Firmware Hacking, Slash the Pineapple for Fun

smrx86
Independent Researcher
ramatrinanda@gmail.com

Abstract

Generally firmware hacking can be defined as an act to customizing firmware content that later have to rewrite/flash into rom memory of related devices. The aim of this act may variated, began to do patching/fix the security hole or even worse to planting malware inside it. In this paper firmware hacking os done to exploiting and porting pineapple Mark 5 firmware and flashing it into GL-inet router.

Keywords: Firmware hacking, reverse engineering, exploitation, flashing.

1. Introduction

At many different times, i often find paper and articles that gave suggestion "learning openwrt is a good way to understanding linux operating systems". With this system you can learn Linux fundamentals, networking, wireless security or even though firmware developing. Its all can be reach because the framework is available and it designed easier to build a firmware for many chipset platforms .[1]

More functionality device can be created with it. Lets takes sample: prototype device can be made with low cost and low power consumtion with a tiny router like TP-link mr3020. And i think because of it openwrt is already become favorite OS that choosen by many firmware developer.

Take a look Hak5, this team is wellknown as top hacking tools maker, if you know this is all because openwrt. Since early versions of pineapple, this OS already choosen to arm all its routers that they sold. Pineapple firmware development it self is quite impressive, more than dozen firmware for MK 1 s/d MK 5 already published during 2008 until 2015. The last version that been launched is ver 2.4.0 which intended for MK 5 costume router.

On other hand,,

Just like sugar among ant colony, hacker get excited to pineapple and hunt it to become an reasearch objects. They think openwrt inside pineapple router is the clue to develop his own penetration tools. Although lack of source code but it's not become an obstacle for them. They try many method to reach it and firmware hacking is one of them.

With the same motivation as mentioned above, this paper is been written. The goal is simple, give you all better explanation about firmware hacking accompanied by proof of implementation MK 5 firmware on GL-inet router. In this case i choose Gl-inet's because it have similar specification's to MK 5 router (CPU chipset, Wlan, ROM and RAM capacity). Besides that this router is really cheap and easy to get it in many online computer store.

1.1. The Concepts

Before going any further i think is good to know and understand the concept of fimware and firmware hacking method first.

1.1.a. Firmware and Firmware Hacking

Fimware hacking can be defined as essential software routines contained in ROM memory of a hardware device. Firmware is always responsible for basic operations, such as starting input / output procedure. Firmware used by almost electonics device which have ROM memory inside it. ex: dvd player. Handphone, digital camera, computer harddisk, medical equipment etc.[2]

To support renewal information or to repair of bug problem, firmware inside device is also equipped with ability to be rewritten or upgraded. This is where firmware hacking usually take an advantage.[3]

1 Alan Holt and Chi-Yu Huang, "Embedded Operating Systems, A Practical Approach," in introduction, virtualisation. London, Springer-Verlag, 2014, pp. 7.

2 Pinzaru. "What is firmware?.", internet:http://incepator.pinzaru.ro/software/what-is-firmware/, Apr. 2015[Jun. 6,2015].

Firmware hacking can be explained as a method to customize content of a firmware then rewritten (flashing) it into rom-related devices. A series of code that customized / modified may make for fixing bugs or even worse plant malware into that firmware. Firmware hacking process itself involves *reverse engineering* technique in the stage of it implementation.

Generally firmware hacking is done through several step below.[4]

1. Recognize/Reconnaisance firmware, Recognize is the stage where we dig as much as possible information contained in a firmware.
2. Extracting / unpacking firmware, at this stage we do upacking all files contained in a firmware.
3. Sorting and find unique files, all files then sorted to search some files that suspect as items need to be modificated later.
4. Debugging test, this is stage where we are trying to run the whole firmware or only specific files and then we take evaluation. Debugging test can be done in emulator or in a device which have same platform with real device.
5. Explotation & Final repacking, the last stage is exploitation which is follow-up results of debugging test. In this step we also repack all files become single firmware file again.

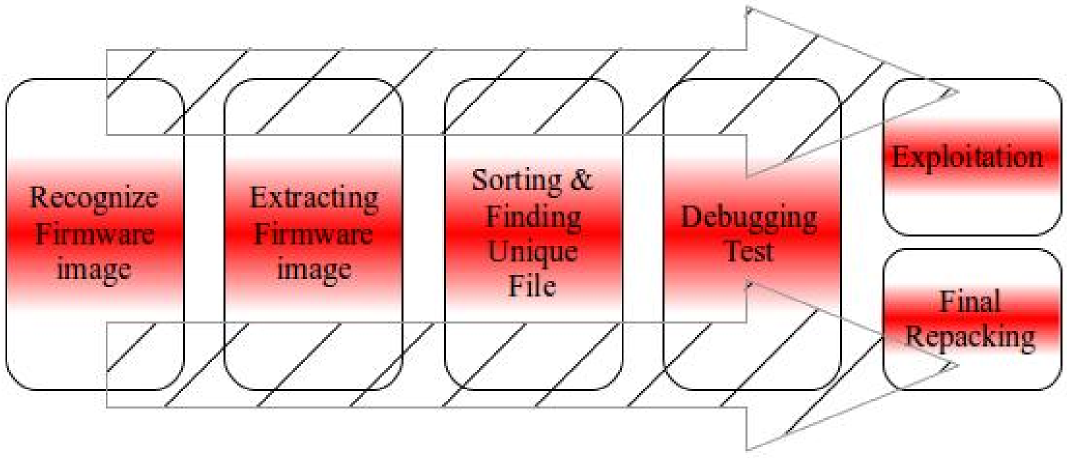


Fig 1. Stages of firmware hacking.

1.2. Basic Hardware Information.

Mark 5 router is a device that specifically made and marketed by hak5. Some features it have rarely found on other routers. Just say, they are sd card slot to store pineapple infusion, 5 pieces of dip switches and two WLAN chipset card in order to maximizing wireless networks penetration.

Here are the specifications and bundling of MK 5 router:[5]

CPU	400 MHz MIPS Atheros AR9331 version 1 SoC
Memory	16 MB ROM (w25q128 (16384 Kbytes)), 64 MB DDR2 RAM (Hynix H5PS5162GFR-Y5C)
Disk	Micro SD support up to 32 GB, FAT or EXT, 2 GB Included
Mode Select	5 DIP Switches 2 System, 3 User configurable
Wireless	Atheros AR9331 IEEE 802.11 b/g/n + Realtek RTL8187 IEEE 802.11 a/b/g
Ports	(2) SMA Antenna, 10/100 Ethernet, USB 2.0, Micro SD, TTL Serial, Expansion Bus
Power	DC in Variable 5-12v, ~1A, 5.5mm*2.1mm connector, International Power

3 Wikipedia. "Firmware.", internet: http://en.m.wikipedia.org/wiki/Firmware, last edited, Jun. 5, 2015[Jun. 10, 2015].

4 Silverbug and Redhidden. "Fu~n of Attacking Firmware .", avaiable: http://www.powerofcommunity.net/poc2012/re&si.pdf, 2012[Jun. 10, 2015].

5 Andy Davis. "Naked WiFi Pineapple Mark V!.", Internet: https://penturalabs.wordpress.com/2013/10/27/naked-wifi-pineapple-mark-v/, Oct. 27, 2013[Apr. 20,2013].

	Supply
Status Indicators	Power LED, Ethernet LED, Wireless 1 LED, Wireless 2 LED
	JTAG (Tags are on reverse of board)(IO6=TDI, IO7=TDO,IO8=TMS)
Default OS	Customized Openwrt
MTD partition info	
0x000000000000-0x000000020000 : "**u-boot**" 0x000000020000-0x000000102744 : "**kernel**" 0x000000102744-0x000000ff0000 : "**rootfs**" 0x000000bb0000-0x000000ff0000 : "**rootfs_data**" 0x000000ff0000-0x000001000000 : "**art**" 0x000000020000-0x000000ff0000 : "**firmware**"	

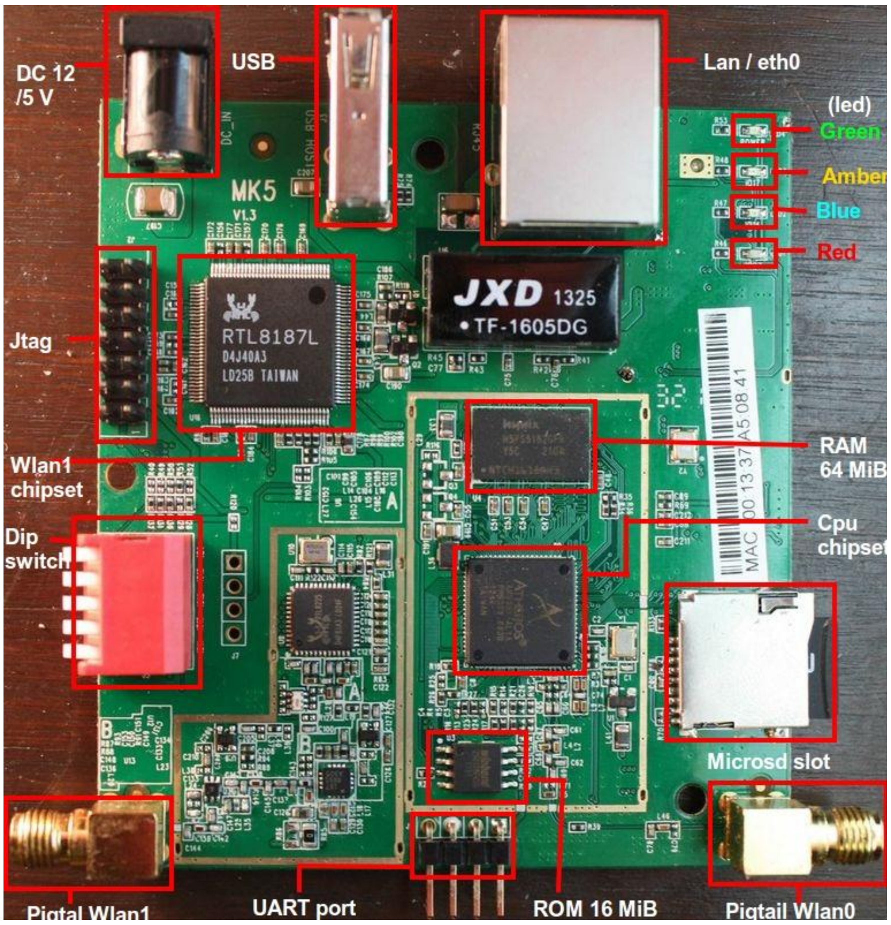


Fig 2. MK 5 router board.

And the device that will we use to flash with MK 5 firmware is a router branded GL-inet. Here specifications of GL-inet router:

CPU	400 MHz MIPS Atheros AR9331 version 1 SoC
Memory	16 MB ROM (winbound w25q128 FYSC (16384 Kbytes)), 64 MB DDR RAM (Etrontech EM6AB160TSD-5D) .
Wireless	Atheros AR9331 IEEE 802.11 b/g/n.
Ports	(2) 10/100 Ethernet, USB 2.0, TTL Serial, (6) exposed unheader GPIO pin.
Power	Miniusb powered 5v, ~1A.

Status Indicators	Power LED, Ethernet LED.
Default Os	Customized Openwrt
MTD partition info	
0x000000000000-0x000000020000 : "**u-boot**" 0x000000020000-0x00000010272c : "**kernel**" 0x00000010272c-0x000000ff0000 : "**rootfs**" 0x000000cc0000-0x000000ff0000 : "**rootfs_data**" 0x000000ff0000-0x000001000000 : "**art**" 0x000000020000-0x000000ff0000 : "**firmware**"	
Uboot costumized by pepe2k, this the most important change of it's modification is an inclusion of a web server, based on **[uIP 0.9 TCP/IP stack](http://www.gaisler.com/doc/net/uip-0.9/doc/html/main.html)**. It allows to upgrade **firmware**, **U-Boot** and **ART** (Atheros Radio Test) images, directly from your web browser, without need to access serial console and running a TFTP server. You can find similar firmware recovery mode, also based on uIP 0.9 TCP/IP stack, in **D-Link** routers.	

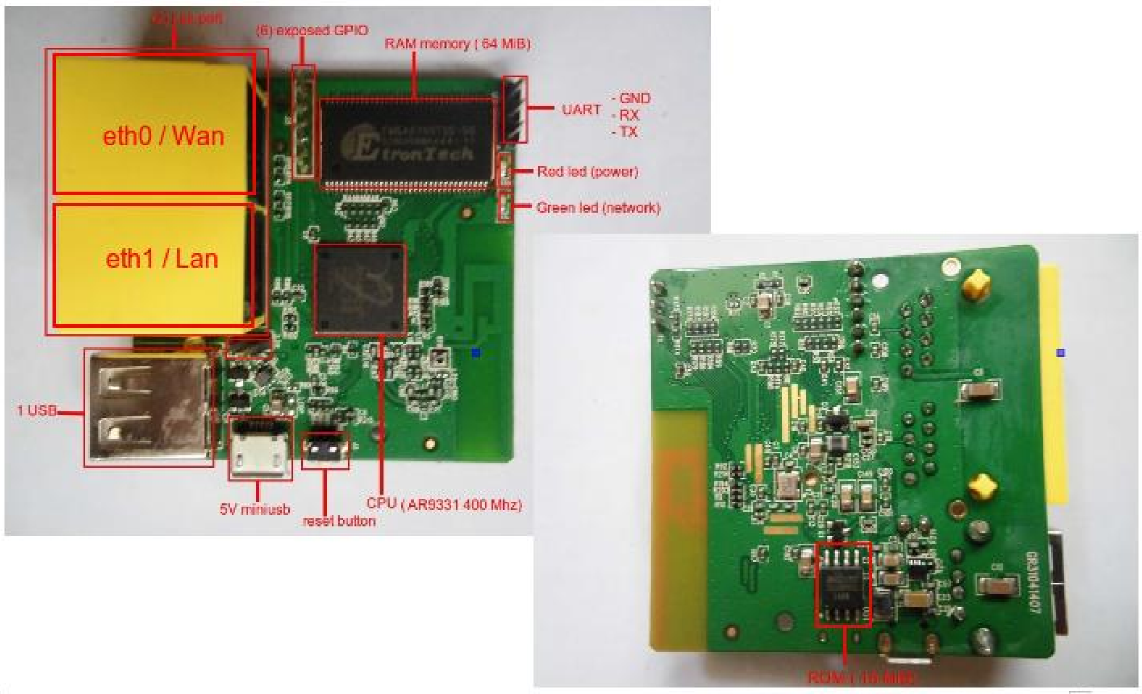


Fig 3. GL-Inet router board.

It may looks many differences between MK 5 routers and GL-inet's, but when we take a look of cpu chipset, ROM and RAM size You will see equivalence between both of them. Experiences from my previous research, i make hypothesized that MK 5 firmware is most likely can be flashing and running on GL-inet router. But previously we must hack that firmware.

1.3. Lab Preparation.

There are several tools and software that needs to be provided before we start. The tools is:

- Binwalk v 2.1.0 (linux software) [6]: Binwalk is an application that designed to perform firmware analysis, extraction and reverse engineering. To install binwalk we need of some dependencies like python 2.7/3 and python lzma. Binwalk can be installed by inputting this command in shell terminal:

```
$ sudo apt-get install python-lzma
```

6 Craig Heffner, "Quick Start Guide.", Internet: https://github.com/devttys0/binwalk/wiki/Quick-Start-Guide, Dec. 1, 2014[Jun. 6, 2015].

```
$ wget https://github.com/devttys0/binwalk/archive/master.zip
$ unzip master.zip
$ (cd binwalk-master && sudo python setup.py install)
```

- Firmware-mod-kit (linux software)[7]: The main function of this tool is to perform extraction and at the same me re-repack firmware image file. Firmware-mod-kit can be installed by inputting this command in shell terminal:

```
$ sudo apt-get install subversion build-essential zlib1g-dev
$ mkdir firmware_mod_kit
$ cd firmware_mod_kit
$ svn checkout http://firmware-mod-kit.googlecode.com/svn/trunk/ firmware-
mod-kit-read-only
```

- Hex editor (software), there are many types of hex editors can be use to perform hex modification. In this paper I using Ghex.

- USB2TTL (hardware) : USB2TTL dongle is used for monitoring and interrupting command through UART connection device. This dongle is very important in debugging process. The dongle running under minicom / putty. Following is display of the dongle.

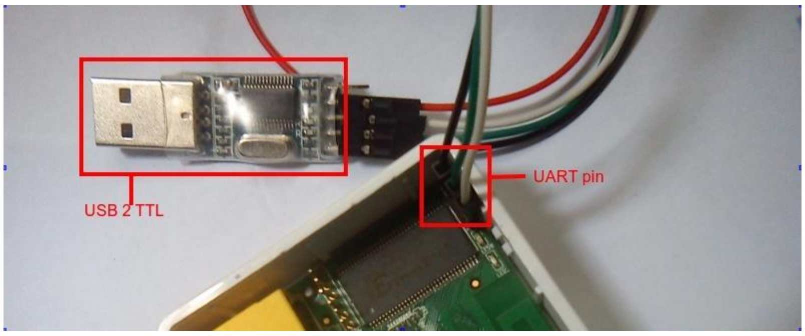


Fig 4. USB2TTL dongle and it installation in GL-inet serial port.

2. Firmware Hacking

2.1. Recognize Firmware Image

Firmware that will we analyzed is firmware version upgrade-2.2.0.bin and upgrade-2.4.0.bin. I choose ver 2.2.0.bin than ver 2.3.0 because based on info at pineapple bugtracker it shows that version upgrade-2.3.0.bin has a serious problem in it karma tool.[8] The upgraded version-2.4.0 file chosen because it was the latest version, although still has a few bugs in control dip-switch.

In this first stage we will identifying the firmware with the help of Binwalk.

Here are the results:

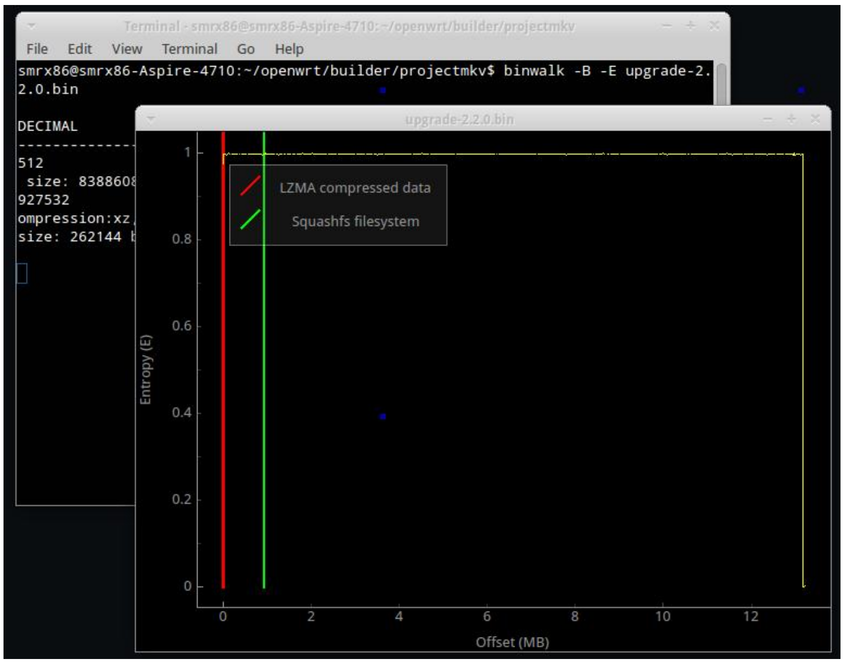
```
$ binwalk upgrade-2.2.0.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
512           0x200           LZMA compressed data, properties: 0x6D, dictionary
size: 8388608 bytes, uncompressed size: 2805816 bytes
927532        0xE272C         Squashfs filesystem, little endian, version 4.0,
compression:xz, size: 12259772 bytes, 2932 inodes, blocksize: 262144 bytes,
blocksize: 262144 bytes, created: 2015-01-12 22:39:23

$ binwalk upgrade-2.4.0.bin
```

7 "DD-WRT Mod Kit.", Internet: http://3iii.dk/wiki/index.php?title=DD-WRT_Mod_Kit&oldid=37, [Dec. 27,2014].

8 Take a look https://www.wifipineapple.com/?bugs_pub&action=view&id=302 that inform there are some bugs in karma blacklist tool in version upgrade-2.3.0.bin.

```
DECIMAL        HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
512             0x200           LZMA compressed data, properties: 0x6D, dictionary
size: 8388608 bytes, uncompressed size: 2805816 bytes
927576          0xE2758           Squashfs filesystem, little endian, version 4.0,
compression:xz,  size:  12316692  bytes,  2963  inodes,  blocksize:  262144  bytes,
blocksize: 262144 bytes, created: 2015-08-04 02:40:49
```

If you install a graph plugin in binwalk you can also analyze this firmware in entrophy graphic signature form. Here's how it looks:



```
smrx86@smrx86-Aspire-4710:~/openwrt/builder/projectmkv$ binwalk -B -E upgrade-2.
2.0.bin

DECIMAL
------------
512
 size: 8388608
927532
ompression:xz,
size: 262144 b
```

Fig 5. Analysis and signature entrophy binwalk result in graphical form.

From the analysis that we get from binwalk, we got conclusion this firmware consists of two main parts:

- Headers Section that compressed with LZMA, this section take position at the beginning of the firmware.
- The rootfs part that have type squashfs and compressed with xz.

2.2. Extracting Firmware Image

Binwalk application actually can do extraction, but it would difficult if we will do repackage. So at this stage we will use firmware mod kit. To make easier extraction process we also have to put the firmware file in a same folder where's firmware mod kit script is located.

Here's how it looks:

```
$ ./extract-firmware.sh upgrade-2.2.0.bin fmk/
Firmware Mod Kit (build-ng) 0.82, (c)2011-2013 Craig Heffner, Jeremy Collake

Scanning firmware...

Scan Time:     2015-06-21 11:24:28
Signatures:    193
Target  File:    /home/smrx86/firmware_mod_kit/firmware-mod-kit-read-only/upgrade-
2.2.0.bin
MD5 Checksum:  457a32d5b78cbdb5cf47fcb0fd3b719a

DECIMAL       HEX           DESCRIPTION
----------------------------------------------------------------------------------
512           0x200         LZMA compressed data, properties: 0x6D, dictionary size:
8388608 bytes, uncompressed size: 2805816 bytes
```

```
927532       0xE272C      Squashfs  filesystem,  little  endian,  version  4.0,
compression:   size:  12259772  bytes,   2932  inodes,  blocksize:  262144  bytes,
created: Tue Jan 13 05:39:23 2015

Extracting 927532 bytes of  header image at offset 0
Extracting squashfs file system at offset 927532
Extracting 96 byte footer from offset 13238180
Extracting squashfs files...
[sudo] password for smrx86:
Firmware extraction successful!
Firmware  parts  can  be  found  in  '/home/smrx86/firmware_mod_kit/firmware-mod-kit-
read-only/fmk/*'
```

```
$ ./extract-firmware.sh upgrade-2.4.0.bin fmk2/
Firmware Mod Kit (build-ng) 0.82, (c)2011-2013 Craig Heffner, Jeremy Collake

Scanning firmware...

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
512           0x200           LZMA compressed data, properties: 0x6D, dictionary
size: 8388608 bytes, uncompressed size: 2805816 bytes
927576         0xE2758          Squashfs filesystem, little endian, version 4.0,
compression:xz,  size:  12316692  bytes,  2963  inodes,  blocksize:  262144  bytes,
blocksize: 262144 bytes, created: 2015-08-04 02:40:49

Extracting 927576 bytes of  header image at offset 0
Extracting squashfs file system at offset 927576
Extracting 96 byte footer from offset 13303716
Extracting squashfs files...
Firmware extraction successful!
Firmware  parts  can  be  found  in  '/home/smrx86/firmware_mod_kit/firmware-mod-kit-
read-only/fmk2/*'
```

The extraction files can be found in folder fmk/ and fmk2/ as follows:

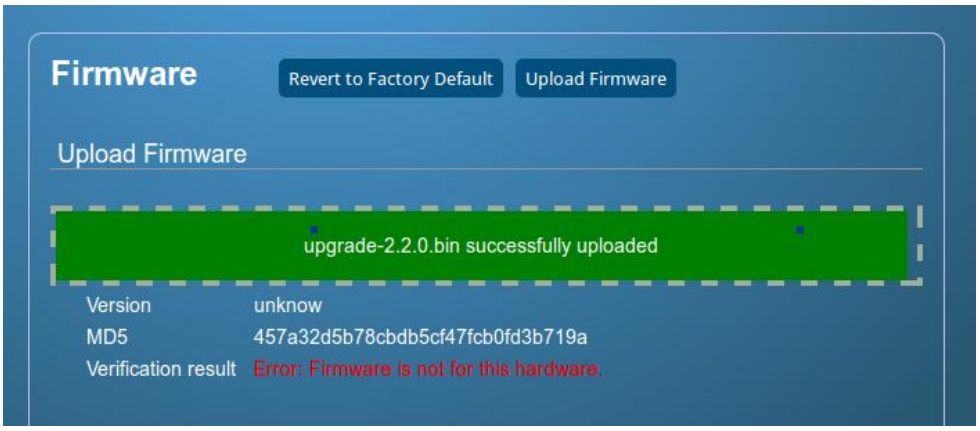
```
$ tree -a fmk/
fmk
├── image_parts
│   ├── footer.img
│   ├── header.img
│   └── rootfs.img
├── list
├── logs
│   ├── binwalk.log
│   └── config.log
├── new-filesystem.squashfs
├── new-firmware.bin
├── openwrt.bin
└── rootfs
    ├── bin
    │   ├── ash -> busybox
    │   ├── bash
…........


$ tree -a fmk2/
fmk
├── image_parts
│   ├── footer.img
│   ├── header.img
│   └── rootfs.img
├── list
├── logs
│   ├── binwalk.log
│   └── config.log
├── new-filesystem.squashfs
├── new-firmware.bin
├── openwrt.bin
└── rootfs
    ├── bin
    │   ├── ash -> busybox
    │   ├── bash
…........
```

If you look closer at information that we got from firmware mod kit, than you will figure that this result is slightly difference with the result that we got from Binwalk analysis. It appears that fmk didn't recognized rootfs compression type. This cause binwalk tool that bundled in Firmware mod kit is an old version. Therefore we will only refer to information that obtained in first stage (binwalk v 2.1.0 results).

2.3. Sorting and Find Unique File.

After extracted firmware, we will do identification check and some modification for pre-test debugging. The files that we need to check is fstab (./fmk/rootfs/etc/config/fstab) and format_sd file (./fmk/rootfs/pineapple/components/system/resources/includes/files/format_sd) cause they are closely related with the use of micro SD slot on MK 5 router. Fstab file doesn't need to be directly edited now because we will done it together with other files in exploitation phase.

But to debug there are a file that need to be modified first. File that we will examine is header.img (./fmk/image_parts/header.img). “Header.img” is part 1 of firmware that is compressed with LZMA. This part store machine/hardware ID code that will be checked every time we do flashing via web UI or SSH shell. Without modification, flashing process will be terminated / halt immediately.

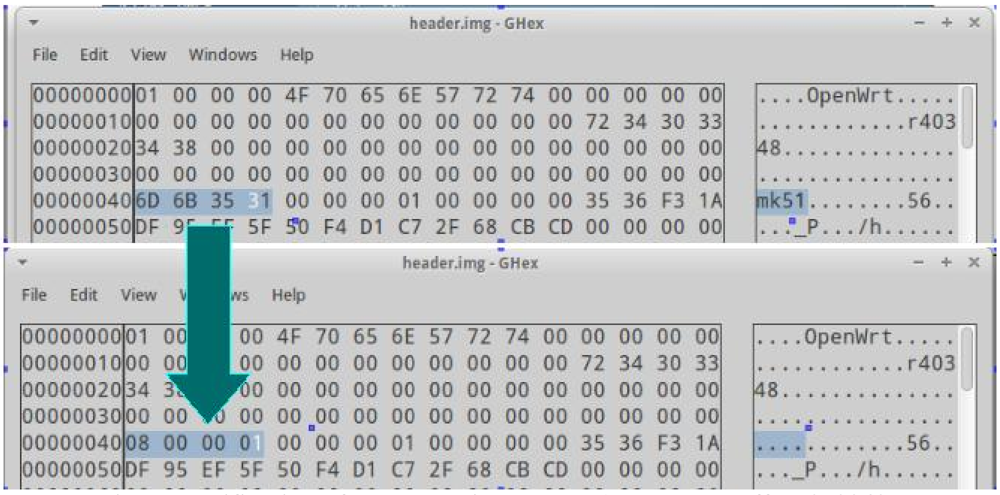


upgrade-2.2.0.bin successfully uploaded

Version	unknow
MD5	457a32d5b78cbdb5cf47fcb0fd3b719a
Verification result	Error: Firmware is not for this hardware.

Fig 6. Failure to verify in flashing due to unmodified id header.

To avoid that we need to edit 0x0040 offset value in header.img with a hex editor.[9] New value is refers to table HW identification in lzma header.[10]



```
00000000 01 00 00 00 4F 70 65 6E 57 72 74 00 00 00 00 00   ....OpenWrt.....
00000010 00 00 00 00 00 00 00 00 00 00 00 00 72 34 30 33   ............r403
00000020 34 38 00 00 00 00 00 00 00 00 00 00 00 00 00 00   48..............
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000040 6D 6B 35 31 00 00 00 01 00 00 00 00 35 36 F3 1A   mk51........56..
00000050 DF 9[illegible] 5F 50 F4 D1 C7 2F 68 CB CD 00 00 00 00   ..._P.../h......
```

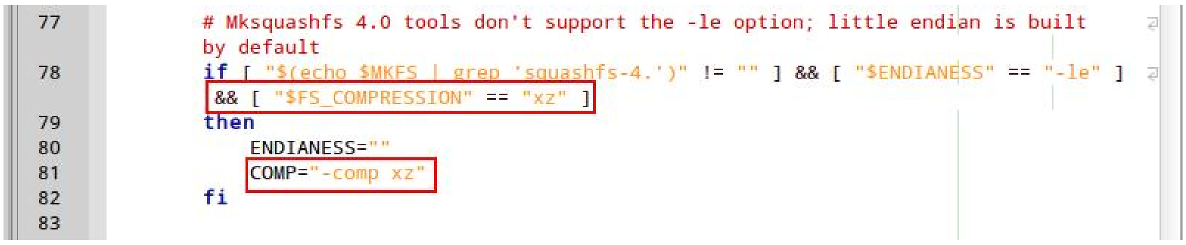
```
00000000 01 00 [illegible] 00 4F 70 65 6E 57 72 74 00 00 00 00 00   ....OpenWrt.....
00000010 00 00 [illegible] 00 00 00 00 00 00 00 00 00 72 34 30 33   ............r403
00000020 34 3[illegible] 00 00 00 00 00 00 00 00 00 00 00 00 00   48..............
00000030 00 00 [illegible]0 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000040 08 00 00 01 00 00 00 01 00 00 00 00 35 36 F3 1A   ............56..
00000050 DF 95 EF 5F 50 F4 D1 C7 2F 68 CB CD 00 00 00 00   ..._P.../h......
```

Fig 7. Modification of hardware id on LZMA header at offset 0x0040.

9 Tomcom "Flashing TP-Link TL-MR3040 v2.0 (Version / Revison 2) with V1.0 Image.", Internet: https://forum.openwrt.org/viewtopic.php?id=44090, May, 9, 2013[Jun. 22, 2015].

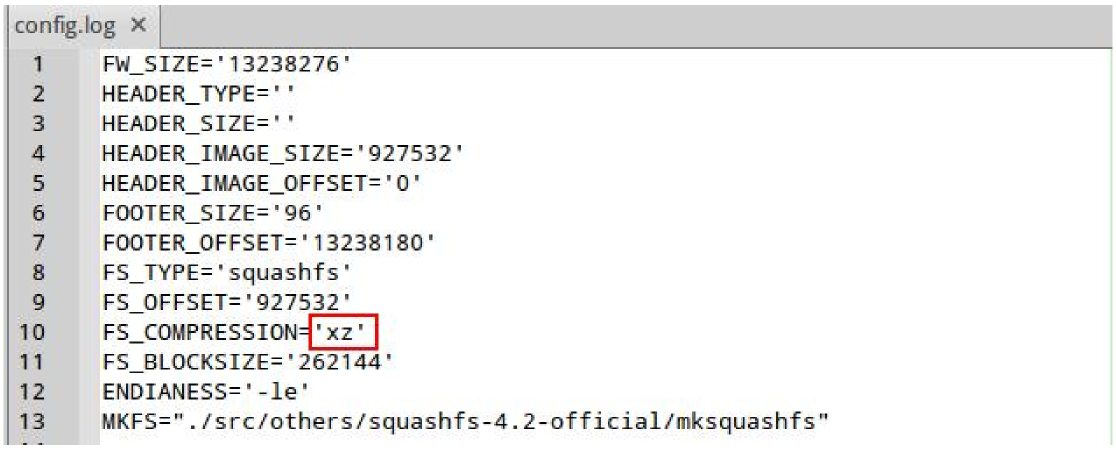
10 Look table in Appendix.

After editing header.img and save them, the process continued with repacking by execute ./build-firmware.sh. But before it we must edit build-firmware.sh file(./ build-firmware.sh) and config.log (./fmk/logs/config.log), in order to call for support repacking with xz compression.



```
        # Mksquashfs 4.0 tools don't support the -le option; little endian is built by default
        if [ "$(echo $MKFS | grep 'squashfs-4.')" != "" ] && [ "$ENDIANESS" == "-le" ] && [ "$FS_COMPRESSION" == "xz" ]
        then
            ENDIANESS=""
            COMP="-comp xz"
        fi
```

Fig 8. Modification build-firmware.sh to support xz compression types.



```
config.log
FW_SIZE='13238276'
HEADER_TYPE=''
HEADER_SIZE=''
HEADER_IMAGE_SIZE='927532'
HEADER_IMAGE_OFFSET='0'
FOOTER_SIZE='96'
FOOTER_OFFSET='13238180'
FS_TYPE='squashfs'
FS_OFFSET='927532'
FS_COMPRESSION='xz'
FS_BLOCKSIZE='262144'
ENDIANESS='-le'
MKFS="./src/others/squashfs-4.2-official/mksquashfs"
```

Fig 9. Modifications config.log on fs_compression part from "gzip" to "xz".

After editing ./build-firmware.sh, and config.log we also need to remove some files, if not we will having problems later in size of firmware that we build. In this case I delete files notification.gif (./fmk/rootfs/pineapple/includes/img/notification.gif) because his role is just not too significant. Then we continue proceed with firmware building.

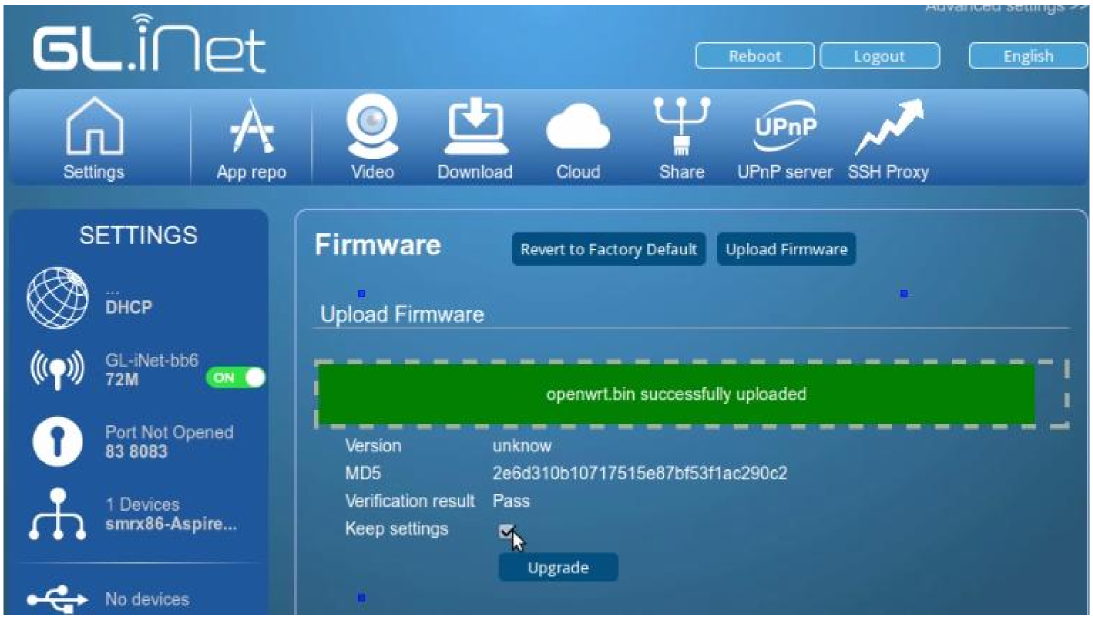
```
$ sudo su
[sudo] password for smrx86:
# rm -rf ./fmk/rootfs/pineapple/includes/img/notification.gif
# ./build-firmware.sh fmk/
Firmware Mod Kit (build-ng) 0.82, (c)2011-2013 Craig Heffner, Jeremy Collake

Building new squashfs file system...
Parallel mksquashfs: Using 2 processors
Creating 4.0 filesystem on /home/smrx86/firmware_mod_kit/firmware-mod-kit-read-only/fmk/new-filesystem.squashfs, block size 262144.
[=============================================================/]
 2538/2538 100%
Exportable Squashfs 4.0 filesystem, xz compressed, data block size 262144
        compressed data, compressed metadata, compressed fragments, compressed xattrs
        duplicates are removed
Filesystem size 12008.79 Kbytes (11.73 Mbytes)
        26.02% of uncompressed filesystem size (46147.47 Kbytes)
Inode table size 24208 bytes (23.64 Kbytes)
        25.29% of uncompressed inode table size (95708 bytes)
Directory table size 27464 bytes (26.82 Kbytes)
        46.95% of uncompressed directory table size (58494 bytes)
Number of duplicate files found 77
…...
Remaining free bytes in firmware image: 10360
Processing 0 header(s) from /home/smrx86/firmware_mod_kit/firmware-mod-kit-read-only/fmk/new-firmware.bin...
CRC update failed.
```

```
Firmware header not supported; firmware checksums may be incorrect. New firmware
image has been saved to: /home/smrx86/firmware_mod_kit/firmware-mod-kit-read-
only/fmk/new-firmware.bin
# mv ./fmk/new-firmware.bin ./fmk/openwrt.bin
```

From the results it shown FMK give warning that "CRC firmware update. Firmware header not support ... ". But these notifications can be ignored. Firmware file are stored in ./fmk/openwrt.bin

2.4. Debugging Test.

Debugging tests can be performed using emulator or vbox with openwrt x86 image, but it ability is limited to run configuration or just front end of its WebUI. In this case I chose to use a actual router, that is GL-inet router. Then process continued by flash the router with firmware that we build before, in a while we also need watching the log through minicom.



openwrt.bin successfully uploaded

Version	unknow
MD5	2e6d310b10717515e87bf53f1ac290c2
Verification result	Pass
Keep settings	✓

Fig 10. After modification header id firmware verification can be bypassed.

Based on mincom log and monitoring via web browser, we can see firmware flashing is successfull without constraints verification check. To start flashing make sure uncheck keep settings before click upgrade button.

A while after upgrade is done, router will reboot and we need to **exchange lan cable position to eth0 port/WAN** port in GL-inet's**.** This must be done because the original MK 5 firmware doesn't build to support 2 ethernet port.

This following log minicom after router reboot:

```
$ sudo minicom
[sudo] password for smrx86:


Welcome to minicom 2.5

OPTIONS: I18n
Compiled on May  2 2011, 00:39:27.
Port /dev/ttyUSB2

Press CTRL-A Z for help on special keys



*****************************************
*      U-Boot 1.1.4  (Jun 25 2014)      *
*****************************************

AP121 (AR9331) U-Boot for GL-iNet
```

```
DRAM:  64 MB
FLASH: Winbond W25Q128 (16 MB)

LED on during eth initialization...

Hit any key to stop autobooting:  0

Booting image at: 0x9F020000

   Image name:   OpenWrt r40348
   Image type:   MIPS Linux Kernel Image (lzma compressed)
   Data size:    927020 Bytes = 905.3 kB
   Load address: 0x80060000
   Entry point:  0x80060000

Uncompressing kernel image... OK!
Starting kernel...

[    0.000000] Linux version 3.3.8 (sebkinne@buildtop) (gcc version 4.6.3 201205
[    0.000000] bootconsole [early0] enabled
[    0.000000] CPU revision is: 00019374 (MIPS 24Kc)
[    0.000000] SoC: Atheros AR9330 rev 1
[    0.000000] Clocks: CPU:400.000MHz, DDR:400.000MHz, AHB:200.000MHz, Ref:25.0z
…...................................
[    0.550000] mtd: partition "rootfs_data" created automatically, ofs=CC0000,
[    0.550000] 0x000000cc0000-0x000000ff0000 : "rootfs_data"
[    0.560000] 0x000000ff0000-0x000001000000 : "art"
[    0.570000] 0x000000020000-0x000000ff0000 : "firmware"
….........
- preinit -
Press the [f] key and hit [enter] to enter failsafe mode
regular preinit -
```

After ensure that flashing done successfully and get you ip address from router, then we can browse to admin page of pineapple in http://172.16.42.1:1471/.

2.4.a. Debugging test version 2.2.0.

From admin page we can find that the MK 5 ver2.2.0 frontend has Unique protection script that trigger 4 led as initial key access..

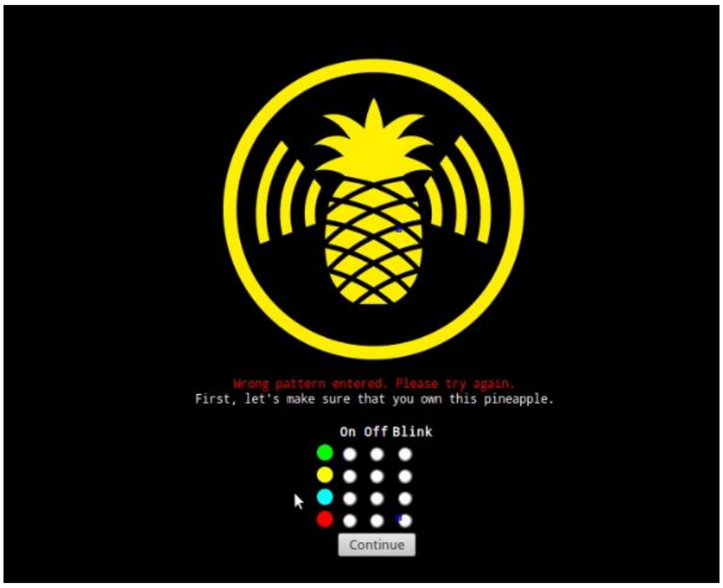

Wrong pattern entered. Please try again.
First, let's make sure that you own this pineapple.

Fig 11. Admin page protection in version 2.2.0 that triggering leds flame in MK 5.

Some trials showed this protection system doesn't have limit attempt and failure response shows with same message repeatly. Using string "wrong pattern entered" we can track the script which is responsible for this puzzle. So from folder rootfs/ pineapple FMK extraction results I enter command:

```
$ grep -lr -e 'wrong\|pattern\|entered' *
```

```
components/system/configuration/help.json
components/system/network/functions.php
includes/welcome/welcome.inc.php
includes/welcome/welcome.php
```

We find that file who responsible for this protection system is welcome.inc.php (./fmk/rootfs/pineapple/includes/welcome/welcome.inc.php).

2.4.b. Debugging test version 2.4.0.

In debugging test of version 2.4.0 we knows that admin protection is a combination of dip-switch control.

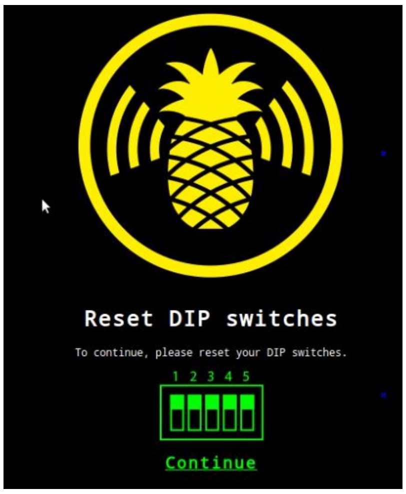


Fig 12. Admin page protection in pineapple ver 2.4.0 firmware.

In the test device there is no dip-switch component, so in this test we will only rely on string message that contained in admin page.

```
$ grep -lr -e "Reset DIP switches" *
includes/welcome/welcome.inc.php
```

From the string search results we can see file which in charge to running this protection is also welcome.inc.php.

From here we move on to exploitation stages.

2.5. Exploitation and Final Repacking.

Besides welcome.inc.php file there are also some other files that already we talked before that need to be edited in this exploitation stage. Here are the files and it modifications:

- fstab (./fmk/rootfs/etc/config/fstab) : This file have a job to manage and mounting additonal storage such as UFD or Micro SD as MK 5 have. In this case Gl-inet use UFD as additional storage media, then fstab needs to be edited into:

```
$  cat ./fmk/rootfs/etc/config/fstab
config global automount
       option from_fstab 1
       option anon_mount 1

config global autoswap
       option from_fstab 1
       option anon_swap 0

config mount
       option target   /sd
       option device   /dev/sda1
       option fstype   auto
       option options  rw,sync
       option enabled  1
       option enabled_fsck 0
```

```
config swap
       option device    /dev/sda2
       option enabled   1
```

- format_sd (./fmk/rootfs/pineapple/components/system/resources/includes/files/format_sd): This shell script is gived task to perform formatting SD card into two partitions, partition 1 with ext4 format and partition 2 as a swap. For that we need first edit up as follow:

```
$cat./fmk/rootfs/pineapple/components/system/resources/includes/files/forma
t_sd
#!/bin/bash
#2013 - WiFiPineapple.com

[[ -f /tmp/sd_format.progress ]] && {
  exit 0
}

#Function to find and reset the SD device
function reset_sd {
   DEVICE=$(find / -name idProduct -exec grep -l 6366 {} + | awk -F '/'
'{ print $(NF-1) }')
  echo $DEVICE  > /sys/bus/usb/drivers/usb/unbind
  sleep 2
  echo $DEVICE  > /sys/bus/usb/drivers/usb/bind
}


touch /tmp/sd_format.progress

reset_sd
sleep 5

umount /sd
swapoff /dev/sda2

sleep 2
cat
/pineapple/components/system/resources/includes/files/fdisk_instructions   |
fdisk /dev/sda
sleep 2

umount /sd
mkfs.ext4 /dev/sda1
sleep 2
mkfs.ext4 /dev/sda2


mkswap /dev/sda2

mount /dev/sda1 /sd
swapon /dev/sda2

rm /tmp/sd_format.progress
```

- welcome.inc.php (./fmk/rootfs/pineapple/includes/welcome/welcome.inc.php), this file have a job to run protection into pineapple admin page.
  They are two version that we talked in this paper, just like down below:
  a. Version 2.2.0
  Reviewing of the script, it looks just three LED that serves as a key accsess (amber, blue, red). While it's value is always changing every time attempt is done and it was obtained from a random array in generateLEDpattern() function. For that we will give it exact value in function verifyPineapple ($ post) that will give us pattern combination amber = blink , blue = off and red=on. This following is the modification results:

```
$ cat ./fmk/rootfs/pineapple/includes/welcome/welcome.inc.php
<?php
```

```
namespace pineapple;
........................

function verifyPineapple($post)
{
    $action_array = array('off', 'on', 'blink');
    if (isset($_SESSION['verify_pattern'])
        && isset($post['amber'])
        && isset($post['blue'])
        && isset($post['red'])
    ) {
        $current_state = str_split($_SESSION['verify_pattern']);
        if (array_search($post['amber'], $action_array) == 2
            && array_search($post['blue'], $action_array) == 0
            && array_search($post['red'], $action_array) == 1
        ) {
            $_SESSION['verified'] = true;
            return passwordForm();
        }
    }
    generateLEDpattern();
    return verifyForm(true);
}

…......................
```

b. Version 2.4.0

In version 2.4.0, we just need to change the equation "==" to "! =" below the resetDips () function.

```
$ cat ./fmk2/rootfs/pineapple/includes/welcome/welcome.inc.php
<?php
namespace pineapple;
........................
function resetDips()
{
   if (checkDIPStatus() != "reset") {
        header("Location: /?action=set_password");
    }
    $text = "<h1>Reset DIP switches</h1>";
    $text .= "<p>To continue, please reset your DIP switches.</p>";
    $text .= "<img src='/includes/welcome/img/11111.gif'>";
    $text .= "<h2><a href='/?action=reset_dips'>Continue</a></h2>";

    return $text;
}
........................
```

Right After edited that three files, what we do then is the final repacking by executing build-firmware.sh script.

```
$ sudo ./build-firmware.sh fmk/
Firmware Mod Kit (build-ng) 0.82, (c)2011-2013 Craig Heffner, Jeremy
Collake

Building new squashfs file system...
Parallel mksquashfs: Using 2 processors
Creating 4.0 filesystem on /home/smrx86/firmware_mod_kit/firmware-mod-kit-
read-only/fmk/new-filesystem.squashfs, block size 262144.
[==============================================================/]  2538/2538
100%
Exportable Squashfs 4.0 filesystem, xz compressed, data block size 262144
       compressed data, compressed metadata, compressed fragments,
compressed xattrs
       duplicates are removed
…...
```

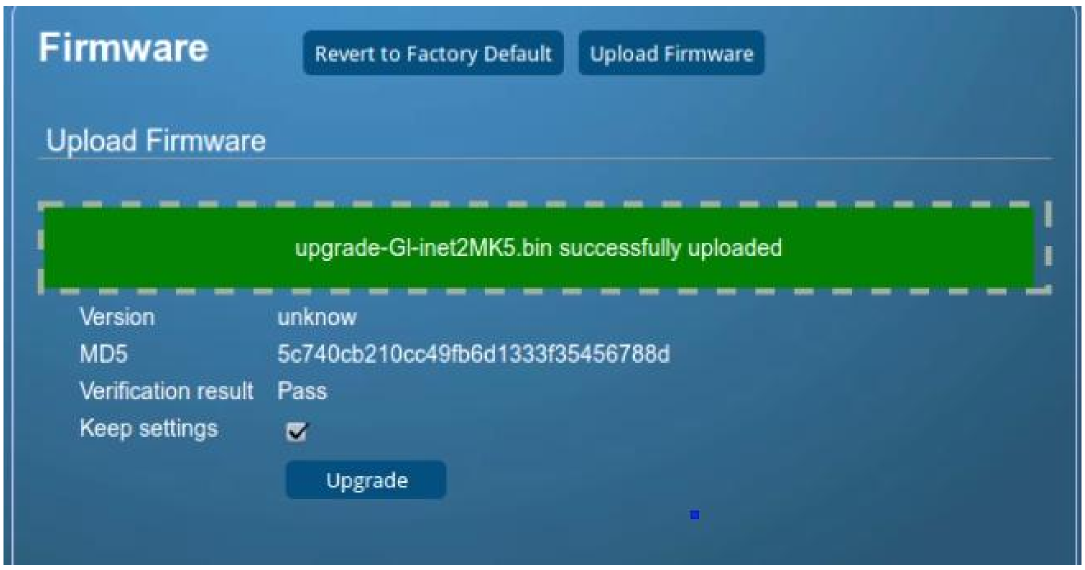
```
Number of uids 1
        root (0)
Number of gids 1
        root (0)
Remaining free bytes in firmware image: 10360
Processing 0 header(s) from /home/smrx86/firmware_mod_kit/firmware-mod-kit-
read-only/fmk/new-firmware.bin...
CRC update failed.
Firmware header not supported; firmware checksums may be incorrect. New
firmware image has been saved to: /home/smrx86/firmware_mod_kit/firmware-
mod-kit-read-only/fmk/new-firmware.bin

$ mv ./fmk/new-firmware.bin /fmk/upgrade-Gl-inet2MK5.bin
$ md5sum upgrade-Gl-inet2MK5.bin
5c740cb210cc49fb6d1333f35456788d  upgrade-Gl-inet2MK5.bin
```

And at last, the firmware file can be found in ./fmk/upgrade-Gl-inet2MK5.bin.

3. Proof of Concepts

3.1. Flashing and Baypass UI Protection

Flashing device can be done too through Uboot console, but in this PoC flashing performed in default firmware upgrade page provided by stock GL-inet firmware. Besides that, it is also carried out to see whether or not it can pass verification check through sysupgrade cgi. From this experiments it showed that this firmware qualify to bypass verification checks.



upgrade-Gl-inet2MK5.bin successfully uploaded

Version	unknow
MD5	5c740cb210cc49fb6d1333f35456788d
Verification result	Pass
Keep settings	✓

Fig 13. Firmware upgrade-Gl-inet2MK5.bin file is qualify to bypass verify check when going to upgrade.

Besides that admin protection in ver 2.2.0 is also successful bypass with slash pattern, such as following spoiler:

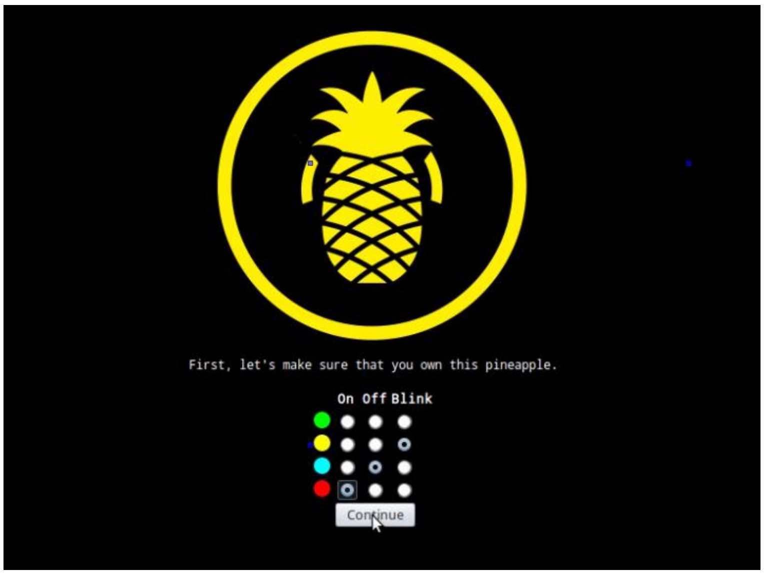


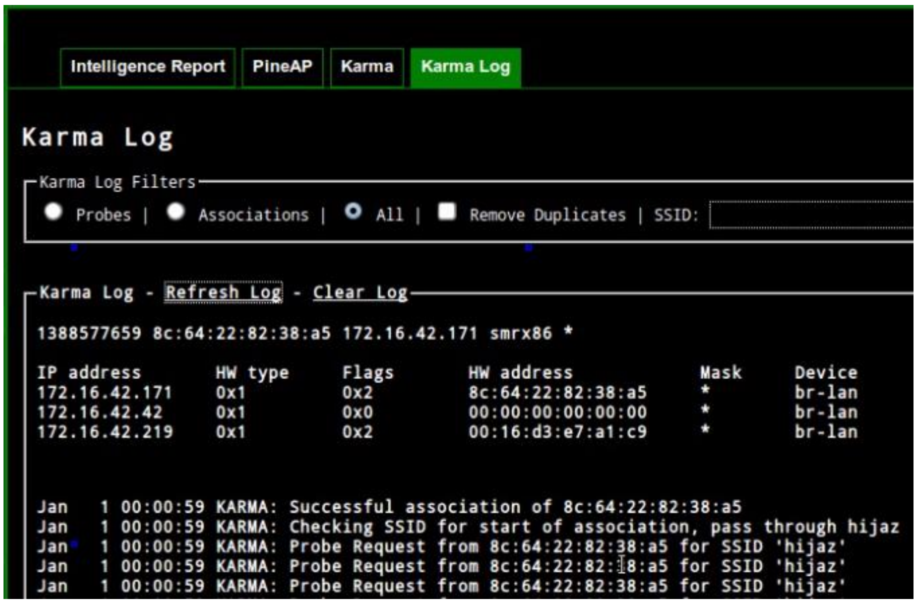
Fig 14. Bypass admin protection page with SLASH pattern.

And on version 2.4.0 mod script will directly bypass admin protection page without any user input. :P

3.2. Karma Test

Karma is a tool that designed and used for wireless hacking. This tool works by automatically make fake SSID according to SSID request that were detected around it.

This is the overview:



Karma Log

Karma Log Filters
Probes | Associations | All | Remove Duplicates | SSID:

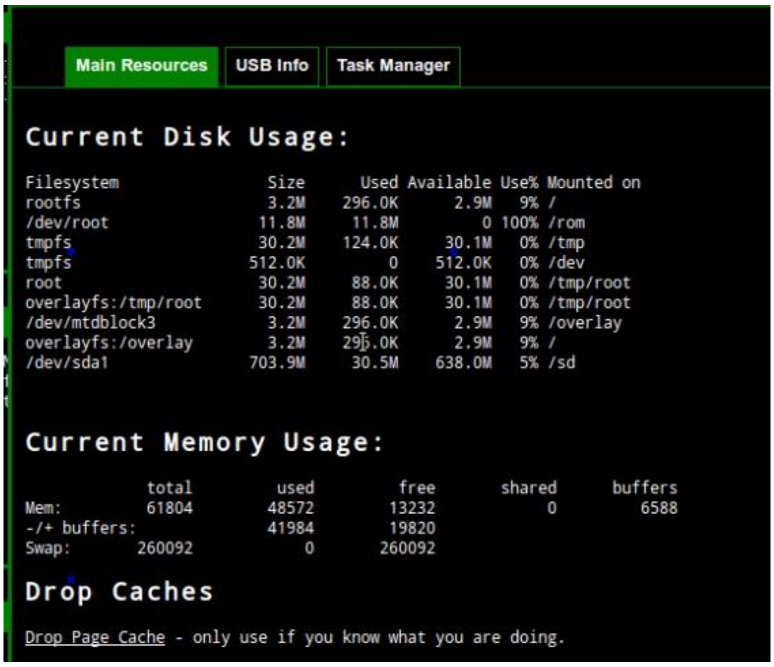
Karma Log - Refresh Log - Clear Log

```
1388577659 8c:64:22:82:38:a5 172.16.42.171 smrx86 *

IP address       HW type     Flags       HW address            Mask     Device
172.16.42.171    0x1         0x2         8c:64:22:82:38:a5     *        br-lan
172.16.42.42     0x1         0x0         00:00:00:00:00:00     *        br-lan
172.16.42.219    0x1         0x2         00:16:d3:e7:a1:c9     *        br-lan


Jan   1 00:00:59 KARMA: Successful association of 8c:64:22:82:38:a5
Jan   1 00:00:59 KARMA: Checking SSID for start of association, pass through hijaz
Jan   1 00:00:59 KARMA: Probe Request from 8c:64:22:82:38:a5 for SSID 'hijaz'
Jan   1 00:00:59 KARMA: Probe Request from 8c:64:22:82:38:a5 for SSID 'hijaz'
Jan   1 00:00:59 KARMA: Probe Request from 8c:64:22:82:38:a5 for SSID 'hijaz'
```

Fig 15. The log when karma is active.

3.3. Fstab and UFD Mounting

From fstab config file modifications it looks fstab working and flashdisk is mounting properly and recognized as /sd:



```
Current Disk Usage:

Filesystem                Size      Used Available Use% Mounted on
rootfs                    3.2M    296.0K      2.9M   9% /
/dev/root                11.8M     11.8M         0 100% /rom
tmpfs                    30.2M    124.0K     30.1M   0% /tmp
tmpfs                   512.0K         0    512.0K   0% /dev
root                     30.2M     88.0K     30.1M   0% /tmp/root
overlayfs:/tmp/root      30.2M     88.0K     30.1M   0% /tmp/root
/dev/mtdblock3            3.2M    296.0K      2.9M   9% /overlay
overlayfs:/overlay        3.2M    296.0K      2.9M   9% /
/dev/sda1               703.9M     30.5M    638.0M   5% /sd


Current Memory Usage:

             total         used         free       shared      buffers
Mem:         61804        48572        13232            0         6588
-/+ buffers:              41984        19820
Swap:       260092            0       260092

Drop Caches

Drop Page Cache - only use if you know what you are doing.
```

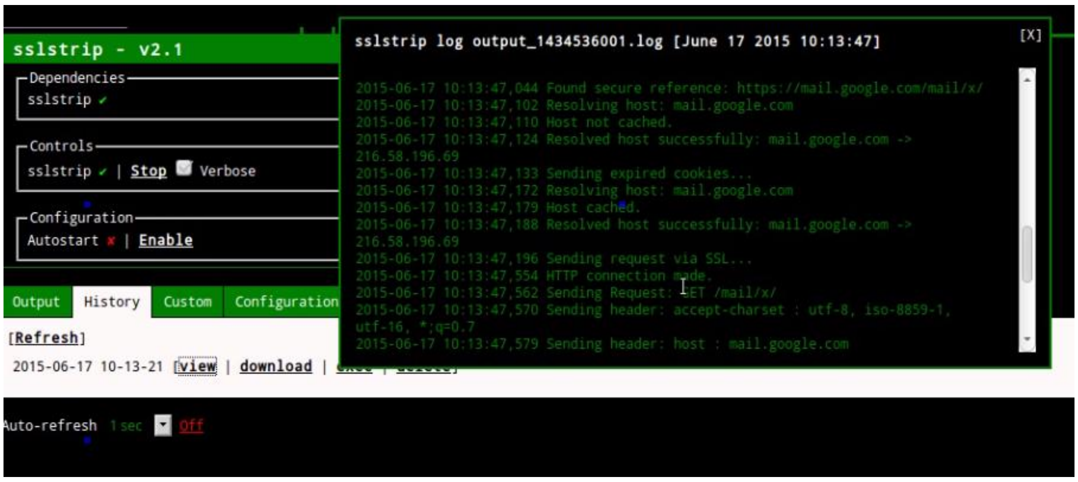
Fig 16. Flashdisk is mounting and successful recognizable as /sd.

3.4. Infusions and it's PoC

Here's a preview of some infusion PoC in router GL-inet

3.4.a. Sslstrip

Sslstrip works by redirecting client requesting for page in https protocol to http protocol.



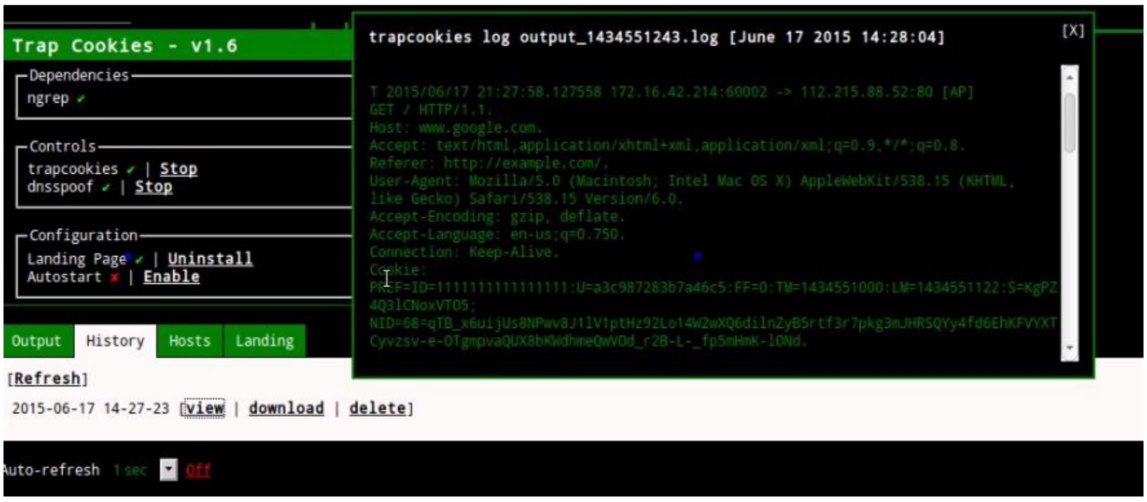
```
sslstrip - v2.1
Dependencies
sslstrip ✓
Controls
sslstrip ✓ | Stop ☐ Verbose
Configuration
Autostart ✗ | Enable
Output  History  Custom  Configuration
[Refresh]
2015-06-17 10-13-21 [view | download |
Auto-refresh 1 sec Off
```

```
sslstrip log output_1434536001.log [June 17 2015 10:13:47]                [X]

2015-06-17 10:13:47,044 Found secure reference: https://mail.google.com/mail/x/
2015-06-17 10:13:47,102 Resolving host: mail.google.com
2015-06-17 10:13:47,110 Host not cached.
2015-06-17 10:13:47,124 Resolved host successfully: mail.google.com ->
216.58.196.69
2015-06-17 10:13:47,133 Sending expired cookies...
2015-06-17 10:13:47,172 Resolving host: mail.google.com
2015-06-17 10:13:47,179 Host cached.
2015-06-17 10:13:47,188 Resolved host successfully: mail.google.com ->
216.58.196.69
2015-06-17 10:13:47,196 Sending request via SSL...
2015-06-17 10:13:47,554 HTTP connection made.
2015-06-17 10:13:47,562 Sending Request: GET /mail/x/
2015-06-17 10:13:47,570 Sending header: accept-charset : utf-8, iso-8859-1,
utf-16, *;q=0.7
2015-06-17 10:13:47,579 Sending header: host : mail.google.com
```

Fig 17. Sslstrip log indicates that https: //mail.google.com is redirected to http protocol.

3.4.b. Trapcookies

Trapcookies work with help of dnsspoof to redirect specific website address to phishing page, then recording cookies and store it into a log file.



```
trapcookies log output_1434551243.log [June 17 2015 14:28:04]

T 2015/06/17 21:27:58.127558 172.16.42.214:60002 -> 112.215.88.52:80 [AP]
GET / HTTP/1.1.
Host: www.google.com.
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8.
Referer: http://example.com/.
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X) AppleWebKit/538.15 (KHTML,
like Gecko) Safari/538.15 Version/6.0.
Accept-Encoding: gzip, deflate.
Accept-Language: en-us;q=0.750.
Connection: Keep-Alive.
Cookie:
PREF=ID=1111111111111111:U=a3c987283b7a46c5:FF=0:TM=1434551000:LM=1434551122:S=KgPZ
4Q3lCNoxVTO5;
NID=68=qTB_x6uijUs8NPwv8J1lV1ptHz92Lo14W2wXQ6dilnZyB5rtf3r7pkg3mJHRSQYy4fd6EhKFVYXT
Cyvzsv-e-OTgmpvaQUX8bKWdhmeQwVOd_r2B-L-_fp5mHmK-lONd.
```

Fig 18. Log of trapcookies that capture and store cookies after client trapted by dnsspoof.

4. Conclusions

Firmware pineapple MK 5 could be porting & flashing into Gl-inet router after perform firmware hacking. The most important thing is modification on hardware_id at LZMA header part. While minor modifications at config file and UI is required just to run cgi smoothly.

5. Greetings

Thanks to Cindy Wijaya, Xopal Unil, Tisaros Kaskus, Openwrt Indonesia, Om Hero lirva32, Brahmanggi Aditya, Richy Hendra, Ade Surya, …all human or not (^^) who always support inspired me. And ofcourse it's U… ra'

7. References

[1] Alan Holt and Chi-Yu Huang, "Embedded Operating Systems, A Practical Approach," in introduction, virtualisation. London, Springer-Verlag, 2014, pp. 7.

[2] Andy Davis. "*Blue for the pineapple*.", Internet: http://penturalabs.wordpress.com/2013/04/25/blue-for-the-pineapple/, Apr. 25, 2013[Apr. 20,2015].

[3] Andy Davis. "Naked WiFi Pineapple Mark V!.", Internet: https://penturalabs.wordpress.com/2013/10/27/naked-wifi-pineapple-mark-v/, Oct. 27, 2013[Apr. 20,2013].

[4] Craig Heffner, "Binwalk, Quick Start Guide.", Internet: https://github.com/devttys0/binwalk/wiki/Quick-Start-Guide, Dec. 1, 2014[Jun. 6, 2015].

[5] Craig Heffner, "Mucking About With SquashFS.", internet: http://www.devttys0.com/2014/08/mucking-about-with-squashfs/, Aug. 20, 2014[Jun. 9, 2015].

[6] Craig Heffner, "Extracting Non-Standard SquashFS Images Extracting Non-Standard SquashFS Images.", internet: http://www.devttys0.com/2011/08/extracting-non-standard-squashfs-images/, Aug. 7, 2011[Jun. 9, 2015].

[7] "DD-WRT Mod Kit.", Internet: http://3iii.dk/wiki/index.php?title=DD-WRT_Mod_Kit&oldid=37, [Dec. 27,2014].

[8] "Firmware mod kit.", Internet: http://www.bitsum.com/firmware_mod_kit.htm, [Apr. 20,2015].

[9] Git openwrt, "mktplikfw.c." internet : http://git.openwrt.org/?p=openwrt.git;a=blob;f=tools/firmware-utils/src/mktplinkfw.c [Jun. 26, 2015]

[10] Hero Suhartono, "Information Theft: Wireless Router Shareport for Phun and profit." Internet: http://www.slideshare.net/idsecconf/idsecconf2014-paper [Jun. 9, 2015]

[11] Hak5. "Pineapple MK V firmware.", avaiable: https://www.wifipineapple.com/index.php?downloads&download_mk5_upgrade=2.2.0, Jan. 12, 2015 [Apr. 20, 2015].

[12] Jonas Zaddach and Andrei Costin."Embedded Devices Security and Firmware Reverse Engineering." , Avaiable: https://media.blackhat.com/us-13/US-13-Zaddach-Workshop-on-

Embedded-Devices-Security-and-Firmware-Reverse-Engineering-WP.pdf, 2013[Jun. 14, 2015].

[13] Pinzaru. "What is firmware?.", internet:http://incepator.pinzaru.ro/software/what-is-firmware/, Apr. 2015[Jun. 6,2015].

[14] Silverbug and Redhidden. “Fu~n of Attacking Firmware.”, avaiable: http://www.powerofcommunity.net/poc2012/re&si.pdf, 2012[Jun. 10, 2015].

[15] Tomcom "Flashing TP-Link TL-MR3040 v2.0 (Version / Revison 2) with V1.0 Image.", Internet: https://forum.openwrt.org/viewtopic.php?id=44090, May, 9, 2013[Jun. 22, 2015].

[16] Wikipedia. "Firmware.", internet: http://en.m.wikipedia.org/wiki/Firmware, last edited, Jun. 5, 2015[Jun. 10, 2015].

[17] Wiki openwrt. “TRX vs. TRX2 vs. BIN.”, Internet: http://wiki.openwrt.org/doc/techref/header, [Jun. 26, 2015]

8. Bio

Smrx86, is a blogger enthusiast, addict to new technology and computer security information. Become speaker at IDSECCONF 2013, IDSECCONF 2014, seminar riset nasional ID-SIRTII/CC tahun 2015 and IDSECCONF 2015 that will hold at Cirebon. Right now work as freelance pentester and product tester for some router factory board from Shenzen. You can get in touch with me through an email: ramatrinanda@gmail.com or twitter https://twitter.com/smrx86.

Appendix

Table 1. HW identification in lzma header[11]

Vendor	Product name	Offset location	hexcode
Hak5	MK 5	0x0040	6D 6B 35 31
GL-INET	gl-inet-v1	0x0040	08 00 00 01
Konke	Kankun smart plug	0x0040	07 03 01 01
TP-LINK	tl-mr3220-v1	0x0040	32 20 00 01
TP-LINK	TLMR3220V2	0x0040	32 20 00 02
TP-LINK	tl-mr3420-v1	0x0040	34 20 00 01
TP-LINK	tl-wa701n-v1	0x0040	07 01 00 01
TP-LINK	TLWA7510NV1	0x0040	75 10 00 01
TP-LINK	TLWA901NV1	0x0040	09 01 00 01
TP-LINK	TLWA901NV2	0x0040	09 01 00 02
TP-LINK	TLWR740NV1	0x0040	07 40 00 01
TP-LINK	TLWR740NV3	0x0040	07 40 00 03
TP-LINK	TLWR741NV1	0x0040	07 41 00 01
TP-LINK	tl-wr743nd-v1	0x0040	07 43 00 01
TP-LINK	tl-wr841nd-v3	0x0040	08 41 00 03
TP-LINK	TLWR841NV5	0x0040	08 41 00 05
TP-LINK	TLWR841NV7	0x0040	08 41 00 07
TP-LINK	TLWR842	0x0040	08 42 00 01
TP-LINK	TLWR941NV2	0x0040	09 41 00 02
TP-LINK	TLWR941NV4	0x0040	09 41 00 04
TP-LINK	tl-wr1043nd-v1	0x0040	10 43 00 01
TP-LINK	tl-mr11u-v1	0x0040	00 11 01 01
TP-LINK	tl-mr3020-v1	0x0040	30 20 00 01
TP-LINK	tl-mr3040-v1	0x0040	30 40 00 01
TP-LINK	tl-mr3040-v2	0x0040	30 40 00 02
TP-LINK	tl-wr703n-v1	0x0040	07 03 01 01
TP-LINK	TLWR740NV4	0x0040	07 40 00 04
TP-LINK	TLWR741NV4	0x0040	07 41 00 04
TP-LINK	TLWR841NV8	0x0040	08 41 00 08
TP-LINK	TL-WR1041N-v2	0x0040	10 41 00 02
TP-LINK	tl-wr2543-v1	0x0040	25 43 00 01
TP-LINK	TLWDR3600V1	0x0040	36 00 00 01
TP-LINK	TLWDR4300V1	0x0040	43 00 00 01
TP-LINK	TLWDR4310V1	0x0040	43 10 00 01

11 Read more at mktplikfw.c." http://git.openwrt.org/?p=openwrt.git;a=blob;f=tools/firmware-utils/src/mktplinkfw.c

TP-LINK BIN Header layout[12]

```
  0   1   2   3   4   5   6   7   8   9   a   b   c   d   e   f
+---------------------------------------------------------------+
|    version    |          vendor_name...                       |
+---------------------------------------------------------------+
|                       ...vendor_name          | fw_version... |
+---------------------------------------------------------------+
|                       ...fw_version...                        |
+---------------------------------------------------------------+
|                       ...fw_version                           |
+---------------------------------------------------------------+
|     hw_id     |    hw_rev     |     unk1      |    md5sum1... |
+---------------------------------------------------------------+
|                         ...md5sum1            |     unk2      |
+---------------------------------------------------------------+
|                            md5sum2                            |
+---------------------------------------------------------------+
|     unk3      |   kernel_la   |   kernel_ep   |   fw_length   |
+---------------------------------------------------------------+
|  kernel_ofs   |   kernel_len  |   rootfs_ofs  |  rootfs_len   |
+---------------------------------------------------------------+
|   boot_ofs    |   boot_len    |ver_hi |ver_mid| ver_lo| pad...|
+---------------------------------------------------------------+
|                           ...pad...                           |
+---------------------------------------------------------------+
```

12 Lihat “TRX vs. TRX2 vs. BIN.”, Internet: http://wiki.openwrt.org/doc/techref/header.