

# **Reverse Engineering Camera Firmware (IP CAM)**

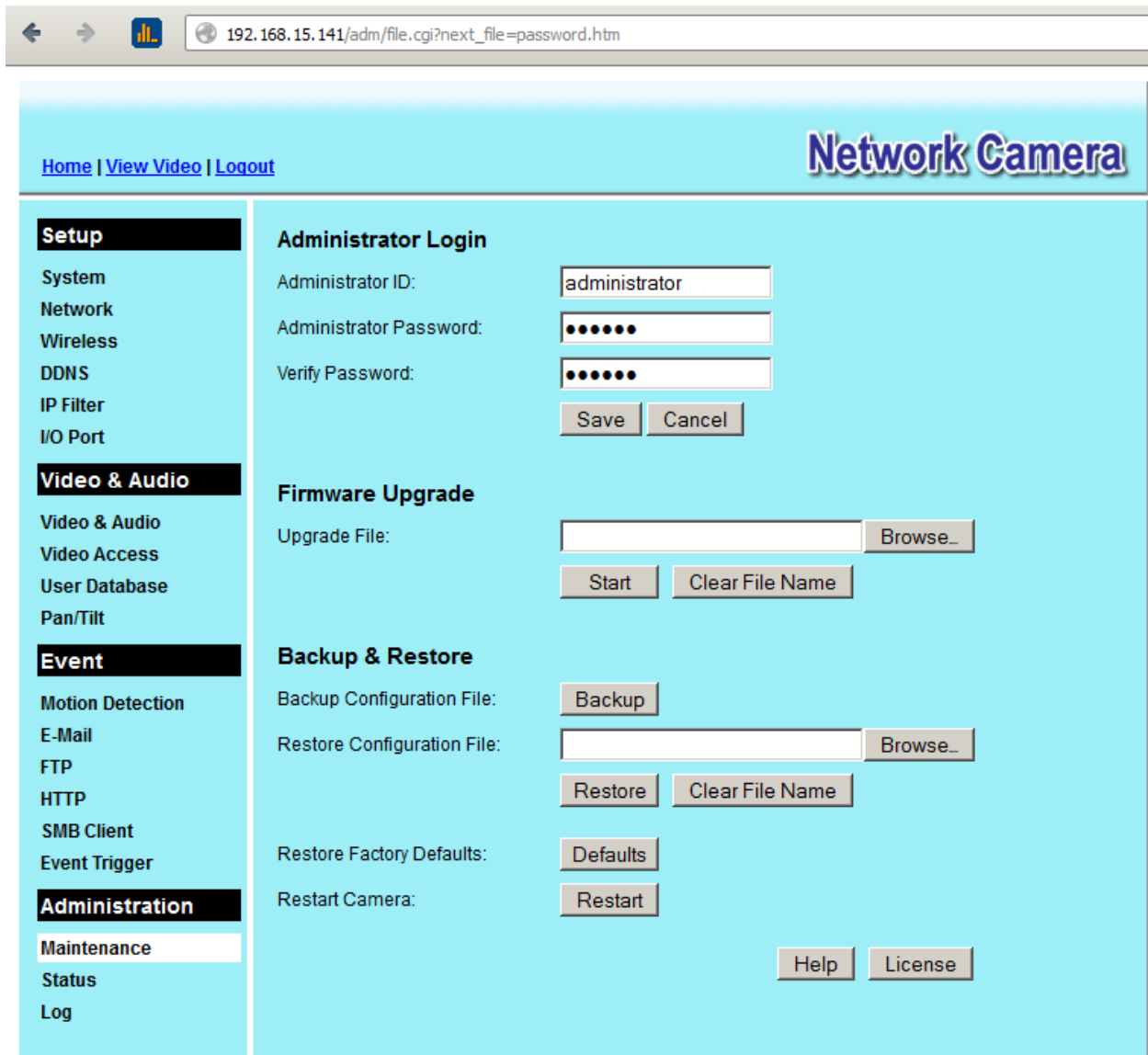
## **Introduction:-**

In this tutorial we are going to understand the ip camera firmware software like what actually running inside ip camera by doing some reverse engineering on the firmware of ip camera.

## **Things we need:-**

For reverse engineering we need to have camera firmware image you can get from you camera vendor website for flush or upgrade the camera firmware.

Camera firmware is nothing but the complied or compressed .bin file for flashing the camera or upgrades the firmware.



Here the interface of the ip camera so that we can browse the .bin firmware file and upgrade the camera firmware.

But here are not going to upgrade the firmware but reverse engineer it for fun and profit.

Now we need to get the camera firmware from the vendor.

```
Applications Places System >_
^ v x root@bt: ~/Desktop/camera
File Edit View Terminal Help
root@bt:~/Desktop/camera# ls
DYM8MF-512-1007R01.bin
root@bt:~/Desktop/camera#
```

Here the required the bin camera file or firmware.

First we need to analyze that what this file is all about (using Linux file utility)

```
Applications Places System >_
^ v x root@bt: ~/Desktop/camera
File Edit View Terminal Help
root@bt:~/Desktop/camera# file DYM8MF-512-1007R01.bin
DYM8MF-512-1007R01.bin: data
root@bt:~/Desktop/camera#
```

So using file utility we did not find any interesting thing. Let move further

Now we are going to use hexdump and string utility against the firmware for more information.

```
Applications Places System >_
^ v x root@bt: ~/Desktop/camera
File Edit View Terminal Help
root@bt:~/Desktop/camera# string -n DYM8MF-512-1007R01.bin > strings.out
```

```
Applications Places System >_
^ v x root@bt: ~/Desktop/camera
File Edit View Terminal Help
root@bt:~/Desktop/camera# hexdump -C DYM8MF-512-1007R01.bin > hex.out
```

So first we need to look at the strings output and analyze it.

```
ran out of input data
-- System halted
Uncompressing Linux...
done, booting the kernel.
pnh!cDouq!
fsl(tyFJ}e
2?>+Ni{(mP
[V:[Ls+"d,F
/@^!e7d*;m
~bt8Jrt|20
eAk$WA[7      Z
>0g<B?p!@v
`kdpSdps(X
m7RlFpF(vPt@
ue0S0{g{gkw
onUyH=Vb}{3k
za0rvuw23-
}e5NYaaYuue
hofY{c>JA[
^ VQ^QEMV>
W0630kX}SY
Z#y!x*=6lF
rvx8x{8836
```

Here we can see that firmware is based on LINUX OPERATING SYSTEM for embedded devices. So let also see the hexdump may be it can reveal more interesting things for us.

```

0000a820 6d 65 6f 75 74 0a 00 00 52 65 63 65 69 76 65 20 |meout...Receive |
0000a830 46 69 6c 65 20 73 69 7a 65 3a 30 78 25 38 78 28 |File size:0x%8x(|
0000a840 25 64 29 0a 00 00 00 00 49 6d 61 67 65 20 4f 4b |%d)....Image OK|
0000a850 21 0a 00 00 43 61 6c 6c 20 53 79 73 74 65 6d 20 |!...Call System |
0000a860 52 65 73 65 74 20 21 0a 00 00 00 00 3c 73 63 72 |Reset !....<scr|
0000a870 69 70 74 20 6c 61 6e 67 75 61 67 65 3d 27 6a 61 |ipt language='ja|
0000a880 76 61 73 63 72 69 70 74 27 3e 3c 21 2d 2d 20 68 |vascript'><!-- h|
0000a890 69 64 65 00 3c 2f 68 65 61 64 3e 3c 62 6f 64 79 |ide.</head><body|
0000a8a0 3e 3c 62 6c 6f 63 6b 71 75 6f 74 65 20 63 6c 61 |><blockquote cla|
0000a8b0 73 73 3d 27 73 74 79 6c 65 31 27 3e 49 6e 76 61 |ss='style1'>Inva|
0000a8c0 6c 69 64 20 66 69 6c 65 2e 3c 2f 62 6c 6f 63 6b |lid file.</block|
0000a8d0 71 75 6f 74 65 3e 3c 2f 62 6f 64 79 3e 3c 2f 68 |quote></body></h|
0000a8e0 74 6d 6c 3e 00 00 00 00 3c 73 63 72 69 70 74 20 |tml>....<script |
0000a8f0 6c 61 6e 67 75 61 67 65 3d 27 4a 61 76 61 53 63 |language='JavaSc|
0000a900 72 69 70 74 27 20 74 79 70 65 3d 27 74 65 78 74 |ript' type='text|
0000a910 2f 6a 61 76 61 73 63 72 69 70 74 27 3e 00 00 00 |/javascript'>...|
0000a920 3c 21 2d 2d 20 53 74 61 72 74 20 53 63 72 69 70 |<!-- Start Scrip|
0000a930 74 09 00 00 76 61 72 20 6d 61 78 63 68 61 72 73 |t...var maxchars|
0000a940 20 3d 20 35 30 3b 76 61 72 20 63 68 61 72 63 6f | = 50;var charco|
0000a950 75 6e 74 20 3d 20 30 3b 00 00 00 00 66 75 6e 63 |unt = 0;....func|
0000a960 74 69 6f 6e 20 70 70 70 28 29 7b 76 61 72 20 63 |tion ppp(){var c|
0000a970 66 20 3d 20 64 6f 63 75 6d 65 6e 74 2e 66 6f 72 |f = document.for|
0000a980 6d 73 5b 30 5d 3b 69 66 20 28 63 68 61 72 63 6f |ms[0];if (charco|
0000a990 75 6e 74 20 3c 20 6d 61 78 63 68 61 72 73 29 7b |unt < maxchars){|

```

But looking at the hexdump we didn't figure out any new interesting things only we can see html and other commands cgi scripts callings.

Now we are going to use binwalk against the firmware image it may give some false result.

```

root@bt:~/Desktop/camera# binwalk DYM8MF-512-1007R01.bin
DECIMAL      HEX          DESCRIPTION
-----
142296      0x22B08      gzip compressed data, from Unix, last modified: Wed Oct 27 01:18:56 2010, max compression
917504      0xE0000      Squashfs filesystem, little endian, version 3.0, size: 2819987 bytes, 562 inodes, blocksize: 65536 bytes, created: Wed Oct 27 01:22:09 2010
4186004     0x3FDF94     gzip compressed data, from NTFS filesystem (NT), DD-WRT date: Wed Dec 31 19:00:00 1969
root@bt:~/Desktop/camera#

```

It give very useful information like the compressesion method used is gzip and the file system is squashfs little endian version 3 and more interesting information.

Now we know that firmware is based on Linux and the file system is squashfs and the compressed data format is gzip now we are going to use another utility so that we can extract and decompress the data.

```

root@bt:~/Desktop/camera/firm# ./extract-nq.sh DYM8MF-512-1007R01.bin
Firmware Mod Kit (build-ng) 0.78 beta

Scanning firmware...

DECIMAL      HEX          DESCRIPTION
-----
917504      0xE0000     Squashfs filesystem, little endian, version 3.0, size: 2819987 bytes, 562 inodes, blocksize: 65536 bytes, created: Wed Oct 27 01:22:09 2010

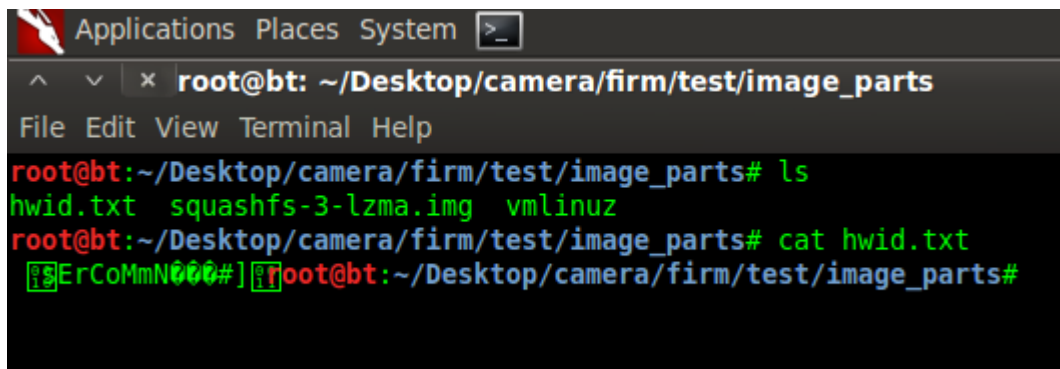
Extracting 917504 bytes of header image at offset 0
Extracting squashfs file system at offset 917504
Extracting 32 byte footer from offset 4194272
Extracting squashfs files...
Firmware extraction successful!
Firmware parts can be found in '/root/Desktop/camera/firm/fmk/*'
root@bt:~/Desktop/camera/firm#

```

Here the firmware has been extracted using the firmware mod kit utility.

Now we can also decompress the data and we find very interesting things inside it (decompression can be done using decompression utility for squashfs system included in firmware mod kit)

Here we can see that the camera firmware is from sercommn (vendor)

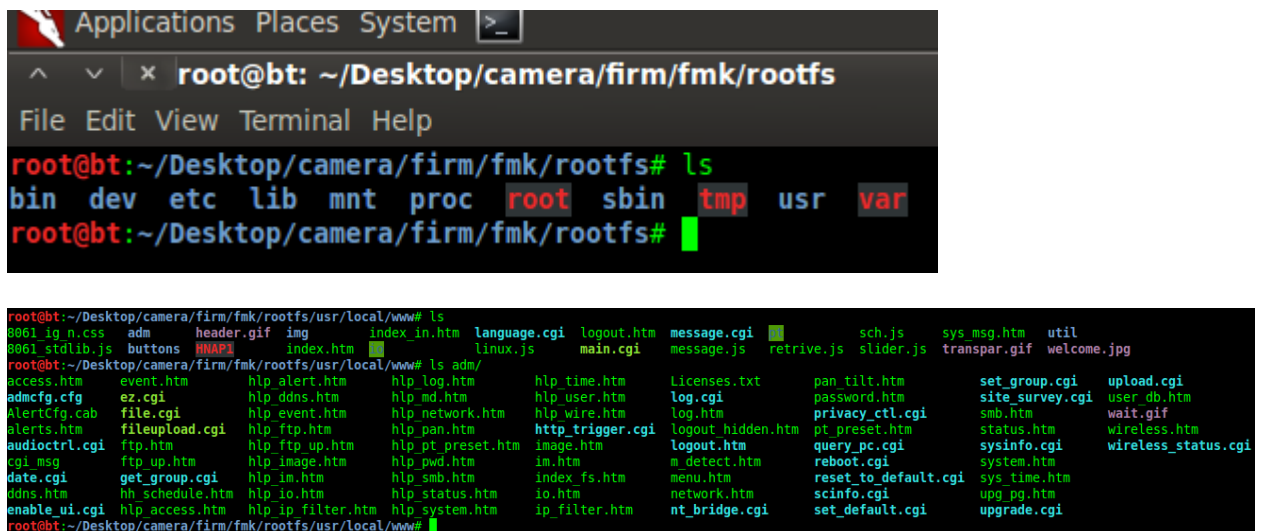


```

Applications Places System >_
root@bt: ~/Desktop/camera/firm/test/image_parts
File Edit View Terminal Help
root@bt:~/Desktop/camera/firm/test/image_parts# ls
hwid.txt  squashfs-3-lzma.img  vmlinuz
root@bt:~/Desktop/camera/firm/test/image_parts# cat hwid.txt
[ErCoMmN000#]
root@bt:~/Desktop/camera/firm/test/image_parts#

```

Now we can see all the file system inside the firmware image and everything



```

Applications Places System >_
root@bt: ~/Desktop/camera/firm/fmk/rootfs
File Edit View Terminal Help
root@bt:~/Desktop/camera/firm/fmk/rootfs# ls
bin dev etc lib mnt proc root sbin tmp usr var
root@bt:~/Desktop/camera/firm/fmk/rootfs#
root@bt:~/Desktop/camera/firm/fmk/rootfs/usr/local/www# ls
0061_ig_n.css adm header.gif img index_in.htm language.cgi logout.htm message.cgi sch.js sys_msg.htm util
0061_stdlib.js buttons HNAPI index.htm linux.js main.cgi message.js retrieve.js slider.js transpar.gif welcome.jpg
root@bt:~/Desktop/camera/firm/fmk/rootfs/usr/local/www# ls adm/
access.htm event.htm hlp_alert.htm hlp_log.htm Licenses.txt pan_tilt.htm set_group.cgi upload.cgi
admcfg.cfg ez.cgi hlp_ddns.htm hlp_md.htm log.cgi password.htm site_survey.cgi user_db.htm
AlertCfg.cab file.cgi hlp_event.htm hlp_network.htm log.htm hlp_wire.htm privacy_ctl.cgi wait.gif
alerts.htm fileupload.cgi hlp_ftp.htm hlp_pan.htm http_trigger.cgi logout_hidden.htm pt_preset.htm status.htm wireless.htm
audioctrl.cgi ftp.htm hlp_ftp_up.htm hlp_pt_preset.htm image.htm logout.htm query_pc.cgi sysinfo.cgi sysinfo.cgi wireless_status.cgi
cgi_msg ftp_up.htm hlp_image.htm hlp_pwd.htm m_detect.htm menu.htm reboot.cgi system.htm
date.cgi get_group.cgi hlp_im.htm hlp_smb.htm menu.htm reset_to_default.cgi sys_time.htm sys_time.htm
ddns.htm hh_schedule.htm hlp_io.htm hlp_status.htm io.htm network.htm scinfo.cgi upg_pg.htm upg_pg.htm
enable_ui.cgi hlp_access.htm hlp_ip_filter.htm hlp_system.htm ip_filter.htm nt_bridge.cgi set_default.cgi upgrade.cgi
root@bt:~/Desktop/camera/firm/fmk/rootfs/usr/local/www#

```

Here we can see each n every cgi and other files and folder inside the camera's webserver.

```
<meta name="description" content="RC8061">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<META http-equiv="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="Cache-Control" CONTENT="no-cache">
<meta HTTP-EQUIV="Expires" CONTENT="Mon, 06 Jan 1990 00:00:01 GMT">

<title>Network Camera</title>

<script language="JavaScript" type="text/javascript" src="linux.js"></script>
<script language="JavaScript" type="text/javascript" src="8061_stdlib.js"></script>
<script language="JavaScript" type="text/javascript" src="message.js"></script>

<script language="JavaScript" type="text/javascript">
<!-- Start Script
```

Here is the index html file which is running inside the camera, model is 8061.

## **Conclusion:-**

So finally you reverse engineered the camera image or firmware hack it according to your own wish (make required manipulation according to need).

## **About the Author:-**

Prayas Kulshrestha

Independent security researcher and pen tester