Sigurnost operacijskih sustava i aplikacija

Docker

Petra Kelković, 4.4.2025.

Pregled predavanja

- Pitanja za ispite i motivacija
- Virtualizacija
- Docker
- Analiza sigurnosti Dockera
- Mehanizmi za poboljšanje sigurnosti
- Zaključak i literatura

Pitanja za ispite

- Kako se postiže izolacija procesa između kontejnera?
- Koja je uloga opcije *nodev* prilikom montiranja datotečnog sustava kontejnera?
- Kako se ostvaruje mrežna veza između Docker kontejnera i koje je njezino sigurnosno ograničenje?
- Koje dvije vrste sustava za dodatnu zaštitu jezgre postoje i na koji način oni pružaju zaštitu?
- Objasnite način rada sigurnosnog modela AppArmour.

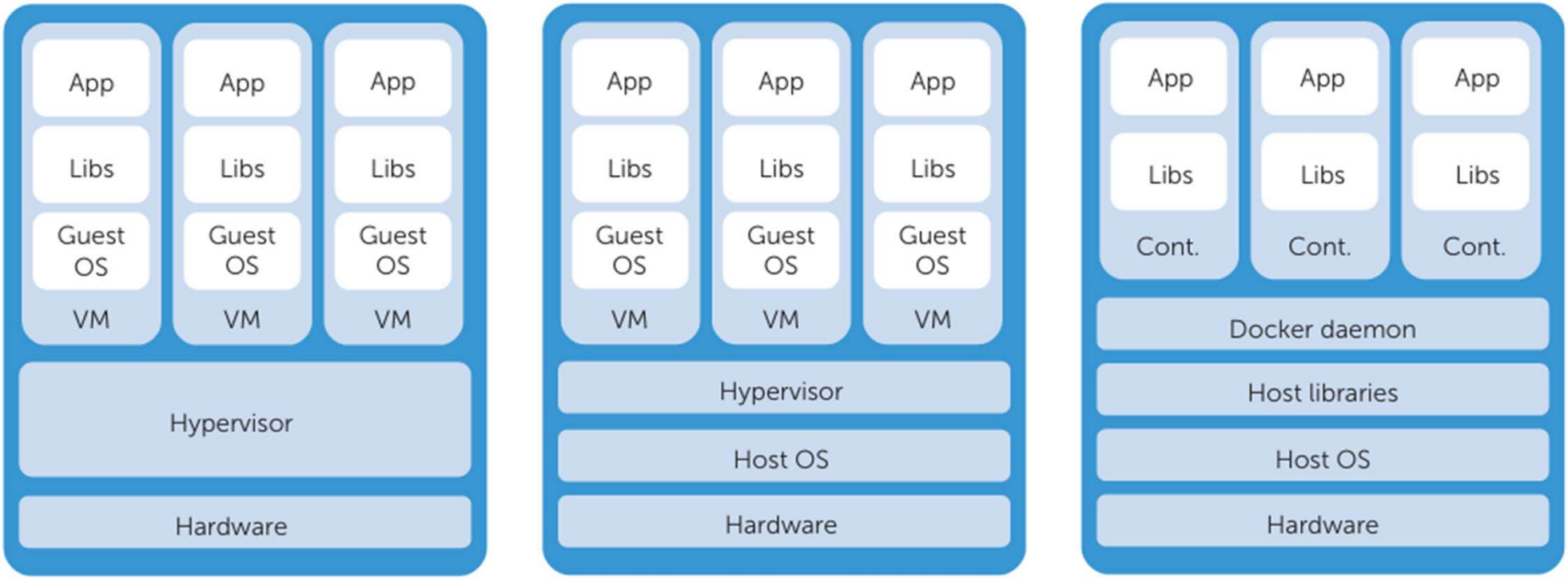
Motivacija

- Ubrzan rast i razvoj tehnologija kontejnera u zadnjem desetljeću
  - Slabija izolacija procesa nego kod “klasične virtualizacije” → veći sigurnosni rizici
- Velika popularnost alata Docker
  - Upoznavanje s glavnim (sigurnosnim) problemima koji se javljaju u Docker okruženju može pomoći u zaštiti rada kontejnera i sustava domaćina
- Shvatiti važnost sigurnosnih dodataka za poboljšanje zaštite kontejnera kako jednog dana ne bismo postali žrtve napada

Virtualizacija

- **Virtualizacija** – tehnika koja omogućuje dijeljenje jednog fizičkog sredstva između više korisnika stvaranjem virtualnih kopija tog sredstva
- Razlozi: više (različitih) instanci operacijskih sustava na istom računalu, virtualizacija poslužitelja, bolje iskorištavanje računalnih resursa
- Dvije glavne vrste su **virtualni strojevi** i **kontejneri**

Virtualni strojevi vs kontejneri (1)

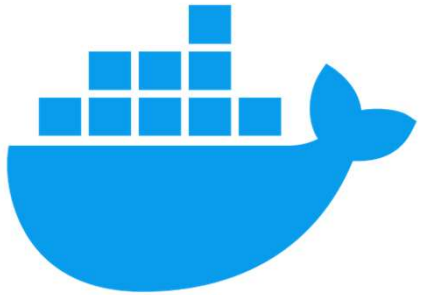


Slika 1: arhitekturne razlike između računala s virtualnim strojevima (lijevo i sredina) i kontejnerima (desno)

Virtualni strojevi vs kontejneri (2)

- **Virtualni strojevi** – virtualizacija na razini sklopovlja, sadrže vlastiti operacijski sustav, njima upravlja hipervizor
- **Kontejneri** – virtualna okruženja koja se izvršavaju na operacijskom sustavu domaćina, njima upravlja virtualni sloj koji se nalazi između OS-a domaćina i aplikacije (kod Dockera → Docker *daemon*)
- **Prednosti kontejnera**: brže pokretanje, zauzimaju manje računalnih resursa, na jedno računalo stane veći broj kontejnera nego VS-ova, bolje performanse
- **Nedostatci kontejnera**: slabija međusobna izolacija, slaba kontrola nad korištenjem resursa, slabija izolacija aplikacije od domaćina (sloj manje nego kod VS-ova)

Docker



- Softverska platforma koja omogućuje izgradnju, isporuku i testiranje (raspodijeljenih) aplikacija korištenjem tehnologije kontejnera
- Izvršava se na Linux operacijskom sustavu → za izvršavanje na ostalim OS-ovima koristi se Linux virtualni stroj
- Vrlo popularan i široko korišten
  - Spotify, Uber, PayPal, Visa, Netflix
- Dvije glavne komponente
  - Docker *engine* – alat koji se bavi pakiranjem, pokretanjem i općenito upravljanjem kontejnerima
  - Docker *hub* - SaaS platforma za dijeljenje Docker slika (eng. *Image*)

Docker – osnovni pojmovi

- ***Docker daemon*** –glavni proces Docker *enginea* koji je odgovoran za upravljanje i izvršavanje kontejnera
- ***Docker client*** – pruža korisničko sučelje kako bi korisnici mogli upravljati kontejnerima, posrednik u komunikaciji korisnik ↔ *daemon*
- ***Docker image*** – serija podatkovnih slojeva povrh bazne slike operacijskog sustava, sadrži sve što je potrebno za pokretanje aplikacije
- ***Docker container*** – izolirani process u izvođenju, aktivna instanca slike

Docker – analiza sigurnosti (1)

- **Model napada**: *na računalu se izvršava N Docker kontejnera, od kojih je K kompromitirano i napadač ima punu kontrolu nad njima, a ostalih N-K kontejnera su i dalje pod kontrolom legitimnih korisnika*

- **Minimalni zahtjevi za mogućnost obrane od takvih napada**
  - Izolacija procesa
  - Izolacija datotečnog sustava
  - Ograničenje pristupa resursima
  - Izolacija uređaja
  - Izolacija međuprocesne komunikacije
  - Izolacija mrežnih resursa

Izolacija procesa (1)

- **Cilj:** spriječiti kompromitirane kontejnere da preko sučelja za upravljanje procesima naštete drugim kontejnerima
- To se postiže korištenjem mehanizma razdvojenog imenskog prostora za identifikacijske brojeve procesa jezgre Linux
  - taj mehanizam nazivamo ***PID namespace***

Izolacija procesa (2)

- ***PID namespace*** – izolira identifikacijske brojeve procesa između različitih okruženja
  - Svaki kontejner ima vlastiti prostor PID brojeva procesa
  - Procesi unutar kontejnera mogu vidjeti samo sebe i svoju djecu procese, ali ne vide izvan svog prostora
  - ***Napadači ne mogu vidjeti druge procese, pa ih je stoga i teže napasti***
- Svaki kontejner ima *init* proces s identifikacijskim brojem 1 koji svojim prekidom izvršavanja terminira sve ostale procese unutar kontejnera
  - Administrator može lako ugasiti “sumnjive” kontejnere

Izolacija datotečnog sustava

- Koristi se mehanizam razdvojenog imenskog prostora za točke montiranja datotečnog sustava jezgre Linux
- ***Mount namespace*** – imenski prostor koji ima jedinstven pogled na strukturu datotečnog sustava te koji ima svojstvo da montiranje novih datotečnih sustava utječe samo na taj imenski prostor
  - **Svaki kontejner je jedan *mount namespace***
- Dio datoteka jezgre nije uključen u imenski prostor
  - Problematično jer im kontejner mora imati pristup za ispravan rad, no pošto nisu dio imenskog prostora njihovo uključivanje daje kontejneru privilegiju izravnog pristupa
  - To se rješava uklanjanjem dozvola za pisanje po tim datotekama i zabranom ponovnog montiranja datotečnih sustava unutar kontejnera

Ograničenje pristupa resursima

- Napadi uskraćivanja usluge su jedni od najčešćih napada u sustavu s više korisnika – do njih dolazi kada jedan proces želi zauzeti sve računalne resurse
  - Rješenje se nalazi u mehanizmu za upravljanje resursima kojeg implementira operacijski sustav Linux, **Cgroups**
- ***Control groups (Cgroups)*** – kontroliraju količinu resursa koju pojedini kontejner može zauzeti
  - Docker može postaviti limit koliko pojedini kontejner može zauzeti resursa
  - Svaki kontejner ima svoju izoliranu kontrolnu grupu

Izolacija uređaja (1)

- U operacijskim sustavima temeljenim na Unixu, jezgra i aplikacije pristupaju sklopovlju preko posebnih datoteka (npr. /dev/mem za memoriju)
  - Ako kontejner ima pristup tim datotekama, to mu daje velike mogućnosti
- ***Device Whitelist Controller*** je kontroler koji djeluje unutar kontrolne grupe procesa i koji slijedi niz definiranih pravila o načinima pristupa uređajima u sustavu
  - Na taj način je onemogućeno procesima u kontejneru da pristupe bitnim uređajima u sustavu

Izolacija uređaja (2)

- ***nodev*** – opcija za montiranje datotečnih sustava u operacijskom sustavu Linux
  - Ona sprječava stvaranje i korištenje datoteka uređaja
  - Kada se datotečni sustav montira s tom opcijom, bilo koji uređaj u tom sustavu neće biti prepoznat kao stvarni uređaj, već samo kao obična datoteka
  - Docker koristi ovu opciju kada montira **datotečni sustav slike kontejnera**
  - Ovime je onemogućeno korištenje čvorova uređaja, koji se već nalaze unutar slike kontejnera, za komunikaciju s jezgrom domaćina

Izolacija međuprocesne komunikacije

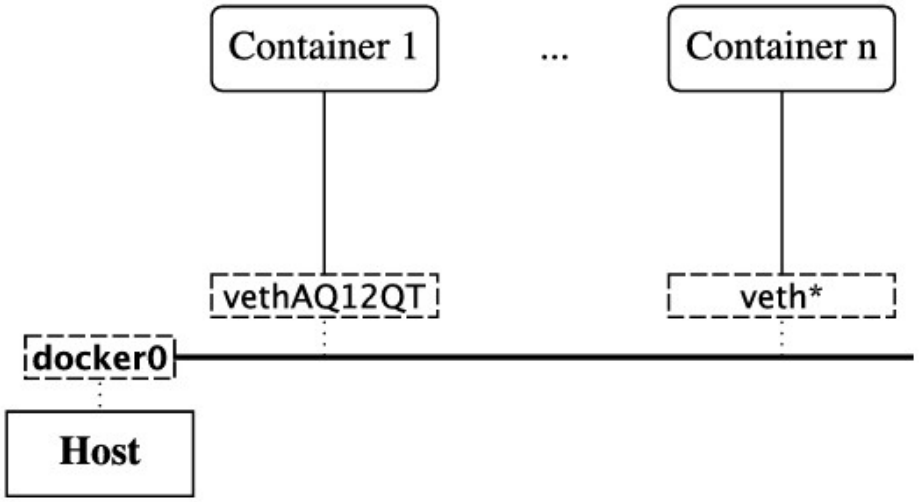
- IPC = set objekata preko kojih procesi komuniciraju
  - Semafori, dijeljena memorija, repovi za poruke…
- Bitno je ograničiti koji IPC objekti se koriste i tko ih koristi
- ***IPC namespace*** – izolirani prostor unutar kojeg se IPC objekti mogu koristiti
  - Nitko izvana ne može koristiti objekte unutar IPC imenskog prostora
  - Svaki kontejner dobiva svoj IPC imenski prostor, što znači da je komunikacija procesa iz različitih kontejnera preko IPC objekata onemogućena

Izolacija mrežnih resursa (1)

- Ako kontejneri nemaju dobru izolaciju mrežnih resursa, zlonamjerni kontejner može **manipulirati** mrežnim prometom drugih kontejnera
  - Izolacija je vrlo bitna za zaštitu od napada čovjeka u sredini i *ARP spoofing* napada
- Docker koristi mehanizam razdvojenog imenskog prostora za izolaciju mrežnih resursa jezgre Linux zvan *network namespace*
- ***Network namespace*** – ima svoj mrežni stog, svoju IP adresu i IP routing tablicu, svoju MAC adresu i postavke vatrozida itd.
  - Svaki kontejner ima svoj mrežni imenski prostor i komunicira s drugim kontejnerima jednako kao i s vanjskim uređajima

Izolacija mrežnih resursa (2)

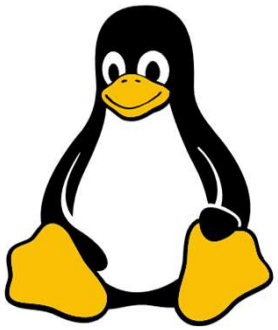
- Veza između kontejnera ostvaruje se **virtualnim Ethernet mostom** kojeg Docker stvara za svoje kontejnere
  - Taj most automatski prosljeđuje pakete između svojih mrežnih sučelja
- Svaki novi kontejner automatski se povezuje sa svojim sučeljem na taj most
- **Ne postoji filtriranje paketa!**
  - Svi paketi se prosljeđuju
  - **Opasnost od ARP spoofing i MAC flooding napada!**



Slika 2: podrazumijevani mrežni model Dockera

Sustavi za dodatnu zaštitu jezgre (1)

- Docker podržava dvije vrste sigurnosnih sustava radi bolje zaštite – Linux sposobnosti *(eng. capabilities)* i Linux sigurnosni model (LSM)
- **Linux sposobnosti** – podjela privilegija administratora na sposobnosti koje se mogu omogućiti ili onemogućiti
  - Posljedica: kontejnerima je većina sposobnosti inicijalno onemogućena
  - Ako kontejner treba obaviti zadatak za koji nema sposobnost, domaćin ili Docker *daemon* ga obave za njega



Sustavi za dodatnu zaštitu jezgre (2)

- Linux sigurnosni model – pruža okvir koji omogućuje uključivanje raznih sigurnosnih modela u sustav
- U službenu verziju Linux jezgre ugrađena su dva modela koja Docker podržava: AppArmor i SELinux
- **AppArmor** – stvara sigurnosne profile za aplikacije
  - Sigurnosni profil ograničava mogućnosti aplikacije
  - Može raditi u dva načina: *enforcement,* koji zahtjeva striktnu primjenu pravila, i *complain,* koji dopušta prekršaje, ali ih bilježi te se najčešće koristi za razvoj novih profila
- U Docker sustavima koji koriste AppArmour, kontejneri se automatski postavljaju u *enforcement* način rada

Sustavi za dodatnu zaštitu jezgre (3)

- **SELinux** – mehanizam koji pruža strožu kontrolu pristupa temeljenu na implementaciji politika
  - Dodaje *Mandatory Access Control* (MAC) uz standardni *Discretionary Access Contro*l (DAC)
  - Sve datoteke/procesi imaju **oznake** koje definiraju tko smije pristupiti čemu
- Docker koristi politike *Type Enforcement* i *MCS enforcement*
  - *Type Enforcement* – politika temeljena na tipovima, npr. određeni tipovi procesa smiju pristupati samo određenim tipovima datoteka
  - Procesi unutar kontejnera označeni su jednim tipom, a procesi domaćina drugim te je ovima iz kontejnera zabranjen pristup sustavskim resursima domaćina
  - *MCS Enforcement* – onemogućuje međusobni napad kontejnera na temelju sigurnosnih kategorija

Zaključak

- Unatoč lošijoj međusobnoj izolaciji kontejnera u odnosu na virtualne strojeve, kontejneri u Docker okruženju ne pružaju značajniji sigurnosni rizik
- Najveći sigurnosni rizik kontejnera u Dockeru su mrežni napadi, no to se može riješiti ručnim dodavanjem filtera prometa na Ethernet most
- Kontejneri zauzimaju manje prostora, brži su i lakše prenosivi i smatram da će njihova popularnost samo nastaviti svoj rast

Literatura

- **Bui, Thanh. “Analysis of Docker security” arXiv preprint arXiv:1501.02967 (2015.)**
- Luo, Yang. “Whispers between containers: High-capacity covert channel attacks in Docker” 2016 IEEE trustcom/bigdatase/ispa. IEEE, 2016
- Combe, Theo, Anthony Martin and Roberto di Pietro. “To Docker or not to Docker: A security perspective” IEEE Cloud computing 3.5 (2016): 54-62
- Martin, Anthony. “Docker ecosystem – vulnerability analysis” Computer communications 122 (2018): 30-43.

Slike

- Slika 1: Combe, Theo, Anthony Martin and Roberto di Pietro. “To Docker or not to Docker: A security perspective” Figure 1
- Ikone: https://www.iconfinder.com/
- Docker icona (slide 8): https://bs.m.wikipedia.org/wiki/Datoteka:Docker_%28container_engine%29_logo.svg
- Slika 2: Bui, Thanh. “Analysis of Docker security” Figure 5

Dodatna literatura

- Ako želite saznati više o mogućim napadima na Docker i kako ih spriječiti:
  - Yasrab, Robail. “Mitigating Docker security issues” arXiv preprint arXiv: 1804.05039 (2018).

Hvala!