Firmware Extraction Analysis Report

1. Introduction

- **Purpose of the Report**: To analyze the firmware extraction process and findings for the specified camera model.
- **Camera Model**: hi350dromfs (image version)
- **Date of Analysis**: 28-05-2025

2. Methodology

- **Tools Used**:
  - Firmware extraction tools: Binwalk, dd, Python scripts (custom scripts for automation)
  - Analysis tools: Python modules (for automations), os, subprocess modules in Python
  -
- **Extraction Process**:
  - Steps taken to extract the firmware from the device:
    We have made a python automated script by integrating the binwalk tool which once executed, extracts the firmware bin file and reveals important directories like squashFS
  - Description of hardware interfaces used: UART
  - Techniques for accessing firmware: downloaded via web interface.

3. Firmware Overview

- **Firmware Version**: 4.0
- **Size and Structure**:
  - Total size of the firmware image: 22,105,604 bytes for the SquashFS filesystem.
  - File system format: SquashFS, little-endian, version 4.0.
- **Key Components**:
  - Kernel version: uImage header indicates the OS is Linux, CPU architecture is ARM.
  - Libraries and dependencies:

SSL Files:
1. /usr/bin/ssl/pwdreset.pem
2. /usr/bin/secboot/public.pem
3. /usr/data/ssl/ca.key
4. /usr/data/ssl/ca.crt
5. /usr/data/ssl/pubkey.pem
6. /usr/data/ssl/cacert.pem
7. /usr/data/ssl/privkey.pem

Configuration Files:
1. /web/html/iscsiconfig.htm
2. /web/html/serialconfig.htm

3. /web/html/snmpconfig.htm
4. /web/html/upnpconfig.htm
5. /web/html/localconfig.htm
6. /web/html/ptzconfig.htm
7. /web/html/ddnsconfig.htm
8. /web/html/emailconfig.htm
9. /web/js/ptzconfig.js
10. /web/js/upnpconfig.js
11. /web/js/emailconfig.js
12. /web/js/ddnsconfig.js
13. /web/js/iscsiconfig.js
14. /web/js/snmpconfig.js
15. /web/js/serialconfig.js
16. /web/js/localconfig.js
17. /etc/memstat.conf
18. /etc/resolv.conf
19. /etc/udev/udev.conf
20. /sbin/ifconfig
21. /usr/sbin/3gconfig
22. /usr/data/config.lua
23. /usr/data/Data/sharePicture/config_storage0.bmp
24. /usr/data/Data/sharePicture/config_net0.bmp
25. /usr/data/Data/sharePicture/config_image0.bmp
26. /usr/data/Data/sharePicture/config_sys0.bmp
27. /usr/data/Data/sharePicture/config_ptz0.bmp
28. /usr/data/Data/sharePicture/config_md0.bmp
29. /usr/data/Data/sharePicture/config_alarm0.bmp
30. /usr/data/Data/sharePicture/config_ipc0.bmp

4. Findings

- **Vulnerabilities Identified**:
  - List and describe any vulnerabilities found during: Possible buffer overflows or exploitable binaries.
  - Potential impact of each vulnerability:
- **Security Mechanisms**:
  - Description of implemented security features (e.g., secure boot, code signing).
- **Malicious Payloads**:
  - Any potential backdoors or malicious code identified within the firmware: NOT FOUND

5. Code Analysis

- **Static Analysis**:
  - Key functions and their purposes:

    Based on the firmware structure, identify and analyze important functions related to:
    Authentication: Check for any weak or hardcoded credentials in scripts or binaries.
    Network Communications: Analyze functions for data handling or network requests.
    Hardware Interfaces: Look for functions controlling hardware, e.g., JTAG or UART configurations.

  - Analysis of configuration files or scripts:

  Startup Scripts: Check scripts like init or bootloader configurations for security risks.
  Configuration Files: Review for hardcoded IPs, default passwords, or insecure settings.
  Log Management: Check logging configurations to see if sensitive data is exposed.

6. Conclusion

- Summary of key findings and overall assessment of firmware security.

The firmware contains a **SquashFS file system** with potential security risks such as:

Weak authentication or hardcoded credentials (if identified in scripts/configurations).

Lack of robust security mechanisms (e.g., no evidence of secure boot or code signing).

Identified vulnerabilities, including possible buffer overflows or exploitable binaries.

Overall Assessment of Firmware Security:

The firmware demonstrates basic functionality but lacks advanced security features.

The presence of vulnerabilities and limited security mechanisms could make it susceptible to exploitation, requiring updates or patches to enhance security.

7. Appendices

- **Tools and Resources**: Binwalk, dd, and python modules for automation
- **Code Snippets**: Relevant code segments that illustrate findings: The below command helped us to find some important files like:

```
./squashfs-root/usr/data/ssl/ca.crt
./squashfs-root/usr/data/ssl/privkey.pem
./squashfs-root/usr/data/ssl/pubkey.pem
./squashfs-root/usr/data/ssl/ca.key
./squashfs-root/usr/data/ssl/cacert.pem
```

```
./squashfs-root/usr/bin/secboot/public.pem
./squashfs-root/usr/bin/ssl/pwdreset.pem
./squashfs-root/etc/udev/udev.conf
./squashfs-root/etc/memstat.conf
```

find extracted_firmware/ -type f \(-name "*.pem" -o -name "*.crt" -o -name "*.key" -o -name "*.cfg" -o -name "*.conf" \)

- **References**:
- https://fr3ak-hacks.medium.com/analysing-and-extracting-firmware-using-binwalk-982012281ff6
- https://blog.pentesteracademy.com/analyzing-firmware-image-using-binwalk-a6e8277310dc

Additional Considerations

- **Ethical and Legal Compliance**: Ensure that the analysis complies with relevant laws and ethical guidelines regarding reverse engineering and firmware analysis.
- **Confidentiality**: If the analysis involves proprietary or sensitive information, ensure that the report is appropriately redacted or shared under NDA.