(Reverse Engineering the Firmware Binary)

Extraction, Analysis and Modification of Router Firmware

Contents:

Dilpreet Singh Bajwa

Background: In the world of ever increasing interconnection of computing, mobile and embedded devices, their security has be-come critical. The security of embedded devices and their firmwares is the new trend in the embedded market. The security requirements and expectations for computingdevices are being constantly raised as the world moves to-wards the Internet of Things. This is especially true for embedded devices and their software counterpart – firmwares– which is also their weakest point.

IoT device Firmware Reverse Engineering: It is a process to understand the device architecture, functionality and vulnerabilities present in the device incorporating different methods.

Firmware: Piece of code written for specific hardware to perform different operations and control the device

Objective: The target device is a router which is running an embedded Linux OS. Routers are the main incoming and outgoing points to the outside world on a computer network, and as such are a prime location for sniffing traffic and performing man in the middle (MITM) attacks. If we control the router then we controls the network traffic.

We reverse engineer thefirmware binary of a Router, analyse and throw some light that how it can be customisedfor our purpose.

The purpose is to provide a demonstration that how we can reverse-engineer a firmware binary, extract the file system, pack it again, analyse it and customized it as per our requirement.

Strategy Followed:

- The firmware for the device can be downloaded from the vendor's website.
- Get as much information about the firmware by using tools like file, readelf, binwalk etc. and from online sources regarding file type, Boot Loader and its version, file system, Architecture, Version, Kernel version, compression used or any encryption used etc.
- File system and kernel can be extracted by using tools like dd, binwalk and firmware-mod-kit.
- Analyse the file system as the file system includes default settings and binaries from the device which then can then be reverse engineered for potential exploits.The purpose of this is to identify the architecture information of the device, as well as finding vulnerabilities and other potential avenues of attack.

Tools Used:

- **File tool**
- **Binwalk**
- **Readelf**
- **Firmware-mod-kit**
- **Firmware-analysis-Tool-Kit**

Demonstration of extracting, analysing and customising the firmware:

Figure: 1

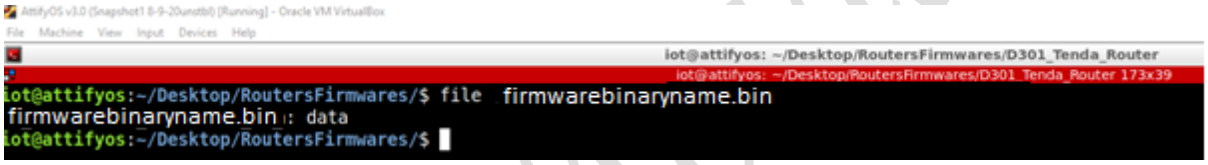


Figure 1 above shows the output of a tool called file, this tool attempts to classify a fileand tests if the file is in a certain format. Here it simply tells that the file type is data. We also ran the file tool on an extracted executable from the bin directory in the filesystem later on.

Figure2:

In Figure2 above, we have used “Binwalk” which is an open-source tool for analyzing, reverse engineering and extracting firmware images. Binwalk is able to scan a firmware image and search for file signatures to identify and extract filesystem images, executable code, compressed archives, bootloader and kernel images, file formats like JPEGs and PDFs, and many more!

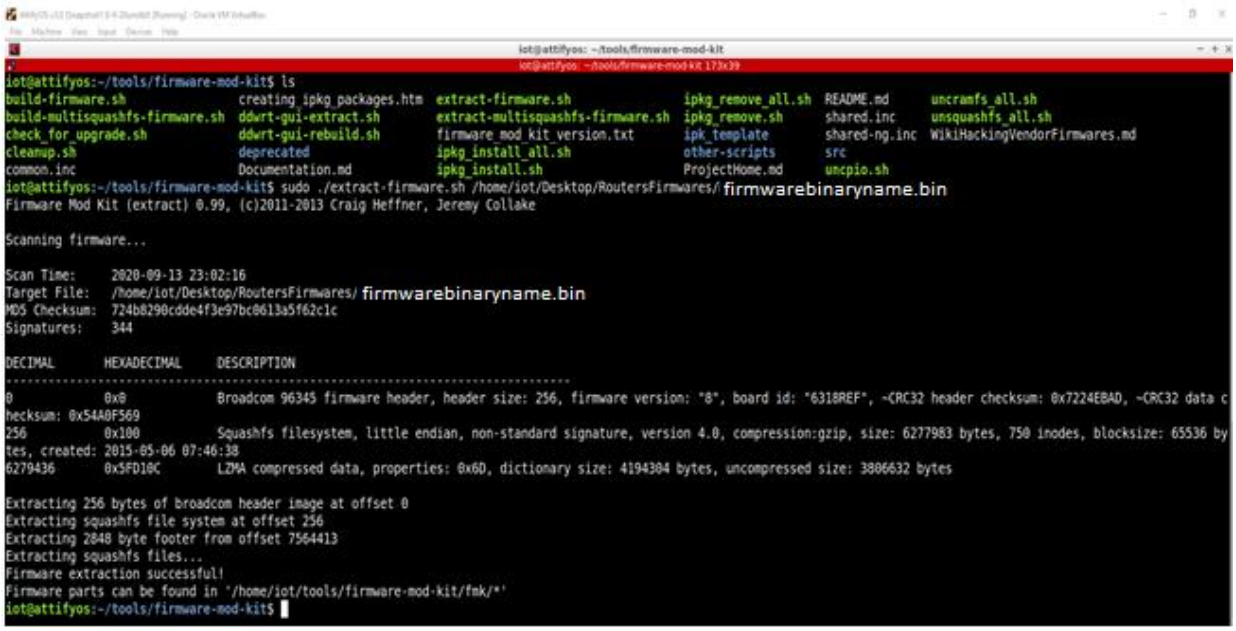
Both *file* and *binwalk* tools use the *libmagic* library to identify file signatures. But binwalk additionally supports a list of custom magic signatures to find compressed/archived files, firmware headers, Linux kernels, bootloaders, filesystems, and so on!

Binwalk has several options you can use but here we only simply use binwalk without any option to get useful information about our target firmware image. Binwalk structure has three sections:

1. File location in decimal format
2. File location in hexadecimal format
3. Description of what was found at that location

The image mentions that the firmware belongs to chipset Broadcom 96345 and also firmware version 8 in first line. And based on the image found at the address 0x100, we can see that the rootfs is a squashfs filesystem in little endian format. Other useful information like gzip compression and size is also shown. Further in last line at address 0x5D10C shows that lzma compression is used.

Figure 3:



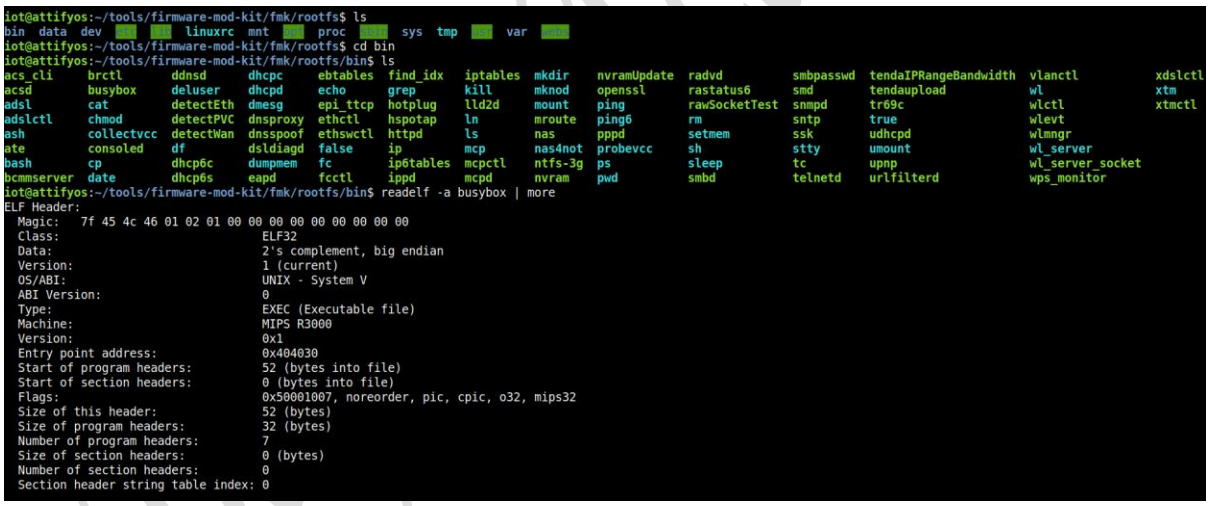
Now we can extract file system from the firmware image by either using dd command or by automatic extracting by using binwalk –e option or by using firmware-mod kit. Here in above figure we have used firmware-mod-kit as we also need to build it again after modification and in that case firmware-mod-kit provides us that functionality.

We extract the firmware by using ./extract-firmware.sh script of firmware-mod-kit and able to extract filesystem successfully. The firmware-mod-kit create a new directory “fmk” and store al extracted files in it.

Figure 4:

As shown above in figure 4, rootfs directory contains the complete filesystem of our firmware image which we can further analyse to find vulnerabilities and how we can modify it in our best interest.

Figure 5:



Now as shown in Figure 5, Figure 6 and Figure 7, we traverse various directories like bin, etc and webs which contains important files which we further analyse to find any loopholes which either helps us in creating any backdoor, bypass authentication or we can modify to brick the device. For example,there are some start-up scripts present in some directories which runs on start-up of the router and do various work like initialization and setting various parameters. We may modify such files to achieve our objective. (The name and location of such files is not mentioned here intentionally).

Figure 6:

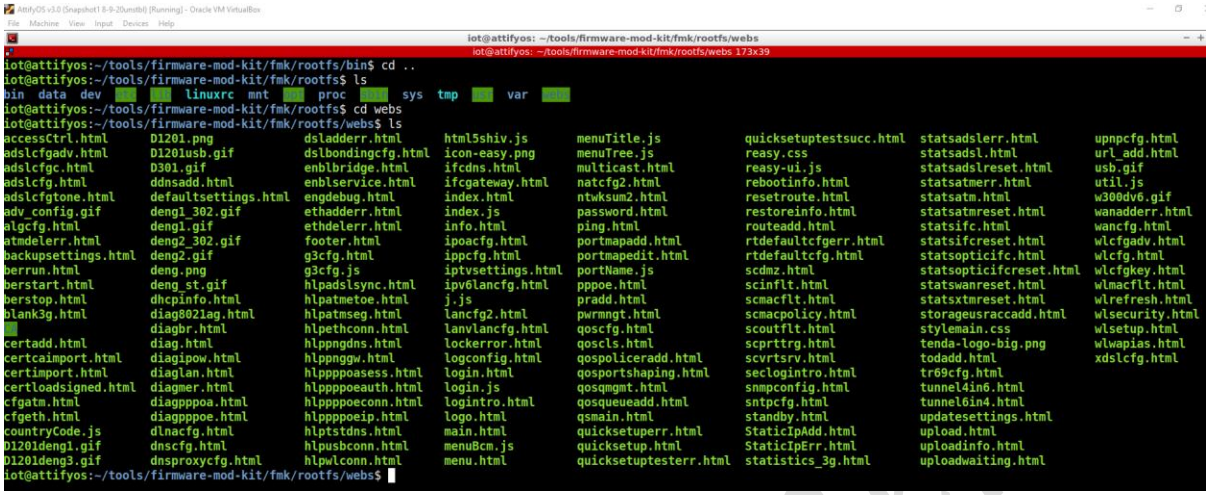


Figure 7:

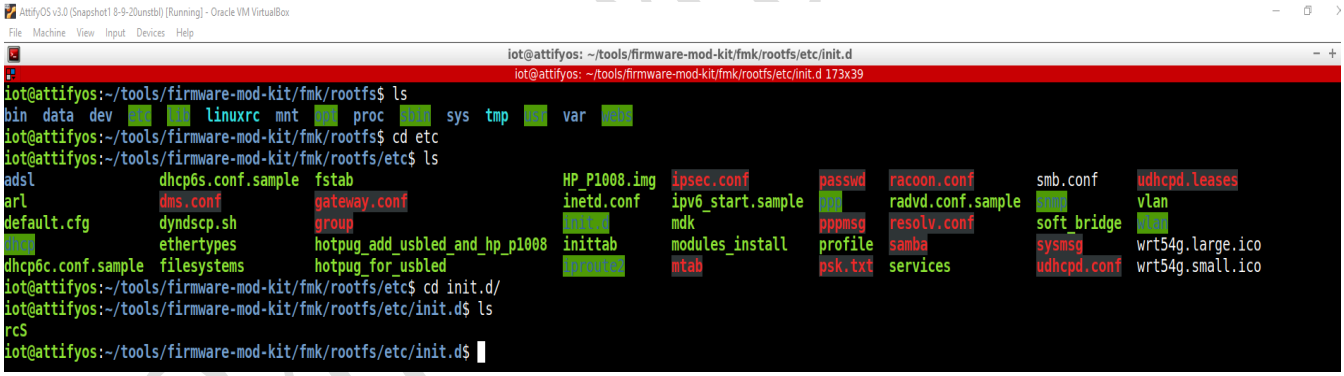
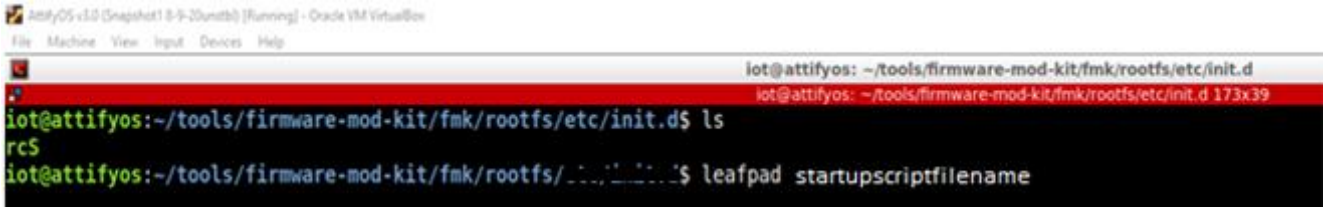
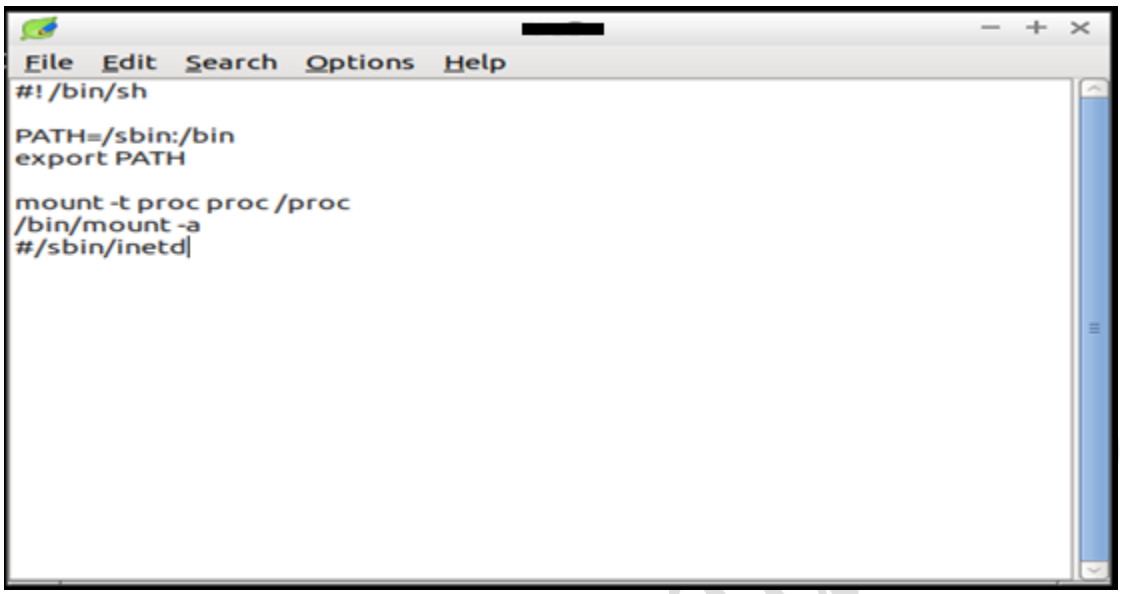


Figure 8:



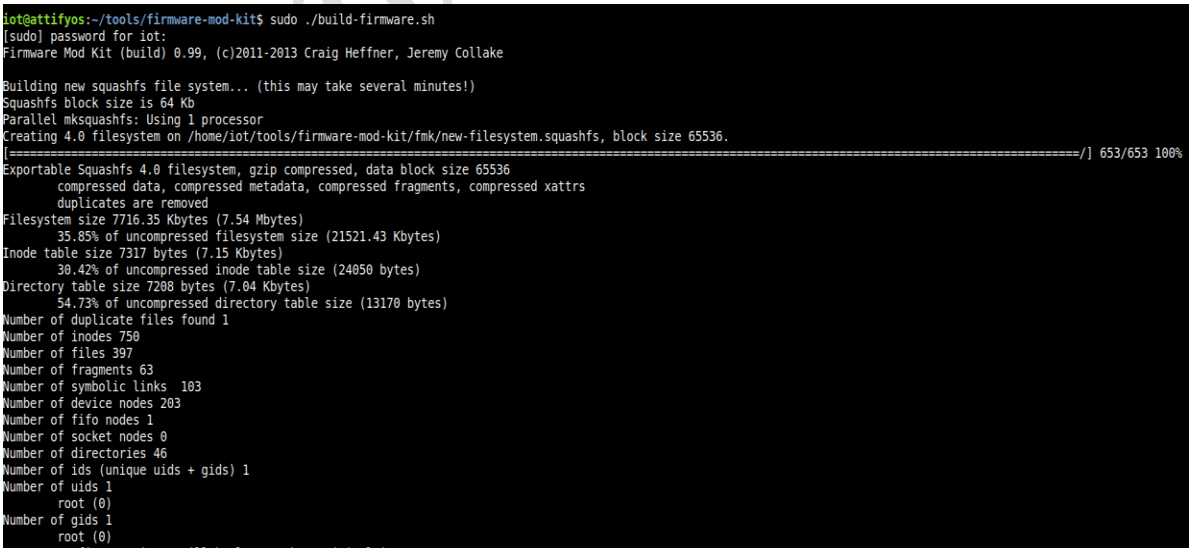
As shown above in Figure 8, we simply accessing the start-up script file and open it withleafpad. The contents of file are shown below in Figure 9.

Figure 9:



As we know from above discussion that this file is important and runs on router start-up and we also knew that the system is a Linux system. We further analyse busybox file present in /bin directory to know its version, commands it supports by using file and emulation software qemu.We can as per our requirement can write our customised code or script and can run through it on system start-up. It is just one method, there are number of files available and we further explore them as per our need. Our malicious process will run in background.

Figure 11:



After Modifying the required files, we simply build our new modified firmware again by using firmware-mod-kit as shown in figure 11 above by using script ./build-firmware.sh available in firmware-mod-kit and the new-firmware file is also stored in fmk directory with name new-firmware.bin as shown below in Figure12.

Figure 12:

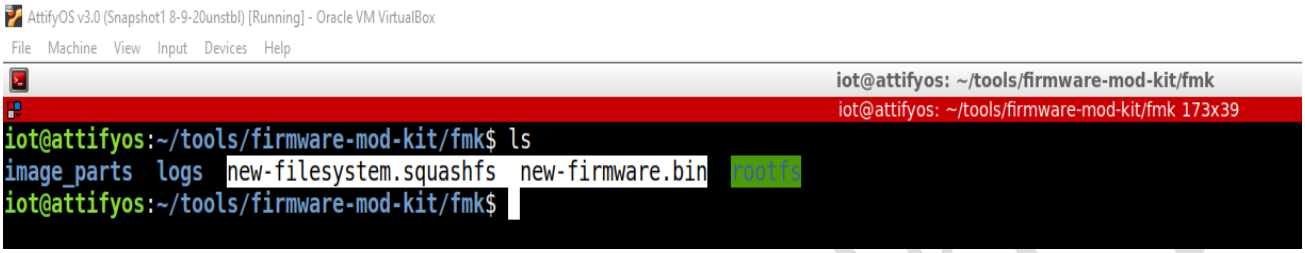
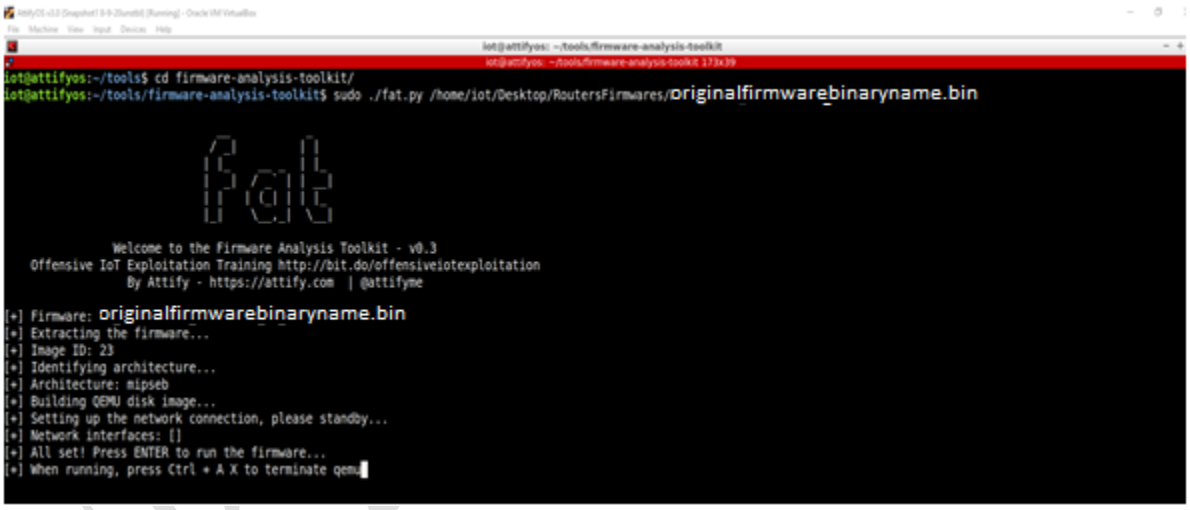


Figure 13:



We modify or customised the firmware binary in a way that after uploading our new customised binary to the router, the router become unusable that it does not work. Here we emulate it through firmware-analysis-tool kit which is based on qemu and automate the whole process.

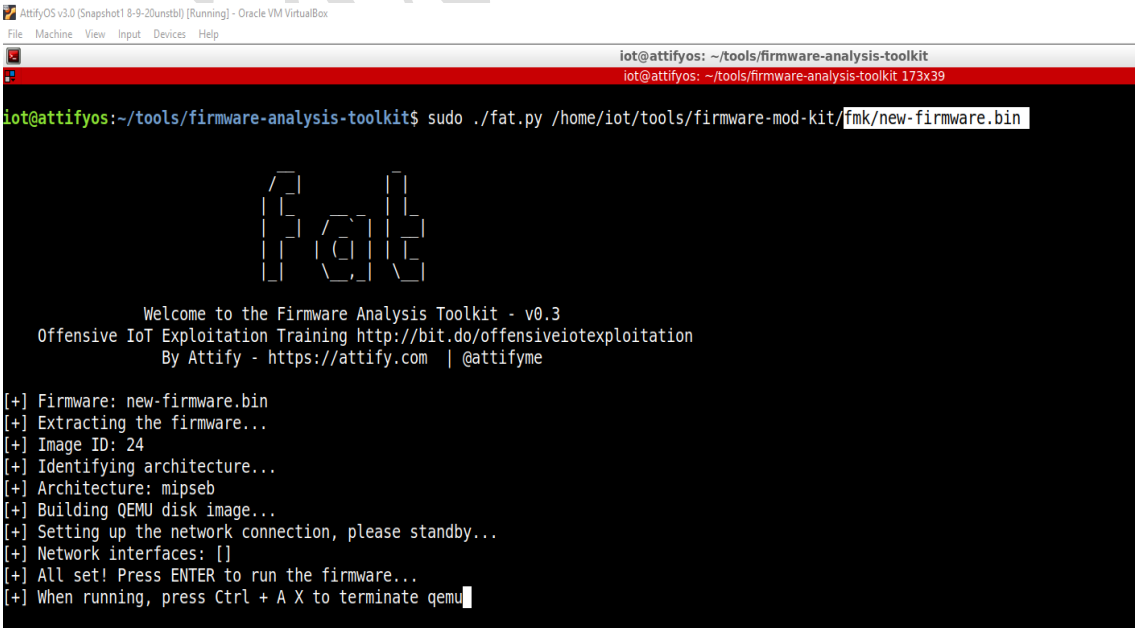
First we emulate our original firmware file as shown in Figure 13 and Figure 14 and finds that it works fine. We able to emulate it and also accessing its file system (Figure 14). This confirms us that the original firmware is working fine.

Now our next task is to emulate our modified new-firmware and check whether we can be able to brick the router or not with our modification.

Figure 14:

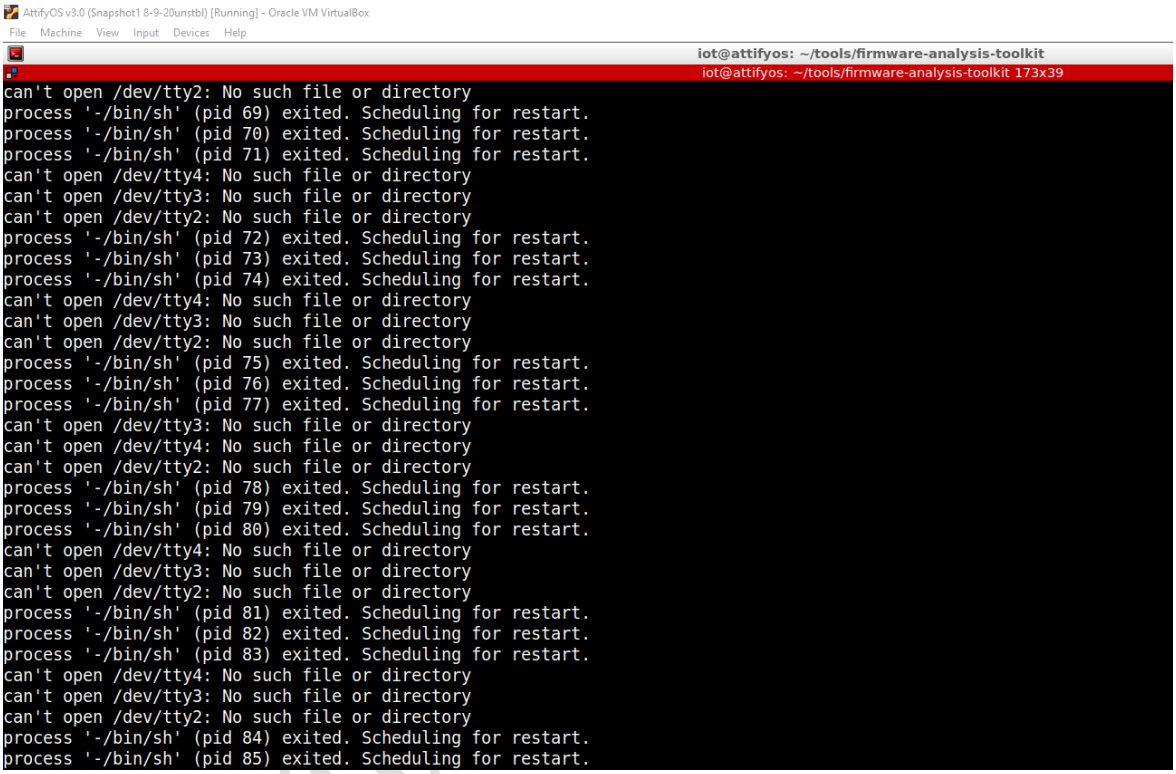
```
# pwd
/
# ls
bin         etc         linuxrc     opt         sys         var
data        firmadyne   lost+found  proc        tmp         webs
dev         lib         mnt         sbin        usr
# cd bin
# ls
acs_cli               ethswctl              radvd
acsd                  false                 rastatus6
adsl                  fc                    rawSocketTest
adslctl               fcctl                 rm
ash                   find_idx              setmem
ate                   grep                  sh
bash                  hotplug               sleep
bcmmserver            hspotap               smbd
brctl                 httpd                 smbpasswd
busybox               ip                    smd
cat                   ip6tables             snmpd
chmod                 ippd                  sntp
collectvcc            iptables              ssk
consoled              kill                  stty
cp                    lld2d                 tc
date                  ln                    telnetd
ddnsd                 ls                    tendaIPRangeBandwidth
deluser               mcp                   tendaupload
detectEth             mcpctl                tr69c
detectPVC             mcpd                  true
detectWan             mkdir                 udhcpd
df                    mknod                 umount
```

Figure 15:



As shown in Figure 15, we are now emulating our modified new-firmware file with firmware-analysis-tool-kit. As shown in Figure 16 below that emulation is failed because of our modification in the firmware and router is restarting again and again. Our purpose and objective is fulfilled.

Figure 16:



Mitigation or How to reduce the chance of Exploitation:

Vendors can perform a few simple steps to make life for the attacker a lot harder than it currently is:-

- Remove all debugging ports (serial, JTAG)
- Remove all unnecessary services such as telnetd
- Audit and lockdown the web interface
- Filter all input into the device.
- Encrypt and Sign firmware.

These steps, especially encrypting the firmware, make offline analysis difficult unless the encryption keys are leaked.

Conclusion:

Here, we have analysed, extracted the entire filesystem of the target Firmware. We further modified the firmware and check through emulation that it did not work after modification. Here we cover only main aspects for firmware extraction, analysis and modification.

Note: Here in the report, the original name of firmware binary, its version, firmware vendor name and start-up scripts files name and location are not mentioned intentionally for the purpose.