ATTACK DEFENSE

by PentesterAcademy

Name	Confining Containers with AppArmor II
URL	https://attackdefense.com/challengedetails?cid=1837
Type	Privilege Escalation : AppArmor

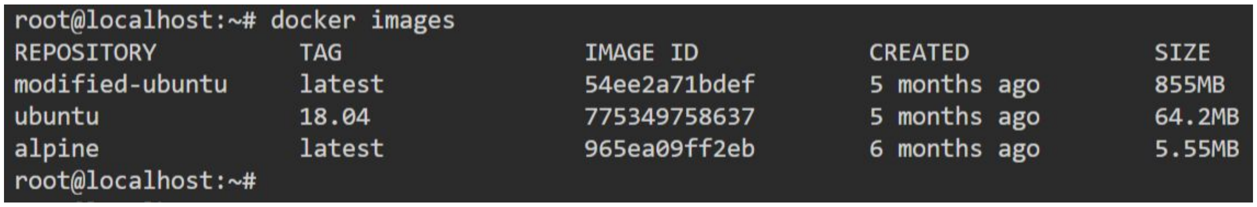
Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this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

Objective: Learn how to apply AppArmor profiles from a user-friendly format using the bane tool.

Solution:

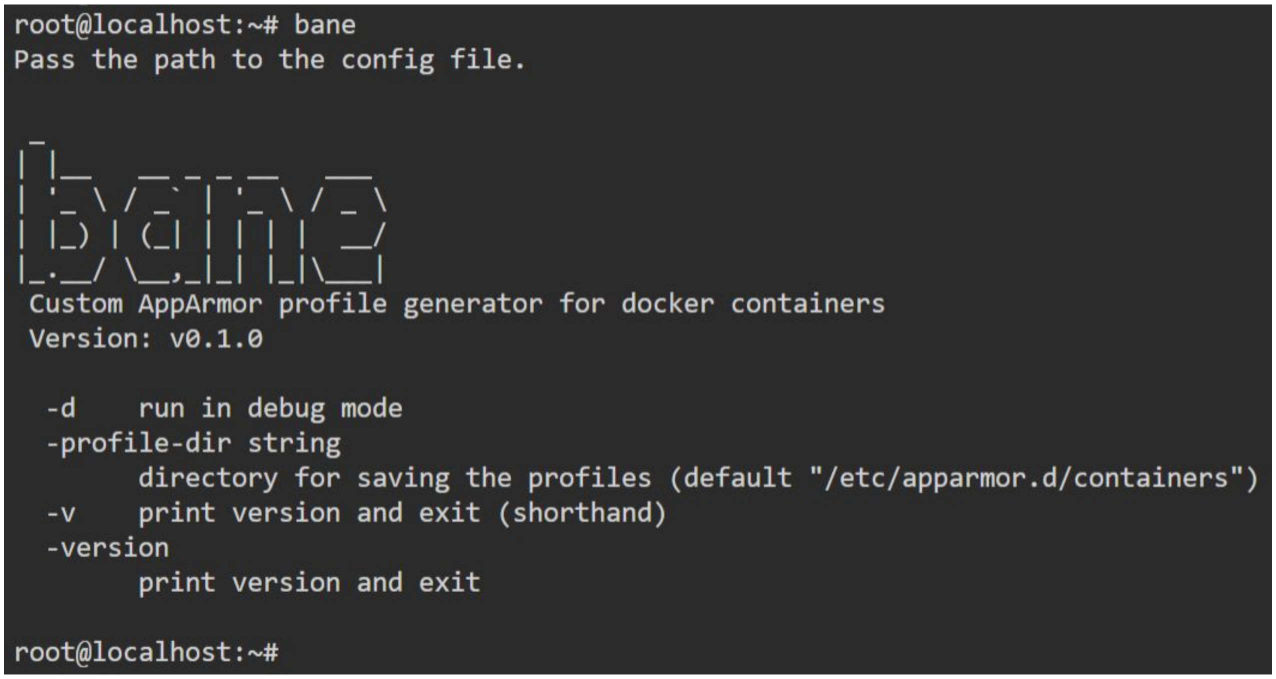
Step 1: List the docker images present on the machine.

Command: docker images


```
root@localhost:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
modified-ubuntu     latest              54ee2a71bdef        5 months ago        855MB
ubuntu              18.04               775349758637        5 months ago        64.2MB
alpine              latest              965ea09ff2eb        6 months ago        5.55MB
root@localhost:~#
```

Step 2: Check the help menu of the bane tool.

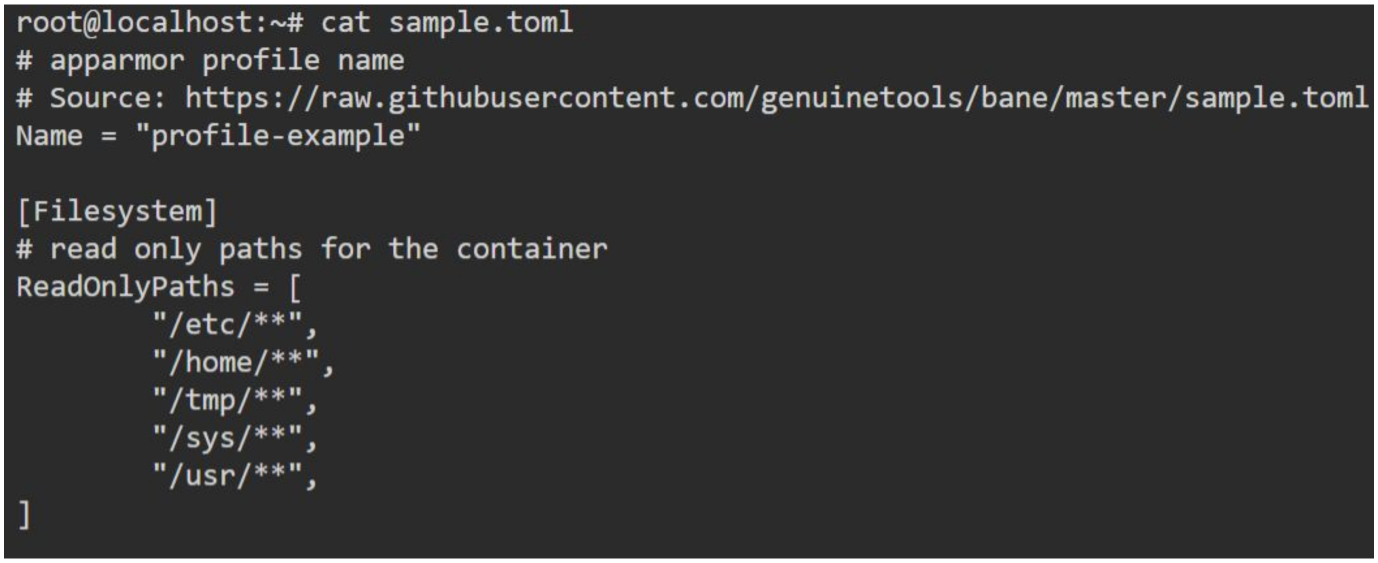
Command: bane



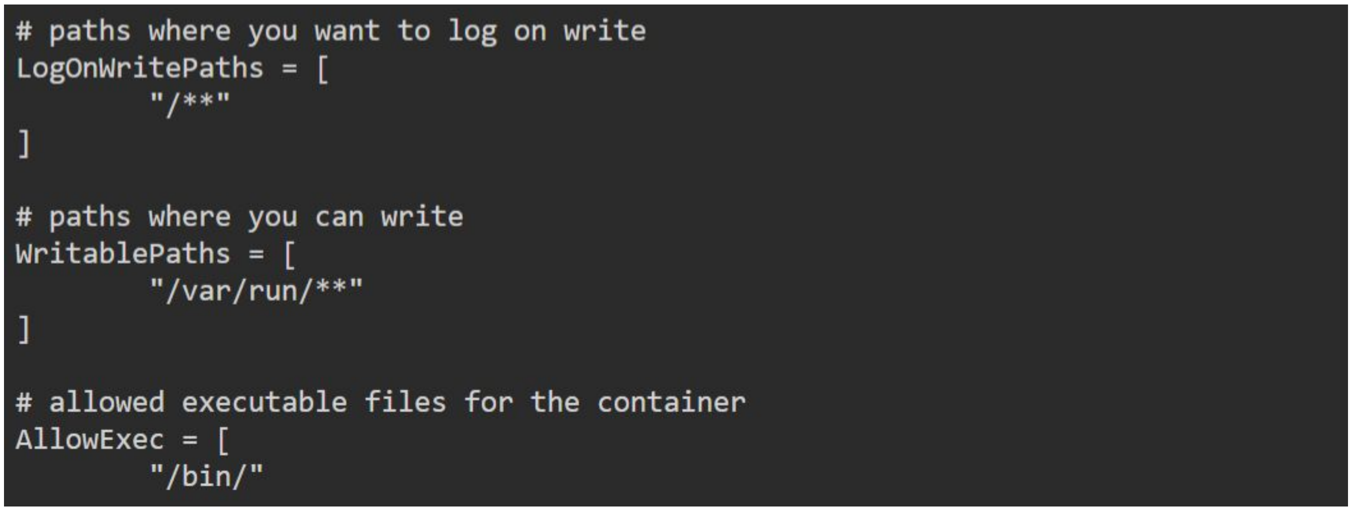
```
root@localhost:~# bane
Pass the path to the config file.

 _
| |__   __ _ _ __   ___
| '_ \ / _` | '_ \ / _ \
| |_) | (_| | | | |  __/
|_.__/ \__,_|_| |_|\___|
 Custom AppArmor profile generator for docker containers
 Version: v0.1.0

  -d    run in debug mode
  -profile-dir string
        directory for saving the profiles (default "/etc/apparmor.d/containers")
  -v    print version and exit (shorthand)
  -version
        print version and exit

root@localhost:~#
```

Step 3: Check the contents of provided sample file i.e. sample.toml

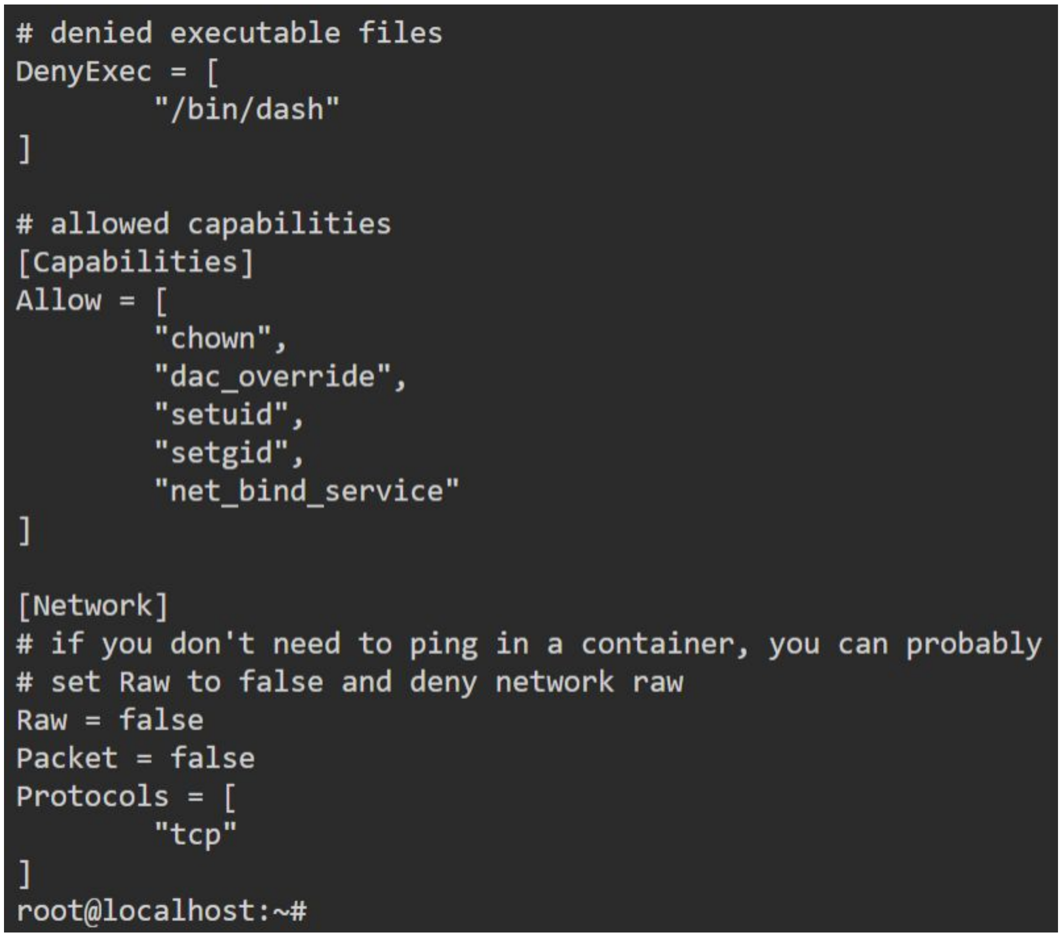
Command: cat sample.toml



```
root@localhost:~# cat sample.toml
# apparmor profile name
# Source: https://raw.githubusercontent.com/genuinetools/bane/master/sample.toml
Name = "profile-example"

[Filesystem]
# read only paths for the container
ReadOnlyPaths = [
        "/etc/**",
        "/home/**",
        "/tmp/**",
        "/sys/**",
        "/usr/**",
]
```



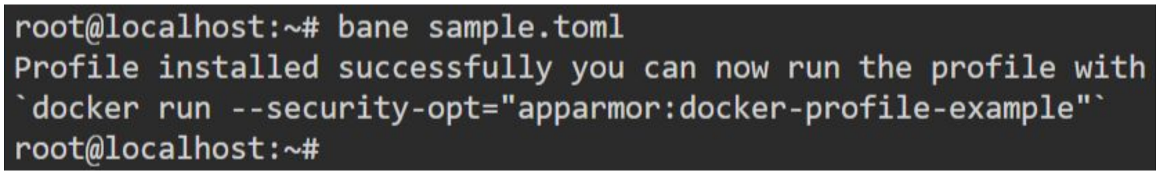
```
# paths where you want to log on write
LogOnWritePaths = [
        "/**"
]

# paths where you can write
WritablePaths = [
        "/var/run/**"
]

# allowed executable files for the container
AllowExec = [
        "/bin/"
```



```
# denied executable files
DenyExec = [
        "/bin/dash"
]

# allowed capabilities
[Capabilities]
Allow = [
        "chown",
        "dac_override",
        "setuid",
        "setgid",
        "net_bind_service"
]

[Network]
# if you don't need to ping in a container, you can probably
# set Raw to false and deny network raw
Raw = false
Packet = false
Protocols = [
        "tcp"
]
root@localhost:~#
```

Step 4: Install the “docker-profile-example” profile (name of the profile mentioned in the config file).

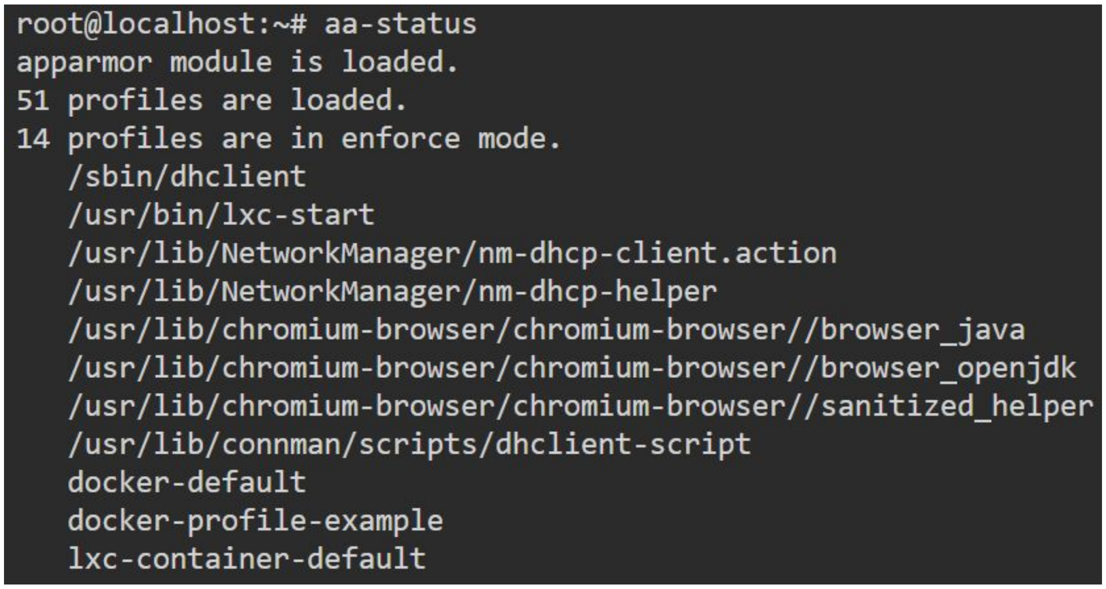
Command: bane sample.toml



```
root@localhost:~# bane sample.toml
Profile installed successfully you can now run the profile with
`docker run --security-opt="apparmor:docker-profile-example"`
root@localhost:~#
```

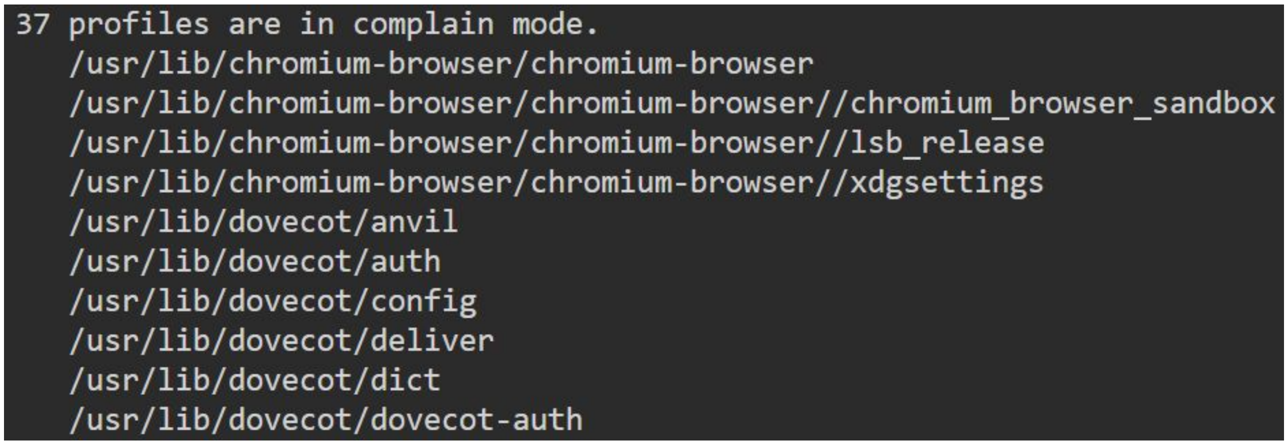
Step 5: Verify that the profile has been installed using the AppArmor status.

Command: aa-status

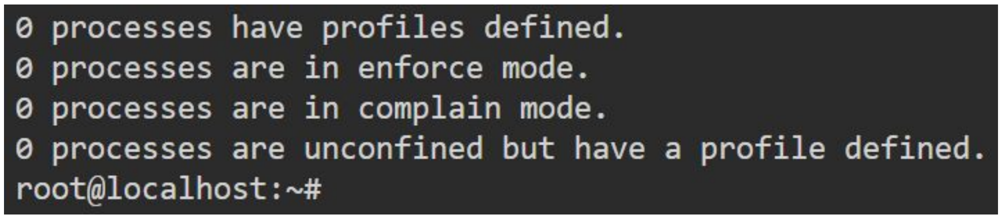


```
root@localhost:~# aa-status
apparmor module is loaded.
51 profiles are loaded.
14 profiles are in enforce mode.
   /sbin/dhclient
   /usr/bin/lxc-start
   /usr/lib/NetworkManager/nm-dhcp-client.action
   /usr/lib/NetworkManager/nm-dhcp-helper
   /usr/lib/chromium-browser/chromium-browser//browser_java
   /usr/lib/chromium-browser/chromium-browser//browser_openjdk
   /usr/lib/chromium-browser/chromium-browser//sanitized_helper
   /usr/lib/connman/scripts/dhclient-script
   docker-default
   docker-profile-example
   lxc-container-default
```

One can observe the “docker-profile-example” in the enforce mode profile list.



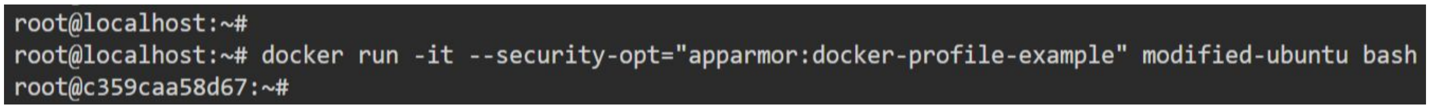
```
37 profiles are in complain mode.
   /usr/lib/chromium-browser/chromium-browser
   /usr/lib/chromium-browser/chromium-browser//chromium_browser_sandbox
   /usr/lib/chromium-browser/chromium-browser//lsb_release
   /usr/lib/chromium-browser/chromium-browser//xdgsettings
   /usr/lib/dovecot/anvil
   /usr/lib/dovecot/auth
   /usr/lib/dovecot/config
   /usr/lib/dovecot/deliver
   /usr/lib/dovecot/dict
   /usr/lib/dovecot/dovecot-auth
```



```
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
root@localhost:~#
```

Step 6: Run the docker image “modified-ubuntu” with this newly added profile.

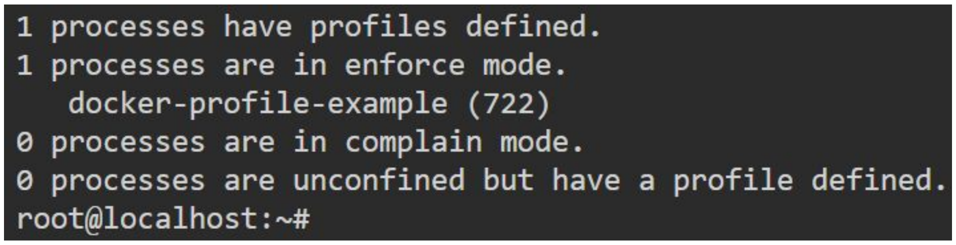
Command: docker run -it --security-opt=”apparmor:docker-profile-example” modified-ubuntu bash



```
root@localhost:~#
root@localhost:~# docker run -it --security-opt="apparmor:docker-profile-example" modified-ubuntu bash
root@c359caa58d67:~#
```

Step 7: Open another terminal T2 and check the AppArmor status to see if the profile is actually applied to the container.

Command: aa-status



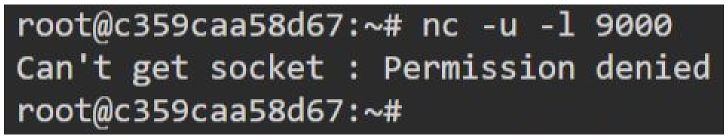
```
1 processes have profiles defined.
1 processes are in enforce mode.
   docker-profile-example (722)
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
root@localhost:~#
```

The profile is listed in the running process section. This means that the profile is containing the container.

Step 8: Switch back to terminal T1 and try different activities inside the container:

Activity I: Run Netcat in listening mode on UDP port 9000

Command: nc -u -l 9000

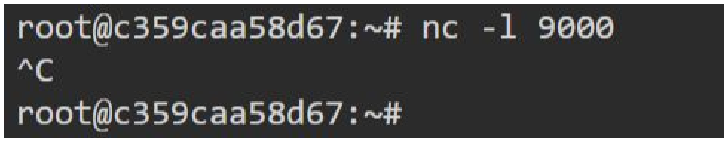


```
root@c359caa58d67:~# nc -u -l 9000
Can't get socket : Permission denied
root@c359caa58d67:~#
```

The UDP network access is not allowed in the profile, so it won't work.

Activity II: Run Netcat in listening mode on TCP port 9000

Command: nc -l 9000

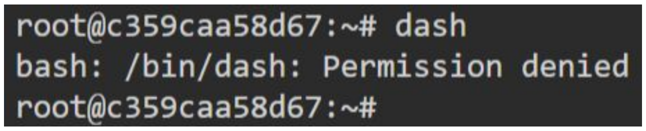


```
root@c359caa58d67:~# nc -l 9000
^C
root@c359caa58d67:~#
```

Activity III: The TCP network access is allowed in the profile, so it will be allowed by AppArmor.

Execute the dash shell.

Command: dash

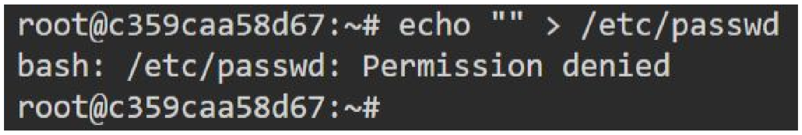


```
root@c359caa58d67:~# dash
bash: /bin/dash: Permission denied
root@c359caa58d67:~#
```

The execution access on binary /bin/dash is not allowed in the profile. Hence, the permission denied message.

Activity IV: Overwrite the passwd file.

Command: echo "" > /etc/passwd



```
root@c359caa58d67:~# echo "" > /etc/passwd
bash: /etc/passwd: Permission denied
root@c359caa58d67:~#
```

The write operation is not allowed in most directories including /etc so the operation failed.

Learning:

- Easy to write configuration file can be used to define Apparmor profiles using the bane tool.
- By defining such custom profiles for all running application/web server containers, the attacker can be restricted to the container itself even after break into the container.

References:

- AppArmor man page (http://manpages.ubuntu.com/manpages/bionic/man7/apparmor.7.html) (http://manpages.ubuntu.com/manpages/bionic/man5/apparmor.d.5.html)
- Beginning AppArmor profile development (https://ubuntu.com/tutorials/beginning-apparmor-profile-development)
- Bane Tool (https://github.com/genuinetools/bane)