**Aïllament de serveis i detecció d’activitat maliciosa amb Docker**

**Objectius**

* Entendre la segmentació de xarxes en entorns contenidoritzats
* Aplicar l’aïllament de serveis per limitar vectors d’atac
* Utilitzar eines de diagnosi i detecció d’activitat sospitosa
* Simular un escenari d’atac lateral entre serveis Docker

**Requisits previs**

* Docker i Docker Compose instal·lats

**1. Crear l’estructura de treball**

**mkdir docker-seguretat**

**cd docker-seguretat**

**touch docker-compose.yml index.html**

**2. Fitxer index.html (servei web bàsic)**

**<!DOCTYPE html>**

**<html>**

**<head><meta charset="utf-8"><title>Web</title></head>**

**<body><h1>Benvinguts al servei web</h1></body>**

**</html>**

**3. Fitxer docker-compose.yml**

version: '3.8'

services:

web:

image: nginx:alpine

volumes:

- ./index.html:/usr/share/nginx/html/index.html

networks:

- frontend

db:

image: postgres:alpine

environment:

POSTGRES\_PASSWORD: exemple

networks:

- backend

attacker:

image: nicolaka/netshoot

command: sleep infinity

networks:

- frontend

- backend

cap\_add:

- NET\_ADMIN

networks:

frontend:

backend:

internal: true

**4. Iniciar els serveis**

**docker compose up -d**

**5. Validar la segmentació de xarxes**

**5.1. El servei web no pot accedir directament al servei de base de dades**

**docker exec -it $(docker ps -qf "name=web") wget --spider db:5432**

*1. docker ps -qf "name=web"*

*Aquesta part retorna l’ID del contenidor que conté “web” al seu nom.*

*-q: mostra només l’ID del contenidor (output curt)*

*-f "name=web": aplica un filtre perquè només es mostrin contenidors amb "web" al nom*

*2. docker exec -it <container\_id>*

*Executa una comanda dins del contenidor:*

*-i: mode interactiu*

*-t: crea una terminal virtual*

*<container\_id>: el contenidor retornat per la comanda anterior*

*3. wget --spider db:5432*

*Aquesta és la comanda que s’executa dins del contenidor.*

*wget: eina per fer connexions HTTP o TCP*

*--spider: fa una comprovació de connexió, sense descarregar dades*

*db:5432: intenta accedir al servei db pel port 5432 (PostgreSQL)*

Resultat esperat: l’accés ha de fallar.

*La comanda intenta que el servei web accedeixi al servei db pel nom de servei Docker (db) i el port 5432 (PostgreSQL). Però retorna un error.*

*Aquest missatge indica que el contenidor web no pot resoldre el nom db, és a dir, no té accés al servei de base de dades a través de la xarxa Docker.*

**5.2. El servei attacker pot veure els dos serveis**

**docker exec -it $(docker ps -qf "name=attacker") ping web**

**docker exec -it $(docker ps -qf "name=attacker") ping db**

**6. Analitzar el trànsit amb tcpdump**

Des de attacker:

**docker exec -it $(docker ps -qf "name=attacker") tcpdump -i any port 5432**

Simula connexions cap a la base de dades des d’altres serveis per veure el trànsit capturat, obre un altre terminal i executa:

**docker exec -it $(docker ps -qf "name=attacker") nmap db**

**7. Afegir polítiques de tallafoc**

**docker exec -it $(docker ps -qf "name=attacker") iptables -A OUTPUT -d db -j DROP**

Aquesta regla impedeix que attacker pugui comunicar-se amb db.

Repeteix ara l’apartat 6 per veure si ara la situació és diferent.

**8. Conclusions**

* S'han creat serveis en xarxes Docker separades per aïllar funcionalitats
* El servei de base de dades està protegit amb una xarxa interna
* S’ha simulat un atac lateral des d’un servei amb accés ampli
* S’han utilitzat eines de diagnosi com tcpdump, ping i iptables

**Notes finals**

* La imatge nicolaka/netshoot inclou eines com nmap, tcpdump, ping i iptables
* Per pràctiques avançades, es pot provar l’ús de IDS o regles de tallafoc a nivell del host