CallRec - Forensic

Le challenge donnait à disposition une sorte de dump partiel squashfs (ou équivalent) d'une ROM d'un téléphone. En observant rapidement les fichiers à notre disposition. On se rend compte qu'ils y a quelques fichiers qui vont nous être intéressant dans notre investigation :

```
$ tree a

a
├── customer
...
│   ├── nvmgr
│   │   ├── Feature.dat
│   │   ├── NvData.dat
│   │   ├── NvInit.dat
│   │   ├── NvOrig.dat
│   │   ├── NvSize.dbg
│   │   └── VariantVersion.dat
...

$ tree sysv

sysv
...
├── heecallrec.pm0
├── heecallrec.pmg
...
├── heeime.pm0
├── heeime.pmg
```

Si on regarde la différence entre les fichiers `pmg` et `pm0`, on remarque vite qu'ils ont le même hash pour la plupart. On va donc s'attarder que sur les `pmg` ici.

Le poids

En faisant un bête `strings` comme tout bon chall de forensic qui se respecte, on apperçoit que dans `NvInit.dat`, il y a a un modèle de téléphone Samsung qui ressort :

```
strings -e l a/customer/nvmgr/NvInit.dat

Samsung
GT-E1050
GSM Dual Band
```

On pense donc que notre téléphone est un GT-E1050.

Avec une simple recherche sur internet on tombe sur un lien disant que le tépéhone fait 67.30 g. On va alors garder `67` pour le flag format.

Le S/N

Après moultes recherches, il semblerait que le S/N soit retrouvable à partir de l'IMEI du téléphone (identifiant international unique du device). Toujours selon internet, ce numéro est composé de 15 chiffres et commence par `35`.

J'ai donc écrit une règle YARA qui va chercher si notre pattern est présent sur le fs donné.

```
rule IMEI
{
    strings:
        $hex_string_wide = {?? ?? ?? ?? ?? ?? ?? 35}
        $hex_string = {35 ?? ?? ?? ?? ?? ?? ??}
        $regex_wide = /35[\d]{13}/ wide
        $regex = /35[\d]{13}/
    condition:
        any of them
}
```

On lance alors cette règle sur tous les fichiers ou alors sur celui qui semble être prometteur (`heeime.pmg`) :

```
yara -r -s imei.yar ./sysv/heeime.pmg

0xc:$hex_string: 35 61 90 04 39 93 15 2F
```

Etant donné que le format est de 15 caractères décimal, on retire le `F` à la fin et on obtient `356190043993152`. Le numéro peut alors ensuite être rentré dans des bases de recherches sur internet afin d'y retrouver notre S/N.

Par exemple sur https://www.imeipro.info/samsung_imei_check.html, on retrouve bien notre `GT-E1050`. Problème : notre S/N contient des lettres, or dans le format flag il y a que des chiffres.

En faisant une recherche style google dorks sur l'IMEI, on trouve une nouvelle ressource :

```
Dork : "356190043993152"
Seul lien trouvé : https://swappa.com/imei/info/356190043993152
```

En fait le S/N était juste la partie suivante (extraite de l'IMEI) : `399315`

La date du dernier appel

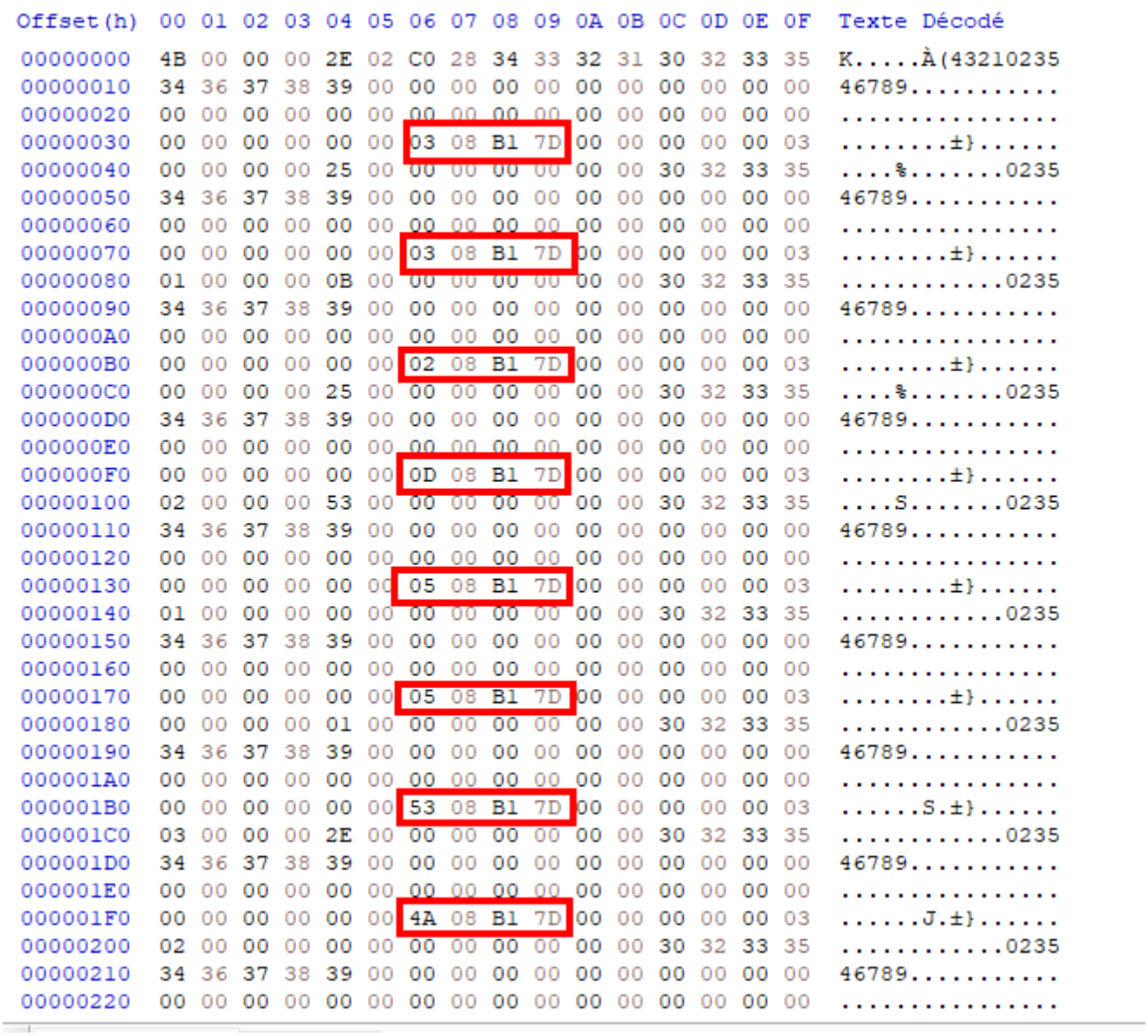
C'est là que la partie se gâte... des jours et des nuits passées à chercher cette fameuse date :clown:.

On commence part chercher le numéro de téléhone dont on nous parle pour savoir où orienter nos recherches :

```
find . -type f -exec bash -c 'cat {} | strings | grep 023546789 && echo {}' \;
2>/dev/null

./sysv/heecallrec.pm0
./sysv/heecallrec.pmg
```

Très bien, si on se penche sur ce fichier on remarque qu'il est relativement vide outre le numéro de téléphone répété plusieurs fois. Toutefois un patterne se répète avec l'octet de poid fort de la suite d'octet qui change



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Texte Décodé
00000000	4B	00	00	00	2E	02	C0	28	34	33	32	31	30	32	33	35	K.....À(43210235
00000010	34	36	37	38	39	00	00	00	00	00	00	00	00	00	00	00	46789..........
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000030	00	00	00	00	00	00	03	08	B1	7D	00	00	00	00	00	03	........±}......
00000040	00	00	00	00	25	00	00	00	00	00	00	00	30	32	33	35	....%.......0235
00000050	34	36	37	38	39	00	00	00	00	00	00	00	00	00	00	00	46789..........
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000070	00	00	00	00	00	00	03	08	B1	7D	00	00	00	00	00	03	........±}......
00000080	01	00	00	00	0B	00	00	00	00	00	00	00	30	32	33	35	0235
00000090	34	36	37	38	39	00	00	00	00	00	00	00	00	00	00	00	46789..........
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000000B0	00	00	00	00	00	00	02	08	B1	7D	00	00	00	00	00	03	........±}......
000000C0	00	00	00	00	25	00	00	00	00	00	00	00	30	32	33	35	....%.......0235
000000D0	34	36	37	38	39	00	00	00	00	00	00	00	00	00	00	00	46789..........
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000000F0	00	00	00	00	00	00	0D	08	B1	7D	00	00	00	00	00	03	........±}......
00000100	02	00	00	00	53	00	00	00	00	00	00	00	30	32	33	35	....S.......0235
00000110	34	36	37	38	39	00	00	00	00	00	00	00	00	00	00	00	46789..........
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000130	00	00	00	00	00	00	05	08	B1	7D	00	00	00	00	00	03	........±}......
00000140	01	00	00	00	00	00	00	00	00	00	00	00	30	32	33	35	0235
00000150	34	36	37	38	39	00	00	00	00	00	00	00	00	00	00	00	46789..........
00000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000170	00	00	00	00	00	00	05	08	B1	7D	00	00	00	00	00	03	........±}......
00000180	00	00	00	00	01	00	00	00	00	00	00	00	30	32	33	35	0235
00000190	34	36	37	38	39	00	00	00	00	00	00	00	00	00	00	00	46789..........
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000001B0	00	00	00	00	00	00	53	08	B1	7D	00	00	00	00	00	03	......S.±}......
000001C0	03	00	00	00	2E	00	00	00	00	00	00	00	30	32	33	35	0235
000001D0	34	36	37	38	39	00	00	00	00	00	00	00	00	00	00	00	46789..........
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000001F0	00	00	00	00	00	00	4A	08	B1	7D	00	00	00	00	00	03	......J.±}......
00000200	02	00	00	00	00	00	00	00	00	00	00	00	30	32	33	35	0235
00000210	34	36	37	38	39	00	00	00	00	00	00	00	00	00	00	00	46789..........
00000220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

Si on extrait toutes ces valeurs et qu'on les trient de manière unique et par valeur (on cherche le dernier appel) on obtient :

```
53 08 B1 7D
4A 08 B1 7D
```

```
49 08 B1 7D
48 08 B1 7D
0D 08 B1 7D
05 08 B1 7D
04 08 B1 7D
03 08 B1 7D
02 08 B1 7D
```

A présent, toujours en supposant que ce sont nos timestamps que l'on cherche, il faut trouver comment les interpréter. Decode, HxD, internet... tout le monde y est passé.

J'ai même tenté de :

- Chercher dans les leaks de Samsung
- Reverse un firmware d'un modèle similaire
- Reverse des logiciels de live forensic de l'époque

J'ai fini au bout de 4 jours à trouver une ressource intéressant en cherchant `samsung "date" format forensic`. Je suis tombé sur cette ressource : https://sqliteforensictoolkit.com/a-brief-history-of-time-stamps/

BitDate (seen on LG and Samsung phones)

A rather arbitrary name applied by me before I was aware of the devices on which this date format could be seen.

The date is stored as a 32-bit integer and the different components of the date are specified by different bits within the integer, in a similar manner to a DOS FAT date and time, however, there are no seconds recorded by the format.

i.e. the lowest 12 bits specify the year, the next 4 bits the month, then 5 bits each for the day & hour and then the last 6 bits for the minute.

	Year	Month	Day	Hour	Min	Seconds
Bits	32-20	19-16	15-11	10-6	5-0	Not recorded

Big loul

Si on applique l'algorithme sur la plus grand valeur hexa précédente on obtient :

```
# Big endian
010100110000 1000 10110 00101 111101
1328         8    22    5     61

# Little endian
011111011011 0001 00001 00001 010011
2011         1    1     1     19
```

Sachant que le téléphone aurait été fait en 2011, d'après notre recherche sur l'IMEI... cette date est intéressant ! Ce serait donc le 01/01/2011 à 01h19. Mais qu'en est-il des secondes ?

Beh fallait les guess loul. Au bout de 4/5 essaies le flag passe :

```
echo -n "67-399315-01:19:00" | md5sum
ECW{2f3399044a56dda1d6b2669f8fb74b65}
```