
facebook

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

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
  - Psaux, 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

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

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

```
-------------------------------------------------------

select
  name, pid, command, privileged, security_options
from
  docker_containers;
```

```
+----------------+-------+-------------------------------+------------+------------------+
| name           | pid   | command                       | privileged | security_options |
+----------------+-------+-------------------------------+------------+------------------+
| /vulnwebserver | 10423 | python -m SimpleHTTPServer 80 | 1          | label=disable    |
| /vulnappdb     | 11458 | docker-entrypoint.sh mysqld   | 0          |                  |
+----------------+-------+-------------------------------+------------+------------------+
```

Use Case: Container Breakout

linux_pstree plugin, uid 999(docker)

```
------------------------------------------------------
Name                 Pid             PPid            Uid
.dockerd             4043            1
...docker-containe   11436            4073
....mysqld           11458           10255           999
..docker-containe    4059            4043
...docker-containe   10397           4059
....python           10423           10397
....nc               14521           10397
```

linux_psaux plugin

```
------------------------------------------------------
Pid    Uid    Gid    Arguments
10423  0      0      python -m SimpleHTTPServer 80
11458  999    999    mysqld
14521  0      0      nc -l 8888
```

Use Case: Container Breakout

linux_getcwd (get working directory)

Name	Pid	CWD
docker-containe	10397	…/moby/b2abeab21db6cf028f19cb610cf8a619c2ddd8dc512f638fb34cd42327ec06ca
docker-containe	11436	…/moby/98d97f45ea9ffc2d5f65602b4562ccaad59eb759ca83b435baee0e9fd28f4df3

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

```
-------------------------------------------------------
```

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

```
+----------------+----------------------------+---------------------------------------+------+
| name           | source                     | destination                           | mode |
+----------------+----------------------------+---------------------------------------+------+
| /vulnwebserver | /                          | /docker_host                          | rw   |
| /vulnappdb     | /vagrant_docker/db/dump.sql | /docker-entrypoint-initdb.d/dump.sql | rw   |
+----------------+----------------------------+---------------------------------------+------+
```

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:

```
cgroup          /docker_host/var/lib/docker/overlay2/99ae583a6a41a0d27b6dc4b8c70d3a549cd42b287050fb61b97c761893ba0ab7/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/b2abeab21db6cf028f19cb610cf8a619c2ddd8dc512f638fb34cd42327ec06ca cgroup      rw,relatime,nosuid,nodev,noexec
cgroup          /sys/fs/cgroup/pids/b2abeab21db6cf028f19cb610cf8a619c2ddd8dc512f638fb34cd42327ec06ca cgroup      rw,relatime,nosuid,nodev,noexec
cgroup          /sys/fs/cgroup/hugetlb/98d97f45ea9ffc2d5f65602b4562ccaad59eb759ca83b435baee0e9fd28f4df3 cgroup      ro,relatime,nosuid,nodev,noexec
tmpfs           /sys/fs/cgroup                  tmpfs       ro,relatime,nosuid,nodev,noexec
cgroup          /sys/fs/cgroup/memory/b2abeab21db6cf028f19cb610cf8a619c2ddd8dc512f638fb34cd42327ec06ca 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/overlay2/99ae583a6a41a0d27b6dc4b8c70d3a549cd42b287050fb61b97c761893ba0ab7/merged… cgroup      rw,relatime,nosuid,nodev,noexec
cgroup          /sys/fs/cgroup/blkio            cgroup      rw,relatime,nosuid,nodev,noexec
cgroup          /sys/fs/cgroup/freezer/98d97f45ea9ffc2d5f65602b4562ccaad59eb759ca83b435baee0e9fd28f4df3 cgroup      ro,relatime,nosuid,nodev,noexec
cgroup          /docker_host/var/lib/docker/overlay2/99ae583a6a41a0d27b6dc4b8c70d3a549cd42b287050fb61b97c761893ba0ab7/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

```
-------------------------------------------------------

select
   name, env_variables
from
   docker_containers;
```

```
+---------------+-----------------------------------------------------------------------------------------------------+
| name          | env_variables                                                                                       |
+---------------+-----------------------------------------------------------------------------------------------------+
| /vulnwebserver | MYSQL_PASSWORD=vulnp55w0rds, MYSQL_USER=vulndbuser, DB_NAME=BOGUS_DB,                              |
| /vulnappdb    | MYSQL_ROOT_PASSWORD=r00t, MYSQL_PASSWORD=vulnp55, MYSQL_USER=vulndbu, MYSQL_DATABASE=BOGUS_DB      |
+---------------+-----------------------------------------------------------------------------------------------------+
```

Use Case: Creds in Environment Vars

linux_psenv (get process environment variables)

```
------------------------------------------------------------

Name    Pid    Environment
----------------------------
mysqld 11458  MYSQL_PASSWORD=vulnp55w0rds HOSTNAME=98d97f45ea9f MYSQL_DATABASE=BOGUS_DB…
python 10423  HOSTNAME=b2abeab21db6 MYSQL_PASSWORD=vulnp55w0rds MYSQL_USER=vulndbuser…
```

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;
```

```
+---------------+----------------------+------------+------------+---------------+-------------------+-------+---------+-----------+
| c_name        | name                 | gateway    | ip_address | ip_prefix_len | mac_address       | port  | host_ip | host_port |
+---------------+----------------------+------------+------------+---------------+-------------------+-------+---------+-----------+
| /vulnwebserver | vagrantdocker_default | 172.18.0.1 | 172.18.0.3 | 16            | 02:42:ac:12:00:03 |   80  | 0.0.0.0 | 8080      |
| /vulnappdb     | vagrantdocker_default | 172.18.0.1 | 172.18.0.2 | 16            | 02:42:ac:12:00:02 | 3306  | 0.0.0.0 | 6033      |
+---------------+----------------------+------------+------------+---------------+-------------------+-------+---------+-----------+
```

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      mysqld/11458
TCP      ::               : 6033 ::               :    0 LISTEN      docker-proxy/8319
```

linux_ifconfig plugin

```
-------------------------------------------------------

Interface        IP Address           MAC Address        Promiscous Mode
---------------- -------------------- ------------------ ---------------
docker0          172.17.0.1           02:42:06:9a:aa:96  False
br-e634dc092402  172.18.0.1           02:42:85:7a:86:20  False
eth0             172.18.0.3           02:42:ac:12:00:03  False
eth0             172.18.0.2           02:42:ac:12:00:02  False
```

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

facebook

Questions? Thank You!