Firmware Analysis and Simulation

Prabhankar Tripathi
Lucideus Technologies

What is a Firmware

Firmware is a set of programs which are stored on a hardware device in order to perform several tasks which a manufacturer wants that device to perform.

Like for example suppose you are operating your smartphone and want to take a picture, what you basically do is you just click on the capture icon present on the touch screen and that camera (which is actually a hardware) takes your beautiful picture. But in the complex world of back-end there is a superhero working around making it possible to pass such information using hardware. That hero is your Firmware.Let's dive into another example. You just open up your desktop. It takes time to boot up that all booting procedure, placing your crucial files into places and making your device functional or operational, all are done by your firmware.

There are various fields where firmware has spread its roots like any networking device like routers,switches uses firmware, your smartphones (android/ios) have firmware installed, your laptops,desktops, smart watches, cameras all of them use firmware.

Advantages of doing Firmware analysis

As we have seen that firmware have a huge applicability.It also comes with vulnerabilities which can lead to :

- Sensitive data exposures like passwords ,API keys,private certificates etc.
- Compromising devices and tampering with data.
- Replicating the firmware image with malicious backdoor embedded.
- Understanding the working of the firmware.

In this paper we will be going with the extraction of a firmware and simulating it in order to perform further pentesting without actually buying one.

Environment used :

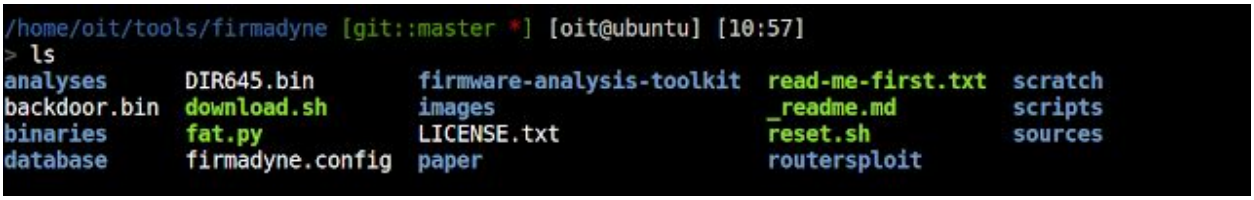
1.IOT pentesting OS named attifyOS you can have it from here : It has all the tools needed to perform analysis on any firmware.

Let's begin :In this paper we will be analysing a router firmware you can take up any other device for analysis.

Step 1: Download the firmware which you want to analyse and simulate.
There are various manufacturers who provide their firmware online so you can download them from there like:
Dlink : http://dlink.co.in/firmware/ftp.aspx
Netgear : https://www.netgear.com/support/download/
Tp-Link : https://www.tp-link.com/in/support/download/
or Google it .

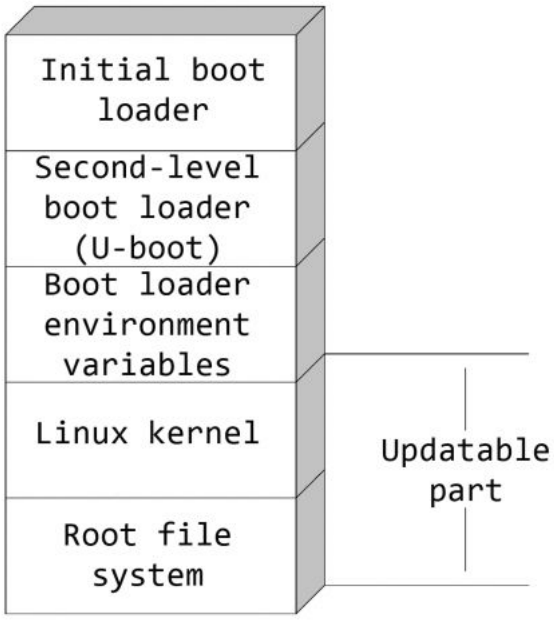
The firmware which I am going to use in this paper is Dlink-DIR645 .

Once you downloaded the firmware, rename it to something simple like **DLINK645.bin** .



DIR645.bin file in firmadyne folder

Step 2 : Analyse the firmware by using a tool like **binwalk** to understand what are the addresses of various segments in the firmware. Most importantly knowing the file system type because it will help us to further during the extraction scenario.



Firmware segments

In the above diagram our major focus needs to be in the lower segment i.e. Root file system because it is the one which contains major files and data of the device.Lets first fetch the information about the firmware using binwalk.

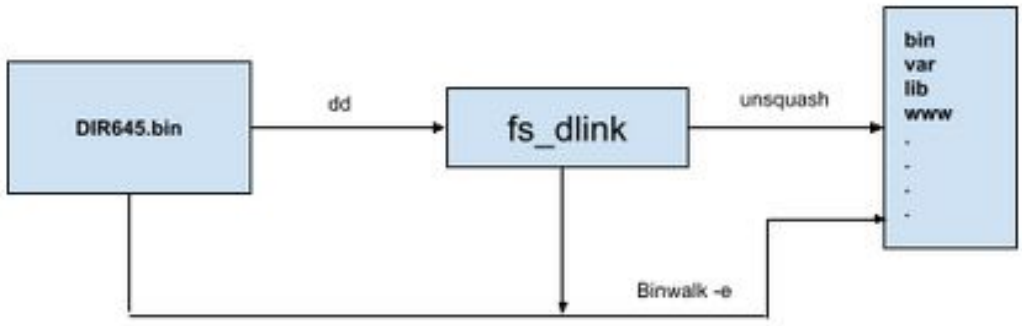
```
/home/oit/tools/firmadyne [git::master *] [oit@ubuntu] [11:14]
> binwalk DIR645.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             DLOB firmware header, boot partition: "dev=/dev/mtdblock/2"
112           0x70            LZMA compressed data, properties: 0x5D, dictionary size: 33554432 b
ytes, uncompressed size: 4229096 bytes
1441904       0x160070        PackImg section delimiter tag, little endian size: 15751680 bytes;
big endian size: 5959680 bytes
1441936       0x160090        Squashfs filesystem, little endian, version 4.0, compression:lzma,
size: 5958022 bytes, 1955 inodes, blocksize: 65536 bytes, created: 2011-11-23 03:10:33
```

Binwalk result of Firmware

From the above result we can see that this firmware is using **LZMA compression** and the file system used is **Squashfs** which starts from the address **1441936** (in decimals).

Things to know:
The common file system which we typically encounter in our IOT devices are: SquashFS, CramFS, JFF S2, YAFF S2, EXT2. On the top the different file systems, there are also varying types of compressions in use.
Some of the common compression which we see in IOT devices are: LZMA, GZIP, normal ZIP, Zlib, ARJ.

Step 3 : Now as we have ample amount of information regarding our firmware we will now start extracting the firmware. There are ample ways to extract the firmware so here we will describe it in two ways.



WAY 1 : By first extracting the file system form the firmware and then extract it .

We simply use dd and segregate a specific file system from firmware and then use tools to extract firmware.

dd if=<firmware_name>.bin skip=<offset in decimal> bs=1 of=<output_filename>

where **if** denotes input file ; **of** denotes output file ; **bs** : block size (in kb by default)
skip denotes after how many offset it should start extracting.

```
/home/oit/tools/firmadyne [git::master *] [oit@ubuntu] [12:10]
> dd if=DIR645.bin skip=1441936 bs=1 of=Dlink.bin
5959680+0 records in
5959680+0 records out
5959680 bytes (6.0 MB) copied, 17.0583 s, 349 kB/s
```

Output after running dd command

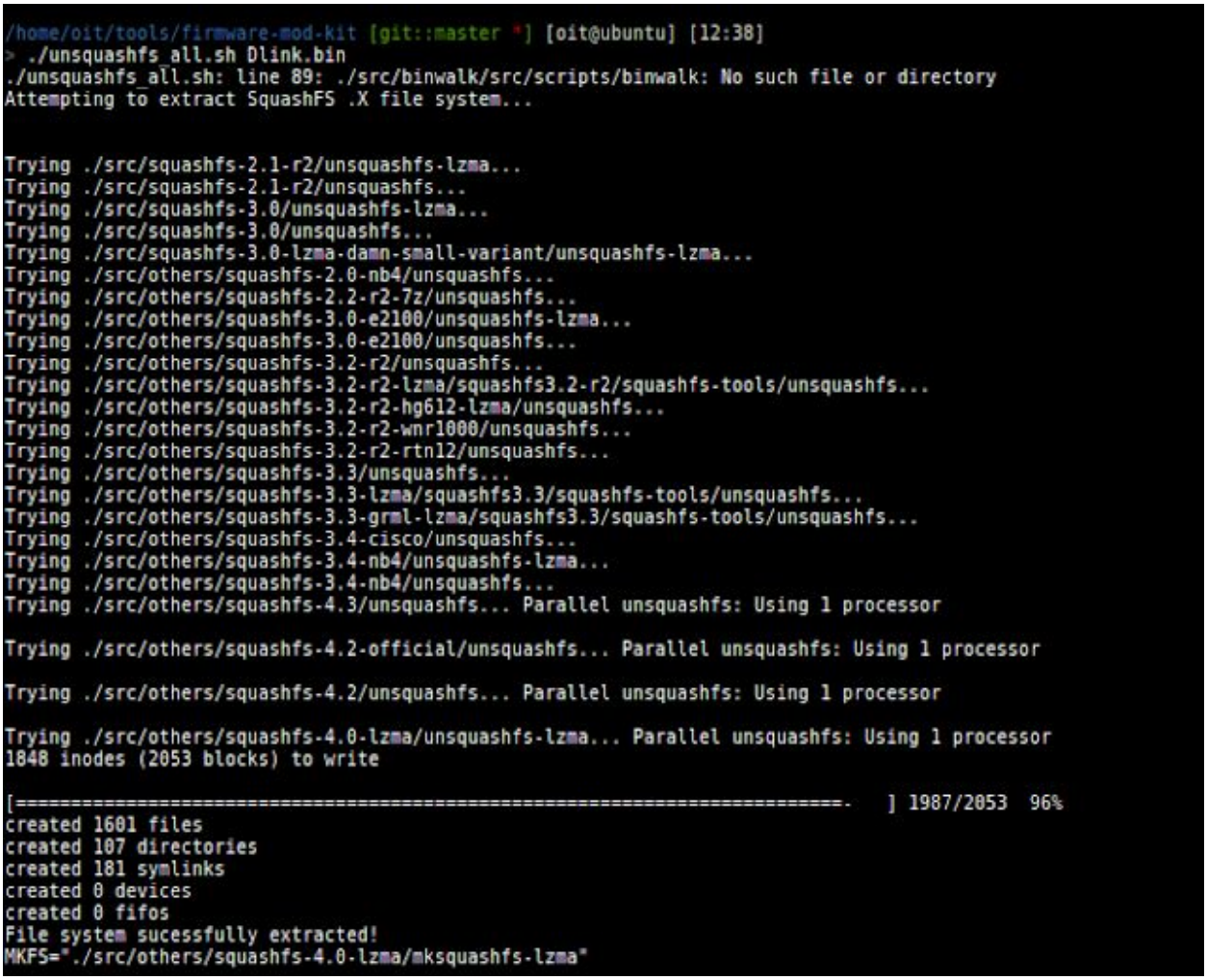
Check the output file you will notice only the file system has been extracted.

```
/home/oit/tools/firmadyne [git::master *] [oit@ubuntu] [12:34]
> binwalk Dlink.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Squashfs filesystem, little endian, version 4.0, compression:lzma,
size: 5958022 bytes, 1955 inodes, blocksize: 65536 bytes, created: 2011-11-23 03:10:33
```

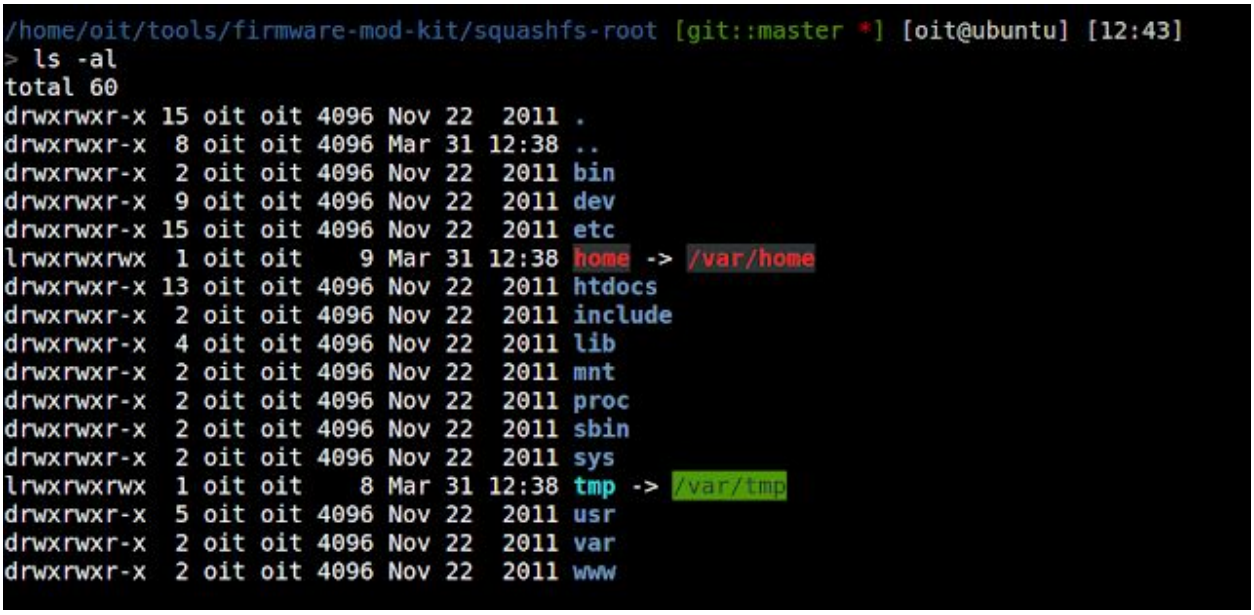
Now simply extract it using unsquashfs_all.sh present in /tools/firmware-mod-kit

```
/home/oit/tools/firmware-mod-kit [git::master *] [oit@ubuntu] [12:31]
> ls
build-firmware.sh           Dlink.bin                    ipk_template      src
check_for_upgrade.sh        Dlink_firmware.bin           kernel.md5        telnetd.sh
cleanup.sh                  extract-firmware.sh          other-scripts     uncpio.sh
common.inc                  firmware_mod_kit_version.txt PGDCSLtelnet.bin uncramfs_all.sh
creating_ipkg_packages.htm  ipkg_install_all.sh          readme.md         unsquashfs_all.sh
ddwrt-gui-extract.sh        ipkg_install.sh              root_fs.md5       vmlinux.gz.uImage
ddwrt-gui-rebuild.sh        ipkg_remove_all.sh           shared.inc
deprecated                  ipkg_remove.sh               shared-ng.inc
```

After running the command ./unsquashfs_all.sh DLink.bin we get a folder named **squashfs-root**



As you move inside it you will see a folder which seems similar to that of the root directory as in Linux systems.



NOTE :
For CPIO archive files
$ cpio -ivd --no-absolute-filenames -F <bin>

For jffs2 filesystems
$ jefferson rootfsfile.jffs2

For ubifs filesystems with NAND flash
$ ubireader_extract_images -u UBI -s <start_offset> <bin>
$ ubidump.py <bin>

WAY 2: Another way of extracting a firmware is very simple we can simply use binwalk -e <firmware name>.bin the extracted folder will contain **squashfs-root** folder going into it you will get the extracted file system.

```
/home/oit/tools/firmadyne [git::master *] [oit@ubuntu] [12:49]
> binwalk -e DIR645.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             DLOB firmware header, boot partition: "dev=/dev/mtdblock/2"
112           0x70            LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes, uncompress
ed size: 4229096 bytes
1441904       0x160070        PackImg section delimiter tag, little endian size: 15751680 bytes; big endian size:
 5959680 bytes
1441936       0x160090        Squashfs filesystem, little endian, version 4.0, compression:lzma, size: 5958022 by
tes, 1955 inodes, blocksize: 65536 bytes, created: 2011-11-23 03:10:33


/home/oit/tools/firmadyne [git::master *] [oit@ubuntu] [12:49]
> ls
analyses      DIR645.bin            fat.py                LICENSE.txt       reset.sh      sources
backdoor.bin  _DIR645.bin.extracted firmadyne.config      paper             routersploit
binaries      Dlink.bin             firmware-analysis-toolkit read-me-first.txt scratch
database      download.sh           images                _readme.md        scripts
```

Now as we have extracted the file system lets search for some sensitive files like : etc/shadow and etc/passwd **or** list out the etc/ssl directory **or** search for SSL related files such as .pem, .crt, etc. **or** search for configuration file **or** look for script files **or** search for other .bin files **or** look for keywords such as admin, password, remote, AWS keys, etc.

Lets try out searching stuff related to telnet so for that we can use grep command like
grep -iRn "telnet"

In that search result we got something like
telnetd -l /usr/sbin/login -u Alphanetworks:$image_sign ---> **etc/init0.d/S80telnetd.sh**

On opening the file we came up to the following result :

```
File Edit View Search Terminal Help
  GNU nano 2.2.6                    File: S80telnetd.sh

#!/bin/sh
echo [$0]: $1 ... > /dev/console
if [ "$1" = "start" ]; then
        if [ -f "/usr/sbin/login" ]; then
                image_sign=`cat /etc/config/image_sign`
                telnetd -l /usr/sbin/login -u Alphanetworks:$image_sign -i br0 &
        else
                telnetd &
        fi
else
        killall telnetd
fi
service ITUNES restart
```

Through the above screen we can see that Telnet username was **Alphanetworks** and password was saved as a variable named image_sign which was reading a file named image_sign

```
/home/oit/tools/firmadyne/_DIR645.bin.extracted/squashfs-root [git::master *] [oit@ubuntu] [13:05]
> cat ./etc/config/image_sign
wrgn39_dlob.hans_dir645
```

So there are so many other things which an analyst can approach after extracting a firmware.

Simulation of a Firmware

Now we know how to extract a firmware and pull out sensitive content from the extracted file system. Next we will be focusing on simulating the firmware on the browser user interface for its web application based exploitation and along with that we will also try to backdoor a firmware so that we can get the access of the firmware when we are not in the network though.

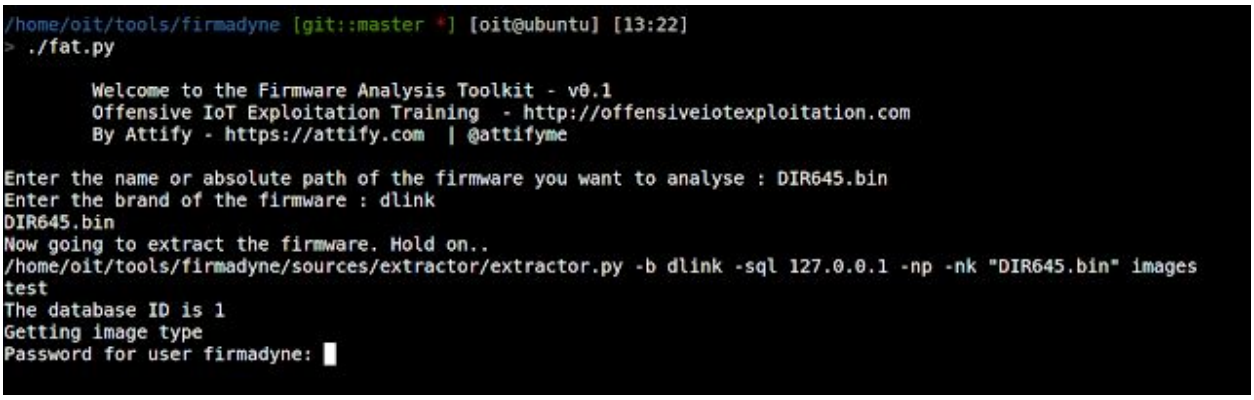
So for the simulation we will be focusing on majorly a fantastic framework named firmadyne and which is embedded with firmware-analysis-toolkit in attifyOS.

Step 1 : Here I am using the same firmware for simulation which was used for extraction in part-1 of this paper i.e. **DIR645.bin**
You can download it from here.

Step 2 : Run ./fat.py file present in **/home/oit/tools/firmadyne** folder and fill out the details needed like name of the file and name you want it to be stored in the database (you can give according to your choice) then it will ask for database password which is **firmadyne** .

This password will be asked by the user three times and sometimes it also asks for the OS password as well which is by default **attify123 .**

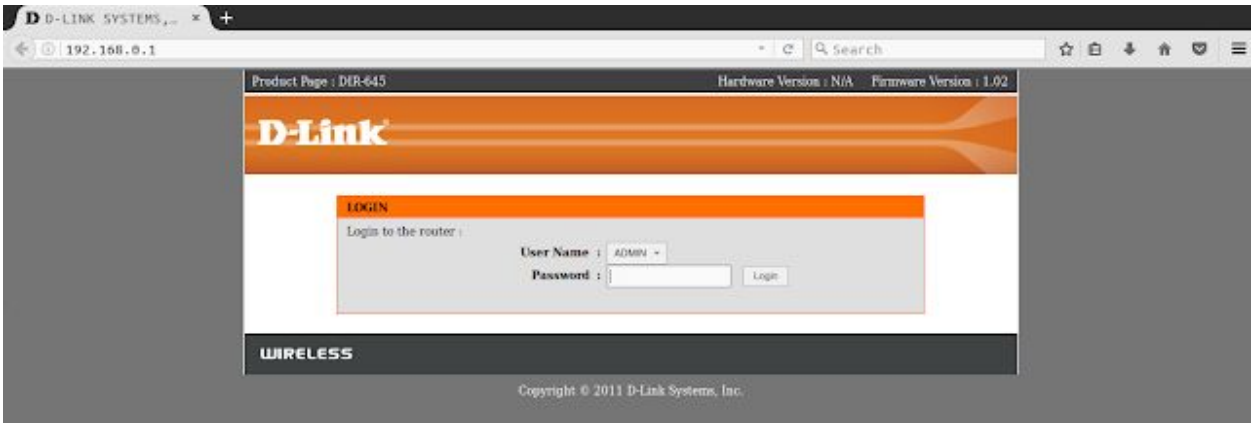
NOTE: Make sure your binary file needs to be in the same folder as that of fat.py



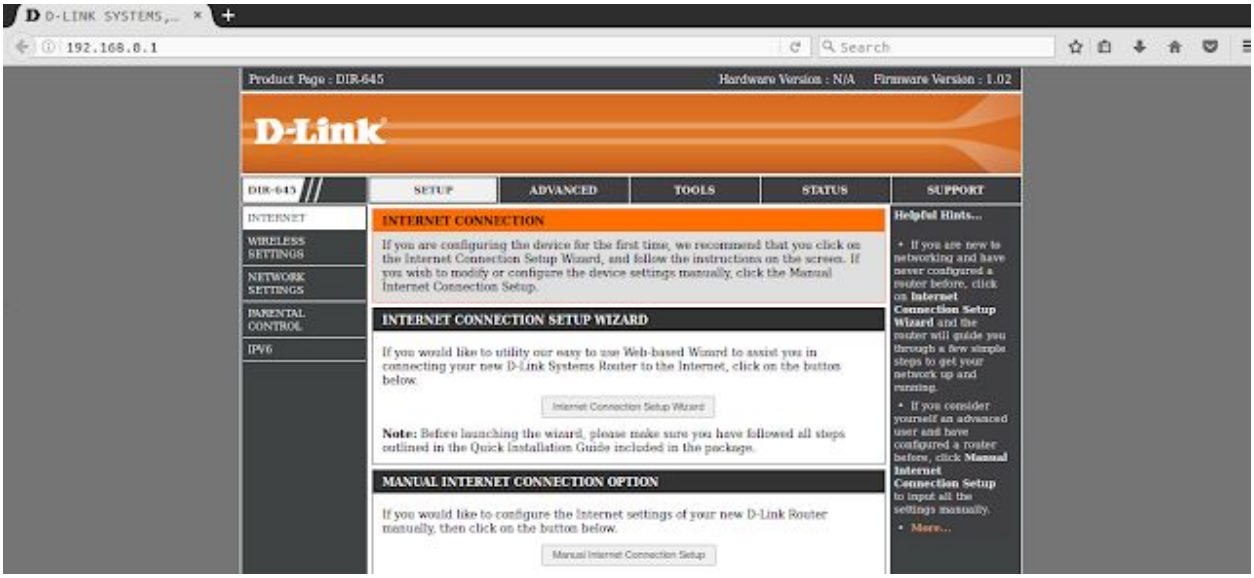
When the password is asked for the third time it will stuck for a while to create an interface and will take around 60 sec. **Don't press enter during that waiting time otherwise it will stop the simulation.**

```
Everything is done for the image id 1
Setting up the network connection
Password for user firmadyne:
qemu: terminating on signal 2 from pid 11085
Querying database for architecture... mipsel
Running firmware 1: terminating after 60 secs...
Inferring network...
Interfaces: [('br0', '192.168.0.1'), ('br1', '192.168.7.1')]
Done!

Running the firmware finally :
```

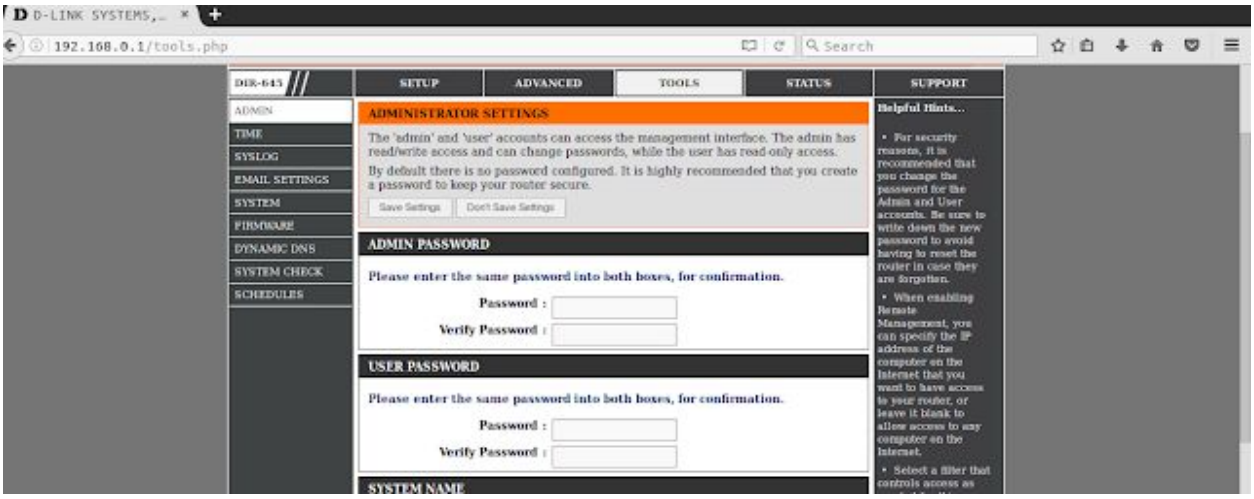
When you see that firmware is finally running just above that it will also give you the IP address to interact with that firmware like here in this case it is 192.168.0.1. Let's try to open this link in the browser.



The default password is set to blank just press login and you are inside the admin panel.

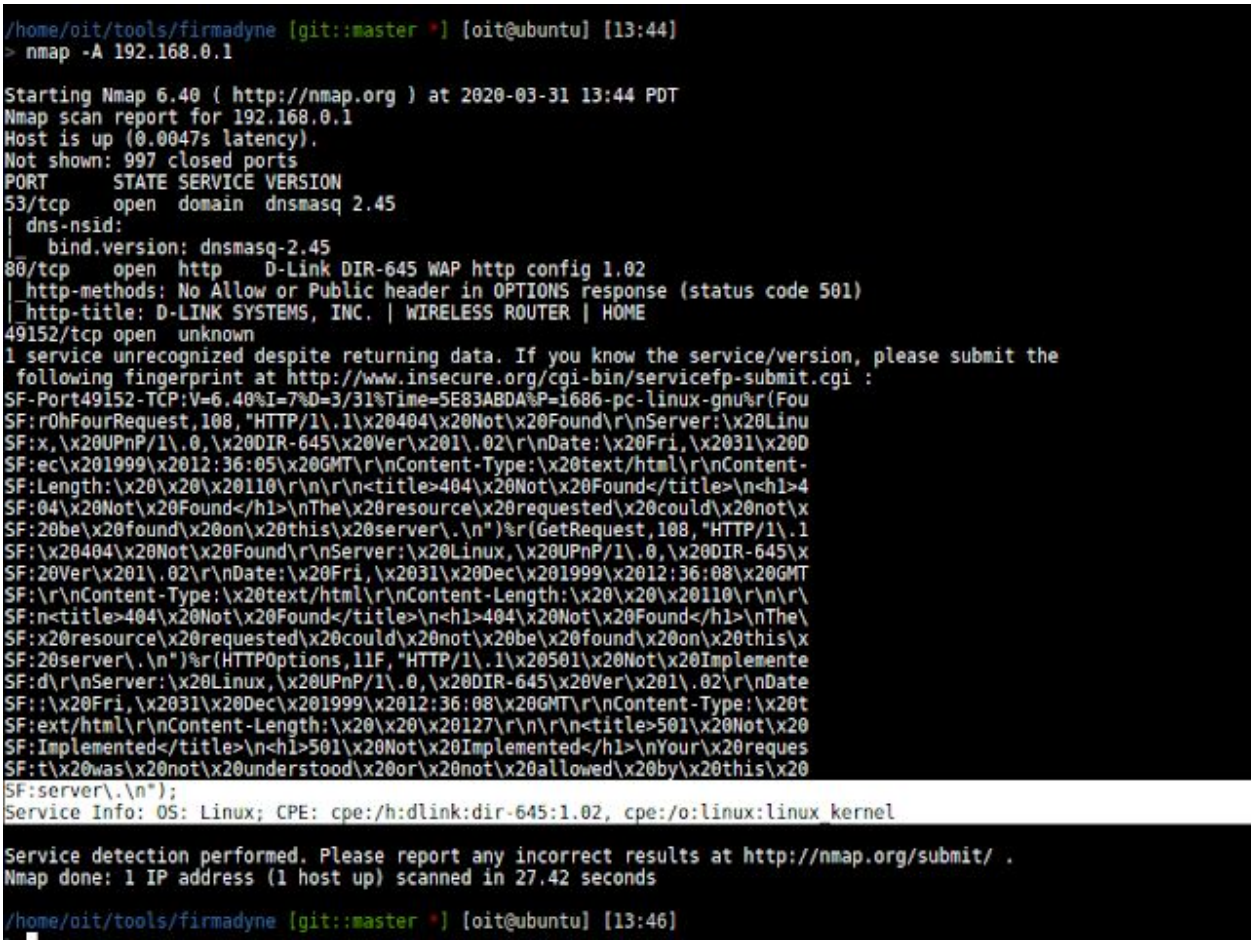


While browsing through it when I started with web pentesting I browsed through **TOOLS** tab which was vulnerable to **csrf** attack as no re-authentication was asked and no csrf token were used a malicious insider can change the admin password if he/she wants to.



This was just one type of web based attack that can be performed. There are many for reference; one can prefer OWASP TOP-10.

As it has also been provided with an IP address like in this case 192.168.0.1 you can try for network based attacks or scanning techniques which might give any network based vulnerability. like here's a screenshot from a **nmap scan** providing us the ports open and operating system architecture running.



You can also pentest it from other frameworks like routersploit which will find vulnerabilities in these devices very easily.

There are ample ways an attacker can exploit such firmwares. One more way of exploiting a framework is by simple adding up your bindshell file in one of the startup programs in the device directory and rebuild the firmware and when any one updates their firmware they will end up giving reverse shell of their router and this can even be possible to perform globally by using an account using cloud servers.