Mendelova univerzita v Brně
Provozně ekonomická fakulta

Návrh loadbalancingu a firewallové politiky pro aplikační servery ve veřejném cloudu

Bakalářská práce

Vedoucí práce:
Ing. Jiří Balej

Jakub Zostal

Brno 2017

Chtěl bych poděkovat mému vedoucímu práce Ing. Jiřímu Balejovi, za cenné rady, odbornou pomoc a konzultace, které mi poskytl při vypracování mé bakalářské práce. Velké poděkování dále patří Ing. Radimovi Klabalovi, který mi nejenže poskytl toto téma bakalářské práce, ale taktéž byl velmi ochotný konzultovat danou problematiku při jakékoliv nejasnosti. V neposlední řadě bych chtěl poděkovat své rodině a přátelům za jejich podporu v průběhu celého studia.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Návrh loadbalancingu a firewallové politiky pro aplikační servery ve veřejném cloudu**
vypracoval/a samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom/a, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 22. května 2017 _____

Abstract

Zostal, J. Design of a loadbalancing and a firewall policy specified for application servers in a public cloud system. Brno: Mendel University in Brno, 2016.
The goal of the bachelor thesis is to propose a loadbalancing and a firewall policy for application servers in a public cloud. In this thesis, various available options for loadbalancing and operation filtering are presented and introduced. Furthermore, there is analyzed the current solution of a network infrastructure and a proposal of its modification with an assistance of the loadbalancing and firewall policy. Accordingly, the adjusted network infrastructure is tested in a laboratory environment.

Keywords

load balancing, firewall, filtering network traffic, network security

Abstrakt

Zostal, J. Návrh loadbalancingu a firewallové politiky pro aplikační servery ve veřejném cloudu. Brno: Mendel University, 2016.
Tato bakalářská práce se zabývá návrhem loadbalancingu a firewallové politiky pro aplikační servery ve veřejném cloudu. Práce zahrnuje seznámení s dostupnými možnostmi jak pro loadbalancing, tak pro filtrování provozu. Další částí práce je analýza stávajícího řešení síťové infrastruktury a návrh její úpravy za pomoci loadbalancingu a firewallové politiky. Upravená síťová infrastruktura je následně otestována v laboratorním prostředí.

Klíčová slova.

vyvažování zátěže, firewall, filtrování síťového provozu, zabezpečení sítě

Obsah

1 Úvod

V době stále se rozšiřujícího internetu můžeme narazit na nebezpečí téměř kdekoliv. Počítačové sítě jsou ohrožovány čím dál více propracovanějšími útoky. Tyto útoky nejsou však jediným ohrožením. Jako ohrožení musíme zvažovat také přírodní jevy a technická selhání, kterým se bohužel někdy vyhnout nedokážeme. Za ohrožení můžeme považovat také sabotované útoky uvnitř naší sítě. Většině zmíněných ohrožení se však můžeme vyvarovat, pokud je správně navrhnuta bezpečnostní politika a celková infrastruktura sítě.

Představme si společnost, která poskytuje službu, jež je využívána miliony uživateli denně. Aby společnost prosperovala, je nutné zajistit dostupnost služby za všech okolností, protože jakýkoli sebemenší výpadek by mohl způsobit obrovské ztráty jak společnosti, tak uživatelům v závislosti na charakteru služby. Nejlepším způsobem, jak se chránit před tímto nebezpečím je zavedení několika bezpečnostních vrstev. Mezi tyto vrstvy můžeme zařadit například firewall, monitoring, load balancer, autentizaci, autorizaci, VPN, antivir, nebo antispam. Je nutné oddělovat zejména privátní sítě od veřejného internetu, protože právě tam je nebezpečí největší. Jak to ale udělat, když k naší aplikaci přistupují uživatelé právě prostřednictvím veřejného internetu?

Firewall je řešením, které dokáže oddělit sítě s různou úrovní integrity a drží kontrolu nad tokem dat mezi těmito sítěmi. S vývojem technologií a všeho kolem nich je také kladen čím dál větší důraz na zvýšení výkonu aplikací a na jejich přizpůsobivost. V tomto ohledu se zaměřujeme zejména na základní požadavky, které se týkají dostupnosti a škálovatelnosti aplikace.

V dřívějších letech se za pojmem load balancer skrývalo zařízení, které sloužilo k vyvažování zátěže a odlehčení provozu. Postupem času ale tato zařízení začala nabývat spousty dalších funkcionalit, které jsou pro dnešní podniky, tedy pro jejich síťovou infrastrukturu, zcela nezbytné.

S rozvojem internetu, zejména tedy s příchodem bezdrátového internetu, sociálních sítí, cloudu atd. se začala zvyšovat potřeba, řešit problém, týkající se dostupnosti aplikace.

Poslední generace load balancing technologií se ukázaly být nezastoupitelným přínosem, a to zejména pro společnosti, které hostují široce využívané webové aplikace. Load balancing poskytuje příležitosti k provedení různých algoritmů pro rozdělení zátěže spojené se zpracováním mezi back-end servery.

2 Cíl práce

Cílem této práce je navrhnout řešení vyvažování zátěže a ochrany aplikačních serverů. Nejdříve bude nutné seznámit se s již existující sítí společnosti. Zároveň bude nutné definovat požadavky k úpravě infrastruktury. Společnost poskytuje webovou aplikaci, tudíž hlavním požadavkem bude, aby bylo její poskytování zabezpečeno jak už před výpadkem služby ze strany technických problémů, tak před útokem zvenčí. Na základě konzultace s vedením firmy je nutné navrhnout v první fázi strukturu, která bude vyhovovat požadavkům rozkladu zátěže a zabezpečení. K tomuto účelu budou do sítě zavedeny nové prvky v podobě load balancerů a bude definována nová bezpečností politika na firewallech.

Dále bude provedeno prozkoumání možných řešení rozdělení zátěže a následně, po konzultaci se společností, výběr nejlepšího řešení. Dále bude vytvořen návrh ochrany proti přetížení aplikačních serverů, který bude zároveň otestován.

Po konzultaci s administrátorem sítě bude navrhnuta bezpečnostní politika pro filtrování provozu, která bude následně taktéž otestována.

V konečné fázi bude sestavena šablona firewallu, kterou bude možné využít při zvyšování počtu serverů.

Tyto kroky by měly společnosti umožnit poskytování aplikace široké veřejnosti bez obav, že by službě hrozil výpadek, nebo že by mohla být přetížena.

3 Rešerše závěrečných prací

Jelikož téma rozložení zátěže a filtrování provozu není žádnou novou, neprozkoumanou oblastí v oboru počítačových sítí, můžeme se setkat s řadou publikací, které se těmto oblastem věnují a tuto problematiku řeší. Využití těchto technik lze však rozprostřít mezi nespočet různých oblastí jejich využití. Je tedy potřeba vytyčit klíčová slova, která jsou pro naši práci důležitá, a podle těchto odborných termínů prozkoumat dostupné informační zdroje. Díky shromáždění a přečtení stávajících prací, můžeme následně zhodnotit, zda už byl tento problém řešen a do jaké míry. V závislosti na tom získáme přehled, jak v dalším řešení postupovat.

Pro vyhledávání byly využity veřejně přístupné databáze závěrečných prací, kde byly vyhledávány bakalářské a diplomové závěrečné práce v rozmezí let 2012-2017, a to podle klíčových slov load balancing, vyvažování zátěže, firewall, filtrování provozu a zabezpečení sítě.

Využili jsme databází jednotlivých univerzit, které jsou dostupné online:

- Mendelova Univerzita v Brně:
  - https://is.mendelu.cz/zp
- Vysoké učení technické v Brně:
  - https://www.vutbr.cz/studium/zaverecne-prace
- Masarykova univerzita V Brně:
  - https://is.muni.cz/thesis

Dále jsme vyhledávali v centrální databázi, v níž jsme mohli dohledat závěrečné práce z více univerzit (databáze obsahuje i některé výše zmíněné): https://theses.cz.

3.1 Jednotlivé práce

Nidl (2015) se ve své diplomové práci zabývá metodikou optimálního využití load balancingu v prostředí datového centra. V teoretické části Nidl (2015) popisuje koncepty vyvažování zátěže, síťovou architekturu vyvažování zátěže, technologie vyvažování zátěže a metody vyvažování zátěže. Dále v teoretické části popisuje aktuální stav vyvažování zátěže v datovém centru, kde popisuje také vyvažování zátěže služeb v datovém centru. Tento popis zahrnuje vyvažování zátěže na webových serverech a databázových serverech. V praktické části Nidl (2015) optimalizuje řešení vyvažování zátěže v datovém centru. Práci lze tedy využít k nastudování load balancingu, který je zde velmi dobře popsán, a taktéž k postupu při řešení problému vyvažování zátěže na webových serverech.

Trnka (2017) ve své diplomové práci řeší clustering a load balancing serveru pro zpracování řeči. V teoretické části popisuje Trnka (2017) load balancing, kde se zaměřuje zejména na to, jak probíhá vyvažování zátěže na vrstvách L4 a L7 modelu ISO/OSI a na jednotlivé metody pro vyvažování zátěže. Dále pak popisuje síťové

programování teorie zpracování řeči. V praktické části pak implementuje vlastní řešení pro vyvažování, jelikož žádné dostupné řešení nepodporuje implementaci vlastní vyvažovací metody. Tuto práci lze opět využít k nastudování vyvažování zátěže a algoritmů pro vyvažování.

V závěrečné práci *Webhostingový cluster postavený na platformě ARM* (Navrátil, 2015) řeší autor návrh clusterového systému poskytující služby pro účely webhostingu za použití energeticky nenáročných zařízení postavených na architektuře ARM. V teoretické části autor seznamuje čtenáře zejména s problematikou clusteru. V praktické části pak navrhuje nastavení serverových aplikací a popisuje situace, kdy ve webhostingovém clusteru selže jedno ze zařízení. Z této práce lze čerpat řešení nastavení virtuálního webového serveru jako vyvažovače zátěže.

Diplomová práce *Analýza a optimalizace softwarových firewall na operačních systémech Linux* (Bartoš, 2016) analyzuje řešení pro firewall. V praktické části optimalizuje nasazení na GNU/Linux ve firemním prostředí. V této práci se lze inspirovat návrhem optimalizace nasazení firewallu.

Závěrečná práce Petrák (2013) řeší softwarový firewall pro filtrování na síťové a linkové vrstvě. Tato diplomová práce rozebírá v teoretické části celkovou problematiku filtrování síťového provozu a porovnává řešení pro filtrování na platformě Linux. V praktické části Petrák (2013) navrhuje a vytváří vlastní aplikaci pro filtrování provozu, kterou staví na základě frameworku Netfilter. Práce nám umožňuje důkladné seznámení s filtrováním provozu.

Vyhnátek (2013) řeší ve své bakalářské práci návrh aplikace pro platformu Android, která slouží jako generátor základních filtrovacích pravidel pro konfiguraci firewallů na síťových zařízeních. V teoretické části Vyhnátek (2013) analyzuje technologie pro filtrování paketů a v praktické části se věnuje návrhu a implementaci aplikace.

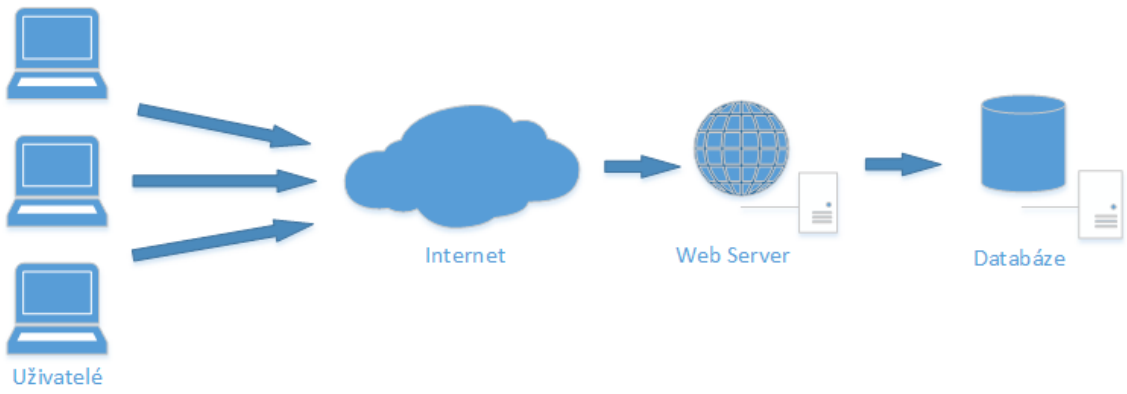
3.2 Zhodnocení

Jak jsme již zmínili na začátku této kapitoly, problematika vyvažování zátěže a filtrování provozu není žádnou novinkou. Lze tedy najít řadu publikací, které tuto problematiku řeší. Žádná z prací ovšem neřeší obě dvě problematiky současně a tím se bude naše práce odlišovat.

Podobnou prací je určitě práce *Metodika optimálního využití load balancingu v prostředí datového centra*, jelikož Nidl (2015) vychází z jakéhosi původního stavu, který se snaží optimalizovat z hlediska rozložení zátěže. Tato práce tedy bude velkou inspirací a možná i přímým využitím řešení problematiky vyvažování zátěže.

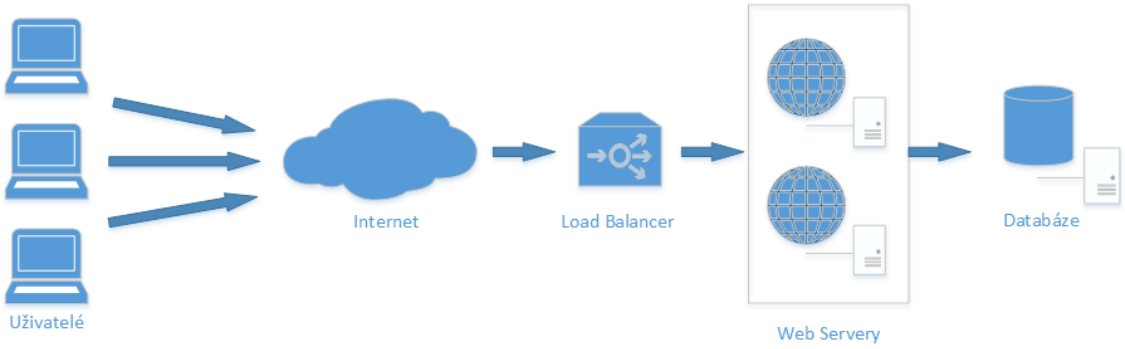
4 Load balancing

Load balancing, server load balancing neboli vyvažování zátěže je jednou ze základních síťových technik. Tato technika je zodpovědná za distribuci požadavků klientů napříč všemi servery, které jsou schopny splnit tyto požadavky způsobem, jenž maximalizuje využití poskytované aplikace a rychlost při zpracování požadavků. Zajišťuje, že žádný ze serverů není přetížen, což by mohlo vést ke snížení výkonu. (Cintrix, 2016) Pokud se jeden ze serverů stane nedostupným, zátěž je vyrovnána přesměrováním požadavků na zbývající servery. Obdobně je tomu i při přidání serveru do skupiny, kdy se zatížení začne vyrovnávat odesíláním požadavků právě k novému serveru. Jinými slovy, vyvažovače zátěže mají velmi efektivní význam pro rozšíření aplikačního prostředí, kdy je současně zajištěna aplikační dostupnost. Klasické prostředí, bez load balanceru můžeme vidět na obrázku *Obr. 1*.



Obr. 1 Prostředí bez load balanceru

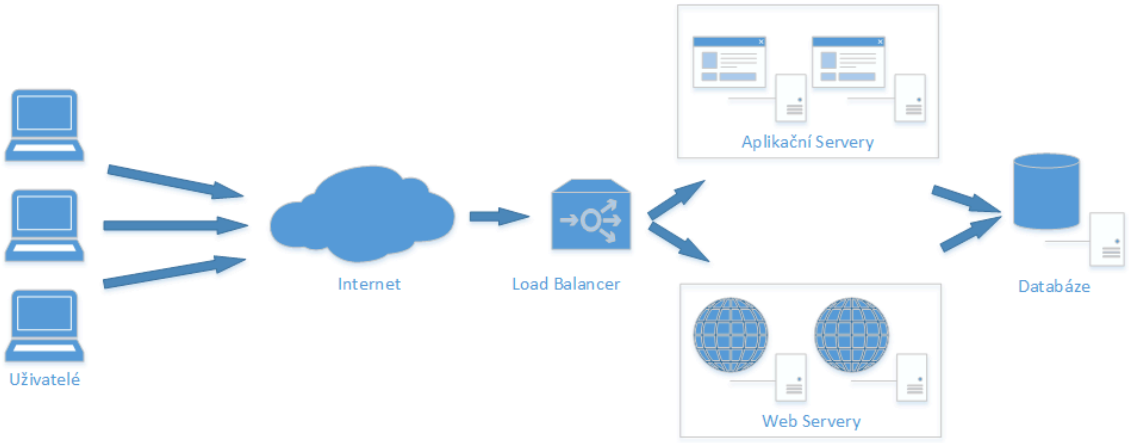
Vyvažování zátěže je realizováno na vrstvě L4 a L7 modelu ISO/OSI. Rozhodování o vyvažování na vrstvě L4 (transportní vrstva) probíhá na základě IP adres a TCP nebo UDP portů u každého z paketů. Nekontroluje tedy obsah paketů. (Langmaid, 2016) Tento způsob vyvažování naznačuje obrázek *Obr. 2*, kde load balancer pracuje se dvěma webovými servery.



Obr. 2 L4 load balancing

Rozhodování o vyvažování na vrstvě L7 (aplikační vrstva) probíhá na základě skutečného obsahu každé zprávy. Rozdělování provozu je mnohem sofistikovanější, než je tomu u vyvažování na L4. Tento způsob vyvažování umožňuje provozovat více aplikačních serverů v rámci stejné domény a portu. Vyvažování na L7 nabízí spoustu výhod v tom, jakým způsobem jsou data směrována a řízena. (Langmaid, 2016)

Na obrázku *Obr. 3* vidíme, že vyvažování zátěže probíhá mezi dvěma různými skupinami serverů.



Obr. 3 L7 load balancing

4.1 Všeobecné výhody vyvažování zátěže

V dnešní době může i krátký výpadek poskytované služby způsobit společnosti ohromné finanční ztráty. Vyvažování zátěže je postaveno tak, aby byla snížena pracovní zátěž na jednotlivých serverech, zvýšena dostupnost poskytovaných služeb, snížena nutnost neustálých upgradů hardwaru a v neposlední řadě zvýšeno zabezpečení poskytované služby. Tyto výhody můžeme shrnout do těchto bodů:

- Dostupnost:
  - Zvýšený výkon aplikace díky rychlejší odezvě.
  - Nepřetržitá kontrola dostupnosti koncových serverů.
  - Automatizovaný proces kontroly bez nutnosti zásahu administrátora.
  - Failover – zamezení výpadku aplikace a dostupnosti řešení.

- Škálovatelnost:
  - Zapojení několika méně výkonných koncových serverů.
  - Snížení nákladů na nákup koncových serverů.
  - Možnost přidání dalšího dodatečného serveru.

- Flexibilita:
  - Prostor pro možnost výměny, úplného vypnutí a pro údržbu koncového serveru bez rizika výpadku jeho poskytované služby.

(Bourke, 2001)

4.2 Metody vyvažování zátěže

Podle (Nidl, 2016) dělíme vyvažování zátěže na metody, které představují způsob distribuce příchozích spojení na koncové servery, jenž se nacházejí právě za síťovým vyvažovačem. Jedná se o různé algoritmy, které na základě rozlišných metrik určují, jak bude spojení distribuováno. Metody vyvažování zátěže řadíme do dvou základních skupin. Jedná se o statické metody vyvažování zátěže a o dynamické metody vyvažování zátěže. Mezi statické metody můžeme zařadit metody Round Robin a Ratio. Mezi dynamické metody pak metody Least Connections, Weighted Least Connections, Observed a Predictive. U jednotlivých dodavatelů se můžeme setkat s dalšími vyvažovacími metodami, které jsou však většinou založeny na již existujících metodách.

4.2.1 Round Robin

Round Robin je výchozí metoda vyvažování zátěže a jedná se o statickou metodu. Výhodou této metody je její jednoduchost, za niž je považována nenáročná implementace. Nevýhodou této metody je však způsob distribuce síťové zátěže, který je cyklický. Metoda přiřazuje každou novou žádost o připojení na další následující koncový server. Zátěž je tedy rozložena rovnoměrně na všech koncových serverech, což dokazuje, že metoda nerozlišuje koncové servery podle jejich výkonnosti. (Nidl, 2016) (University of Tennessee, Knoxville, 2015)

4.2.2 Ratio

Ratio je pokročilejší modifikace metody Round Robin, je tedy taktéž statickou metodou. Všechny žádosti o připojení na koncové servery jsou opět distribuovány cyklicky, tentokrát však v souladu s hodnotou, kterou je nutné předem definovat. Každé příchozí připojení je distribuováno na koncové servery v určitém poměru. Pokud tedy máme některé koncové servery výkonnější než ostatní, lze této metody využít a největší část žádostí o připojení směřovat právě na výkonnější servery, což je výhoda oproti Roud Robin. Ve chvíli, kdy je dosaženo definované hodnoty pro distribuci spojení, jsou následující spojení distribuována na další dostupný koncový server, kam je síťová zátěž distribuována opět v souladu s definovanou hodnotou. Metoda se dělí podle zaměření na Ratio Member a Ratio Node. Ratio Member umožňuje distribuci na koncové servery podle skupin, do kterých jsou koncové servery přiřazeny. Ratio Node umožňuje distribuci zátěže na jednotlivé koncové servery. (Nidl, 2016) (University of Tennessee, Knoxville, 2015)

4.2.3 Least Connections

Least Connection je dynamická metoda založená na aktivní kontrole vytížení jednotlivých koncových serverů. Zjišťuje aktuální počet udržovaných spojení na koncových serverech. Metoda předá nové připojení koncovému serveru, který má nejmenší počet současných připojení. Tato metoda funguje nejlépe v prostředí, ve kterém mají jednotlivé servery podobný výkon. Nevýhodou této metody je způsob ověření počtu aktivních spojení, jelikož ověřuje všechna udržovaná spojení, ale již nezjišťuje, která spojení jsou aktivně využita. Tato nevýhoda se může projevit u síťových služeb, které udržují spojení typu klient-server, aniž by spojení bylo využíváno ke komunikaci. (Nidl, 2016) (University of Tennessee, Knoxville, 2015)

4.2.4 Weighted Least Connections

Metoda Weighted Least Connections je dynamickou metodou a modifikací metody Least Connection, kde kromě zjišťování aktuálního počtu udržovaných připojení na jednotlivých koncových serverech přidává také kontrolu maximálního množství připojení, jenž jsou jednotlivé koncové servery schopny udržovat. Při výběru této metody je tedy nutné nastavit váhu, tedy maximální počet připojení, pro každý koncový server. Modifikovaná metoda je efektivnější, jelikož distribuuje příchozí spojení na koncové servery o různých výkonech. Tento algoritmus využívá sice více výpočetního času než metoda Least Connections, nicméně delší výpočetní čas se odráží v efektivnější distribuci provozu na server, který je pak nejvíce schopen zvládnout požadavky. (Nidl, 2016) (University of Tennessee, Knoxville, 2015)

4.2.5 Observed

Metoda Observed je dynamická metoda založená na řazení koncových serverů podle počtu spojů v čase. Na základě všech odeslaných spojení je vytvořena hodnota proměnné (ratio), která je následně dosazována do vzorce pro výpočet vyvažování zátěže. Algoritmus tak pracuje se zátěží na základě aktivních spojení, maximálního počtu udrži-

telných spojení. Současně také pracuje s hodnotou reprezentující počty spojení v minulosti. Tento způsob vyrovnání zátěže pracuje dobře v každém prostředí, ale může být užitečný zejména v prostředí, kde se výkon jednotlivých koncových serverů výrazně liší. (Nidl, 2016) (University of Tennessee, Knoxville, 2015)

4.2.6 Predictive

Predictive je dynamická metoda, která má schopnost vyhodnocování počtu udržovaných spojení a počtu odeslaných spojení, čímž se tato metoda stává efektivní při distribuci zátěže. Pracuje na podobném principu jako metoda Weighted Least Connections. Nová spojení na koncové servery distribuuje na základě počtu aktivních spojení a současně na základě maximálního počtu udržitelných spojení. Metoda přidává k algoritmu pro vyvažování hodnotu, která pracuje s hodnocením počtu odeslaných spojení v minulosti. Tento postup umožňuje vyvažovat zátěž na základě předpovědi, která se opírá o data z minulosti. Je tak možné vytvořit přehled, v němž jsou sledovány trendy v počtu spojení na koncové servery, a na základě toho lze lépe plánovat vyvažování zátěže. (Nidl, 2016) (University of Tennessee, Knoxville, 2015)

4.3 Hardwarový load balancing

Hardwarový load balancer je síťové zařízení, které distribuuje provoz aplikace mezi více serverů. Load balancer je umístěný na front-end části webových serverů, tudíž mezi servery a klientem, kde zachycuje aplikační požadavky a řídí dopravu mezi zdravými servery s cílem zajistit vždy dostupné aplikační služby. Taktéž snižuje zatížení na back-end části serverů mezi servery a databázemi pro zlepšení výkonnosti a uživatelské zkušenosti. Tyto load balancery jsou realizovány na L4 a L7 v ISO/OSI modelu. (Load Balancer FAQ, 2015)

Výhody hardwarového load balancingu jsou:

- vysoká dostupnost a odolnost pro aplikace,
- nárůst kapacity systému,
- škálovatelnost operací pro přizpůsobení více uživatelům a novým aplikacím,
- zvýšený výkon aplikací,
- zvýšená bezpečnost.

(Load Balancer FAQ, 2015)

4.4 Aplikační load balancing

Po mnoho let byl load balancing spojen pouze s dedikovaným hardwarem. Vzhledem k tomu, že se výkon standartního počítačového hardwaru vyvíjel podle Mooreova zákona, začaly vznikat i jiné možnosti, jak tento problém řešit a vynechat hardwarové load balancery. S rostoucí výkonností kapacit standartního serverového hardwaru a velkých

pokroků v samotné architektuře lze využít právě tohoto trendu. Softwarové řešení pro vyvažování zátěže poskytuje maximální flexibilitu a škálovatelnost pro uživatele a umožňuje nasazení na téměř každé serverovém hardwaru. V mnoha případech to také může být šetrnější k životnímu prostředí a to proto, že využijeme starší hardware serverů a nemusíme opět kupovat nový kvůli náročnějším aplikacím. (Software Load Balancer, 2015)

Roli aplikačního load balanceru vykonává zpravidla webový server, který obsahuje modul podporující load balancing.

Výhody aplikačního load balancingu:

- zvýšení škálovatelnosti a flexibility,
- eliminace požadavků na dedikovaný hardware,
- snadnost nasazení,
- šetrnost k životnímu prostředí,
- stejné funkce jako hardarové load balancery,
- virtualizování do cloud prostředí, tedy bez potřeby HW řešení.

(Software Load Balancer, 2015)

5 Porovnání komerčních a open-source řešení rozkladu zátěže

Na trhu se můžeme setkat s mnoha řešeními pro vyvažování zátěže. Jak ale vybrat to pravé?

V době, kdy se stal internet komerčním, se na trhu objevily společnosti poskytující tuto službu. Ty se odlišovaly v mnoha aspektech, ať už se jednalo o cenovou dostupnost služby, nebo funkcionální odlišnost. V dnešní době se však jednotliví dodavatelé již tolik neliší. Nidl (2016) ve své závěrečné práci zmiňuje, že se dodávaná řešení liší pouze v detailnějším pohledu na technickou specifikaci, cenovou politiku a garantovanou podporu.

Mezi nejznámější komerční řešení patří Cintrix NetScaler, Barracuda Load Balancer ADC a F5 BIG-IP. Na trhu se však můžeme setkat také s open source řešeními jako jsou například Nginx a HAproxy.

Nyní se seznámíme s komerčními řešeními, které následně porovnáme podle vlastností, které jsou pro naši společnost prioritní a taktéž z hlediska cenové politiky. Klíčové vlastnosti pro porovnání jsou:

- rozklad zátěže http provozu,
- počet L7 žádostí za sekundu,
- počet SSL transakcí za sekundu.

5.1 Komerční řešení

5.1.1 Citrix NetScaler

Citrix NetScaler je jak hardwarovým, tak softwarovým řešením pro vyrovnání zátěže. Jedná se o aplikační přepínač, který inteligentně distribuuje, optimalizuje a zabezpečuje vrstvy L4 a L7. Můžeme se setkat s řadami NetScaler MPX, NetScaler SDX a NetScaler VPX.

NetScaler MPX je dedikované hardwarové zařízení, které se hodí pro správu webových aplikací, jenž realizují různě velké datové toky (výkon 500 Mbps - 160 Gbps). Tato řada má díky vestavěnému firewallu mimořádně vysoký výkon zabezpečení webových aplikací. Je vhodná pro malé podniky.

NetScaler SDX je opět hardwarovým řešením. Tato řada je určená pro optimalizaci datového toku v datových centrech.

NetScaler VPX je virtualizované softwarové zařízení, které je vhodné pro veřejné i privátní cloudové infrastruktury. Poskytuje aplikační vyvažování zátěže, bezpečný vzdálený přístup, akceleraci zpracování požadavků a zabezpečení poskytované služby. Vhodné pro telekomunikační společnosti a malé i velké podniky.

NetScaler rozlišuje tři druhy licencí: Standard Edition, Enterprise Edition a Platinum Edition. Tyto licence přicházejí s určitými softwarovými nástroji. Základní licence přináší spolehlivou dostupnost aplikací, komplexní vyvažování zátěže L4 –L7, optima-

lizaci výkonu a bezpečný vzdálený přístup. Další dvě edice přicházejí s vylepšeními jako je například řízení dopravy, podpora clusterů, optimalizace provozu aplikací v cloudu, rozšířené funkce pro akceleraci aplikací atd.

Citrix využívá k vyvažování zátěže metody Least Connection a Round Robin. Dále pak využívá výše neuvedené vyvažovací metody Least Response Time, Least Bandwidth, Least Packets nebo hashovací metody. (Cintrix, 2017).

5.1.2 Barracuda

Barracuda Networks, Inc. je dalším z dodavatelů jak dedikovaného hardwaru pro vyvažování zátěže, tak virtualizovaného softwaru pro vyvažování zátěže. V hardwarové podobě společnost nabízí zařízení ADC 240, ADC 340, ADC 440, ADC 540, ADC 640/641/624 a nejvýkonnější ADC 840/841/842.

V softwarové podobě pak nabízí verze ADC 340 Vx, ADC 440 Vx, ADC 540 Vx a nejvýkonnější ADC 640 Vx.

Tato zařízení jsou kombinována s IPS (Intrusion Prevention System), čímž navíc zajišťují ochranu před útoky na síť. Mezi hlavní přednosti zařízení patří poskytnutí flexibilní možnosti volby algoritmu rozdělení zátěže, podpora zálohování zařízení, neomezená licence v rámci množství serverů a kompletní kryptografické zajištění provozu SSL. Barracuda využívá 3 typy metod pro vyvažování zátěže: Round robin, Weighted round robin a Least connection. (Barracuda, 2017)

5.1.3 F5 BIG-IP

Společnost F5 Networks, Inc. je jednou z hlavních společností dodávající jak dedikovaný hardware, tak virtualizovaný software pro vyvažování zátěže. Tyto produkty jsou označovány jako BIG-IP. Hardwarová řada produktů nabízí opět řešení rozdělená do řad podle výkonnosti, jako tomu je i u konkurence. Můžeme tedy vybírat ze standartní řady 2000, 4000, 5000, 7000, 10000 a nejvýkonnější 12000.

Na všech řadách běží operační systém TMOS (Traffic Menagement Operating System). F5 dále nabízí velké množství nástrojů, které se dají dokoupit v podobě modulů. Můžeme tedy pořídit navíc nástroje jako jsou například:

- LTM (Local Traffic Manager), který slouží pro optimalizaci datového toku v lokální síti,
- GTM (Global Traffic Manager), který zabezpečuje DNS před DDoS útoky,
- APM (Access Policy Manager), který zjednodušuje uživatelům přístup k aplikacím a datům,
- ASM (Aplication Security Manager), který zabezpečuje aplikace,
- AFM (Advanced Firewall Manager), který slouží pro zabezpečení dat.

F5 využívá pro vyvažování zátěže tyto algoritmy: Round robin, Ratio, Least Connections, Weighted Least Connections, Observed, Predictive, Least Session a Faster. (F5, 2017)

5.1.4 Zhodnocení

Za pomoci klíčových vlastností jsme provedli výběr zařízení od jednotlivých dodavatelů. Počet L7 žádostí za sekundu jsme stanovili na 10 000 – 50 000 a počet SSL transakcí za sekundu na 1000 transakcí. Informace o schopnostech jednotlivých řešení jsou uvedeny v tabulce *Tab. 1*, která vychází z dokumentace dodavatelů.

Tab. 1 Porovnání komerčních řešení pro load balancing

	počet L4 připojení za sekundu	počet L7 žádostí za sekundu	maximální počet konkurenčních připojení na L4	maximální propustnost na L4	maximální propustnost na L7	maximum SSL transakcí za sekundu	přibližná cena(zdroj www.cdw.com)
Citrix NetScaler MPX 5550	neuvedeno	175 000	neuvedeno	neuvedeno	neuvedeno	1500	436 000 Kč(Standart Edition)
Barracuda ADC 540	120 000	neuvedeno	14M (nejspíš na L4, není uvedeno)	5 Gbps	3.6 Gbps	1400	251 000 Kč
F5 BIG-IP 2000s	75 000	212 000	5M	5 Gbps	5 Gbps	2000	484 000 Kč

Jak můžeme vidět v tabulce *Tab. 1*, ne každý z dodavatelů udává takové parametry, které se shodují s klíčovými vlastnostmi. Můžeme tedy dále pouze podle okolních parametrů odhadovat vlastnosti zbylé. Taktéž si můžeme všimnout, že počty žádostí na L7 za sekundu jsou oproti našim požadavkům poměrně vyšší. Zařízení s těmito parametry jsou však nejméně výkonnými, a proto bychom museli využít právě těchto zařízení, k uspokojení našich požadavků.

Dále si můžeme všimnout, že cena u Cintrix a F5 je poměrně vyšší, než je tomu u Barracuda. Je tomu tak nejspíš pouze kvůli tomu, že tito dodavatelé jsou na trhu déle, což znamená, že mají vybudovanou jakousi záruku kvality, postavenou na jejich jméně. Barracuda přišla na trh později a vychází z open source řešení.

5.2 Open-source řešení v podobě aplikačních proxy serverů

V případě, že hledáme dostupná řešení typu open-source, dostáváme se k pojmu aplikační proxy server.

Aplikační proxy server je speciální server, určený pro protokol nebo aplikaci. Slouží jako prostředník mezi klientem a nějakou síťovou službou. Klient vysílá požadavky, které proxy služba přejímá a předává dále serveru. Výsledek je pak poslán zpět

přes proxy klientovi. Proxy provádí funkci přesměrování a umí také hlídat obsah komunikace, který může zároveň modifikovat.

Dalším typem proxy serveru je reverzní proxy a tento typ je pro nás mnohem podstatnější, protože jedna z jeho vlastností je právě vyvažování zátěže. Může také ukládat požadavky do vyrovnávací paměti, čímž jsou při opakovaném požadavku odpovědi poskytnuty rychleji. Taktéž zvládá šifrování a dešifrování protokolem SSL. Reverzní proxy server se jeví jako obyčejný server, ale může za sebou ukrývat několik serverů, které řeší příchozí požadavky. (Sosinsky, 2010)

Většina aplikačních proxy serverů vychází z původních webových serverů projektu Apache a projektu Nginx.

5.2.1 HAProxy

HAProxy je open source řešení, které umožňuje vyvažování zátěže, vysokou dostupnost a proxy server pro jakoukoli aplikaci využívající protokoly TCP a HTTP. Toto řešení je vhodné pro aplikace, které musí snášet vysoký provoz, a tedy velký počet souběžných požadavků. Průběhem let se toto řešení stalo open source standartem pro vyvažování zátěže a je dodáváno s většinou běžných distribucí Linuxu. Je však dostupné taktéž pro platformy FreeBSD a Solaris.

Existuje několik verzí, které se odlišují v sadě funkcionalit, ale taktéž v letech zveřejnění jednotlivých verzí. Nyní se tedy setkáme s možnostmi jako: v. 1.3, v. 1.4, v. 1.5, v. 1.6, v 1.7. s

Pro vyvažování zátěže HAproxy využívá algoritmy Round Robin, Weight round robin, Dynamic round robin, Least connection a Source.

HAProxy je využíváno mnoha velkými společnostmi včetně společností jako jsou Stack Overflow, Reddit, Tumblr, Twitter nebo Instagram. (HAProxy, 2017)

5.2.2 Nginx

V případě Nginx mluvíme o open-source virtuálním webovém serveru, který podporuje protokoly HTTP, HTTPS, SMTP, POP3 a IMAP. Server je velmi rychlý a umí souběžně zpracovávat požadavky s nízkou náročností na systémové prostředky. Server ovšem poskytuje i další služby. Nginx obsahuje množství modulů, díky kterým můžete nakonfigurovat load balancer, jednoduchý file server, ale i komplikovanější reverzní proxy, a dokonce i mailové proxy servery. (Kutáč, 2016)

V Nginx serveru mohou moduly zastupovat jednu ze tří rolí:

- obslužná rutina (handler), která zpracovává požadavek a produkuje výstup,
- filtr, který upravuje výstup obslužných rutin (komprese, úprava hlavičekss,
- vyvažování zátěže, které vybírá koncový server a přeposílá mu požadavek.

Pokud se ukáže, že obslužná rutina má přeposlat požadavek jinam, volají se moduly pro vyvažování zátěže. (Kameníčková, 2015)

Zpracování HTTP požadavku z pohledu modulů probíhá tak, že klient pošle HTTP serveru požadavek, se kterým jádro Nginx serveru následně pracuje. Jádro vybere obslužnou rutinu na základě konfigurace odpovídající požadavku. V případě, že je stanovená konfigurace pro vyvažování zátěže, zvolí se cíl přesměrování. Rutina, sskterá tento

požadavek obsluhuje odvede svoji práci a pošle výstup filtrům. Po zpracování všemi filtry se výsledná odpověď pošle zpět klientovi. (Kameníčková, 2015)

Nginx podporuje tři vyvažovací metody: Round Robin, IP hash a Least Connected. (Nginx, 2017) Mezi jeho hlavní přednosti určitě můžeme zařadit vysokou výkonnost poskytované aplikace, jednoduché nasazení a snížení nákladů na infrastrukturu. Jako další výhodu bychom mohli zařadit i to, že oproti HAproxy má Nginx více využití a neslouží pouze pro rozklad zátěže. Díky své rychlosti je využíván ve velké míře například společnostmi Yandex, WordPress.com nebo Netflix. (Kutáč, 2016)

5.2.3 Zhodnocení

U open-source řešení nejsme schopni porovnávat vyvažovače zátěže tak, jako jsme porovnávání prováděli u hardwarových řešení. Je to z toho důvodu, že výkon udává dodávaný hardware. Výkon open-source řešení bude tedy záviset na hardwaru, na němž bude toto řešení nasazeno. Load balancery na softwarové bázi jsou mnohem levnějším řešním oproti hardwarovým alternativám.

6 Filtrování provozu

Filtrování provozu je další věcí, která by se v dnešní době neměla podceňovat. K tomuto účelu slouží firewall, který je jakousi stěnou mezi internetem a privátní sítí, již dokáže ochránit před různými útoky z veřejného internetu a zároveň dokáže řídit odchozí a příchozí data. Díky firewallu můžeme v naší síti nastavit spoustu věcí, jako například to, které protokoly budou mít do naší sítě přístup a naopak, které ho mít nebudou. Jinými slovy dokážeme omezit provoz některých protokolů. Správným nastavením filtrovacích pravidel dokážeme zabránit jak finančním ztrátám zaviněným například napadením našeho serveru, a tím pádem výpadku naší služby, ale dokážeme také zabránit únikům důležitých dat, které by mohly ve špatných rukou způsobit daleko větší ztráty.

Při řešení filtrování provozu se můžeme setkat jak se hardwarovým řešením, tak se softwarovými firewally, které pracují nad operačním systémem serveru.

Typy firewallů:
- paketový
- stavový
- aplikační

6.1 Paketový firewall

Filtrování paketovým firewallem je nejjednodušší a nejstarší formou. Veškerý odchozí i příchozí provoz mezi dvěma různými sítěmi prochází přes firewall. Filtr paketů prověří každou hlavičku paketu v izolovaném prostředí a určí, zda mu bude umožněno projít do sítě, kam je směřován, nebo zda by měl být jeho přenos zrušen na základě pravidel specifikovaných správcem sítě. (Ross & Kurose, 2012)

Zrušení může proběhnout dvěma způsoby. Jedním způsobem je zamítnutí paketu a jeho zahození. Druhým způsobem je zamítnutí paketu, jeho zahození a informování odesílatele paketu o jeho zamítnutí. Doporučuje se však naprostá anonymita, a proto se využívá zahození bez poskytování informace odesilateli. (Strebe, Perkins, 2003)

Filtrování u paketového filtru probíhá na síťové vrstvě modelu ISO/OSI a filtrovací rozhodnutí jsou založena na:

- zdrojové/cílové IP adrese,
- typu protokolu (TCP, UDP, ICMP, OSPF…),
- TCP nebo UDP zdrojové/cílovém portu,
- TCP inicializačním/potvrzovacím bitové značení (ACK, SYN …),
- typu ICMP zprávy,
- různých pravidlech pro odchozí a příchozí datagramy,
- různých pravidlech na různých rozhraní směrovače.

(Ross & Kurose, 2012)

U paketového filtrování lze filtrovat i další informace jako je přímé směrování a fragmentace. Pokud síť přímé směrování nepoužívá, mělo by být nastaveno, aby filtr všechny pakety s volbou přímého směrování zahazoval. Nebylo by tomu tak, hrozí nebezpečí nežádoucího útoku. Stejně tak se vyskytuje hrozba při fragmentaci, kdy útočník může zneužít fragmentaci a obejít tak filtr paketů při zpětném znovu sestavování paketu. (Strebe & Perkins, 2003)

Výhodou paketového filtru je jeho vysoká rychlost a taktéž to, že je tato funkce integrována na většině směrovačů. Nevýhodou je, že filtrace dosahuje pouze síťové úrovně, neumí tedy kontrolovat datovou část paketů, neuchovává stav připojení a neexistuje zabezpečení pro konkrétní služby. (Strebe & Perkins, 2003)

Typickými představiteli paketových filtrů jsou například ACL na Cisco IOS.

6.2 Stavový firewall

Stavový filtr řeší nedostatky paketového filtru tím, že uchovává stav celé komunikace, která prochází přes firewall, v paměti. Na základě zapamatovaného stavu následně určuje, zda by měly být jednotlivé pakety zahozeny či nikoliv. Stavový filtr pracuje na třetí a čtvrté vrstvě modelu ISO/OSI a zaznamenává informace o ustanovení relace v tabulce spojení tzv. stavové tabulce. Dokáže tedy rozpoznat, zda je paket součástí nějaké již povolené relace nebo zakládá relaci novou. Sada nakonfigurovaných pravidel pak určuje, zda má být žádost o novou relaci povolena či nikoliv.

Stavová tabulka obsahuje atributy jako jsou zdrojová/cílová IP adresa, čísla portů a sekvenční čísla, která odečítá ze všech procházejících paketů. Založená spojení na firewallu existují při nečinnosti jen určitou dobu. Není-li detekován žádný provoz, spojení se považuje za uzavřené a záznam o této relaci je z tabulky odstraněn. (Sosinsky, 2010) (Strebe, Perkins, 2003)

Typickým představitelem stavového firewallu je framework Netfilter, dále pak například Cisco ASA nebo Juniper SRX.

6.3 Aplikační firewall

Aplikační filtr, nebo také Proxy firewall, je rozšířením stavového filtru o filtraci na základě obsahu známých protokolů aplikační vrstvy. Filtr je schopný rozpoznat, zda komunikace na daném portu nese znaky odpovídajícího protokolu či aplikace, jež se na tomto portu očekává. Aplikační filtr pracuje na sedmé vrstvě neboli aplikační vrstvě, jak již z názvu vyplývá. Veškerá komunikace probíhá formou dvou spojení. Nejprve se klient připojí na proxy firewall, ten příchozí spojení zpracuje a na základě požadavků klienta otevře nové spojení k serveru. V novém spojení k serveru je pak v roli klienta právě proxy firewall. Data, která firewall dostane od serveru, pak zase v původním spojení předá klientovi. Firewall tedy vždy skrývá IP adresu klienta a jako klient vystupuje on sám. (Firewall, 2017)

Užitečnost tohoto filtru vyplývá zejména v jeho schopnosti inteligentně reagovat na změny podmínek. Představme si situaci, kdy máme filtrování nastavené na port 53 (DNS port) a DNS server se stane cílem útoku DoS. Stavový firewall by v takovéto

situaci tento port zavřel, jelikož by byl přetížen. Aplikační filtr však dokáže i pod útokem port otevřít pouze pro potřebné interní překládání adres a neomezovat tak tuto službu. (Sosinsky, 2010)

Nevýhodou aplikačního firewallu jsou zejména vyšší hardwarové nároky a netransparentnost, kde pro každou aplikaci musí být zvláštní proxy a samotná aplikace musí podporovat připojení pomocí proxy. (Strebe & Perkins, 2003)

Na trhu existuje spoustu proxy firewallů, například Zorp firewall, Novell BorderManager nebo Squid.

6.4 NAT

NAT (Network Address Translation) je technologie překladu síťových adres, která umožňuje nahradit adresy v paketech na základě položek nacházejících se v tzv. mapovací tabulce. Tato technologie je velkým přínosem v mnoha směrech. Díky NAT technologii dokážeme směrovat provoz do a z interních sítí, které jsou adresovány privátními adresami, jež jsou nesměrovatelné. Řeší to tedy problém rozsahu adres IPv4 a zároveň je skryt celý rozsah prvků v interní síti před veřejným internetem. NAT funkce spočívá v tom, že dovoluje mapovat jednu adresu na jiné adresy. (Sosinsky, 2010)

7 Porovnání komerčních a open-source řešení pro filtrování provozu

Stejně jako tomu bylo u vyvažování zátěže, i u filtrování provozu se můžeme setkat s řešeními, která jsou dedikovaná hardwarem, nebo jsou dodávaná formou licence a s řešeními, která jsou dostupná jako open-source. V následujících podkapitolách se s některými z nich seznámíme. Pro náš provoz dále uvažujeme stavový firewall. Mezi hardwarová řešení jsme vybrali Cisco ASA a Juniper SRX a mezi open-source IPFW a Netfilter.

Nyní se seznámíme s komerčními řešeními, která následně porovnáme podle vlastností, jež jsou pro společnost prioritní a taktéž z hlediska cenové politiky. Klíčové vlastnosti pro porovnání jsou:

- propustnost firewallu,
- propustnost VPN,
- propustnost IPS.

7.1 Komerční řešení

7.1.1 Cisco ASA

ASA (Adaptive Security Appliance) je zařízení dedikované hardwarem od společnosti Cisco, které běží na operačním systému Cisco IOS (Internetwork Operating Systém). Toto zařízení slouží jako stavový firewall, ale pro vyšší zabezpečení lze využít i IPS (Intrusion Prevention System) nebo VPN (Virtual Private Network). Toto řešení nabízí zabezpečení všech typů sítí. Pomocí zásuvných modulů lze dosáhnout ještě většího počtu bezpečnostních možností, a tak navýšit zabezpečení sítě. Některé z vlastností sou však licencované samostatně. (Lynt, 2013)

Cisco ASA lze konfigurovat jak pomocí konzole, tak pomocí uživatelského rozhraní. Může fungovat ve 2 základních módech: transparentním a routovaném. V transparentním módu funguje pouze na L2 pro filtrování provozu a lze nasadit beze změny konfigurace sítě. V routovacím módu má firewall svou vlastní IP adresu v síti, na kterou je nasměrován provoz ostatních strojů.

Existuje několik řad těchto zařízení: ASA 5505, ASA 5510, ASA 5520, ASA 5540, ASA 5550, ASA 5580. Tyto řady se odlišují především v jednotlivých propustnostech firewallu a v počtu konkurenčních připojení. Od těchto parametrů se samozřejmě odráží také cena jednotlivých zařízení. (Cisco, 2016)

Stanovování pravidel u Cisco zařízení probíhá pomocí ACL (Access Control List) manažera. ACL je seznam pravidel, který řídí přístup a je součástí IOS.

7.1.2 Juniper SRX

Společnost Juniper Networks nabízí hardwarové řešení filtrování provozu ve formě produktu Juniper SRX, který kombinuje Next-Generation firewall a služby UTM (Unified Threat Management) se směrováním a přepínáním v jednom. Firewally příští gene-

race jsou schopny provádět plnou inspekci paketů a mohou používat bezpečností zásady založené na informacích vrstvy L7.

Dodavatel rozděluje svá řešení do tří kategorií podle toho, o jak rozsáhlou společnost se jedná. Pro malé podniky a pobočky nabízí produkty SRX110, SRX220, SRX 300 a SRX 550. Pro středně velké podniky a datová centra nabízí produkty SRX1400, SRX1500, SRX3400, SRX3600 a SRX4000. Pro ty nejnáročnější nabízí produkty SRX5400, SRX5600 a SRX5800, které využijí už jen opravdu velká datová centra a poskytovatelé služeb. Všechny zmíněné produkty se liší opět v propustnostech a zejména v samostatných rozšířeních jednotlivých služeb, či jejich celkového zavedení.

Zařízení SRX běží na systému Junos OS, který je založen na FreeBSD unixovém jádře. Pro nastavování pravidel zde slouží rozhraní příkazového řádku CLI (Command Line Interface). (Juniper Networks, 2017)

7.1.3 Zhodnocení

Za pomoci klíčových vlastností jsme provedli výběr zařízení Cisco ASA a Juniper SRX, která vyhovují našim požadavkům. Propustnost firewallu jsme zvolili do 200 Mbps, propustnost VPN do 100 Mbps a propustnost IPS taktéž do 100 Mbps. Po prostudování dokumentací jednotlivých dodavatelů získáváme tabulku *Tab. 2*.

Tab. 2 Porovnání komerčních řešení pro filtrování provozu

	propustnost firewall	propustnost VPN	propustnost IPS	přibližná cena
Cisco ASA 5510	300 Mbps	170 Mbps	150 Mbps	17 000 Kč
Juniper SRX220	300 Mbps	100 Mbps	80 Mbps	23 000 Kč

V tabulce *Tab. 2* můžeme vidět výběr jednotlivých produktů, které vyhovují našim požadavkům. Juniper SRX220 nevyhovuje pouze v maximální IPS propustnosti. Pokud bychom však chtěli, aby i tento parametr vyhovoval přesně našim požadavkům, museli bychom přistoupit k vyššímu modelu SRX, což by však mělo razantní dopad na cenu.

7.2 Open-source řešení

7.2.1 IPFW

Ipfirewall je nativní paketový filr, který je součástí unixové platformy FreeBSD (Berkeley Software Distribution). Tento filtr se ovládá pomocí uživatelského rozhraní IPFW. Filtr poskytuje jak paketové, tak stavové filtrování.

IPFW se skládá z několika komponent:

- systém paketového filtrování, pro zpracování pravidel a evidence procházejících paketů,

- logovací systém,
- NAT,
- komponenta, pro vyvažování zátěže,
- komponenta, k přesměrování paketů,
- komponenta, přepínání paketů.

Jednotlivá pravidla vytvořená za účelem řízení toku paketů vytvářejí sadu pravidel firewallu. Pořadí pravidel sady je dáno číslem každého z nich. To, jestli bude dané pravidlo sady uplatněno na procházející paket stanovuje tzv. selektivní parametr. Tímto parametrem se specifikuje hodnota určitého atributu paketu. Pokud se hodnota zkoumaného atributu paketu procházejícího firewallem shoduje s hodnotou selektivního parametru, je dané pravidlo na paket uplatněno a další pravidla se již netestují. Pokud procházející paket nevyhoví selektivní podmínce jednoho pravidla, je porovnán s následujícím pravidlem sady. Pokud paket projde všemi pravidly, stará se o něj dále výchozí firewallová politika, která je nastavena. Tato politika může takový paket dále povolovat nebo zakazovat. (Milanov, 2006)

7.2.2 Netfilter

Jak už jsme dříve poznamenali, Netfilter je frameworkem, který je součásti samotného jádra Linuxu. Tento framework umožňuje filtrování provozu na samotném linuxovém serveru. Kromě paketového filtrování nabízí taktéž stavové filtrování, překlad síťových adres NAT a překlad síťových portů PAT. Díky Netfilteru můžeme tedy vybudovat kvalitní firewall. (Netfilter,2014)

7.2.2.1 Tabulky a řetězy

Netfilter se skládá ze čtyř tabulek:

- Filter:
  - Slouží k filtrování provozu (odmítání, zahazování a přijímání paketů).
  - Pracuje s řetězy INPUT, OUTPUT, FORWARD.

- NAT:
  - slouží pro překlad adres a směrování provozu.
  - pracuje s řetězy PREROUTING, POSTROUTING a OUTPUT.

- Mangle:
  - Slouží pro modifikaci hlavičky paketu (např. modifikace ToS, QoS, TTL …).
  - Pracuje s řetězy PREROUTING, INPUT, FORWARD, OUTPUT a POSTROUTING.

- Raw:
  - Slouží k označení provozu, který nemá jít do connection trackingu.
  - Pracuje s řetězy PREROUTING a OUTPUT.

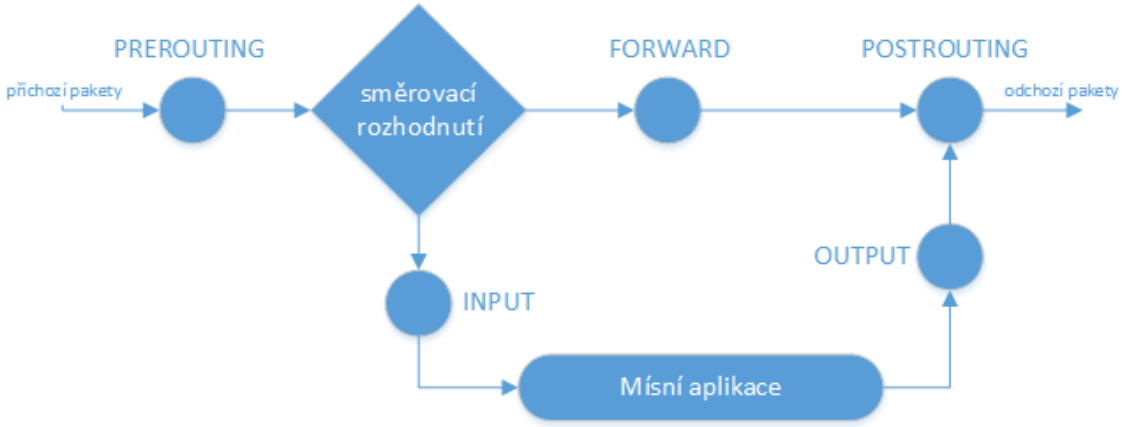
(Balej, 2016)

Řetězy lze vytvářet i vlastní, avšak původní nelze mazat. Jedná se o řetězy administrátorem nastavených pravidel.

Na paket, který prochází určitým řetězem, jsou pak aplikována jednotlivá pravidla od prvního pravidla k následujícímu, dokud paket některému z pravidel nevyhoví. Pokud některému z pravidel vyhoví, pravidlo určí, co se sním stane dál. Nejdůležitějšími a také nejčastějšími akcemi jsou akce DROP, REJECT a ACCEPT. Existují však i další akce, kterým se však nebudeme v této práci věnovat. Paket může být tedy zahozen (DROP), odmítnut (REJECT), přijat (ACCEPT) nebo předán jinému řetězu.

Pokud by paket nevyhověl žádnému z pravidel, pak o jeho osudu rozhoduje politika, která je nastavená na daném řetězu, kde může být paket buď přijat nebo zahozen. (Dočekal, 2010)

Jednotlivé řetězy, jak už název napovídá, určují, kam paket míří. Pokud např. přijímáme paket, aplikují se na něj pravidla z řetězu INPUT, pro odchozí pakety se aplikují pravidla z řetězu OUTPUT. Řetěz FORWARD použijeme, pokud náš server funguje jako směrovač a přeposílá pakety mezi sítěmi. Díky PREROUTING pravidlům můžeme modifikovat DNAT u příchozích paketů a díky POSROUTING pravidlům zase SNAT u odchozích paketů. (Root,2006)



Obr. 4 Netfilter – tok paketů

7.2.2.2 IPtables

Iptables je rozhraní, které nám umožňuje práci s firewallem, tedy nastavování filtračních pravidel. Filtrační pravidla nastavujeme jednotlivým řetězům pravidel. Procházející paket je pak řízen právě těmito pravidly v závislosti na tom, v jakém směru se aktuálně pohybuje. Pravidla mohou být stavová i nestavová.

Příkazy v rozhraní iptables mají následující syntaxi:

```
iptables [tabulka] [akce] [řetězec] [match] [cíl]
```

Jako první parametr příkazu je název tabulky, se kterou budeme manipulovat. Pokud tuto část vynecháme, příkaz se bude vztahovat k tabulce „filter“. V části „akce“ udáváme jeden z následujících parametrů:

- -A (apend), pro přidání nového pravidla na konec řetězu,
- -D (delete), pro smazání pravidla,
- -R (replace), pro nahrazení pravidla, specifikovaného jeho číslem, jiným pravidlem,
- -I (insert), pro vložení pravidla na začátek řetězu,
- -L (list), pro vypsání všech pravidel,
- -F (flush), pro odstranění všech pravidel,
- -N (new chain), pro vytvoření vlastního řetězu,
- -X (delete chain), pro smazání vlastního řetězu,
- -P (policy), pro nastavení výchozí politiky řetězu,
- -E (rename chain), pro přejmenování vlastního řetězu,
- -Z (zero), pro vynulování počítadel datagramů.

(Duc & Lesník, 2012)

Jako další parametr se uvádí název řetězu, ke kterému se pravidlo bude vztahovat. V části „*match*“ pak uvádíme několik parametrů, podle toho, co všechno v daném pravidlu omezujeme. Pokud některý z parametrů nezadáme, použije se jeho implicitní hodnota. Nyní se seznámíme s některými z těchto parametrů:

- -i (input interface), pro definici vstupního rozhraní,
- -o (output interface), pro definici výstupního rozhraní,
- -p (protocol), pro definici typu protokolu,
- -s (source), pro definici zdrojové IP adresy,
- -d (destination), pro definici cílové IP adresy,
- --sport (source port), pro definici zdrojového portu,
- --dport (destination port), pro definici cílového portu,
- -m multiport, pro definici kombinace cílových portů,
- -m conntrack --ctstate, pro definici stavu u stavového filtrování.

(Duc & Lesník, 2012)

Poslední částí příkazu je „*cíl*“. V této části definujeme, co dané pravidlo provede, pokud mu paket vyhoví. K této definici využíváme parametr „-j“ (jump), za kterým uvedeme volbu cíle. (Duc & Lesník, 2012)

7.2.2.3 Stavové filtrování

Jak jsme již zmínili, Netfilter dokáže filtrovat i podle stavu spojení. Pomocí této vlastnosti je možné pravidla pro filtrování nastavit tak, že se budou vztahovat pouze k paketům s žádostí o vytvoření nového spojení a pakety, které již patří některému z vytvořených spojení, budou rovnou propuštěny. (Dočekal, 2010)

Stavové filtrování pak rozlišuje čtyři různé stavy:

- NEW – pro paket, který vytváří nové spojení nebo se vztahuje ke spojení, kde doposud neproběhla obousměrná komunikace,
- ESTABLISHED – pro paket, který je již součástí existujícího spojení, kde probíhá komunikace oběma směry,
- RELATED – pro paket, vytváří nové spojení, ale vztahuje se již k některému z existujících (např. FTP, ICMP chybová zpráva, …),
- INVALID – pro paket, který se nevztahuje k žádnému existujícímu spojení.

(Dočekal, 2010)

7.2.3 Zhodnocení

Stejně jako tomu bylo u softwarových řešení pro vyvažování zátěže, tak i zde nejsme schopni porovnávat open-source filtrovací řešení. Výkon udává opět až hardware, na kterém je řešení nasazeno. Nicméně v našem případě můžeme porovnávat to, že každý z open-source nástrojů běží na jiné platformě. Vybíráme Netfilter, protože všechny stroje, které se v síťové infrastruktuře momentálně nacházejí jsou stavěné za pomoci linuxových distribucí. Právě z důvodu použití nástroje Netfilteru pro další práci, je mu v této kapitole věnováno více prostoru.

8 Charakteristika firmy a popis současného řešení síťové infrastruktury

Dříve, než začneme popisovat síťovou infrastrukturu firmy v jejím původním stavu, přiblížíme si základní charakteristiky této společnosti.

Společnost, pro níž zpracováváme řešení infrastruktury, se jmenuje Aliacte s.r.o. a zabývá se především vývojem aplikací pro řízení oběhu produktů a výrobků. Pro vývoj aplikací je zcela jistě potřebný rozsáhlý tým zaměstnanců, kteří kooperují a řeší vývoj na odlišných úrovních. Jejich činnost je tedy různorodá. Ve společnosti se tak můžeme setkat se zaměstnanci na pozicích programátor, infrastrukturní specialista, helpdesk, SW architekt, SW grafik, web programátor a grafik. Strukturu společnosti lze dále rozdělit na tým programátorů a tým infrastruktury, kde je každý z týmu zastřešen team leadrem.

Celá firma podléhá přísnému projektovému řízení, přičemž hlavní slovo mají SCRUM masteři pro vývojové práce. Pod ně pak spadají programátoři, dle určení a grafik, dle určení.

Na základě toho, že společnost poskytuje zejména aplikace, skládá se její síťová infrastruktura z následujících prvků:

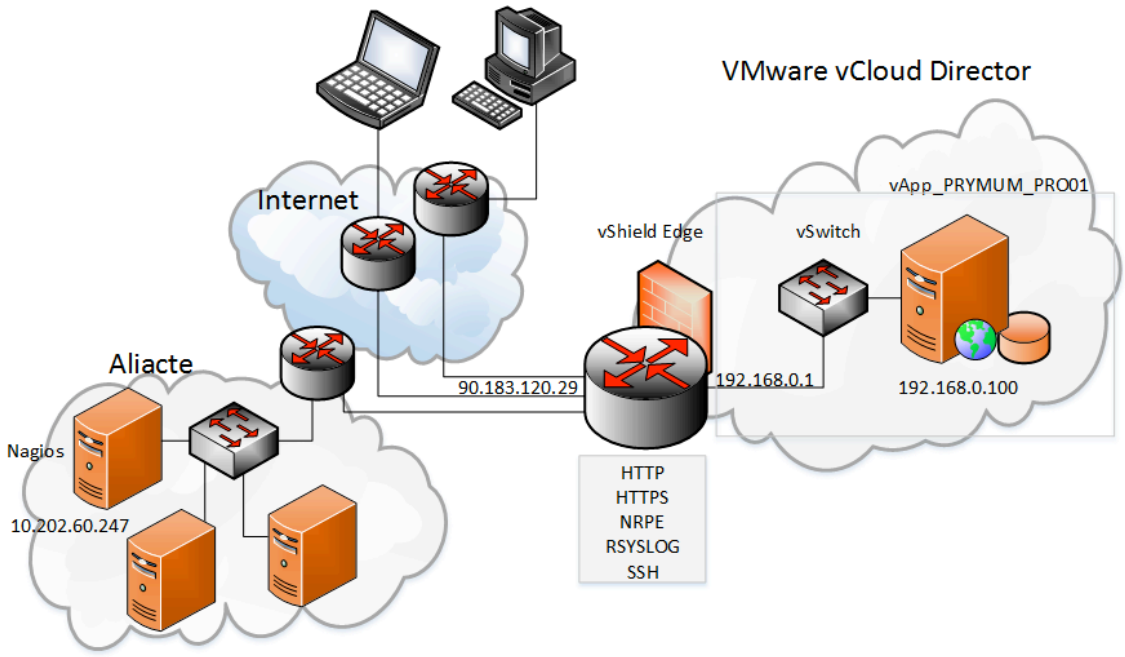
- **Veřejný cloud**, v němž firma poskytuje cloud z vlastních sdílených prostředků jako službu zákazníkům z řad veřejnosti, kterým tímto způsobem konkrétně poskytuje přístup do aplikačního a databázového prostředí pro provoz služeb.
- **Interní cloud**, který slouží firmě pro vlastní interní využití. Je provozován na zařízeních firmy, je plně virtualizovaný a slouží k testování operačních systémů, databází, aplikací a v neposlední řadě slouží také k ukládání a zálohování dat a pro provoz monitorovacích systémů.
- **Veřejný SaaS cloud** je určen pro provoz verzovacích systémů typu GitHub a systémů pro řízení práce programátorských týmů.

Webová aplikace, kterou společnost vyvíjí, bude přístupná uživatelům jak na webovém portálu, tak v podobě mobilní aplikace. Je tedy nutné zabezpečit síť jak před přetížením, tak před útoky zvenčí. Úprava síťové infrastruktury bude prováděna právě tak, aby byla co neodolnější a společnost se vyvarovala jakýchkoliv ztrát. Na úvod se ale seznámíme se sítí, jež se nachází v prostředí veřejného cloudu, v jejím původním stavu.

8.1 Síťová infrastruktura v původním stavu

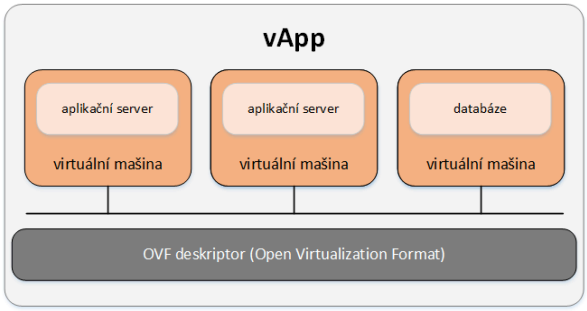
Celá infrastruktura se nachází na interním cloudu, a to konkrétně na cloudu společnosti O2, který nabízí svá řešení v podobě VMware vCloud Director. VMware vCloud Director umožnuje vytvořit a doručovat koncovým zákazníkům infrastrukturu jako službu. Síťová infrastruktura je tedy celkově virtualizovaná, což je značnou výhodou, neboť toto řešení urychluje modernizaci sítí a poskytování nových služeb. Taktéž snižu-

je náklady společnosti, která si zde může vytvářet své vlastní virtuální datová centra. Momentálně firma využívá jedno datové centrum. Jak můžeme vidět na obrázku *Obr. 5*, původní síť se skládá z jednoho routeru, jednoho webového serveru a jedné databáze.



Obr. 5 Síťová infrastruktura v původním stavu

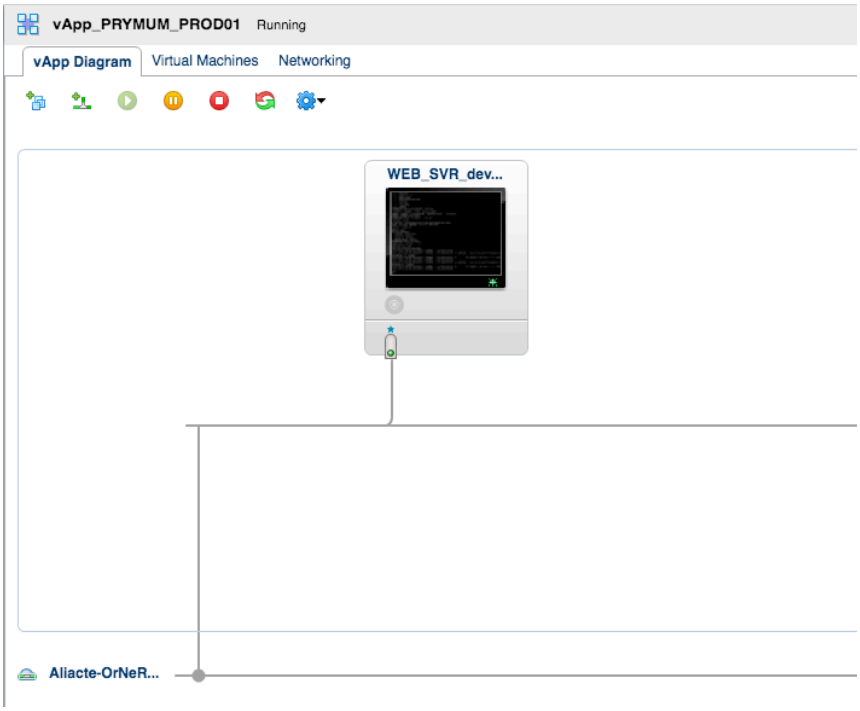
Pro virtualizaci v prostředí VMware slouží kontejnery vApp, které si můžeme představit jako balík IT služeb, v němž může být jedna nebo více virtuálních strojů. Každý kontejner vApp taktéž obsahuje jednu nebo více vApp sítí, které mohou být připojeny přímo na podnikovou síť, mohou být směrovány na podnikovou síť nebo mohou sloužit jako interní síť. (YouTube). Strukturu vApp znázorňuje obrázek *Obr. 6*.



Obr. 6 Znázornění virtuálních strojů uvnitř vApp

Společnost momentálně využívá jednu vApp, v níž se nachází virtuální stroj s webovým serverem, a druhou vApp, již je zcela stejná a slouží pro testování. Jak jsme

si již výše zmínili, každý takový kontejner obsahuje alespoň jednu vApp síť. V našem případě jde o interní síť, která je následně směrována do veřejného internetu. Toto směrování zařizuje virtuální router vShield Edge. Na obrázku *Obr. 7* vidíme připojenou produkční vApp k interní síti.



Obr. 7 Původní vApp kontejner a jeho interní síť

Po nastudování toho, jak vlastně původní infrastruktura v prostředí VMware vypadá, získáváme další podrobné informace, jež jsou pro nás klíčové.

8.1.1 Router

Síťová infrastruktura využívá virtuální router v podobě služby vShield Edge, který zde slouží jako výchozí brána a má na starost několik funkcí. Představuje několik možností, jak zabezpečit prostředí veVMware:

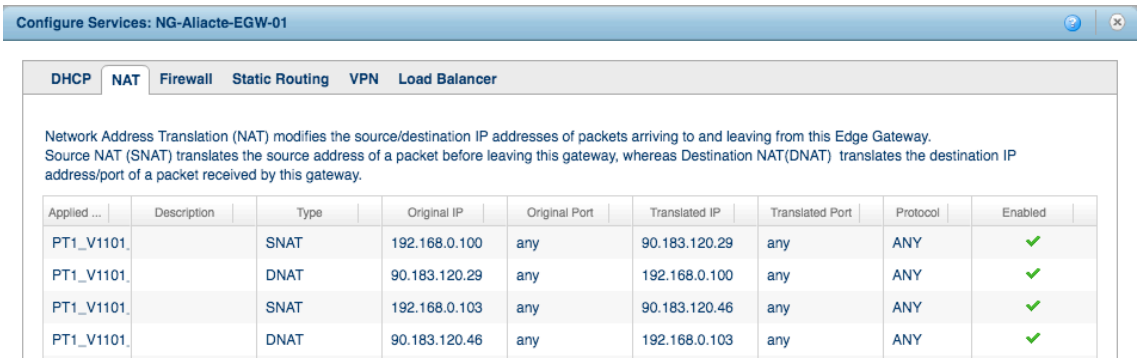
- firewall, pro filtrování provozu,
- rozdělení sítě na jednotlivé VLAN,
- překlad síťových adres pomocí NAT,
- přesměrování portů pomocí port-forwarding,
- routing,
- IPsec VPN pro ochranu komunikace,
- load balancer pro vyvažování zátěže.

Router nám zde odděluje privátní/VIP síť od veřejného internetu. IP adresa 192.168.0.1/24, která je nastavena na routeru, slouží jako výchozí brána privátní sítě.

Dále je také nastaven statický rozsah IP adres v rozmezí 192.168.0.100/24 – 192.168.0.199.

VMware vShield je konfigurován pomocí vShield Manager, který umožňuje centrální správu skrze webové rozhraní. Když otevřeme okno pro konfiguraci služeb, zobrazí se nám záložky a v každé záložce konfigurujeme danou službu zvlášť.

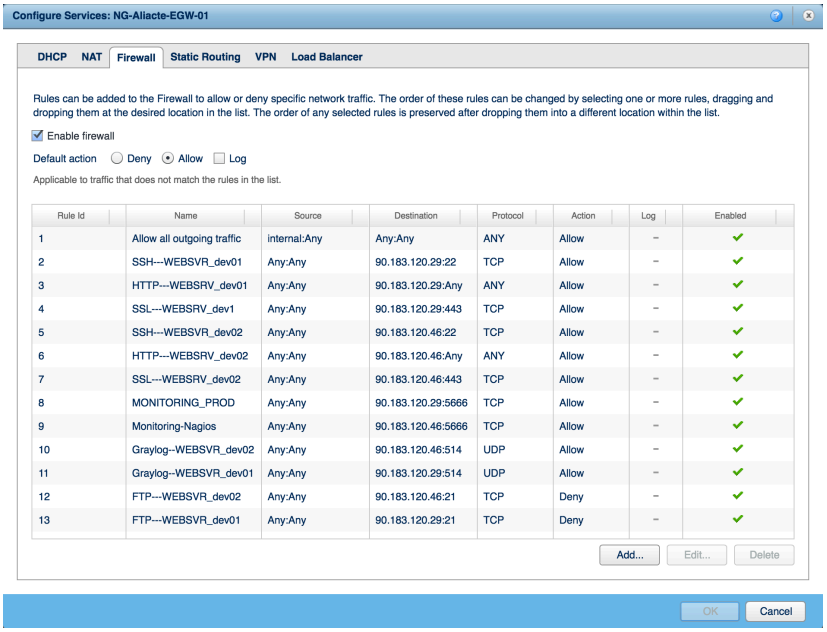
V tabulce NAT nacházíme překlady adres pro obě dvě zmiňované vApp, jak můžeme vidět na obrázku *Obr. 8*.



Obr. 8 Tabulka NAT v prostředí WMware

V záložce Firewall pak nacházíme pravidla pro filtrování provozu, která se opět vztahují k oběma vApp. Platí, že všechno, co není výslovně povoleno, je zakázáno. Jak můžeme vidět na Obr. 9, nastavena jsou zde pouze pravidla pro služby:

- SSH, pro možnost vzdáleného přístupu,
- HTTP, pro výměnu hypertextových dokumentů,
- SSL, pro zabezpečený datový přenos,
- Monitoring – Nagios, pro monitorování sítě,
- Graylog, pro záznam programových zpráv.



Obr. 9 Tabulka Firewall v prostředí WMware

8.1.2 Web Server

Web server, díky kterému je aplikace poskytována, běží na platformě Linux, konkrétně na distribuci Ubuntu Server verze 14.04. Pro softwarový webový server je zde využíván Apache HTTP Server verze 2, který je nejpoužívanějším webovým serverem vůbec. Jako další služby na webovém serveru běží např. MongoDB, SSH nebo SNMP. Server má nastavenou statickou IP adresu 192.168.0.100/24.

V současné době nejsou na webovém serveru nastavena žádná pravidla pro filtrování. Filtrování řeší pouze vShield Edge jak jsme již dříve zmínili.

8.1.3 Databáze

Jako databázové řešení společnost využívá MongoDB, které je open-source a řadí se do kategorie NoSQL. Databáze typu NoSQL jsou velmi oblíbené v cloud computingu. Místo tabulek v tradičních relačních databázích, které mohou omezovat požadavky aplikace, využívá MongoDB soubory BSON a dynamické databázové schéma, které umožňuje vytváření a integraci dat pro aplikace jednodušeji a rychleji. (MongoDB, 2016) Tato databáze běží jako služba přímo na webovém serveru.

Tato infrastruktura má nedostatečné zabezpečení. Jak jsme již zmínili dříve, poskytování veřejně dostupné aplikace by mělo být zabezpečeno mnohem více. Původní stav je zabezpečen pouze jedním firewallem.

9 Úprava síťové infrastruktury pro rozložení zátěže

Hlavním cílem práce, jak již bylo několikrát zmíněno, je zajištění dostupnosti webové aplikace, v podobě rozkladu zátěže a zabezpečení proti výpadku, zabezpečení proti útokům a celková optimalizace řešení. Infrastrukturu budeme upravovat tak, aby byla v souladu s GDPR (General Data Protection Regulation), tedy v souladu s novou legislativou EU, která zatím nevstoupila v platnost, ale byla již schválena. Jde o zvýšení ochrany osobních dat občanů, s cílem hájit zacházení s jejich daty a osobními údaji.

9.1 Postup úpravy

Původní řešení obsahuje pouze jeden webový server, na kterém zároveň běží i databáze jako služba. Toto řešení není efektivní a není jím zajištěna dostupnost služby v případě výpadku. Firewall je v původním řešení postavený na routeru a filtrace provozu tedy probíhá pouze na jedné úrovni. Pro lepší zabezpečení by bylo vhodné zavést víceúrovňové filtrování. V prvním kroku bychom tedy měli oddělit databázi od samotného webového serveru, abychom dosáhli toho, že každý ze serverů bude poskytovat pouze svou specifickou službu.

9.1.1 Oddělení databáze od webového serveru

Aby bylo řešení efektivní, je nutné izolovat webový server a databázi. Jelikož se síťová infrastruktura nachází v cloud prostředí, stačí v prostředí VMware zavést vApp, ve které by byly vytvořeny dva virtuální stroje. Jedna pro webový server a druhá pro databázový server.

V testovacím prostředí budeme vytvářet taktéž dva virtuální stroje pro demonstraci webového a databázového serveru. Co se týče operačních systémů jednotlivých strojů, budeme pokračovat na open-source platformách, protože přinášejí spoustu výhod zejména v nasazování, následné administraci či zabezpečení. V neposlední řadě je důležité, že jsou dostupné zdarma.

Pro webový server tedy budeme i nadále využívat operační systém Ubuntu Server, ale tentokrát jeho novější verzi, a to verzi 16.04. Ta drží krok v nejnovějších trendech a je taktéž využívána v cloud computingu. Jako službu pro webový server budeme opět využívat Apache verze 2. Co se týče databázového serveru budeme dále využívat open-source řešení v podobě MongoDB, které je pro cloudové řešení ideální.

9.1.2 Vytvoření množiny webových serverů pro rozložení zátěže

Pokud chceme zajistit škálovatelnost a dostupnost služby, je potřeba zavést více než pouze jeden webový server, který naši službu poskytuje. V případě výpadku serveru v architektuře právě s jedním serverem nezmůžeme nic a naše služba se stane nedostupnou.

Vytvoření množiny serverů nám umožní postavení load balanceru přímo před tuto množinu. Následná komunikace pak bude směřována na tento load balancer, který za sebou bude ukrývat množinu serverů, na než bude rozkládat zátěž. Load balancer se tedy bude tvářit jako konečný webový server, ale ve skutečnosti jich za sebou bude ukrývat více.

Pro vysokou dostupnost zavedeme tedy více serverů, konkrétně tři, což je jakýmsi optimálním minimem pro rozložení zátěže. V případě výpadku jednoho ze serverů lze dále rozkládat zátěž na zbylé dva servery.

Jak v prostředí VMware, tak v našem testovacím prostředí stačí přidat další dva virtuální stroje, které budou nakonfigurovány zcela stejně jako první webový server.

9.1.3 Výběr load balanceru a jeho zavedení

Při porovnávání možností vyvažování zátěže jsme se setkali s komerčními prvky, které mohou být jak v podobě dedikovaného zařízení, tak v podobě softwaru, jenž je ovšem licencovaný. Dále jsme se setkali s open-source řešeními, které se vyskytují v podobě aplikačních proxy serverů.

Jelikož řešíme takovou infrastrukturu sítě, která se nachází v prostředí cloudu, dedikovaná zařízení jako možnost řešení rovnou vynecháme.

Zbývají tedy dodavatelé, kteří poskytují komerční řešení v podobě licencovaného softwaru a open-source řešení.

Ve společnosti jsme se shodli, že load balancer v podobě open-source je pro toto řešení zcela dostačující, protože v našem případě nepotřebujeme žádné další přídavné služby, které dodavatelé poskytují. Stačí nám základní funkce pro rozkládání zátěže. Pro společnost by tedy bylo komerční řešení zbytečným nákladem navíc, jenž by se projevoval zejména platbou licence.

Pro rozklad zátěže jsme vybrali řešení v podobě Nginx, které je podporováno platformou Ubuntu od vezre 14.04.

V produkčním i testovacím prostředí je opět nutné vytvořit další virtuální stroj, který bude sloužit jako load balancer. Opět budeme virtuální stroj stavět za pomoci operačního systému Ubuntu Server, na němž následně nainstalujeme Nginx, který nakonfigurujeme tak, aby sloužil pro rozklad zátěže na jednotlivé servery.

9.1.4 Vytvoření logického rozdělení sítě pomocí VLAN

Po uvážení všech předchozích kroků dostáváme infrastrukturu, kde se nacházejí tři webové servery, které jsou skryty za load balancerem a databázový server.

Pro ještě lepší zabezpečení můžeme naši síť segmentovat na dvě menší sítě pomocí VLAN. Dosáhneme tak dalšího stupně zabezpečení a zároveň jakési organizace a elegance. Do jedné VLAN zařadíme load balancer s aplikačními servery a do druhé VLAN pak databázový server.

V prostředí VMware dosáhneme dvou virtuálních sítí tak, že vytvoříme dvě vApp. V každé vApp je pak potřeba vytvořit jednotlivé virtuální stroje, které náleží každé z VLAN.

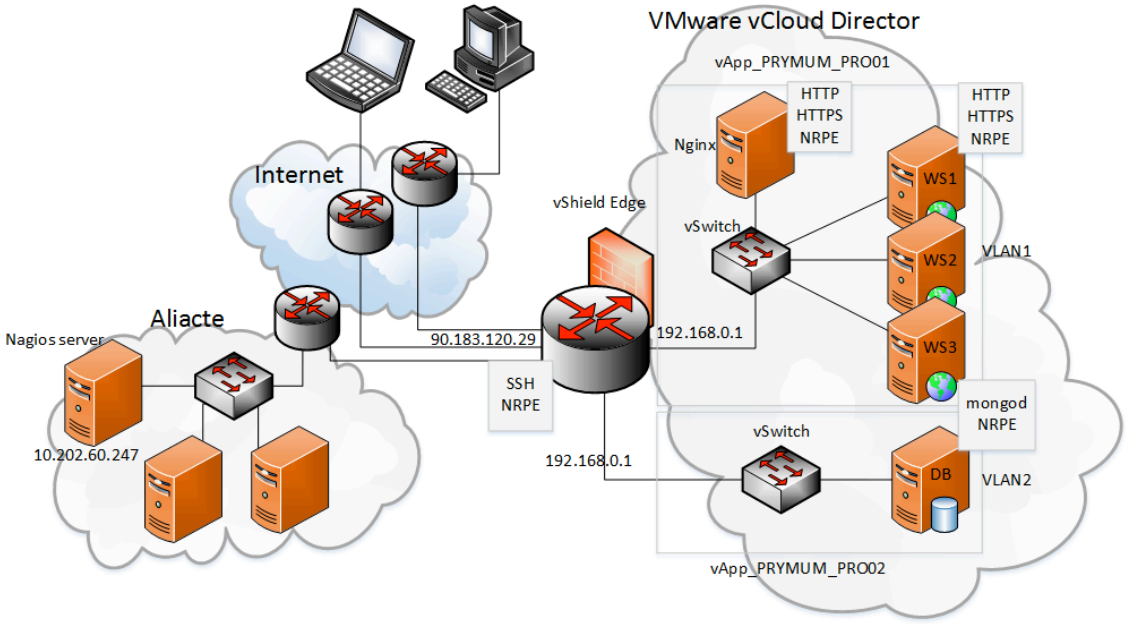
V testovacím prostředí budeme simulovat toto rozdělení pomocí dvou různých sítí.

9.1.5 Zavedení firewallové politiky

V poslední řadě je třeba upravit a zavést bezpečnostní politiku. Budeme vycházet z původní sady pravidel, kterou jsme viděli nastavenou na výchozí bráně původní infrastruktury. Jako brána zde vykonává svou službu vShield Edge a v původní síti byla nastavena pravidla pouze na tomto místě. V upravené infrastruktuře zavedeme firewallová pravidla stejně tak na jednotlivé servery, čímž opět zvýšíme zabezpečení.

V prostředí VMware je pak nutné změnit původní pravidla na routeru, a to z důvodu segmentace služeb do jednotlivých VLAN. V testovacím prostředí vytvoříme přímo pravidla, která se budou odvíjet od směru provozu. V obou prostředích je pak nutné zavést filtrování i na virtuálních strojích, které reprezentují servery.

Na obrázku *Obr. 10* můžeme vidět finální návrh řešení, obsahující všechny popsané kroky.



Obr. 10 Návrh nového řešení síťové infrastruktury

10 Bezpečnostní politika pro filtrování provozu k serverům

Některé společnosti rychle zavádějí technologii, aniž by nejdříve věnovaly nějaký čas vytvoření zásad zabezpečení. Nesmíme zapomínat, že by pravidla, která budeme implementovat, měla být v souladu s našimi zásadami. Pravidla pro filtrování provozu by měla být tedy stanovena ještě před samotnou implementací. V infrastrukturní úpravě budeme stavět firewall jak na routeru, tak na samotných serverech. Vytvoříme tak víceúrovňové zabezpečení. Musíme si zejména ujasnit, jaký provoz bude přes každý z firewallů procházet. Jak bychom ale měli postupovat při vytváření filtrovací politiky?

Podle Watkins a Walance (2008) je při vytváření filtrovací politiky nejlepší držet se následujících postupů:

- zákaz veškerého provozu ve výchozím nastavení a povolení pouze těch služeb, které jsou pro naše podnikání podstatné,
- pochopení toho, jaké informace uživatelé potřebují a podle toho poskytnutí přístupu pouze k těmto informacím,
- zajištění toho, aby se pravidla neopakovala, nebo tam některá nebyla zbytečně,
- zákaz fyzického přístupu k filtrovacímu zařízení, přístup pouze oprávněným osobám,
- vypracování seznamu protokolů, které jsou nezbytné pro podnikové operace
- využívání logovacích záznamů,
- vytvoření demilitarizovaných zón pro omezení přístupu k zónám,
- využívání firewallu jako samostatného zařízení (nevyužívat jej zároveň jako server nebo pracovní stanici),
- nastavení limitů pro připojení,
- vytvoření přístupových účtů k firewallu pouze pro správce (silná hesla),
- kombinování paketového filtrovaní se stavovým filtrováním, kontrolou protokolů a aplikační inspekcí,
- kombinování firewallů ve spojení s dalšími bezpečnostními prvky (antivirus, osobní firewally...),
- udržování aktuálních verzí softwaru.

Služby, se kterými bude naše infastruktura pracovat jsou následující:

- SSH (Secure Shell)
- HTTP (Hypertext Transfer Protocol)
- HTTPS (Hypertext Transfer Protocol Secure)
- MongoDB
- NRPE (Nagios Remote Plugin Executor)

10.1 Filtrování na úrovni routeru

První zařízení dostupné z internetu je router. Na samotný router bude umožněn vzdálený přístup z podnikové sítě společnosti. Dále bude umožněno router monitorovat pomocí monitorovacího systému Nagios. Tabulka Tab. *3* znázorňuje vstupní a výstupní provoz na routeru.

Tab. 3 Služby povolené v provozu mezi routerem a internetem

provoz	služby
internet <=> router	SSH, NRPE

Firewall nasazený na úrovni routeru bude zejména směrovat provoz mezi jednotlivými VLAN sítěmi, viz *Vytvoření logického rozdělení sítě pomocí VLAN*. Tato segmentace je pro nás výchozí, protože se filtrování bude odvíjet od směru provozu. Filtrování bude tedy probíhat následovně:

- z internetu do VLAN1;
- z VLAN1 do internetu;
- z internetu do VLAN2;
- z VLAN2 do internetu;
- z VLAN1 do VLAN2;
- z VLAN2 do VLAN1.

Každý provoz bude pracovat s rozdílnými službami a bude tedy mít svá vlastní omezující pravidla Jednotlivé služby v těchto směrech ilustruje tabulka *Tab. 4.*

Tab. 4 Služby v povolené ve směrovaných provozech

provoz	služby
internet <=> vlan1	HTTP, HTTPS, NRPE
internet <=> vlan2	NRPE
vlan1 <=> vlan2	MongoDB

Vše ostatní, co se netýká zmíněných služeb, bude zakázáno.

10.2 Filtrování na úrovni webových serverů

Pro zvýšení zabezpečení budeme firewall taktéž stavět na samotných webových serverech. Opět budeme vycházet z toho, že budou povoleny pouze nutné služby a ostatní provoz bude zakázán. Povolené sužby budou omezeny taktéž na určité adresy, ze kterých se bude možné připojovat.

11 Postup konfigurace v laboratorním prostředí

Dříve, než začneme řešit naši problematiku přímo v produkčním prostředí, vyzkoušíme veškerou konfiguraci v laboratorním prostředí. Díky laboratornímu prostředí tak máme jistotu, že pokud se stane během konfigurace jakákoliv chyba, nemá to dopad na naše produkční prostředí, které je využíváno. Vyhneme se tak možným komplikacím.

11.1 Popis testovacího prostředí

Pro testování využijeme prostředí Proxmox Virtual Environment, což je virtualizační platforma, která je dostupná jako open-source. Proxmox VE slouží jako nástroj pro správu virtuálních serverů. Vybíráme toto prostředí z důvodu jeho vlastností, které spočívají v jednoduché instalaci, přehledné konfiguraci a správě serveru z centrální konzole, která je založena na webovém rozhraní. (Virtualizace Proxmox, 2016) Toto prostředí nainstalujeme na server, do kterého přivedeme lokální firemní síť.

11.2 Vytvoření virtuálních strojů

V první řadě musíme vytvořit jednotlivé virtuální stroje, které budou sloužit pro demonstraci produkčního prostředí. Tyto stroje budou založené na platformě Ubuntu 16.04. Celkem budeme potřebovat 6 strojů, které budou reprezentovat síťovou infrastrukturu. Pro otestování potřebných věcí využijeme dva počítače, které budou připojeny k lokální firemní síti. Díky těmto počítačům budeme moci provádět testování přístupu z internetu. V interní síti budou tři stroje sloužit jako webové servery, jeden stroj jako load balancer, jeden jako databázový server a poslední jako router.

Aby bylo možné směřovat provoz, je nutné povolit samotné směrování. Jinak řečeno, je nutné zajistit, že stroj sloužící jako router bude schopný využívat více síťových karet a bude schopný předávat datagramy z jednoho síťového rozhraní na druhé. Toho dosáhneme tak, že v souboru `/etc/sysctl.conf` smažeme znak sloužící pro uvedení poznámky v textovém editoru na řádku, jenž slouží pro nastavení směrování. Tento řádek vypadá následovně: `#net.ipv4.ip_forward = 1`.

Na virtuálních strojích je dále nutné naistalovat patřičné služby, které náleží jednotlivým strojům. Pro webové servery to bude instalace Apache2, pro load balancer Nginx, pro databázový server MongoDB a pro stroje, které mají být monitorovány pomocí služby Nagios, je nutné naistalovat modul NRPE (Nagios Remote Plugin Executor).

11.3 Nastavení IP adres

V dalším kroku je nutné nastavit veškerá rozhraní tím, že jim přidělíme patřičnou IP adresu. Jelikož se nacházíme v prostředí Proxmox, které je připojené k LAN síti společnosti, získáváme připojení k síti 10.202.60.0/24. Prostředí Proxmox propojuje virtuální stroje se sítí přemostěním díky tzv. VMBR (Virtual Machine Bridge). V počátečním stavu se v prostředí nachází pouze vmbr0 se síťovým rozsahem právě

zmiňované LAN sítě. Abychom mohli oddělit síť LAN, která bude simulovat internet, je nutné vytvořit další mosty. Vytvoříme tedy další dva mosty. Jeden, který bude vytvářet privátní síť 192.168.0.0/24 a bude sloužit pro síť s webovými servery a druhý, s privátním síťovým rozsahem 192.168.1.0/24 pro databázový server.

V první řadě je potřeba propojit jednotlivé sítě tak, že nastavíme na stroji simulující router jednotlivá rozhraní. Pro vnější rozhraní bude sloužit rozhraní eth0 a pro interní sítě eth1 a eth2. Jednotlivé adresy na rozhraních znázorňuje tabulka *Tab. 5*.

Tab. 5 Seznam IP adres na routeru

rozhraní	IP adresa	IP adresa sítě
eth0	10.202.60.231/24	10.202.60.0
eth1	192.168.0.1/24	192.168.0.0
eth2	192.168.1.1/24	192.168.1.0

V dalším kroku je potřeba připojit jednotlivé stroje do jejich patřičných sítí. IP adresy jednotlivých strojů znázorňuje tabulka *Tab. 6*.

Tab. 6 Seznam IP adres jednotlivých strojů

jméno stroje	role	IP adresa	IP adresa sítě
nginx	Nginx Load Balancer	192.168.0.100/24	192.168.0.0
webserver1	Apache Web Server	192.168.0.101/24	192.168.0.0
webserver2	Apache Web Server	192.168.0.102/24	192.168.0.0
webserver3	Apache Web Server	192.168.0.103/24	192.168.0.0
databaze	MongoDB Server	192.168.1.100/24	192.168.1.0

Zbývají pouze testovací počítače, u kterých pro nás není jejich IP adresa podstatná. Tyto adresy jsou automaticky přiděleny pomocí DHCP serveru interní podnikové sítě.

11.4 Konfigurace load balanceru

Nyní se dostáváme k samotné konfiguraci load balanceru. Jak již bylo zmíněno v kapitole 9.1.3, jako řešení pro vyvažování zátěže jsme zvolili Nginx. Jeho instalace je jednoduchá. Na linuxovém stroji zadáme pouze příkaz „`apt-get install nginx`“, který nainstaluje Nginx ze softwarového repozitáře balíčků. Po skončení instalace systém Ubuntu spustí Nginx. Zda služba opravdu běží můžeme zkontrolovat příkazem „`systemctl status nginx.service`“, který zobrazí stav služby.

```
aliacte@nginx:~$ systemctl status nginx.service
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset:
enabled)
   Active: active (running) since Čt 2017-05-11 09:03:13 CEST; 3 days ago
     Docs: man:nginx(8)
  Process: 1186 ExecStart=/usr/sbin/nginx -g daemon on; master_process on;
(code=exited, status=0/SUCCESS)
  Process: 1080 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; mas-
ter_process on; (code=exited, status=0/SUCCESS)
 Main PID: 1188 (nginx)
    Tasks: 5
   Memory: 12.1M
      CPU: 38.197s
   CGroup: /system.slice/nginx.service
           ├─1188 nginx: master process /usr/sbin/nginx -g daemon on; mas-
ter_process on
           ├─1189 nginx: worker process
           ├─1190 nginx: worker process
           ├─1191 nginx: worker process
           └─1192 nginx: worker process

kvě 11 09:03:12 nginx systemd[1]: Starting A high performance web server and a
reverse proxy server...
kvě 11 09:03:13 nginx systemd[1]: Started A high performance web server and a
reverse proxy server.
```

Taktéž můžeme zkontrolovat, zda server odpovídá na HTTP požadavek a to tak, že zadáme jeho IP adresu do webového prohlížeče. Zadáme-li tedy adresu 192.168.0.100, měla by se nám ve webovém prohlížeči zobrazit defaultní úvodní stránka Nginx.

Po samotné instalaci se dostáváme ke konfiguraci Nginx jako load balanceru. Systém Ubuntu se řídí pravidly pro ukládání virtuálních hostitelských souborů v souboru `/etc/nginx/sites-available/`, které jsou povoleny prostřednictvím symbolických odkazů v `/etc/nginx/sites-enabled/`.

Po instalaci se ve složce `/etc/nginx/sites-available/` nachází defaultní konfigurační soubor a ve složce `/etc/nginx/sites-enabled/` defaultní symbolický odkaz na konfigurační soubor.

Při konfiguraci vyvažování zátěže je nutné nastavit, jakým typům připojení bude naslouchat a kam tato připojení bude přesměrovávat. Tento problém řeší v Nginx modul „`ngx_http_upstream_module`“, který se využívá k definování skupin serverů, na než se lze odkazovat.

Pro toto nastavení využijeme některý z dostupných textových editorů Ubuntu a vytvoříme nový konfigurační soubor, jenž umístíme do složky `/etc/nginx/conf.d/`. S použitím textového editoru nano bude příkaz vypadat následovně:

```
nano /etc/angin/conf.d/nginxLB.conf
```

V tomto souboru je v první řadě nutné nakonfigurovat pool IP adres, na které bude load balancer přesměrovávat provoz. K této konfiguraci slouží v modulu následující syntaxe:

```
upstream name { ... }
```

Jméno „upstremu“ zvolíme pro přehlednost backend, protože v poolu se budou nacházet servery postavené právě za tímto zařízením. Naše syntaxe v konfiguračním souboru bude tedy vypadat následovně:

```
upstream backend {
	server 192.168.0.101;
	server 192.168.0.102;
	server 192.168.0.103;
}
```

Jak můžeme vidět, do poolu adres jsme napsali adresy webových serverů Apache. V dalším kroku je nutné do konfiguračního souboru napsat, jakým typům připojení má server naslouchat. Tato konfigurace se provede následující syntaxí:

```
server { ... }
```

Tato část slouží pro konfiguraci virtuálního serveru. Nachází se v ní pokyny pro poslech. Popisuje všechny adresy a porty, které by měly přijímat připojení. Doplnění konfiguračního souboru bude vypadat následovně:

```
server {
	listen 80;

	location / {
		proxy_pass http://backend;
		error_page 500 502 503 504 /50x.html;
	}

	location = /50x.html {
		root /usr/share/nginx/html;
	}
}
```

Vidíme, že na prvním řádku je určeno, s jakým protokolem bude vyvažování pracovat, tedy s protokolem HTTP. Dále si můžeme všimnout lokalizačního bloku, který se používá k rozhodování o tom, jak zpracovat požadavek URI (Uniform Resource Identifier). V tomto bloku se nacházejí dvě směrnice. První směrnice ukazuje pool webových serverů a druhou směrnici využíváme k tomu, abychom určili, co by se mělo stát při výskytu určitých stavových kódů.

Chceme-li dosáhnout taktéž HTTPS provozu, musíme přidat další blok „`server`“, jenž bude tentokrát poslouchat na portu 443. Tato konfigurace může vypadat následovně:

```
server {
        listen 443 ssl;
        ssl_certificate /etc/nginx/ssl/prymum.cz/server.crt;
        ssl_certificate_key /etc/nginx/ssl/prymum.cz/server.key;

        location / {
                proxy_pass http://backend;
                error_page 404 /404.html;
                error_page 500 502 503 504 /50x.html;
        }
```

Můžeme si všimnout, že nám při této konfiguraci přibylo několik málo věcí. Pro zabezpečený provoz jsme museli v první řadě aktivovat SSL (Secure Sockets Layer). Toho jsme dosáhli prvním řádkem v bloku. Následně bylo nutné specifikovat lokaci certifikačního souboru a lokaci souboru, který obsahuje tajný klíč.

Pokud nyní konfigurační soubor uložíme bude využito defaultního algoritmu pro vyvažování zátěže Round Robin. Tato vyvažovací metoda je pro naše řešení dostačující. Vyvažování bude tedy probíhat tak, že se bude postupně cyklicky přesměrovávat na jednotlivé webové servery.

Posledním krokem je zakázání defaultní konfigurace serveru. Defaultní konfiguraci zakážeme tak, že ze složky `/etc/nginx/sites-enabled/` smažeme defaultní symbolický odkaz na původní konfiguraci příkazem:

```
rm /etc/nginx/sites-enabled/default
```

Po provedení všech zmíněných kroků musíme službu restartovat příkazem:

```
service nginx restart
```

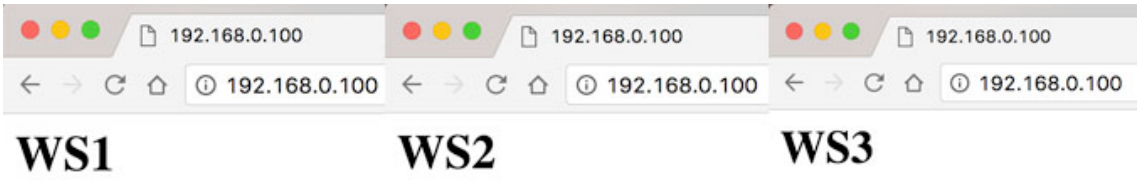
11.4.1 Test vyvažování zátěže

Jak již bylo zmíněno v kapitole 11.2, pro testování použijeme dva počítače, které se nacházejí mimo naši lokální virtuální síť. Otestování bude prováděno následovně:

- zasíláním HTTP požadavků na adresu load balanceru prostřednictvím webového prohlížeče,
- výpisem logu souboru „access.log“,
- vypnutím jednoho serveru.

Abychom viděli i přes webový prohlížeč, jak se vyvažovač zátěže chová, pozměnili jsme na jednotlivých webových serverech jejich soubor „index.html“. Původní soubory by zobrazovaly úvodní stánku Apache, která je na všech strojích stejná a nepoznali bychom tak, zda se vlastně něco děje. V pozměněných souborech se nyní nachází pouze zkrácený název každého stroje.

Po zadání adresy 192.168.0.100 do webového prohlížeče, dostáváme nyní tyto tři různé odpovědi, které můžeme vidět na *Obr. 11*. Vidíme, že dostáváme pokaždé jiný výsledek v podobě zkráceného názvu serverů.



Obr. 11 Test vyvažování zátěže ve webovém prohlížeči

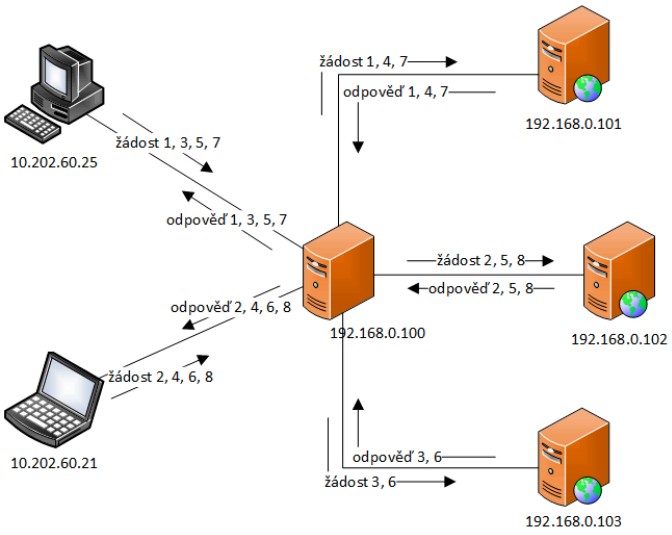
Nyní zkontrolujeme, co se děje, vypsáním logovacího souboru „access.log“. Tento soubor nalezneme v adresáři `/var/log/nginx/`. Strukturu zaznamenávání logu jsme pozměnili, aby byl výpis přehlednější.

Ve výpisu můžeme vidět tyto atributy:

- čas, kdy byl požadavek vyslán,
- adresu stroje, odkud byl požadavek vyslán,
- adresu stroje, na kterou byl požadavek přesměrován,
- dobu odezvy.

```
[14/May/2017:10:42:20 +0200] from: 10.202.60.25 to: 192.168.0.101:80 upstre-
am_response_time 0.002
[14/May/2017:10:42:21 +0200] from: 10.202.60.21 to: 192.168.0.102:80 upstre-
am_response_time 0.002
[14/May/2017:10:42:21 +0200] from: 10.202.60.25 to: 192.168.0.103:80 upstre-
am_response_time 0.002
[14/May/2017:10:42:21 +0200] from: 10.202.60.21 to: 192.168.0.101:80 upstre-
am_response_time 0.001
[14/May/2017:10:42:21 +0200] from: 10.202.60.25 to: 192.168.0.102:80 upstre-
am_response_time 0.001
[14/May/2017:10:42:21 +0200] from: 10.202.60.21 to: 192.168.0.103:80 upstre-
am_response_time 0.001
[14/May/2017:10:42:21 +0200] from: 10.202.60.25 to: 192.168.0.101:80 upstre-
am_response_time 0.002
[14/May/2017:10:42:22 +0200] from: 10.202.60.21 to: 192.168.0.102:80 upstre-
am_response_time 0.001
[14/May/2017:10:42:22 +0200] from: 10.202.60.25 to: 192.168.0.103:80 upstre-
am_response_time 0.001
[14/May/2017:10:42:22 +0200] from: 10.202.60.21 to: 192.168.0.101:80 upstre-
am_response_time 0.001
[14/May/2017:10:42:22 +0200] from: 10.202.60.25 to: 192.168.0.102:80 upstre-
am_response_time 0.001
```

Jak vidíme, požadavky jsou odesílány z adres 10.202.60.25 a 10.202.60.21. Dále vidíme, že požadavky byly z jednotlivých strojů odesílány ve stejných časech, přičemž se jejich cílová adresa měnila na jednotlivé adresy webových serverů. Můžeme si všimnout, že cílová adresa se mění stále ve stejném pořadí. Je tomu tak právě díky algoritmu Round Robin. Znázornění tohoto algoritmu ilustruje *Obr. 12*.



Obr. 12 Round Robin

Pokud by se s jedním ze serverů něco stalo, vyvažování by mělo pokračovat na zbylých serverech. Můžeme to vyzkoušet například tak, že vypneme službu Apache na jednom ze serverů. Pro ukázku vypneme službu například na druhém webovém serveru příkazem „`systemctl stop apache2`“ a vyšleme opět několik požadavků.

```
[14/May/2017:10:47:29 +0200] from: 10.202.60.21 to: 192.168.0.102:80, 192.168.0.103:80
upstream_response_time 0.000, 0.002
[14/May/2017:10:47:30 +0200] from: 10.202.60.25 to: 192.168.0.101:80 upstre-
am_response_time 0.002
[14/May/2017:10:47:30 +0200] from: 10.202.60.21 to: 192.168.0.103:80 upstre-
am_response_time 0.002
[14/May/2017:10:47:31 +0200] from: 10.202.60.25 to: 192.168.0.101:80 upstre-
am_response_time 0.001
[14/May/2017:10:47:31 +0200] from: 10.202.60.21 to: 192.168.0.103:80 upstre-
am_response_time 0.001
[14/May/2017:10:47:31 +0200] from: 10.202.60.25 to: 192.168.0.101:80 upstre-
am_response_time 0.001
```

Vidíme, že další server v pořadí, ke kterému se mělo přistupovat, byl právě druhý webový server. Následující řádky pak znázorňují již jen přesměrování na zbylé dva servery.

11.5 Nastavení firewallové politiky

Jak bylo zmíněno v kapitole 10, filtrování bude probíhat jak na úrovni routeru, tak na úrovni serverů. Firewallová pravidla budou nastavena za pomoci nástroje iptables a budeme se držet pravidla, že všechno, co nebude výslovně povoleno, bude zakázáno.

Ještě před samotným nastavováním pravidel si musíme uvědomit, že je nutné tato pravidla někde ukládat a při restartování stroje je opět načítat. K tomuto účelu využijeme balíčku „*iptables-persistent*“, díky kterému budou pravidla při startu načtena automaticky. Tento balíček naistalujeme z repozitáře pomocí příkazu „`apt-get install iptables-persistent`“.

11.5.1 Nastavení firewallových pravidel na routeru

V našem testovacím prostředí jsme do role routeru postavili virtuální stroj postavený na platformě Ubuntu, kde jsme povolili směrování. Nyní je potřeba nastavit pravidla v jednotlivých směrech, která byla uvedena v kapitole 10.1.

V prvním kroku vytvoříme vlastní řetězy, jež budou sloužit pro seskupení pravidel v jednotlivých směrech. Tyto řetězy budou pojmenovány následovně:

- IntKVlan1, pro provoz směřující z internetu do sítě vlan1,
- Vlan1KInt, pro provoz směřující z vlan1 do internetu,
- Vlan1KVlan2, pro provoz směřující z vlan1 do vlan2,
- Vlan2KVlan1, pro provoz směřující z vlan2 do vlan1,
- IntKVlan2, pro provoz směřující z internetu do vlan2,
- Vlan2KInt, pro provoz směřující z vlan2 do internetu.

Jednotlivé řetězy vytvoříme příkazem „`iptables –N <název řetězu>`“.

```
iptables –N IntKVlan1
iptables –N Vlan1KInt
iptables –N Vlan1KVlan2
iptables –N Vlan2KVlan1
iptables –N IntKVlan2
iptables –N Vlan2KInt
```

V dalším kroku nastavíme výchozí politiku hlavních řetězů.

```
iptables –P INPUT DROP
iptables –P FORWARD DROP
iptables –P OUTPUT DROP
```

Výše uvedené příkazy zakážou veškerý provoz, který by se vztahoval k jednotlivým řetězům. Musíme tedy povolit právě ten provoz, který je pro nás důležitý. Jelikož se nacházíme na routeru, na řetězech INPUT a OUTPUT nastavíme pouze pravidla pro vzdálený přístup a monitoring. Následujícími příkazy nastavíme možnost připojení všech příchozích připojení z místní smyčky, možnost vzdáleného připojení z podnikové sítě společnosti pomocí SSH a přístup pro monitorovací systém, který bude možný pouze ze serveru Nagios.

```
iptables –A INPUT –i lo –j ACCEPT
iptables –A INPUT –m conntrack ––ctstate ESTABLISHED –j ACCEPT
iptables –A INPUT –s 10.202.60.247 –p tcp ––dport 5666 –m conntrack ––ctstate
NEW –j ACCEPT
iptables –A INPUT –s 10.202.60.0/24 –p tcp ––dport 22 –m conntrack ––ctstate
NEW –j ACCEPT
```

V opačném směru, tedy v řetězu OUTPUT, nastavíme opět pravidlo pro místní smyčku. Dále pak pravidlo pro zpětnou komunikaci zmíněných protokolů tak, že povolíme pouze již navázaná spojení.

```
iptables –A OUTPUT –o lo –j ACCEPT
iptables –A OUTPUT –m conntrack ––ctstate ESTABLISHED –j ACCEPT
```

Důležitým řetězem je pro nás v tuto chvíli řetěz FORWARD, ve kterém nastavíme pravidla, jež budou určovat z jakého rozhraní na jaké rozhraní se má směrovat provoz.

```
iptables –A FORWARD –i eth0 –o eth1 –d 192.168.0.0/24 –j IntKVlan1
iptables –A FORWARD –i eth1 –o eth0 –s 192.168.0.0/24 –j Vlan1KInt
iptables –A FORWARD –i eth1 –s 192.168.0.0/24 –o eth2 –d 192.168.1.0/24 –j
Vlan1KVlan2
iptables –A FORWARD –i eth2 –s 192.168.1.0/24 –o eth1 –d 192.168.0.0/24 –j
Vlan2KVlan1
iptables –A FORWARD –i eth0 –o eth2 –d 192.168.1.0/24 –j IntKVlan2
iptables –A FORWARD –i eth2 –o eth0 –s 192.168.1.0/24 –j Vlan2KInt
```

Výše uvedené příkazy určují provoz mezi jednotlivými rozhraními. Můžeme si všimnout, že jako parametr příkazu uvádíme taktéž cílovou nebo zdrojovou IP adresu. Co se následně bude dít s každým z provozů, určuje parametr „*-j*“ (jump). Jak vidíme, na tomto místě přecházíme do vlastních jednotlivých řetězů, kde už každý z nich bude rozhodovat o tom, co projde dál či nikoliv. Jakýkoliv ostatní provoz prostřednictvím řetězu FORWARD bude zahozen.

Posledním krokem je nastavení pravidel v jednotlivých řetězech, které jsme si vytvořili. Začneme provozem z internetu směrem do vlan1.

```
iptables –A IntKVlan1 –m conntrack ––ctstate ESTABLISHED –j ACCEPT
iptables –A IntKVlan1 –p tcp ––dport 80 –d 192.168.0.100 –m conntrack ––
ctstate NEW –j ACCEPT
iptables –A IntKVlan1 –p tcp ––dport 443 –d 192.168.0.100 –m conntrack ––
ctstate NEW –j ACCEPT
iptables –A IntKVlan1 –s 10.202.60.247 –p tcp ––dport 5666 –m conntrack ––
ctstate NEW –j ACCEPT
```

Tato posloupnost příkazů pro nastavení pravidel nám na prvním řádku povoluje pakety, které patří již k založeným komunikacím. Další dva řádky povolují nový provoz HTTP a HTTPS s cílovou adresou, jež náleží load balanceru. Příkaz, který následuje nastaví pravidlo, jenž povoluje serveru Nagios přístup do celé vlan1.

Pravidla pro provoz v opačném směru, tedy z vlan1 do internetu, nastavíme následovně:

```
iptables –A Vlan1KInt –p tcp ––sport 80 –m conntrack ––ctstate ESTABLISHED –j
ACCEPT
```

```
iptables –A Vlan1KInt –p tcp ––sport 443 –m conntrack ––ctstate ESTABLISHED –j
ACCEPT
iptables –A Vlan1KInt –p tcp ––sport 5666 –m conntrack ––ctstate ESTABLISHED –
j ACCEPT
```

Tato pravidla povolují všem využívaným protokolům pouze takový provoz ven, který byl již založen.

Dále nastavíme pravidla pro provoz mezi vlan1 a vlan2.

```
iptables –A Vlan1KVlan2 –m conntrack ––ctstate ESTABLISHED –j ACCEPT
iptables –A Vlan1KVlan2 –p tcp ––dport 27017 –m conntrack ––ctstate NEW –j
ACCEPT

iptables –A Vlan2KVlan1 –p tcp ––sport 27017 –m conntrack ––ctstate ESTA–
BLISHED –j ACCEPT
```

Jak můžeme vidět, komunikace probíhá pouze na portu 27017, který náleží démonu mongod, který je součástí MongoDB databáze. Tato pravidla umožňují webovým serverům komunikovat s databází.

Nyní nám zbývá už jen nastavit pravidla pro komunikaci mezi internetem a vlan2. V těchto směrech bude povolena pouze komunikace typu Nagios.

```
iptables –A IntKVlan2 –m conntrack ––ctstate ESTABLISHED –j ACCEPT
iptables –A IntKVlan2 –s 10.202.60.247 –p tcp –m tcp ––dport 5666 –m conntrack
––ctstate  NEW –j ACCEPT

iptables –A Vlan2KInt –p tcp ––sport 5666 –m conntrack ––ctstate ESTABLISHED –
j ACCEPT
```

První příkaz nastaví pravidlo, které povoluje průchod založených komunikací. Následující pravidlo povoluje přístup nové komunikace ze serveru Nagios. Poslední pravidlo se vztahuje k provozu opačným směrem a povoluje již založený odchozí provoz pro Nagios.

Samotná pravidla je nutné uložit, abychom o tuto konfiguraci při restartování stroje nepřišli. Uložení provedeme příkazem „`/etc/init.d/iptables-persistent save`“ .
V poslední řadě si můžeme pravidla zkontrolovat jejich výpisem, a to příkazem „`iptables –L –line-number`“, který nám zároveň očísluje jednotlivé řádky.

```
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  anywhere             anywhere
2    ACCEPT     all  --  anywhere             anywhere             ctstate ESTABLISHED
3    ACCEPT     tcp  --  10.202.60.247        anywhere             tcp dpt:nrpe ctstate
NEW
4    ACCEPT     tcp  --  10.202.60.0/24       anywhere             tcp dpt:ssh ctstate
NEW

Chain FORWARD (policy DROP)
num  target     prot opt source               destination
1    IntKVlan1  all  --  anywhere             192.168.0.0/24
2    Vlan1KInt  all  --  192.168.0.0/24       anywhere
```

```
3    Vlan1KVlan2  all  --  192.168.0.0/24       192.168.1.0/24
4    Vlan2KVlan1  all  --  192.168.1.0/24       192.168.0.0/24
5    IntKVlan2  all  --  anywhere             192.168.1.0/24
6    Vlan2KInt  all  --  192.168.1.0/24       anywhere

Chain OUTPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  anywhere             anywhere
2    ACCEPT     all  --  anywhere             anywhere             ctstate ESTABLISHED

Chain IntKVlan1 (1 references)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
2    ACCEPT     all  --  anywhere             anywhere             ctstate ESTABLISHED
3    ACCEPT     tcp  --  anywhere             192.168.0.100        tcp dpt:http ctstate
NEW
4    ACCEPT     tcp  --  anywhere             192.168.0.100        tcp dpt:https ctstate
NEW
5    ACCEPT     tcp  --  10.202.60.247        anywhere             tcp dpt:nrpe ctstate
NEW

Chain IntKVlan2 (1 references)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
2    ACCEPT     all  --  anywhere             anywhere             ctstate ESTABLISHED
3    ACCEPT     tcp  --  10.202.60.247        anywhere             tcp dpt:nrpe ctstate
NEW

Chain Vlan1KInt (1 references)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  anywhere             anywhere             tcp spt:ssh
2    ACCEPT     tcp  --  anywhere             anywhere             tcp spt:http ctstate
ESTABLISHED
3    ACCEPT     tcp  --  anywhere             anywhere             tcp spt:https ctstate
ESTABLISHED
4    ACCEPT     tcp  --  anywhere             anywhere             tcp spt:nrpe ctstate
ESTABLISHED

Chain Vlan1KVlan2 (1 references)
num  target     prot opt source               destination
1    ACCEPT     all  --  anywhere             anywhere             ctstate ESTABLISHED
2    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:27017 ctstate
NEW

Chain Vlan2KInt (1 references)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  anywhere             anywhere             tcp spt:ssh
2    ACCEPT     tcp  --  anywhere             anywhere             tcp spt:nrpe ctstate
ESTABLISHED

Chain Vlan2KVlan1 (1 references)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  anywhere             anywhere             tcp spt:27017 ctstate
ESTABLISHED
```

V přílohách této bakalářské práce nalezneme shallový skript, díky kterému můžeme tato pravidla v testovacím prostředí nasadit na router postavený na platformě Ubuntu server. Tento skript je přiložen zejména z důvodu podrobnějšího popisu jednotlivých pravidel.

11.5.2 Nastavení firewallových pravidel na webových serverech

Jednotlivé webové servery se od sebe nijak neliší. Vyplývá tedy, že sada pravidel na jednotlivých serverech bude zcela stejná. Opět budeme vycházet z pravidla, že všechno, co nebude výslovně povoleno, bude zakázáno. Dle tohoto pravidla nastavíme veškerou výchozí politiku na DROP.

```
iptables –P INPUT DROP
iptables –P FORWARD DROP
iptables –P OUTPUT DROP
```

Řetěz FORWARD nás v další části zajímat nebude, protože zde již nic nesměrujeme. Dále tedy využíváme řetěz INPUT a řetěz OUTPUT. Příkazy pro konfiguraci řetězu INPUT jsou následující:

```
iptables –A INPUT –i lo –j ACCEPT
iptables –A INPUT –m conntrack ––ctstate ESTABLISHED –j ACCEPT
iptables –A INPUT –p tcp –m multiport ––dports 80,443 –s 192.168.0.100 –m
conntrack ––ctstate NEW –j ACCEPT
iptables –A INPUT –p tcp ––dport 5666 –s 10.202.60.247 –m conntrack ––ctstate
NEW –j ACCEPT
```

Prvním řádkem nastavíme pravidlo, které povolí všechna příchozí spojení z místní smyčky. Další příkazy povolují již založená spojení, nová spojení HTTP A HTTPS ze zdrojové adresy load balanceru a přístup monitorovacímu systému.

Následující posloupnost příkazů povoluje komunikaci v opačném směru. Navíc se zde objevuje pravidlo, které umožňuje webovému serveru přistupovat k databázi MongoDB.

```
iptables –A OUTPUT –o lo –j ACCEPT
iptables –A OUTPUT –p tcp –m multiport ––sports 80,443 –m conntrack ––ctstate
ESTABLISHED –j ACCEPT
iptables –A OUTPUT –p tcp ––dport 27017 –m conntrack ––ctstate NEW,ESTABLISHED
–j ACCEPT
iptables –A OUTPUT –p tcp ––sport 5666 –m conntrack ––ctstate ESTABLISHED –j
ACCEPT
```

Opět je nutné uložit pravidla na jednotlivých webových serverech. Výpis pravidel na webovém serveru vypadá následovně:

```
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  anywhere             anywhere
2    ACCEPT     all  --  anywhere             anywhere             ctstate ESTABLISHED
3    ACCEPT     tcp  --  192.168.0.100        anywhere             multiport dports
http,https ctstate NEW
4    ACCEPT     tcp  --  10.202.60.247        anywhere             tcp dpt:nrpe ctstate
NEW
```

```
Chain FORWARD (policy DROP)
num  target     prot opt source               destination

Chain OUTPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  anywhere             anywhere
1    ACCEPT     tcp  --  anywhere             anywhere             multiport sports
http,https ctstate ESTABLISHED
4    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:27017 ctstate
NEW,ESTABLISHED
```

Všechna pravidla byla stavěna tak, abychom v případě přidání dalšího webového serveru za load balancer mohli využít pouze šablonu, která tato pravidla na novém uzlu nastaví. Tato šablona je vytvořena za pomoci shallového skriptu, který obsahuje sérii příkazů. Mimo samotné příkazy nalezneme v šabloně taktéž komentáře, popisující jednotlivá pravidla. Tuto šablonu nalezneme v přílohách této práce.

11.5.3 Test firewallových pravidel

Testování toho, zda jsou pravidla správně nastavena provede za pomoci nástroje Nmap, který slouží pro skenování portů.

Test povolených portů na routeru

V první řadě otestujeme router, na kterém jsme povolili vzdálený přístup a monitoring. Test bude proveden díky skenování zevnitř a zvenčí. Tyto skeny budou dodávat odlišné výsledky, protože pravidlo pro monitorování bylo nastaveno tak, aby se na router dalo přistupovat pouze z konkrétní adresy, a to adresy serveru Nagios. V prvním kroku tedy otestujeme náš stroj zevnitř pomocí příkazu „`nmap localhost`“. Výsledek tohoto testu vypadá následovně:

```
root@ROUTER:/home/aliacte# nmap localhost

Starting Nmap 6.40 ( http://nmap.org ) at 2017-05-19 17:45 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000011s latency).
Other addresses for localhost (not scanned): 127.0.0.1
Not shown: 998 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
5666/tcp open  nrpe

Nmap done: 1 IP address (1 host up) scanned in 1.61 seconds
```

Z testu vidíme, že porty, které jsou povolené (open), jsou právě ty, které jsme definovali v pravidlech. Dále si na pátém řádku můžeme všimnout, že ostatní porty jsou uzavřené (closed).

Nyní provedeme test našeho routeru zvenčí. V roli naší venkovní sítě je právě síť, která byla specifikována v pravidlu pro vzdálený přístup. Vyplívá z toho tedy, že bychom měli opět vidět otevřený port 22 (SSH). Což bychom však z testovacího stroje vidět neměli, je port 5666 (NRPE), protože jsme v pravidlu specifikovali kon-

krétní adresu monitorovacího serveru. Test z testovacího stroje provedeme příkazem „nmap 10.202.60.231“.

```
root@test1:/home/aliacte# nmap 10.202.60.231

Starting Nmap 7.40 ( https://nmap.org ) at 2017-05-19 18:05 CEST
Nmap scan report for 10.202.60.231
Host is up (0.00053s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 9A:0A:60:A9:1C:35 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 4.62 seconds
```

Výsledek našeho testu vyšel podle očekávání. Dostupná je pouze služba SSH.

Pro otestování toho, zda je pravidlo pro monitoring správné můžeme zkusit test přímo ze serveru Nagios. Jelikož se tento server nachází ve stejné síti, která je zároveň povolena pro vzdálený přístup, měli bychom vidět otevřené porty obou služeb.

```
root@nagiosAli:/home/aliacte# nmap 10.202.60.231

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2017-05-19 15:59 CEST
Nmap scan report for 10.202.60.231
Host is up (0.00087s latency).
Not shown: 998 filtered ports
PORT     STATE SERVICE
22/tcp   open  ssh
5666/tcp open  nrpe
MAC Address: 9A:0A:60:A9:1C:35 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 19.33 seconds
```

Vidíme, že poslední sken zvenčí vyšel taktéž podle našich očekávání. Znamená to, tedy, že tato pravidla jsou nastavena správně.

Test povolených portů v jednotlivých směrech

Nyní se dostáváme k části, kdy je potřeba otestovat, zda pravidla nastavená pro řetěz FORWARD, byla také správná.

V síti VLAN1 provedeme sken dostupných portů na load balanceru. V prvním kroku opět provedeme sken zevnitř. Výsledek tohoto skenu vypadá následovně:

```
root@nginx:/home/aliacte# nmap localhost

Starting Nmap 7.40 ( https://nmap.org ) at 2017-05-19 18:33 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000060s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 997 closed ports
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
5666/tcp open  nrpe

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
```

Ve směru z internetu do VLAN1 byly povoleny služby HTTP, HTTPS a pro monitoring služba Nagios. Jak můžeme z testu vidět, jsou otevřené pouze tyto porty. Test je tedy na místní smyčce úspěšný.

Jako další provedeme opět test z testovacího stroje. Výsledek testu:

```
root@test1:/home/aliacte# nmap 192.168.0.100

Starting Nmap 7.40 ( https://nmap.org ) at 2017-05-19 18:35 CEST
Nmap scan report for 192.168.0.100
Host is up (0.00080s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 4.10 seconds
```

Tento sken opět vyhodnocujeme jako úspěšný, protože jediný provoz, který by měl být povolen z jakéhokoliv nedefinovaného stroje v internetu do sítě VLAN1 je právě provoz HTTP a HTTPS.

Poslední služba, která by měla být přístupná zvenčí je služba Nagios, pro kterou je opět v pravidlech definován přístup, který je omezen pouze na adresu Nagios serveru. Sken portů ze serveru Nagios mířený do naší sítě vypadá následovně:

```
root@nagiosAli:/home/aliacte# nmap 192.168.0.100

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2017-05-19 18:36 CEST
Nmap scan report for 192.168.0.100
Host is up (0.0021s latency).
Not shown: 997 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
5666/tcp open  nrpe

Nmap done: 1 IP address (1 host up) scanned in 16.72 seconds
```

Můžeme si všimnou, že mimo port 5666 (NRPE), jsou dostupné taktéž porty služeb HTTP a HTTPS. To je zcela v pořádku. Jak bylo zmíněno u předcházejícího testu, přístup zvenčí do VLAN1 je u služeb HTTP a HTTPS povolen všem strojům.

Skeny portů z testovacího stroje a ze serveru Nagios do VLAN2 vypadají následovně:

```
aliacte@test1:~$ nmap 192.168.1.196

Starting Nmap 7.40 ( https://nmap.org ) at 2017-05-19 22:26 CEST
Nmap scan report for 192.168.1.196
Host is up.
All 1000 scanned ports on 192.168.1.196 are filtered

Nmap done: 1 IP address (1 host up) scanned in 201.73 seconds

root@nagiosAli:/home/aliacte# nmap 192.168.1.101
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2017-05-19 22:22 CEST
Nmap scan report for 192.168.1.101
Host is up (0.0030s latency).
Not shown: 999 filtered ports
PORT     STATE SERVICE
5666/tcp open  nrpe

Nmap done: 1 IP address (1 host up) scanned in 13.79 seconds
```

Opět vidíme, že jsou pravidla nastavena správně, protože jediný přístup, který je do VLAN2 povolen z internetu, je přístup ze serveru Nagios.

Test povolených portů na webových serverech

Posledním skupina testů je věnována testování pravidel na webových serverech. Opět jako první test provedeme test zevnitř, tedy na lokální smyčce některého z webových serverů.

```
root@webserver2:/home/aliacte# nmap localhost

Starting Nmap 7.01 ( https://nmap.org ) at 2017-05-19 18:40 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000011s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 996 closed ports
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
5666/tcp open  nrpe

Nmap done: 1 IP address (1 host up) scanned in 1.63 seconds
```

První z testů vyšel tak, jak měl. Vidíme, že jsou dostupné pouze služby, které jsme nastavili pravidly a ostatní služby jsou nedostupné.

Další testy provádíme zvenčí prostřednictvím testovacího stroje a serveru Nagios.

```
root@test1:/home/aliacte# nmap 192.168.0.102

Starting Nmap 7.40 ( https://nmap.org ) at 2017-05-19 18:47 CEST
Nmap scan report for 192.168.0.102
Host is up.
All 1000 scanned ports on 192.168.0.102 are filtered

Nmap done: 1 IP address (1 host up) scanned in 217.77 seconds
root@nagiosAli:/home/aliacte# nmap 192.168.0.102

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2017-05-19 18:48 CEST
Nmap scan report for 192.168.0.102
Host is up (0.0025s latency).
Not shown: 999 filtered ports
PORT     STATE SERVICE
5666/tcp open  nrpe

Nmap done: 1 IP address (1 host up) scanned in 150.29 seconds
```

Opět vidíme, že se testy zdařily, protože na webové servery má podle stanovených pravidel přístup pouze server Nagios.

Jako poslední otestujeme, zda je databáze dostupná z našich serverů. Nástroj nmap v základu skenuje pouze tisíc nejpoužívanějších portů. Pokud bychom chtěli otestovat veškeré otevřené porty typu TCP, zabralo by to mnoho času. Test můžeme však provést zadáním čísla portu, jehož stav chceme na dané adrese zjistit. Zadáme-li příkaz „`nmap -p 27017 192.168.1.101`“ zjistíme, zda máme přístup k databázi.

```
root@webserver2:/home/aliacte# nmap -p 27017 192.168.1.101

Starting Nmap 7.01 ( https://nmap.org ) at 2017-05-20 10:51 CEST
Nmap scan report for 192.168.1.101
Host is up (0.0013s latency).
PORT      STATE SERVICE
27017/tcp open  mongod

Nmap done: 1 IP address (1 host up) scanned in 13.28 seconds
```

I poslední z našich testů vyhodnocujeme jako úspěšný.

12 Ekonomické zhodnocení

Jak víme, síťová infrastruktura společnosti se nachází v prostředí cloudu. V tomto prostředí se náklady společnosti za dané řešení odvíjí od pronájmu výkonu. Cena za pronájem výkonu je stanovena včetně synchronní zálohované konektivity. Další náklad vzniká v podobě implementace řešení vyvažování zátěže a jejo správy. V této práci jsme firewall stavěli přímo na webových serverech pomocí nástroje iptables. Pokud bychom však stavěli firewall jako samostatné zařízení, vznikal by taktéž náklad v podobě pronájmu výkonu. Pořizovací náklady load balanceru a firewallu jsou nulové, jelikož jsme se soustředili na řešení v podobě open-source.

Během porovnávání komerčních a open-source řešení jednotlivých zařízení jsme se setkali taktéž s dedikovaným hardwarem. Představme si situaci, kdy by naše společnost nevyužívala cloud jako řešení síťové infrastruktury. V tom případě by mohla využívat například pronájem rackového prostoru, kde by se její zařízení nacházela. Pokud by se společnost rozhodla pořizovat potřebná zařízení v hardwarové podobě, je potřeba si uvědomit, že jejich pořizovací cena není jediným nákladem. Další náklady vznikají za pronájem racku, implementaci, spotřebu, správu zařízení a za konektivitu.

Díky tomu, že víme, jaké náklady v každém z řešení síťové infrastruktury vznikají, můžeme nyní provést vyčíslení celkových nákladů pro naši společnost. Budeme tak schopni porovnat celkové jednorázové náklady a celkové měsíční náklady. Většina potřebných informací k tomuto porovnání byla poskytnuta společností. Zbývající informace byly dohledány.

Abychom viděli rozdíl v nákladech i u firewallu, budeme dále uvažovat, že v prostředí cloudu stavíme firewall jako samostatné zařízení.

Pro lepší orientaci zavedeme následující jednotky:

- 1MD– ohodnocení pracovního dne společností =6000 Kč/den,
- 1U (Rack Unit) – cena za 1,75 palce v racku = 350 Kč/měsíc,
- 1E – jednotková cena za elektřinu + chlazení =4,3 Kč,
- 1M – průměrný počet hodin v měsíci =720 h/měsíc,
- 1K – cena konektivity za symetrický 100 Mbp v režimu vysoké dostupnosti (zálohová linka) = 2000 Kč/měsíc.

12.1 Cloud

Nyní si uvedeme, jaké náklady by pro společnost vznikly v prostředí cloudu za jednotlivá zařízení, která jsou dostupná jako open-source. V tabulce *Tab. 7* můžeme vidět jednorázový náklad na implementaci load balanceru a měsíční náklady v podobě pronájmu výkonu a správy zařízení.

Tab. 7 Náklady na load balancer v prostředí cloudu

	množství	cena	popis
pronájem výkonu	1	9 759,00 Kč	měsíční náklad za pronájem 9 GHz CPU a 32 GB ram

implementace	2*MD	12 000,00 Kč	jednorázový náklad
správa	0,25*MD	1 500,00 Kč	měsíční náklad za správu

V tabulce *Tab. 8* vidíme opět jednotlivé náklady, tentokrát však na pořízení firewallu.

Tab. 8 Náklady na firewal v prostředí cloudu

	množství	cena	popis
pronájem výkonu	1	5 172,00 Kč	měsíční náklad za pronájem 2,6 GHz CPU a 16 GB ram
implementace	2*MD	12 000,00 Kč	jednorázový náklad
správa	0,25*MD	1 500,00 Kč	měsíční náklad za správu

Poslední tabulka *Tab. 9* znázorňuje celkové náklady na obě zařízení v prostředí cloudu.

Tab. 9 Celkové náklady v prostředí cloudu

	celkem
jednorázové náklady	24 000,00 Kč
měsíční náklady	17 931,00 Kč

12.2 Housing

Pokud by společnost řešila infrastrukturu takovým způsobem, že by svá zařízení situovala do prostředí datového centra, hovoří se o pojmu „housing“. Měsíční náklady by se projevovaly formou pronájmu racku, konektivity a spotřebou energie.

Dále uvažujeme, že by společnost zakoupila BIG-IP load balancer a Cisco ASA firewall, tedy právě to řešení, které by vyhovovalo požadavkům společnosti. K zajištění vysoké dostupnosti musíme uvažovat vždy pořízení dvou kusů jednotlivých zařízení. V prostředí cloudu je vysoká dostupnost již zajištěná strukturou, proto jsme u cloudu počítali jednotlivá zařízení pouze jedenkrát. V tabulce *Tab. 10* můžeme vidět jednotlivé náklady, které by společnosti vznikaly nákupem load balancer v podobě dedikovaného hardwaru.

Tab. 10 Náklady na load balancer v prostředí datového centra

	množství	cena	popis
F5 BIG-IP 2000s	2	968 000,00 Kč	jednorázový náklad za pořízení
implementace	2	200 000,00 Kč	jednorázový náklad za implementaci
RACK	6*1U	2 100,00 Kč	měsíční náklad v podobě pronájmu RACKu
spotřeba	3*1E*1M	9 288,00 Kč	měsíční náklad v podobě ceny za elektřinu
správa	3*MD1	18 000,00 Kč	měsíční náklad za správu
konektivita	1K	2 000,00 Kč	měsíční náklad v podobě ceny danou konektivitu

V tabulce *Tab. 11* pak můžeme vidět jednotlivé náklady, které by vznikaly nákupem fyzického firewallu.

Tab. 11 Náklady na firewall v prostředí datového centra

	množství	cena	popis
Cisco ASA 5510	2	34 000 Kč	jednorázový náklad za pořízení
implementace	2	20 000	jednorázový náklad za implementaci
RACK	6*1U	2 100,00 Kč	měsíční náklad v podobě pronájmu RACKu
spotřeba	1,4*1E*1M	4 334,40 Kč	měsíční náklad v podobě ceny za elektřinu
správa	0,75*MD1	4 500,00 Kč	měsíční náklad za správu
konektivita	1K	2 000,00 Kč	měsíční náklad v podobě ceny danou konektivitu

Poslední tabulka *Tab. 12* zachycuje opět celkové náklady, které by společnosti vznikly, tentokrát však v prostředí datového centra.

Tab. 12 Celkové náklady v prostředí datového centra

	celkem
jednorázové náklady	1 222 000,00 Kč
měsíční náklady	44 322,40 Kč

12.3 Zhodnocení

Z výše uvedených tabulek si můžeme všimnout, kolik společnost ušetří tím, že své řešení staví do prostředí cloudu a tím, že jsme vybrali jednotlivé řešení v podobě open-source. V našem řešení jsme firewall stavěli přímo na linuxových strojích. To znamená, že při našem řešení by byly náklady ještě menší, protože nepotřebujeme navíc žádný další pronájem výkonu pro firewall. V této kapitole jsme uváděli pronájem výkonu pro firewall pouze proto, abychom viděli, jaký cenový rozdíl by v takové situaci nastal.

Jak již bylo zmíněno dříve, open-source řešení jsou pro společnost momentálně dostačující a s přehledem zvládnou to, co společnost vyžaduje.

13 Závěr

Cílem bakalářské práce bylo navrhnou load balancing a firewallovou politiku pro společnost, která chce poskytovat veřejně dostupnou službu a svou síťovou infrastrukturu staví v prostředí cloudu.

Nejprve jsme se detailně seznámili s problematikou load balancingu, přičemž jsme si taktéž uvedli výčet algoritmů pro rozklad zátěže. Následně jsme se seznámili s řešeními, která jsou na trhu dostupná. Z dostupných řešení byl uskutečněn výběr jednotlivých zařízení podle stanovených požadavků a tento výběr byl porovnán.

V dalším kroku této práce jsme se stejně tak seznámili s problematikou filtrování provozu, kde jsme uvedli, jaké typu firewallů můžeme užívat. Taktéž jako tomu bylo u load balancerů, i tady bylo provedeno seznámení s dostupnými řešeními na trhu, která byla následně porovnána.

Po detailním seznámení s původní infrastrukturou, která za firewallem skrývala pouze jeden webový server, na kterém rovněž běžela i databáze, bylo navrhnuto řešení, které bude zaručovat jak dostupnost služby, tak její zabezpečení. Výsledné řešení je stavěno na open-source platformách. Pro vyvažování zátěže byla vybrána platforma Nginx a pro filtrování provozu nástroj iptables, který je součástí linuxových distribucí. Tento výběr splňuje požadavky a je zcela dostačující pro naši infrastrukturu. Navrhnuté řešení upravuje síťovou infrastrukturu v podobě zvýšení počtu serverů, konkrétně z jednoho serveru na celkové tři, což je minimální počet pro zajištění vysoké dostupnosti. Dále pak nastala úprava přidáním load balanceru před tyto servery a oddělením databáze od webového serveru. Zvýšení zabezpečení v této úpravě dále proběhlo návrhem VLAN sítí a postavením firewallů přímo na webových serverech. Pro tuto skutečnost bylo nutné definovat služby, které mají mít povolený přístup na jednotlivá zařízení. Tato definice služeb proběhla s administrátorem sítě.

Pro test našeho návrhu jsme vytvořili simulaci každého potřebného zařízení ve virtuálním prostředí, kde bylo nutné naimplementovat patřičné služby. Dále pak bylo nutné nakonfigurovat vyvažování zátěže k webovým serverům, kde bylo využito základního algoritmu vyvažování zátěže. Vybraný algoritmus je momentálně dostačující a v případě nějakých změn není na zvolené platformě nijak náročné nakonfigurovat vyhovující algoritmus. Po tom, co bylo zprovozněno vyvažování zátěže, bylo nutné nakonfigurovat firewally. Pro simulaci routeru, který se nachází v prostředí cloudu a je součástí tohoto prostředí, jsme zvolili další stroj, který nám v testovacím prostředí odděloval jednotlivé sítě. Na tomto stroji jsme stavěli firewall v podobě open-source řešení nástroje iptables. Pravidla byla tedy nastavena na routeru a na jednotlivých webových serverech. Výstupem konfigurace pravidel na webových serverech je taktéž šablona, která je použitelná v případě přidání dalšího webového serveru za load balancer.

Po celkové konfiguraci jsme podrobili upravenou infrastrukturu v laboratorním prostředí jednotlivým testům, jejichž výsledky byly pozitivní. Můžeme tedy výsledné řešení nasadit i mimo laboratorní prostředí.

14 Literatura

BALEJ, Jiří. Bezpečnostní technologie. 2016.

Barracuda: Models [online]. 2017 [cit. 2017-05-18]. Dostupné z: https://www.barracuda.com/products/loadbalancer/models

BARTOŠ, Miroslav. Analýza a optimalizace softwarových firewall na operačních systémech Linux [online]. Hradec Králové, 2016 [cit. 2017-04-27]. Dostupné z: <http://theses.cz/id/yryss6/>. Diplomová práce. Univerzita Hradec Králové, Fakulta informatiky a managementu. Vedoucí práce Mgr. Josef Horálek, Ph.D..

BOURKE, Tony. Server Load Balancing [online]. 1. O'Reilly Media, 2001 [cit. 2017-04-06]. ISBN 0-596-00050-2. Dostupné z: http://index-of.es/Networking/Server%20Load%20Balancing.pdf

Cisco: Cisco ASA 5500 Series Adaptive Security Appliances [online]. 2016 [cit. 2017-05-18]. Dostupné z: http://www.cisco.com/c/en/us/products/security/asa-5500-series-next-generation-firewalls/data_sheet_c78-345385.pdf

Citrix: Choosing the right platform and edition options [online]. 2017 [cit. 2017-05-18]. Dostupné z: https://www.citrix.cz/products/netscaler-adc/platforms.html

Citrix: What is load balancing? [online]. 2017 [cit. 2017-04-06]. Dostupné z: https://www.citrix.cz/glossary/load-balancing.html

DOČEKAL, Michal. LinuxExpres: Správa linuxového serveru: Linuxový firewall, základy iptables II. : 2010[online]. [cit. 2017-04-06]. Dostupné z: https://www.quora.com/What-is-the-difference-between-layer-3-and-layer-4-load-balancing-Why-is-layer-7-LB-used-inspite-of-its-drawbacks-of-being-a-bottleneck

DUC, Václav a David LESNÍK. Stavový firewall na linuxu [online]. 2012 [cit. 2017-05-19]. Dostupné z: http://wh.cs.vsb.cz/sps/images/3/34/Stavovy_firewall_na_linuxu.pdf

Firewall [online]. 2017 [cit. 2017-04-06]. Dostupné z: https://cs.wikipedia.org/wiki/Firewall

F5 [online]. 2017 [cit. 2017-05-18]. Dostupné z: http://f5.com/

F5: BIG-IP System [online]. 2017 [cit. 2017-05-18]. Dostupné z: https://www.f5.com/pdf/products/big-ip-platforms-datasheet.pdf

HAProxy [online]. 2017 [cit. 2017-04-06]. Dostupné z: http://www.haproxy.org

Juniper Networks: SRX Series Services Gateways [online]. 2017 [cit. 2017-05-18]. Dostupné z: http://www.juniper.net/us/en/products-services/security/srx-series/

KAMENÍČKOVÁ, Petra. Implementace externích autentizačních modulů pro Nginx [online]. Brno, 2015 [cit. 2017-05-19]. Bakalářská práce. Vysoké učení technické v Brně. Vedoucí práce RNDr. Marek Rychlý Ph.D.

KUTÁČ, Pavel. Jednoduchý proxy server s nginx [online]. 2016 [cit. 2017-05-18]. Dostupné z: http://www.kutac.cz/blog/weby-a-vse-okolo/jednoduchy-proxy-server-s-nginx/

LANGMAID, Martin. Quora: What is the difference between layer-3 and layer-4 load balancing? Why is layer-7 LB used inspite of its drawbacks of being a bottleneck? : 2016 [online]. [cit. 2017-04-06]. Dostupné z: https://www.quora.com/What-is-the-difference-between-layer-3-and-layer-4-load-balancing-Why-is-layer-7-LB-used-inspite-of-its-drawbacks-of-being-a-bottleneck

Load Balancer FAQ: Why use a Hardware Load Balancer? [online]. 2015 [cit. 2017-05-18]. Dostupné z: http://www.hardwareloadbalancer.com/

Lynt: Cisco ASA [online]. 2013 [cit. 2017-05-18]. Dostupné z: https://lynt.cz/blog/predstaveni-cisco-asa

MILANOV, Petr. IP filtr a detektor útoků [online]. 2006 [cit. 2017-05-19]. Dostupné z: https://dip.felk.cvut.cz/browse/pdfcache/milanp1_2006bach.pdf. Bakalářská práce. České vysoké učení technické v Praze. Vedoucí práce Doc. Ing. Jan Janeček, CSc.

MongoDB. Wikipedie [online]. 2016 [cit. 2017-05-19]. Dostupné z: https://cs.wikipedia.org/wiki/MongoDB

NAVRÁTIL, Martin. Webhostingový cluster postavený na platformě ARM [online]. Zlín, 2015 [cit. 2017-04-28]. Dostupné z: <http://theses.cz/id/zx5hwt/>. Diplomová práce. Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky. Vedoucí práce Ing. David Malaník, Ph.D..

Netfilter [online]. 2014 [cit. 2017-04-06]. Dostupné z: http://www.netfilter.org

Nginx [online]. 2017 [cit. 2017-05-18]. Dostupné z: https://www.nginx.com

NIDL, Michal. Metodika optimálního využití load balancingu v prostředí datového centra [online]. Praha, 2015 [cit. 2017-04-27]. Dostupné z: <http://theses.cz/id/qqc561/>. Diplomová práce. Vysoká škola ekonomická v Praze. Vedoucí práce Libor Gála.

PETRÁK, Martin. Softwarový firewall pro filtrování na síťové a linkové vrstvě [online]. Plzeň, 2013 [cit. 2017-04-27]. Dostupné z: <http://theses.cz/id/02r6uz/>. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd. Vedoucí práce Ing. Tomáš Chmelař.

ROOT: Vše o iptables: úvod [online]. 2006 [cit. 2017-04-06]. Dostupné z: https://www.root.cz/clanky/vse-o-iptables-uvod/

ROSS, Keith W., KUROSE, James F. Computer Networking: A Top-Down Approach [online]. 6th Edition. 2012 [cit. 2017-04-06]. ISBN 0132856204. ISBN-13: 9780132856201. Dostupné z: http://www.bau.edu.jo/UserPortal/UserProfile/PostsAttach/10617_1870_1.pdf

Software Load Balancer: Advantages of a Software Load Balancer [online]. 2015 [cit. 2017-05-18]. Dostupné z: http://www.hardwareloadbalancer.com/software-load-balancer/

The University of Tennessee, Knoxville: What load balancing methods are available? [online]. 2015 [cit. 2017-05-18]. Dostupné z: https://help.utk.edu/kb/index2.php?func=show&e=1699.

TRNKA, Miroslav. Clustering a load balancing serveru pro zpracování řeči [online]. Brno, 2017 [cit. 2017-04-27]. Dostupné z: <http://theses.cz/id/td64j7/>. Diplomová práce. Mendelova univerzita v Brně, Provozně ekonomická fakulta. Vedoucí práce Ing. Jan Přichystal, Ph.D..

VCloud Director: Building a vApp [online]. VMware Education & Certification, 2012 [cit. 2017-05-18]. Dostupné z: https://www.youtube.com/watch?v=jt4y6XZnaxE

Virtualizace Proxmox – instalace a konfigurace. Nápovědy [online]. 2016 [cit. 2017-05-19]. Dostupné z: http://napovedy.cz/post/315

VYHNÁNEK, Jan. Generátor základních filtrovacích pravidel pro konfiguraci firewallů na síťových zařízeních [online]. Brno, 2013 [cit. 2017-04-27]. Dostupné z: <http://theses.cz/id/gv5gtx/>. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce RNDr. Martin Jakubička.

WATKINS, Michael a Kevin WALLACE. CCNA security official exam certification guide. Indianapolis, IN: Cisco Press, c2008. ISBN 15-872-0220-4.

15 Seznam obrázků

16 Seznam tabulek

Přílohy

A Skript pro zavedení firewallových pravidel na routeru

```
#!/bin/bash
#odstranění veškerých pravidel a následné smazání veškerých nezabudovaných řetězů
iptables -F
iptables -X


#######################################################
#nastavení výchozí politiky jednotlivých řetězců
#######################################################
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP


#######################################################
#vytvoření vlastních řetězů
#######################################################
#IntKVlan1, pro stanovení pravidel ve směru komunikace internet k VLAN1
iptables -N IntKVlan1
#Vlan1KInt, pro stanovení pravidel ve směru komunikace VLAN1 k internetu
iptables -N Vlan1KInt
#Vlan1KVlan2, pro stanovení pravidel ve směru komunikace VLAN1 k VLAN2
iptables -N Vlan1KVlan2
#Vlan2KVlan1, pro stanovení pravidel ve směru komunikace VLAN2 k VLAN1
iptables -N Vlan2KVlan1
#IntKVlan1, pro stanovení pravidel ve směru komunikace internet k VLAN2
iptables -N IntKVlan2
#Vlan1KInt, pro stanovení pravidel ve směru komunikace VLAN2 k internetu
iptables -N Vlan2KInt


#######################################################
#INPUT pravidla
#######################################################
#povolení všech příchozích spojení z místní smyčky (local loopback)
iptables -A INPUT -i lo -j ACCEPT
#povolení všech příchozích spojení, která náleží již vytvořeným spojením
iptables -A INPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
#povolení nového tcp spojení pocházejícího ze serveru Nagios
iptables -A INPUT -s 10.202.60.247 -p tcp --dport 5666 -m conntrack --ctstate  NEW -j ACCEPT
#povolení nového tcp spojení SSH pocházejícího z podnikové sítě
iptables -A INPUT -s 10.202.60.0/24 -p tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT


#######################################################
#OUTPUT pravidla
#######################################################
#povolení všech odchozích spojení z místní smyčky (local loopback)
iptables -A OUTPUT -o lo -j ACCEPT
#povolení všech odchozích spojení, která náleží již vytvořeným spojením
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT


#######################################################
#FORWARD pravidla
#######################################################
#předání příchozího spojení na rozhraní eth0 rozhraní eth1, s cílovou adresou sítě VLAN1, řetězci IntKVlan1
```

```
iptables -A FORWARD -i eth0 -o eth1 -d 192.168.0.0/24 -j IntKVlan1
#předání příchozího spojení na rozhraní eth1 rozhraní eth0, se zdrojovu adresou sítě VLAN1, řetězci Vlan1KInt
iptables -A FORWARD -i eth1 -o eth0 -s 192.168.0.0/24 -j Vlan1KInt
#předání příchozího spojení na rozhraní eth1, o zdrojové adrese sítě VLAN1, rozhraní eth2, s cílovou adresou sítě VLAN2, řetězci Vlan1KVlan2
iptables -A FORWARD -i eth1 -s 192.168.0.0/24 -o eth2 -d 192.168.1.0/24 -j Vlan1KVlan2
#předání příchozího spojení na rozhraní eth2, o zdrojové adrese sítě VLAN2, rozhraní eth1, s cílovou adresou sítě VLAN1, řetězci Vlan2KVlan1
iptables -A FORWARD -i eth2 -s 192.168.1.0/24 -o eth1 -d 192.168.0.0/24 -j Vlan2KVlan1
#předání příchozího spojení na rozhraní eth0 rozhraní eth2, s cílovou adresou sítě VLAN2, řetězci IntKVlan2
iptables -A FORWARD -i eth0 -o eth2 -d 192.168.1.0/24 -j IntKVlan2
#předání příchozího spojení na rozhraní eth2 rozhraní eth0, se zdrojovu adresou sítě VLAN1, řetězci Vlan2KInt
iptables -A FORWARD -i eth2 -o eth0 -s 192.168.1.0/24 -j Vlan2KInt


#######################################################
#IntKVlan1 pravidla
#######################################################
#povolení všech příchozích spojení, která náleží již vytvořeným spojením
iptables -A IntKVlan1 -m conntrack --ctstate ESTABLISHED -j ACCEPT
#povolení nového tcp spojení HTTP směřujícího na adresu load balanceru
iptables -A IntKVlan1 -p tcp --dport 80 -d 192.168.0.100 -m conntrack --ctstate NEW -j ACCEPT
#povolení nového tcp spojení HTTPS směřujícího na adresu load balanceru
iptables -A IntKVlan1 -p tcp --dport 443 -d 192.168.0.100 -m conntrack --ctstate NEW -j ACCEPT
#povolení nového tcp spojení pocházejícího ze serveru Nagios
iptables -A IntKVlan1 -s 10.202.60.247 -p tcp --dport 5666 -m conntrack --ctstate NEW -j ACCEPT


#######################################################
#Vlan1KInt pravidla
#######################################################
#povolení odchozího spojení tcp, které náleží již vytvořenému spojení HTTP
iptables -A Vlan1KInt -p tcp --sport 80 -m conntrack --ctstate ESTABLISHED -j ACCEPT
#povolení odchozího spojení tcp, které náleží již vytvořenému spojení HTTPS
iptables -A Vlan1KInt -p tcp --sport 443 -m conntrack --ctstate ESTABLISHED -j ACCEPT
#povolení odchozího spojení tcp, které náleží již vytvořenému spojení Nagios
iptables -A Vlan1KInt -p tcp --sport 5666 -m conntrack --ctstate ESTABLISHED -j ACCEPT


#######################################################
#Vlan1KVlan2 pravidla
#######################################################
#povolení všech příchozích spojení, která náleží již vytvořeným spojením
iptables -A Vlan1KVlan2 -m conntrack --ctstate ESTABLISHED -j ACCEPT
#povolení nového tcp spojení pro přistup do databáze MongoDB
iptables -A Vlan1KVlan2 -p tcp --dport 27017 -m conntrack --ctstate NEW -j ACCEPT


#######################################################
#Vlan2KVlan1 pravidla
#######################################################
#povolení odchozího spojení tcp, které náleží již vytvořenému spojení MongoDB
iptables -A Vlan2KVlan1 -p tcp --sport 27017 -m conntrack --ctstate ESTABLISHED -j ACCEPT


#######################################################
#IntKVlan2 pravidla
#######################################################
#povolení všech příchozích spojení, která náleží již vytvořeným spojením
iptables -A IntKVlan2 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

```
#povolení nového tcp spojení pocházejícího ze serveru Nagios
iptables -A IntKVlan2 -s 10.202.60.247 -p tcp -m tcp --dport 5666 -m conntrack --ctstate  NEW -j ACCEPT


#######################################################
#Vlan2KInt pravidla
#######################################################
#povolení odchozího spojení tcp, které náleží již vytvořenému spojení Nagios
iptables -A Vlan2KInt -p tcp --sport 5666 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

B Skript pro zavedení firewallových pravidel na webovém serveru

```
#!/bin/bash
#odstranění veškerých pravidel a následné smazání veškerých nezabudovaných řetězů
iptables -F
iptables -X


#######################################################
#nastavení výchozí politiky jednotlivých řetězců
#######################################################
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP


#######################################################
#INPUT pravidla
#######################################################
#povolení všech příchozích spojení z místní smyčky (local loopback)
iptables -A INPUT -i lo -j ACCEPT
#povolení všech příchozích spojení, která náleží již vytvořeným spojením
iptables -A INPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
#povolení nového příchozího tcp spojení HTTP a HTTPS pocházejícího z adresy load balanceru
iptables -A INPUT -p tcp -m multiport --dports 80,443 -s 192.168.0.100 -m conntrack --ctstate NEW -j ACCEPT
#povolení nového příchozího tcp spojení pocházejícího ze serveru Nagios
iptables -A INPUT -p tcp --dport 5666 -s 10.202.60.247 -m conntrack --ctstate NEW -j ACCEPT


#######################################################
#OUTPUT pravidla
#######################################################
#povolení všech odchozích spojení z místní smyčky (local loopback)
iptables -A OUTPUT -o lo -j ACCEPT
#povolení odchozího spojení tcp, které náleží již vytvořenému spojení HTTP a HTTPS
iptables -A OUTPUT -p tcp -m multiport --sports 80,443 -m conntrack --ctstate ESTABLISHED -j ACCEPT
#povolení nového a již vytvořeného odchozího tcp spojení pro přistup do databáze MongoDB
iptables -A OUTPUT -p tcp --dport 27017 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
#povolení odchozího spojení tcp, které náleží již vytvořenému spojení Nagios
iptables -A OUTPUT -p tcp --sport 5666 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```