Reversing a Japanese Wireless SD Card With r2

@guedou

#r2con2018

radare2 And Me

mainly for this project

r2scapy, jupyter-radare2

r2con 2016 speaker

r2m2

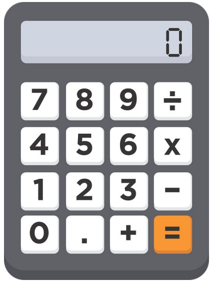
issues / PR

fuzzing r2 commands

```
=< dt
pd0~{}
V`````````````````````````h|`iVKd|`
dd .,,.,,,,,,,,,
dd A
omn
pvj 30~{
ds 10~{pd
.dd dd)ddd dddr{G
p=p" 1 " 1 EN
aek-
*455%R*:**************
(((((((,(*;(((($(($(((((*<((()(((d
tk -!!!! d'B%CCCC!9!!!!_
55ddd ddD@@ @ d
22.22.22ddd d do@
pbij 10g{j 10g{|
aespd +oobt
paadd [d,[,R,1d
axf.
td B()17ecS? [
woDj01
p==pv 1oom 6e
ge;aespe
```

2015

?



2018

Get the slides at https://goo.gl/PpMjNf

Black Hat US slides at https://goo.gl/oijvdN

Toshiba FlashAir

Main Features

access files over Wi-Fi

SSID: flashair_{MAC address}
PSK: 12345678

provide some services

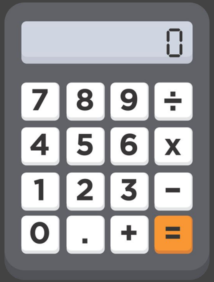
DHCP, DNS, HTTP

configured with SD_WLAN/CONFIG

See https://www.flashair-developers.com/en/documents/api/config/

Game Plan

- □ memory dump
- □ architecture
- □ Operating System
- □ execution vector



Inspecting Firmwares Updates

This talk focuses
on firmware v3.00.00

Extracting the Firmware

download the Mac OS zip file

unzip the .app

explore `Contents/Resources`

CONFIG files
fwupdate.fbn (~1MB)

```
$ r2 zip://FlashAirFWUpdateToolV3_v30002.zip::36
```

reading fwupdate.fbn

Operation of The Software Update Tool

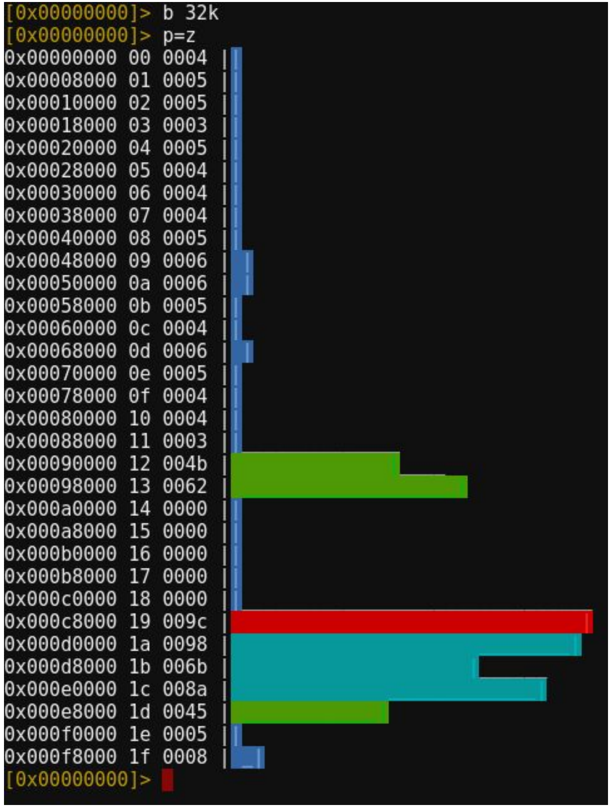
copy fwupdate.fbn to the card

add the following line to SD_WLAN/CONFIG

```
COMMAND=update -f fwupdate.fbn -rm -reboot
```

eject & insert the card

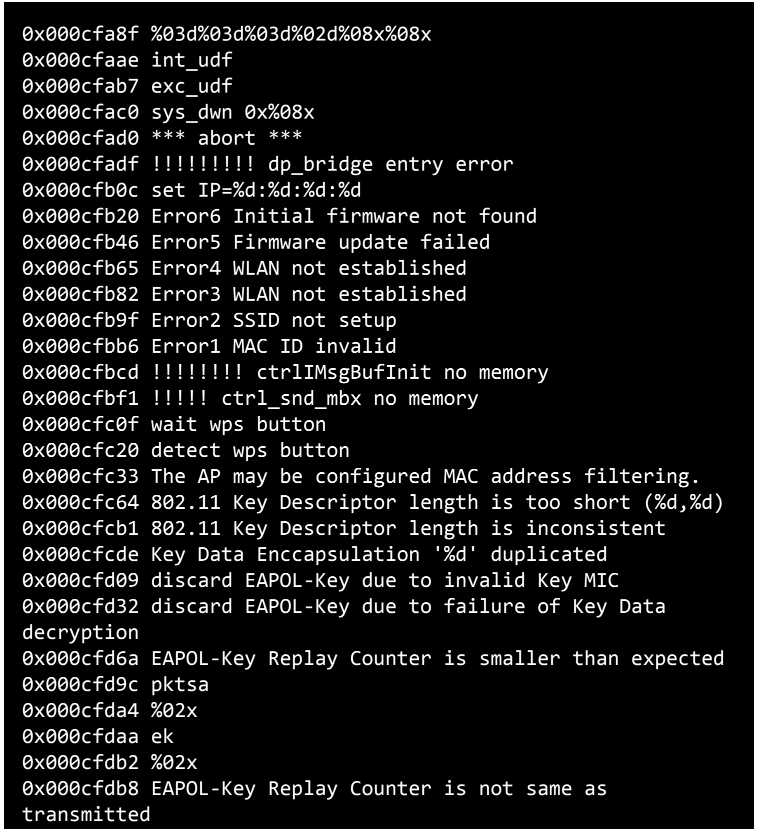
Searching Strings with radare2



```
[0x00000000]> b 32k
[0x00000000]> p=z
0x00000000 00 0004 |
0x00008000 01 0005 |
0x00010000 02 0005 |
0x00018000 03 0003 |
0x00020000 04 0005 |
0x00028000 05 0004 |
0x00030000 06 0004 |
0x00038000 07 0004 |
0x00040000 08 0005 |
0x00048000 09 0006 |
0x00050000 0a 0006 |
0x00058000 0b 0005 |
0x00060000 0c 0004 |
0x00068000 0d 0006 |
0x00070000 0e 0005 |
0x00078000 0f 0004 |
0x00080000 10 0004 |
0x00088000 11 0003 |
0x00090000 12 004b |
0x00098000 13 0062 |
0x000a0000 14 0000 |
0x000a8000 15 0000 |
0x000b0000 16 0000 |
0x000b8000 17 0000 |
0x000c0000 18 0000 |
0x000c8000 19 009c |
0x000d0000 1a 0098 |
0x000d8000 1b 006b |
0x000e0000 1c 008a |
0x000e8000 1d 0045 |
0x000f0000 1e 0005 |
0x000f8000 1f 0008 |
[0x00000000]>
```

```
b 32k
p=z
```

```
s 0xc80000
psb
```



```
0x000cfa8f %03d%03d%03d%02d%08x%08x
0x000cfaae int_udf
0x000cfab7 exc_udf
0x000cfac0 sys_dwn 0x%08x
0x000cfad0 *** abort ***
0x000cfadf !!!!!!!!! dp_bridge entry error
0x000cfb0c set IP=%d:%d:%d:%d
0x000cfb20 Error6 Initial firmware not found
0x000cfb46 Error5 Firmware update failed
0x000cfb65 Error4 WLAN not established
0x000cfb82 Error3 WLAN not established
0x000cfb9f Error2 SSID not setup
0x000cfbb6 Error1 MAC ID invalid
0x000cfbcd !!!!!!!! ctrlIMsgBufInit no memory
0x000cfbf1 !!!!! ctrl_snd_mbx no memory
0x000cfc0f wait wps button
0x000cfc20 detect wps button
0x000cfc33 The AP may be configured MAC address filtering.
0x000cfc64 802.11 Key Descriptor length is too short (%d,%d)
0x000cfcb1 802.11 Key Descriptor length is inconsistent
0x000cfcde Key Data Enccapsulation '%d' duplicated
0x000cfd09 discard EAPOL-Key due to invalid Key MIC
0x000cfd32 discard EAPOL-Key due to failure of Key Data
decryption
0x000cfd6a EAPOL-Key Replay Counter is smaller than expected
0x000cfd9c pktsa
0x000cfda4 %02x
0x000cfdaa ek
0x000cfdb2 %02x
0x000cfdb8 EAPOL-Key Replay Counter is not same as
transmitted
```

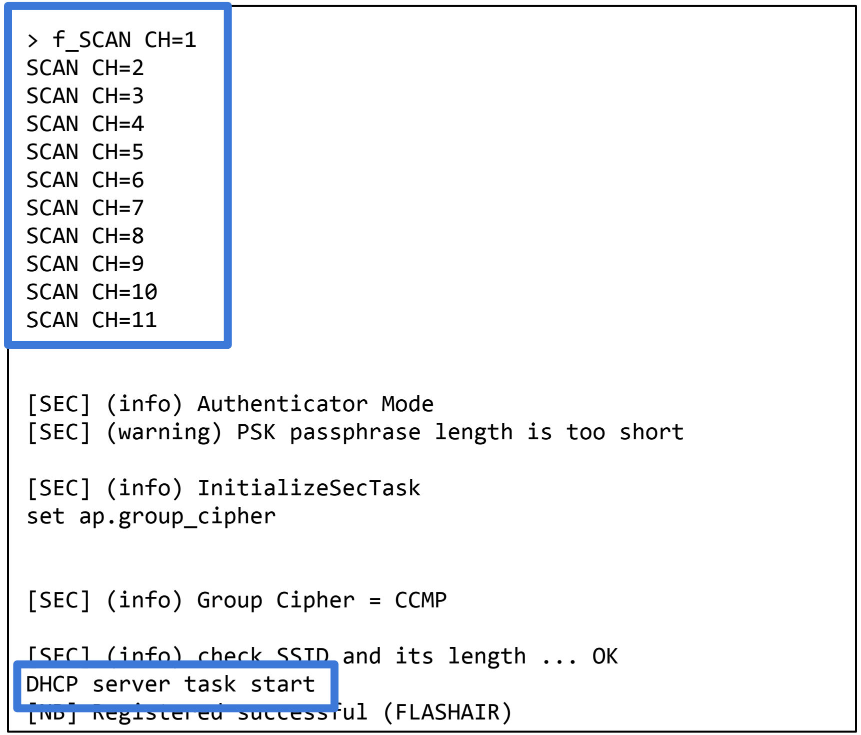
“/eva.cgi”

access it over HTTP

http://192.168.0.1/eva.cgi

looks like the output buffer

information, warnings ...



```
> f_SCAN CH=1
SCAN CH=2
SCAN CH=3
SCAN CH=4
SCAN CH=5
SCAN CH=6
SCAN CH=7
SCAN CH=8
SCAN CH=9
SCAN CH=10
SCAN CH=11


[SEC] (info) Authenticator Mode
[SEC] (warning) PSK passphrase length is too short

[SEC] (info) InitializeSecTask
set ap.group_cipher


[SEC] (info) Group Cipher = CCMP

[SEC] (info) check SSID and its length ... OK
DHCP server task start
[NB] Registered successful (FLASHAIR)
```

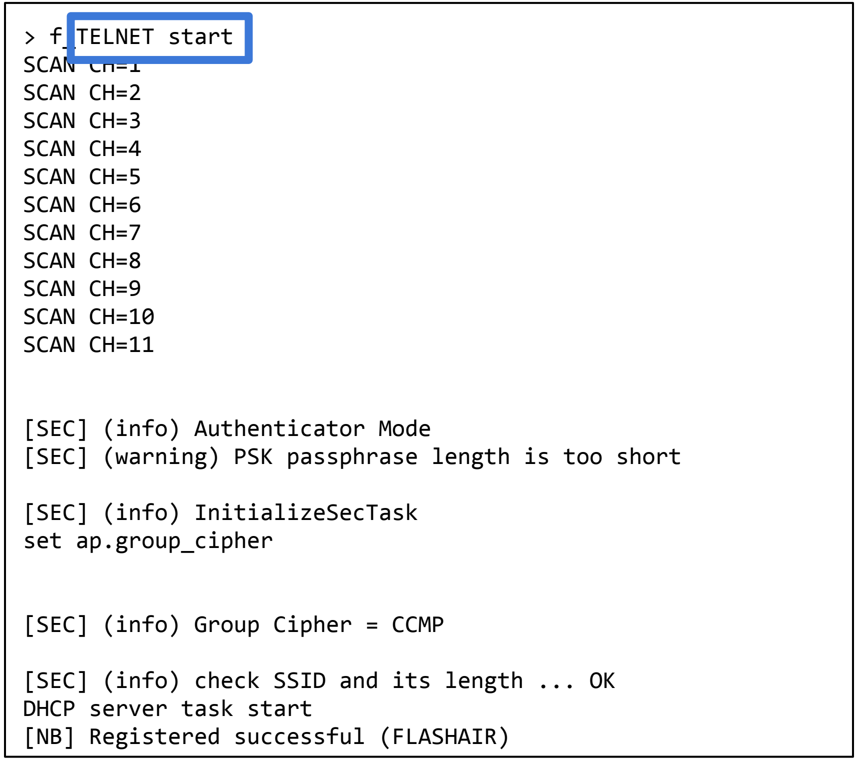
“TELNET”

edit SD_WLAN/CONFIG with

TELNET=1

telnet daemon on 23/tcp

character per character



```
> f_TELNET start
SCAN CH=1
SCAN CH=2
SCAN CH=3
SCAN CH=4
SCAN CH=5
SCAN CH=6
SCAN CH=7
SCAN CH=8
SCAN CH=9
SCAN CH=10
SCAN CH=11


[SEC] (info) Authenticator Mode
[SEC] (warning) PSK passphrase length is too short

[SEC] (info) InitializeSecTask
set ap.group_cipher


[SEC] (info) Group Cipher = CCMP

[SEC] (info) check SSID and its length ... OK
DHCP server task start
[NB] Registered successful (FLASHAIR)
```

Asking for Help

COMMAND=help in CONFIG

restart & check /eva.cgi

TELNET=1 in CONFIG

type `help` in telnet session

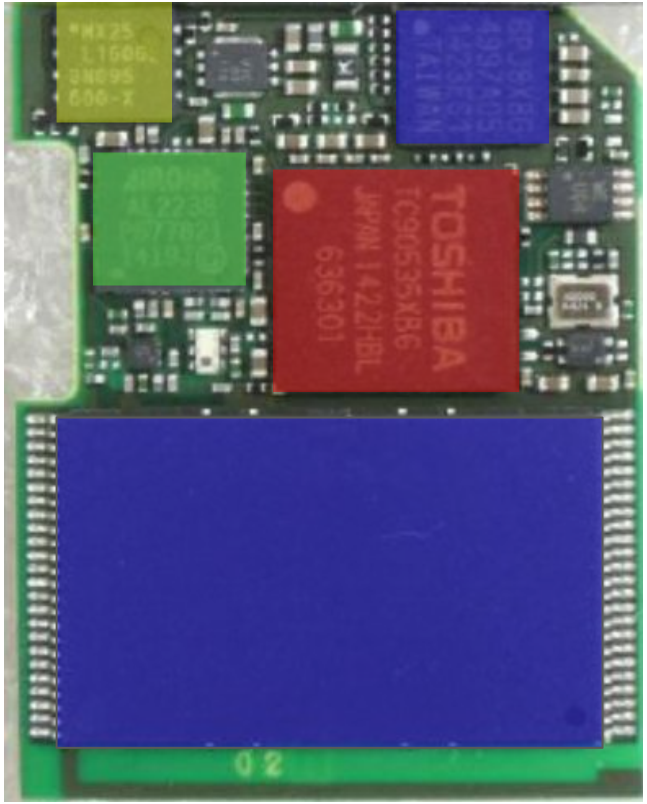
```
help              show help
version           show version
mod               Modify Memory
fdump             Memory dump to file
dump              Dump Memory
-- >8 --
```

32

commands

Inspecting the Card

FlashAir W-03 Innards



🟦 Toshiba TC58TFG7DDLTAID: Flash memory

Toshiba 6PJ8XBG: Flash Memory controller

🟨 SPI - USON-8 4x4 mm - 2MB

- Macronix - MX25L1606E
- Winbond - Q16DVUZIG

🟩 Airoha AL2238: 802.11 b/g - RF transceiver

🟥 Toshiba TC90535XBG: the SoC

- 32-bit RISC
- released in 2013

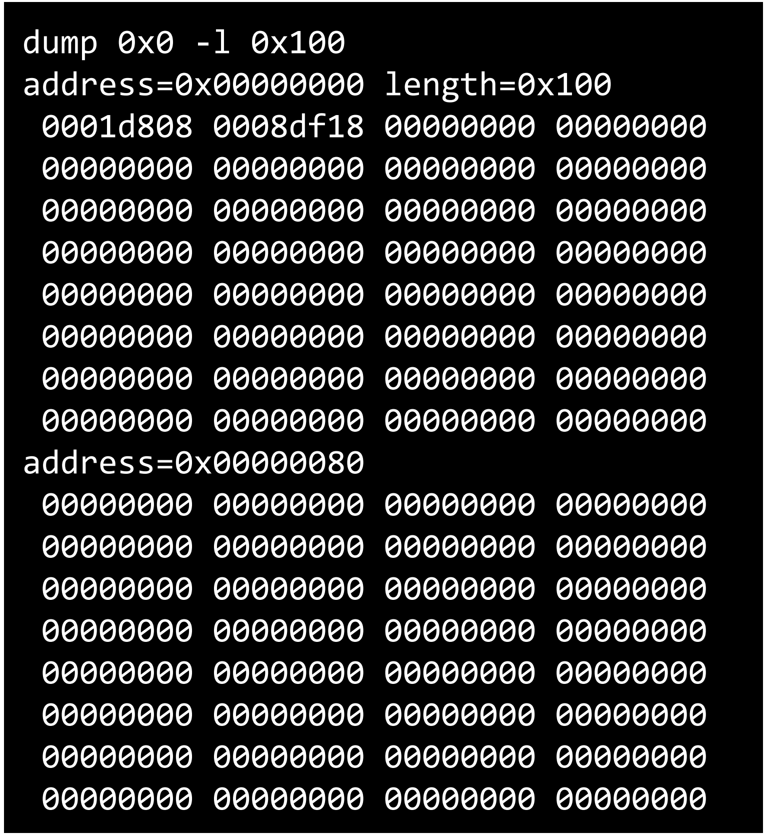
Dumping Memory

Software Based Dump

CONFIG & TELNET commands

`fdump` - write memory to files

`dump` - print memory content



```
dump 0x0 -l 0x100
address=0x00000000 length=0x100
 0001d808 0008df18 00000000 00000000
 00000000 00000000 00000000 00000000
 00000000 00000000 00000000 00000000
 00000000 00000000 00000000 00000000
 00000000 00000000 00000000 00000000
 00000000 00000000 00000000 00000000
 00000000 00000000 00000000 00000000
 00000000 00000000 00000000 00000000
address=0x00000080
 00000000 00000000 00000000 00000000
 00000000 00000000 00000000 00000000
 00000000 00000000 00000000 00000000
 00000000 00000000 00000000 00000000
 00000000 00000000 00000000 00000000
 00000000 00000000 00000000 00000000
 00000000 00000000 00000000 00000000
 00000000 00000000 00000000 00000000
```

flashre Tools - https://github.com/guedou/flashre

simplify reversing FlashAir cards

telnet, update, xref ...

automate useful tasks with r2pipe

dump, naming ...

Docker image available

```
$ docker pull guedou/flashre
```

Dumping Memory with `flashre`

```
$ flashre dump dump_w03.txt
```

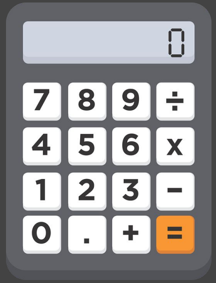
dump

```
$ flashre dump --convert dump_w03.txt > dump_w03.bin
$ ls -alh dump_w03.bin
-rw-rw-r--. 1 guedou guedou 2.0M Aug 08 13:30 dump_w03.bin
```

conversion

Game Plan

- ☒ memory dump
- □ architecture
- □ Operating System
- □ execution vector

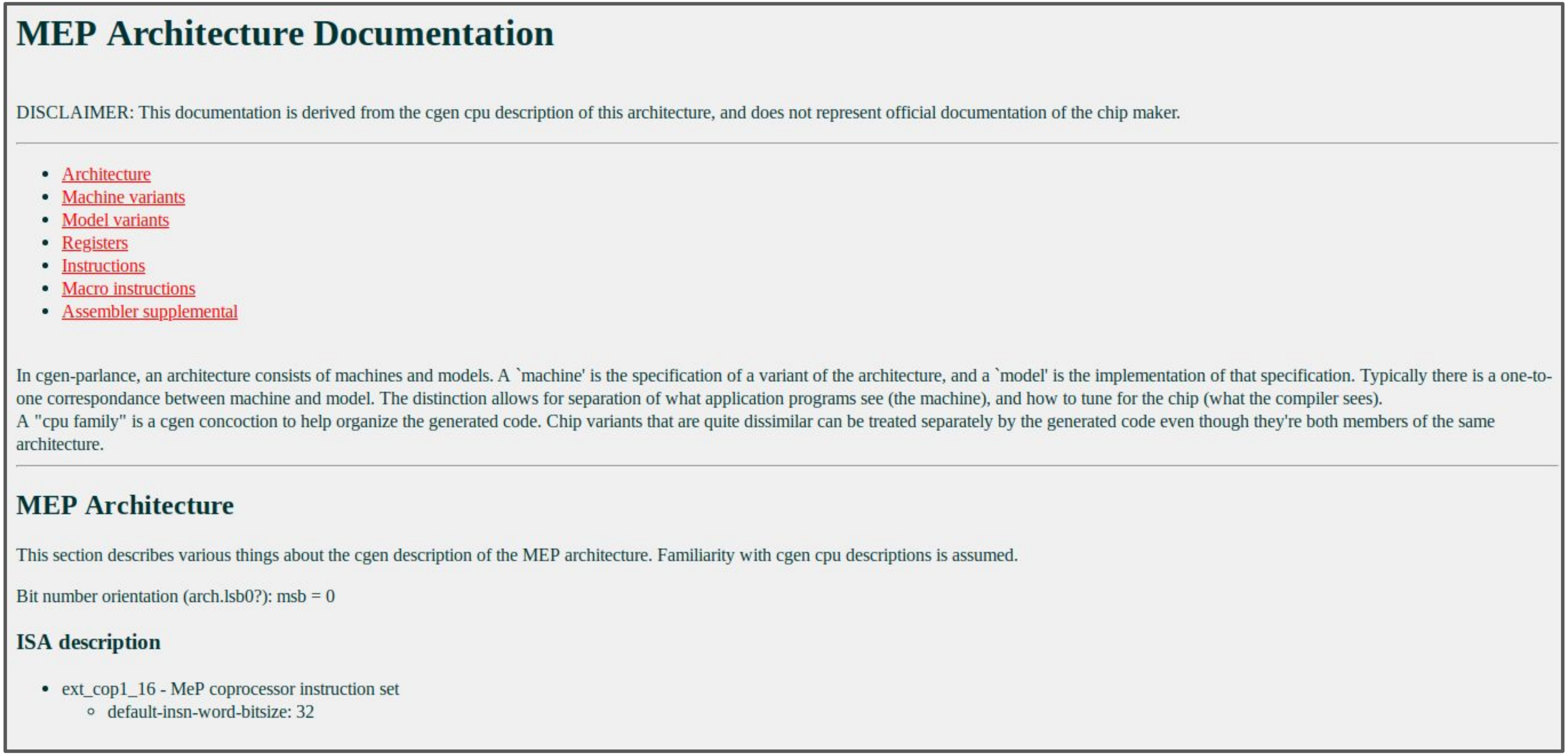


Identifying the CPU

Magic Format Strings

```
R%-2d:%08x  R%-2d:%08x  R%-2d:%08x R%-2d:%08x\n

PSW:%08x LP:%08x NPC:%08x EXC:%08x EPC:%08x\n
```

print registers contents



MEP Architecture Documentation

DISCLAIMER: This documentation is derived from the cgen cpu description of this architecture, and does not represent official documentation of the chip maker.

- Architecture
- Machine variants
- Model variants
- Registers
- Instructions
- Macro instructions
- Assembler supplemental

In cgen-parlance, an architecture consists of machines and models. A `machine' is the specification of a variant of the architecture, and a `model' is the implementation of that specification. Typically there is a one-to-one correspondance between machine and model. The distinction allows for separation of what application programs see (the machine), and how to tune for the chip (what the compiler sees).
A "cpu family" is a cgen concoction to help organize the generated code. Chip variants that are quite dissimilar can be treated separately by the generated code even though they're both members of the same architecture.

MEP Architecture

This section describes various things about the cgen description of the MEP architecture. Familiarity with cgen cpu descriptions is assumed.

Bit number orientation (arch.lsb0?): msb = 0

ISA description

- ext_cop1_16 - MeP coprocessor instruction set
  - default-insn-word-bitsize: 32

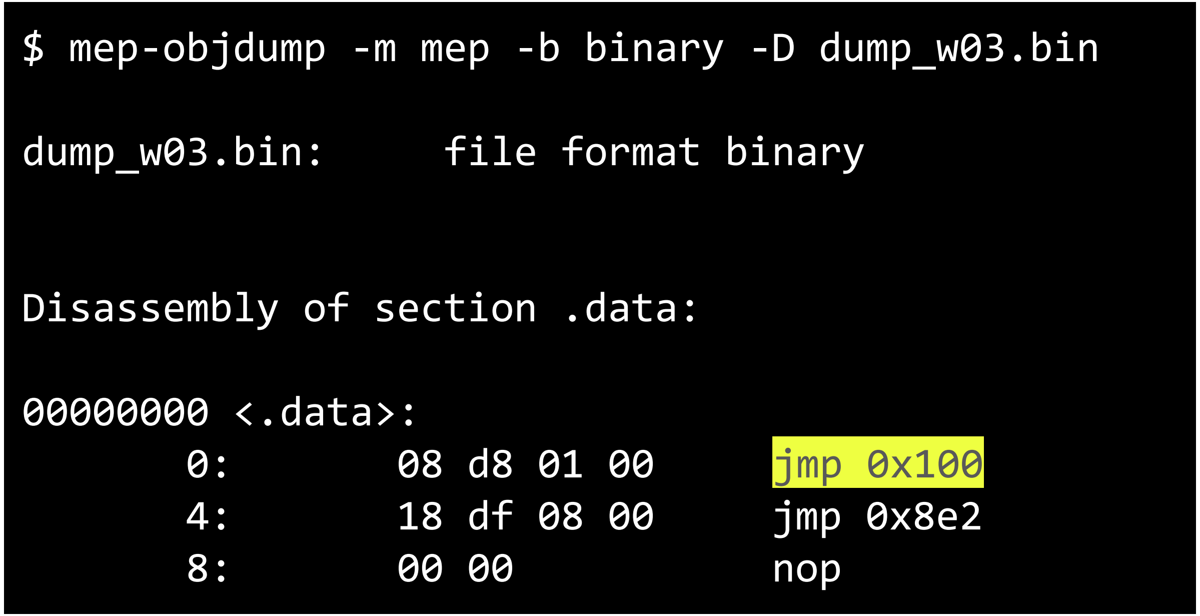
See https://sourceware.org/cgen/gen-doc/mep.htm

cgen - MeP architecture description in guile

Disassembling the Dump

compile binutils with MeP support

```
tar xzf binutils-2.31.tar.gz && cd binutils-2.30 && ./configure --target=mep && make
```



```
$ mep-objdump -m mep -b binary -D dump_w03.bin

dump_w03.bin:     file format binary


Disassembly of section .data:

00000000 <.data>:
       0:       08 d8 01 00     jmp 0x100
       4:       18 df 08 00     jmp 0x8e2
       8:       00 00           nop
```

Where is it Used?

Gigabeat U Info

Image Recognition

Sony PlayStation Vita

Toshiba Media-embedded Processor

MIPS like

load/store, ...

16 general-purpose registers

33 control/special registers

~200 instructions

2 or 4 bytes each

no privileged mode

calling convention

first four registers then stack

32 bits addresses

up to 4GB

Little-Endian or Big-Endian

LEND field in the CFG register

*REPEAT Instructions

REPEAT and EREPEAT

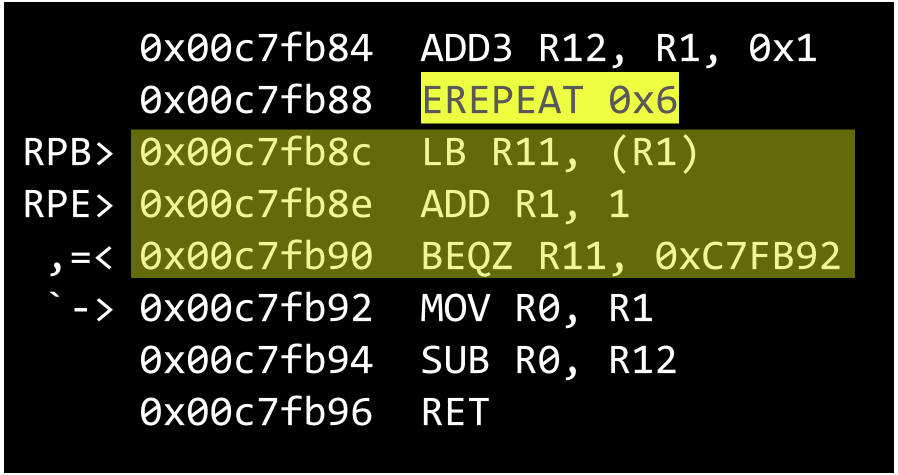
E stands for Endless

three dedicated registers

RPB, RPC, RPE

loop over a block

two instructions executed at RPE



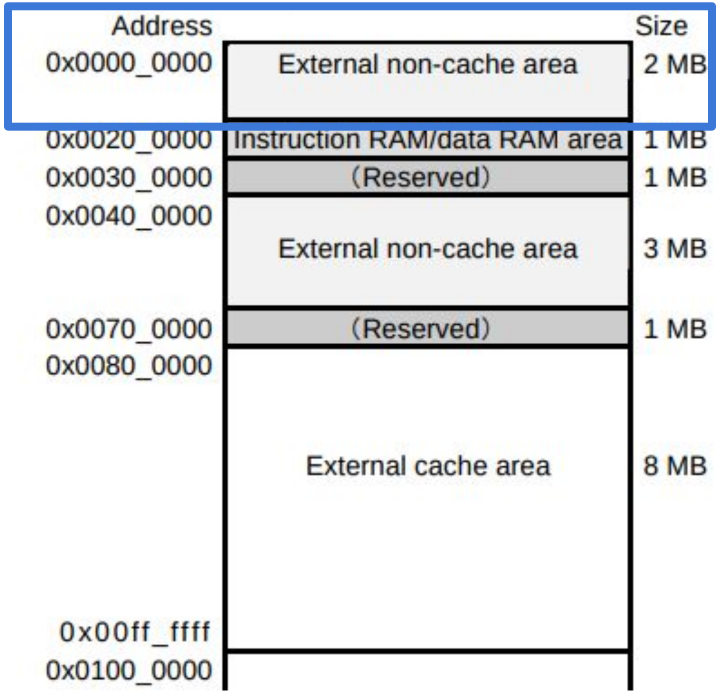
```
      0x00c7fb84  ADD3 R12, R1, 0x1
      0x00c7fb88  EREPEAT 0x6
RPB>  0x00c7fb8c  LB R11, (R1)
RPE>  0x00c7fb8e  ADD R1, 1
 ,=<  0x00c7fb90  BEQZ R11, 0xC7FB92
 `->  0x00c7fb92  MOV R0, R1
      0x00c7fb94  SUB R0, R12
      0x00c7fb96  RET
```

strlen()

Memory Map

flash likely located at 0x00000

- boot program
- reset and NMI handlers



Address		Size
0x0000_0000	External non-cache area	2 MB
0x0020_0000	Instruction RAM/data RAM area	1 MB
0x0030_0000	(Reserved)	1 MB
0x0040_0000	External non-cache area	3 MB
0x0070_0000	(Reserved)	1 MB
0x0080_0000	External cache area	8 MB
0x00ff_ffff		
0x0100_0000		

Guessing The Main Base Address

BSR use signed offset!

offset related to PC

calls can go to lower or higher addresses

```
$ mep-objdump -m mep -b binary -D dump_w03.bin
-- >8 --
   fd27a:   69 d9 26 00 bsr 0xff8a6
```

incorrect BSR address

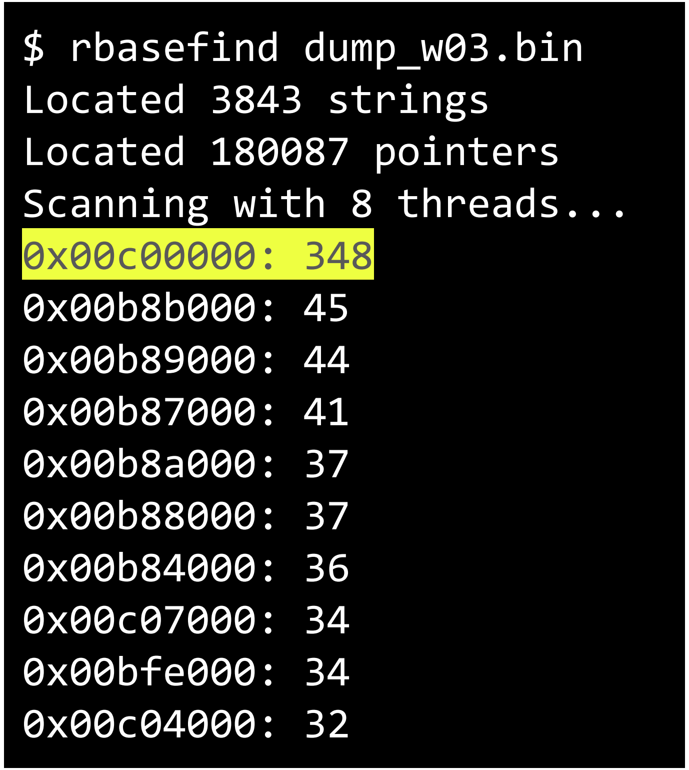
Basefind - Feature request #10725

brute-force base address

in Python2, C++, Rust

steps

1. get string offsets
2. use all words as pointers
3. subtract base from pointers
4. score valid pointers



```
$ rbasefind dump_w03.bin
Located 3843 strings
Located 180087 pointers
Scanning with 8 threads...
0x00c00000: 348
0x00b8b000: 45
0x00b89000: 44
0x00b87000: 41
0x00b8a000: 37
0x00b88000: 37
0x00b84000: 36
0x00c07000: 34
0x00bfe000: 34
0x00c04000: 32
```

See https://github.com/sgayou/rbasefind

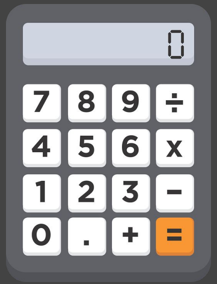
Disassembling Using the Main Base Address

```
$ mep-objdump -m mep -b binary -D dump_w03.bin
-- >8 --
   fd27a:    69 d9 26 00  bsr 0xff8a6

$ mep-objdump -m mep -b binary -D dump_w03.bin --adjust-vma=0xC00000
-- >8 --
  cfd27a:    69 d9 26 00  bsr 0xcff8a6
```

correct BSR address

Game Plan

- ☒ memory dump
- ☒ architecture
- □ operating System
- □ execution vector



MeP Tools

Wish List

disassembly with semantics

split basic blocks

instructions emulation

validate functions behavior

graphical interface (i.e. radare2)

navigate call-graphs, analyse functions, ...

miasm2

Python-based reverse engineering framework

- assemble & disassemble x86, ARM, MIPS, ...
- symbolic execution using intermediate language
- emulation using JIT

simplify defining new architectures

- assembling & disassembling
- expressing semantics

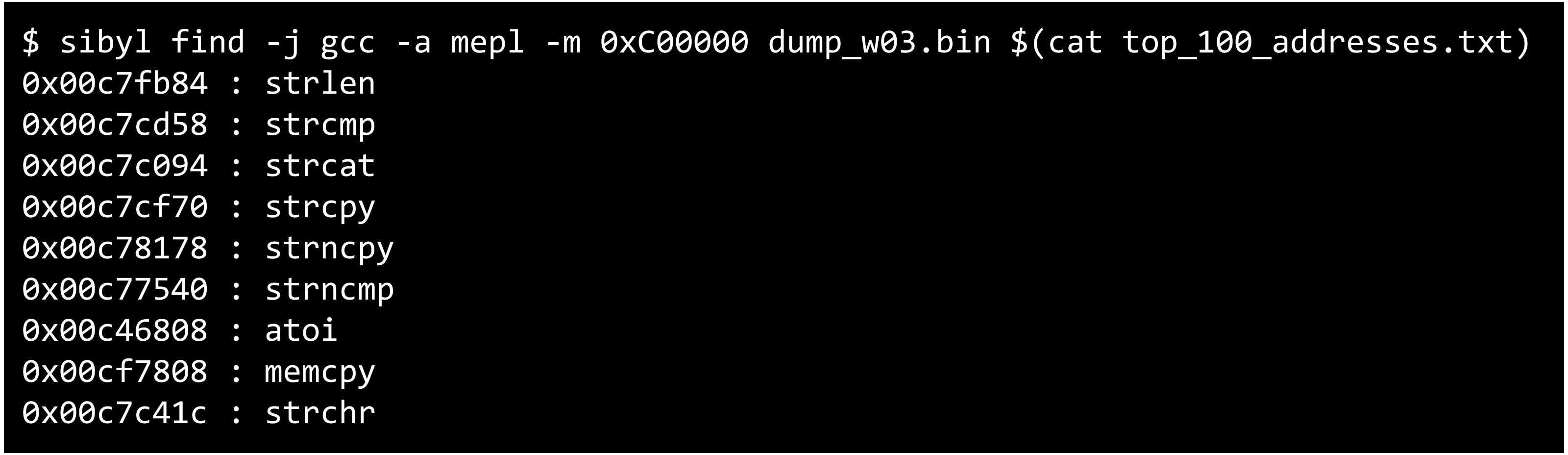
See http://miasm.re & https://github.com/cea-sec/miasm

Sibyl

discover functions using jitters

emulate functions and verify their side effects

an API bruteforcer



```
$ sibyl find -j gcc -a mepl -m 0xC00000 dump_w03.bin $(cat top_100_addresses.txt)
0x00c7fb84 : strlen
0x00c7cd58 : strcmp
0x00c7c094 : strcat
0x00c7cf70 : strcpy
0x00c78178 : strncpy
0x00c77540 : strncmp
0x00c46808 : atoi
0x00cf7808 : memcpy
0x00c7c41c : strchr
```

See https://github.com/cea-sec/Sibyl

9

automatically discovered functions

r2m2 - radare2 + miasm2 = ❤️

use miasm2 features from radare2

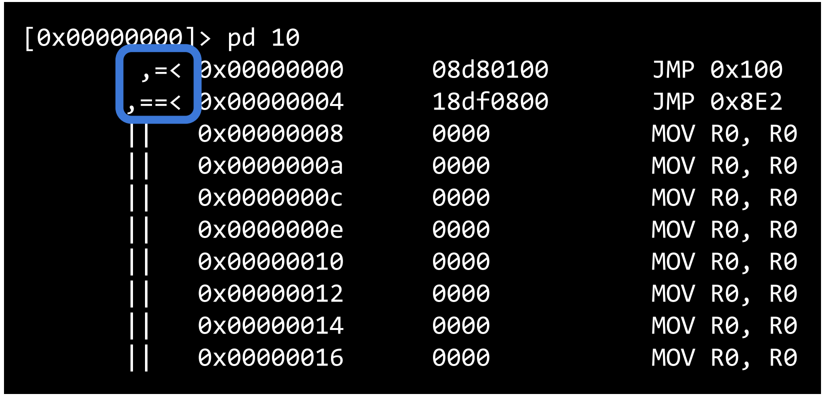
assemble, disassemble, split blocks
convert miasm2 expression to radare2 ESIL

provides two radare2 plugins

ad: assembly & disassembly
Ae: Analysis & emulation

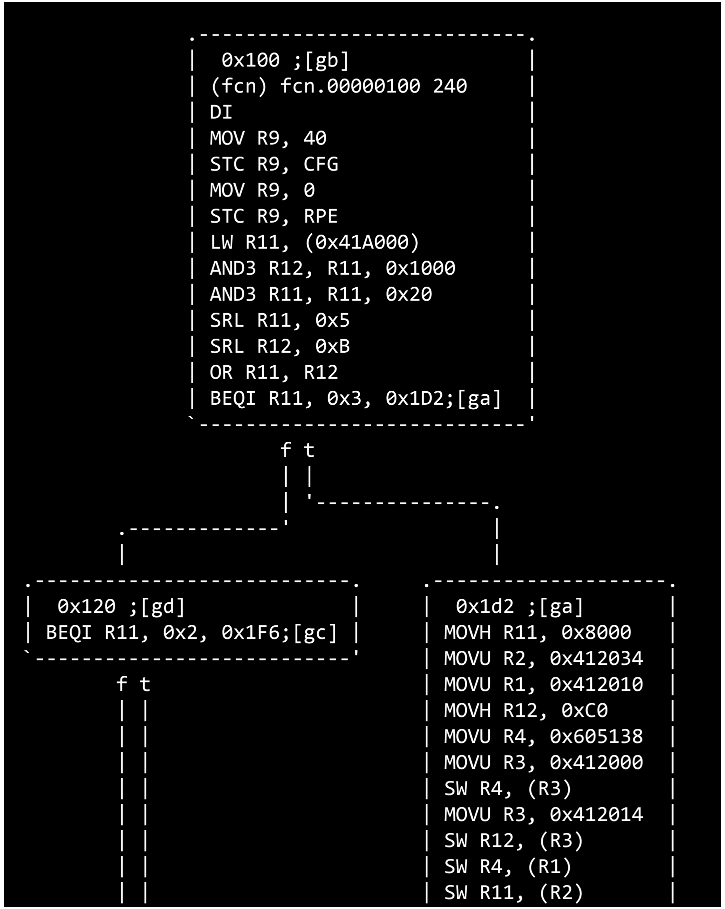
See https://github.com/guedou/r2m2

r2m2_Ae.so - Analysis



```
[0x00000000]> pd 10
        ,=< 0x00000000      08d80100        JMP 0x100
       ,==< 0x00000004      18df0800        JMP 0x8E2
       ||   0x00000008      0000            MOV R0, R0
       ||   0x0000000a      0000            MOV R0, R0
       ||   0x0000000c      0000            MOV R0, R0
       ||   0x0000000e      0000            MOV R0, R0
       ||   0x00000010      0000            MOV R0, R0
       ||   0x00000012      0000            MOV R0, R0
       ||   0x00000014      0000            MOV R0, R0
       ||   0x00000016      0000            MOV R0, R0
```

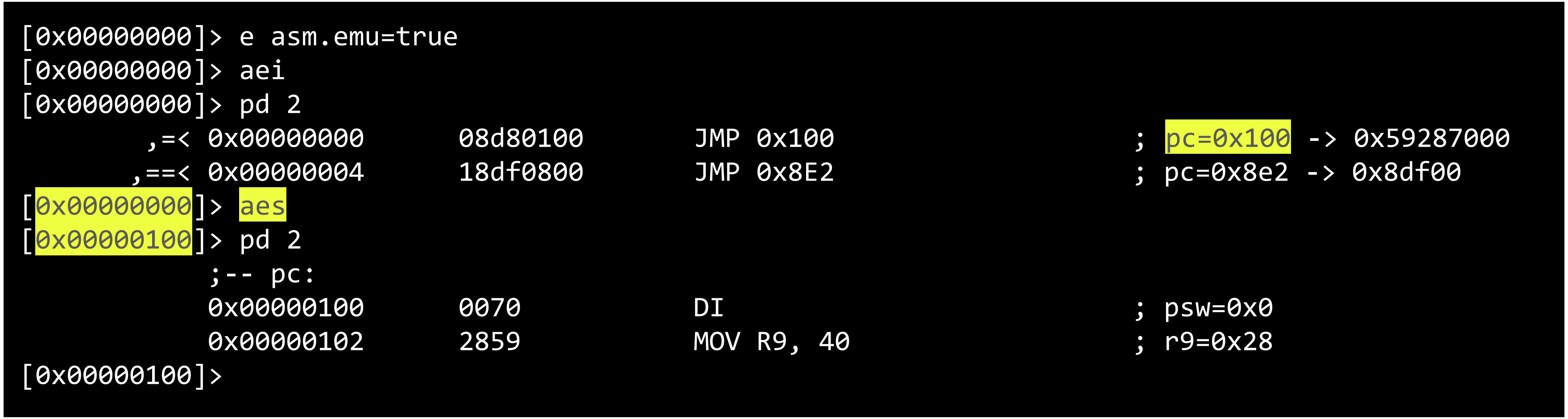
known destinations



```
                .-----------------------------.
                |  0x100 ;[gb]                |
                | (fcn) fcn.00000100 240      |
                | DI                          |
                | MOV R9, 40                  |
                | STC R9, CFG                 |
                | MOV R9, 0                   |
                | STC R9, RPE                 |
                | LW R11, (0x41A000)          |
                | AND3 R12, R11, 0x1000       |
                | AND3 R11, R11, 0x20         |
                | SRL R11, 0x5                |
                | SRL R12, 0xB                |
                | OR R11, R12                 |
                | BEQI R11, 0x3, 0x1D2;[ga]   |
                `-----------------------------'
                       f t
                       | |
                       | '---------------.
          .------------'                 |
          |                              |
 .-----------------------------.   .---------------------.
 |  0x120 ;[gd]                |   |  0x1d2 ;[ga]        |
 | BEQI R11, 0x2, 0x1F6;[gc]   |   | MOVH R11, 0x8000    |
 `-----------------------------'   | MOVU R2, 0x412034   |
         f t                       | MOVU R1, 0x412010   |
         | |                       | MOVH R12, 0xC0      |
         | |                       | MOVU R4, 0x605138   |
         | |                       | MOVU R3, 0x412000   |
         | |                       | SW R4, (R3)         |
         | |                       | MOVU R3, 0x412014   |
         | |                       | SW R12, (R3)        |
         | |                       | SW R4, (R1)         |
         | |                       | SW R11, (R2)        |
```

callgraph

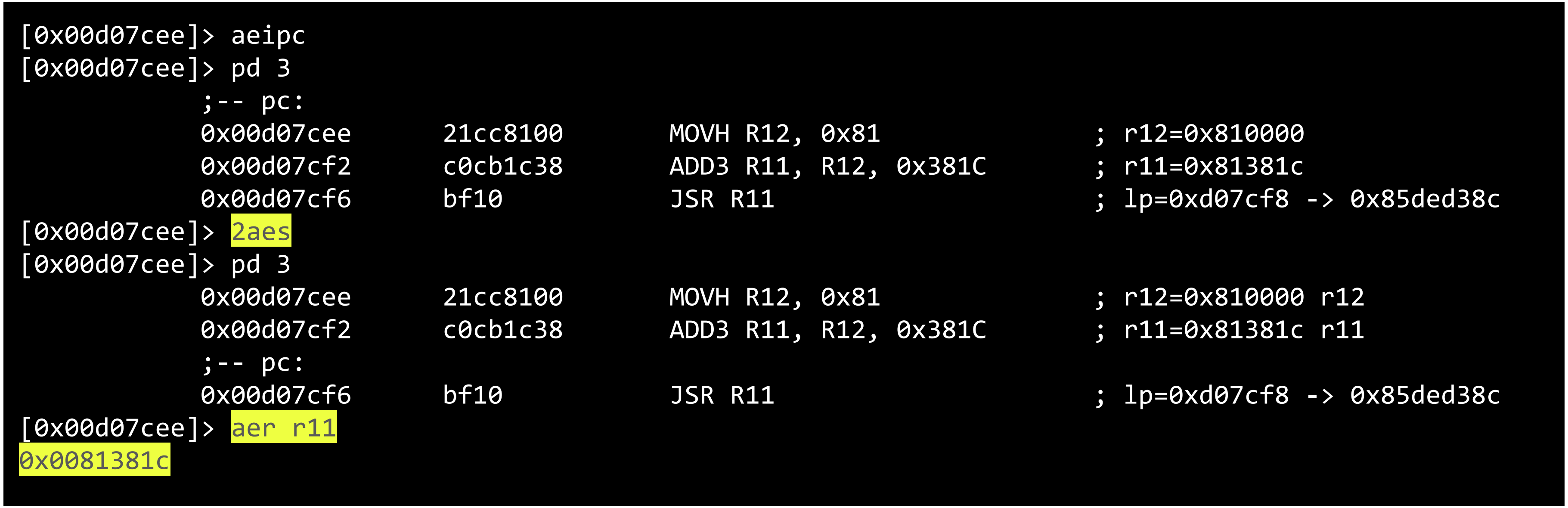
r2m2_Ae.so - emulation



```
[0x00000000]> e asm.emu=true
[0x00000000]> aei
[0x00000000]> pd 2
        ,=< 0x00000000      08d80100       JMP 0x100                     ; pc=0x100 -> 0x59287000
       ,==< 0x00000004      18df0800       JMP 0x8E2                     ; pc=0x8e2 -> 0x8df00
[0x00000000]> aes
[0x00000100]> pd 2
            ;-- pc:
            0x00000100      0070           DI                            ; psw=0x0
            0x00000102      2859           MOV R9, 40                    ; r9=0x28
[0x00000100]>
```

JMP emulation with ESIL

r2m2_Ae.so - emulation

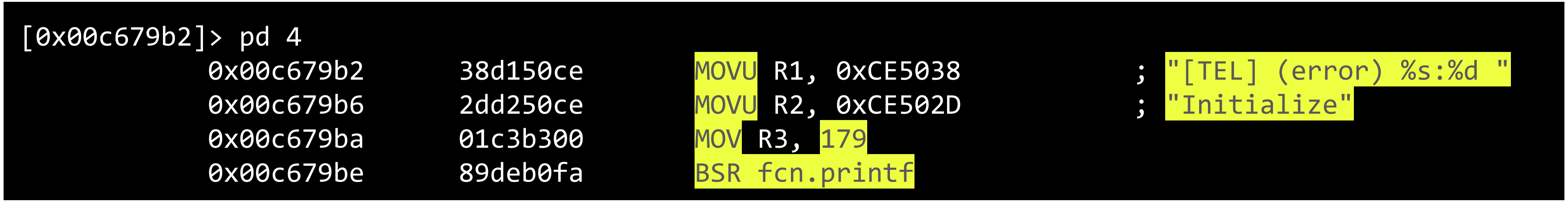


```
[0x00d07cee]> aeipc
[0x00d07cee]> pd 3
            ;-- pc:
            0x00d07cee      21cc8100        MOVH R12, 0x81            ; r12=0x810000
            0x00d07cf2      c0cb1c38        ADD3 R11, R12, 0x381C     ; r11=0x81381c
            0x00d07cf6      bf10            JSR R11                   ; lp=0xd07cf8 -> 0x85ded38c
[0x00d07cee]> 2aes
[0x00d07cee]> pd 3
            0x00d07cee      21cc8100        MOVH R12, 0x81            ; r12=0x810000 r12
            0x00d07cf2      c0cb1c38        ADD3 R11, R12, 0x381C     ; r11=0x81381c r11
            ;-- pc:
            0x00d07cf6      bf10            JSR R11                   ; lp=0xd07cf8 -> 0x85ded38c
[0x00d07cee]> aer r11
0x0081381c
```

retrieving a register value with ESIL

Reversing With Strings

Auto-naming Functions



```
[0x00c679b2]> pd 4
            0x00c679b2      38d150ce        MOVU R1, 0xCE5038           ; "[TEL] (error) %s:%d "
            0x00c679b6      2dd250ce        MOVU R2, 0xCE502D           ; "Initialize"
            0x00c679ba      01c3b300        MOV R3, 179
            0x00c679be      89deb0fa        BSR fcn.printf
```

typical error message pattern

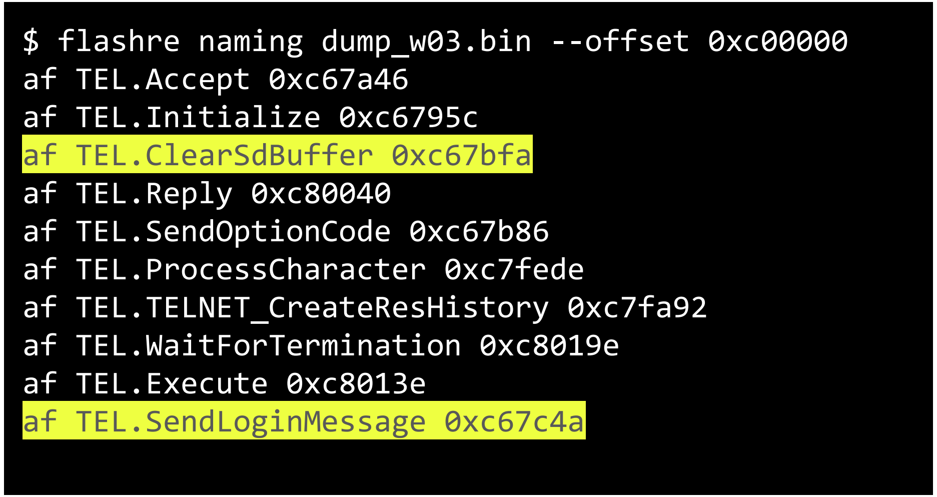
strategy with r2pipe

1. assemble `MOVU R1,<error format string address>`
2. search corresponding bytes
3. disassemble and check the `MOVU, MOVU, MOV, BSR` pattern
4. find the closest function prologue

~150

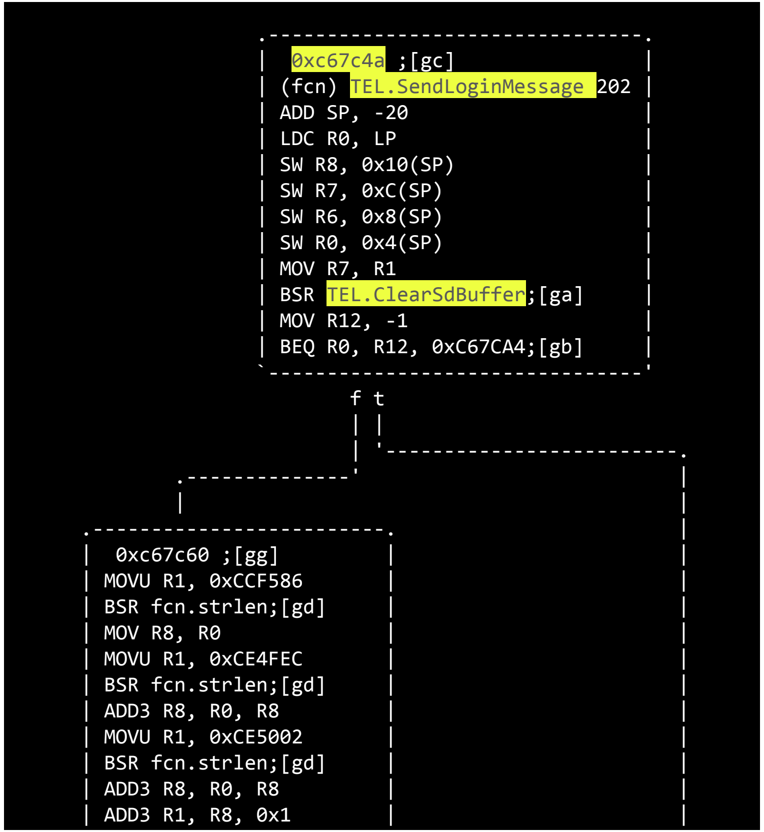
functions automatically named

Telnet Related Functions



```
$ flashre naming dump_w03.bin --offset 0xc00000
af TEL.Accept 0xc67a46
af TEL.Initialize 0xc6795c
af TEL.ClearSdBuffer 0xc67bfa
af TEL.Reply 0xc80040
af TEL.SendOptionCode 0xc67b86
af TEL.ProcessCharacter 0xc7fede
af TEL.TELNET_CreateResHistory 0xc7fa92
af TEL.WaitForTermination 0xc8019e
af TEL.Execute 0xc8013e
af TEL.SendLoginMessage 0xc67c4a
```

auto-named telnet functions



```
               .--------------------------------.
               |  0xc67c4a ;[gc]                |
               | (fcn) TEL.SendLoginMessage 202 |
               | ADD SP, -20                    |
               | LDC R0, LP                     |
               | SW R8, 0x10(SP)                |
               | SW R7, 0xC(SP)                 |
               | SW R6, 0x8(SP)                 |
               | SW R0, 0x4(SP)                 |
               | MOV R7, R1                     |
               | BSR TEL.ClearSdBuffer;[ga]     |
               | MOV R12, -1                    |
               | BEQ R0, R12, 0xC67CA4;[gb]     |
               `--------------------------------'
                       f t
                       | |
                       | '-------------------------.
        .--------------'                           |
        |                                          |
.-------------------------.                        |
|  0xc67c60 ;[gg]         |                        |
| MOVU R1, 0xCCF586       |                        |
| BSR fcn.strlen;[gd]     |                        |
| MOV R8, R0              |                        |
| MOVU R1, 0xCE4FEC       |                        |
| BSR fcn.strlen;[gd]     |                        |
| ADD3 R8, R0, R8         |                        |
| MOVU R1, 0xCE5002       |                        |
| BSR fcn.strlen;[gd]     |                        |
| ADD3 R8, R0, R8         |                        |
| ADD3 R1, R8, 0x1        |                        |
```

TEL.SendLoginMessage()

Two RE targets

1. update mechanism

   discover the binary format

2. configuration parser

   parameters effects

   understand commands

Update Mechanism

Update Header

32 bytes long

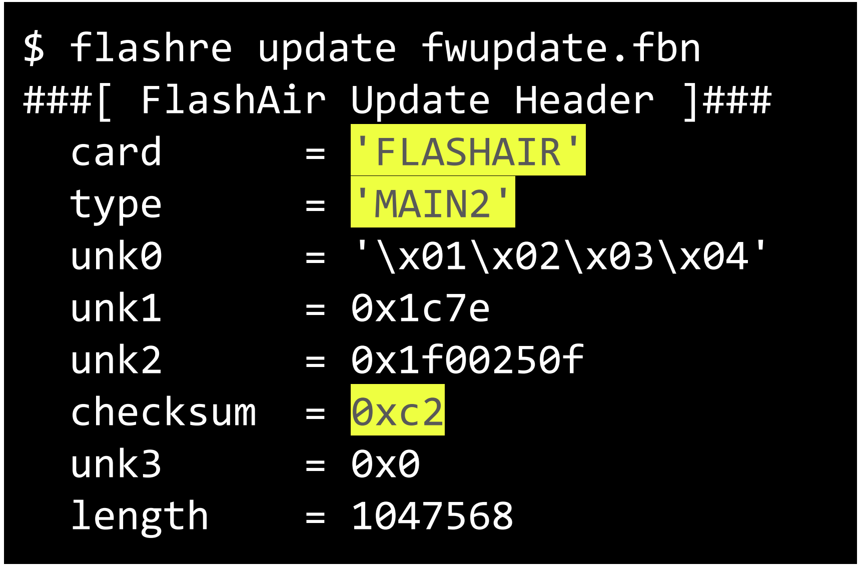
starts with “FLASHAIR”

defines five different types

MAIN2, BOOT, MAC, RF, USRPRG

one-byte checksum

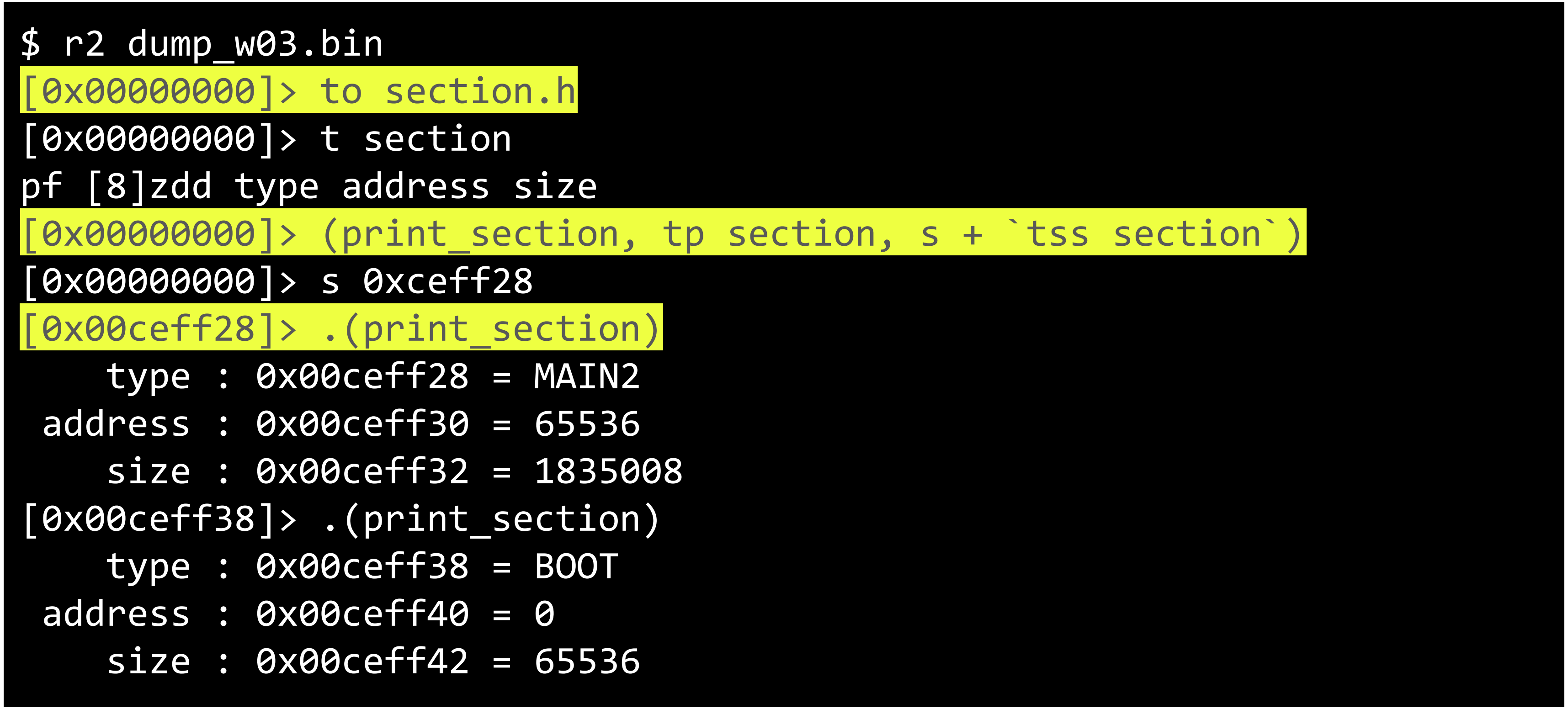
sum of all data bytes modulo 255



```
$ flashre update fwupdate.fbn
###[ FlashAir Update Header ]###
  card      = 'FLASHAIR'
  type      = 'MAIN2'
  unk0      = '\x01\x02\x03\x04'
  unk1      = 0x1c7e
  unk2      = 0x1f00250f
  checksum  = 0xc2
  unk3      = 0x0
  length    = 1047568
```

SPI Memory Map Array at 0xceff28

Type	Content	Address	Size
BOOT	**MeP code**	**0x000000**	**64 KB**
MAIN2	**MeP code**	**0x010000**	**1.8 MB**
MAC	MAC address ...	0x1d0000	24 KB
RF	starts with “2230”	0x1d8000	32 KB
USRPRG	full of 0xFF bytes	0x1e0000	128 KB



```
$ r2 dump_w03.bin
[0x00000000]> to section.h
[0x00000000]> t section
pf [8]zdd type address size
[0x00000000]> (print_section, tp section, s + `tss section`)
[0x00000000]> s 0xceff28
[0x00ceff28]> .(print_section)
    type : 0x00ceff28 = MAIN2
 address : 0x00ceff30 = 65536
    size : 0x00ceff32 = 1835008
[0x00ceff38]> .(print_section)
    type : 0x00ceff38 = BOOT
 address : 0x00ceff40 = 0
    size : 0x00ceff42 = 65536
```

dumping section structures

Reversing the Configuration Parser

Three Uses

configure values

APPSSID, APPNETWORKEY ...

start daemons

TELNET, DHCP_Enabled ...

execute commands

COMMAND

parse_config() - 0xc15f4e

```
0x00c1633e   41d1d1c9   MOVU R1, 0xC9D141   ; “APPSSID”
0x00c16342   6002       MOV R2, R6          ; parameter
0x00c16344   a9d86a06   BSR fcn.strcmp
[..]
```

testing a parameter name

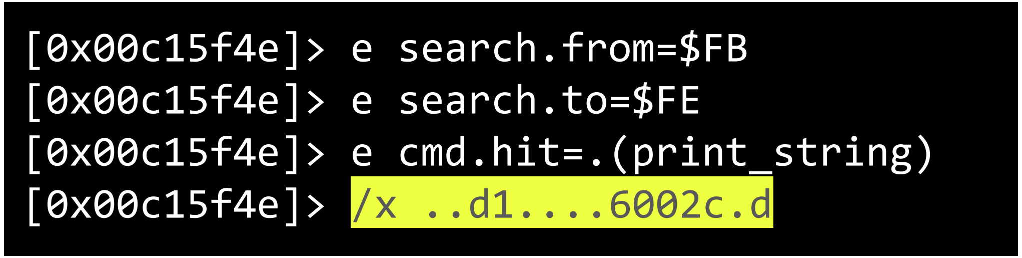
```
[0x00c1633e]> (print_string, ps @ `pd 1~[4]`)
[0x00c1633e]> .(print_string)
APPSSID
```

extracting the parameter name with a macro

Listing Undocumented Parameters

1. search the MOVU, MOV, BSR pattern

2. print the string

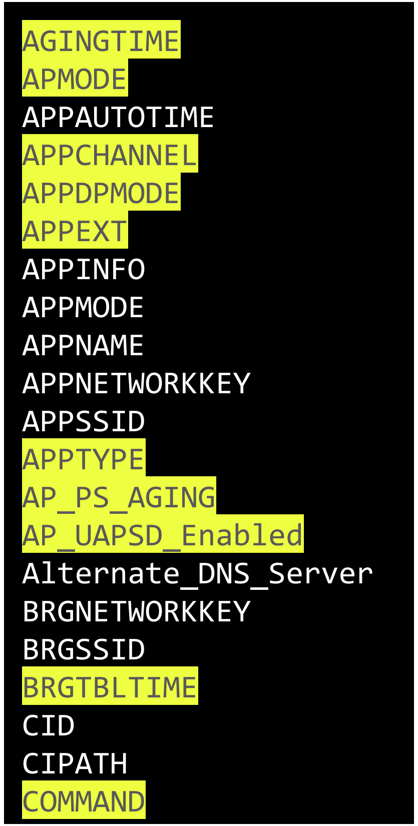


```
[0x00c15f4e]> e search.from=$FB
[0x00c15f4e]> e search.to=$FE
[0x00c15f4e]> e cmd.hit=.(print_string)
[0x00c15f4e]> /x ..d1....6002c.d
```

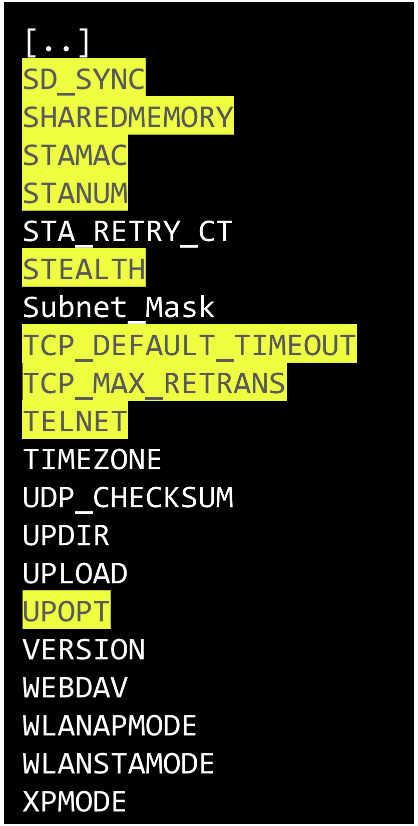
call command on hit

~30 documented

~70 extracted

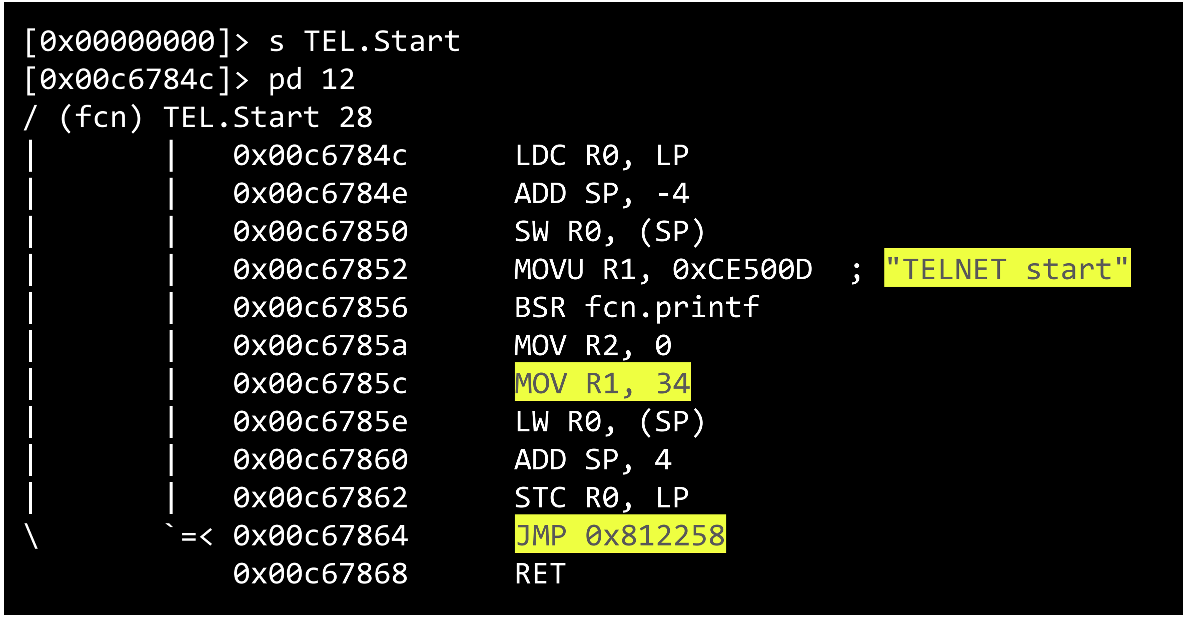


```
AGINGTIME
APMODE
APPAUTOTIME
APPCHANNEL
APPDPMODE
APPEXT
APPINFO
APPMODE
APPNAME
APPNETWORKKEY
APPSSID
APPTYPE
AP_PS_AGING
AP_UAPSD_Enabled
Alternate_DNS_Server
BRGNETWORKKEY
BRGSSID
BRGTBLTIME
CID
CIPATH
COMMAND
```



```
[..]
SD_SYNC
SHAREDMEMORY
STAMAC
STANUM
STA_RETRY_CT
STEALTH
Subnet_Mask
TCP_DEFAULT_TIMEOUT
TCP_MAX_RETRANS
TELNET
TIMEZONE
UDP_CHECKSUM
UPDIR
UPLOAD
UPOPT
VERSION
WEBDAV
WLANAPMODE
WLANSTAMODE
XPMODE
```

Starting the Telnet Daemon



```
[0x00000000]> s TEL.Start
[0x00c6784c]> pd 12
/ (fcn) TEL.Start 28
|       |   0x00c6784c      LDC R0, LP
|       |   0x00c6784e      ADD SP, -4
|       |   0x00c67850      SW R0, (SP)
|       |   0x00c67852      MOVU R1, 0xCE500D   ; "TELNET start"
|       |   0x00c67856      BSR fcn.printf
|       |   0x00c6785a      MOV R2, 0
|       |   0x00c6785c      MOV R1, 34
|       |   0x00c6785e      LW R0, (SP)
|       |   0x00c67860      ADD SP, 4
|       |   0x00c67862      STC R0, LP
\       `=< 0x00c67864      JMP 0x812258
            0x00c67868      RET
```

jumps to 0x812258

first argument is 34

execute_command() - 0xc29cce

- two functions access an array at 0xc9ff18
  - is_valid() at 0xc29462
  - is_authorized() at 0xc29078

- command_t structures array
  - 47 elements
  - function address and name

```
typedef struct command {
  char* name;
  void* function;
  char* default_argument;
  char* long_name;
  char* help;
  int level;
} command_t;
```

Listing All Available Commands

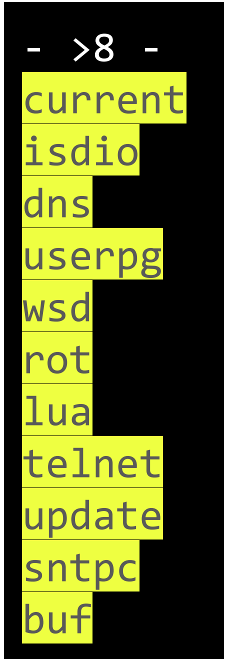
```
[0x00000000]> pv @@= `?s 0xc9ff18 0xc9ff18+24*47 24` > offsets.txt
```

extracting command_t offsets

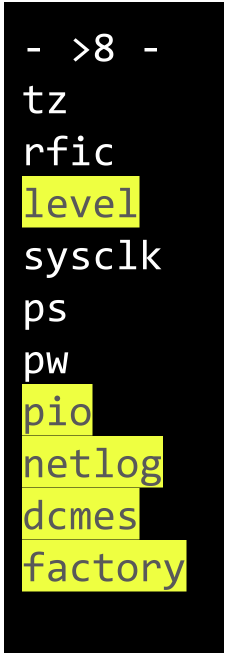
```
[0x00000000]> ps @@= `cat offsets.txt`
```

printing commands

15 new commands



```
- >8 -
current
isdio
dns
userpg
wsd
rot
lua
telnet
update
sntpc
buf
```

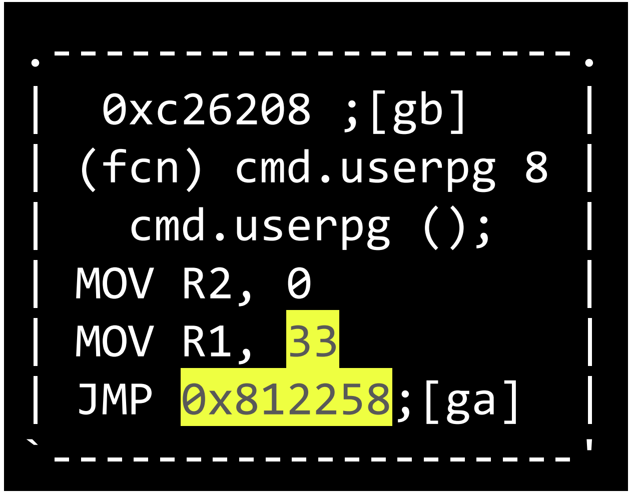


```
- >8 -
tz
rfic
level
sysclk
ps
pw
pio
netlog
dcmes
factory
```

The userpg command

jumps to 0x812258

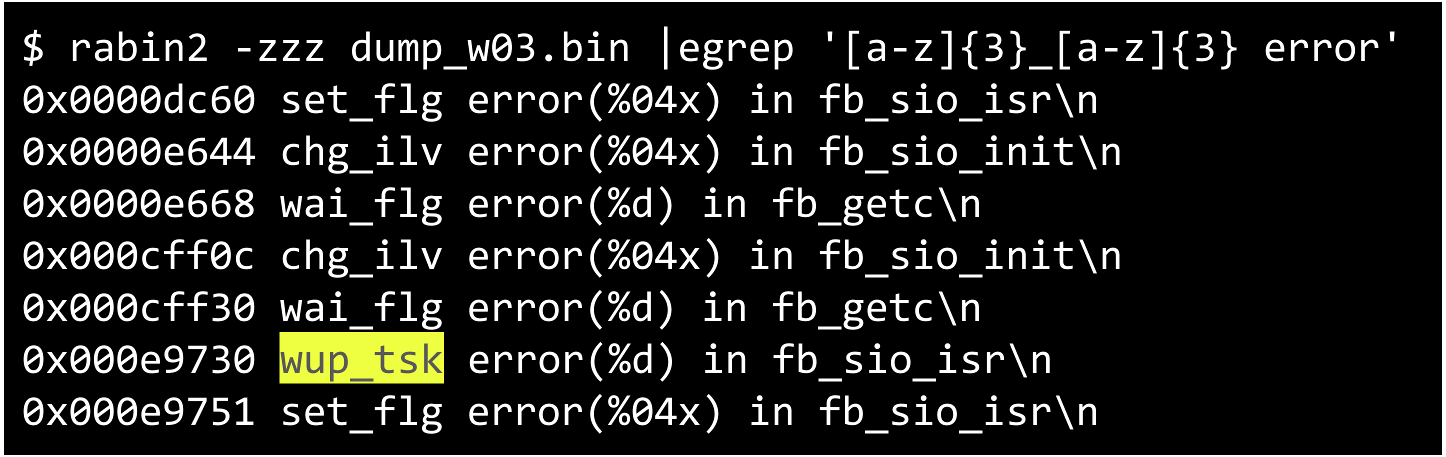
- also called in parse_config()
- first argument was 34



```
.--------------------.
|  0xc26208 ;[gb]    |
| (fcn) cmd.userpg 8 |
|   cmd.userpg ();   |
| MOV R2, 0          |
| MOV R1, 33         |
| JMP 0x812258;[ga]  |
`--------------------'
```

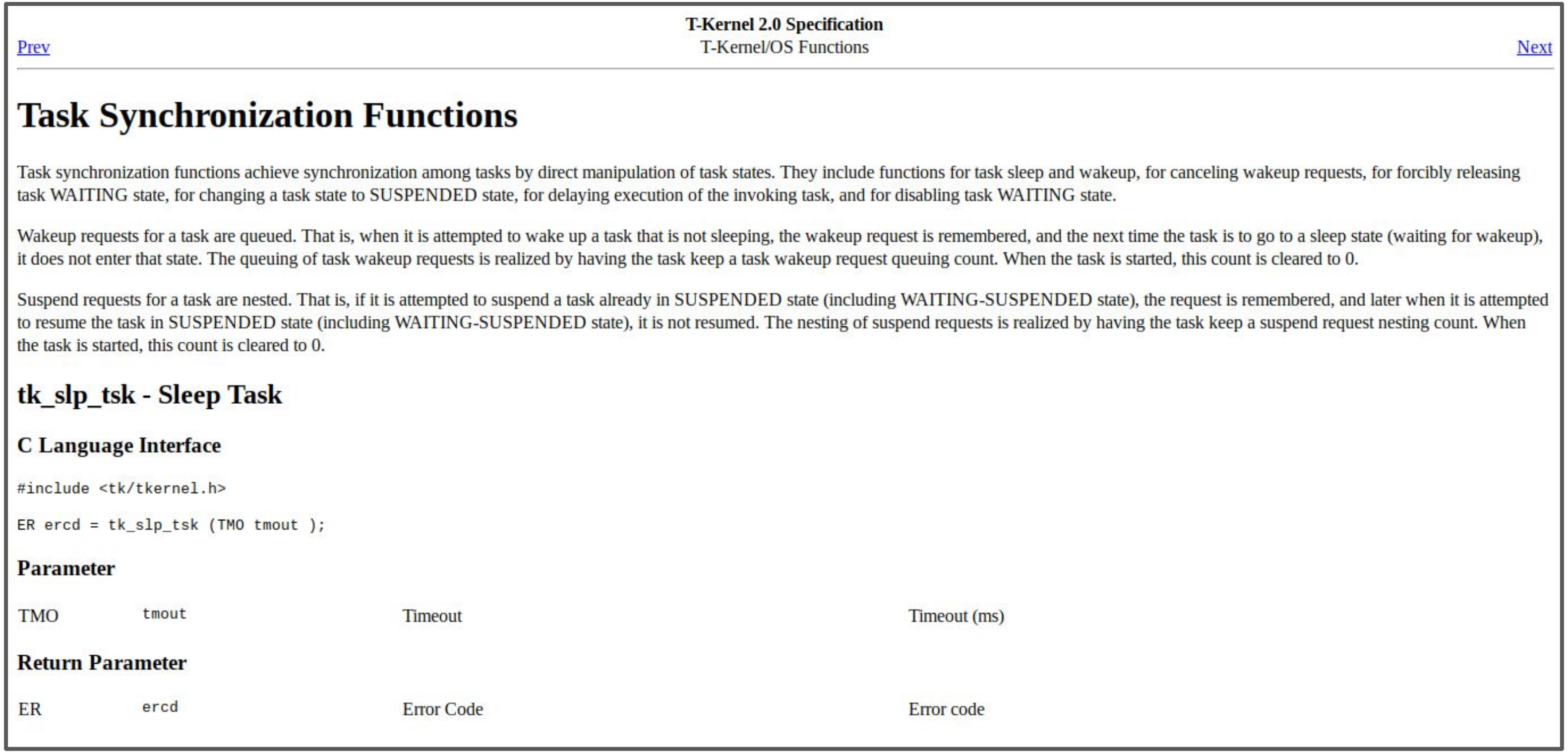
Identifying the OS

More Error Strings!



```
$ rabin2 -zzz dump_w03.bin |egrep '[a-z]{3}_[a-z]{3} error'
0x0000dc60 set_flg error(%04x) in fb_sio_isr\n
0x0000e644 chg_ilv error(%04x) in fb_sio_init\n
0x0000e668 wai_flg error(%d) in fb_getc\n
0x000cff0c chg_ilv error(%04x) in fb_sio_init\n
0x000cff30 wai_flg error(%d) in fb_getc\n
0x000e9730 wup_tsk error(%d) in fb_sio_isr\n
0x000e9751 set_flg error(%04x) in fb_sio_isr\n
```

wup_tsk() looks promising!



T-Kernel 2.0 Specification

Prev T-Kernel/OS Functions Next

Task Synchronization Functions

Task synchronization functions achieve synchronization among tasks by direct manipulation of task states. They include functions for task sleep and wakeup, for canceling wakeup requests, for forcibly releasing task WAITING state, for changing a task state to SUSPENDED state, for delaying execution of the invoking task, and for disabling task WAITING state.

Wakeup requests for a task are queued. That is, when it is attempted to wake up a task that is not sleeping, the wakeup request is remembered, and the next time the task is to go to a sleep state (waiting for wakeup), it does not enter that state. The queuing of task wakeup requests is realized by having the task keep a task wakeup request queuing count. When the task is started, this count is cleared to 0.

Suspend requests for a task are nested. That is, if it is attempted to suspend a task already in SUSPENDED state (including WAITING-SUSPENDED state), the request is remembered, and later when it is attempted to resume the task in SUSPENDED state (including WAITING-SUSPENDED state), it is not resumed. The nesting of suspend requests is realized by having the task keep a suspend request nesting count. When the task is started, this count is cleared to 0.

tk_slp_tsk - Sleep Task

C Language Interface

```
#include <tk/tkernel.h>
```

```
ER ercd = tk_slp_tsk (TMO tmout );
```

Parameter

TMO	`tmout`	Timeout	Timeout (ms)

Return Parameter

ER	`ercd`	Error Code	Error code

wup_tsk - wake up a task in T-Kernel

http://www.tron.org/wp-content/themes/dp-magjam/pdf/t-kernel_2.0/html_en/task_dependent_synchronization_functions.html

The Real-time Operating system Nucleus

See https://www.tron.org

Japanese RTOS

launched in 1984

specifications maintained by the TRON Forum

typical version: MITRON (Micro Industrial Tron)

many implementations

T-Kernel, TOPPERS, RTEMS, UDEOS, PrKERNEL, DryOS, …

~150 supported architectures

Where is it Used?

Casio Exilim EX-FC100

Joy-Con

Canon 5D Mark III

Asteroid Explorer Hayabusa

The FlashAir W-03 uses UDEOS4.

Game Plan

- ☒ memory dump
- ☒ architecture
- ☒ Operating System
- □ execution vector

Solving the 0x812258() Mystery!

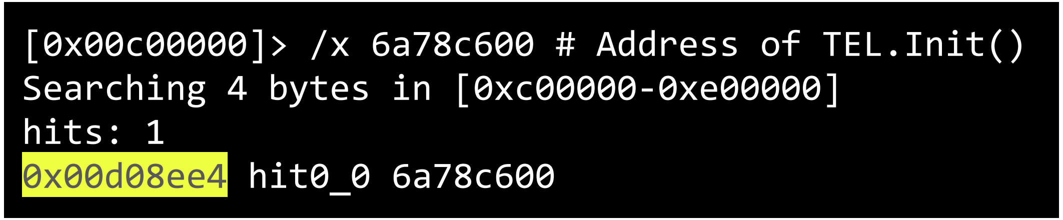
TEL.Init() - 0xc6786a

a single match in the dump

search result at 0xd08ee4

used in a potential tasks array

located at 0xd08c50



```
[0x00c00000]> /x 6a78c600 # Address of TEL.Init()
Searching 4 bytes in [0xc00000-0xe00000]
hits: 1
0x00d08ee4 hit0_0 6a78c600
```

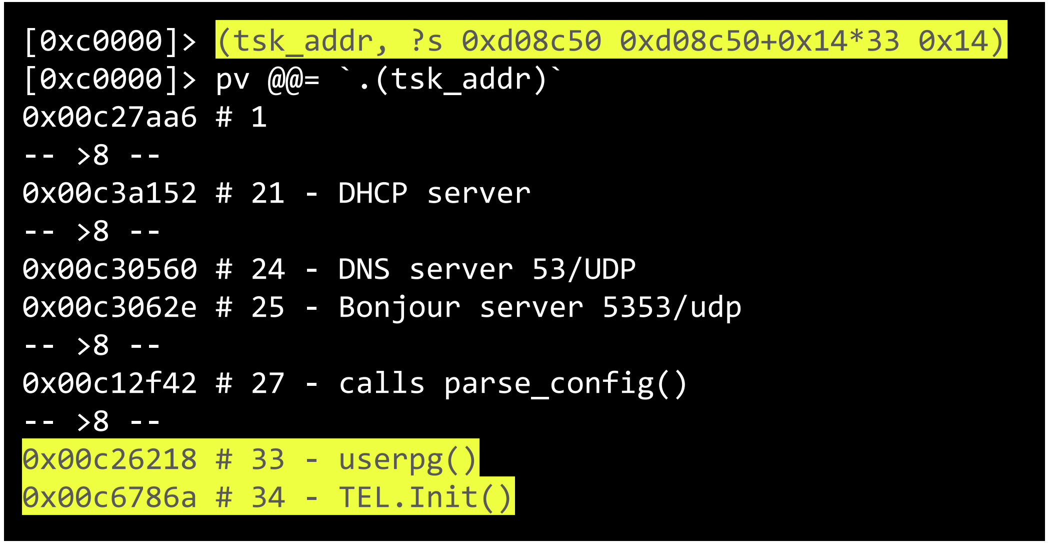
searching TEL.Init() address

34 tasks identified

elements of 20 bytes

0x812258() is sta_tsk()

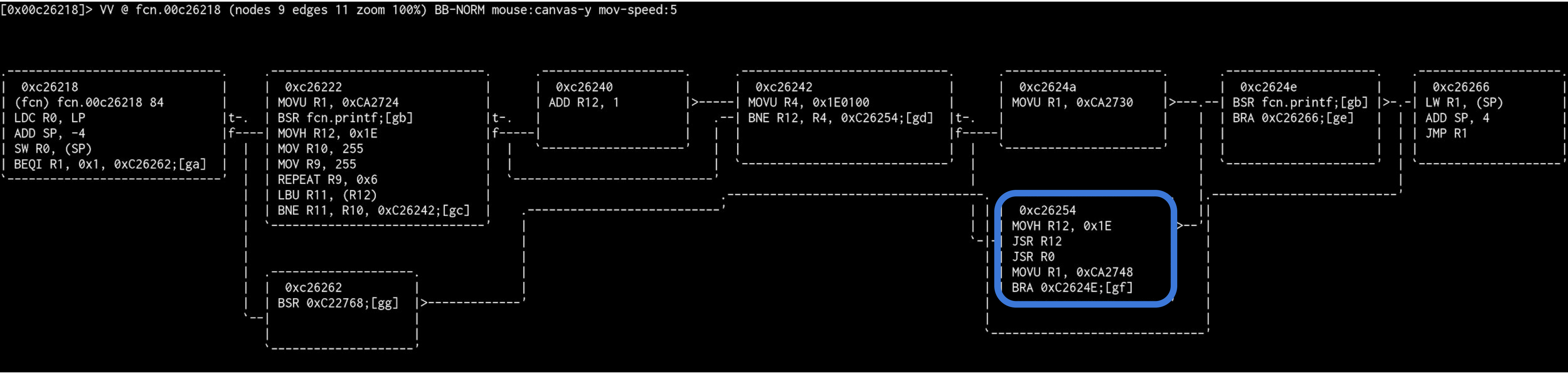
move task to READY state



```
[0xc0000]> (tsk_addr, ?s 0xd08c50 0xd08c50+0x14*33 0x14)
[0xc0000]> pv @@= `.(tsk_addr)`
0x00c27aa6 # 1
-- >8 --
0x00c3a152 # 21 - DHCP server
-- >8 --
0x00c30560 # 24 - DNS server 53/UDP
0x00c3062e # 25 - Bonjour server 5353/udp
-- >8 --
0x00c12f42 # 27 - calls parse_config()
-- >8 --
0x00c26218 # 33 - userpg()
0x00c6786a # 34 - TEL.Init()
```

tasks addresses

The userpg task - 0xc26218



```
[0x00c26218]> VV @ fcn.00c26218 (nodes 9 edges 11 zoom 100%) BB-NORM mouse:canvas-y mov-speed:5

 0xc26218
(fcn) fcn.00c26218 84
LDC R0, LP
ADD SP, -4
SW R0, (SP)
BEQI R1, 0x1, 0xC26262;[ga]

 0xc26222
MOVU R1, 0xCA2724
BSR fcn.printf;[gb]
MOVH R12, 0x1E
MOV R10, 255
MOV R9, 255
REPEAT R9, 0x6
LBU R11, (R12)
BNE R11, R10, 0xC26242;[gc]

 0xc26240
ADD R12, 1

 0xc26242
MOVU R4, 0x1E0100
BNE R12, R4, 0xC26254;[gd]

 0xc2624a
MOVU R1, 0xCA2730

 0xc2624e
BSR fcn.printf;[gb]
BRA 0xC26266;[ge]

 0xc26266
LW R1, (SP)
ADD SP, 4
JMP R1

 0xc26262
BSR 0xC22768;[gg]

 0xc26254
MOVH R12, 0x1E
JSR R12
JSR R0
MOVU R1, 0xCA2748
BRA 0xC2624E;[gf]
```

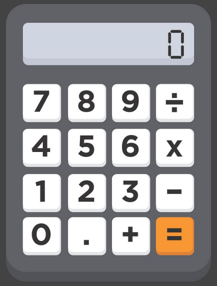
checks that the **USRPRG** section (0x1e0000) is not 0xff

jumps 0x1e0000

calls the function stored at R0

Game Plan

- ☒ memory dump
- ☒ architecture
- ☒ Operating System
- ☒ execution vector

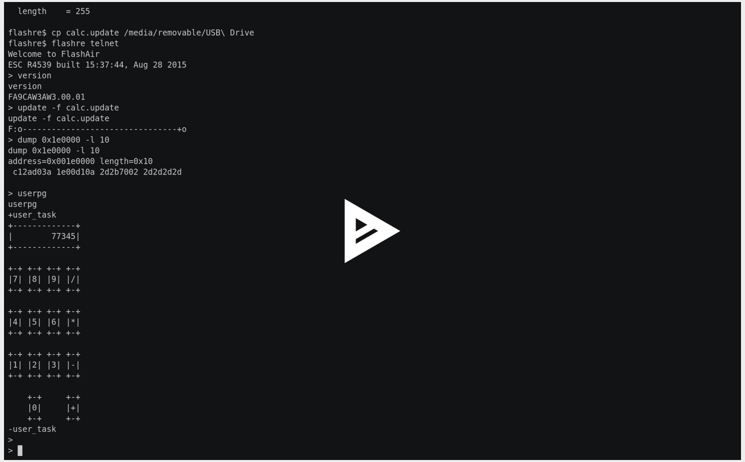


Thanks to JPCERT/CC, Toshiba is aware of these results since June.

Putting Everything Together

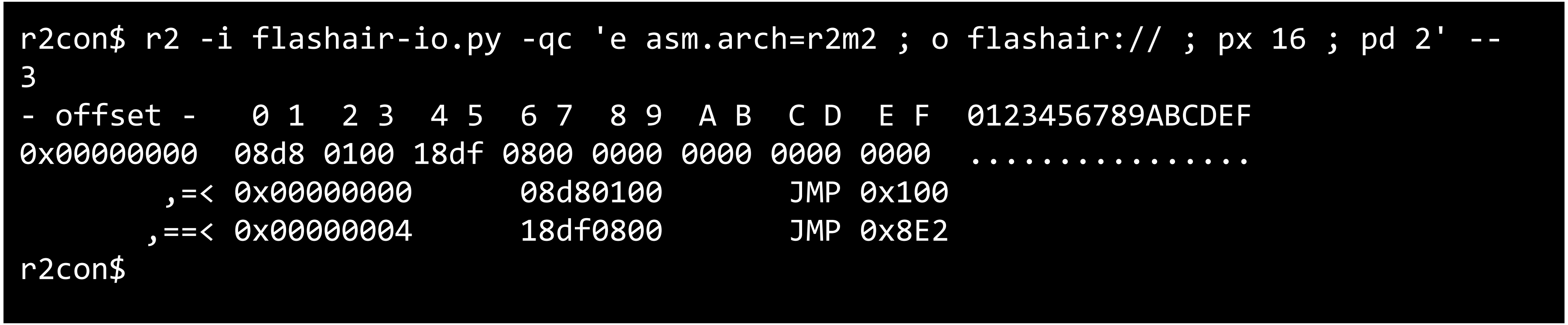
Created by Sarah
from Noun Project

1. build a fake **`USRPRG`** update
2. write it to the card
3. call **`update -f usrprg.bin`**
4. call **`userpg`**



```
  length    = 255

flashre$ cp calc.update /media/removable/USB\ Drive
flashre$ flashre telnet
Welcome to FlashAir
ESC R4539 built 15:37:44, Aug 28 2015
> version
version
FA9CAW3AW3.00.01
> update -f calc.update
update -f calc.update
F:o--------------------------------+o
> dump 0x1e0000 -l 10
dump 0x1e0000 -l 10
address=0x001e0000 length=0x10
 c12ad03a 1e00d10a 2d2b7002 2d2d2d2d

> userpg
userpg
+user_task
+-------------+
|        77345|
+-------------+

+-+ +-+ +-+ +-+
|7| |8| |9| |/|
+-+ +-+ +-+ +-+

+-+ +-+ +-+ +-+
|4| |5| |6| |*|
+-+ +-+ +-+ +-+

+-+ +-+ +-+ +-+
|1| |2| |3| |-|
+-+ +-+ +-+ +-+

    +-+     +-+
    |0|     |+|
    +-+     +-+
-user_task
>
> 
```

Tools Wish List



```
r2con$ r2 -i flashair-io.py -qc 'e asm.arch=r2m2 ; o flashair:// ; px 16 ; pd 2' --
3
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00000000  08d8 0100 18df 0800 0000 0000 0000 0000  ................
        ,=< 0x00000000      08d80100       JMP 0x100
       ,==< 0x00000004      18df0800       JMP 0x8E2
r2con$
```

binutils based MeP disassembly

in Rust using guedou/binutils-rs

discover functions in .01 and .02 firmwares

binary diffing with diaphora

Last Words

unexpected

a Japanese SoC and a Japanese OS

original

detailed FlashAir analysis and code execution

reproducible

open-source tools & addresses published

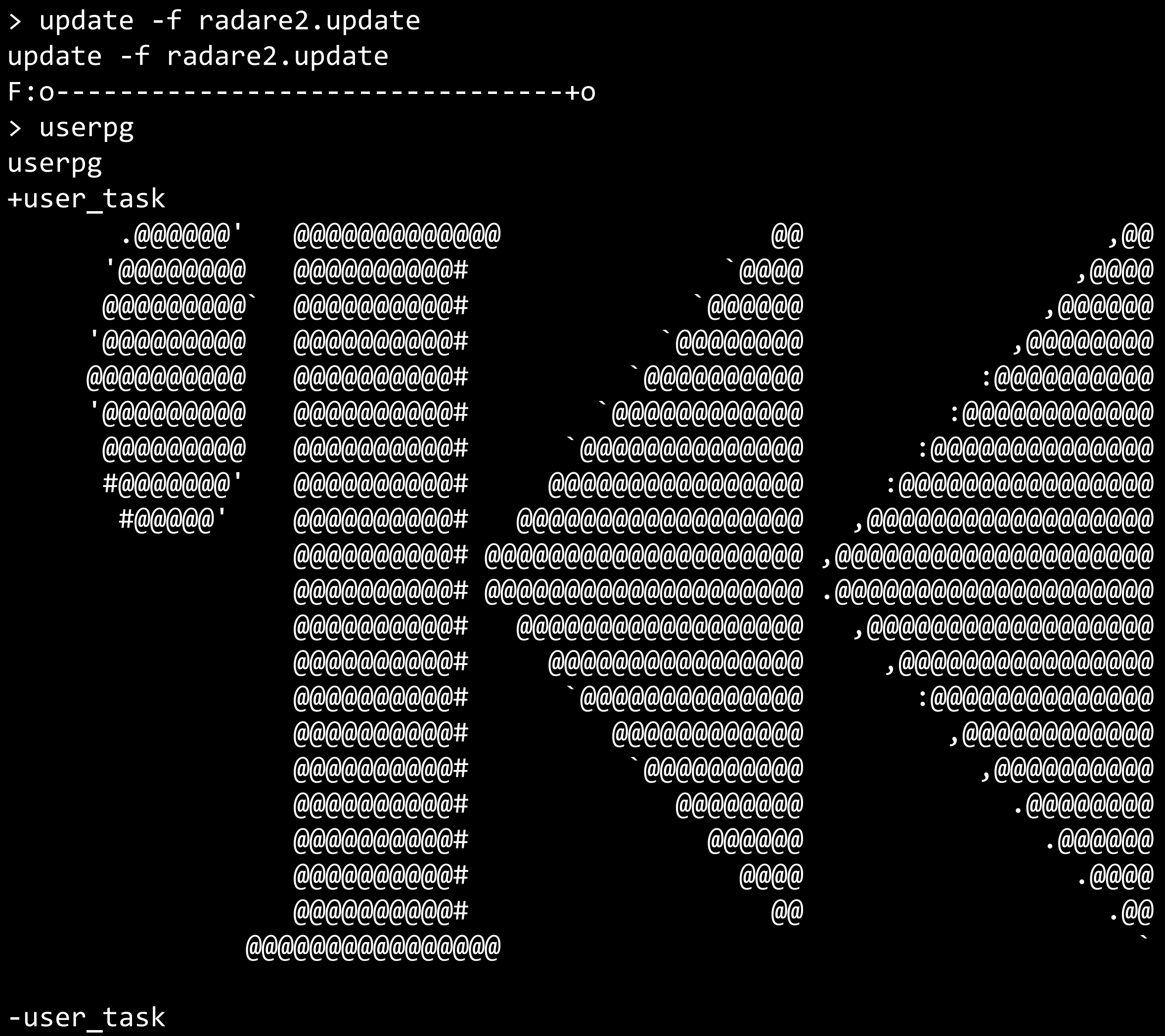
Tools!

guedou/flashre

- guedou/r2m2
- radare/radare2
- cea-sec/miasm
- cea-sec/sibyl
- sgayou/rbasefind

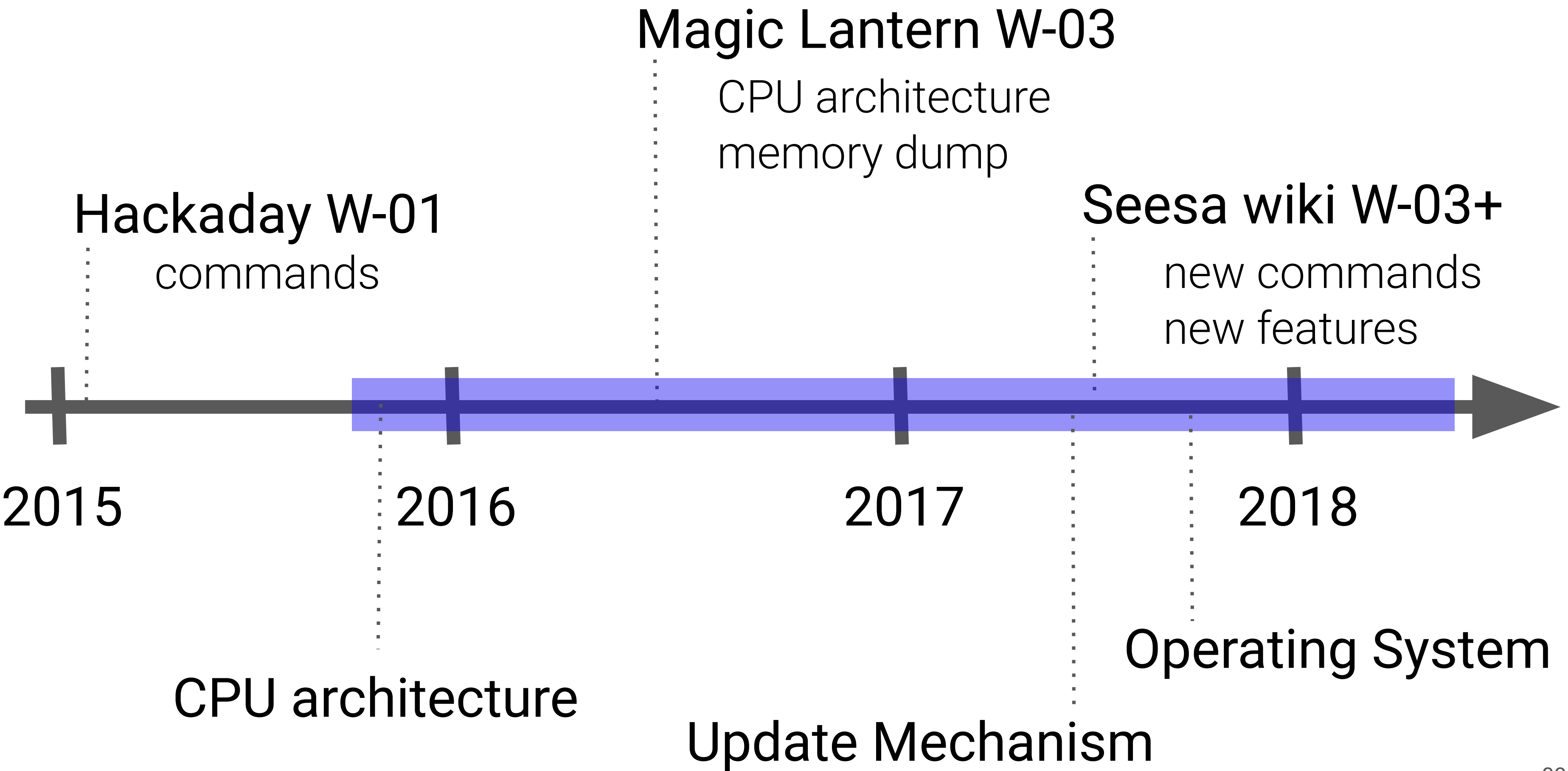
guedou/jupyter-radare2
guedou/r2scapy

guedou/binutils-rs



```
> update -f radare2.update
update -f radare2.update
F:o--------------------------------+o
> userpg
userpg
+user_task
       .@@@@@@'   @@@@@@@@@@@@@                 @@                   ,@@
      '@@@@@@@@   @@@@@@@@@@#                `@@@@                 ,@@@@
      @@@@@@@@@`  @@@@@@@@@@#              `@@@@@@               ,@@@@@@
     '@@@@@@@@@   @@@@@@@@@@#            `@@@@@@@@             ,@@@@@@@@
     @@@@@@@@@@   @@@@@@@@@@#          `@@@@@@@@@@           :@@@@@@@@@@
     '@@@@@@@@@   @@@@@@@@@@#        `@@@@@@@@@@@@         :@@@@@@@@@@@@
      @@@@@@@@@   @@@@@@@@@@#      `@@@@@@@@@@@@@@       :@@@@@@@@@@@@@@
      #@@@@@@@'   @@@@@@@@@@#     @@@@@@@@@@@@@@@@     :@@@@@@@@@@@@@@@@
       #@@@@@'    @@@@@@@@@@#   @@@@@@@@@@@@@@@@@@   ,@@@@@@@@@@@@@@@@@@
                  @@@@@@@@@@# @@@@@@@@@@@@@@@@@@@@ ,@@@@@@@@@@@@@@@@@@@@
                  @@@@@@@@@@# @@@@@@@@@@@@@@@@@@@@ .@@@@@@@@@@@@@@@@@@@@
                  @@@@@@@@@@#   @@@@@@@@@@@@@@@@@@   ,@@@@@@@@@@@@@@@@@@
                  @@@@@@@@@@#     @@@@@@@@@@@@@@@@     ,@@@@@@@@@@@@@@@@
                  @@@@@@@@@@#      `@@@@@@@@@@@@@@       :@@@@@@@@@@@@@@
                  @@@@@@@@@@#         @@@@@@@@@@@@         ,@@@@@@@@@@@@
                  @@@@@@@@@@#          `@@@@@@@@@@           ,@@@@@@@@@@
                  @@@@@@@@@@#             @@@@@@@@             .@@@@@@@@
                  @@@@@@@@@@#               @@@@@@               .@@@@@@
                  @@@@@@@@@@#                 @@@@                 .@@@@
                  @@@@@@@@@@#                   @@                   .@@
               @@@@@@@@@@@@@@@@                                        `

-user_task
```

Few More Things



Magic Lantern W-03

CPU architecture
memory dump

Hackaday W-01

commands

Seesa wiki W-03+

new commands
new features

2015

2016

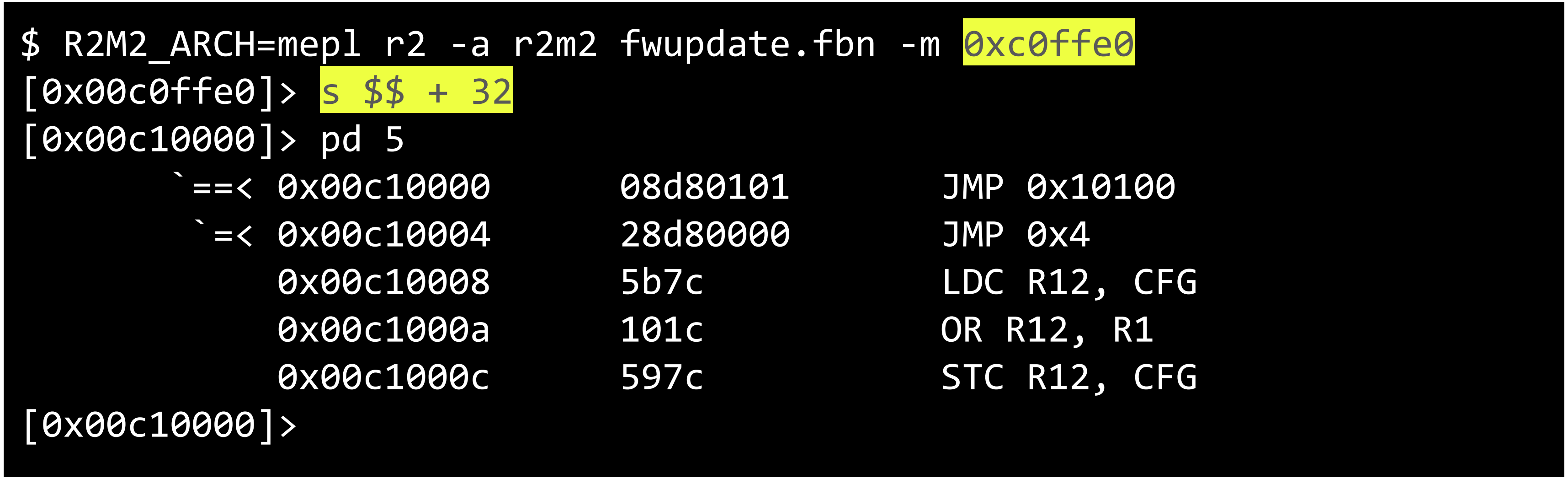
2017

2018

CPU architecture

Operating System

Update Mechanism



```
$ R2M2_ARCH=mepl r2 -a r2m2 fwupdate.fbn -m 0xc0ffe0
[0x00c0ffe0]> s $$ + 32
[0x00c10000]> pd 5
       `==< 0x00c10000      08d80101       JMP 0x10100
        `=< 0x00c10004      28d80000       JMP 0x4
            0x00c10008      5b7c           LDC R12, CFG
            0x00c1000a      101c           OR R12, R1
            0x00c1000c      597c           STC R12, CFG
[0x00c10000]>
```

mapping fwupdate.fbn correctly

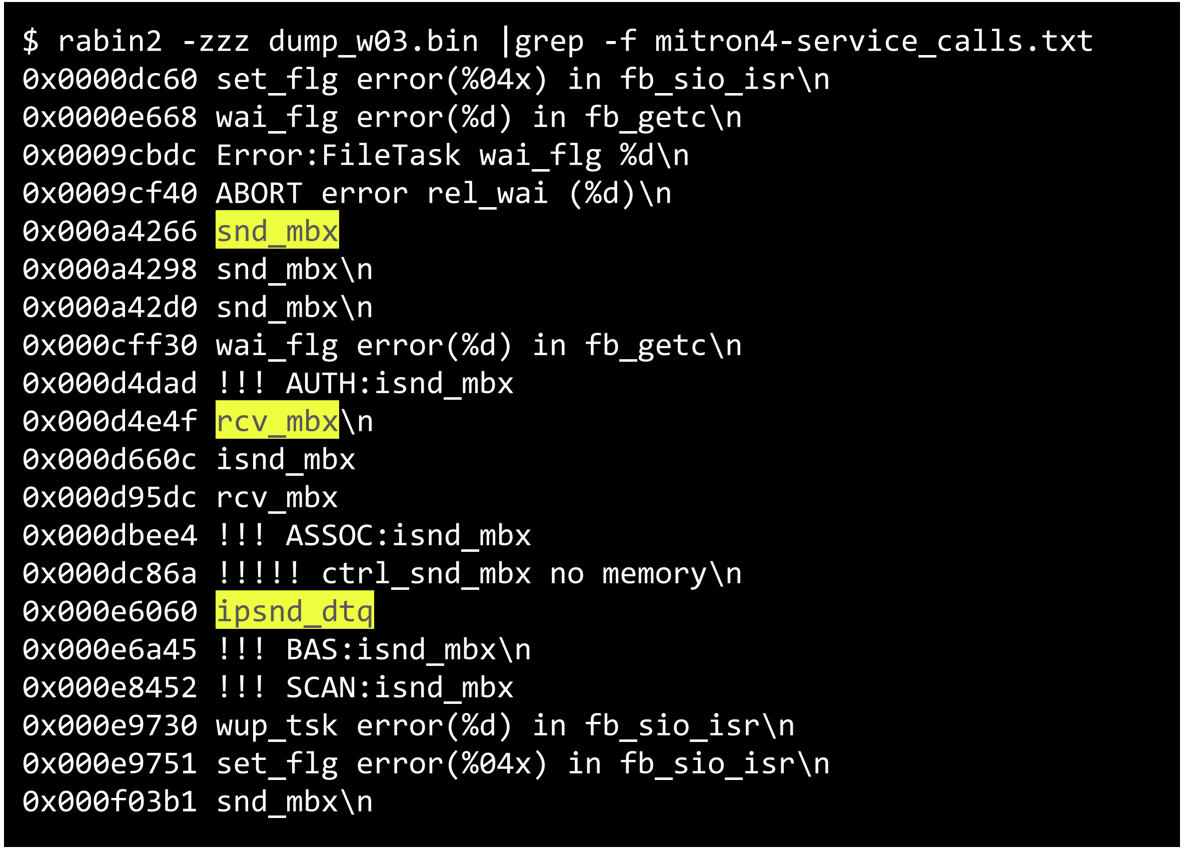
Extract Configuration Parameters from Memory

some stored at fixed offsets from 0x817ae8

APPSSID, APPNETWORKEY, CIPATH ...

```
[0x00c15f4e]> ps @ 0x817aE8 + 0x22b
flashair
[0x00c15f4e]> ps @ 0x817aE8 + 0x24c
2018%bhus&GV!
[0x00c15f4e]> ps @ 0x817aE8 + 0x12a
/DCIM/100__TSB/FA000001.JPG
```

retrieving parameter values



```
$ rabin2 -zzz dump_w03.bin |grep -f mitron4-service_calls.txt
0x0000dc60 set_flg error(%04x) in fb_sio_isr\n
0x0000e668 wai_flg error(%d) in fb_getc\n
0x0009cbdc Error:FileTask wai_flg %d\n
0x0009cf40 ABORT error rel_wai (%d)\n
0x000a4266 snd_mbx
0x000a4298 snd_mbx\n
0x000a42d0 snd_mbx\n
0x000cff30 wai_flg error(%d) in fb_getc\n
0x000d4dad !!! AUTH:isnd_mbx
0x000d4e4f rcv_mbx\n
0x000d660c isnd_mbx
0x000d95dc rcv_mbx
0x000dbee4 !!! ASSOC:isnd_mbx
0x000dc86a !!!!! ctrl_snd_mbx no memory\n
0x000e6060 ipsnd_dtq
0x000e6a45 !!! BAS:isnd_mbx\n
0x000e8452 !!! SCAN:isnd_mbx
0x000e9730 wup_tsk error(%d) in fb_sio_isr\n
0x000e9751 set_flg error(%04x) in fb_sio_isr\n
0x000f03b1 snd_mbx\n
```

identifying new mitron Service Calls

“ITRON is the most used OS in 2003”

Wikipedia

See https://en.wikipedia.org/wiki/TRON_project