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**: HiSilicon Hi3520D-based Camera
- **Date of Analysis**: 28 January 2025

2. Methodology

- **Tools Used**:
  - Firmware extraction tools: Binwalk, UnSquashFS, Sasquatch, dd
  - Analysis tool: Ghidra, Radare2
- **Extraction Process**:
  - Steps taken to extract the firmware from the device.
    1) The firmware image was analyzed using Binwalk to identify its structure, including the uImage header and it was discovered to be a SquashFS filesystem compressed with LZMA which is usual for SquashFS.
    2) UnSquashFS was unable to extract the file system, so Sasquatch was used to extract it.
  - Techniques for accessing firmware: Direct extraction and analysis of the provided binary file using software tools like Binwalk, Ghidra.
  - Some binary files had a lot of embedded files within them such as certificates and databases and had to be extracted using dd, specifying their locations and offsets.

3. Firmware Overview

- **Firmware Version**: 1.02
- **Size and Structure**:
  - Total size of the firmware image: 13,144,064 bytes(~13MB).
  - File system format: SquashFS (version 4.0, LZMA compression).
- **Key Components**:
  - Kernel version: 3.10.0
  - Libraries and dependencies:
    ./libhive_RES.so
    ./libutil-0.9.33.2.so
    ./libdl-0.9.33.2.so
    ./libuClibc-0.9.33.2.so
    ./ld-uClibc-0.9.33.2.so
    ./libthread_db-0.9.33.2.so
    ./libm-0.9.33.2.so
    ./libstdc++.so.6
    ./libgcc_s.so.1
    ./libhive_common.so

./librt-0.9.33.2.so
./libpthread-0.9.33.2.so
./libcrypt-0.9.33.2.so

4. Findings

Vulnerabilities Identified:

Vulnerability	Description	Path	Tools Used
Root Password Exposure [Encrypted with Salted MD5 Hash]	Username: root Password Hash: 1jSqQv.uP$jgz4lwEx2pnDh4QwXkh06 Cracked Password: vizxv	/etc/passwd	John the Ripper, Hashcat
Root Password Exposure [Encrypted with Salted DES]	Username: root Password Hash: ab8nBoH3mb8.g Cracked Password: helpme	/etc/passwd-	John the Ripper, Hashcat
Exposed Telnet Credentials	Username: aditya Password: BXw6K8YB	/usr/etc/telnet_cfg	Binwalk, Sasquatch, Linux OS Commands
Exposed Private Keys	Link	/usr/data/ssl	Binwalk
Hardcoded Credentials	Link	/usr/data/config.lua	Strings
Command Injection	Allows arbitrary command execution, click for more info	/usr/bin/DahuaExec	Ghidra, Qemu-ARM
Web Directory Exposed	Can directly access the web directory with all its contents, and can upload files without any restriction	/web	UnSquashFS, Binwalk, Linux OS Commands
Malicious Scripts	Malicious scripts download and run a base64-encoded daemon to steal /etc/passwd and send it to an external server (203.0.113.25)	/etc/init.d/ /tmp/daemon	Bash, Linux OS Commands
Weak Authentication	Default credentials (admin:admin) found in a commented line within configuration files.	/usr/data/config.lua	Binwalk, Sasquatch

Security Mechanisms:

1. **Secure Boot**:

- Evidence found in `./usr/bin/secboot/public.pem`, suggesting signature verification during boot.
- No bootloader configuration or related binaries were found, requiring further analysis.

2. **TLS/SSL Encryption**:

- TLS/SSL keys and certificates (`ca.key`, `privkey.pem`, `cacert.pem`) were found in `./usr/data/ssl`.
- These are likely used for encrypted communication, but exposure of private keys (`privkey.pem`) could compromise security.

3. **Password and Authentication**:

- The `/etc/passwd` file was found, but further inspection is needed to confirm password hashing.
- Potentially insecure hardcoded credentials (`username=admin &`

`password=admin`) were found in `web/config/index.htm`, posing a security risk.

4. File System Security:

- The `fstab` file indicates that basic filesystems (proc, sysfs, tmpfs) are mounted without encryption.

5. Secure Updates:

- Public key (`public.pem`) in the `secboot` directory indicates potential for firmware signature verification, but no update scripts were found to confirm this.

Malicious Payloads:

1. Malicious Scripts in `/etc/init.d/`

- **Description**: A malicious script found in the **/etc/init.d/** directory is responsible for downloading and executing a base64-encoded daemon. The daemon is designed to steal sensitive files, such as `/etc/passwd`, and exfiltrate them to an external server (IP address `203.0.113.25`).
- **Location**: `/etc/init.d/`
- **Exploitation**: Upon execution, the script runs in the background, sending stolen data (e.g., `/etc/passwd`) to the attacker's server.
- **Tools Used to Expose Vulnerability**: Bash (script execution)

2. Base64-Encoded Daemon in `/tmp/daemon`

- **Description**: The **/tmp/daemon** file is a base64-encoded daemon responsible for stealing system files like `/etc/passwd` and sending them to a remote attacker-controlled server.
- **Location**: `/tmp/daemon`
- **Exploitation**: The executable runs after being triggered by the script, transmitting stolen data to an external server, facilitating unauthorized access and compromise of the system.
- **Tools Used to Expose Vulnerability**: Bash, Binwalk

5. Code Analysis

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

1) BusyBox Init (Called via /sbin/init)

Purpose:

Acts as the first program run during boot. It sets up the environment and then passes control to other programs.

Details:

/sbin/init is just a link to BusyBox (../bin/busybox).

BusyBox is a multi-tool that can act as many different programs (like shell commands).

2) rcS Script (/etc/init.d/rcS)

Purpose:

Runs early during boot to start up the system.

Details:

It mounts all necessary filesystems using /bin/mount -a.

It then goes through and runs other startup scripts in /etc/init.d/ in a set order.

3) Device Setup Scripts (e.g., S00devs, S01udev, S02wndev)

Purpose:

Create necessary device files so that the system can interact with hardware.

Details:

They use the mknod command to create devices like serial ports and input devices.

This ensures the hardware is ready for use later in the boot process.

4) Network Configuration Script (e.g., S80network)

Purpose:

Sets up the network connection.

Details:

Reads network settings from /proc/cmdline (like IP address, gateway, etc.).

Configures the network interface with commands like ifconfig and route.

Also starts a basic service like telnetd for remote access.

5) Service and Environment Scripts (e.g., S81toe, S99dh)

Purpose:

Set up additional services and tune system settings.

Details:

One script downloads a daemon from a remote location and prepares it for running.

Other scripts mount extra filesystems (like JFFS2 or cramfs) and adjust kernel settings (like memory usage).

They help get all parts of the system working correctly before normal operation begins.

- Analysis of configuration files or scripts:After analyzing we found :
  1) Presence of Hardcoded Credentials
  2) Insecure File Permissions
  3) Unsecure Features
  4) Lack of Error Handling
  5) Redundant Configuration
  6) Absence of Encryption

Examples:

This script `/etc/init.d/S02wndev` attempts to download a file named daemon from an external IP address (203.0.113.25) and moves it to /tmp/.

This script `/etc/init.d/S80network` parses network configuration parameters from /proc/cmdline. It configures the network interface, enables Telnet (telnetd), and executes a hidden decryption script (/usr/sbin/.dec.sh) on the downloaded service daemon.

This script `/etc/init.d/S99dh` mounts multiple partitions, including JFFS2, CRAMFS, and RAMFS filesystems, and extracts sensitive files using p7zip. It also sets up PPPoE configuration files and starts network services like net3g.

The configuration file `/usr/data/config.lua` exposes 4 username and password pairs along with a variable `Secret_Resetpwd_Pubver`, which is presumably used to reset the password.

6. Conclusion

- Summary of key findings and overall assessment of firmware security.
- **Firmware Structure and Content**
  - The firmware contains a variety of file types, including configuration files, script files, and UI resources.

    - Extraction tools such as `binwalk` and `strings` have provided insight into the structure, revealing hidden archives, cryptographic algorithms, and sensitive information.
- **Cryptographic Algorithms and Keys**
    - Several cryptographic algorithms such as AES, RSA, and MD5 have been identified, indicating the presence of encryption mechanisms.
    - Potential keys and tokens related to cryptographic processes or authentication were detected.
- **Potential Passwords**
    - The analysis has identified numerous potential password-related strings (e.g., "password", "passwd", "api_key").
    - Several confirmed passwords were extracted in cleartext and others in hashed format, which indicates areas for concern.
- **Sensitive Files and Configuration**
    - Configuration files related to system setup, SSL keys, and scripts were found. These include sensitive data like admin credentials, private keys and password reset keys.
- **Security Vulnerabilities**
    - Common binaries such as `wget`, `curl`, and `openssl` were found in the extracted content.

---

7. Appendices

- **Tools and Resources**: List of tools used in the analysis and links to documentation.

- Binwalk: https://github.com/ReFirmLabs/binwalk
- Python: https://www.python.org/
- Sasquatch: https://github.com/devttys0/sasquatch
- Ghidra: https://ghidra-sre.org/
- Readelf: https://github.com/ghosind/readelf
- Qemu: https://github.com/qemu/qemu
- GDB: https://github.com/bminor/binutils-gdb
- Linux Operating System Commands: https://www.linux.org/

- **Code Snippets:** Relevant code segments that illustrate findings.

Function to find the most potential passwords from our python script.

```
def find_potential_passwords(directory):
        regex_pattern = r'(?i)(?:passwd|pwd|password)\s*=\s*"?([^"\s]+)"?\s*(?:,|$)'
        passwords_with_paths = []

        for root, _, files in os.walk(directory):
        for file in files:
```

```
        file_path = os.path.join(root, file)
        try:
                with open(file_path, 'r', encoding='utf-8', errors='ignore') as f:
                content = f.read()
                matches = re.findall(regex_pattern, content)

                for pwd in matches:
                relative_path = os.path.relpath(file_path, directory)
                passwords_with_paths.append((pwd, relative_path))

        except Exception as e:
                print(f"Error reading {file_path}: {e}")

        # Regex to filter out common placeholders or junk
        valid_pwd_pattern = r'^(?!.*[%\*\{\}\(\)\[\]])([a-zA-Z0-9]{4,12})$'

        # Filter out non-sensitive passwords
        sensitive_passwords = [
        (pwd, path) for pwd, path in passwords_with_paths
        if re.match(valid_pwd_pattern, str(pwd)) and pwd not in ['******', '%s', '%d']
        ]

        return sensitive_passwords
```

Exposed Credentials

Telnet credentials found in cleartext in `/usr/etc/telnet_cfg`

username=aditya
passwd=BXw6K8YB

Leaked credentials in the program /usr/data/config.lua

```
    -- group
      INI_GROUP_NAME_ADMIN          = "admin";
      INI_GROUP_NAME_USER           = "user";
      -- user
      INI_SYS_USER_ADMIN            = "admin",
      INI_SYS_USER_ADMIN_PWD        = "admin",
      INI_SYS_USER_LOCAL            = "888888",
      INI_SYS_USER_LOCAL_PWD        = "888888",
```

```
INI_USER_USER_LOCAL            = "666666",
INI_USER_USER_LOCAL_PWD        = "666666",
INI_DEFAULT_USER_NAME          = "default",
INI_DEFAULT_USER_PWD           = "tluafed",
```

A comment in /web/config/index.htm reveals the username, password

```
//兼容明文username=admin&password=admin
which translates to
//Compatible with plain text username=admin&password=admin
```

- **References**:

- Dahua Documentation: https://dahuawiki.com/
- OSDev Wiki: https://wiki.osdev.org/
- OWASP IoT Security Guidelines: https://owasp.org/
- OpenAI - ChatGPT: https://chat.com/
- Exploit Database: https://www.exploit-db.com/
- IBM Documentation:https://www.ibm.com/docs