TLP: WHITE

Issue Date: 4.4.16

Risk Factor- High

Threat Advisory: "BillGates" Botnet

1.0 / OVERVIEW / Akamai's Security Intelligence Research Team (SIRT) continues to see the BillGates trojan/bot family of malware being used to launch DDoS attacks. Attackers who control the malware — first disclosed on a Russian IT website in February 2014 — can gain full control of the infected systems.

The attack vectors available within the toolkit include: ICMP flood, TCP flood, UDP flood, SYN flood, HTTP Flood (Layer7), and DNS query-of-reflection flood. This malware is an update and reuse of the Elknot's malware source code. It's been detected in the wild for a few years now. Over the years, the botnets composed of it have grown, and today's botnets are launching significantly large attacks.

Like the XOR botnet, this malware is believed to be of Asian origin. The attackers are using the same methods for infection, which are primarily SSH brute force attempts for root login credentials (previously it was reported that infection methods include a vulnerability in ElasticSearch Java VM).

The botnet targets are the same as the XOR botnet, most of which are hosted in Asia and online gaming institutions.

Akamai SIRT observed inactivity from the XOR C2 back in Q4 2015, which was publicly announced and believed to be part of a takedown operation. Once that occurred, the attackers started using the BillGates Botnet to launch attacks against the same target list.

This advisory includes validated DDoS attack campaigns and an example of this botnet being used as one of the sources, along with the use of a booter site. We also cover the detection of malware infections, identifying attack patterns from this botnet and how to clean an infected machine.

1.1 / NOT ONE, BUT MANY BOTNETS / The malware sample is generated by what's called a "Builder" — a piece of software that creates a variant of the actual BillGates malware. This allows anybody to build their own customized version of the malware, with their own C2 they control, and start infecting machines — thus creating their own botnet.
Because of this process, there are many active BillGates botnets operated by different individuals, attacking different targets across the globe.

A little over a year ago, the Malware Must Die team released a screen recording of how anybody can use the builder to generate their own version of the malware as well as spreading methods.

2.0 / TOOLKIT ANALYSIS / Upon infection, the malware is capable of launching DDoS attacks, opening ports and services, and nearly taking full control over the infected system. The malware consists of multiple stages, each responsible for certain tasks within its operation.

The executed binary initially does two anti-debugging/anti-tampering checks. It compares its size with a preset value; if both values do not match, it simply stops there. The next check is by searching for the **gdb** string within its parent process' full execution path. This ensures the malware was not started inside the GDB debugger.

The next step of its execution is to decrypt its own configuration using the RSA algorithm. The format for the configuration is **<C2-ip>:<C2-port>:<Is Listener ?>:<Is Service ?>:<Campaign Name>:<Enable Backdoor ?>**

The list below includes each configuration parameter and associated global variable:

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

Figure 1: Configuration parameter table

Once the malware has decrypted its configuration file, execution jumps directly to the malware's main functionality, which first checks the value of the **`g_GatesTypes`** global variable. Based on the value, the malware performs one of the following functions. The value is based on the filename and path the malware is executed from.

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

Figure 2: The 4 main functions executed based on associated file path

Let's review each of the 4 functions since they play a specific role in the process. The order in which they are explained matches the malware execution flow sequence.

2.1 / GATE 1, MAINBEIKONG / Since the initial infection of the malware will start with a random filename, the malware will enter this stage first. In this stage, the malware first checks if the malware is already running on the current machine by checking if the file **`/tmp/bill.lod`** exists. This file contains the process ID of the malicious instance of the already running process. If it finds an active process, it kills the process id and replaces the file's content with its current running-process id. Next it checks the value of **`g_IsService`** global variable (derived from the previously decrypted configuration), and if it's 1, it configures persistence on the system by creating the following startup scripts:

1. **`/etc/init.d/DbSecuritySpt`**
2. **`/etc/rc1.d/S97DbSecuritySpt`**
3. **`/etc/rc2.d/S97DbSecuritySpt`**
4. **`/etc/rc3.d/S97DbSecuritySpt`**
5. **`/etc/rc4.d/S97DbSecuritySpt`**
6. **`/etc/rc5.d/S97DbSecuritySpt`**

The contents of **`DbSecuritySpt`** and all of the files in the **`rc.d`** directories are symbolic links to **`/etc/init.d/DbSecuritySpt`**.

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

Figure 3: Contents of DbSecuritySpt script

After the persistence configuration is complete, the malware checks the value of the **g_iDoBackDoor** global variable. If set to 1, it checks if files **/usr/bin/bsd-port/getty.lock** and **/usr/bin/bsd-port/udev.lock** exist and kills the process ID contained inside these files. Next, it copies itself to **/usr/bin/bsd-port/getty** and **/usr/bin/bsd-port/.sshd**, which it executes via a call to **system().** It then pauses the current process until the child process has exited; at that point, it invokes the function **MainProcess()** explained later.

2.2 / GATE 2, MAINBACKDOOR / Under this instance of the malware, it checks if file **/usr/bin/bsd-port/getty.lock** exists and if there is an already running process with the process id that matches the content of that file. If this proves true, it simply exits; otherwise, it continues setting up more persistence measures and corrupting vital system tools.
For persistence of the Gate 2 instance of the malware, it creates the file **/etc/init.d/selinux** with the following content:

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

Figure 4: Contents of /etc/init.d/selinux file

It also creates the following symbolic links to it:

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

Figure 5: Symbolic links pointed at /etc/init.d/selinux

Each link ensures the malware will be included in the startup processes upon boot for each of the Linux run levels.

It also checks if any of the following system tools exist:

```
/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
```

Figure 6: List of system tools the malware checks for on the infected machine

It replaces each tool in the above list with a copy of itself, and it moves the origin files to the **`/usr/bin/dpkgd/`** directory. It also sets the appropriate permissions required to ensure they can be executed as normal. The malware then proceeds to the function **`MainProcess()`**.

2.3 / GATE 0, MAINMONITOR / If the process was started via the **`.sshd`** file, the malware executes stage Gate 0. This phase first reads the file "**`/tmp/notify.file`**" that was created by Gate 1, which contains the full path to the original executable. This path is then used to initialize a new thread object which activates **`CThreadMonGate::MainProcess`**, which simply monitors if any process currently has an active lock on the file **`/tmp/gates.lock`**, and repeats this check every 60 seconds. If this check fails, it simply reruns the main binary to restart the infection. Essentially, this process is safe-guarding the malware's main process, ensuring it is alive and well.

2.4 / GATE 3, MAINSYSTOOL / This phase is responsible for the stealthy and continued operation of the malware. Previously in Gate 2, some core system tools were moved by the malware, with copies of itself dropped in their place. Gate 3 acts as a type of pass-thru to those relocated binaries and is responsible for processing and removing indications of compromise from the original binaries' returned output. It does this by checking its own execution path: if it matches one of the replaced core system tools, it proceeds to execute the original utility using its new path and parsing the returned output line by line. If specific keywords are detected on each line, such as /**`usr/bin/bsd-port`** or a specific port number, it simply scrubs this line —stopping it from being output to the user.

2.5 / MAIN PROCESS / Once all of the initial phases have completed and the malware has rooted itself into the system, the malware runs the **`MainProcess`** function. This is a multi-threaded stage responsible for opening communication with the C2 server(s), parsing commands, and launching DDoS attacks.

The malware is going to try to connect to the C2 IP address and port number from the decrypted configuration. Upon successful connection, the malware registers itself with the C2 by sending information about the infected host. Some of the information includes output of the **`uname -sr`** command, number of CPU cores on the system, CPU speed, etc.

```
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.
```

Figure 7: C2 "registration" packet

The above payload consists of:

- IP address of the infected machine
- DNS Addresses
- Number of CPUs
- CPU Mhz derived from /proc/cpuinfo
- Total memory
- System kernel name and kernel version from "uname -sr"
- Static string, possibly malware version

The C2 responds right away with a command for the bot. Some of the available commands are **`Start DDoS Attack`**, **`Stop DDoS Attack`**, and **`Execute Shell Command`**. An example payload for **`Start DDoS Attack`** includes the following data:

```
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..
```

Figure 8: C2 "Start DDoS Attack" packet

The above payload consists of:

- Command for C2 to perform an attack
- Size of the rest of the payload
- Attack Type (In this case we have a SYN Flood)
- TCP Flags (In this case we have 02 for SYN)
- Number of targets
- Target IP in plain readable ASCII
- Following the IP in ASCII is the destination port address in hex

3.0 / DDOS ATTACK PAYLOADS / The DDoS Attacks most frequently observed are SYN and DNS Floods. Each of these attacks have their own distinguishable characteristics. Here is a sample SYN Flood:

```
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                   ..
```

Figure 9: SYN Flood attack packet

Every SYN packet consists of a 20-byte IP header, 20-byte TCP header, exactly 970 bytes of null padded payload, and no TCP header options. The TCP window size, sequence numbers, and TTL are all fully randomized. The malware has the ability to spoof the source IP address; however, some of the monitored attack campaigns did not use a spoofed source IP address. This is likely due to an inability to route spoofed traffic from the infected machine's network.

```
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                                  ...
```

Figure 10: DNS Flood attack packet

In the example malformed query in Figure 10, we see a highly randomized subdomain being requested, which is represented in **RED**. This is the most distinguishable characteristic and enough to identify DNS attack traffic generated by the BillGates malware.

4.0 / OBSERVED CAMPAIGNS / Much like the XOR botnet, the BillGates botnet is also believed to be of Asian origin. The botnet appears to mostly target Asia-based organizations, focusing heavily on the gaming and entertainment industries.

The attack campaigns observed on our network vary from several Gbps to hundreds of Gbps. Most of the campaigns observed include signatures from multiple types of malware and their associated botnets as well as BillGates.

Our most recently observed attack campaign, consisting only of traffic from the BillGates Malware in 2015, peaked around 6.5 Gbps — achieving nearly 1 Million packets per second.

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

Figure 11: Mitigated DDoS campaign measurements targeting an Akamai customer

The biggest attack campaign observed, including malicious traffic from the BillGates botnet along with other various attack vectors, was on December 30, 2015 and had a well-distributed peak bandwidth of about 308 Gbps across our scrubbing centers.

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

Figure 12: Largest observed attack that utilized BillGates

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

Figure 13: Sample malicious packet from mitigated DDoS campaigns

The BillGates malware has the ability to spoof the source address from infected machines; however, from our investigation we can confirm this is not commonly observed in the attack traffic, and the attacking source IPs are in fact the actual infected machines. This is likely due to an inability to route spoofed traffic from the infected host's network.

5.0 / INDICATORS OF BINARY INFECTION / First, check for the presence of the files **`gates.lod`** and **`moni.lod`** in your **`/tmp`** directory. If they exist, the content of **`gates.lod`** will be the process id of the main malware and the content of **`moni.lod`** is the process id of the process responsible for the safeguarding/persistence of the malware's primary process.

```
$ 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
```

Figure 14: Indication of Infection: PID files linked to infection

We can see the associated malicious files of these processes. (Remember, don't use the ps utility from **`/bin`** because the original utility is substituted with a copy of the malware.) The original utility is in **`/usr/bin/dpkgd/ directory`**)

```
$ cat /proc/13640/cmdline
/usr/bin/.sshd
$ cat /proc/13550/cmdline
/home/user/Desktop/billgates_variant
```

Figure 15: Finding the execution path of the associated PID

Another indication is if you have the following startup scripts in your **/etc/init.d/** directory.

```
$ 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
```

Figure 16: Indication of infection: **/etc/init.d** startup scripts

Consequently, there's going to be a symbolic link to these startup scripts for each run level in the **/etc/rc[run-level-number].d/** directories.

```
$ 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
```

Figure 17: Indications of Infection: symlinks to **/etc/init.d/DbSecuritySpt** in **/rcX.d/** directories

We can check the hashes of the utility binaries **ps**, **lsof**, **netstat**, and **ss** and confirm they are identical to the other confirmed infections.

```
$ 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
```

Figure 18: Indication of Infection: Hashes of infected files and core system tools

As you can see, the hashes are identical for all binaries, meaning the malware has successfully replaced these core system tools with copies of itself.

To disinfect your system, first kill the primary process **id** found in the file **/tmp/moni.lod**. By killing this safe-guarding process first, we ensure the malware will not restart itself when we kill the main process. Next, kill the process id found in the file **/tmp/gates.lod**. As a third step, we have to kill the process id found in the file **/usr/bin/bsd-port/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
```

Figure 19: Killing the BillGates processes in order to disinfect the machine

As a final step, we have to delete all copies of the malware and startup scripts.

```
$ 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
```

Figure 20: Removing malware and associated persistence measures

6.0 / RECOMMENDED DETECTION METHODS / The following steps are designed to assist organizations in identifying the BillGates malware on systems.

6.1 / DETECTING NETWORK TRAFFIC / To detect the network traffic from the initial communication bot to C2, we can use the following **tcpdump** filters:

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

Figure 21: Tcpdump filter that can be used to find BillGates C2 communications

This filter is of two parts. The first part is looking within the first 4 bytes of the payload for the bytes 0x01000000, which are found in the "register" command. The second part is looking for the **ASCII :G** and counting 7 bytes backwards from the end of the payload.

Part 1: **'tcp[((tcp[12] >> 4) << 2):4] == 0x01000000'**

Part 2: **'ip[(ip[2:2] - 7):2] == 0x3a47'**

To capture the heartbeat from the C2 to the bot, there are many parts of the packet that change and could generate false positives and false negatives. This filter is a bit more involved to prevent false hits.

```
'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]'
```

Figure 22: Tcpdump filter that can be used to find BillGates C2 heartbeats

This filter is based on two parts as well. The first part is detecting the command from the C2 within the first 4 bytes of the tcp payload. The second part is calculating the size of the payload based on the command issued. It sums the size of the IP Header's length + TCP Header's length + size of payload, as instructed by the C2 + 8, which needs to match the IP header's total packet length.

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]'
```

6.2 / YARA RULES FOR DETECTING INFECTION / There are pre-existing rules for detecting the BillGates malware from the Yara community. However, just for good measure, we built another additional rule that can be used.

```
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
}
```

Figure 23: Yara rule used to identify BillGates malware

7.0 / CONCLUSION / The BillGates malware is capable of launching Layer 3/4- and Layer 7-based DDoS attacks. BillGates botnets have grown significantly and are large enough to launch 100 + Gbps of attack traffic independently but are also used in conjunction with other DDoS frameworks. They appear to attack the same target industries as botnets built using the XOR malware and are very active in targeting Asia-based companies and their digital properties.

There is a possibility that after the takedown of XOR botnet, the malware actors began using different means and/or different botnets to continue their onslaught of attacks directed at the same primary group of targets. This awareness of activity observed by Akamai over the last 6 months has warranted the release of this advisory.

Disclaimer:

The malware was named after Microsoft's former CEO, Bill Gates based on the fact that it targets Linux machines instead of Windows. However, the malware is not affiliated in any way with Microsoft Corporation or the Founder, Bill Gates.

Akamai FASTER FORWARD

About Akamai Security Intelligence Research Team (SIRT) Focuses on mitigating malicious global cyber threats and vulnerabilities, the Akamai Security Intelligence Research Team (SIRT) conducts and shares digital forensics and post-event analysis with the security community to proactively protect against threats and attacks. As part of its mission, the Akamai SIRT maintains close contact with peer organizations around the world and trains Akamai's Professional Services and Customer Care teams to both recognize and counter attacks from a wide range of adversaries. The research performed by the Akamai SIRT is intended to help ensure Akamai's cloud security products are best of breed and can protect against any of the latest threats impacting the industry.

About Akamai® As the global leader in Content Delivery Network (CDN) services, Akamai makes the Internet fast, reliable and secure for its customers. The company's advanced web performance, mobile performance, cloud security and media delivery solutions are revolutionizing how businesses optimize consumer, enterprise and entertainment experiences for any device, anywhere. To learn how Akamai solutions and its team of Internet experts are helping businesses move faster forward, please visit www.akamai.com or blogs.akamai.com, and follow @Akamai on Twitter.

Akamai is headquartered in Cambridge, Massachusetts in the United States with operations in more than 57 offices around the world. Our services and renowned customer care are designed to enable businesses to provide an unparalleled Internet experience for their customers worldwide. Addresses, phone numbers and contact information for all locations are listed on www.akamai.com/locations.