COMP8053 – Embedded Software Security

Lab 2 – Understanding what Binwalk does

Make sure binwalk is installed (re-install following the steps in lab #1 if not).

- Download the DVRF_v03.bin firmware from https://github.com/praetorian-inc/DVRF

We shall use this sample firmware for this lab.

We shall try to understand a little better how binwalk works.

Binwalk searches for "magic numbers" (also known as File Signatures) in the .bin file which indicate the starting point of different file types

- By running: `binwalk DVRF_v03.bin` we see something like this:

```
cadmin@cadmin-virtual-machine ~/Firmware $ binwalk DVRF_v03.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             BIN-Header, board ID: 1550, hardware version: 4702, firmware version: 1.0.0, build da
te: 2012-02-08
32            0x20            TRX firmware header, little endian, image size: 7753728 bytes, CRC32: 0x436822F6, fla
gs: 0x0, version: 1, header size: 28 bytes, loader offset: 0x1C, linux kernel offset: 0x192708, rootfs offset: 0x0
60            0x3C            gzip compressed data, maximum compression, has original file name: "piggy", from Unix
, last modified: 2016-03-09 08:08:31
1648424       0x192728        Squashfs filesystem, little endian, non-standard signature, version 3.0, size: 609921
5 bytes, 447 inodes, blocksize: 65536 bytes, created: 2016-03-10 04:34:22
```

Here we see the different file signatures binwalk identified. If we wanted to try and find these manually instead, we can type: hexdump DVRF_v03.bin which will show us the hexcode for the entire binary. However this is a bit too much for us to process so we will filter what is necessary to show using:

- `hexdump -C DVRF_v03.bin | grep -i shsq`

```
cadmin@cadmin-virtual-machine ~/Firmware $ hexdump -C DVRF_v03.bin | grep -i shsq
00192720  2f 28 4f bb 24 20 3b 00  73 68 73 71 bf 01 00 00  |/(O.$ ;.shsq....|
```

Here we can see that shsq starts after 8 characters, so it is the 9th-12th hex values (73, 68, 73, 71) at addresses 00192728 to 0019272B inclusive.

If you do a bit of googling, you can find that some vendors use the magic number "shsq" for an lzma compressed Squashfs file system. Thus we can see that binwalk effectively searched through the hex of the binary file to find magic numbers like this, comparing against its own list of known magic numbers/file signatures. And then in the end, it output a nice list of all the ones it identified.

Next, hexdump the address where binwalk found the TRX header, and the gzip file.

- `hexdump -C DVRF_v03.bin | grep -i 00000020`
- `hexdump -C DVRF_v03.bin | grep -i 00000030`

```
cadmin@cadmin-virtual-machine ~/Firmware $ hexdump -C DVRF_v03.bin | grep -i 00000020
00000020  48 44 52 30 00 50 76 00  f6 22 68 43 00 00 01 00  |HDR0.Pv.."hC....|
cadmin@cadmin-virtual-machine ~/Firmware $ hexdump -C DVRF_v03.bin | grep -i 00000030
00000030  1c 00 00 00 08 27 19 00  00 00 00 00 1f 8b 08 08  |.....'..........|
```

With a bit of googling we can determine that the magic number 'HDR0' is related to TRX Headers, while the hex sequence 1f 8b 08 (which has no string representation) is for gzip files.

So, if we knew what file signatures we were searching for, we could find them in a .bin without binwalk, it just makes our life easier.

Extracting without binwalk:

Binwalk also enables us to extract the file system from the .bin, and we'll now see how we can do that without binwalk. Let's assume we used hexdump and identified the starting address of our filesystem, 00192728, because that is where we found 'shsq'.

We can use the 'dd' command to prune off the blocks before the starting address. We must convert the starting address to decimal to know how many blocks we wish to skip.

- We can use the command: `echo $((<hex value>))` to convert a hex value into decimal

```
cadmin@cadmin-virtual-machine ~/Firmware $ echo $((0x00192728))
1648424
```

- Then the `'dd'` command has multiple options, we can use 'if=<filename>' to specify the input file is the file with name <filename>, 'of=<outfile>' specifies that where to store the output of the split, 'skip=<num>' specifies how many blocks to skip before copying the rest of <filename> into <outfile>, and finally 'bs=1' specifies to read and write 1 byte at a time.
- Check "man dd" for the full manual of dd options.

```
cadmin@cadmin-virtual-machine ~/Firmware $ dd if=DVRF_v03.bin of=filesystem.bin skip=1648424 bs=1
6106328+0 records in
6106328+0 records out
6106328 bytes (6.1 MB, 5.8 MiB) copied, 7.3971 s, 826 kB/s
cadmin@cadmin-virtual-machine ~/Firmware $ ls
DVRF_v03.bin  filesystem.bin
```

Next since we know this is a squashfs filesystem, we can use 'unsquashfs' to extract it. If we use the '–l' (lowercase L) option, we can get a list of the contents of the filesystem.bin, while running it on the .bin with no options will extract the files.

If your system does not recognise the unsquashfs command, install squashfs tools:

- `sudo apt-get install squashfs-tools`

Upon running it, we are presented with this error:

```
cadmin@cadmin-virtual-machine ~/Firmware $ unsquashfs -l filesystem.bin
Parallel unsquashfs: Using 2 processors
lzma uncompress failed with error code 9
read_block: failed to read block @0x5d0c6b
read_fragment_table: failed to read fragment table block
FATAL ERROR:failed to read fragment table
```

- After googling the error code, we discover that certain vendors firmware will require that you use 'sasquatch' instead of 'squashfs'. Try extracting using sasquatch instead! https://github.com/devttys0/sasquatch

  For Ubuntu and Kali operating systems, you may need to use these commands to use a fork that fixes some issues:
  git clone https://github.com/threadexio/sasquatch.git
  cd sasquatch
  ./build.sh

Next we will experiment with an automated tool called 'Firmwalker' which attempts to identify files which may be of interest within an extracted firmware. To download it, enter in:

- `git clone https://github.com/craigz28/firmwalker.git`
- `cd firmwalker`

and then run ./firmwalker.sh giving it the base directory of the extracted file-system as a parameter when executing it. E.g. (alter this to suit the locations you extracted to and installed firmwalker in…)

- `./firmwalker.sh ../Firmware/squashfs-root/`

Try it on some of the DLink firmwares from Lab #1 which contained a filesystem (the DIR line of firmwares seemed to have a lot of file systems in them) and see if you find anything interesting!

Exercises:

1. Try to manually extract the file systems from some of the firmware from Lab #1, which binwalk was able to extract from.

2. Can you find any filesystems which binwalk was not able to detect/extract (e.g. in the firmware which only seemed to contain a small GIF or HTML file despite being ~3MB in size)? Use google to find other file signatures/magic numbers to search for.

    a. Note that how to determine the ending of a file is different depending on the type. For example, with Bitmaps the 4 bytes after the signature (which is 42 4D) specify the length of the file. For jpeg, the signature is FF D8 FF and we can determine the end of the file by locating the trailer FF D9.
    b. Helpful link: https://web.archive.org/web/20210503094655/http://www.devttys0.com/2011/05/reverse-engineering-firmware-linksys-wag120n/
    c. I make no guarantees that it is possible to find a filesystem in the cases where binwalk failed!

3. Instead of a whole file system, can you extract some file(s) which binwalk did not seem to? e.g. lzma encrypted files, GIF files, HTML files, etc…