"BILLGATES" 僵尸网络分析

文/ jm33_ng

原文：BillGates Botnet Malware Used in Large DDoS Attacks
原文地址：https://www.akamai.com/kr/ko/multimedia/documents/state-of-the-internet/bill-gates-botnet-threat-advisory.pdf
本文由 jm33_ng 翻译整理

0x00 综述

Akamai 的安全情报研究团队（SIRT）持续观测到 BillGates 家族木马被用于 DDoS 攻击。控制该恶意软件的攻击者 2014 年 2 月在一家俄文网站首次披露可以获取被感染系统的完整权限。该 rootkit 可用的攻击方式包括：ICMP 泛洪, TCP 泛洪, UDP 泛洪，SYN 泛洪，HTTP 泛洪 (Layer7)，DNS 反射泛洪攻击。这个恶意软件是基于 Elknot's malware 进行的更新重写。现在它已经被发现了好几年。这些年来，由它组成的僵尸网络成长了起来，如今的僵尸网络正在发动相当规模的攻击。像 XOR botnet 一样，研究者相信 gates 也源自亚洲（本人猜测来自中国）。攻击者使用和 XOR 相同的传播手法，主要为 SSH 暴力破解（root），之前有报告说它也使用了一个 ElasticSearch Java VM 的漏洞进行传播。
和 XOR 僵尸网络一样，gates 也主要针对亚洲在线游戏服务器发动攻击。
Akamai SIRT 观察到自从 2015 年 Q4，XOR 的 C2 服务器被拿下之后，攻击者开始使用 gates 去攻击和之前相同的目标。

本文使用已知的 DDoS 攻击案例和一个 gates 僵尸网络的案例作为资料来源，同时使用了一个用于测试该木马的站点服务器作为参考。也涵盖了恶意程序检测，僵尸网络攻击特征分析，以及恶意程序的清除方法。

gates 样本由一个叫做 builder 的生成器生成，这个工具允许用户创建自己的定制化恶意程序（gates），也造成了 gates 的大面积传播。

一年多前，Malware Must Die 团队发布了一个演示构建和传播 gates 木马的视频。

0x01 rootkit 分析

感染之后，gates 可以发动 DDoS 攻击，打开端口服务，几乎完全控制被感染主机。Gates 包括多个运行阶段，每个阶段都负责特定的任务。

主程序具备两个反调试和反修改检查，第一个是比较自己的文件大小和预设值差异，如果不匹配，它会停止运行。第二个检查是在父进程的关键字里搜索 gdb 字样，避免自己在 gdb 调试器中运行。

下一步是解密自己的配置文件（RSA 算法），配置的格式

```
<C2-ip>:<C2-
port>:<Is Listener ?>:<Is Service ?>:<Campaign Name>:<Enable Backdoor ?>
```

下表显示了每个配置参数和对应的全局变

Configuration parameter	Global Variable Name
<C2-ip>	g_strConnTgt
<C2-port>	g_iGatsPorts
<Is Listener Boolean>	g_iGatsIsFx
<Is Service Boolean>	g_iIsService
<Campaign Name>	g_strForceNote
<Enable Backdoor Boolean>	g_iDoBackdoor

下表显示了每个配置参数和对应的全局变量

g_GatesTypes value	Function	Filename
0	MainMonitor()	/usr/bin/.sshd
1	MainBeikong()	Anything else
2	MainBackdoor()	/usr/bin/bsd-port/getty
3	MainSystool()	/bin/netstat /usr/bin/ps /usr/sbin/ps /bin/ss /usr/sbin/ss /usr/bin/netstat /bin/lsof /usr/sbin/lsof /bin/ps /usr/sbin/ps /usr/bin/lsof /usr/sbin/ss

1. **GATE 1, MAINBEIKONG**

初次感染之后该恶意软件会以随机文件名启动，这个阶段，gates 首先检查自己是不是已经运行，方法是检查/tmp/bill.lod 是否存在，该文件的内容是已运行的恶意程序 PID。如果已经运行，则 kill 掉原进程，然后运行自己并更新/tmp/bill.lod 的内容。下一步它检查 g_IsService 的值，如果是 1，则在该主机上配置自己的持久化服务。创建以下脚本：

```
/etc/init.d/DbSecuritySpt
/etc/rc1.d/S97DbSecuritySpt
/etc/rc2.d/S97DbSecuritySpt
/etc/rc3.d/S97DbSecuritySpt
/etc/rc4.d/S97DbSecuritySpt
/etc/rc5.d/S97DbSecuritySpt
```

DbSecuritySpt 的内容如下，所有 rc.d 下的软链接都指向这个内容的文件 /etc/init.d/DbSecuritySpt

```
#!/bin/bash
<full-path-to-malware-filename>
```

驻留服务配置完之后，恶意软件会检查 g_iDoBackDoor 的值，如果是 1，则检查 usr/bin/bsd-port/getty.lock 和 /usr/bin/bsd-port/udev.lock。如果存在，则 kill 掉它们含有的 PID，下一步，它使用 system()调用把自己复制到/usr/bin/bsd-port/getty 和/usr/bin/bsd-port/.sshd 。然

后暂停当前进程直到子进程退出，这个点上，它调用函数 MainProcess(), 这个函数将在稍后解释。

2. 1.2GATE 2, MAINBACKDOOR

在此阶段的 gates 会检查 usr/bin/bsd-port/getty.lock 是否存在，如果有一个已经运行的进程，且其 PID 匹配该文件里的 PID，它直接退出。否则，它会继续配置它的驻留服务，进一步损坏重要的系统工具文件。在 gates2 阶段，它创建如下内容的/etc/init.d/selinux

```
#!/bin/bash
/usr/bin/bsd-port/getty
```

然后建立软链接指向它：

```
/etc/rc1.d/S99selinux
/etc/rc2.d/S99selinux
/etc/rc3.d/S99selinux
/etc/rc4.d/S99selinux
/etc/rc5.d/S99selinux
```

每个链接保证自启动的进程会出现在每个 Linux 运行级别中
它也检测以下系统工具是否存在：

```
/bin/netstat
/bin/lsof
/bin/ps
/bin/ss
/usr/bin/netstat
/usr/bin/lsof
/usr/bin/ps
/usr/bin/ss
/usr/sbin/netstat
/usr/sbin/lsof
/usr/sbin/ps
/usr/sbin/ss
```

每个文件都被它替换成自己的主程序文件，然后原文件被移到/usr/bin/dpkgd/目录。同时它也会设置合适的执行权限以保证正常执行。现在 gates 继续执行到 MainProcess()。

3. GATE 0, MAINMONITOR

如果用 .sshd 启动，恶意程序会执行 gates0 阶段。这个阶段先读取 gates1 生成的 /tmp/notify.file ，该文件包含了原主程序的完整执行路径。然后这个路径被用于初始化一个新的线程对象，该对象激活 CThreadMonGate::MainProcess，主要功能就是监控是否有进程锁了 /tmp/gates.lock，每 60 秒检测一次。如果检测失败，则重新运行主程序进行重新感染。主要来说，这个东西用于保证恶意软件主程序健康运行。

4. GATE 3, MAINSYSTOOL

这个阶段负责隐蔽和持续的恶意软件操作。之前在 gates2，一些系统工具被移动然后原位置被恶意软件主程序替换。Gates3 负责执行那些被替换的系统命令然后隐藏掉它相关的输

出信息。实现方法：检查自己的执行路径，如果匹配一个被替换的系统命令，它就执行原本的系统命令（使用移动后的新的路径），然后把返回的结果逐行处理，如果有关键字，则去掉这一行。

5. MAIN PROCESS

当所有初始阶段都完成之后，恶意软件就植入了系统中，现在它运行 MainProcess 函数。这是个多线程阶段，负责和 C2 服务器通信，处理命令（如发动 DDoS 攻击）。
恶意软件使用解密的配置信息连接 C2 服务器，成功连接之后，它会用宿主机的信息在 C2 注册自己。如 uname –sr 和 CPU 信息之类。

```
0x0818c0b0 : 01 00 00 00 72 00 00 00 00 f4 01 00 00 32 00 00   ....r........2..
0x0818c0c0 : 00 e8 03 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0818c0d0 : 00 00 01 01 00 00 00 00 01 00 00 00 ac 10 6c 8c   ..............l.
0x0818c0e0 : c0 a8 ac cf c0 a8 ac cf ac 10 6c 89 ac 10 6c 89   ..........l...l.
0x0818c0f0 : ff ff 01 00 00 00 00 00 d2 af d2 af 3a 00 01 00   ............:...
0x0818c100 : 00 00 f5 08 00 00 df 07 00 00 4c 69 6e 75 78 20   ..........Linux
0x0818c110 : 33 2e 31 33 2e 30 2d 37 37 2d 67 65 6e 65 72 69   3.13.0-77-generi
0x0818c120 : 63 00 31 3a 47 32 2e 34 30 00                     c.1:G2.40.
```

以上注册信息包括

- 感染机 IP 地址
- DNS 地址
- CPU 数量
- CPU 速度
- 内存大小
- uname -sr 输出
- 静态字符串，可能是恶意软件版本

C2 回复信息如下：

```
0000000: 0100 0000 5100 0000 00f4 0100 0032 0000  ....Q........2.. 0000010: 00e8
0300 0056 0300 0000 0000 0001 0000  .....V..........  0000020: 0001 0000 0010 0200
d007 0000 0000 0000  ..............  0000030: 0000 2000 0000 0400 0000 0400 0001
0000   ..  ............  0000040:  000a  0000  0000  0001  0000  00xx  xx2e
xxxx  ..............
0000050: 2e31 332e 3738 0050 000a               .13.78.P..
```

具体包括

- 要执行的命令
- 剩余的载荷大小（本次的载荷，用于确定完整性）
- 攻击类型
- TCP Flags (这里是 02 代表 SYN)
- 目标数量
- 目标 IP, ASCII 形式
- 端口号，以十六进制形式跟随 ASCII 形式的 IP 地址

0x02 DDOS 攻击载荷

发起的 DDoS 攻击大多是 SYN 和 DNS 泛洪，每种攻击都有特征，这里是一个 SYN 示例

```
14:16:29.472419 IP xxx.xxx.xxx.xxx.55670 > xxx.xxx.xxx.xxx.80: Flags [S], seq
1158631255:1158632225, win 60311, length 970
0x0000:  4500 03f2 574f 4000 fc06 48ca xxxx xxxx  E...WO@...H.....
0x0010:  xxxx xxxx d976 0050 450f 4f57 0000 0000  .....v.PE.OW....
0x0020:  5002 eb97 7767 0000 0000 0000 0000 0000  P...wg..........
0x0030:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0040:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0050:  0000 0000 0000 0000 0000 0000 0000 0000  ................ <! *****
Redacted for space ***** !> 0x03c0:  0000 0000 0000 0000 0000 0000 0000
0000  ................
0x03d0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x03e0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x03f0:  0000..
```

每个 SYN 包由 20 字节的 IP 头，20 字节 TCP 头（一共 970 字节 null 构建的载荷）组成，没有 TCP 头选项。TCP 窗口大小，序列号，TTL 都是完全随机的。恶意程序有能力伪造源 IP 地址。但有些观测到的攻击并没有伪造源 IP，这大概是因为它无法把伪造后的流量路由出感染主机的网络。

```
15:23:36.184608 IP xxx.xxx.xxx.xxx.32521 > xxx.xxx.xxx.xxx.53: 22148 [1au]
A? ghaxofybqxsmut.example.com. (55)
0x0000:  4500 0053 fa33 0000 3411 3bff xxxx xxxx  E..R.3..4.;..... 0x0010:  xxxx
xxxx 7f09 0035 003f e2ac 5684 0000  .......5.>..V... 0x0020:  0001 0000 0000 0001
0e67 6861 786f 6679  .........ghaxofy 0x0030:  6271 7873 6f75 7407 6578 616d 706c
6503   bqxsmut.example.  0x0040:   636f  6d00  0001  0001  0000  2910  0000
0080  com.......).....
0x0050:  0000 00
```

在这个用于演示的请求中，我们可以看到它在请求一个高度随机化的子域名。这是最容易辨识的 gates DNS 攻击流量特征。

0x03 已知攻击案例

和 XOR 僵尸网络很相似，BillGates 据推测也是源自亚洲。这个僵尸网络主要针对亚洲的游戏娱乐行业。
在 Akamai 的网络上观测到的攻击案例中，流量从几个 Gbps 到几百个 Gbps 不等。它们大多数包含不止一个恶意软件。Akamai 最新观测到的攻击（只包含 gates 的流量）出现在 2015 年。最高达到约 6.5Gbps——达到几乎 1 兆 packet 每秒。

Akamai Scrubbing Center	Peak Gigabits per Second	Peak Packets per Second
Hong Kong	0.24 Gbps	36.45 Kpps
Virginia	2.0 Gbps	243.37 Kpps
San Jose	1.16 Gbps	138.04 Kpps
Frankfurt	2.54 Gbps	302.16 Kpps
London	0.45 Gbps	55.93 Kpps
Tokyo	0.83 Gbps	113.12 Kpps

最大规模的攻击发生在 2015 年 12 月 30 日，流量达到 308Gbps，此次攻击来自 gates 僵尸网络，同时加入了其它攻击手段。

Akamai Scrubbing Center	Peak Gigabits per Second	Peak Packets per Second
Hong Kong	30.32 Gbps	18.94 Mpps
Virginia	47.73 Gbps	32.91 Mpps
San Jose	55.59 Gbps	23.89 Mpps
Frankfurt	56.68 Gbps	34.37 Mpps
London	47.18 Gbps	28.40 Mpps
Tokyo	71.13 Gbps	63.03 Mpps

```
05:43:39.199265 IP xx.xxx.18.106.33776 &gt; xx.xxx.43.26.80: Flags [S], seq
952308329:952309299, win 64713, length 970
05:43:39.199269 IP xxx.xx.117.78.28153 &gt; xx.xxx.43.26.80: Flags [S], seq
123035470:123036440, win 64398, length 970
05:43:39.199279 IP xxx.xx.34.183.57723 &gt; xx.xxx.43.26.80: Flags [S], seq
1883570416:1883571386, win 60240, length 970
05:43:39.199567 IP xxx.xx.34.183.53230 &gt; xx.xxx.43.26.80: Flags [S], seq
1865413005:1865413975, win 61837, length 970
05:43:39.199571 IP xxx.xxx.216.176.29140 &gt; xx.xxx.43.26.80: Flags [S], seq
1284626607:1284627577, win 60471, length 970
05:43:39.199578 IP xxx.xx.84.51.17549 &gt; xx.xxx.43.26.80: Flags [S], seq
2016171058:2016172028, win 61554, length 970
```

gates 有能力伪造源 IP，但是 Akamai 的调查发现这并不多见，大概是因为它无法把伪造的流量路由出被感染主机的网络。

0x04 二进制文件感染特征

首先检查 gates.lod， moni.lod 是否在/tmp 目录. 如果存在，gates.lod 会含有恶意程序主进程的 PID，moni.lod 则含有守护进程的 PID

```
$ ls -lha /tmp total 52K drwxrwxrwt 10 root root 4.0K Feb 10 18:17 . drwxr-xr-x
22 root root 4.0K Dec  9 11:09 ..
-rwxr-xr-x  1 root root    4 Feb  9 15:48 gates.lod
-rwxr-xr-x  1 root root    4 Feb  9 15:48 moni.lod
$ cat /tmp/gates.lod
13550
$ cat /tmp/moni.lod
13640
$ cat /proc/13640/cmdline
/usr/bin/.sshd
$ cat /proc/13550/cmdline
/home/user/Desktop/billgates_variant
```

检查 /etc/init.d/ 下是否有恶意脚本

```
$ ls -la /etc/init.d/selinux
-rwxr-xr-x 1 root root 36 Feb 4 15:48 /etc/init.d/selinux

$ ls -la /etc/init.d/DbSecuritySpt
-rwxr-xr-x 1 root root 39 Feb 4 15:48 /etc/init.d/DbSecuritySpt

$ cat /etc/init.d/selinux
#!/bin/bash
/usr/bin/bsd-port/getty

$ cat /etc/init.d/DbSecuritySpt
#!/bin/bash
/home/user/Desktop/billgates\_variant
```

etc/rc[run-level-number].d/ 下也可能会有恶意脚本

```
$ ls -la /etc/rc1.d/S99selinux
lrwxrwxrwx  1  root  root  19  Feb  4  15:48  /etc/rc1.d/S99selinux  ->
/etc/init.d/selinux
$ ls -la /etc/rc1.d/S97DbSecuritySpt
lrwxrwxrwx  1  root  root  25  Feb  4  15:48  /etc/rc1.d/S97DbSecuritySpt  ->
/etc/init.d/DbSecuritySpt
$ ls -la /etc/rc2.d/S99selinux
lrwxrwxrwx  1  root  root  19  Feb  4  15:48  /etc/rc2.d/S99selinux  ->
/etc/init.d/selinux
$ ls -la /etc/rc2.d/S97DbSecuritySpt
lrwxrwxrwx  1  root  root  25  Feb  4  15:48  /etc/rc2.d/S97DbSecuritySpt  ->
/etc/init.d/DbSecuritySpt
$ ls -la /etc/rc3.d/S99selinux
```

```
lrwxrwxrwx 1 root root 19 Feb 4 15:48 /etc/rc3.d/S99selinux ->
/etc/init.d/selinux
$ ls -la /etc/rc3.d/S97DbSecuritySpt
lrwxrwxrwx 1 root root 25 Feb 4 15:48 /etc/rc3.d/S97DbSecuritySpt ->
/etc/init.d/DbSecuritySpt
$ ls -la /etc/rc4.d/S99selinux
lrwxrwxrwx 1 root root 19 Feb 4 15:48 /etc/rc4.d/S99selinux ->
/etc/init.d/selinux
$ ls -la /etc/rc4.d/S97DbSecuritySpt
lrwxrwxrwx 1 root root 25 Feb 4 15:48 /etc/rc4.d/S97DbSecuritySpt ->
/etc/init.d/DbSecuritySpt
$ ls -la /etc/rc5.d/S99selinux
lrwxrwxrwx 1 root root 19 Feb 4 15:48 /etc/rc5.d/S99selinux ->
/etc/init.d/selinux
$ ls -la /etc/rc5.d/S97DbSecuritySpt lrwxrwxrwx 1 root root 25 Feb 4 15:48
/etc/rc5.d/S97DbSecuritySpt -> /etc/init.d/DbSecuritySpt
```

检查 ps , lsof , netstat , ss 看它们是不是同一个文件

```
$ md5sum /bin/ps
22e2cda565a857b1d78414ce50ac074d  /bin/ps
$ md5sum /bin/ss
22e2cda565a857b1d78414ce50ac074d  /bin/ss
$ md5sum /bin/netstat
22e2cda565a857b1d78414ce50ac074d  /bin/netstat
$ md5sum /usr/bin/lsof
22e2cda565a857b1d78414ce50ac074d  /usr/bin/lsof
$ md5sum /usr/bin/.sshd
22e2cda565a857b1d78414ce50ac074d  /usr/bin/.sshd
$ md5sum /home/user/Desktop/billgates_variant
22e2cda565a857b1d78414ce50ac074d  /home/user/Desktop/billgates_variant
$ md5sum /usr/bin/bsd-port/getty
22e2cda565a857b1d78414ce50ac074d  /usr/bin/bsd-port/getty
```

可以看到，这些文件是同一个，均由 gates 自我复制而来。

先 kill 掉 /tmp/moni.lod 含的 PID，这是守护进程，干掉之后可以确保恶意程序不自启。然后 kill 掉 /tmp/gates.lod， 再然后 kill 掉 /usr/bin/bsdport/getty.lock。

```
$ cat /tmp/moni.lod
13640
$ sudo kill -9 13640
$ cat /tmp/gates.lod
13550
$ sudo kill -9 13550
```

```
$ cat /usr/bin/bsd-port/getty.lock
13593
$ sudo kill -9 13593
```

最后一步，删除所有相关文件

```
# rm       -rf       /tmp/moni.lod       /tmp/gates.lod       /etc/init.d/selinux
/etc/init.d/DbSecuritySpt   /etc/rc1.d/S97DbSecuritySpt   /etc/rc1.d/S99selinux
/etc/rc2.d/S97DbSecuritySpt   /etc/rc2.d/S99selinux   /etc/rc3.d/S97DbSecuritySpt
/etc/rc3.d/S99selinux     /etc/rc4.d/S97DbSecuritySpt     /etc/rc4.d/S99selinux
/etc/rc5.d/S97DbSecuritySpt       /etc/rc5.d/S99selinux       /usr/bin/bsd-port/
/usr/bin/lsof /bin/ps /bin/ss /bin/netstat
```

0x05 检测方法

以下方法推荐受影响的机构用来检测 gates 木马

1. tcpdump 使用如下过滤参数

```
‘tcp[((tcp[12] >> 4) << 2):4] == 0x01000000’ and ‘ip[(ip[2:2] - 7):2] == 0x3a47’
```

过滤参数有两部分，第一部分在最先的 4 字节中定位 0x01000000, 它和 register 命令有关. 第二部分搜索 ASCII :G，从载荷的末尾往前数 7 字节。

Part 1:

```
‘tcp[((tcp[12] >> 4) << 2):4] == 0x01000000’
```

Part 2:

```
‘ip[(ip[2:2] - 7):2] == 0x3a47’
```

```
‘tcp[((tcp[12] >> 4) << 2):4] == 0x08000000’ and ‘((tcp[((tcp[12] >> 4) << 2) +
4:1]) + ((tcp[12] >> 4)<<2) + 8) + ((ip[0] & 0xf) << 2) == ip[2:2]’
```

这个过滤参数也是两部分，第一部分和刚才类似，第二部分根据命令计算包长度。

Part 1:

```
‘tcp[((tcp[12] >> 4) << 2):4] == 0x08000000’
```

Part 2:

```
‘((tcp[((tcp[12] >> 4) << 2) + 4:1]) + ((tcp[12] >> 4)<<2) + 8) + ((ip[0] & 0xf)
<< 2) == ip[2:2]’
```

2. 5.2 YARA 社区的检测规则

Yara 已经有现成的 gates 检测规则，不过，为了更准确的检测，我们加入了额外的规则，可以使用。

```
rule BillGatesv1 { meta:
   author = “Akamai SIRT”     description = “Rule to detect BillGates infection”
strings:
            $st0 = “xpacket.ko”
            $st1 = “libamplify.so”
```

```
            $st2 = “12CUpdateGates”
            $st3 = “11CUpdateBill”
            $st4 = “10CTcpAttack”
            $st5 = “10CAttackDns”
            $st6 = “10CAttackAmp”
            condition: all of them
}
```

0x06 参考链接

https://www.akamai.com/kr/ko/multimedia/documents/state-of-the-internet/bill-gates-botnet-threat-advisory.pdf