Firmware Analysis of Embedded Systems

Dimitrios-Georgios Akestoridis

Carnegie Mellon University

14-829 / 18-638: Mobile and IoT Security (Fall 2019)

Reminders

- University Policies: https://www.cmu.edu/policies/index.html
- Course Policies: http://mews.sv.cmu.edu/teaching/14829/f19/policy.html
- Be aware of potential ethical and legal implications of your actions
- Use isolated networks for your assignments and research

What is an embedded system?

- An embedded system consists of **special-purpose** computer hardware and software, often as part of a larger system and with limited resources

- Embedded systems can be found in a plethora of devices, including:
  - Thermostats
  - Washing machines
  - Pacemakers

- Most IoT devices are just embedded systems with networking capabilities, such as:
  - IP cameras
  - Fitness trackers
  - Smart locks

How do embedded systems work?

- The special-purpose computer software that controls an embedded system is often referred to as **firmware** and it is stored in non-volatile memory

How do embedded systems work?

- The special-purpose computer software that controls an embedded system is often referred to as **firmware** and it is stored in non-volatile memory
- Many vendors use flash memory in their devices to store their firmware, which enables them to later:
    - Improve the system's functionality
    - Fix security vulnerabilities

How do embedded systems work?

- The special-purpose computer software that controls an embedded system is often referred to as **firmware** and it is stored in non-volatile memory

- Many vendors use flash memory in their devices to store their firmware, which enables them to later:
    - Improve the system's functionality
    - Fix security vulnerabilities

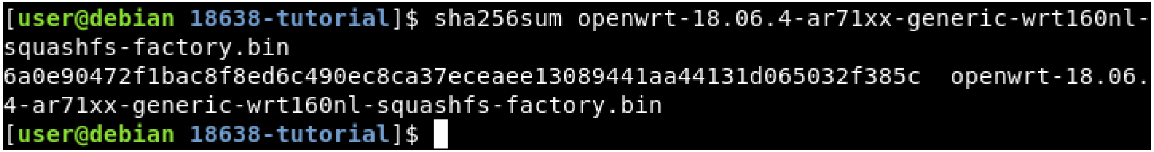
- A **firmware image** may be provided in order to update the firmware of a device, which can be done either manually or automatically

What does a firmware image look like?

- Possible methods for obtaining the firmware image of a device:
  - Downloading it from the vendor’s website
  - Capturing it during the device’s firmware update process
  - Extracting it from the hardware

What does a firmware image look like?

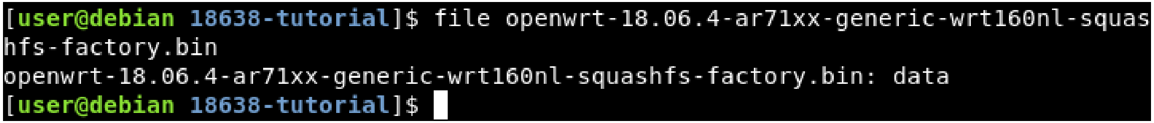
- Possible methods for obtaining the firmware image of a device:
  - Downloading it from the vendor’s website
  - Capturing it during the device’s firmware update process
  - Extracting it from the hardware
- For illustration purposes, we will use a firmware image from the OpenWrt project:
  - https://downloads.openwrt.org/releases/18.06.4/targets/ar71xx/generic/
  - https://git.openwrt.org/openwrt/openwrt.git/



```
[user@debian 18638-tutorial]$ sha256sum openwrt-18.06.4-ar71xx-generic-wrt160nl-squashfs-factory.bin
6a0e90472f1bac8f8ed6c490ec8ca37eceaee13089441aa44131d065032f385c  openwrt-18.06.4-ar71xx-generic-wrt160nl-squashfs-factory.bin
[user@debian 18638-tutorial]$
```

$ file

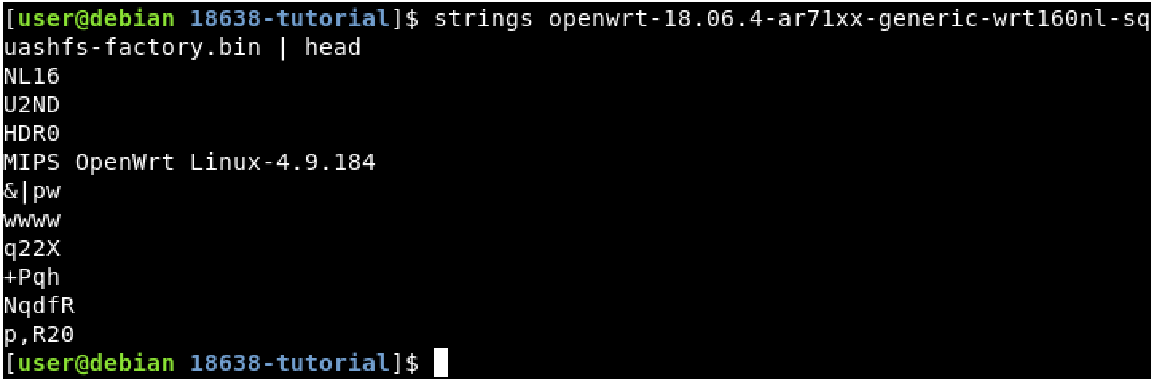
- The firmware image could be in a standard archive format that the `file` command can identify
- If the file format of the provided firmware image is unknown, then `file` will simply report that it contains binary data



```
[user@debian 18638-tutorial]$ file openwrt-18.06.4-ar71xx-generic-wrt160nl-squas
hfs-factory.bin
openwrt-18.06.4-ar71xx-generic-wrt160nl-squashfs-factory.bin: data
[user@debian 18638-tutorial]$
```

$ strings

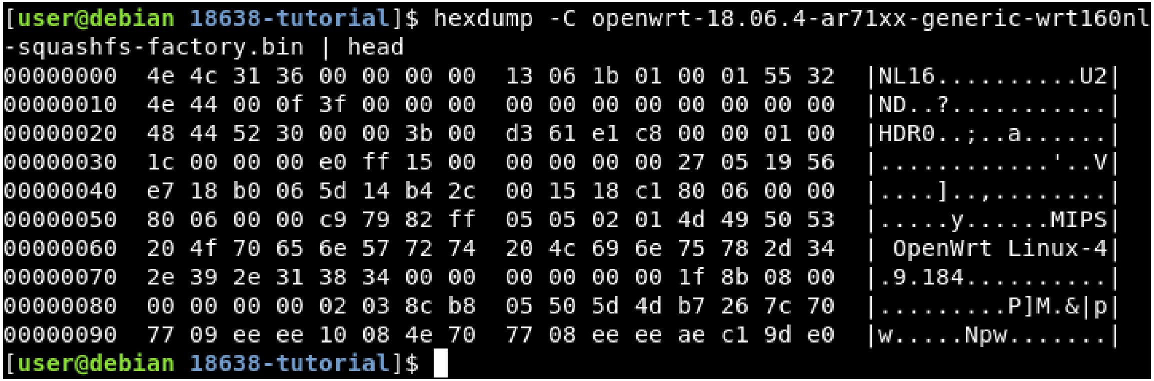
- We can inspect sequences of printable characters in the firmware image with the `strings` command



```
[user@debian 18638-tutorial]$ strings openwrt-18.06.4-ar71xx-generic-wrt160nl-squashfs-factory.bin | head
NL16
U2ND
HDR0
MIPS OpenWrt Linux-4.9.184
&|pw
wwww
q22X
+Pqh
NqdfR
p,R20
[user@debian 18638-tutorial]$
```

$ hexdump

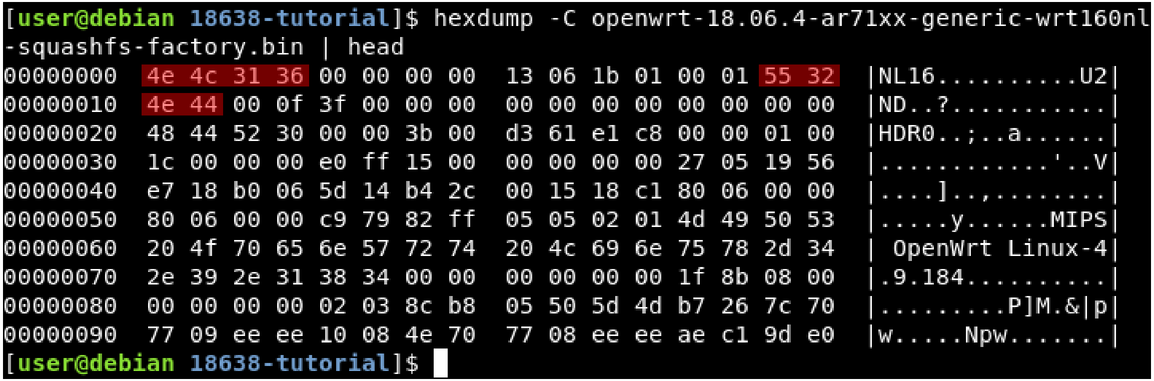
- We can examine the bytes of the firmware image with the hexdump command



```
[user@debian 18638-tutorial]$ hexdump -C openwrt-18.06.4-ar71xx-generic-wrt160nl
-squashfs-factory.bin | head
00000000  4e 4c 31 36 00 00 00 00  13 06 1b 01 00 01 55 32  |NL16..........U2|
00000010  4e 44 00 0f 3f 00 00 00  00 00 00 00 00 00 00 00  |ND..?...........|
00000020  48 44 52 30 00 00 3b 00  d3 61 e1 c8 00 00 01 00  |HDR0..;..a......|
00000030  1c 00 00 00 e0 ff 15 00  00 00 00 00 27 05 19 56  |............'..V|
00000040  e7 18 b0 06 5d 14 b4 2c  00 15 18 c1 80 06 00 00  |....]..,........|
00000050  80 06 00 00 c9 79 82 ff  05 05 02 01 4d 49 50 53  |.....y......MIPS|
00000060  20 4f 70 65 6e 57 72 74  20 4c 69 6e 75 78 2d 34  | OpenWrt Linux-4|
00000070  2e 39 2e 31 38 34 00 00  00 00 00 00 1f 8b 08 00  |.9.184..........|
00000080  00 00 00 00 02 03 8c b8  05 50 5d 4d b7 26 7c 70  |.........P]M.&|p|
00000090  77 09 ee ee 10 08 4e 70  77 08 ee ee ae c1 9d e0  |w.....Npw.......|
[user@debian 18638-tutorial]$
```

$ hexdump

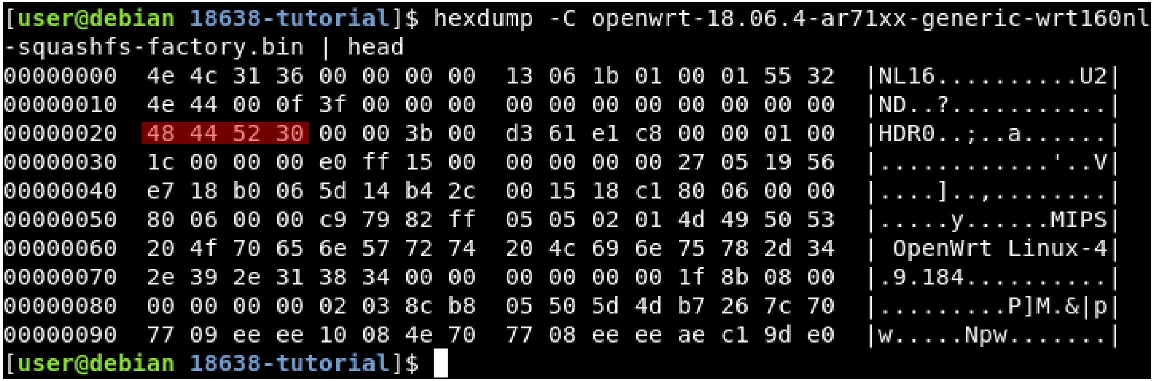
- 0x4e4c3136 (NL16) and 0x55324e44 (U2ND) correspond to the magic number and ID number of the BIN header:
  - https://git.openwrt.org/?p=openwrt/openwrt.git;a=blob_plain;f=tools/firmware-utils/src/addpattern.c;hb=HEAD



```
[user@debian 18638-tutorial]$ hexdump -C openwrt-18.06.4-ar71xx-generic-wrt160nl
-squashfs-factory.bin | head
00000000  4e 4c 31 36 00 00 00 00  13 06 1b 01 00 01 55 32  |NL16..........U2|
00000010  4e 44 00 0f 3f 00 00 00  00 00 00 00 00 00 00 00  |ND..?...........|
00000020  48 44 52 30 00 00 3b 00  d3 61 e1 c8 00 00 01 00  |HDR0..;..a......|
00000030  1c 00 00 00 e0 ff 15 00  00 00 00 00 27 05 19 56  |............'..V|
00000040  e7 18 b0 06 5d 14 b4 2c  00 15 18 c1 80 06 00 00  |....]..,........|
00000050  80 06 00 00 c9 79 82 ff  05 05 02 01 4d 49 50 53  |.....y......MIPS|
00000060  20 4f 70 65 6e 57 72 74  20 4c 69 6e 75 78 2d 34  | OpenWrt Linux-4|
00000070  2e 39 2e 31 38 34 00 00  00 00 00 00 1f 8b 08 00  |.9.184..........|
00000080  00 00 00 00 02 03 8c b8  05 50 5d 4d b7 26 7c 70  |.........P]M.&|p|
00000090  77 09 ee ee 10 08 4e 70  77 08 ee ee ae c1 9d e0  |w.....Npw.......|
[user@debian 18638-tutorial]$
```

$ hexdump

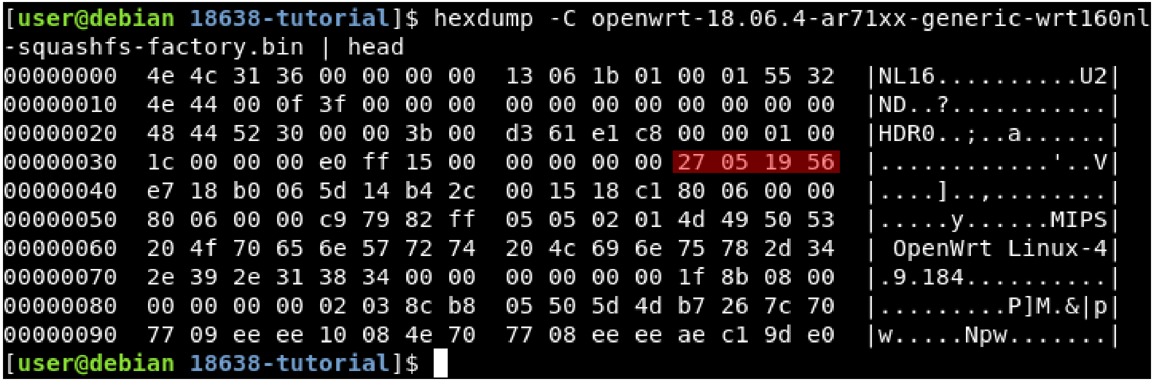
- 0x48445230 (HDR0) corresponds to the magic number of the TRX header:
  - https://git.openwrt.org/?p=openwrt/openwrt.git;a=blob_plain;f=package/system/mtd/src/trx.c;hb=HEAD



```
[user@debian 18638-tutorial]$ hexdump -C openwrt-18.06.4-ar71xx-generic-wrt160nl
-squashfs-factory.bin | head
00000000  4e 4c 31 36 00 00 00 00  13 06 1b 01 00 01 55 32  |NL16..........U2|
00000010  4e 44 00 0f 3f 00 00 00  00 00 00 00 00 00 00 00  |ND..?...........|
00000020  48 44 52 30 00 00 3b 00  d3 61 e1 c8 00 00 01 00  |HDR0..;..a......|
00000030  1c 00 00 00 e0 ff 15 00  00 00 00 00 27 05 19 56  |............'..V|
00000040  e7 18 b0 06 5d 14 b4 2c  00 15 18 c1 80 06 00 00  |....]..,........|
00000050  80 06 00 00 c9 79 82 ff  05 05 02 01 4d 49 50 53  |.....y......MIPS|
00000060  20 4f 70 65 6e 57 72 74  20 4c 69 6e 75 78 2d 34  | OpenWrt Linux-4|
00000070  2e 39 2e 31 38 34 00 00  00 00 00 00 1f 8b 08 00  |.9.184..........|
00000080  00 00 00 00 02 03 8c b8  05 50 5d 4d b7 26 7c 70  |.........P]M.&|p|
00000090  77 09 ee ee 10 08 4e 70  77 08 ee ee ae c1 9d e0  |w.....Npw.......|
[user@debian 18638-tutorial]$
```

$ hexdump

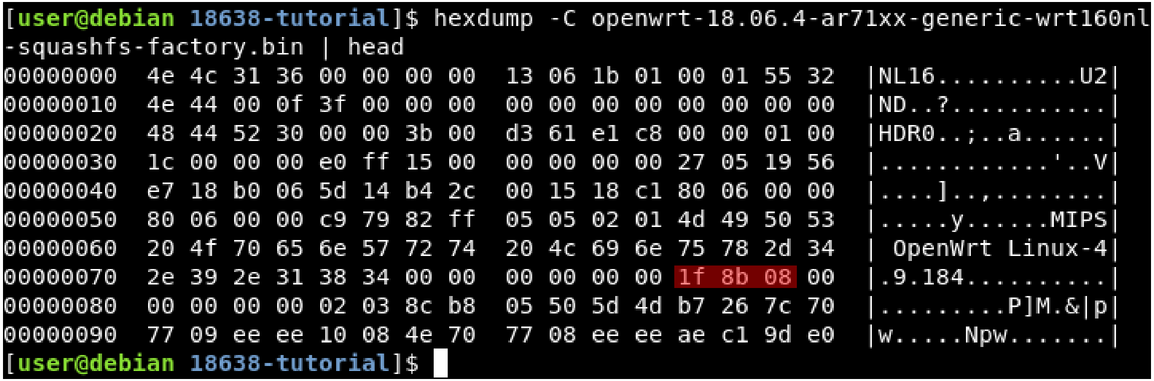
- `0x27051956` corresponds to the magic number of the uImage header:
  - https://git.denx.de/?p=u-boot.git;a=blob_plain;f=include/image.h;hb=HEAD



```
[user@debian 18638-tutorial]$ hexdump -C openwrt-18.06.4-ar71xx-generic-wrt160nl
-squashfs-factory.bin | head
00000000  4e 4c 31 36 00 00 00 00  13 06 1b 01 00 01 55 32  |NL16..........U2|
00000010  4e 44 00 0f 3f 00 00 00  00 00 00 00 00 00 00 00  |ND..?...........|
00000020  48 44 52 30 00 00 3b 00  d3 61 e1 c8 00 00 01 00  |HDR0..;..a......|
00000030  1c 00 00 00 e0 ff 15 00  00 00 00 00 27 05 19 56  |............'..V|
00000040  e7 18 b0 06 5d 14 b4 2c  00 15 18 c1 80 06 00 00  |....]..,........|
00000050  80 06 00 00 c9 79 82 ff  05 05 02 01 4d 49 50 53  |.....y......MIPS|
00000060  20 4f 70 65 6e 57 72 74  20 4c 69 6e 75 78 2d 34  | OpenWrt Linux-4|
00000070  2e 39 2e 31 38 34 00 00  00 00 00 00 1f 8b 08 00  |.9.184..........|
00000080  00 00 00 00 02 03 8c b8  05 50 5d 4d b7 26 7c 70  |..........P]M.&|p|
00000090  77 09 ee ee 10 08 4e 70  77 08 ee ee ae c1 9d e0  |w.....Npw.......|
[user@debian 18638-tutorial]$
```

$ hexdump

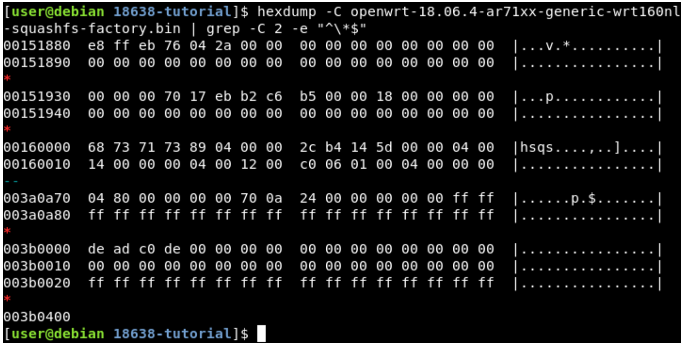
- `0x1f8b08` corresponds to the magic number of the gzip file format with the "deflate" compression method:
  - https://tools.ietf.org/html/rfc1952



```
[user@debian 18638-tutorial]$ hexdump -C openwrt-18.06.4-ar71xx-generic-wrt160nl
-squashfs-factory.bin | head
00000000  4e 4c 31 36 00 00 00 00  13 06 1b 01 00 01 55 32  |NL16..........U2|
00000010  4e 44 00 0f 3f 00 00 00  00 00 00 00 00 00 00 00  |ND..?...........|
00000020  48 44 52 30 00 00 3b 00  d3 61 e1 c8 00 00 01 00  |HDR0..;..a......|
00000030  1c 00 00 00 e0 ff 15 00  00 00 00 00 27 05 19 56  |............'..V|
00000040  e7 18 b0 06 5d 14 b4 2c  00 15 18 c1 80 06 00 00  |....]..,........|
00000050  80 06 00 00 c9 79 82 ff  05 05 02 01 4d 49 50 53  |.....y......MIPS|
00000060  20 4f 70 65 6e 57 72 74  20 4c 69 6e 75 78 2d 34  | OpenWrt Linux-4|
00000070  2e 39 2e 31 38 34 00 00  00 00 00 00 1f 8b 08 00  |.9.184..........|
00000080  00 00 00 00 02 03 8c b8  05 50 5d 4d b7 26 7c 70  |.........P]M.&|p|
00000090  77 09 ee ee 10 08 4e 70  77 08 ee ee ae c1 9d e0  |w.....Npw.......|
[user@debian 18638-tutorial]$
```

$ hexdump

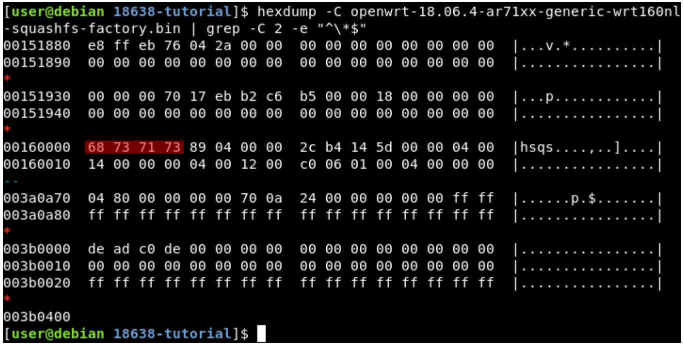
- If the -v option is not provided, hexdump replaces repeating lines with a single asterisk (*)



```
[user@debian 18638-tutorial]$ hexdump -C openwrt-18.06.4-ar71xx-generic-wrt160nl
-squashfs-factory.bin | grep -C 2 -e "^\*$"
00151880  e8 ff eb 76 04 2a 00 00  00 00 00 00 00 00 00 00  |...v.*..........|
00151890  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00151930  00 00 00 70 17 eb b2 c6  b5 00 00 18 00 00 00 00  |...p............|
00151940  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00160000  68 73 71 73 89 04 00 00  2c b4 14 5d 00 00 04 00  |hsqs....,..]....|
00160010  14 00 00 00 04 00 12 00  c0 06 01 00 04 00 00 00  |................|
--
003a0a70  04 80 00 00 00 00 70 0a  24 00 00 00 00 00 ff ff  |......p.$.......|
003a0a80  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
003b0000  de ad c0 de 00 00 00 00  00 00 00 00 00 00 00 00  |................|
003b0010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
003b0020  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
003b0400
[user@debian 18638-tutorial]$
```

$ hexdump

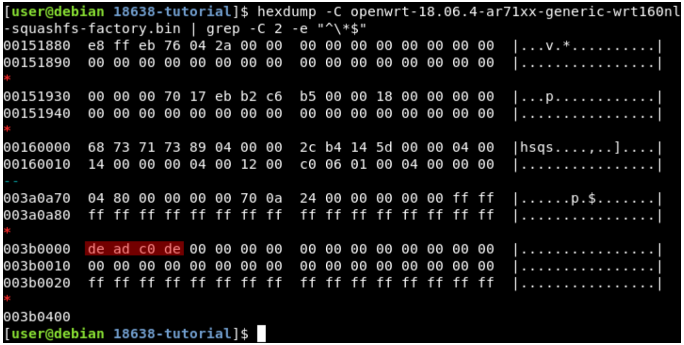
- `0x68737173` (hsqs) corresponds to the magic number of the little-endian SquashFS filesystem:
  - https://sourceforge.net/p/squashfs/code/ci/master/tree/squashfs-tools/squashfs_fs.h



```
[user@debian 18638-tutorial]$ hexdump -C openwrt-18.06.4-ar71xx-generic-wrt160nl
-squashfs-factory.bin | grep -C 2 -e "^\*$"
00151880  e8 ff eb 76 04 2a 00 00  00 00 00 00 00 00 00 00  |...v.*..........|
00151890  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00151930  00 00 00 70 17 eb b2 c6  b5 00 00 18 00 00 00 00  |...p............|
00151940  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00160000  68 73 71 73 89 04 00 00  2c b4 14 5d 00 00 04 00  |hsqs....,..]....|
00160010  14 00 00 00 04 00 12 00  c0 06 01 00 04 00 00 00  |................|
--
003a0a70  04 80 00 00 00 00 70 0a  24 00 00 00 00 00 ff ff  |......p.$.......|
003a0a80  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
003b0000  de ad c0 de 00 00 00 00  00 00 00 00 00 00 00 00  |................|
003b0010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
003b0020  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
003b0400
[user@debian 18638-tutorial]$ 
```

$ hexdump

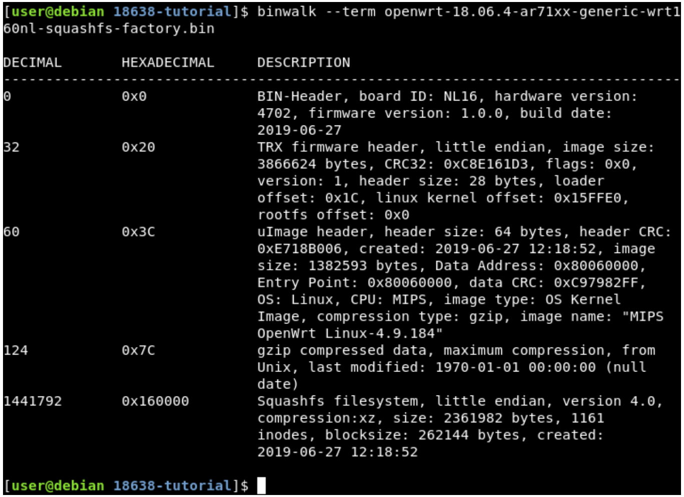
- `0xdeadc0de` indicates the start of the reformatted JFFS2 partition:
  - https://openwrt.org/docs/techref/filesystems



```
[user@debian 18638-tutorial]$ hexdump -C openwrt-18.06.4-ar71xx-generic-wrt160nl
-squashfs-factory.bin | grep -C 2 -e "^\*$"
00151880  e8 ff eb 76 04 2a 00 00  00 00 00 00 00 00 00 00  |...v.*..........|
00151890  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00151930  00 00 00 70 17 eb b2 c6  b5 00 00 18 00 00 00 00  |...p............|
00151940  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00160000  68 73 71 73 89 04 00 00  2c b4 14 5d 00 00 04 00  |hsqs....,..]....|
00160010  14 00 00 00 04 00 12 00  c0 06 01 00 04 00 00 00  |................|
--
003a0a70  04 80 00 00 00 00 70 0a  24 00 00 00 00 00 ff ff  |......p.$.......|
003a0a80  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
003b0000  de ad c0 de 00 00 00 00  00 00 00 00 00 00 00 00  |................|
003b0010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
003b0020  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
003b0400
[user@debian 18638-tutorial]$
```

$ binwalk

- We can use `binwalk` to scan for known signatures
- Custom signatures can easily be incorporated
- Wide variety of analysis options available
- https://github.com/ReFirmLabs/binwalk



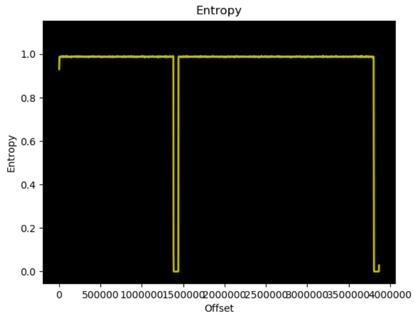
```
[user@debian 18638-tutorial]$ binwalk --term openwrt-18.06.4-ar71xx-generic-wrt1
60nl-squashfs-factory.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             BIN-Header, board ID: NL16, hardware version:
                              4702, firmware version: 1.0.0, build date:
                              2019-06-27
32            0x20            TRX firmware header, little endian, image size:
                              3866624 bytes, CRC32: 0xC8E161D3, flags: 0x0,
                              version: 1, header size: 28 bytes, loader
                              offset: 0x1C, linux kernel offset: 0x15FFE0,
                              rootfs offset: 0x0
60            0x3C            uImage header, header size: 64 bytes, header CRC:
                              0xE718B006, created: 2019-06-27 12:18:52, image
                              size: 1382593 bytes, Data Address: 0x80060000,
                              Entry Point: 0x80060000, data CRC: 0xC97982FF,
                              OS: Linux, CPU: MIPS, image type: OS Kernel
                              Image, compression type: gzip, image name: "MIPS
                              OpenWrt Linux-4.9.184"
124           0x7C            gzip compressed data, maximum compression, from
                              Unix, last modified: 1970-01-01 00:00:00 (null
                              date)
1441792       0x160000        Squashfs filesystem, little endian, version 4.0,
                              compression:xz, size: 2361982 bytes, 1161
                              inodes, blocksize: 262144 bytes, created:
                              2019-06-27 12:18:52

[user@debian 18638-tutorial]$ 
```

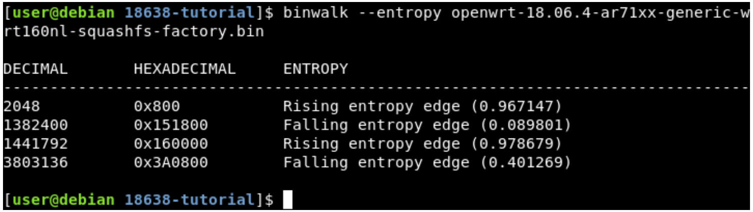
$ binwalk

- Regions that contain compressed or encrypted data tend to have high values of **entropy**
- Useful for the inspection of regions that contain data in an unknown format

$ binwalk

- Regions that contain compressed or encrypted data tend to have high values of **entropy**
- Useful for the inspection of regions that contain data in an unknown format





```
[user@debian 18638-tutorial]$ binwalk --entropy openwrt-18.06.4-ar71xx-generic-w
rt160nl-squashfs-factory.bin

DECIMAL       HEXADECIMAL     ENTROPY
--------------------------------------------------------------------------------
2048          0x800           Rising entropy edge (0.967147)
1382400       0x151800        Falling entropy edge (0.089801)
1441792       0x160000        Rising entropy edge (0.978679)
3803136       0x3A0800        Falling entropy edge (0.401269)

[user@debian 18638-tutorial]$ 
```

`$ binvis`

- We can use `binvis` to generate a **visualization** of the firmware image with space-filling curves in order to identify regions with non-random data

- Coloring scheme:
  - `0x00`: `[0,0,0]`
  - `0xff`: `[255,255,255]`
  - Printable character: `[55,126,184]`
  - Everything else: `[228,26,28]`

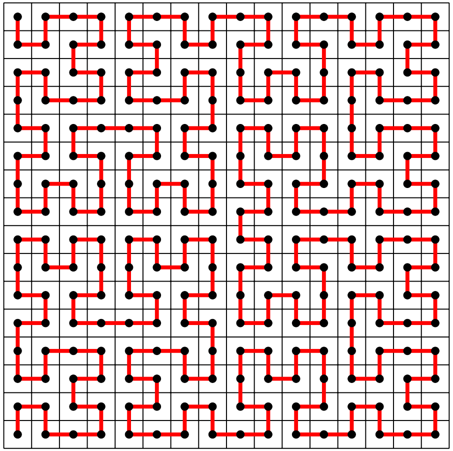
- https://github.com/cortesi/scurve

$ binvis

- We can use `binvis` to generate a **visualization** of the firmware image with space-filling curves in order to identify regions with non-random data

- Coloring scheme:
  - `0x00`: `[0,0,0]`
  - `0xff`: `[255,255,255]`
  - Printable character: `[55,126,184]`
  - Everything else: `[228,26,28]`

- https://github.com/cortesi/scurve

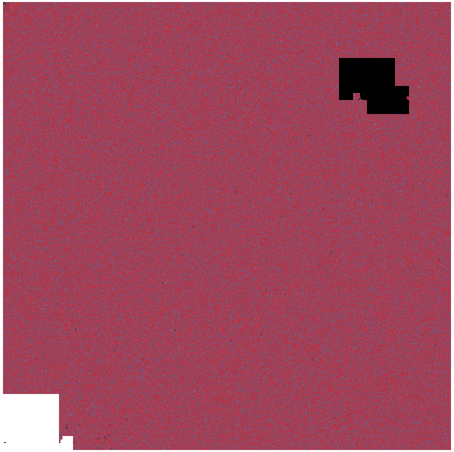


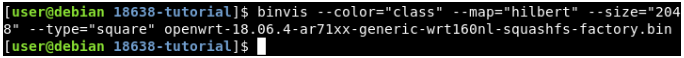
$ binvis

- We can use `binvis` to generate a **visualization** of the firmware image with space-filling curves in order to identify regions with non-random data

- Coloring scheme:
  - `0x00`: `[0,0,0]`
  - `0xff`: `[255,255,255]`
  - Printable character: `[55,126,184]`
  - Everything else: `[228,26,28]`

- https://github.com/cortesi/scurve

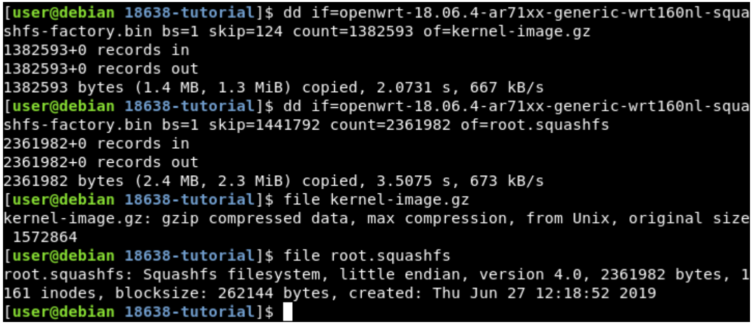




```
[user@debian 18638-tutorial]$ binvis --color="class" --map="hilbert" --size="2048" --type="square" openwrt-18.06.4-ar71xx-generic-wrt160nl-squashfs-factory.bin
[user@debian 18638-tutorial]$ 
```

$ dd

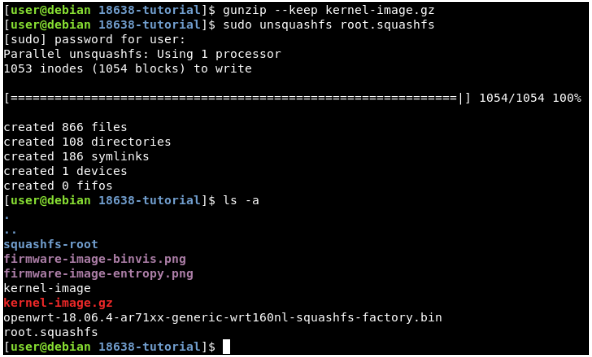
- We can duplicate regions of the firmware image with the dd command:
    - `if` option: Input file
    - `bs` option: Number of bytes in a block (in decimal notation)
    - `skip` option: Number of blocks to skip (in decimal notation)
    - `count` option: Number of blocks to copy (in decimal notation)
    - `of` option: Output file



```
[user@debian 18638-tutorial]$ dd if=openwrt-18.06.4-ar71xx-generic-wrt160nl-squa
shfs-factory.bin bs=1 skip=124 count=1382593 of=kernel-image.gz
1382593+0 records in
1382593+0 records out
1382593 bytes (1.4 MB, 1.3 MiB) copied, 2.0731 s, 667 kB/s
[user@debian 18638-tutorial]$ dd if=openwrt-18.06.4-ar71xx-generic-wrt160nl-squa
shfs-factory.bin bs=1 skip=1441792 count=2361982 of=root.squashfs
2361982+0 records in
2361982+0 records out
2361982 bytes (2.4 MB, 2.3 MiB) copied, 3.5075 s, 673 kB/s
[user@debian 18638-tutorial]$ file kernel-image.gz
kernel-image.gz: gzip compressed data, max compression, from Unix, original size
 1572864
[user@debian 18638-tutorial]$ file root.squashfs
root.squashfs: Squashfs filesystem, little endian, version 4.0, 2361982 bytes, 1
161 inodes, blocksize: 262144 bytes, created: Thu Jun 27 12:18:52 2019
[user@debian 18638-tutorial]$ 
```

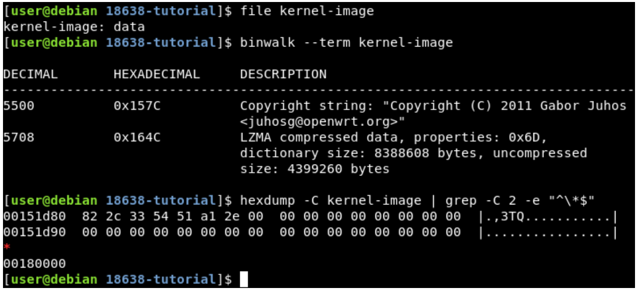
Data extraction tools

- We can extract gzip compressed data with `gunzip` and SquashFS filesystems with `unsquashfs`
- Vendors often use **non-standard** SquashFS filesystems that `unsquashfs` is unable to extract:
    - https://github.com/devttys0/sasquatch
- With the `--extract` option, `binwalk` uses common tools to extract the files that it identified

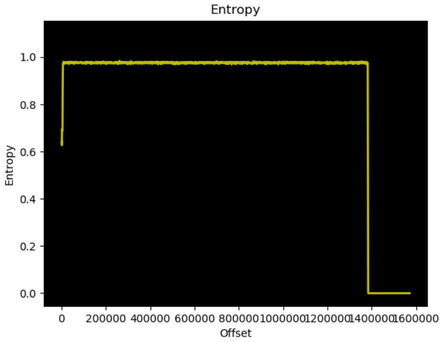


```
[user@debian 18638-tutorial]$ gunzip --keep kernel-image.gz
[user@debian 18638-tutorial]$ sudo unsquashfs root.squashfs
[sudo] password for user:
Parallel unsquashfs: Using 1 processor
1053 inodes (1054 blocks) to write

[=============================================================|] 1054/1054 100%

created 866 files
created 108 directories
created 186 symlinks
created 1 devices
created 0 fifos
[user@debian 18638-tutorial]$ ls -a
.
..
squashfs-root
firmware-image-binvis.png
firmware-image-entropy.png
kernel-image
kernel-image.gz
openwrt-18.06.4-ar71xx-generic-wrt160nl-squashfs-factory.bin
root.squashfs
[user@debian 18638-tutorial]$
```

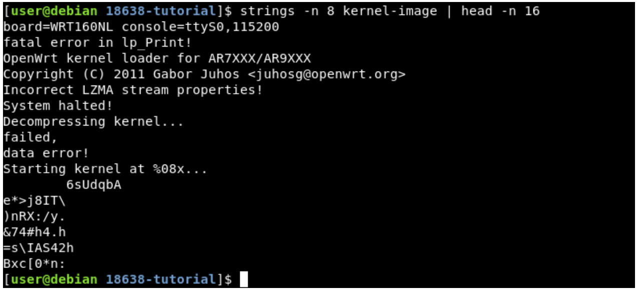
Inspecting the kernel image



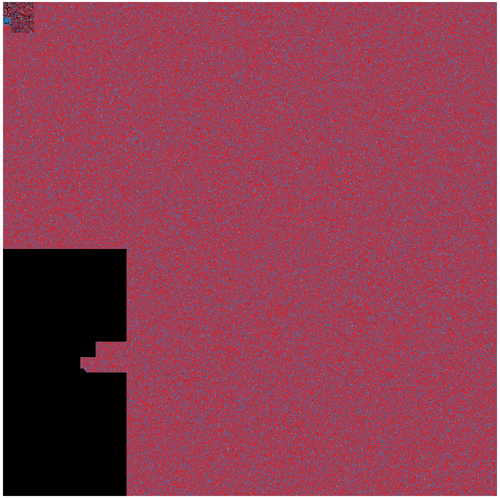
```
[user@debian 18638-tutorial]$ file kernel-image
kernel-image: data
[user@debian 18638-tutorial]$ binwalk --term kernel-image

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
5500          0x157C          Copyright string: "Copyright (C) 2011 Gabor Juhos
                              <juhosg@openwrt.org>"
5708          0x164C          LZMA compressed data, properties: 0x6D,
                              dictionary size: 8388608 bytes, uncompressed
                              size: 4399260 bytes

[user@debian 18638-tutorial]$ hexdump -C kernel-image | grep -C 2 -e "^\*$"
00151d80  82 2c 33 54 51 a1 2e 00  00 00 00 00 00 00 00 00  |.,3TQ...........|
00151d90  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00180000
[user@debian 18638-tutorial]$
```



Inspecting the kernel image

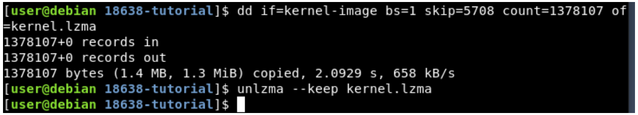


```
[user@debian 18638-tutorial]$ strings -n 8 kernel-image | head -n 16
board=WRT160NL console=ttyS0,115200
fatal error in lp_Print!
OpenWrt kernel loader for AR7XXX/AR9XXX
Copyright (C) 2011 Gabor Juhos <juhosg@openwrt.org>
Incorrect LZMA stream properties!
System halted!
Decompressing kernel...
failed,
data error!
Starting kernel at %08x...
        6sUdqbA
e*>j8IT\
)nRX:/y.
&74#h4.h
=s\IAS42h
Bxc[0*n:
[user@debian 18638-tutorial]$ 
```



Decompressing the kernel

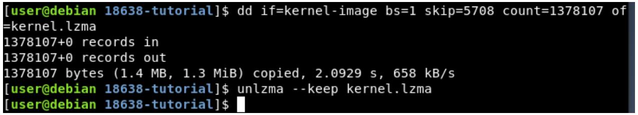
- We can extract LZMA compressed data with the `unlzma` command
- For recursive scanning and extraction of known files, we can use `binwalk` with the `--extract` and `--matryoshka` options, or simply `-eM`



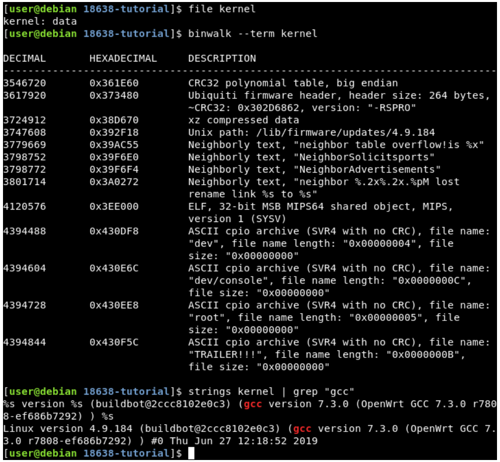
```
[user@debian 18638-tutorial]$ dd if=kernel-image bs=1 skip=5708 count=1378107 of
=kernel.lzma
1378107+0 records in
1378107+0 records out
1378107 bytes (1.4 MB, 1.3 MiB) copied, 2.0929 s, 658 kB/s
[user@debian 18638-tutorial]$ unlzma --keep kernel.lzma
[user@debian 18638-tutorial]$
```

Decompressing the kernel

- We can extract LZMA compressed data with the `unlzma` command
- For recursive scanning and extraction of known files, we can use `binwalk` with the `--extract` and `--matryoshka` options, or simply `-eM`

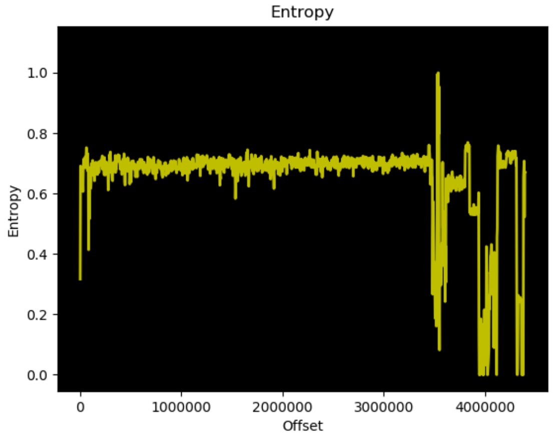


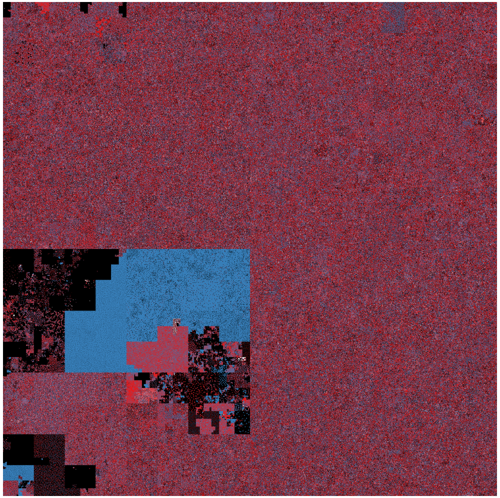
```
[user@debian 18638-tutorial]$ dd if=kernel-image bs=1 skip=5708 count=1378107 of
=kernel.lzma
1378107+0 records in
1378107+0 records out
1378107 bytes (1.4 MB, 1.3 MiB) copied, 2.0929 s, 658 kB/s
[user@debian 18638-tutorial]$ unlzma --keep kernel.lzma
[user@debian 18638-tutorial]$
```



```
[user@debian 18638-tutorial]$ file kernel
kernel: data
[user@debian 18638-tutorial]$ binwalk --term kernel

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
3546720       0x361E60        CRC32 polynomial table, big endian
3617920       0x373480        Ubiquiti firmware header, header size: 264 bytes,
                              ~CRC32: 0x302D6862, version: "-RSPRO"
3724912       0x38D670        xz compressed data
3747608       0x392F18        Unix path: /lib/firmware/updates/4.9.184
3779669       0x39AC55        Neighborly text, "neighbor table overflow!is %x"
3798752       0x39F6E0        Neighborly text, "NeighborSolicitsports"
3798772       0x39F6F4        Neighborly text, "NeighborAdvertisements"
3801714       0x3A0272        Neighborly text, "neighbor %.2x%.2x.%pM lost
                              rename link %s to %s"
4120576       0x3EE000        ELF, 32-bit MSB MIPS64 shared object, MIPS,
                              version 1 (SYSV)
4394488       0x430DF8        ASCII cpio archive (SVR4 with no CRC), file name:
                              "dev", file name length: "0x00000004", file
                              size: "0x00000000"
4394604       0x430E6C        ASCII cpio archive (SVR4 with no CRC), file name:
                              "dev/console", file name length: "0x0000000C",
                              file size: "0x00000000"
4394728       0x430EE8        ASCII cpio archive (SVR4 with no CRC), file name:
                              "root", file name length: "0x00000005", file
                              size: "0x00000000"
4394844       0x430F5C        ASCII cpio archive (SVR4 with no CRC), file name:
                              "TRAILER!!!", file name length: "0x0000000B",
                              file size: "0x00000000"

[user@debian 18638-tutorial]$ strings kernel | grep "gcc"
%s version %s (buildbot@2ccc8102e0c3) (gcc version 7.3.0 (OpenWrt GCC 7.3.0 r780
8-ef686b7292) ) %s
Linux version 4.9.184 (buildbot@2ccc8102e0c3) (gcc version 7.3.0 (OpenWrt GCC 7.
3.0 r7808-ef686b7292) ) #0 Thu Jun 27 12:18:52 2019
[user@debian 18638-tutorial]$
```

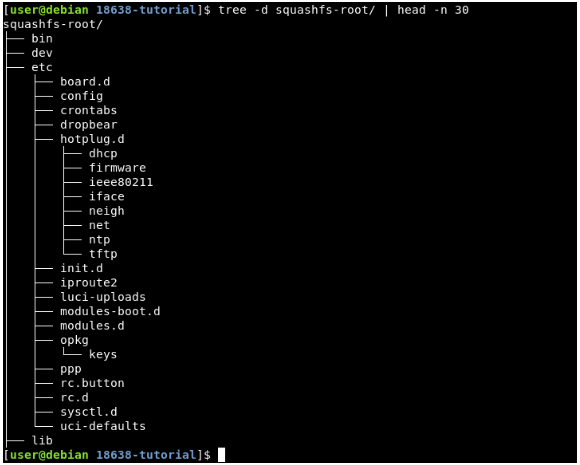
Inspecting the kernel





Inspecting the filesystem

- What to look for in the filesystem?
  - Password files
  - Encryption keys
  - Public key certificates
  - Executable files
  - Configuration files
  - Interesting keywords
- We can use `firmwalker` to search for some common files and keywords in the filesystem:
  - https://github.com/craigz28/firmwalker



```
[user@debian 18638-tutorial]$ tree -d squashfs-root/ | head -n 30
squashfs-root/
├── bin
├── dev
├── etc
│   ├── board.d
│   ├── config
│   ├── crontabs
│   ├── dropbear
│   ├── hotplug.d
│   │   ├── dhcp
│   │   ├── firmware
│   │   ├── ieee80211
│   │   ├── iface
│   │   ├── neigh
│   │   ├── net
│   │   ├── ntp
│   │   └── tftp
│   ├── init.d
│   ├── iproute2
│   ├── luci-uploads
│   ├── modules-boot.d
│   ├── modules.d
│   ├── opkg
│   │   └── keys
│   ├── ppp
│   ├── rc.button
│   ├── rc.d
│   ├── sysctl.d
│   └── uci-defaults
├── lib
[user@debian 18638-tutorial]$
```

Password files

- Usually, the system's accounts can be found in the `/etc/passwd` file and their hashed passwords are stored in the `/etc/shadow` file
- For more information regarding the format of those files:
  - `$ man 5 passwd`
  - `$ man 5 shadow`
  - `$ man 3 crypt`
- Password-cracking software:
  - https://www.openwall.com/john/
  - https://hashcat.net/hashcat/

Encryption keys

- Many devices contain hard-coded private keys in their firmware in order to support HTTPS:
    - http://www.devttys0.com/2010/12/breaking-ssl-on-embedded-devices/

Encryption keys

- Many devices contain hard-coded private keys in their firmware in order to support HTTPS:
  - http://www.devttys0.com/2010/12/breaking-ssl-on-embedded-devices/

- Multiple devices may be using the same encryption keys, sometimes even devices of different vendors:
  - https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/costin
  - https://www.sec-consult.com/en/blog/2016/09/house-of-keys-9-months-later-40-worse/index.html

Encryption keys

- Many devices contain hard-coded private keys in their firmware in order to support HTTPS:
  - http://www.devttys0.com/2010/12/breaking-ssl-on-embedded-devices/
- Multiple devices may be using the same encryption keys, sometimes even devices of different vendors:
  - https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/costin
  - https://www.sec-consult.com/en/blog/2016/09/house-of-keys-9-months-later-40-worse/index.html
- Datasets of private keys that were found in embedded systems:
  - https://github.com/devttys0/littleblackbox
  - https://github.com/sec-consult/houseofkeys

Public key certificates

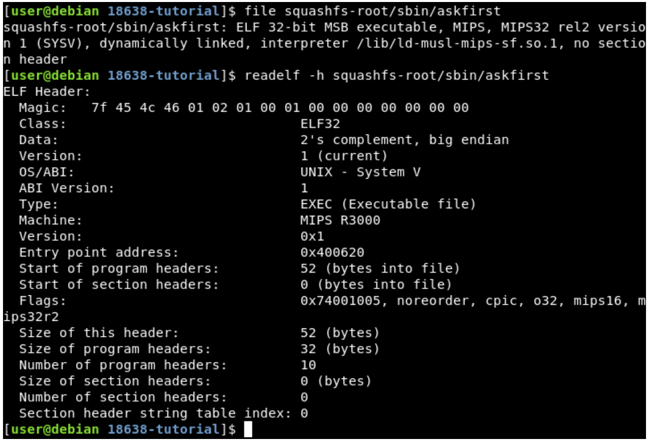
- We can process private keys, public keys, and X.509 certificates with the `openssl` program
- For example, we can view the contents of an X.509 certificate in PEM format with the following command:
    - `$ openssl x509 -in certificate.pem -text -noout`

Public key certificates

- We can process private keys, public keys, and X.509 certificates with the `openssl` program
- For example, we can view the contents of an X.509 certificate in PEM format with the following command:
  - `$ openssl x509 -in certificate.pem -text -noout`
- We can estimate the number of Internet-connected devices that use the same public key certificate by searching for its fingerprint on computer search engines:
  - https://www.shodan.io/
  - https://censys.io/

Executable files

- We can examine executable files in ELF format with the `readelf` command
- For example, with the `-h` option, `readelf` displays the information that is contained in the header of the ELF file
- We can disassemble ELF files with the appropriate `objdump` command

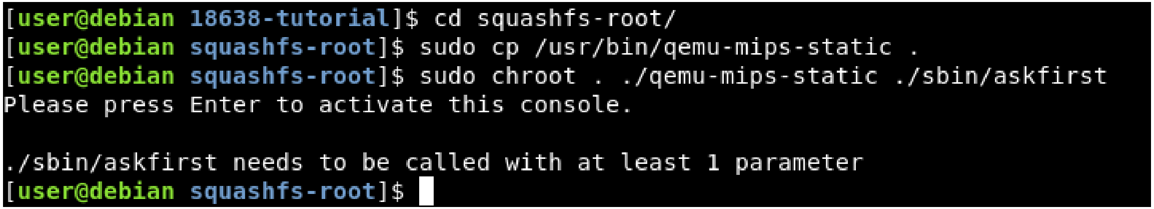


```
[user@debian 18638-tutorial]$ file squashfs-root/sbin/askfirst
squashfs-root/sbin/askfirst: ELF 32-bit MSB executable, MIPS, MIPS32 rel2 versio
n 1 (SYSV), dynamically linked, interpreter /lib/ld-musl-mips-sf.so.1, no sectio
n header
[user@debian 18638-tutorial]$ readelf -h squashfs-root/sbin/askfirst
ELF Header:
  Magic:   7f 45 4c 46 01 02 01 00 01 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, big endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       1
  Type:                              EXEC (Executable file)
  Machine:                           MIPS R3000
  Version:                           0x1
  Entry point address:               0x400620
  Start of program headers:          52 (bytes into file)
  Start of section headers:          0 (bytes into file)
  Flags:                             0x74001005, noreorder, cpic, o32, mips16, m
ips32r2
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         10
  Size of section headers:           0 (bytes)
  Number of section headers:         0
  Section header string table index: 0
[user@debian 18638-tutorial]$ 
```

QEMU user mode emulation

- We can use QEMU in user mode to execute binary files that were compiled for a different computer architecture than that of our host system:
  - https://www.qemu.org/

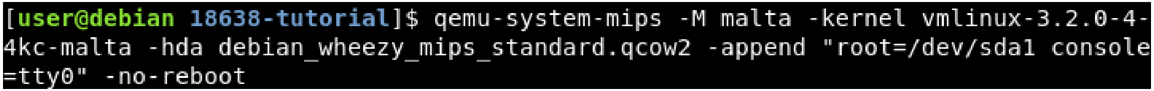
- We use the `chroot` command to execute the ELF file with the extracted SquashFS filesystem as root directory



```
[user@debian 18638-tutorial]$ cd squashfs-root/
[user@debian squashfs-root]$ sudo cp /usr/bin/qemu-mips-static .
[user@debian squashfs-root]$ sudo chroot . ./qemu-mips-static ./sbin/askfirst
Please press Enter to activate this console.

./sbin/askfirst needs to be called with at least 1 parameter
[user@debian squashfs-root]$ 
```

QEMU full system emulation

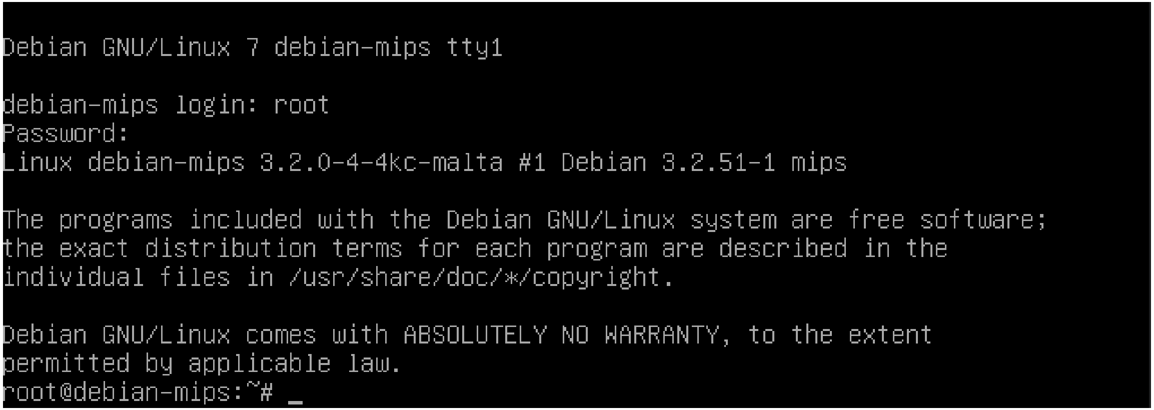
- QEMU also supports full system emulation using prebuilt images:
  - https://people.debian.org/~aurel32/qemu/



```
[user@debian 18638-tutorial]$ qemu-system-mips -M malta -kernel vmlinux-3.2.0-4-
4kc-malta -hda debian_wheezy_mips_standard.qcow2 -append "root=/dev/sda1 console
=tty0" -no-reboot
```

QEMU full system emulation

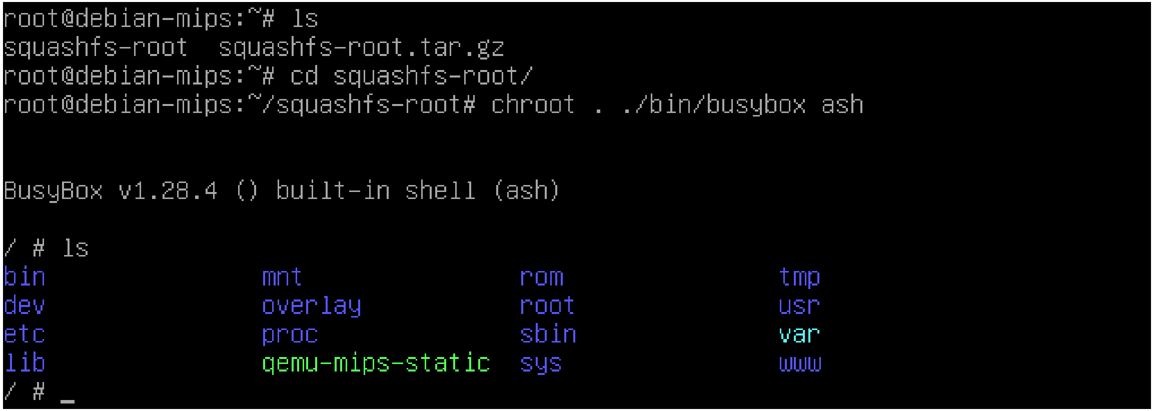
- QEMU also supports full system emulation using prebuilt images:
  - https://people.debian.org/~aurel32/qemu/



```
Debian GNU/Linux 7 debian-mips tty1

debian-mips login: root
Password:
Linux debian-mips 3.2.0-4-4kc-malta #1 Debian 3.2.51-1 mips

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@debian-mips:~# _
```

QEMU full system emulation

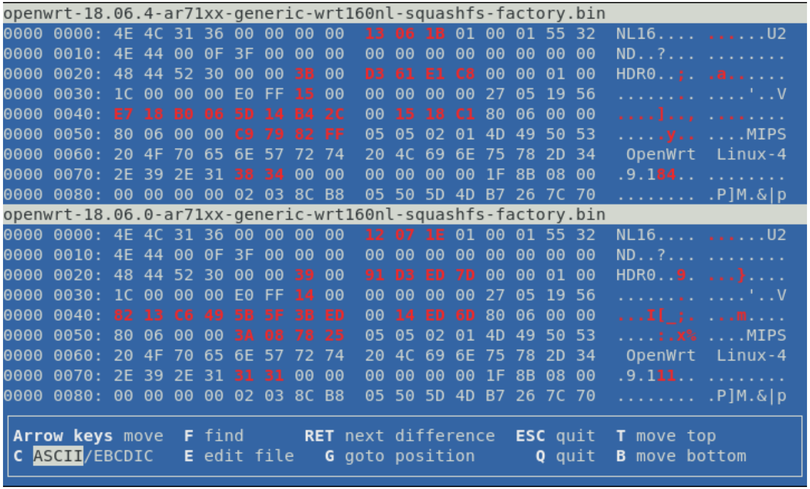
- We can copy the extracted filesystem in the hard disk image and then initiate a command interpreter (shell) with `chroot`



```
root@debian-mips:~# ls
squashfs-root  squashfs-root.tar.gz
root@debian-mips:~# cd squashfs-root/
root@debian-mips:~/squashfs-root# chroot . ./bin/busybox ash


BusyBox v1.28.4 () built-in shell (ash)

/ # ls
bin                 mnt                 rom                 tmp
dev                 overlay             root                usr
etc                 proc                sbin                var
lib                 qemu-mips-static    sys                 www
/ # _
```

$ vbindiff

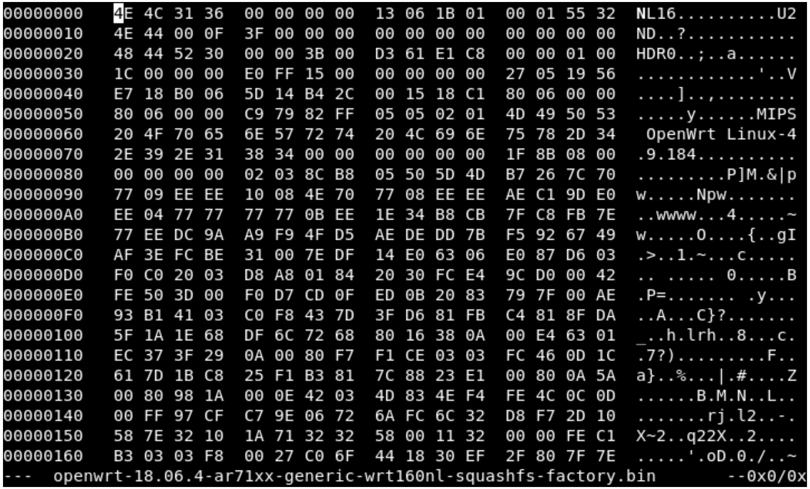
- We can use `vbindiff` to compare different versions of a firmware image



```
openwrt-18.06.4-ar71xx-generic-wrt160nl-squashfs-factory.bin
0000 0000: 4E 4C 31 36 00 00 00 00  13 06 1B 01 00 01 55 32  NL16.... ......U2
0000 0010: 4E 44 00 0F 3F 00 00 00  00 00 00 00 00 00 00 00  ND..?... ........
0000 0020: 48 44 52 30 00 00 3B 00  D3 61 E1 C8 00 00 01 00  HDR0..;. .a......
0000 0030: 1C 00 00 00 E0 FF 15 00  00 00 00 00 27 05 19 56  ........ ....'..V
0000 0040: E7 18 B0 06 5D 14 B4 2C  00 15 18 C1 80 06 00 00  ....]..,  ........
0000 0050: 80 06 00 00 C9 79 82 FF  05 05 02 01 4D 49 50 53  .....y.. ....MIPS
0000 0060: 20 4F 70 65 6E 57 72 74  20 4C 69 6E 75 78 2D 34   OpenWrt  Linux-4
0000 0070: 2E 39 2E 31 38 34 00 00  00 00 00 00 1F 8B 08 00  .9.184.. ........
0000 0080: 00 00 00 00 02 03 8C B8  05 50 5D 4D B7 26 7C 70  ........ .P]M.&|p
openwrt-18.06.0-ar71xx-generic-wrt160nl-squashfs-factory.bin
0000 0000: 4E 4C 31 36 00 00 00 00  12 07 1E 01 00 01 55 32  NL16.... ......U2
0000 0010: 4E 44 00 0F 3F 00 00 00  00 00 00 00 00 00 00 00  ND..?... ........
0000 0020: 48 44 52 30 00 00 39 00  91 D3 ED 7D 00 00 01 00  HDR0..9. ...}....
0000 0030: 1C 00 00 00 E0 FF 14 00  00 00 00 00 27 05 19 56  ........ ....'..V
0000 0040: 82 13 C6 49 5B 5F 3B ED  00 14 ED 6D 80 06 00 00  ...I[_;. ...m....
0000 0050: 80 06 00 00 3A 08 78 25  05 05 02 01 4D 49 50 53  ....:.x% ....MIPS
0000 0060: 20 4F 70 65 6E 57 72 74  20 4C 69 6E 75 78 2D 34   OpenWrt  Linux-4
0000 0070: 2E 39 2E 31 31 31 00 00  00 00 00 00 1F 8B 08 00  .9.111.. ........
0000 0080: 00 00 00 00 02 03 8C B8  05 50 5D 4D B7 26 7C 70  ........ .P]M.&|p

 Arrow keys move   F find       RET next difference  ESC quit  T move top
 C ASCII/EBCDIC    E edit file    G goto position      Q quit  B move bottom
```

$ hexedit

- We can use `hexedit` to modify a firmware image



```
00000000   4E 4C 31 36  00 00 00 00  13 06 1B 01  00 01 55 32  NL16..........U2
00000010   4E 44 00 0F  3F 00 00 00  00 00 00 00  00 00 00 00  ND..?...........
00000020   48 44 52 30  00 00 3B 00  D3 61 E1 C8  00 00 01 00  HDR0..;..a......
00000030   1C 00 00 00  E0 FF 15 00  00 00 00 00  27 05 19 56  ............'..V
00000040   E7 18 B0 06  5D 14 B4 2C  00 15 18 C1  80 06 00 00  ....]..,........
00000050   80 06 00 00  C9 79 82 FF  05 05 02 01  4D 49 50 53  .....y......MIPS
00000060   20 4F 70 65  6E 57 72 74  20 4C 69 6E  75 78 2D 34   OpenWrt Linux-4
00000070   2E 39 2E 31  38 34 00 00  00 00 00 00  1F 8B 08 00  .9.184..........
00000080   00 00 00 00  02 03 8C B8  05 50 5D 4D  B7 26 7C 70  .........P]M.&|p
00000090   77 09 EE EE  10 08 4E 70  77 08 EE EE  AE C1 9D E0  w.....Npw.......
000000A0   EE 04 77 77  77 77 0B EE  1E 34 B8 CB  7F C8 FB 7E  ..wwww...4.....~
000000B0   77 EE DC 9A  A9 F9 4F D5  AE DE DD 7B  F5 92 67 49  w.....O....{..gI
000000C0   AF 3E FC BE  31 00 7E DF  14 E0 63 06  E0 87 D6 03  .>..1.~...c.....
000000D0   F0 C0 20 03  D8 A8 01 84  20 30 FC E4  9C D0 00 42  .. ..... 0.....B
000000E0   FE 50 3D 00  F0 D7 CD 0F  ED 0B 20 83  79 7F 00 AE  .P=....... .y...
000000F0   93 B1 41 03  C0 F8 43 7D  3F D6 81 FB  C4 81 8F DA  ..A...C}?.......
00000100   5F 1A 1E 68  DF 6C 72 68  80 16 38 0A  00 E4 63 01  _..h.lrh..8...c.
00000110   EC 37 3F 29  0A 00 80 F7  F1 CE 03 03  FC 46 0D 1C  .7?)........F...
00000120   61 7D 1B C8  25 F1 B3 81  7C 88 23 E1  00 80 0A 5A  a}..%...|.#....Z
00000130   00 80 98 1A  00 0E 42 03  4D 83 4E F4  FE 4C 0C 0D  ......B.M.N..L..
00000140   00 FF 97 CF  C7 9E 06 72  6A FC 6C 32  D8 F7 2D 10  .......rj.l2..-.
00000150   58 7E 32 10  1A 71 32 32  58 00 11 32  00 00 FE C1  X~2..q22X..2....
00000160   B3 03 03 F8  00 27 C0 6F  44 18 30 EF  2F 80 7F 7E  .....'.oD.0./..~
---  openwrt-18.06.4-ar71xx-generic-wrt160nl-squashfs-factory.bin       --0x0/0x
```

General security concerns

- Is there any information leakage from the device?

General security concerns

- Is there any information leakage from the device?
- Does the device accept unauthenticated commands?

General security concerns

- Is there any information leakage from the device?
- Does the device accept unauthenticated commands?
- Is the device susceptible to replay attacks?

General security concerns

- Is there any information leakage from the device?
- Does the device accept unauthenticated commands?
- Is the device susceptible to replay attacks?
- Is the firmware image digitally signed?

General security concerns

- Is there any information leakage from the device?
- Does the device accept unauthenticated commands?
- Is the device susceptible to replay attacks?
- Is the firmware image digitally signed?
- Is the device running any unnecessary services?

General security concerns

- Is there any information leakage from the device?
- Does the device accept unauthenticated commands?
- Is the device susceptible to replay attacks?
- Is the firmware image digitally signed?
- Is the device running any unnecessary services?
- Are there any backdoors in the firmware?

General security concerns

- Is there any information leakage from the device?
- Does the device accept unauthenticated commands?
- Is the device susceptible to replay attacks?
- Is the firmware image digitally signed?
- Is the device running any unnecessary services?
- Are there any backdoors in the firmware?
- Is the device using outdated software with known vulnerabilities?