Pink, a botnet that competed with the vendor to control the massive infected devices

By 360Netlab

Published: 2021-10-29 · Archived: 2026-04-05 21:10:51 UTC

Most of the following article was completed around early 2020, at that time the vendor was trying different ways to recover the massive amount of infected devices, we shared our findings with the vendor, as well as to CNCERT, and decided to not publish the blog while the vendor's working was in progress. Last week, CNCERT finally disclosed this botnet, and we have learned that the infected devices have “mostly” been taken care of, so here we are..

Overview

On November 21, 2019, we got an interesting new botnet sample from the security community, the sample contained a large number of function names starting with “pink”, and we named it pink botnet.

Pink is the largest botnet we have first hand observed in the last six years, during peak time, it had a total infection of over 1.6 million devices (96% are located in China) Pink targets mainly mips based fiber router, and has very strong and robust architecture, it uses a combination of third-party services, P2P and central C2s for its’ bots to controller communications, and has complete verification of the C2 communications, doing this ensures that the bot nodes will not be easily cut off or taken over Pink raced with the vendor to retain control over the infected devices, while vendor made repeated attempts to fix the problem, the bot master noticed the vendor’s action also in real time, and made multiple firmware updates on the fiber routers corresponsly.

Pink adopts DNS-Over-HTTPS protocol, which is also not typical.

Scale and Impact

We have infection numbers from three differentness sources, they are:

1. 2019-11-30 A trusted security partner in US informed us they saw 1,962,308 unique daily active IPs from this botnet hitting them.
2. 2020-01-02 CNCERT shared us the following note

> "From the data measured in multiple dimensions such as NetFlow data, active probing, and real-time monitoring, the number of Bot node IP addresses associated with this botnet exceeds 5 million. As home broadband IPs are dynamically assigned, the true size of the infected devices behind them cannot be accurately estimated, and it is presumed that the actual number of infected devices is in the millions. One of the main bases of the measurement is that the number of IPs connected to C2 in one minute was well over one million."

3. 2020-01-08 Our (360NetLab) assessment based on continuous network-wide probing gave us the the number of 1.65 million. The infected IPs are concentrated in China (96%), spanning 33 provinces across the country. Affected carriers involve China Unicom (>80%) and China Telecom (>15%).

Pink Architecture

Pink is a hybrid architecture botnet that uses both "P2P" and central “C2" to communicate to its bots . In general, it delivers less time-sensitive commands (e.g. management configuration information) via P2P, while more time-sensitive commands are distributed centrally via the C2s (e.g. launching ddos attacks, inserting advertisements into HTTP websites visited by users).

Configuration information

One key step for every Bot is to find the controller. The controller information is contained in the "configuration", and the following is the latest configuration we intercepted.

```
 {
  "verify": "1585065675",
  "cncip1": "144.202.109.110",
  "cncport1": "32876",
  "dlc": "5b62596bc1453d51cc7241086464f294",
  "dl": "http【:】//155.138.140.245/dlist.txt",
  "dlc1": "484417e6f65c8e18e684d60c03c4680a",
  "dl1": "https【:】//*****.com/johncase/zip/raw/master/dlist.txt",
  "sd0": "1.1.1.1",
  "sdp0": "443",
  "srvk": "FJAz37XiKgnTLtVpmhjxZcavTJUU5r4XN3Wl5nhTpg0=",
  "pxy": "1"
}
```

1. The verify field is the timestamp when the command is issued, and the Bot will get the latest valid commands based on this timestamp.

2. The subsequent cncip and cncport fields specify the latest C2 address of the botnet, and the attacker can switch this control address easily at any time.

3. The subsequent "dlc/dl" and "dlc1/dl1" field groups are the latest Bot update addresses, where dlc and dlc1 are the corresponding content Hash check fields, and the algorithm pseudo-code is: `MD5(MD5(dlist_content)+SHA256(dlist_content))` .

4. The "sd0/sdp0" field is the secure DNS server address. For each Bot, when a DNS resolution record needs to be queried, it will check the DNS service specified here using `DNS-Over-HTTPS` .

5. The srvk field is the public key content (base64 encoding) of the server side. For each bot, its communication with CNC is encrypted. So a unique private key has to be obtained through ECDH key

negotiation before the actual communication. After specifying the public key on the CNC side here, it also completes the verification of the Bot's identity to the CNC by the way. This is an extension of the original ECDH.

6. The pxy field is presumably an option for the proxy usage. There is no sign of its being used, and it is not clear how the logic works.

Protection of configuration information

After reading the previous section, it is easy to see that the "configuration information" is actually the core of the botnet, which guarantees the attacker’s absolute control over the botnet.

To prevent others from discovering the configuration information, the configuration is encrypted. The decryption algorithm is symmetric, and the decryption code logic is as follows.

```
def easydecrpyt(message):
   res = ""
   for cursor in range(0, len(message)):
       mbt = ord(message[cursor])
       res += (chr((mbt ^ (cursor%0xff) ^ 0xae ^ 0xac ^ 0xbe ^ 0xef) & 0xff))
   return res
```

The bot master also signed the configuration information using ecdsa to make sure no one can mess with the message, with the following signature details.

1. The cryptographic library used for signature verification is `mbedtls`.
2. The signature algorithm is `ECDSA`.
3. The curve used for the signature is: `MBEDTLS_ECP_DP_SECP192R1`.
4. The public key used for signature checking is: `04 8D 54 71 71 44 A0 61 DA 5A B4 EA 40 55 2F 21 B1 9B 6C A5 17 92 0F 10 B5 11 56 ED 14 DB 54 47 1A 94 48 06 06 3C 7A B4 3B 25 D1 AC 9F 85 AF 33 9E`

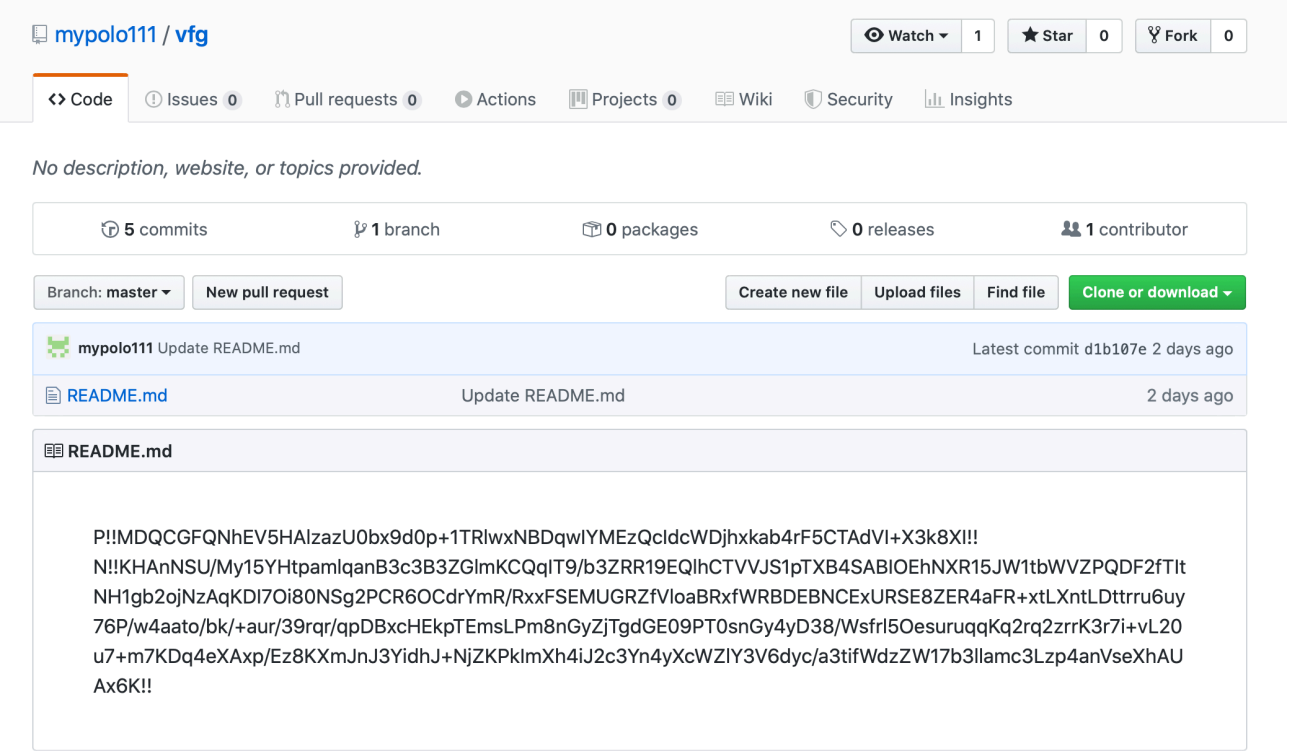
Distribution of Configuration Information

In addition to ensuring the confidentiality and integrity of the configuration information, the bot master also uses several means to distribute the configuration information to ensure its availability.

a) Distribution of configuration information through third-party services

1. Distribution of configuration information via BTC+GITHUB

The core of this distribution channel is a project hidden on GITHUB, the most recently seen project is (mypolo111/vfg), and you can see two lines in the README of this project.



Where the pattern is `P!!! <base64>I!!!` line is the configuration signature and the schema is `N!!! <base64>K!!!` line is the ciphertext of the configuration information.

However, for each Bot, trying to find this hidden item is a multiple steps process:

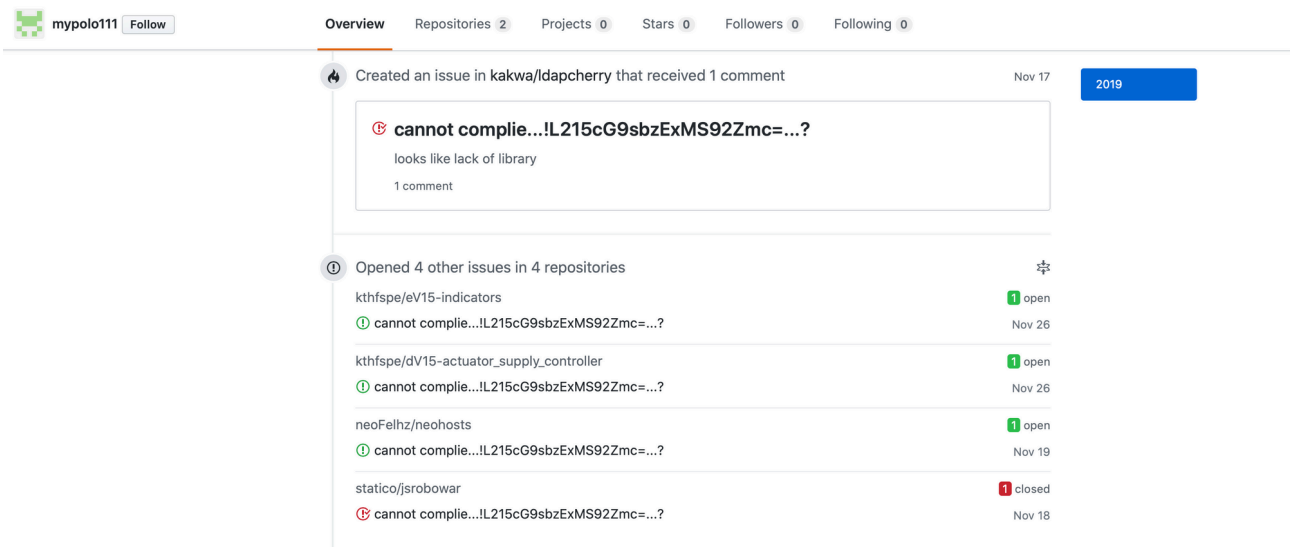
Step 1: A topic tag was first generated from the transfer record of a fixed BTC wallet [1GQNam6xhzYVLWWXvRfu3EjsFon6n6GxMF], (after reversing the sample, I reverted the process as shown below, the query of the BTC wallet uses four web services, and the specific addresses are also shown in the figure).

```python
import sys

def genericTopic(hashstr):
    tablelist = "abcdefghijklmnopqrstuvwxyz0123456789"
    cursor1  = ord(hashstr[0])
    cursor1 += ord(hashstr[3])
    cursor1 += ord(hashstr[6])
    cursor1 += ord(hashstr[9])
    cursor1 += ord(hashstr[0xC])
    cursor1 += ord(hashstr[0xF])
    char1 = tablelist[cursor1 % 0x1A]

    cursor2  = ord(hashstr[1])
    cursor2 += ord(hashstr[4])
    cursor2 += ord(hashstr[7])
    cursor2 += ord(hashstr[0xA])
    cursor2 += ord(hashstr[0xD])
    cursor2 += ord(hashstr[0x10])
    char2 = tablelist[cursor2 % 0x1A]
    topicUrl = "https://github.com/topic/%c%c" % (char1,char2)
    return topicUrl

if __name__=='__main__':
    hashlists = [
            # 1GQNam6xhzYVLWWXvRfu3EjsFon6n6GxMF -> %s
            # https://blockchain.info/rawaddr/%s?limit=2               | jq | grep "\"hash\""
            # https://api-r.bitcoinchain.com/v1/address/txs/%s?limit=2 | jq | grep "\"self_hash\""
            # https://chain.api.btc.com/v3/address/%s                  | jq | grep "\"last_tx\"
            # https://api.blockcypher.com/v1/btc/main/addrs/%s?limit=2 | jq | grep "\"tx_hash\""
            "a221570e2c6219604dc3a50e5f5ebdf63d80bf8e659ae198500b0ce8df4080a6",
            # -> https://github.com/topic/ad
            "5a6169e66a8c42c721952a55816cc4a89ffcfeae7628c7e150ff12048a403c35",
            # -> https://github.com/topic/rx
    ]

    for hashstr in hashlists:
        print genericTopic(hashstr)

```

Step 2: For each of the possible topic tags, there are many related github projects. Going through the ISSUES of these projects and looking for a file with the format `...!<base64>...?` String,when a match is located, restoring the base64 will give you the address (for example, the most recent one we found is `...!L215cG9sbzExMS92Zmc=...?` By restoring this base64, the address of the hidden item is `mypolo111/vfg`)

Step 3: Searching GITHUB Topics and ISSUES to find the hidden GIT project.

PS: `mypolo111` has submitted ISSUES in several projects, the relevant screenshots are shown below.



Important note here: With these set of addressing logic above, the bot master can easily switch the real GITHUB project address by adding transaction records to a specific BTC wallet. In such a case, the specified BTC wallet must be blocked in order to disrupt the GITHUB-based distribution logic of the botnet, or else the defender would just be playing whack a mole.

2. Distribution of configuration information through a Chinese website

In a few samples, the bot master also distributed configuration information via a website in China, using similar logic to that of GITHUB distribution.

b) Distribution of configuration information via P2P

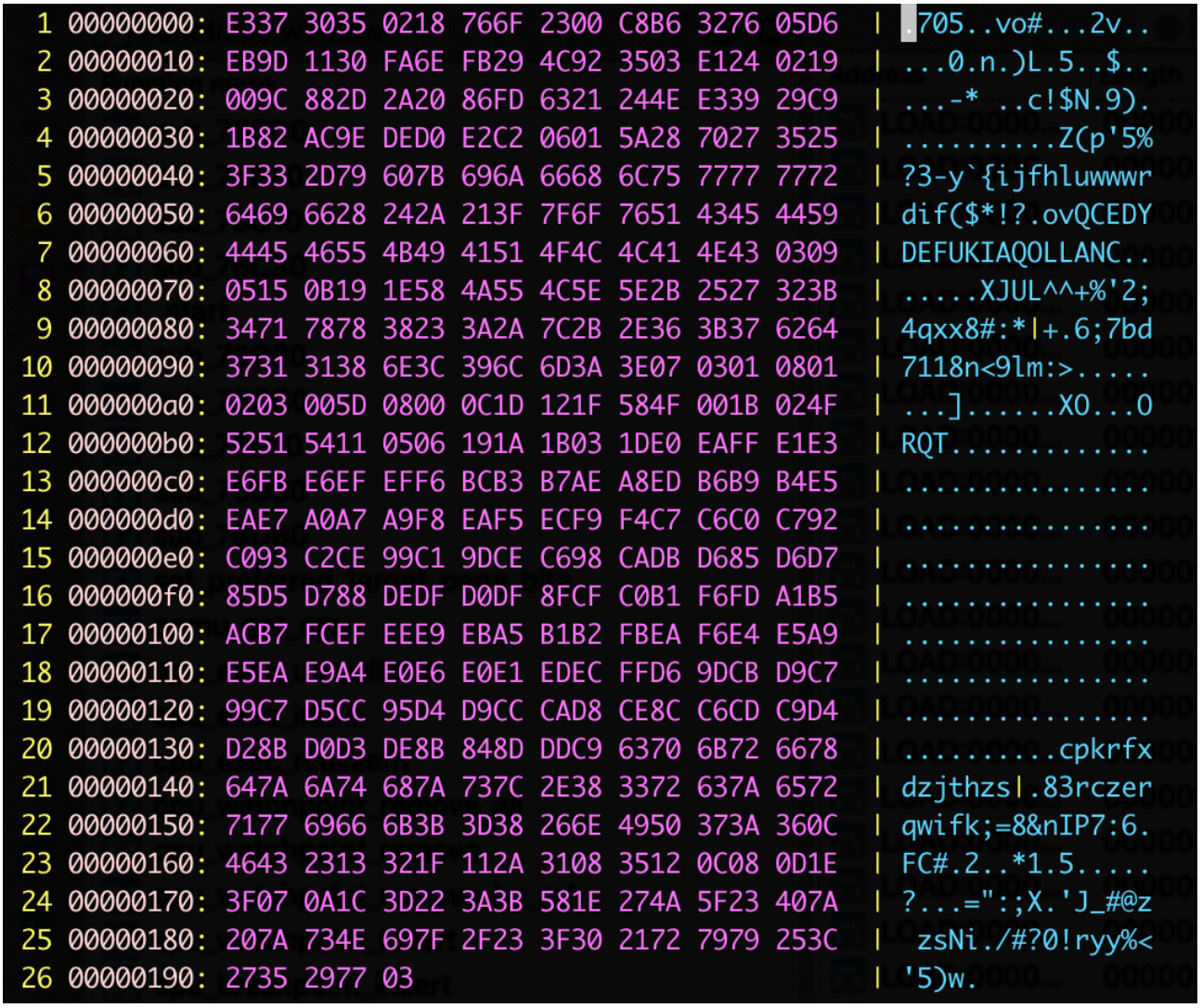
Two methods are supported here:

1. P2P-Over-UDP123 distribution

For this method, when the bot starts, it will listen to the UDP-123 port, which is originally the default port of NTP service, doing this might trick some users to think this is just a normal NTP service and nothing needs to be looked here. Then it will launch a Peer probe request to four B-segment addresses on the public network (`"114.25.0.0/16"` , `"36.227.0.0/16"` , `"59.115.0.0/16"` , `"1.224.0.0/16"`) with the content `1C 00 00 00` .

- When the target is a real NTP server it will respond with NTP time, while if the target is a bot node, two possible responses are expected.
- If the target Bot does not have the c2 information it will respond with `1D 00 00 00`
- When the target Bot has already gotten the C2 information, it replies with the signature of the C2 information and the corresponding cipher text, and before sending, 0xE3 will be added to the message header.

The following figure shows the recently captured configuration information via UDP-123.



c) Distribution of configuration information via C2

The attacker has a domain name cnc.pinklander.com built into some of the samples. When the domain name is enabled, a web page is displayed with the same content as the GITHUB project. It is also the base64 encoded configuration information.

PinkBot Command

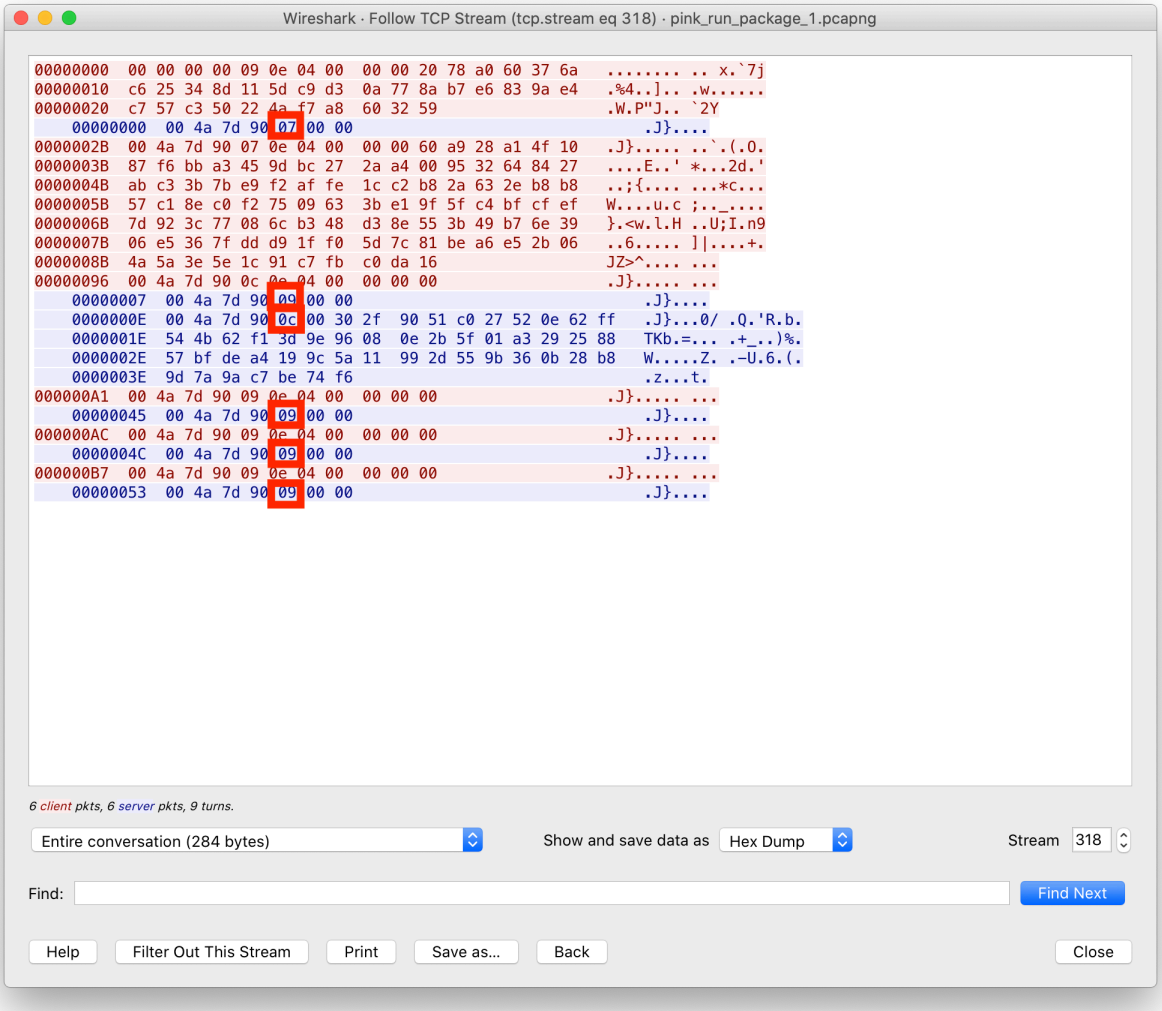
Command Format

Each command contains at least 7 bytes, and the meaning is as follows.

1. Token field, 4 bytes long, the value of this field is specified by the server and will be used all the time after it is specified. When the server side accepts it, it will assign a Token value to the Bot, which is considered successful.
2. Command field, length 1 byte, after C2 sends out the command, Bot should return the execution result with the same command code.

3. Content length field, 2 bytes in length. When the instruction does not contain specific content, it is set to zero, otherwise it is filled with the number of bytes of content length and appended with the cipher content.
4. Instruction. When the instruction contains content, this field is filled with the instruction content of the cipher text. See below for decryption method.

Here is a screenshot for reference, where the instruction field is marked in red.



Instruction communications

a) Encryption

The cncip1 and cncport1 in the above configuration information are the actual C2 nodes used by the attacker. The communication details are as follows.

1. The cryptographic library used is: `mbedtls`;
2. The exchange algorithm used in the key exchange phase is ecdh, and the loading curve is `MBEDTLS_ECP_DP_CURVE25519`;

3. The server-side ECDH public key is hard-coded in the samples in the early stages of development , and later on, it is changed to be specified in the configuration information. However, the content has not changed so far: `14 90 33 DF B5 E2 2A 09 D3 2E D5 69 9A 18 F1 65 C6 AF 4C 95 14 E6 BE 17 37 75 A5 E6 78 53 A6 0D`
4. The algorithm used in the encryption/decryption phase of the message is AES, the key is the secret after key exchange, and the loading parameters are `MBEDTLS_AES_ENCRYPT` and `MBEDTLS_AES_DECRYPT`;
5. For ECDH, generally the public and private keys of both parties are regenerated every time. In Pink, however, only the Bot side is required to be different each time, while the server side specifies a fixed pair of public-private keys. This built-in server-side public key is equivalent to giving the Bot the ability to authenticate the CNC, thus eliminating the possibility of man-in-the-middle attacks.

b) Instruction Content

In order to adapt to the different distribution of byte sequences in mipsb/mipsl models at the same time, the transmitted content is transformed by the open source library nanopb, which can abstract the serialization and deserialization process by agreeing on a template, thereby ignoring the interference of big/little endian memory.

The instructions

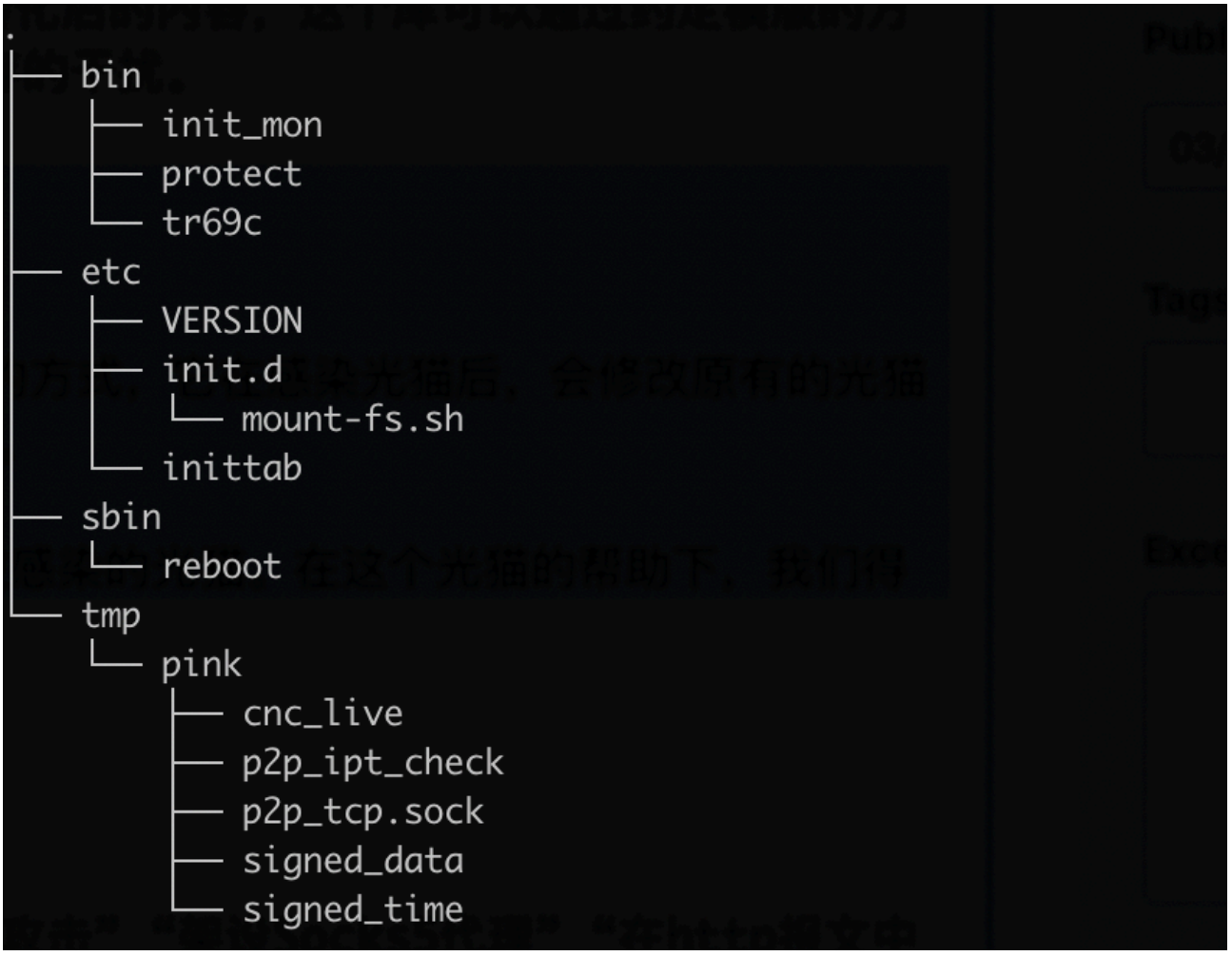
The Pink instruction has a rich set of controls.

> 1. File download
> 2. System command execution
> 3. DDoS attacks (HTTP attacks and UDP attacks)
> 4. Scan(the specifics of the scan can be set by the command)
> 5. Report device information (CPU / system type / memory information / system version / hardware information)
> 6. Self-update (save new version to /tmp/client and then run)
> 7. P2P node list synchronization (push a set of P2P nodes directly to the Bot)
> 8. Http message injection (on the victim device, advertising js scripts will be injected when traffic type is http)
> 9. Sock5 proxy service (set up Socks5 proxy service on Bot side, account password set by command)
> 10. Download the file and execute
> 11. Stop the attack
> 12. Reset watchdog

PinkBot persistence method

Unlike other the botnets we commonly see, Pink will flash the original firmware of the fiber router after infecting it in order to maintain absolute controls. In the rewritten firmware, PinkBot's downloader c2 and the supporting bootloader are included.

The following figure shows the list of files added/modified by Pink.



The tmp directory can be ignored, as it contains temporary files generated when the sample runs.

Key file descriptions.

- /bin/protect: `md5:9ec5bd857b998e60663e88a70480b828`

The protect file is actually more like a downloader, in the sample you can see codes that support above-mentioned 5 methods to get configuration information. The main function is to get the latest samples from the configuration information and start them up.

- /bin/tr69c: `md5:451a3cf94191c64b5cd1be1a80be7799`

The tc69c file is a patch version of tr69c of the original firmware of the fiber router. It removes the update function from the firmware, which basically makes it impossible to update the firmware through tr69c.

Command Tracking

We simulated PinkBot node to receive commands distributed by the C2 in real time. In addition to the daily maintenance-type instructions (heartbeat instructions/peerlist synchronization instructions), we also received several instructions to insert advertisements to WEB pages, such as:

```
 <script async src="http【:】//45.32.21.251/j/?$$"></script>
<script async src="http【:】//167.179.80.159/j/?$$"></script>
<script async src="http【:】//114.55.124.13/j/?$$"></script>
```

We have noticed that the configuration information for pink changes intermittently, mainly in the CNC and DL* fields, and it has stabilized in recent months. Below is the latest configuration information we captured on 2021/10/5 (BST).

```
{
"verify":"1611936001",
"cncip1":"140.82.40.29",
"cncport1":"26007",
"dlc":"450aa79da035a8a55ca4c0e6b1025b50",
"dl":"http://209.250.247.60/dlist.txt",
"dlc1":"47ed94977b45099f1ef5c7701b2d25dc",
"dl1":"https://****.com/****/dlist.txt",
"sd0":"1.1.1.1",
"sdp0":"443",
"srvk":"FJAz37XiKgnTLtVpmhjxZcavTJUU5r4XN3Wl5nhTpg0=",
"pxy":"1"
}
```

Current Size

As mentioned earlier, Pink uses P2P to distribute non-real-time command information. Using this feature, we were able to evaluate the number of PinkBot infections on a global basis. The major vendor has been exploring all possible methods to eliminate infected devices and the infection number has dropped significantly. While at the same time, there are still a good amount of infected units out there, on 2021/10/20, we still see 103024 daily active IPs.

DDoS attacks

DDos does not seem to be a big thing with Pink, we have seen Pink launching around 100 DDos attacks all together, for example:

```
 2020/3/24 UDP-DDoS,  Victim 203.56.252.137:26999
2020/4/8  HTTP-DDoS, Victim 180.101.192.199:27020
```

In the process of analyzing and tracking the Pink botnet. We noticed that the bot master and vendor have conducted several offensive and defensive cyber wars.

Round 0: According to information provided by one major vendor involved, the confrontation first occurred in mid-November 2019. The vulnerability under attack originated from a TCP-17998 control service, which is an

interface provided to operators to the device. Due to a misconfiguration and implementation of the service, access was opened to the public network, through which the attacker gained control of the relevant fiber routers.

Round 1: After discovering this problem, the vendor started to try to fix its own device through the same vulnerability on the public network. However, it was soon discovered and immediately acted upon by the bot master, who shut down the TCP-17998's extranet access via iptables, preventing further fixes by the vendor using this method.

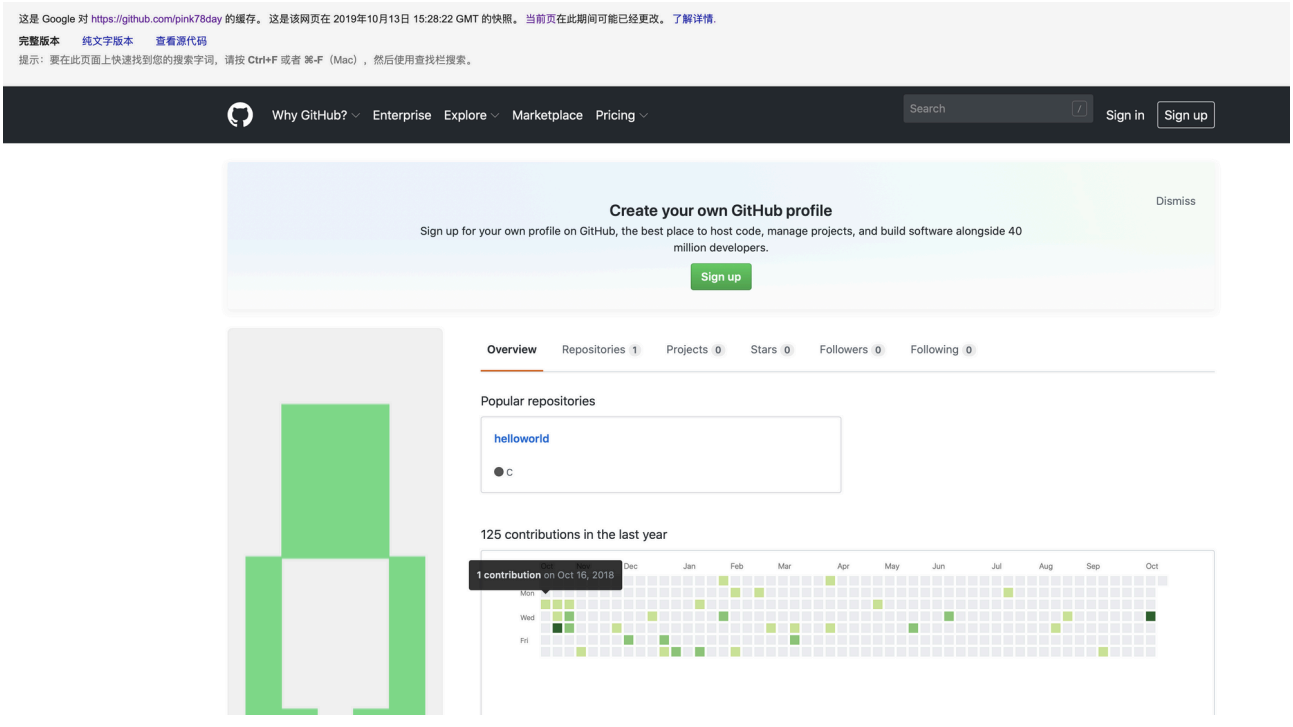
Round 2: This time the attack and defense focused on the tr069 upgrade channel. The vendor from the operator side can use tr096 to get in and repair the device when the unit boots up. However, the attacker had great visibility and quickly updated the firmware to turn off the tr096 update channel.

Round 3: Now the vendor tried to use the TCP-80 HTTP service on the LAN side of the device to repair the device, however, the attacker quickly noticed what was going on and updated the firmware again to take out the HTTP service file on the device. At this point, all devices have no control plane for the vendor to provide management service.

The finally: The only option left for the vendor to dispatch people to the end customers to physically access the fiber router, disassemble the debugging interface or simply replace the unit.

Our take: The bot master easily got an upper hand in multiple rounds of offensive and defensive confrontations, the vendor had no control and no visibility to where the devices are, how many of them are infected and what was going on on these devices and had no way to easy access the devices. On the other hand, the bot master had full visibility and full control over the infected devices and could issue whatever service command such as blocking ports, removing services through the centralized C2 in real time.

We traced back the Pink and found out that its first existence went back as early as October 16, 2018, at that time the Github account used was pink78day (this account is not exist now, seet Google snapshot here).



The account currently used by PinkBot is mypolo111 registered in late November 2019, and pink78day is no longer searchable on Github.

Note here as we mentioned before pink78day is not used as a centralized channel for distributing instructions, so simply blocking this account would have no real effect. The attacker can instantly redirect Bot to a new account by just adding a new transaction record to its BTC wallet.

IOC

C&C address

The attacker used the following C2 addresses.

```
 cnc.pinklander[.]com
144.202.109.110:40080
144.202.109.110:32876
207.148.70.25:12368
45.32.125.150:12368
45.32.125.188:12368
45.32.174.105:12368
5.45.79.32:12368
```

Synchronization Service

Pink synchronizes samples via HTTP services. Some of the HTTP services used are publicly services and some are temporally created.

The following URLs have been used for sample synchronization. These URLs were all extracted from PinkBot's configuration.

```
 http[:]//1.198.50.63:1088/dlist.txt
http[:]//1.63.19.10:19010/var/sss/dlist.txt
http[:]//104.207.142.132/dlist.txt
http[:]//108.61.158.59/dlist.txt
http[:]//111.61.248.32:1088/dlist.txt
http[:]//112.26.43.199:81/dlist.txt
http[:]//113.106.175.43:19010/tmp/pinkdown/dlist.txt
http[:]//117.131.10.102:1088/d/dlist.txt
http[:]//123.13.215.89:8005/d/dlist.txt
http[:]//125.74.208.220:81/dlist.txt
http[:]//140.82.24.94/dlist.txt
http[:]//140.82.30.245/d/dlist.txt
http[:]//140.82.53.129/dlist.txt
http[:]//144.202.38.129/dlist.txt
http[:]//149.28.142.167/p/dlist.txt
```

```
http[:]//149.28.142.167/p1/dlist.txt
http[:]//155.138.140.245/dlist.txt
http[:]//167.179.110.44/dlist.txt
http[:]//173.254.204.124:81/dlist.txt
http[:]//182.139.215.4:82/dlist.txt
http[:]//207.148.4.202/dlist.txt
http[:]//218.25.236.62:1987/d/dlist.txt
http[:]//218.25.236.62:1988/d/dlist.txt
http[:]//222.216.226.29:81/dlist.txt
http[:]//45.32.26.220/dlist.txt
http[:]//45.76.104.146/dlist.txt
http[:]//45.77.165.83/p1/dlist.txt
http[:]//45.77.198.232/p1/dlist.txt
http[:]//45.88.42.38/p1/dlist.txt
http[:]//61.149.204.230:81/dlist.txt
http[:]//66.42.114.73/dlist.txt
http[:]//66.42.67.148/dlist.txt
http[:]//8.6.193.191/dlist.txt
http[:]//95.179.238.22/dlist.txt
https[:]//***.com/**/dlist.txt
https[:]//raw.githubusercontent.com/pink78day/helloworld/master/dlist.txt
```

MD5

The relevant samples (ELFs).

```
 9ec5bd857b998e60663e88a70480b828 /bin/protect
451a3cf94191c64b5cd1be1a80be7799 /bin/tr69c
06d6ad872e97e47e55f5b2777f78c1ba slient_l
07cd100c7187e9f4c94b54ebc60c0965 slient_b
0f25b0d54d05e58f5900c61f219341d3 client_b
0f89e43ea433fdfd18a551f755473388 slient_l
1197994610b2ffb60edbb5ab0c125bc0 client_b
167364ad0d623d17332f09dbb23a980e client_b
175b603082599838d9760b2ab264da6f slient_l
1a6dce9916b9b6ae50c1457f5f1dfbbd slient_l
229503686c854bb39efdc84f05b071b9 slient_b
25a07e3ef483672b4160aa12d67f5201 client_l
262a4e242c9ebeba79aa018d8b38d229 client_l
29d0afd2a244c9941976ebf2f0f6597f client_l
2befedd020748ff6d9470afad41bd28c slient_b
2ca5810744173889b2440e4f25b39bd4 client_l
36e48e141943a67c6fdeaa84d7af21cc client_b
3a620ff356686b461e0e1a12535bea24 slient_l
41bbe8421c0a78067bae74832c375fe8 slient_l
45ee78d11db54acfdda27c19e44c3126 client_l
```

```
4830c3950957093dac27d4e87556721e slient_l
484761f281cb2e64d9db963a463efca5 client_l
48a7f2799bf452f10f960159f6a405d3 client_l
494412638dc8d573172c1991200e1399 client_l
4c83ad66189a7c4d2f2afdbfb94d0e65 slient_b
50270de8d5783bb0092bf1677b93c97b slient_l
54aa9e716567bd0159f4751916f7f0d1 client_l
5ae1fec20c2f720269c2dc94732187e8 slient_b
5b62a9bd3431c2fd55283380d81c00fa client_b
5c322610e1845d0be9ccfc8a8b6a4c4f client_l
5c4f8dae67dad8cac141afa00847b418 slient_b
5d0d034845bd69179bf678104c046dc1 client_b
60658ef214c960147200d432eece3e13 slient_l
60a2b1bb02a60ac49f7cc1b47abdf60c client_l
610f0aadba3be1467125607bf2ba2aaf slient_l
66a068fd860bda7950fde8673d1b5511 client_b
6c4de9bd490841f0a6c68638f7253c65 client_b
72c531a813b637af3ea56f288d65cdb7 slient_b
7608b24c8dcf3cd7253dbd5390df8b1f client_b
7645a30a92863041cf93a7d8a9bfba1a client_b
857fc3c7630859c20d35d47899b75699 slient_b
861af6b5a3fea01f2e95c90594c62e9d client_l
8e86be3be36094e0f5b1a6e954dbe7c2 client_l
8fbcd7397d451e87c60a0328efe8cd5d client_b
987a9befb715b6346e7ad0f6ac87201f slient_b
9eb147e3636a4bb35f0ee1540d639a1b slient_b
aa2fc46dd94cbf52aef5e66cdd066a40 client_l
ae8b519504afc52ee3aceef087647d36 slient_b
b0202f1e8bded9c451c734e3e7f4e5d8 slient_b
b6f91ad027ded41e2b1f5bea375c4a42 slient_b
b9935859b3682c5023d9bcb71ee2fece slient_b
b9d1c31f59c67289928e1bb7710ec0ba client_l
bec2f560b7c771d7066da0bee5f2e001 client_b
c2efa35b34f67a932a814fd4636dd7cb slient_l
c839aff2a2680fb5676f12531fecba3b slient_b
c94504531159b8614b95c62cca6c50c9 slient_l
dfe0c9d36062dd3797de403a777577a6 client_b
e19a1106030e306cc027d56f0827f5ce slient_l
f09b45daadc872f2ac3cc6c4fe9cff90 client_b
f5381892ea8bd7f5c5b4556b31fd4b26 client_b
f55ad7afbe637efdaf03d4f96e432d10 slient_b
f62d4921e3cb32e229258b4e4790b63a client_b
f81c8227b964ddc92910890effff179b slient_b
fc5b55e9c6a9ddef54a256cc6bda3804 client_b
fe8e830229bda85921877f606d75e96d slient_l
fee6f8d44275dcd2e4d7c28189c5f5be client_l
```

Readers are always welcomed to reach us on Twitter or email us to netlab at 360 dot cn.
