ZAKŁAD SYSTEMÓW ZŁOŻONYCH

Cyberbezpieczeństwo

Laboratorium 7

Docker – Wprowadzenie do bezpieczeństwa

W sprawozdaniu zamieść zrzuty ekranu z istotnych etapów oraz odpowiedzi na pytania.

Taki symbol oznacza, że trzeba w sprawozdaniu dodać zrzut ekranu (najczęściej 1) z wyniku działania polecenie.

Taki symbol oznacza, że należy dodać opis (najczęściej 1 zdanie) z wyniku działania polecenia.

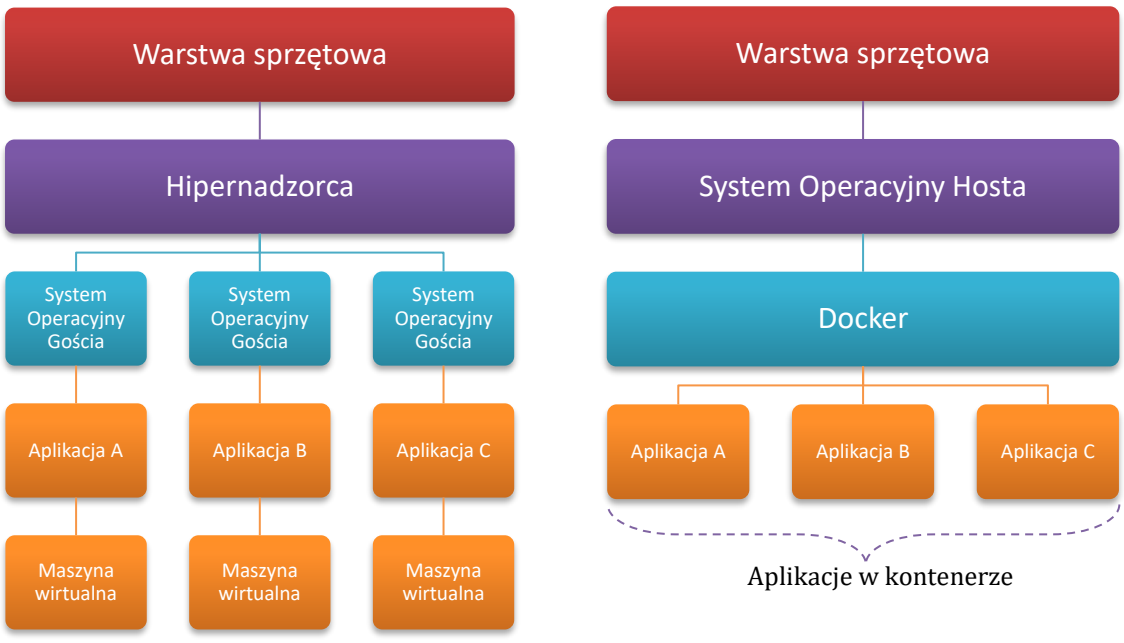
Wstęp

Początki wirtualizacji sięgają lat 60. XX wieku, kiedy IBM opracował system logicznego podziału zasobów komputerów mainframe o nazwie CP/CMS. Mechanizm ten umożliwiał efektywniejsze wykorzystanie zasobów sprzętowych, minimalizując konieczność uruchamiania kolejnych urządzeń. W efekcie ograniczono zużycie energii oraz wprowadzono separację środowisk, dzięki czemu błędy w jednej wirtualnej maszynie nie wpływały na działanie innych instancji.

W 2013 roku, dzięki inicjatywie Solomona Hykesa, powstało nowe narzędzie – Docker. W odróżnieniu od tradycyjnych mechanizmów wirtualizacji, które opierają się na emulacji całego sprzętu komputerowego, Docker wykorzystuje tzw. kontenery. Ta lekka architektura pozwala na efektywniejsze wykorzystanie zasobów systemowych, umożliwiając uruchomienie wielu aplikacji na tej samej infrastrukturze bez symulowania sprzętu. W rezultacie zmniejsza się obciążenie systemu.

W modelu wirtualizacji (po lewej) hipernadzorca, zarządzający procesem, działa bezpośrednio na warstwie sprzętowej. Na nim tworzone są odseparowane maszyny wirtualne. Każda maszyna wirtualna zawiera pełną kopię systemu operacyjnego, aplikacji, niezbędnych plików binarnych i bibliotek – zajmując dziesiątki GB. Maszyny wirtualne mogą się również wolno uruchamiać..

Z kolei Docker działa w ramach infrastruktury, na której zainstalowany jest system operacyjny hosta (np. Linux). Wykorzystując jądro hosta, Docker tworzy odseparowane środowiska dla uruchamianych aplikacji. Dzięki temu, że nie ma potrzeby wirtualizacji komponentów systemu gościa, możliwe jest znaczne ograniczenie zużycia zasobów przy uruchamianiu tych samych aplikacji.



Rysunek 1. Porównanie wirtualizacji i konteneryzacji.*

* https://www.docker.com/resources/what-container/

I. Dockerfile

Jednym z podstawowych elementów Dockera są pliki Dockerfile, które zawierają polecenia używane do budowy obrazów. Zanim przejdziemy do tworzenia obrazu, zainstalujmy Dockera.

1. Uruchom maszynę wirtualną opartą na Kali Linux.

2. Otwórz terminal.

3. Zaktualizuj listę pakietów:

```
sudo apt update
```

4. Zainstaluj Dockera:

```
sudo apt install -y docker.io
```

 5. Sprawdź wersję Dockera:

```
sudo docker --version
```

6. Aby używać Dockera bez konieczności wpisywania `sudo`, dodaj użytkownika do grupy `docker`:

```
sudo usermod -aG docker $USER
```

7. Uruchom ponownie maszynę wirtualną, aby zastosować zmiany.

8. Otwórz edytor `nano` i utwórz plik `Dockerfile`:

```
nano Dockerfile
```

9. Wypełnij plik następującym kodem:

```
FROM alpine:3.23.2
RUN apk update && apk add --no-cache \
    bash \
    curl
RUN adduser -D limited -s /bin/bash
USER limited:limited
WORKDIR /app
COPY . /app
```

Pierwsze polecenie `FROM alpine:3.23.2` wskazuje obraz bazowy, który będzie podstawą do budowy kolejnych warstw.

Polecenie `RUN apk update && apk add --no-cache bash curl` odpowiada za:

- aktualizację repozytoriów pakietów,
- instalację `bash` i `curl`.

Te komendy zostaną wykonane podczas budowy obrazu.

Kolejne polecenia:

- `RUN adduser -D limited -s /bin/bash` – tworzy nowego użytkownika o nazwie `limited`,
- `USER limited:limited` – przełącza kontekst na konto tego użytkownika.

Na końcu:

- `WORKDIR /app` ustawia katalog roboczy na `/app`,
- `COPY . /app` kopiuje całą zawartość bieżącego katalogu (na maszynie hosta) do katalogu `/app` w kontenerze.

Każdy obraz Dockera może mieć przypisany dowolny tag – w tym przypadku jest to `3.23.2`.

Warto zwrócić uwagę na tag `latest`, który może być mylący, ponieważ nie zawsze odnosi się do najnowszej wersji oprogramowania. Dlatego zaleca się ręczne określanie wersji, jak w tym przykładzie.

10. Zbuduj obraz Dockera o nazwie cyberbezpieczenstwo, nadając mu tag odpowiadający Twojemu numerowi indeksu:

```
docker build . -t cyberbezpieczenstwo:twoj_numer_indeksu
```

```
┌──(kali㉿kali)-[~]
└─$ docker build . -t cyberbezpieczenstwo:112233
[+] Building 5.9s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 199B
 => [internal] load metadata for docker.io/library/alpine:3.23.2
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/alpine:3.23.2@sha256:865b95f46d98cf867a156fe4a135ad3fe50d2056aa3f25ed31662dff6da4eb62
 => => resolve docker.io/library/alpine:3.23.2@sha256:865b95f46d98cf867a156fe4a135ad3fe50d2056aa3f25ed31662dff6da4eb62
 => => sha256:1882fa4569e0c591ea092d3766c4893e19b8901a8e649de7067188aba3cc0679 1.02kB / 1.02kB
 => => sha256:e7b39c54cdeca0d2aae83114bb605753a5f5bc511fe8be7590e38f6d9f915dad 611B / 611B
 => => sha256:1074353eec0db2c1d81d5af2671e56e00cf5738486f5762609ea33d606f88612 3.86MB / 3.86MB
 => => sha256:865b95f46d98cf867a156fe4a135ad3fe50d2056aa3f25ed31662dff6da4eb62 9.22kB / 9.22kB
 => => extracting sha256:1074353eec0db2c1d81d5af2671e56e00cf5738486f5762609ea33d606f88612
 => [internal] load build context
 => => transferring context: 37.35MB
 => [2/5] RUN apk update && apk add --no-cache     bash     curl
 => [3/5] RUN adduser -D limited -s /bin/bash
 => [4/5] WORKDIR /app
 => [5/5] COPY . /app
 => exporting to image
 => => exporting layers
 => => writing image sha256:ac0b322c3b3c8239079f2f0ef4d4a109c7ed334c759e311641f3c2569c626221
 => => naming to docker.io/library/cyberbezpieczenstwo:112233
```

11. Sprawdź utworzony obraz:

```
docker images
```

```
┌──(kali㉿kali)-[~]
└─$ docker images
REPOSITORY            TAG       IMAGE ID       CREATED          SIZE
cyberbezpieczenstwo   112233    7ec8f5d3f946   6 minutes ago    959MB
```

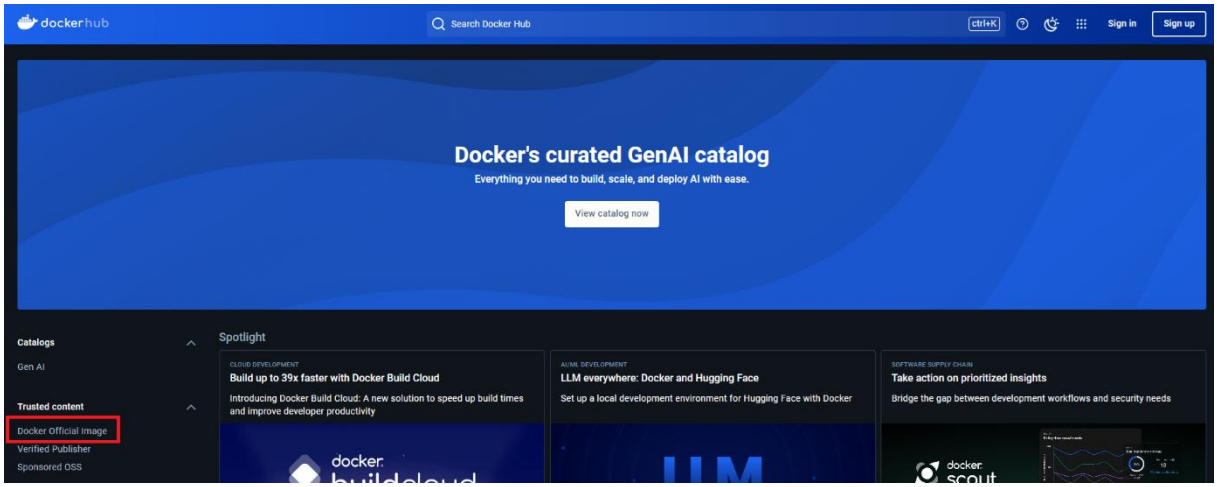
Przed uruchomieniem pierwszego kontenera warto zwrócić uwagę na potencjalne zagrożenia, które mogą się pojawić:

- **Nieaktualne oprogramowanie** – może zawierać znane podatności lub błędy w konfiguracji.
- **Złośliwe oprogramowanie (malware)** – może być umieszczone w obrazie lub pobierane podczas jego uruchamiania.

Aby zwiększyć pewność co do bezpieczeństwa wybranego obrazu, należy go zweryfikować jeszcze przed pobraniem. Najpopularniejszym repozytorium obrazów jest Docker Hub, które oferuje szeroką gamę obrazów. Niestety, nawet tam mogą znajdować się obrazy zawierające błędy bezpieczeństwa. Dlatego zaleca się korzystanie wyłącznie z oficjalnych i zweryfikowanych obrazów.

12. Otwórz przeglądarkę i przejdź do Docker Hub.

13. Filtruj tylko oficjalne obrazy, zaznaczając opcję `DOCKER OFFICIAL IMAGE`.



14. Przejrzyj dostępne obrazy.

Mimo to, jak omawialiśmy na pierwszych zajęciach z utwardzania Linuksa, nawet oficjalne obrazy mogą być zainfekowane. Jednym z rozwiązań jest weryfikacja warstw, z których składa się obraz, oraz przeskanowanie go pod kątem obecności złośliwego kodu za pomocą narzędzi takich jak `Snyk` czy `Trivy`. W tym przypadku skorzystamy z narzędzia `Trivy`.

15. Pobierz skaner `Trivy`:

```
wget https://github.com/aquasecurity/trivy/releases/download/v0.68.2/trivy_0.68.2_Linux-64bit.deb
```

16. Zainstaluj skaner `Trivy`:

```
sudo dpkg -i trivy_0.68.2_Linux-64bit.deb
```

17. Przeskanuj obraz `alpine:3.23.2`:

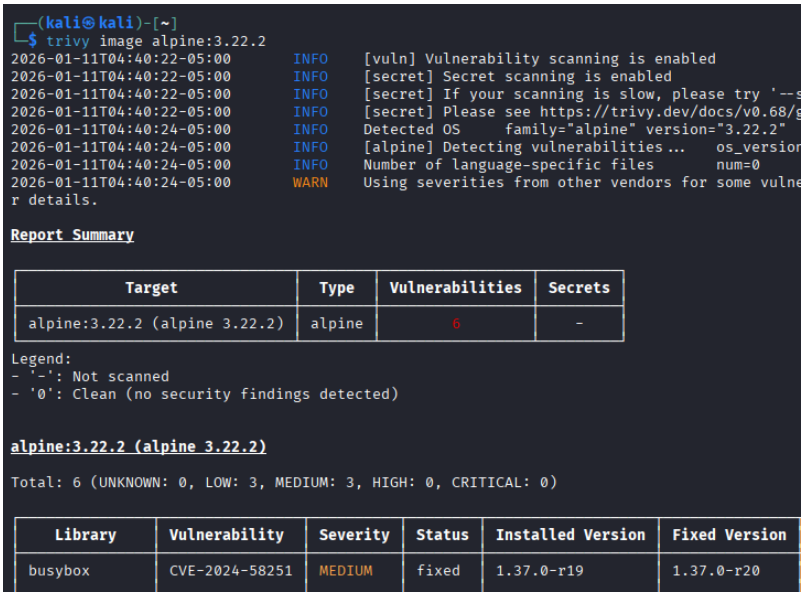
```
trivy image alpine:3.23.2
```

```
┌──(kali㉿kali)-[~]
└─$ trivy image alpine:3.23.2
2026-01-11T04:38:00-05:00       INFO    [vuln] Vulnerability scanning is enabled
2026-01-11T04:38:00-05:00       INFO    [secret] Secret scanning is enabled
2026-01-11T04:38:00-05:00       INFO    [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2026-01-11T04:38:00-05:00       INFO    [secret] Please see https://trivy.dev/docs/v0.68/guide/scanner/secret#recommendation for faster secret detection
2026-01-11T04:38:02-05:00       INFO    Detected OS     family="alpine" version="3.23.2"
2026-01-11T04:38:02-05:00       WARN    This OS version is not on the EOL list  family="alpine" version="3.23"
2026-01-11T04:38:02-05:00       INFO    [alpine] Detecting vulnerabilities ...  os_version="3.23" repository="3.23" pkg_num=16
2026-01-11T04:38:02-05:00       INFO    Number of language-specific files       num=0

Report Summary

┌──────────────────────────────┬────────┬─────────────────┬─────────┐
│            Target            │  Type  │ Vulnerabilities │ Secrets │
├──────────────────────────────┼────────┼─────────────────┼─────────┤
│ alpine:3.23.2 (alpine 3.23.2)│ alpine │        0        │    -    │
└──────────────────────────────┴────────┴─────────────────┴─────────┘
Legend:
- '-': Not scanned
- '0': Clean (no security findings detected)
```

Jak można zauważyć, obecna wersja obrazu wydaje się bezpieczna. Sprawdźmy jednak, jak wyglądała sytuacja w przypadku wcześniejszej wersji, oznaczonej jako `3.20.0`.

18. Porównaj z poprzednią wersją obrazu:

```
trivy image alpine:3.22.2
```



Jak widać, poprzednia wersja obrazu okazała się podatna na zagrożenia. Po zakończeniu weryfikacji możemy teraz przystąpić do uruchomienia kontenera.

19. Uruchom kontener:

```
docker run -it cyberbezpieczenstwo:twoj_numer_indeksu /bin/bash
```

20. Sprawdź, czy pliki zostały prawidłowo skopiowane:

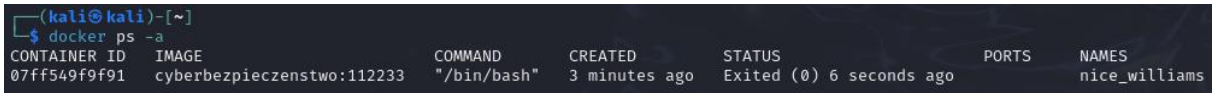
```
ls
```

21. Wyjdź z kontenera:

```
exit
```

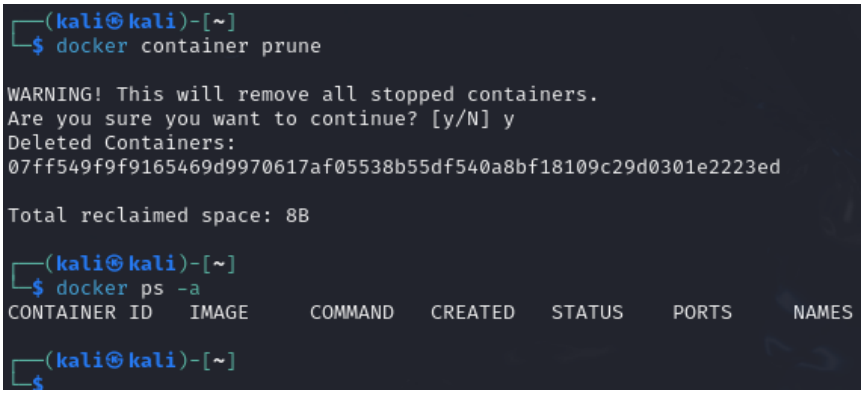
22. Sprawdź, czy kontener został zamknięty (nie działa w tle):

```
docker ps -a
```



23. Usuń wszystkie utworzone kontenery:

```
docker container prune
```



II. Typowe problemy i błędy bezpieczeństwa dotyczące Dockera

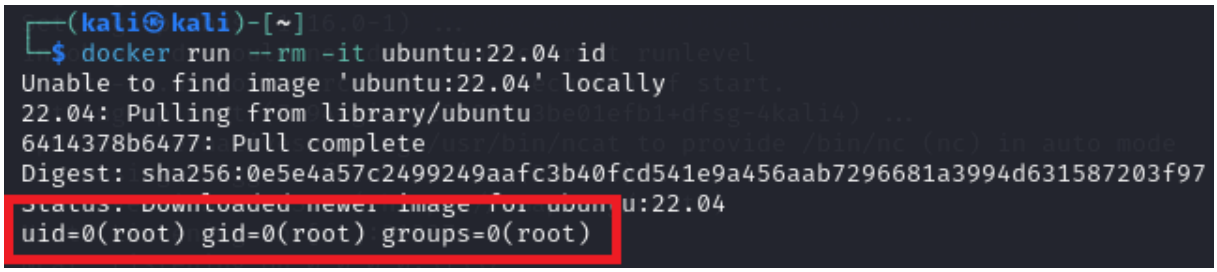
Docker nie działa w izolacji – jest usługą uruchamianą na określonym systemie operacyjnym. Dlatego to właśnie system operacyjny jest pierwszym i podstawowym elementem, o który należy zadbać. W środowisku pentesterów jedną z najczęściej wykrywanych podatności jest nieaktualny system operacyjny z jądrem zawierającym znane błędy bezpieczeństwa, które umożliwiają atak na hosta z poziomu kontenera. Podobnie brak aktualizacji samego Dockera może prowadzić do przejęcia infrastruktury. W związku z tym należy pamiętać o regularnym aktualizowaniu systemu i oprogramowania oraz wdrażaniu zasad utwardzania.

Kontenery z uprawnieniami administratora

Jak wielokrotnie podkreślano na zajęciach, należy unikać pracy na Linuksie jako użytkownik root w maksymalnym możliwym stopniu. Ta zasada dotyczy również Dockera. Warto jednak pamiętać, że domyślnie procesy uruchamiane w nowym kontenerze działają w kontekście użytkownika root. Sprawdźmy to.

1. Uruchom kontener na podstawie obrazu `ubuntu:22.04` z poleceniem `id`:

```
docker run --rm -it ubuntu:22.04 id
```



Jak widać, procesy w kontenerze uruchamiane są z uprawnieniami roota. Jest to sytuacja, którą należy unikać, ponieważ może prowadzić do ataków typu RCE (Remote Code Execution), które pozwalają na uruchomienie kodu w kontekście aplikacji, traktowanego jak jej część. Taki atak umożliwia przejęcie uprawnień i przywilejów aplikacji. Można jednak zminimalizować ryzyko, uruchamiając procesy w kontenerze z minimalnymi, niezbędnymi uprawnieniami.

2. Uruchom proces w kontekście użytkownika `www-data`:

```
docker run --rm --user www-data -it ubuntu:22.04 id
```

```
┌──(kali㉿kali)-[~]
└─$ docker run --rm --user www-data -it ubuntu:22.04 id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Płaska sieć

Administratorzy często nie zmieniają domyślnych ustawień sieciowych (problem ten dotyczy nawet dużych systemów), w wyniku czego kontenery są zwykle konfigurowane w taki sposób, że znajdują się w tym samym segmencie sieci, umożliwiając im wzajemną komunikację. Taki sposób konfiguracji jest standardowy w przypadku domyślnego sterownika sieciowego Dockera. Czy jest to jednak podejście właściwe? To kwestia dyskusyjna. Z pewnością jednak nie jest to optymalne rozwiązanie z perspektywy bezpieczeństwa. Sprawdźmy, czy kontenery rzeczywiście domyślnie mogą się wzajemnie "widzieć".

3. Uruchom trzy instancje terminala.

4. W pierwszym terminalu wpisz następującą komendę, aby utworzyć kontener:

```
docker run --rm -it --name "kali1" kalilinux/kali-rolling bash -c "apt-get update && apt-get install -y ncat && ncat -lvp 31337"
```

```
update-alternatives: using /usr/bin/ncat to provi
Processing triggers for libc-bin (2.40-3) ...
Ncat: Version 7.94SVN ( https://nmap.org/ncat )
Ncat: Listening on [::]:31337
Ncat: Listening on 0.0.0.0:31337
```

Powyższa komenda uruchamia nowy kontener na podstawie obrazu Kali Linux w wersji `rolling`, instaluje narzędzie `ncat`, a następnie ustawia je w trybie nasłuchiwania na porcie `31337`.

5. Przejdź do drugiego terminala. Zainstaluj narzędzie jq (do przetwarzania danych w formacie JSON):

```
sudo apt install jq -y
```

6. Wykonaj weryfikację adresu IP kontenera uruchomionego w domyślnej sieci bridge:

```
docker inspect bridge | jq '.[0].Containers'
```

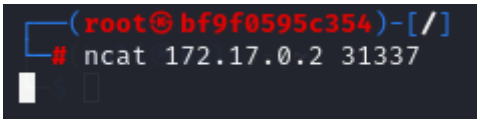
```
┌──(kali㉿kali)-[~]
└─$ docker inspect bridge | jq '.[0].Containers'
{
  "94ee850d0ec4e46e1a875fa7fe2ff991025d104019f2239e20c3e86e11ef0e36": {
    "Name": "kali1",
    "EndpointID": "d716c113a742de1b62cde637ac8f0e3534e074c90bec56ff239481e7ce1da759",
    "MacAddress": "02:42:ac:11:00:02",
    "IPv4Address": "172.17.0.2/16",
    "IPv6Address": ""
  }
}
```

7. Przejdź do trzeciego terminala i uruchom kolejny kontener:

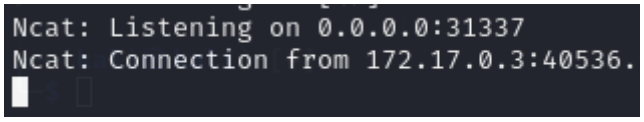
```
docker run --rm -it --name "kali2" kalilinux/kali-rolling bash -c "apt-get update && apt-get install -y ncat && bash"
```

8. Nawiąż połączenie z drugim kontenerem przy użyciu narzędzia `ncat`:

```
ncat adres_ip_pierwszego_kontenera port
```



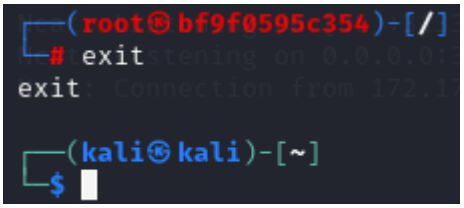
9. Upewnij się, że komunikacja faktycznie nastąpiła. Przejdź do pierwszego terminala.



Udowodniliśmy, że domyślnie Docker konfiguruje sieć w taki sposób, aby kontenery były w tym samym segmencie. Ograniczmy takie zachowanie poprzez umieszczenie kontenerów w osobnych podsieciach.

10. Przejdź do trzeciego terminala i przerwij komunikację kombinacją klawiszy `Ctrl + C`.

11. W tym samym terminalu wyjdź z kontenera komendą `exit`.



12. Następnie utwórz dwie nowe sieci typu most:

```
docker network create net-kali1
docker network create net-kali2
```

Teraz należy uruchomić kontenery, wskazując, w jakiej sieci mają się znaleźć.

13. Przejdź do pierwszego terminala i uruchom następującą komendę:

```
docker run --rm -it --name "kali1" --network "net-kali1" kalilinux/kali-rolling bash
-c "apt-get update && apt-get install -y ncat && ncat -lvp 31337"
```

Jak widzisz, kontener ponownie przeszedł w tryb nasłuchiwania.

14. Przejdź do drugiego terminala i uruchom drugi kontener:

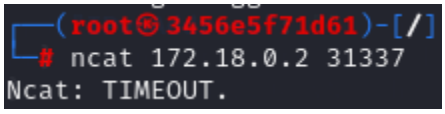
```
docker run --rm -it --name "kali2" --network "net-kali2" kalilinux/kali-rolling bash
-c "apt-get update && apt-get install -y ncat && bash"
```

15. Przejdź do terminala numer 3 i wykonaj poniższe komendy, aby sprawdzić, w jakich podsieciach znajdują się kontenery:

```
docker inspect net-kali1 | jq '.[0].Containers'
docker inspect net-kali2 | jq '.[0].Containers'
```

Znający podstawy sieci komputerowych już teraz widzą, że komunikacja między kontenerami jest niemożliwa. Mimo to, upewnijmy się, że tak jest...

16. Przejdź do drugiego terminala i spróbuj nawiązać komunikację z kontenerem `kali1`, podobnie jak w punkcie 2.8. Komunikacja nie powinna być możliwa.



Brak ograniczeń zasobów

Wyobraźmy sobie sytuację, w której kontener z aplikacją zostaje zaatakowany, a napastnik uzyskuje dostęp do niego. Jeśli dostęp do zasobów nie został określony, kontener będzie mógł wykorzystać wszystkie zasoby pamięciowe/CPU hosta. Zasymulujmy taką sytuację, używając tzw. zip bomby. Atak polega na rozpakowaniu odpowiednio przygotowanego pliku .zip, co powoduje zajęcie całej dostępnej pamięci operacyjnej, procesora lub przestrzeni dyskowej.

17. Wyjdź z kontenerów i zamknij wszystkie terminale.

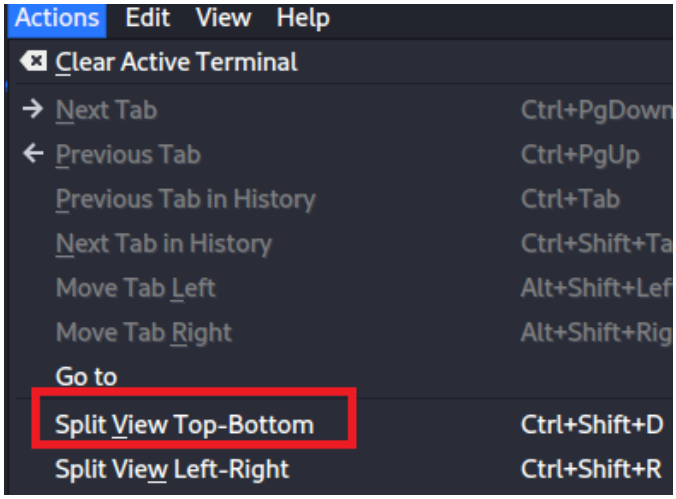
18. Sprawdź, czy masz uruchomione jakieś kontenery (ppkt 1.22). Jeśli tak, zatrzymaj je i usuń.

19. Uruchom terminal Kali Linux.

20. Zainstaluj monitor zasobów `htop`:

```
sudo apt install htop
```

21. W menu terminala wybierz `Actions -> Split View Top-Bottom`.

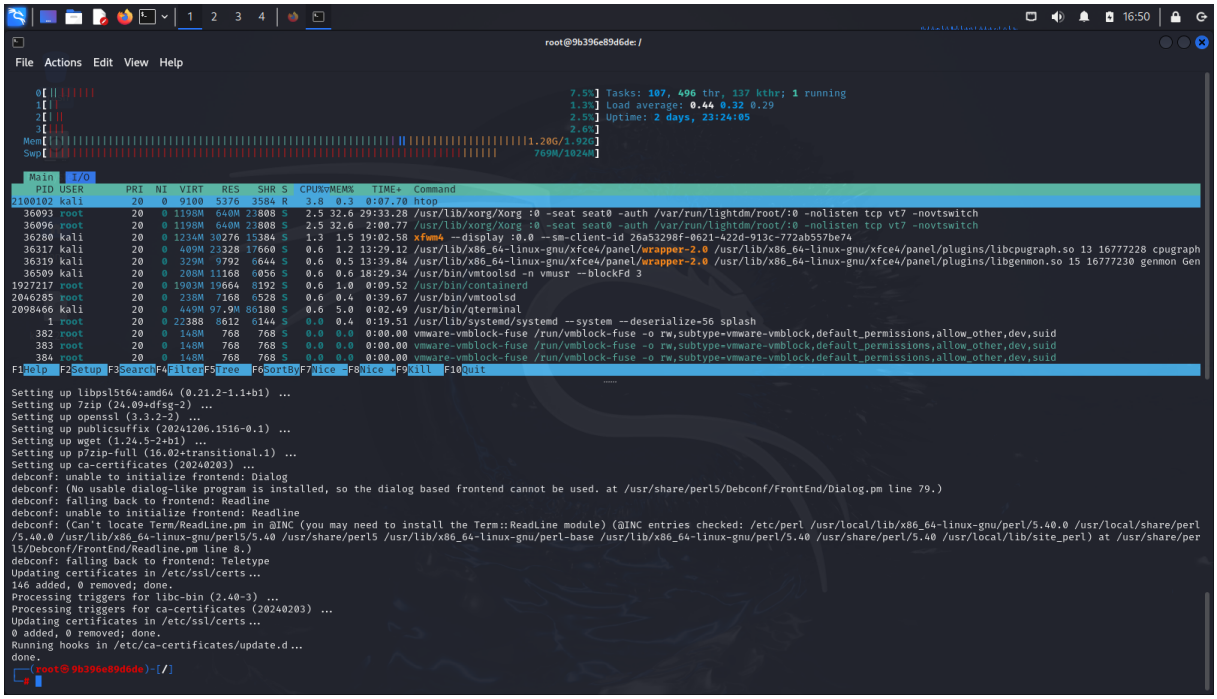


22. W górnej części terminala uruchom `htop`:

```
htop
```

23. W dolnej części uruchom kontener z Kali Linux:

```
docker run --rm -it kalilinux/kali-rolling bash -c "apt-get update && apt-get install -y p7zip-full wget && bash"
```



24. Pobierz zip bombę w kontenerze:

```
wget https://www.bamsoftware.com/hacks/zipbomb/zbsm.zip
```

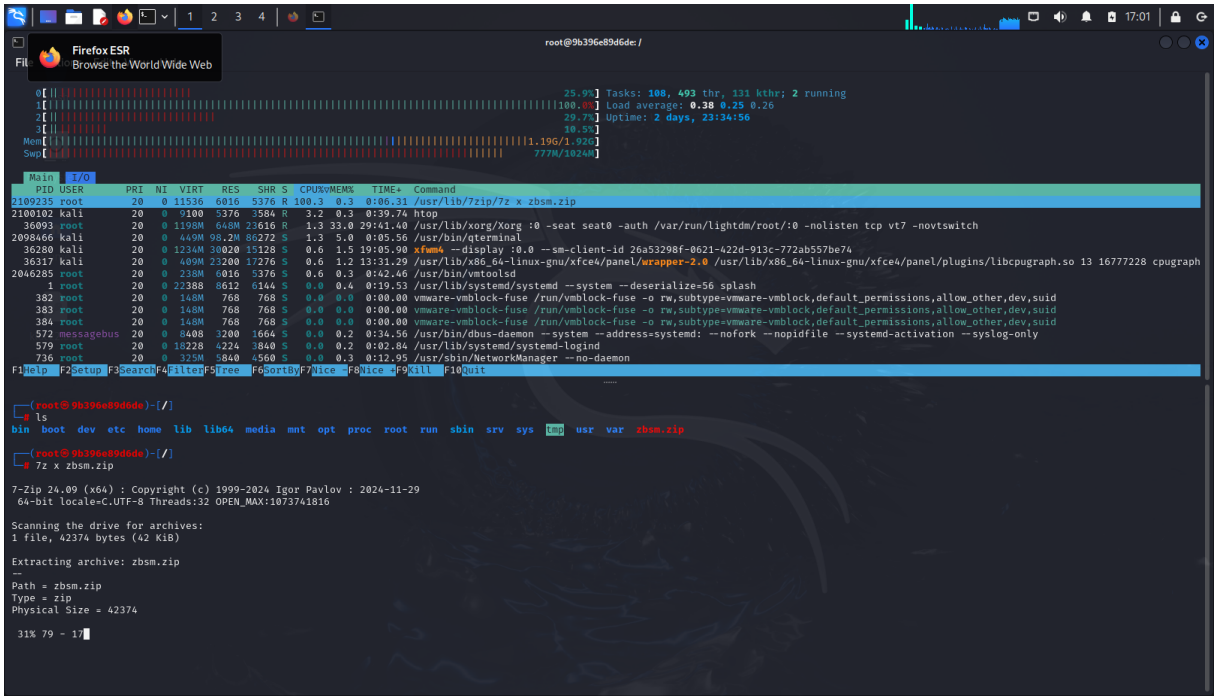
Uwaga: Zip bomba może całkowicie „zapchać” twój system operacyjny. W tym przypadku pobieramy 42 kB plik, który po wypakowaniu zajmuje 5 GB miejsca na dysku twardym! Istnieją również warianty, np. 10 MB, które po wypakowaniu zajmują 281 TB. **Zawsze działaj na maszynie wirtualnej!**

Uruchamianie zip bomby w sposób niekontrolowany jest nieodpowiedzialne i może być nielegalne, jeśli robisz to na cudzym sprzęcie lub w sieci!!!

Jeśli interesuje Cię temat zip bomb, możesz zapoznać się z artykułem Davida Fifielda, autorem jednej z najbardziej znanych i powszechnie cytowanych prac na temat tego typu ataku. Artykuł jest dostępny pod tym linkiem: https://www.bamsoftware.com/hacks/zipbomb/

25. Przyjrzyj się aktualnemu zachowaniu procesora, a następnie uruchom wypakowanie zip bomby:

```
7z x zbsm.zip
```



Jak widzisz, obciążenie procesora znacznie wzrosło – na poszczególnych rdzeniach procesora często osiąga 100%. Oczywiście wartości te będą się różnić w zależności od posiadanego sprzętu. Aby zapobiec takim sytuacjom, musimy skonfigurować ograniczenia nakładane na kontener.

26. Jeżeli proces wypakowania trwa zbyt długo, przerwij go kombinacją klawiszy `Ctrl + C`.

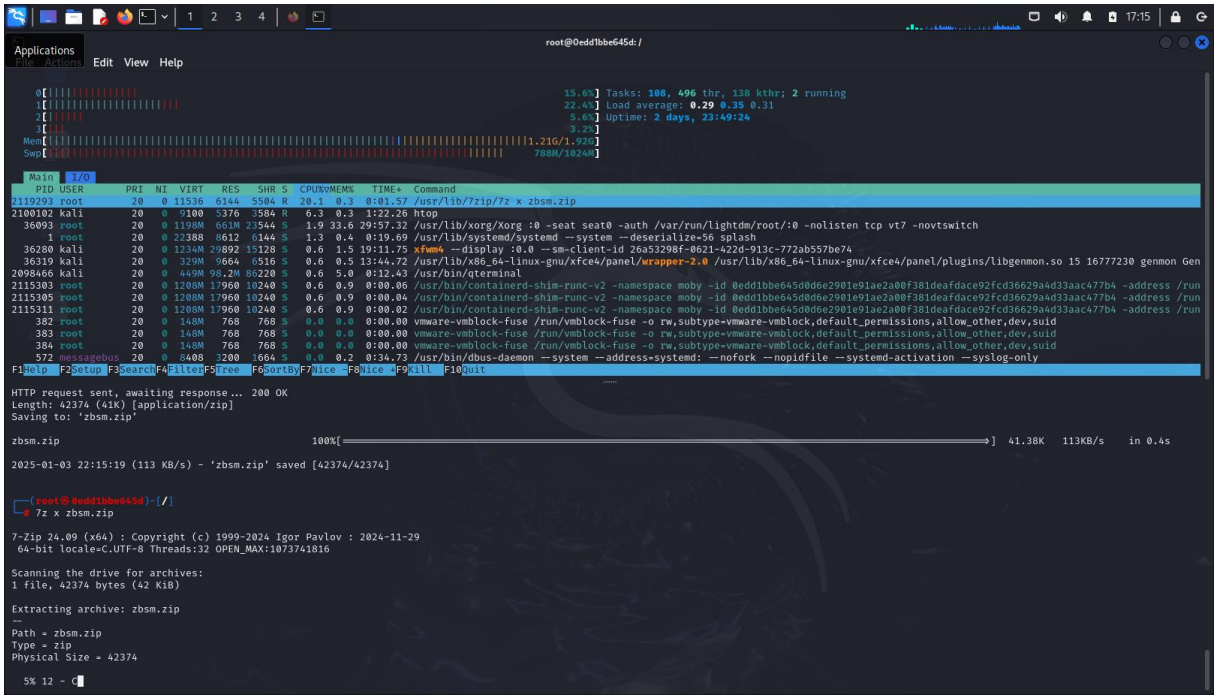
27. Zamknij kontener komendą `exit`.

28. Ponownie uruchom kontener, ale z dodatkowymi parametrami, które ograniczą dostęp do pamięci do poziomu 150 MB oraz do 20% wykorzystania procesora:

```
docker run --rm -it --memory="150m" --cpus="0.2" kalilinux/kali-rolling bash -c "apt-get update && apt-get install -y p7zip-full wget && bash"
```

29. Powtórz kroki 2.24 oraz 2.25.

30. Obserwuj zachowanie procesora. Jak zauważysz, obciążenie rdzeni oscyluje wokół 20%, co jest zgodne z oczekiwaniami.



31. Jeżeli proces wypakowania trwa zbyt długo, przerwij go kombinacją klawiszy `Ctrl + C`, a następnie wyjdź z kontenera poleceniem `exit`.

Uprawnienia R/W na plikach hosta

Często przy uruchamianiu usług w kontenerze należy przekazać pliki konfiguracyjne znajdujące się na hoście. Przekazywanie konfiguracji opiera się na podpięciu wybranego pliku z hosta do kontenera. Zasymulujmy tę sytuację.

32. Utwórz plik `config.conf` i wpisz do niego swój numer indeksu.

33. Uruchom kontener z podpiętym plikiem konfiguracyjnym, który modyfikuje zawartość pliku:

```
docker run --rm -v /home/kali/config.conf:/app/config.conf -it ubuntu:22.04 bash -c 'echo "Zhakowany" >> /app/config.conf'
```

34. Wyświetl zawartość pliku:

Jak widać, kontener mógł zmodyfikować plik znajdujący się na hoście, więc możliwa jest nieuprawniona zmiana działania usługi w kontenerze. W tym przypadku to był tylko plik konfiguracyjny, ale co, gdyby podmontowany był główny katalog hosta (`/`) lub katalog konfiguracyjny (`/etc`)?

Jeżeli mamy pewność, że edytowanie danego pliku nie jest wymagane, należy stosować tryb tylko do odczytu.

📷 35. Uruchom zmodyfikowane polecenie z poprzedniego punktu z parametrem `:ro` (tylko do odczytu):

```
docker run --rm -v /home/kali/config.conf:/app/config.conf:ro -it ubuntu:22.04 bash -c 'echo "Zhakowany2" >> /app/config.conf'
```

Jak widać, zabezpieczenie spełniło swoją rolę, ale dla pewności sprawdźmy ponownie zawartość pliku.

📷 36. Wyświetl zawartość `config.conf`:

```
cat config.conf
```