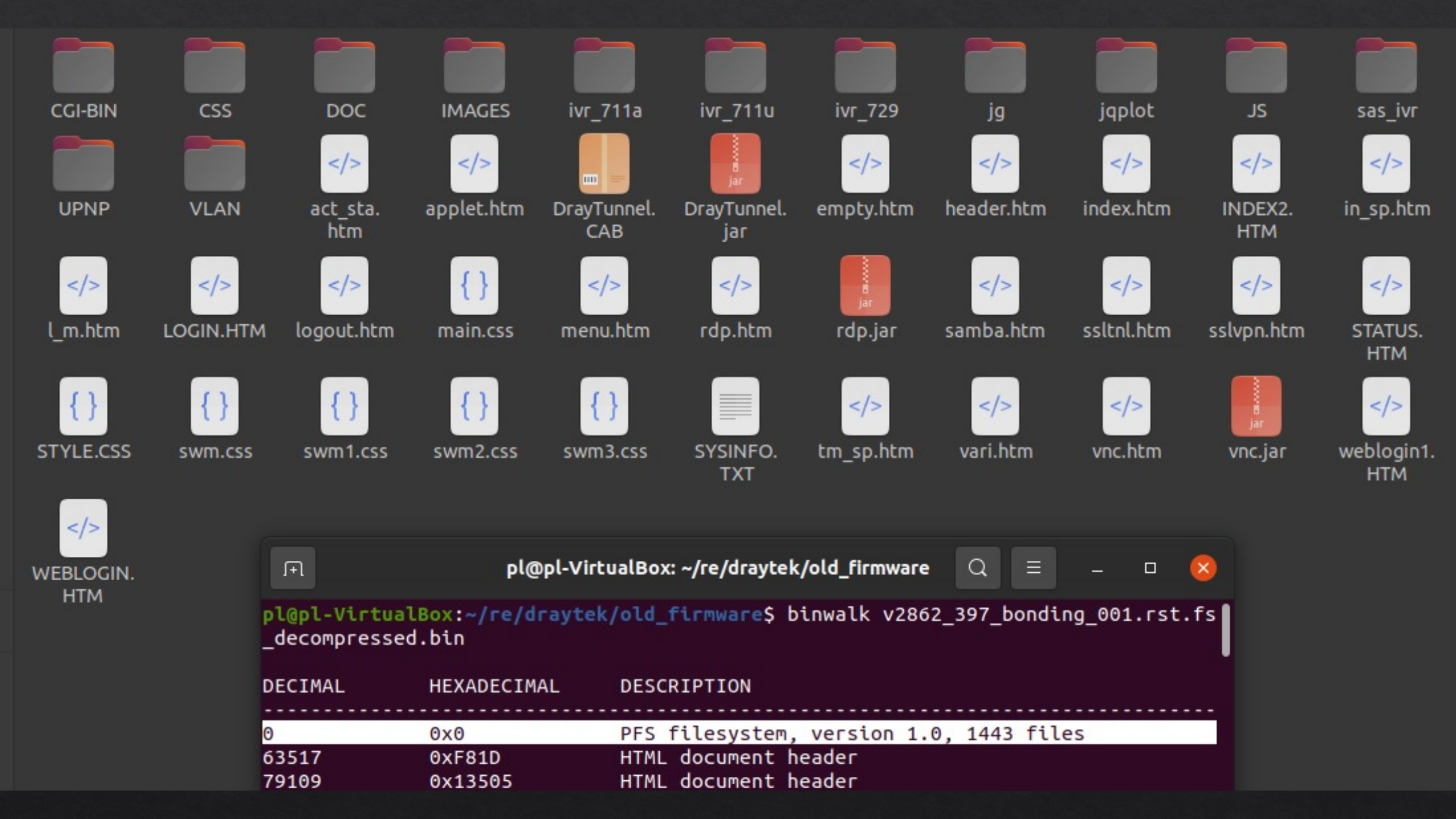
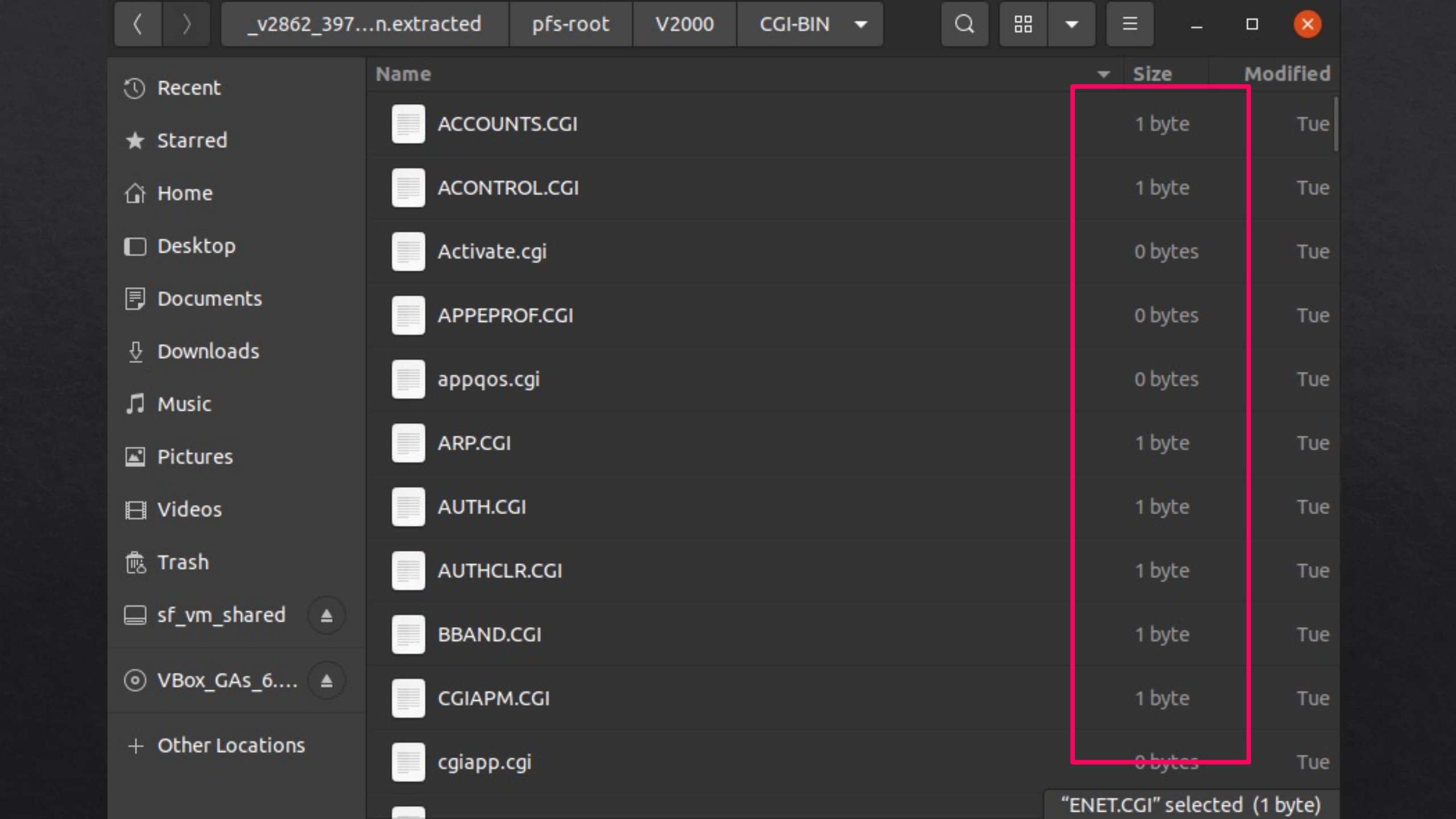
```
def find_fs(data):
    offset = struct.unpack(">L", data[:4])[0]
    sz1, sz2, magic = struct.unpack(">LLL", data[offset:offset+4*3])
    if magic !=  0xAA1D7F50:
        raise(Exception("Failed to find magic bytes for fs (val:{:x} expected:{:x})".format(magic2, 0xAA1D7F50) ))

    return data[offset+8:offset+8+sz2] #needs to be sz2 or get a error chunk size



def find_extract_blob(filename):
    with open(filename, "rb") as f:
        data = f.read()

    pos = data.find(b"\xA5\x5A\xA5\x5A")
    if (pos == -1):
        raise("Failed to find magic byte")
    # PLEASE NOTE: if you want to have a full firmware
    # (decompression routine + decompressed firmware), then you need to replace
    # the data starting at pos (from above) with the decompressed firmware
    # as this is the first bytes that get overwritten by decompressed firmware
    # as the process moves along
    # with pos = data.find(b"\xA5\x5A\xA5\x5A")


    loader_fw = data[0x100:pos]
    magic1, size, magic2 = struct.unpack(">III", data[pos:pos+4*3])

    if magic2 !=  0xAA1D7F50:
        raise(Exception("Failed to find magic bytes part2 (val:{:x} expected:{:x})".format(magic2, 0xAA1D7F50) ))
    print("Blob size: {:x}".format(size))

    blob_start = pos + 8

    fs_blob = find_fs(data)

    return loader_fw, data[blob_start:blob_start + size], fs_blob
```


CGI-BIN
CSS
DOC
IMAGES
ivr_711a
ivr_711u
ivr_729
jg
jqplot
JS
sas_ivr
UPNP
VLAN
act_sta.htm
applet.htm
DrayTunnel.CAB
DrayTunnel.jar
empty.htm
header.htm
index.htm
INDEX2.HTM
in_sp.htm
l_m.htm
LOGIN.HTM
logout.htm
main.css
menu.htm
rdp.htm
rdp.jar
samba.htm
ssltnl.htm
sslvpn.htm
STATUS.HTM
STYLE.CSS
swm.css
swm1.css
swm2.css
swm3.css
SYSINFO.TXT
tm_sp.htm
vari.htm
vnc.htm
vnc.jar
weblogin1.HTM
WEBLOGIN.HTM
pl@pl-VirtualBox: ~/re/draytek/old_firmware
pl@pl-VirtualBox:~/re/draytek/old_firmware$ binwalk v2862_397_bonding_001.rst.fs_decompressed.bin
DECIMAL HEXADECIMAL DESCRIPTION
0 0x0 PFS filesystem, version 1.0, 1443 files
63517 0xF81D HTML document header
79109 0x13505 HTML document header

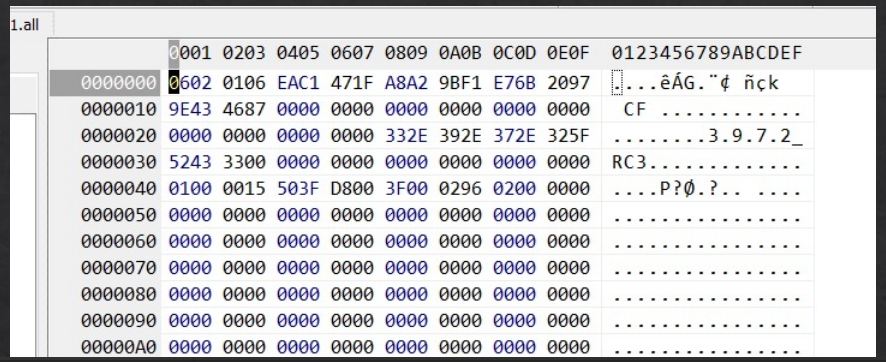

_v2862_397…n.extracted
pfs-root
V2000
CGI-BIN
Recent
Starred
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
sf_vm_shared
VBox_GAs_6.…
Other Locations
Name
Size
Modified
ACCOUNTS.CGI 1 byte Tue
ACONTROL.CGI 1 byte Tue
Activate.cgi 0 bytes Tue
APPEPROF.CGI 0 bytes Tue
appqos.cgi 0 bytes Tue
ARP.CGI 1 byte Tue
AUTH.CGI 1 byte Tue
AUTHCLR.CGI 1 byte Tue
BBAND.CGI 1 byte Tue
CGIAPM.CGI 1 byte Tue
cgiapp.cgi Tue
"ENET.CGI" selected (1 byte)

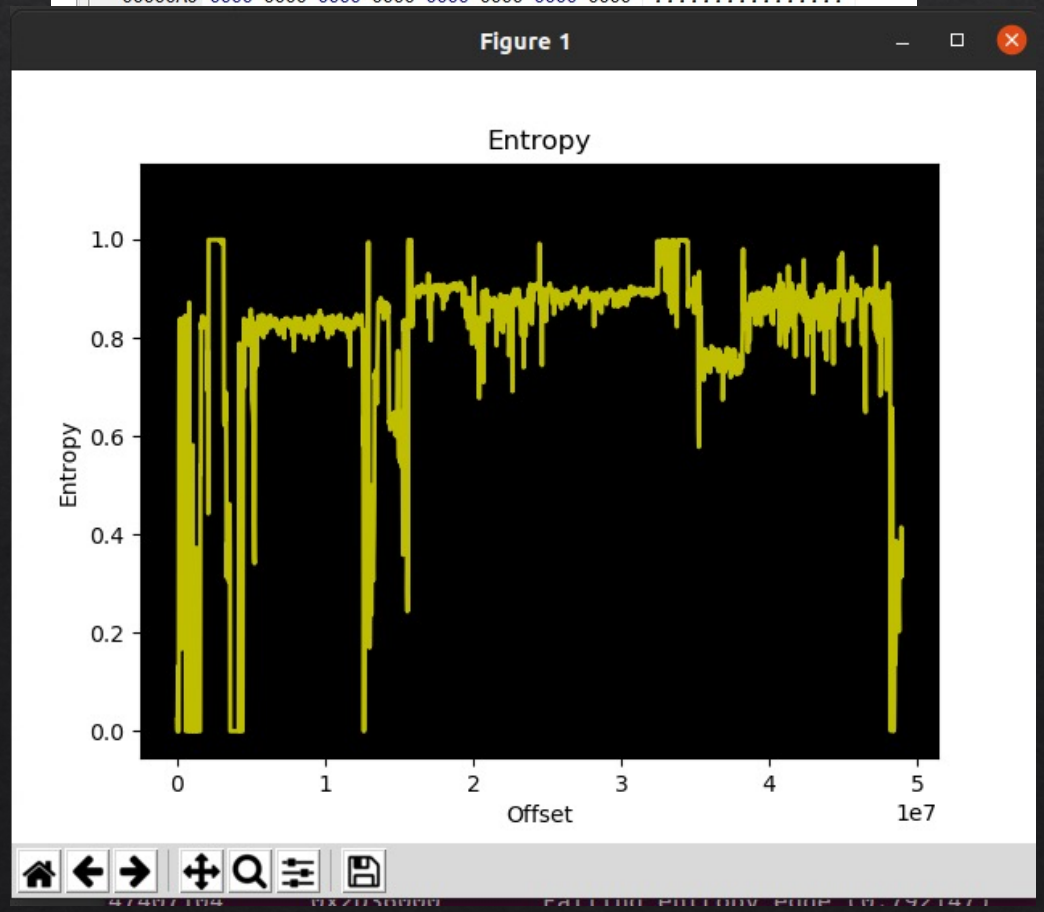
Decompression Result

- Firmware looks legit in IDA!
- Filesystem is also compressed. Decompression leads to PFS filesystem
  - binwalk to extract PFS
  - Some of the files are compressed with the same algo as well
- CGI are empty ☹
  - Need to find them in a different castle

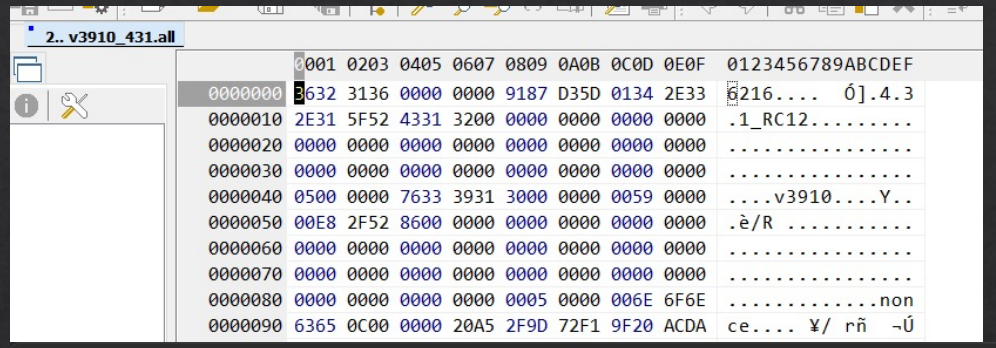
Looking back at the Vigor 3910

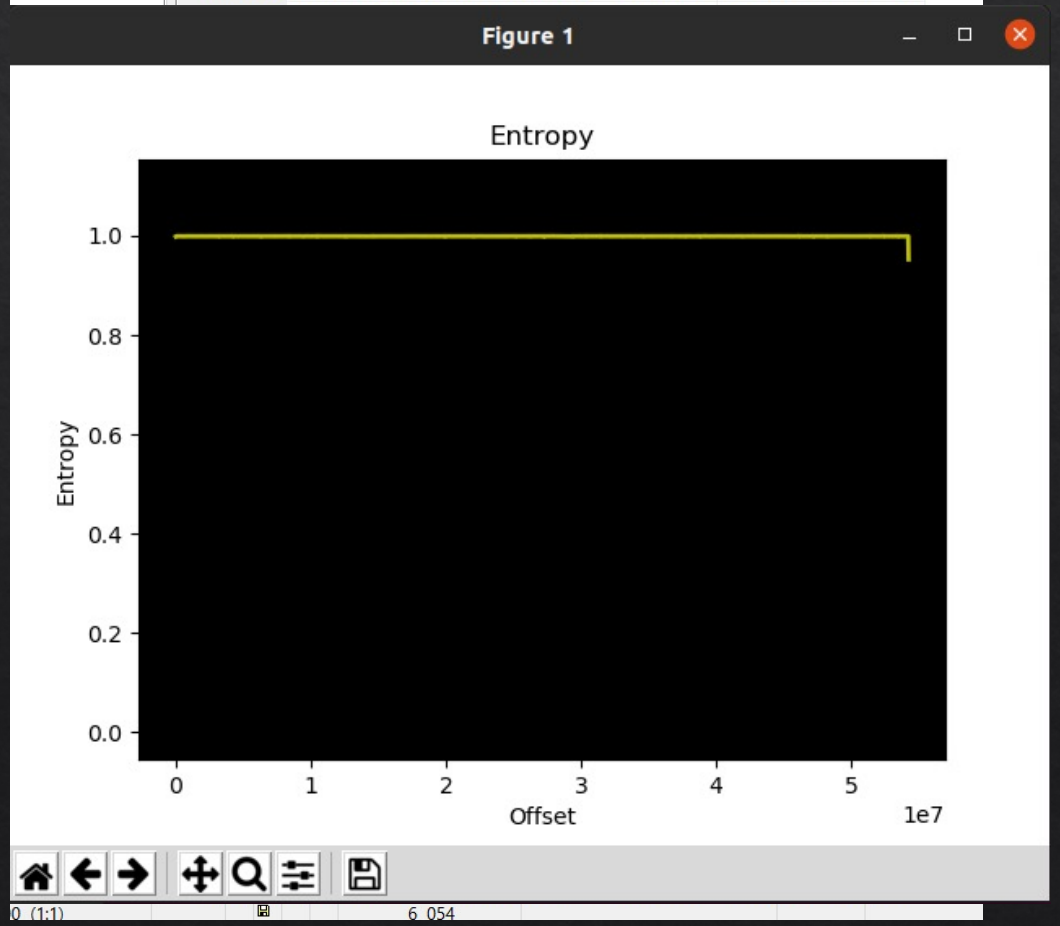
Vigor 3910 Firmware 3.9.y





Vigor 3910 Firmware 4.x



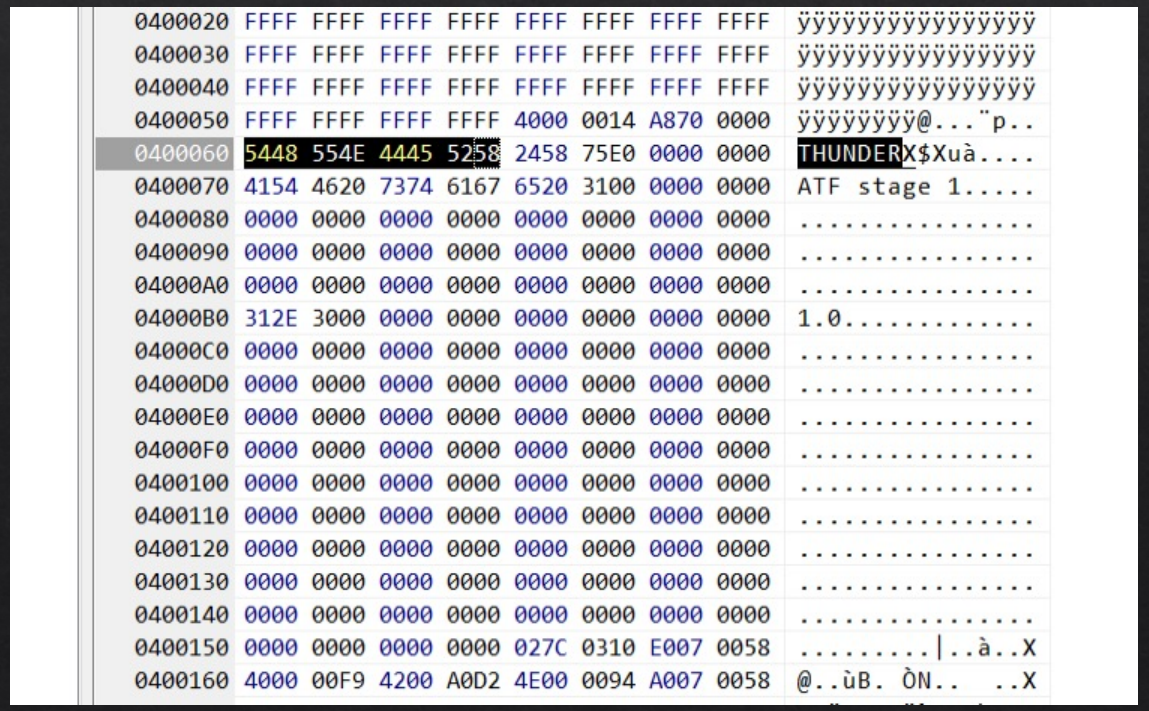


Gimme da decryption Keeeyzz

- Where's the key?
  - Either inside FW update code (decryption upon install)
  - At boot time (FW encrypted "at rest")
- Flash dump would answer the question….
- …. But the title is "pwning it before getting out of the box, so, …."

Vigor 3910 Early Boot

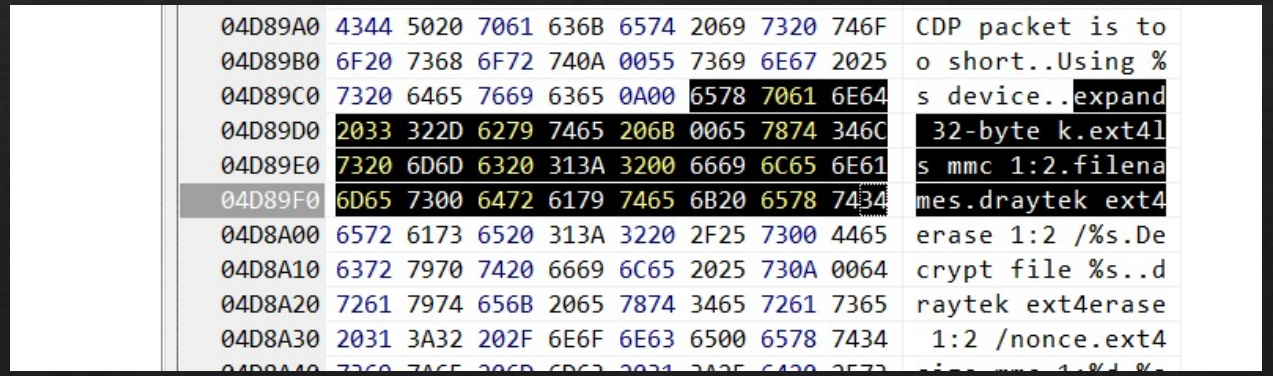
- Looking at firmware file v3.9.7.1
- THUNDERX Files
  - From OCTEON THUNDER SDK
  - Yolo split (surrounded by \x00\x00…\x00 or \xff\xff..\xff)
    - boot.bin
    - boot.bin (again)
    - init.bin
    - ATF stage 1 (later called stage1.bin)



Vigor 3910 Early Boot

- ARM stuff in stage1.bin
  - BL1, BL2, BL31
  - Funky Strings:
    - “Decrypt File”
    - “expand 32-byte k”

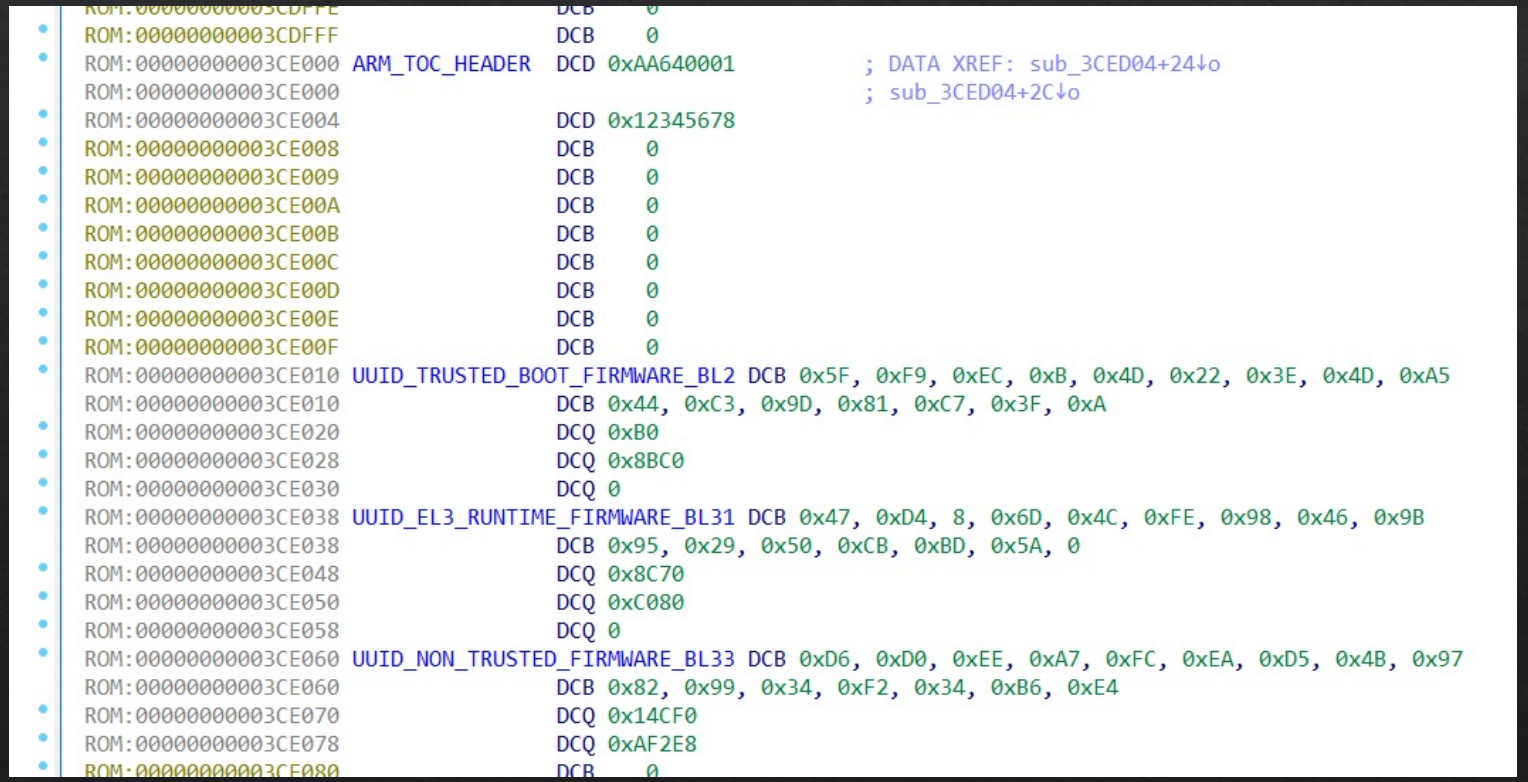
```
04055E0 320A 004E 4F54 4943 453A 2020 424C 312D  2..NOTICE:  BL1-
04055F0 4657 553A 202A 2A2A 2A2A 2A2A 4657 5520  FWU: *******FWU
0405600 5072 6F63 6573 7320 5374 6172 7465 642A  Process Started*
0405610 2A2A 2A2A 2A2A 0A00 4E4F 5449 4345 3A20  ******..NOTICE:
0405620 2042 4C31 3A20 426F 6F74 696E 6720 424C   BL1: Booting BL
0405630 3331 0A00 0000 0000 D467 39FD CB72 9A4D  31......Ôg9ýËr M
0405640 B575 6715 D6F4 BB4A 03EF EFEF EF03 EFEF  µug.Öô»J.ïïïï.ïï
0405650 77EF EFEF 3EEF EF29 EF79 EFEF 57EF 0030  wïïï>ïï)ïyïïWï.0
0405660 7800 0000 0C00 8803 2700 8803 8803 8803  x..... .'. . . .
0405670 8803 3300 2200 8803 2400 0900 8803 5800   .3.". .$... .X.
```



```
04D89A0 4344 5020 7061 636B 6574 2069 7320 746F  CDP packet is to
04D89B0 6F20 7368 6F72 740A 0055 7369 6E67 2025  o short..Using %
04D89C0 7320 6465 7669 6365 0A00 6578 7061 6E64  s device..expand
04D89D0 2033 322D 6279 7465 206B 0065 7874 346C   32-byte k.ext4l
04D89E0 7320 6D6D 6320 313A 3200 6669 6C65 6E61  s mmc 1:2.filena
04D89F0 6D65 7300 6472 6179 7465 6B20 6578 7434  mes.draytek ext4
04D8A00 6572 6173 6520 313A 3220 2F25 7300 4465  erase 1:2 /%s.De
04D8A10 6372 7970 7420 6669 6C65 2025 730A 0064  crypt file %s..d
04D8A20 7261 7974 656B 2065 7874 3465 7261 7365  raytek ext4erase
04D8A30 2031 3A32 202F 6E6F 6E63 6500 6578 7434   1:2 /nonce.ext4
```

Vigor 3910 Early Boot (stage1.bin)

- TOC_HEADER_NAME (0xAA640001)
  - UUID of various sections (BL2, BL31, BL33) defined in ARM specs
  - Easy to binary search



```
ROM:00000000003CDFFF                    DCB    0
ROM:00000000003CE000 ARM_TOC_HEADER     DCD 0xAA640001          ; DATA XREF: sub_3CED04+24↓o
ROM:00000000003CE000                                            ; sub_3CED04+2C↓o
ROM:00000000003CE004                    DCD 0x12345678
ROM:00000000003CE008                    DCB    0
ROM:00000000003CE009                    DCB    0
ROM:00000000003CE00A                    DCB    0
ROM:00000000003CE00B                    DCB    0
ROM:00000000003CE00C                    DCB    0
ROM:00000000003CE00D                    DCB    0
ROM:00000000003CE00E                    DCB    0
ROM:00000000003CE00F                    DCB    0
ROM:00000000003CE010 UUID_TRUSTED_BOOT_FIRMWARE_BL2 DCB 0x5F, 0xF9, 0xEC, 0xB, 0x4D, 0x22, 0x3E, 0x4D, 0xA5
ROM:00000000003CE010                    DCB 0x44, 0xC3, 0x9D, 0x81, 0xC7, 0x3F, 0xA
ROM:00000000003CE020                    DCQ 0xB0
ROM:00000000003CE028                    DCQ 0x8BC0
ROM:00000000003CE030                    DCQ 0
ROM:00000000003CE038 UUID_EL3_RUNTIME_FIRMWARE_BL31 DCB 0x47, 0xD4, 8, 0x6D, 0x4C, 0xFE, 0x98, 0x46, 0x9B
ROM:00000000003CE038                    DCB 0x95, 0x29, 0x50, 0xCB, 0xBD, 0x5A, 0
ROM:00000000003CE048                    DCQ 0x8C70
ROM:00000000003CE050                    DCQ 0xC080
ROM:00000000003CE058                    DCQ 0
ROM:00000000003CE060 UUID_NON_TRUSTED_FIRMWARE_BL33 DCB 0xD6, 0xD0, 0xEE, 0xA7, 0xFC, 0xEA, 0xD5, 0x4B, 0x97
ROM:00000000003CE060                    DCB 0x82, 0x99, 0x34, 0xF2, 0x34, 0xB6, 0xE4
ROM:00000000003CE070                    DCQ 0x14CF0
ROM:00000000003CE078                    DCQ 0xAF2E8
ROM:00000000003CE080                    DCB    0
```

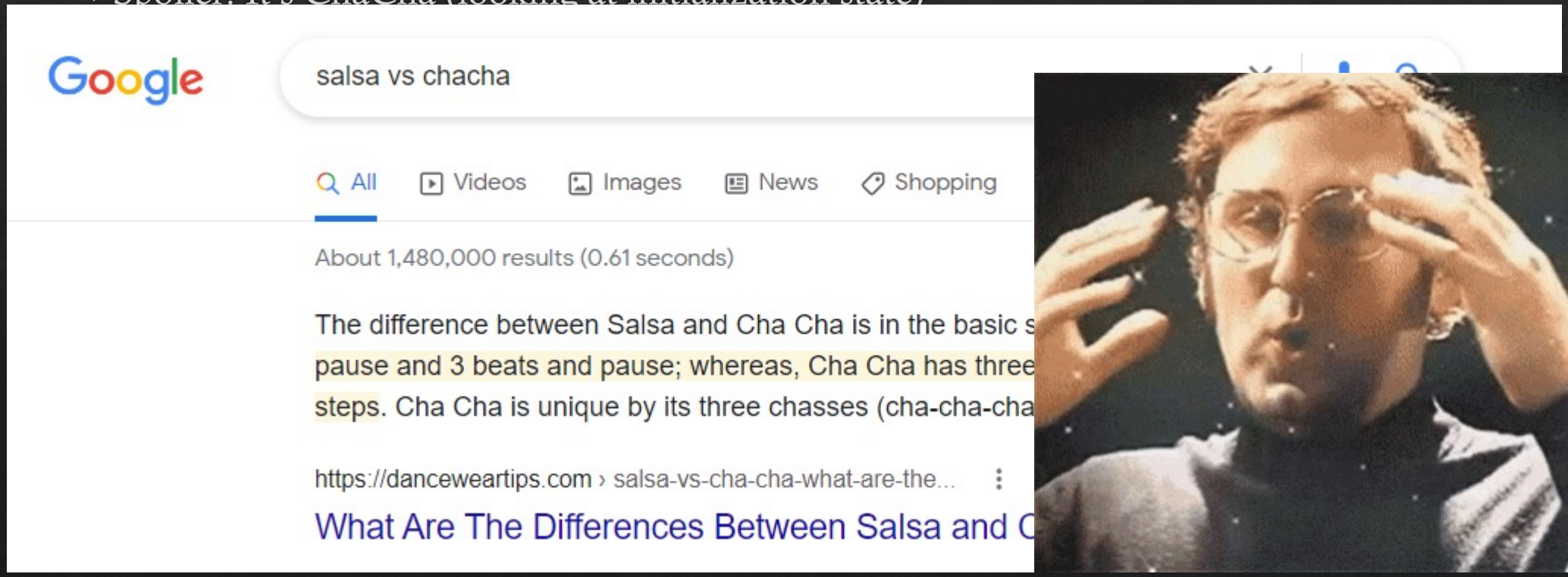
```
ROM:00000000003CDFFF                    DCB    0
ROM:00000000003CE000 ARM_TOC_HEADER     DCD 0xAA640001          ; DATA XREF: sub_3CED04+24↓o
ROM:00000000003CE000                                            ; sub_3CED04+2C↓o
ROM:00000000003CE004                    DCD 0x12345678
ROM:00000000003CE008                    DCB    0
ROM:00000000003CE009                    DCB    0
ROM:00000000003CE00A                    DCB    0
ROM:00000000003CE00B                    DCB    0
ROM:00000000003CE00C                    DCB    0
ROM:00000000003CE00D                    DCB    0
ROM:00000000003CE00E                    DCB    0
ROM:00000000003CE00F                    DCB    0
ROM:00000000003CE010 UUID_TRUSTED_BOOT_FIRMWARE_BL2 DCB 0x5F, 0xF9, 0xEC, 0xB, 0x4D, 0x22, 0x3E, 0x4D, 0xA5
ROM:00000000003CE010                    DCB 0x44, 0xC3, 0x9D, 0x81, 0xC7, 0x3F, 0xA
ROM:00000000003CE020                    DCQ 0xB0 <-- Offset (from TOC)
ROM:00000000003CE028                    DCQ 0x8BC0 <-- Size
ROM:00000000003CE030                    DCQ 0
ROM:00000000003CE038 UUID_EL3_RUNTIME_FIRMWARE_BL31 DCB 0x47, 0xD4, 8, 0x6D, 0x4C, 0xFE, 0x98, 0x46, 0x9B
ROM:00000000003CE038                    DCB 0x95, 0x29, 0x50, 0xCB, 0xBD, 0x5A, 0
ROM:00000000003CE048                    DCQ 0x8C70
ROM:00000000003CE050                    DCQ 0xC080
ROM:00000000003CE058                    DCQ 0
ROM:00000000003CE060 UUID_NON_TRUSTED_FIRMWARE_BL33 DCB 0xD6, 0xD0, 0xEE, 0xA7, 0xFC, 0xEA, 0xD5, 0x4B, 0x97
ROM:00000000003CE060                    DCB 0x82, 0x99, 0x34, 0xF2, 0x34, 0xB6, 0xE4
ROM:00000000003CE070                    DCQ 0x14CF0
ROM:00000000003CE078                    DCQ 0xAF2E8
ROM:00000000003CE080                    DCB    0
```

BL33 is where the fun is at!

- Load address: 0x500000
- Contains the "decrypt file" and "expand 32-byte k" strings

  → ChaCha or Salsa encryption

  → Spoiler: It's ChaCha (looking at initialization state)



BL33 is where the fun is at!

- Load address: 0x500000
- Contains the “decrypt file” and “expand 32-byte k” strings

  → ChaCha or Salsa encryption

  → Spoiler: It’s ChaCha (looking at initialization state)

Initial state of Salsa20

"expa"	Key	Key	Key
Key	"nd 3"	Nonce	Nonce
Pos.	Pos.	"2-by"	Key
Key	Key	Key	"te k"

Initial state of ChaCha

"expa"	"nd 3"	"2-by"	"te k"
Key	Key	Key	Key
Key	Key	Key	Key
Counter	Counter	Nonce	Nonce

BL33 is where the fun is at!

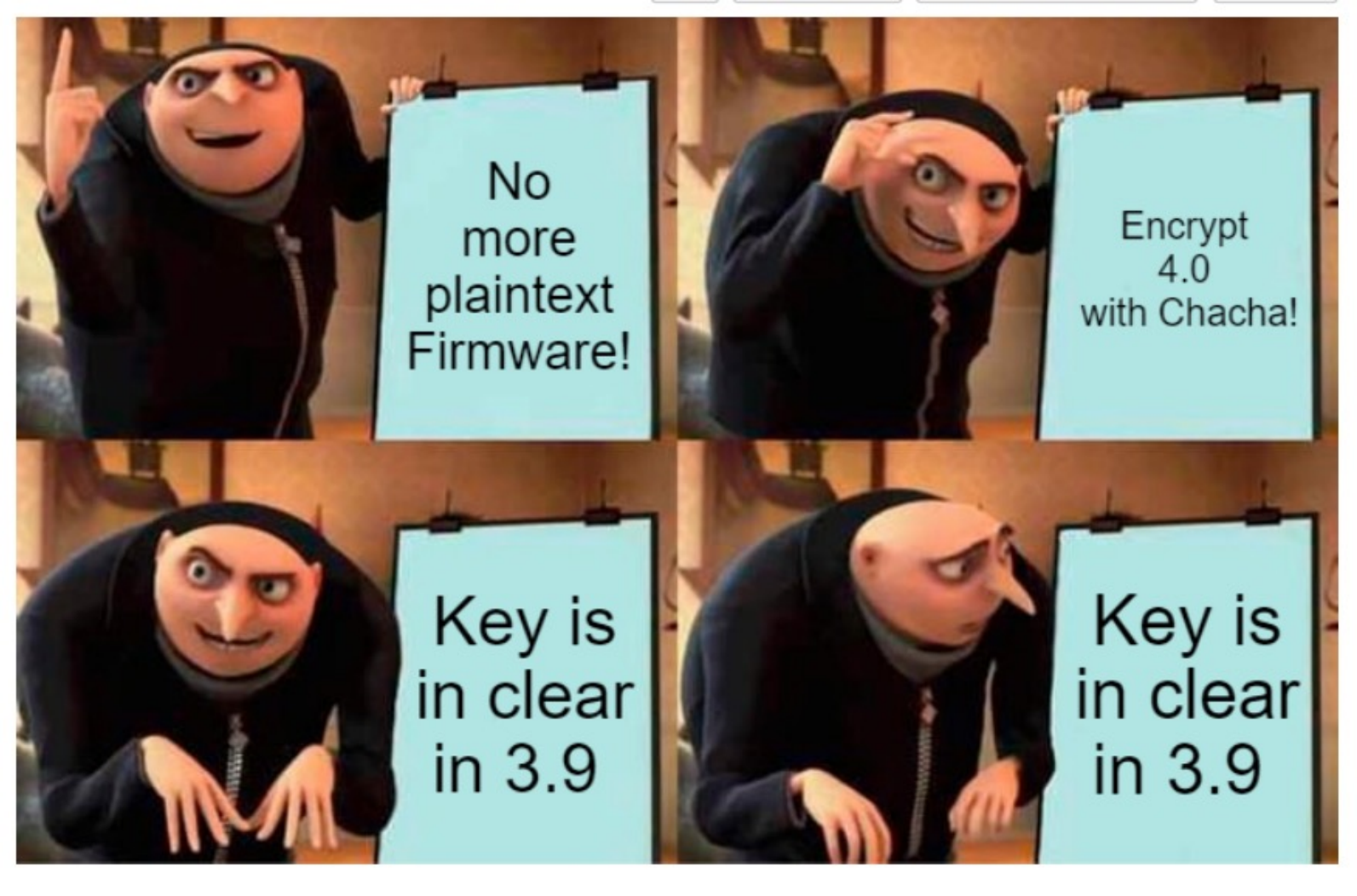
- Load address: 0x5000
- Contains the "decryp
  - → ChaCha or Salsa
  - → Spoiler: It's ChaC
- Cute key mangling

```
STP             X29, X30, [SP,#var_180]!
MOV             X29, SP
STP             X21, X22, [SP,#0x180+var_160]
ADRP            X21, #qword_57C310@PAGE
STP             X19, X20, [SP,#0x180+var_170]
ADD             X7, X21, #qword_57C310@PAGEOFF
MOV             X19, X1
ADRL            X1, a0draytekkd5jas ; "0DraytekKd5Jason[redacted]"
MOV             X20, X0
LDR             X6, [X7]
STR             X6, [X29,#0x180+var_8]
MOV             X6, #0
ADD             X0, X29, #0x180+chacha_key ; dst
STP             X23, X24, [SP,#0x180+var_150]
MOV             X24, X2
MOV             X2, #0x20 ; ' ' ; sz
ADRL            X22, aJason ; "Jason"
MOV             W23, #0x45 ; 'E'
BL              memcpy
STR             XZR, [X29,#0x180+nonce]
MOV             X1, X22
ADD             X0, X29, #0x180+chacha_key
STR             WZR, [X29,#0x180+var_70]
BL              findstr
```

```
loc_505658
STRB            W23, [X0]
MOV             X1, X22
ADD             X0, X29, #0x180+chacha_key
BL              findstr
B               loc_5055C8 ; I think it replaces the key from
                           ; 0DraytekKd5Jason[redacted]
                           ; to 0DraytekKd5Eason[redacted]
```

```python
import sys
from Crypto.Cipher import ChaCha20

def usage():
    print("{} path_to_file path_to_nonce".format(sys.argv[0]))

def go():
    if len(sys.argv) < 2:
        return usage()

    with open(sys.argv[1], "rb") as f:
            data = f.read()
    with open(sys.argv[2], "rb") as f:
            msg_nonce = f.read()

    if len(msg_nonce) != 0xC:
        print("Invalid nonce len")
        return

    secret = b"0DraytekKd5Easc[redacted]
    cipher = ChaCha20.new(key=secret, nonce= msg_nonce)
    plaintext = cipher.decrypt(data)

    with open(sys.argv[1] + "_decrypted", "wb") as f:
            f.write(plaintext)


if __name__ == "__main__":
        go()
```

Firmware Encryption



The FW is decrypted, now what?

```
pl@pl-VirtualBox:~/re/draytek/scripts/from_vm/fw_unpacker/out$ binwalk enc_Image_decrypted

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             xz compressed data
240144        0x3A9D0         Unix path: /lib/firmware/updates/4.9.0-OCTEONTX_SDK_6_2_0_p3_build_38
7434240       0x717C40        Copyright string: "Copyright(c) 1999-2006 Intel Corporation"
7482232       0x7295F8        Copyright string: "Copyright 2005-2007 Rodolfo Giometti <giometti@linux.it>"
7483232       0x7299E0        Copyright string: "Copyright(c) Pierre Ossman"
7484232       0x729DC8        Unix path: /sys/firmware/devicetree/base
7485232       0x72A1B0        Unix path: /sys/firmware/fdt': CRC check failed
7485232       0x72A1B0        Neighborly text, "neighbor table overflow!ate is %x"
7485232       0x72A1B0        LZ4 compressed data, legacy
7485232       0x72A1B0        Unix path: /home/ruby/X
7485232       0x72A1B0        Unix path: /home/ruby/X
7485232       0x72A1B0        Copyright string: "Copyright (C) 2018, Thomas G. Lane, Guido Vollbeding"
7485232       0x72A1B0        Copyright string: "Copyright (C) 2018, Thomas G. Lane, Guido Vollbeding"
7485232       0x72A1B0        Neighborly text, "neighbor C %s"
7485232       0x72A1B0        gzip compressed data, ASCII, last modified: 2043-12-28 15:17:23 (bogus date)
7485232       0x72A1B0        ELF, 64-bit LSB
7485232       0x72A1B0        Copyright string: "Copyright (C) 2018, Thomas G. Lane, Guido Vollbeding"
7485232       0x72A1B0        Executable script, shebang: "/bin/bash"
7485232       0x72A1B0        Executable script, shebang: "/bin/bash"
7485232       0x72A1B0        gzip compressed data, ASCII, last modified: 2042-08-01 06:45:20 (bogus date)
```

```
pl@pl-VirtualBox:~/re/draytek/scripts/from_vm/fw_unpacker/out$ strings enc_Image_decrypted | grep "Linux"
Linux version 4.9.0-OCTEONTX_SDK_6_2_0_p3_build_38 (jenkins@cavium-autobuild) (gcc version 6.2.0 (Cavium Inc. Version 2.1 build 97) )
 #2 SMP PREEMPT Tue Apr 19 00:49:46 CST 2022
Linux
No working init found.  Try passing init= option to kernel. See Linux Documentation/init.txt for guidance.
6Booting Linux on physical CPU 0x%lx
0-0     Linux                   [kernel]
```

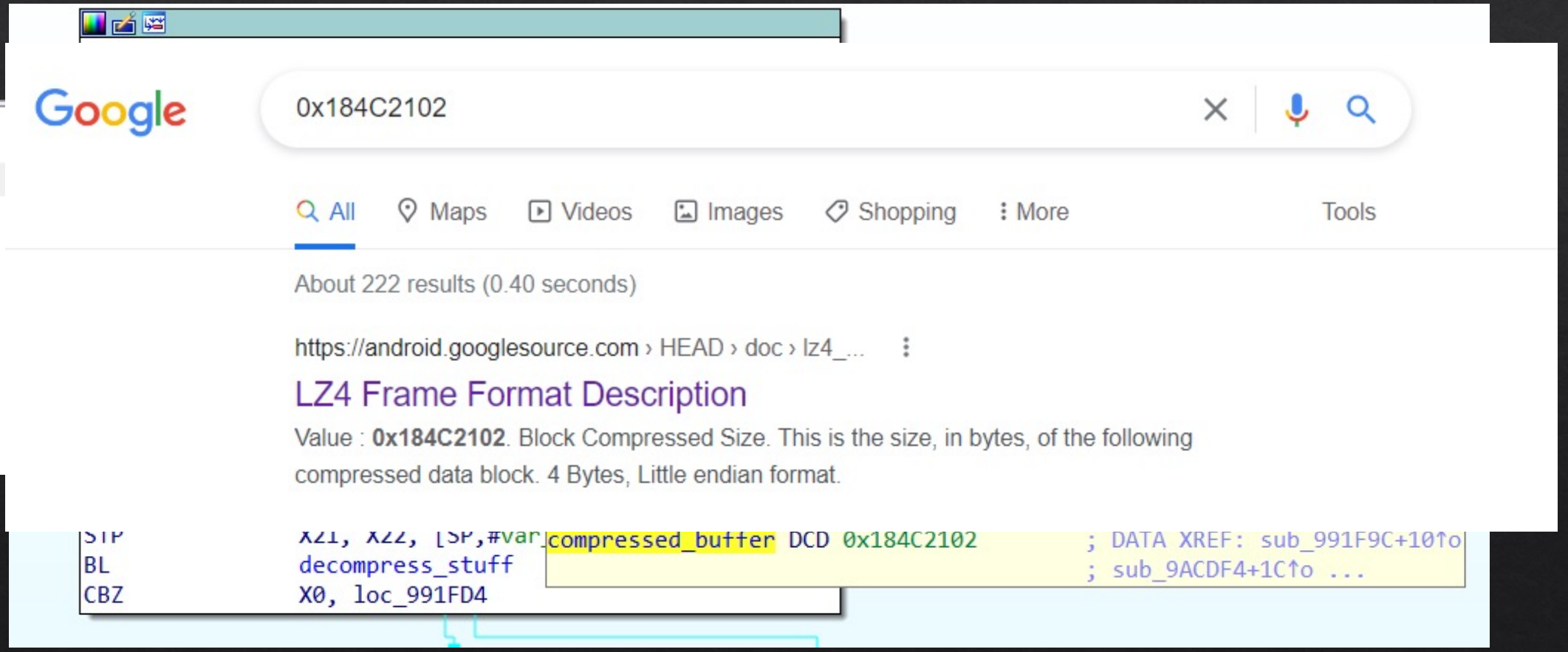
Linux? Show me your filesystem!

Filesystem

- The FS is embedded inside the Linux Kernel
- Linux Kernel is somehow a PE file inside the firmware fil
- Let's:
  - Extract Linux Kernel with binwalk
  - Import it in IDA
  - See how it decompresses the FS

Decompression

- Linux Kernel in Vigor 3910 firmware:



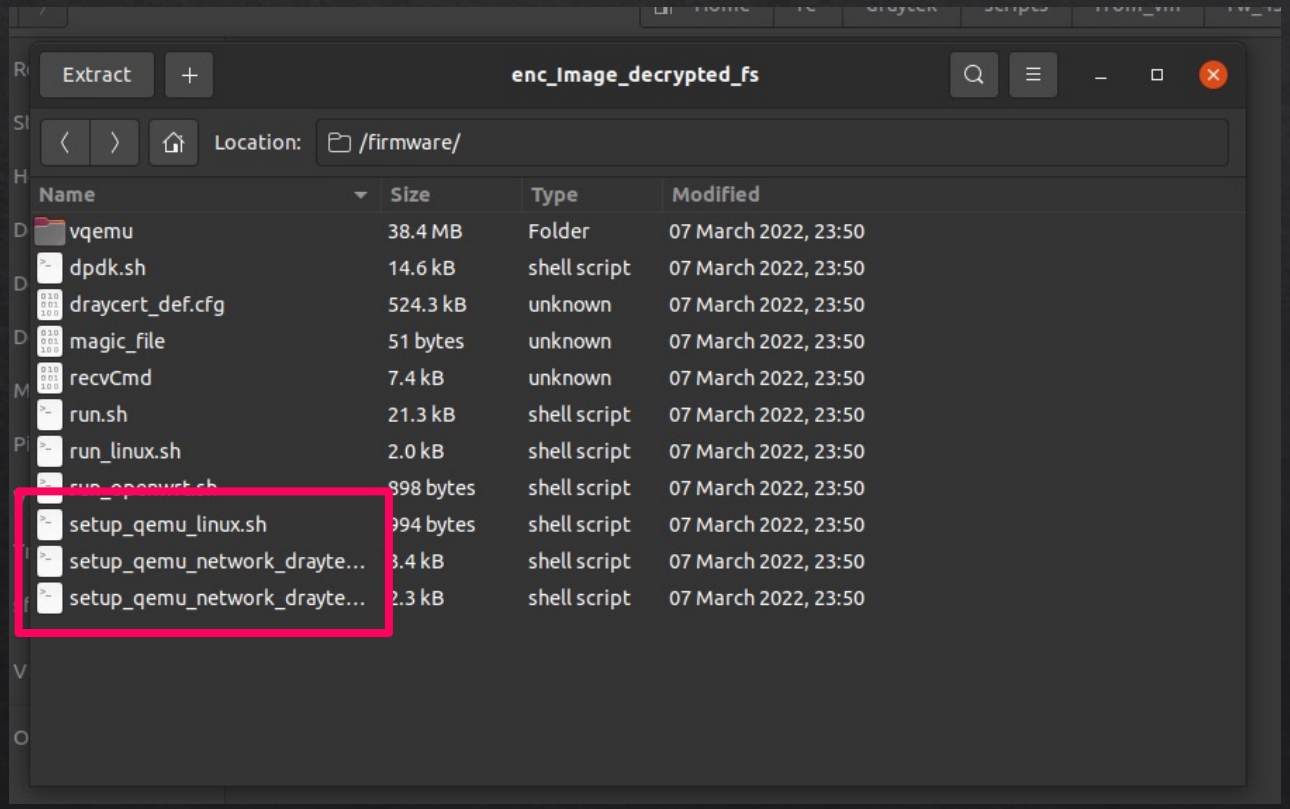
Decompression

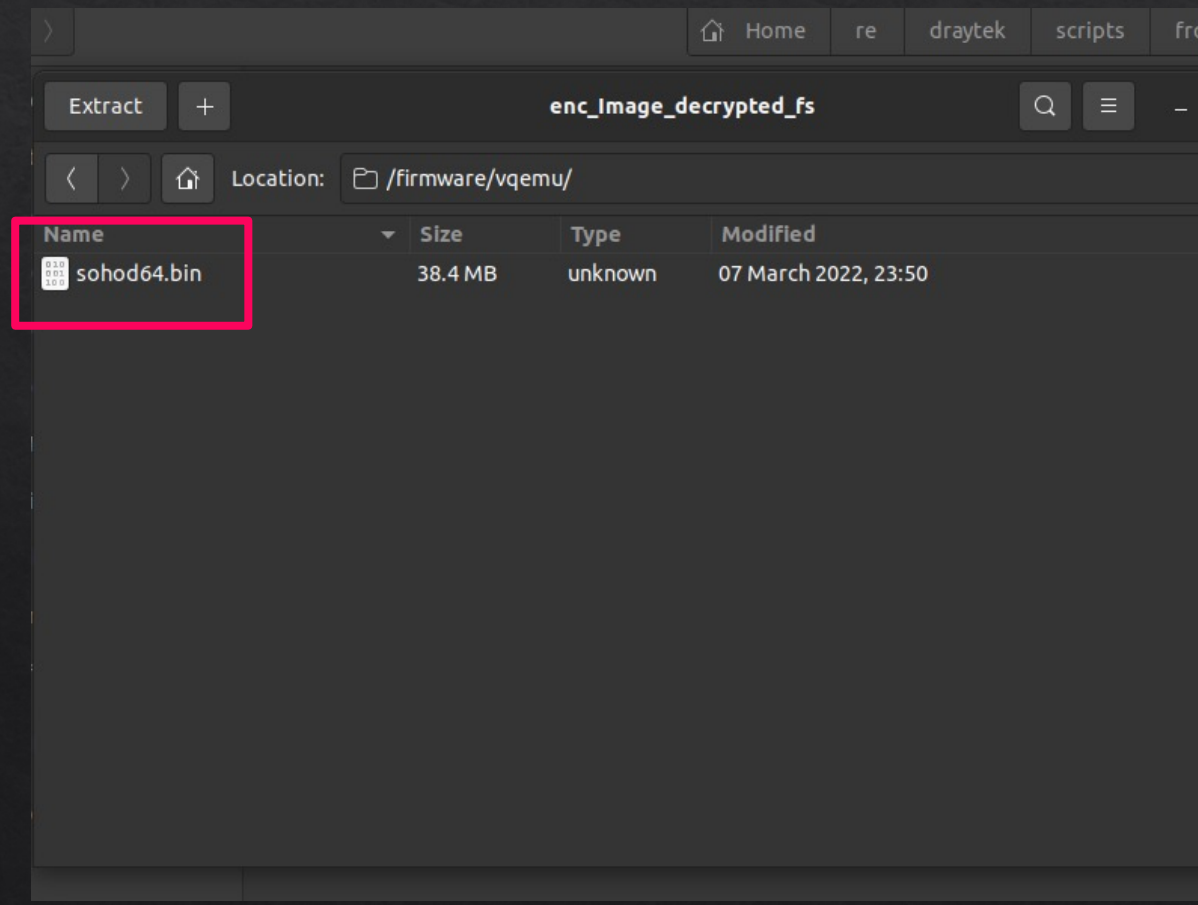
- Dump compressed buffer to a file, run lz4 utility on it.
- This is tedious
  - Especially if you want to analyze as many version as possible…
  - Automation could be an option (e.g. IDA headless) but also tedious
- Yolo mode: instead of importing into IDA we can look for magic strings
  - LZ4 files starts with 0x184C2102
  - End with TRAILER!!!
  - Try to extract the blob and run lz4 on that

```python
import sys
import os
#import lz4.frame # pip install lz4



def go():
    if len(sys.argv) < 2:
        print("Usage: {} image_to_extract".format(sys.argv[0]))


    with open(sys.argv[1], "rb") as f:
        data = f.read()

    """
    The linux kernel is a MZ/PE File (lol why!!) it has embedded the filesystem compressed as LZ4.
      -> search for string "decompression failed" and see calling function
      -> see see \Vigor3910_v3.9.7.2  -> mz_file_good -> decompress_stuff
           -> calling function will push address of the LZ4 blob and its size (dereferenced from different location)
           -> LZ4 will start with  0x184C2102     and end with TRAILER!!! (...) followed by 0x00000000  (it might be 5 null bytes, not sure)
           -> If LZ4 throws an error " Error 52 : Read error : cannot access compressed block ! " it might be because there's are a few bytes missing on the last block.

    """


    data_start = data.find(b"\x02\x21\x4c\x18") # we assume 1. this is little endian and 2. the first occurence is the good one
    if data_start == -1:
        print("LZ4 header not found")
        return False

    #data_end = data.find(b"TRAILER!!!", data_start)
    data_end = data.find(b"R!!!", data_start)
    temp_end = data_end
    while temp_end != -1:
        data_end = temp_end
        temp_end = data.find(b"R!!!", data_end +1)
    if data_end == -1:
        print("LZ4 trailer not found")
        return False
    data_end += (0x14-5) # data ends with TRAILER!!! followed by a bunch of stuff and a few null bytes or it can be TR_R!!!)

    print("Found LZ4 from {:x} to {:x}".format(data_start, data_end))
    #decompressed = lz4.frame.decompress(data[data_start:data_end])

    filename_out = sys.argv[1] + "_fs.lz4"
    with open(filename_out, "wb") as f:
        f.write(data[data_start:data_end])


    os.system("lz4 -d {}".format(filename_out))
    return True
```

Filesystem?

- The LZ4 decompression gives a CPIO filesystem
- Regular Linux stuff…
- …but firmware folder looks fun!





Vuln Research Time!

How to hack the planet?

- Import sohod64.bin in IDA ?
  - DrayOS image (similar to the ones found on other models)
  - Easiest target → CGI endpoints
- What's up with QEMU?
  - Used to emulate DrayOS (!!!) on the device
  - If they can do it, so can we!

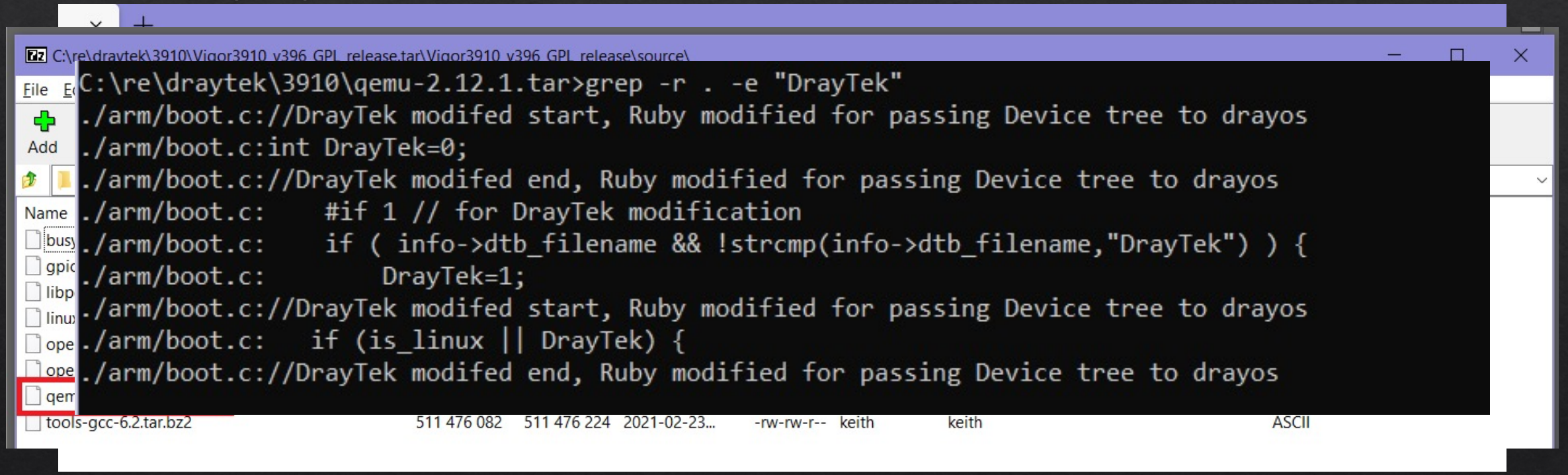
QEMU

- Let's look at this first!
  - Easy semantically, but really both (qemu+reverse) happened in parallel
- Cool script run_linux.sh

```
GCI_PATH="./app/gci"
GCI_FAIL="./app/gci_exp_fail"
GDEF_FILE="$GCI_PATH/draycfg.def"
GEXP_FLAG="$GCI_PATH/EXP_FLAG"
GEXP_FILE="$GCI_PATH/draycfg.exp"
GDEF_FILE_ADDR="0x4de0000"
GEXP_FLAG_ADDR="0x55e0000"
GEXP_FILE_ADDR="0x55e0010"

echo "kyrofang" > $GDEF_FILE
echo "0#" > $GEXP_FLAG
#echo "19831026" > $GEXP_FILE


# Make sure the lan works... shrug lol
(sleep 20 && ethtool -K qemu-lan tx off)&


          #-netdev tap,id=network-lan,ifname=qemu-lan,script=no,downscript=no \
           # -cpu host \
           # --enable-kvm \
           #  -dtb DrayTek
./qemu-system-aarch64 draytek -M virt,gic_version=3   -cpu cortex-a57  \
          -m 1024  -dtb DrayTek \
          -kernel ./vqemu/sohod64.bin $serial_option \
          -nographic  $gdb_serial_option $gdb_remote_option \
          -device virtio-net-pci,netdev=network-lan,mac=${LAN_MAC} \
          -netdev tap,id=network-lan,ifname=qemu-lan,script=no,downscript=no  \
          -device virtio-net-pci,netdev=network-wan,mac=00:1d:aa:${A}:${B}:$(wan_mac 1) \
          -netdev tap,id=network-wan,ifname=qemu-wan,script=no,downscript=no \
          -device virtio-serial-pci -chardev pipe,id=ch0,path=serial0 \
          -device virtserialport,chardev=ch0,name=serial0 \
          -device virtio-serial-pci -chardev pipe,id=ch1,path=serial1 \
          -device virtserialport,chardev=ch1,name=serial1 \
          -monitor telnet:127.0.0.1:7777,server,nowait \
          -device loader,file=$platform_path,addr=0x25fff0 \
          -device loader,file=$cfg_path,addr=0x260000 \
          -device loader,file=gci_magic,addr=0x4de0000 \
          -device loader,file=$enable_kvm_path,addr=0x25ffe0 \
          -device loader,file=$uffs_flash,addr=0x00be0000 \
          -device loader,file=memsize,addr=0x25ff67 \
          -device loader,file=$model_path,addr=0x25ff69 \
          -device loader,file=$GEXP_FLAG,addr=$GEXP_FLAG_ADDR \
          -device loader,file=$GEXP_FILE,addr=$GEXP_FILE_ADDR \
         #-device loader,file=$GDEF_FILE,addr=$GDEF_FILE_ADDR \
         # -device loader,file=$test_verbose_path,addr=0x46B000A8
```

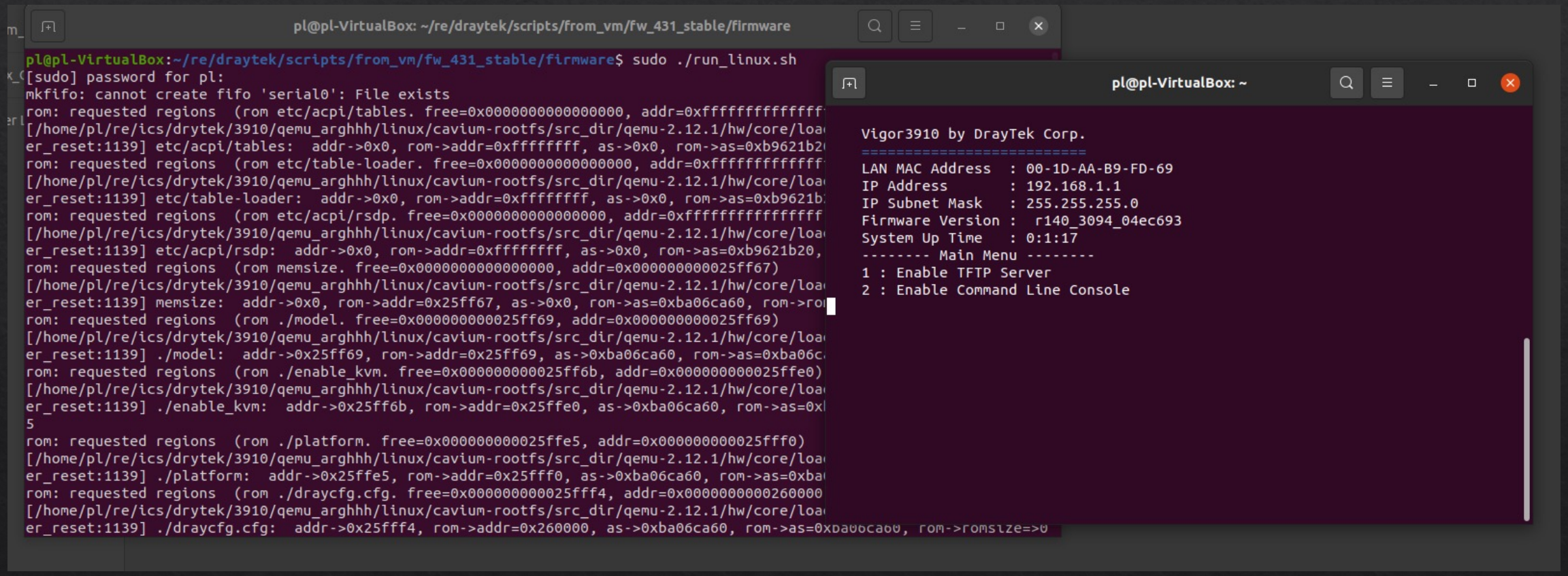
QEMU

- -dtb DrayTek ?
  - Lol! Not legit flag….



QEMU

- It boots!!!!



QEMU

- It boots!!!!
- Answer to pings….
- ……. But can't connect to the web interface.

The connection has timed out

An error occurred during a connection to 192.168.1.1.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

Try Again