ESERCIZIO CONTAINER/NAMESPACES

Obiettivo: Lanciare un processo esterno condividendo i namespace di un container e verificarne la visibilità eseguendo “ps” al suo interno.
Di seguito riporto i passaggi svolti e l’analisi dei risultati ottenuti.

1. AVVIO CONTAINER E ANALISI PROCESSI SUL HOST

Ho avviato il container alpine con un processo “sleep 3600”. A differenza dell’esecuzione su Mac, eseguendo una “ps aux” direttamente sul terminale della macchina Kali, il processo del container è visibile. Questo conferma che su Linux i container sono semplici processi isolati e non macchine virtuali nascoste.

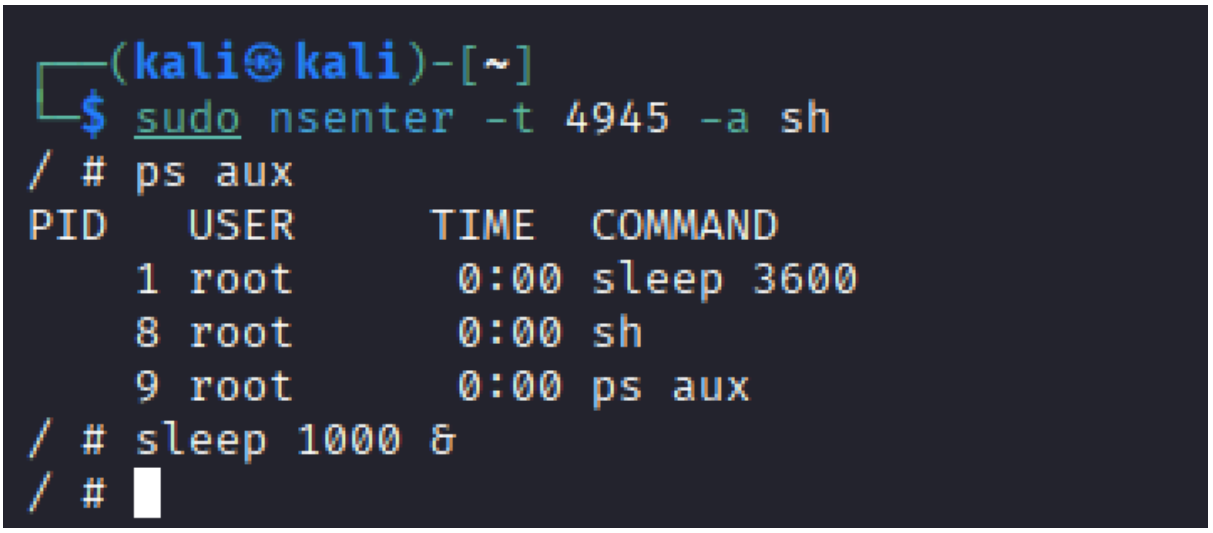
Risultato: il sistema operativo vede il processo PID 4945.

2. CONFERMA PID E INGRESSO NEL NAMESPACE

Ho utilizzato il comando “docker inspect” per confermare che il PID visto dall’host fosse corretto. Successivamente (vedere prossima slide), ho utilizzato “nsenter” direttamente dalla shell di Kali (stavolta senza utilizzare container privilegiati) per “entrare” nel namespace del processo.

3. VERIFICA ISOLAMENTO

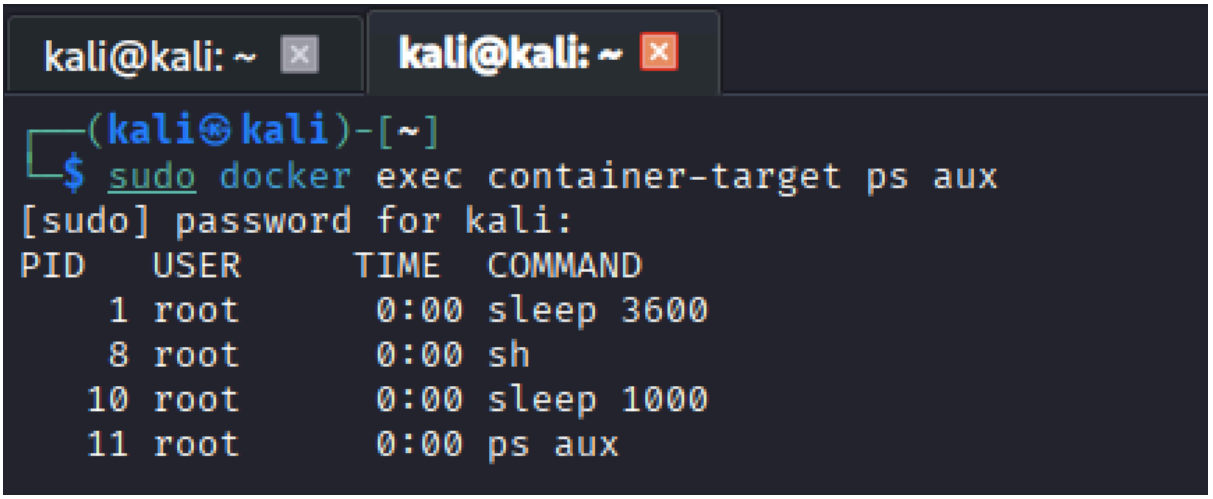
Una volta lanciata la shell all'interno dei namespace agganciati (con -a prendo tutti i namespace), ho eseguito ps aux.



Risultato: il processo che fuori era 4945, qui dentro appare come PID 1. Questo perchè sarebbe il processo di init del container.
Il PID Namespace ha funzionato correttamente, fornendo al processo una vista virtualizzata degli ID.

4. ULTERIORE PROCESSO

Per completare l'esercizio, ho lanciato un processo "sleep 1000" dalla sessione nsenter (come si può vedere sopra). Poi, da un secondo terminale sul Host, ho interrogato il container con "docker exec".



Risultato: Il comando "docker exec" mostra correttamente il processo "sleep 1000" con PID 10.

Conclusione: A differenza di quanto osservato su Mac, dove l'architettura richiede uno strato di virtualizzazione intermedio, su Linux è stato possibile osservare come i container siamo a tutti gli effetti processi del sistema operativo a cui vengono applicati dei filtri (Namespaces).
L'uso di nsenter mi ha fatto capire che le barriere di un container non sono impermeabili dato che, conoscendo il PID del processo su Host, è possibile "teletrasportare" un processo all'interno del contesto isolato.