The Gafgyt variant vbot seen in its 31 campaigns

blog.netlab.360.com/the-gafgyt-variant-vbot-and-its-31-campaigns/

LIU Ya July 6, 2020

6 July 2020

Overview

Gafgyt botnets have a long history of infecting Linux devices to launch DDoS attacks. While dozens of variants have been detected, new variants are constantly emerging with changes in terms of register message, exploits, and attacking methods. On the other hand, their new botnets are usually short lived, with most of the C2s watched keeping active for only a few days. In this blog, I will introduce such a sort of variant. The key findings are as follow:

1. This variant was active from mid-April to mid-June. In total 31 campaigns for this variant were detected, from which 572 samples were captured. They were spread to build 19 botnets.
2. This variant evolved through 2 versions. Both have a characteristic register message template “ver:%f:%s:%d” that includes a rarely seen format specifier “%f”.
3. Mirai code was heavily used in both versions, which makes it possible analyze them with the extracted Mirai configurations.
4. The same infrastructures, e.g., download servers, and filenames were observed being used in other families of botnet campaigns.
   This variant was named as vbot because vbot is found being used in an unstripped sample by the author. Accordingly the 2 versions are named as vbot1 and vbot2 in this blog.

vbot1

Only 1 vbot1 campaign was seen, with 26 samples captured, as shown by the following honeypot records.

time	yara	md5	down_server	filename
20-04-15 05:11:48+08:00	vbot_v1	2a141cd2930536f74f51fb57adbb0236	185.225.19.200	RHOMBUS
20-04-15 05:11:53+08:00	vbot_v1	8717baf17660d8e96813ccd99f32c0be	185.225.19.200	RHOMBUS
20-04-15 05:12:00+08:00	vbot_v1	cc559b487e1ec18727f37006bd3395e0	185.225.19.200	RHOMBUS
20-04-15 05:12:09+08:00	vbot_v1	f666c3398601cd1b017f8d4556cabbbc	185.225.19.200	RHOMBUS
20-04-15 05:12:18+08:00	vbot_v1	6fb6aaa253c165636ee63a4fdcdb1b9e	185.225.19.200	RHOMBUS
20-04-15 05:12:18+08:00	vbot_v1	f422707ac869240bfeea648b6f9b90ad	185.225.19.200	RHOMBUS
20-04-15 05:12:28+08:00	vbot_v1	36997fd129a5ff09311da94c3814379c	185.225.19.200	RHOMBUS
20-04-15 05:12:28+08:00	vbot_v1	790ae71c097662bf6efba92d2d633076	185.225.19.200	RHOMBUS
20-04-15 05:12:39+08:00	vbot_v1	e420df68941cc7ce2d8dd4ba92fd360e	185.225.19.200	RHOMBUS
20-04-15 05:12:49+08:00	vbot_v1	3e36440871a6e39ee87e6d7d1a42155a	185.225.19.200	RHOMBUS
20-04-15 05:12:49+08:00	vbot_v1	ae50829a02e5265c590f2fff35e64c52	185.225.19.200	RHOMBUS
20-04-15 05:12:58+08:00	vbot_v1	09ab7435c76df627a813fb75db15ce5d	185.225.19.200	RHOMBUS
20-04-15 05:12:58+08:00	vbot_v1	43ee98318945a475b555045aed4f0e01	185.225.19.200	RHOMBUS
20-04-15 06:15:38+08:00	vbot_v1	e4db8addb5123021e358576157e5e1c0	185.225.19.200	RHOMBUS
20-04-15 06:15:44+08:00	vbot_v1	4147fb0fe442173558f86fe37728ecae	185.225.19.200	RHOMBUS
20-04-15 06:15:53+08:00	vbot_v1	846d6ad9ea86e331f2e071eac6a269de	185.225.19.200	RHOMBUS
20-04-15 06:16:02+08:00	vbot_v1	40b1bf1e415ae508f8a5b831c2f4e994	185.225.19.200	RHOMBUS
20-04-15 06:16:02+08:00	vbot_v1	f696375452d08eecbde14d64c74acdde	185.225.19.200	RHOMBUS
20-04-15 06:16:23+08:00	vbot_v1	98b07b087b98b8d679c9938b16ae4df3	185.225.19.200	RHOMBUS
20-04-15 06:16:23+08:00	vbot_v1	aea960687f0e43b465198be7ffafcf82	185.225.19.200	RHOMBUS
20-04-15 06:16:34+08:00	vbot_v1	3d596d37fe6536a2c759923d920f3e08	185.225.19.200	RHOMBUS
20-04-15 06:16:36+08:00	vbot_v1	52c462f3b22646774219f91bfb44ae66	185.225.19.200	RHOMBUS
20-04-15 06:16:44+08:00	vbot_v1	d2c273e758fd4ac2759ca1d63aafcf6c	185.225.19.200	RHOMBUS
20-04-15 06:16:52+08:00	vbot_v1	bbee73ed05730ad95df7a77241207ea5	185.225.19.200	RHOMBUS
20-04-15 06:16:53+08:00	vbot_v1	0f492673eb249fa1209512575040f62d	185.225.19.200	RHOMBUS
20-04-15 06:16:57+08:00	vbot_v1	0e59d4a40bba390314ffa0713b18441c	185.225.19.200	RHOMBUS
20-04-16 17:27:37+08:00	vbot_v2	efabd7e734490b9ad12812982347f237	185.225.19.200	Slsmodsd
20-04-16 17:27:43+08:00	vbot_v2	614581bba324c3550a18268a8cb9c221	185.225.19.200	Slsmodsd
20-04-16 17:27:51+08:00	vbot_v2	86310b514c55d31db288a2bb2c1e6114	185.225.19.200	zte

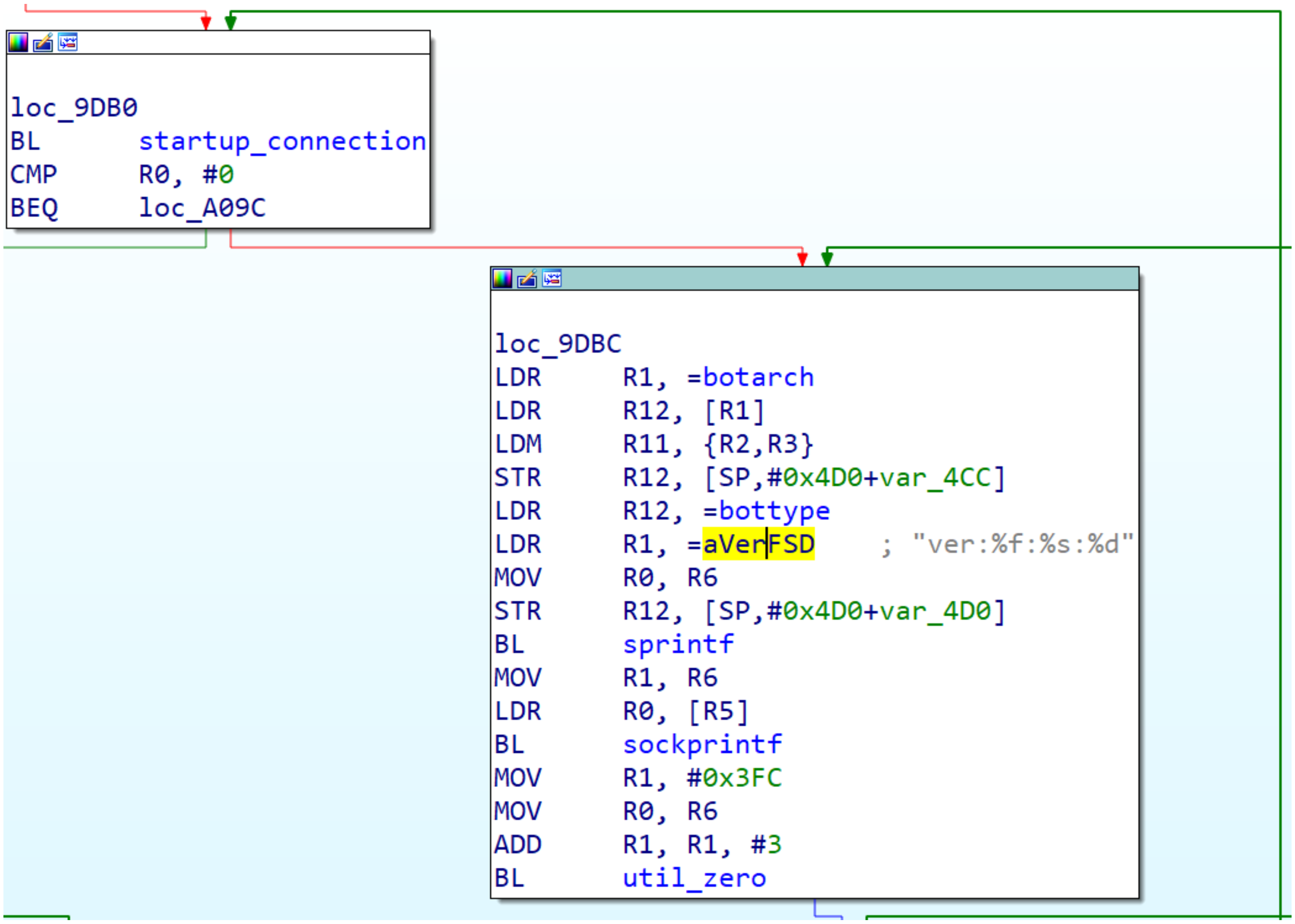
vbot v1

vbot v2

All samples share the same C2 `185.225.19.200:2017`. Since in Gafgyt it's common that the same source code will be compiled into binaries for different processor architectures, for simplicity, the following analysis is based on the unstripped ARM sample of `f696375452d08eecbde14d64c74acdde`. Compared with previous variants, vbot1 has a more concise main() function because most of its code was moved into 2 new functions named init_vbot() and main_c2_handler().

```
loc_A294
BL      init_vbot
BL      main_c2_handler
MOV     R0, #0
ADD     SP, SP, #4
ADD     SP, SP, #0x400
POP     {R4-R7,LR}
BX      LR
```

The function name init_vbot indicates that the author code named their botnet as vbot. It's responsible for initializing things including watchdog, configurations, and scanner. C2 communications are done in main_c2_handler(), where a loop of connection, registration and receiving command can be found, as shown below.



The characteristic register message template `“ver:%f:%s:%d”` is used in the registration block that tightly follows the connection block. From the unstripped symbols we can show that the 3 specifiers separately represent version, bot type and arch. The analyzed sample has version of 4.1.

Actually it’s just the rarely seen specifier `“%f”` that caused my attention to this variant because as far as I knew “%f” was not supported by Gafgyt. The original authors borrowed the design of C library functions `printf` and `sprintf`, and implemented a new function named `sockprintf` which can generate message according to the assigned string format and send it to the C2. A custom yet simple format controls is done inside sockprintf with “%f” not implemented. That function has been kept by most Gafgyt variants. When firstly encountering vbot’s register template, I imagined a new version of sockprintf. However, that’s obviously not true. To reuse sockprintf but avoid complex programming, vbot author turned to sprintf to generate the expected message then passed it to sockprintf with the supported specifier `“%s”`.

Similar to many Gafgyt variants, Mirai code can be found in vbot1. Due to its tight connection with the encrypted configurations, the borrowed code can be well analyzed with the extracted configurations. If you don’t know how to extract, please go to our VB2018 paper. The extracted configurations are shown below, with items annotated with its owner modules.

```
[0x00]: "1gba4cdom53\x00", size=12
[0x01]: "/dev/watchdog\x00", size=14
[0x02]: "/dev/misc/watchdog\x00", size=19
[0x03]: "/dev/FTWDT101_watchdog\x00", size=23
[0x04]: "/dev/FTWDT101\ watchdog\x00", size=24
[0x05]: "/dev/watchdog0\x00", size=15
[0x06]: "/etc/default/watchdog\x00", size=22
[0x07]: "/sbin/watchdog\x00", size=15
[0x08]: "shell\x00", size=6
[0x09]: "enable\x00", size=7
[0x0a]: "system\x00\x17", size=8
[0x0b]: "sh\x00", size=3
[0x0c]: "echo "check"\x00", size=13
[0x0d]: "check\x00", size=6
[0x0e]: "assword\x00", size=8
[0x0f]: "ogin\x00", size=5
[0x10]: "enter\x00", size=6
[0x11]: "ccount\x00", size=7
[0x12]: "ser\x00", size=4
[0x13]: "ncorrect\x00\x17", size=10
[0x14]: "nvalid\x00", size=7
[0x15]: "ncomplete\x00", size=10
[0x16]: "attempt failed\x00", size=15
[0x17]: "IVEBEENEXECUTED\x00", size=16
[0x18]: "GET\x00", size=4
[0x19]: "UDP\x00", size=4
[0x1a]: "ASTD\x00", size=5
[0x1b]: "TCP\x00", size=4
[0x1c]: "GRE\x00", size=4
[0x1d]: "AHTTP\x00", size=6
[0x1e]: "KT\x00", size=3
[0x1f]: "UPDATE\x00", size=7
[0x20]: "execnew\x00", size=8
[0x21]: "./execnew\x00", size=10
[0x22]: "mv\x00", size=3
[0x23]: "/tmp/vbot.exe\x00", size=14
[0x24]: "/etc/init.d/.vbot.sh\x00", size=21
[0x25]: "/etc/rc.d/rc.local\x00", size=19
[0x26]: "/etc/rc.local\x00", size=14
[0x27]: "PING\x00", size=5
[0x28]: "PONG\x00", size=5
[0x29]: "KILL\x00", size=5
[0x2a]: "/proc/\x00", size=7
[0x2b]: "/exe\x00", size=5
[0x2c]: "/fd\x00", size=4
[0x2d]: "/proc/net/tcp\x00", size=14
[0x2e]: "cnc.kowaiontop.xyz\x00", size=19
```

random_string_generation

watchdog

scanner

command

persistence

killer

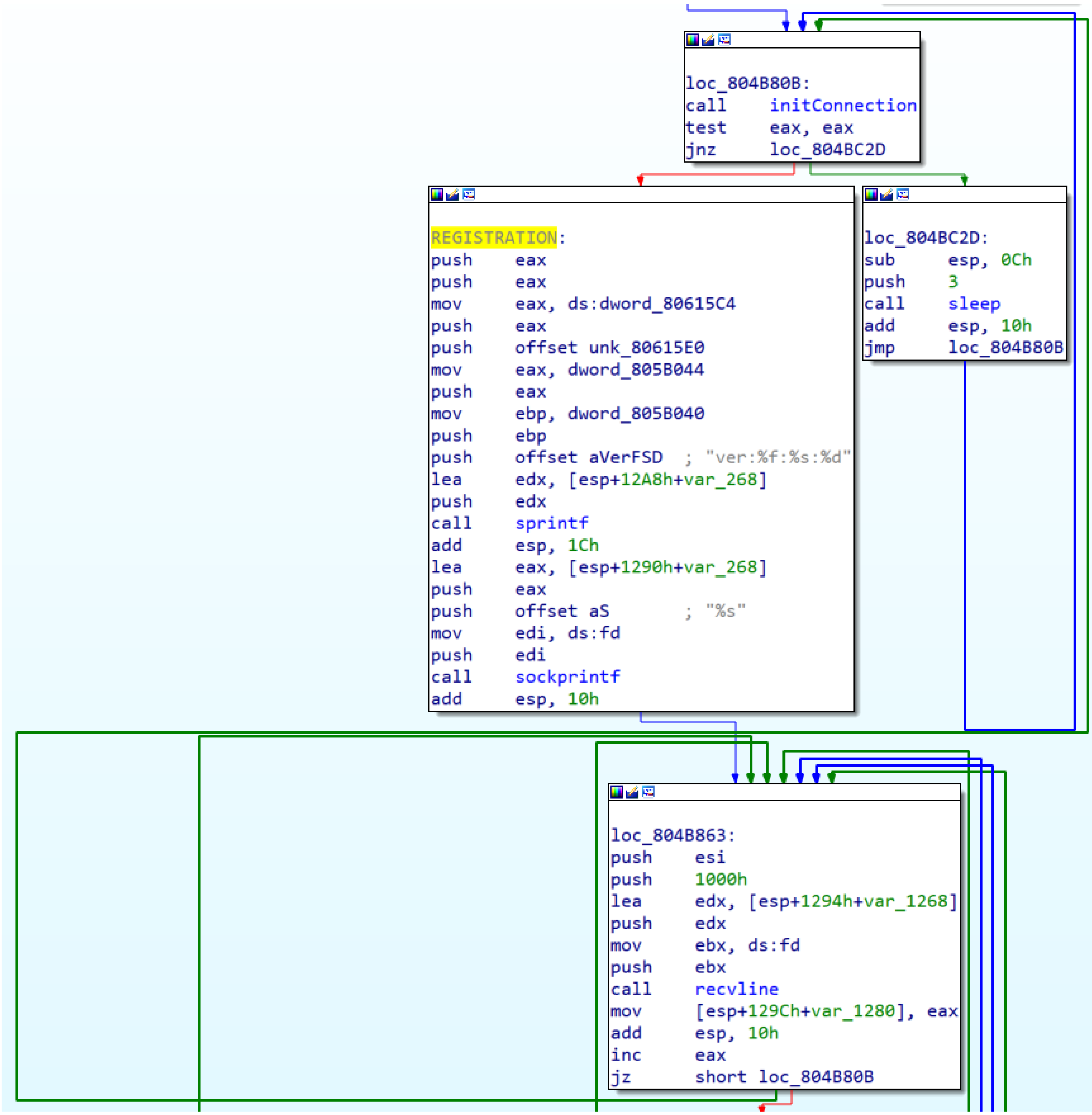
The commands are hidden in configurations. Except for attacking methods, vbot1 also supports remote update with the command `UPDATE`. Another worth mentioning feature is persistence mechanism, which is done by modifying crontab.

It’s strange that vbot1 was spread only once. After its campaign was firstly detected, 35 hours, or 1.5 days, later the first vbot2 campaign was seen from the same download server. Obviously the operators wanted to replace vbot1 with vbot2. The reason might be its buggy registration which always sends a 191-byte register message back to its C2 but only 18 bytes there are really useful, as shown by the following figure.

```
00000000  76 65 72 3a 34 2e 31 30  30 30 30 30 3a 31 31 3a  |ver:4.100000:11:|
00000010  30 0a be f1 fc f9 79 6b  52 14 13 e9 e2 2d 51 8e  |0.....ykR....-Q.|
00000020  1f 56 08 57 27 a7 05 d4  d0 52 82 77 75 1b 99 4a  |.V.W'....R.wu..J|
00000030  ed 58 3d 6a 52 36 d5 24  4a 68 8e ad 95 5f 3c 35  |.X=jR6.$Jh..._<5|
00000040  b5 c4 8c dd 6c 11 32 3d  e2 b4 b4 59 cf ce 23 3d  |....l.2=...Y..#=|
00000050  27 df a7 f9 96 fc 1e e0  66 2c 0e 7b 8c ca 30 42  |'.......f,.{..0B|
00000060  8f bc 9f 7b ce d1 b8 b1  87 ec 8a d6 bb 2e 15 63  |...{...........c|
00000070  0e 3c dc a4 3a 7a 06 20  a7 93 1b 34 dd 4c f5 ec  |.<..:z. ...4.L..|
00000080  88 96 68 d6 68 a0 09 6f  8e 93 47 c9 41 db ac cf  |..h.h..o..G.A...|
00000090  97 89 f3 51 05 79 71 2c  0e 0d 60 6b 59 d5 59 e1  |...Q.yq,..`kY.Y.|
000000a0  6c c1 b9 55 63 42 44 71  55 0b ba 97 e6 68 67 fe  |l..UcBDqU....hg.|
000000b0  71 5b 50 76 55 c2 22 63  4f 02 ce a8 d8 a7 0a     |q[PvU."cO......|
000000bf
```

vbot2

In total 30 vbot2 campaigns were seen from April 16 to June 12, 2020, with 546 samples captured from 12 download servers. From those samples 13 C2 servers were checked. Detailed analysis shows except the registration code, vbot2 actually differs a lot from vbot1 in terms of code structure, attacking methods and Mirai configuration. The following analysis is based on the x86 sample `f5b0ebebc924e69e34a4ddd145916594`. It’s stripped but key function names have been manually restored.
Different from vbot1 but similar to many other variants, vbot2's C2 communications are done in main(), as shown below.



Nearly the same registration block as vbot1 can be found, with the 3 specifiers holding the same semantics. The analyzed sample has version of 1.5. The loop composed of "loc_804B80B -> REGISTRATION -> loc_804B863" is very similar to previous Gafgyt variants in terms of CFG node number and semantics. The blocks are separately responsible for establishing connection, registration, and receiving commands.

5 attacking methods were checked. All of them have been seen in other variants.

```
.rodata:080587BD aUdpBypass      db 'UDP-BYPASS',0       ; DATA XREF: processCmd+A↑o
.rodata:080587C8 aTcpBypass      db 'TCP-BYPASS',0       ; DATA XREF: processCmd+C1↑o
.rodata:080587D3 asc_80587D3     db ',',0                ; DATA XREF: processCmd+1BF↑o
.rodata:080587D3                                         ; processCmd+282↑o ...
.rodata:080587D5 aNfoDrop        db 'NFO-DROP',0         ; DATA XREF: processCmd+2A9↑o
.rodata:080587DE aOvhKill        db 'OVH-KILL',0         ; DATA XREF: processCmd+429↑o
.rodata:080587E7 aUdpRape        db 'UDP-RAPE',0         ; DATA XREF: processCmd+595↑o
```

Some vbot2 samples, e.g., `e36d96a74236038a348cfd667ca83528`, have slightly different attacking method names, as shown below.

```
data:080571C4 aUdp            db 'UDP',0           ; DATA XREF: processCmd+7↑o
data:080571C8 aTcp            db 'TCP',0           ; DATA XREF: processCmd+B6↑o
data:080571CC asc_80571CC     db ',',0             ; DATA XREF: processCmd+1B4↑o
data:080571CC                                      ; processCmd+271↑o ...
data:080571CE aStd            db 'STD',0           ; DATA XREF: processCmd+298↑o
data:080571D2 aUdpRape        db 'UDP-RAPE',0      ; DATA XREF: processCmd+3FC↑o
data:080571DB aStop           db 'STOP',0          ; DATA XREF: processCmd+56E↑o
```

2 Mirai configurations were found. The only difference lies in the 0x28 item, as shown by the following 2 figures.

```
[0x01]: "C\", size=2
[0x02]: "(null)\x00", size=7                          sockprintf
[0x03]: "/dev/watchdog\x00", size=14
[0x04]: "/dev/misc/watchdog\x00", size=19                 watchdog
[0x05]: "/dev/watchdog0\x00", size=15
[0x06]: "/bin/watchdog\x00", size=14
[0x07]: "/etc/watchdog\x00", size=14
[0x0a]: "shell\x00", size=6
[0x0b]: "enable\x00", size=7                                    scanner
[0x0c]: "system\x00", size=7
[0x0d]: "linuxshell\x00", size=11
[0x0e]: "bah\x00", size=4
[0x0f]: "sh\x00", size=3
[0x10]: "ncorrect\x00", size=9
[0x11]: "nvalid\x00", size=7
[0x12]: "ogin\x00", size=5
[0x13]: "ame\x00", size=4
[0x14]: "ccount\x00", size=7
[0x15]: "enter\x00", size=6
[0x16]: "assword\x00", size=8
[0x17]: "/bin/busybox echo "test"\x00", size=25
[0x18]: "test\x00", size=5
[0x19]: "/proc/\x00", size=7
[0x1a]: "/exe\x00", size=5
[0x1b]: "/fd\x00", size=4
[0x1c]: "/maps\x00", size=6
[0x1d]: "/proc/net/tcp\x00", size=14                     killer
[0x1e]: "0\x16\x00\x17H$\x02\x00", size=8
[0x1f]: "/dev/null\x00", size=10
[0x20]: "STD\x00", size=4
[0x21]: "/proc/net/route\x00", size=16
[0x22]: "/proc/net/tcp\x00", size=14
[0x23]: "/proc/self/exe\x00", size=15
[0x24]: "UPX!\x00", size=5
[0x25]: "/proc/net/route\x00", size=16
[0x26]: "/etc/rc.d/rc.local\x00", size=19
[0x27]: "/bin/sh\x00", size=8                             random_string_generation
[0x28]: "ya that high keeps me alive\x00", size=28
[0x29]: "qC8cVuGTnRH6cfv7sjcYPFv7guAmZxbQRc57fV77IUUj5b6wocpfFJPmHC\x00", size=59
```

```
[0x28]: "-\x0a\x02\x01\x07\x10\x01\x00", size=8
```

From the annotations we can see the Mirai code was mainly used in modules of watchdog, killer, scanner and rand alpha string generation. Since the 0x28 item corresponds to a message to be written to the STDOUT, and the second unprintable 0x28 item is probably

caused by a typo from the author.
With the extracted configurations the differences from vbot1 are obvious. They are:

1. vbot2 has different attacking methods from vbot1.
2. While vbot1 hides commands in its configuration, vbot2 directly uses them.
3. No remote update and persistence mechanism were found in vbot2.

Although those great differences suggest that vbot1 and vbot2 were actually derived from different code bases, I still think they were written by the same author(s) because:

1. The shared register message template and registration implementation are unique enough.
2. The first vbot2 campaign shared the same download and C2 server as vbot1 within a relatively short period of time (1.5 days).

vbot and the RHOMBUS malware

While the filename RHOMBUS was seen 4 times in vbot campaigns, its use in Gafgyt campaigns was much earlier[1], with the variant called RHOMBUS analysed in [2][3]. Here I make a simple comparison. In the blogged RHOMBUS malware dropper mechanism was found, with the dropper having the persistence ability across restart by modifying crontab. The dropped binaries, e.g., `269029c1554b13c3eccfaacf0196ff72` and `ba42665872ea41e3d2edd8978bc38c24`, actually belong to another Gafgyt variant that also heavily borrowed code from Mirai, as shown by the below figure.

```
[0x00]: "1gba4cdom53\x00", size=12        random string generation
[0x01]: "/dev/watchdog\x00", size=14
[0x02]: "/dev/misc/watchdog\x00", size=19
[0x03]: "/dev/FTWDT101_watchdog\x00", size=23     watchdog
[0x04]: "/dev/FTWDT101\ watchdog\x00", size=24
[0x05]: "/dev/watchdog0\x00", size=15
[0x06]: "/etc/default/watchdog\x00", size=22
[0x07]: "/sbin/watchdog\x00", size=15
[0x08]: "shell\x00", size=6
[0x09]: "enable\x00", size=7
[0x0a]: "system\x00\x17", size=8
[0x0b]: "sh\x00", size=3
[0x0c]: "echo "check"\x00", size=13
[0x0d]: "check\x00", size=6
[0x0e]: "assword\x00", size=8                 scanner
[0x0f]: "ogin\x00", size=5
[0x10]: "enter\x00", size=6
[0x11]: "ccount\x00", size=7
[0x12]: "ser\x00", size=4
[0x13]: "ncorrect\x00\x17", size=10
[0x14]: "nvalid\x00", size=7
[0x15]: "ncomplete\x00", size=10
[0x16]: "attempt failed\x00", size=15
[0x17]: "IVEBEENEXECUTED\x00", size=16
[0x19]: "GET\x00", size=4
[0x1a]: "shell\x00", size=6
[0x1b]: "KILLBOT\x00", size=8
[0x1c]: "UDP\x00", size=4
[0x1d]: "TCP\x00", size=4                     command
[0x1e]: "GRE\x00", size=4
[0x1f]: "THREADHTTP\x00", size=11
[0x20]: "KT\x00", size=3
[0x21]: "UPDATE\x00", size=7
[0x22]: "EgrsFc\x00", size=7
[0x23]: "chmod +x EgrsFc; ./EgrsFc\x00", size=26
[0x24]: "cnc.kowaiontop.xyz\x00", size=19
```

From the above configuration we can see that obvious similarities exist between the RHOMBUS dropped binaries and vbot1. I think the most possibility is that vbot1 evolved from RHOMBUS malware with the following modifications:

1. The dropper's persistence mechanism was grafted to its payload. That's why persistence items could be found in vbot1 configuration but not in the above figure.
2. The register template was updated.
3. c2 communications were moved to the so called main_c2_handler() function.

Other key points about RHOMBUS malware include:

1. The register message template is "jm:_:%d" or jm:%s:%d.
2. Similar to many Gafgyt variants, C2 communications were done in main().

3. The Gafgyt characteristic function initConnection() was removed with its code broken down into snippets that can be found in main().

Conclusion

I have introduced a short lived Gafgyt variant vbot. During its 2 month life, 31 campaigns were seen to build 19 botnets. From vbot we can learn that it's easy for Linux IoT botnet authors to quickly write new variants, which might be due to the fact that dozens of Gafgyt and Mirai source has been leaked online. Once a new variant is written, the behind operators usually will spread it over and over with different campaigns to build multiple botnets. Such patterns have also been observed in other variants and families, e.g., Mirai. To fight such sort of fast emerging while short living botnets, automatic IoC extraction would play an import role for quick blocking or tracking. In VB2020 conference to be held in October, I will give a talk on that topic. I hope it will help you fight against Gafgyt botnets better.

IoC

download servers

```
104.244.75.12
142.11.194.209
185.172.110.248
185.172.110.249
185.225.19.200
192.119.66.66
192.129.188.98
205.185.123.101
23.254.164.76
45.84.196.148
50.115.173.131
85.92.108.211
```

vbot1 MD5

```
2a141cd2930536f74f51fb57adbb0236
8717baf17660d8e96813ccd99f32c0be
cc559b487e1ec18727f37006bd3395e0
f666c3398601cd1b017f8d4556cabbbc
6fb6aaa253c165636ee63a4fdcdb1b9e
f422707ac869240bfeea648b6f9b90ad
36997fd129a5ff09311da94c3814379c
790ae71c097662bf6efba92d2d633076
e420df68941cc7ce2d8dd4ba92fd360e
3e36440871a6e39ee87e6d7d1a42155a
ae50829a02e5265c590f2fff35e64c52
09ab7435c76df627a813fb75db15ce5d
43ee98318945a475b555045aed4f0e01
e4db8addb5123021e358576157e5e1c0
4147fb0fe442173558f86fe37728ecae
846d6ad9ea86e331f2e071eac6a269de
40b1bf1e415ae508f8a5b831c2f4e994
f696375452d08eecbde14d64c74acdde
98b07b087b98b8d679c9938b16ae4df3
aea960687f0e43b465198be7ffafcf82
3d596d37fe6536a2c759923d920f3e08
52c462f3b22646774219f91bfb44ae66
d2c273e758fd4ac2759ca1d63aafcf6c
bbee73ed05730ad95df7a77241207ea5
0f492673eb249fa1209512575040f62d
0e59d4a40bba390314ffa0713b18441c
```

vbot1 C2

```
185.225.19.200 -port 2017
```

vbot2 MD5

```
efabd7e734490b9ad12812982347f237
614581bba324c3550a18268a8cb9c221
86310b514c55d31db288a2bb2c1e6114
76d9c69036f1eaac8f7a90eba3a36bfc
e36d96a74236038a348cfd667ca83528
d45da804fd35cf502bf942ebfeb64064
90a633f30bdbb2b80642bb229d1605d1
c4391301645cc9df4da3657f4c88f7dc
8bef47e420d0cdf8d0ee69a5d1f5b74c
4c8cdcbaf16f39a461b0bf7052fe1ec3
d936a9226fbbe97993bbe604c8cd5458
125b99cc79808679a7461f1841fd80a5
3b7da3d39db6ec08373c1e4af79aff85
23f764f5f918746b9ffff952dd25cc21
6f24268273573fd5f07cacb00031f1a0
ecae928b4e4093489bd221986da39aba
d883d5a2bedf0c3a3da79358c06fa429
3e26626d4563f3199fde498d0ff9fe32
11c1d777b18ffc0f23d2435fdb4645dc
...
```

vbot2 C2

```
104.244.75.12_666
142.11.194.209_1337
142.11.194.209_17911
142.11.194.209_34
142.11.194.209_44
184.172.110.248_666
184.172.110.249_666
185.172.110.248_323
185.172.110.248_666
185.225.19.200_666
192.119.66.66_7331
192.129.188.98_323
205.185.123.101_666
23.254.164.76_107
23.254.164.76_33
23.254.164.76_89
45.84.196.148_1227
50.115.173.131_111
85.92.108.211_1447
```