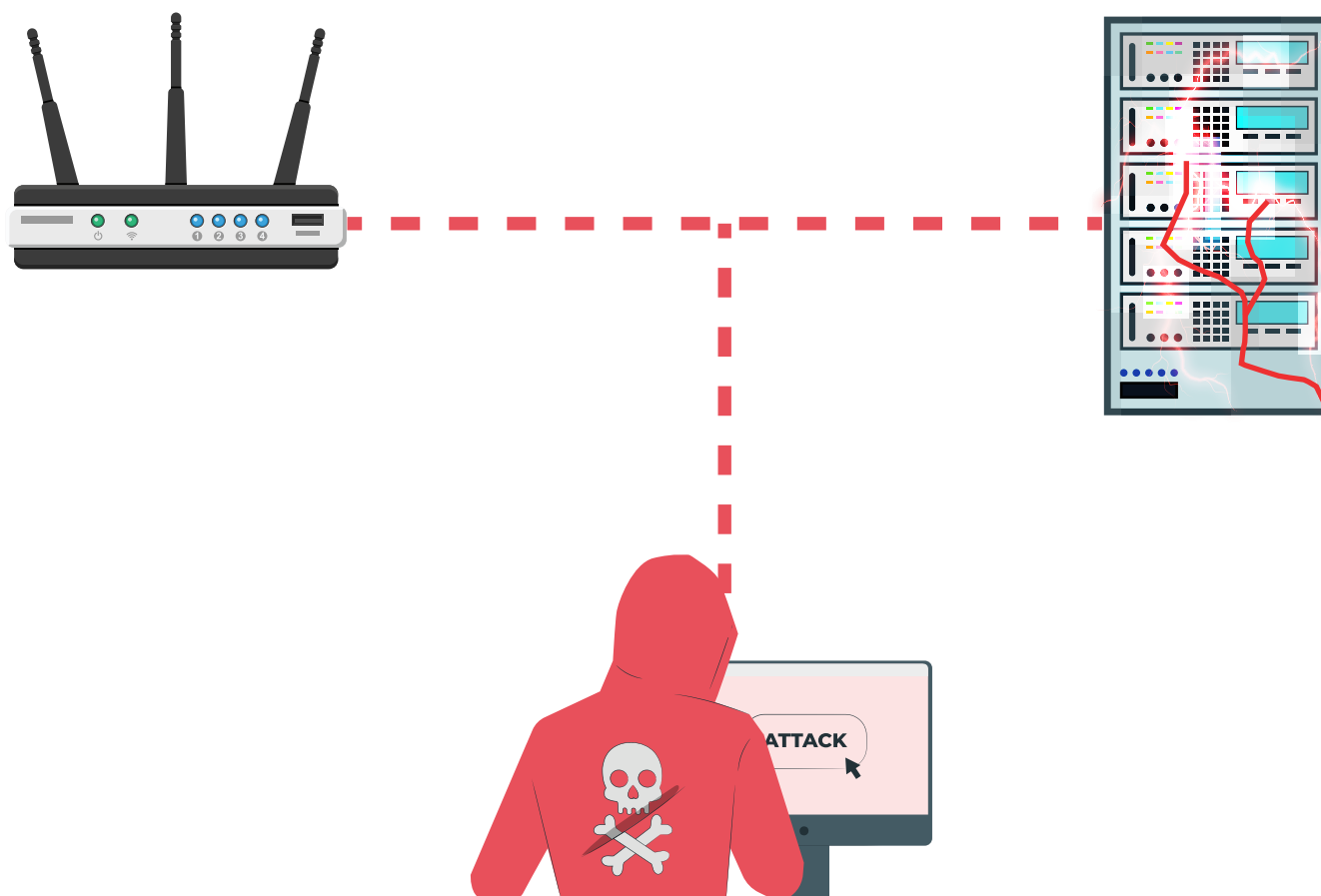




## 3.2 Software-Based Extraction

### 3.2.1 Firmware Update Mechanisms

Firmware update mechanisms involve leveraging the device's built-in firmware update functionality to extract the firmware. This method typically requires intercepting and analyzing the firmware update process, either by capturing firmware update files transmitted over the network or by intercepting the update process during installation on the device. Security analysts can then extract the firmware files from these captured updates and analyze them for vulnerabilities or modifications. However, this approach may require reverse engineering of the firmware update mechanism and careful monitoring of network traffic or device interactions to successfully intercept the firmware update process.



### 3.2.2 Remote Debugging

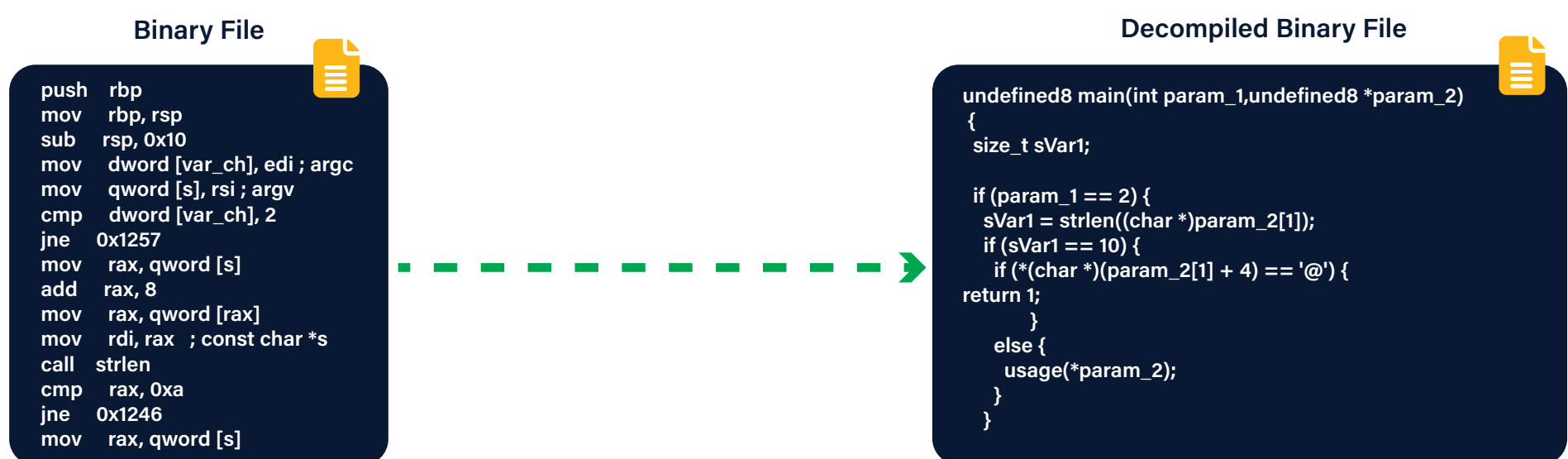
Extraction using remote debugging, such as GDB (GNU Debugger), involves establishing a connection between the debugger tool and the target device over a network. This connection allows security analysts to remotely access and interact with the device's firmware and memory. By leveraging GDB's capabilities, analysts can inspect the device's running firmware, set breakpoints, and examine memory contents in real-time. This method enables the extraction of firmware data without physical access to the device, making it particularly useful for analyzing embedded systems and IoT devices deployed in remote or inaccessible locations.



## 4. Firmware Reverse Engineering

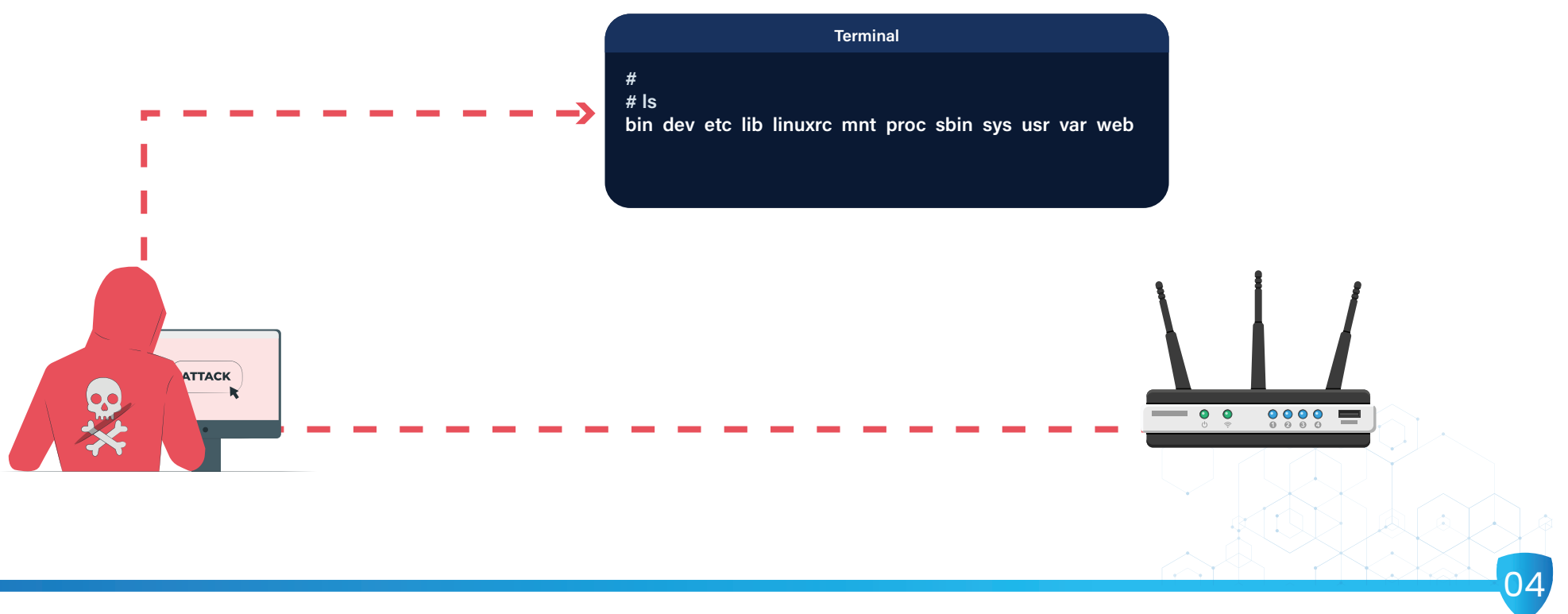
### 4.1 Static Analysis

Firmware reverse engineering through static analysis involves examining the firmware code without executing it, to understand its structure, functionality, and potential vulnerabilities. This process starts by extracting the firmware from the device and then using tools like disassemblers or decompilers to translate the binary file code into a readable format. Analysts then review the code, mapping out functions, data structures, and control flows. Static analysis helps identify issues such as security vulnerabilities, hidden features, or malicious code.



### 4.2 Dynamic Analysis

Firmware reverse engineering through dynamic analysis involves examining the firmware while it is running to observe its behavior and identify vulnerabilities. This method includes loading the firmware onto a device or an emulator and using debugging tools to monitor its execution in real-time. Analysts can set breakpoints, step through the code, and inspect memory and register values to see how the firmware responds to different inputs and conditions.



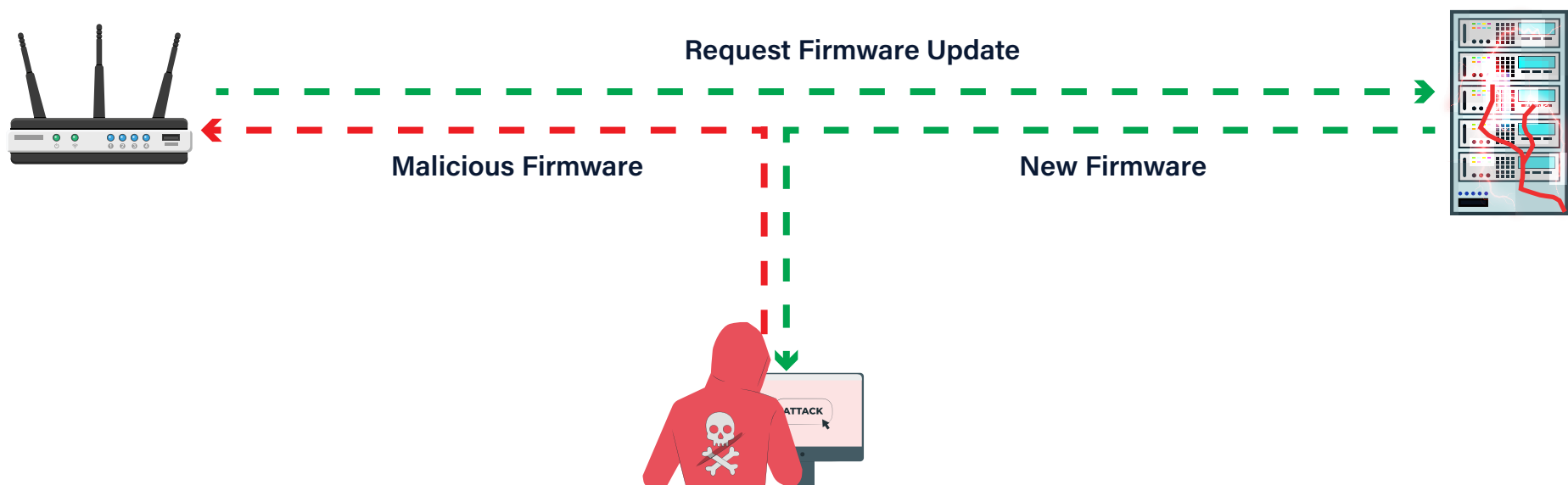


## 5. Bypass

### 5.1 Bypass update mechanisms

Bypassing firmware update mechanisms involves finding ways to go around the standard procedures designed to authenticate and apply firmware updates to a device. This can be accomplished through various methods, which are as follows:

- **Tampering with Update Files:** This method involves intercepting and modifying the update files during transmission or injecting malicious code into the update package. Attackers can manipulate the update content to introduce unauthorized changes or malicious software into the device.
- **Exploiting Vulnerabilities in the Update Process:** Attackers can exploit weaknesses in authentication mechanisms or insufficient validation checks to trick the device into accepting unauthorized firmware updates. This involves identifying and leveraging flaws within the update procedure itself.
- **Targeting Firmware Update Servers or Distribution Channels:** Another approach is to compromise the integrity of legitimate update packages by attacking the firmware update server or distribution channels. This can involve distributing updates or tampering with the legitimate ones to ensure they carry malicious payloads.



### 5.2 Bypass serial/debug interface protection

Bypassing the serial/debug interface protection involves methods to access a device's protected interfaces, such as UART, JTAG, or SWD, which are typically secured to prevent unauthorized access. One common method is glitching, where brief, intentional disturbances (like power or clock glitches) are introduced to the device to momentarily disable security features. Another approach is exploiting software vulnerabilities in the device's firmware that allow bypassing security checks. Physical tampering can also be used, such as soldering directly onto the board to access debug pins or using advanced tools like logic analyzers to intercept and manipulate communication.



## 6. Tools

### 6.1 Binwalk

Binwalk is a tool for analyzing, extracting, and reverse-engineering firmware images and binary files.

🔗 <https://github.com/ReFirmLabs/binwalk>

### 6.2 Ghidra

Ghidra is a software reverse engineering tool developed by the NSA, offering features like disassembly, decompilation, and graphical representation for analyzing binary files.

🔗 <https://github.com/NationalSecurityAgency/ghidra>

### 6.3 Logic Analyzer

A logic analyzer is an electronic tool used to capture and display multiple signals from a digital system, enabling analysis of the system's operation and debugging of hardware issues.

## 7. Impact

- **Security Breaches:** Attackers can find and exploit vulnerabilities in the extracted firmware, leading to unauthorized access and data theft.
- **Intellectual property theft:** Proprietary code can be copied, leading to loss of unique features or competitive edge for the device maker.
- **Malware Insertion:** Extracted firmware can be modified to include malicious code, which can then be reinstalled on devices to compromise their operation.
- **Fake Devices:** Extracted firmware can be used to produce fake versions of the device, harming the original manufacturer's sales and reputation.
- **Device Failure:** Tampered or incorrectly modified firmware can cause devices to malfunction or stop working, affecting user experience and safety.







## 8. Prevention

### 8.1 Physical

#### 8.1.1 Removing/Blocking connection

Preventing firmware extraction by removing connections to serial and debug interfaces involves physically disconnecting or not including these interfaces, such as UART, JTAG, or SWD, on the device's circuit board. These interfaces are commonly used for debugging and can be exploited to access the firmware. By removing or disabling these connections, the direct hardware access needed to extract the firmware is blocked, making it much harder for attackers to retrieve and analyze the firmware. This approach enhances the device's security by eliminating easy entry points for unauthorized firmware extraction.

#### 8.1.2 Chip Encryption

This method involves using encryption directly on the memory chips where the firmware is stored. Technically, this means that the data on the chip is encoded in such a way that it can only be read if you have the correct decryption key. Even if an attacker physically removes the memory chip from the device, they won't be able to access the firmware data without this key. This adds a strong layer of security, making it much harder for unauthorized individuals to extract and understand the firmware.

### 8.2 Software

#### 8.2.1 Secure update mechanisms

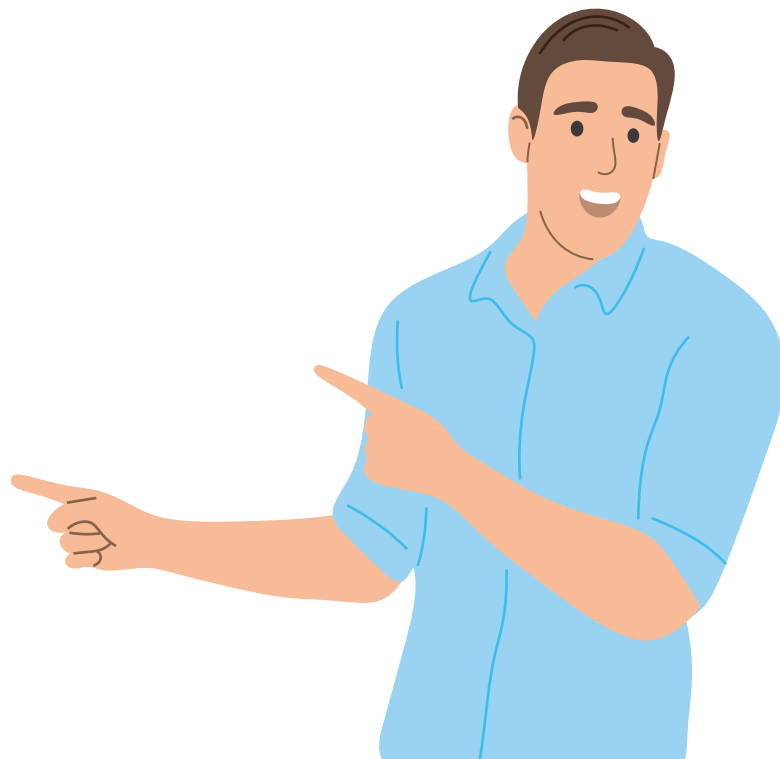
It involves implementing robust procedures to ensure that only authorized firmware updates can be installed on a device. This includes digitally signing firmware updates with cryptographic keys to verify their authenticity before installation. Secure channels, such as encrypted communication protocols, are used to transmit these updates, preventing interception or tampering during transit. Additionally, devices may employ secure boot mechanisms that verify the integrity of the firmware before it is loaded, safeguarding against unauthorized modifications.



**SECURITYBOAT**

Frontline Of Your Business

# FIRMWARE EXTRACTION HANDBOOK



**Scan QR Code to Download Handbook**

[www.securityboat.net](http://www.securityboat.net)