CTF Challenge Answers: Incident Response

Incident Response Report: Hunt the IoT monkey device

Created by: Karthick Sivanantham

Challenge:

Due to the recent attention to IoT devices being able intrude your network and spy, thought I'll make this challenge as close as possible to a real-world incident – News Links

"HALB Corporation is a Fortune 500 company who has 12 office locations across the globe. Especially 5 offices in Australia. They celebrated their new year by gifting every employee a sponge monkey doll. Most of the employees kept the doll in their cubical.

A Intel was received to the company's investigations team stating that one of the monkey doll is an IoT device that is capable of intruding the network and steal information. An incident responder started his hunt for this device by capturing traffic at the internet perimeter. He was able to identify some strange traffic and confirm the presence of the device within the network. He immediately called in for help from other incident responders for further analysis"

The HALB corp management team is after the following details

1. *collect basic Information on this IoT device (Trivial)*
2. *Retrieve the "**complete**" information sent from this IoT device.(Challenging)*
3. *Find the IoT device location to which branch office.(Tough)*

Executive Summary:

1. The IoT device Basic information

IP Address: 10.11.0.5
MAC Address: 00:12:9f:12:33:44
Destination Address: 198.91.81.7
Destination domain: hunt.pcriot.com

2. Retrieve the complete information

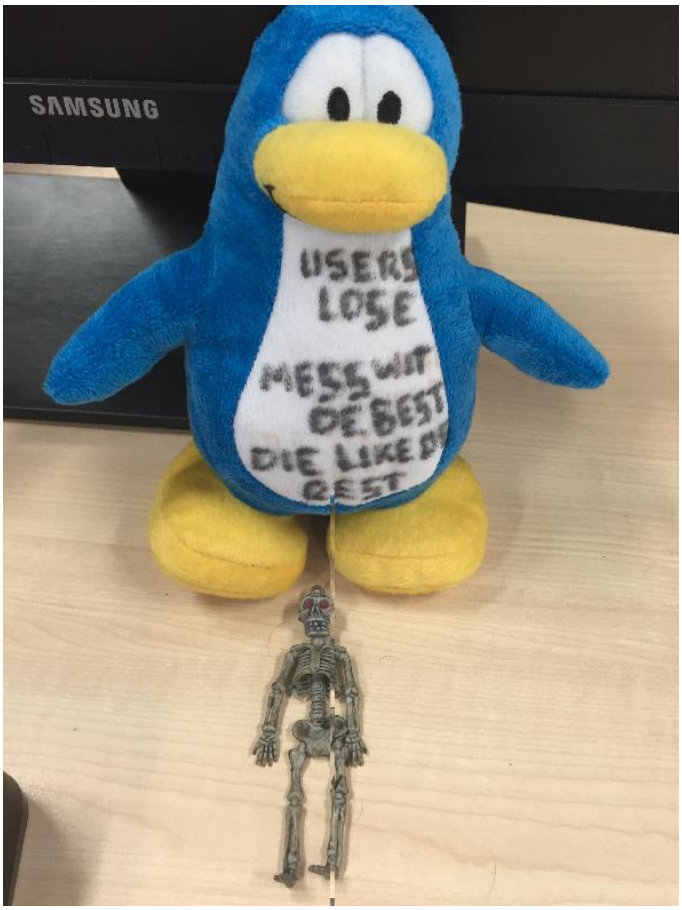
The device has performed a reconnaissance on the HALB_Guest Network and has the capability to take snapshots with hidden camera. Not sure about the further capabilities of this device.
All the data collected has been base64 encoded with added cruft to evade detection.

Retrieved information:
root$ ./startshell-1
wlan0 IEEE 802.11abgn ESSID:"HALB-Guest"
Mode:Managed Frequency:2.412 GHz Cell: 3C:D4:C7:6F:B5:4B
Tx-Power=20 dBm
Retry short limit:7 RTS thr:off Fragment thr:off
Encryption key:off
Power Management:off
lo no wireless extensions.
eth0 no wireless extensions.

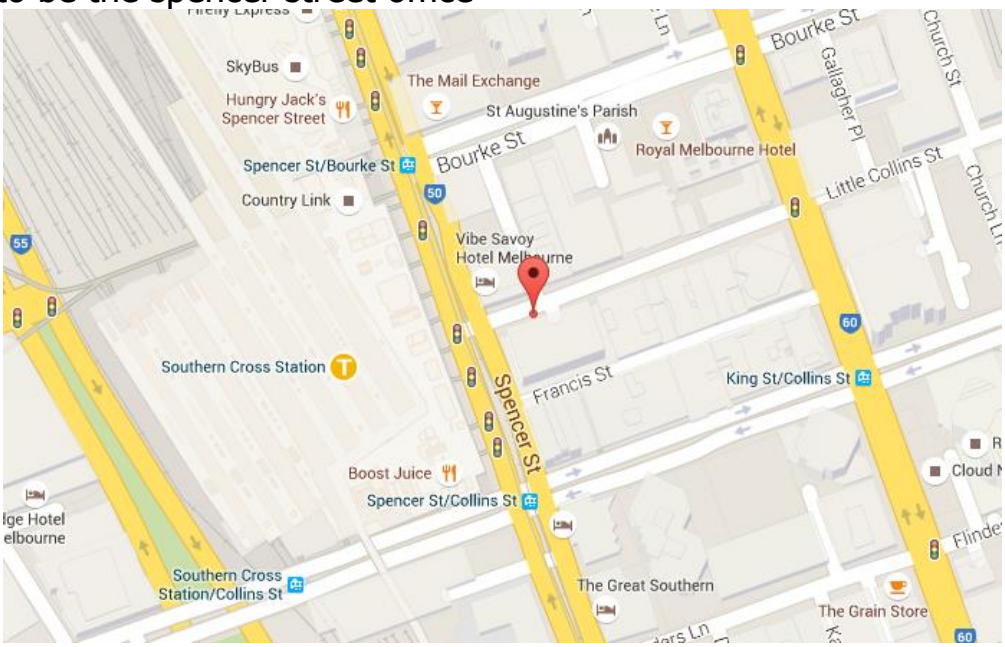
root$ ./wlan0-scan
wlan0 Scan completed :
Cell 01 - Address: 00:7F:28:35:9A:C7
Channel:1
Frequency:2.412 GHz (Channel 1)
Quality=29/70 Signal level=-81 dBm
Encryption key:on
ESSID:"CHC"
Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
9 Mb/s; 12 Mb/s; 18 Mb/s
Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
Mode:Master
Extra:tsf=000000412e67cddf
Extra: Last beacon: 5408ms ago
IE: Unknown: 00055837335A36
IE: Unknown: 010882848B960C121824
IE: Unknown: 030101
IE: Unknown: 200100
IE: IEEE 802.11i/WPA2 Version 1
Group Cipher : CCMP
Pairwise Ciphers (1) : CCMP
Authentication Suites (1) : PSK
IE: Unknown: 2A0100
IE: Unknown: 32043048606C
IE: Unknown: DD180050F2020101040003A4000027A4000042435E0062322F00
IE: Unknown: 2D1A8C131BFFFF000000000000000000000000000000000000000000
IE: Unknown: 3D160108080000000000000000000000000000000000000000
IE: Unknown: DD0900037F01010000FF7F
IE: Unknown: DD0A00037F04010000000000
IE: Unknown: 0706555320010B1B

```
Cell 02 - Address: 48:5D:36:08:68:DC
Channel:6
Frequency:2.412 GHz (Channel 1)
Quality=59/70  Signal level=-51 dBm
Encryption key:on
ESSID:"HALB-CORP"
Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
24 Mb/s; 36 Mb/s; 54 Mb/s
Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 48 Mb/s
Mode:Master
Extra:tsf=00000021701d828b
Extra: Last beacon: 4532ms ago
IE: Unknown: 000F736F6D657468696E67636C65766572
IE: Unknown: 010882848B962430486C
IE: Unknown: 030106
IE: Unknown: 0706555320010B1E
IE: Unknown: 2A0100
IE: Unknown: 2F0100
IE: IEEE 802.11i/WPA2 Version 1
Group Cipher : CCMP
Pairwise Ciphers (1) : CCMP
Authentication Suites (1) : PSK
Cell 03 - Address: 68:5C:12:97:75:3A
Channel:6
Frequency:2.412 GHz (Channel 1)
Quality=62/70  Signal level=-49 dBm
Encryption key:off
ESSID:"HALB-Guest"
Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
24 Mb/s; 36 Mb/s; 54 Mb/s
Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 48 Mb/s
Mode:Master
Extra:tsf=00000021701d8913
Extra: Last beacon: 5936ms ago
IE: Unknown: 000F736F6D657468696E67636C65766572
IE: Unknown: 010882848B962430486C
IE: Unknown: 030106
IE: Unknown: 0706555320010B1E
IE: Unknown: 2A0100
IE: Unknown: 2F0100

root$ ./sweep
10.11.0.12
10.11.0.45
10.11.0.64
10.11.0.89
10.11.0.92
10.11.0.121
10.11.0.1
10.11.0.3
10.11.0.62
10.11.0.78
10.11.0.90
10.11.0.32
10.11.0.87
10.11.0.61
10.11.0.42
10.11.0.98
10.11.0.151
10.11.0.25
```

```
10.11.0.13
10.11.0.48
root$ ./capture
root$ ./transfer,NAME=/root/Pictures/snapshot_CURRENT.jpg
```



3. Location of the device was retrieved by the geo co-ordinates in the image's metadata, to be the spencer street office



Technical Detail:

File: "internet_perimeter.pcap"
md5: 449f068399889883fd2d2a93a0bafc41

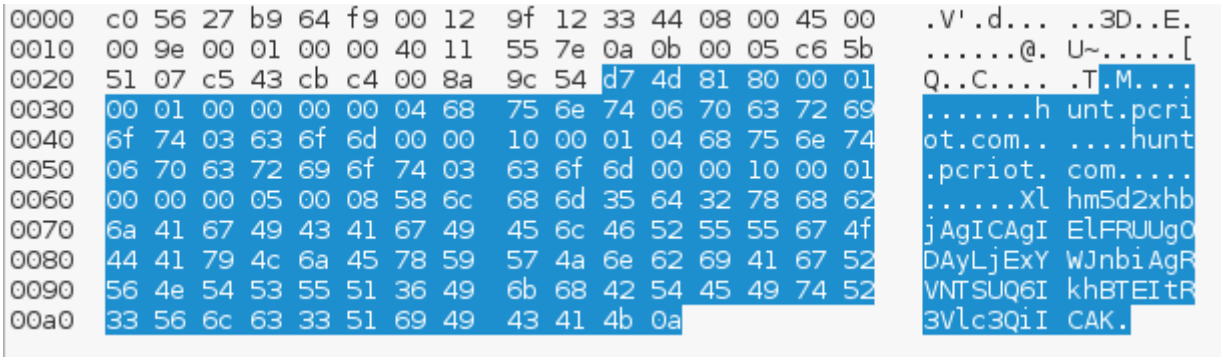
A quick check on the summary of the overall traffic collected reveals that "10.11.0.5" is been sending out a huge number of data outside

Address A	Address B	Packets	Bytes	Packets A→B	Bytes A→B
10.11.5.15	216.58.220.142	200	15 600	200	15 600
10.11.5.2	179.60.193.36	200	15 800	200	15 800
10.11.1.14	216.58.220.97	8 441	751 249	8 441	751 249
10.11.0.5	198.91.81.7	8 241	3 198 383	8 241	3 198 383
1.2.3.4	10.11.2.24	8 318	640 486	0	0
8.8.8.8	10.11.0.12	77	5 929	0	0

Details on other IP:

10.11.0.12 - 8.8.8.8(Google)
10.11.2.24 - 1.2.3.4 (Halbcorp domain)
10.11.1.14 - 216.58.220.97 (memoryexploit.blogspot.com.au - "My blog" :-p)
10.11.5.2 - 179.60.193.36 (facebook)
10.11.5.15 - 216.58.220.142 (youtube)

our suspicious traffic from 10.11.0.5 is towards 198.91.81.7 - hunt.pcriot.com with some strange data after the domain request in every single packet sent. chances can be this request traffic can be used to tunnel the data along with domain request.



hunt.pcriot.com - does not have any details other than some funny videos

Let's deep dive into dissecting and analysing more on this specific traffic. (Export traffic only from 10.11.0.5 and dissect with Scapy)

The pcap has total of 8241 UDP packets and nothing else
Upon inspecting the 1st packet the RAW.load section has our suspicious bits and bytes

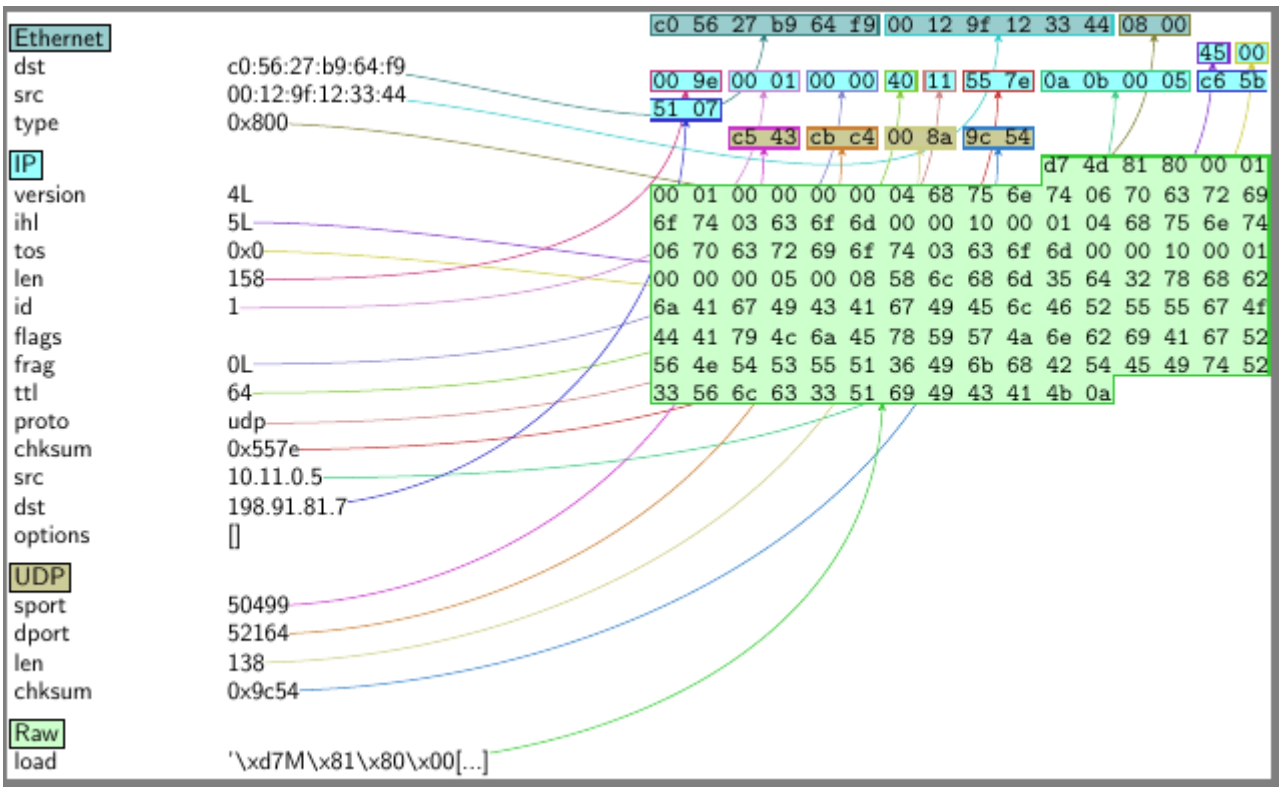
```
>>> pkt.show
<bound method PacketList.show of <hunt.pcap: TCP:0 UDP:8241 ICMP:0 Other:0>>
>>> pkt[0].show()
###[ Ethernet ]###
  dst= c0:56:27:b9:64:f9
  src= 00:12:9f:12:33:44
  type= 0x800
###[ IP ]###
     version= 4L
     ihl= 5L
     tos= 0x0
     len= 119
     id= 1
     flags=
     frag= 0L
     ttl= 64
     proto= udp
     chksum= 0x557e
     src= 10.11.0.5
     dst= 198.91.81.7
     \options\
###[ UDP ]###
        sport= 60770
        dport= 52164
        len= 99
        chksum= 0x9c54
###[ Raw ]###
           load= 'X\x07\x81\x80\x00\x01\x00\x01\x00\x00\x00\x00\x04hunt\x06pcriot\x03com\x00\x00\x10\x00\x01\x
04hunt\x06pcriot\x03com\x00\x00\x10\x00\x01\x00\x00\x00\x05\x00\x08cm9vdCQgLi9zdGFydHNoZWxsLTEK==\n'
```

Based on the "==" at the end we are able to determine that the data is base64 encoding

cm9vdCQgLi9zdGFydHNoZWxsLTEK==
decoded: root$ ./startshell-1==

looks like the == was manually added. since without that even the base64 math satisfies

cm9vdCQgLi9zdGFydHNoZWxsLTEK
decoded: root$ ./startshell-1



After inspecting the other samples our suspicious data sits on the same section of the packets " RAW - load ". Chances these packets may be injected with data by the device to hide information in base64 format

But our encode information starts after specific number of bytes and ends before the "\n" as per the red highlights below

```
>>> pkt[0].load
'X\x07\x81\x80\x00\x01\x00\x01\x00\x00\x00\x00\x04hunt\x06pcriot\x03com\x00\x00\x10\x00\x01\x04hunt\x06pcriot\
x03com\x00\x00\x10\x00\x01\x00\x00\x00\x05\x00\x08cm9vdCQgLi9zdGFydHNoZWxsLTEK==\n'
>>> pkt[1].load
'dB\x81\x80\x00\x01\x00\x01\x00\x00\x00\x00\x04hunt\x06pcriot\x03com\x00\x00\x10\x00\x01\x04hunt\x06pcriot\x03
com\x00\x00\x10\x00\x01\x00\x00\x00\x05\x00\x08\n'
>>> pkt[2].load
'\xd7M\x81\x80\x00\x01\x00\x01\x00\x00\x00\x00\x04hunt\x06pcriot\x03com\x00\x00\x10\x00\x01\x04hunt\x06pcriot\
x03com\x00\x00\x10\x00\x01\x00\x00\x00\x05\x00\x08Xlhm5d2xhbjAgICAgIElFRUUgODAyLjExYWJnbiAgRVNTSUQ6IkhBTEItR3V
lc3QiICAK\n'
>>> pkt[4].load
'\xf52\x81\x80\x00\x01\x00\x01\x00\x00\x00\x00\x04hunt\x06pcriot\x03com\x00\x00\x10\x00\x01\x04hunt\x06pcriot\
x03com\x00\x00\x10\x00\x01\x00\x00\x00\x05\x00\x08Xlhm5TW9kZTpNYW5hZ2VkICBGcmVxdWVuY3k6Mi40MTIgR0h6ICBDZWxsOiA
zQzpENDpDNzo2RjpCNTo0QiAgIAo=\n'
```

quick test on one of the packets, get our needed information

```
>>> pkt[4].load[60:-1]
'Xlhm5TW9kZTpNYW5hZ2VkICBGcmVxdWVuY3k6Mi40MTIgR0h6ICBDZWxsOiAzQzpENDpDNzo2RjpCNTo0QiAgIAo='
```

Time to automate & unravel the information . bring in the pythons :-p

quick code to iterate through every packet and extract the information. in the below code hunt.pcap is the specific suspicious IP's pcap file. "scapy.all" is imported to dissect the packet as we did in the pcap file

```
#!/usr/bin/python

from scapy.all import *
import base64

packets=rdpcap("hunt.pcap")

for pkt in packets:
        p_data=base64.b64decode(pkt.load[60:-1])
        print p_data
```

Results:

```
WARNING: No route found for IPv6 destination :: (no default route?)
root$ ./startshell-1


Traceback (most recent call last):
  File "./pcap_decode.py", line 9, in <module>
    p_data=base64.b64decode(pkt.load[60:-1])
  File "/usr/lib/python2.7/base64.py", line 76, in b64decode
    raise TypeError(msg)
TypeError: Incorrect padding
```

The first data got decode successfully but the code ended with error "incorrect padding" for the next data in the packet
So let’s pipe the output before base64 conversion and inspect further

The encoded data that caused the error :
Xlhm5d2xhbjAgICAgIElFRUUgODAyLjExYWJnbiAgRVNTSUQ6IkhBTEItR3Vlc3QiICAK

Decoding into base64 with HEX view for deeper look

XIhm5d2xhbjAgICAgIElFRUUgODAyLjExYWJnbiAgRVNTSUQ6IkhBTEItR3Vlc3QiICAK

0	5e	58	66	e5	dd	b1	85	b8	c0	80	80	80	80	81	25	15	^XfåÝ±□¸Àxxxx□%␕
1	15	14	80	e0	c0	c8	b8	c4	c5	85	89	9d	b8	80	81	15	␕␔xàÀÈ¸ÄÅ□□□¸x□␕
2	4d	4d	25	10	e8	89	21	05	31	08	b5	1d	d5	95	cd	d0	MM%◄è□!∴1√µ∝Õ□ÍÐ
3	88	80	80	4b	--	--	--	--	--	--	--	--	--	--	--	--	□xxK

69 characters in total - "not divisible by 4"
If it's a base64 it should be multiples of 4. so something wrong with this data

"chances the attacker might have added some cruft to make the make the decode throw some garbage to evade detection from the platforms" or "some random data which we might need to exclude in the decode" or "this might not be base64 at all"

let's sample more of these odd numbered characters to see if they have any pattern of noise

```
1  cm9vdCQgLi9zdGFydHNoZWxsLTEK==
2
3  Xlhm5d2xhbjAgICAgIElFRUUgODAyLjExYWJnbiAgRVNTSUQ6IkhBTEItR3Vlc3QiICAK
4
5  Xlhm5TW9kZTpNYW5hZ2VkICBGcmVxdWVuY3k6Mi40MTIgR0h6ICBDZWxsOiAzQzpENDpDNzo2RjpCNTo0QiAgIAo=
6
7  Xlhm5VHgtUG93ZXI9MjAgZEJtICAgCg==
8
9  Xlhm5UmV0cnkgc2hvcnQgbGltaXQ6NyAgIFJUUyB0aHI6b2ZmICAgRnJhZ21lbnQgdGhyOm9mZgo=
10
11 Xlhm5RW5jcnlwdGlvbiBrZXk6b2ZmCg==
12
13 Xlhm5UG93ZXIgTWFuYWdlbWVudDpvZmYK
14
15 Xlhm5ICAgICAgCg==
16
17 Xlhm5bG8gICAgICAgIG5vIHdpcmVsZXNzIGV4dGVuc2lvbnMuCg==
18
19 Xlhm5Cg==
20
21 Xlhm5ZXRoMCAgICAgIG5vIHdpcmVsZXNzIGV4dGVuc2lvbnMuCg==
22
23 cm9vdCQgLi93bGFuMC1zY2FuCg==
24
25 Xlhm5d2xhbjAgICAgIFNjYW4gY29tcGxldGVkIDoK
26
27 Xlhm5Q2VsbCAwMSAtIEFkZHJlc3M6IDAwOjdGOjI4OjM1OjlBOkM3Cg==
28
29 Xlhm5Q2hhbm5lbDoxCg==
30
31 Xlhm5RnJlcXVlbmN5OjIuNDEyIEdIeiAoQ2hhbm5lbCAxKQo=
```

Looks like there is a pattern in the odd numbered data. let's give a quick decode ignoring the highlighted pattern

Encoded data (Line 3 from above image):
d2xhbjAgICAgIElFRUUgODAyLjExYWJnbiAgRVNTSUQ6IkhBTEItR3Vlc3QiICAK
Decoded data:
wlan0   IEEE 802.11abgn ESSID:"HALB-Guest"

let’s now add this logic to our python code

```
#!/usr/bin/python

from scapy.all import *
import base64

file_n=open("output.txt",'w')
packets=rdpcap("hunt.pcap")

for pkt in packets:
        p_data=pkt.load[60:-1]
        if(p_data.startswith('Xlhm5')):
                line = p_data[5:]
                deco2=base64.b64decode(line)
                sys.stdout=file_n
                print deco2
        else:
                deco1=base64.b64decode(p_data)
                sys.stdout=file_n
                print deco1
```

The changes from previous code are the logic to check for the cruft and remove it before decode. Also I have piped the output to txt file even if the data is not interpreted correctly by my terminal, just for later inspection

Results:

```
root$ ./startshell-1
wlan0     IEEE 802.11abgn  ESSID:"HALB-Guest"
Mode:Managed  Frequency:2.412 GHz  Cell: 3C:D4:C7:6F:B5:4B
Tx-Power=20 dBm
Retry short limit:7   RTS thr:off   Fragment thr:off
Encryption key:off
Power Management:off
lo        no wireless extensions.
eth0      no wireless extensions.

root$ ./wlan0-scan
wlan0     Scan completed :
Cell 01 - Address: 00:7F:28:35:9A:C7
Channel:1
Frequency:2.412 GHz (Channel 1)
Quality=29/70  Signal level=-81 dBm
Encryption key:on
ESSID:"CHC"
Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
9 Mb/s; 12 Mb/s; 18 Mb/s
Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
Mode:Master
Extra:tsf=000000412e67cddf
Extra: Last beacon: 5408ms ago
IE: Unknown: 00055837335A36
IE: Unknown: 010882848B960C121824
IE: Unknown: 030101
IE: Unknown: 200100
IE: IEEE 802.11i/WPA2 Version 1
```

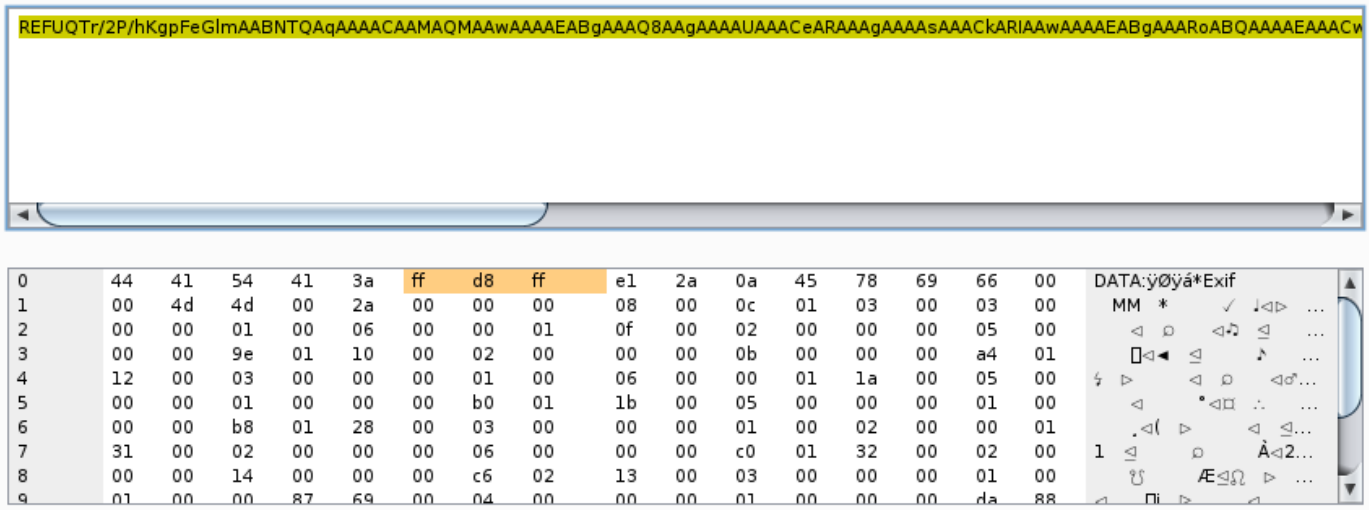
```
Group Cipher : CCMP
Pairwise Ciphers (1) : CCMP
Authentication Suites (1) : PSK
IE: Unknown: 2A0100
IE: Unknown: 32043048606C
IE: Unknown: DD180050F2020101040003A4000027A4000042435E0062322F00
IE: Unknown: 2D1A8C131BFFFF000000000000000000000000000000000000000000
IE: Unknown: 3D160108080000000000000000000000000000000000000000
IE: Unknown: DD0900037F01010000FF7F
IE: Unknown: DD0A00037F04010000000000
IE: Unknown: 0706555320010B1B
Cell 02 - Address: 48:5D:36:08:68:DC
Channel:6
Frequency:2.412 GHz (Channel 1)
Quality=59/70  Signal level=-51 dBm
Encryption key:on
ESSID:"HALB-CORP"
Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
24 Mb/s; 36 Mb/s; 54 Mb/s
Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 48 Mb/s
Mode:Master
Extra:tsf=00000021701d828b
Extra: Last beacon: 4532ms ago
IE: Unknown: 000F736F6D657468696E67636C65766572
IE: Unknown: 010882848B962430486C
IE: Unknown: 030106
IE: Unknown: 0706555320010B1E
IE: Unknown: 2A0100
IE: Unknown: 2F0100
IE: IEEE 802.11i/WPA2 Version 1
Group Cipher : CCMP
Pairwise Ciphers (1) : CCMP
Authentication Suites (1) : PSK
Cell 03 - Address: 68:5C:12:97:75:3A
Channel:6
Frequency:2.412 GHz (Channel 1)
Quality=62/70  Signal level=-49 dBm
Encryption key:off
ESSID:"HALB-Guest"
Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
24 Mb/s; 36 Mb/s; 54 Mb/s
Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 48 Mb/s
Mode:Master
Extra:tsf=00000021701d8913
Extra: Last beacon: 5936ms ago
IE: Unknown: 000F736F6D657468696E67636C65766572
IE: Unknown: 010882848B962430486C
IE: Unknown: 030106
IE: Unknown: 0706555320010B1E
IE: Unknown: 2A0100
IE: Unknown: 2F0100

root$ ./sweep
10.11.0.12
10.11.0.45
10.11.0.64
10.11.0.89
10.11.0.92
10.11.0.121
10.11.0.1
```

10.11.0.3
10.11.0.62
10.11.0.78
10.11.0.90
10.11.0.32
10.11.0.87
10.11.0.61
10.11.0.42
10.11.0.98
10.11.0.151
10.11.0.25
10.11.0.13
10.11.0.48
root$ ./capture
root$ ./transfer,NAME=/root/Pictures/snapshot_CURRENT.jpg
*DATA:ÿØÿá**

Based on the above output looks like the device is capable of running scripts to check on the adapters, perform a wlan scan, sweep on the network & capable of taking a picture with an hidden camera

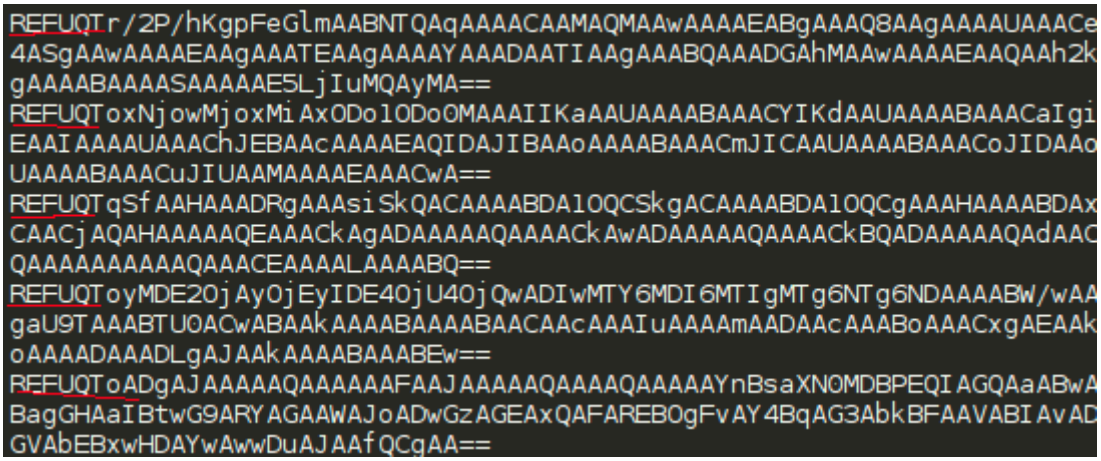
root$ ./transfer,NAME=/root/Pictures/snapshot_CURRENT.jpg
*DATA:ÿØÿá**

From the last bit of strings shown above. looks like after the snapshot some data is being passed which is successfully decoding, but not human readable. chances that this can be image data. let's perform an Hex view base64 decode to have a closer look at the bits and bytes



based on the above highlights, it's confirmed that the JPEG image is being transferred. Even checking the trailer of the final data packet gives FF D9.

Upon sampling further data section, we kind of got the logic to decode, so the data lines start with "REFUQT" before decode



and starts with "DATA:" after decode. so we need to remove the "DATA:" and group the rest of data as image file.

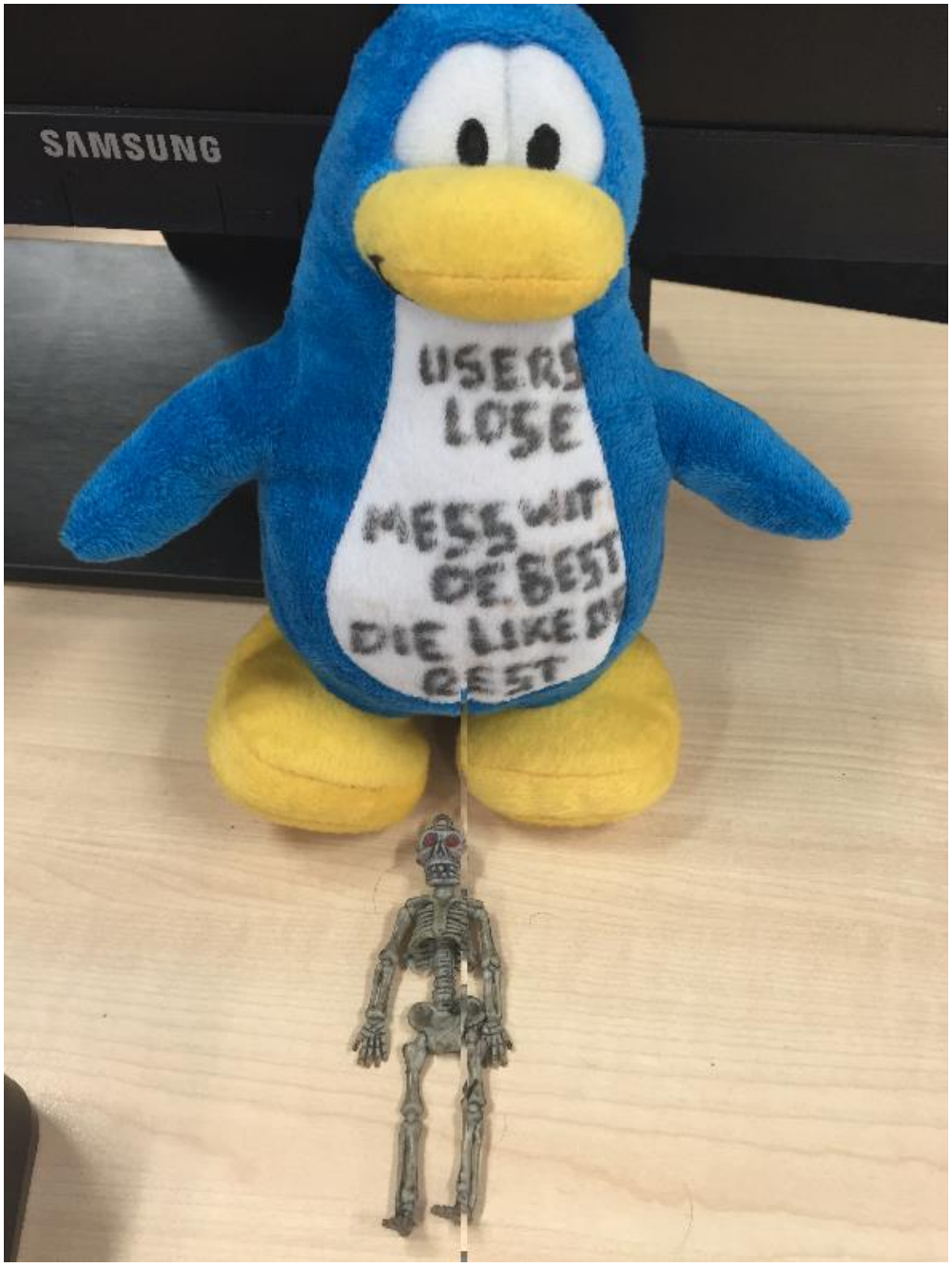
Time to teach our python to carve files :-p

As per the logic analysed. the code bits added were checking if the data section starts with 'REFUQ' and strip of the "DATA:" after decoding and save it in a file.

```
if(p_data.startswith('REFUQ')):
        deco2=base64.b64decode(p_data)
        if 'DATA' in deco2:
                image_data +=deco2.replace('DATA:', '')
                image = True
        continue

with open('picture_new.jpg','wb') as f:
        f.write(image_data)
```

Results:

Carved Image



Taking a look at the metadata of the image gives us further details

```
File Name                       : picture_new.jpg
Directory                       : .
File Size                       : 1663 kB
File Modification Date/Time     : 2016:02:16 00:42:31+11:00
File Access Date/Time           : 2016:02:21 21:47:34+11:00
File Inode Change Date/Time     : 2016:02:16 00:46:03+11:00
File Permissions                : rw-r--r--
File Type                       : JPEG
MIME Type                       : image/jpeg
Exif Byte Order                 : Big-endian (Motorola, MM)
Compression                     : JPEG (old-style)
Make                            : W00t
Camera Model Name               : W00t iOS 9
Orientation                     : Rotate 90 CW
X Resolution                    : 72
Y Resolution                    : 72
Resolution Unit                 : inches
Software                        : 9.2.1
Modify Date                     : 2016:02:12 18:58:40
Y Cb Cr Positioning             : Centered
Exposure Time                   : 1/33
F Number                        : 2.2
Exposure Program                : Program AE
ISO                             : 125
Exif Version                    : 0221
Date/Time Original              : 2016:02:12 18:58:40
Create Date                     : 2016:02:12 18:58:40
Components Configuration        : Y, Cb, Cr, -
Shutter Speed Value             : 1/33
Aperture Value                  : 2.2
Brightness Value                : 2.915427986
Exposure Compensation           : 0
Metering Mode                   : Spot
Flash                           : Off, Did not fire
Focal Length                    : 4.2 mm
Subject Area                    : 1845 969 610 612
Run Time Flags                  : Valid
Run Time Value                  : 27328974285416
Run Time Epoch                  : 0
Run Time Scale                  : 1000000000
Sub Sec Time Original           : 059
Sub Sec Time Digitized          : 059
Flashpix Version                : 0100
Color Space                     : sRGB
Exif Image Width                : 3264
Exif Image Height               : 2448
Sensing Method                  : One-chip color area
Scene Type                      : Directly photographed
Exposure Mode                   : Auto
White Balance                   : Auto
Focal Length In 35mm Format     : 29 mm
```

```
Scene Capture Type              : Standard
Lens Info                       : 4.15mm f/2.2
Lens Make                       : W00t
Lens Model                      : W00t 9 back camera 4.15mm f/2.2
GPS Latitude Ref                : South
GPS Longitude Ref               : East
Thumbnail Offset                : 1828
Thumbnail Length                : 8938
Image Width                     : 3264
Image Height                    : 2448
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Aperture                        : 2.2
GPS Latitude                    : 37 deg 49' 4.74" S
GPS Longitude                   : 144 deg 57' 15.13" E
GPS Position                    : 37 deg 49' 4.74" S, 144 deg 57' 15.13" E
Image Size                      : 3264x2448
Run Time Since Power Up         : 7:35:28
Scale Factor To 35 mm Equivalent: 6.9
Shutter Speed                   : 1/33
Create Date                     : 2016:02:12 18:58:40.059
Date/Time Original              : 2016:02:12 18:58:40.059
Thumbnail Image                 : (Binary data 8938 bytes, use -b option to extract)
Circle Of Confusion             : 0.004 mm
Field Of View                   : 63.7 deg
Focal Length                    : 4.2 mm (35 mm equivalent: 29.0 mm)
Hyperfocal Distance             : 1.84 m
Light Value                     : 7.0
```

Looks the hidden cam is capable of storing the geo co-ordinates.

so we have gathered all the information for the management to perform the sweep for the device in the Spencer street office

Game Over :-)