Firmware Extraction Analysis Report

1. Introduction

- **Purpose of the Report**: The Firmware Analysis Report aims to analyze the firmware extraction process for a specified camera model. It includes metadata, extracted files, and other relevant findings to assess the firmware's structure, security, and potential vulnerabilities.
- **Camera Model**: hi3520Dromfs (Image Firmware Name)
- **Date of Analysis**: 28-01-2025

2. Methodology

- **Tools Used**:
  - Firmware extraction tools: Binwalk
  - Analysis tools: Binwalk, numpy, chardet

• **Extraction Process**:
  - Steps taken to extract the firmware from the device.

Save Script and bin file in one directory execute script and then give a bin file path.

  - Description of hardware interfaces used:UART
  - Techniques for accessing firmware: Downloading via web interface.

3. Firmware Overview

- **Firmware Version**: 4.0
- **Size and Structure**:
  - Total size of the firmware image: 22105604 bytes, 21.0 MB.
  - File system format: SquashFS, little-endian, filesystem.
- **Key Components**:
  - Kernel version: uIage 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`
31. `/usr/data/Data/sharePicture/config_chnname0.bmp`
32. `/usr/data/Data/sharePicture/config_sysinfo0.bmp`
33. `/usr/data/Data/mainPage/config1.bmp`
34. `/usr/data/Data/mainPage/config2.bmp`
35. `/usr/data/Data/player/ivs_config.png`
36. `/usr/data/Data/mainMenu/config_guide_normal.png`
37. `/usr/data/Data/title48/config1.bmp`
38. `/usr/data/Data/title48/config2.bmp`

4. Findings

Script Files:

1. `/usr/etc/load_modules.sh`

2. `/usr/sbin/.dec.sh`
3. `/usr/lib/lib.7z.extracted/0/lib/pinmux_hi3521a_vicap.sh`
4. `/usr/lib/lib.7z.extracted/0/lib/pinmux_hi3521a_i2s.sh`
5. `/usr/lib/lib.7z.extracted/0/lib/pinmux_hi3521a_vga_hdmi_spi.sh`
6. `/usr/lib/lib.7z.extracted/0/lib/sysctl_hi3521a_asic.sh`
7. `/usr/lib/lib.7z.extracted/0/lib/crgctrl_hi3521a.sh`
8. `/usr/lib/lib.7z.extracted/0/lib/load_hisimod.sh`

Top 10 **Potential Passwords**:
'HenZii', ' 'BXw6K8YB', 'H6zjSw', 'O1Nmuf', 'Oh11OPj', 'QFxjunqO', 'zbKrAh', 'DQMpuxv', 'nz_3KH', 'OvW5K3'

5. Code Analysis

- **Static Analysis**:
- The provided Python script contains a range of functions for analyzing firmware files, extracting information, and generating reports. Here's a detailed analysis of the code, with a focus on static analysis, key functions, and configuration files:
- **Static Analysis:**
- **1. Key Functions and Their Purposes:**

get_file_size(file_path): Returns the size of a file in bytes.
get_md5_hash(file_path): Computes the MD5 hash of the file for integrity checking.
get_file_format(file_path): Determines the file format (ELF, PE, or BIN) by reading the first few bytes of the file.
find_urls(file_path): Extracts URLs from the file by searching for strings that match the pattern of a URL.
find_ip_addresses(file_path): Extracts IP addresses by searching for patterns that match IP formats (IPv4 and IPv6).
get_packing_info(file_path): Uses binwalk to extract packed information from the firmware binary.
calculate_entropy(file_path): Calculates the entropy (a measure of randomness) of the file. High entropy indicates compressed or packed data.
analyze_entropy(file_path): Determines whether a file is "packed" or "unpacked" based on its entropy value.
get_file_metadata(file_path): Runs the file command to gather metadata about the file (like type, architecture, etc.).
extract_architecture(metadata): Extracts the system architecture from the metadata (e.g., x86_64, ARM).
get_ui_resources(file_path): Detects UI resources such as buttons, icons, and text by scanning for certain keywords in the file's strings.
get_cryptographic_algorithms(file_path): Identifies cryptographic algorithms (e.g., AES, RSA) used in the file.

get_top_10_passwords(file_path): Finds the most likely password candidates by searching for common password patterns in the file's strings.
process_extracted_files(extracted_dir): Processes extracted files from a firmware image, detecting SSL certificates, configuration files, scripts, and other metadata like URLs, emails, and IP addresses.
write_output_to_file(output_file, file_info, ssl_files, config_files, script_files, bin_files, urls, emails, ip_addresses): Writes the analysis results (including metadata and extracted information) to a text file.
extract_firmware(firmware_path): Uses binwalk to extract the contents of a firmware image into a specified directory.
generate_file_tree(extracted_dir): Uses the tree command to generate a file structure of the extracted firmware contents.
analyze_files_accessed(extracted_dir): Analyzes the creation and modification times of files in the extracted directory.

2. Configuration Files or Scripts Analysis:
Configuration Files: Files like config.json, config.yaml, .conf, settings, and config.txt are detected and processed.
Script Files: Files with extensions such as .sh, .py, .pl, .cgi are identified as potential scripts and logged.
SSL Files: Files with extensions like .pem, .crt, .key are classified as SSL-related files.
The process_extracted_files function handles these detections by walking through the extracted directory and categorizing files accordingly.
Key Outputs:
The script generates detailed reports containing:
Basic information about the firmware file (size, hash, format, metadata).
Extracted information, including SSL certificates, configuration files, scripts, and any URLs, emails, and IP addresses found.
A detailed file tree structure of the extracted files.
Analysis of file access times (creation and modification).
The reports are saved as text files (e.g., firmware_analysis_report.txt) and a markdown file (e.g., filesystem.md).

- Analysis of configuration files or scripts

  Configuration Files Analysis:
  In the script, configuration files are processed by the process_extracted_files function. The function detects configuration files based on their names or extensions, such as config.json,

settings, config.txt, config.yaml, and .conf. These files are then added to the config_files list, which is later included in the output report.

Script Files Analysis:
Script files are handled in a similar manner. The script identifies files with extensions such as .sh, .py, .pl, and .cgi. These files are collected in the script_files list and also written to the final report.

6. Conclusion

The firmware uses shell scripts and configuration files to manage system initialization, hardware setup, and network communication. Scripts like `pinmux_hi3521a_*` configure the Hi3521a chipset, while `/usr/etc/load_modules.sh` loads kernel modules. Networking is handled by `ifconfig` and `3gconfig`, and SSL files enable secure communication. Web-based management is supported through JavaScript and HTML files. These scripts ensure secure, flexible, and efficient operation of the device.

Additional Considerations

- **Tools and Resources:**

1. binwalk – A tool for analyzing and extracting firmware images.
2. numpy – A popular library for numerical computing in Python.
3. chardet – A library used for character encoding detection.

The rest, like os, hashlib, re, subprocess, collections.Counter, math, pwd, grp, and datetime, are part of Python's standard library and should be pre-installed.

- **Code Snippets:**

1. Using binwalk to extract packing information:

```
def get_packing_info(file_path):
    binwalk_output = subprocess.run(['binwalk', '--extract', file_path], capture_output=True, text=True)
    return binwalk_output.stdout
```

2. Using `hashlib` to calculate MD5 hash:

```
def get_md5_hash(file_path):
    hash_md5 = hashlib.md5()
    with open(file_path, "rb") as f:
        for chunk in iter(lambda: f.read(4096), b""):
            hash_md5.update(chunk)
    return hash_md5.hexdigest()
```

3. Using `re` for extracting URLs:

```
def find_urls(file_path):
    with open(file_path, 'rb') as f:
        data = f.read()
    urls = re.findall(r'(https?://[^\s]+)', data.decode(errors='ignore'))
    return list(set(urls))
```

4. Using `subprocess` to execute shell commands:

```
def get_file_metadata(file_path):
    try:
        file_output = subprocess.check_output(['file', file_path]).decode(errors='ignore')
        return file_output.strip()  # Return the output as a string
    except subprocess.CalledProcessError as e:
        return f"Error retrieving metadata: {e}"
```

5. Using `collections.Counter` to count most common passwords:

```
def get_top_10_passwords(file_path):
    passwords = []
    with open(file_path, 'rb') as f:
        strings = subprocess.check_output(['strings', file_path]).decode(errors='ignore')
    passwords = re.findall(r'(\w{6,})', strings)
    return [item[0] for item in Counter(passwords).most_common(10)]
```

6. Using `os` for file size:

```
def get_file_size(file_path):
    return os.path.getsize(file_path)
```


References:

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