Container Detection and Forensics, ‘Gotta catch them all’!

Cem Gürkök
Detection Infra
Summary

- Docker?
- What is osquery and Docker capabilities
- What is Volatility and Docker capabilities
- Use cases
- How to detect with osquery and Volatility
- Conclusion
Containers?

“Containers are processes born from tarballs anchored to namespaces controlled by cgroups”

-- Alice Goldfuss
What?

- **Control Groups (cgroups)** provide a mechanism for limiting resources (i.e. mem, cpu) for a hierarchy or set of tasks.

- **Namespaces** partition kernel resources (isolation: i.e. mnt, pid), hence containers.
Docker Containers?

- A container platform/runtime
- Runs on Linux, Mac, Windows, various clouds (i.e. AWS)
- Large ecosystem of infra, devs, apps

Docker Components

Client
- `docker build`
- `docker push`
- `docker run`

Docker Host
- Daemon
- Containers
- Images

Registry
- Repositories
- Notary
Why care about Docker Forensics?

- The reality of things:
  - Latest applications, including enterprise, use Docker
  - New hires most probably used Docker for dev + CI/CD
  - The momentum will carry over...
Why care about Docker Forensics?

- Risks:
- Minimal visibility
- Container breakouts [CVE-2016-5195, CVE-2017-5123, CVE-2014-9357]
- Leverage as persistence and lateral movement
- Source poisoning (bad public images)
- Vulnerabilities (exposed as users and as a provider)
- IP, cred. and data leaks as images get pushed to public repos
Cool… what tooling?
osquery

- An operating system instrumentation, monitoring, and analytics framework
- Live monitoring
- OSX/macOS, Windows, and Linux
- SQL powered
- More at osquery.io
osquery: Docker Forensics Capabilities

- `docker_*` tables
- Equivalent to ‘docker inspect’
  - network, volumes, environment variables, security, and more
- Linux and Mac visibility at the moment
- Leverages the Docker engine API socket
- Image and container information
The Volatility Framework

- A framework to extract digital artifacts from volatile memory (RAM) samples
  - Paux, psenv, mount, ifconfig, netstat, recovered_files
- Linux, Windows, OS X/macOS
- Python powered
- More at volatilityfoundation.org
Tool Use

- osquery mostly for continuous monitoring
- Volatility Framework mostly for incident response
- Both great for detection
## Containers in the Mitre Att&ck Kill Chain and Tool Coverage

<table>
<thead>
<tr>
<th>Vector/Scenarios</th>
<th>Initial Access</th>
<th>Execution</th>
<th>Persistence</th>
<th>Privilege Escalation</th>
<th>Defense Evasion</th>
<th>Credential Access</th>
<th>Discovery</th>
<th>Lateral Movement</th>
<th>Collection</th>
<th>Exfiltration</th>
<th>Command and Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitoring, Detection, and IR</td>
<td>Bad Image, breakout</td>
<td>Rogue Container Launch</td>
<td>Enable restart</td>
<td>Privileged mode via breakout</td>
<td>Masquerade</td>
<td>Env variables, configs</td>
<td>Other containers, network</td>
<td>Other containers or other resources on host + network</td>
<td>Host or network resources (db, fs)</td>
<td>As image push or regular exfil routes</td>
<td>Remote access, obfuscation</td>
</tr>
</tbody>
</table>

Monitoring, Detection, and IR: Monitoring, detection, and incident response.
Use Cases
Use Case: Container Breakout

- Vulnerable Python Web Application
- Running as *privileged* or *no seccomp* profile
- On compromise, provides the attacker with root level access
## Use Case: Container Breakout

### docker_containers table

<table>
<thead>
<tr>
<th>name</th>
<th>pid</th>
<th>command</th>
<th>privileged</th>
<th>security_options</th>
</tr>
</thead>
<tbody>
<tr>
<td>/vulnwebserver</td>
<td>10423</td>
<td>python -m SimpleHTTPServer 80</td>
<td>1</td>
<td>label=disable</td>
</tr>
<tr>
<td>/vulnappdb</td>
<td>11458</td>
<td>docker-entrypoint.sh mysqld</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Use Case: Container Breakout

**linux_pstree plugin, uid 999(docker)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Pid</th>
<th>PPid</th>
<th>Uid</th>
</tr>
</thead>
<tbody>
<tr>
<td>.dockerd</td>
<td>4043</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>...docker-containe</td>
<td>11436</td>
<td>4073</td>
<td></td>
</tr>
<tr>
<td>....mysql</td>
<td>11458</td>
<td>10255</td>
<td>999</td>
</tr>
<tr>
<td>..docker-containe</td>
<td>4059</td>
<td>4043</td>
<td></td>
</tr>
<tr>
<td>...docker-containe</td>
<td>10397</td>
<td>4059</td>
<td></td>
</tr>
<tr>
<td>....python</td>
<td>10423</td>
<td>10397</td>
<td></td>
</tr>
<tr>
<td>....nc</td>
<td>14521</td>
<td>10397</td>
<td></td>
</tr>
</tbody>
</table>

**linux_psaux plugin**

<table>
<thead>
<tr>
<th>Pid</th>
<th>Uid</th>
<th>Gid</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>10423</td>
<td>0</td>
<td>0</td>
<td>python -m SimpleHTTPServer 80</td>
</tr>
<tr>
<td>11458</td>
<td>999</td>
<td>999</td>
<td>mysql</td>
</tr>
<tr>
<td>14521</td>
<td>0</td>
<td>0</td>
<td>nc -l 8888</td>
</tr>
</tbody>
</table>
## Use Case: Container Breakout

`linux_getcwd (get working directory)`

<table>
<thead>
<tr>
<th>Name</th>
<th>Pid</th>
<th>CWD</th>
</tr>
</thead>
<tbody>
<tr>
<td>docker-containe</td>
<td>10397</td>
<td><code>/moby</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>b2abeab21db6cf028f19cb610cf8a619c2dd8dc512f638fb34cd42327ec06ca</code></td>
</tr>
<tr>
<td>docker-containe</td>
<td>11436</td>
<td><code>/moby</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>98d97f45ea9fccc2d5f65602b4562ccaad59eb759ca83b435baee0e9fd28f4df3</code></td>
</tr>
</tbody>
</table>
Use Case: Exposed Host Filesystem

- Vulnerable Web Application
- Has access to host root filesystem
- Potential to modify binaries, config files, logs
## Use Case: Exposed Host Filesystem

### docker_containers and docker_container_mounts tables

```sql
select
c.name, m.source, m.destination, m.mode
from
docker_containers c, docker_container_mounts m
where
c.id=m.id;
```

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>destination</th>
<th>mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>/vulnwebserver</td>
<td>/vagrant_docker/db/dump.sql</td>
<td>/docker-entrypoint-initdb.d/dump.sql</td>
<td>rw</td>
</tr>
<tr>
<td>/vulnappdb</td>
<td>/vagrant_docker/db/dump.sql</td>
<td>/docker-entrypoint-initdb.d/dump.sql</td>
<td>rw</td>
</tr>
</tbody>
</table>
Use Case: Exposed Host Filesystem

• It is possible to join mount namespace information for tasks and mounts in Volatility, work in progress.

• Current data on container mounts:

```plaintext
cgroup   /docker_host/var/lib/docker/overlay2/99ae583a6a4a0d27b6dc6b8c70d3a549cd42b287050fb61b97c761893ba0ab7/merged... cgroup   rw,relatime,nosuid,nodev,noexec
tmpfs    /sys/fs/cgroup               tmpfs   rw,relatime,nosuid,nodev,noexec
tmpfs    /sys/fs/cgroup               tmpfs   ro,nosuid,nodev,noexec

cgroup   /sys/fs/cgroup/cpuset/b2abeab21d6bf028f19cb610cf8a6192c2dd8dc42e3fbc51263fb14cd42327ce06ca cgroup   rw,relatime,nosuid,nodev,noexec
cgroup   /sys/fs/cgroup/pids/b2abeab21d6bf028f19cb610cf8a6192c2dd8dc42e3fbc51263fb14cd42327ce06ca cgroup   rw,relatime,nosuid,nodev,noexec
cgroup   /sys/fs/cgroup/hugetlb/98d97f45ea9ff0565602b4562caaa859eb759ca83b435bae0e9f38f4df3 cgroup   ro,relatime,nosuid,nodev,noexec
tmpfs    /sys/fs/cgroup               tmpfs   ro,relatime,nosuid,nodev,noexec
cgroup   /sys/fs/cgroup/memory/b2abeab21d6bf028f19cb610cf8a6192c2dd8dc42e3fbc51263fb14cd42327ce06ca cgroup   rw,relatime,nosuid,nodev,noexec
cgroup   /docker_host/sys/fs/cgroup/net_cls,net_prio cgroup   rw,relatime,nosuid,nodev,noexec
cgroup   /sys/fs/cgroup/systemd        cgroup   rw,relatime,nosuid,nodev,noexec
cgroup   /docker_host/var/lib/docker/overlay2/99ae583a6a4a0d27b6dc6b8c70d3a549cd42b287050fb61b97c761893ba0ab7/merged... cgroup   rw,relatime,nosuid,nodev,noexec
cgroup   /sys/fs/cgroup/bkio           cgroup   rw,relatime,nosuid,nodev,noexec
cgroup   /sys/fs/cgroup/freezer/98d97f45ea9ff0565602b4562caaa859eb759ca83b435bae0e9f38f4df3 cgroup   ro,relatime,nosuid,nodev,noexec
cgroup   /docker_host/var/lib/docker/overlay2/99ae583a6a4a0d27b6dc6b8c70d3a549cd42b287050fb61b97c761893ba0ab7/merged... cgroup   rw,relatime,nosuid,nodev,noexec
```

Use Case: Exposed Host Filesystem

- Vulnerable Web Application
- Has access to environment variables with sensitive data
- Potential for lateral movement, access to other resources
## Use Case: Creds in Environment Vars

### docker_containers table

```sql
select
    name, env_variables
from
    docker_containers;
```

<table>
<thead>
<tr>
<th>name</th>
<th>env_variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>/vulnwebserver</td>
<td>MYSQL_PASSWORD=vulnp55w0rds, MYSQL_USER=vulndbuser, DB_NAME=BOGUS_DB,</td>
</tr>
<tr>
<td>/vulnappdb</td>
<td>MYSQL_ROOT_PASSWORD=r00t, MYSQL_PASSWORD=vulnp55, MYSQL_USER=vulndbu, MYSQL_DATABASE=BOGUS_DB</td>
</tr>
</tbody>
</table>
### Use Case: Creds in Environment Vars

`linux_psenv (get process environment variables)`

<table>
<thead>
<tr>
<th>Name</th>
<th>Pid</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>mysqld</td>
<td>11458</td>
<td>MYSQL_PASSWORD=vulnp55w0rds HOSTNAME=98d97f45ea9f MYSQL_DATABASE=BOGUS_DB...</td>
</tr>
<tr>
<td>python</td>
<td>10423</td>
<td>HOSTNAME=b2abeab21db6 MYSQL_PASSWORD=vulnp55w0rds MYSQL_USER=vulndbuser...</td>
</tr>
</tbody>
</table>
Use Case: Network Connections

docker_containers and docker_container_networks tables

```
select
c.name as c_name, n.name, n.gateway, n.ip_address, n.ip_prefix_len, n.mac_address, p.port, p.host_ip, p.host_port
from
  docker_containers c, docker_container_networks n left outer join docker_container_ports p
on c.id=p.id
where
c.id=n.id;
```

<table>
<thead>
<tr>
<th>c_name</th>
<th>name</th>
<th>gateway</th>
<th>ip_address</th>
<th>ip_prefix_len</th>
<th>mac_address</th>
<th>port</th>
<th>host_ip</th>
<th>host_port</th>
</tr>
</thead>
<tbody>
<tr>
<td>/vulnwebserver</td>
<td>vagrantdocker_default</td>
<td>172.18.0.1</td>
<td>172.18.0.3</td>
<td>16</td>
<td>02:42:ac:12:00:03</td>
<td>80</td>
<td>0.0.0.0</td>
<td>8080</td>
</tr>
<tr>
<td>/vulnappdb</td>
<td>vagrantdocker_default</td>
<td>172.18.0.1</td>
<td>172.18.0.2</td>
<td>16</td>
<td>02:42:ac:12:00:02</td>
<td>3306</td>
<td>0.0.0.0</td>
<td>6033</td>
</tr>
</tbody>
</table>

Then select * from process_open_sockets where port in [8080, 6033] to see what’s connected.
Same table to see any outbound connections mapped to processes.
Use Case: Network Connections

linux_netstat plugin, search with pids

TCP          0.0.0.0         : 80   0.0.0.0         :    0 LISTEN     python/10423
TCP          ::              : 8080 ::              :    0 LISTEN      docker-proxy/8612
TCP          ::              : 3306 ::              :    0 LISTEN      mysql/11458
TCP          ::              : 6033 ::              :    0 LISTEN      docker-proxy/8319

linux_ifconfig plugin

<table>
<thead>
<tr>
<th>Interface</th>
<th>IP Address</th>
<th>MAC Address</th>
<th>Promiscous Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>docker0</td>
<td>172.17.0.1</td>
<td>02:42:06:9a:aa:96</td>
<td>False</td>
</tr>
<tr>
<td>br-e634dc092402</td>
<td>172.18.0.1</td>
<td>02:42:85:7a:86:20</td>
<td>False</td>
</tr>
<tr>
<td>eth0</td>
<td>172.18.0.3</td>
<td>02:42:ac:12:00:03</td>
<td>False</td>
</tr>
<tr>
<td>eth0</td>
<td>172.18.0.2</td>
<td>02:42:ac:12:00:02</td>
<td>False</td>
</tr>
</tbody>
</table>
The Volatility Framework

• Complicated? No worries!
• Dump file system in memory with the `linux_recover_filesystem` plugin
• Analyze the `bolt db *.db files` associated with `dockerd`
• These database files have most container configs
• Similar to running the ‘docker inspect’ cmd
Conclusion

- Monitor your Docker installations
- osquery for continuous monitoring
- Volatility Framework for deep dives + forensics
- Before all this, reduce risks by:
  - Hardening hosts and containers
  - Following security best practices, i.e. CIS Benchmarks
Questions? Thank You!