FIRMWARE ANALYSIS

BlackBox Data Exfiltration

PREPARED BY

Kolby MacDonald

Information Systems Security

PREPARED ON

25 / 04 / 2023

Project Overview

1. Background

Directly sourced from an unknown microprocessor, the project's core objective revolves around two key points. Initially, our objective was to efficiently extract relevant data. Subsequently, this data forms the basis for constructing a detailed profile of the device. However, the project's overal focus is to test the security posture of the device. This involves a comprehensive evaluation of the device's security practices, essentially gauging its strengths and vulnerabilities.

2. Goals

1. Retrieve and Reconstruct Binaries.
2. Extract stored file information.
3. Determine the devices architecture.
4. Entropy analysis.
5. Compression analysis.
6. Determine firmware versions.
7. Highlight relevant information.

3. Intended Outcome

The main goal of this project is two-fold. First, it aims to uncover the identity and purpose of the device in question—essentially, what it is and what it does. Second, it's all about determining how secure the device is. This involves a thorough assessment to gauge its resiliency to such data exfiltration.

Exfiltration Process

1. Tools & Descriptions

1. Binwalk: General data extraction tool for binaries. Great for quick analysis of firmware and generating entropy analysis views.

2. Ghidra: Open-source software reverse engineering tool. Usefully for its string extraction and ability to analyse samples without running their assembly code. Ghidra also has the ability to reconstruct certain program files.

2. Process Design

1. Reconstruction of partitions from binaries.
2. Manual partition viewing for poorly hidden data.
3. Binwalk entropy analysis graphing.
4. Ghidra binary analysis & String Viewing.
5. Notes and remarks.

Relevant Results

1. Reconstruction of Partitions

1. Given the binary:

   ```
   Firmware.bin
   ```

2. Run: Binwalk -e {filename}.bin to reconstruct useful binaries.
3. Change directory into:

   ```
   _Firmware.bin.extracted
   ```

2. Manual Data Extraction

1. View reconstructed files:

   ```
   └─$ ls
   17F5D4  17F5D4.7z  2A470.squashfs  2CD81C.squashfs  62A840.jffs2  jffs2-root  squashfs-root  squashfs-root-0
   ```

2. Now the squashfs-root folder will container has the following:

   ```
   └─$ ls
   app  base  homever  lib  rt
   ```

3. And for example, the app file has the following contents:

   ```
   └─$ ls
   aacplay   audio_file  cloud     crypt_file  init.sh      localbin  locallib    mp4record  p2p_tnp    rmm     upgrade_firmware  wp_cmd
   arp_test  ca          cloudAPI  dispatch    init.sh.bak  localko   log_server  oss        recbackup  script  watch_process
   ```

4. For every reconstructed files, I went through and collected all relevant plaintext information:
   a. Users and Passwords

```
root::0:0:root:/:/bin/sh
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
```

```
"0123456789ABCDEF"
```

```
1 A:*
2 /secret:tonywu:123qwe
```

b. Last stored IP location:

```
3 127.0.0.1        localhost.localdomain   localhost
```

c. CA certificate:

```
$ cat ca.crt
-----BEGIN CERTIFICATE-----
```

MIIDdTCCAl2gAwIBAgILBAAAAAABFUtaw5QwDQYJKoZIhvcNAQEFBQAwVzELMAkG
A1UEBhMCQkUxGTAXBgNVBAoTEEdsb2JhbFNpZ24gbnYtc2ExEDAOBgNVBAsTB1Jv
b3QgQ0ExGzAZBgNVBAMTEkdsb2JhbFNpZ24gUm9vdCBDQTAeFw05ODA5MDExMjAw
MDBaFw0yODAxMjgxMjAwMDBaMFcxCzAJBgNVBAYTAkJFMRkwFwYDVQQKExBHbG9i
YWxTaWduIG52LXNhMRAwDgYDVQQLEwdSb290IENBMRswGQYDVQQDExJHbG9iYWxT
aWduIFJvb3QgQ0EwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDaDuaZ
jc6j40+Kfvvxi4Mla+pIH/EqsLmVEQS98GPR4mdmzxzdzxtIK+6NiY6arymAZavp
xy0Sy6scTHAHoT0KMM0VjU/43dSMUBUc71DuxC73/OlS8pF94G3VNTCOXkNz8kHp
1Wrjsok6Vjk4bwY8iGlbKk3Fp1S4bInMm/k8yuX9ifUSPJJ4ltbcdG6TRGHRjcdG
snUOhugZitVtbNV4FpWi6cgKOOvyJBNPc1STE4U6G7weNLWLBYy5d4ux2x8gkasJ
U26Qzns3dLlwR5EiUWMWea6xrkEmCMgZK9FGqkjWZCrXgzT/LCrBbBlDSgeF59N8
9iFo7+ryUp9/k5DPAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNVHRMBAf8E
BTADAQH/MB0GA1UdDgQWBBRge2YaRQ2XyolQL30EzTSo//z9SzANBgkqhkiG9w0B
AQUFAAOCAQEA1nPnfE920I2/7LqivjTFKDK1fPxsnCwrvQmeU79rXqoRSLblCKOz
yj1hTdNGCbM+w6DjY1Ub8rrvrTnhQ7k4o+YviiY776BQVvnGCv04zcQLcFGUl5gE
38NflNUVyRRBnMRddWQVDf9VMOyGj/8N7yy5Y0b2qvzfvGn9LhJIZJrglfCm7ymP
AbEVtQwdpf5pLGkkeB6zpxxxYu7KyJesF12KwvhHhm4qxFYxldBniYUr+WymXUad
DKqC5JlR3XC321Y9YeRq4VzW9v493kHMB65jUr9TU/Qr6cf9tveCX4XSQRjbgbME
HMUfpIBvFSDJ3gyICh3WZlXi/EjJKSZp4A==

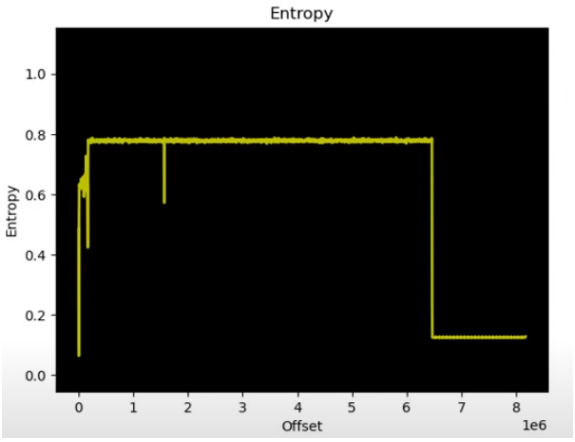
```
-----END CERTIFICATE-----
```

d. A potential device name:

```
/xiaoyicamera/
```

e. A potential internal script banner:

3. Binwalk Entropy Analysis Graphing:



Notes: Likely encrypted and compressed.

Using binwalk we can also see:

```
173264        0x2A4D0         xz compressed data
261764        0x3FE84         xz compressed data
```

```
1570260       0x17F5D4        LZMA compressed data,
```

3. Binary Analysis:

1. Using Ghidra, the following information was extracted:
    a. Board Name:

```
"spi nand boot mode\n"
"emmc boot mode\n"
"Board: IPCAM RTS3903 CPU: %dM :"
"unknown"
" prid=0x%x\n"
```

    b. String and time formatting for internal scripts:

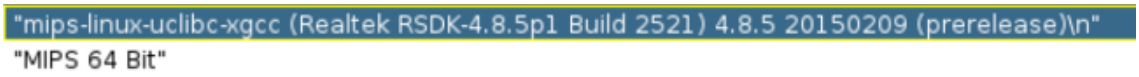
```
"%4d-%02d-%02d  %2d:%02d:%02d UTC\n"
```

    c. Firmware Version:

```
7.0.00.79A_201903291120
```

d. Cloud camera support:

e. System architecture:



3. Notes and Remarks

1. This device is an IP Camera with cloud support.
2. It's name is the Xiaoyicamera and its running on MIPS 64 Bit.
3. It's encoded and partially encrypted.
4. Furthermore, it has cloud support and stores important usernames/passwords/ca certs and more in unencrypted files around the device.
5. Googling the IPCAM board seems to suggest the camera has night vision capabilities.
6. Xiaoyicamera's have remote access through their app.
7. Another report of someone completing similar attacks on similar devices has been conducted in the past with examples of other exploits as well https://arxiv.org/pdf/2201.07462.pdf.

Conclusion

The Xiaoyicamera, an IP Camera with cloud support, poses significant vulnerabilities to remote access attacks due to several critical factors. The device's MIPS 64 Bit architecture, while powerful, can be exploited if proper security measures aren't in place. The encoding and partial encryption of its binaries might deter some attackers, but determined adversaries can still reverse-engineer the code to find vulnerabilities. The presence of unencrypted files on the device containing sensitive information like usernames, passwords, and CA certificates creates a potential goldmine for attackers. With a more comprehensive attack, potentially using enma, a large amount og, if not all sensitive information could likely be extract. These vulnerabilities are compounded by the device's cloud support, which can expose the stored information to external threats. The reported accessibility of night vision capabilities raises concerns as attackers could exploit these features to evade detection during unauthorized access. The documented history of successful attacks on similar devices, as detailed in the following report (https://arxiv.org/pdf/2201.07462.pdf), emphasizes the urgent need for remediation.