



## Analyzing Firmware With Binwalk

Author: <https://github.com/small-goblin>

Firmware is the collection of code, images, data, etc that's packaged and "flashed" (programmed onto) small devices. Stuff like renta-scooters, your home internet router, digital cameras, jet engines.... Etc.

If you're researching a device, there's a lot of ways to get firmware. Either off the device itself, or maybe downloaded from the manufacturer. Once you have the firmware how do you get out all the bundled data? Usually it isn't in a human readable format.

Binwalk is a program that "file slices" or uses heuristics to determine not only what kind of files exist in the binary, but roughly where those files begin and end. It isn't always perfect, but it's usually one of the first steps to reverse engineering firmware to find vulnerabilities.

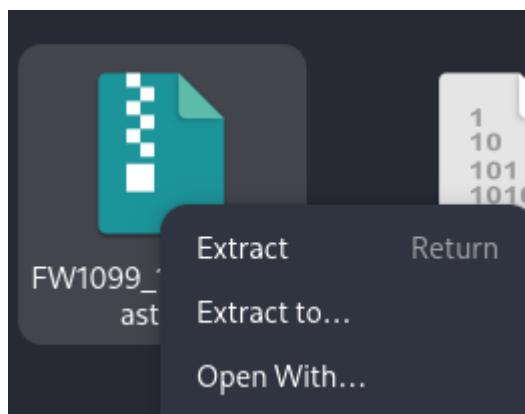
## 1. Download the firmware

<https://github.com/navaismo/Ender-3V3-SE/releases/tag/v1.0.9.9>

Download: FW1099\_115Kbps\_Fast.zip

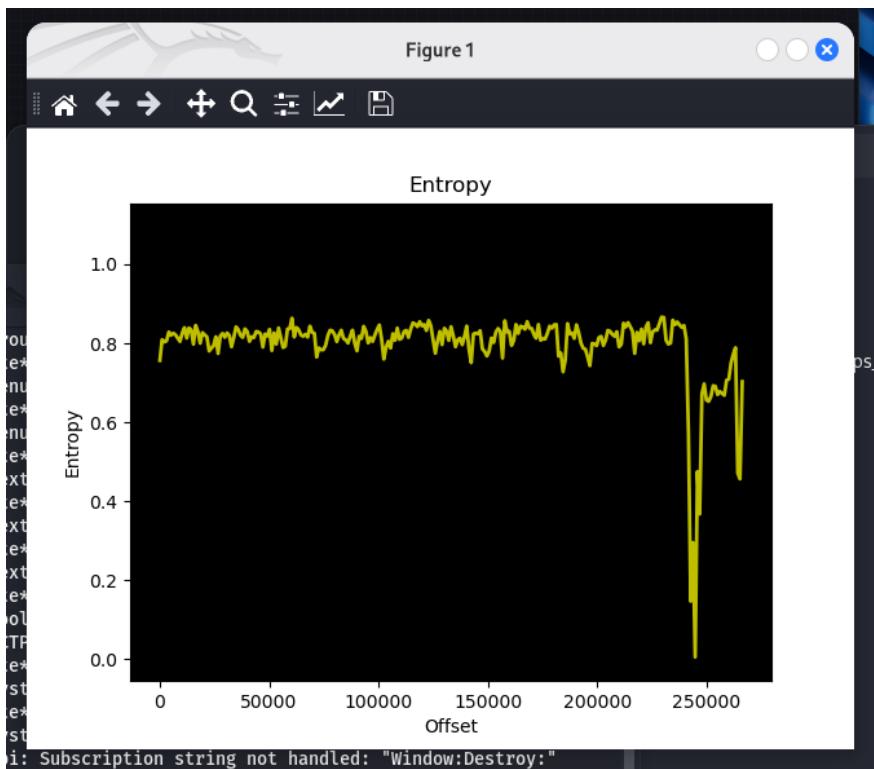
This is a modified firmware for the ender3, a common hobbyist 3d printer.

Right Click, Extract the ZIP



## 2. Check the entropy of the file.

Terminal: binwalk FW1099\_115Kbps\_Fast.bin -E



Our firmware shows low entropy.

This means: the firmware is either plain text, binary data, or structured coherently in some other unknown format. Not encrypted! Not compressed! If it was, it would have a much higher entropy, because the bytes would be more or less random.

3. So what kind of firmware is this? We can check

terminal: binwalk FW1099\_115Kbps\_Fast.bin -Y

```
(user㉿kali)-[~/Downloads]
$ binwalk FW1099_115Kbps_Fast.bin -Y

DECIMAL      HEXADECIMAL      DESCRIPTION
-----      -----
0            0x0              ARM executable code, 16-bit (Thumb), little endian, at least 1602 valid instructions
```

Seems like this firmware was compiled for an ender3 that uses an ARM processor.

#### 4. Can we recover any embedded files?

terminal: binwalk FW1099\_115Kbps\_Fast.bin

```
0x0          ARM executable code, 16-bit (Thumb), little endian, at least 1802 valid instructions
[~] Pictures
[(user㉿kali)-[~/Downloads]]$ binwalk FW1099_115Kbps_Fast.bin Videos

DECIMAL      HEXADECIMAL      DESCRIPTION
-----      -----      -----
247878      0x3C846        Unix path: /home/shaka/.platformio/packages/framework-arduinoSTM32-maple/STM32F1/cores/maple/HardwareTimer.cpp
248685      0x3CB6D        Unix path: /home/shaka/.platformio/packages/framework-arduinoSTM32-maple/STM32F1/cores/maple/libmaple/timer.c
249268      0x3CDB4        Unix path: /home/shaka/.platformio/packages/framework-arduinoSTM32-maple/STM32F1/cores/maple/stm32f1/wirish_digital_f1.c
PP
```

It looks like binwalk was able to locate at least three files.

Lets extract the files into a folder for additional analysis.

terminal: binwalk -e FW1099\_115Kbps\_Fast.bin

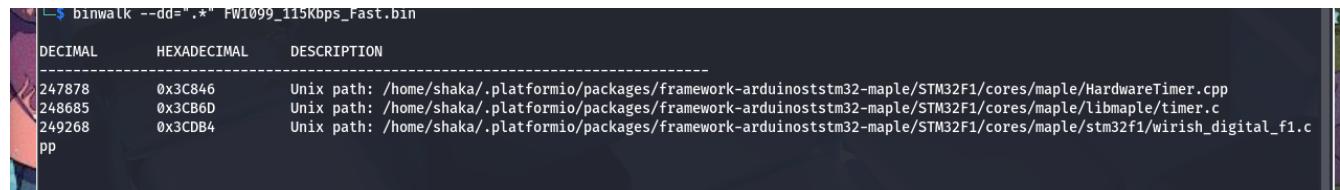
```
(user㉿kali)-[~/Downloads]
$ binwalk -e FW1099_115Kbps_Fast.bin

DECIMAL      HEXADECIMAL      DESCRIPTION
-----
WARNING: One or more files failed to extract: either no utility was found or it's unimplemented
```

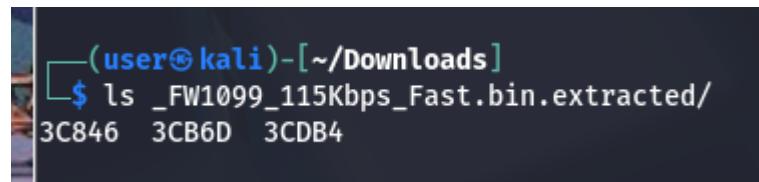
This fails, which isn't uncommon for difficult to extract files or unrecognized formats.

We can try a different command, to tell binwalk to extract everything it can.

terminal: binwalk --dd=".\*" FW1099\_115Kbps\_Fast.bin



DECIMAL	HEXADECIMAL	DESCRIPTION
247878	0x3C846	Unix path: /home/shaka/.platformio/packages/framework-arduinoostm32-maple/STM32F1/cores/maple/HardwareTimer.cpp
248685	0x3CB6D	Unix path: /home/shaka/.platformio/packages/framework-arduinoostm32-maple/STM32F1/cores/maple/libmaple/timer.c
249268	0x3CDB4	Unix path: /home/shaka/.platformio/packages/framework-arduinoostm32-maple/STM32F1/cores/maple/stm32f1/wirish_digital_f1.c



```
(user㉿kali)-[~/Downloads]
$ ls _FW1099_115Kbps_Fast.bin.extracted/
3C846  3CB6D  3CDB4
```

These files are named after their hex value. You can either rename them, or use the output table as a reference.

5. Lets see if there's any printable strings in these files

terminal: strings ./\_FW1099\_115Kbps\_Fast.bin.extracted/\*

We can see a lot of what looks like output for exception handling.

```
SD Init Fail
Subcall Overflow
!#%&(+-/1
#%(+-0369<?BFJMQ
""#&(+-0358;>AD
!$T'
1T5$9
N9__gnu_cxx20recursive_init_errorE
std::bad_exception
std::exception
N10__cxxabiv115__forced_unwindE
N10__cxxabiv119__foreign_exceptionE
St13bad_exception
St9exception
terminate called recursively
terminate called after throwing an instance of '
terminate called without an active exception
what():
N10__cxxabiv120__si_class_type_infoE
St9type_info
N10__cxxabiv117__class_type_infoE
<pow
sqrtf
!      @|
```

Inside of these files is also a lot of nonsense strings, which could likely be code being interpreted as text.

terminal: cat \_FW1099\_115Kbps\_Fast.bin.extracted/3C846

```
358;>AD
? $0@HXn*****4Rl****0L***,d**          Documents
( `**$\
$4H`l**** @h***"X***f***t***:Xv***, @Xp***4`***L***0***?V***F***Tb***~
F***6***D***g7'7g *      *
*+{+!$T'*.*.1T5$9
<KD*H          Pictures
M*QKV*Y*^*ci`n*s*aA5K          Videos
cSL**N9      gnu
```

From here, we'd want to review these files further using a dissembler tool, and hex editor.

[Read more](#)

- <https://github.com/ReFirmLabs/binwalk>
- [https://en.wikipedia.org/wiki/Entropy\\_\(information\\_theory\)](https://en.wikipedia.org/wiki/Entropy_(information_theory))
- <https://en.wikipedia.org/wiki/Firmware>
- <https://owasp.org/owasp-istg/index.html>