Is Hajime botnet dead?

blog.netlab.360.com/hajime-status-report-en/

RootKiter September 20, 2017

20 September 2017 / IoT Botnet

Overview

The mysterious Hajime botnet was first discovered by Rapiditynetworks in Oct 2016, and it was all over the news earlier this year, but it seems that nobody talks about it any more now, is this botnet gone?

The answer is no, our team has been tracking this botnet for quite some time, and we have recently noticed the number of infections has been going up, and a very interesting feature, an x64 config has been added in the code(is the botnet author eying PC platform as the next target? Or key leaked?).

Unlike the traditional Hajime-runs-in-sandbox solution, which really has no control over the bot besides merely observing it(a good example, researcher cannot ask the bot in sandbox to go over all the peering nodes and pull all the files).

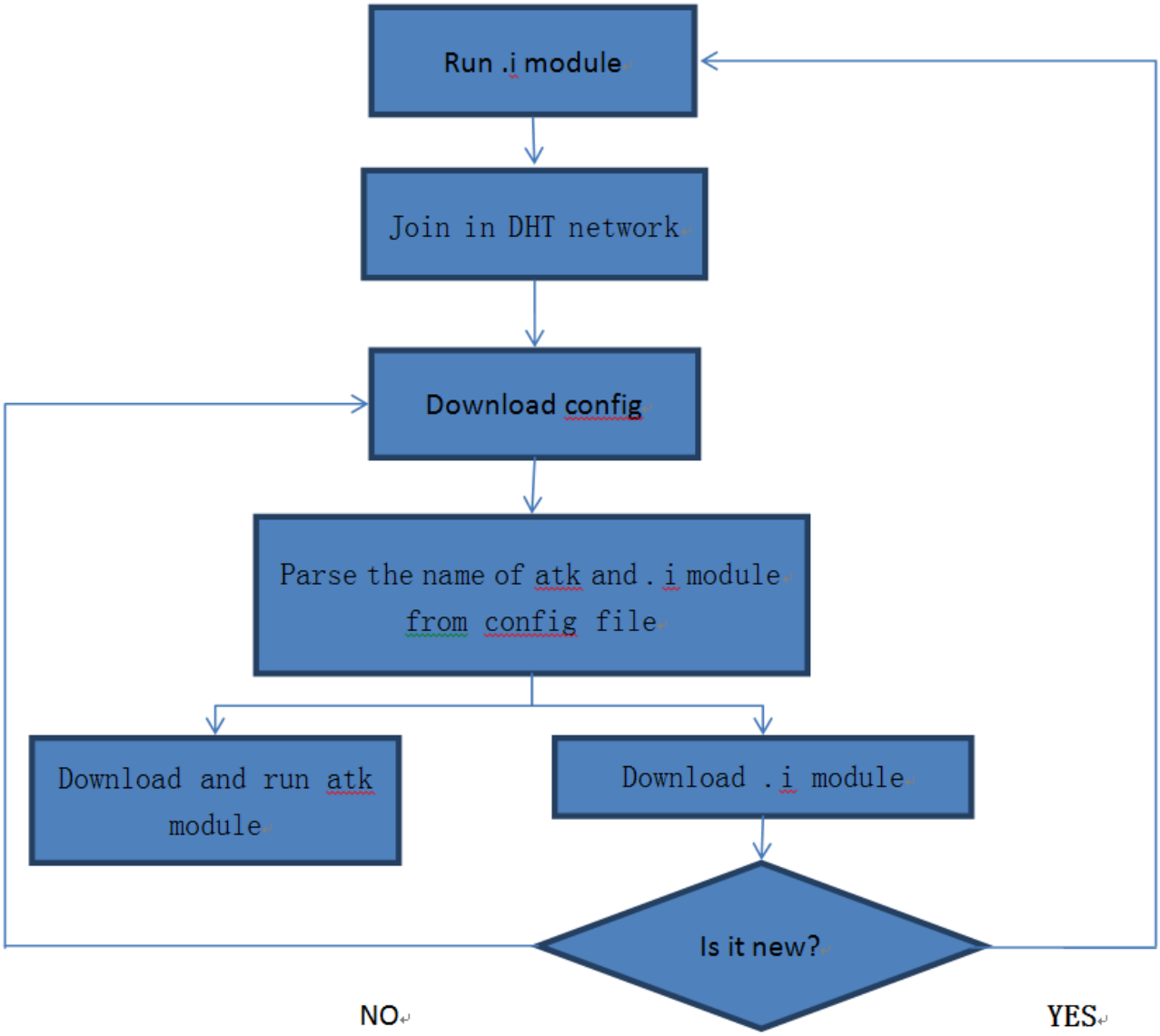
Our team was able to figure out the key exchange scheme of this botnet so we are able to participate in the Hajime network with full control of the bot behaviors, with that, we have been able to gain more insights. The visibility of this botnet is fairly poor in security research field, with the two observations above, we decide to talk about this botnet a little bit and also start a public accessible Hajime tracking project here

How Hajime works

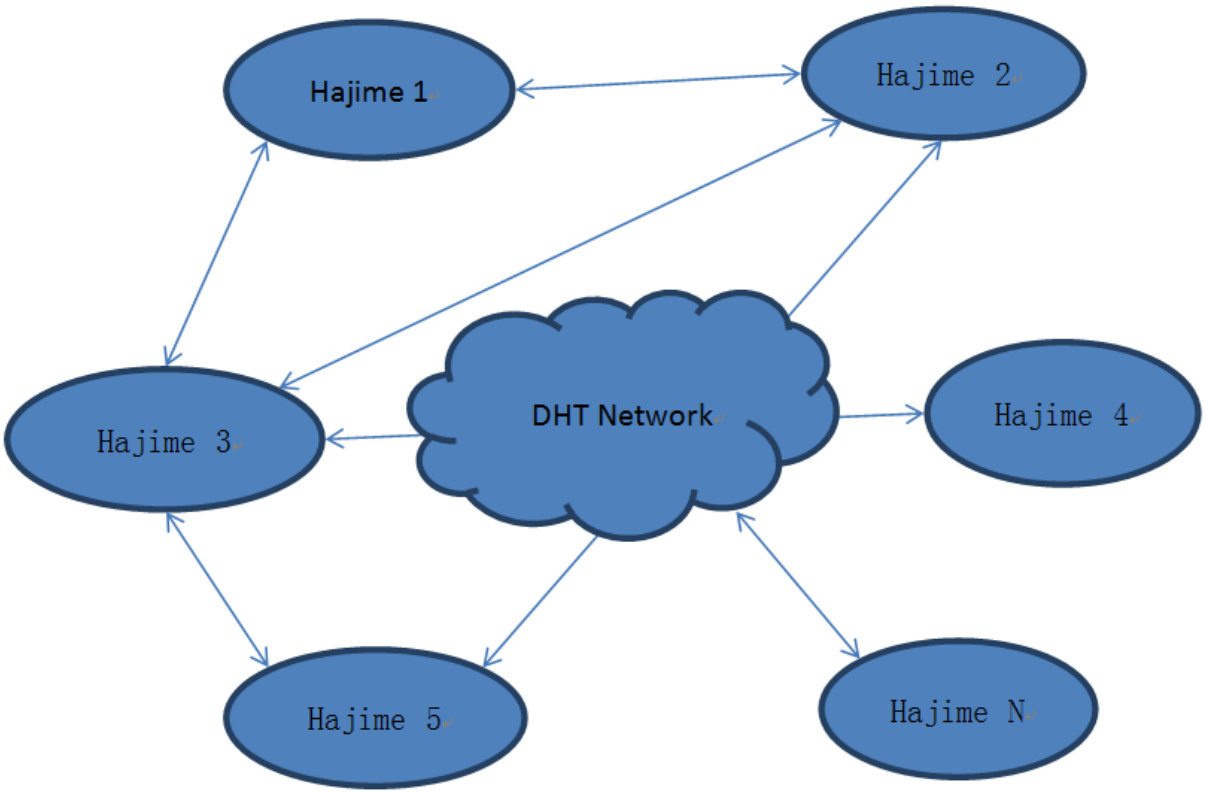
For some basic concepts of Hajime botnet, we suggest readers to take a good look at the Rapiditynetworks paper here so we don't need to repeat known facts here.

Overall framework

Hajime's implementation follows the modular programming paradigm. There are two common executable modules: "execution module" (also known as stage2 or .i module) and "propagation module" (also known as atk module).



In the propagation module (.i module), the Hajime nodes will establish a P2P network based on the DHT protocol. On top of this P2P network, Hajime nodes can perform inter-node negotiation, control command transmission and file synchronization. Any node can obtain the control instructions from the administrator without direct communication to the administrator. This not only provides protection for the administrator, but also maintains the stability of the botnet itself. Once the network is set up, it is extremely difficult to take it down.



In the DHT network, Hajime encodes the config file as a special infohash value. This is a 160 bit binary number, changed daily. By indexing the infohash value, Hajime can get the latest config file.

```
[modules]
atk.arm6.1501950328
.i.mipseb.1501955178
atk.mipseb.1502568282
.i.mipsel.1501955168
.i.arm5.1501967080
atk.mipsel.1502568291
atk.arm5.1501967394
.i.arm6.1501897866
atk.arm7.1502046207
.i.arm7.1501901744
[peers]
router.utorrent.com
router.bittorrent.com
[info]
    [32mJust a white hat, securing some systems.
Important messages will be signed like this!
Hajime Author.    [0m
    [1m    [31mContact CLOSED
Stay sharp!
    [0m

```

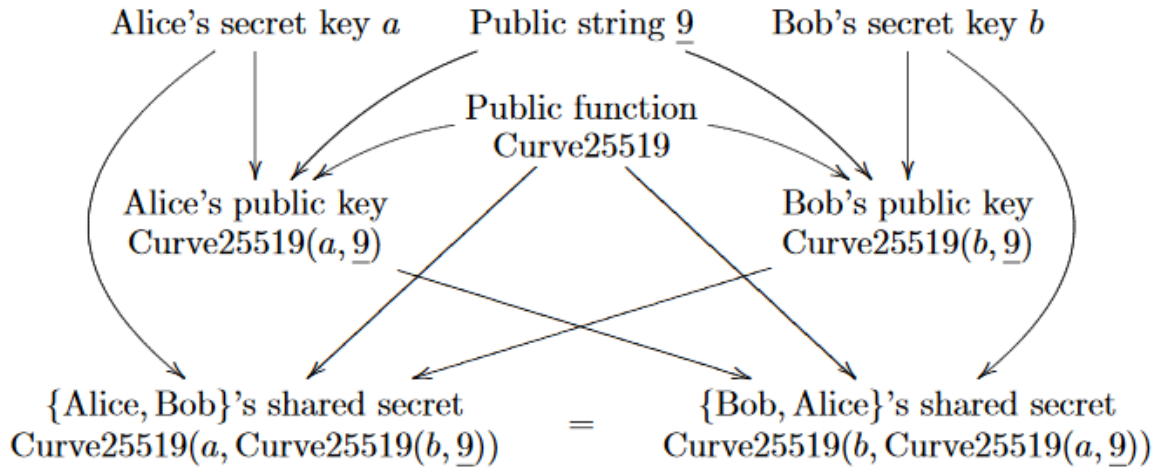
The above figure shows a config file in August 12, 2017. You can see it is a text file. The modules field contains the module file name for different CPU architectures. The peers field specifies the entry domain name for the DHT network. The two domain names are both valid and publicly accessible. The configuration file will direct Hajime to synchronize to the latest "propagation module (atk module)" or “execution module (.i module)".

File synchronization

In a DHT network, each Hajime node can correspond to a peer. By searching in the DHT network, the Hajime node can easily get the address information of other Hajime nodes. During file synchronization, it will traverse other nodes sequentially and perform synchronization on each node. The inter-node communication relies on the uTP protocol. This protocol implements mechanisms such as three-way handshake, session retransmission and session interruption based on UDP to ensure trusted communication between Hajime nodes.

Key exchange

On the channel provided by uTP, Hajime made the RC4 communication encryption for each session to ensure that others could not restore the content by packet capture. Besides, Hajime uses a key negotiation algorithm to ensure that the RC4 key is not stealable.



Hajime uses ECDH as the key exchange protocol and selects a key exchange algorithm based on Curve25519 elliptic curve. Although there are many open ECDH implementations on the Internet, the author of Hajime did not use the existing code directly. He has implemented a more efficient key exchange process. This implementation features on projecting points from the original elliptical space to the new one-dimensional series. Besides, in the "double point" calculation process, it only calculates the X coordinates, without considering the Y coordinates. Therefore he can improve the computational efficiency.

Reference for Curve25519 can be found at: https://cr.yp.to/ecdh.html
The fundamentals for the new key exchange algorithm can be found at https://cr.yp.to/ecdh/curvezero-20060726.pdf

Document Signature

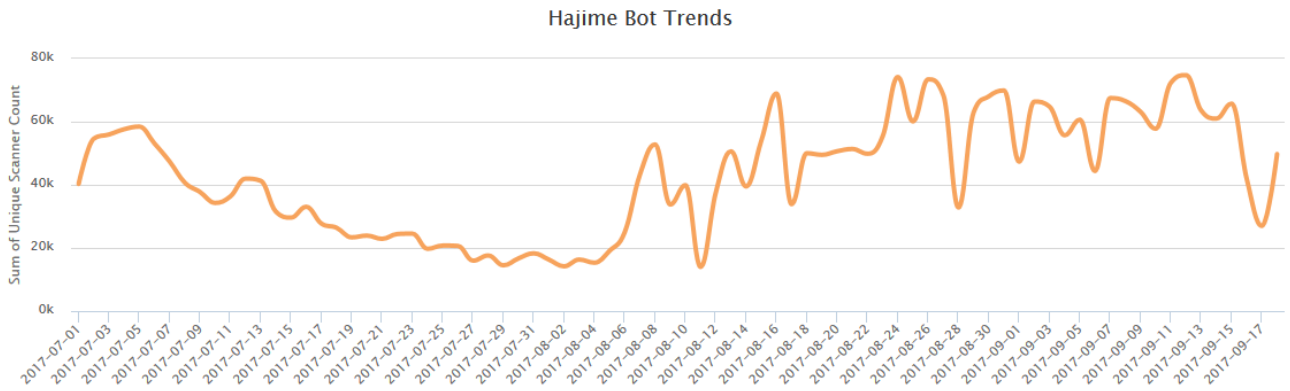
In P2P networks, nodes are untrustworthy, and anyone can fake a Hajime node at a very low cost. To ensure that the Hajime network is completely controllable and not stolen by others, Hajime nodes need to verify the signature of each synchronized file before acceptance and execution. Hajime adopts a public digital signature algorithm called ED25519 and uses `A55CEED41FECB3AC66B6515AB5D383791B00FEC166A590D7626A04C2466B3F54` as public key, which is integrated into each Hajime's execution module. Thus every Hajime nodes can verify the integrity of an new synchronized file.

Current status

The following diagram shows daily active hijime nodes since the beginning of July.

Daily active nodes

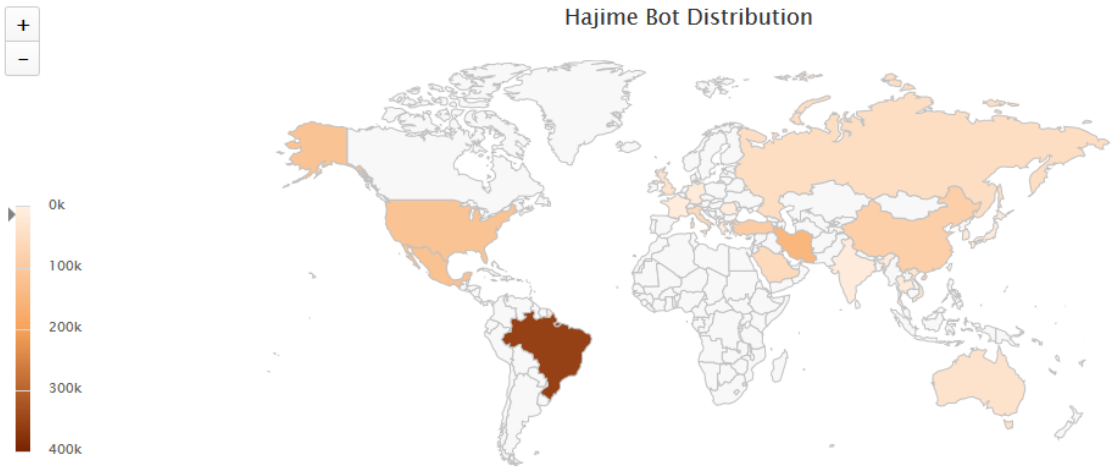
Below shows the number of daily active IPs:



It is not difficult to spot that in the second half of July, Hajime infection went down. Correspondently there were no botnet file update during that period(did the author go for vacation?) Starting from August, the author started to update the Hajime files, and the bot numbers started to going up, noticeably on 2017-08-06 and 2017-08-12, with the author pushing update, the botnet climbed up really quickly. Subsequently, Hajime completed several updates on 2017-08-18, 2017-08-19, 2017-08-22 and 2017-09-04. This opened the new normality of frequent updates.

Geographical distribution

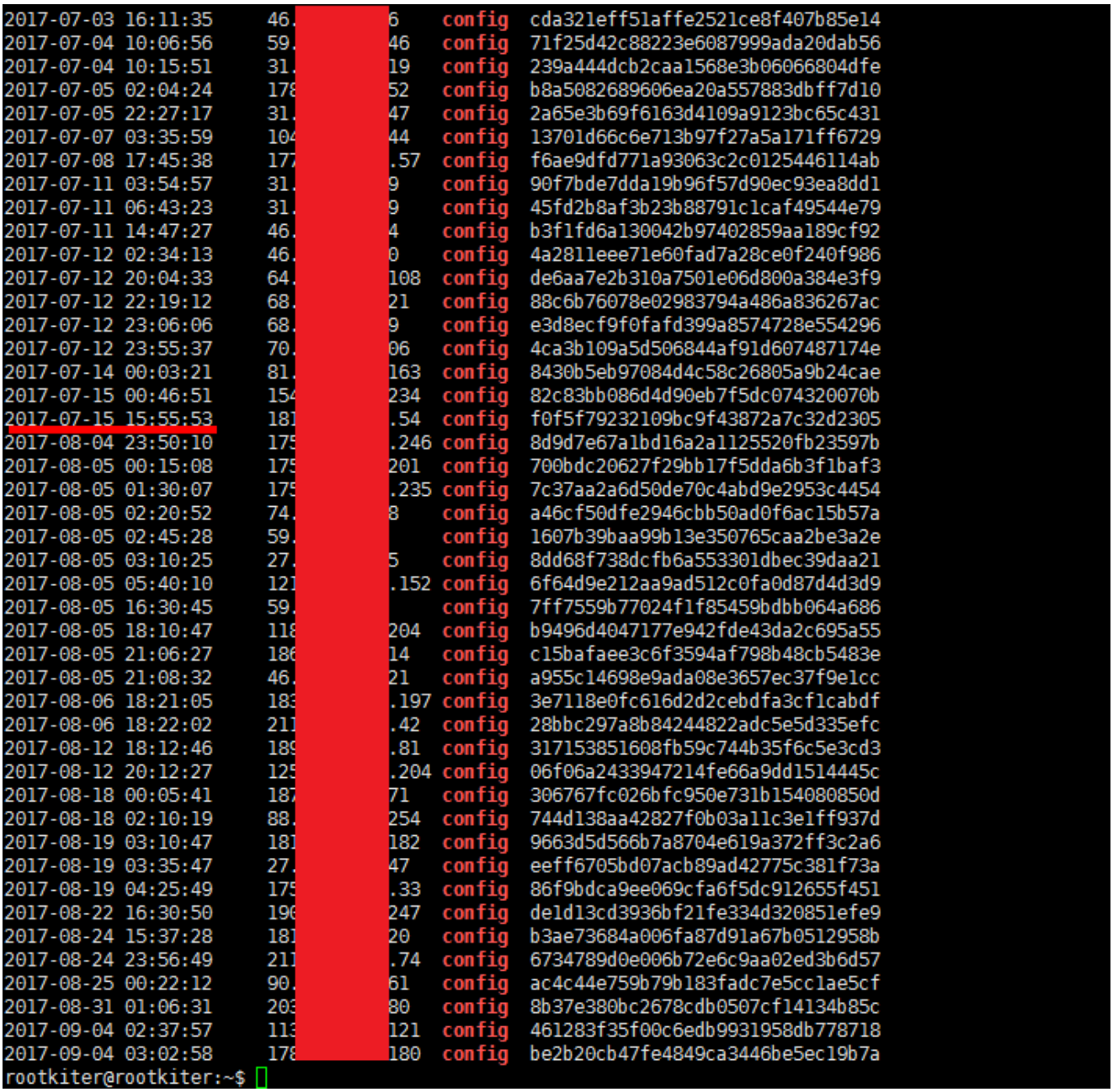
The following figure shows the geographical activity of Hajime in the past two months:



The deeper the color, the more serious the affected area.

Update frequency

Hajime was not updated frequently in July, and there was a two-week break. However, after August 5, Hajime began to regain activity:



In fact, all updates in the past month are patches on the existing code. There's no dramatic change.

Update on bot propagation

Originally,Hajime propagated mainly thought TR_069 vulnerability and weak telnet account, as time goes by, more vulnerabilities have been added, and the following is the newest vulnerability breakdown.

Port List	Protocol	Spreading Method
80/81/82 8080/8081/8082 9000/7547/5555	HTTP	1. TR-069 exploitation: First seen in MIRAI and later was used by Hajime as well. 2. Go-Ahead exploitation: First seen in HTTP81(also known as Persia) and was used by Hajime as well. 3. HI_SRDK_NET_SetPppoeAttr RCE exploitation: a unique method used by Hajime. This RCE vulnerability was exposed in 2013 and widely existed in DVR devices. The exploitation code was first seen in (2487b4ed4a2f55bfd743b2e6b98f8121) Hajime sample in 201701, but broke out at the end of 201705, which lead to a sharp rise of traffic on port 9000. This exploitation was only found in ARM5 Hajime sample, but not in other CPU architecture samples. Btw. A Japanese security researcher mentioned this on twitter, but did not get much attention: https://twitter.com/masafuminegishi/status/870182653797871617
5358/23	Telnet	Telnet user/password cracking, which was widely used by IoT botnet and was adopt by Hajime as well.

- `2017-01-17 atk.arm5.1481380646 2487b4ed4a2f55bfd743b2e6b98f8121`
- `2017-05-29 atk.arm5.1496096402 a238462e1e758792c5d1f04b82f4a6a0`
- `http://console-cowboys.blogspot.com/2013/01/swann-song-dvr-insecurity.html`
- `https://gist.github.com/ylluminate/fcee91965b58695460ce849c424488f7`
- `https://twitter.com/masafuminegishi/status/870182653797871617`

CPU Architecture Distribution of Hajime Node

The propagation of Hajime is limited among a small number of different CPUs. Our analysis on 18433 Hajime nodes shows that the Mipseb CPU is affected the most.

CPU	Hajime Node Count
Arm5	1506
Arm6	465
Arm7	2431
Mipsel	953
Mipseb	13078

More Observations on Hajime

Support for x64 platform?

Our tracking system captured a special config file `b8a5082689606ea20a557883dbff7d10` on 2017-08-29.
This config file can be verified successfully and contained settings for X64 CPU, which indicated that Hajime author is planning to support PC platforms.

The config file is shown as followed:

b8a5082689606ea20a557883dbff7d10

```
[modules]
atk.arm6.1480149376
.i.mipseb.1480592953
atk.mipseb.1480464620
atk.x64.1480589934
.i.mipsel.1480592455
.i.arm5.1480593905
atk.mipsel.1480464626
atk.arm5.1480464605
.i.arm6.1479760645
atk.arm7.1480464611
.i.arm7.1480592218
[peers]
router.utorrent.com
router.bittorrent.com

```

However there are some doubts in this config file:

- We can only get two modules (atk.mipseb/.i.mipseb) listed in the config file while other CPU architecture's modules are unavailable in Hajime network.
- The "info" field, which contains the author's whitehat declaration, is missing in this config file. This is quite abnormal according to author's previous habit.

```
filemd5             f221e5906293038c6ee8699b725db3ef
filename            config
isCompressed        0100
filetype            0000
timeStamp           584011f3
bodysize            000000e0
head_sig            5d3ad6aa42dadb14b15d971df2f183acfd099a087fe8ef8e2f2cc4cd1e2ae49dd24adf3ad0e2ddcd15a36643e8bd9bdf815d6268df145b08e1ef06d28ba4c008
head_verify         confirm
content_sig         2d27f533530b96995ffe5f807ee79585752ab85c1b72b6e9b98d3b793b9d24e6c3dc6c64bea7d3f3ac59139ab2864978126d2b21632f3bd798805426b5ad1b02
data_verify         confirm
decomSize           0000011b
comSize             000000d4
startHex            f1185b6d6f64756c65735d0a61746b2e
```

So we have two possibilities here:

- The author has intention to support x64 platform. As mentioned before, all Hajime files are signed by the private key and need to be verified before any Hajime node takes the update.

- The private key has been leaked, someone else is trying to poison the Hajime network.

We prefer the first scenario at this point, and it will be big change for Hajime itself if it is moving to the pc battleground. We will keep a close eye on what will unfold in the future.

1: https://x86.re/blog/hajime-a-follow-up/
2: https://github.com/Psychotropos/hajime_hashes
3: https://security.rapiditynetworks.com/publications/2016-10-16/hajime.pdf
4: https://securelist.com/hajime-the-mysterious-evolving-botnet/78160/