Configuring the Docker Daemon for Security

Nigel Brown

@n_brownuk www.windsock.io

Module Outline

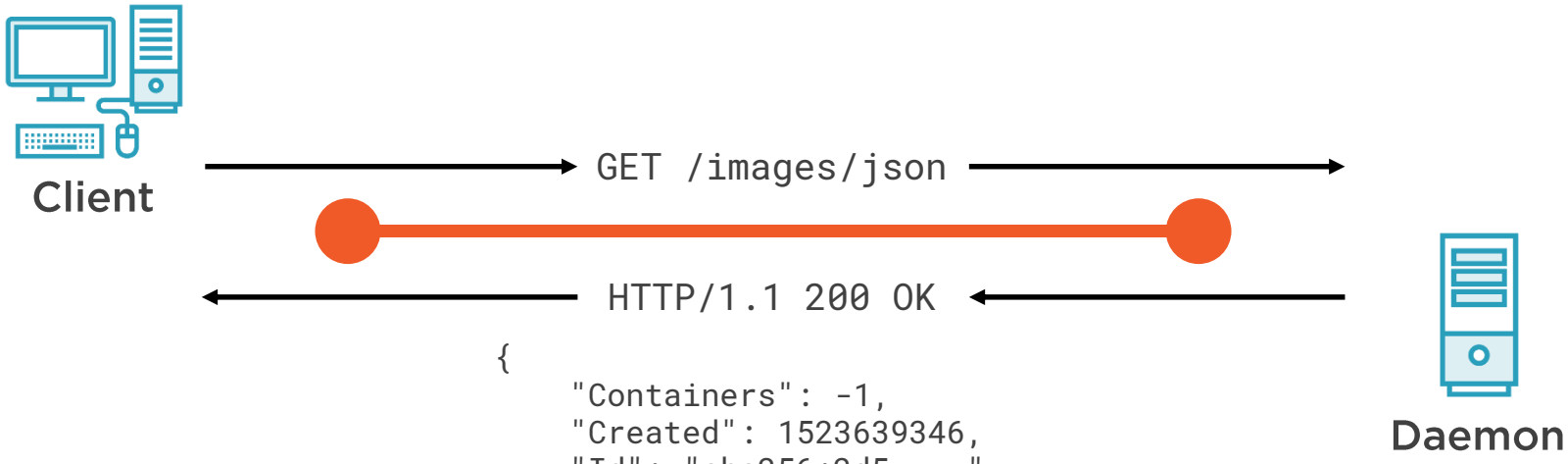
Controlling access to the Docker daemon, from a local client

Establishing secure communication between remote client and daemon

Configuring the Docker daemon, to mitigate against container breakout

Local Docker UNIX Domain Socket

```
$ docker image ls
```




```
{
    "Containers": -1,
    "Created": 1523639346,
    "Id": "sha256:2d5 ...",
    "Labels": {},
    "ParentId": "sha256:afc ...",
    "RepoDigests": null,
    "RepoTags": [
      "nginx:ghost"
    ],
    .
    .
}
```

Addressing the Docker Daemon

The `--host` or `-H` config option is used to specify the Docker daemon socket (e.g. `-H unix:///var/run/docker.sock`)

The Docker daemon can be configured to listen for requests on multiple sockets, simultaneously

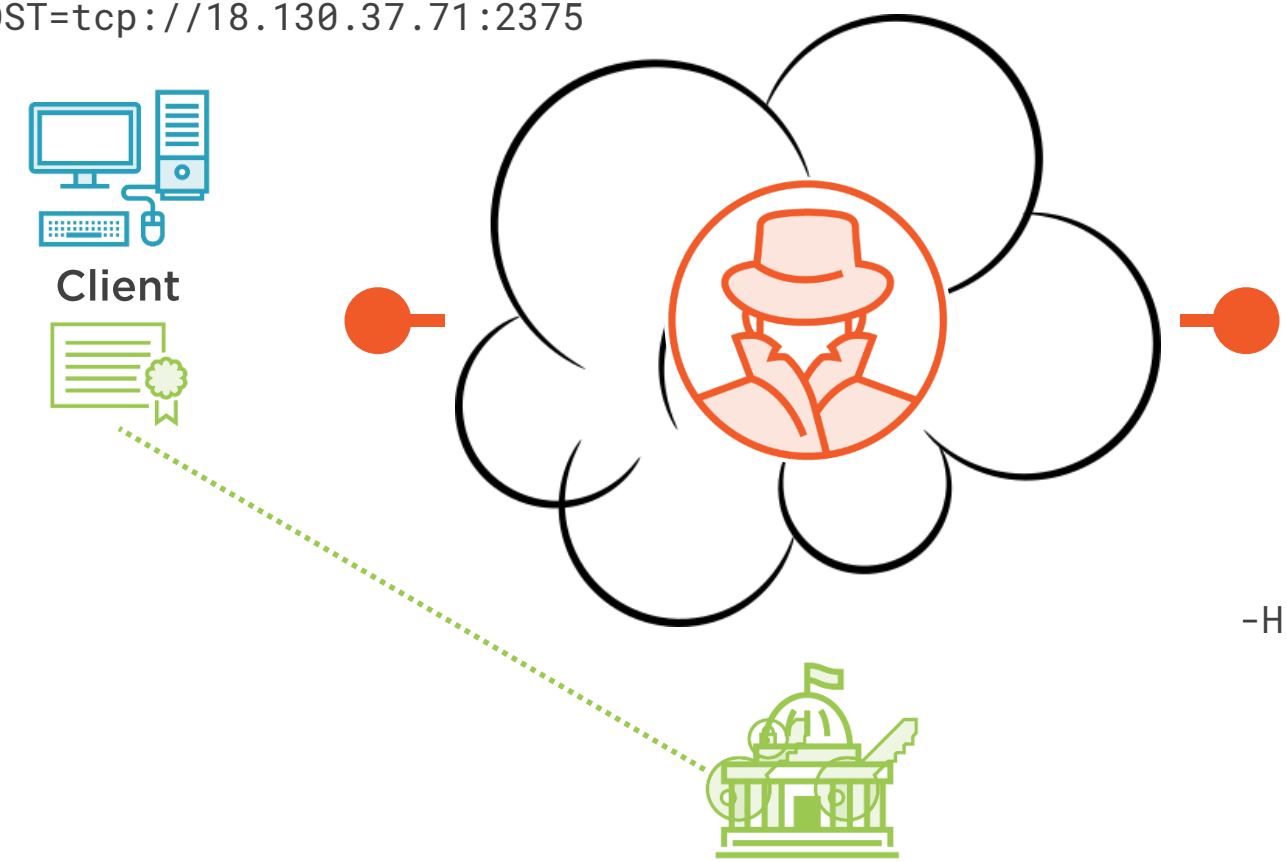
Docker client can be configured to address daemon using config flags, or environment variable

```
[Unit]
Description=Docker Socket for API
PartOf=docker.service

[Socket]
ListenStream=/var/run/docker.sock
SocketMode=0660
SocketUser=root
SocketGroup=docker

[Install]
WantedBy=sockets.target
```

- Unit file is part of the Systemd service for the Docker daemon
- UNIX domain socket definition
- Creates dependency for boot sequence on install

Remote Docker TCP Socket

```
export DOCKER_HOST=tcp://18.130.37.71:2375
```



TLS Config Options for Daemon

Option	Behavior
`--tls`	Explicitly instruct daemon to use TLS
`--tlskey`	Pathname of daemon's private key
`--tlscert`	Pathname of daemon's signed certificate
`--tlsverify`	Verify the authenticity of client connections
`--tlscacert`	Pathname of CA certificate

1. `--tlsverify, --tlscacert, --tlscert, --tlskey`
   Daemon configured to be authenticated by client, and vice versa

2. `--tls, --tlscert, --tlskey`
   Daemon configured to be authenticated by client

TLS Config Options for Client

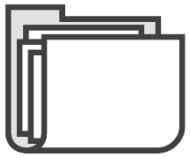
Option	Behavior
`--tls`	Authenticate daemon using system CA certificate
`--tlskey`	Pathname of client's private key
`--tlscert`	Pathname of client's signed certificate
`--tlsverify`	Verify authenticity of daemon using CA certificate
`--tlscacert`	Pathname of CA's certificate

1. `--tls`
   Client authenticates daemon using public CA certificate
2. `--tlsverify, --tlscacert`
   Client authenticates daemon using the CA certificate specified
3. `--tlsverify, --tlscacert, --tlscert, --tlskey`
   Client configured to be authenticated by daemon, and vice versa
4. `--tls, --tlscert, --tlskey`
   Client configured to be authenticated by daemon and authenticates daemon using public CA certificate

Configuring the Docker Client

```
$ docker --tlsverify \
> --tlscacert=ca.pem \
> --tlscert=client.pem \
> --tlskey=client-key.pem \
> -H tcp://18.130.37.71:2376 \
> container run -itd --publish 443:443 nginx
```

Configuring the Docker Client



`DOCKER_CERT_PATH` – **Specifies the location of key and certificates the client uses for TLS-enabled communication**

`DOCKER_TLS_VERIFY` – **When set, the client authenticates the Docker daemon it attempts to communicate with**

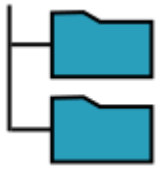
`DOCKER_HOST` – **Informs the client of the socket location to use when communicating with the Docker daemon**

Namespace

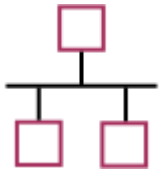
A Linux namespace is a kernel construct, which allows for the isolation of an operating system resource from the perspective of a running process.

Linux Namespaces

PID namespace



Mount namespace



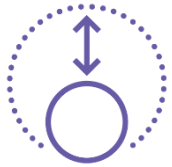
Network namespace

IPC namespace

UTS namespace

User namespace

User Namespaces



User namespaces re-map the set of user and group IDs from one namespace to another

A process can be non-privileged in one user namespace, yet privileged in another

User namespaces mitigate against a breakout from the confines of a container

Docker does not enable user namespace use as the default container creation behavior

Subordinate User and Group IDs

Non-privileged User

```
rackham
uid=1000,gid=1000
```

Subordinate UID/GID File

```
lxd:100000:65536
root:100000:65536
rackham:165536:65536
baxter:231072:65536
bolt:296608:65536
```

```
165536 -> 0
```

```
# Specify the user/group for re-mapping
--userns-remap=default|uid:gid|user:group|user|uid


# Override daemon re-mapping with client
--userns=host
```

Configuring the Daemon for User Namespaces

Docker can perform re-mapping, or it can be pre-configured

Re-mapping can be overridden on a per-container basis

User Namespace Gotchas

Check to ensure the user employed to configure user namespaces has a subordinate mapping configured

Images and other Docker objects may need to be re-created as object content is stored in a sub-directory

Configure the permissions on host volumes mounted in containers so that the volume can be read and written to

Module Summary

Carefully control access to the Docker daemon's UNIX domain socket

Implement mutual TLS between Docker client and daemon

Re-map UID/GIDs for containers, using the user namespace config option