root@tester-desktop:~# rm -rf /*

LINUXOIDS ERROR OR EPIC FAIL

MONDAY, FEBRUARY 4, 2013

Firmware DVR Asenware DN8004 (HI3515 based)

The story begins with the fact that this device Asenware DN8004 was purchased in China, through AliExpress. Its functionality, respectively, was quite trivial, and is similar to many similar devices, but he came home quite well. The only thing I do not really like - video transmission over the network was carried out in 2 ways. The first - port 2200, without encryption, in its own proprietary utility for Windows, the second - in ActiveX-component in IE. Both are fine in Windows, but not for Linux, on which I work. In principle, a utility for viewing, after installing VC + + 6 in wine starts, and even more or less worked, but the result did not satisfy me. It was decided to get to the guts devaysa to understand how to correct discrimination on the OS.
Port scans revealed the presence of telnetd-service in the device, but in, as expected, proved to recovery record. Traditional options do not come, and it became clear that without the firmware distribution itself it will not work.

```
root@tester-HP:~# nmap -v -sS 10.10.0.175

Starting Nmap 5.21 ( http://nmap.org ) at 2013-02-03 18:36 MSK
Initiating ARP Ping Scan at 18:36
Scanning 10.10.0.175 [1 port]
Completed ARP Ping Scan at 18:36, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 18:36
Completed Parallel DNS resolution of 1 host. at 18:37, 13.00s elapsed
Initiating SYN Stealth Scan at 18:37
Scanning 10.10.0.175 [1000 ports]
```

The search began with the inside page of the device. It is noteworthy that the absence of established ActiveX-component, or just when coming from any browser, we see the line:

If your browser does not support the ActiveX to download, please click here

The link leads to http://dy.vip620.com/network/dfcb/Install.exe
When trying directly to http://dy.vip620.com/ I was disappointed - there is a login page with a captcha. Experiments with the selection of a login failed, so I got the idea to go to the other end, and, as it turned out for good reason. Link http://dy.vip620.com/firmware/ led me exactly where it is needed.
was downloaded firmware http://dy.vip620.com/firmware/upgrade/commonnew/dn8004_v0401.rar
on the device at the time, 0302 version was installed.
Next it was for small. Unpack and see what's what. Unpack the archive consists of 3 files - app_dn8004_v0401, kernel_dn8004_sdkv051_v0002 and rootfs_dn8004_v0401 binwalk findings made clear with what I was dealing with:

```
tester@tester-HP:~/Downloads/dn8004_v0401$ binwalk kernel_dn8004_sdkv051_v00

DECIMAL         HEX             DESCRIPTION
----------------------------------------------------------------------------
-----------------------
0               0x0             uImage header, header size: 64 bytes, header
CRC: 0xCCC4A14, created: Sat Apr 28 06:04:43 2012, image size: 1531236 bytes
Data Address: 0xC0800000, Entry Point: 0xC0800000, data CRC: 0x4E6B0B04, OS:
Linux, CPU: ARM, image type: OS Kernel Image, compression type: none, image
name: xsjlinux
```

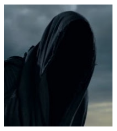
I was mainly interested in the rootfs and the app, so I continued to work further with them.
app_dn8004_v0401 could mount no problems with:

TAG CLOUD

BLOG ARCHIVE

▼ 2013 (2)
▼ February (2)
Firmware DVR Asenware DN8004 (HI3515 based) part ...
Firmware DVR Asenware DN8004 (HI3515 based)
► 2011 (3)
► 2010 (21)

ABOUT ME



AZ

Previously, technical support analyst, a large company, for the production of printed circuit technology. Linuhovod, sometimes programmer.

VIEW PROFILE

mount-t cramfs-o loop ./app_dn8004_v0401 / mnt / temp_app

This file contained the entire program shell that used by DVR. Also there are kernel modules and libraries for video input.
Root filesystem is mounted more difficult (thanks http://bitjuggling.blogspot.ru/2008/08/jffs2-to-targz.html):

modprobe mtdram total_size = 131072 erase_size = 128

modprobe mtdblock

dd if = rootfs_dn8004_v0401 of = / dev/mtdblock0

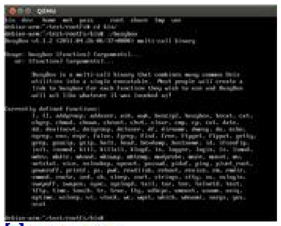
mount-t jffs2 / dev/mtdblock0 / mnt / temp_rootfs

Once mounted, I saw exactly what he wanted - the file structure emb linux.
Cursory review made it clear that the system is quite truncated (which, in general, is not surprising.) Almost all of the possible commands are symlinks to busybox. The / etc contains the most basic config system:

```
/etc # ls
dhclient.script  init.d          passwd          resolv.conf
fs-version       inittab         ppp             services
fstab            memstat.conf    profile         shadow
group            mtab            protocols       udev
```

Since the initial goal was to go to a working device, I decided to try to decipher the root hash and user «ken».
Used for decoding the infamous John The Ripper and several dictionaries with the Internet. The result was not, exactly as after 5 days of brute force. It became clear that the problem can be solved only modifying the firmware.

Was chosen to modify rootfs (as it turned out later - for good reason).
A journaling file system JFFS2 is quite common on various mobile devices, tablets, routers, etc, but with her personally, I have come across for the first time.
To get that I need to replace the hashes into CFS firmware, and to view the version and the team built busybox, I picked up a virtual OS ARM-architecture. Helped me in this way and QEMU emulator installed ARM Debian Lenny. After copying the firmware libraries CFS, I was able to run busybox. The output of:



Увеличить

Hashes of root and user I took of the same system.
After that, only needed to pack our FS and pour it on the DVR. I understand that I will attempt only 1-a, as was done through the shell filling device.
FS was packed with:

mkfs.jffs2-p-d / mnt / temp_rootfs-e 128KiB-o rootfs_dn8004_v0401

I was just confused Erase Block size memory chips (WINBOND W29GL064CB7S) . In Datasheet for the chip such information was not specified.
The long search for a network, so do not give anything specific, so I had to pick that. Compare files in hexedit to

say, that the source and the file are structurally similar, minus the change.
However, the error was still somewhere. Whether it was necessary to collect a key 64KiB, it was still there a reason that I missed. Flash was successful, but after rebooting the device is no longer on readiness.

As I have previously dealt with a variety of devices and firmware, as well as the add fills a self-assembled programmer (or more wires and LPT-connector :)) then, of course, the first thought that came to my mind was just a flash manual rewrite watering details .
But there were problems - first is the lack of a hair dryer is available, and you also can not flash "simple" way.
After consulting with the guys from ROM.by, I realized that manual pereproshika Parallel Flash, Kojima was W29GL064CB7S, it will be very difficult, as the assembly programmer for it. Leaving this option as a last resort, I began to study the board for possible solutions:

Увеличить

And it was found in a UART-out on the board. Since I was not quite sure what it is UART (rx \ tx certainly talk about it, but I did not believe that the Chinese would bother with output), I used an oscilloscope to confirm. This is, without a doubt he was.
Next, find a stock of old cable for the phone to PL-2303, I joined the unfortunate through minicom (arguments start minicom-l -8-c on-s, hardware flow control), and saw exactly what he wanted.

Увеличить

```
U-Boot 2008.10 (Aug  3 2011 - 05:16:21)

DRAM:  128 MB
manufacturer id is 0x15
Flash:  8 MB
In:    serial
Out:   serial
Err:   serial
Press CTRL-C to abort autoboot in jpeg decoding ...
```

Many things became clear:
In the role of boot second level to use U-Boot, which controls the loading of the kernel, which in turn loads the CFS and Cramfs.
Analyse.log shows that the kernel can not load a flooded CFS and falls into the Kernel Panic.
By pressing Ctrl + C at startup devaysa might go to the command line U-Boot. From there, I set the Environment to download the image CFS network, bringing it to the following form:

```
bootcmd = bootm 0x80080000

bootdelay = 1

baudrate = 115200

ethaddr = 00:11:00:34:76:21

bootfile = "uImage"
```

```
filesize = 14BCD8

fileaddr = 80200000

netmask = 255.255.255.0

bootargs = mem = 48M console = ttyAMA0, 115200 root = 1f02 rootfstype = jffs2
mtdparts = physmap-flash.0: 512K (boot), 1536K (kernel), 2M (rootfs), 4M (app), 8M
@ 0 (flash) pcimod = host pciclksel = 1

gatewayip = 10.10.0.1

serverip = 10.10.0.254

ipaddr = 10.10.0.180

stdin = serial

stdout = serial

stderr = serial

verify = n

ver = U-Boot 2008.10 (Aug 3 2011 - 5:16:21)
```

Attempts to browse through the file system ls fsinfo or hang the device. The team also mtest or display Fatal Error, or hang the system.
Conclusion flinfo showed:

```
Bank # 1: CFI conformant FLASH (8 x 8) Size: 8 MB in 135 Sectors

AMD Standard command set, Manufacturer ID: 0x15, Device ID: 0x00

Erase timeout: 2048 ms, write timeout: 1 ms

Buffer write timeout: 1 ms, buffer size: 32 bytes
```

Arguments during boot:

```
Kernel command line: mem = 48M console = ttyAMA0, 115200 root = 1f02
rootfstype = JFFS2 mtdparts = physmap-flash.0: 512K (Boot), 1536K (kernel), 2M
(rootfs), 4M (app), 8M @ 0 ( flash) 1
```

Log loading also gave an idea of how the data is arranged in memory:

```
Creating 5 MTD partitions on "physmap-flash.0":

0x00000000-0x00080000: "boot"

0x00080000-0x00200000: "kernel"

0x00200000-0x00400000: "rootfs"

0x00400000-0x00800000: "app"

0x00000000-0x00800000: "flash"
```

Next, it was only necessary to fill in the original firmware rootfs in temporary memory and write it back to where it should be.
The actual location of sections in memory was at 0x80000000, figuring out which took a few hours browsing through the listing dump manually from the command md (not would immediately draw attention to the withdrawal of Environment).
The installed version of U-Boot was somewhat curtailed, so many teams simply were not available (such as working with USB support), so I stopped to fill a form tftp.
Fill in the following manner:

```
xsjlinux # tftp 0xc0000000 rootfs_dn8004_v0401
```

```
Hisilicon ETH net controler

MAC: 00-11-00-34-76-21

UP_PORT: phy status change: LINK = UP: DUPLEX = FULL: SPEED = 100M

TFTP from server 10.10.0.254; our IP address is 10.10.0.180

Download Filename 'rootfs_dn8004_v0401'.

Download to address: 0xc0000000

Downloading:% # [Connected]

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # [1.000 MB]

# # # # # # # # #

1.307 MB download ok.


done

Bytes transferred = 1363752 (14cf28 hex)
```

Further purification of memory before casting:

```
erase 0x80200000 +0 x14cf28
```

Next fill CFS in place:

```
cp.b 0xc0000000 0x80200000 0x14cf28
```

After all the action the system was restarted, but it's not over. Now booting the kernel and CFS is successful, but could not mount a cramfs-section.
I assumed that the firmware through the software I slipped placement cramfs-section, for which reason, now-loaded, he did not.
By analogy with CFS was filled cramfs (start position it according 0x80400000). Writing errors have arisen, but now the situation was of a different kind - to download kernel and CFS held, cramfs volume mounted, but then the conclusion:

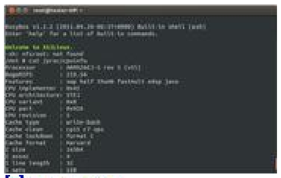
```
cramfs: bad compressed blocksize 4292875699

cramfs: bad compressed blocksize 4291992747
```

Software DVR-and still does not run.
Checking the firmware using fsck, it became clear that the problem is probably in existing bad-blocks, and it did not bode well. In the case of NAND, was the team that made "REMAP", but then it was not. The only solution that came to mind is to cut the size of the firmware, and hope that it reaches a bedy.
From the firmware has been removed entirely unnecessary cab for ActiveX viewing through IE, whom I never used, then the file system has been re-assembled and sewn into place.
Device again come to life, and on the screen, I saw a familiar image divided into 4 of the screen. In minicom terminal, I saw not a very pleasant or Save, as a login. It was unfortunate because I was counting on the fact that the terminal output recovery record.
Did not want to give up, so I went back for a detailed examination of the firmware, but now paid special attention to the file name to cramfs, and, as it turned out, for good reason.
In the depths of the FS was found script load kernel modules are probably responsible for the video input. Because modules can load only the root, the solution was obvious.
Modified before the file was placed shadow root filesystem cramfs, and the script has been added to a very simple line:

```
mv. / shadow / etc / shadow
```

FS was re-packed and laced. Restart, login, and we finally get into the system.



Увеличить

In the future will need to understand how to organize video-output flows in the network, but most of the work is still left behind.

AUTHOR: AZ AT 2:24 Рекомендовать в Google

LABELS: CRAMFS , DVR , FIRMWARE , HI3515 , JFFS2 , LINUX , NMAP , ROOT , U-BOOT , UART

NO COMMENTS YET:

POST A COMMENT

Введите комментарий...

Подпись комментария: Аккаунт Google

Публикация Просмотр

Next Home Previous

Subscribe to: Post Comments (Atom)