

# 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 .

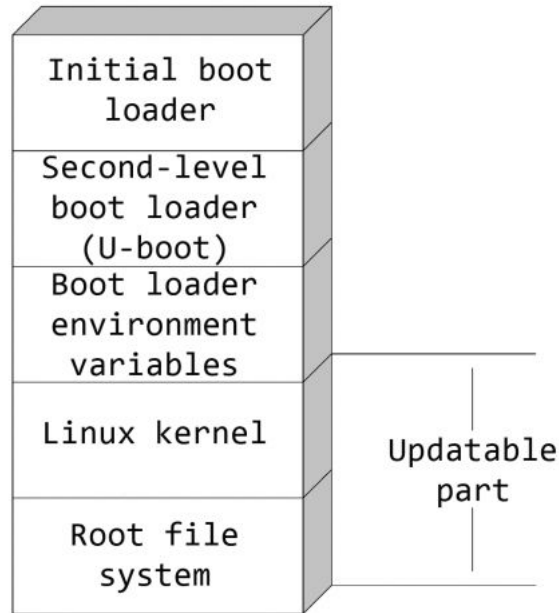
The firmware which I am going to use in this paper is [Dlink-DIR645](#) .

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

```
/home/oit/tools/firmadyne [git::master *] [oit@ubuntu] [10:57]
> ls
analyses      DIR645.bin      firmware-analysis-toolkit  read-me-first.txt  scratch
backdoor.bin  download.sh     images                    _readme.md         scripts
binaries      fat.py          LICENSE.txt              reset.sh            sources
database      firmadyne.config  paper                    routersploit
```

### **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: 0x50, 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**

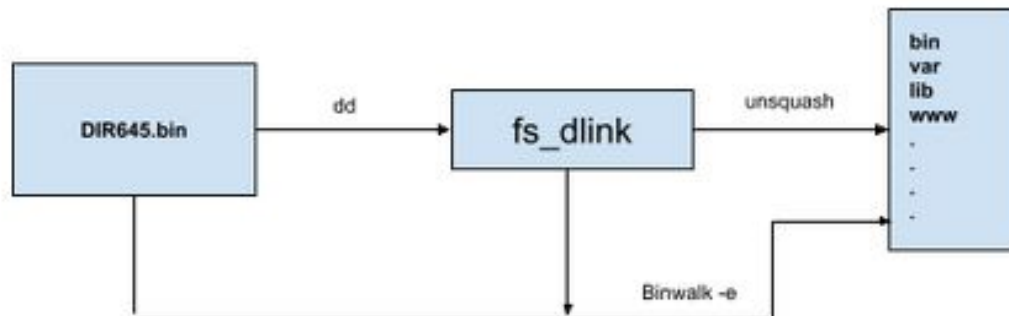
From the above result we can see that this firmware is using **LZMA compression** and the file system used is **Squashfs** which starts from the address **1441936** ( in decimals).

**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 from 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**

```

/home/oit/tools/firmware-mod-kit [git::master *] [oit@ubuntu] [12:38]
> ./unsquashfs all.sh Dlink.bin
./unsquashfs_all.sh: line 89: ./src/binwalk/src/scripts/binwalk: No such file or directory
Attempting to extract SquashFS .X file system...

Trying ./src/squashfs-2.1-r2/unsquashfs-lzma...
Trying ./src/squashfs-2.1-r2/unsquashfs...
Trying ./src/squashfs-3.0/unsquashfs-lzma...
Trying ./src/squashfs-3.0/unsquashfs...
Trying ./src/squashfs-3.0-lzma-damn-small-variant/unsquashfs-lzma...
Trying ./src/others/squashfs-2.0-nb4/unsquashfs...
Trying ./src/others/squashfs-2.2-r2-7z/unsquashfs...
Trying ./src/others/squashfs-3.0-e2100/unsquashfs-lzma...
Trying ./src/others/squashfs-3.0-e2100/unsquashfs...
Trying ./src/others/squashfs-3.2-r2/unsquashfs...
Trying ./src/others/squashfs-3.2-r2-lzma/squashfs3.2-r2/squashfs-tools/unsquashfs...
Trying ./src/others/squashfs-3.2-r2-hg612-lzma/unsquashfs...
Trying ./src/others/squashfs-3.2-r2-wnr1000/unsquashfs...
Trying ./src/others/squashfs-3.2-r2-rtn12/unsquashfs...
Trying ./src/others/squashfs-3.3/unsquashfs...
Trying ./src/others/squashfs-3.3-lzma/squashfs3.3/squashfs-tools/unsquashfs...
Trying ./src/others/squashfs-3.3-grml-lzma/squashfs3.3/squashfs-tools/unsquashfs...
Trying ./src/others/squashfs-3.4-cisco/unsquashfs...
Trying ./src/others/squashfs-3.4-nb4/unsquashfs-lzma...
Trying ./src/others/squashfs-3.4-nb4/unsquashfs...
Trying ./src/others/squashfs-4.3/unsquashfs... Parallel unsquashfs: Using 1 processor

Trying ./src/others/squashfs-4.2-official/unsquashfs... Parallel unsquashfs: Using 1 processor

Trying ./src/others/squashfs-4.2/unsquashfs... Parallel unsquashfs: Using 1 processor

Trying ./src/others/squashfs-4.0-lzma/unsquashfs-lzma... Parallel unsquashfs: Using 1 processor
1848 inodes (2053 blocks) to write

[=====] 1987/2053 96%
created 1601 files
created 107 directories
created 181 symlinks
created 0 devices
created 0 fifos
File system successfully extracted!
MKFS="./src/others/squashfs-4.0-lzma/mksquashfs-lzma"

```

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

```

/home/oit/tools/firmware-mod-kit/squashfs-root [git::master *] [oit@ubuntu] [12:43]
> ls -al
total 60
drwxrwxr-x 15 oit oit 4096 Nov 22  2011 .
drwxrwxr-x  8 oit oit 4096 Mar 31 12:38 ..
drwxrwxr-x  2 oit oit 4096 Nov 22  2011 bin
drwxrwxr-x  9 oit oit 4096 Nov 22  2011 dev
drwxrwxr-x 15 oit oit 4096 Nov 22  2011 etc
lrwxrwxrwx  1 oit oit   9 Mar 31 12:38 home -> /var/home
drwxrwxr-x 13 oit oit 4096 Nov 22  2011 htdocs
drwxrwxr-x  2 oit oit 4096 Nov 22  2011 include
drwxrwxr-x  4 oit oit 4096 Nov 22  2011 lib
drwxrwxr-x  2 oit oit 4096 Nov 22  2011 mnt
drwxrwxr-x  2 oit oit 4096 Nov 22  2011 proc
drwxrwxr-x  2 oit oit 4096 Nov 22  2011 sbin
drwxrwxr-x  2 oit oit 4096 Nov 22  2011 sys
lrwxrwxrwx  1 oit oit   8 Mar 31 12:38 tmp -> /var/tmp
drwxrwxr-x  5 oit oit 4096 Nov 22  2011 usr
drwxrwxr-x  2 oit oit 4096 Nov 22  2011 var
drwxrwxr-x  2 oit oit 4096 Nov 22  2011 www

```

#### 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, uncompressed size: 4229096 bytes
1441904     0x160070     PackImg section delimiter tag, little endian size: 15751600 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

/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`

```
/home/oit/tools/firmadyne [git::master *] [oit@ubuntu] [13:22]
> ./fat.py

Welcome to the Firmware Analysis Toolkit - v0.1
Offensive IoT Exploitation Training - http://offensiveiotexploitation.com
By Attify - https://attify.com | @attifyme

Enter the name or absolute path of the firmware you want to analyse : DIR645.bin
Enter the brand of the firmware : dlink
DIR645.bin
Now going to extract the firmware. Hold on..
/home/oit/tools/firmadyne/sources/extractor/extractor.py -b dlink -sql 127.0.0.1 -np -nk "DIR645.bin" images
test
The database ID is 1
Getting image type
Password for user firmadyne: █
```

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.**

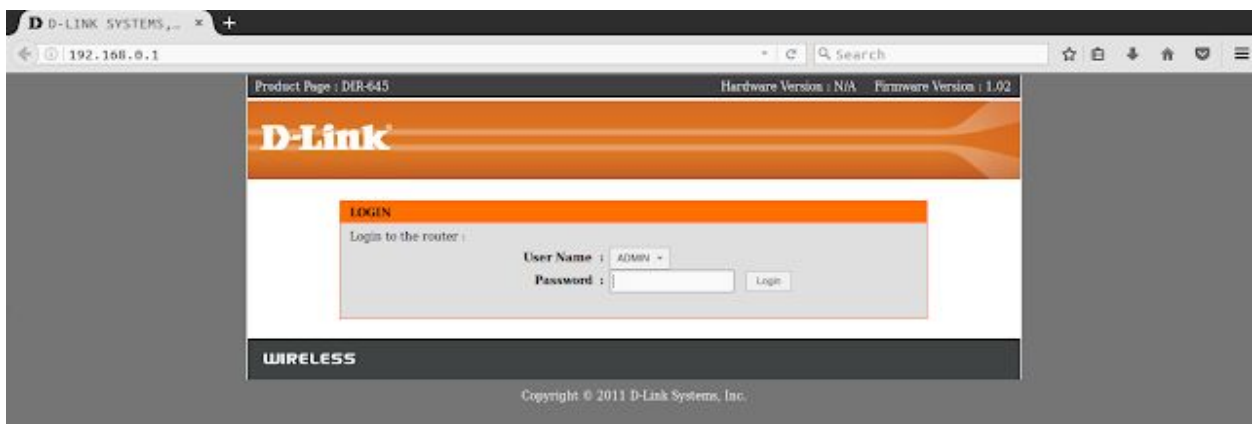
```

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!

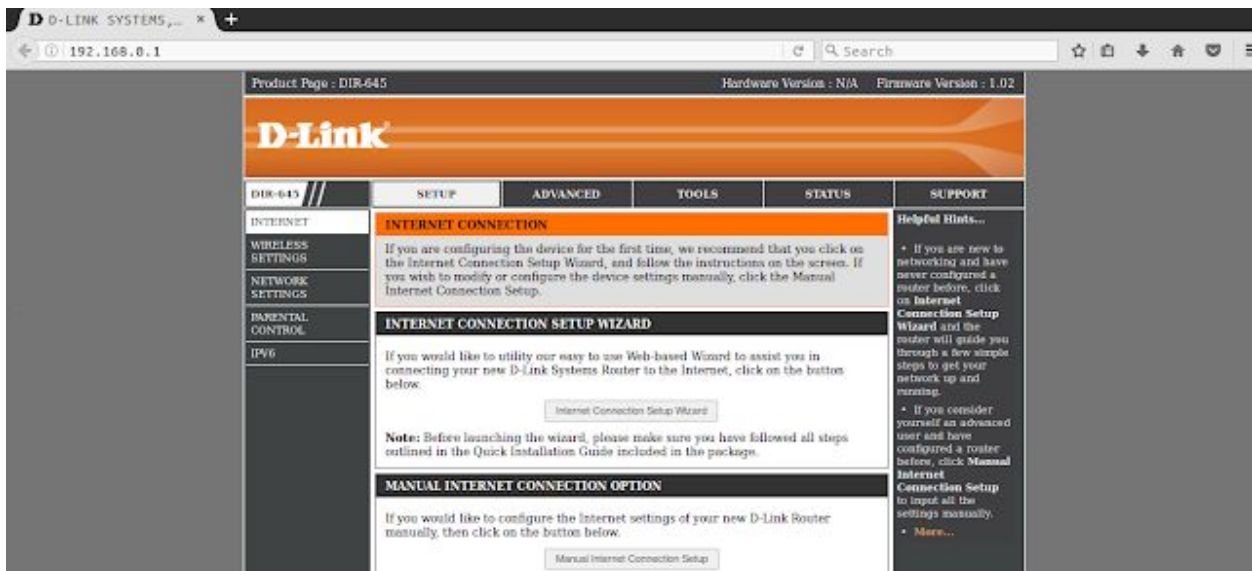
Running the firmware finally :

```

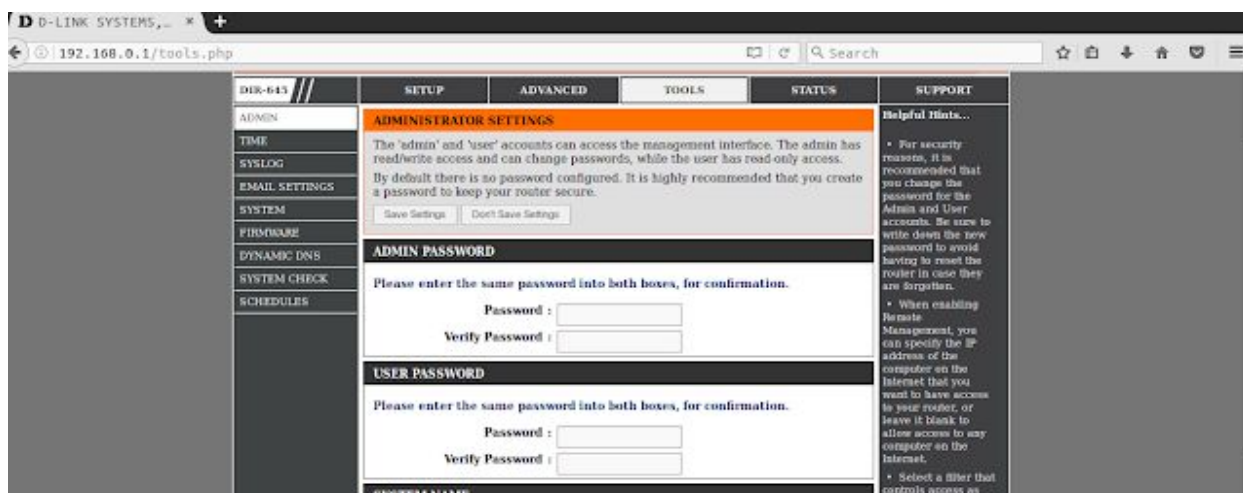
When you see that firmware is finally running just above that it will also give you the IP address to interact with that firmware like here in this case it is 192.168.0.1. Let's try to open this link in the browser.



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.

```

/home/oit/tools/firmadyne [git::master *] [oit@ubuntu] [13:44]
> nmap -A 192.168.0.1

Starting Nmap 6.40 ( http://nmap.org ) at 2020-03-31 13:44 PDT
Nmap scan report for 192.168.0.1
Host is up (0.0047s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
53/tcp    open  domain  dnsmasq 2.45
| dns-nsid:
|_ bind.version: dnsmasq-2.45
80/tcp    open  http    D-Link DIR-645 WAP http config 1.02
|_ http-methods: No Allow or Public header in OPTIONS response (status code 501)
|_ http-title: D-LINK SYSTEMS, INC. | WIRELESS ROUTER | HOME
49152/tcp  open  unknown
1 service unrecognized despite returning data. If you know the service/version, please submit the
following fingerprint at http://www.insecure.org/cgi-bin/servicefp-submit.cgi :
SF:Port49152-TCP:V=6.40%I=7%D=3/31%Time=5E83ABDA%P=I686-pc-linux-gnu%(Fou
SF:roHFourRequest,108,"HTTP/1.1\x20404\x20Not\x20Found\r\nServer:\x20Linu
SF:x,\x20UPnP/1\0,\x20DIR-645\x20Ver\x201\02\r\nDate:\x20Fri,\x2031\x20
SF:ec\x201999\x2012:36:05\x20GMT\r\nContent-Type:\x20text/html\r\nContent-
SF:Length:\x20\x20\x20110\r\n\r\n<title>404\x20Not\x20Found</title>\n<h1>4
SF:04\x20Not\x20Found</h1>\nThe\x20resource\x20requested\x20could\x20not\x20
SF:be\x20found\x20on\x20this\x20server\.\n")%(GetRequest,108,"HTTP/1.1
SF:\x20404\x20Not\x20Found\r\nServer:\x20Linux,\x20UPnP/1\0,\x20DIR-645\x
SF:20Ver\x201\02\r\nDate:\x20Fri,\x2031\x20Dec\x201999\x2012:36:08\x20GMT
SF:\r\nContent-Type:\x20text/html\r\nContent-Length:\x20\x20\x20110\r\n\r
SF:n<title>404\x20Not\x20Found</title>\n<h1>404\x20Not\x20Found</h1>\nTh
SF:x20resource\x20requested\x20could\x20not\x20be\x20found\x20on\x20this\x2
SF:0server\.\n")%(HTTPOptions,11F,"HTTP/1.1\x20501\x20Not\x20Implemente
SF:d\r\nServer:\x20Linux,\x20UPnP/1\0,\x20DIR-645\x20Ver\x201\02\r\nDate
SF::\x20Fri,\x2031\x20Dec\x201999\x2012:36:08\x20GMT\r\nContent-Type:\x20t
SF:ext/html\r\nContent-Length:\x20\x20\x20127\r\n\r\n<title>501\x20Not\x20
SF:Implemented</title>\n<h1>501\x20Not\x20Implemented</h1>\nYour\x20reques
SF:t\x20was\x20not\x20understood\x20or\x20not\x20allowed\x20by\x20this\x20
SF:server\.\n");
Service Info: OS: Linux; CPE: cpe:/h:dlink:dir-645:1.02, cpe:/o:linux:linux kernel

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 27.42 seconds

/home/oit/tools/firmadyne [git::master *] [oit@ubuntu] [13:46]

```

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.