|  |
| --- |
| Close-up image showing the leaf-sides of two oversized books side-by-side on a bookshelf, with additional books in soft focus background |
| IOT REDBOOK  FIRMWARE ANALYSIS |
| |  |  | | --- | --- | | NAVNEET MISHRA, JACOB VICTOR |  | |

Table of Contents

[Firmware 2](#_Toc534809108)

[Firmwareanalysis 2](#_Toc534809109)

[How to get Firmware 3](#_Toc534809110)

[Extracting Filesystem 3](#_Toc534809111)

[Manual Process 3](#_Toc534809112)

[Common file system and magic bytes 5](#_Toc534809113)

[Automatic Process 5](#_Toc534809114)

[Binwalk 5](#_Toc534809115)

[Firmware Mod Kit 6](#_Toc534809116)

[Analyzing Filesystem 7](#_Toc534809117)

[Manual Process 7](#_Toc534809118)

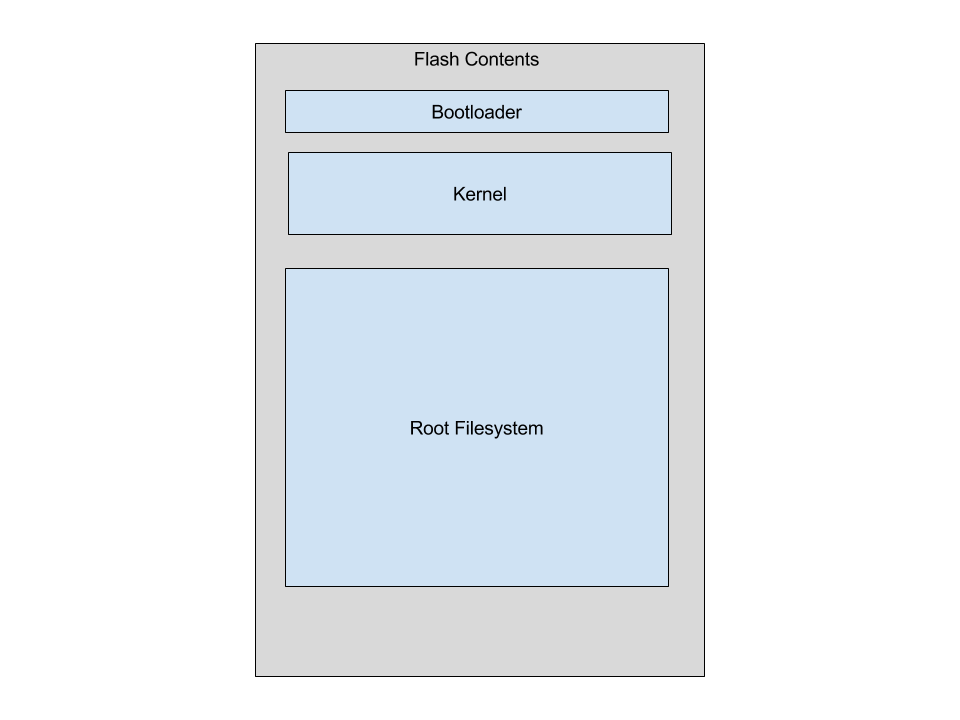
[Automatic approach 8](#_Toc534809119)

[Reference 9](#_Toc534809120)

# Firmware

Firmware is a software program or set of instructions programmed on a hardware device. It provides the necessary instructions for how the device communicates with the other computer hardware. The firmware contains low level programming code that enables software to access hardware functions. Devices that run firmware are known as embedded systems which have limited hardware resources, such as storage capabilities as well as memory. Examples of embedded devices that run firmware are smartphones, traffic lights, connected vehicles, some types of computers, drones, and cable set-top boxes. Firmware is typically stored in the flash ROM of a hardware device. While ROM is "read-only memory," flash ROM can be erased and rewritten because it is actually a type of flash memory. Unlike normal software, firmware cannot be changed or deleted by an end-user without the aid of special programs and remains on that device whether or not it's on or off.

In simple terms, Firmware = flash contents, the bootloader, the kernel, and a root filesystem.



# Firmwareanalysis

Firmware is the center of controlling IoT devices, which is why we may want to start analyzing its contents before other pieces of the device's components. Depending on the industry your IoT device is manufactured for, obtaining a firmware image and disassembling its contents may be trivial. Similarly, some industry verticals require certain safeguards that may make reverse engineering more difficult and/or time-consuming.

What to search for in Firmware

* Filesystem
* Custom user space binaries
* Web code
* Hardcoded credentials
* Configuration files
* Private Certificates, keys
* Backdoor accounts
* Vulnerable services
* API Keys, tokens
* API endpoints (URLs)
* Source code
* How data is stored

# How to get Firmware

There are several methods to obtain firmware from an IoT device. Firmware images can be obtained via the following approaches:

* Download from the vendor's website ([read more](https://learning.oreilly.com/library/view/iot-penetration-testing/9781787280571/ef613ada-1e88-48e8-b28d-70beb95fad21.xhtml))
* Proxy or mirror traffic during device updates ([read more](https://learning.oreilly.com/library/view/iot-penetration-testing/9781787280571/f56d34f1-a279-46f2-8afb-39d4bc2617cc.xhtml))
* Dump firmware directly from the device Googling/researching ([read more](https://learning.oreilly.com/library/view/iot-penetration-testing/9781787280571/1ac59502-2a43-4625-abae-d3e4710a88b5.xhtml))
* Decompile associated mobile apps ([read more](https://learning.oreilly.com/library/view/iot-penetration-testing/9781787280571/7544cc30-b272-4321-bf8d-af5fae6d523d.xhtml))

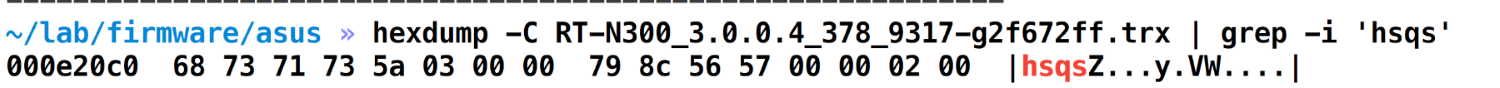
Once we have the firmware with us, the main step now is to analyze the firmware. This involves looking inside the firmware to identify as many security issues as possible. Firmware is composed of a bootloader, kernel, filesystem, and additional resources. The most interesting part, however, is the filesystem. Firmware, is a binary file package and the filesystem is just one of the components which could be stored at a specific offset in the binary and with a specific size.

# Extracting Filesystem

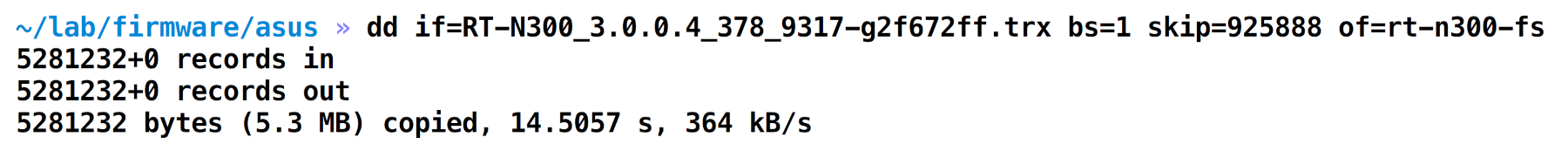
## Manual Process

Find out the filesystem of firmware.

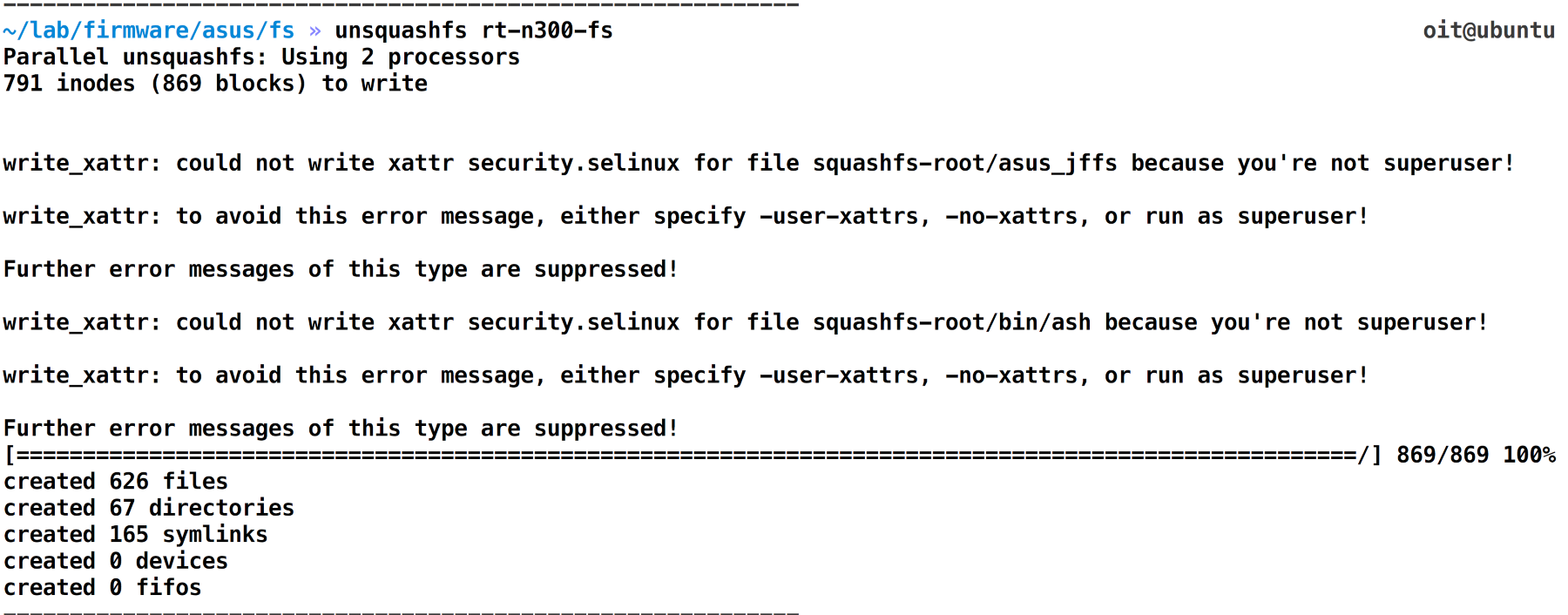
1. If we want to look for the Squashfs filesystem, we can grep the hexdump output for shsq (which is the magic byte for any Squashfs filesystem) in reverse order as follows. Similarly, we can grep magic byte for other filesystem.



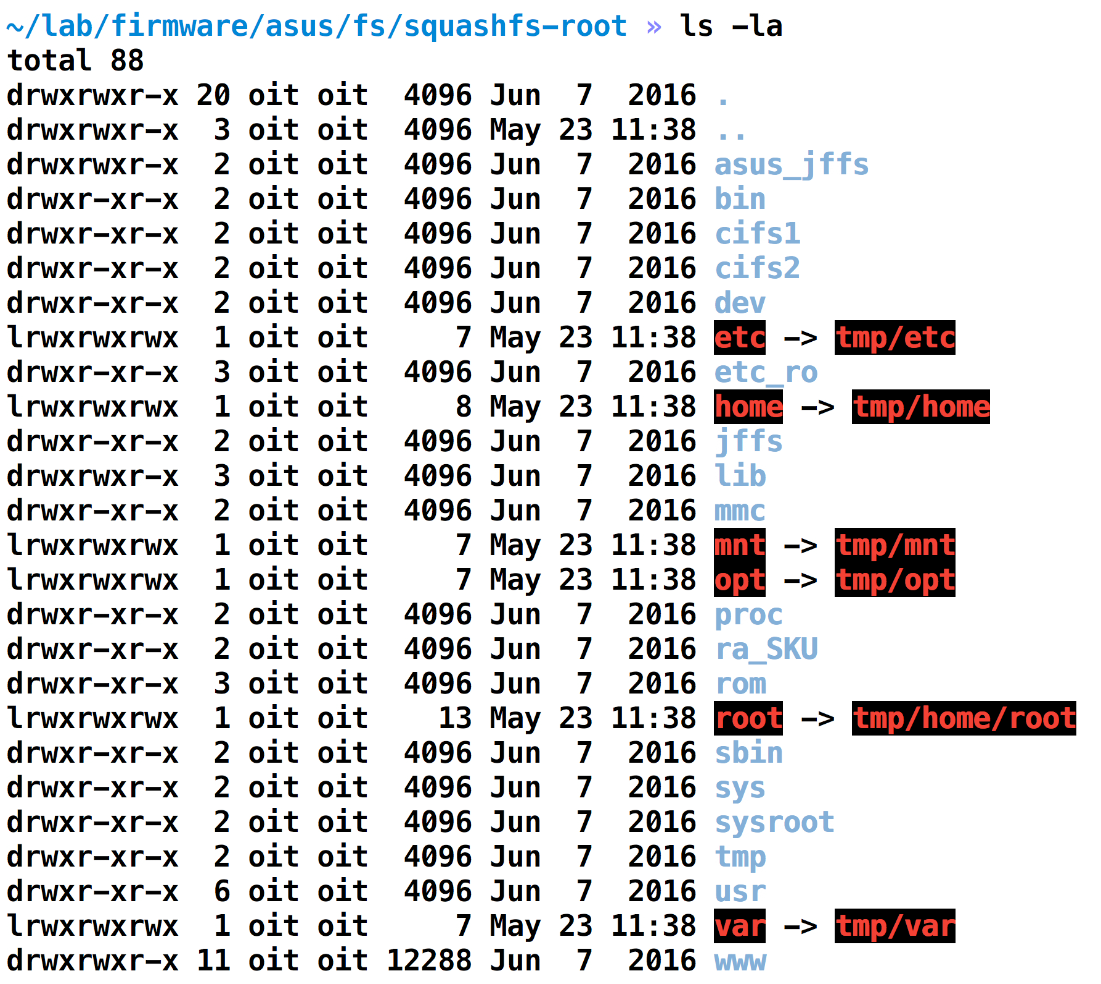
2. As you can see, we are able to identify that the Squashfs filesystem begins from the address 0x000e20c0. Once we have this information, we can use the dd utility to dump contents starting from this location till the end, as follows:



3. Run a utility such as unsquashfs to look at the entire filesystem.



4. The following is a screenshot of how the entire filesystem looks



## Common file system and magic bytes

|  |  |
| --- | --- |
| **File System** | **Magic Byte** |
| Squashfs, big endian | sqsh |
| Squashfs, little endian | hsqs |
| Squashfs with LZMA compression | sqlz |
| TROC | TROC |
| PFS | PFS |
| MPFS | MPFS |

[magic bytes for different file systems](https://github.com/devttys0/binwalk/blob/62e9caa164305a18d7d1f037ab27d14ac933d3cf/src/binwalk/magic/filesystems)

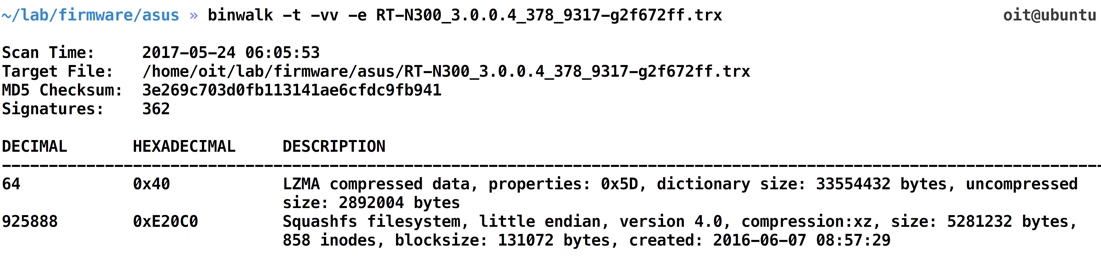
## Automatic Process

All of the steps mentioned earlier can be done automatically with tools such as Binwalk by Craig Heffner or Firmware Mod Kit.

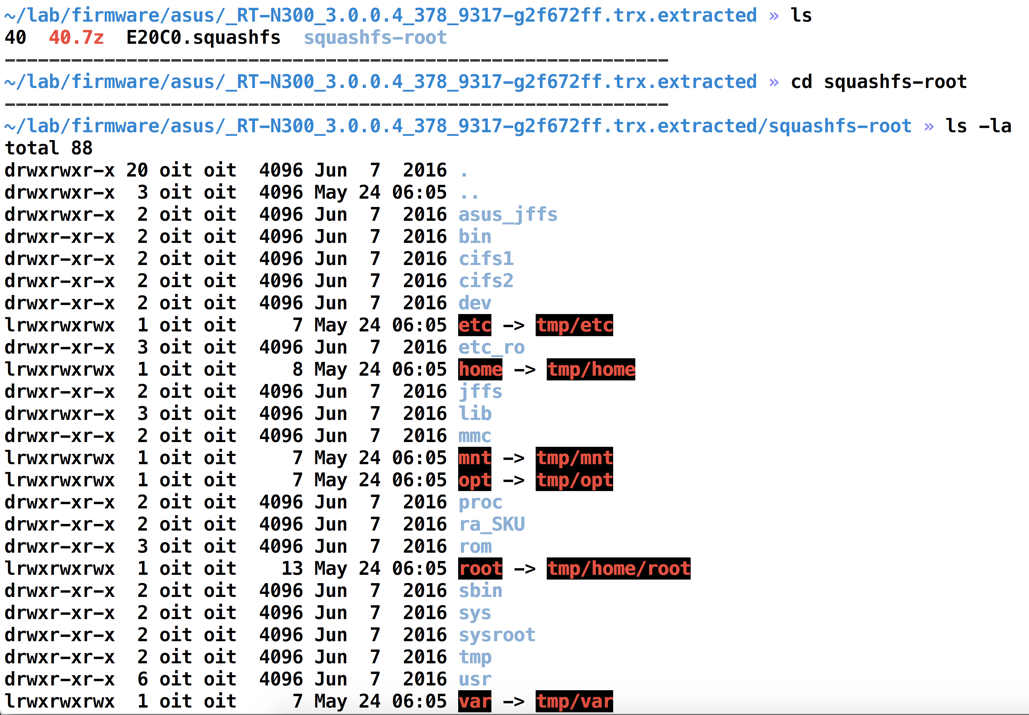
### Binwalk

It is a fast, easy to use tool for analyzing, reverse engineering, and extracting firmware images.

The t and vv flags simply allow us to print the output in a more readable and verbose format.



After the Binwalk execution, we can go to the directory with the name \_[firmwarename].extracted, which will hold the entire filesystem for us as shown in the following screenshot:



In order to perform entropy analysis (check if firmware is encrypted or zipped), run binwalk with the -E flag followed by the firmware name

The number of filesystems detected by Binwalk can be found at this URL: <https://github.com/devttys0/binwalk/blob/62e9caa164305a18d7d1f037ab27d14ac933d3cf/src/binwalk/magic/filesystems>

### Firmware Mod Kit

The Firmware Mod Kit allows for easy deconstruction and reconstruction of firmware images for various embedded devices. While it primarily targets Linux based routers, it should be compatible with most firmware that makes use of common firmware formats and file systems such as TRX/uImage and SquashFS/CramFS. The Firmware Mod Kit is a collection of utilities and shell scripts.

Extracting Firmware

$ ./extract-firmware.sh firmware.bin

Re-Building Firmware

$ ./build-firmware.sh [-nopad] [-min]

# Analyzing Filesystem

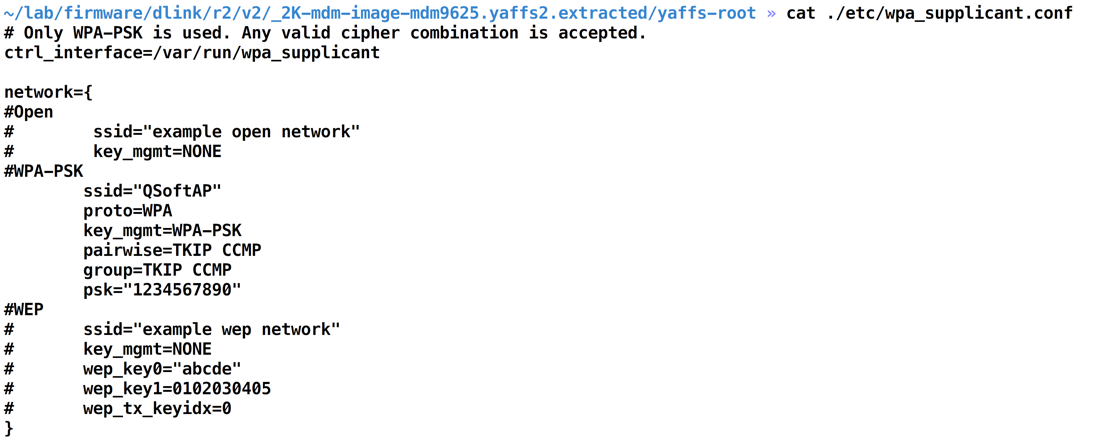
## Manual Process

Analyzing and understanding a filesystem and its internal contents is all about your manual assessment skills. This is how you will be able to identify vulnerabilities. Even while working with various tools, you will realize that, in the end, it comes down to analyzing the binary file manually and figuring out the vulnerability.

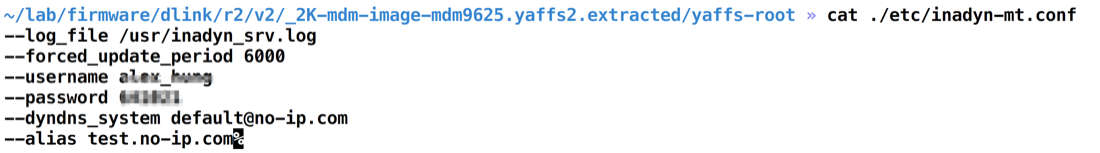
From here on, we can start navigating inside various directories and look at files which look interesting from a security point of view. One of the first things we could do is look for all the configuration files by running a find query for all the \*.conf files, as shown in the following screenshot:



For instance, this is what is present inside the wpa-supplicant.conf file



Let's look at other files such as inadyn-mt.conf:



Surprisingly, this file has highly sensitive information which in no way should have been able to be accessed. As we can see from the preceding screenshot, this file stores the no-IP configuration for the router, including the username and password combination which is used for the https://www.no-ip.com access.

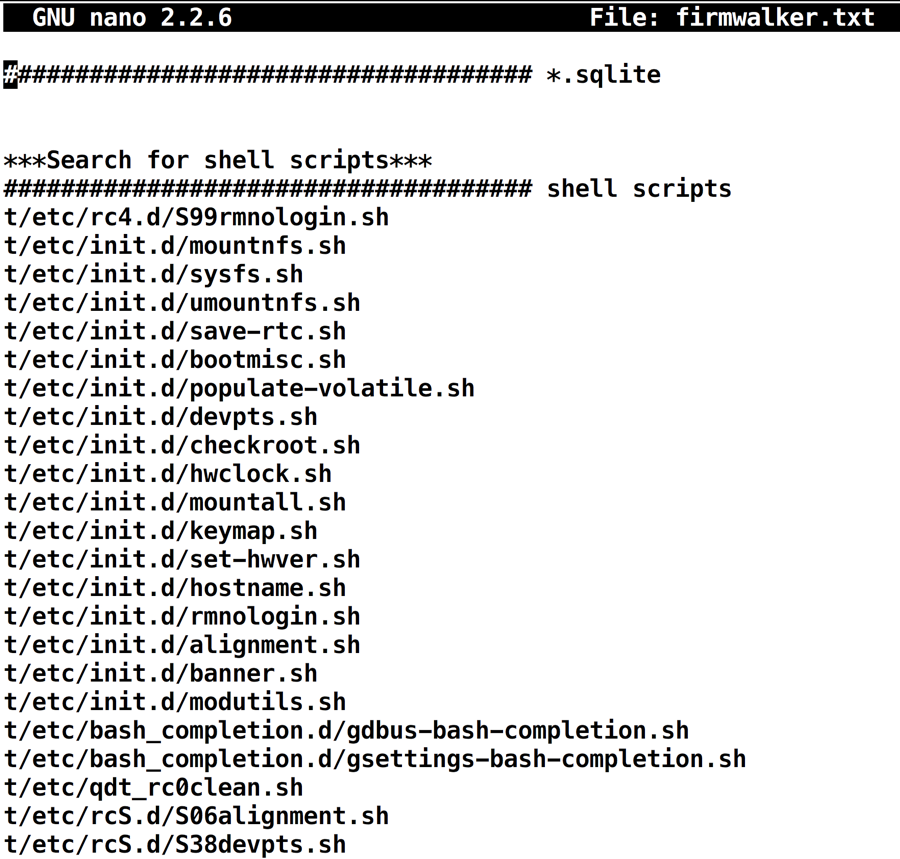
This is how we can find sensitive information hidden in firmware. You can obviously look around more and identify more sensitive information within the firmware's filesystem.

## Automatic approach

For this, we will use a tool called Firmwalker, written by Craig Smith, which helps identify some of the common sensitive information in a firmware through static analysis. we just need to run the ./firmwalker.sh script followed by the extracted filesystem location as follows:

./firmwalker.sh ~/lab/firmware/dlink/r2/v2/\_2K-mdm-image-mdm9625.yaffs2.extracted/yaffs-root.

The Firmwalker script identifies a number of different things for us, including additional binary files, certificates, IP addresses, private keys, and so on. It also stores the output in a file called firmwalker.txt (unless a different file is specified by the user) which looks as shown in the following screenshot



Once we have the report generated by Firmwalker, we can look at all the different files individually and analyze them further.

Another thing which we could do with firmware filesystem content is look at the various libraries and components which have been used and see whether those components are outdated versions with vulnerabilities in them.

# Reference

* <https://www.packtpub.com/networking-and-servers/iot-penetration-testing-cookbook>
* <https://payatu.com/>