ESERCIZIO CONTAINER/CGROUPS

1. AVVIO DEL CONTAINER E ALLOCAZIONE MEMORIA

Prima di tutto ho avviato un container python:3.9-slim ed ho utilizzato lo script Python per allocare staticamente esattamente 100MB di RAM.
Inoltre ho impostato la swap uguale alla memoria RAM in modo da disabilitare l'uso della swap per questo container. Questo per forzare il kernel a terminare il processo invece di provare a scambiare memoria su disco quando il limite viene raggiunto.

```
┌──(kali㉿kali)-[~]
└─$ sudo docker run -d \
> --name es_oom \
> --memory=300m \
> --memory-swap=300m \
> python:3.9-slim \
> python -c "import time; x = 'a' * 1024 * 1024 * 100; print('Allocati 100MB');
time.sleep(3600)"
26ff11945d1dd133f157fe370b5a4c3b182f40aba4653dffd56e6b1ebd7ef139
```

2. MONITORAGGIO INTERNO COL COMANDO FREE

Prima di tutto ho dovuto installare il pacchetto procps che include il comando free.
Dopodichè il comando "free -m" mi ha dato un output che mi ha fatto riflettere.
Lanciando quel comando all'interno del container, l'output che mi da è in riferimento alla VM Kali Linux e non del solo container.
Tuttavia so che il processo python sta contribuendo per circa 100 MB a quel totale di 756 MB.

```
┌──(kali㉿kali)-[~]
└─$ sudo docker exec -u 0 -it es_oom sh -c "export DEBIAN_FRONTEND=noninteractive
&& apt-get update && apt-get install -y procps"
Get:1 http://deb.debian.org/debian trixie InRelease [140 kB]
Get:2 http://deb.debian.org/debian trixie-updates InRelease [47.3 kB]
Get:3 http://deb.debian.org/debian-security trixie-security InRelease [43.4 kB]
Get:4 http://deb.debian.org/debian trixie/main amd64 Packages [9670 kB]
Get:5 http://deb.debian.org/debian trixie-updates/main amd64 Packages [5412 B]
Get:6 http://deb.debian.org/debian-security trixie-security/main amd64 Packages [73.1 kB]
Fetched 9979 kB in 2s (4728 kB/s)
Reading package lists... Done
N: Repository 'http://deb.debian.org/debian trixie InRelease' changed its 'Version' value from
'13.1' to '13.2'
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libproc2-0 linux-sysctl-defaults psmisc
```

```
The following NEW packages will be installed:
  libproc2-0 linux-sysctl-defaults procps psmisc
0 upgraded, 4 newly installed, 0 to remove and 6 not upgraded.
Need to get 1220 kB of archives.
After this operation, 3687 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian trixie/main amd64 linux-sysctl-defaults all 4.12 [5624 B]
Get:2 http://deb.debian.org/debian trixie/main amd64 libproc2-0 amd64 2:4.0.4-9 [65.6 kB]
Get:3 http://deb.debian.org/debian trixie/main amd64 procps amd64 2:4.0.4-9 [882 kB]
Get:4 http://deb.debian.org/debian trixie/main amd64 psmisc amd64 23.7-2 [267 kB]
Fetched 1220 kB in 0s (3887 kB/s)
Selecting previously unselected package linux-sysctl-defaults.
(Reading database ... 5644 files and directories currently installed.)
Preparing to unpack .../linux-sysctl-defaults_4.12_all.deb ...
Unpacking linux-sysctl-defaults (4.12) ...
Selecting previously unselected package libproc2-0:amd64.
Preparing to unpack .../libproc2-0_2%3a4.0.4-9_amd64.deb ...
Unpacking libproc2-0:amd64 (2:4.0.4-9) ...
Selecting previously unselected package procps.
Preparing to unpack .../procps_2%3a4.0.4-9_amd64.deb ...
Unpacking procps (2:4.0.4-9) ...
Selecting previously unselected package psmisc.
Preparing to unpack .../psmisc_23.7-2_amd64.deb ...
Unpacking psmisc (23.7-2) ...
Setting up psmisc (23.7-2) ...
Setting up linux-sysctl-defaults (4.12) ...
Setting up libproc2-0:amd64 (2:4.0.4-9) ...
Setting up procps (2:4.0.4-9) ...
Processing triggers for libc-bin (2.41-12) ...

┌──(kali㉿kali)-[~]
└─$ sudo docker exec -it es_oom free -m
               total        used        free      shared  buff/cache   available
Mem:            1973         756         644          12         728        1216
Swap:            953         161         792
```

3. MANIPOLAZIONE CGROUP (SIMULAZIONE OOM)

Per innescare l'OOM Killer, ho dovuto individuare il percorso del Cgroup associato al container e ho dovuto imporre un limite di memoria inferiore all'utilizzo corrente.
Il limite l'ho impostato a 50MB (52.428.800 Byte), nettamente inferiore al centinaio di MB attualmente in uso.

```
┌──(kali㉿kali)-[~]
└─$ CTR_ID=$(sudo docker inspect --format="{{.Id}}" es_oom)

┌──(kali㉿kali)-[~]
└─$ CGROUP_PATH=$(sudo find /sys/fs/cgroup -name "*$CTR_ID*" | grep "docker" | head
-n 1)
```

```
┌──(kali㉿kali)-[~]
└─$ echo 52428800 | sudo tee "$CGROUP_PATH/memory.max"
52428800

# 4. VERIFICA
Immediatamente dopo l'applicazione del limite, ho verificato lo stato del container.
Ovviamente, il container ha riportato lo stato Exited (137).

┌──(kali㉿kali)-[~]
└─$ docker ps -a
CONTAINER ID   IMAGE            COMMAND                  CREATED         STATUS
PORTS     NAMES
26ff11945d1d   python:3.9-slim   "python -c 'import t…"   3 minutes ago   Exited (137) 23
seconds ago             es_oom
```

RISPOSTE ALLE DOMANDE

D1: L'uccisione da parte dell'OOM killer del container è istantanea oppure no?
R1: Si, nel contesto di questo esercizio l'uccisione è stata istantanea. Normalmente il kernel tenta di recuperare memoria utilizzando lo spazio di swap. Avendo disabilitato lo swap, però, il Kernel chiama immediatamente l'OOM Killer inviando un segnale SIGKILL (9), terminando il processo all'istante.

D2: Una volta impostato un memory.max inferiore alla memoria attualmente utilizzata cosa si rileva all'interno del container? La memoria viene deallocata o non c'è alcun effetto?
R2: Non è più possibile rilevare nulla all'interno del container poiché il processo va in crash. Tutta la memoria che era stata allocata (processo python) viene immediatamente deallocata e restituita al sistema operativo Host.