分布式拒绝服务攻击预警初探

马西兴　　金山云安珀实验室资深研究员

分布式拒绝服务攻击概念

FIT 2019

分布式 + 拒绝服务攻击 = DDoS

distributed + denial-of-service attack

访问卡顿

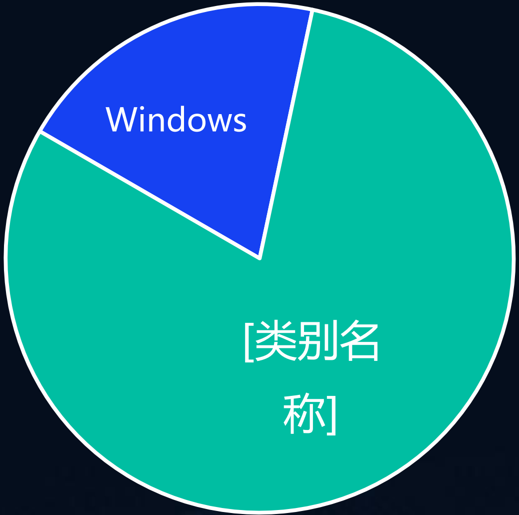
频繁掉线

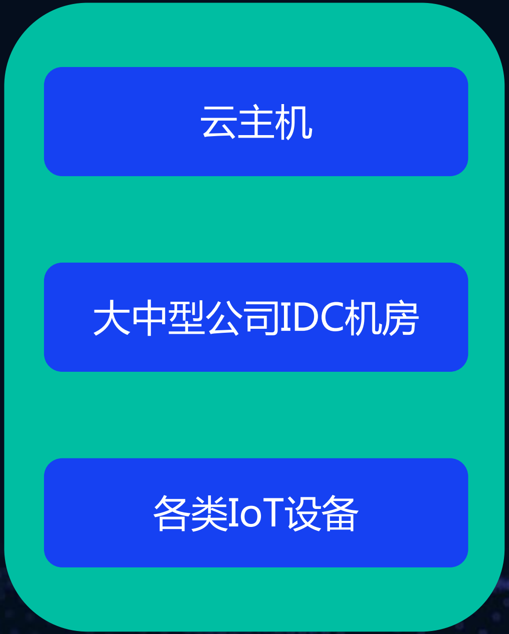
服务终止

声誉下降

肉鸡来源渠道分析及占比

Linux主机、 IoT设备占比超过80%





现阶段的防护手段

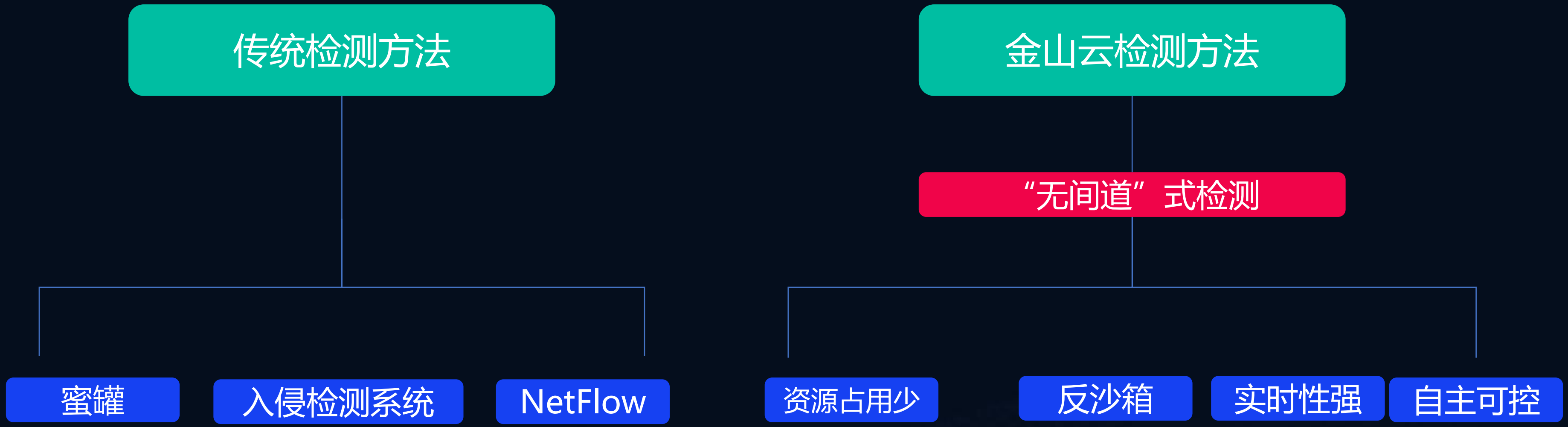
高防清洗

当流量突破报警阈值时，高防系统会自动对流量进行清洗

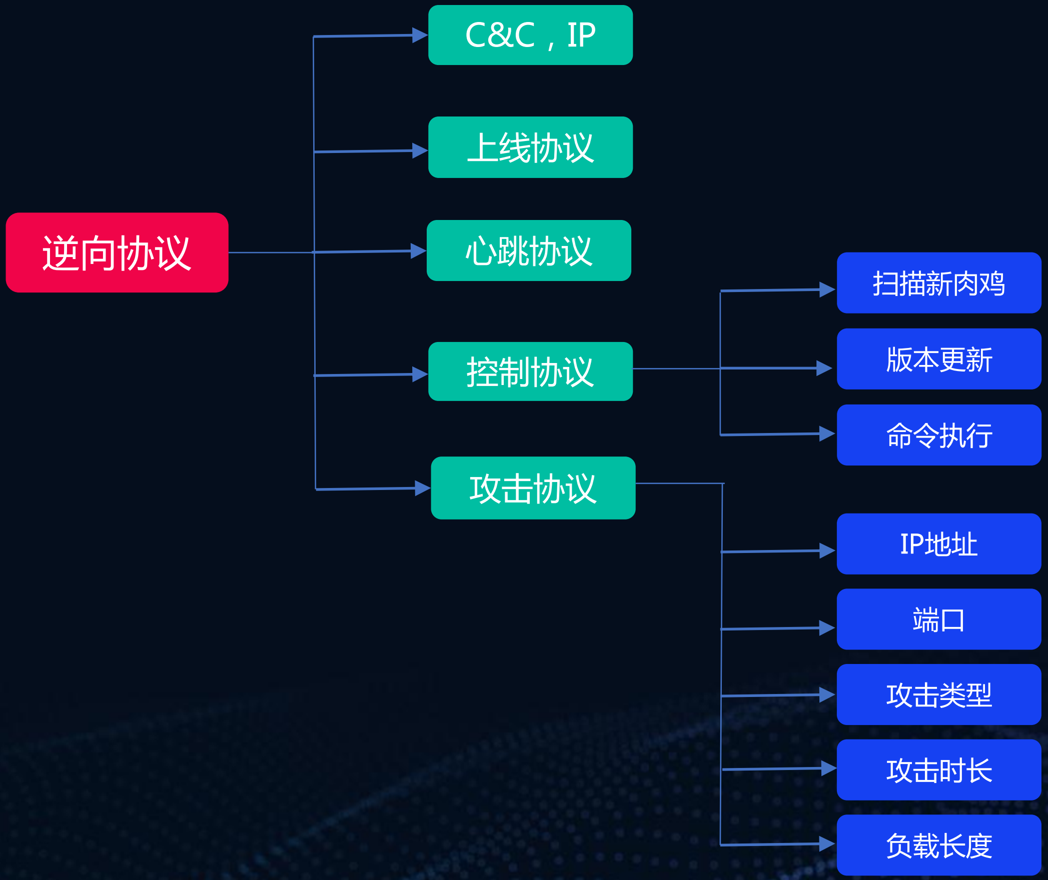
不足之处

当流量突破报警阈值时，肉鸡已经开始发动大规模攻击

预警检测思路

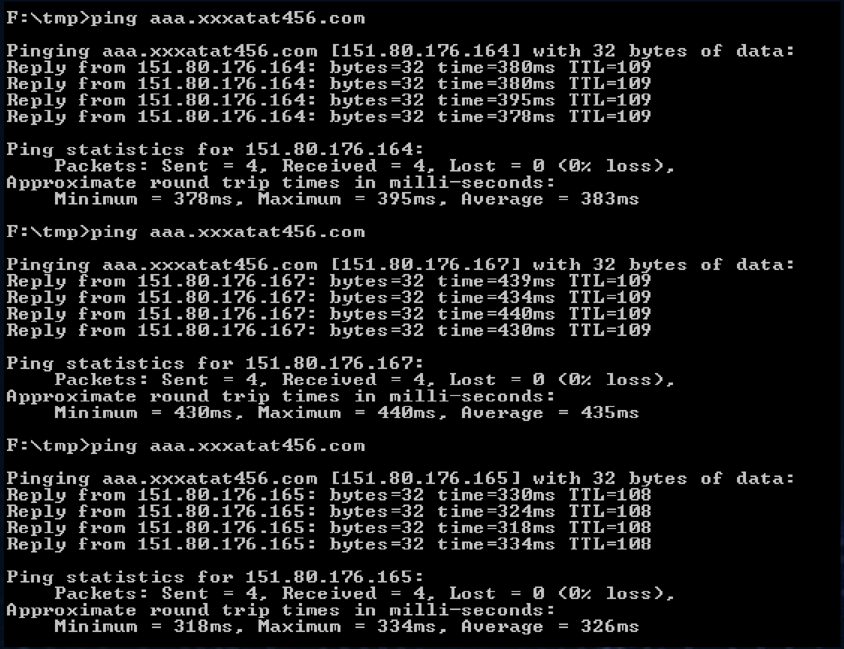

传统检测方法
金山云检测方法
“无间道”式检测
蜜罐
入侵检测系统
NetFlow
资源占用少
反沙箱
实时性强
自主可控

协议分析流程


逆向协议
C&C，IP
上线协议
心跳协议
控制协议
攻击协议
扫描新肉鸡
版本更新
命令执行
IP地址
端口
攻击类型
攻击时长
负载长度

连接 c&c

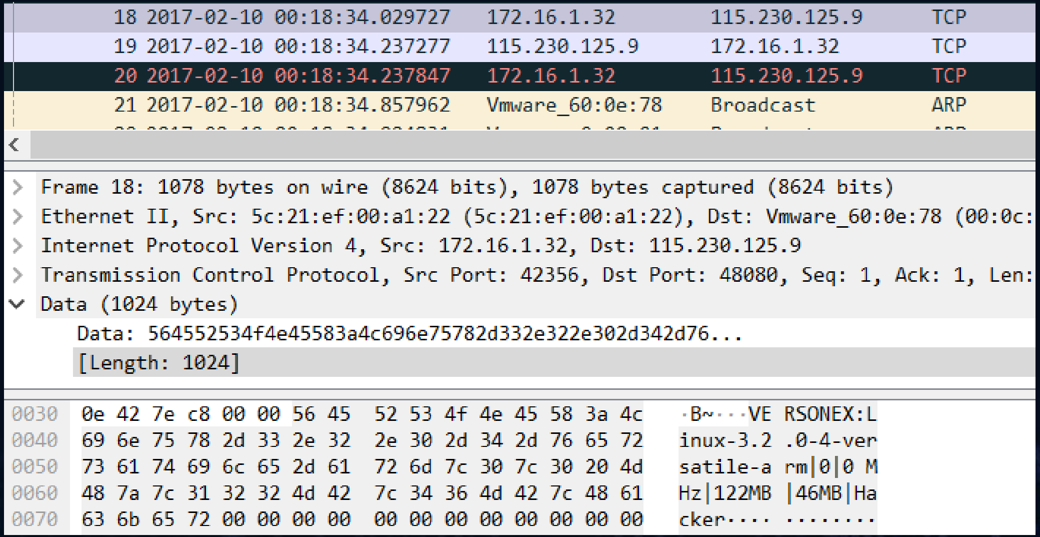
当存在多个不同的IP地址时，到底该连接哪个IP地址？



```
F:\tmp>ping aaa.xxxatat456.com

Pinging aaa.xxxatat456.com [151.80.176.164] with 32 bytes of data:
Reply from 151.80.176.164: bytes=32 time=380ms TTL=109
Reply from 151.80.176.164: bytes=32 time=380ms TTL=109
Reply from 151.80.176.164: bytes=32 time=395ms TTL=109
Reply from 151.80.176.164: bytes=32 time=378ms TTL=109

Ping statistics for 151.80.176.164:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 378ms, Maximum = 395ms, Average = 383ms

F:\tmp>ping aaa.xxxatat456.com

Pinging aaa.xxxatat456.com [151.80.176.167] with 32 bytes of data:
Reply from 151.80.176.167: bytes=32 time=439ms TTL=109
Reply from 151.80.176.167: bytes=32 time=434ms TTL=109
Reply from 151.80.176.167: bytes=32 time=440ms TTL=109
Reply from 151.80.176.167: bytes=32 time=430ms TTL=109

Ping statistics for 151.80.176.167:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 430ms, Maximum = 440ms, Average = 435ms

F:\tmp>ping aaa.xxxatat456.com

Pinging aaa.xxxatat456.com [151.80.176.165] with 32 bytes of data:
Reply from 151.80.176.165: bytes=32 time=330ms TTL=108
Reply from 151.80.176.165: bytes=32 time=324ms TTL=108
Reply from 151.80.176.165: bytes=32 time=318ms TTL=108
Reply from 151.80.176.165: bytes=32 time=334ms TTL=108

Ping statistics for 151.80.176.165:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 318ms, Maximum = 334ms, Average = 326ms
```

肉鸡上线协议

Linux.Flooder家族的上线包格式



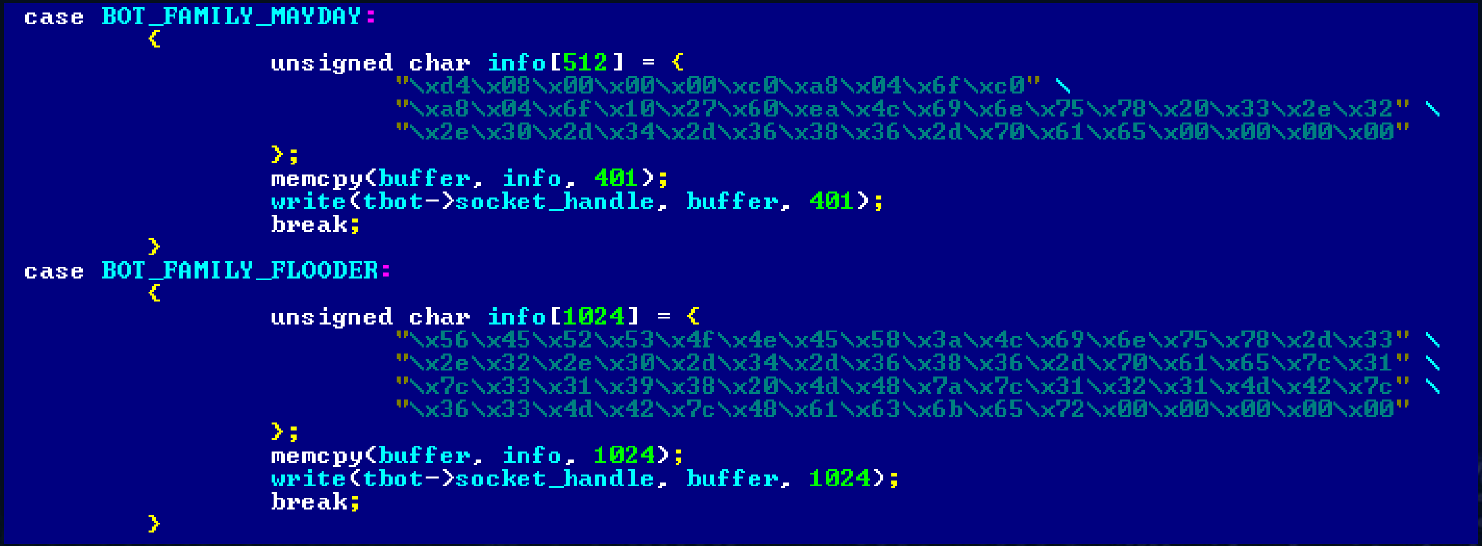
```
18 2017-02-10 00:18:34.029727   172.16.1.32     115.230.125.9   TCP
19 2017-02-10 00:18:34.237277   115.230.125.9   172.16.1.32     TCP
20 2017-02-10 00:18:34.237847   172.16.1.32     115.230.125.9   TCP
21 2017-02-10 00:18:34.857962   Vmware_60:0e:78 Broadcast       ARP
```

```
> Frame 18: 1078 bytes on wire (8624 bits), 1078 bytes captured (8624 bits)
> Ethernet II, Src: 5c:21:ef:00:a1:22 (5c:21:ef:00:a1:22), Dst: Vmware_60:0e:78 (00:0c:
> Internet Protocol Version 4, Src: 172.16.1.32, Dst: 115.230.125.9
> Transmission Control Protocol, Src Port: 42356, Dst Port: 48080, Seq: 1, Ack: 1, Len:
v Data (1024 bytes)
    Data: 564552534f4e45583a4c696e75782d332e322e302d342d76...
    [Length: 1024]
```

```
0030  0e 42 7e c8 00 00 56 45  52 53 4f 4e 45 58 3a 4c   ·B~···VE RSONEX:L
0040  69 6e 75 78 2d 33 2e 32  2e 30 2d 34 2d 76 65 72   inux-3.2 .0-4-ver
0050  73 61 74 69 6c 65 2d 61  72 6d 7c 30 7c 30 20 4d   satile-a rm|0|0 M
0060  48 7a 7c 31 32 32 4d 42  7c 34 36 4d 42 7c 48 61   Hz|122MB |46MB|Ha
0070  63 6b 65 72 00 00 00 00  00 00 00 00 00 00 00 00   cker···· ········
```

肉鸡上线代码

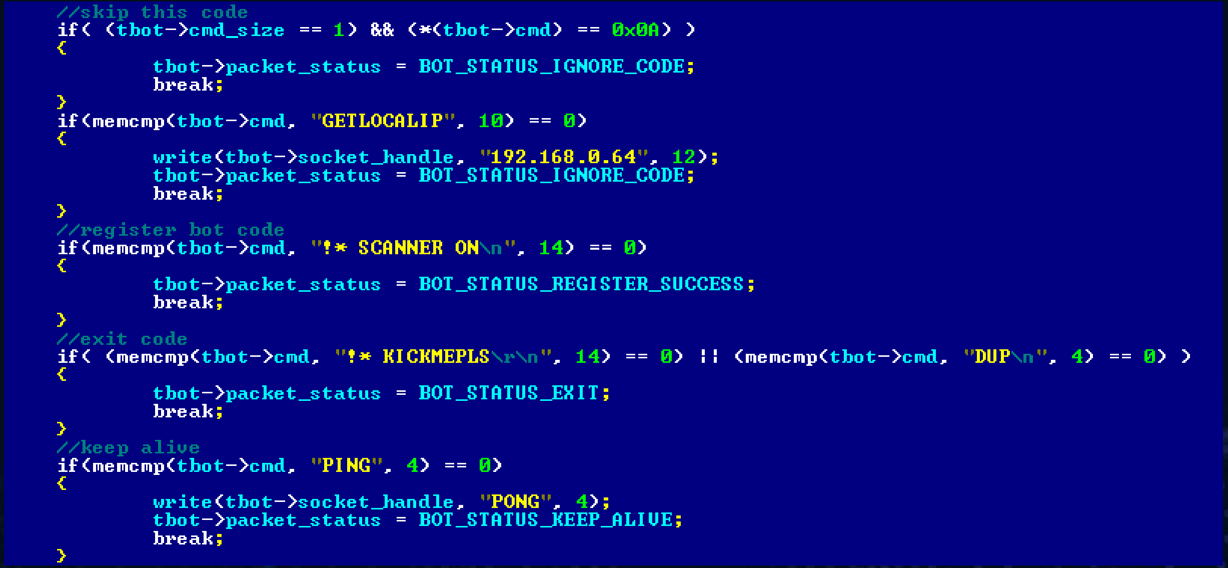
金山云上线代码



```
case BOT_FAMILY_MAYDAY:
        {
                unsigned char info[512] = {
                        "\xd4\x08\x00\x00\x00\xc0\xa8\x04\x6f\xc0" \
                        "\xa8\x04\x6f\x10\x27\x60\xea\x4c\x69\x6e\x75\x78\x20\x33\x2e\x32" \
                        "\x2e\x30\x2d\x34\x2d\x36\x38\x36\x2d\x70\x61\x65\x00\x00\x00\x00"
                };
                memcpy(buffer, info, 401);
                write(tbot->socket_handle, buffer, 401);
                break;
        }
case BOT_FAMILY_FLOODER:
        {
                unsigned char info[1024] = {
                        "\x56\x45\x52\x53\x4f\x4e\x45\x58\x3a\x4c\x69\x6e\x75\x78\x2d\x33" \
                        "\x2e\x32\x2e\x30\x2d\x34\x2d\x36\x38\x36\x2d\x70\x61\x65\x7c\x31" \
                        "\x7c\x33\x31\x39\x38\x20\x4d\x48\x7a\x7c\x31\x32\x31\x4d\x42\x7c" \
                        "\x36\x33\x4d\x42\x7c\x48\x61\x63\x6b\x65\x72\x00\x00\x00\x00\x00"
                };
                memcpy(buffer, info, 1024);
                write(tbot->socket_handle, buffer, 1024);
                break;
        }
```

命令控制和心跳协议

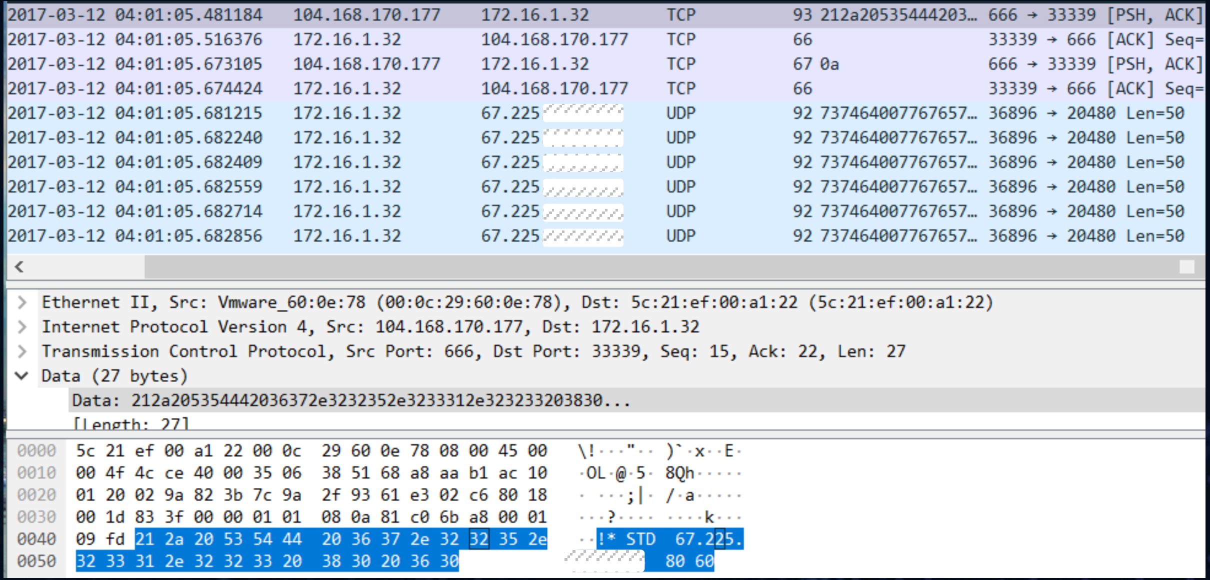
gafgyt家族控制协议响应代码



```
    //skip this code
    if( (tbot->cmd_size == 1) && (*(tbot->cmd) == 0x0A) )
    {
            tbot->packet_status = BOT_STATUS_IGNORE_CODE;
            break;
    }
    if(memcmp(tbot->cmd, "GETLOCALIP", 10) == 0)
    {
            write(tbot->socket_handle, "192.168.0.64", 12);
            tbot->packet_status = BOT_STATUS_IGNORE_CODE;
            break;
    }
    //register bot code
    if(memcmp(tbot->cmd, "!* SCANNER ON\n", 14) == 0)
    {
            tbot->packet_status = BOT_STATUS_REGISTER_SUCCESS;
            break;
    }
    //exit code
    if( (memcmp(tbot->cmd, "!* KICKMEPLS\r\n", 14) == 0) || (memcmp(tbot->cmd, "DUP\n", 4) == 0) )
    {
            tbot->packet_status = BOT_STATUS_EXIT;
            break;
    }
    //keep alive
    if(memcmp(tbot->cmd, "PING", 4) == 0)
    {
            write(tbot->socket_handle, "PONG", 4);
            tbot->packet_status = BOT_STATUS_KEEP_ALIVE;
            break;
    }
```

攻击协议明文解析

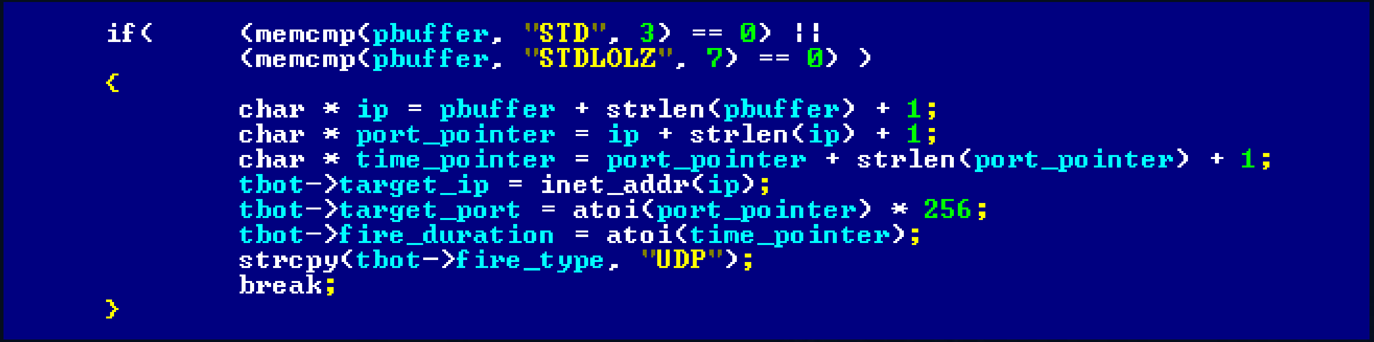
gafgyt家族攻击协议



```
2017-03-12 04:01:05.481184  104.168.170.177  172.16.1.32      TCP  93 212a20535444203…  666 → 33339 [PSH, ACK]
2017-03-12 04:01:05.516376  172.16.1.32      104.168.170.177  TCP  66                 33339 → 666 [ACK] Seq=
2017-03-12 04:01:05.673105  104.168.170.177  172.16.1.32      TCP  67 0a              666 → 33339 [PSH, ACK]
2017-03-12 04:01:05.674424  172.16.1.32      104.168.170.177  TCP  66                 33339 → 666 [ACK] Seq=
2017-03-12 04:01:05.681215  172.16.1.32      67.225[redacted]  UDP  92 737464007767657…  36896 → 20480 Len=50
2017-03-12 04:01:05.682240  172.16.1.32      67.225[redacted]  UDP  92 737464007767657…  36896 → 20480 Len=50
2017-03-12 04:01:05.682409  172.16.1.32      67.225[redacted]  UDP  92 737464007767657…  36896 → 20480 Len=50
2017-03-12 04:01:05.682559  172.16.1.32      67.225[redacted]  UDP  92 737464007767657…  36896 → 20480 Len=50
2017-03-12 04:01:05.682714  172.16.1.32      67.225[redacted]  UDP  92 737464007767657…  36896 → 20480 Len=50
2017-03-12 04:01:05.682856  172.16.1.32      67.225[redacted]  UDP  92 737464007767657…  36896 → 20480 Len=50

> Ethernet II, Src: Vmware_60:0e:78 (00:0c:29:60:0e:78), Dst: 5c:21:ef:00:a1:22 (5c:21:ef:00:a1:22)
> Internet Protocol Version 4, Src: 104.168.170.177, Dst: 172.16.1.32
> Transmission Control Protocol, Src Port: 666, Dst Port: 33339, Seq: 15, Ack: 22, Len: 27
v Data (27 bytes)
    Data: 212a205354442036372e3232352e3233312e323233203830...
    [Length: 27]

0000  5c 21 ef 00 a1 22 00 0c  29 60 0e 78 08 00 45 00   \!···"·· )`·x··E·
0010  00 4f 4c ce 40 00 35 06  38 51 68 a8 aa b1 ac 10   ·OL·@·5· 8Qh·····
0020  01 20 02 9a 82 3b 7c 9a  2f 93 61 e3 02 c6 80 18   · ···;|· /·a·····
0030  00 1d 83 3f 00 00 01 01  08 0a 81 c0 6b a8 00 01   ···?···· ····k···
0040  09 fd 21 2a 20 53 54 44  20 36 37 2e 32 32 35 2e   ··!* STD  67.225.
0050  32 33 31 2e 32 32 33 20  38 30 20 36 30            [redacted]  80 60
```

攻击协议明文解析

gafgyt家族攻击协议解析代码

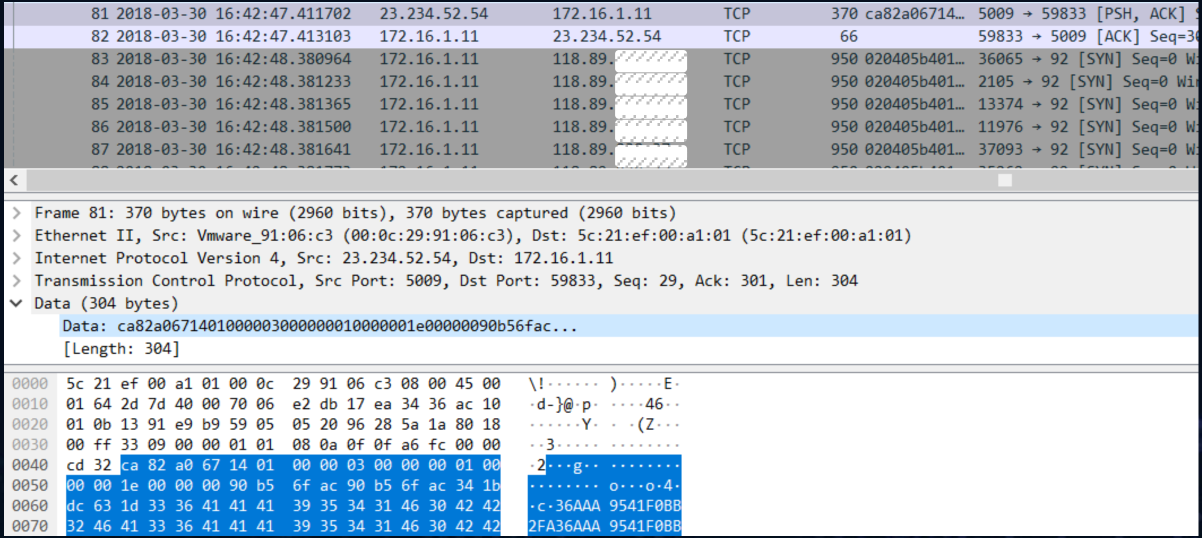


```
if(      (memcmp(pbuffer, "STD", 3) == 0) ||
         (memcmp(pbuffer, "STDLOLZ", 7) == 0) )
{
         char * ip = pbuffer + strlen(pbuffer) + 1;
         char * port_pointer = ip + strlen(ip) + 1;
         char * time_pointer = port_pointer + strlen(port_pointer) + 1;
         tbot->target_ip = inet_addr(ip);
         tbot->target_port = atoi(port_pointer) * 256;
         tbot->fire_duration = atoi(time_pointer);
         strcpy(tbot->fire_type, "UDP");
         break;
}
```

攻击协议密文解析

FIT 2019

xorddos家族攻击协议



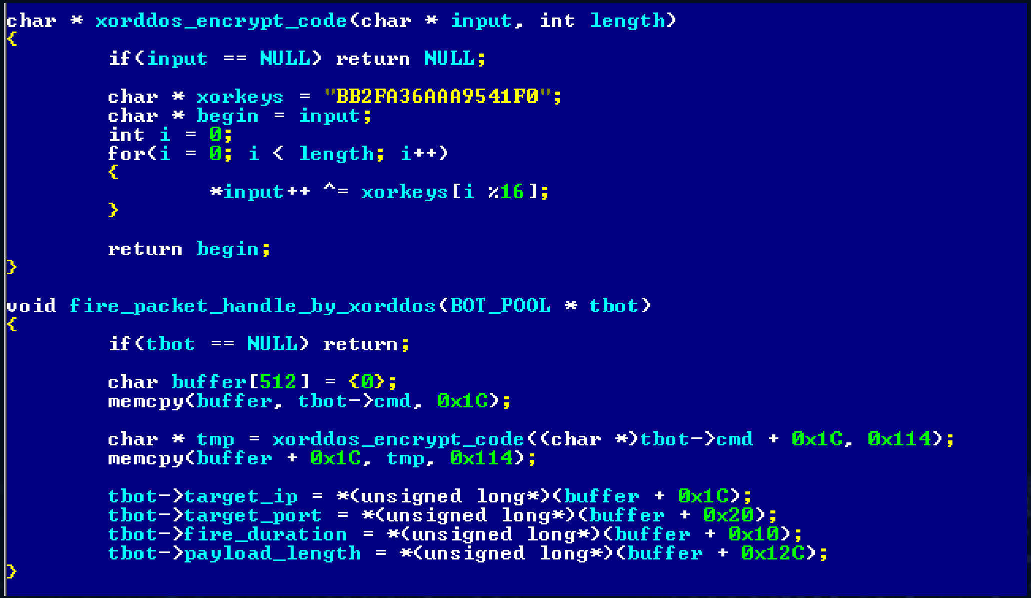
```
81 2018-03-30 16:42:47.411702   23.234.52.54   172.16.1.11   TCP   370 ca82a06714…  5009 → 59833 [PSH, ACK] S
82 2018-03-30 16:42:47.413103   172.16.1.11    23.234.52.54  TCP    66               59833 → 5009 [ACK] Seq=3
83 2018-03-30 16:42:48.380964   172.16.1.11    118.89.[redacted]   TCP   950 020405b401…  36065 → 92 [SYN] Seq=0 W
84 2018-03-30 16:42:48.381233   172.16.1.11    118.89.[redacted]   TCP   950 020405b401…  2105 → 92 [SYN] Seq=0 Wi
85 2018-03-30 16:42:48.381365   172.16.1.11    118.89.[redacted]   TCP   950 020405b401…  13374 → 92 [SYN] Seq=0 W
86 2018-03-30 16:42:48.381500   172.16.1.11    118.89.[redacted]   TCP   950 020405b401…  11976 → 92 [SYN] Seq=0 W
87 2018-03-30 16:42:48.381641   172.16.1.11    118.89.[redacted]   TCP   950 020405b401…  37093 → 92 [SYN] Seq=0 W
```

```
> Frame 81: 370 bytes on wire (2960 bits), 370 bytes captured (2960 bits)
> Ethernet II, Src: Vmware_91:06:c3 (00:0c:29:91:06:c3), Dst: 5c:21:ef:00:a1:01 (5c:21:ef:00:a1:01)
> Internet Protocol Version 4, Src: 23.234.52.54, Dst: 172.16.1.11
> Transmission Control Protocol, Src Port: 5009, Dst Port: 59833, Seq: 29, Ack: 301, Len: 304
v Data (304 bytes)
     Data: ca82a06714010000030000000010000001e00000090b56fac...
     [Length: 304]
```

```
0000  5c 21 ef 00 a1 01 00 0c  29 91 06 c3 08 00 45 00   \!······ )·····E·
0010  01 64 2d 7d 40 00 70 06  e2 db 17 ea 34 36 ac 10   ·d-}@·p· ····46··
0020  01 0b 13 91 e9 b9 59 05  05 20 96 28 5a 1a 80 18   ······Y· · ·(Z···
0030  00 ff 33 09 00 00 01 01  08 0a 0f 0f a6 fc 00 00   ··3····· ········
0040  cd 32 ca 82 a0 67 14 01  00 00 03 00 00 00 01 00   ·2···g·· ········
0050  00 00 1e 00 00 00 90 b5  6f ac 90 b5 6f ac 34 1b   ········ o···o·4·
0060  dc 63 1d 33 36 41 41 41  39 35 34 31 46 30 42 42   ·c·36AAA 9541F0BB
0070  32 46 41 33 36 41 41 41  39 35 34 31 46 30 42 42   2FA36AAA 9541F0BB
```

攻击协议密文解析

xorddos家族攻击协议解析代码



```c
char * xorddos_encrypt_code(char * input, int length)
{
        if(input == NULL) return NULL;

        char * xorkeys = "BB2FA36AAA9541F0";
        char * begin = input;
        int i = 0;
        for(i = 0; i < length; i++)
        {
                *input++ ^= xorkeys[i %16];
        }

        return begin;
}

void fire_packet_handle_by_xorddos(BOT_POOL * tbot)
{
        if(tbot == NULL) return;

        char buffer[512] = {0};
        memcpy(buffer, tbot->cmd, 0x1C);

        char * tmp = xorddos_encrypt_code((char *)tbot->cmd + 0x1C, 0x114);
        memcpy(buffer + 0x1C, tmp, 0x114);

        tbot->target_ip = *(unsigned long*)(buffer + 0x1C);
        tbot->target_port = *(unsigned long*)(buffer + 0x20);
        tbot->fire_duration = *(unsigned long*)(buffer + 0x10);
        tbot->payload_length = *(unsigned long*)(buffer + 0x12C);
}
```

代码测试

对特定IP的请求转发到本机进行处理

```
iptables -t nat -A OUTPUT -d 112.74.200.55 -j DNAT --to-destination 127.0.0.1
```

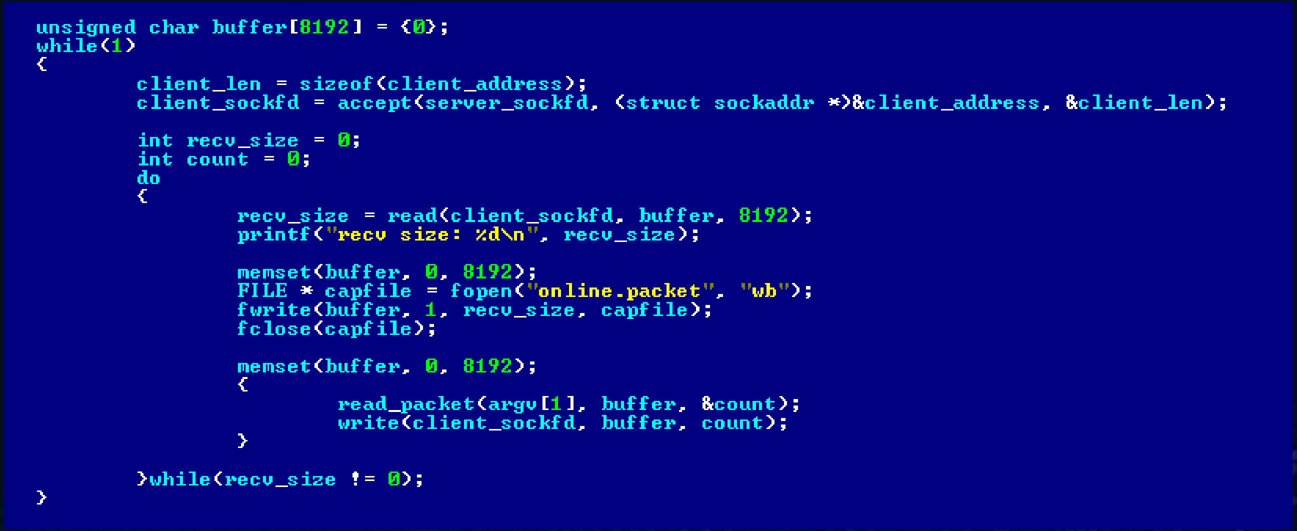
搭建伪服务器端，监听指定端口

运行原始样本，肉鸡上线后，对其发送经过我们重组的攻击指令

如果肉鸡能够成功发出攻击包，表明我们的攻击指令构造正确

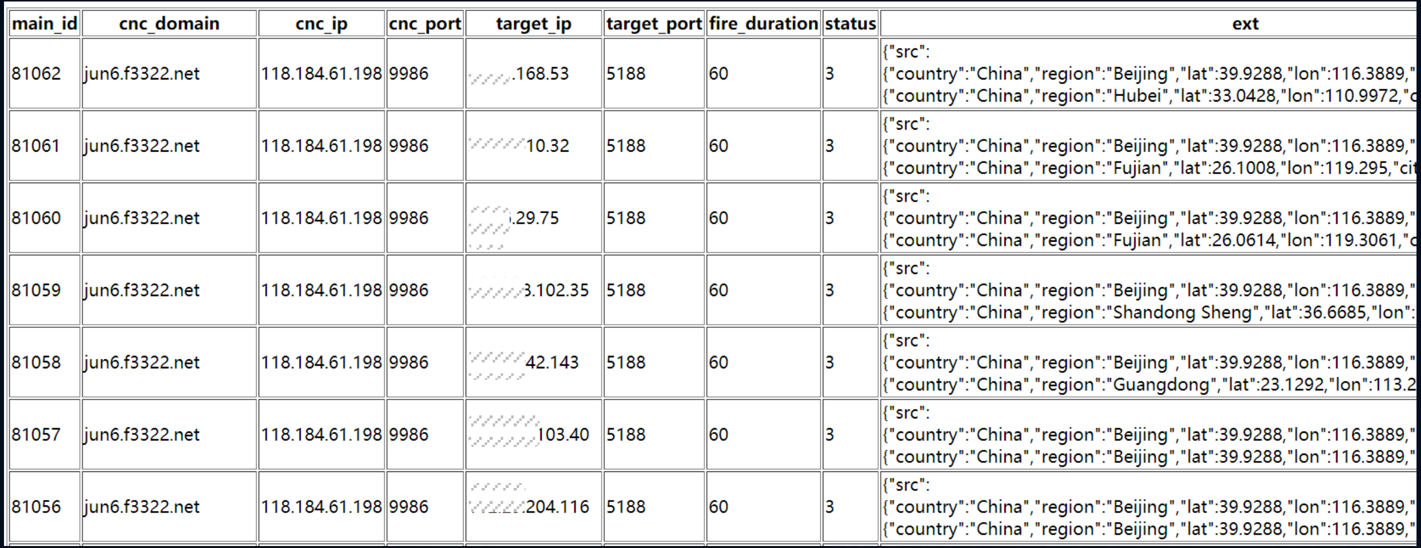
代码测试

伪服务器端代码



```
unsigned char buffer[8192] = {0};
while(1)
{
        client_len = sizeof(client_address);
        client_sockfd = accept(server_sockfd, (struct sockaddr *)&client_address, &client_len);

        int recv_size = 0;
        int count = 0;
        do
        {
                recv_size = read(client_sockfd, buffer, 8192);
                printf("recv size: %d\n", recv_size);

                memset(buffer, 0, 8192);
                FILE * capfile = fopen("online.packet", "wb");
                fwrite(buffer, 1, recv_size, capfile);
                fclose(capfile);

                memset(buffer, 0, 8192);
                {
                        read_packet(argv[1], buffer, &count);
                        write(client_sockfd, buffer, count);
                }

        }while(recv_size != 0);
}
```

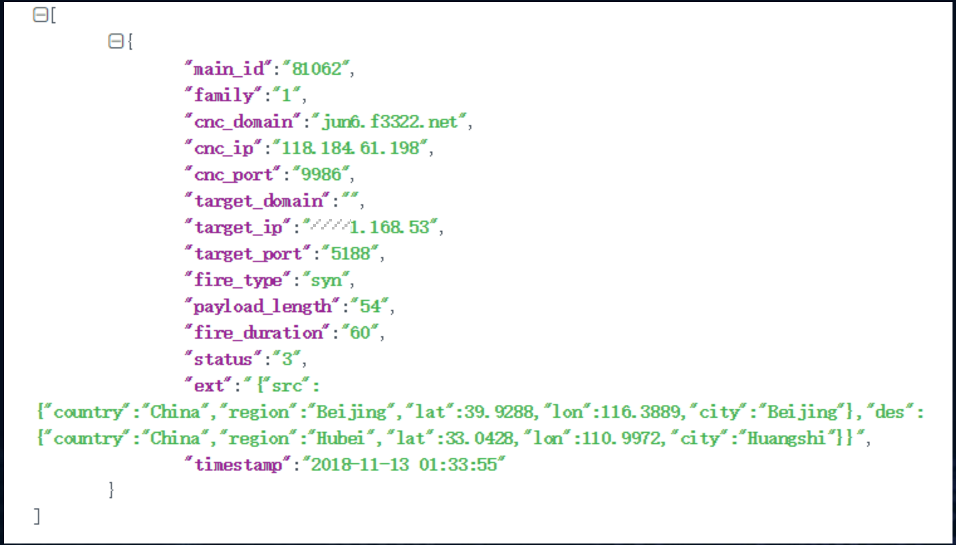
历史攻击指令数据库



main_id	cnc_domain	cnc_ip	cnc_port	target_ip	target_port	fire_duration	status	ext
81062	jun6.f3322.net	118.184.61.198	9986	[redacted].168.53	5188	60	3	{"src": {"country":"China","region":"Beijing","lat":39.9288,"lon":116.3889," {"country":"China","region":"Hubei","lat":33.0428,"lon":110.9972,"c
81061	jun6.f3322.net	118.184.61.198	9986	[redacted]10.32	5188	60	3	{"src": {"country":"China","region":"Beijing","lat":39.9288,"lon":116.3889," {"country":"China","region":"Fujian","lat":26.1008,"lon":119.295,"cit
81060	jun6.f3322.net	118.184.61.198	9986	[redacted].29.75	5188	60	3	{"src": {"country":"China","region":"Beijing","lat":39.9288,"lon":116.3889," {"country":"China","region":"Fujian","lat":26.0614,"lon":119.3061,"c
81059	jun6.f3322.net	118.184.61.198	9986	[redacted]3.102.35	5188	60	3	{"src": {"country":"China","region":"Beijing","lat":39.9288,"lon":116.3889," {"country":"China","region":"Shandong Sheng","lat":36.6685,"lon":
81058	jun6.f3322.net	118.184.61.198	9986	[redacted]42.143	5188	60	3	{"src": {"country":"China","region":"Beijing","lat":39.9288,"lon":116.3889," {"country":"China","region":"Guangdong","lat":23.1292,"lon":113.2
81057	jun6.f3322.net	118.184.61.198	9986	[redacted]103.40	5188	60	3	{"src": {"country":"China","region":"Beijing","lat":39.9288,"lon":116.3889," {"country":"China","region":"Beijing","lat":39.9288,"lon":116.3889,"
81056	jun6.f3322.net	118.184.61.198	9986	[redacted].204.116	5188	60	3	{"src": {"country":"China","region":"Beijing","lat":39.9288,"lon":116.3889," {"country":"China","region":"Beijing","lat":39.9288,"lon":116.3889,"

预警接口

json格式的输出接口



```
[
        {
                "main_id":"81062",
                "family":"1",
                "cnc_domain":"jun6.f3322.net",
                "cnc_ip":"118.184.61.198",
                "cnc_port":"9986",
                "target_domain":"",
                "target_ip":"[redacted]1.168.53",
                "target_port":"5188",
                "fire_type":"syn",
                "payload_length":"54",
                "fire_duration":"60",
                "status":"3",
                "ext":"{"src":
{"country":"China","region":"Beijing","lat":39.9288,"lon":116.3889,"city":"Beijing"},"des":
{"country":"China","region":"Hubei","lat":33.0428,"lon":110.9972,"city":"Huangshi"}}",
                "timestamp":"2018-11-13 01:33:55"
        }
]
```

统计信息

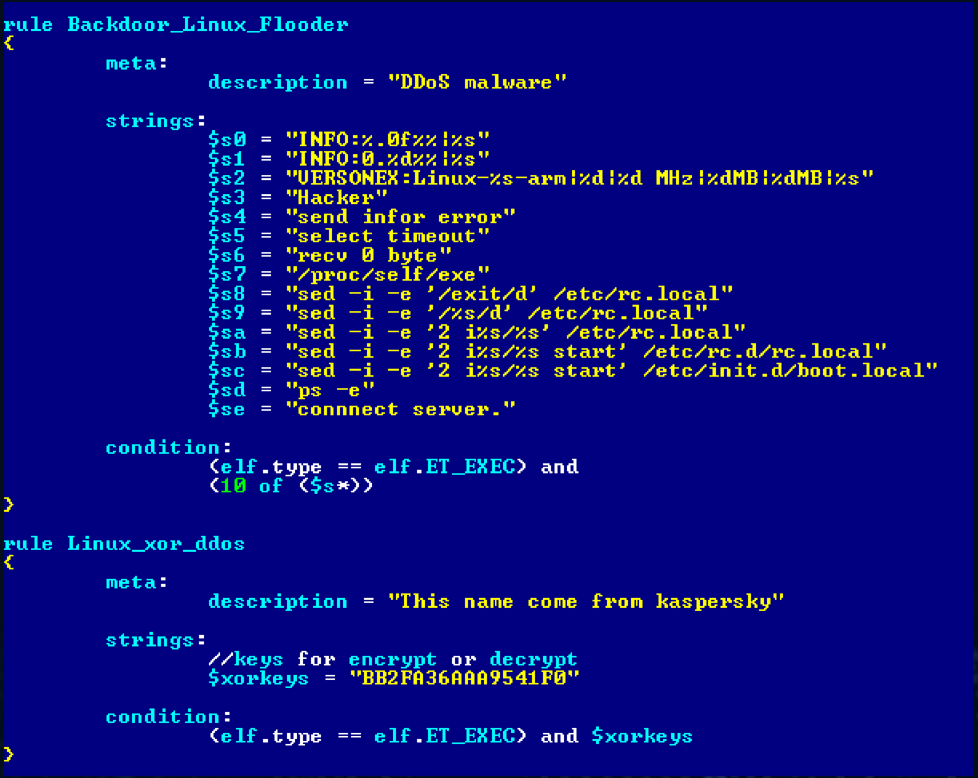
FIT 2019

某c&c在2018-07-31的攻击情况

开始时间	11:00:41
结束时间	23:58:48
攻击目标	275个
攻击次数	9139次
国内IP占比	63%
单个目标最大攻击次数	913次
单个目标最长攻击时长	7小时

每30秒下发一次攻击指令，每35秒下发一次停止指令

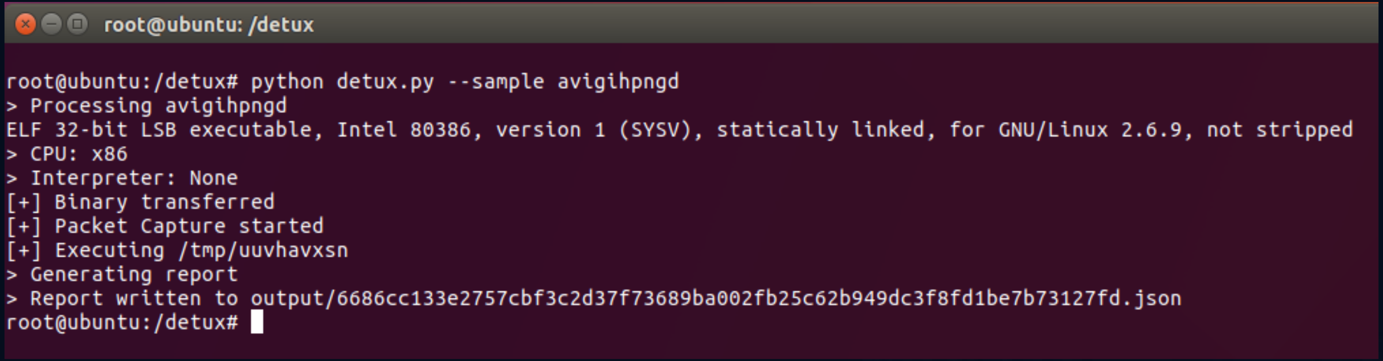
辅助分析工具-样本家族分类



```
rule Backdoor_Linux_Flooder
{
        meta:
                description = "DDoS malware"

        strings:
                $s0 = "INFO:%.0f%%|%s"
                $s1 = "INFO:0.%d%%|%s"
                $s2 = "VERSONEX:Linux-%s-arm|%d|%d MHz|%dMB|%dMB|%s"
                $s3 = "Hacker"
                $s4 = "send infor error"
                $s5 = "select timeout"
                $s6 = "recv 0 byte"
                $s7 = "/proc/self/exe"
                $s8 = "sed -i -e '/exit/d' /etc/rc.local"
                $s9 = "sed -i -e '/%s/d' /etc/rc.local"
                $sa = "sed -i -e '2 i%s/%s' /etc/rc.local"
                $sb = "sed -i -e '2 i%s/%s start' /etc/rc.d/rc.local"
                $sc = "sed -i -e '2 i%s/%s start' /etc/init.d/boot.local"
                $sd = "ps -e"
                $se = "connnect server."

        condition:
                (elf.type == elf.ET_EXEC) and
                (10 of ($s*))
}

rule Linux_xor_ddos
{
        meta:
                description = "This name come from kaspersky"

        strings:
                //keys for encrypt or decrypt
                $xorkeys = "BB2FA36AAA9541F0"

        condition:
                (elf.type == elf.ET_EXEC) and $xorkeys
}
```

辅助分析工具-开源沙盒

https://github.com/detuxsandbox/detux
输出网络流量，运行时行为等信息

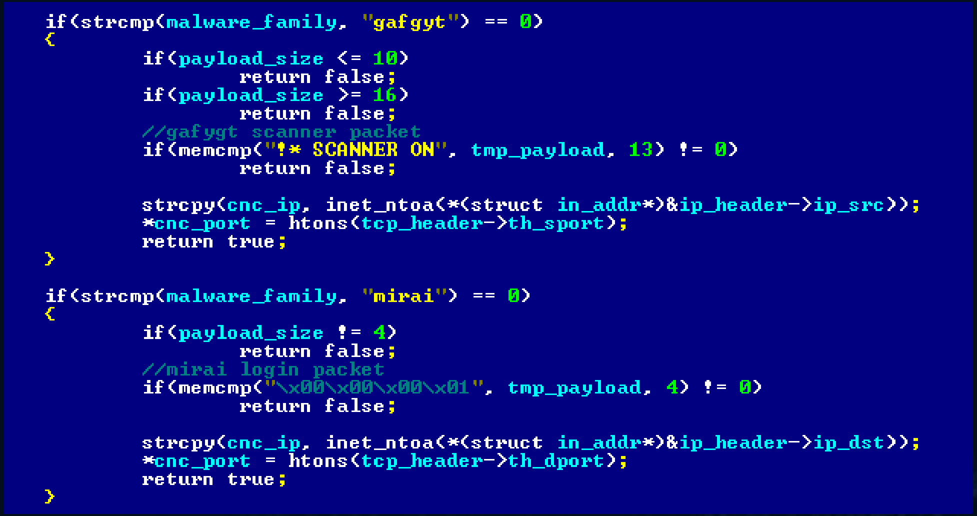

root@ubuntu: /detux

```
root@ubuntu:/detux# python detux.py --sample avigihpngd
> Processing avigihpngd
ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, for GNU/Linux 2.6.9, not stripped
> CPU: x86
> Interpreter: None
[+] Binary transferred
[+] Packet Capture started
[+] Executing /tmp/uuvhavxsn
> Generating report
> Report written to output/6686cc133e2757cbf3c2d37f73689ba002fb25c62b949dc3f8fd1be7b73127fd.json
root@ubuntu:/detux# 
```

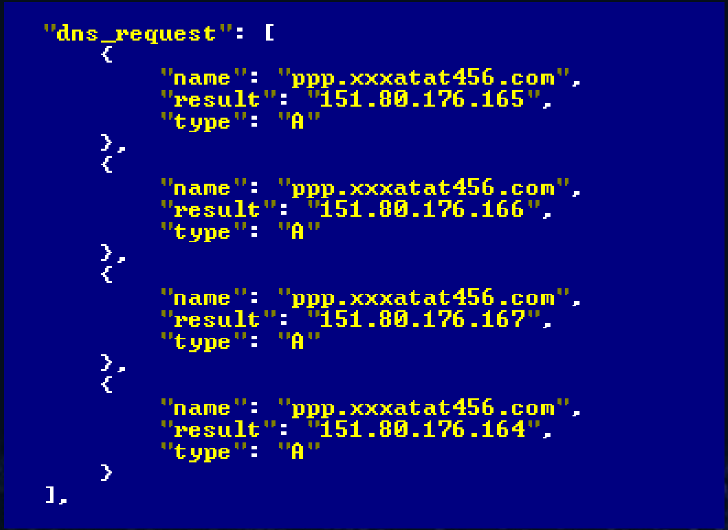
辅助分析工具-C&C提取

FIT 2019

提取流量中的C&C IP，域名，端口信息。存储到数据库，做威胁情报数据源

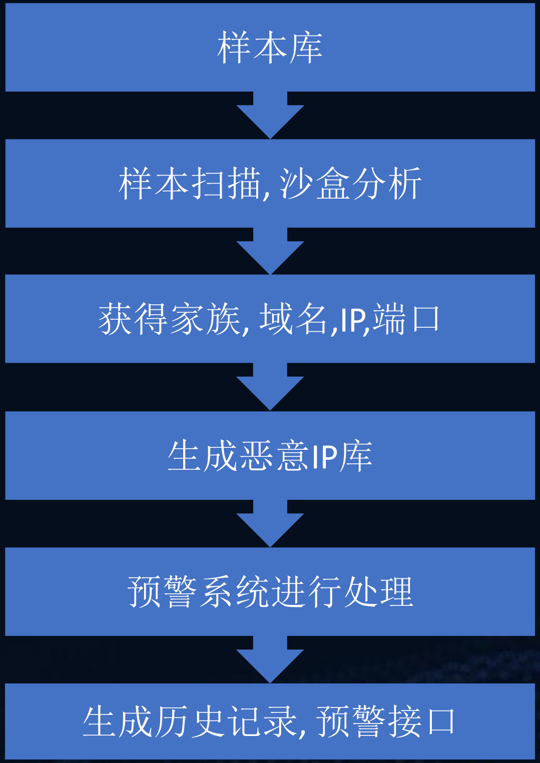


```
if(strcmp(malware_family, "gafgyt") == 0)
{
        if(payload_size <= 10)
                return false;
        if(payload_size >= 16)
                return false;
        //gafgyt scanner packet
        if(memcmp("!* SCANNER ON", tmp_payload, 13) != 0)
                return false;

        strcpy(cnc_ip, inet_ntoa(*(struct in_addr*)&ip_header->ip_src));
        *cnc_port = htons(tcp_header->th_sport);
        return true;
}

if(strcmp(malware_family, "mirai") == 0)
{
        if(payload_size != 4)
                return false;
        //mirai login packet
        if(memcmp("\x00\x00\x00\x01", tmp_payload, 4) != 0)
                return false;

        strcpy(cnc_ip, inet_ntoa(*(struct in_addr*)&ip_header->ip_dst));
        *cnc_port = htons(tcp_header->th_dport);
        return true;
}
```



```
"dns_request": [
    {
        "name": "ppp.xxxatat456.com",
        "result": "151.80.176.165",
        "type": "A"
    },
    {
        "name": "ppp.xxxatat456.com",
        "result": "151.80.176.166",
        "type": "A"
    },
    {
        "name": "ppp.xxxatat456.com",
        "result": "151.80.176.167",
        "type": "A"
    },
    {
        "name": "ppp.xxxatat456.com",
        "result": "151.80.176.164",
        "type": "A"
    }
],
```

DDoS预警流程图



THANKS