# *BASICS OF BINARY FIRMWARE ANALYSIS*

Pivot Project – http://www.pivotproject.org

*Jaime Geiger*

*GRIMM | jaime [at] grimm-co.com*

## Summary

Device firmware blobs can be obtained by downloading software updates from the device manufacturer or by pulling them off of the device itself. Once an image is obtained it can be analyzed to figure out how it is put together and what it contains.

This exercise is designed to give an introduction to analyzing a firmware binary through guided exploration. This lab walks through the analysis and extraction of a Dlink DCS-930L Camera firmware binary. The image is NOT a factory image and has malicious content, which you will identify.

## Tools

**binwalk** – A tool for analyzing and identifying the contents of binary blobs. Its features include detecting and extracting known file types, analyzing binary entropy, and detecting differences between binaries. It is the focus of this exercise.
**dd** – Affectionately known as "Data Destroyer." It has many uses, but you will use it for copying out parts of an image file.
**file** – Identifies files based on their "magic" numbers. (see https://en.wikipedia.org/wiki/List_of_file_signatures)
**xxd** – Command line hex dump tool.
**lzma** – Compression tool.
**cpio** – Compression tool.

## Requirements

- An Ubuntu VM (14.04 or higher)
    - cpio, lzma, python 2.7, dd, xxd, and file should already be installed
- pivot_bfa.bin – the binary file you will be analyzing

## Procedure

1. **Log into the VM and open a Terminal**

2. **Install binwalk**
    i. wget https://github.com/devttys0/binwalk/archive/v2.1.1.tar.gz
    ii. tar xf v2.1.1.tar.gz
    iii. cd binwalk-2.1.1
    iv. sudo python setup.py install

3. **Run binwalk on the binary file**
    i. binwalk pivot_bfa.bin

Questions

a. What are the two uImages?

b. What is Uboot?

c. What CPU architecture was this firmware designed to run on?

d. What OS type is this firmware?

e. How is the firmware compressed (compression type)?

4. **Extract the Linux image from pivot_bfa.bin**
   i. `dd if=pivot_bfa.bin of=linux.bin.lzma skip=327744 bs=1`

if – the source bin
of – the destination file
skip – how many blocks to skip (if block size is 1, then this is the number of bytes to skip)
bs – block size in bytes

5. **Try to decompress the extracted Linux image**
   i. `lzma --help`
   ii. `lzma -d linux.bin.lzma`

You can use the --help option to discover other useful options. When running new commands, you should always try and understand what they are doing. In this case the option “-d” is for decompressing the image.

The extraction should **fail**.

Question

a. Can you think of a reason for the failure?

6. **Examine the end of the extracted Linux image**
   i. `xxd linux.bin.lzma | tail`

The xxd command shows a hex dump of the provided file. Tail will cut the output down to the last 10 lines of output. Since xxd outputs 16 bytes per line, this will output the last 160 bytes of the file.

Question

a. What do you notice about the bytes at the end of the file? How might this affect the extraction above?

7. **Re-extract the Linux image without trailing bytes**
   i. We need to find the last line where there isn't just ffff's and then take them out.
   ii. xxd linux.bin.lzma | fgrep -v "................" | tail
   iii. We can take the last address and use the count option to stop copying right before the ffff's start. They are just padding!
   iv. dd if=pivot_bfa.bin of=linux.bin.lzma skip=327744 bs=1 count=$((0x02f9da0 + 12))
   v. lzma -d linux.bin.lzma

Again, xxd is used to dump out the hex bytes of the file. The hexdump output is then fed through fgrep with the -v option to get the inverse of the match. The quoted argument provided to fgrep represents the left hand column of the xxd output. We need to find the offset at which the padding with ff's starts so the fgrep command as it is used here will essentially get rid of all xxd output lines that are non-printable (0xff is not printable). Then the tail command is used to see the end of the output.

The count argument that is fed to dd is the number of bytes to copy from the output file starting from the skip position. The syntax used after count= may be confusing. Essentially, it calculates the integer value of the address at which the ff's start. The +12 means that the ff's start 12 bytes after the start of the last line of non-padding bytes. If you count the bytes on the last line of the xxd output from the above step, this will make sense.

8. **Run binwalk on the extracted and decompressed image**
   i. binwalk linux.bin

Questions

a. How many LZMA images are in this binary file?

b. Based on the entries below the first LZMA entry, what do you think it is?

9. **Extract known file types from the extracted and decompressed image**
   i. binwalk -ez linux.bin

   The -e option extracts known file types from the binary file. The -z option just carves them out instead of attempting to extract them fully.

   This is an alternative to using dd:
   dd if=linux.bin of=file bs=1 skip=<start number> count=$((<start of next file area> - <start number>))

10. **Analyze the extracted files**
    i. cd _linux.bin.extracted
    ii. binwalk 3AC000.7z
    iii. mv 3AC000.7z fs.bin.lzma
    iv. lzma -d fs.bin.lzma
    v. binwalk fs.bin
    vi. file fs.bin

    The 3AC000.7z file is actually an lzma compressed archive. Since the lzma utility requires that the extension of a file it is told to extract is lzma, the file has to be renamed. After lzma extraction is done, the extracted file is analyzed with binwalk and file.

    <u>Questions</u>
    a. What do all of these entries have in common? (after running binwalk on fs.bin)

    b. What kind of archive is this? (after running file on fs.bin)

11. **Extract the file system**
    i. cpio -i --no-absolute-filenames < fs.bin
    ii. ls

    We have now extracted the root filesystem!

12. **Analyze the files on the firmware file system for anything malicious**
    i. Explore the files inside the etc_ro directory (don't go into any folders under it)
    ii. Find the internet.sh script on the filesystem. Hint: use the "find" command!

Questions

a. What file appears to be a startup script?

b. Is there any malicious content in the startup script? If so what is it and why is it malicious?

c. Is there any malicious content in the internet.sh script? If so what is it and why is it malicious?

## Conclusion

Through the use of tools like binwalk, binaries can be examined and analyzed for known file types contained within them. This exercise was meant to acquaint you with some of the beginner techniques that firmware reverse engineers use to identify binary files containing device firmwares. Through the lab you were able to analyze the firmware update file, extract the filesystem, and identify malicious components contained within it.

I hope you have enjoyed this exercise. Please send any feedback or questions to me at jaime [at] grimm-co.com