VILNIAUS UNIVERSITETO

MATEMATIKOS IR INFORMATIKOS FAKULTETO

INFORMATIKOS KATEDRA

Kursinis darbas

**FreeBSD įkalintos aplinkos (jail) ir Linux OpenVZ palyginimas**

Atliko: 3 kurso, 2 grupės studentas

Gytis Karčiauskas

Darbo vadovas:

Kęstutis Mizara

Vilnius

2011

Turinys

[1. Sutartinių terminų sąrašas 3](#_Toc296355722)

[2. Anotacija 4](#_Toc296355723)

[3. Summary 5](#_Toc296355724)

[4. Įvadas 6](#_Toc296355725)

[5. Pagrindinė dalis 7](#_Toc296355726)

[5.1. Tradicinis \*NIX failų prieigos kontrolės modelis 7](#_Toc296355727)

[5.2. FreeBSD įkalintų aplinkų (jail) sprendimas 8](#_Toc296355728)

[5.2.1 Kalėjimo įgyvendinimas 9](#_Toc296355729)

[5.2.2. Apribojimas vienam IP adresui 10](#_Toc296355730)

[5.3. OpenVZ 11](#_Toc296355731)

[5.3.1. OpenVZ programos 11](#_Toc296355732)

[5.3.2. OS virtualizacija 12](#_Toc296355733)

[5.3.3. Tinklo virtualizacija 12](#_Toc296355734)

[5.3.4 Šablonai 12](#_Toc296355735)

[5.3.5 Resursų valdymas 13](#_Toc296355736)

[5.3.6. OpenVZ technologijos principai 13](#_Toc296355737)

[5.4. Palyginimas 15](#_Toc296355738)

[5.4.1. Bendrosios savybės 15](#_Toc296355739)

[5.4.2. Chroot 15](#_Toc296355740)

[5.4.3. Jails 15](#_Toc296355741)

[5.4.4. OpenVZ 16](#_Toc296355742)

[6. Išvados 17](#_Toc296355743)

[7. Literatūros sąrašas 18](#_Toc296355744)

# 1. Sutartinių terminų sąrašas

* \*NIX – terminas, apibrėžiantis Unix ir Linux operacines sistemas.
* FreeBSD - laisva daugiaplatformė Unix šeimos operacinė sistema. [[Wik11a](#_7._Literatūros_sąrašas)]
* Linux - laisvos operacinės sistemos branduolio (kernel) pavadinimas. Dažnai taip sutrumpintai vadinama ir bendrai visa Unix tipo operacinė sistema naudojanti Linux branduolį kartu su sisteminėmis programomis bei bibliotekomis iš GNU projekto. [[Wik11b](#_7._Literatūros_sąrašas)]
* Chroot – Bill Joy sukurta sistema, naudojanti failus ir operacijas su jais (pridėjimas, pašalinimas ir t.t.) viename medyje. Programa, veikianti šioje aplinkoje, negali pasiekti failų, neesančių medyje.
* Host‘as – virtualias aplinkas kontroliuojanti sistema. Ji turi priėjimą prie visų aparatinės įrangos resursų. Vienas iš pagrindinių skirtumų tarp host‘o ir virtualių aplinkų yra tas, kad super vartotojų (root) negali įtakoti procesų host‘o sistemoje.
* IP adresas - kompiuterio identifikatorius IP tinkluose. Tai tam tikrame tinkle unikalus skaičius, naudojamas vienareikšmei duomenų paketo siuntėjo ir gavėjo identifikacijai ir skiriamas žmogaus ar organizacijos, administruojančios duotąjį IP tinklą. [[Wik11c](#_7._Literatūros_sąrašas)]

# 2. Anotacija

Kursinio darbo tikslas – išanalizuoti ir palyginti UNIX šeimos FreeBSD operacinės sistemos įkalintos aplinkos (jail) ir Linux OpenVZ virtualizacijas. Tikslas pasiektas išanalizuojant kiekvienos pusės naudojamus principus, technologijas, privalumus ir trūkumus.

Darbe aprašomas tradicinis \*NIX sistemų failų prieigos kontrolės modelis, surandami jo trūkumai. Apsvarstomos galimos alternatyvos – FreeBSD jail ir Linux OpenVZ. Trumpai aprašomas alternatyvų elgesys izoliuotose aplinkose, darbas su failais, procesais, resursais ir vartotojais. Palyginant pateikiami sprendimų bendrosios savybės, privalumai ir trūkumai.

# 3. Summary

**The Jailed Environment of FreeBSD and Linux OpenVZ Comparison**

The aim of this work is to analyze and compare the FreeBSD operating system imprisoned environment (jail) and Linux OpenVZ virtualization. Aim has achieved by analyzing each side used principles, techniques, advantages and disadvantages.

This work describes the traditional \*NIX file system access control model, detects his disadvantages and considers alternatives. Possible alternatives to be considered – FreeBSD jails and Linux OpenVZ VPS (Virtual Private Servers). This work also describes a behavior of alternatives in isolated environments, work with files, processes, resources and users, compares the general properties of solutions, advantages and disadvantages.

# 4. Įvadas

Tradicinis \*NIX sistemų saugumo modelis yra paprastas, bet netobulas. Egzistuojanti failų prieigos kontrolės mechanizmas atlieka puikiai savo darbą, tačiau dažnai padidina tiek sistemos valdymo išlaidas, tiek įgyvendinimo sudėtingumą. Jis veikia aplinkoje, kurioje egzistuoja dviejų tipų vartotojai: tie, kurie turi administravimo teises, ir tie, kurie neturi.

Norint tai labiau suprasti, pateiksime realų pavyzdį: daug žiniatinklio paslaugų tiekėjų teikia klientams interneto svetainių talpinimo paslaugas aukšto našumo serveriuose. Tiekėjas turi rūpintis tiek savo failų ir veikiančių bei klientus aptarnaujančių servisų vientisumu ir konfidencialumu, tiek apsaugoti klientų failus, kad kiti negalėtų jų pasiekti (atsitiktinai ar tyčia).

Ši problema eina link skirstymo sprendimo (principas „*skaldyk ir valdyk*“), kuris kliento procesus ir failus izoliuotų nuo kitų klientų taip, kad apsaugotų nuo konfidencialios informacijos atskleidimo ir keitimo iš išorės. Egzistuojantys tokie mechanizmai kaip *chroot* suteikia kuklų skirstymą, taip pat žinoma, kad šie mechanizmai turi rimtų trūkumų tiek taikymo srityse, tiek veiksmingume.

Kalbant apie *chroot*, proceso matomumas yra apribojimas ties viena šaka, tačiau neapima kitų procesų ar tinklo, todėl įmanomas išoriškas stebėjimas ir įsikišimas. Norint pilnai išspręsti problemą, palyginsime UNIX šeimos FreeBSD operacinės sistemos jail mechanizmą su Linux OpenVZ.

# 5. Pagrindinė dalis

## 5.1. Tradicinis \*NIX failų prieigos kontrolės modelis

Tradicinis \*NIX failų prieigos kontrolės modelis priskiria unikalų skaičių *uid* kiekvienam sistemos naudotojui. Savo ruožtu kiekvienas naudotojo sukurtas procesas bus pažymėtas to naudotojo *uid*. Šis skaičius tarnauja dviems tikslams: pirma, jis nustato, kaip savo nuožiūra priėjimo kontrolės mechanizmas bus taikomas, antra, jie yra naudojami nustatyti ar yra suteiktų specialių privilegijų.

Šiuo atveju pirminis objektas yra failas. Skaičius *uid* (ir *gid*, nurodantis grupę) yra susietas į teisių aibę kiekvienam objektui. Toks modelis veikia kaip riboto priėjimo prie objekto reguliavimas.

Siekiant nustatyti, ar specialios privilegijos yra suteiktos procesui, vyksta paprastas patikrinimas „ar skaitinė *uid* reikšmė lygi 0“? Jei taip, šis procesas veikia „super naudotojo“ privilegijomis, kurios leidžia jam daryti tai, ko jis nori. Žmogaus patogumo tikslais, naudotojas, kuris turi *uid* = 0, vadinasi *root*. Šios privilegijos gali būti panaudojamos valdyti aparatinę įrangą, konfigūraciją ar tinklo operacijas.

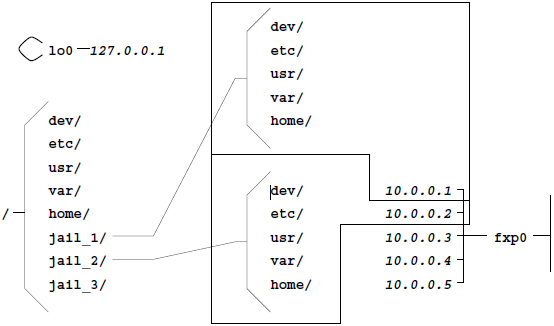
Tačiau šis modelis turi keletą aiškių trūkumų: toks naudotojas yra vienintelis sistemoje, tad gali būti atakuojamas iš programinės įrangos pusės siekiant įsilaužti. Sėkmės atveju, įsilaužėlis turės visas sistemos privilegijas. Be šio trūkumo, sistemą gali sugadinti nepatyręs sistemų administratorius. Visos šios savybės daro visagalį root naudotoją veiksmingu ir labai pavojingu įrankiu.

BSD šeimos operacinės sistemos įgyvendino „*secure-level*“ mechanizmą, kuris leidžia, administratoriui blokuoti tam tikras konfigūravimo ir valdymo funkcijas, kurios gali būti atliekamos „super naudotojo“ privilegijomis, kol nėra perkraunama ir užkraunama į *single-user* režimą. Nors iš tiesų tai suteikia tam tikrą kiekį apsaugos bandant sukompromituoti sistemą, tačiau jis neturi delegavimo tam tikrų *root* savybių galimybės.

## 5.2. FreeBSD įkalintų aplinkų (jail) sprendimas

Įkalintos aplinkos išsprendžia daugumą šių problemų. Užuot įvedus papildomą kontrolės mechanizmą, panašų į tradicinį, suskaldome FreeBSD aplinką (procesus, failų sistemą, tinklo resursus) į valdymo aplinką. Be to, išsaugome tradicinį \*NIX saugumo modelį, leidžiantį daugelių vartotojų ir *root* buvimą kiekvienoje įkalintoje aplinkoje, tuo pačiu apribojant root veiklos sritį tik savo kalėjime. Todėl sistemų administratorius FreeBSD sistemą gali suskaldyti į atskirus kalėjimus ir suteikti prieigą prie „super naudotojo“ kiekvienam iš jų neprarandant kontrolės.

Procesas, esantis kalėjime, vadinamas „in jail“. Kai FreeBSD sistema yra įkraunama po instaliacijos, nė vienas procesas nebus „in jail“. Kai procesas yra patalpinamas į kalėjimą, viskas, kas bus proceso sukurta, taip pat bus kalėjime. Procesas gali būti tik viename kalėjime, ir po patalpinimo jis negali kalėjimo palikti. Kalėjimai yra sukuriami, kai privilegijuotas procesas *call*‘ina į *jail* *syscall*‘ą su kalėjimo apibūdinimu *call*‘o argumente. Kiekvienas *call*‘as į *jail* sukuria naują kalėjimą. Vienintelis būdas naujam procesui patekti į kalėjimą yra prieigos į kalėjimą paveldėjimas iš kito proceso, jau esančio kalėjime. Procesai negali palikti kalėjimo, net jei jie buvo sukurti prieš kalėjimą, ar susikūrė jame.



Paveikslėlis 1 – Sistemos diagrama su dvejomis sukonfigūruotomis įkalintomis aplinkomis

Narystė kalėjime susijęs su tam tikrais apribojimais:

* priėjimas prie failo yra apribojamas *chroot* principu;
* gebėjimas naudotis tiklo resursais yra apribojamas konkrečiam IP adresui;
* gebėjimas manipuliuoti sistemos resursais ir atlikti privilegijuotas operacijas yra smarkiai apribojamas;
* gebėjimas bendrauti su kitais procesais yra apribojamas – galima bendrauti tik su tais procesais, esančiais kalėjime.

Įkalinta aplinka išnaudoja teigiamą *chroot* savybę riboti priėjimą prie failų sistemos įkalintiems procesams. Kai yra sukuriamas kalėjimas, jis yra susiejamas prie failų sistemos šakninės (*root*) direktorijos. Procesai negali manipuliuoti su failais, esančiais kalėjimo išorėje. Tradiciniai mechanizmai, skirti išeiti iš *chroot*, buvo užblokuoti. Kiekvienas standartinis kalėjimas turi standartinį FreeBSD katalogų išdėstymą, tačiau tai nėra privaloma pagal implementaciją.

Kiekvienas kalėjimas yra susietas su vienu IP adresu. Procesai, esantys kalėjime, negali pasinaudoti jokiu kitu IP adresu įeinantiems ir išeinantiems prisijungimams. Tai apima galimybę pasiūlyti kiekvienam kalėjimui skirtingą servisą.

### 5.2.1 Kalėjimo įgyvendinimas

Procesai, vykdomi su root privilegijomis, susidurs su rimtais apribojimais ko jie negalės padaryti būnant kalėjime:

* tiesiogiai modifikuoti branduolį (*kernel*) ar užkrauti branduolio modulius yra draudžiamas;
* modifikavimas bet kokios tinklo konfigūracijos yra draudžiamas;
* failų sistemų prijungimas ir atjungimas yra draudžiamas;
* įrenginių kūrimas yra draudžiamas;
* keisti su „secure-level“ susijusius failus yra draudžiama;
* priėjimas prie tinklo resursų, kurie nėra susieti su kalėjimu, yra draudžiama.

Kiti privilegijuoti veiksmai yra leistini tol, kol jie vykdomi įkalintoje aplinkoje:

* signalų perdavimas per įkalintą procesą yra leidžiamas;
* failo teisių keitimas yra leidžiamas tol, kol failo *flag*‘ai leidžia tai;
* failo ištrynimas yra leidžiamas tol, kol failo *flag*‘as leidžia tai;
* sieti rezervuotus TCP ir UDP portų numerius ant įkalinto IP adreso yra leidžiama (bandymai susieti TCP ir UDP portus naudojant IN\_ADDRANY bus perradresuojami į įkalintą IP adresą;
* funkcijos, kurios dirba su *uid/gid*, yra leidžiamos

Šie root apribojimai apriboja *root* procesus taip, kad leidžia daugumai aplikacijų dirbti netrukdomai bei užkerta kelią pasiekti išorę ir turėti įtakos kitų procesų ar visos sistemos konfigūracijai.

### 5.2.2. Apribojimas vienam IP adresui

Apribojus TCP ir UDP priėjimą tik vienam IP buvo beveik atliktas kode, kuris valdo „protokolo valdymo blokus“. Kai įkalintas procesas kreipiasi į *socket*‘ą, IP adresas, gautas iš proceso, nebus naudojamas, vietoj to bus naudojamas sukonfigūruotas IP, skirtas įkalintai aplinkai.

BSD pagrįstas TCP/IP turi specialų interfeisą, kuris turi „magišką“ IP adresą 127.0.0.1. Jis yra dažnai naudojamas procesų susisiekti su serveriais lokaliame kompiuteryje, todėl reikėjo pasirūpinti jo apdorojimu kalėjimuose. Norint aptarnauti šį adresą, reikėjo keisti prisijungimo elgesį – kai iš įkalinto proceso pamatomas šis adresas, jis pakeičiamas į kalėjimui skirtą IP adresą. [[Kwa00](#_7._Literatūros_sąrašas)]

## 5.3. OpenVZ

OpenVZ yra pilnas serverio automatizavimo ir virtualizavimo sprendimas, kuris gali sukurti daug izoliuotų Virtualių Privačių Serverių (VPS) viename fiziniame serveryje ir valdyti aparatinę įrangą maksimaliai efektyviai. Kiekvienas VPS dirba kaip atskiras serveris, turintis savo vartotojus bei root priėjimą, programas, failus, IP adresą, atmintį, procesus, sistemos bibliotekas, konfigūracinius failus. Taip pat kiekvienas VPS gali būti perkrautas nepriklausomai nuo kitų VPS.

Pagrindinės OpenVZ galimybės yra:

* dinaminis realaus laiko skirstymas – dalina fizinį serverį į VPS, kiekvienas turi pilną dedikuoto serverio funkcionalumą;
* resursų valdymas – susieja ir valdo VPS resursų parametrus ir perskirsto resursus realiu laiku;
* daugumos valdymas – valdo fizinius ir VPS serverius vieningai.

### 5.3.1. OpenVZ programos

OpenVZ pateikia išsamius sprendimus talpinimo paslaugų (*hosting*) tiekėjams, kurie leidžia jiems:

* turėti šimtus klientų su jiems sukurtais VPS viename fiziniame serveryje;
* garantuoti kiekvienam klientui paslaugų kokybę;
* perkelti klientus tarp serverių be jokio papildomo perkonfigūravimo.

Jei egzistuoja tokie dedikuoti serveriai, kur kiekviename vykdomas tam tikras servisas, galima panaudoti OpenVZ konsoliduojant visus tokius serverius į vieną serverį neprarandant informacijos ir nesumažinant našumo.

Virtualūs Privatūs Serveriai dirba panašiai kaip atskiras izoliuotas serveris:

* kiekvienas VPS turi savo procesus, vartotojus, failus ir suteikia pilnavertę root prieigą;
* kiekvienas VPS turi savo IP adresą, prievadus, filtravimo ir maršrutų taisykles;
* kiekvienas VPS gali turėti savo konfigūraciją sistemos arba programinės įrangos poreikiams. Įmanoma įrašyti arba modifikuoti programinės įrangos paketus VPS viduje nepriklausomai nuo kitų VPS ar host sistemos.

### 5.3.2. OS virtualizacija

Kiekvienas VPS yra nepriklausoma sistema. Ši nepriklausomybė pasiekiama per virtualizacijos sluoksnį *host‘o* OS branduolyje. Pažymėtina, kad tik dalis procesoriaus išteklių yra skiriama virtualizacijai (apie 1-2%). Pagrindinės OpenVZ virtualizacijos sluoksnio funkcijos yra šios:

* VPS atrodo kaip normali Linux sistema. Ji turi standartinius paleidimo scenarijus (scripts), programinė įranga gali būti vykdoma VPS viduje be konkrečių pakeitimų ar derinimų;
* vartotojas gali keisti bet kurį konfigūracijos failą ar įdiegti papildomą programinę įrangą;
* VPS yra pilnai izoliuoti vienas nuo kito;
* procesai, priklausantys VPS, yra planuojami įvykdyti visuose įmanomuose procesoriuose. Todėl VPS neprivalo turėti tik vieną procesorių ir gali išnaudoti visų procesorių galią.

### 5.3.3. Tinklo virtualizacija

OpenVZ tinklo virtualizacijos sluoksnis yra sukurtas izoliuoti VPS vieną nuo kito, taip pat ir nuo fizinio tinklo:

* kiekvienas VPS turi nuosavą IP adresą. Vieno VPS galimybė turėti kelis IP adresus yra leidžiama;
* kiekvieno VPS tinklo srautas yra izoliuotas nuo kitų VPS. Kitaip sakant, Virtualūs Privatūs Serveriai yra saugomi vienas nuo kito, kad tinklo stebėjimas (snooping) taptų neįmanomas;
* ugniasienės (firewalls) gali būti naudojamos VPS viduje (vartotojas gali sukurti taisyklės, kurios apribotų priėjimą prie tam tikrų paslaugų, panaudojant įrankį *iptables*;
* maršrutizavimo lentelių keitimas yra galimas, tarkim norint gauti daugiau naudos iš sudėtingesnių maršrutizavimo privalumų. Pavyzdžiui, nurodant skirtingus didžiausius perdavimo vienetus (MTU) skirtingoms vietovėms ir pan.

### 5.3.4 Šablonai

Operacinės sistemos šablonas iš esmės yra paketų aibė, kuri naudojama sukurti vieną ar daugiau VPS. OS šabloną sudaro sisteminės programos, bibliotekos ir scenarijai (scripts), reikalingi pakrauti ir vykdyti sistemą (VPS). Tokia programinė įranga kaip kompiliatoriai ar SQL serveriai paprastai nėra įtraukiama į OS šabloną.

### 5.3.5 Resursų valdymas

OpenVZ resursų valdymas kontroliuoja reikalingą resursų kiekį Virtualies Privatiems Serveriams. Yra kontroliuojami tokie resursai kaip procesoriaus galia, su atmintimi susiję parametrai. Resursų valdymas leidžia OpenVZ:

* efektyviai dalintis turimais aparatinės įrangos ištekliais tarp VPS;
* suteikti našumo ir resursų izoliaciją ir apsisaugoti nuo atsisakymo aptarnauti (*denial-of-service*) išpuolių;
* vienu metu paskirti ir kontroliuoti resursus konkrečiam skaičiui VPS.

OpenVZ sistemai resursų valdymas yra daug svarbesnis už atskirą kompiuterį, nes kompiuterio išteklių panaudojimas OpenVZ pagrįstoje sistemoje yra žymiai didesnis už tipišką sistemą.

### 5.3.6. OpenVZ technologijos principai

Paveikslėlis 2 – OpenVZ technologija

Darome prielaidą, kad turime tam tikrą skaičių fizinių serverių, sujungtų į tinklą. Iš tiesų norint kuo efektyviau išnaudoti OpenVZ, galime turėti tik vieną fizinį serverį. Kiekvienas fizinis serveris, pagrįstas OpenVZ, turės panašią architektūrą. Tokie serveriai OpenVZ terminologijoje yra vadinami aparatūros mazgais (*hardware nodes*), nes jie atstovauja aparatūros vienetą tinkle.

OpenVZ yra įrašomas taip, kad galėtume paleisti kompiuterį su OpenVZ palaikymu, arba be jo. Šis palaikymas yra pateikiamas kaip „OpenVZ“ įkrovos pakrovėjuje (boot-loader) ir parodomas kaip „OpenVZ Layer“ antrame paveikslėlyje.

Tačiau šiame žingnyje dar negalima sukurti Virtualaus Privataus Serverio. VPS yra funkciškai identiškas izoliuotam atskiram serveriui, turinčiam savo IP adresą, procesus, failus, vartotojus, konfigūracinius failus, programinę įrangą, sistemines bibliotekas ir t.t. VPS dalinasi tuo pačiu aparatūros mazgu ir tuo pačiu OS branduoliu. Tačiau jie yra izoliuoti vienas nuo kito. Virtualus Privatus Serveris yra lyg smėliadėžė (sandbox) procesams ir vartotojams.

Į skirtingus VPS galima įrašyti skirtingas Linux versijas. Tokiu atveju sakome, kad VPS yra pagrįstas tam tikru OS šablonu. OS šablonai yra paketai, siunčiami kartu su OpenVZ. Prieš sukuriant VPS, reikia įrašyti atitinkamą šabloną į OpenVZ. Tai parodoma kaip kaip „OpenVZ Templates“ antrame paveikslėlyje.

Įdiegus bent vieną OS šabloną, galima sukurti bet kokį skaičių VPS su standartiniais OpenVZ įrankiais, konfigūruoti tinklo ir/ar kitus nustatymus ir dirbti šiaip su VPS kaip pilnai funkcionaliais Linux serveriais. [[Sws05](#_7._Literatūros_sąrašas)]

## 5.4. Palyginimas

### 5.4.1. Bendrosios savybės

Tiek FreeBSD įkalintos aplinkos, tiek Linux OpenVZ Virtualūs Privatūs Serveriai turi šias bendras savybes:

* virtualizacija – kiekviena virtuali aplinka turi savo failus, procesus, vartotojus, root prieigą;
* saugumas – kiekviena virtuali aplinka neįtakoja kitų aplinkų darbui, viena nuo kitos nepriklausoma;
* lengva delegacija – ribotas priėjimas prie virtualių aplinkų leidžia administratoriams paskirstyti servisus;
* nevirtualizuoja aparatinės įrangos – vienas branduolys.

### 5.4.2. Chroot

Privalumai:

* vykdomas toj pačioj operacinėj sistemoj, tad virtualizacija nereikalinga;
* parašyta daug dokumentacijos.

Trūkumai:

* reikalauja visų sisteminių dvejetainių failų kopijų;
* jei vartotojas gali įgauti *root* teises, jis gali išeiti iš *chroot*’intos aplinkos;
* proceso matomumas yra apribojimas ties viena šaka, neapima kitų procesų ar tinklo, todėl įmanomas išoriškas stebėjimas ir įsikišimas;
* nėra nepriklausoma virtuali mašina su savo resursais. [[Ser10](#_7._Literatūros_sąrašas)]

### 5.4.3. Jails

Privalumai:

* reikalauja tik vienos sisteminių dvejetainių failų kopijos visiems kalėjimams;
* galima atnaujinti visus kalėjimus per vieną bazinę direktoriją;

Trūkumai:

* negali sukurti ar pakeisti tiklo įrenginių;
* sunkiai realizuojamas NFS serveris.

### 5.4.4. OpenVZ

Privalumai:

* reikalauja labai mažai resursų: kiekvienas VPS papildomai užima tik 8MB atminties ir kelis megabaitus disko vietus, nereikia transliacijų irašymo/nuskaitymo ar tinklo operacijoms.

Trūkumai:

* reikalauja branduolio pataisymo (patch);
* nėra pilnai palaikoma su naujausiomis Linux distribucijomis;
* veikia tik su Linux distribucijomis. [[Sto08](#_7._Literatūros_sąrašas)]

# 6. Išvados

Šiame darbe buvo analizuojamos dvi virtualizavimo technologijos – FreeBSD jails ir Linux OpenVZ. Išaiškėjo, kad nė viena technologija neturi akivaizdaus pranašumo, visos turi savų pliusų ir minusų. Funkcionalumu ir našumu išsiskyrė OpenVZ, savo reikmėms naudojanti šablonus – tai leidžia per kelias sekundes sukurti visiškai naują virtualią mašiną su švaria OS, tačiau ji keičia *host*‘o branduolį, o OS pasirinkimas apsiriboja ties Linux šeima. Paprastumu išsiskyrė FreeBSD jails virtualizacijos būdas, kurį lengva valdyti ir atnaujinti, bet turi problemų su NFS serveriais bei tinklo įrenginių valdymu. Visi virtualizavimo sprendimai geri, tačiau tinka tik konkretiems poreikiams ir atvejams.

# 7. Literatūros sąrašas

[Kwa00] Poul-Henning Kamp and Robert N. M. Watson „Jails: Confining the omnipotent root.“ (<http://phk.freebsd.dk/pubs/sane2000-jail.pdf>)

[Ser10] ServerFault: [Isolating websites on a shared host using container solutions](http://serverfault.com/questions/164337/isolating-websites-on-a-shared-host-using-container-solutions) (<http://serverfault.com/questions/164337/isolating-websites-on-a-shared-host-using-container-solutions>)

[Sto08] Introduction to OpenVZ based virtualisation

[<https://stoned-it.com/blog/?p=33>]

[Sws05] OpenVZ User‘s Guide, SWsoft Inc., 2005, Version 2.7.0-8 (<http://download.openvz.org/doc/OpenVZ-Users-Guide.pdf>)

[Wik11a] Wikipedia: FreeBSD (<http://lt.wikipedia.org/wiki/FreeBSD>)

[Wik11b] Wikipedia: Linux (<http://lt.wikipedia.org/wiki/Linux>)

[Wik11c] Wikipedia: IP adresas (<http://lt.wikipedia.org/wiki/IP_adresas>)