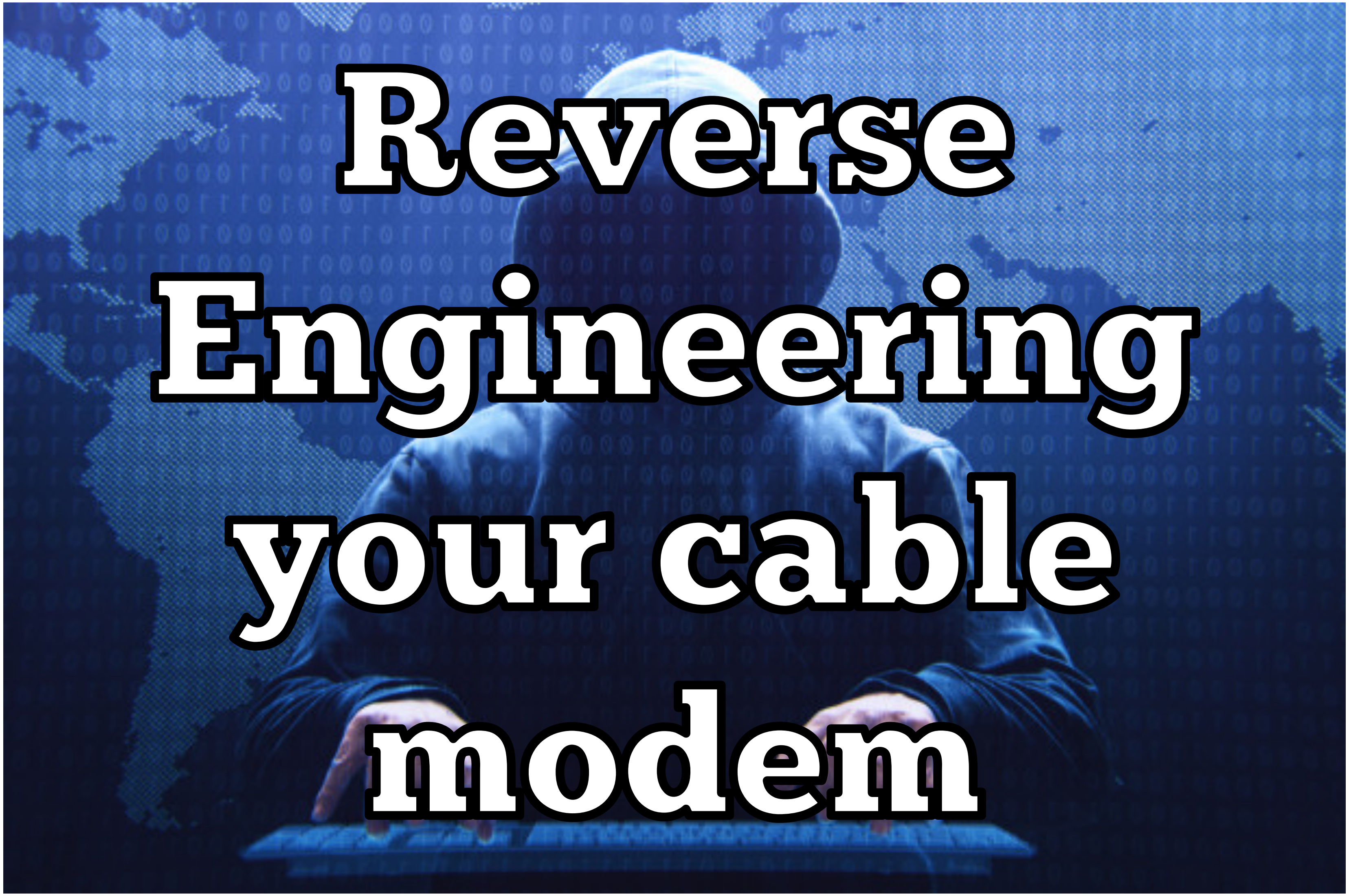


Reverse Engineering your cable modem

fG! @ 0xOPOSEC OUT 2019

Last month

- How to:
  - Achieve serial console access.
  - Dump firmware.
  - Extract filesystem.
  - Patch firmware into privilege escalation.

Today's agenda

- How to build and how attach a debugger.
- How to decrypt all passwords.
- Remote updates security.

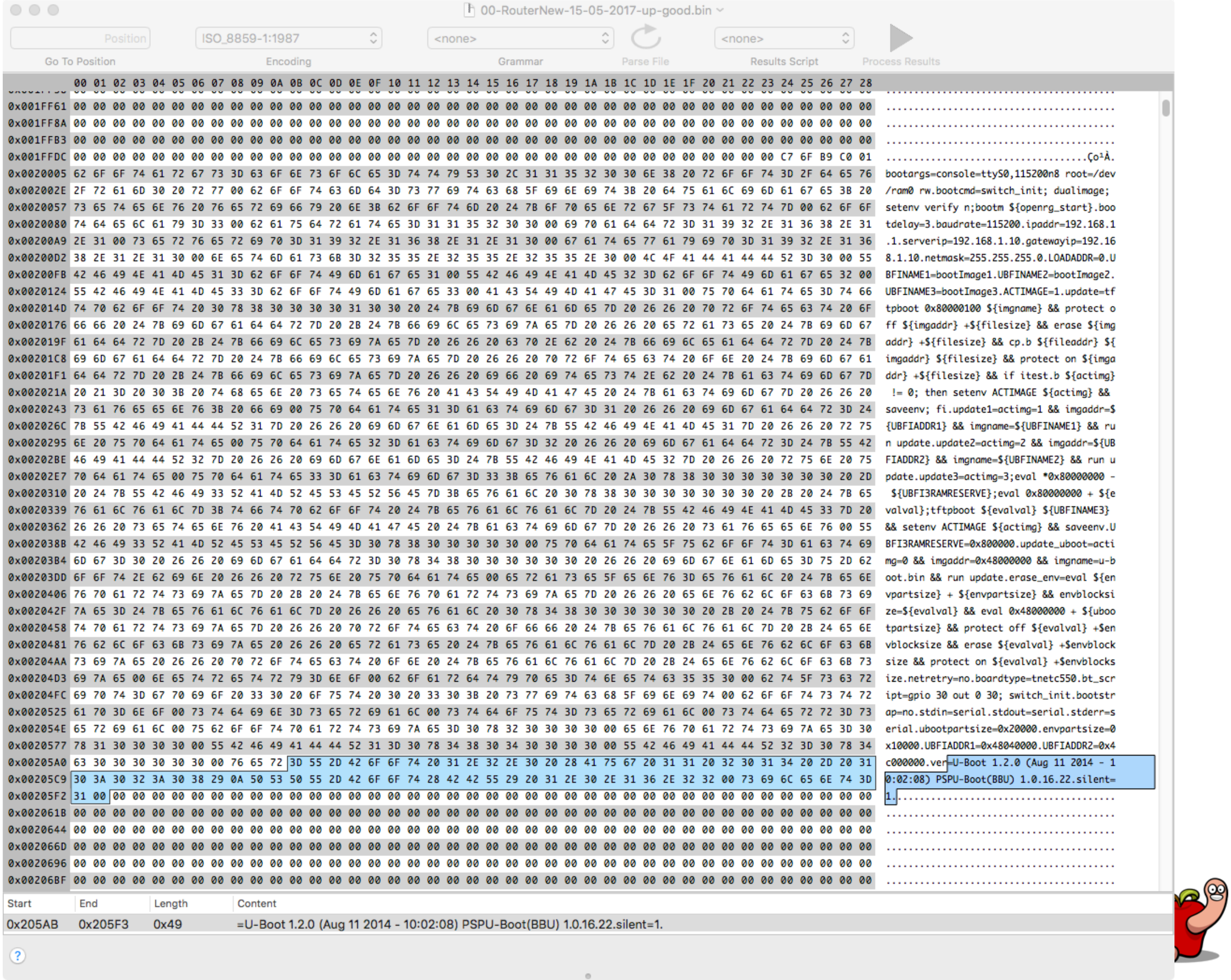
Target

- NOS CVE-30360 cable modem.
- OpenRG software by Jungo (now Cisco RG).
- Firmware version 4.11.3.7.62.3.52.

WARNING
ASSUMPTIONS
AHEAD

Assumptions

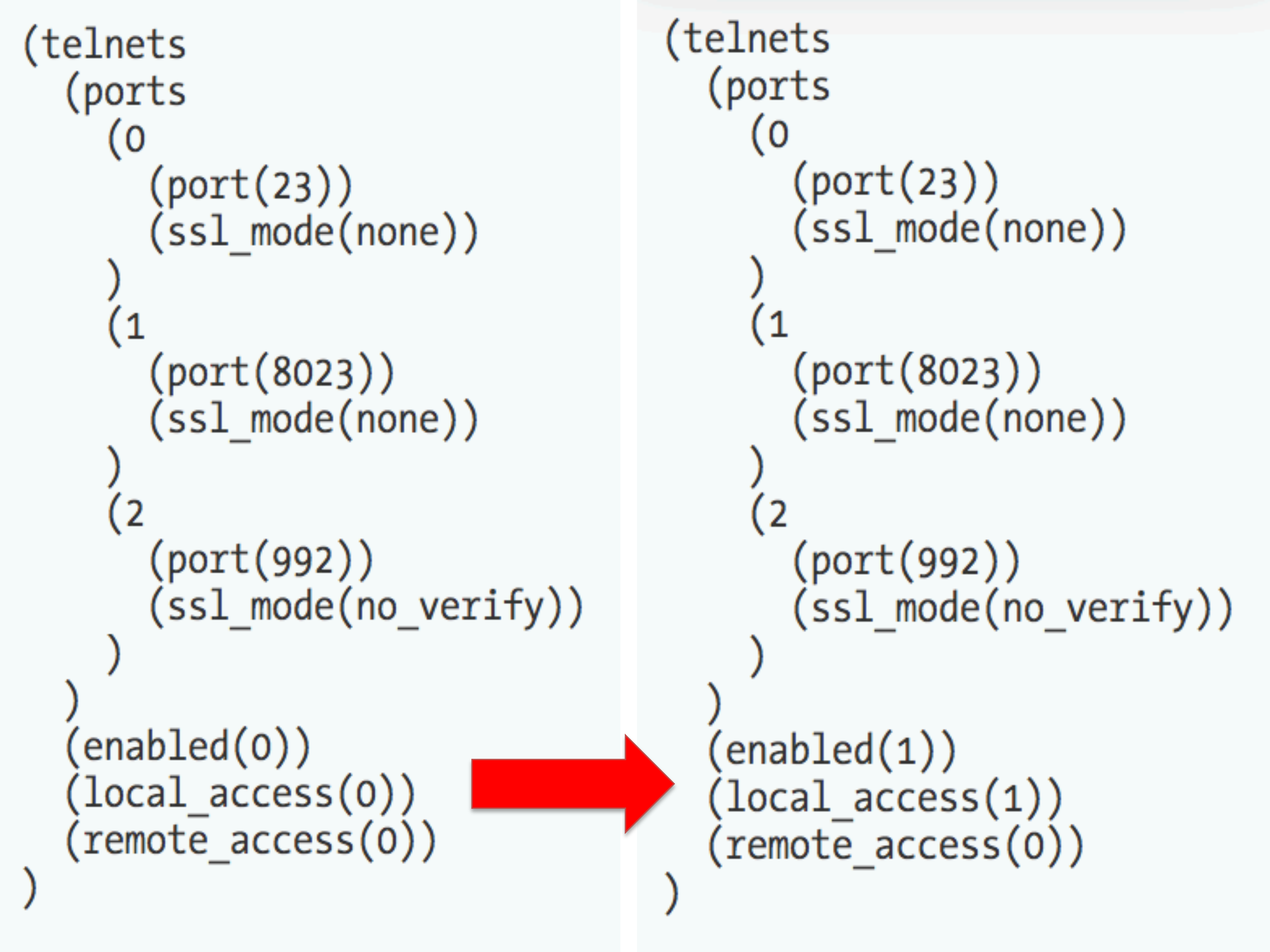
- Patched firmware with:
    - Serial console.
    - Telnet.
    - Administrator privileges.



00-RouterNew-15-05-2017-up-good.bin

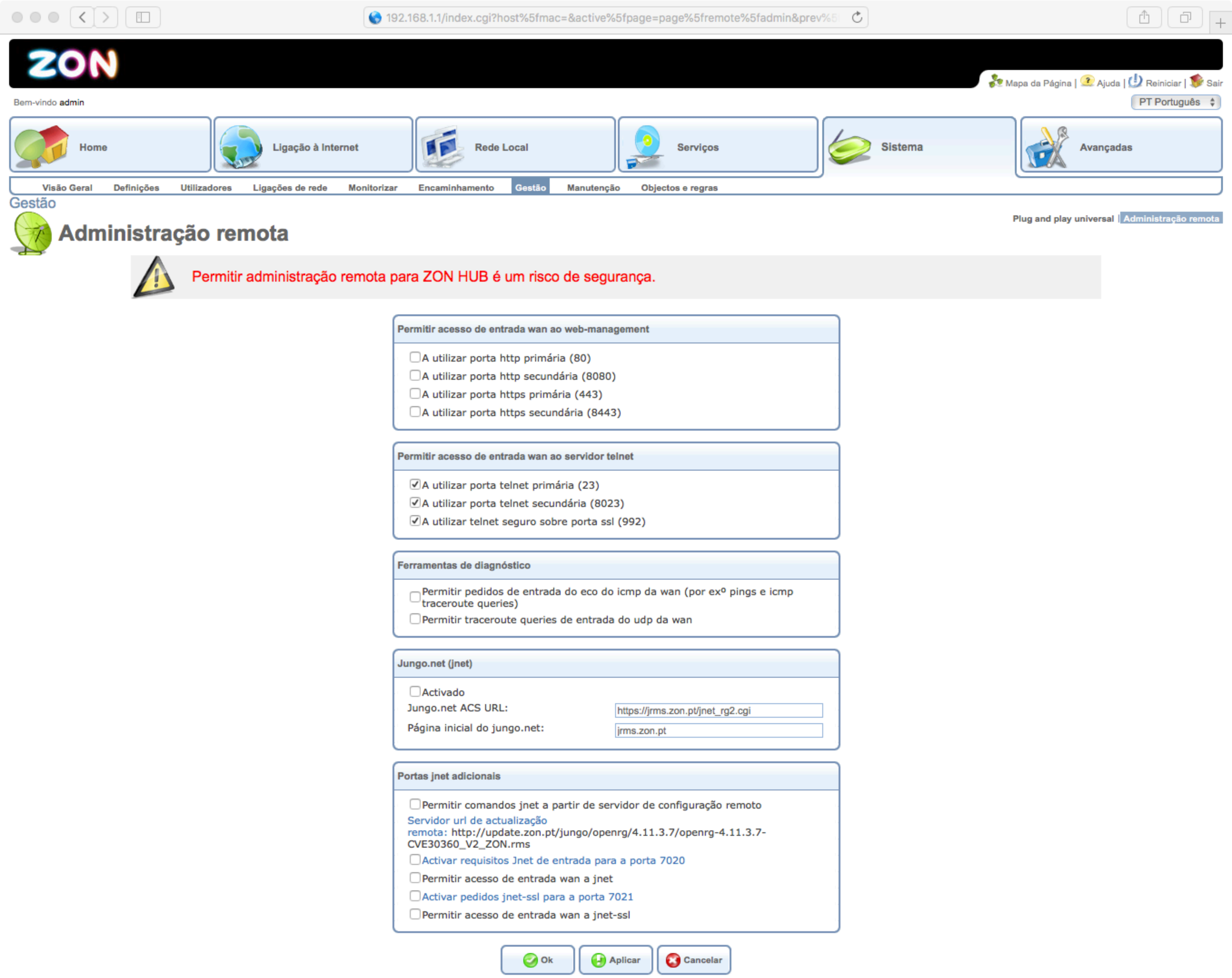
```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28
0x001FF61 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .........................................
0x001FF8A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .........................................
0x001FFB3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .........................................
0x001FFDC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 C7 6F B9 C0 01  ....................................Ço¹À.
0x0020005 62 6F 6F 74 61 72 67 73 3D 63 6F 6E 73 6F 6C 65 3D 74 74 79 53 30 2C 31 31 35 32 30 30 6E 38 20 72 6F 6F 74 3D 2F 64 65 76  bootargs=console=ttyS0,115200n8 root=/dev
0x002002E 2F 72 61 6D 30 20 72 77 00 62 6F 6F 74 63 6D 64 3D 73 77 69 74 63 68 5F 69 6E 69 74 3B 20 64 75 61 6C 69 6D 61 67 65 3B 20  /ram0 rw.bootcmd=switch_init; dualimage; 
0x0020057 73 65 74 65 6E 76 20 76 65 72 69 66 79 20 6E 3B 62 6F 6F 74 6D 20 24 7B 6F 70 65 6E 72 67 5F 73 74 61 72 74 7D 00 62 6F 6F  setenv verify n;bootm ${openrg_start}.boo
0x0020080 74 64 65 6C 61 79 3D 33 00 62 61 75 64 72 61 74 65 3D 31 31 35 32 30 30 00 69 70 61 64 64 72 3D 31 39 32 2E 31 36 38 2E 31  tdelay=3.baudrate=115200.ipaddr=192.168.1
0x00200A9 2E 31 00 73 65 72 76 65 72 69 70 3D 31 39 32 2E 31 36 38 2E 31 2E 31 30 00 67 61 74 65 77 61 79 69 70 3D 31 39 32 2E 31 36  .1.serverip=192.168.1.10.gatewayip=192.16
0x00200D2 38 2E 31 2E 31 30 00 6E 65 74 6D 61 73 6B 3D 32 35 35 2E 32 35 35 2E 32 35 35 2E 30 00 4C 4F 41 44 41 44 44 52 3D 30 00 55  8.1.10.netmask=255.255.255.0.LOADADDR=0.U
0x00200FB 42 46 49 4E 41 4D 45 31 3D 62 6F 6F 74 49 6D 61 67 65 31 00 55 42 46 49 4E 41 4D 45 32 3D 62 6F 6F 74 49 6D 61 67 65 32 00  BFINAME1=bootImage1.UBFINAME2=bootImage2.
0x0020124 55 42 46 49 4E 41 4D 45 33 3D 62 6F 6F 74 49 6D 61 67 65 33 00 41 43 54 49 4D 41 47 45 3D 31 00 75 70 64 61 74 65 3D 74 66  UBFINAME3=bootImage3.ACTIMAGE=1.update=tf
0x002014D 74 70 62 6F 6F 74 20 30 78 38 30 30 30 30 31 30 30 20 24 7B 69 6D 67 6E 61 6D 65 7D 20 26 26 20 70 72 6F 74 65 63 74 20 6F  tpboot 0x80000100 ${imgname} && protect o
0x0020176 66 66 20 24 7B 69 6D 67 61 64 64 72 7D 20 2B 24 7B 66 69 6C 65 73 69 7A 65 7D 20 26 26 20 65 72 61 73 65 20 24 7B 69 6D 67  ff ${imgaddr} +${filesize} && erase ${img
0x002019F 61 64 64 72 7D 20 2B 24 7B 66 69 6C 65 73 69 7A 65 7D 20 26 26 20 63 70 2E 62 20 24 7B 66 69 6C 65 61 64 64 72 7D 20 24 7B  addr} +${filesize} && cp.b ${fileaddr} ${
0x00201C8 69 6D 67 61 64 64 72 7D 20 24 7B 66 69 6C 65 73 69 7A 65 7D 20 26 26 20 70 72 6F 74 65 63 74 20 6F 6E 20 24 7B 69 6D 67 61  imgaddr} ${filesize} && protect on ${imga
0x00201F1 64 64 72 7D 20 2B 24 7B 66 69 6C 65 73 69 7A 65 7D 20 26 26 20 69 66 20 69 74 65 73 74 2E 62 20 24 7B 61 63 74 69 6D 67 7D  ddr} +${filesize} && if itest.b ${actimg}
0x002021A 20 21 3D 20 30 3B 20 74 68 65 6E 20 73 65 74 65 6E 76 20 41 43 54 49 4D 41 47 45 20 24 7B 61 63 74 69 6D 67 7D 20 26 26 20   != 0; then setenv ACTIMAGE ${actimg} && 
0x0020243 73 61 76 65 65 6E 76 3B 20 66 69 00 75 70 64 61 74 65 31 3D 61 63 74 69 6D 67 3D 31 20 26 26 20 69 6D 67 61 64 64 72 3D 24  saveenv; fi.update1=actimg=1 && imgaddr=$
0x002026C 7B 55 42 46 49 41 44 44 52 31 7D 20 26 26 20 69 6D 67 6E 61 6D 65 3D 24 7B 55 42 46 49 4E 41 4D 45 31 7D 20 26 26 20 72 75  {UBFIADDR1} && imgname=${UBFINAME1} && ru
0x0020295 6E 20 75 70 64 61 74 65 00 75 70 64 61 74 65 32 3D 61 63 74 69 6D 67 3D 32 20 26 26 20 69 6D 67 61 64 64 72 3D 24 7B 55 42  n update.update2=actimg=2 && imgaddr=${UB
0x00202BE 46 49 41 44 44 52 32 7D 20 26 26 20 69 6D 67 6E 61 6D 65 3D 24 7B 55 42 46 49 4E 41 4D 45 32 7D 20 26 26 20 72 75 6E 20 75  FIADDR2} && imgname=${UBFINAME2} && run u
0x00202E7 70 64 61 74 65 00 75 70 64 61 74 65 33 3D 61 63 74 69 6D 67 3D 33 3B 65 76 61 6C 20 2A 30 78 38 30 30 30 30 30 30 30 20 2D  pdate.update3=actimg=3;eval *0x80000000 -
0x0020310 20 24 7B 55 42 46 49 33 52 41 4D 52 45 53 45 52 56 45 7D 3B 65 76 61 6C 20 30 78 38 30 30 30 30 30 30 30 20 2B 20 24 7B 65   ${UBFI3RAMRESERVE};eval 0x80000000 + ${e
0x0020339 76 61 6C 76 61 6C 7D 3B 74 66 74 70 62 6F 6F 74 20 24 7B 65 76 61 6C 76 61 6C 7D 20 24 7B 55 42 46 49 4E 41 4D 45 33 7D 20  valval};tftpboot ${evalval} ${UBFINAME3} 
0x0020362 26 26 20 73 65 74 65 6E 76 20 41 43 54 49 4D 41 47 45 20 24 7B 61 63 74 69 6D 67 7D 20 26 26 20 73 61 76 65 65 6E 76 00 55  && setenv ACTIMAGE ${actimg} && saveenv.U
0x002038B 42 46 49 33 52 41 4D 52 45 53 45 52 56 45 3D 30 78 38 30 30 30 30 30 00 75 70 64 61 74 65 5F 75 62 6F 6F 74 3D 61 63 74 69  BFI3RAMRESERVE=0x800000.update_uboot=acti
0x00203B4 6D 67 3D 30 20 26 26 20 69 6D 67 61 64 64 72 3D 30 78 34 38 30 30 30 30 30 30 20 26 26 20 69 6D 67 6E 61 6D 65 3D 75 2D 62  mg=0 && imgaddr=0x48000000 && imgname=u-b
0x00203DD 6F 6F 74 2E 62 69 6E 20 26 26 20 72 75 6E 20 75 70 64 61 74 65 00 65 72 61 73 65 5F 65 6E 76 3D 65 76 61 6C 20 24 7B 65 6E  oot.bin && run update.erase_env=eval ${en
0x0020406 76 70 61 72 74 73 69 7A 65 7D 20 2B 20 24 7B 65 6E 76 70 61 72 74 73 69 7A 65 7D 20 26 26 20 65 6E 76 62 6C 6F 63 6B 73 69  vpartsize} + ${envpartsize} && envblocksi
0x002042F 7A 65 3D 24 7B 65 76 61 6C 76 61 6C 7D 20 26 26 20 65 76 61 6C 20 30 78 34 38 30 30 30 30 30 30 20 2B 20 24 7B 75 62 6F 6F  ze=${evalval} && eval 0x48000000 + ${uboo
0x0020458 74 70 61 72 74 73 69 7A 65 7D 20 26 26 20 70 72 6F 74 65 63 74 20 6F 66 66 20 24 7B 65 76 61 6C 76 61 6C 7D 20 2B 24 65 6E  tpartsize} && protect off ${evalval} +$en
0x0020481 76 62 6C 6F 63 6B 73 69 7A 65 20 26 26 20 65 72 61 73 65 20 24 7B 65 76 61 6C 76 61 6C 7D 20 2B 24 65 6E 76 62 6C 6F 63 6B  vblocksize && erase ${evalval} +$envblock
0x00204AA 73 69 7A 65 20 26 26 20 70 72 6F 74 65 63 74 20 6F 6E 20 24 7B 65 76 61 6C 76 61 6C 7D 20 2B 24 65 6E 76 62 6C 6F 63 6B 73  size && protect on ${evalval} +$envblocks
0x00204D3 69 7A 65 00 6E 65 74 72 65 74 72 79 3D 6E 6F 00 62 6F 61 72 64 74 79 70 65 3D 74 6E 65 74 63 35 35 30 00 62 74 5F 73 63 72  ize.netretry=no.boardtype=tnetc550.bt_scr
0x00204FC 69 70 74 3D 67 70 69 6F 20 33 30 20 6F 75 74 20 30 20 33 30 3B 20 73 77 69 74 63 68 5F 69 6E 69 74 00 62 6F 6F 74 73 74 72  ipt=gpio 30 out 0 30; switch_init.bootstr
0x0020525 61 70 3D 6E 6F 00 73 74 64 69 6E 3D 73 65 72 69 61 6C 00 73 74 64 6F 75 74 3D 73 65 72 69 61 6C 00 73 74 64 65 72 72 3D 73  ap=no.stdin=serial.stdout=serial.stderr=s
0x002054E 65 72 69 61 6C 00 75 62 6F 6F 74 70 61 72 74 73 69 7A 65 3D 30 78 32 30 30 30 30 00 65 6E 76 70 61 72 74 73 69 7A 65 3D 30  erial.ubootpartsize=0x20000.envpartsize=0
0x0020577 78 31 30 30 30 30 00 55 42 46 49 41 44 44 52 31 3D 30 78 34 38 30 34 30 30 30 30 00 55 42 46 49 41 44 44 52 32 3D 30 78 34  x10000.UBFIADDR1=0x48040000.UBFIADDR2=0x4
0x00205A0 63 30 30 30 30 30 30 00 76 65 72 3D 55 2D 42 6F 6F 74 20 31 2E 32 2E 30 20 28 41 75 67 20 31 31 20 32 30 31 34 20 2D 20 31  c000000.ver=U-Boot 1.2.0 (Aug 11 2014 - 1
0x00205C9 30 3A 30 32 3A 30 38 29 0A 50 53 50 55 2D 42 6F 6F 74 28 42 42 55 29 20 31 2E 30 2E 31 36 2E 32 32 00 73 69 6C 65 6E 74 3D  0:02:08) PSPU-Boot(BBU) 1.0.16.22.silent=
0x00205F2 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  1........................................
0x002061B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .........................................
0x0020644 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .........................................
0x002066D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .........................................
0x0020696 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .........................................
0x00206BF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .........................................
```

Start	End	Length	Content
0x205AB	0x205F3	0x49	=U-Boot 1.2.0 (Aug 11 2014 - 10:02:08) PSPU-Boot(BBU) 1.0.16.22.silent=1.



```
(telnets
  (ports
    (0
      (port(23))
      (ssl_mode(none))
    )
    (1
      (port(8023))
      (ssl_mode(none))
    )
    (2
      (port(992))
      (ssl_mode(no_verify))
    )
  )
  (enabled(0))
  (local_access(0))
  (remote_access(0))
)
```

```
(telnets
  (ports
    (0
      (port(23))
      (ssl_mode(none))
    )
    (1
      (port(8023))
      (ssl_mode(none))
    )
    (2
      (port(992))
      (ssl_mode(no_verify))
    )
  )
  (enabled(1))
  (local_access(1))
  (remote_access(0))
)
```



Gestão

Administração remota

Permitir administração remota para ZON HUB é um risco de segurança.

Permitir acesso de entrada wan ao web-management

- ☐ A utilizar porta http primária (80)
- ☐ A utilizar porta http secundária (8080)
- ☐ A utilizar porta https primária (443)
- ☐ A utilizar porta https secundária (8443)

Permitir acesso de entrada wan ao servidor telnet

- ☑ A utilizar porta telnet primária (23)
- ☑ A utilizar porta telnet secundária (8023)
- ☑ A utilizar telnet seguro sobre porta ssl (992)

Ferramentas de diagnóstico

- ☐ Permitir pedidos de entrada do eco do icmp da wan (por exº pings e icmp traceroute queries)
- ☐ Permitir traceroute queries de entrada do udp da wan

Jungo.net (jnet)

☐ Activado

Jungo.net ACS URL: https://jrms.zon.pt/jnet_rg2.cgi

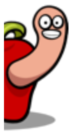
Página inicial do jungo.net: jrms.zon.pt

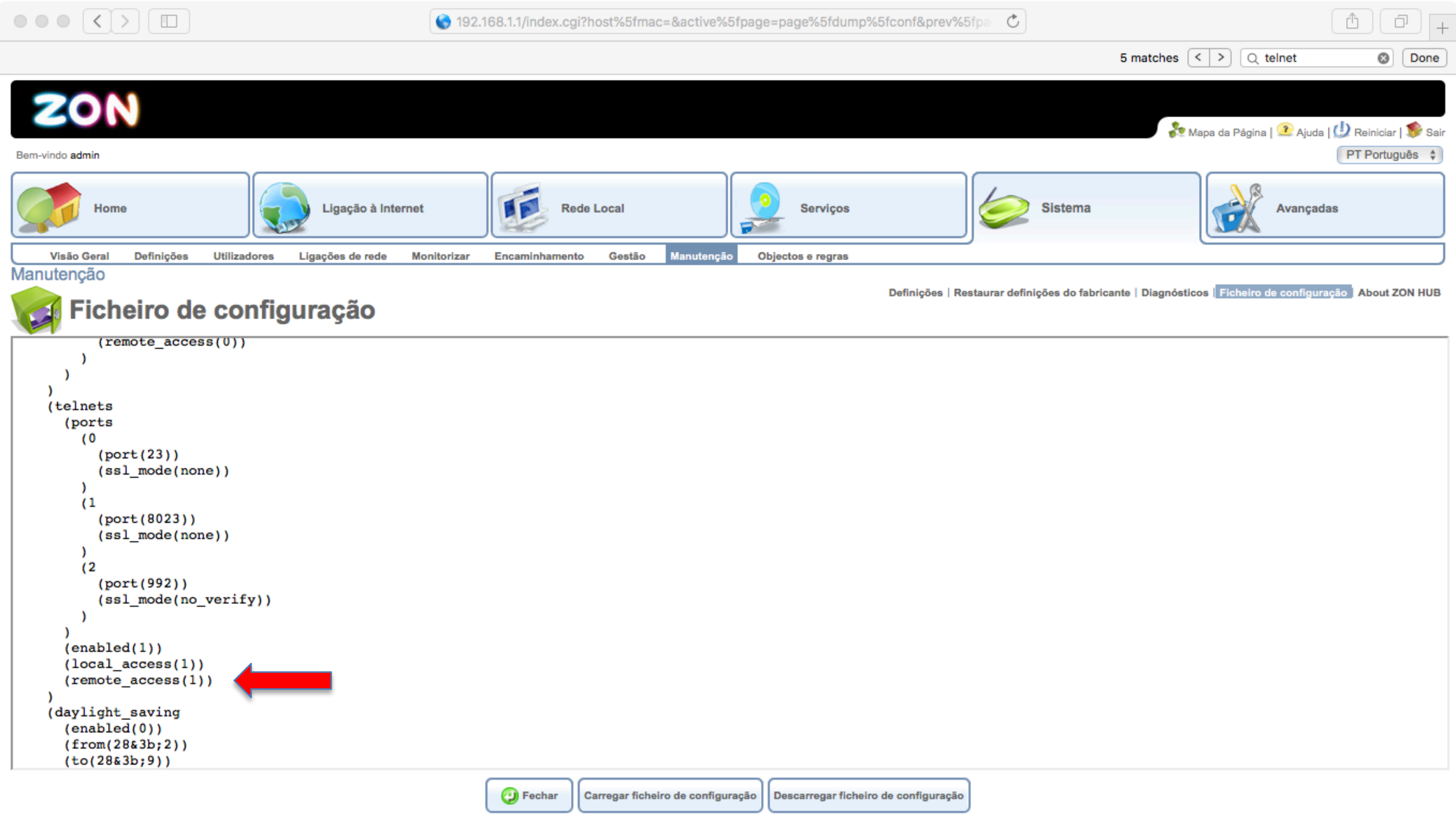
Portas jnet adicionais

☐ Permitir comandos jnet a partir de servidor de configuração remoto

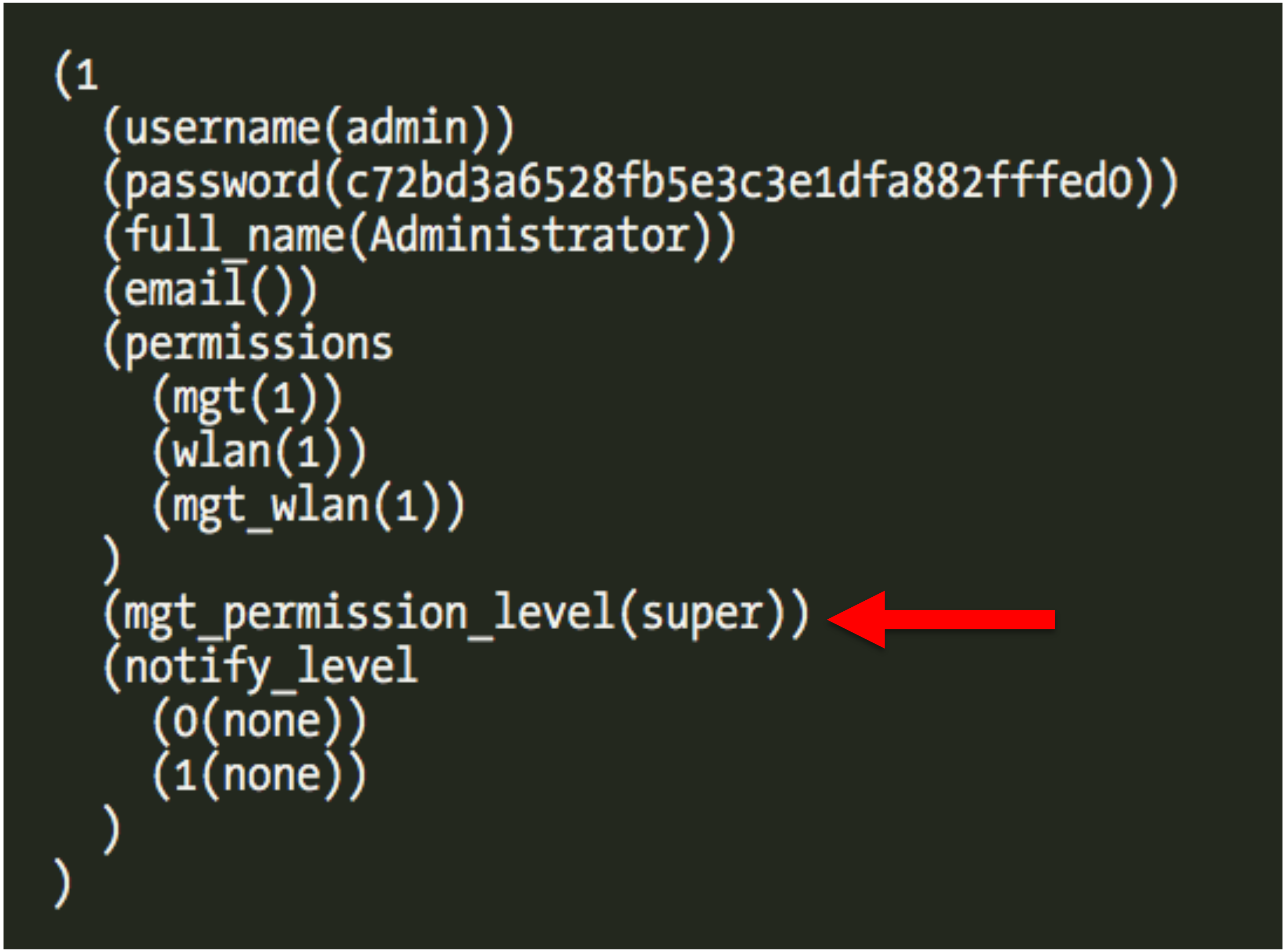
Servidor url de actualização remota: http://update.zon.pt/jungo/openrg/4.11.3.7/openrg-4.11.3.7-CVE30360_V2_ZON.rms

- ☐ Activar requisitos Jnet de entrada para a porta 7020
- ☐ Permitir acesso de entrada wan a jnet
- ☐ Activar pedidos jnet-ssl para a porta 7021
- ☐ Permitir acesso de entrada wan a jnet-ssl




192.168.1.1/index.cgi?host%5fmac=&active%5fpage=page%5fdump%5fconf&prev%5fpa
5 matches
telnet
Done
ZON
Mapa da Página | Ajuda | Reiniciar | Sair
Bem-vindo admin
PT Português
Home
Ligação à Internet
Rede Local
Serviços
Sistema
Avançadas
Visão Geral
Definições
Utilizadores
Ligações de rede
Monitorizar
Encaminhamento
Gestão
Manutenção
Objectos e regras
Manutenção
Definições | Restaurar definições do fabricante | Diagnósticos | Ficheiro de configuração | About ZON HUB
Ficheiro de configuração
Fechar
Carregar ficheiro de configuração
Descarregar ficheiro de configuração

```
          (remote_access(0))
        )
      )
    )
    (telnets
      (ports
        (0
          (port(23))
          (ssl_mode(none))
        )
        (1
          (port(8023))
          (ssl_mode(none))
        )
        (2
          (port(992))
          (ssl_mode(no_verify))
        )
      )
      (enabled(1))
      (local_access(1))
      (remote_access(1))
    )
    (daylight_saving
      (enabled(0))
      (from(28&3b;2))
      (to(28&3b;9))
```

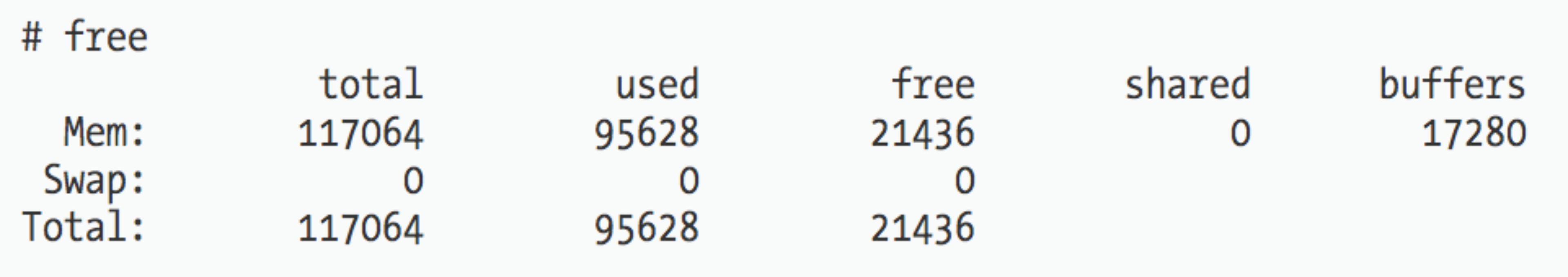


```
(1
  (username(admin))
  (password(c72bd3a6528fb5e3c3e1dfa882fffed0))
  (full_name(Administrator))
  (email())
  (permissions
    (mgt(1))
    (wlan(1))
    (mgt_wlan(1))
  )
  (mgt_permission_level(super))
  (notify_level
    (0(none))
    (1(none))
  )
)
```

DEBUGGER HOW-TO

How to attach a debugger

- Remote debugging session.
- **gdbserver** and **gdb** combo.
- Prebuilt or built from source.



```
# free
             total       used       free     shared    buffers
  Mem:      117064      95628      21436          0      17280
 Swap:           0          0          0
Total:      117064      95628      21436
```

How to attach a debugger

- We need a Puma5 toolchain.
- Usually the toolchains are published.
  - GPL had to be useful someday…
- Some Google-fu and luck required.

How to attach a debugger

- Someone already published it.
- For Motorola modems but it works anyway.
  - https://github.com/bmaia/cross-utils
  - https://github.com/bmaia/cross-utils/tree/master/armeb/puma5_toolchain

Welcome to HELL!

How to attach a debugger

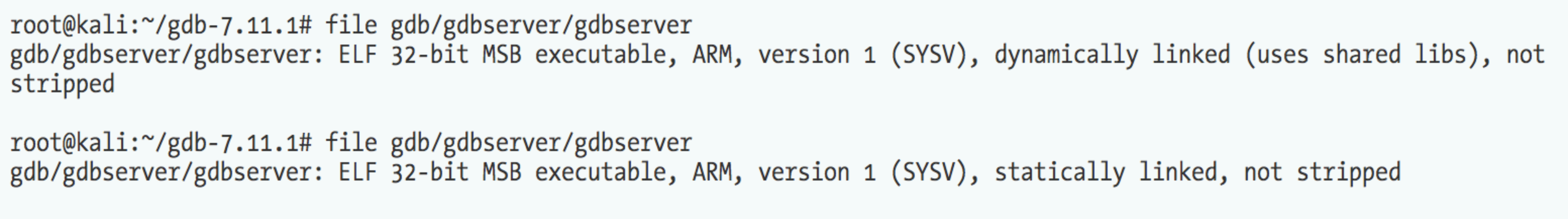
- Kali 1.1.0a.
- Trust the Internet and use **prebuilt** toolchain.
  - https://github.com/bmaia/cross-utils/raw/master/armeb/puma5_toolchain/armeb-linux.tar.xz
- GDB 7.11.1 is fine. Everything else, good luck!
- Static binary crashes, use dynamic.

How to attach a debugger

- Build your own toolchain.
- Ubuntu Server 9.04.
    - Need to fix apt sources.
- SSL/TLS deprecation.
    - Deprecation is all fun until everything blows up!
- Need to fix toolchain/buildroot scripts.

How to attach a debugger

- Result should be something like this:



```
root@kali:~/gdb-7.11.1# file gdb/gdbserver/gdbserver
gdb/gdbserver/gdbserver: ELF 32-bit MSB executable, ARM, version 1 (SYSV), dynamically linked (uses shared libs), not
stripped

root@kali:~/gdb-7.11.1# file gdb/gdbserver/gdbserver
gdb/gdbserver/gdbserver: ELF 32-bit MSB executable, ARM, version 1 (SYSV), statically linked, not stripped
```

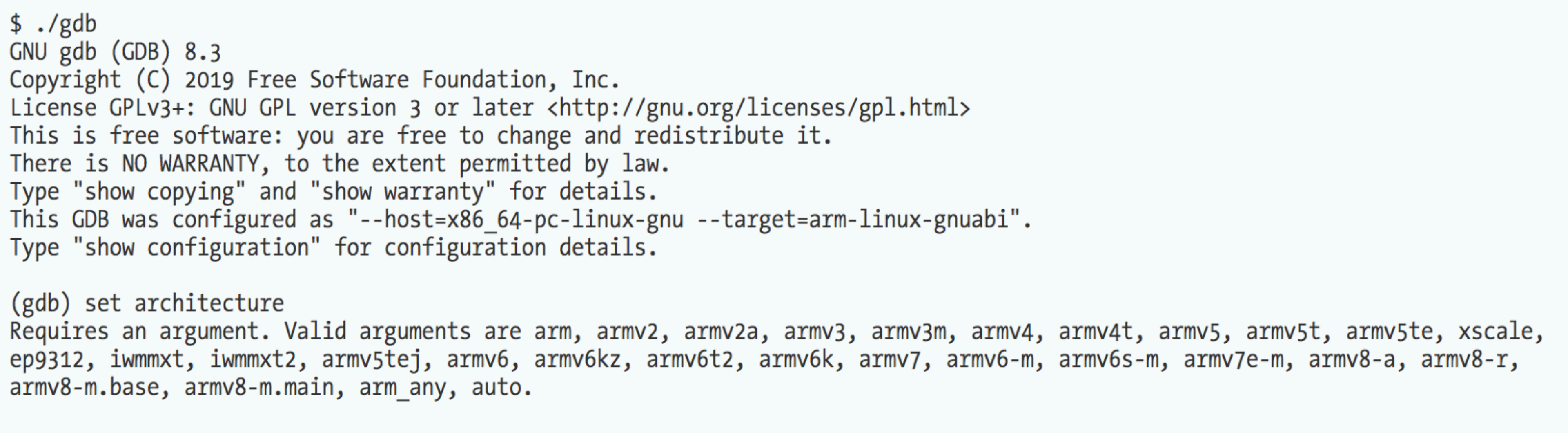
- Don’t forget to strip binaries to save mem.
  - armeb-linux-strip

How to attach a debugger

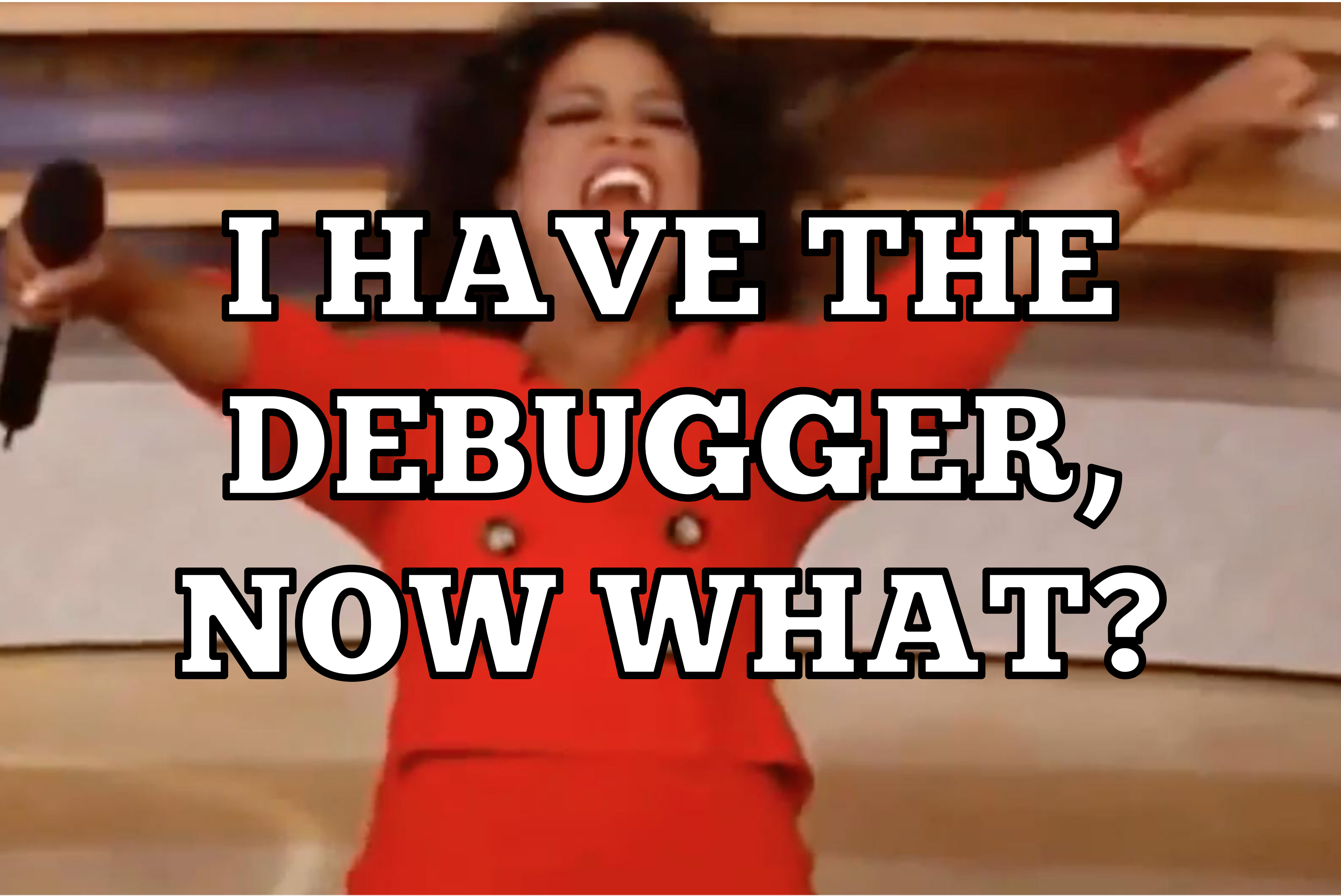
- We still need to compile host **gdb**.
- Latest 8.3 works fine.
- ./configure --host=x86_64-pc-linux-gnu --build=x86_64-pc-linux-gnu --target=arm-linux-gnuabi
- You can compile with multi-arch support.

How to attach a debugger

- Result should be something like this:



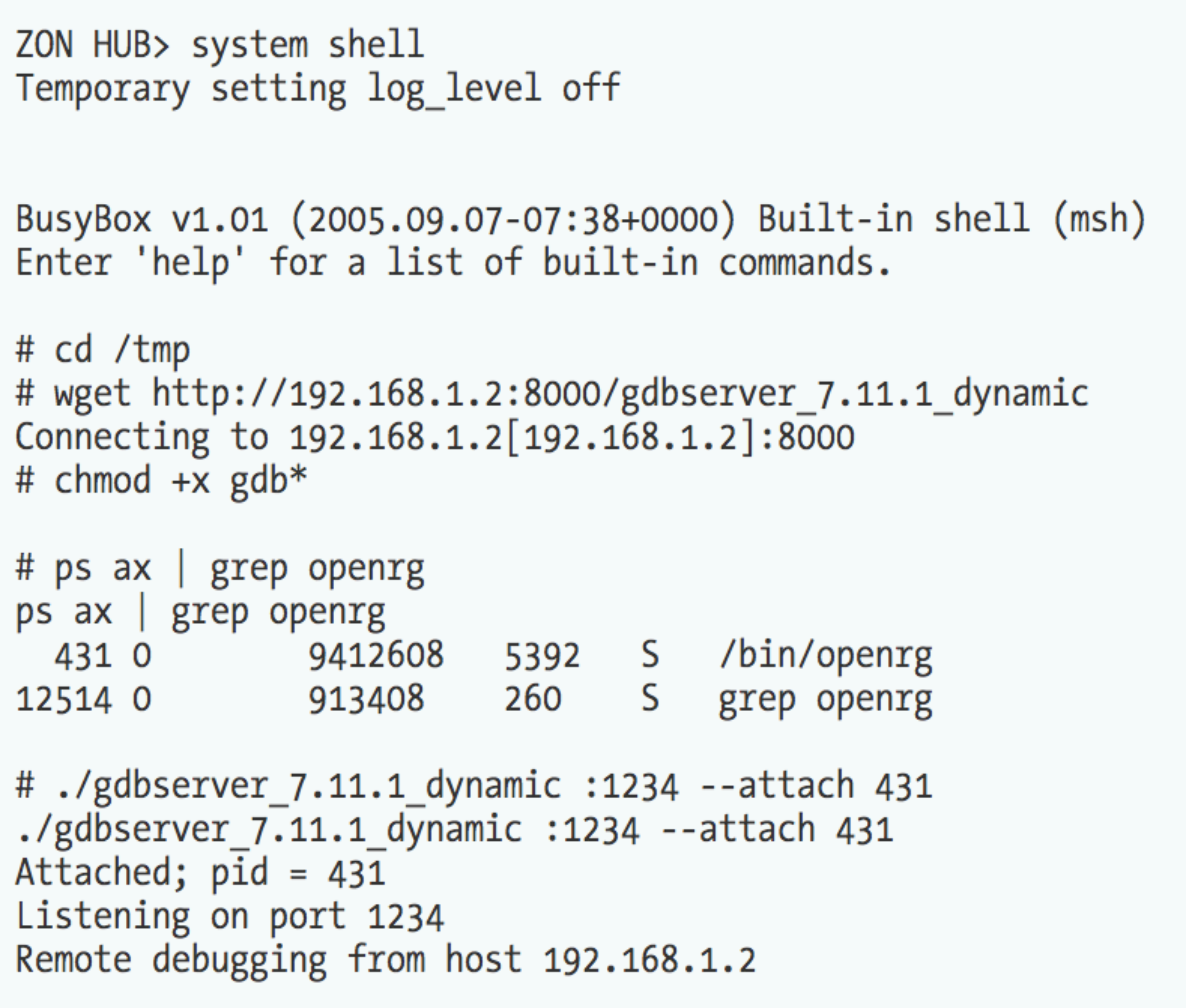
```
$ ./gdb
GNU gdb (GDB) 8.3
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-pc-linux-gnu --target=arm-linux-gnuabi".
Type "show configuration" for configuration details.

(gdb) set architecture
Requires an argument. Valid arguments are arm, armv2, armv2a, armv3, armv3m, armv4, armv4t, armv5, armv5t, armv5te, xscale,
ep9312, iwmmxt, iwmmxt2, armv5tej, armv6, armv6kz, armv6t2, armv6k, armv7, armv6-m, armv6s-m, armv7e-m, armv8-a, armv8-r,
armv8-m.base, armv8-m.main, arm_any, auto.
```



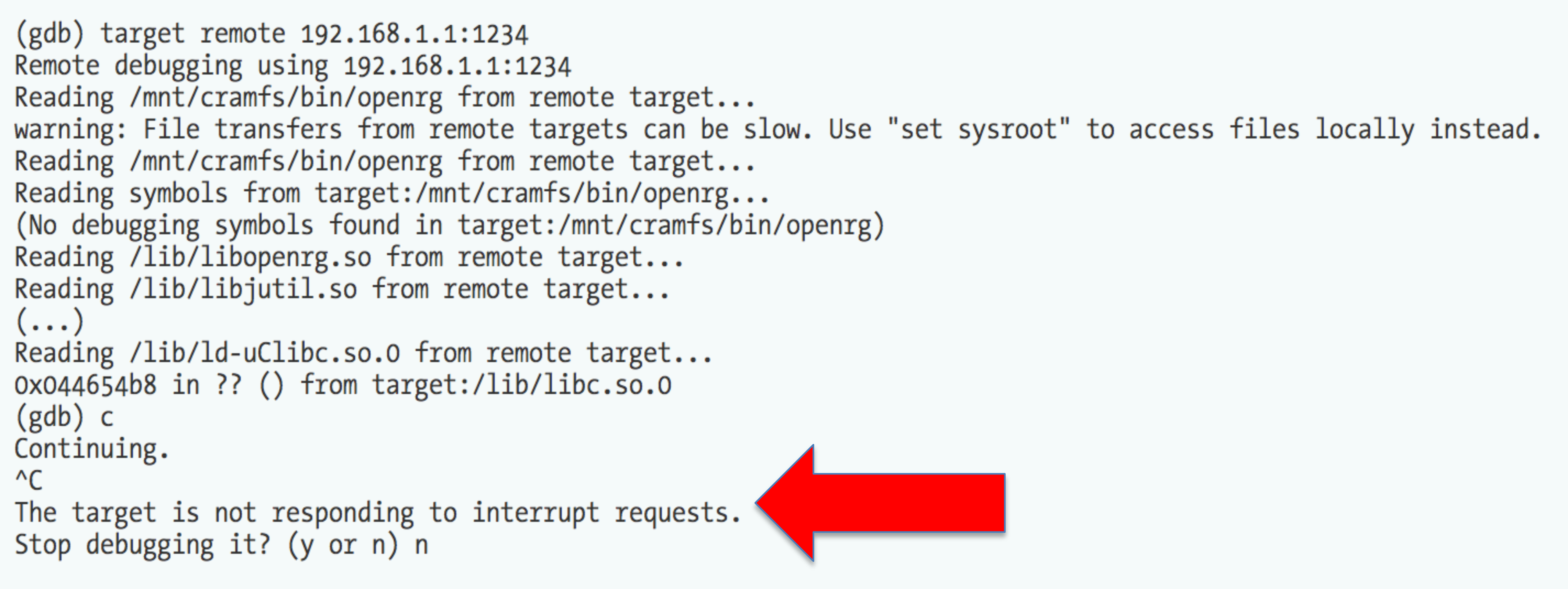
I HAVE THE DEBUGGER, NOW WHAT?

How to attach a debugger

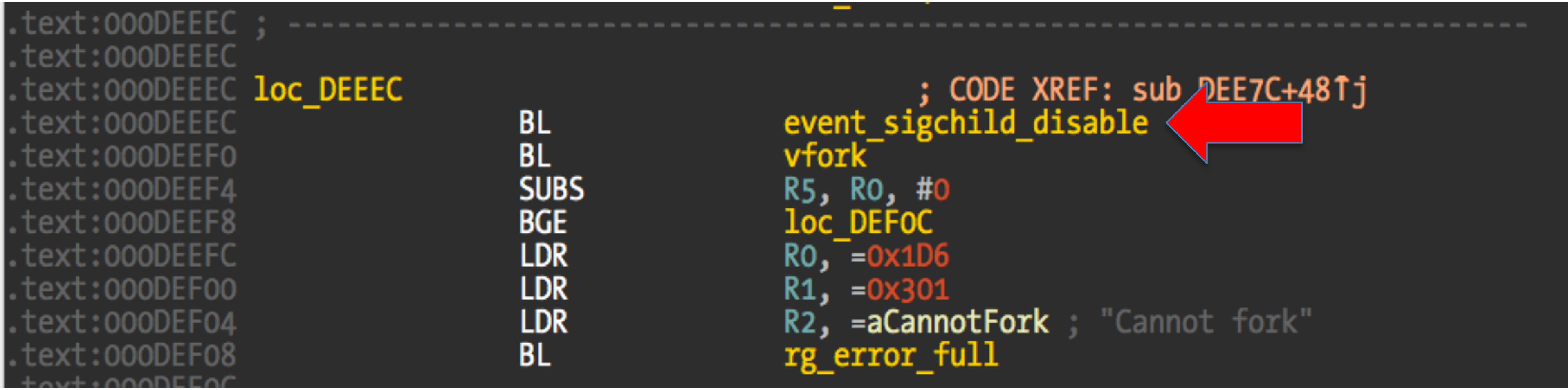
- Use SimpleHTTPServer or TFTP to transfer gdbserver binary.
- Attach to a process.
- Doesn’t really work ☹.
- Intercepted signals?



```
ZON HUB> system shell
Temporary setting log_level off


BusyBox v1.01 (2005.09.07-07:38+0000) Built-in shell (msh)
Enter 'help' for a list of built-in commands.

# cd /tmp
# wget http://192.168.1.2:8000/gdbserver_7.11.1_dynamic
Connecting to 192.168.1.2[192.168.1.2]:8000
# chmod +x gdb*

# ps ax | grep openrg
ps ax | grep openrg
  431 0        9412608   5392   S   /bin/openrg
12514 0        913408    260    S   grep openrg

# ./gdbserver_7.11.1_dynamic :1234 --attach 431
./gdbserver_7.11.1_dynamic :1234 --attach 431
Attached; pid = 431
Listening on port 1234
Remote debugging from host 192.168.1.2
```



```
(gdb) target remote 192.168.1.1:1234
Remote debugging using 192.168.1.1:1234
Reading /mnt/cramfs/bin/openrg from remote target...
warning: File transfers from remote targets can be slow. Use "set sysroot" to access files locally instead.
Reading /mnt/cramfs/bin/openrg from remote target...
Reading symbols from target:/mnt/cramfs/bin/openrg...
(No debugging symbols found in target:/mnt/cramfs/bin/openrg)
Reading /lib/libopenrg.so from remote target...
Reading /lib/libjutil.so from remote target...
(...)
Reading /lib/ld-uClibc.so.0 from remote target...
0x044654b8 in ?? () from target:/lib/libc.so.0
(gdb) c
Continuing.
^C
The target is not responding to interrupt requests.
Stop debugging it? (y or n) n
```



```
.text:000DEEEC ; ---------------------------------------------------------------------------
.text:000DEEEC
.text:000DEEEC loc_DEEEC                               ; CODE XREF: sub_DEE7C+48↑j
.text:000DEEEC                 BL              event_sigchild_disable
.text:000DEEF0                 BL              vfork
.text:000DEEF4                 SUBS            R5, R0, #0
.text:000DEEF8                 BGE             loc_DEF0C
.text:000DEEFC                 LDR             R0, =0x1D6
.text:000DEF00                 LDR             R1, =0x301
.text:000DEF04                 LDR             R2, =aCannotFork ; "Cannot fork"
.text:000DEF08                 BL              rg_error_full
```

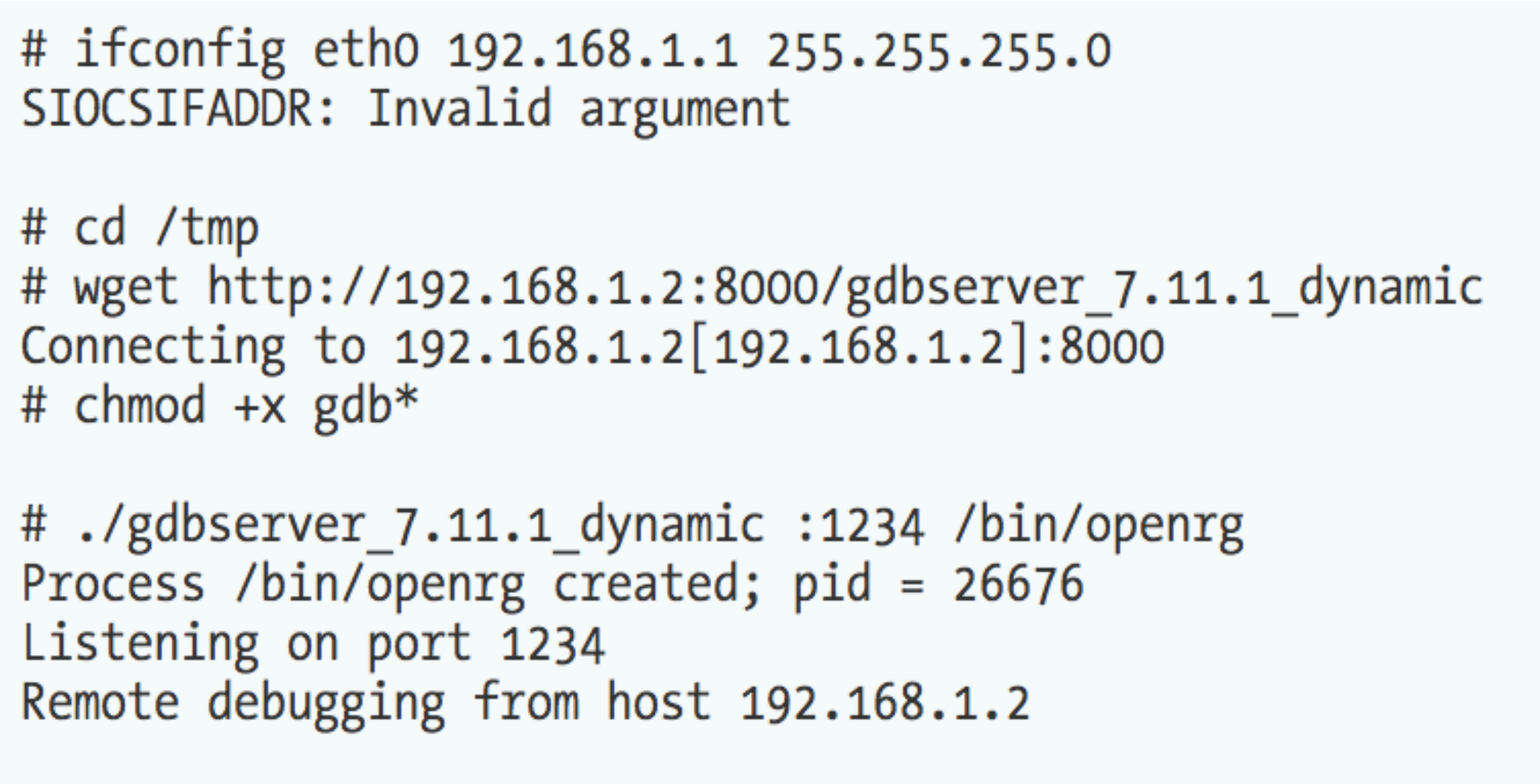
How to attach a debugger

- We can attach to processes.
- But we have no real control over them.
- There is an hidden trick in OpenRG developer manuals ☺.

```
ZON HUB> help system

Command Category system - Commands to control ZON HUB execution
todc                          Commands to update the todc task from docsis
die                           Exit from ZON HUB and return ret
ps                            Print ZON HUB's tasks
entity_close                  Close an entity
etask_list_dump               Dump back trace of all etasks
restore_factory_settings      Restore factory configuration
restore_home_admin_password   Restore Home Admin Password
reboot                        Reboot the system
delayed_reboot                Reboot the system asynchronously
ver                           Display version information
print_config                  Print compilation configuration. Search for
                              option if specified
exec                          Execute program
cat                           Print file contents to console
shell                         Spawn busybox shell in foreground
date                          Print the current UTC and local time
echo                          Echo arguments to console
exit                          Exit sub menu
help                          Show help for commands within this menu

Returned 0

ZON HUB> help system exit_and_shell
exit_and_shell   Exit from ZON HUB and open a shell on the serial console

Returned 0
```

How to attach a debugger

- openrg process is killed.
- Network interface goes down.
- No telnet anymore.
- We need to restore everything via serial console.



```
# ifconfig eth0 192.168.1.1 255.255.255.0
SIOCSIFADDR: Invalid argument

# cd /tmp
# wget http://192.168.1.2:8000/gdbserver_7.11.1_dynamic
Connecting to 192.168.1.2[192.168.1.2]:8000
# chmod +x gdb*

# ./gdbserver_7.11.1_dynamic :1234 /bin/openrg
Process /bin/openrg created; pid = 26676
Listening on port 1234
Remote debugging from host 192.168.1.2
```

```
(gdb) target remote 192.168.1.1:1234
Remote debugging using 192.168.1.1:1234
Reading /mnt/cramfs/bin/openrg from remote target...
warning: File transfers from remote targets can be slow. Use "set sysroot" to access files locally instead.
Reading /mnt/cramfs/bin/openrg from remote target...
Reading symbols from target:/mnt/cramfs/bin/openrg...
(No debugging symbols found in target:/mnt/cramfs/bin/openrg)
Reading /lib/ld-uClibc.so.0 from remote target...
Reading /lib/ld-uClibc.so.0 from remote target...
Reading symbols from target:/lib/ld-uClibc.so.0...
(No debugging symbols found in target:/lib/ld-uClibc.so.0)
0x04001010 in _start () from target:/lib/ld-uClibc.so.0
(gdb) c
Continuing.
(...)
[Detaching after vfork from child process 23005]
[Detaching after vfork from child process 23053]
[Detaching after fork from child process 23056]
[Detaching after vfork from child process 23109]
[Detaching after vfork from child process 23110]
[Detaching after vfork from child process 23112]
^C
Program received signal SIGINT, Interrupt.
0x044654b8 in ?? () from target:/lib/libc.so.0
(gdb) c
Continuing.
```

How to attach a debugger

- Now we have full control of **openrg** process.
- Debugger interrupts work.
- We can insert breakpoints.
- And other services are active again.

Reverse engineering!

Reverse engineering

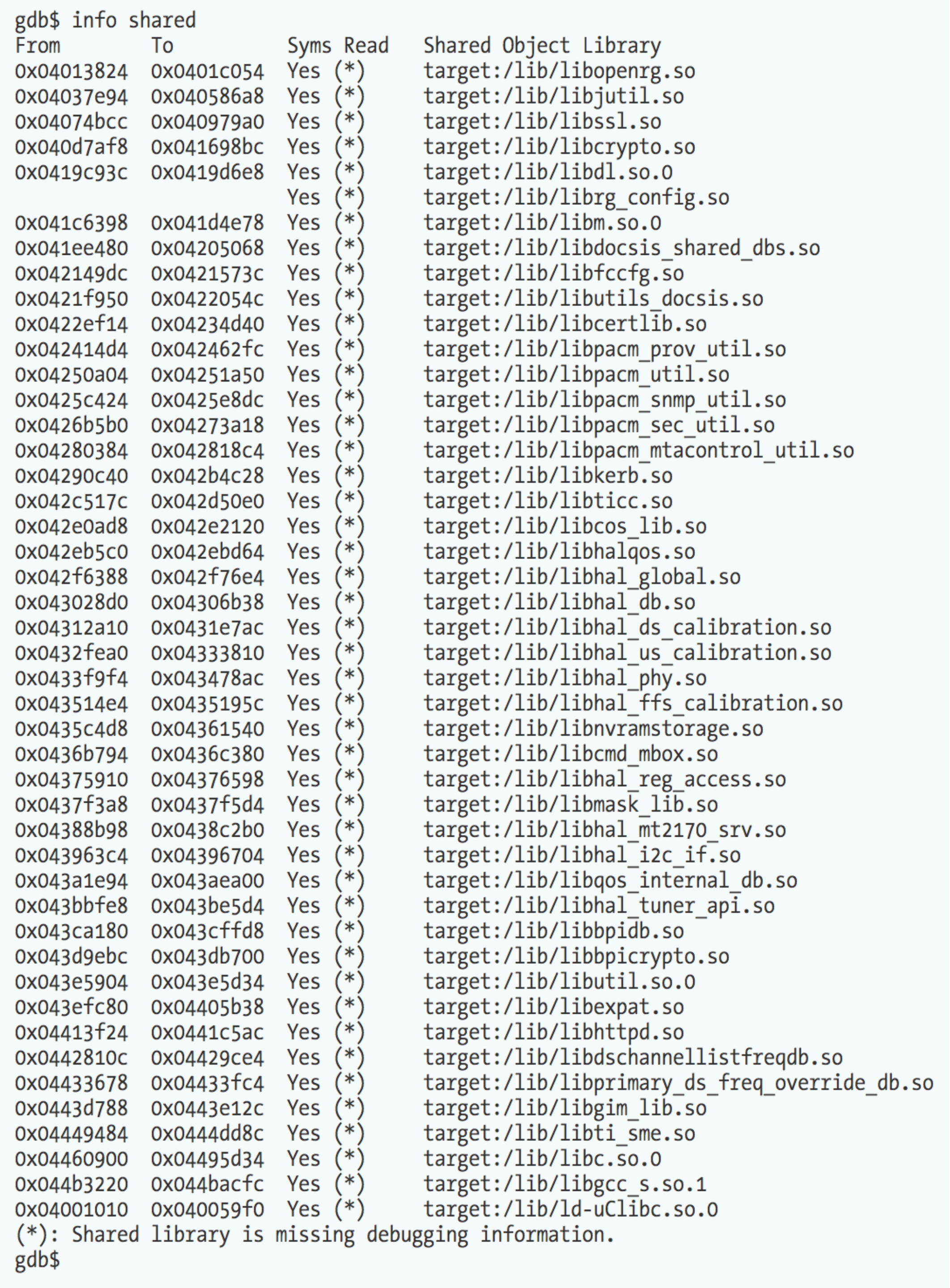
- It’s not a linear process.
- More like of chaotic and fractal nature.
- Lots of trial and error.
- Experience plays an important role.
  - Practice makes perfect.

Reverse engineering

- I will try to present some kind of ordered process.
- Many things I don't even remember how I found them ☺.
- More art than science.

Reverse engineering

- Main target is **openrg** binary.
- 32 bit ARM, dynamically linked, stripped, big endian.
- Decent size, around 7k functions.
- Linked against 60 libraries.



```
gdb$ info shared
From        To          Syms Read   Shared Object Library
0x04013824  0x0401c054  Yes (*)     target:/lib/libopenrg.so
0x04037e94  0x040586a8  Yes (*)     target:/lib/libjutil.so
0x04074bcc  0x040979a0  Yes (*)     target:/lib/libssl.so
0x040d7af8  0x041698bc  Yes (*)     target:/lib/libcrypto.so
0x0419c93c  0x0419d6e8  Yes (*)     target:/lib/libdl.so.0
                        Yes (*)     target:/lib/librg_config.so
0x041c6398  0x041d4e78  Yes (*)     target:/lib/libm.so.0
0x041ee480  0x04205068  Yes (*)     target:/lib/libdocsis_shared_dbs.so
0x042149dc  0x0421573c  Yes (*)     target:/lib/libfccfg.so
0x0421f950  0x0422054c  Yes (*)     target:/lib/libutils_docsis.so
0x0422ef14  0x04234d40  Yes (*)     target:/lib/libcertlib.so
0x042414d4  0x042462fc  Yes (*)     target:/lib/libpacm_prov_util.so
0x04250a04  0x04251a50  Yes (*)     target:/lib/libpacm_util.so
0x0425c424  0x0425e8dc  Yes (*)     target:/lib/libpacm_snmp_util.so
0x0426b5b0  0x04273a18  Yes (*)     target:/lib/libpacm_sec_util.so
0x04280384  0x042818c4  Yes (*)     target:/lib/libpacm_mtacontrol_util.so
0x04290c40  0x042b4c28  Yes (*)     target:/lib/libkerb.so
0x042c517c  0x042d50e0  Yes (*)     target:/lib/libticc.so
0x042e0ad8  0x042e2120  Yes (*)     target:/lib/libcos_lib.so
0x042eb5c0  0x042ebd64  Yes (*)     target:/lib/libhalqos.so
0x042f6388  0x042f76e4  Yes (*)     target:/lib/libhal_global.so
0x043028d0  0x04306b38  Yes (*)     target:/lib/libhal_db.so
0x04312a10  0x0431e7ac  Yes (*)     target:/lib/libhal_ds_calibration.so
0x0432fea0  0x04333810  Yes (*)     target:/lib/libhal_us_calibration.so
0x0433f9f4  0x043478ac  Yes (*)     target:/lib/libhal_phy.so
0x043514e4  0x0435195c  Yes (*)     target:/lib/libhal_ffs_calibration.so
0x0435c4d8  0x04361540  Yes (*)     target:/lib/libnvramstorage.so
0x0436b794  0x0436c380  Yes (*)     target:/lib/libcmd_mbox.so
0x04375910  0x04376598  Yes (*)     target:/lib/libhal_reg_access.so
0x0437f3a8  0x0437f5d4  Yes (*)     target:/lib/libmask_lib.so
0x04388b98  0x0438c2b0  Yes (*)     target:/lib/libhal_mt2170_srv.so
0x043963c4  0x04396704  Yes (*)     target:/lib/libhal_i2c_if.so
0x043a1e94  0x043aea00  Yes (*)     target:/lib/libqos_internal_db.so
0x043bbfe8  0x043be5d4  Yes (*)     target:/lib/libhal_tuner_api.so
0x043ca180  0x043cffd8  Yes (*)     target:/lib/libbpidb.so
0x043d9ebc  0x043db700  Yes (*)     target:/lib/libbpicrypto.so
0x043e5904  0x043e5d34  Yes (*)     target:/lib/libutil.so.0
0x043efc80  0x04405b38  Yes (*)     target:/lib/libexpat.so
0x04413f24  0x0441c5ac  Yes (*)     target:/lib/libhttpd.so
0x0442810c  0x04429ce4  Yes (*)     target:/lib/libdschannellistfreqdb.so
0x04433678  0x04433fc4  Yes (*)     target:/lib/libprimary_ds_freq_override_db.so
0x0443d788  0x0443e12c  Yes (*)     target:/lib/libgim_lib.so
0x04449484  0x0444dd8c  Yes (*)     target:/lib/libti_sme.so
0x04460900  0x04495d34  Yes (*)     target:/lib/libc.so.0
0x044b3220  0x044bacfc  Yes (*)     target:/lib/libgcc_s.so.1
0x04001010  0x040059f0  Yes (*)     target:/lib/ld-uClibc.so.0
(*): Shared library is missing debugging information.
gdb$
```

WE NEED TO HAVE A GOAL!

Reverse engineering

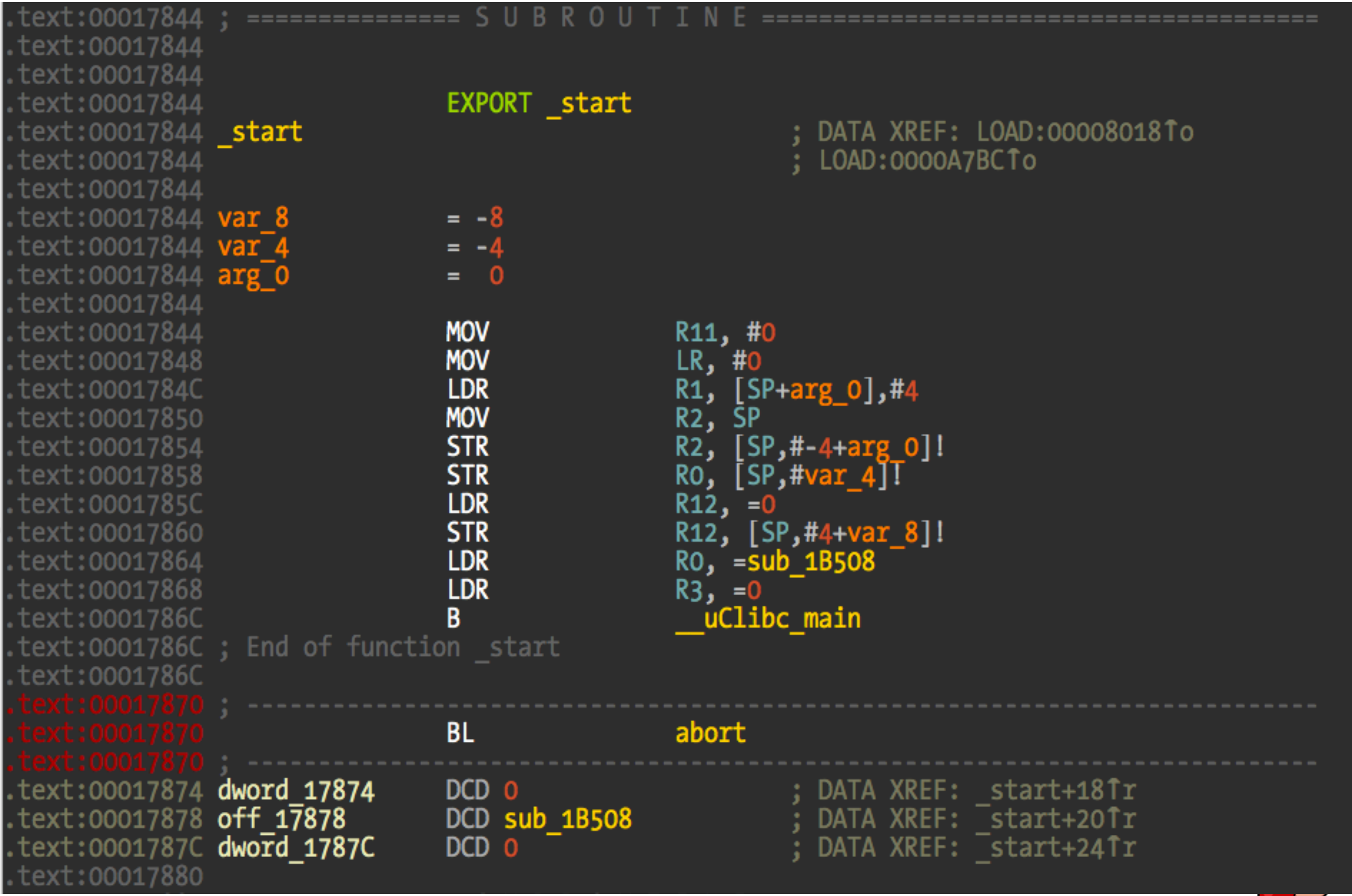
- You need a goal to kickstart the process.
- Otherwise very easy to get lost and/or frustrated.
- My initial goal was to find the default passwords.

ATTACK PLAN

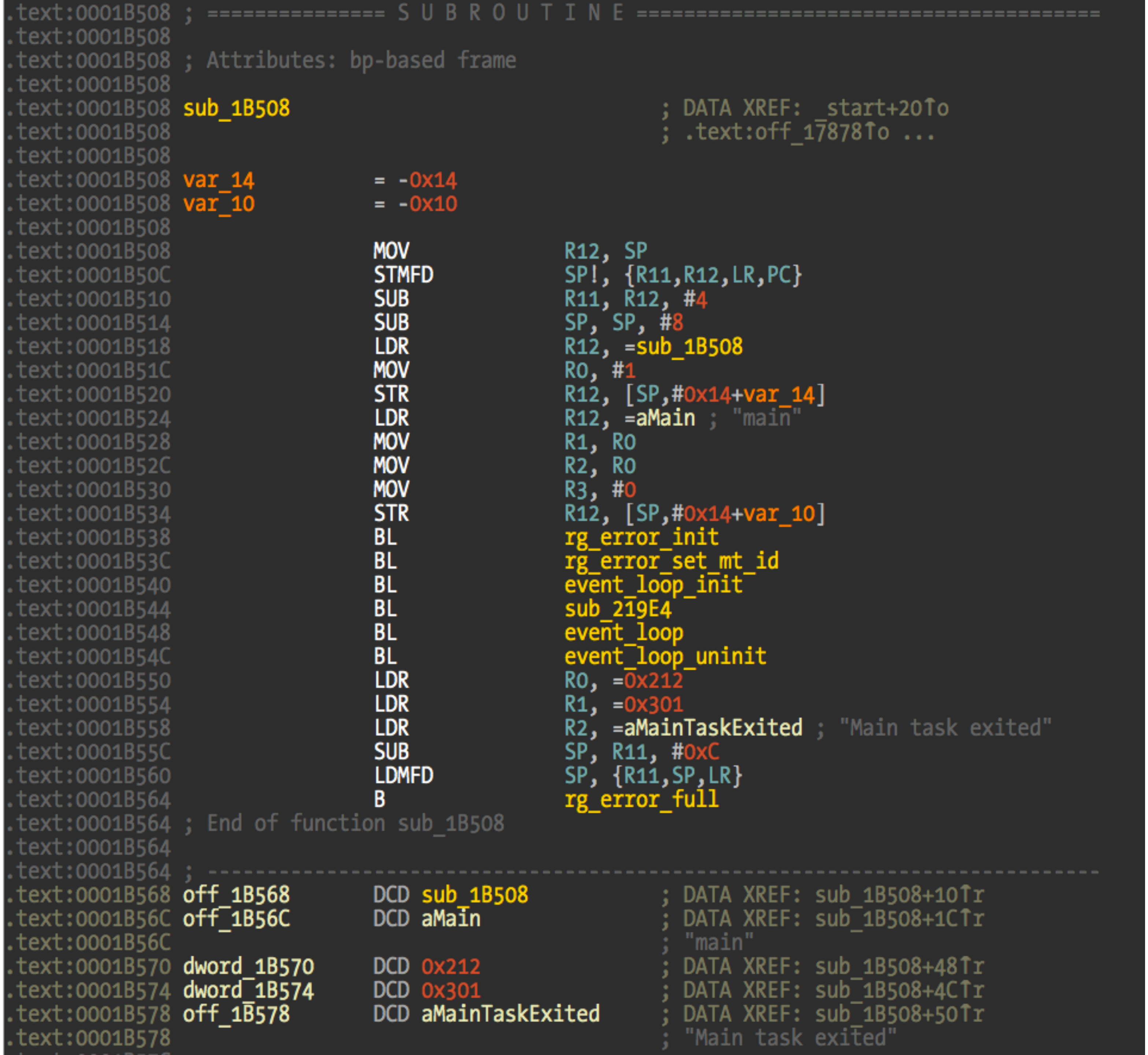
NETH.
Brussels
BELGIUM
Rhine R.
GER
eims
Paris
LORRAINE
Marne R.
Seine R.
FRANCE
Allied
wers
Neutral
nations
Central
Powers

Reverse engineering

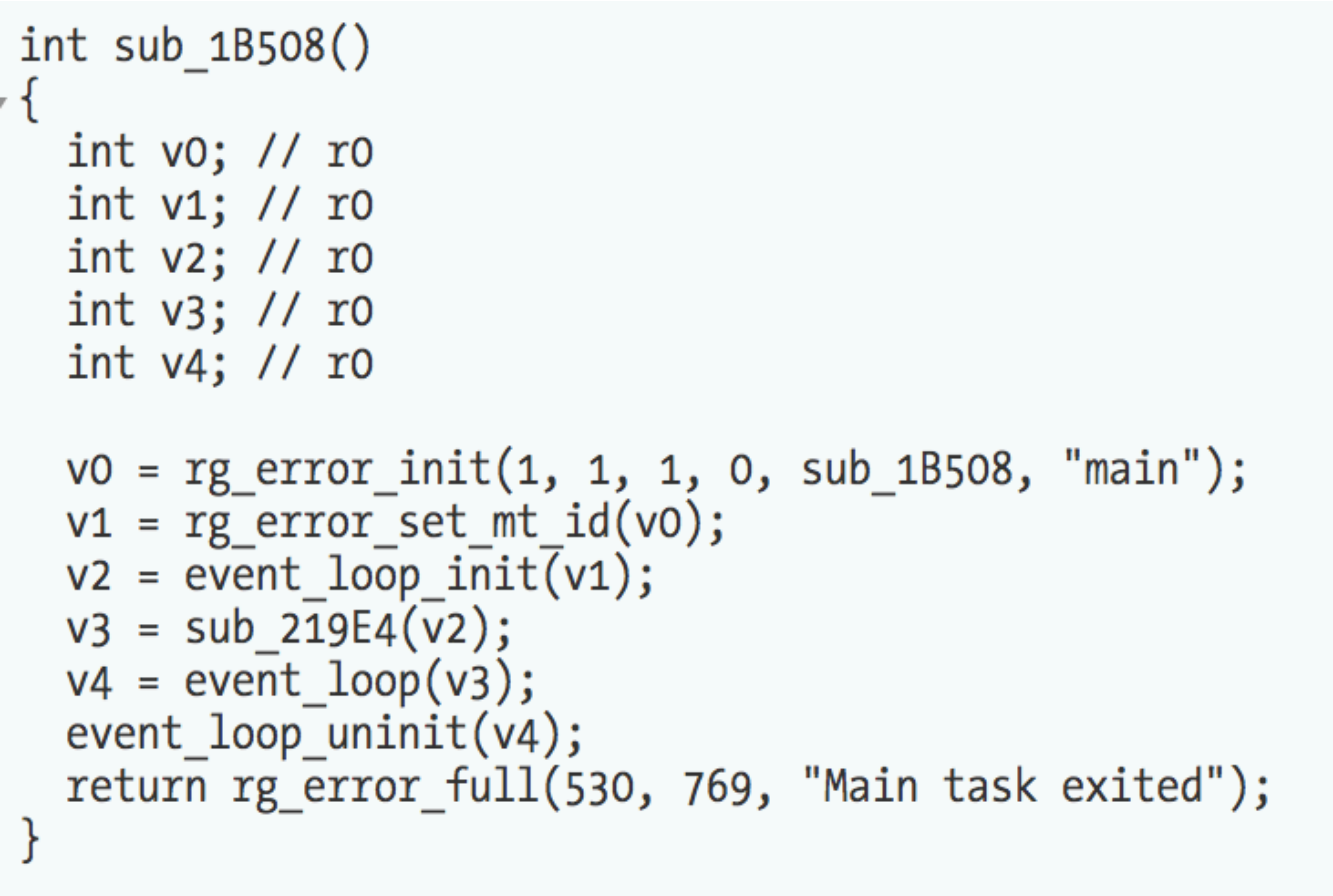
- Poke around with strings.
  - Lots of Portuguese text. Translation files suck.
- Load binary into IDA.
  - Maybe Ghidra: Java + NSA = too much for me.
- main() as starting point.



```
.text:00017844 ; =============== S U B R O U T I N E =======================================
.text:00017844
.text:00017844
.text:00017844                 EXPORT _start
.text:00017844 _start                                  ; DATA XREF: LOAD:00008018↑o
.text:00017844                                         ; LOAD:0000A7BC↑o
.text:00017844
.text:00017844 var_8           = -8
.text:00017844 var_4           = -4
.text:00017844 arg_0           =  0
.text:00017844
.text:00017844                 MOV             R11, #0
.text:00017848                 MOV             LR, #0
.text:0001784C                 LDR             R1, [SP+arg_0],#4
.text:00017850                 MOV             R2, SP
.text:00017854                 STR             R2, [SP,#-4+arg_0]!
.text:00017858                 STR             R0, [SP,#var_4]!
.text:0001785C                 LDR             R12, =0
.text:00017860                 STR             R12, [SP,#4+var_8]!
.text:00017864                 LDR             R0, =sub_1B508
.text:00017868                 LDR             R3, =0
.text:0001786C                 B               __uClibc_main
.text:0001786C ; End of function _start
.text:0001786C
.text:00017870 ; ---------------------------------------------------------------------------
.text:00017870                 BL              abort
.text:00017870 ; ---------------------------------------------------------------------------
.text:00017874 dword_17874     DCD 0                   ; DATA XREF: _start+18↑r
.text:00017878 off_17878       DCD sub_1B508           ; DATA XREF: _start+20↑r
.text:0001787C dword_1787C     DCD 0                   ; DATA XREF: _start+24↑r
.text:00017880
```



```
.text:0001B508 ; =============== S U B R O U T I N E =======================================
.text:0001B508
.text:0001B508 ; Attributes: bp-based frame
.text:0001B508
.text:0001B508 sub_1B508                               ; DATA XREF: _start+20↑o
.text:0001B508                                         ; .text:off_17878↑o ...
.text:0001B508
.text:0001B508 var_14          = -0x14
.text:0001B508 var_10          = -0x10
.text:0001B508
.text:0001B508                 MOV             R12, SP
.text:0001B50C                 STMFD           SP!, {R11,R12,LR,PC}
.text:0001B510                 SUB             R11, R12, #4
.text:0001B514                 SUB             SP, SP, #8
.text:0001B518                 LDR             R12, =sub_1B508
.text:0001B51C                 MOV             R0, #1
.text:0001B520                 STR             R12, [SP,#0x14+var_14]
.text:0001B524                 LDR             R12, =aMain ; "main"
.text:0001B528                 MOV             R1, R0
.text:0001B52C                 MOV             R2, R0
.text:0001B530                 MOV             R3, #0
.text:0001B534                 STR             R12, [SP,#0x14+var_10]
.text:0001B538                 BL              rg_error_init
.text:0001B53C                 BL              rg_error_set_mt_id
.text:0001B540                 BL              event_loop_init
.text:0001B544                 BL              sub_219E4
.text:0001B548                 BL              event_loop
.text:0001B54C                 BL              event_loop_uninit
.text:0001B550                 LDR             R0, =0x212
.text:0001B554                 LDR             R1, =0x301
.text:0001B558                 LDR             R2, =aMainTaskExited ; "Main task exited"
.text:0001B55C                 SUB             SP, R11, #0xC
.text:0001B560                 LDMFD           SP, {R11,SP,LR}
.text:0001B564                 B               rg_error_full
.text:0001B564 ; End of function sub_1B508
.text:0001B564
.text:0001B564 ; ---------------------------------------------------------------------------
.text:0001B568 off_1B568       DCD sub_1B508           ; DATA XREF: sub_1B508+10↑r
.text:0001B56C off_1B56C       DCD aMain               ; DATA XREF: sub_1B508+1C↑r
.text:0001B56C                                         ; "main"
.text:0001B570 dword_1B570     DCD 0x212               ; DATA XREF: sub_1B508+48↑r
.text:0001B574 dword_1B574     DCD 0x301               ; DATA XREF: sub_1B508+4C↑r
.text:0001B578 off_1B578       DCD aMainTaskExited     ; DATA XREF: sub_1B508+50↑r
.text:0001B578                                         ; "Main task exited"
```



```
int sub_1B508()
{
  int v0; // r0
  int v1; // r0
  int v2; // r0
  int v3; // r0
  int v4; // r0

  v0 = rg_error_init(1, 1, 1, 0, sub_1B508, "main");
  v1 = rg_error_set_mt_id(v0);
  v2 = event_loop_init(v1);
  v3 = sub_219E4(v2);
  v4 = event_loop(v3);
  event_loop_uninit(v4);
  return rg_error_full(530, 769, "Main task exited");
}
```

Reverse engineering

- Function **sub_1B508** is **main**.
- Function **sub_219E4** appears to initialize a bunch of stuff.
- Event driven loop.
- Not much to (easily) trace from **main**.

ATTACK PLAN
ACT II

Reverse engineering

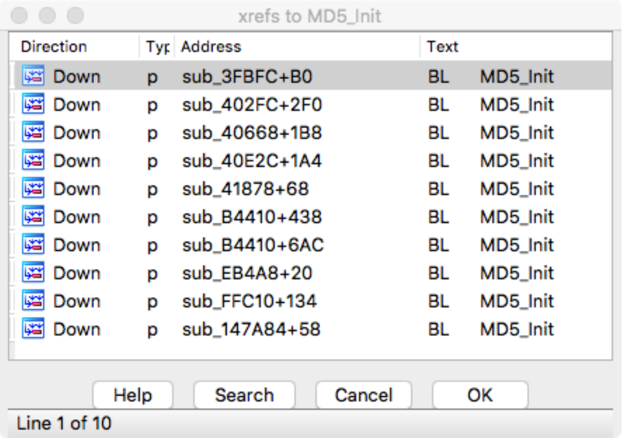
- Originally I had no debugger access.
- Too boring to browse every call from **main**.
- Would need to find events registration function and event handlers.
- Better shortcuts required.

Reverse engineering

- Passwords appear to be **MD5.**

```
(username(admin))
(password(a609bd56d33840a1f314793459ea7fa9))
(full_name(Administrator))
```

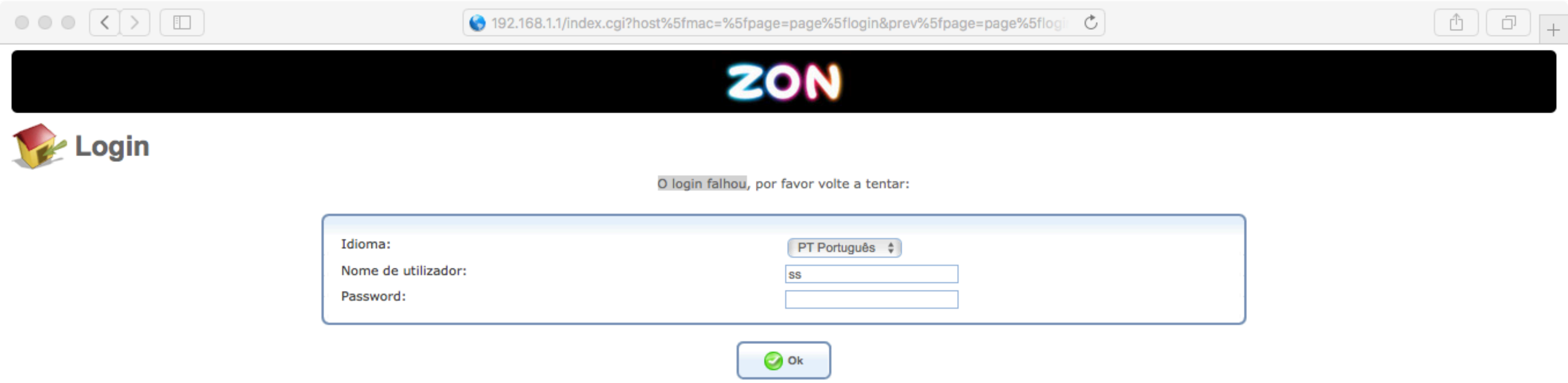
- But "too many" calls to **MD5_Init.**



Direction	Typ	Address	Text
Down	p	sub_3FBFC+B0	BL MD5_Init
Down	p	sub_402FC+2F0	BL MD5_Init
Down	p	sub_40668+1B8	BL MD5_Init
Down	p	sub_40E2C+1A4	BL MD5_Init
Down	p	sub_41878+68	BL MD5_Init
Down	p	sub_B4410+438	BL MD5_Init
Down	p	sub_B4410+6AC	BL MD5_Init
Down	p	sub_EB4A8+20	BL MD5_Init
Down	p	sub_FFC10+134	BL MD5_Init
Down	p	sub_147A84+58	BL MD5_Init

Reverse engineering

- String references can be helpful.



192.168.1.1/index.cgi?host%5fmac=%5fpage=page%5flogin&prev%5fpage=page%5flogi

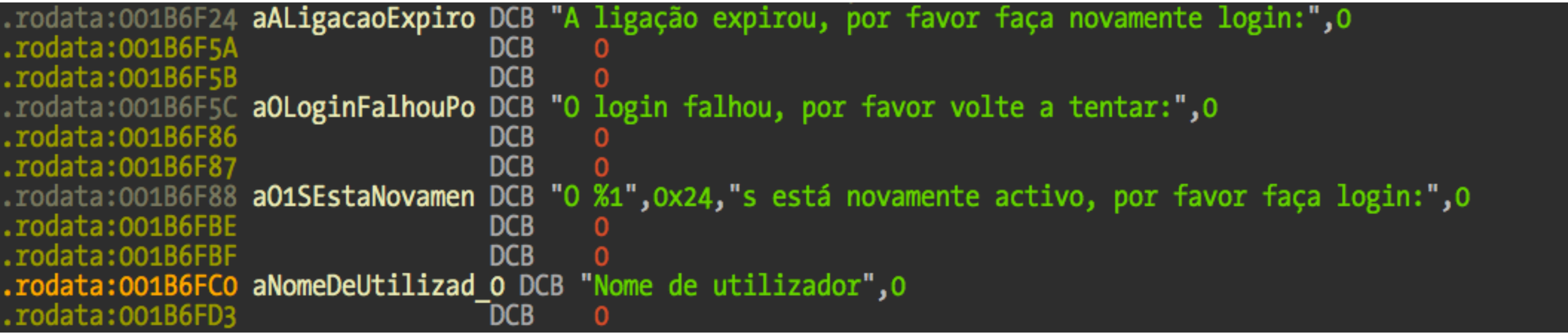
Login

O login falhou, por favor volte a tentar:

Idioma:	PT Português
Nome de utilizador:	ss
Password:	

Reverse engineering

- No such luck, there are no cross references to the text strings we want.



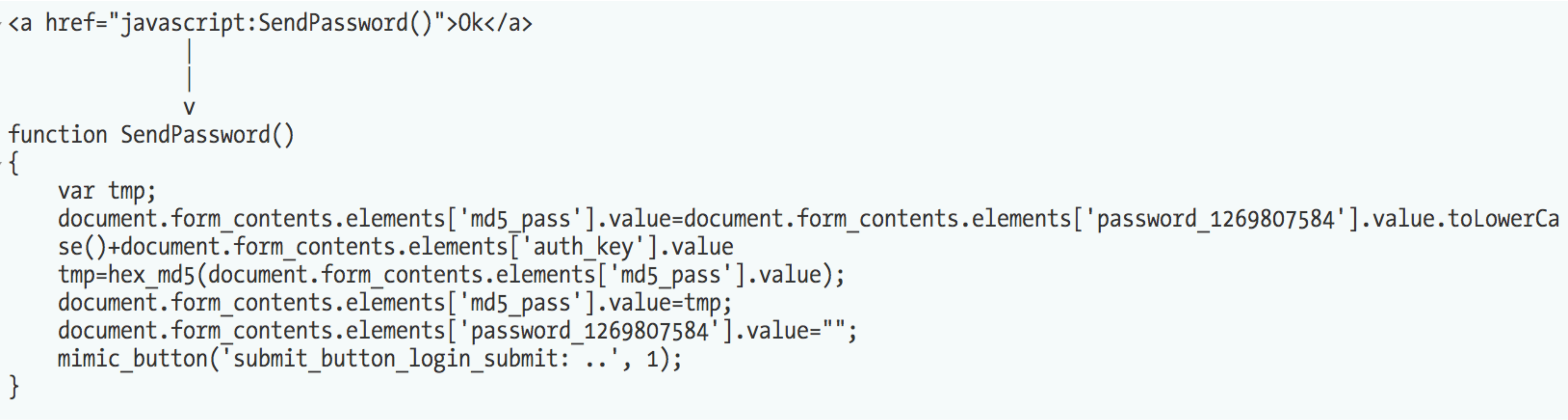
```
.rodata:001B6F24 aALigacaoExpiro DCB "A ligação expirou, por favor faça novamente login:",0
.rodata:001B6F5A                 DCB    0
.rodata:001B6F5B                 DCB    0
.rodata:001B6F5C aOLoginFalhouPo DCB "O login falhou, por favor volte a tentar:",0
.rodata:001B6F86                 DCB    0
.rodata:001B6F87                 DCB    0
.rodata:001B6F88 a01SEstaNovamen DCB "O %1",0x24,"s está novamente activo, por favor faça login:",0
.rodata:001B6FBE                 DCB    0
.rodata:001B6FBF                 DCB    0
.rodata:001B6FC0 aNomeDeUtilizad_0 DCB "Nome de utilizador",0
.rodata:001B6FD3                 DCB    0
```

Reverse engineering

- We talk to a web interface.
- That uses CGIs.
- There must be some code reading our form submission.
- Check login page source.

Reverse engineering

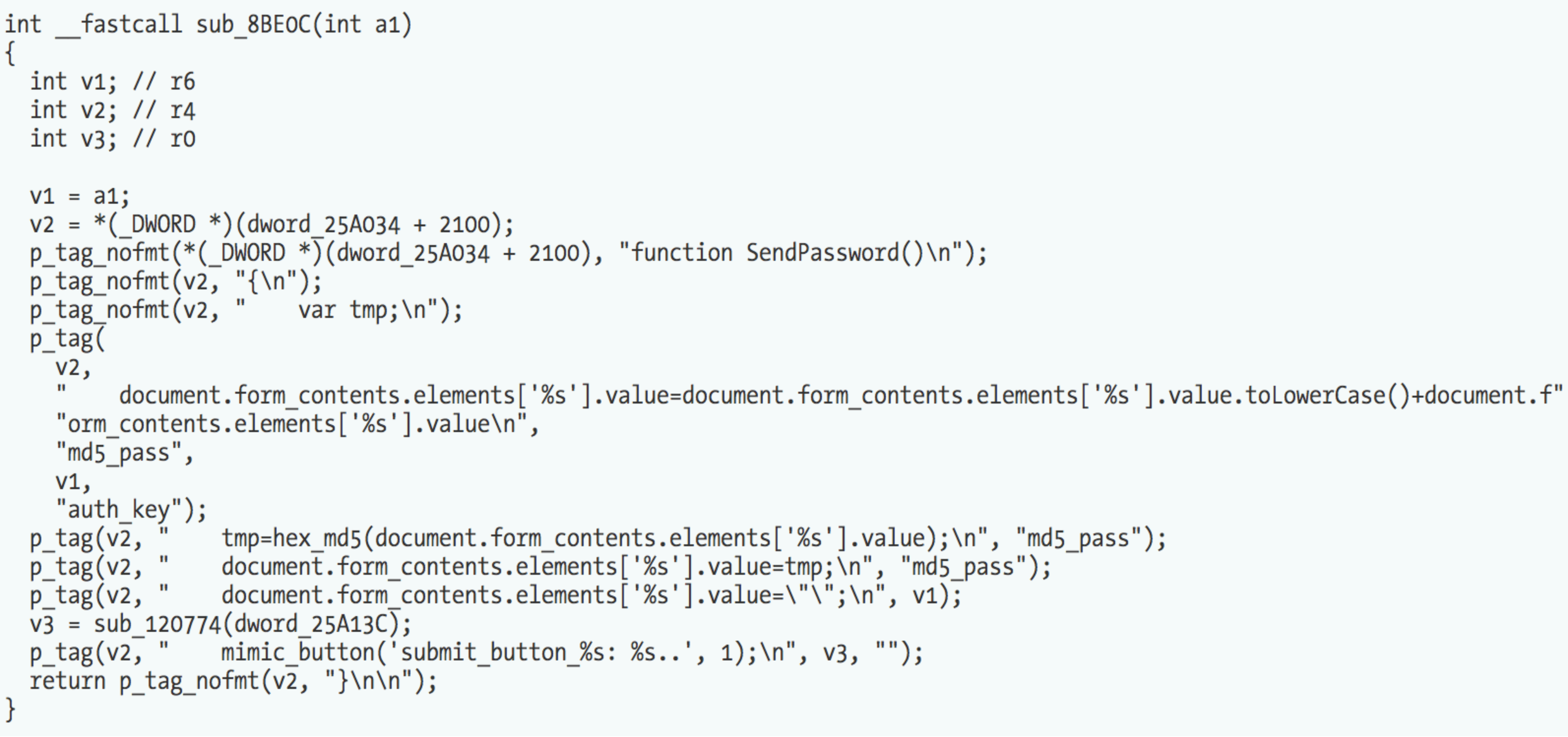
- Submit button uses JS to submit form contents.
- Search xrefs to “**SendPassword()**”



```
<a href="javascript:SendPassword()">Ok</a>
                |
                |
                v
function SendPassword()
{
    var tmp;
    document.form_contents.elements['md5_pass'].value=document.form_contents.elements['password_1269807584'].value.toLowerCa
    se()+document.form_contents.elements['auth_key'].value
    tmp=hex_md5(document.form_contents.elements['md5_pass'].value);
    document.form_contents.elements['md5_pass'].value=tmp;
    document.form_contents.elements['password_1269807584'].value="";
    mimic_button('submit_button_login_submit: ..', 1);
}
```

Reverse engineering

- There are two hits:
  - “function SendPassword()”
  - “SendPassword()”
- First appears to be in a function that just formats the HTML output.



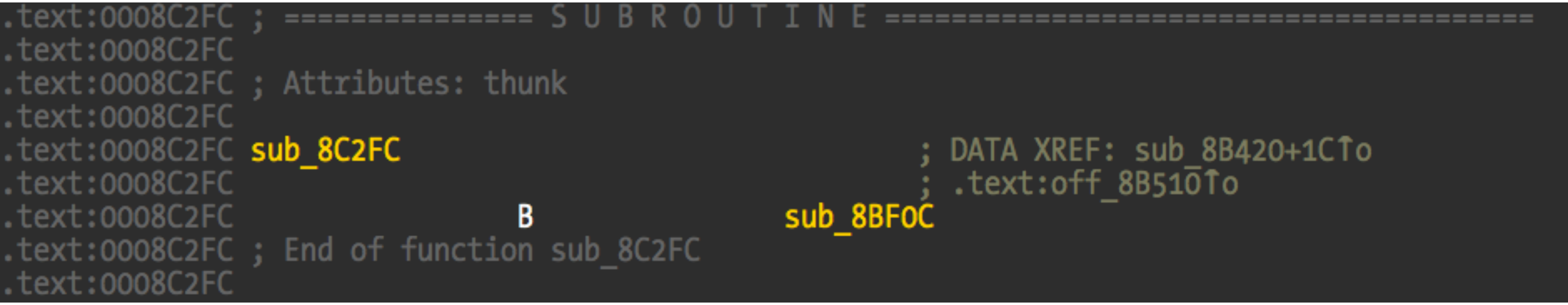
```
int __fastcall sub_8BE0C(int a1)
{
  int v1; // r6
  int v2; // r4
  int v3; // r0

  v1 = a1;
  v2 = *(_DWORD *)(dword_25A034 + 2100);
  p_tag_nofmt(*(_DWORD *)(dword_25A034 + 2100), "function SendPassword()\n");
  p_tag_nofmt(v2, "{\n");
  p_tag_nofmt(v2, "    var tmp;\n");
  p_tag(
    v2,
    "    document.form_contents.elements['%s'].value=document.form_contents.elements['%s'].value.toLowerCase()+document.f"
    "orm_contents.elements['%s'].value\n",
    "md5_pass",
    v1,
    "auth_key");
  p_tag(v2, "    tmp=hex_md5(document.form_contents.elements['%s'].value);\n", "md5_pass");
  p_tag(v2, "    document.form_contents.elements['%s'].value=tmp;\n", "md5_pass");
  p_tag(v2, "    document.form_contents.elements['%s'].value=\"\";\n", v1);
  v3 = sub_120774(dword_25A13C);
  p_tag(v2, "    mimic_button('submit_button_%s: %s..', 1);\n", v3, "");
  return p_tag_nofmt(v2, "}\n\n");
}
```

Reverse engineering

- Second hit on a reasonably long function **sub_8BF0C**.
- References to “**username**”, “**password**”, “**md5_pass**” strings.
- Before starting to reverse it, check its callers path (backtrace).

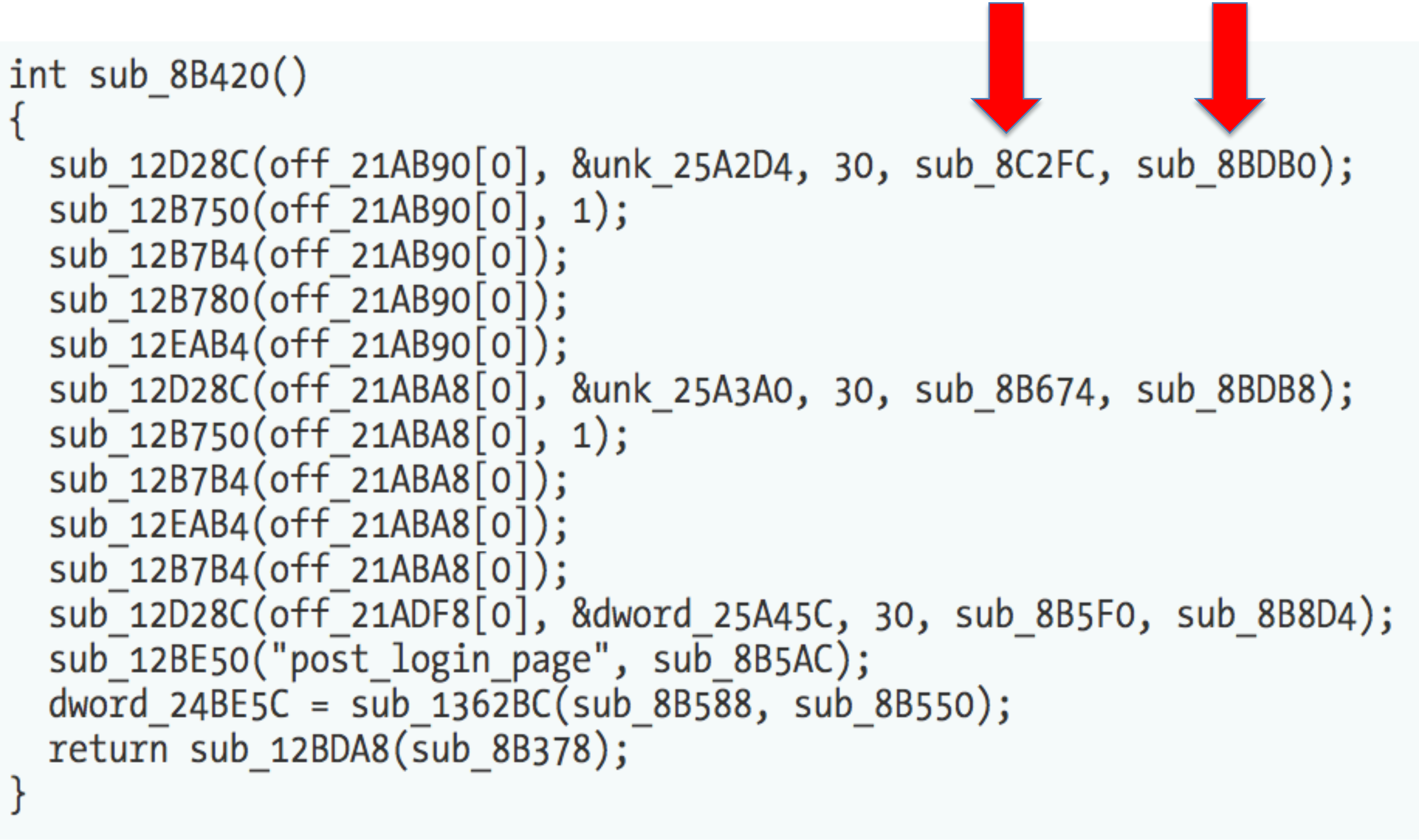
Reverse engineering

- A single caller to this function.



```
.text:0008C2FC ; =============== S U B R O U T I N E =======================================
.text:0008C2FC
.text:0008C2FC ; Attributes: thunk
.text:0008C2FC
.text:0008C2FC sub_8C2FC                               ; DATA XREF: sub_8B420+1C↑o
.text:0008C2FC                                         ; .text:off_8B510↑o
.text:0008C2FC                 B               sub_8BF0C
.text:0008C2FC ; End of function sub_8C2FC
.text:0008C2FC
```

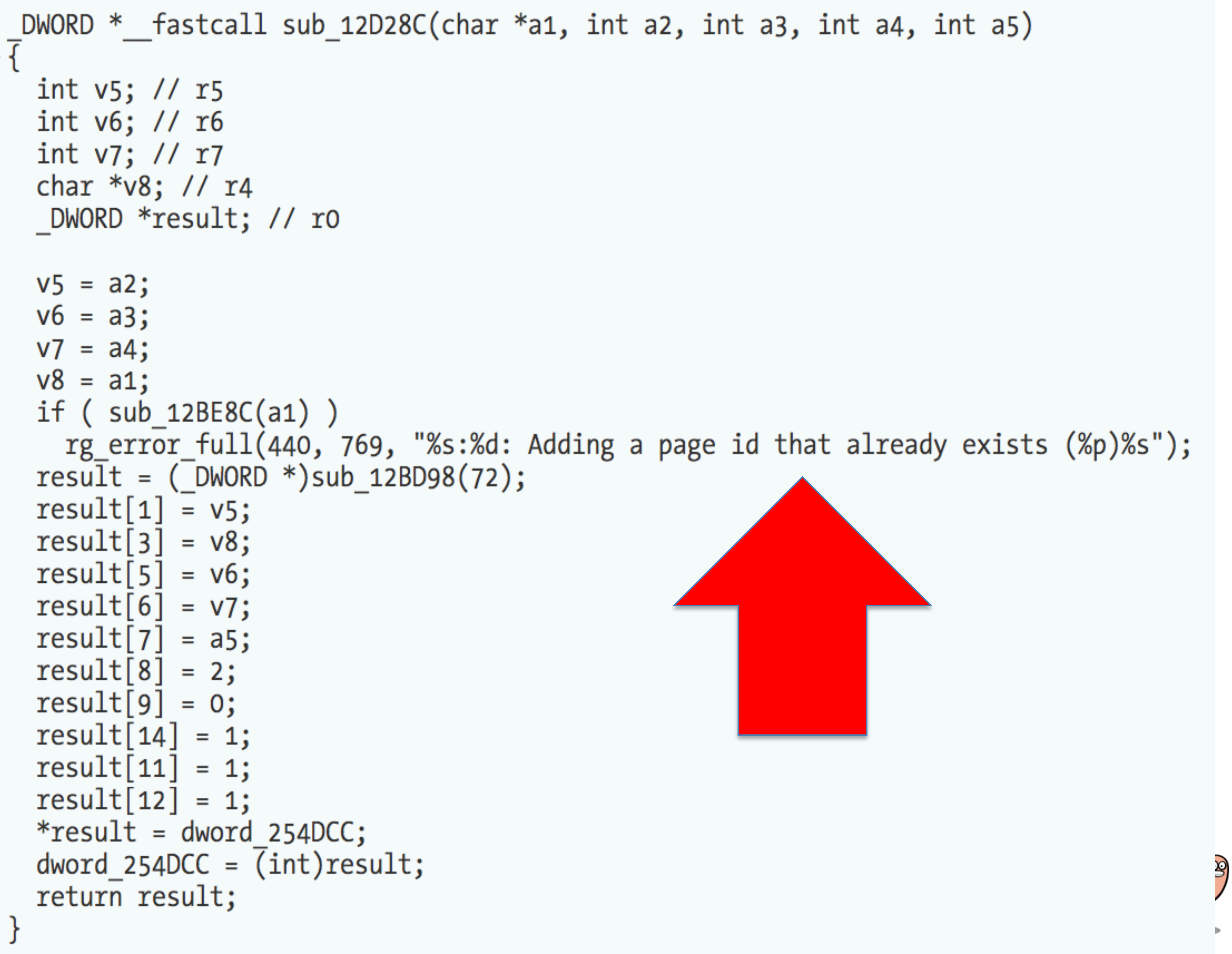
- Continue to backtrace xrefs.



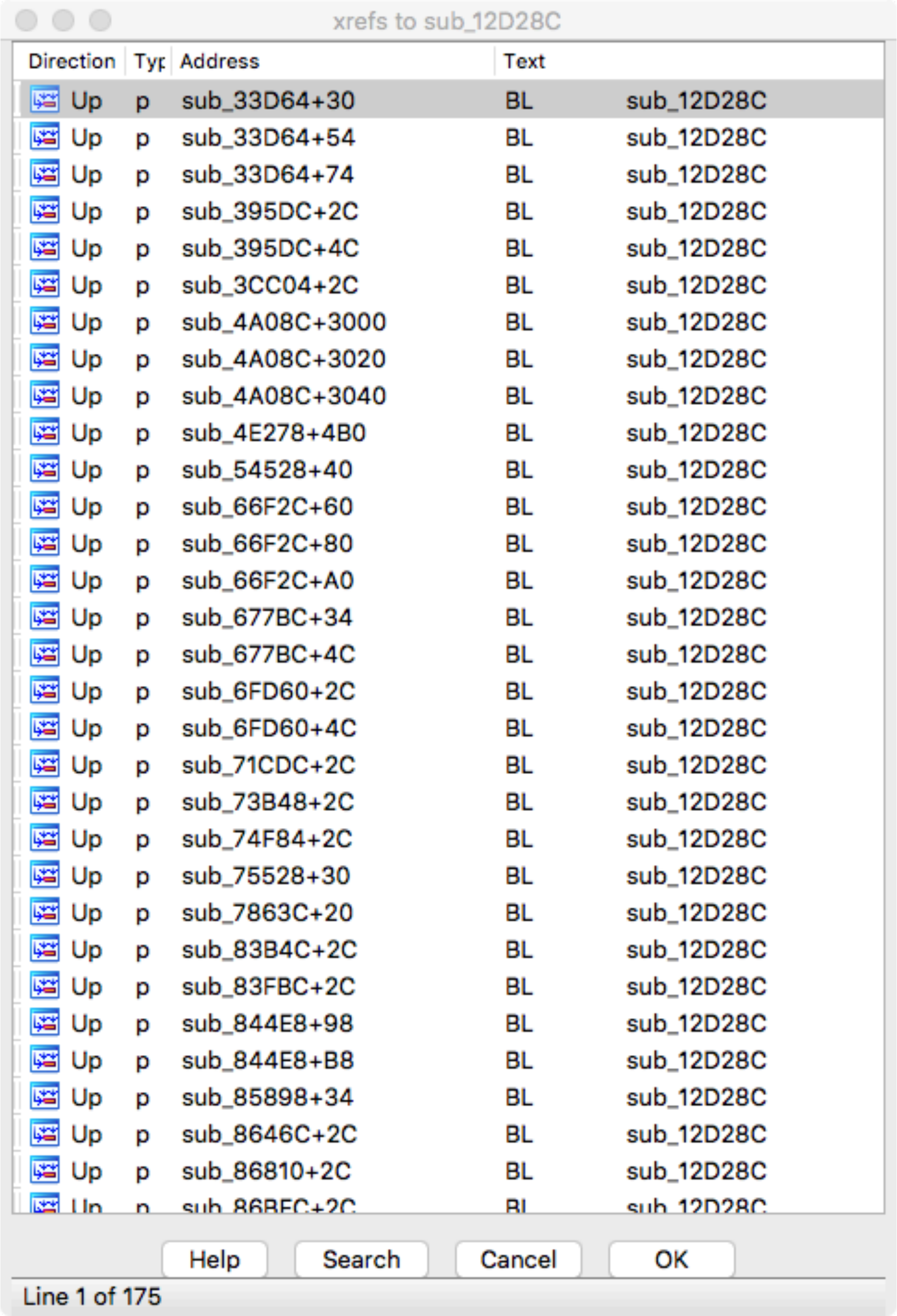
```
int sub_8B420()
{
  sub_12D28C(off_21AB90[0], &unk_25A2D4, 30, sub_8C2FC, sub_8BDB0);
  sub_12B750(off_21AB90[0], 1);
  sub_12B7B4(off_21AB90[0]);
  sub_12B780(off_21AB90[0]);
  sub_12EAB4(off_21AB90[0]);
  sub_12D28C(off_21ABA8[0], &unk_25A3A0, 30, sub_8B674, sub_8BDB8);
  sub_12B750(off_21ABA8[0], 1);
  sub_12B7B4(off_21ABA8[0]);
  sub_12EAB4(off_21ABA8[0]);
  sub_12B7B4(off_21ABA8[0]);
  sub_12D28C(off_21ADF8[0], &dword_25A45C, 30, sub_8B5F0, sub_8B8D4);
  sub_12BE50("post_login_page", sub_8B5AC);
  dword_24BE5C = sub_1362BC(sub_8B588, sub_8B550);
  return sub_12BDA8(sub_8B378);
}
```

Reverse engineering

- sub_12D28C is suspicious.
- Feels like some kind of callback registration.
- Contains a string to confirm our hypothesis.



```
_DWORD *__fastcall sub_12D28C(char *a1, int a2, int a3, int a4, int a5)
{
  int v5; // r5
  int v6; // r6
  int v7; // r7
  char *v8; // r4
  _DWORD *result; // r0

  v5 = a2;
  v6 = a3;
  v7 = a4;
  v8 = a1;
  if ( sub_12BE8C(a1) )
    rg_error_full(440, 769, "%s:%d: Adding a page id that already exists (%p)%s");
  result = (_DWORD *)sub_12BD98(72);
  result[1] = v5;
  result[3] = v8;
  result[5] = v6;
  result[6] = v7;
  result[7] = a5;
  result[8] = 2;
  result[9] = 0;
  result[14] = 1;
  result[11] = 1;
  result[12] = 1;
  *result = dword_254DCC;
  dword_254DCC = (int)result;
  return result;
}
```



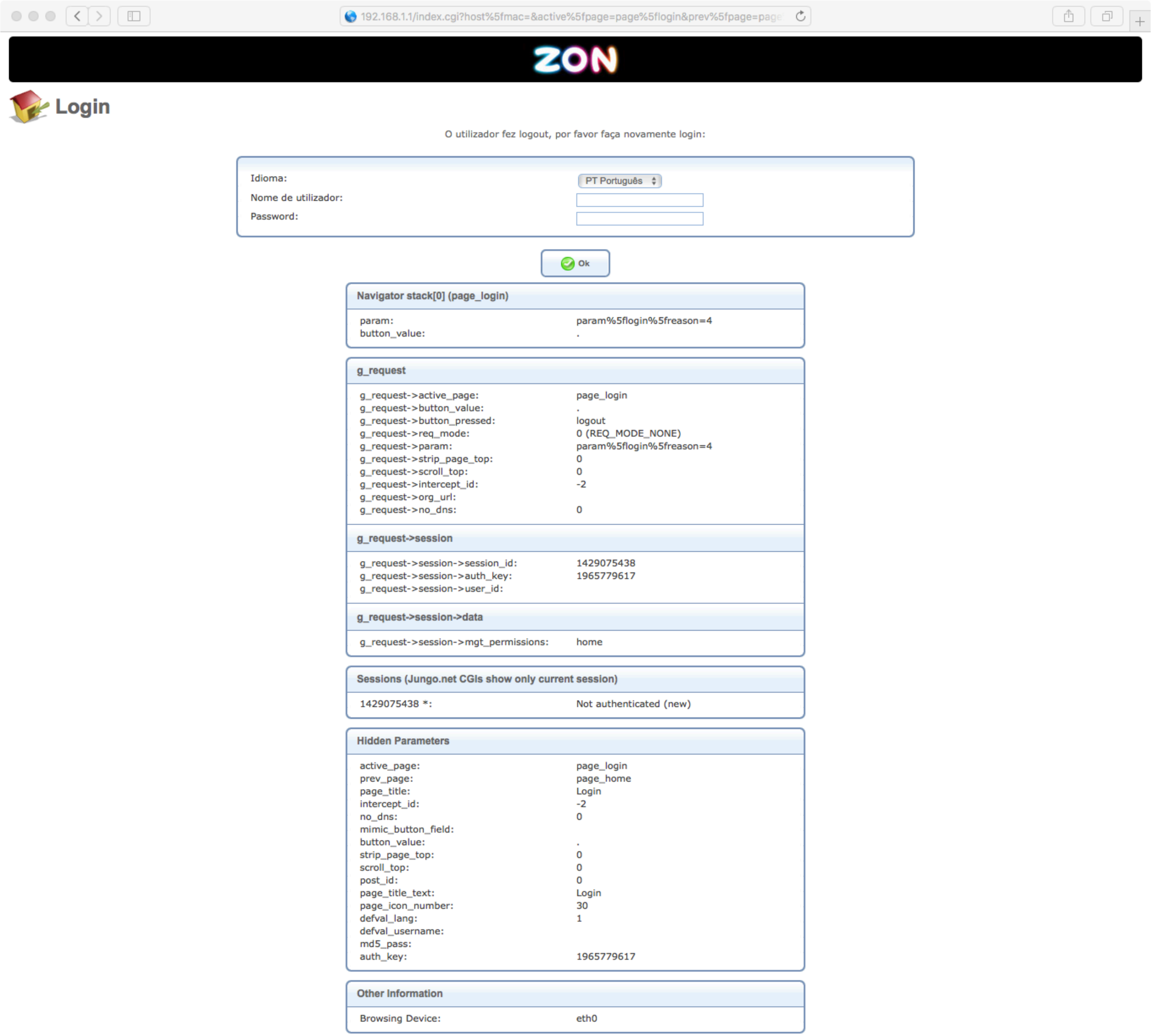
xrefs to sub_12D28C

Direction	Typ	Address	Text
Up	p	sub_33D64+30	BL sub_12D28C
Up	p	sub_33D64+54	BL sub_12D28C
Up	p	sub_33D64+74	BL sub_12D28C
Up	p	sub_395DC+2C	BL sub_12D28C
Up	p	sub_395DC+4C	BL sub_12D28C
Up	p	sub_3CC04+2C	BL sub_12D28C
Up	p	sub_4A08C+3000	BL sub_12D28C
Up	p	sub_4A08C+3020	BL sub_12D28C
Up	p	sub_4A08C+3040	BL sub_12D28C
Up	p	sub_4E278+4B0	BL sub_12D28C
Up	p	sub_54528+40	BL sub_12D28C
Up	p	sub_66F2C+60	BL sub_12D28C
Up	p	sub_66F2C+80	BL sub_12D28C
Up	p	sub_66F2C+A0	BL sub_12D28C
Up	p	sub_677BC+34	BL sub_12D28C
Up	p	sub_677BC+4C	BL sub_12D28C
Up	p	sub_6FD60+2C	BL sub_12D28C
Up	p	sub_6FD60+4C	BL sub_12D28C
Up	p	sub_71CDC+2C	BL sub_12D28C
Up	p	sub_73B48+2C	BL sub_12D28C
Up	p	sub_74F84+2C	BL sub_12D28C
Up	p	sub_75528+30	BL sub_12D28C
Up	p	sub_7863C+20	BL sub_12D28C
Up	p	sub_83B4C+2C	BL sub_12D28C
Up	p	sub_83FBC+2C	BL sub_12D28C
Up	p	sub_844E8+98	BL sub_12D28C
Up	p	sub_844E8+B8	BL sub_12D28C
Up	p	sub_85898+34	BL sub_12D28C
Up	p	sub_8646C+2C	BL sub_12D28C
Up	p	sub_86810+2C	BL sub_12D28C
Up	p	sub_86BEC+2C	BL sub_12D28C

Line 1 of 175

Reverse engineering

- We found the code that registers the events (at least for the web interface).
- Good ”choking” point to rename lots of functions and understand available events.
- Tip: “**misc wbm_debug_set 1**”
    - Web interface debugging output.



192.168.1.1/index.cgi?host%5fmac=&active%5fpage=page%5flogin&prev%5fpage=page
ZON

Login

O utilizador fez logout, por favor faça novamente login:

Idioma:	PT Português
Nome de utilizador:	
Password:	

Navigator stack[0] (page_login)

param:	param%5flogin%5freason=4
button_value:	.

g_request

g_request->active_page:	page_login
g_request->button_value:	.
g_request->button_pressed:	logout
g_request->req_mode:	0 (REQ_MODE_NONE)
g_request->param:	param%5flogin%5freason=4
g_request->strip_page_top:	0
g_request->scroll_top:	0
g_request->intercept_id:	-2
g_request->org_url:	
g_request->no_dns:	0

g_request->session

g_request->session->session_id:	1429075438
g_request->session->auth_key:	1965779617
g_request->session->user_id:	

g_request->session->data

g_request->session->mgt_permissions:	home

Sessions (Jungo.net CGIs show only current session)

1429075438 *:	Not authenticated (new)

Hidden Parameters

active_page:	page_login
prev_page:	page_home
page_title:	Login
intercept_id:	-2
no_dns:	0
mimic_button_field:	
button_value:	.
strip_page_top:	0
scroll_top:	0
post_id:	0
page_title_text:	Login
page_icon_number:	30
defval_lang:	1
defval_username:	
md5_pass:	
auth_key:	1965779617

Other Information

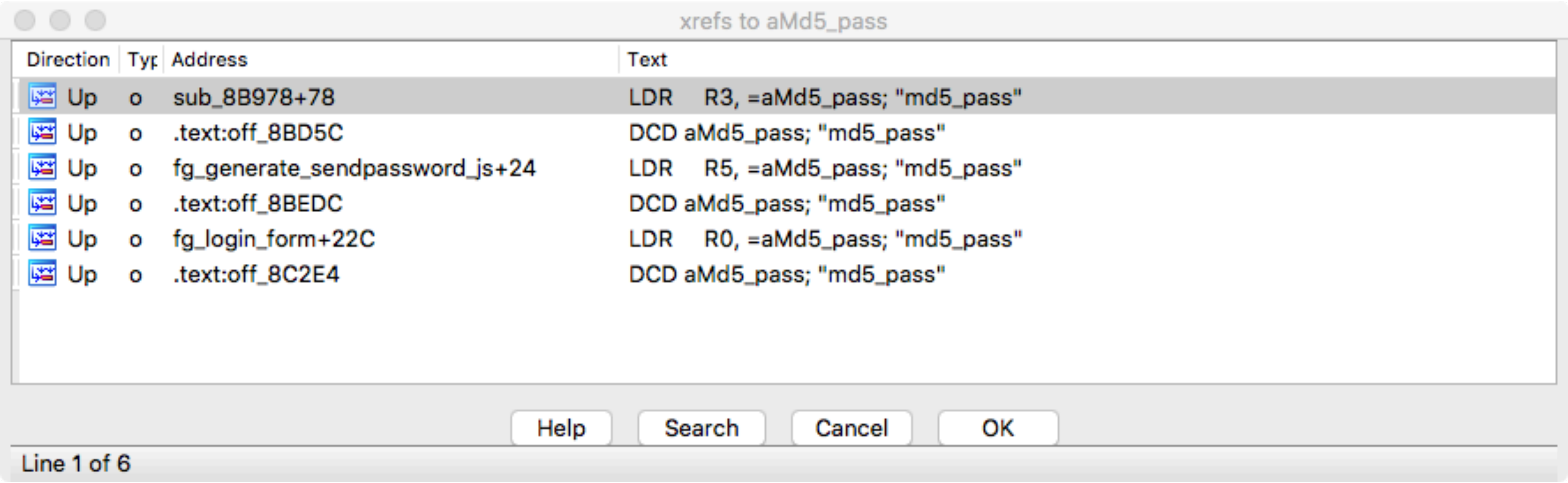
Browsing Device:	eth0

Reverse engineering

- We know where the login page is generated.
- Don’t know where the form is processed.
  - Didn’t notice at the time that it was the next argument to sub_12D28C.
- But from the JS we know which variables are submitted.

Reverse engineering

- Only three hits on “**md5_pass**”.
- Two to generate the form, one unknown.



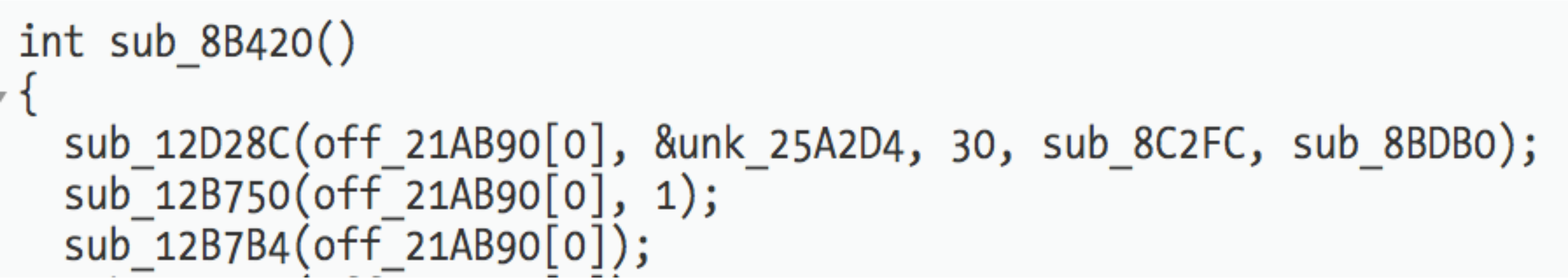
xrefs to aMd5_pass

Direction	Typ	Address	Text
Up	o	sub_8B978+78	LDR R3, =aMd5_pass; "md5_pass"
Up	o	.text:off_8BD5C	DCD aMd5_pass; "md5_pass"
Up	o	fg_generate_sendpassword_js+24	LDR R5, =aMd5_pass; "md5_pass"
Up	o	.text:off_8BEDC	DCD aMd5_pass; "md5_pass"
Up	o	fg_login_form+22C	LDR R0, =aMd5_pass; "md5_pass"
Up	o	.text:off_8C2E4	DCD aMd5_pass; "md5_pass"

Line 1 of 6

Reverse engineering

- Function **sub_8B978** is called from **sub_8BDB0**.

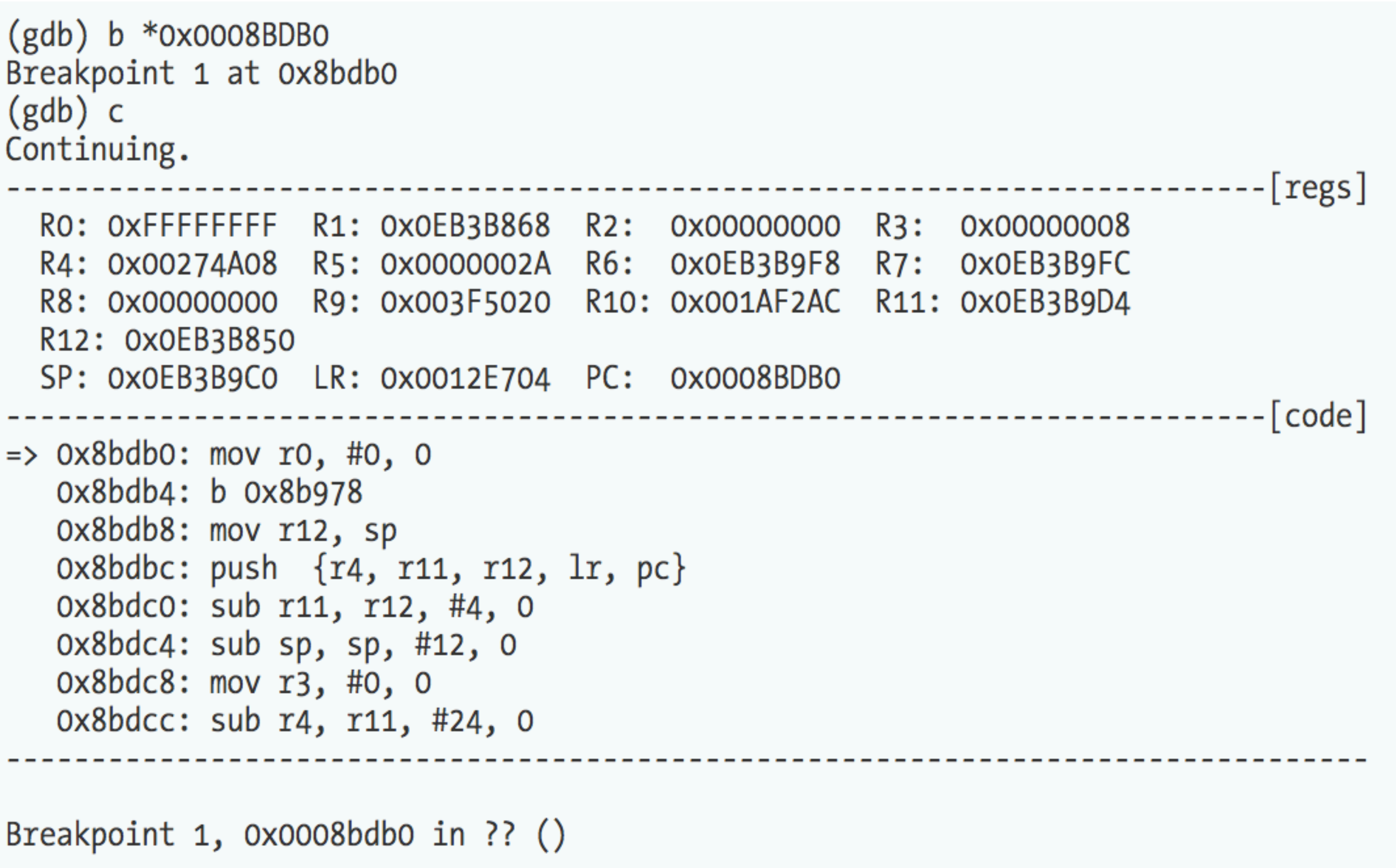


```
int sub_8B420()
{
  sub_12D28C(off_21AB90[0], &unk_25A2D4, 30, sub_8C2FC, sub_8BDB0);
  sub_12B750(off_21AB90[0], 1);
  sub_12B7B4(off_21AB90[0]);
```

- First pointer is to draw HTML, second to parse POST.

Reverse engineering

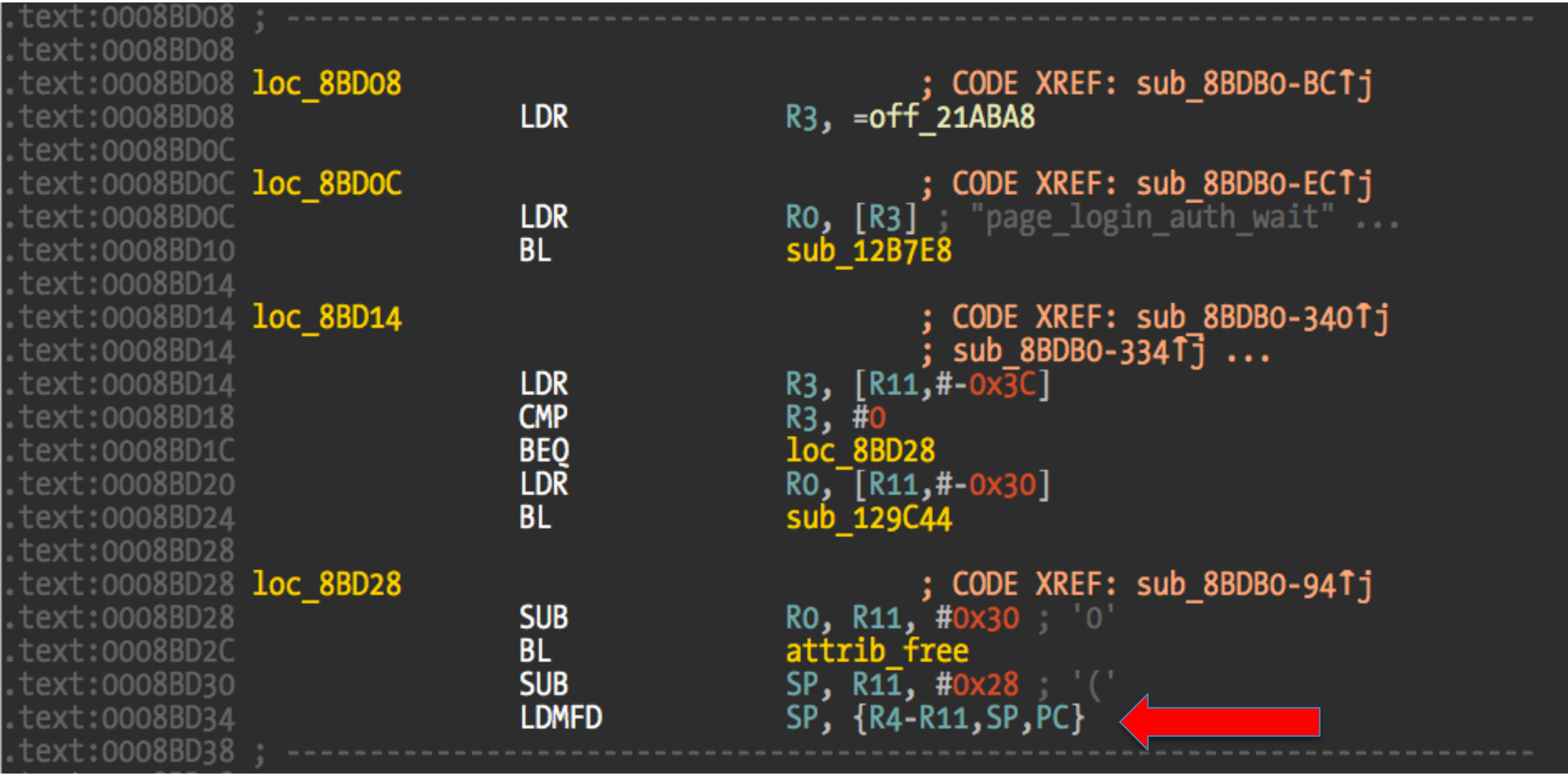
- We can confirm this using the debugger.
- Set a breakpoint at sub_8BDB0.
- Should hit when we press the “Ok” button in the login form.



```
(gdb) b *0x0008BDB0
Breakpoint 1 at 0x8bdb0
(gdb) c
Continuing.
---------------------------------------------------------------------------[regs]
  R0: 0xFFFFFFFF  R1: 0x0EB3B868  R2:  0x00000000  R3:  0x00000008
  R4: 0x00274A08  R5: 0x0000002A  R6:  0x0EB3B9F8  R7:  0x0EB3B9FC
  R8: 0x00000000  R9: 0x003F5020  R10: 0x001AF2AC  R11: 0x0EB3B9D4
  R12: 0x0EB3B850
  SP: 0x0EB3B9C0  LR: 0x0012E704  PC:  0x0008BDB0
---------------------------------------------------------------------------[code]
=> 0x8bdb0: mov r0, #0, 0
   0x8bdb4: b 0x8b978
   0x8bdb8: mov r12, sp
   0x8bdbc: push  {r4, r11, r12, lr, pc}
   0x8bdc0: sub r11, r12, #4, 0
   0x8bdc4: sub sp, sp, #12, 0
   0x8bdc8: mov r3, #0, 0
   0x8bdcc: sub r4, r11, #24, 0
--------------------------------------------------------------------------------


Breakpoint 1, 0x0008bdb0 in ?? ()
```

Reverse engineering

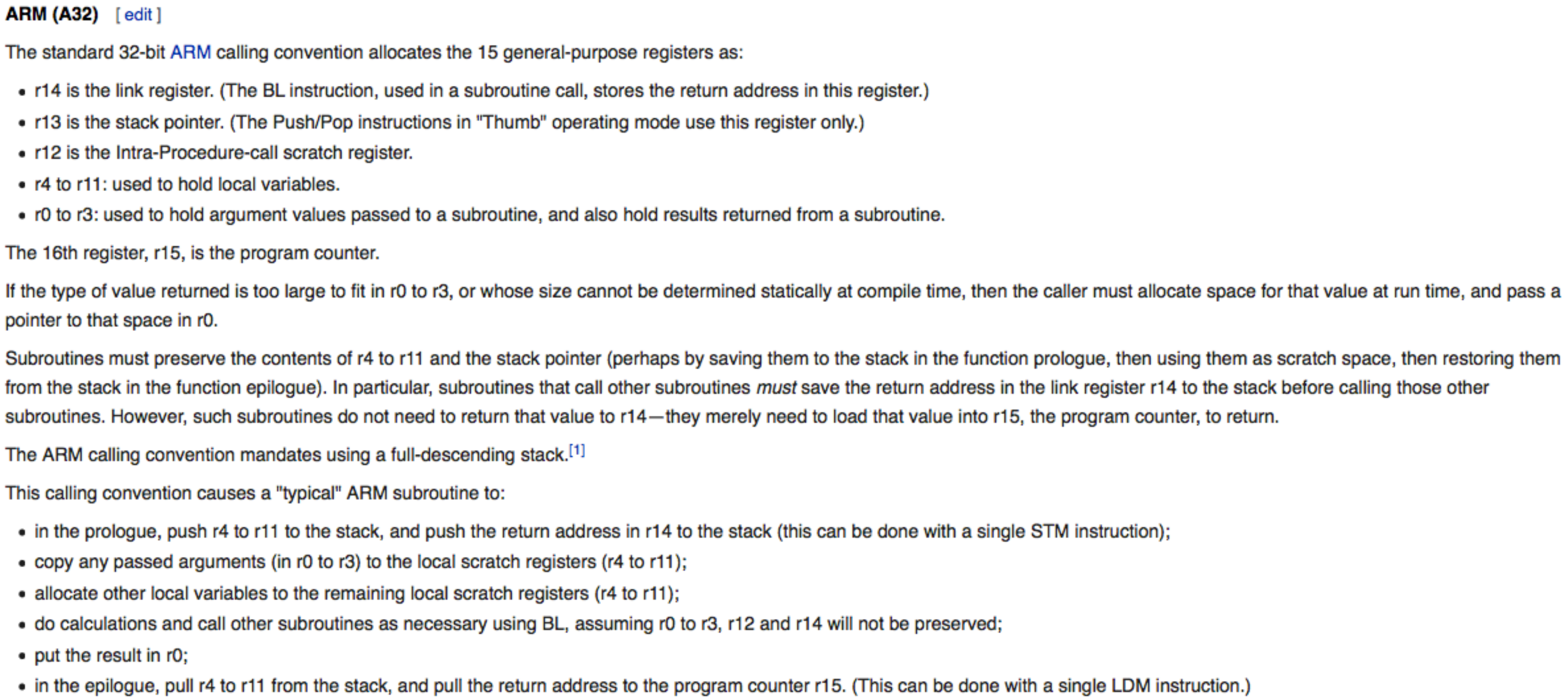
- Function **sub_8B978** is our prime target.
- Decent sized function (~980 bytes).
- Not obvious what it does (I don’t like ARM!).
- Start by doing basic tracing.
- **First**, check the return values.



```
.text:0008BD08 ; ---------------------------------------------------------------------------
.text:0008BD08
.text:0008BD08 loc_8BD08                               ; CODE XREF: sub_8BDB0-BC↑j
.text:0008BD08                 LDR             R3, =off_21ABA8
.text:0008BD0C
.text:0008BD0C loc_8BD0C                               ; CODE XREF: sub_8BDB0-EC↑j
.text:0008BD0C                 LDR             R0, [R3] ; "page_login_auth_wait" ...
.text:0008BD10                 BL              sub_12B7E8
.text:0008BD14
.text:0008BD14 loc_8BD14                               ; CODE XREF: sub_8BDB0-340↑j
.text:0008BD14                                         ; sub_8BDB0-334↑j ...
.text:0008BD14                 LDR             R3, [R11,#-0x3C]
.text:0008BD18                 CMP             R3, #0
.text:0008BD1C                 BEQ             loc_8BD28
.text:0008BD20                 LDR             R0, [R11,#-0x30]
.text:0008BD24                 BL              sub_129C44
.text:0008BD28
.text:0008BD28 loc_8BD28                               ; CODE XREF: sub_8BDB0-94↑j
.text:0008BD28                 SUB             R0, R11, #0x30 ; '0'
.text:0008BD2C                 BL              attrib_free
.text:0008BD30                 SUB             SP, R11, #0x28 ; '('
.text:0008BD34                 LDMFD           SP, {R4-R11,SP,PC}
.text:0008BD38 ; ---------------------------------------------------------------------------
```

Reverse engineering

- A single exit point at address 0x8BD34.
- LDMFD instruction to restore stack and all must preserve registers.
- Breakpoint and compare return values with good and bad password.
- Nothing interesting.



ARM (A32) [edit]

The standard 32-bit ARM calling convention allocates the 15 general-purpose registers as:

- r14 is the link register. (The BL instruction, used in a subroutine call, stores the return address in this register.)
- r13 is the stack pointer. (The Push/Pop instructions in "Thumb" operating mode use this register only.)
- r12 is the Intra-Procedure-call scratch register.
- r4 to r11: used to hold local variables.
- r0 to r3: used to hold argument values passed to a subroutine, and also hold results returned from a subroutine.

The 16th register, r15, is the program counter.

If the type of value returned is too large to fit in r0 to r3, or whose size cannot be determined statically at compile time, then the caller must allocate space for that value at run time, and pass a pointer to that space in r0.

Subroutines must preserve the contents of r4 to r11 and the stack pointer (perhaps by saving them to the stack in the function prologue, then using them as scratch space, then restoring them from the stack in the function epilogue). In particular, subroutines that call other subroutines *must* save the return address in the link register r14 to the stack before calling those other subroutines. However, such subroutines do not need to return that value to r14—they merely need to load that value into r15, the program counter, to return.

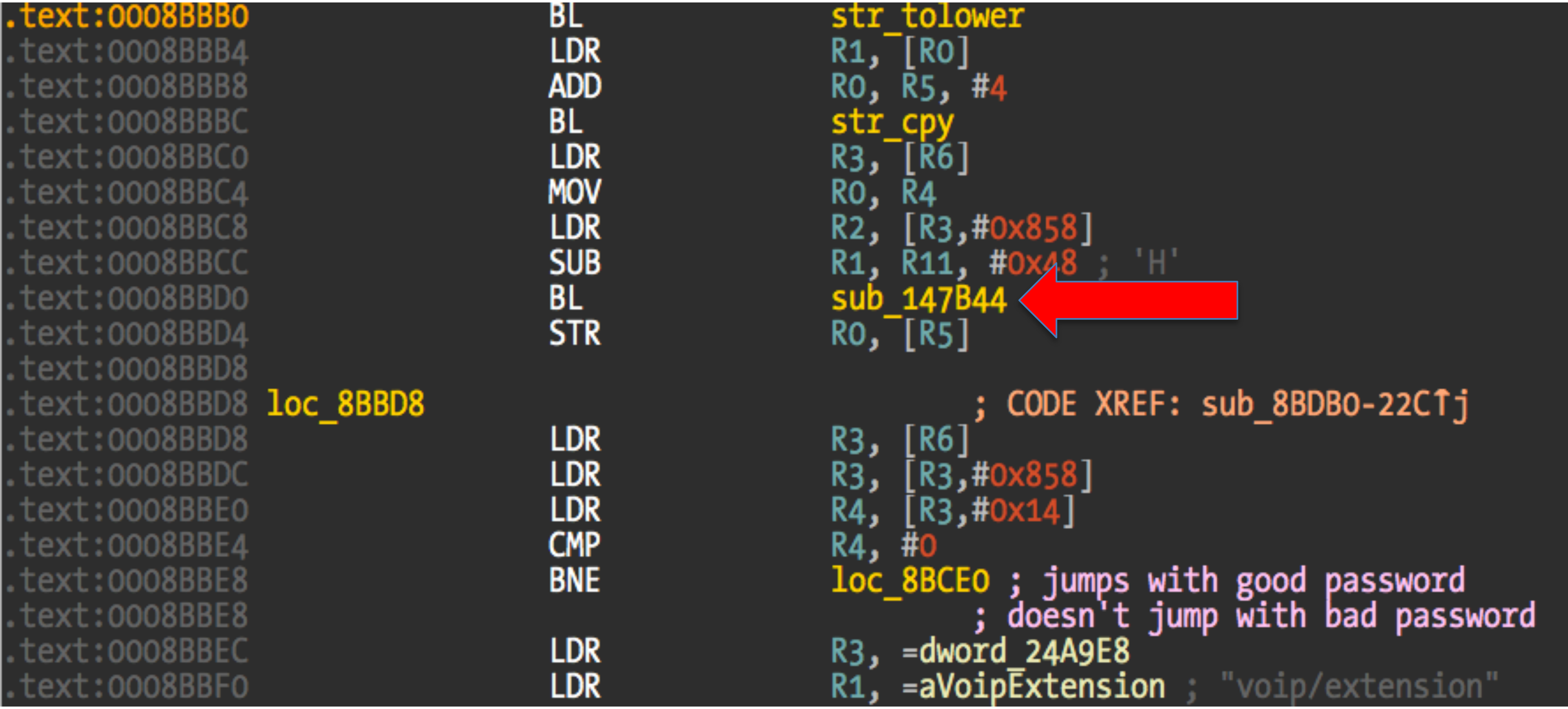
The ARM calling convention mandates using a full-descending stack.[1]

This calling convention causes a "typical" ARM subroutine to:

- in the prologue, push r4 to r11 to the stack, and push the return address in r14 to the stack (this can be done with a single STM instruction);
- copy any passed arguments (in r0 to r3) to the local scratch registers (r4 to r11);
- allocate other local variables to the remaining local scratch registers (r4 to r11);
- do calculations and call other subroutines as necessary using BL, assuming r0 to r3, r12 and r14 will not be preserved;
- put the result in r0;
- in the epilogue, pull r4 to r11 from the stack, and pull the return address to the program counter r15. (This can be done with a single LDM instruction.)

Reverse engineering

- Next attempt is to diff execution flow with good and bad passwords.
- Cheap method to find where to focus and avoid understand everything the function does.
- I get sleepy reading IDA output.



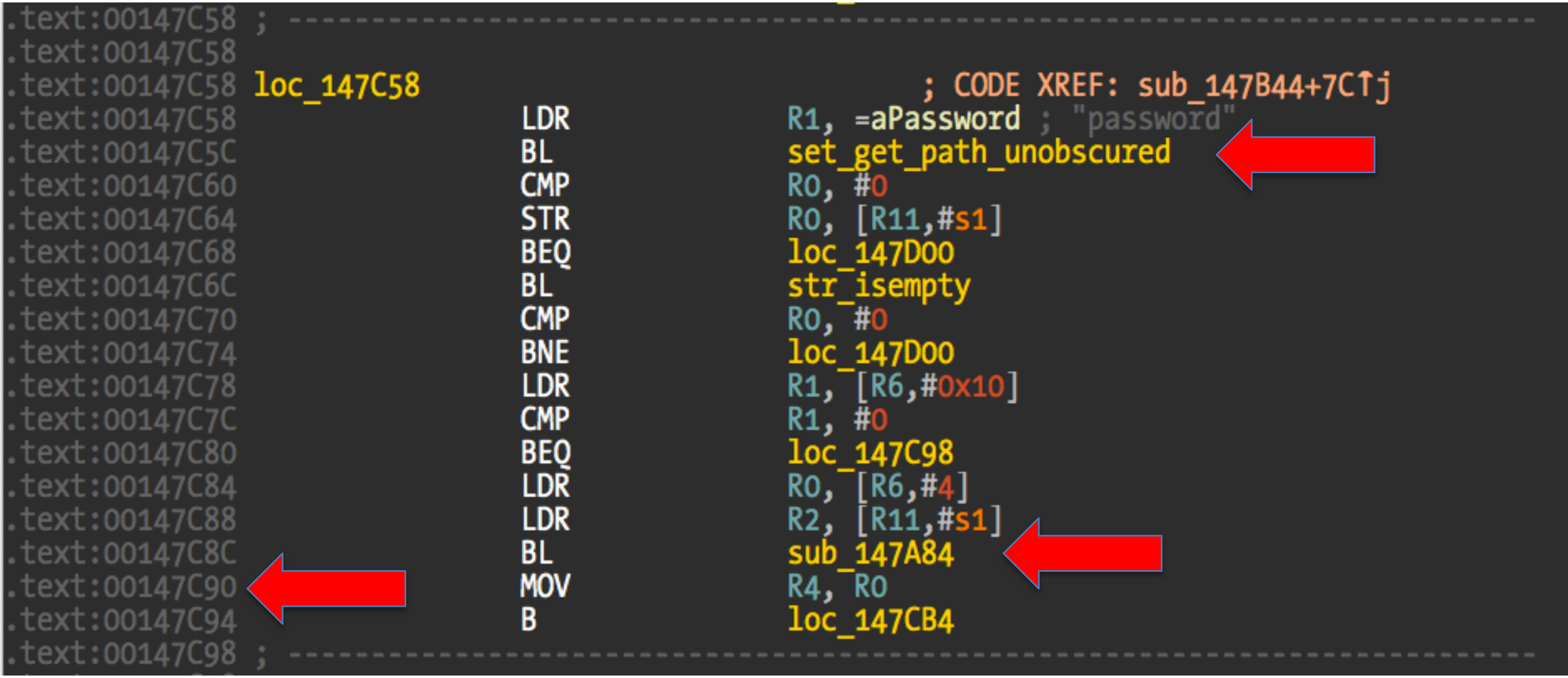
```
.text:0008BBB0                 BL              str_tolower
.text:0008BBB4                 LDR             R1, [R0]
.text:0008BBB8                 ADD             R0, R5, #4
.text:0008BBBC                 BL              str_cpy
.text:0008BBC0                 LDR             R3, [R6]
.text:0008BBC4                 MOV             R0, R4
.text:0008BBC8                 LDR             R2, [R3,#0x858]
.text:0008BBCC                 SUB             R1, R11, #0x48 ; 'H'
.text:0008BBD0                 BL              sub_147B44
.text:0008BBD4                 STR             R0, [R5]
.text:0008BBD8
.text:0008BBD8 loc_8BBD8                                ; CODE XREF: sub_8BDB0-22C↑j
.text:0008BBD8                 LDR             R3, [R6]
.text:0008BBDC                 LDR             R3, [R3,#0x858]
.text:0008BBE0                 LDR             R4, [R3,#0x14]
.text:0008BBE4                 CMP             R4, #0
.text:0008BBE8                 BNE             loc_8BCE0 ; jumps with good password
.text:0008BBE8                                          ; doesn't jump with bad password
.text:0008BBEC                 LDR             R3, =dword_24A9E8
.text:0008BBF0                 LDR             R1, =aVoipExtension ; "voip/extension"
```

Reverse engineering

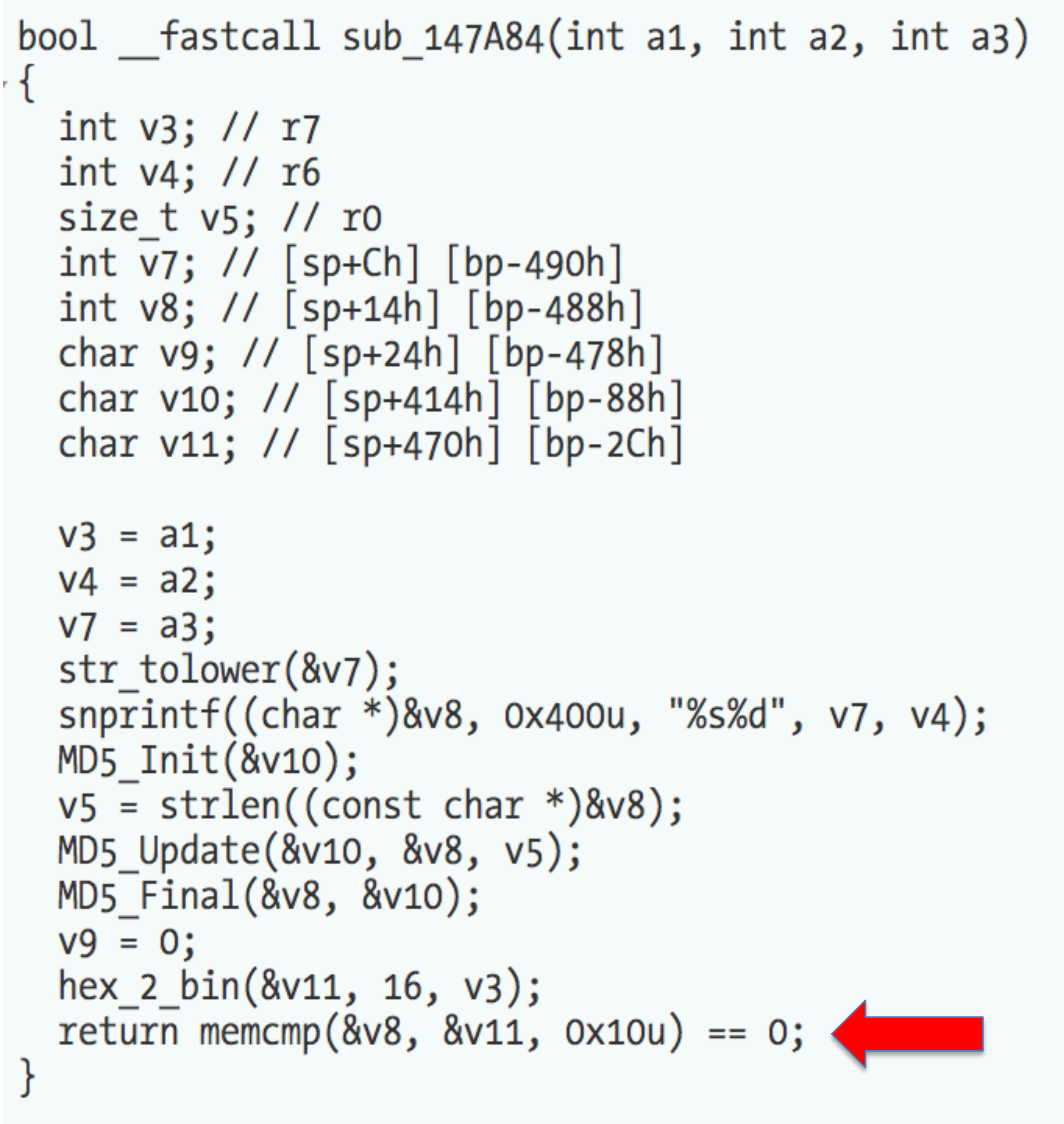
- We can find a spot where things go different.
- So **sub_147B44** is a function we want to explore next.
- The **str_cpy** copies login username.

Reverse engineering

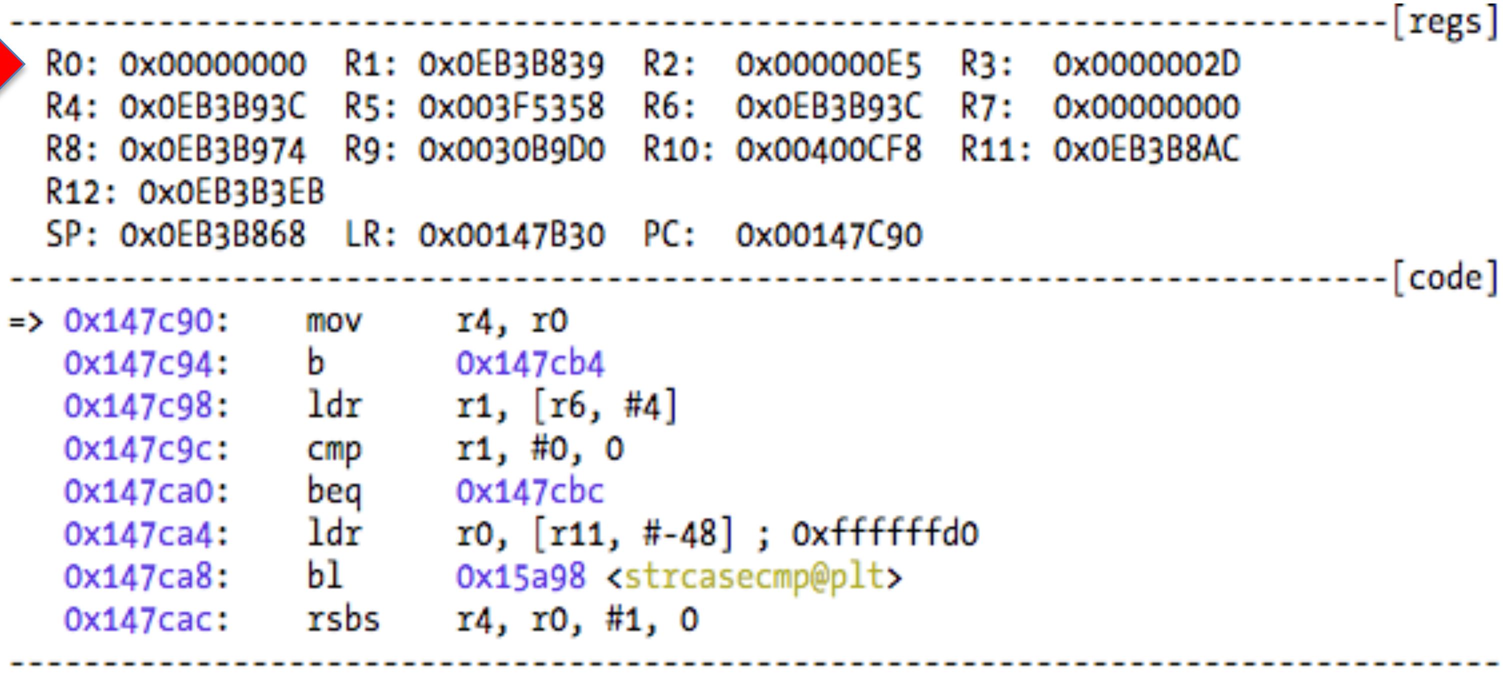
- Time to explore **sub_147B44**.
- We can find a function **sub_147A84** that does MD5 hashing.
- Good place to diff execution.
- Breakpoint at return address and compare return values with good/bad passwords.



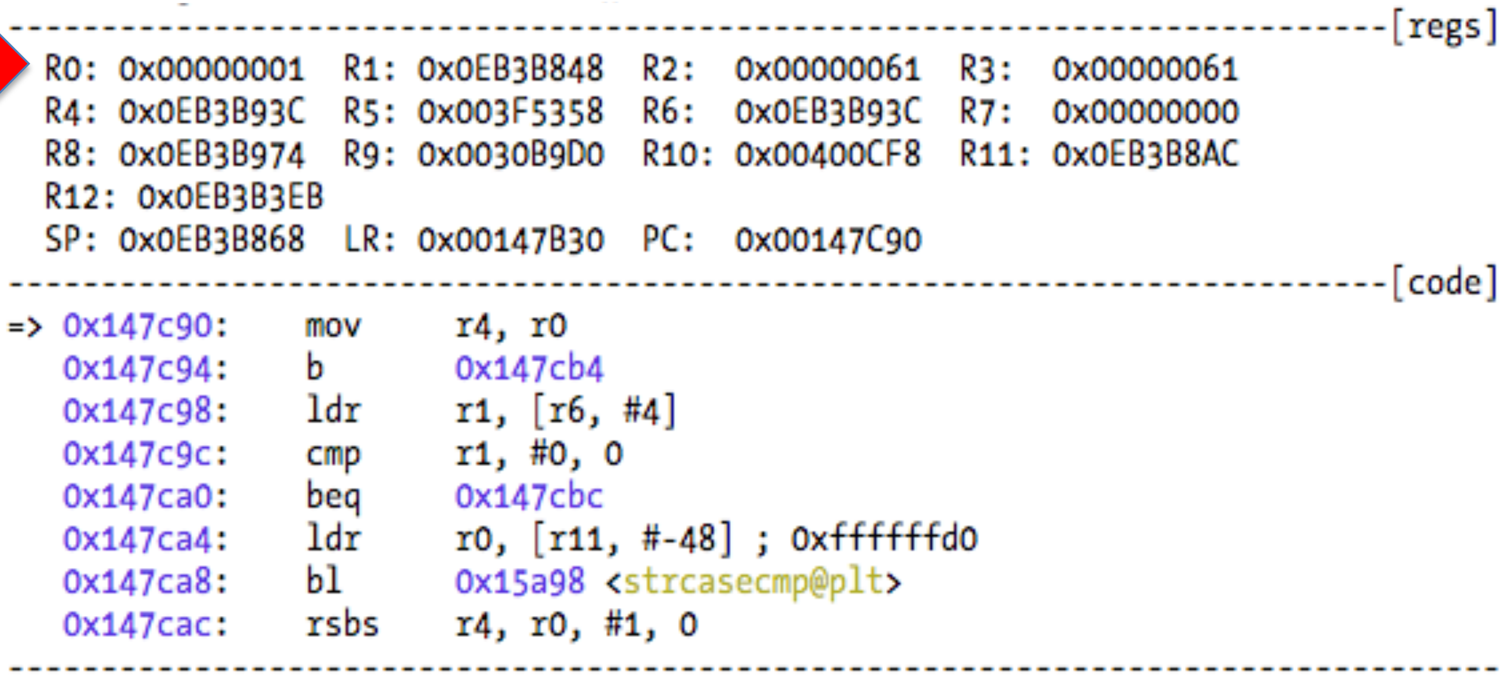
```
.text:00147C58 ; ---------------------------------------------------------------
.text:00147C58
.text:00147C58 loc_147C58                              ; CODE XREF: sub_147B44+7C↑j
.text:00147C58                 LDR             R1, =aPassword ; "password"
.text:00147C5C                 BL              set_get_path_unobscured
.text:00147C60                 CMP             R0, #0
.text:00147C64                 STR             R0, [R11,#s1]
.text:00147C68                 BEQ             loc_147D00
.text:00147C6C                 BL              str_isempty
.text:00147C70                 CMP             R0, #0
.text:00147C74                 BNE             loc_147D00
.text:00147C78                 LDR             R1, [R6,#0x10]
.text:00147C7C                 CMP             R1, #0
.text:00147C80                 BEQ             loc_147C98
.text:00147C84                 LDR             R0, [R6,#4]
.text:00147C88                 LDR             R2, [R11,#s1]
.text:00147C8C                 BL              sub_147A84
.text:00147C90                 MOV             R4, R0
.text:00147C94                 B               loc_147CB4
.text:00147C98 ; ---------------------------------------------------------------
```



```
bool __fastcall sub_147A84(int a1, int a2, int a3)
{
  int v3; // r7
  int v4; // r6
  size_t v5; // r0
  int v7; // [sp+Ch] [bp-490h]
  int v8; // [sp+14h] [bp-488h]
  char v9; // [sp+24h] [bp-478h]
  char v10; // [sp+414h] [bp-88h]
  char v11; // [sp+470h] [bp-2Ch]

  v3 = a1;
  v4 = a2;
  v7 = a3;
  str_tolower(&v7);
  snprintf((char *)&v8, 0x400u, "%s%d", v7, v4);
  MD5_Init(&v10);
  v5 = strlen((const char *)&v8);
  MD5_Update(&v10, &v8, v5);
  MD5_Final(&v8, &v10);
  v9 = 0;
  hex_2_bin(&v11, 16, v3);
  return memcmp(&v8, &v11, 0x10u) == 0;
}
```



```
--------------------------------------------------------------------------[regs]
  R0: 0x00000000  R1: 0x0EB3B839  R2:  0x000000E5  R3:  0x0000002D
  R4: 0x0EB3B93C  R5: 0x003F5358  R6:  0x0EB3B93C  R7:  0x00000000
  R8: 0x0EB3B974  R9: 0x0030B9D0  R10: 0x00400CF8  R11: 0x0EB3B8AC
  R12: 0x0EB3B3EB
  SP: 0x0EB3B868  LR: 0x00147B30  PC:  0x00147C90
--------------------------------------------------------------------------[code]
=> 0x147c90:    mov     r4, r0
   0x147c94:    b       0x147cb4
   0x147c98:    ldr     r1, [r6, #4]
   0x147c9c:    cmp     r1, #0, 0
   0x147ca0:    beq     0x147cbc
   0x147ca4:    ldr     r0, [r11, #-48] ; 0xffffffd0
   0x147ca8:    bl      0x15a98 <strcasecmp@plt>
   0x147cac:    rsbs    r4, r0, #1, 0
--------------------------------------------------------------------------------
```



```
--------------------------------------------------------------------------[regs]
  R0: 0x00000001  R1: 0x0EB3B848  R2:  0x00000061  R3:  0x00000061
  R4: 0x0EB3B93C  R5: 0x003F5358  R6:  0x0EB3B93C  R7:  0x00000000
  R8: 0x0EB3B974  R9: 0x0030B9D0  R10: 0x00400CF8  R11: 0x0EB3B8AC
  R12: 0x0EB3B3EB
  SP: 0x0EB3B868  LR: 0x00147B30  PC:  0x00147C90
--------------------------------------------------------------------------[code]
=> 0x147c90:    mov     r4, r0
   0x147c94:    b       0x147cb4
   0x147c98:    ldr     r1, [r6, #4]
   0x147c9c:    cmp     r1, #0, 0
   0x147ca0:    beq     0x147cbc
   0x147ca4:    ldr     r0, [r11, #-48] ; 0xffffffd0
   0x147ca8:    bl      0x15a98 <strcasecmp@plt>
   0x147cac:    rsbs    r4, r0, #1, 0
--------------------------------------------------------------------------------
```

Reverse engineering

- Return values:
  - 1: password ok
  - 0: bad password
- Just insert bad password and modify R0 to 1 when breakpoint is hit.

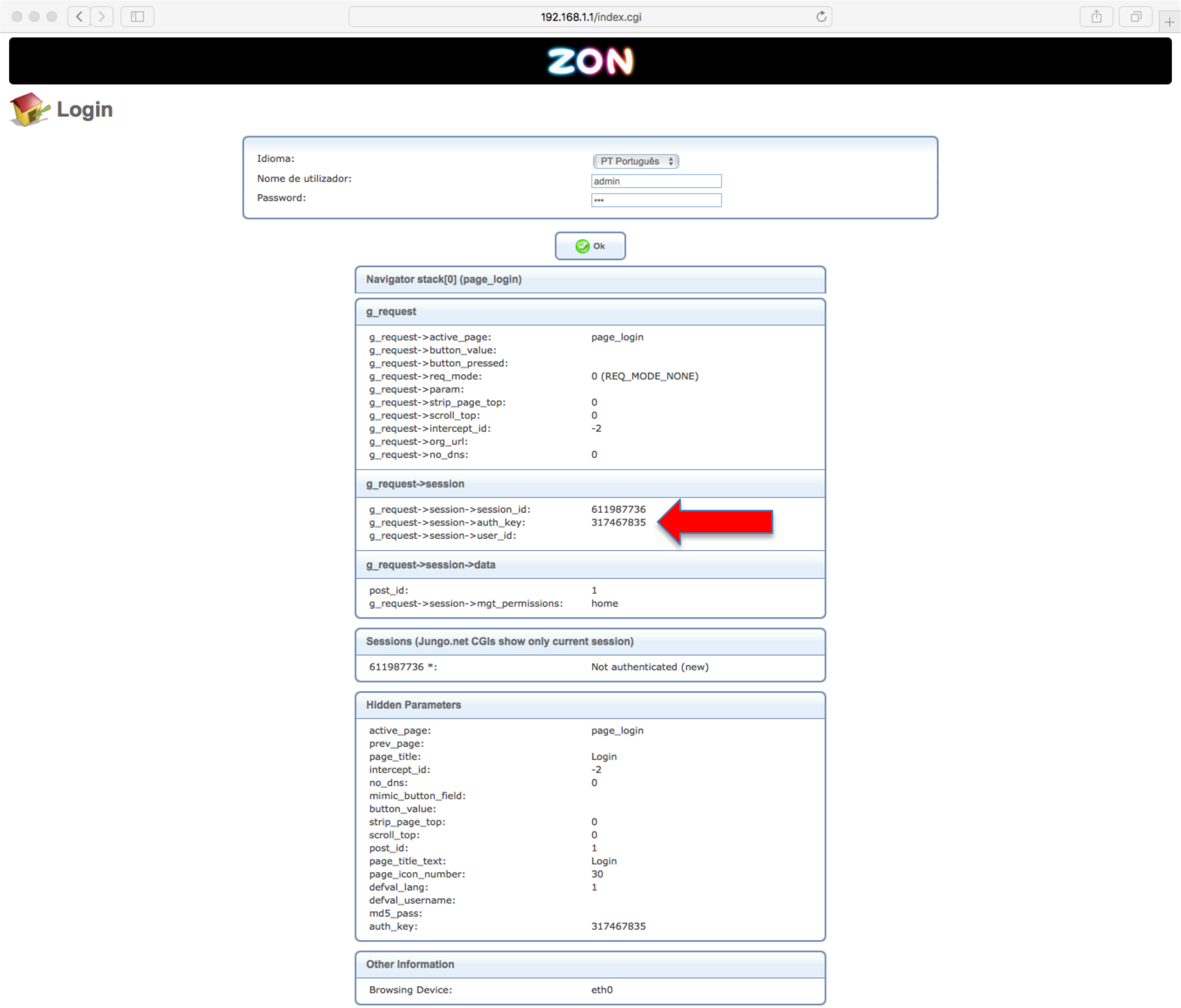
Patch here, patch there, patch everywhere!

Reverse engineering

- We can log in with any account we want without knowing its password.
- That's nice but still not able to find out the default passwords.

Reverse engineering

- Let’s take a look at the MD5 function arguments.
- int function(char* MD5, int auth_key, char *password)
- auth_key is some sort of session key.
    - Rotated after successful login (and timer?).



Login

Idioma:	PT Português
Nome de utilizador:	admin
Password:	•••

Navigator stack[0] (page_login)

g_request

g_request->active_page:	page_login
g_request->button_value:	
g_request->button_pressed:	
g_request->req_mode:	0 (REQ_MODE_NONE)
g_request->param:	
g_request->strip_page_top:	0
g_request->scroll_top:	0
g_request->intercept_id:	-2
g_request->org_url:	
g_request->no_dns:	0

g_request->session

g_request->session->session_id:	611987736
g_request->session->auth_key:	317467835
g_request->session->user_id:	

g_request->session->data

post_id:	1
g_request->session->mgt_permissions:	home

Sessions (Jungo.net CGIs show only current session)

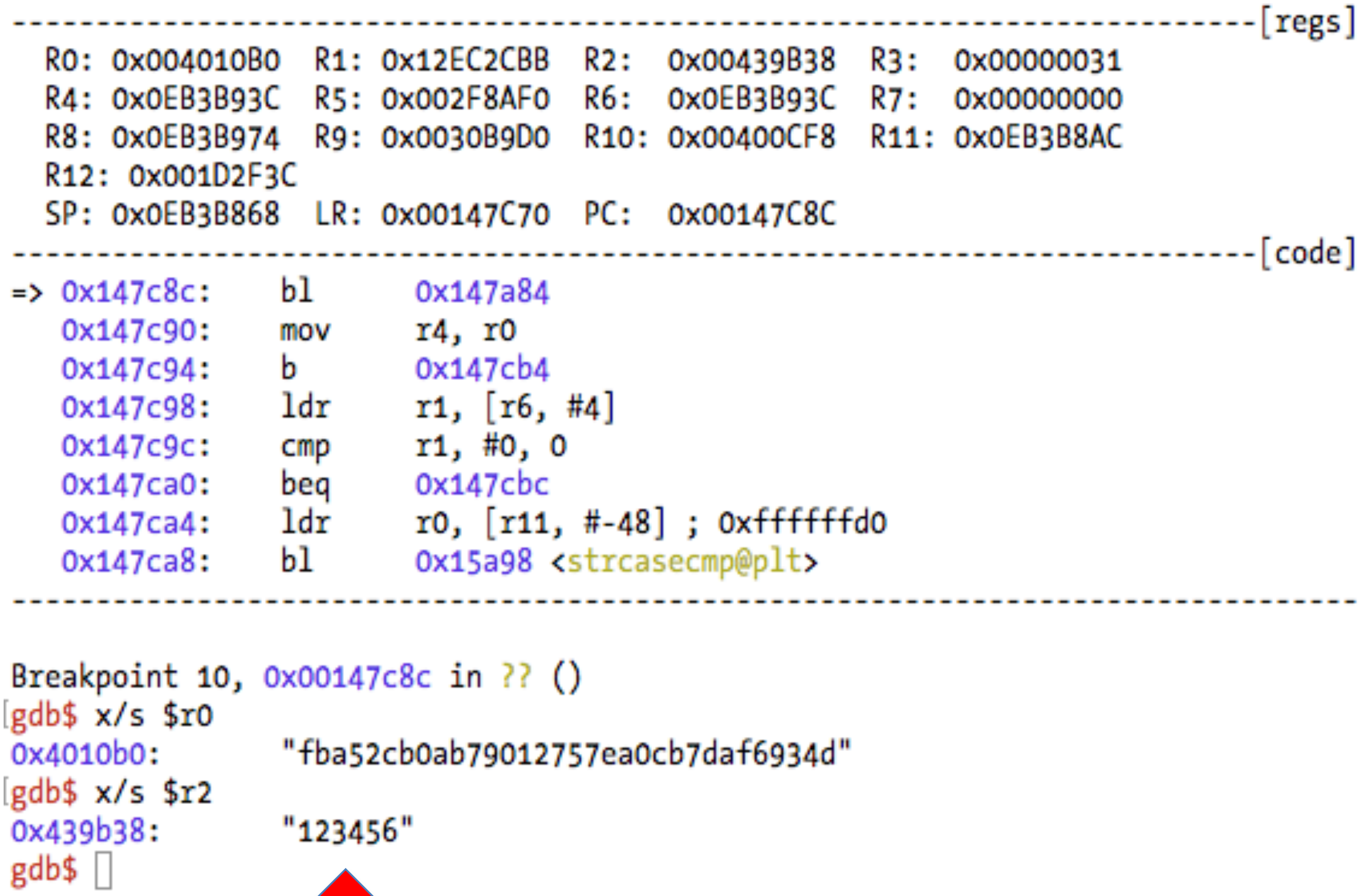
611987736 *:	Not authenticated (new)

Hidden Parameters

active_page:	page_login
prev_page:	
page_title:	Login
intercept_id:	-2
no_dns:	0
mimic_button_field:	
button_value:	
strip_page_top:	0
scroll_top:	0
post_id:	1
page_title_text:	Login
page_icon_number:	30
defval_lang:	1
defval_username:	
md5_pass:	
auth_key:	317467835

Other Information

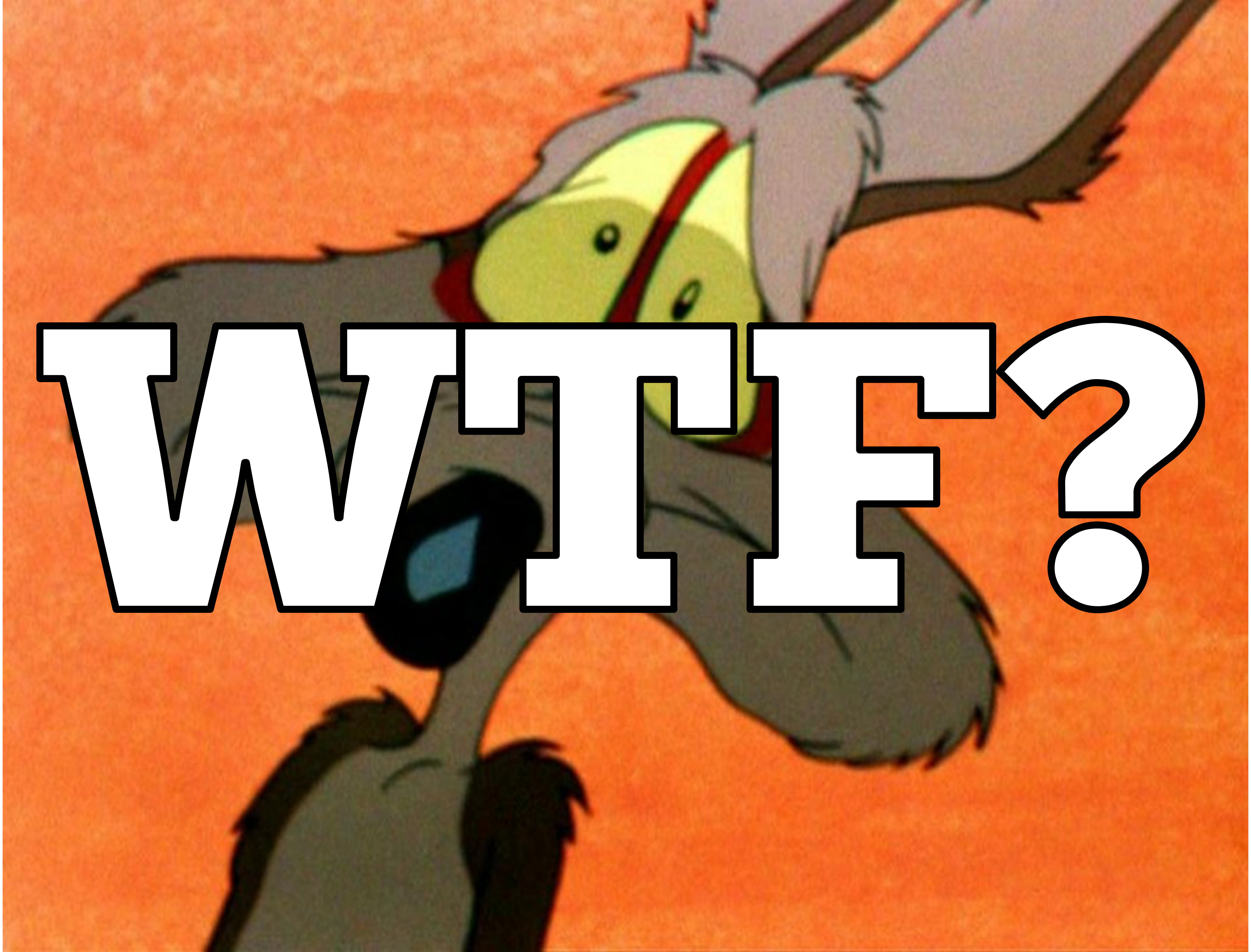
Browsing Device:	eth0



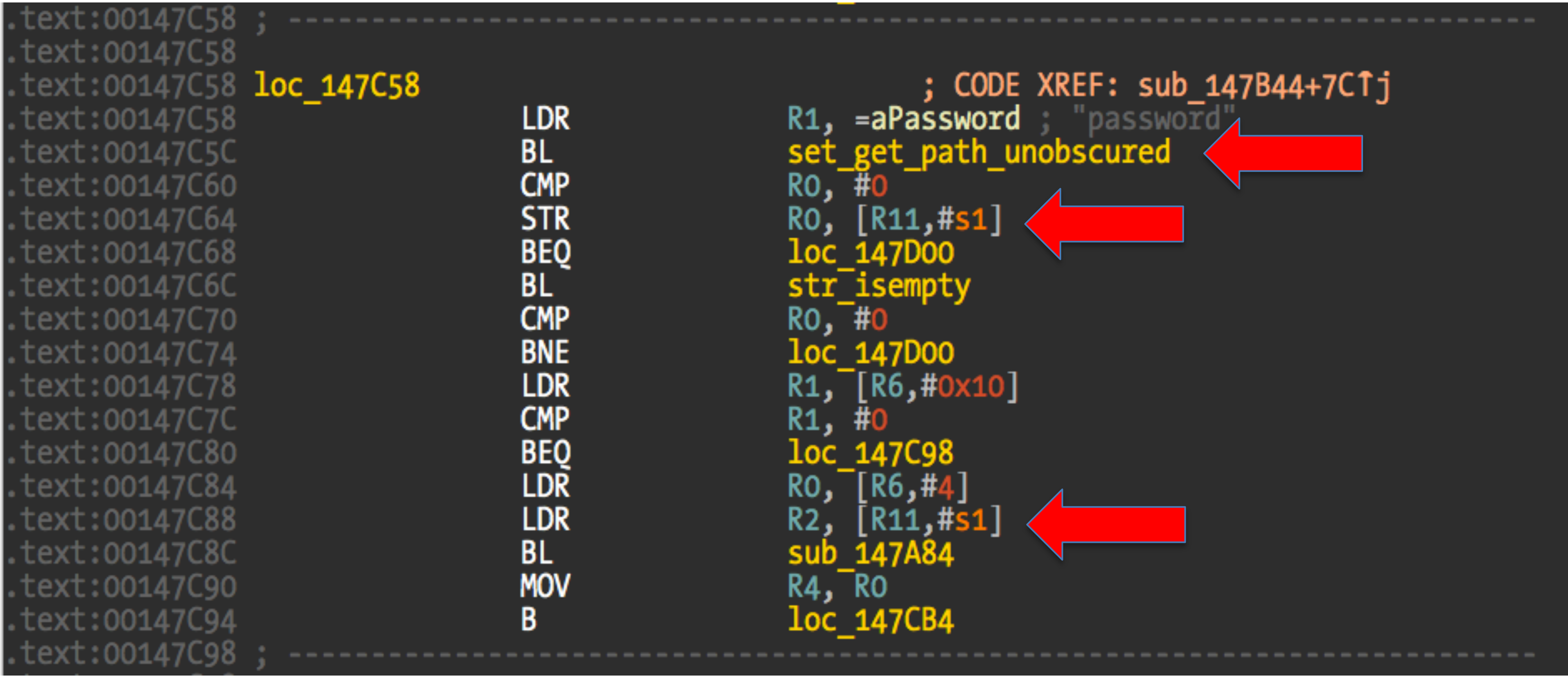
```
----------------------------------------------------------------------------[regs]
  R0: 0x004010B0  R1: 0x12EC2CBB  R2:  0x00439B38  R3:  0x00000031
  R4: 0x0EB3B93C  R5: 0x002F8AF0  R6:  0x0EB3B93C  R7:  0x00000000
  R8: 0x0EB3B974  R9: 0x0030B9D0  R10: 0x00400CF8  R11: 0x0EB3B8AC
  R12: 0x001D2F3C
  SP: 0x0EB3B868  LR: 0x00147C70  PC:  0x00147C8C
----------------------------------------------------------------------------[code]
=> 0x147c8c:    bl      0x147a84
   0x147c90:    mov     r4, r0
   0x147c94:    b       0x147cb4
   0x147c98:    ldr     r1, [r6, #4]
   0x147c9c:    cmp     r1, #0, 0
   0x147ca0:    beq     0x147cbc
   0x147ca4:    ldr     r0, [r11, #-48] ; 0xffffffd0
   0x147ca8:    bl      0x15a98 <strcasecmp@plt>
----------------------------------------------------------------------------------

Breakpoint 10, 0x00147c8c in ?? ()
gdb$ x/s $r0
0x4010b0:        "fba52cb0ab79012757ea0cb7daf6934d"
gdb$ x/s $r2
0x439b38:        "123456"
gdb$ 
```

Reverse engineering

- MD5 is the hash generated at the browser.
- R2 contains the **good plaintext** password.
- We just need to breakpoint one instruction before previous patch and we can recover the original password for any account.

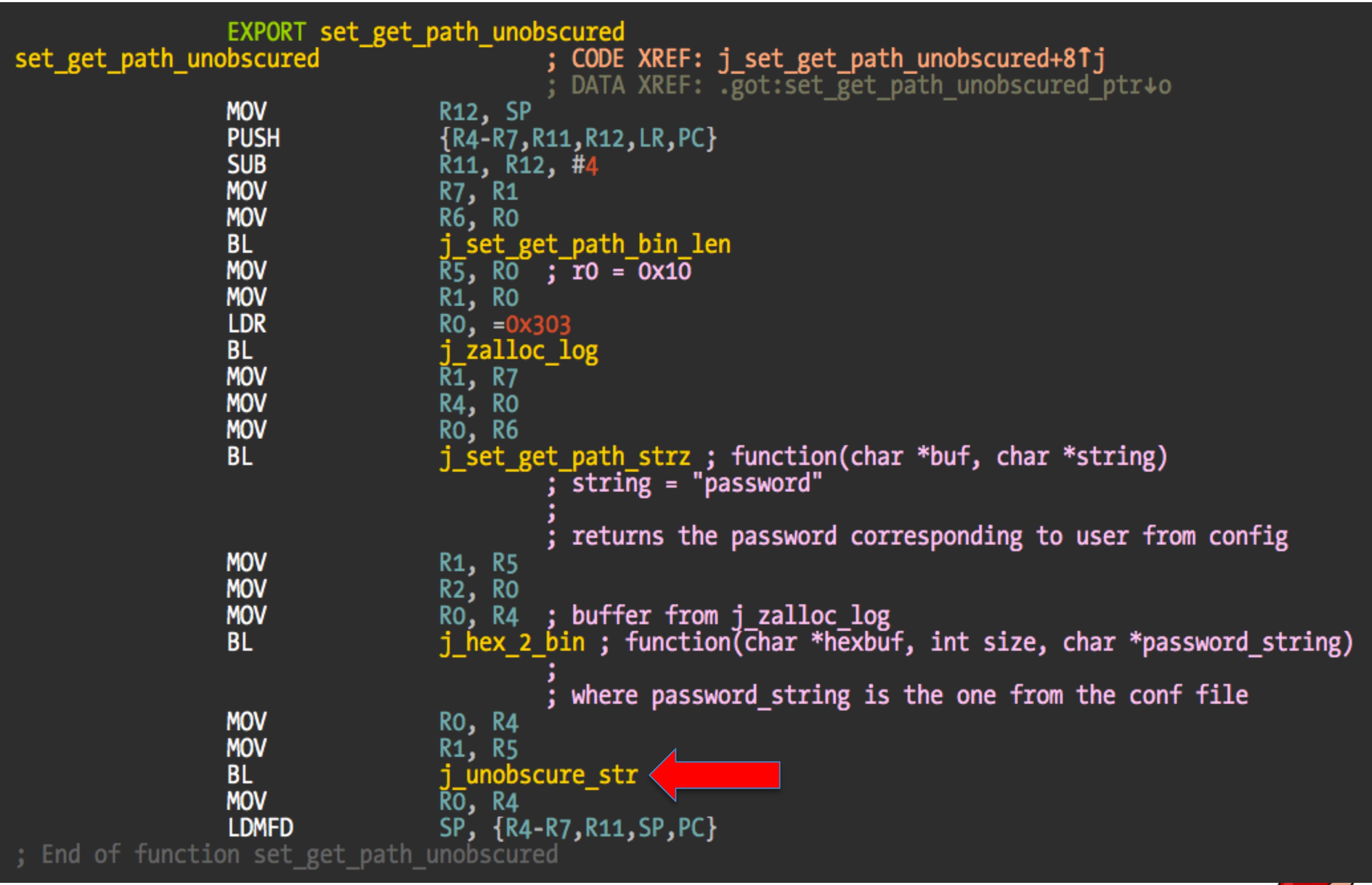


WTF?

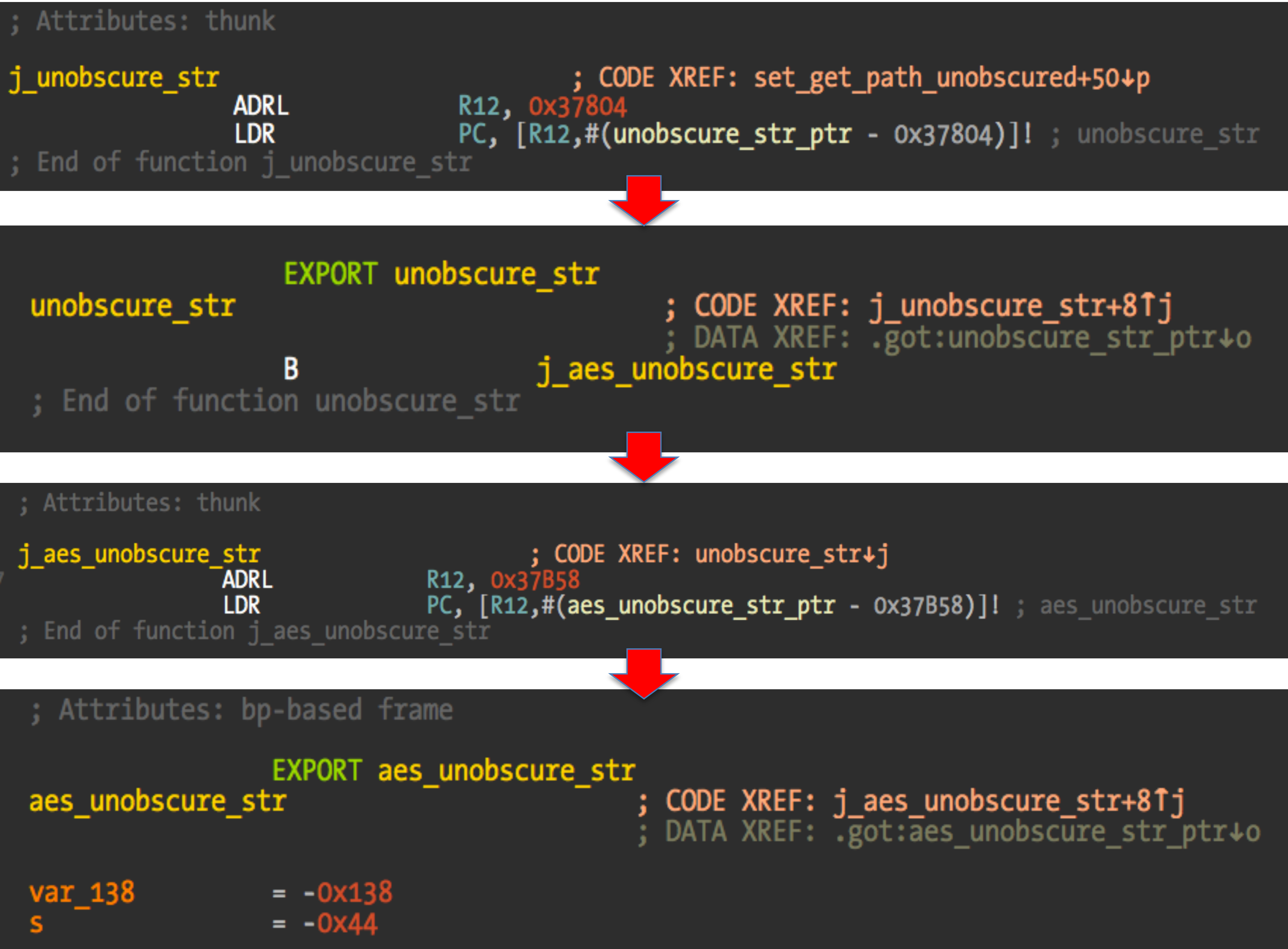


```
.text:00147C58 ; ---------------------------------------------------------------
.text:00147C58
.text:00147C58 loc_147C58                              ; CODE XREF: sub_147B44+7C↑j
.text:00147C58                 LDR             R1, =aPassword ; "password"
.text:00147C5C                 BL              set_get_path_unobscured
.text:00147C60                 CMP             R0, #0
.text:00147C64                 STR             R0, [R11,#s1]
.text:00147C68                 BEQ             loc_147D00
.text:00147C6C                 BL              str_isempty
.text:00147C70                 CMP             R0, #0
.text:00147C74                 BNE             loc_147D00
.text:00147C78                 LDR             R1, [R6,#0x10]
.text:00147C7C                 CMP             R1, #0
.text:00147C80                 BEQ             loc_147C98
.text:00147C84                 LDR             R0, [R6,#4]
.text:00147C88                 LDR             R2, [R11,#s1]
.text:00147C8C                 BL              sub_147A84
.text:00147C90                 MOV             R4, R0
.text:00147C94                 B               loc_147CB4
.text:00147C98 ; ---------------------------------------------------------------
```

- It seems the plaintext password is retrieved inside set_get_path_unobscured.
- This is an imported function from libjutil.so.
- Retrieves the password from configuration and decrypts it.



```
                EXPORT set_get_path_unobscured
set_get_path_unobscured                 ; CODE XREF: j_set_get_path_unobscured+8↑j
                                        ; DATA XREF: .got:set_get_path_unobscured_ptr↓o
                MOV             R12, SP
                PUSH            {R4-R7,R11,R12,LR,PC}
                SUB             R11, R12, #4
                MOV             R7, R1
                MOV             R6, R0
                BL              j_set_get_path_bin_len
                MOV             R5, R0  ; r0 = 0x10
                MOV             R1, R0
                LDR             R0, =0x303
                BL              j_zalloc_log
                MOV             R1, R7
                MOV             R4, R0
                MOV             R0, R6
                BL              j_set_get_path_strz ; function(char *buf, char *string)
                                        ; string = "password"
                                        ;
                                        ; returns the password corresponding to user from config
                MOV             R1, R5
                MOV             R2, R0
                MOV             R0, R4  ; buffer from j_zalloc_log
                BL              j_hex_2_bin ; function(char *hexbuf, int size, char *password_string)
                                        ;
                                        ; where password_string is the one from the conf file
                MOV             R0, R4
                MOV             R1, R5
                BL              j_unobscure_str
                MOV             R0, R4
                LDMFD           SP, {R4-R7,R11,SP,PC}
; End of function set_get_path_unobscured
```

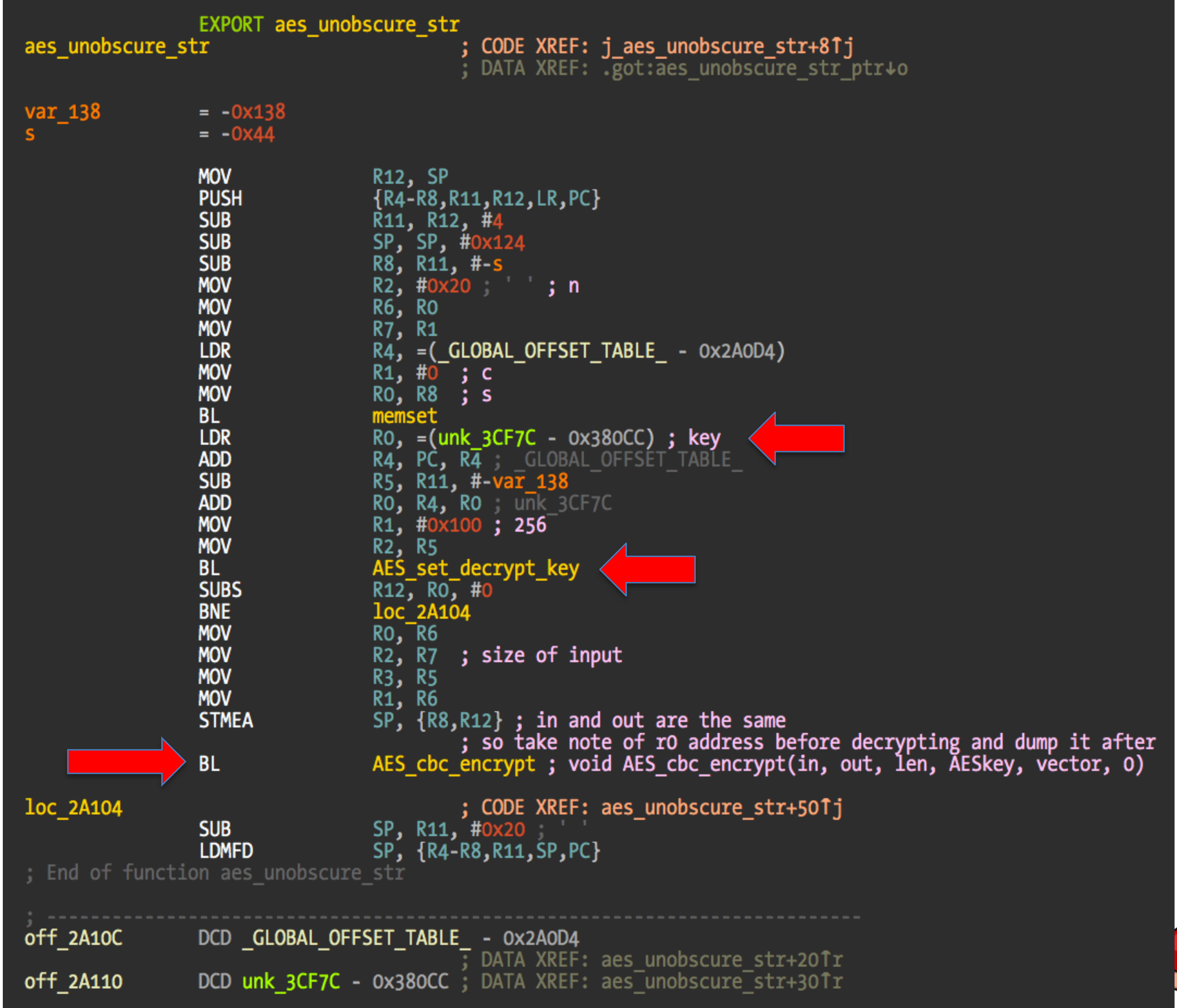


```
; Attributes: thunk

j_unobscure_str                         ; CODE XREF: set_get_path_unobscured+50↓p
                ADRL            R12, 0x37804
                LDR             PC, [R12,#(unobscure_str_ptr - 0x37804)]! ; unobscure_str
; End of function j_unobscure_str
```

```
                EXPORT unobscure_str
unobscure_str                           ; CODE XREF: j_unobscure_str+8↑j
                                        ; DATA XREF: .got:unobscure_str_ptr↓o
                B               j_aes_unobscure_str
; End of function unobscure_str
```

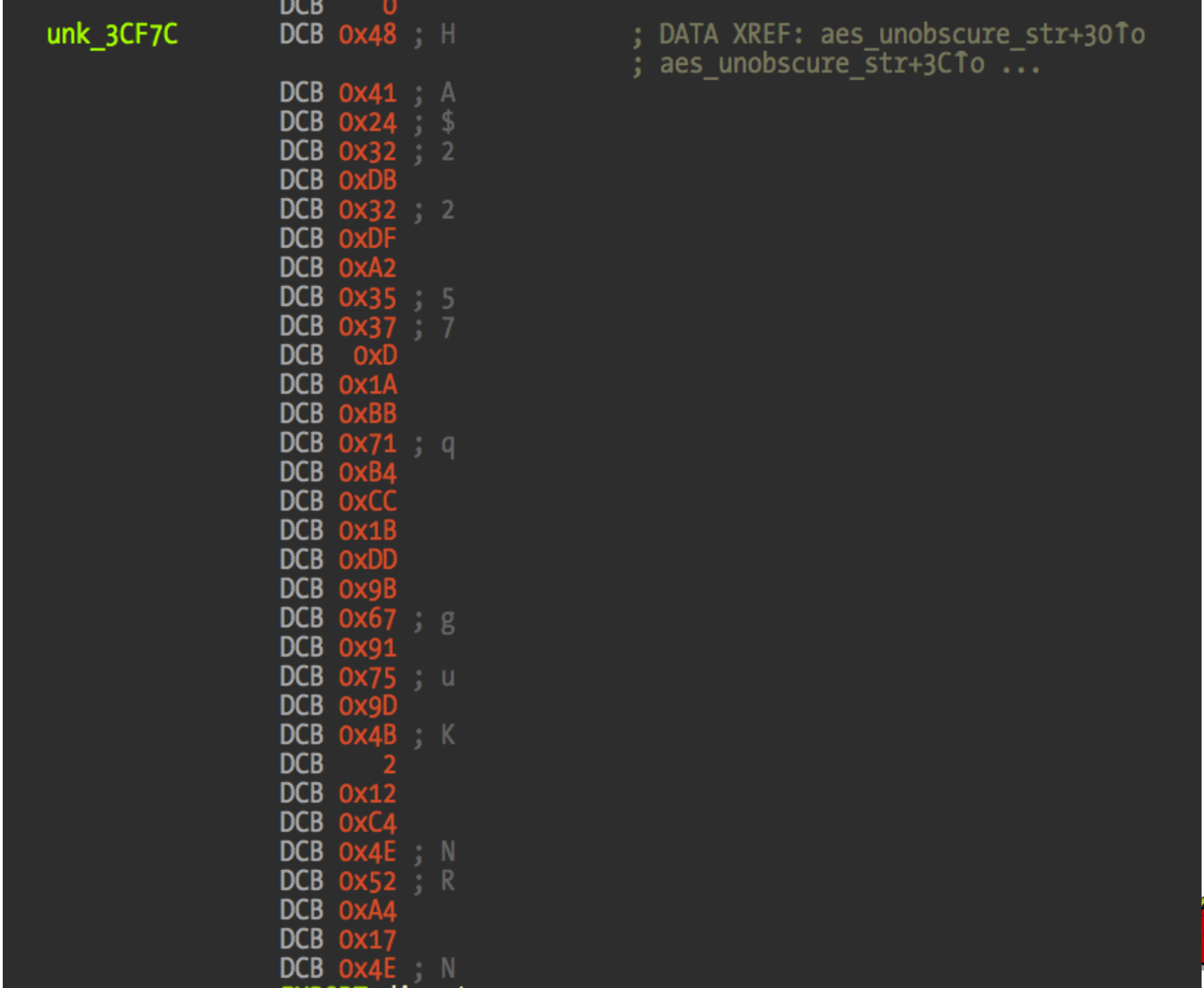
```
; Attributes: thunk

j_aes_unobscure_str                     ; CODE XREF: unobscure_str↓j
                ADRL            R12, 0x37B58
                LDR             PC, [R12,#(aes_unobscure_str_ptr - 0x37B58)]! ; aes_unobscure_str
; End of function j_aes_unobscure_str
```

```
; Attributes: bp-based frame

                EXPORT aes_unobscure_str
aes_unobscure_str                       ; CODE XREF: j_aes_unobscure_str+8↑j
                                        ; DATA XREF: .got:aes_unobscure_str_ptr↓o

var_138         = -0x138
s               = -0x44
```



```
                EXPORT aes_unobscure_str
aes_unobscure_str                       ; CODE XREF: j_aes_unobscure_str+8↑j
                                        ; DATA XREF: .got:aes_unobscure_str_ptr↓o

var_138         = -0x138
s               = -0x44

                MOV             R12, SP
                PUSH            {R4-R8,R11,R12,LR,PC}
                SUB             R11, R12, #4
                SUB             SP, SP, #0x124
                SUB             R8, R11, #-s
                MOV             R2, #0x20 ; ' ' ; n
                MOV             R6, R0
                MOV             R7, R1
                LDR             R4, =(_GLOBAL_OFFSET_TABLE_ - 0x2A0D4)
                MOV             R1, #0  ; c
                MOV             R0, R8  ; s
                BL              memset
                LDR             R0, =(unk_3CF7C - 0x380CC) ; key
                ADD             R4, PC, R4 ; _GLOBAL_OFFSET_TABLE_
                SUB             R5, R11, #-var_138
                ADD             R0, R4, R0 ; unk_3CF7C
                MOV             R1, #0x100 ; 256
                MOV             R2, R5
                BL              AES_set_decrypt_key
                SUBS            R12, R0, #0
                BNE             loc_2A104
                MOV             R0, R6
                MOV             R2, R7  ; size of input
                MOV             R3, R5
                MOV             R1, R6
                STMEA           SP, {R8,R12} ; in and out are the same
                                        ; so take note of r0 address before decrypting and dump it after
                BL              AES_cbc_encrypt ; void AES_cbc_encrypt(in, out, len, AESkey, vector, 0)

loc_2A104                               ; CODE XREF: aes_unobscure_str+50↑j
                SUB             SP, R11, #0x20 ; ' '
                LDMFD           SP, {R4-R8,R11,SP,PC}
; End of function aes_unobscure_str

; ---------------------------------------------------------------------------
off_2A10C       DCD _GLOBAL_OFFSET_TABLE_ - 0x2A0D4
                                        ; DATA XREF: aes_unobscure_str+20↑r
off_2A110       DCD unk_3CF7C - 0x380CC ; DATA XREF: aes_unobscure_str+30↑r
```

Game over!



```
                DCB    0
unk_3CF7C       DCB 0x48 ; H               ; DATA XREF: aes_unobscure_str+30↑o
                                           ; aes_unobscure_str+3C↑o ...
                DCB 0x41 ; A
                DCB 0x24 ; $
                DCB 0x32 ; 2
                DCB 0xDB
                DCB 0x32 ; 2
                DCB 0xDF
                DCB 0xA2
                DCB 0x35 ; 5
                DCB 0x37 ; 7
                DCB  0xD
                DCB 0x1A
                DCB 0xBB
                DCB 0x71 ; q
                DCB 0xB4
                DCB 0xCC
                DCB 0x1B
                DCB 0xDD
                DCB 0x9B
                DCB 0x67 ; g
                DCB 0x91
                DCB 0x75 ; u
                DCB 0x9D
                DCB 0x4B ; K
                DCB    2
                DCB 0x12
                DCB 0xC4
                DCB 0x4E ; N
                DCB 0x52 ; R
                DCB 0xA4
                DCB 0x17
                DCB 0x4E ; N
```

Reverse engineering

- We recovered the encryption key.
- We can use OpenSSL to decrypt any passwords.
- Just pay attention that the values in configuration files are the content bytes.

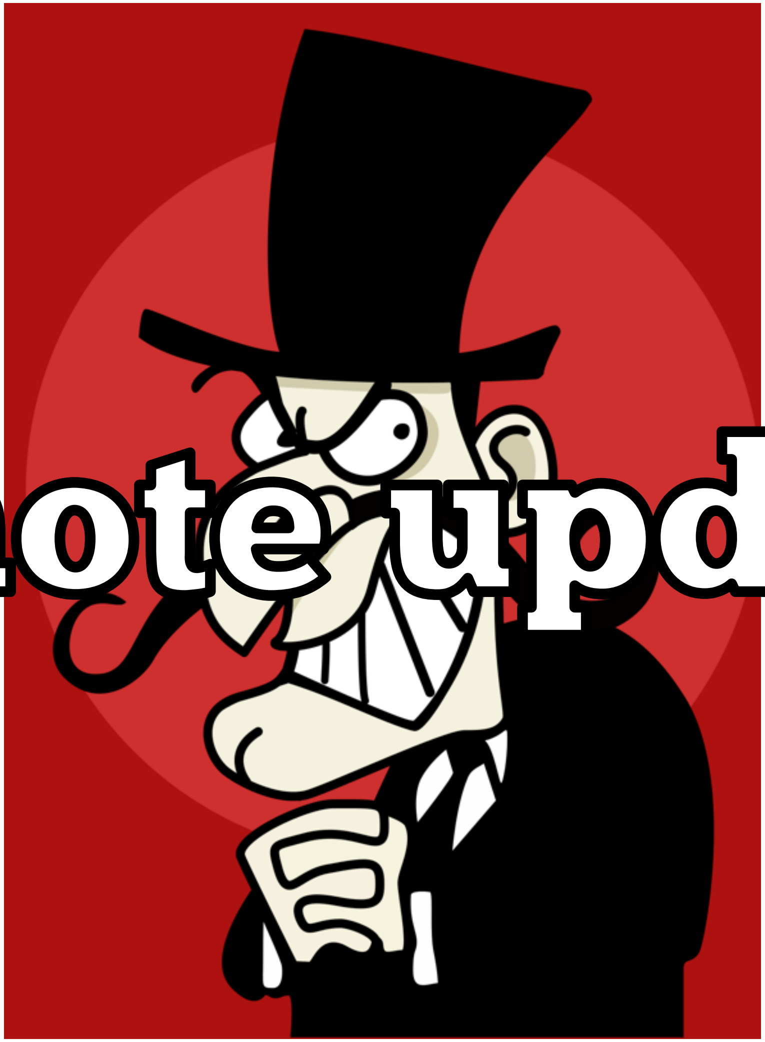
```
$ KEY="48412432DB32DFA235370D1ABB71B4CC1BDD9B6791759D4B0212C44E52A4174E"

$ PASS="c72bd3a6528fb5e3c3e1dfa882fffed0"
$ echo $PASS | xxd -r -p | openssl enc -aes-256-cbc -d -K $KEY -iv 0 -nopad
123456

$ PASS="3c65360ec5ed0e9ce38a2a20ae816358"
$ echo $PASS | xxd -r -p | openssl enc -aes-256-cbc -d -K $KEY -iv 0 -nopad
password

$ PASS="fb5438bcd8a8a226240de52c3bf03633"
$ echo $PASS | xxd -r -p | openssl enc -aes-256-cbc -d -K $KEY -iv 0 -nopad
zonnet

$ PASS="1380ed296134ad56f9e879c97956f90a"
$ echo $PASS | xxd -r -p | openssl enc -aes-256-cbc -d -K $KEY -iv 0 -nopad
72151950

$ PASS="1721cffdd6fa0354079963f1f67c0f51"
$ echo $PASS | xxd -r -p | openssl enc -aes-256-cbc -d -K $KEY -iv 0 -nopad
acs
```

FAIL

Reverse engineering

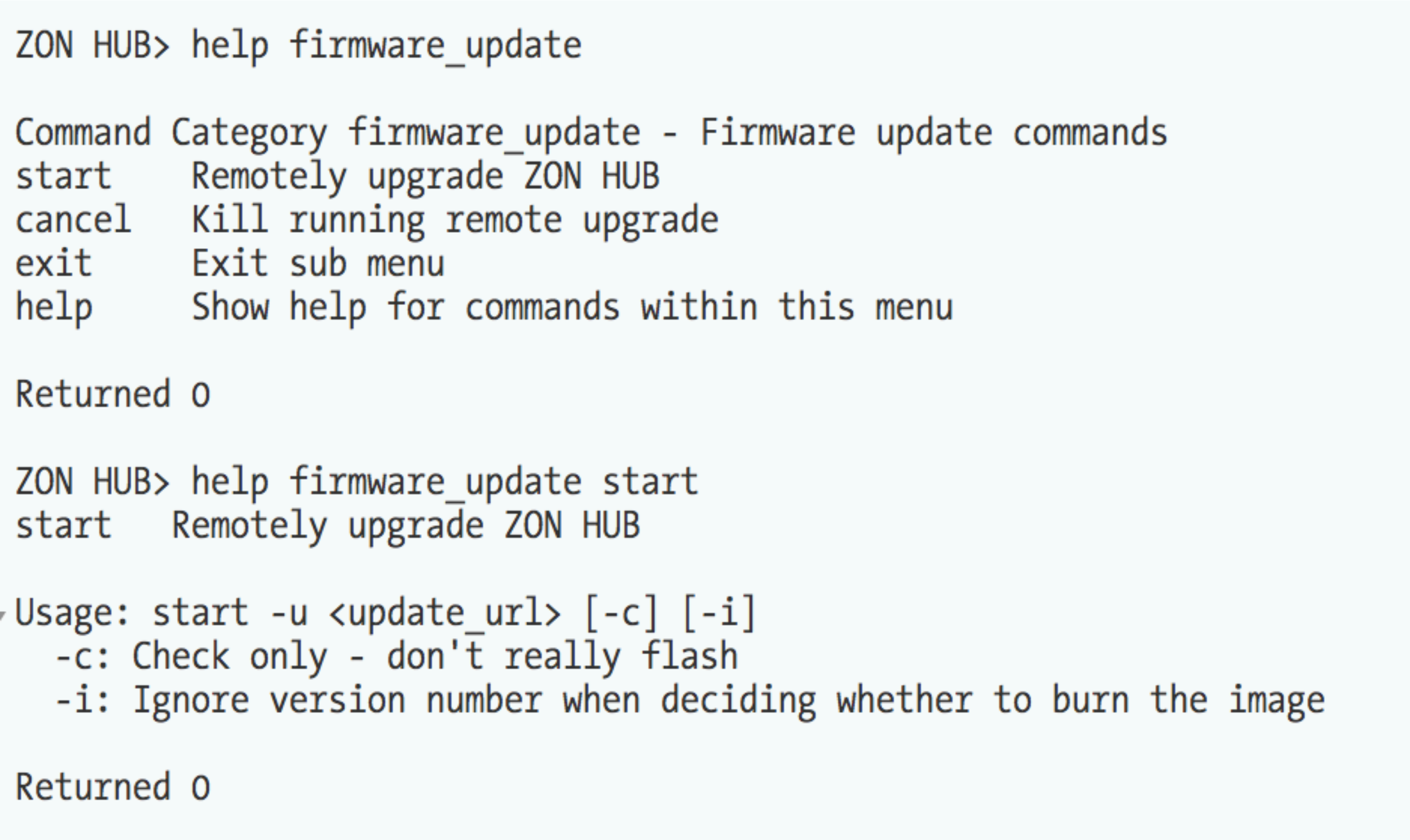
- No idea why they (Jungo? NOS?) have done it this way.
- It’s just **dumb**.
- At least hashes you might need some computing power to break.

Remote updates

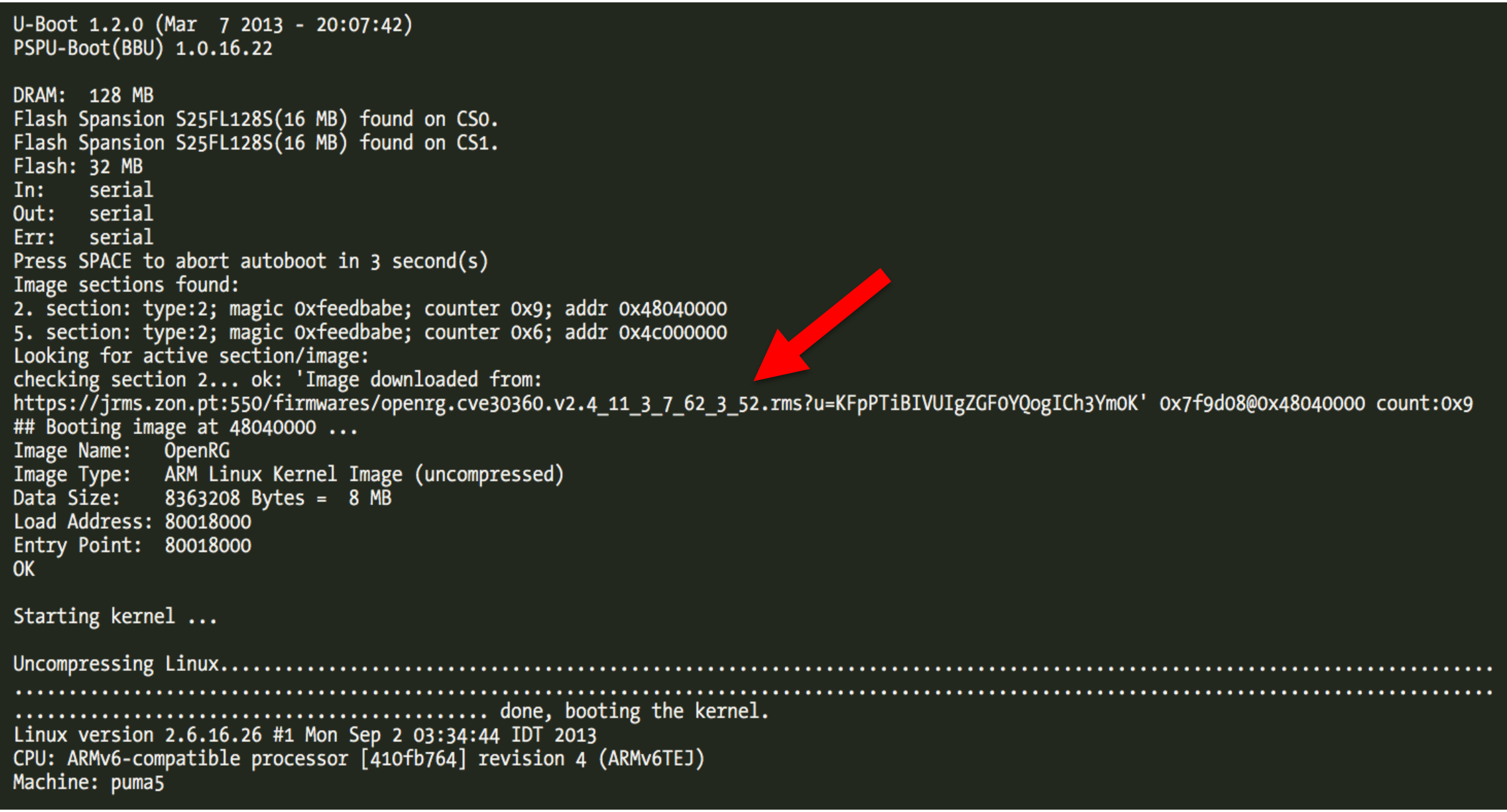


Remote Updates

- NOS has the ability to push remote updates.
- Are they reasonably secure?
- Can we MiTM?
- We can play around locally without DNS tricks.



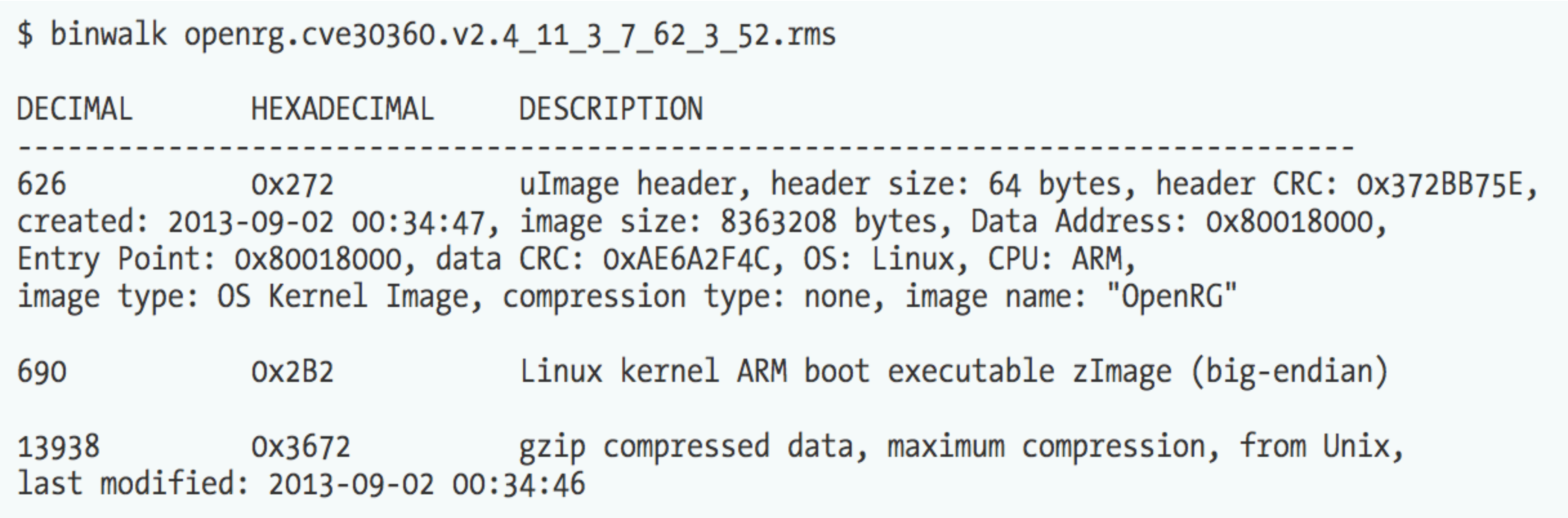
```
ZON HUB> help firmware_update

Command Category firmware_update - Firmware update commands
start    Remotely upgrade ZON HUB
cancel   Kill running remote upgrade
exit     Exit sub menu
help     Show help for commands within this menu

Returned 0

ZON HUB> help firmware_update start
start   Remotely upgrade ZON HUB

Usage: start -u <update_url> [-c] [-i]
  -c: Check only - don't really flash
  -i: Ignore version number when deciding whether to burn the image

Returned 0
```



```
U-Boot 1.2.0 (Mar  7 2013 - 20:07:42)
PSPU-Boot(BBU) 1.0.16.22

DRAM:  128 MB
Flash Spansion S25FL128S(16 MB) found on CS0.
Flash Spansion S25FL128S(16 MB) found on CS1.
Flash: 32 MB
In:    serial
Out:   serial
Err:   serial
Press SPACE to abort autoboot in 3 second(s)
Image sections found:
2. section: type:2; magic 0xfeedbabe; counter 0x9; addr 0x48040000
5. section: type:2; magic 0xfeedbabe; counter 0x6; addr 0x4c000000
Looking for active section/image:
checking section 2... ok: 'Image downloaded from:
https://jrms.zon.pt:550/firmwares/openrg.cve30360.v2.4_11_3_7_62_3_52.rms?u=KFpPTiBIVUIgZGFOYQogICh3YmOK' 0x7f9d08@0x48040000 count:0x9
## Booting image at 48040000 ...
Image Name:   OpenRG
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    8363208 Bytes =  8 MB
Load Address: 80018000
Entry Point:  80018000
OK

Starting kernel ...

Uncompressing Linux.......................................................................................................................
..........................................................................................................................................
............................................ done, booting the kernel.
Linux version 2.6.16.26 #1 Mon Sep 2 03:34:44 IDT 2013
CPU: ARMv6-compatible processor [410fb764] revision 4 (ARMv6TEJ)
Machine: puma5
```

Remote Updates

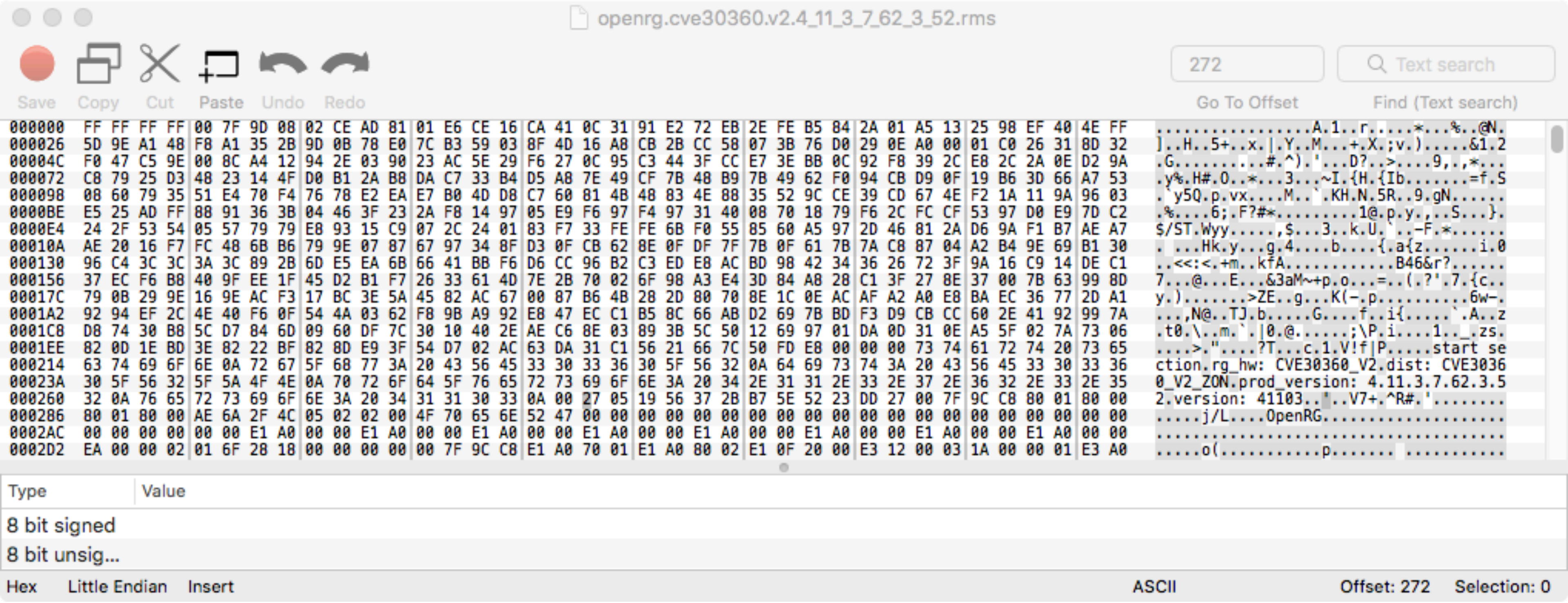
- Updates are delivered in a RMS binary file.



```
$ binwalk openrg.cve30360.v2.4_11_3_7_62_3_52.rms

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
626           0x272           uImage header, header size: 64 bytes, header CRC: 0x372BB75E,
created: 2013-09-02 00:34:47, image size: 8363208 bytes, Data Address: 0x80018000,
Entry Point: 0x80018000, data CRC: 0xAE6A2F4C, OS: Linux, CPU: ARM,
image type: OS Kernel Image, compression type: none, image name: "OpenRG"

690           0x2B2           Linux kernel ARM boot executable zImage (big-endian)

13938         0x3672          gzip compressed data, maximum compression, from Unix,
last modified: 2013-09-02 00:34:46
```

Remote Updates

- Contains the kernel and data that we have already seen in flash dumps.
- But also what seems to be some kind of header.
- ALWAYS LOAD BINARY DATA INTO AN HEX EDITOR!



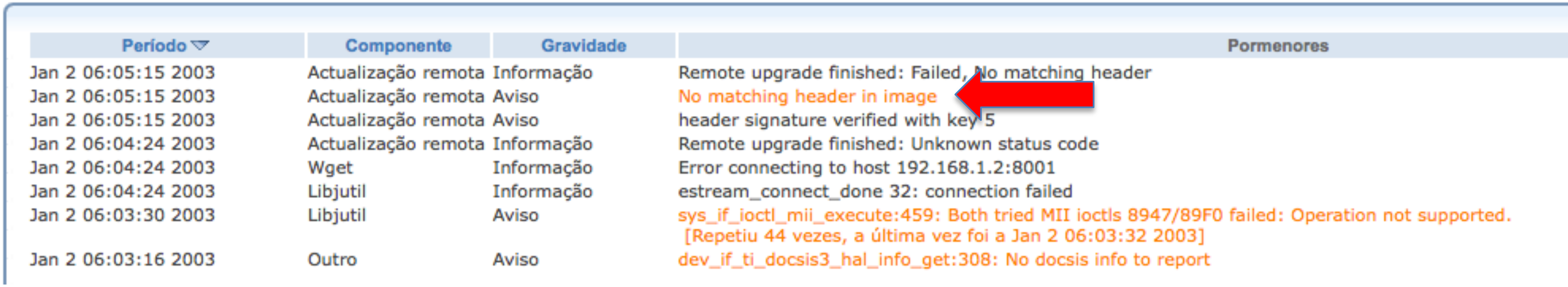
```
000000  FF FF FF FF 00 7F 9D 08 02 CE AD 81 01 E6 CE 16 CA 41 0C 31 91 E2 72 EB 2E FE B5 84 2A 01 A5 13 25 98 EF 40 4E FF   ....................A.1..r.....*....%..@N.
000026  5D 9E A1 48 F8 A1 35 2B 9D 0B 78 E0 7C B3 59 03 8F 4D 16 A8 CB 2B CC 58 07 3B 76 D0 29 0E A0 00 01 C0 26 31 8D 32   ]..H..5+..x.|.Y..M...+.X.;v.).....&1.2
00004C  F0 47 C5 9E 00 8C A4 12 94 2E 03 90 23 AC 5E 29 F6 27 0C 95 C3 44 3F CC E7 3E BB 0C 92 F8 39 2C E8 2C 2A 0E D2 9A   .G.........#.^).'....D?..>.....9,.,*...
000072  C8 79 25 D3 48 23 14 4F D0 B1 2A B8 DA C7 33 B4 D5 A8 7E 49 CF 7B 48 B9 7B 49 62 F0 94 CB D9 0F 19 B6 3D 66 A7 53   .y%.H#.O..*....3....~I.{H.{Ib.........=f.S
000098  08 60 79 35 51 E4 70 F4 76 78 E2 EA E7 B0 4D D8 C7 60 81 4B 48 83 4E 88 35 52 9C CE 39 CD 67 4E F2 1A 11 9A 96 03   .`y5Q.p.vx.....M...`.KH.N.5R..9.gN......
0000BE  E5 25 AD FF 88 91 36 3B 04 46 3F 23 2A F8 14 97 05 E9 F6 97 F4 97 31 40 08 70 18 79 F6 2C FC CF 53 97 D0 E9 7D C2   .%.....6;.F?#*..........1@.p.y.,..S...}.
0000E4  24 2F 53 54 05 57 79 79 E8 93 15 C9 07 2C 24 01 83 F7 33 FE FE 6B F0 55 85 60 A5 97 2D 46 81 2A D6 9A F1 B7 AE A7   $/ST.Wyy......,$....3..k.U.`...-F.*.......
00010A  AE 20 16 F7 FC 48 6B B6 79 9E 07 87 67 97 34 8F D3 0F CB 62 8E 0F DF 7F 7B 0F 61 7B 7A C8 87 04 A2 B4 9E 69 B1 30   . ...Hk.y...g.4....b....{.a{z.......i.0
000130  96 C4 3C 3C 3A 3C 89 2B 6D E5 EA 6B 66 41 BB F6 D6 CC 96 B2 C3 ED E8 AC BD 98 42 34 36 26 72 3F 9A 16 C9 14 DE C1   ..<<:<.+m..kfA............B46&r?......
000156  37 EC F6 B8 40 9F EE 1F 45 D2 B1 F7 26 33 61 4D 7E 2B 70 02 6F 98 A3 E4 3D 84 A8 28 C1 3F 27 8E 37 00 7B 63 99 8D   7...@...E...&3aM~+p.o....=..(.?'.7.{c..
00017C  79 0B 29 9E 16 9E AC F3 17 BC 3E 5A 45 82 AC 67 00 87 B6 4B 28 2D 80 70 8E 1C 0E AC AF A2 A0 E8 BA EC 36 77 2D A1   y.)........>ZE..g...K(-.p............6w-.
0001A2  92 94 EF 2C 4E 40 F6 0F 54 4A 03 62 F8 9B A9 92 E8 47 EC C1 B5 8C 66 AB D2 69 7B BD F3 D9 CB CC 60 2E 41 92 99 7A   ...,N@..TJ.b......G.....f..i{......`.A..z
0001C8  D8 74 30 B8 5C D7 84 6D 09 60 DF 7C 30 10 40 2E AE C6 8E 03 89 3B 5C 50 12 69 97 01 DA 0D 31 0E A5 5F 02 7A 73 06   .t0.\..m.`.|0.@.......;\P.i.....1.._.zs.
0001EE  82 0D 1E BD 3E 82 22 BF 82 8D E9 3F 54 D7 02 AC 63 DA 31 C1 56 21 66 7C 50 FD E8 00 00 00 73 74 61 72 74 20 73 65   ....>."....?T....c.1.V!f|P......start se
000214  63 74 69 6F 6E 0A 72 67 5F 68 77 3A 20 43 56 45 33 30 33 36 30 5F 56 32 0A 64 69 73 74 3A 20 43 56 45 33 30 33 36   ction.rg_hw: CVE30360_V2.dist: CVE3036
00023A  30 5F 56 32 5F 5A 4F 4E 0A 70 72 6F 64 5F 76 65 72 73 69 6F 6E 3A 20 34 2E 31 31 2E 33 2E 37 2E 36 32 2E 33 2E 35   0_V2_ZON.prod_version: 4.11.3.7.62.3.5
000260  32 0A 76 65 72 73 69 6F 6E 3A 20 34 31 31 30 33 0A 00 27 05 19 56 37 2B B7 5E 52 23 DD 27 00 7F 9C C8 80 01 80 00   2.version: 41103..'..V7+.^R#.'.........
000286  80 01 80 00 AE 6A 2F 4C 05 02 02 00 4F 70 65 6E 52 47 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ......j/L.....OpenRG....................
0002AC  00 00 00 00 00 00 E1 A0 00 00 E1 A0 00 00 E1 A0 00 00 E1 A0 00 00 E1 A0 00 00 E1 A0 00 00 E1 A0 00 00 E1 A0 00 00   ......................................
0002D2  EA 00 00 02 01 6F 28 18 00 00 00 00 00 7F 9C C8 E1 A0 70 01 E1 A0 80 02 E1 0F 20 00 E3 12 00 03 1A 00 00 01 E3 A0   ......o(............p....... ..........
```

Type	Value
8 bit signed	
8 bit unsig...	

Hex   Little Endian   Insert   ASCII   Offset: 272   Selection: 0

Remote Updates

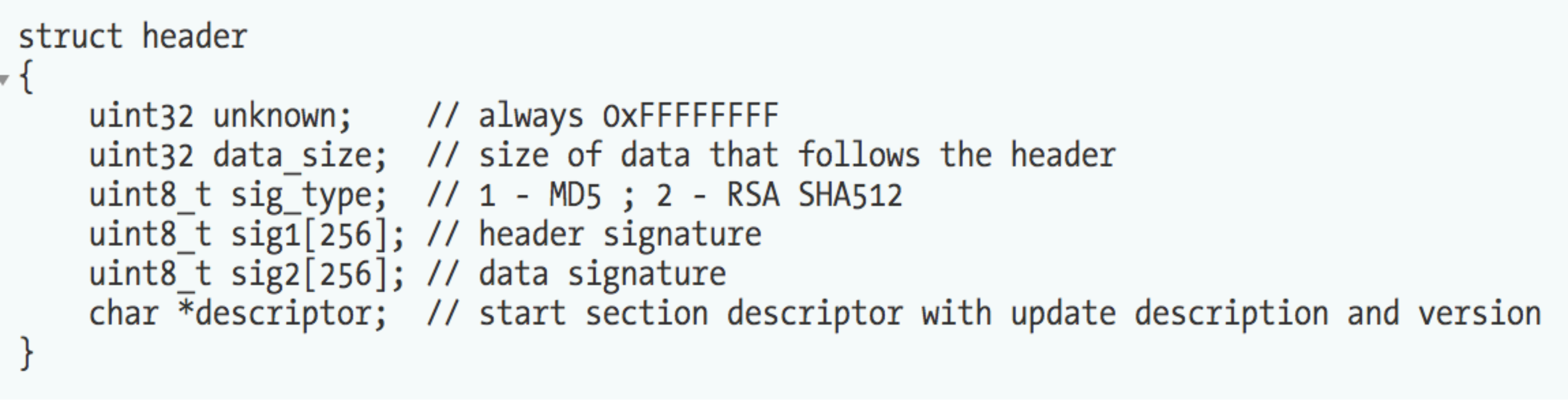
- Upgrade function at **sub_B4410**.
- Found it via log messages.



Período	Componente	Gravidade	Pormenores
Jan 2 06:05:15 2003	Actualização remota	Informação	Remote upgrade finished: Failed, No matching header
Jan 2 06:05:15 2003	Actualização remota	Aviso	No matching header in image
Jan 2 06:05:15 2003	Actualização remota	Aviso	header signature verified with key 5
Jan 2 06:04:24 2003	Actualização remota	Informação	Remote upgrade finished: Unknown status code
Jan 2 06:04:24 2003	Wget	Informação	Error connecting to host 192.168.1.2:8001
Jan 2 06:04:24 2003	Libjutil	Informação	estream_connect_done 32: connection failed
Jan 2 06:03:30 2003	Libjutil	Aviso	sys_if_ioctl_mii_execute:459: Both tried MII ioctls 8947/89F0 failed: Operation not supported. [Repetiu 44 vezes, a última vez foi a Jan 2 06:03:32 2003]
Jan 2 06:03:16 2003	Outro	Aviso	dev_if_ti_docsis3_hal_info_get:308: No docsis info to report

Remote Updates

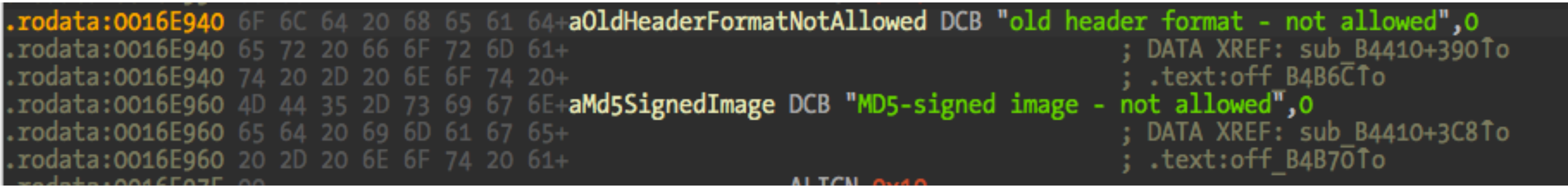
- Poking around we are able to define an header structure.



```
struct header
{
    uint32 unknown;     // always 0xFFFFFFFF
    uint32 data_size;   // size of data that follows the header
    uint8_t sig_type;   // 1 - MD5 ; 2 - RSA SHA512
    uint8_t sig1[256]; // header signature
    uint8_t sig2[256]; // data signature
    char *descriptor;   // start section descriptor with update description and version
}
```

Remote Updates

- MD5 comes from here.

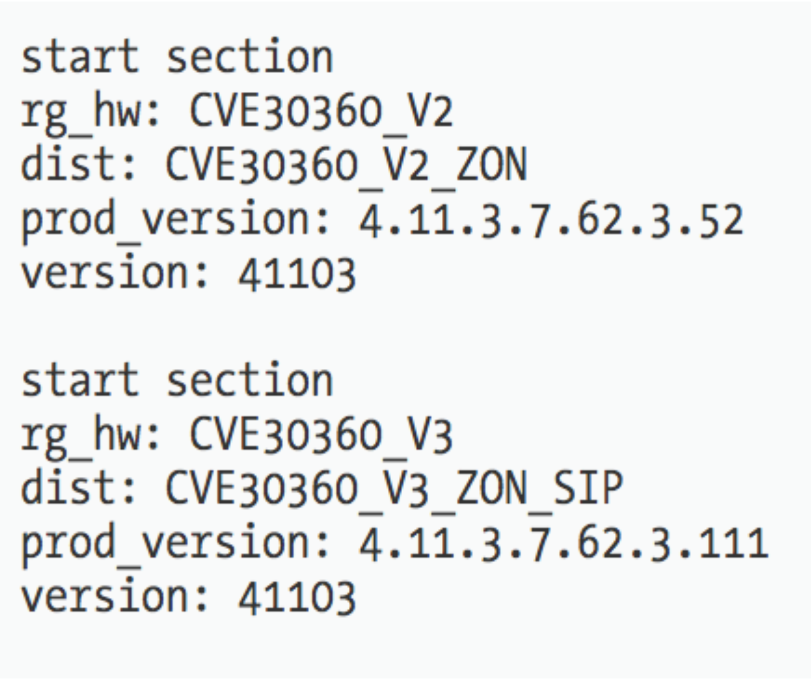


```
.rodata:0016E940 6F 6C 64 20 68 65 61 64+aOldHeaderFormatNotAllowed DCB "old header format - not allowed",0
.rodata:0016E940 65 72 20 66 6F 72 6D 61+                                         ; DATA XREF: sub_B4410+390↑o
.rodata:0016E940 74 20 2D 20 6E 6F 74 20+                                         ; .text:off_B4B6C↑o
.rodata:0016E960 4D 44 35 2D 73 69 67 6E+aMd5SignedImage DCB "MD5-signed image - not allowed",0
.rodata:0016E960 65 64 20 69 6D 61 67 65+                                         ; DATA XREF: sub_B4410+3C8↑o
.rodata:0016E960 20 2D 20 6E 6F 74 20 61+                                         ; .text:off_B4B70↑o
```

- I guess things were quite bad somewhere in the past ☺.

Remote Updates

- The descriptor is just a text string that is parsed for information (commands?).



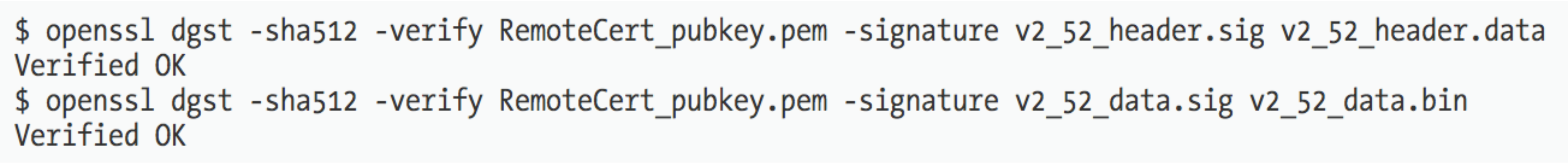
```
start section
rg_hw: CVE30360_V2
dist: CVE30360_V2_ZON
prod_version: 4.11.3.7.62.3.52
version: 41103

start section
rg_hw: CVE30360_V3
dist: CVE30360_V3_ZON_SIP
prod_version: 4.11.3.7.62.3.111
version: 41103
```

Remote Updates

- Two RSA-SHA512 signatures.
- One for the header.
- Another for header + data.
- No idea why?
- Maybe to faster reject updates because of wrong descriptor.

Remote Updates

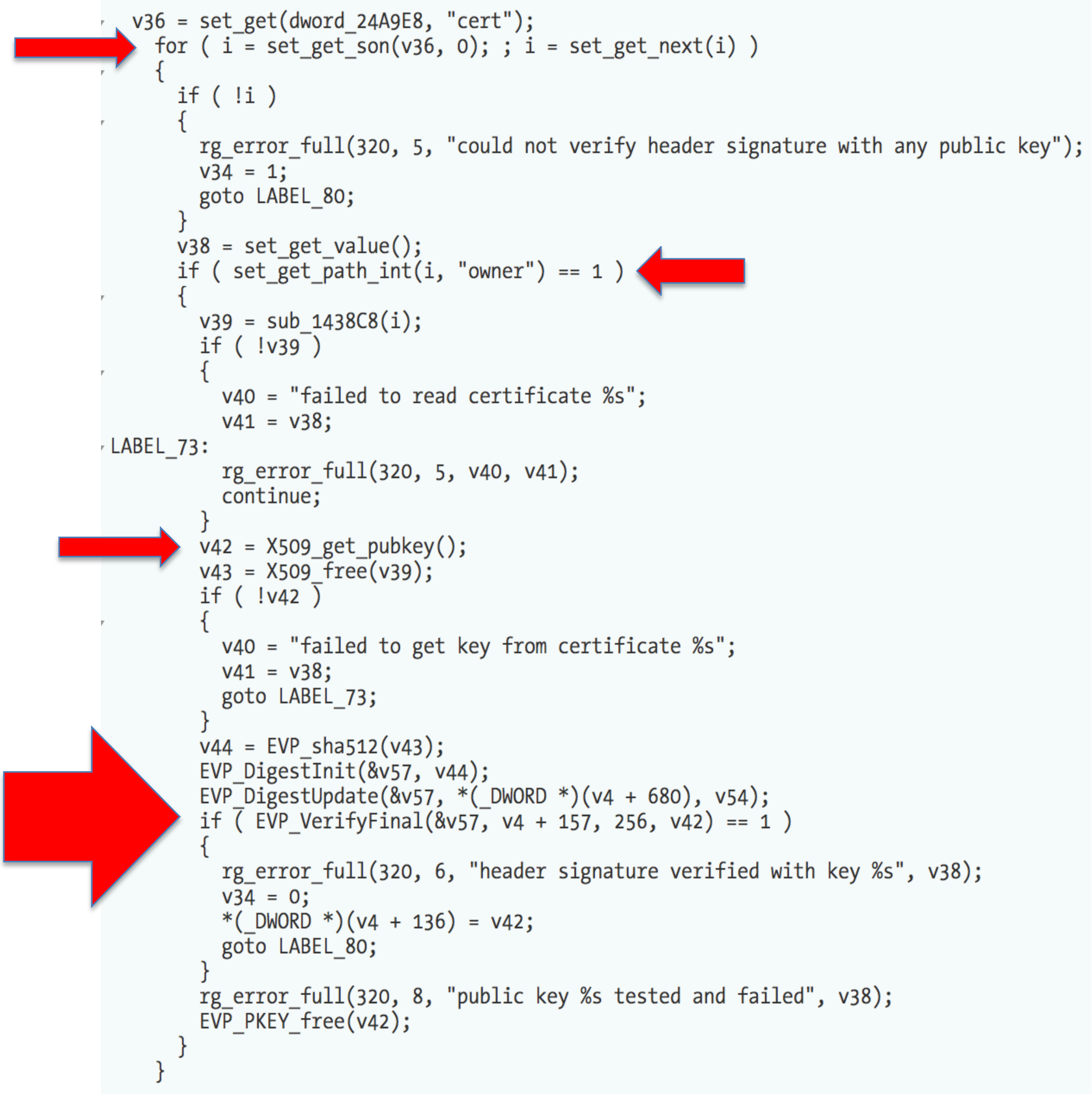
- Remote updates can be signed by any* of the keys available in the configuration file.
- Signed by "ZON HUB remote update" key.
- We can extract it and verify ourselves.



```
$ openssl dgst -sha512 -verify RemoteCert_pubkey.pem -signature v2_52_header.sig v2_52_header.data
Verified OK
$ openssl dgst -sha512 -verify RemoteCert_pubkey.pem -signature v2_52_data.sig v2_52_data.bin
Verified OK
```

Remote Updates

- Remove the “ZON HUB remote update” cert from configuration files.
- No remote updates possible ☺.
- Backdoors -= 1.

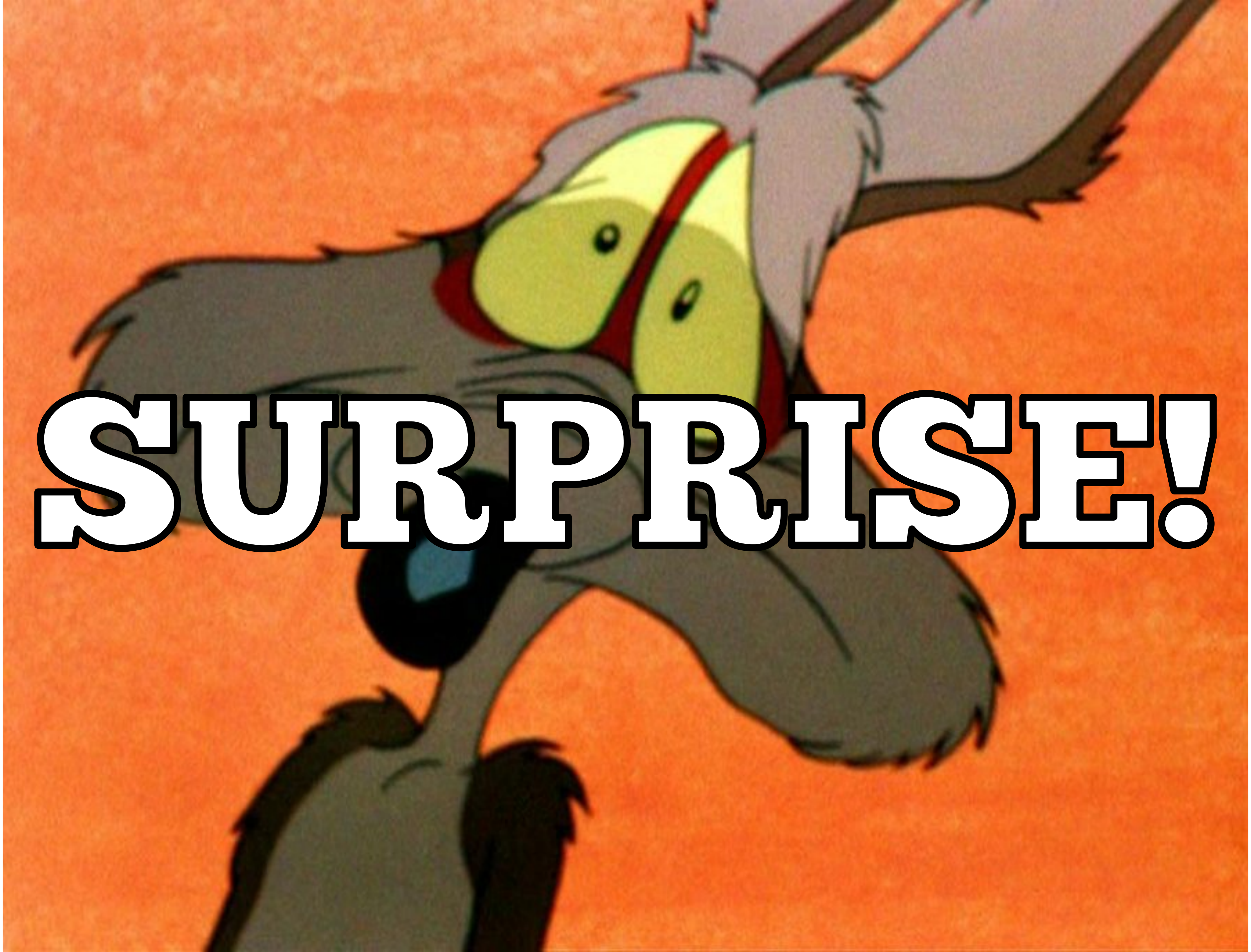


```
  v36 = set_get(dword_24A9E8, "cert");
    for ( i = set_get_son(v36, 0); ; i = set_get_next(i) )
    {
      if ( !i )
      {
        rg_error_full(320, 5, "could not verify header signature with any public key");
        v34 = 1;
        goto LABEL_80;
      }
      v38 = set_get_value();
      if ( set_get_path_int(i, "owner") == 1 )
      {
        v39 = sub_1438C8(i);
        if ( !v39 )
        {
          v40 = "failed to read certificate %s";
          v41 = v38;
LABEL_73:
          rg_error_full(320, 5, v40, v41);
          continue;
        }
        v42 = X509_get_pubkey();
        v43 = X509_free(v39);
        if ( !v42 )
        {
          v40 = "failed to get key from certificate %s";
          v41 = v38;
          goto LABEL_73;
        }
        v44 = EVP_sha512(v43);
        EVP_DigestInit(&v57, v44);
        EVP_DigestUpdate(&v57, *(_DWORD *)(v4 + 680), v54);
        if ( EVP_VerifyFinal(&v57, v4 + 157, 256, v42) == 1 )
        {
          rg_error_full(320, 6, "header signature verified with key %s", v38);
          v34 = 0;
          *(_DWORD *)(v4 + 136) = v42;
          goto LABEL_80;
        }
        rg_error_full(320, 8, "public key %s tested and failed", v38);
        EVP_PKEY_free(v42);
      }
    }
```

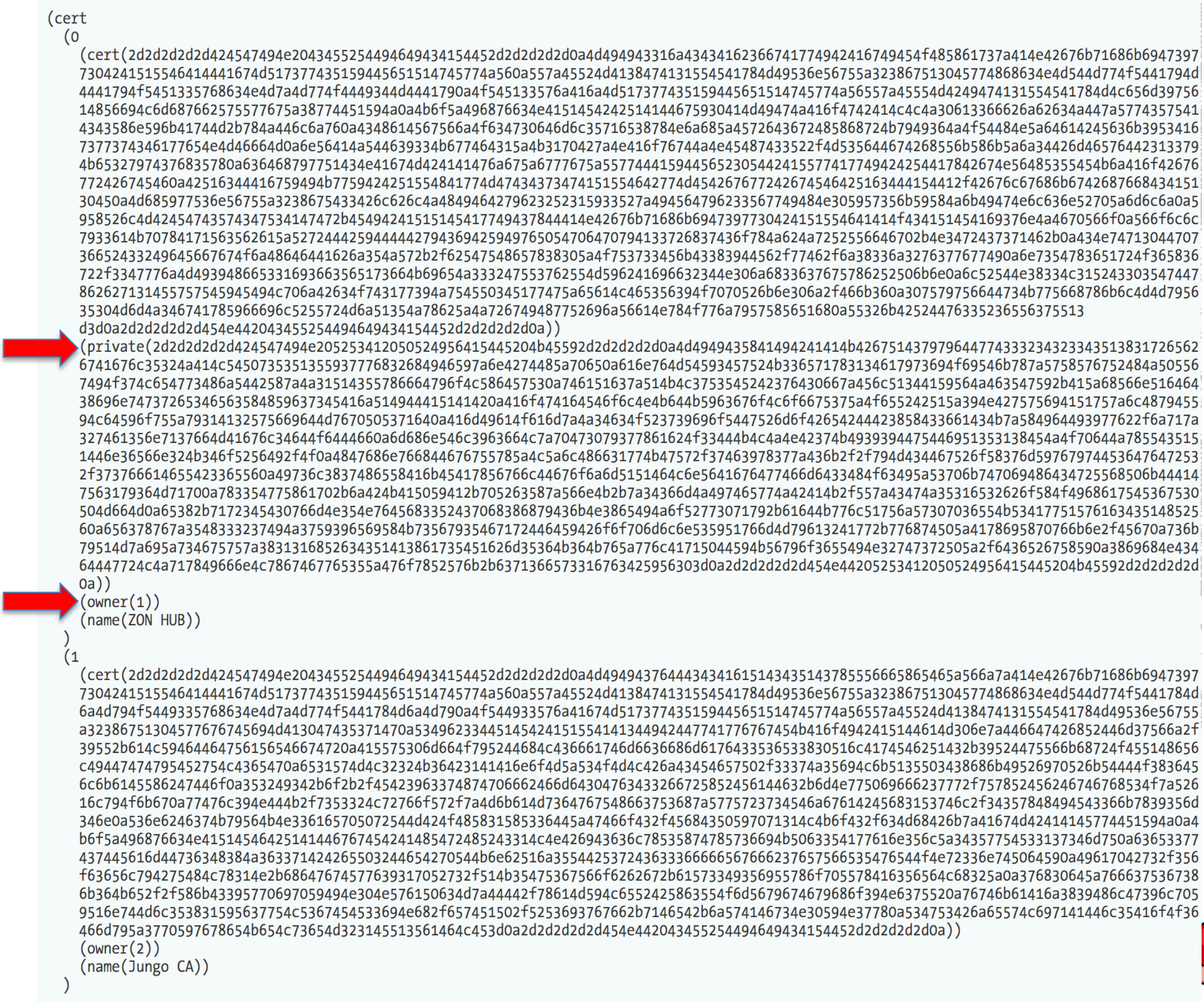
```
(5
  (cert(2d2d2d2d2d424547494e20434552544946494341544452d2d2d2d2d0a4d494943764443434161514351437855566658654a566a7a414e42676b7
  1686b69473977304241515546414441674d5173774351594456515145774a560a557a45524d41384741315545417884d49536e56755a32386751304577748
  68634e4d544d774f5441784d6a4d794f544933576863634e4d7a4d774f5441784d6a4d790a4f544933576a41674d517377435159445651514745774a56557a4
  5524d41384741315545417884d49536e56755a32386751304577676745694d413047435371470a5349623344514541415155414134494244477417767454b
  416f4942415144614d306e7a446647426852446d37566a2f39552b614c594644647561565466674720a415575306d664f795244684c436661746d6636686d6
  17643353653830516c417454625143b39524475566b68724f455148656c494474747954522754c4365470a6531574d4c32324b3642314416e6f4d5a534f
  4d4c426a4345465750 2f33374a35694c6b5135503438686b49526970526b54444f3836456c6b6145586247446f0a353249342b6f2b2f4542963374874706
  662466d64304763433266725852456144632b6d4e775069666237772f757852456246746768534f7a52616c794f6b670a77476c394e444b2f7353324c7276
  6f572f7a4d6b614d7364767548663753687a577572373454 6a676142456831537 46c2f343578484945433666b7839356d346e0a536e6246374b79564b4e336
  165705072544d424f4858315853364 45a47466f432f456843505970 71314c4b6f432f634d68426b7a41674d4241414577445159 4a0a4b6f5a496876634e41
  514546425141446767454241485472485243314c4e426943636c7853587478736694b50633541 77616e356c5a343577545331373 46d750a6365337743744
  5616d4473634838 4a363371424265503244654270544 4b6e62516a35544425372436333666665676662376575665354 76544f4e72336e745064590a49617042
  732f356f63656c794275484c78314e2b686476745776393170 52732f514b35475367566f626 2672b6157334935695578 6f7055784163565 64c68325a0a376
  830645a7666375367386b364b652f2f586b433957706970594 94e304e576150634d7a44442f78614d594c6552425863554f6d5679674679686f394e637552
  0a76746b61416a3839486c47396c7059516e744d6c3538315956377 54c53674545 33694e682f657451502f5253693767662b7146542b6a574146734e30594
  e37780a534753426a65574c697141446c35416f4f36466d795a377059767865 4b654c73654d32314551356 1464c453d0a2d2d2d2d2d454e44204345525449
  4649434154452d2d2d2d2d0a))
  (owner(1))
  (name(ZON HUB remote update))
)
```

Remote Updates

- Loops all available certificates in configuration file.
- Only cares about those with **owner** == 1.



SURPRISE!



```
(cert
  (0
    (cert(2d2d2d2d2d424547494e2043455254494649434154452d2d2d2d2d0a4d494943316a43434162366741774942416749454f485861737a414e42676b71686b6947397
    7304241515546414441674d517377435159445651514745774a560a557a45524d4138474131554541784d49536e56755a323867513045774868634e4d544d774f5441794d
    4441794f5451335768634e4d7a4d774f4449344d4441790a4f545133576a416a4d517377435159445651514745774a56557a45554d4249474131554541784d4c656d39756
    14856694c6d687662575577675a38774451594a0a4b6f5a496876634e4151454242514144675930414d49474a416f4742414c4c4a30613366626a62634a447a5774357541
    4343586e596b41744d2b784a446c6a760a4348614567566a4f634730646d6c35716538784e6a685a4572643672485868724b7949364a4f54484e5a64614245636b3953416
    7377374346177654e4d46664d0a6e56414a544639334b677464315a4b3170427a4e416f76744a4e45487433522f4d535644674268556b586b5a6a34426d46576442313379
    4b65327974376835780a636468797751434e41674d424141476a675a6777675a5577444159445652305442415577417749424254417842674e56485355454b6a416f42676
    772426745460a42516344416759494b7759424251554841774d47434373474151554642774d4542676777242674546425163444154412f42676c67686b6742687668434151
    30450a4d685977536e56755a3238675433426c626c4a484946427962325231593352 7a494564796233567749484e305957356b59584a6b49474e6c636e52705a6d6c6a0a5
    958526c4d42454743574347534147472b45494241515145417749437844414e42676b71686b6947397730424151554641414f434151454169376e4a4670566f0a566f6c6c6c
    7933614b70784171563562615a5272444259444442794369425949765054706470794133726837436f784a624a7252556646702b4e3472437371462b0a434e74713044707
    36652433249645667674f6a48646441626a354a572b2f62547548657838305a4f753733456b43383944562f77462f6a38336a327637767749 0a6e7354783651724f365836
    722f3347776a4d493948665331693663565173664b69654a333247553762554d596241696632344e306a6833637675786252506b6e0a6c52544e38334c315243303547447
    862627131455757545945494c706a42634f743177394a754550345177475a65614c465356394f7070526b6e306a2f466b360a307579756644734b775668786b6c4d4d7956
    35304d6d4a346741785966696c5255724d6a51354a78625a4a726749487752696a56614e784f776a7957585651680a55326b42524476335236556375513
    d3d0a2d2d2d2d2d454e442043455254494649434154452d2d2d2d2d0a))
    (private(2d2d2d2d2d424547494e205253412050524956415445204b45592d2d2d2d2d0a4d4949435841494241414b4267514379796447743332343233343513831726562
    6741676c35324a414c5450735351355937776832684946597a6e4274485a70650a616e764d54593457524b336571783134617973694f69546b787a5758576752484a50556
    7494f374c654773486a5442587a4a31514355786664796f4c586457530a746151637a514b4c3753545242376430667a456c51344159564a463547592b415a68566e516464
    38696e74737265346563584859637345416a514944415141420a416f474164546f6c4e4b644b5963676f4c6f6675375a4f655242515a394e427575694151757a6c4879455
    94c64596f755a79314132575669644d7670505371640a416d49614f616d7a4a34634f523739696f5447526d6f42654244423858433661434b7a584964493977622f6a717a
    327461356e7137664d41676c34644f6444660a6d686e546c3963664c7a70473079377861624f33444b4c4a4e42374b493939394475446951353138454a4f70644a785543515
    1446e36566e324b346f5256492f4f0a4847686e766844676755785a4c5a6c486631774b47572f37463978377a436b2f2f794d434467526f58376d59767974453647647253
    2f373766614655423365560a49736c3837486558416b45417856766c44676f6a6d5151464c6e5641676477466d6433484f63495a53706b747069486434725568506b44414
    7563179364d71700a783354775861702b6a424b415059412b705263587a566e4b2b7a34366d4a497465774a42414b2f557a43474a35316532626f584f4968617545367530
    504d664d0a65382b7172345430766d4e354e7645683352437068386879436b4e3865494a6f52773071792b61644b776c51756a57307036554b53417751576163435148525
    60a656378767a3548333237494a3759396569584b73567935467172446459426f6f706d6c6e535951766d4d79613241772b776874505a4178695870766b6e2f45670a736b
    79514d7a695a734675757a3831316852634351413861735451626d35364b364b765a776c41715044594b56796f3655494e32747372505a2f6436526758590a3869684e434
    64447724c4a717849666e4c7867467765355a476f7852576b2b6371366573316763425956303d0a2d2d2d2d2d454e44205253412050524956415445204b45592d2d2d2d2d
    0a))
    (owner(1))
    (name(ZON HUB))
  )
  (1
    (cert(2d2d2d2d2d424547494e2043455254494649434154452d2d2d2d2d0a4d4949437644434341615143435143785556665865465a566a7a414e42676b71686b6947397
    7304241515546414441674d517377435159445651514745774a560a557a45524d4138474131554541784d49536e56755a323867513045774868634e4d544d774f5441784d
    6a4d794f5449335768634e4d7a4d774f5441784d6a4d790a4f544933576a41674d517377435159445651514745774a56557a45524d4138474131554541784d49536e56755
    a32386751304577676745694d413047435371470a53496233445145424251554141344942447741776767454b416f49424151446647426852446d37566a2f
    39552b614c59464464756156546674720a415575306d664f795244684c436661746d6636686d6176433536533830516c4174546251432b39524475566b68724f455148656
    c49447474795452754c4365470a6531574d4c32324b36423141416e6f4d5a534f4d4c426a43454657502f33374a35694c6b5135503438686b49526970526b54444f383645
    6c6b6145586247446f0a353249342b6f2b2f45423963374874706662466d6430476343326672585245614463 2b6d4e775069666237772f757852456246746768534f7a526
    16c794f6b670a77476c394e444b2f7353324c72766f572f7a4d6b614d7364767548663753687a5775723734546a6761424568315374 6c2f34357848494543366b7839356d
    346e0a536e6246374b79564b4e336165705072544d424f48583158533644 5a47466f432f45684350597071314c4b6f432f634d68426b7a41674d42414145774451594a0a4
    b6f5a496876634e41514542425141446767454241485472485243314c4e426943636c78535874785736694b50633541776 16e356c5a34357754533137346d750a63653377
    437445616d44736348384a36337142426550324465427054 4b6e62516a35544253724363333666665676662376575665354765 44f4e72336e745064590a49617042732f356
    f63656c794275484c78314e2b68647674577639317052732f514b35475367566f6262672b61573349356955786f705578416356564c68325a0a376830645a766637536738
    6b364b652f2f586b43395770697059494e304e576150634d7a44442f78614d594c6552425863554f6d5679674679686f394e6375520a76746b61416a3839486c47396c705
    9516e744d6c353831595637754c5367454533694e682f657451502f5253693767662b7146542b6a574146734e30594e37780a534753426a65574c697141446c35416f4f36
    466d795a3770597678654b654c73654d323145513561464c453d0a2d2d2d2d2d454e442043455254494649434154452d2d2d2d2d0a))
    (owner(2))
    (name(Jungo CA))
  )
```

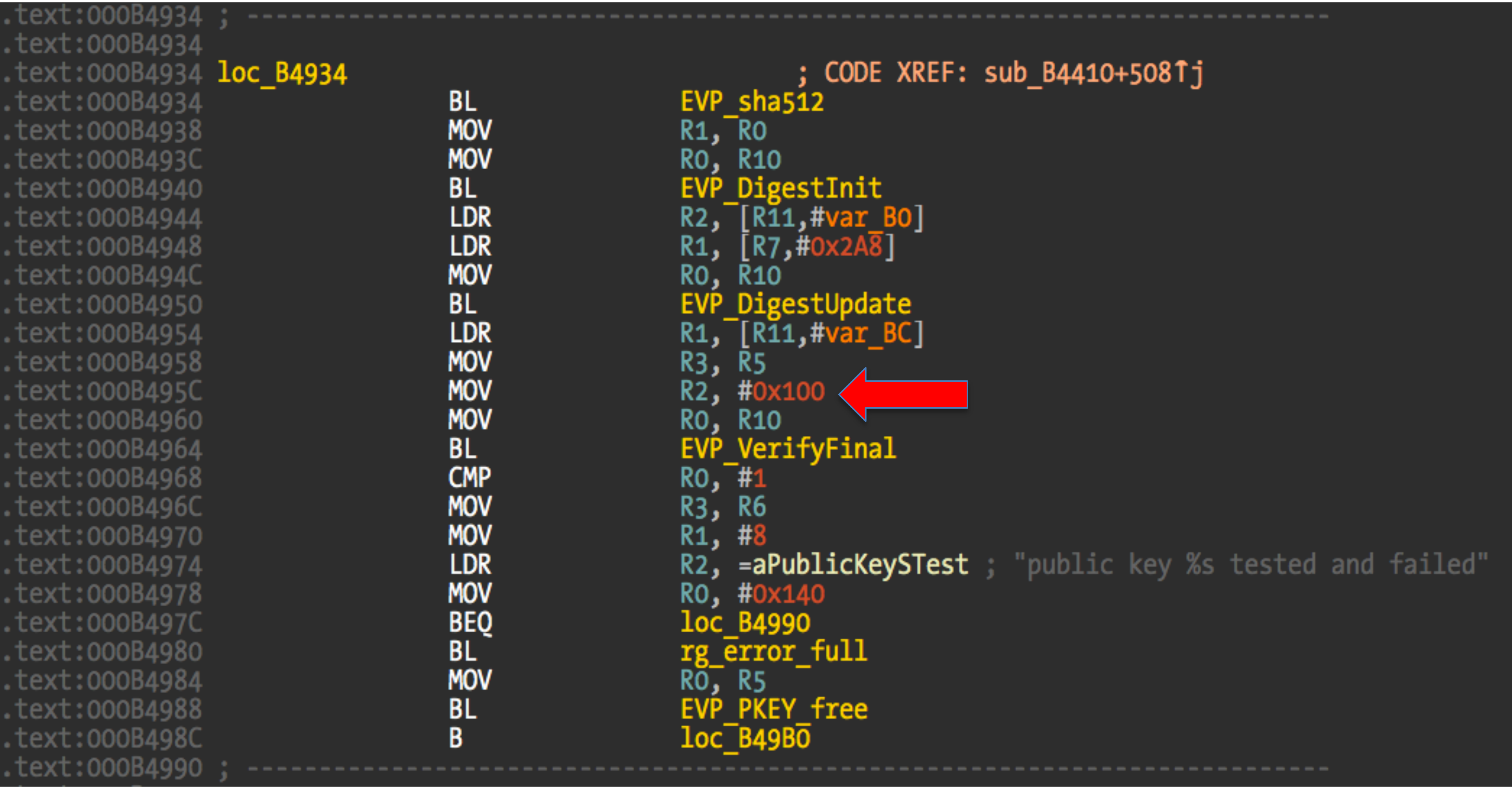
Remote Updates

- The client certificate has **owner** == 1.
- WE HAVE A PRIVATE KEY!
- WE CAN RESIGN UPDATES!

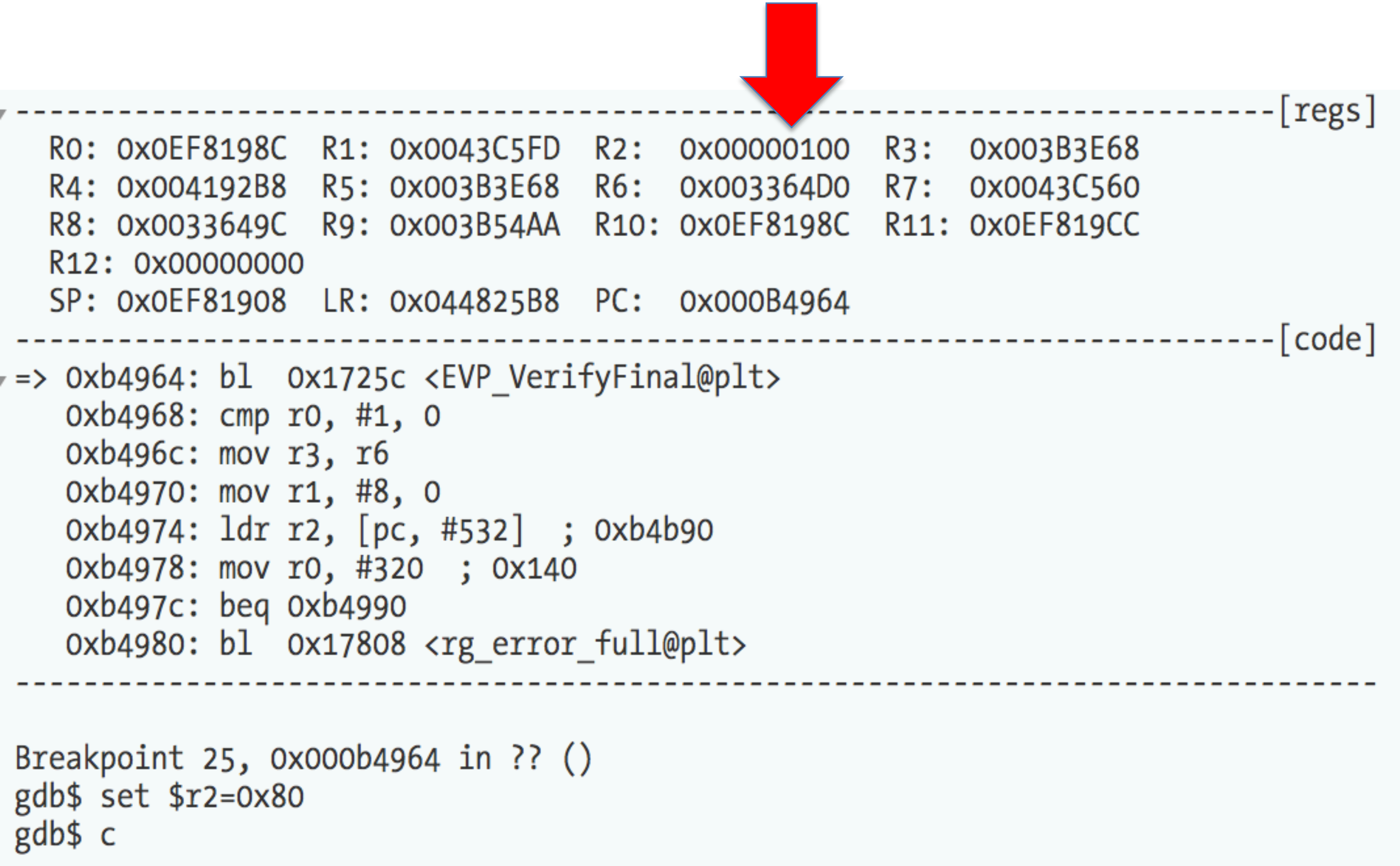
VICTORY!

Remote Updates

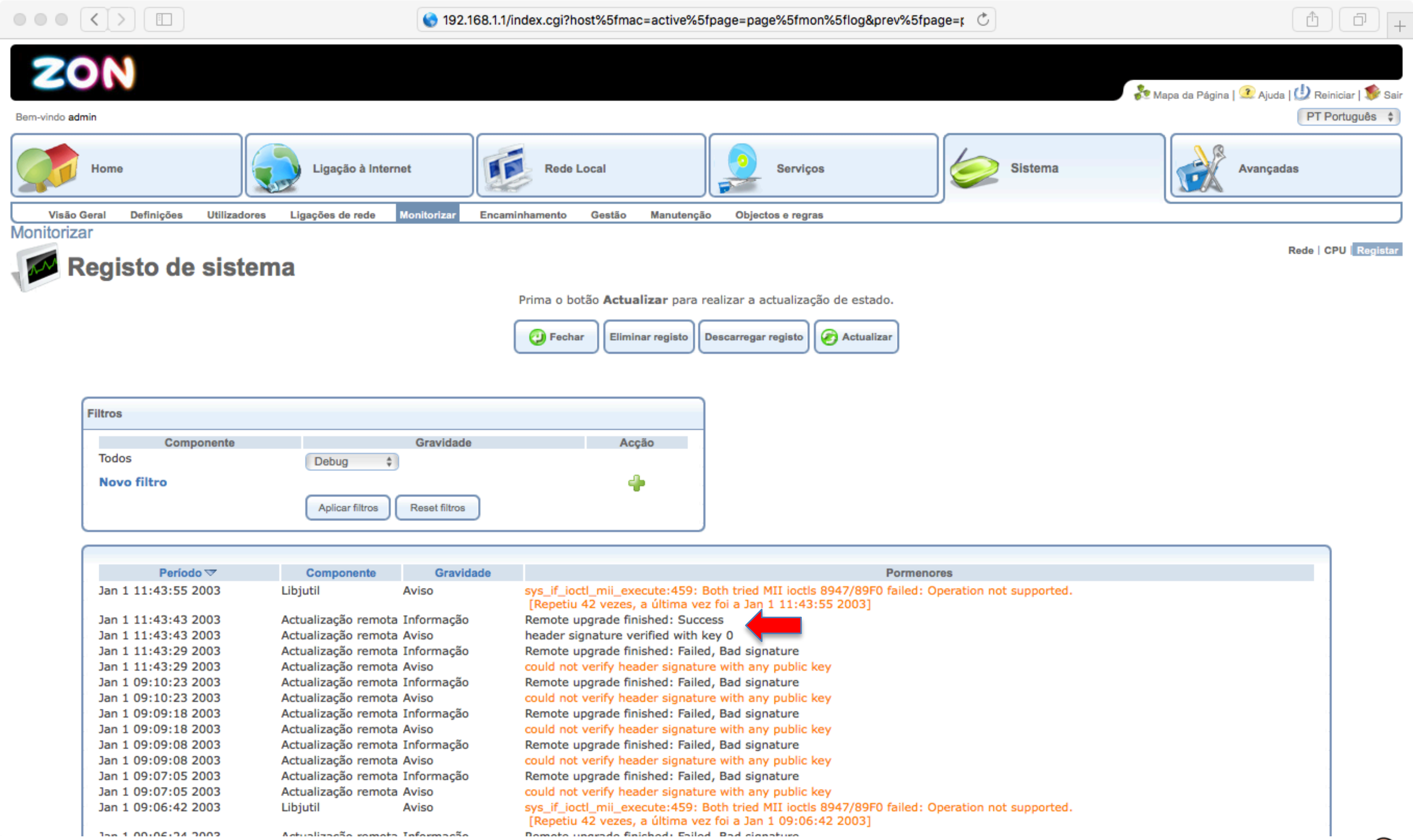
- The signature length is fixed at 256 bytes.
- The RSA key is only 1024 bits.
- EVP_VerifyFinal() will return failure.
- ☹ ☹ ☹ ☹ ☹ ☹ ☹ ☹ ☹ ☹ ☹ ☹ ☹ ☹ ☹ ☹ ☹



```
.text:000B4934 ; ---------------------------------------------------------------------------
.text:000B4934
.text:000B4934 loc_B4934                               ; CODE XREF: sub_B4410+508↑j
.text:000B4934                 BL              EVP_sha512
.text:000B4938                 MOV             R1, R0
.text:000B493C                 MOV             R0, R10
.text:000B4940                 BL              EVP_DigestInit
.text:000B4944                 LDR             R2, [R11,#var_B0]
.text:000B4948                 LDR             R1, [R7,#0x2A8]
.text:000B494C                 MOV             R0, R10
.text:000B4950                 BL              EVP_DigestUpdate
.text:000B4954                 LDR             R1, [R11,#var_BC]
.text:000B4958                 MOV             R3, R5
.text:000B495C                 MOV             R2, #0x100
.text:000B4960                 MOV             R0, R10
.text:000B4964                 BL              EVP_VerifyFinal
.text:000B4968                 CMP             R0, #1
.text:000B496C                 MOV             R3, R6
.text:000B4970                 MOV             R1, #8
.text:000B4974                 LDR             R2, =aPublicKeySTest ; "public key %s tested and failed"
.text:000B4978                 MOV             R0, #0x140
.text:000B497C                 BEQ             loc_B4990
.text:000B4980                 BL              rg_error_full
.text:000B4984                 MOV             R0, R5
.text:000B4988                 BL              EVP_PKEY_free
.text:000B498C                 B               loc_B49B0
.text:000B4990 ; ---------------------------------------------------------------------------
```



```
-----------------------------------------------------------------------[regs]
  R0: 0x0EF8198C  R1: 0x0043C5FD  R2:   0x00000100  R3:   0x003B3E68
  R4: 0x004192B8  R5: 0x003B3E68  R6:   0x003364D0  R7:   0x0043C560
  R8: 0x0033649C  R9: 0x003B54AA  R10:  0x0EF8198C  R11:  0x0EF819CC
  R12: 0x00000000
  SP: 0x0EF81908  LR: 0x044825B8  PC:   0x000B4964
-----------------------------------------------------------------------[code]
=> 0xb4964: bl  0x1725c <EVP_VerifyFinal@plt>
   0xb4968: cmp r0, #1, 0
   0xb496c: mov r3, r6
   0xb4970: mov r1, #8, 0
   0xb4974: ldr r2, [pc, #532]  ; 0xb4b90
   0xb4978: mov r0, #320  ; 0x140
   0xb497c: beq 0xb4990
   0xb4980: bl  0x17808 <rg_error_full@plt>
-----------------------------------------------------------------------------


Breakpoint 25, 0x000b4964 in ?? ()
gdb$ set $r2=0x80
gdb$ c
```



Monitorizar

Registo de sistema

Prima o botão **Actualizar** para realizar a actualização de estado.

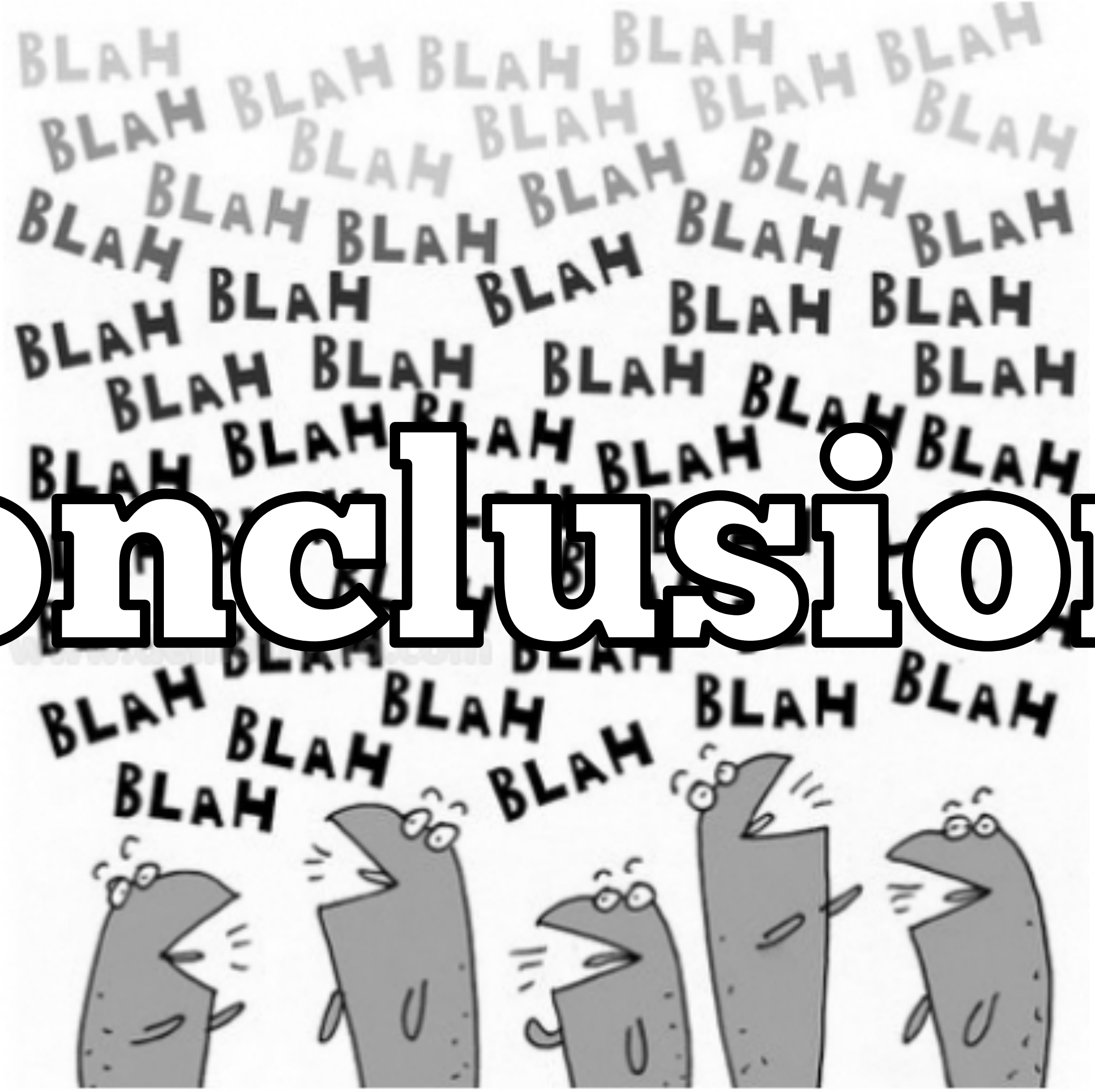
Filtros

Componente	Gravidade	Acção
Todos	Debug	
Novo filtro		✚

Período	Componente	Gravidade	Pormenores
Jan 1 11:43:55 2003	Libjutil	Aviso	sys_if_ioctl_mii_execute:459: Both tried MII ioctls 8947/89F0 failed: Operation not supported. [Repetiu 42 vezes, a última vez foi a Jan 1 11:43:55 2003]
Jan 1 11:43:43 2003	Actualização remota	Informação	Remote upgrade finished: Success
Jan 1 11:43:43 2003	Actualização remota	Aviso	header signature verified with key 0
Jan 1 11:43:29 2003	Actualização remota	Informação	Remote upgrade finished: Failed, Bad signature
Jan 1 11:43:29 2003	Actualização remota	Aviso	could not verify header signature with any public key
Jan 1 09:10:23 2003	Actualização remota	Informação	Remote upgrade finished: Failed, Bad signature
Jan 1 09:10:23 2003	Actualização remota	Aviso	could not verify header signature with any public key
Jan 1 09:09:18 2003	Actualização remota	Informação	Remote upgrade finished: Failed, Bad signature
Jan 1 09:09:18 2003	Actualização remota	Aviso	could not verify header signature with any public key
Jan 1 09:09:08 2003	Actualização remota	Informação	Remote upgrade finished: Failed, Bad signature
Jan 1 09:09:08 2003	Actualização remota	Aviso	could not verify header signature with any public key
Jan 1 09:07:05 2003	Actualização remota	Informação	Remote upgrade finished: Failed, Bad signature
Jan 1 09:07:05 2003	Actualização remota	Aviso	could not verify header signature with any public key
Jan 1 09:06:42 2003	Libjutil	Aviso	sys_if_ioctl_mii_execute:459: Both tried MII ioctls 8947/89F0 failed: Operation not supported. [Repetiu 42 vezes, a última vez foi a Jan 1 09:06:42 2003]
Jan 1 09:06:24 2003	Actualização remota	Informação	Remote upgrade finished: Failed, Bad signature

DEFEAT! ☹

Conclusions



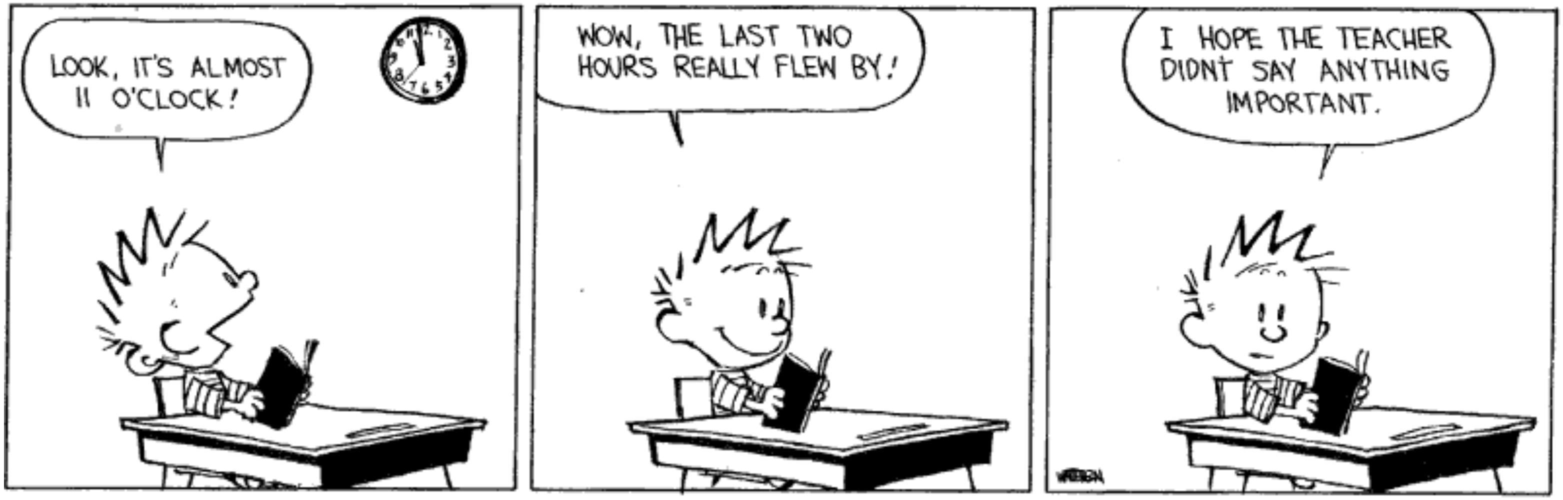
It’s a Mickey Mouse operation!

Conclusions

- Now we have further control because we can RE everything.
- Easily recovered passwords plaintext.
- Remote updates seem fine today but not in the past*.
- Next step is to find (RCE) vulnerabilities.

Greetings

- 0xOPOSEC team.


LOOK, IT'S ALMOST 11 O'CLOCK!
WOW, THE LAST TWO HOURS REALLY FLEW BY!
I HOPE THE TEACHER DIDN'T SAY ANYTHING IMPORTANT.
WATTERSON

https://reverse.put.as

https://github.com/gdbinit

reverser@put.as

@osxreverser

#osxre @ irc.freenode.net

PGP key

https://reverse.put.as/wp-content/uploads/2008/06/publickey.txt

PGP Fingerprint

7B05 44D1 A1D5 3078 7F4C E745 9BB7 2A44 ED41 BF05

References

- Images from images.google.com. Credit due to all their authors.
- http://www.devttys0.com/2012/11/reverse-engineering-serial-ports/
- http://jcjc-dev.com/2016/04/08/reversing-huawei-router-1-find-uart/
- https://wikidevi.com/wiki/Hitron
- https://wikidevi.com/wiki/Hitron_BVW-3653
- http://www.hitrontech.com/product/cve-30360/

References

- https://www.zerodayinitiative.com/blog/2019/9/2/mindshare-hardware-reversing-with-the-tp-link-tl-wr841n-router