

A Deeper Look Into Reversing the Toshiba FlashAir SD Card

@guedou

PacSec2018



@guedou?

hobbyist reverser

Scapy co-maintainer

Python-based packet manipulation program

network security researcher

IPv6, DNS, TLS, BGP, DDoS mitigation, ...

Head Of Security @ Netatmo

IoT Home Automation





2015



?



2018



Get the slides at
<https://goo.gl/RAzWSE>



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



FlashAir Extended Features

Lua script executed on the card
on boot, write events, or over HTTP

specific FlashAir API
interface with SPI, I2C, Wi-Fi ...



bitcoin rate display with I2C

Four Generations



2012



2013



2015

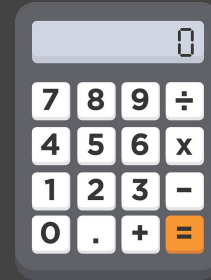
2017

¥2,800

Game Plan



- memory dump
- architecture
- Operating System
- execution vector



Inspecting Firmwares Updates

Firmwares Versions & Installers

v3.00.00

October 2014

v3.00.01

August 2015

v3.00.02

August 2016



INTERNET ARCHIVE
Wayback Machine

https://www.toshiba.co.jp/p-media/english/download/wl/updates002_w03.htm

20 captures
21 Feb 2016 - 11 Dec 2017

JAN FEB 21 MAR
2015 2016 2017

FlashAir™ W-03 Class10

FlashAir™ W-03
Wireless LAN
32 GB
TOSHIBA

FlashAir™ W-03
Wireless LAN
16 GB
TOSHIBA

FlashAir™ W-03
Wireless LAN
8 GB
TOSHIBA

For Windows®

Software Update ▶
(EXE:7.92MB)

User's manual ▶
(PDF:397KB/13pages)

For Mac

Software Update ▶
(ZIP:3.15MB)

User's manual ▶
(PDF:431KB/12pages)

Update History

Date	Version	Description
August 20, 2015	V3.00.01	<ul style="list-style-type: none">• A malfunction in which FlashAir™ drive is not available with Windows Vista® is modified.• A malfunction in which some thumbnails of photo data stored in FlashAir™ might not be displayed is modified.

This talk focuses
on 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

“/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  
[ND] 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 --
```

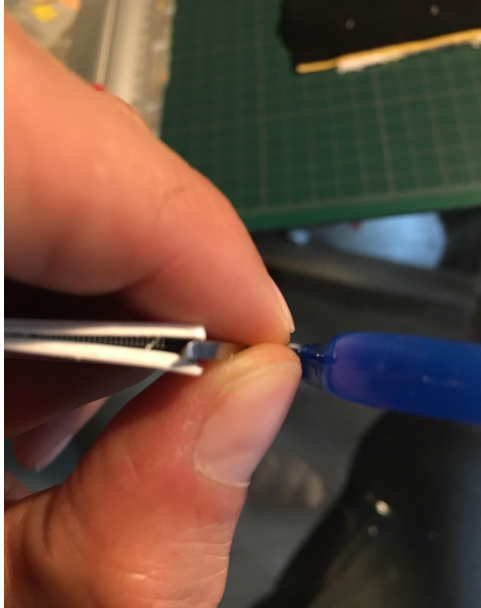
32

commands

Inspecting the Card

Getting Inside

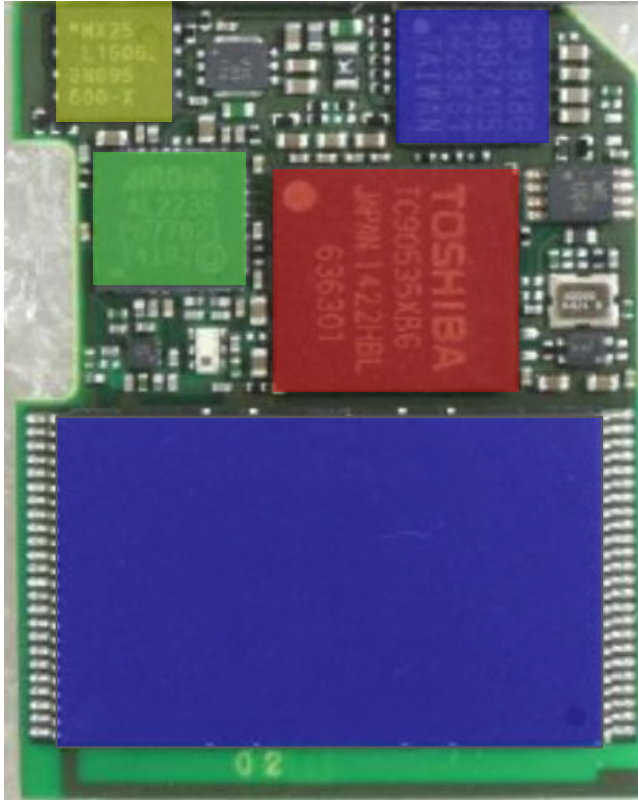
1. opening the card
using a sharp blade



2. searching FCC applications
FlashAir FCC ID: ZVZP42350FA3



FlashAir W-03 Innards

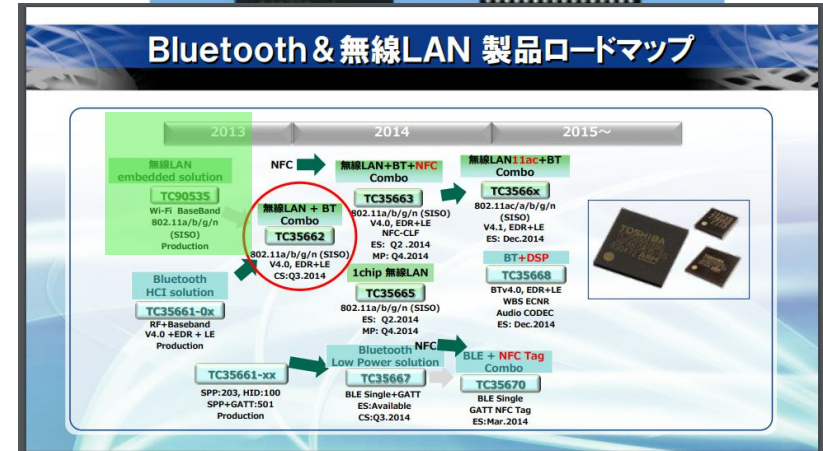
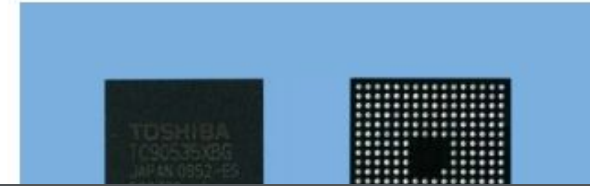


- Toshiba TC58TFG7DDLTAID: Flash memory
- Toshiba 6PJ8XBG: Flash Memory controller
- SPI - USON-8 4x4 mm - 2MB
 - Macronix - MX25L1606E
 - Winbond - Q16DVUZIG
- Airoha AL2238: 802.11 b/g - RF transceiver
- Toshiba TC90535XBG: ?

Toshiba TC90535XBG

the SoC

802.11n MAC
32-bit RISC
released in 2013



of 72 Mbits/s and one of the lowest levels of power consumption for such devices in the world.

To achieve this low power consumption, the circuit blocks are divided into multiple power domains to dynamically control the power supply voltage. Technologies for clock frequency control and original low-power flip-flops are also employed. In addition, the embedded 32-bit reduced instruction set computer (RISC) processor core is capable of handling tasks up to the level of communication middleware and applications in order to reduce the load of the host processor.

Dumping Memory

Software Based Dump

CONFIG & TELNET commands

fdump - write memory to files

dump - print memory content

```
dump 0x0 -1 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
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

dump, naming ...

Docker image available

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

Dumping Memory with flashre

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

dump

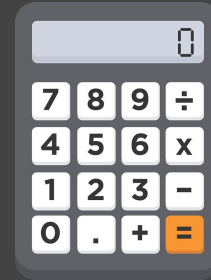
```
$ flashre dump --convert dump_w03.txt > dump_w03.bin  
$ ls -alh dump_w03.bin  
-rw-rw-r--. 1 guedou guedou 2.0M Aug 08 13:30 dump_w03.bin
```

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

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

User's Manual

MeP Core (MeP-c4)
User's Manual
(Architecture)

MeP manual in English

Toshiba Media-embedded Processor

MIPS like

load/store, ...

calling convention

first four registers then stack

16 general-purpose registers

33 control/special registers

32 bits addresses

up to 4GB

~200 instructions

2 or 4 bytes each

Little-Endian or Big-Endian

LEND field in the CFG register

no privileged mode

Memory Map

flash likely located at 0x000000

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

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
```

Disassembling Using the Main Base Address

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

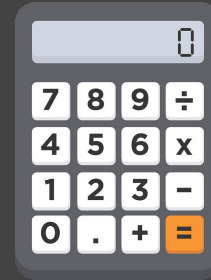
$ mep-objdump -m mep -b binary -D dump_w03.bin -adjust-vma=0xC00000
-- >8 --
  cfd27a:  69 d9 26 00 bsr 0xcff8a6
```

correct BSR address

Game Plan



- memory dump
- architecture
- operating System
- execution vector



~6500

BSR-based functions

MeP Tools

Wish List

disassembly with semantics

split basic blocks

instructions emulation

validate functions behavior

graphical interface

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

miasm2 - Adding the MeP MOV Instruction

MOV Rn,Rm

`0000_nnnn_mmmm_0000` (Rn=nnnn, Rm=mmm)

MeP manual

```
reg04 = bs(l=4, cls=(mep_reg,))
```

```
addop("MOV", [bs("0000"), reg04, reg04, bs("0000")])
```

arch/mep/arch.py

```
@sbuild.parse
```

```
def mov(regn, regm):
```

```
    regn = regm
```

arch/mep/sem.py

Sibyl

discover functions using jitters

emulate functions and verify their side effects

an API bruteforcer

```
$ sibyl find -j gcc -a mep1 -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
```

9

automatically discovered functions


radare2

RE framework
console based
set of command line utilities
extendable with plugins

```
$ r2 /bin/ls
[0x00005060]> pd 10
      |-- entry0:
      |-- rip:
0x00005060      31ed          xor ebp, ebp
0x00005062      4989d1        mov r9, rdx
0x00005065      5e           pop rsi
0x00005066      4889e2        mov rdx, rsp
0x00005069      4883e4f0     and rsp, 0xfffffffffffffff0
0x0000506d      50           push rax
0x0000506e      54           push rsp
0x0000506f      4c8d058a0c01. lea r8, [0x00015d00]
0x00005076      488d0d130c01. lea rcx, [0x00015c90]
0x0000507d      488d3d9ce5ff. lea rdi, [0x00003620]

[0x00005060]> px
- offset -  0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00005060  31ed 4989 d15e 4889 e248 83e4 f050 544c 1.I.^H..H...PTL
0x00005070  8d05 8a0c 0100 488d 0d13 0c01 0048 8d3d .....H.....H.=
0x00005080  9ce5 ffff ff15 6ead 2100 f40f 1f44 0000 .....n.!...D..
0x00005090  488d 3dd1 b121 0055 488d 05c9 b121 0048 H.=..!.UH...!.H
0x000050a0  39f8 4889 e574 1948 8b05 eaab 2100 4885 9.H..t.H...!.H
0x000050b0  c074 0d5d ffe0 662e 0f1f 8400 0000 0000 .t.]..f.....
0x000050c0  5dc3 0f1f 4000 662e 0f1f 8400 0000 0000 ]...@.f.....
0x000050d0  488d 3d91 b121 0048 8d35 8ab1 2100 5548 H.=..!.H.5..!.UH
0x000050e0  29fe 4889 e548 c1fe 0348 89f0 48c1 e83f ).H..H...H..H..?
0x000050f0  4801 c648 d1fe 7418 488b 05a9 ae21 0048 H..H..t.H...!.H
0x00005100  85c0 740c 5dff e066 0f1f 8400 0000 0000 ..t.]..f.....
0x00005110  5dc3 0f1f 4000 662e 0f1f 8400 0000 0000 ]...@.f.....
0x00005120  803d a1b1 2100 0075 2f48 833d 97ae 2100 .=..!.u/H.=..!.
0x00005130  0055 4889 e574 0c48 8b3d caae 2100 e8cd .UH..t.H.=..!.
0x00005140  e4ff ffe8 48ff ffff c605 79b1 2100 015d ....H....y.!..]
0x00005150  c30f 1f80 0000 0000 f3c3 660f 1f44 0000 .....f..D..

[0x00005060]>
```


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

Dedicated radare2 flashair:// IO plugin

interact with the card from radare2

call `dump` commands over Telnet

ease exploring the card memory

using radare2 commands

```
$ r2 -i io_flashair.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
```

Reversing With Strings

Goals

auto-name functions

using errors format strings

high-level knowledge

using strings as hints

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

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

TEL.SendLoginMessage()

```
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]
```

```
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
```

High-Level Knowledge

use strings as RE hints

discover functions manipulating specific strings

strategy

1. assemble `MOVU R1,<string address>`
2. find the closest function prologue


```
$ flashre hints dump_w03.bin --offset 0xc0000 update
0xc20580 0xc20c82 update -f %s
====
0xc96870 0xc969c6 FwUpdate error f_open(%s) ret=%d\n
0xc96870 0xc96a36 \nUpdate fail. Unexpected target name.\n
0xc96870 0xc96b3e \nUpdate reserved.\n
====
0xc9b502 0xc9b51a USAGE: sd update filename
0xc9b502 0xc9b65a \nUpdate fail. Unexpected target name.\n
0xc9b502 0xc9b722 \nUpdate success.\n
0xc9b502 0xc9b780 Update error.(checksum)\n
```

update hints

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
```

W-04 Update Header

same as W-03

the third field is “W-04”

```
$ flashre update fwupdate.fbn
###[ FlashAir Update Header ]###
card      = 'FLASHAIR'
type      = 'MAIN2'
unk0      = 'W-04'
unk1      = 0xd485
unk2      = 0x1f002b0f
checksum  = 0x45
unk3      = 0x0
length    = 1500492
```

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

```
$ R2M2_ARCH=mep1 r2 -a r2m2 fwupdate.fbn -m $((0xc10000 - 32))
[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

Reversing the Configuration Parser

parse_config() - 0xc15f4e

configure values

APPSSID, APPNETWORKKEY ...

start daemons

TELNET, DHCP_Enabled ...

execute commands

COMMAND

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;
```

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\  
0x0000e644 chg_ilv error(%04x) in fb_sio_init\  
0x0000e668 wai_flg error(%d) in fb_getc\  
0x000cfff0c chg_ilv error(%04x) in fb_sio_init\  
0x000cfff30 wai_flg error(%d) in fb_getc\  
0x000e9730 wup_tsk error(%d) in fb_sio_isr\  
0x000e9751 set_flg error(%04x) in fb_sio_isr\  

```

wup_tsk() looks promising!

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

The Real-time Operating system Nucleus



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

Which TRON Implementation?

```
$ rabin2 -zzz dump_w03.bin |grep -i nucleus
0x000a4103 NetNucleus WPS version %d.%d.%d
0x000eafcd NetNucleus WPS version %d.%d.%d
```

NetNucleus - IP stack from Toshiba for UDEOS

Reading μ TRON 4.0 Specification

[Differences from the μ TRON3.0 Specification]

The task state names are now in the adjective form. They **have been renamed** from **RUN** to **RUNNING**, from **WAIT** to **WAITING**, from **SUSPEND** to **SUSPENDED**, and from **WAIT-SUSPEND** to **WAITING-SUSPENDED**. [..]

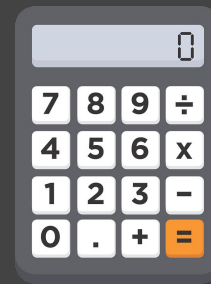
```
$ rabin2 -zzz dump_w03.bin |egrep 'RUN|WAIT|SUSPEND'  
0x000d7574 WAITING-SUSPENDED  
0x000d7586 SUSPENDED  
0x000d7590 WAITING  
0x000d759e RUNNING
```



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()

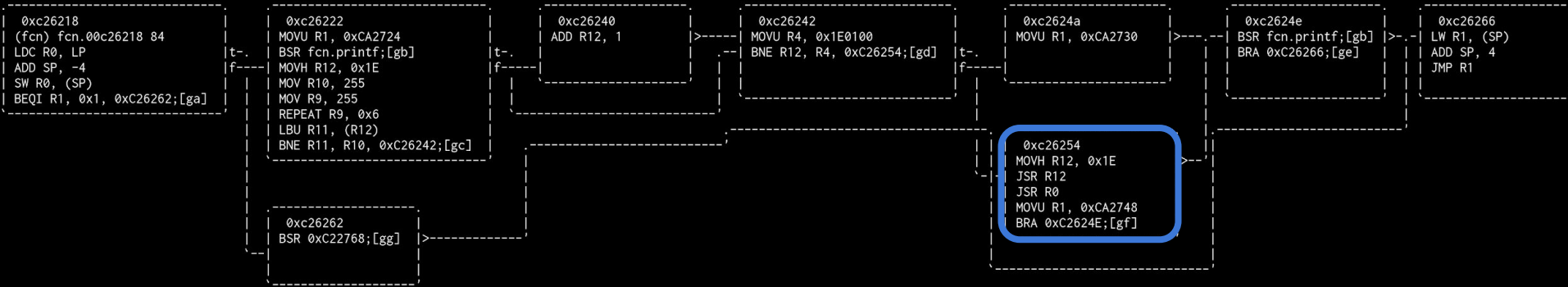
move task to READY state

```
[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

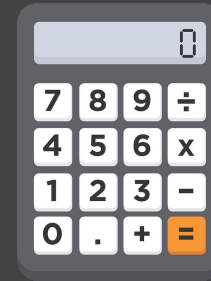


checks that the **USRPRG** section (0x1e0000) is not 0xff
jumps 0x1e0000
calls the function stored at R0

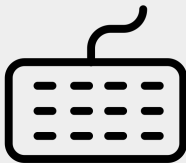
Game Plan



- memory dump
- architecture
- Operating System
- execution vector



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



Created by Sarah
from Noun Project

Putting Everything Together

0. write the payload
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 FlashR!r
ESC R4539 built 15:37:44, Aug 28 2015
> version
version
FA0CA3A03.00.01
> update -f calc.update
update -f calc.update
F:-----40
> dump 0x1e0000 -l 10
dump 0x1e0000 -l 10
address=0x01e0000 length=0x10
c12ad83a 1e00d10a 202b7002 202d2d2d

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

+++ ++ ++ ++
[7] [8] [9] [1]
+++ ++ ++ ++

+++ ++ ++ ++
[4] [5] [6] [1]
+++ ++ ++ ++

+++ ++ ++ ++
[1] [2] [3] [1]
+++ ++ ++ ++

+++ ++
[0] [1]
+++ ++
-user_task
>
█
```



Project Outlook

identify remote vulnerabilities

DHCP, HTTP, 802.11, ...

SDK

gcc supports MeP

new firmwares

encrypt or hide pictures

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 thank_you.update
update -f thank_you.update
F:o-----+o
```

```
> userpg
userpg
+user_task
```

```
##### ##      ##      ###      ##      ## ##      ##      ##      ##      #####      ##      ##      #####
##      ##      ##      ## ##      ###      ## ##      ##      ##      ##      ##      ##      ##      ##      #####
##      ##      ##      ##      ##      #####      ## ##      ##      ##      ##      ##      ##      ##      ##      #####
##      #####      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##
##      ##      ##      #####      ##      #####      ##      ##      ##      ##      ##      ##      ##      ##
##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##
##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##      ##
```

```
-user_task
```


Few More Things

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)
0x000cfc61 802.11 Key Descriptor length is inconsistent
0x000cfcde Key Data Encapsulation '%' 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
```

```

$ r2 fwupdate.fbn
[0x00000000]> px 512
- offset -  0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00000000  464c 4153 4841 4952 4d41 494e 3200 0000  FLASHAIRMAIN2...
0x00000010  0102 0304 1c7e 1f00 250f c200 10fc 0f00  .....~..%.....
0x00000020  08d8 0101 28d8 0000 5b7c 101c 597c 0000  ....(...[|..Y|..
0x00000030  0270 0000 0000 0000 0000 0000 0000 0000  .p.....
0x00000040  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x00000050  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x00000060  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x00000070  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x00000080  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x00000090  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x000000a0  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x000000b0  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x000000c0  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x000000d0  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x000000e0  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x000000f0  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x00000100  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x00000110  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x00000120  0070 2859 5979 0059 5879 03eb a041 b5cc  .p(YYy.YXy...A..
0x00000130  0010 b5cb 2000 2a6b 5a6c c01b 30eb 5b00  .... .*kZl..0.[.

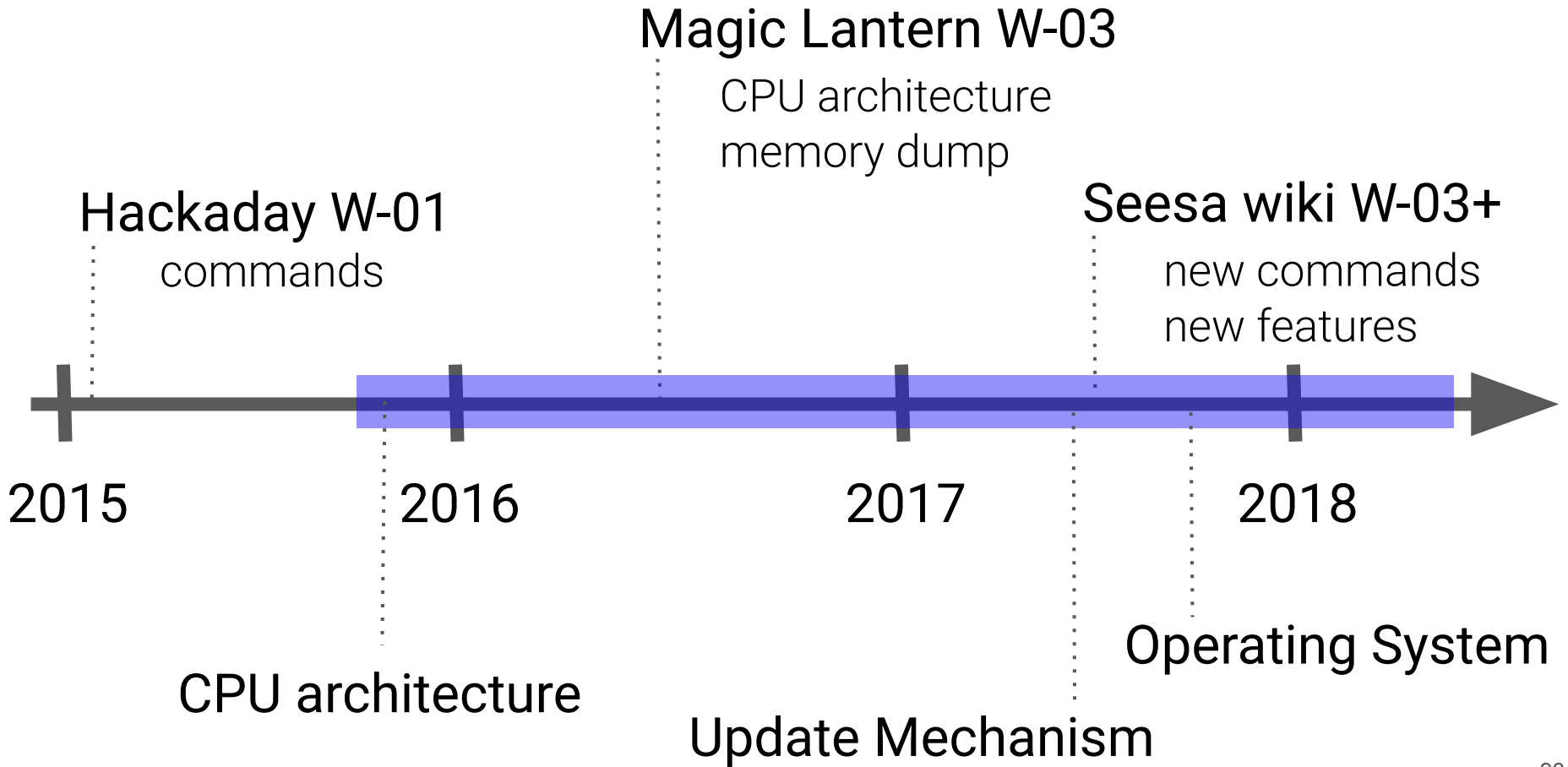
```

fwupdate.fbn content

```
$ telnet 192.168.0.1
Telnet escape character is '^]'.
Trying 192.168.0.1...
Connected to 192.168.0.1.
Escape character is '^]'.
Welcome to FlashAir
ESC R4539 built 15:37:44, Aug 28 2015
>
telnet> mode character

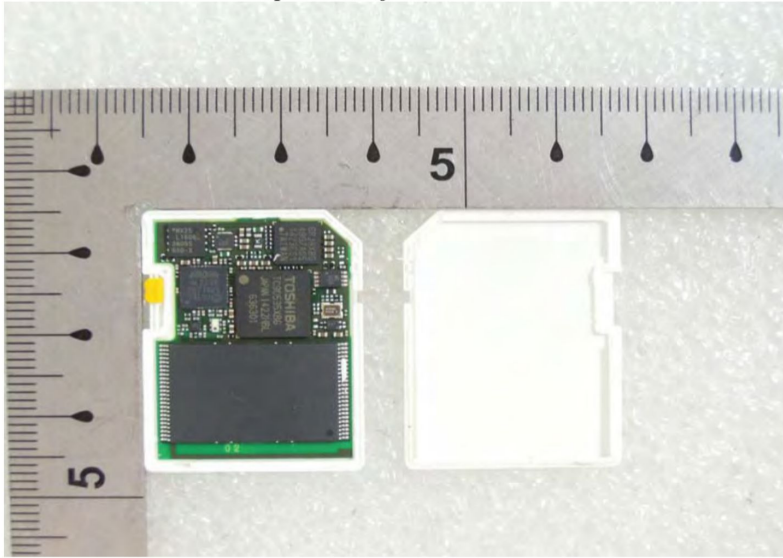
> version
version
FA9CAW3AW3.00.01
> exit
```

telnet character mode



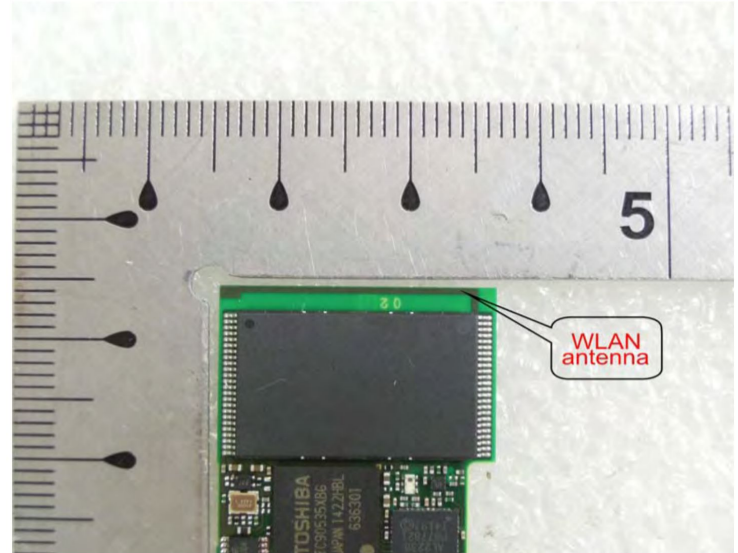
Pictures From the FCC Application

Open View of EUT – 1



chips markings

Antenna- WLAN



bonus information

*REPEAT Instructions

REPEAT and EREPEAT

E stands for Endless

three dedicated registers

RPB, RPC, RPE

loop over a block

two instructions executed at RPE

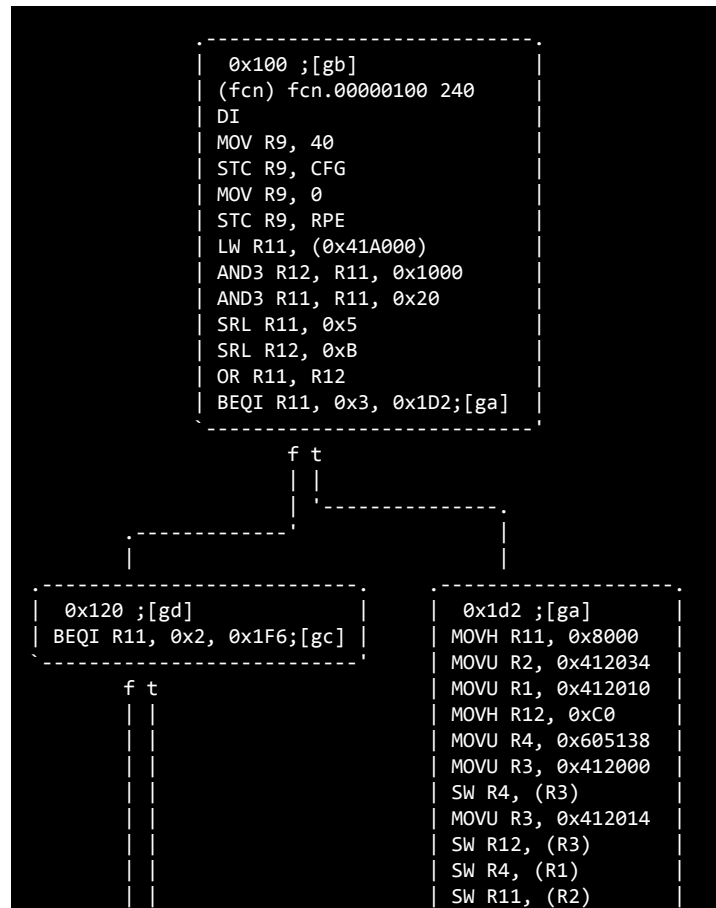
```
0x00c7fb84 ADD3 R12, R1, 0x1
0x00c7fb88 EREPEAT 0x6
RPB> 0x00c7fb8c LB R11, (R1)
RPE> 0x00c7fb8e ADD R1, 1
,=< 0x00c7fb90 BEQZ R11, 0xC7FB92
`-> 0x00c7fb92 MOV R0, R1
0x00c7fb94 SUB R0, R12
0x00c7fb96 RET
```

strlen()

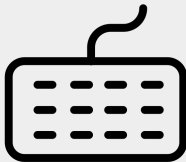
r2m2_Ae.so - Analysis

```
[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



callgraph



Created by Sarah
from Noun Project

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

```
$ 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 = B00T
    address : 0x00ceff40 = 0
    size : 0x00ceff42 = 65536
```

dumping section structures

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

Extract Configuration Parameters from Memory

some stored at fixed offsets from 0x817ae8

APPSSID, APPNETWORKKEY, 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

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
```

Executing Commands



command(char* command) # 0xc29e1c

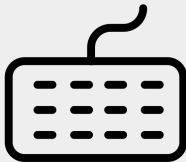
strcpy(0x81d6d8, command)

parse_argc_argv(0x81d6d8) # 0xc29bfc

execute_command(0x81d6d8) # 0xc29cce

memset(0x81d6d8, 0x284)

```
0xc29e1c ;[gg]
(fcn) fcn.command 120
LDC R0, LP
ADD SP, -4
SW R0, (SP)
MOV R2, R1
MOVU R1, 0x81B6D8
BSR fcn.strcpy;[gc]
MOVU R2, 0x81B6D8
MOVU R1, 0xCCF6BE
BSR fcn.printf;[gb]
MOVU R1, 0x81B6D8
BSR fcn.parse_argc_argv;[gd]
MOVU R1, 0x81B6D8
BSR fcn.execute_command;[ge]
MOVU R1, 0x81B6D8
MOV R2, 0
MOV R3, 284
BSR fcn.memset;[gf]
MOVU R2, 0xCCF6C2
MOVU R1, 0xCCF6C5
LW R0, (SP)
ADD SP, 4
STC R0, LP
JMP fcn.printf;[gb]
```



Created by Sarah
from Noun Project

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


```
$ 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