jamk.fi

Microsegmentation as part of organization's network architecture

Investigating VMware NSX for vSphere

Juha Koskinen

Master's thesis
May 2020
School of Technology
Master's Degree Programme in Information Technology
Cyber Security

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Description

Author(s)	Type of publication	Date
Koskinen, Juha	Master's thesis	May 2020
		Language of publication: English
	Number of pages: 89	Permission for web publication: x

Title of publication

Microsegmentation as part of organization's network architecture

Investigating VMware NSX for vSphere

Degree programme

Master´s Degree Programme in Information Technology, Cyber Security

Supervisor(s)

Karo Saharinen, Tero Kokkonen

Assigned by

Petri Kiiskilä, Finnish Defence Forces Logistics Command

Abstract

The rise of cloud computing and services needed for digitalization has brought organizations from simple centralized mainframe and self-hosted type of environments to complex, distributed and networked systems which are ever harder to secure and manage. The number of networked devices and connected services has exponentially increased. As more systems are exposed to a wider audience, and data contains more value for people and organizations, the need for security has risen. Implementing and managing systems security, however, has never been a simple problem to solve.

Software based revolution is taking place in the industry and it is disrupting how we build, scale, maintain, upgrade the systems in organizations. Hardware is becoming increasingly more generic, and the abstraction layer between hardware and end systems is becoming more sophisticated in being able to serve systems on top while management is efficient and the underlying hardware layer is hidden. The ultimate end goal for data centers will likely be just generic server hardware with necessary compute, storage capacity and interfaces, which then can be programmed to be part of a clustered pool of resources and made to serve multiple functionalities dynamically and in a distributed fashion.

The thesis looks critically at the problems faced with networked computer security and how different implementation models can have contrasting approach angles with distinctive architectural and management outcomes. The research explored how technological transformation of network segmentation implementation with a modern solution can benefit most organizations with certain kinds of computing environments. Microsegmentation was found to have several benefits for technical environments from management, complexity, adaptability and security perspectives.

Keywords/tags

Software defined networking, network segmentation, microsegmentation, virtualization

Miscellaneous

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Koskinen, Juha	Opinnäytetyö, ylempi AMK	Toukokuu 2020
	Sivumäärä 89	Julkaisun kieli Englanti
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Mikrosegmentointi osana organisaation verkkoarkkitehtuuria
VMware NSX for vSphere tutkinnassa

Tutkinto-ohjelma
Master´s Degree Programme in Information Technology, Cyber Security

Työn ohjaaja(t)
Karo Saharinen, Tero Kokkonen

Toimeksiantaja(t)
Petri Kiiskilä, Puolustusvoimien logistiikkalaitos

Tiivistelmä

Kasvava pilvilaskenta ja sen palvelut digitaalisaation tarpeisiin ovat tuoneet meidät yksinkertaisista keskitetyistä keskustietokoneista ja itse ylläpidetyistä ympäristöistä kohti monimutkaisempia, hajautetumpia ja verkotetumpia järjestelmiä, joita on yhä vaikeampi turvata ja hallita. Verkottuneiden laitteiden ja palveluiden määrä on eksponentiaalisesti lisääntynyt. Kun yhä useammat järjestelmät altistuvat laajemmalle yleisölle ja tiedolla on enemmän arvoa ihmisille ja organisaatioille, on turvallisuuden tarve kasvanut. Järjestelmäturvallisuuden toteuttaminen ja hallinta ei kuitenkaan ole koskaan ollut yksinkertainen ongelma ratkoa.

Ohjelmistopohjainen vallankumous on tapahtumassa alalla, ja se on häiritsevä voima siinä, kuinka teemme järjestelmien rakentamista, laajentamista, ylläpitämistä ja päivittämistä. Laitteistosta tulee yhä yleisempiä. Laitteistojen ja loppujärjestelmien välinen abstraktiokerros kehittyy entistä hienostuneemmaksi, kun pystytään palvelemaan päällä olevia järjestelmiä tehokkaasti ja piilottamaan alla oleva laitteistokerros. Palvelinkeskukset tulevat loputtua todennäköisesti koostumaan laitteistoista, jossa tarvittava laskenta-, tallennuskapasiteetti rajapintojen avulla voidaan ohjelmoida osaksi klusteroitua ”resurssiallasta” ja tuottamaan useita toimintoja dynaamisesti ja hajautetulla tavalla.

Työssä on tarkasteltu kriittisesti verkottuneen tietoturvan ongelmia ja sitä, kuinka erilaisilla ratkaisumalleilla voi olla erottuvat lähestymiskulmat arkkitehtuurisesti ja hallinnollisesti eroavilla tuloksilla. Tutkimuksessa selvitettiin, kuinka verkkosegmentoinnin teknologinen muutos nykyaikaisella ratkaisulla pystyy hyödyntämään organisaatioita, joilla on tietyntyyppisiä ympäristöjä. Mikrosegmentoinnista löydettiin selviä hyötyjä ympäristön hallinnan, monimutkaisuuden, muuntautumisen ja turvallisuuden näkökulmista.

Avainsanat
Ohjelmisto-ohjatut verkot, verkkosegmentointi, mikrosegmentointi, virtualisointi

Muut tiedot

Contents

1 Introduction 6
1.1 Background 6
1.2 Data as valuable resource 7
1.3 Defence perimeters 8
1.4 The need for a change in networking 9

2 Research basis 11

3 Infrastructure virtualization 14
3.1 Software-defined infrastructure 14
3.2 Compute virtualization 15
3.3 Network virtualization 16
3.3.1 Software defined networking 18
3.3.2 VMware NSX 20

4 Network segmentation 25
4.1 Challenges in segmentation 25
4.2 Designing secure networks 27
4.3 Traditional segmentation 29
4.4 Zero trust model 31
4.5 Microsegmentation 34
4.6 Segmentation policy 38

5 Research 44
5.1 Test environment 44
5.2 Test scenarios 46
5.2.1 Scenario A - Traditional segmentation with external firewall 47
5.2.2 Scenario B - NSX microsegmentation with flat logical network 48

5.2.3 Unused alternative scenarios 49
5.3 Test methods 51
5.3.1 Method 1 - Security control 51
5.3.2 Method 2 - Network performance 51
5.3.3 Method 3 - Application performance 52
5.4 Test results 53
5.5 Analysis of results 56

6 Conclusions 58
6.1 Outcome of results 58
6.2 Promising potential 59
6.3 Organizational readiness 60
6.4 Reflection and further research 64

References 66

Appendices 70
Appendix 1. Test scenario VM networks and IP addressing 70
Appendix 2. SRX firewall rule configuration 71
Appendix 3. Test run output - nmap 71
Appendix 4. Test run output - qperf 72
Appendix 5. Test run output - sysbench 74
Appendix 6. Test virtual machine specifications and DRS rule 84
Appendix 7. Test virtual machine network cards and memberships 85
Appendix 8. SRX firewall rule management 85
Appendix 9. NSX firewall rule management 85
Appendix 10. NSX network trace between VMs through a logical switch 86

Figures

Figure 1. Simplified separation of management, control and data planes with SDN implementation 19
Figure 2. NSX microsegmentation in a datacenter 22
Figure 3. NSX context-aware firewall policy 23
Figure 4. East-west traffic patterns for traditional and NSX implementations 24
Figure 5. Flat network of hosts segmented with Virtual LANs 29
Figure 6. Private VLAN port types and allowed data flow 30
Figure 7. Multi-tenancy enabled with virtualization of routing by VRF 31
Figure 8. Microsegmentation can be used to protect both flat and VLAN & VRF enabled networks 37
Figure 9. Network topology for simple 3-tiered internet accessible web system 39
Figure 10. Example of using virtual machine tags to define security groupings 42
Figure 11. Test environment network topology 46
Figure 12. Layer 2 data path between VMs with traditional segmentation 47
Figure 13. Layer 2 data path between VMs with virtualized networking 48
Figure 14. Layer 2 data path between VMs inside same hypervisor host 49
Figure 15. Layer 2 data path between VMs with virtual firewall 50
Figure 16. Layer 2 path between VMs and through virtual firewall on same host 51
Figure 17. TCP & UDP bandwidth test results 54
Figure 18. TCP & UDP latency test results 55
Figure 19. Database benchmarking throughput results 55
Figure 20. Database benchmark latency results 56

Tables

Table 1. Traditional firewall ruleset using IP addressing 40
Table 2. Firewall ruleset using object groups 40
Table 3. Security group definitions and criteria rules for tiered web system 41
Table 4. Definition of security groups and tag use for multi-environment system 42
Table 5. Summation of results from all test methods 53

Acronyms

API	Application Programming Interface
CMDB	Configuration Management Database
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DFW	Distributed Firewall
DHCP	Dynamic Host Configuration Protocol
DLR	Distributed Logical Routing
DNS	Domain Name System
DRS	Distributed Resource Scheduler
ECMP	Equal-Cost Multi-Path
ESG	Edge Services Gateway
HA	High Availability
I/O	Input/Output
ICT	Information and Communication Technology
IDFW	Identity Firewall
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPS	Intrusion Prevention System
IT	Information Technology
L2VPN	Layer 2 Virtual Private Network
LAN	Local Area Network
MAC	Media Access Control
MPLS	Multiprotocol Label Switching
NAT	Network Address Translation
NFV	Network Function Virtualization
NGFW	Next Generation Firewall
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
NVGRE	Network Virtualization using Generic Routing Encapsulation
OFDMA	Orthogonal Frequency-Division Multiple Access
OLTP	Online Transaction Processing
OSI	Open Systems Interconnection
PCI DSS	Payment Card Industry Data Security Standard
RAM	Random Access Memory
SaaS	Software as a Service
SDLC	Software Development Life Cycle
TDM	Time-Division Multiplexing
vCPU	Virtual Central Processing Unit
VLAN	Virtual Local Area Network
VM	Virtual Machine
VPLS	Virtual Private LAN Service
VPN	Virtual Private Network

VRF	Virtual Routing and Forwarding
VTEP	VXLAN Tunnel Endpoint
VXLAN	Virtual Extensible LAN
WAN	Wide Area Network
WDM	Wavelength-Division Multiplexing

1 Introduction

1.1 Background

The importance of information and communication technology in our modern society and economy has grown remarkably as the everyday living and critical infrastructure are more reliant on computer networks and systems. (Jang-Jaccard 2014) These complex solutions of digital era have made businesses and lives easier, cost-effective and empowered. Computers enable their users in many ways; however, as these systems have become more important to people - logically their value and power to influence has too increased. For example, information about people's lifestyles and behaviours are collected at a massive scale, and these data stores have become very valuable business tools for purposes such as marketing, influencing and predicting globally.

The Finnish Defence Forces Logistics Command is responsible for acquisitions of defense material for the Defence Forces including its availability, support and management of technical life cycle. The Logistics Command, as a subordinate to the Defence Command, is tasked to ensure operational capability and effectiveness of troops and systems in national and international environments. The Joint Systems Centre, as a unit of the Logistics Command, employs approriately 500 people and is responsible for procurement preparations, life cycle management, maintenance management and technical inspections of the Defence Forces' systems and equipment. (Finnish Defence Forces.)

Research and development of information systems and underlying technical platforms to support the Defence Forces is a vital part of increasing military efficiency and capability. Evaluating new technologies is critical part of this. Microsegmentation and network virtualization is researched and evaluated to provide understanding of the phenomenon and how it could be utilized in the Defence Forces.

Investigating microsegmentation with NSX originated from an actual work assignment from the Finnish Air Force Command for the Logistics Command to explore if the technology could be used to provide security and operational

enhancements into virtualized compute environments used by the Air Force. This preceding investigation included implementing an actual full-scale Proof of Concept lab environment to explore benefits and usage of NSX with real military command and control systems deployed on top and attempting to utilize NSX features to improve deployment and operation of systems. This thesis work is an extension of this aforementioned assignment in order to gain more comprehensive understanding into organizational benefits for benefit of the assigner and broader audience in the industry. The knowledge and conclusions gained from this thesis will be used to support the technical evaluation of the NSX solution for the Air Force.

1.2 Data as valuable resource

For most of the industries data has become the lifeblood for organizations, and it is a very valued merchandise for the criminal underground. Europol reports in their Internet Organized Crime Threat Assessment (IOCTA) report of 2018 that over 55% of EU member states have investigated some form of network attack, other than DDoS, and attacks are being reported more frequently. Data is often acquired through these network intrusions and used for a number of purposes based on the type of the data stolen. A multitude of tactics is utilized; however, most common are different forms of hacking and malware usage usually delivered through malicious email. (IOCTA 2018, 22)

In 2019 report of IOCTA, Europol highlights data being the key element in all cybercrime that year. Data security and regulations is made more important as organizational and personal consumer data is being targeted by cyber criminals. (IOCTA 2019, 6) Ransomware exists as the leading threat even with decreased volume; however, attacks are being increasingly targeted for resulting more profits and greater economic damage (IOCTA 2019, 15).

Gartner, a global research and advisory firm, estimates worldwide IT security spending will have hit over $124 billion in year 2019, which is a 9% increase to previous year. Over half of that spending is measured to end up in Security Services market segment with Infrastructure Protection and Network Security Equipment segments coming second and third respectively. Based on a Gartner survey, top three drivers

for spending to security are security risks, business needs and industry changes. As organizations are going through their digital transformation journeys, security will be a key factor in that process as data is more regulated, critical operations and intellectual property need to be protected while utilizing new frontiers such as public cloud, SaaS solutions and Internet of Things devices. (Gartner Forecasts Worldwide Information Security Spending to Exceed $124 Billion in 2019; 2018)

Traditionally organizations have implemented perimeter defense strategy which puts a wall around all the assets and protects from unwanted outside access. The perimeter defense is technically most often built with a firewall and intrusion prevention system. The whole concept relies on having assets and access node groups segmented into their own silos and access control is then implemented by forcing network traffic to go through the centralized perimeter mechanisms which allows for interception and examination. This defense model has been generally favored due to its ease of implementation and for low cost of securing large volume of systems or networks. Other access control mechanisms are usually implemented on top of individual assets or systems in addition to perimeter defense if more security presence is required. (Jang-Jaccard 2014.)

In March 2019, one of the world's largest aluminum producing company based in Oslo suffered a major ransomware attack. The systems for office IT and factory equipment went dark. Operations had to fallback to manual procedures which slowed or halted production. Business was severely impacted. The malware was able to spread easily inside the company's infrastructure that was not hardened enough and not patched. The CEO of a New York-based cybersecurity firm commented on the case as follows: "The incident also underscores the need to keep critical systems isolated from one another." (Korolov 2019.)

1.3 Defence perimeters

As cyber threats and tools have evolved and become more sophisticated, attacks have found loopholes to bypass these perimeter defenses. In addition, people's environment, habits, tools and have evolved as well, which has opened new opportunities for exploitation to get around the perimeter defense. One example is

the rise of mobile way of working where the endpoint device can move freely in and out of the perimeter. (Jang-Jaccard 2014.)

In modern data centers the perimeter defense strategy has been deemed lacking by most enterprise IT professionals. Once the perimeter has been breached, an attacker is able to move around from system to system with small effort and access internal resources. Lateral movement inside the network or system is made possible when there are no adequate network controls implemented inside the perimeter to protect individual assets. (Micro-Segmentation Builds Security into Your Data Center's DNA 2016.)

As threats get more sophisticated and grow uncontrollably in volume every day, the cost of data breaches is rising and in July 2019 Ponemon Institute with IBM Security reported global average cost of $3,92 million with malicious attacks as the leading cause for breaches. (Ponemon 2019.)

Breakthroughs in recent years have made it possible to enforce finer control over data center security allowing creation of automated and distributed security policies that can separate workloads and safeguard them individually or as a logically grouped segment. This security model is most often called microsegmentation, it is constructed in software and it enables more effective workload segmentation in enterprise networks. (Micro-Segmentation Builds Security into Your Data Center's DNA 2016.)

> *The advancement of technology is continuous, cyclical, non-linear and is based on previous innovations. Nothing is created from scratch; instead every technology has its history. Technology is related to its time, a phenomenon is utilized by technology, when the maturity of a certain technology is at right level.* (Translated by the author.) (Lehto 2014, 157.)

1.4 The need for a change in networking

In a article about healthcare cybersecurity a vice president and CISO of a hospital in Florida states that they have setup a dedicated virtual LAN for medical devices separate from the production network but that they also have implemented mini-firewall like network devices in front of each medical device. This implementation of

microsegmentation is needed since devices are otherwise vulnerable for attacks that have not been patched or cannot be replaced without costing millions of dollars. The acceleration of cyber related threats against healthcare data and IT security is forcing organizations to re-think how they approach device segmentation to reduce the risks related to increasing threat vectors such as malware. However, the majority of healthcare IT is still behind and carrying out manually costly and ineffective traditional segmentation. Software-defined configuration or networking is called to answer these problems in terms of speed and policy-driven enforcement access controls. In cases like these, a managing consultant from CyberEdge Group company recommends a granular approach for segmentation where policies are defined in terms of application concepts rather than network constructs such as IP addresses, subnets and VLANs. Abstraction of underlying infrastructure and focusing on application concepts creates dynamic policy that can be automated and rapidly deployed. (Hagland 2018.)

The requirements and methods of operating in today's digitally networked world have changed for businesses and end users. Traditionally built network architectures are no longer able to properly meet the new requirements. (Software-Defined Networking: The New Norm for Networks 2012, 2.)

The new trends, and thus driving forces of change, in the networking industry include rapidly growing phenomena such as smart mobile devices and applications, server virtualization and cloud service adoption. Conventionally networks were designed for a different age with needs and a usage profile fitting that. Those designs made sense back then; however, they resulted in a static architecture not able to adapt to new needs demanded by the industry. The client-server computing was very dominant before and is still heavily utilized; yet, modern computing and storage systems are dynamic in their nature and require the networking platform to support that. (Software-Defined Networking: The New Norm for Networks 2012, 3.)

Traffic patterns have changed inside data centers. Previously most of the communication took place between one client and one server in "north-south" type of pattern. In today's networked and distributed applications communication takes place between multiple different servers (machine-to-machine) creating a great amount of "east-west" traffic. Users also expect to access these applications from multiple types

of devices, from anywhere in the world and at any time. Availability requirements have changed drastically during the computer age. Compared to the old terminal workstation and mainframe server type of computing the expectations have changed a great deal. The cloud computing trend also has brought the industry new ways of operating and matters to consider. Enterprises either have already considered or will need to consider usage of private cloud, public cloud or hybrid model, which will also affect traffic patterns and can result in increased exposure of network traffic through public networks. Although usage of cloud services brings agility and efficiency with on-demand IT resources to enterprises, it will also add complexity to planning with increased requirements in security, compliance and auditing. Businesses are also more dynamic, and assumptions can change overnight with reorganizations, consolidations and mergers of businesses. (Software-Defined Networking: The New Norm for Networks 2012, 3-4.)

In short, the limits of current technologies that have been in use for decades already are already reached. New solutions have been built on top of old standards and protocols; however, the networking platforms and the way of thinking have not yet been fully transformed to the next level to better meet today's business needs.

2 Research basis

Due to the assigner specifications of this thesis work, the research scope is aimed at a single product named VMware NSX. The NSX solution offers a network virtualization and security platform in-line with VMware's Software-Defined Data Center umbrella model. NSX greatly expands the capabilities of the existing virtualization platform and enables several new features such as allowing microsegmentation, virtualized network functions and automation of operational tasks in the virtualizated infrastructure. NSX for vSphere offers easy first step into the SDN world and its benefits without the complexities of full-on deployment retransforming the whole network infrastructure. The ease of deployment and ability to co-exist with the old environment enables painless migrations and allows the use of many primary features of NSX without making changes to the infrastructure.

Following problems have been identified from traditional segmentation implementations in the past. This identification is based on the researcher's industry knowledge and observations from the assigner environments including conversations with people responsible for infrastructure platforms (servers and networking) and deployed systems on the platforms.

1. Segmentation of workloads for high security environments with traditional methods is difficult to implement and maintain, is complex and is not cost-effective.
2. Traditional network architecture is static and adapts poorly to new needs.
3. Network changes require man hours for planning and configuration of each individual network device with different feature sets and management interfaces.
4. Traditional segmentation is a compromise between security, resources, complexity and ease of management.

These can roughly be summarized to be a problem of ease of implementing segmentation to improve security and operations. The research question thus is: **can an organization use microsegmentation implemented with NSX to introduce security and operational improvements for workload segmentation without meaningful loss of performance?**

The objective of this research is to provide an informational overview and understanding of how microsegmentation enabled by software defined networking solution can be utilized to improve security management and agility of an organization operating a datacenter or suitable virtualization platform. The study relies on comparisons to current segmentation practices in the industry as they are applied to secure different types of workloads most commonly seen at enterprise environments.

Due to the problems identified for this research being both subjective and partly difficult to quantify, both qualitative and quantative research approaches were chosen for this thesis work.

Case study research strategy is used to acquire a stronger understanding of the researched phenomenon with narrow focus on single case to gain detailed information on the subject. Surrounding processes and structures around the

phenomenon can be explored to recognize their impact. The flexiblity offered by the case study framework is very approriate for this applied research since it is difficult to achieve pragmatic and productive results with other research strategies with this subject. (Routio 2007.)

Action research methodology is used to improve existing activity by researcher proposing improvements and then carrying out an investigation. It is known to be effective for handling complicated issues in working environments and to solve problems faced by the community. Action is executed and results are evaluated on the effects. Distance is taken to reflect afterwards for understanding on why the process or situation is now as it is. (Routio 2007.)

The aim is to explore interaction with microsegmentation phenomenon in a technical environment using action research method. Two primary viewpoints are deducted from aforementioned research problems, security and operational performance. Policy management would have been interesting to study as well; however, this was not practically achievable in the researching environment available. Research data collection will be done via unstructured behavioral observations by the researcher while investigating the phenomenon and controlled lab experiments. Observations will not be recorded or documented as these cannot be so strictly defined, instead subjective interpretations are reported in conclusions and reflections with a critical attitude.

When observations for data collection are made in a natural setting without pre-defined plans and instruments to guarantee standardization and precision, it is called uncontrolled observation method and enables the researcher to obtain natural impromptu views of the research subject often for more complete picture. Subjective interpretation is fundamental drawback of this method in contrast to strictly controlled observation and can raise uncertainty to results. For this reason, controlled lab exams are also made here to support credibility in research outcomes. (Kothari 2004, 97.)

The plan is to investigate the microsegmentation and software-defined networking phenomonen on theoretical plane in relation to the aforementioned research problems and test out the capabilities of the implemented solution in practice. The

effectiveness of segmentation security can be studied with simple technical auditing tools. The operational impact of the phenomonen can be examined with use of assessment software from which key performance metrics can be analyzed to provide an informational basis for evaluating the technological solution.

3 Infrastructure virtualization

3.1 Software-defined infrastructure

The networking field is currently in the middle of transformative revolution similar to what server virtualization has made to computing services. In a similar fashion, networking is moving from hardware to software mode and solutions. (Pujolle 2015, ix.)

The solutions based on software defined networking are currently quite promising especially for enterprise and carrier networks. As an example of one such prominent open standard SDN based technology is OpenFlow and according to the non-profit industry consortium the Open Networking Foundation (Software-Defined Networking: The New Norm for Networks 2012, 2-3), its benefits include:

1. Centralized management and control for multi-vendor network devices
2. Common APIs are used to improve automation and management by abstracting the underlying networking details
3. Enabling faster innovation with new network services and capabilities with less dependency on device vendor and need for device configurations
4. Programmability is allowing new opportunities for multiple parties to implement new solutions to drive revenue and differentiation
5. Centralized management, device automation and enforced uniform policies increase the reliability and security of network with fewer configuration errors
6. Increased granularity in network control policies allowing separation of individual sessions, users, devices and applications
7. Improved user-experience as applications have awareness of network state and have capability to make adjustments based on user needs
8. Dynamic network architecture that future-proofs the network for future investments and is able to adapt quickly to changes in requirements

One significant commercial approach has been taken by the virtualization giant VMware with its Software-Defined Data Center (SDDC) concept where higher utilization of software is being used to abstract, pool and automate data center resources and services. VMware is stating that the current mobile and cloud enabled era is bringing new challenges to IT organizations and to answer this, organizations would need to virtualize more of the data center infrastructure services in order to gain cost-efficiency, security benefits and improve management. In SDDC enabled clouds compute, storage and network resources are provisioned and managed automatically by policies defined by the organization and thus providing efficiency and agility for IT operations. The SDDC concept combines VMware's multiple software-defined products into a package with high level of integration and automation, software defined networking being one significant component of that resulting solution. SDDC products work on any x86 server and any IP transport network. (VMware SDDC.)

The main components of SDDC consist of the following products:

1. Compute virtualization with vSphere
2. Storage virtualization with vSAN
3. Network virtualization with NSX
4. Cloud management with vCenter and various vRealize products (VMware SDDC.)

3.2 Compute virtualization

Virtualization has been a disruptive technology that has transformed the way computing services are provided for consumers and businesses. At the core of these services is the technological ability to abstract physical components into scalable, elastic and lean virtualized objects. By means of virtualizing an object, more utility is gained out of the resource produced by the object. (Portnoy 2016, 1-2.)

For datacenters around the world, virtualization has enabled to consolidate physical servers into fewer servers running virtual machines and utilization of computational resources at much higher rate. This has allowed companies to decommission large portions of their physical servers and thus cut down costs on many aspects such as

hardware maintenance, administrative tasks and physical expansion. (Portnoy 2016, 10-11.)

Virtualization is a fundamental technology for enabling cloud-based services as it pools multiple hardware systems into a shared platform of compute resources such as networking, CPU, memory and storage. At the same time, abstracting the hardware and complexity of running computing platform while providing ease of scalability and built-in multi-tenancy isolation through software-based virtualization. (Dawoud, Takouna & Meinel 2010, 4.)

When talking about virtualization in the scope of IT infrastructure, for most companies and professionals it often refers to "x86 server hardware virtualization" provided by multiple different commercial and open source products, the best known solution in the industry is VMware's vSphere. Virtualization of compute resources allows the OS and applications once tied to physical hardware to share the same resource components with others through hypervisor's resource management such as CPU scheduling, memory management, I/O for storage and networking. Each application and service and its OS reside in an isolated virtual machine object created in software and thereupon abstracted from direct access to the hardware. (Mitchell & Keegan 2011.)

3.3 Network virtualization

The concept of network virtualization is similar to server virtualization, in that the aim is to create components or functions in software that were previously physical and thus in the process make it more cost efficient, secure, scalable and even automated. Network components such as switches, routers, firewalls, load balancers, network cards/adapters, logical ports and connections can all be implemented in software without any additional hardware or manual labor, assuming a supported and configured virtualization platform exists in the environment.

The most prominent network "virtualization" technology currently has been IEEE 802.1Q enabled virtual LANs (VLANs) that are used to partition Ethernet networks for the purpose of performance and security as multiple isolated logical network segments can co-exist within the same physical networking infrastructure. For many organizations VLANs create flexibility in work environments as same cabling can be

used for different types of employee roles and needs, otherwise lot of unnecessary extra cabling would be required for each case.

VLANs were first introduced in 1993 with two major drivers to justify their implementation; lowering cost of change management and improving performance of client-server applications. As the technology started to gain momentum, in 1996 Virtual LANs were already a hot topic in the industry and businesses contemplated on deciding if virtualizing their organizations' networks made sense. LAN switching was compared to slower frame transfer speeds of "router-based LAN microsegmentation". The value of virtualized LANs with shared medium was rather questionable for people as different conflicting methodologies for implementations existed in the networking field during that time. Application fingerprinting was considered essential to guide decision-making. (Business communication review 1996)

The use of VLANs makes network design easier and supports businesses in their adaptation to growth and changes. The use of VLANs include benefits such as (Cisco Networking Academy's Introduction to VLANs 2014.):

1. Security, through separating computers with sensitive data from rest of the network
2. Cost reduction is gained from more efficient use of existing network hardware and uplinks
3. Better performance when layer 2 broadcast segments are divided into multiple logical workgroups which reduces unnecessary network-wide traffic
4. Limit failure domain when a logical network fails or causes disturbance
5. Simplify management for projects and applications when these can be logically grouped together

For most enterprises and cloud service operators with large networks, further isolation is required to separate customers or departments from each other. Most commonly private IP space is overlapped between so-called tenants of shared infrastructure, which leads traditionally to separation of layer 3 routing space. Without virtualization capabilities this would mean additional clustered physical router hardware for each instance of the needed routing space. Various enterprise routers and firewalls

have supported creating virtual routing instances with separate routing and forwarding information tables through technology called Virtual Route Forwarding (VRF). VRF combined with VLANs allows tenants to have multiple logical network segments that have no knowledge of other tenant networks. VRF functionality is entirely created in software that is part of the physical router operating system. (Virtual Route Forwarding Design Guide 2008.)

The use of virtualized firewalls has risen in data centers as they do not carry some of the disadvantages and limitations of physical firewalls. They share compute, storage and network resources with other virtual machines within the deployed virtualization infrastructure and can scale easily with expansion of the platform. There are two types of virtual firewalls. Subnet-level firewall operates on a dedicated VM with multiple virtual NICs, each connected to a different virtual segment. These operate much like physical firewalls but the hardware is just virtualized. Kernel-level virtual firewall instead functions inside the virtualization hypervisor operating system as loadable module and can directly intercept every packet entering or leaving the protected VM. The actual traffic filtering can also be offloaded from kernel to a dedicated VM for more fine-grained policy processing or monitoring/logging purposes. (Chandramouli 2016, 15-18.)

3.3.1 Software defined networking

In the last decade, the software defined networking (SDN) field grew substantially in the industry, and it has already seen production-ready products deployed to many big customers. Most of these early adopters are large telco or cloud provider organizations with much to gain from taking their services to the next level and improving their competitive edge.

SDN is built to use application programming interfaces (APIs) to allow software developers and networkers to easily configure devices, services and applications in the network. In traditional networking, network components are managed and controlled individually, i.e. the control plane is distributed in each network device. In software defined networking, the control of the networking infrastructure is transfered to a separate centralized control plane where SDN controllers dictate data plane actions and interface with network services and business applications.

Capabilities and features come from controllers and services instead of the individual hardware devices inside the network. Figure 1 illustrates the difference between traditional and software-defined implementations where data, control and management planes are separated from individual devices to a centralized location. SDN based network infrastructure will act as a platform for various functions and allows new kind of implementations in the areas of automation, security and service insertion. One of the most sought early benefits from SDN is the ability to do dynamic network segmentation and overlay networking. (Kirkpatrick 2013, 1-3.)

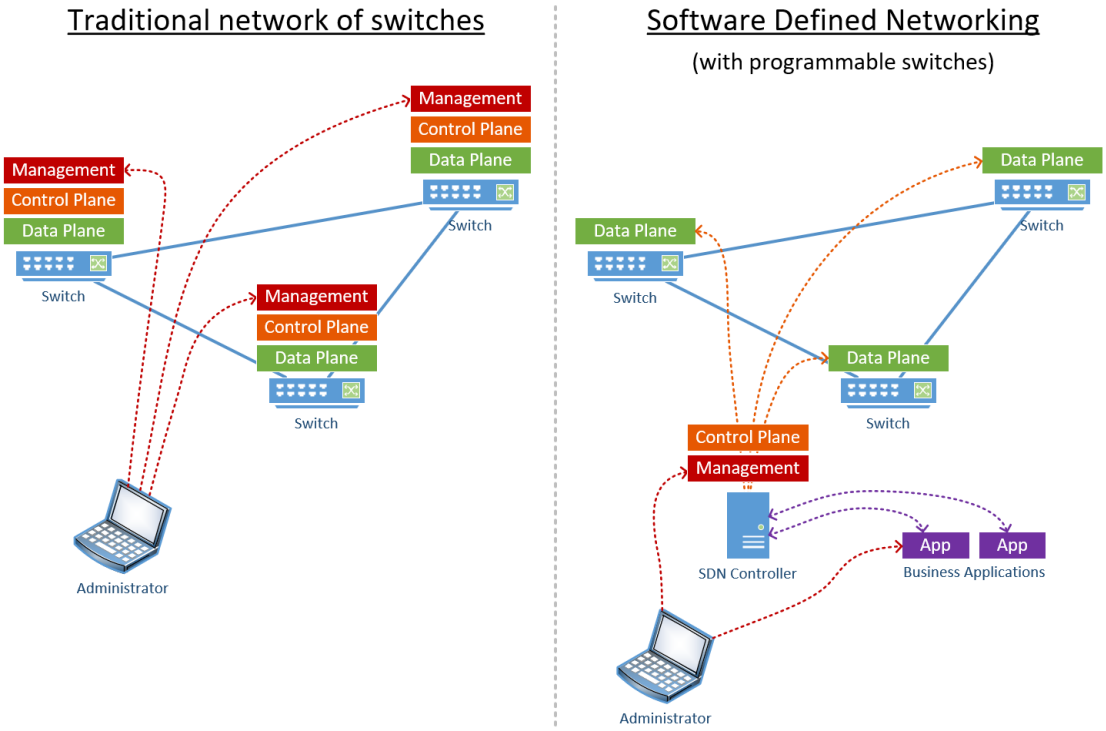


Figure 1. Simplified separation of management, control and data planes with SDN implementation

Virtualization of the network infrastructure through SDN enables efficient microsegmentation implementations and integration of applications to provide various services. Applications can be programmed to instruct the network on how to function and run optimizations for optimal application performance. (Jaworski 2017, 9-10.)

Software-defined approach to implementing network security posture can facilitate modern Software Development Life Cycle (SDLC) practices such as DevOps and new deployment methods for applications and services when the application can interact with the network and hence be optimized to serve it better. Adaptive microsegmentation can automatically and dynamically change based on the workload and allow or deny specific traffic patterns. The security policy is attached to the workload, not to network, and thus follows it when moved and gets removed neatly when not needed anymore at the end of application lifecycle. Automation and microsegmentation implementations create value for application deployments by increasing efficiency and level of security. (Shackleford 2019, 5-8.)

3.3.2 VMware NSX

VMware currently has two overlapping NSX products for datacenters due to them first developing their own vCloud based network virtualization for vSphere platform and later acquiring a company named Nicira in 2012 to procure a network virtualization platform independent of underlying infrastructure. The first product is named NSX-V or NSX for vSphere and depends on VMware's virtualization platform; it is also more robust and mature with features. The latter is called NSX-T or NSX for Transformers and it can be considered to be more extensive SDN solution as it is meant to be implemented further into enterprise network infrastructure than just virtualization platforms. NSX-V currently enjoys a quite painless and non-intrusive deployment method into a new or existing vSphere environment making it very attractive for organizations in need for its unique features. In the terms of microsegmentation implementation, both products are mostly identical in how they operate and are managed; yet, in practice only NSX-V is used in this thesis and will be refered to NSX in this thesis. In addition to scalable software-based microsegmentation firewalling, switching, routing, load balancing and VPN tunneling, NSX encompasses unique advantages such as allowing extending networks across physical networks with VXLAN overlay networking and enabling smooth management of security controls across multiple cloud platforms for unified policy and monitoring. (The History of NSX and the Future of Network Virtualization 2016.)

According to VMware's NSX whitepaper, NSX can be deployed non-distruptively on any hypervisor connected to physical network infrastructure and supports network fabric implementations from any vendor. The existing networking, applications and workloads require no changes. NSX allows incremental implementation of virtual networks and security policy at organization's own pace. The solution requires less from the physical networking hardware as its functionality and processing is offloaded to virtualization servers, thus reducing need for expensive networking equipment and feature licencing. It is easier and faster to acquire new features and bandwidth capacity and scale down/up as networking occurs at the hypervisor and closer to the workloads, e.g. increasing switching and routing capacity to tens and hundreds of Gbps in virtual networking is likely easier and cheaper than getting equivalent throughput from traditional networking. NSX also allows integrations with multiple cloud providers and cloud management solutions. (The VMware NSX Network Virtualization Platform 2013, 6-12.)

VMware is claiming that their NSX solution provides true microsegmentation to cases where east-west traffic needs to be protected without need for additional hardware. Traditionally in these cases traffic has been forced through an inspection point which most often is deployed separately, needing traffic to pass multiple network hops just to turn around and come back, hence incurring unnecessary use of underlying infrastructure when there is a more direct path available between the endpoints. (Jaworski 2017, 10.)

In the distributed firewall (DFW), NSX utilizes a grouping system that allows inclusion or exclusion of virtual machines by variety of static or dynamic factors. Instead of just relying on network constructs (MAC/IP addressing, interfaces) and payload inspection (application identification), the system allows referencing to various logical objects of the virtual infrastructure such as VM name, OS name, object location (vApp, resource pool etc.), user-set security tags and network membership. As presented in Figure 2, this allows for distinctively different logical segmentation results. (Jaworski 2017, 13)

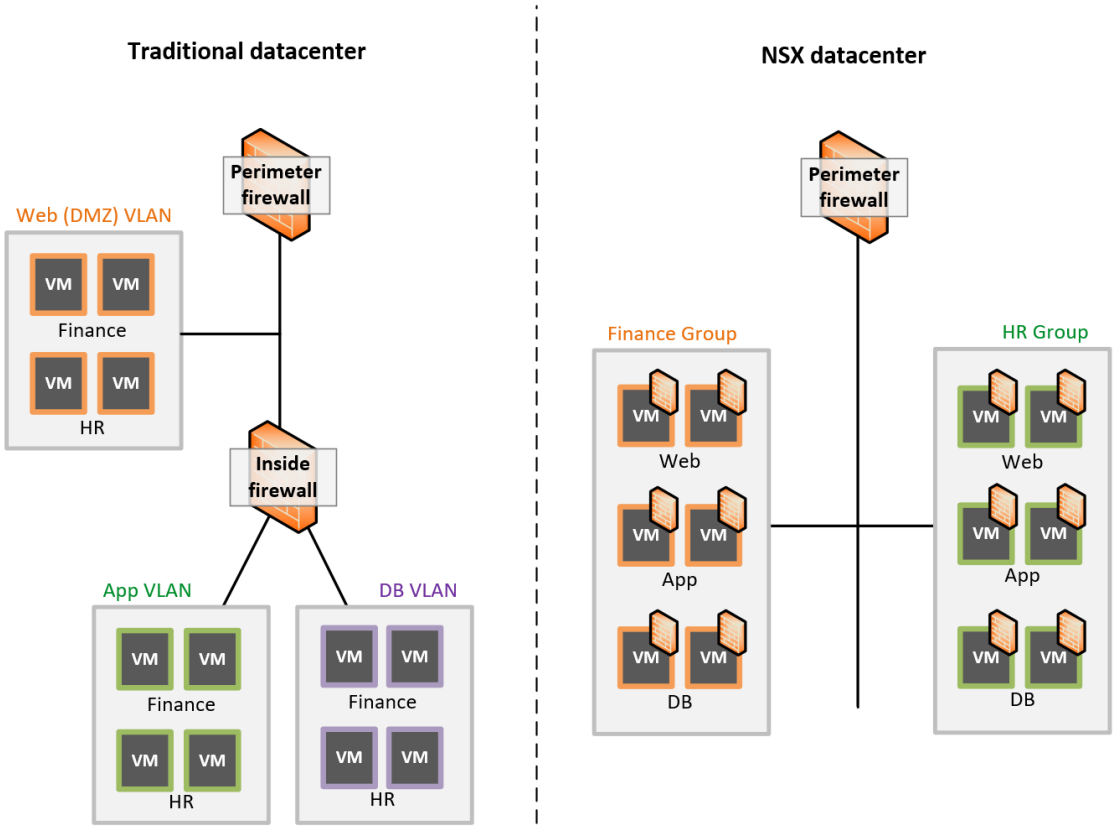


Figure 2. NSX microsegmentation in a datacenter

The distributed firewall of NSX allows for context-aware microsegmentation and enables application and user identification with its built-in context-engine as seen in Figure 3. Every connection has tracked context attributes that can be used in mapping to filter rules of a security policy. Packet payloads of application flows are matched to pattern signatures enabling OSI Layer 7 firewalling capability. Context awareness enables full visibility into application network flows inside the perimeter. (Vanveerdeghem 2018.)

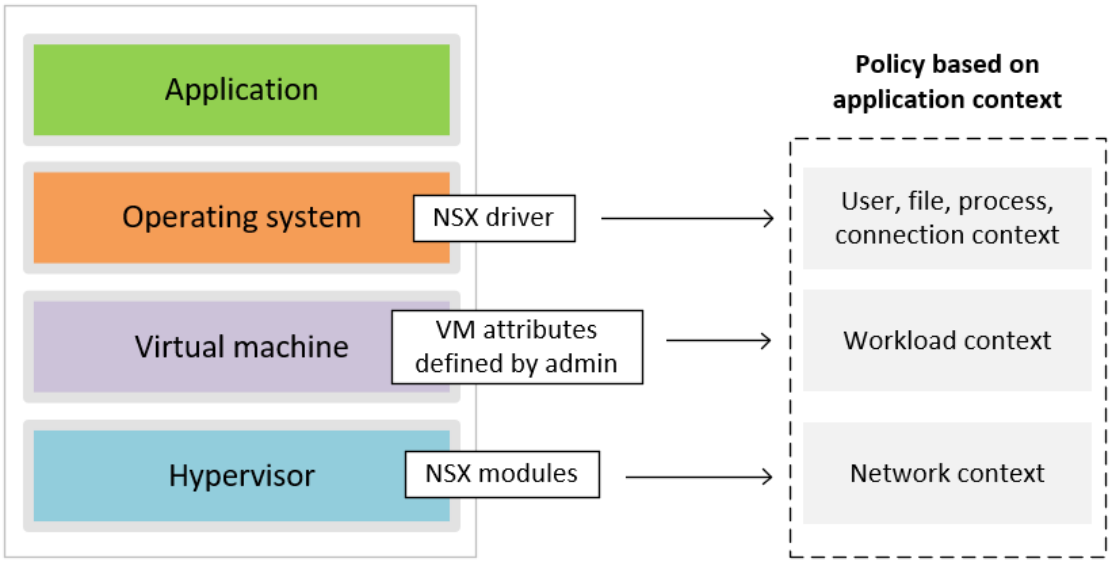


Figure 3. NSX context-aware firewall policy

Identity firewall is enabled through either using Guest Introspection feature of NSX or scraping Active Directory logs and with the use of VMware Tools Thin Agent inside the virtual machine, user information is mapped to connection flows. The Security Identifiers (SIDs) of a user is inserted directly to the data plane of hypervisors where the filtering rules are enforced. (Vanveerdeghem 2018.)

Implementing NSX with microsegmentation will lead to more efficient traffic patterns and eliminate overprovisioning resulting from downsides of physical security implementations such as hair-pinning. With NSX the traffic never leaves the physical server and thus consumes zero bandwidth capacity of the physical network infrastructure. (Micro-Segmentation Builds Security Into Your Data Center's DNA 2016, 6)

Figure 4 illustrates the implementation difference in communication path for virtual machines communicating via same hypervisor host or between two different hosts. In both cases the distributed logical routing (DLR) functionality of NSX also enables the same effect in routing as hair-pinning is not needed. DLR is also built as hypervisor kernel module like DFW; however, it includes a control VM to provide routing protocol peering with neighboring routers. With firewalling and routing (DFW & DLR)

both implemented in distributed way, the traffic pattern for east-west communications in the network infrastructure effectively changes and becomes more direct between endpoints while segmentation is enforced strongly.

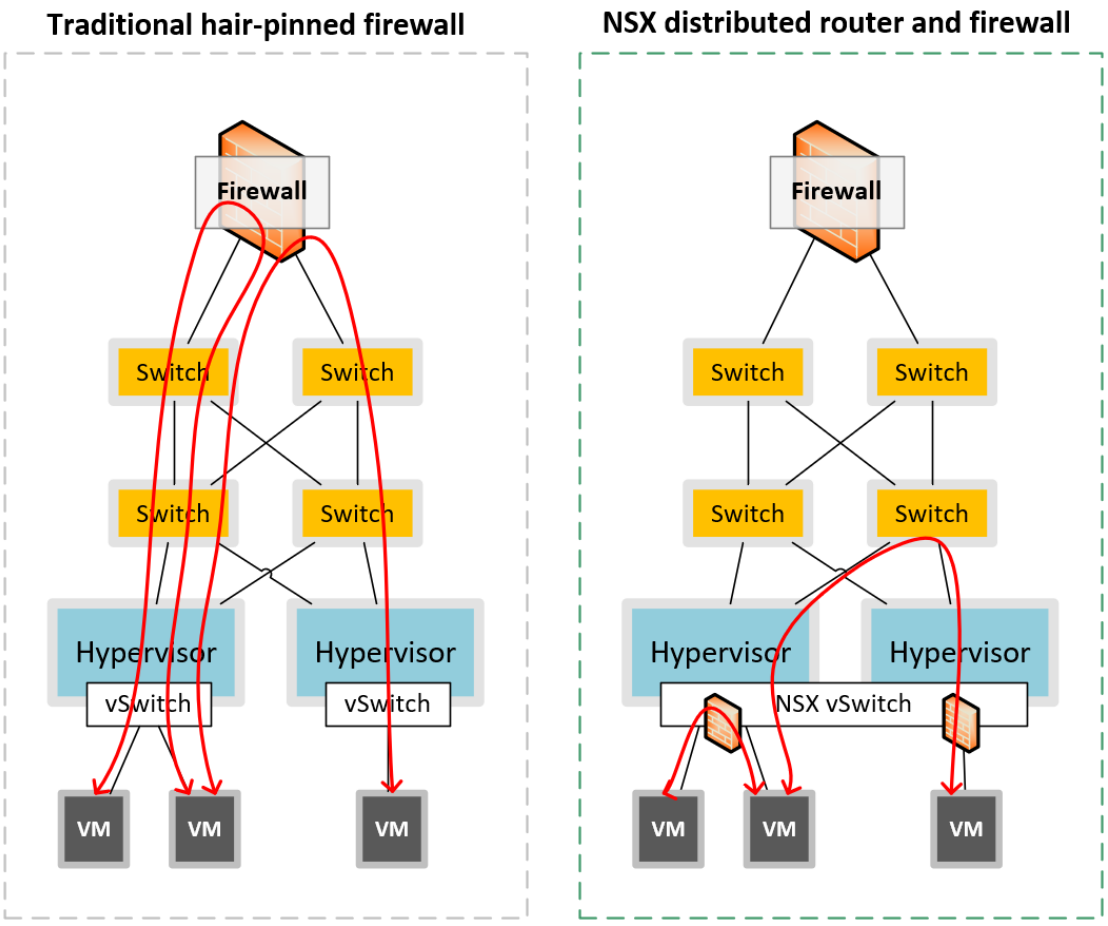


Figure 4. East-west traffic patterns for traditional and NSX implementations

As certain abstraction and simplicity is achieved with SDN solutions, they generally hide rather complex architectures and technologies under the hood. In November 2016, VMware employee, architect and owner of their highest level of certification VCDX (VMware Certified Design Expert), Chris Mutchler tweeted “I like #NSX, but sometimes I think it adds a little too much complexity for operational simplicity. #vExpert #VCDX #KeepItSimple” while illustrating a level of complexity for a topology of possible NSX implementation. (Mutchler 2016.) The rich feature set and scalability of NSX platform is powerful but it can become complex to build and manage.

4 Network segmentation

4.1 Challenges in segmentation

Providing proper segmentation is a dynamic challenge. New applications, networks, users and devices are deployed. Business, partnerships, staff and ways of working change while access to resources needs to be securily available and adapted efficiently to new status quo. The adaptation to changes needs to be operationally lightweight and least distruptive to business. (Terranova 2018.)

Application architectures have changed much from monolithic client-to-server to multiple tiered server-to-server with increased traffic volumes and for this purpose workload segmentation is ever more important. (Santana 2017, 74) The current standard practice of workload segmentation and controlling network level access involves distributing machines into separate multiple network segments with a firewall controlling access policy on the perimeter between segments. Inside the segment network hosts are able to freely traverse unless further limited or filtered by additional access controls such as local firewall or application level control mechanisms. Free movement inside or between segments would enable lateral movement by potential attackers. Most of the traffic in these segments takes place within the datacenter or as the industry calls it east-west type of traffic pattern.

Local access control relies entirely on the machine's operating system and the security software available for it. Some machines do not even have any capability for controlling access. There is also the burden and complexity of managing all these access control policies on different decentralized management interfaces. Although with added layer of automation it is possible to have better management, but it is rarely close to an optimal solution. Visibility into one's global access control policy and dynamic management does not automatically exist and is often built on top of legacy software.

For every new customer, a system or application in a shared infrastructure new instance of segmentation will likely need to be built or configured each time. This usually involves multiple teams and a change management process including the actual tasks such as reserving numbered network resources (VLAN and IP space),

configuring network devices (switches, routers, firewalls, load balancers), configuring a virtualization platform, creating a firewall policy for the segment and documenting changes. Much of this work is assigned to the network team and for most part they just replicate a configuration that already exists in the current infrastructure. Possible even a separate security team needs to be involved depending on the organization. This work needs to be scheduled, planned and perhaps executed repeatedly with each new addition. In summary – the required work and complexity for simple segmentation using traditional methods can be really costly and difficult to manage. The constructs and standards for computer networks built by the industry have become an hindrance for some security needs. New solutions have been are needed for security features such as microsegmentation of workloads.

Ultimately companies are working with systems that are fully built of different pieces of software (networking, virtualization, storage, firmware, operating systems, applications); however, in order for everything to work together for most parts industry standards have to be adhered to and compability ensured. Isolated solutions have been built with standardized compability layers between different type of solutions; however, as new technologies are stacked on top and new previously unthinkable ways of utilizing these solutions are found, new problems emerge when the underlying solutions were not made to serve these new purposes.

One notable example is the TCP/IP stack built in almost every network device, which was originally designed for end-to-end connection between computers and assumed network to be stateless and simple. Good willing cooperation between participants of the global network was assumed and security mechanisms were left out originally. Network middle boxes and functions such as firewalls, proxies, network address translators compelled to alter design principles. (Blumenthal & Clark 2001, 2-5.)

However, in theory one could do anything one wanted to with the software if one just had the capabilities for it; however, the problem often comes when one's systems need to interface outside of themselves, and this is where those standards are needed as protocol on how to talk between systems or even directly with hardware. However, inside one's own world one can do whatever one wants as long as you can translate and interface that to the outside world if required. Not many, however, have the resources or interest in programming one's own security platform

and logic; nevertheless, an organization such as Google has done this at a global scale with their in-house software defined networking solution. (Salisbury 2013)

There is clearly a need for a higher level of security, automation and better management in the environments where the workloads and critical data exist, however, at the same time most organizations do not have the resources to deal with the increasing complexity that comes with it. The advancement of technology has driven companies to certain kind of implementations where one has stacked new solutions on top of each other and laid out the infrastructure in a certain way. When building new solutions or features on top of existing systems, one most often ends up with sub-optimal end results with a plenty of complexity. Sometimes efficient problem solving requires changing one's viewpoint or doing things differently. Software based microsegmentation as a concept is a driving force for thinking differently how networks are provisioned and connectivity between systems is established. Instead of building a segmentation based on network constructs, the focus could be on the things that actually need protection, i.e. the workloads and data itself. This way of thinking is part of what VMware is attempting to promote with their NSX mindset. To think about what is possible to create and change without having to be limited to traditional constraints.

4.2 Designing secure networks

Network design is used to provide a blueprint for how the network is laid out and how the assets are protected againts a variety of network threats. The design can help block attacks altogether, mitigate some and in some cases shut down or fail appropriately in a predictable manner. Network segmentation is used as defense in depth strategy to prevent lateral movement by potential attackers inside an organization's network. Segmentation of the network is considered one of the primary tools for establishing control mechanisms in an environment as it forces traffic flows through controlled perimeter points in the network, where security controls such as firewalls and intrusion prevention systems can be implemented. In addition to security benefits, segmentation is also used for performance gains and limiting technical issues to a specific part of the network which helps when attempting to pinpoint issues. (Andress 2011.)

Network level traffic separation can be achieved in different layers of OSI model with virtualization technologies (Virtual Route Forwarding Design Guide 2008.):

1. Physical (OSI Layer 1) segmentation can be implemented by the medium, i.e. cabling or wireless radio. Virtual segmentation of shared medium can be achieved with time, frequency and wavelength division multiplexer techniques such as TDM, WDM, OFDMA etc.
2. Datalink (OSI Layer 2) segmentation always involves active network device such as Ethernet switch or router with specific capabilities. Virtualization is created with technologies such as VLANs, L2VPN/VPLS, tunneling/overlays e.g. NVGRE and VXLAN etc.
3. Network (OSI Layer 3) segmentation always involves a device with routing capabilities since it needs to be part of it. Network virtualization is done with VRFs, VPNs, generic tunneling protocols, MPLS etc.

Partitioning a network to segments limit's a potential attacker's ability to move around the network and access important assets or resources, forcing attacks to actively attempt crossing between segments and thus becoming easier to detect as they pass monitored control points. (Wagner, Sahin, Winterrose, Riordan, Pena, Hanson & Streilein 2016, 1-2.)

The approach to implementing networking segmentation and policy enforcement can vary depending on the organizations' security objectives, cost-benefit estimations and technical complexity. The most commonly used segmentation approaches are as follows:

1. **Per role** – hosts with similar function, such as databases, are grouped together and policies are defined between these groups. Communication inside the group is not controlled by the network.
2. **Per system** – hosts are grouped together based on what system or service they belong to. For example e-commerce system would be one group where all the necessary web, app and database hosts reside together. Communication inside the system group is trusted and policy is defined between system groupings.
3. **Per data classification** – hosts or whole systems are grouped based data requirements and security compliance such as PCI DSS or classification of sensitive information.
4. **Mixing all of the above** – when more segmenation is required. The resulting number of segmentation groups and complexity of security policy management will often become a problem with this method.

4.3 Traditional segmentation

Traditional segmentation relies mainly on creating choke points or hair-pinning traffic through one or more security device, usually a firewall appliance. Partitioning the network into segments is often accomplished with VLANs or physical cabling. The hair-pinning firewall does the job of an inter-VLAN routing that forwards traffic between segments. This is also sometimes referred to as router-on-a-stick configuration. (Inter-VLAN Routing.)

Sometimes Access Control Lists (ACLs) are also used on routers or L3 switches if Layer 3 routing is implemented before firewalls to break up too large segments. ACLs offer much less granularity and are complex to manage without a centralized solution. (Jaworski 2017.)

An example of simplified segmentation of a flat network with VLANs and hair-pinning firewall can be seen in Figure 5 with different VLAN memberships on access links and all-inclusive trunk between switch and firewall on shared medium.

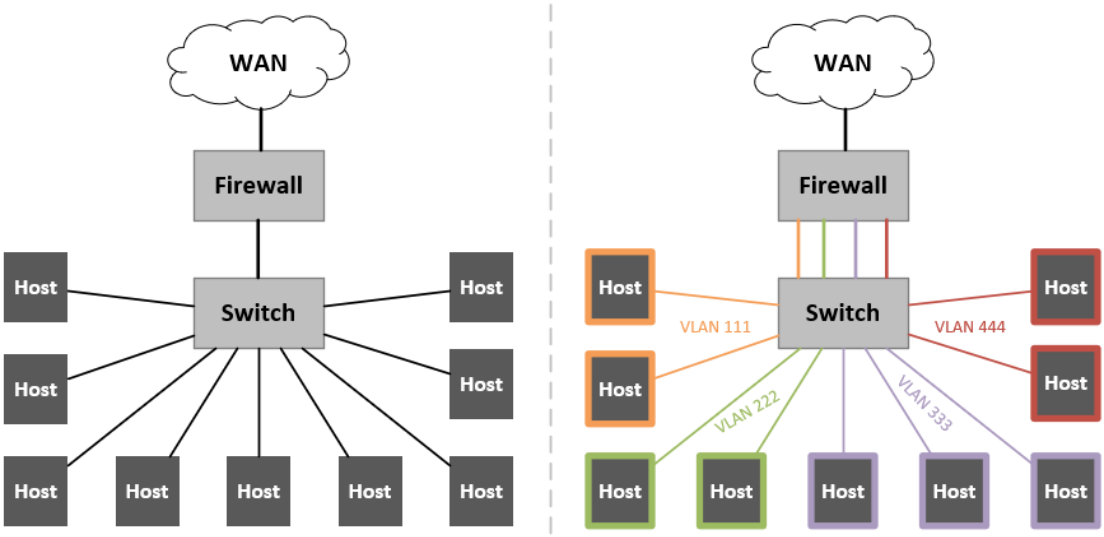


Figure 5. Flat network of hosts segmented with Virtual LANs

Private VLAN (PVLAN), also known as port isolation, is also sometimes used in switches and augment VLANs with more isolation options as detailed in Figure 6. It is a fair example of how the use of existing technology has been extended beyond its original design by software implementation. Although it is not common feature for

switches and interoperability with other switches not great. Management and scalability of PVLAN is considered poor; however, it can serve adequately in some specific cases such as hotels and offices where multiple organizations share the infrastructure and some devices such as printers.

PVLANs enforces a total isolation of hosts by using different types of configurations on switch ports that change how communication between those ports can take place, e.g. hosts behind “isolated” type ports are unable to see each other while they exist in the same VLAN and IP subnet; however, they are still able to communicate to a network gateway located behind a “promiscuous” port. (VandenBrink 2010.)

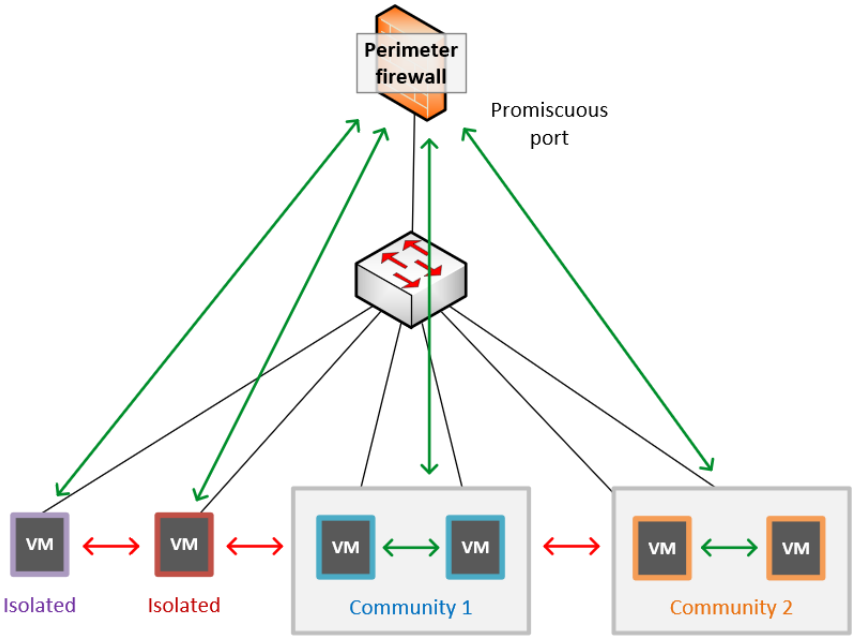


Figure 6. Private VLAN port types and allowed data flow

For environments with shared physical infrastructure, where multiple customers or tenants exist, proper segmentation is a key factor for security and compliance. At this level, virtualization of routing infrastructure is needed and traditionally has been solved with VRF. Figure 7 shows a simplified example of multi-tenant network with two tenants separated from each other using a combination of VLAN segments and VRF routing instances.

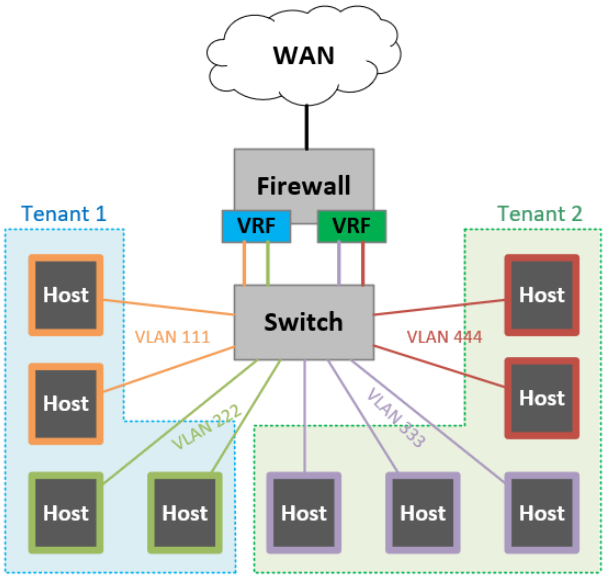


Figure 7. Multi-tenancy enabled with virtualization of routing by VRF

Taking into consideration the operational practice and steps required to create new or make changes to workload segmentation in a traditional datacenter, usually multiple tasks are partly or fully needed. Usually the request comes from a person responsible for an application or project, the requestee needs to involve organization's teams responsible for security, network and virtualization - with change management process likely in the mix. Following steps are common for most organizations when making changes to segmentation:

1. Reserving and documenting a new VLAN and IP subnet
2. Configuring VLAN into physical switches and virtual switches in hypervisors
3. Configuring Layer 3 gateway VLAN interface into firewall/router
4. Configuring firewall rules for inter-segment communication

4.4 Zero trust model

Zero trust model of information security described by Forrester Research Inc. is based on a simple philosophy: stop trusting packets and networks. The point is to phase out the idea of trusted internal and untrusted external networks. In the zero trust model all network traffic is untrusted, inspected and logged. Instead, the access

to resources is verified, secured and strictly enforced by access control. (Kindervag 2010, 2.)

Legacy networks were commonly built from outside in, with infrastructure rather than data in mind. Network professionals would begin building networks at the edge of Internet connectivity and start laying out the core infrastructure (routing protocols and switching) inwards while mostly unconcerned for placement or security of individual resources or data. The zero trust model proposes to instead first protect the data and then secondly work out how to build the networking to enable communications. Getting things connected is easy; however, making them secure is hard. (Kindervag 2010, 2-3.)

Users, devices, applications and data in today's increasingly digitally transformed world are moving out from the relative safety of enterprise networks and zones protected by perimeters and security controls - according to major cloud service provider Akamai Technologies. The "Trust, but verify" model where the user was admitted access to resources if a request came from inside the perimeter or has just user credentials is no longer considered an option where targeted and increasingly advanced threats are able to penetrate corporate perimeters. Perimeter is needed at every point where applications, data, users and devices exist. Trust and authentication is end-to-end, not in a network or location. Akamai describes the zero-trust model with the following concepts. (Terranova 2018.)

1. Never trust, always verify
2. Least privilege and default deny
3. Full visibility and inspection
4. Centralized management

Akamai states that one approach to zero-trust architecture is microsegmentation of the network which expands the traditional firewall setup with next generation firewall (NGFW) features and slicing networks into even smaller micro-segments. This is based on a vision by Forrester of a new type of device called segmentation gateway which would implement all the necessary security capability for different types of network hosts. (Terranova 2018.)

Forrester Research report lists five steps to redesigning network security for zero trust model through data centric model (Balaouras, Cunningham & Cerrato 2018, 4-9.):

1. Identification of sensitive data
    1. Sensitive data must be identified and classified using appropriate data security and control framework. This will determine how data is to be protected.
    2. Creation of microsegments is based on data sensitivity.
2. Mapping flows of sensitive data
    1. A map of data flows should exist that illustrates connections and interactions between resources, applications and users. The map will expose where sensitive data is accessed, possible weak points might exist and could help optimize flow of data.
    2. The application flow is also useful for revealing dependencies on systems and helps in planning for disaster recovery.
3. Define microsegmentation perimeters
    1. Once the application flow map around sensitive data is defined, it can be used to identify points where protection should exist and thereupon have microperimeter to protect it.
    2. Microperimeters can be enforced with physical or virtual security controls such as NGFW appliances from various vendors or software-defined microsegmentation solutions such as VMware NSX.
    3. Access policy needs to be defined for the perimeter based on the flow map. If possible, automation and dynamic rulesets should be used to help with the management and ease operations as long as they do not compromise the policy for malicious actors to exploit.
4. Monitor and analyze all traffic for malicious activity and improvement
    1. Security information can be obtained from multiple sources such as logs, networks, applications, endpoints, data loss prevention and identity access management systems.
    2. Security analytics can be carried out with various kinds of solutions from different vendors. Optimal deployment model also depends on how the company operates and does business. Cloud, on-premises or hybrid solutions exist and they have significant differences.
5. Embrace policy-driven security automation
    1. New technology is progressively more automated; however, many security tasks in organizations still rely on manual processes with slow breach detection and response times.

2. Automation should be made use of decisively with policies established by defined business principles of the organization.

Google transitioned from the usual corporate network design to a device and user centric security model where each one is authenticated individually regardless of the network location. The network itself is not trusted; however, the device and user are if authenticated and predetermined conditions match. Access control to enterprise resources and applications is fine-grained and specificly based on each user and device. Every device can thus be considered to be isolated in its individual segment which can be described as microsegmentation. Google's approach removes trust from the network and places it into user credentials and device states. Google publicly promotes this security model as BeyondCorp. The implementation for segmentation is technically very different from datacenters; nevertheless, it shows trend and value for zero-trust and microsegmentation philosophies also in the end user computing sector. (Ward & Beyer 2014, 6-7.)

Implementing infrastructure similar to Google's approach however can be too high cost and complex for most organizations with current technologies and solutions available. Although if an organization has fully transitioned to cloud apps or SaaS solutions, this can already be a built-in infrastructure service from the cloud provider and therefore allows making use of zero-trust enabling technological benefits with lower cost of implementation. Cloud platforms from Google and Microsoft already support this and also provide identity and device solutions to support services hosted outside datacenters and on-premises. However, transitioning fully to zero-trust model in on-premises or traditional datacenter can be very costly; yet, this predicament is likely to change in the future as more advances and innovations are made. (Wagner et al. 2016, 2-3.)

4.5 Microsegmentation

Microsegmentation is used to lower the level of security risk and strenghten security posture for modern data centers by utilizing following functionality to achieve the following results (Holmes 2017.):

1. **Stateful distributed firewall** enables protection on individual application level and scales effortlessly with the compute capacity from the virtualization infrastructure.
2. **Segmentation independent of topology** allows for protection agnostic of the underlying network topology.
3. **Centralized policy management** makes creation and provisioning of security policy easy from single glass-plane with straightforward API access for automation and integrations.
4. **Granular policy controls** for identifying each application by different type of static and dynamic constructs provided by network and virtualization infrastructure.
5. **Extensibility of the platform** for integrating new capability and services to provide new functionality and added value for microsegmentation.

The demand for microsegmentation comes from the need to protect hosts inside the same security zone (IP subnet, VLAN or broadcast domain) where communication directly with each other is possible without passing through a security control. (Jaworski 2017, 9-10.)

Microsegmentation can benefit organization's by (Micro-Segmentation Builds Security Into Your Data Center's DNA 2016, 4.):

1. Halting the spread of malware inside the data center by prevention of lateral movement between hosts
2. Speeding up deployment of network and security services
3. Providing higher level of automation and adaptation capability to answer changing business needs and security posture

In March 2016, the Computer Security Division of NIST made recommendations for secure virtual networks of protected virtual machines based on their findings and analysis for special publication 800-125B. One of the network segmentation related recommendations for large data center networks is to use overlay-based virtual networking techniques for scaling purposes in order to maintain segmentation guarantees. (Chandramouli 2016, 10-11.)

NIST recommends using virtual firewalls for virtualized environments with VMs running delay-sensitive or I/O intensive applications. Kernel-based filtering is

especially suggested for the latter. It is also desired that the virtual firewall is integrated with the virtualization platform instead of separate management console for unified management of multiple firewall instances. Use of higher-level abstractions such as groups in addition to basic network constructs is preferred. (Chandramouli 2016, 14-18.)

When VMware presented the microsegmentation security concept for data centers in 2014 with their NSX product, they pointed out that the idea of using network based microsegmentation for controlling lateral movement is not anything new, but until now it was not practical at all from cost-efficiency and operational viewpoints. Microsegmentation built completely with network-centric approach cannot deliver thorough security at a scale to data centers and cloud environments. As computing service operations have evolved from simple client-server model to today's dynamic, distributed and heterogeneous interconnected applications and systems, the implementation of network chokepoint enforcement model is getting even more complex and difficult to scale when new network services such as traffic steering and service-chaining are being added on top. (Cohen 2017.)

An important distinction to be made is that traditional original segmentation model based on forcing traffic through a control points or centralized chokepoint firewall is essentially only based on specifically built network constructs. Segmentation rules are purely based on information provided by the networking plane such as addressing and I/O interface. The network level processing is very objective and from a single inspection point. The firewall really has no solid information about origins or context of the traffic passing through it. It does not know from which device, user or application really constructed it; instead it relies on network constructs that have been purposefully designed to create these logical constraints to flow packets between segments. The resulting security policy depends on the trust provided by networking infrastructure. (Cohen 2017.)

When granularity is needed in building security segmentation, even modern NGFW appliances struggle as they attempt to increase granularity through building features such as application detection and user identification on top of the traditional firewall capability. A network based firewall alone can only offer limited means for implementation of security segmentation. A more granular approach requires policy

that can make decisions on information from software instead of network, thus software-centric approach is appropriate for achieving efficient microsegmentation as seen example of in Figure 8. (Cohen 2017.)

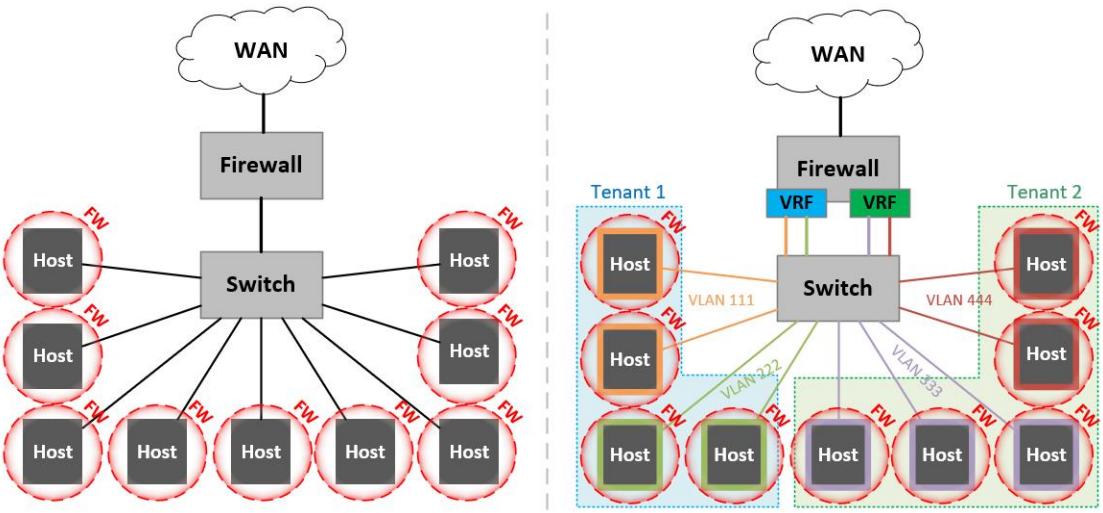


Figure 8. Microsegmentation can be used to protect both flat and VLAN & VRF enabled networks

Due to the isolating ability enabled by microsegmentation, networks can be drastically simplified and flattened to reduce complexity and management overhead without compromizing security or performance. (The VMware NSX Network Virtualization Platform 2013, 9.)

Implementing a strong security policy with microsegmentation requires a clear visibility into how applications work and communicate. This visibility is most often gained from a process called application dependency mapping and it cannot be fully built based on only monitoring network activity. The mapping should be real-time in order to stay relevant to currently enforced security policy. In real world applications there are so many dependencies and connections between different entities that manual application mapping and policy rules will be unless paired with automation and intelligent processing in order to build functioning dynamic rulesets. (Cohen 2017.)

For implementing microsegmentation in organizations, most face the obvious problem of where to begin. Microsegmentation is not something that can be just

turned on and involves more than just technological aspects. Usually several people need to get involved and planning is essential. Following list explains the common steps for starting out. (Wilmington 2019.)

1. **Understand the application**, how it works and what dependencies it has.
2. **Define methodology** for approaching security policy e.g. what constructs (IP addressing, VM objects, virtual network segments and groupings) to use to simplify management for overall security.
3. **Break the application** to tiers. Typically, applications consists of processes, containers or servers fulfilling a certain role, service or function. In the eyes of security these have different requirements for connectivity.
4. **Document** methodologies, rationales, application breakdowns for straightforward reference when examined by existing and future employees.
5. **Apply security policy** to the application. Test functionality and refine process.

4.6 Segmentation policy

In most cases of segmentation, full isolation is not wanted, but instead what is needed is access based on definable control rules also known as security policy. The segmentation method, in addition to how rulesets are created and managed, can affect efficiency and complexity of the resulting security policy. Static rules require administrators to keep them updated and handle their removal at the end of the lifecycle. Dynamic rules can at best manage themselves after the policy is defined. Automated policy management through dynamic properties can adapt to changes, such as scaling, in the compute environment or be network-agnostic in nature.

Figure 9 illustrates a typical 3-tier web application, such as e-commerce system, which is being scaled out by adding additional host for each tier to compensate for increased loads. This example can be considered a cutoff from an environment where multiple similar systems co-exist, meaning that in real world scenarios there usually exists hundreds of hosts with same roles (web, app and db) belonging to different systems.

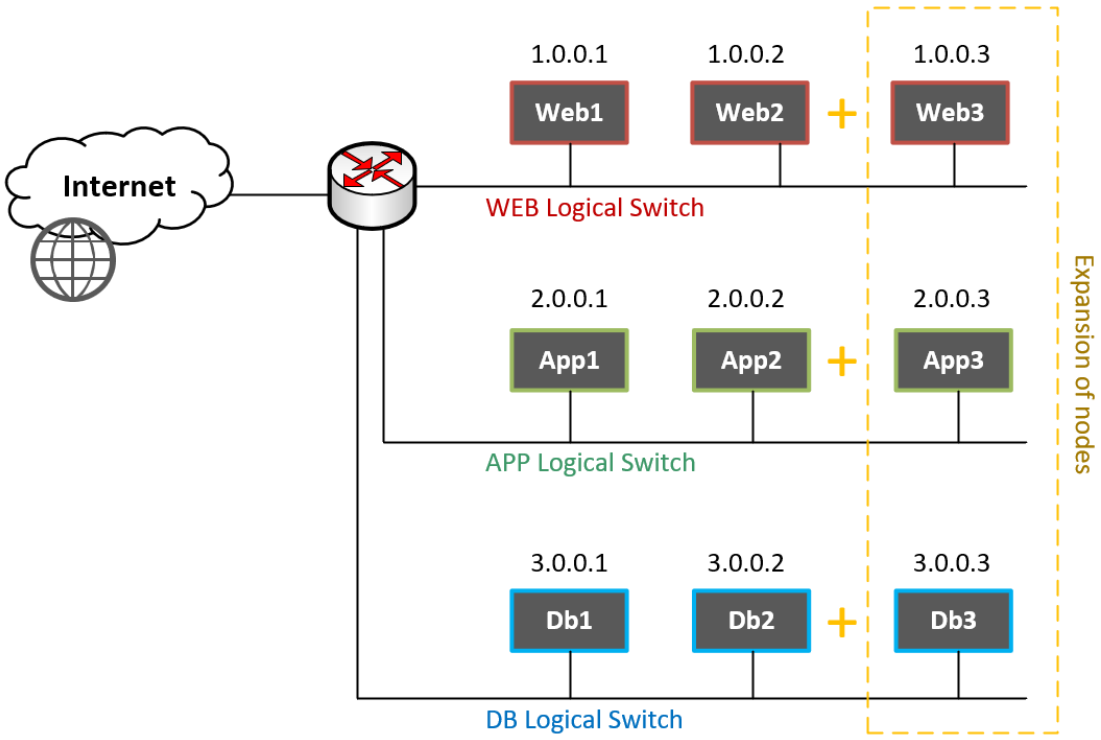


Figure 9. Network topology for simple 3-tiered internet accessible web system

Traditional security policies based on network constructs mostly employ rules using static source and destination IP addressing as seen example of in Table 1 representing collection of firewall rules for the scenario of Figure 9 with new additions highlighted. The expansion of an environment requires a policy change that modifies every relevant rule in order to add new host addresses; nevertheless, one could use subnet criteria to match the whole segment in this example; however, in real world cases this is not often possible or prudent.

Table 1. Traditional firewall ruleset using IP addressing

Source IP	Destination IP	Destination Port	Action
(Internet)	1.0.0.1 1.0.0.2 *1.0.0.3*	TCP/443 (HTTPS)	Accept
1.0.0.1 1.0.0.2 *1.0.0.3*	2.0.0.1 2.0.0.2 *2.0.0.3*	TCP/8080 (HTTP-ALT)	Accept
2.0.0.1 2.0.0.2 *2.0.0.3*	3.0.0.1 3.0.0.2 *3.0.0.3*	TCP/3306 (MySQL)	Accept
Any	Any	Any	Block

In contrast to traditional static firewall rules, microsegmentation presents a different kind of approach through dynamic object based management. Table 2 presents the firewall ruleset based on objects for the aforementioned example scenario. Although modern NGFW appliances already offer object based management, these are actually in most cases very static unless integrated with external source of information (such as virtualization platform manager) to provide dynamic group memberships to objects.

Table 2. Firewall ruleset using object groups

Source	Destination	Service	Action
(Internet)	SG_WEB	HTTPS	Accept
SG_WEB	SG_APP	HTTP-8080	Accept
SG_APP	SG_DB	MySQL	Accept
Any	Any	Any	Block

Using NSX, security groups are used as group object mechanism and can be defined as presented in Table 3, which demonstrates use of multiple different dynamic criteria available for defining which hosts belong to the security group. Web hosts are grouped by membership to a logical segment dedicated for this role. Application group is defined by the virtual machine name prefix which exposes the role as well. In many cases database hosts are attached to logical pools of compute resources, which ensure performance; this information can be utilized for security group selection criteria aswell.

Table 3. Security group definitions and criteria rules for tiered web system

Security Group	Criteria for Selection	Resultant Pool of VMs
SG_WEB	Member of WEB Logical Switch	VM-WEB1 VM-WEB2 *VM-WEB3*
SG_APP	VM name begins with “VM-APP”	VM-APP1 VM-APP2 *VM-APP3*
SG_DB	Member of “DB” Resource Pool	VM-DB1 VM-DB2 *VM-DB3*

In a real world scenario, a different solution would be more approriate; however, this is just an example to highlight possible usage. The result of using this type of policy and dynamic groups results in automated policy management for many cases such as scaling out as presented in this example. New hosts are automatically included in the groups and since the policy does not need to change, no firewall configuration changes are needed.

The possibilities for the use of security groups are many depending on the technical environment and needs. Security tags are a user managed feature to give additional context to VM objects. Figure 10 illustrates the use of security tags to define which environment hosts belong to. This can be used to create environment specific security groups. Tagging can also be carried out by an automated system such as IPS and then used to automate response from the virtualization platform, such as to quarantine workload from other hosts in case of suspected infection.

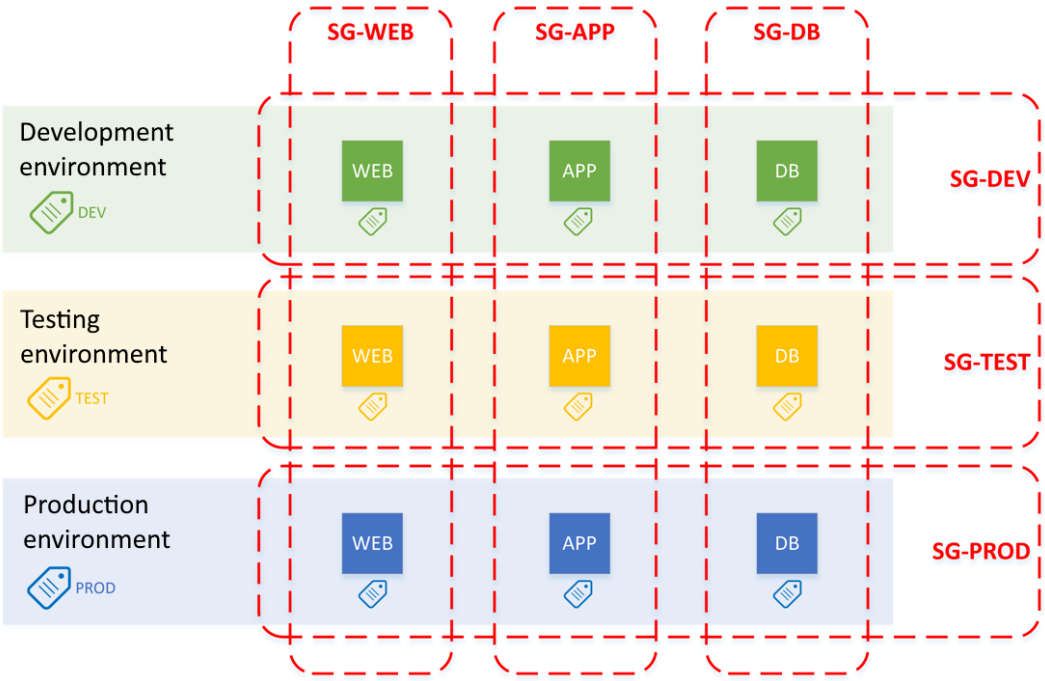


Figure 10. Example of using virtual machine tags to define security groupings

In this example, each VM is given a security tag based on the type of environment (development, testing or production) it belongs to. The same model can be applied to classify a department of an organization (HR, finance etc.), data sensitivity (for e.g. compliance), security state (e.g. unpatched system, malware detected). VMs in this example are also named according to their role (Web, App, DB). Table 4 presents an example of group definitions by just using security tag and VM name information to achieve all inclusive role and environment groups and then creating environment specific role based groups for explicit use in security policy.

Table 4. Definition of security groups and tag use for multi-environment system

Security Group	**Criteria for Selection**	**Resultant Pool of VMs**
SG_DEV_ENV	Security Tag contains ”DEV_ENV”	DEV-WEB DEV-APP DEV-DB
SG_TEST_ENV	Security Tag contains ”TEST_ENV”	TEST-WEB TEST-APP TEST-DB
SG_PROD_ENV	Security Tag contains ”PROD_ENV”	PROD-WEB PROD-APP

		PROD-DB
SG_ALL_WEB	VM name contains ”WEB”	DEV-WEB TEST-WEB PROD-WEB
SG_ALL_APP	VM name contains ”APP”	DEV-APP TEST-APP PROD-APP
SG_ALL_DB	VM name contains ”DB”	DEV-DB TEST-DB PROD-DB
SG_DEV_WEB	Member of ”SG_DEV_ENV” and “SG_ALL_WEB” Security Groups	DEV-WEB
SG_DEV_APP	Member of ”SG_DEV_ENV” and “SG_ALL_APP” Security Groups	DEV-APP
SG_DEV_DB	Member of ”SG_DEV_ENV” and “SG_ALL_DB” Security Groups	DEV-DB
SG_TEST_WEB	Member of ”SG_TEST_ENV” and “SG_ALL_WEB” Security Groups	TEST-WEB
SG_TEST_APP	Member of ”SG_TEST_ENV” and “SG_ALL_APP” Security Groups	TEST-APP
SG_TEST_DB	Member of ”SG_TEST_ENV” and “SG_ALL_DB” Security Groups	TEST-DB
SG_PROD_WEB	Member of ”SG_PROD_ENV” and “SG_ALL_WEB” Security Groups	PROD-WEB
SG_PROD_APP	Member of ”SG_PROD_ENV” and “SG_ALL_APP” Security Groups	PROD-APP
SG_PROD_DB	Member of ”SG_PROD_ENV” and “SG_ALL_DB” Security Groups	PROD-DB

In systems such as NSX, the dynamic nature of security groups also abstracts security policy definitions, which allows the use of the same policy for multiple systems. For example one could have a security policy defined where the rules for the aforementioned 3-tier system are defined as listed previously in Table 2, and that same policy could be used for each separate environment with just linked groups differing. This allows for a smaller policy footprint where multiple identical environments or cases exist of the same system or similar policy needs occur. In NSX this feature is called Service Composer and allows the separation of security policies and security groups.

5 Research

5.1 Test environment

The research conducted in this thesis seeks to examine the technical suitability from the workload perspective of the researched phenomen (NSX microsegmentation) as a replacement for a traditionally built network architecture which most organizations currently own and maintain. If the new implementation performs well and similarly compared to traditional segmentation, the conclusion can be expected to be favorable for microsegmentation.

The research consists of a test environent created in lab facilities that is mostly on par with a modern production datacenter environment with both its hardware and software. Some notable exceptions include hypervisor network cards being limited to 1 Gbps ports instead of 10 Gbps or higher, and storage system having spinning disks instead of full flash array. Two testing scenarios were constructed for which three tests were executed on.

The test environment resembles minimal and relevant part of a typical datacenter virtualization environment to research the effects of microsegmentation compared to a traditional segmentation with external firewall appliances. The environment description is simplified, and the hardware specifications cannot be detailed in this study; however, it should be considered to be high-end and very consolidated. The components and topology presented here are just one limited part of a larger lab environment that is not relevant to this study; this however, can affect the results.

The testing environment presented here consists of the following components.

1. Two firewall appliances (Juniper SRX) in active-passive cluster
2. Two core switches (HPE) in a stacked configuration
3. Two access switches (HPE) in a stacked configuration
4. Two VMware ESXi 6.7 hypervisor hosts (HPE) in a cluster
5. VM-APP with CentOS 7.7.1908 OS and necessary testing tools (detailed later) installed
6. VM-DB with CentOS 7.7.1908 OS and PostgreSQL 9.2.24 database engine installed

7. APP and DB VLANs connected to respective VMs (appendix 7) and to external firewall cluster acting as IP gateway
8. Unrouted internal VXLAN logical switch (overlay network) existing only inside both hypervisor hosts

The hypervisor cluster is enabled with HA and DRS with NSX-V 6.4.3 implemented on top. The hosts include redundant physical NIC connectivity to the switch fabric, and the network topology is tiered full-mesh.

Both virtual machines reside on separate hypervisor hosts with 100% reserved CPU and RAM resources of 2 vCPU 2.4 GHz cores and 8 GB memory respectively as seen on the output presented in Appendix 6. The resource reservation is made to ensure a more stable performance and test results.

In relation to the tests of this research, the VXLAN implementation to provide logical segment for the VMs is effectively the same as a single VLAN would provide. VXLAN has different kind of overhead and behavior, which might affect the results; however, this is intentionally included as it also represents a replacement for VLAN.

Detailed network interface and IP addressing information of the virtual machines can be seen in Appendix 1.

The full topology of the test environment is illustrated in Figure 11.

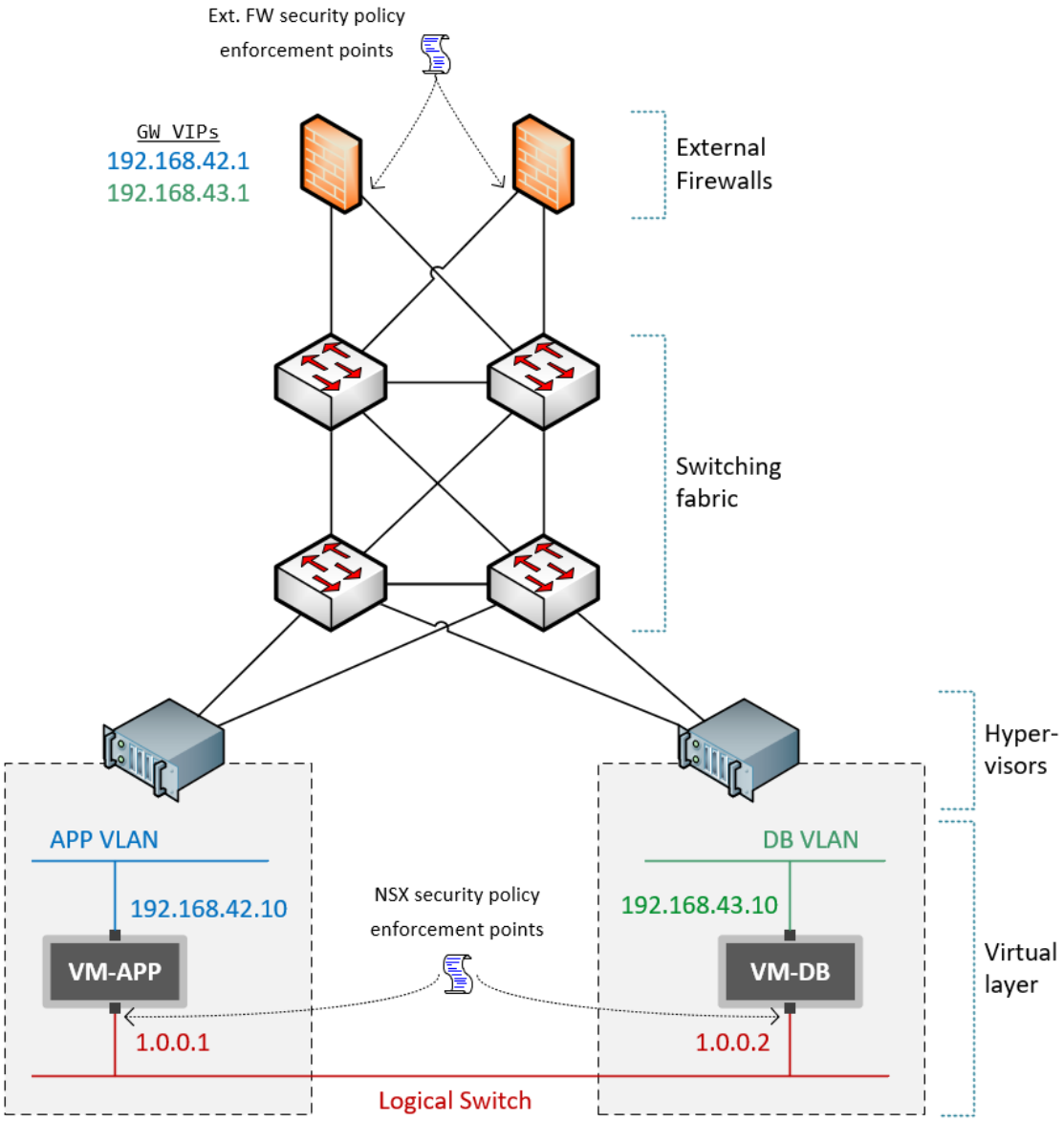


Figure 11. Test environment network topology

5.2 Test scenarios

Two small test scenarios were chosen to highlight and test the differences between the very common traditional segmentation implementation and the proposed new implementation utilizing microsegmentation without VLANs.

In terms of utilizing action research method here, the scenarios represent before and after states of taking action to implement microsegmentation architechture. The tests highlight different workload operational aspects of segmentation: effective security and performance penalties. The test participants are two virtual machines with application and database roles. The effect of segmentation implementation is tested by different methods between these two VMs.

The differing scenarios aim to present the same outcome for security state with problems of segmentation difficulty and manageability solved through new kind of implementation method.

5.2.1 Scenario A - Traditional segmentation with external firewall

The first scenario represents a traditional implementation with hosts segmented to separate VLANs based on roles and security control enabled by external firewall appliance through inter-VLAN routing. Security policy is enforced when the traffic hits firewall interfaces as illustrated in Figure 11 with policy enforcement points. The rule enforcement configuration needed in this scenario is presented in Appendix 2 in its configuration format and in Appendix 8 from the policy management interface point of view.

As the traffic between two endpoints needs to be hair-pinned through the firewall, network packets need to pass through multiple virtual and physical network components as illustrated in Figure 12. Every NIC, switch and layer 3 device such as the firewall on the data path incurs latency from processing and consumes link bandwidth. This type of implementation most often leads to the firewall being the bottleneck for net-to-net traffic passing through it.

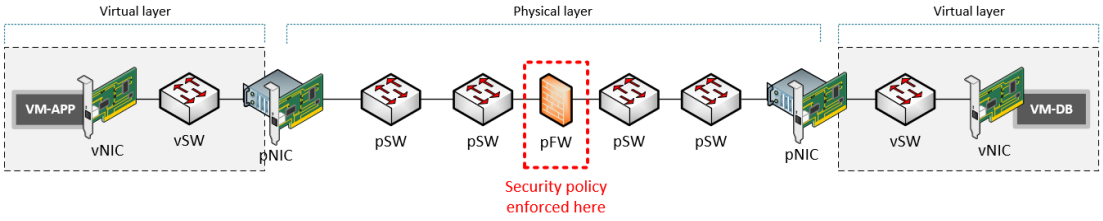


Figure 12. Layer 2 data path between VMs with traditional segmentation

5.2.2 Scenario B - NSX microsegmentation with flat logical network

The second scenario introduces an implementation where the segmentation is implemented in the virtual network layer without passing through a firewall appliance or any layer 3 device. The VMs are connected through a single unrouted logical network segment. The actual security policy is applied to hypervisor hosts and enforced at the virtual switch ports thus allowing for VM level microsegmentation and use of a simple flat network structure. In addition to microsegmentation itself, this scenario demonstrates the capability to design logical segments more efficiently and disassociate them from security zone type of thinking. In addition, the security policy, illustrated in Appendix 9, is always simple to change compared to re-designing logical network architecture when protection requirements shift.

Figure 13 illustrates the data path between the VMs, and security enforcement can be seen to take place directly after leaving the virtual NIC. As the traffic needs hairpinning through any physical appliance, its path is as direct as possible, and it only traverses through the necessary switching fabric from one hypervisor to another. This data path topology was verified with NSX network trace tool and its result is evident in appendix 10. The same data path would also apply if NSX distributed logical routing (DLR) was used as it is also kernel level functionality of the SDN platform. With DLR, if separate VXLAN logical segments or VLANs were used as in scenario A, it would make no difference in the data path presented here.

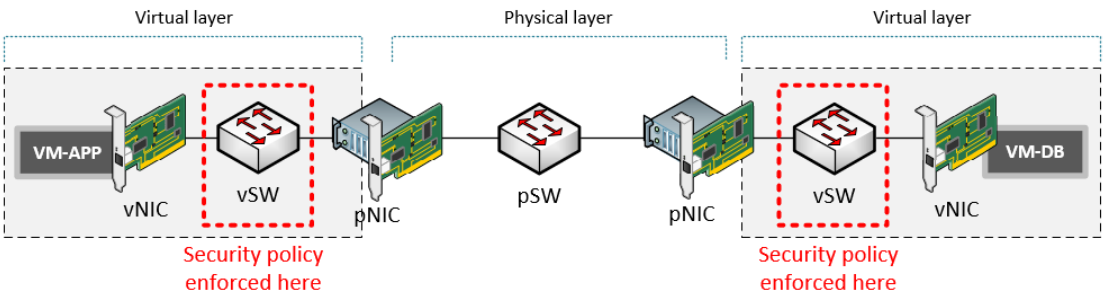


Figure 13. Layer 2 data path between VMs with virtualized networking

5.2.3 Unused alternative scenarios

Other alternative scenarios of course exist that were chosen to not be included in the tests of this research. Few of these should be examined in order to recognize their existence and understand why they were left out.

The first alternative scenario is the same as the microsegmentation scenario B, but with VMs existing on same the hypervisor host and thus packets not hitting physical network at any point. As seen Figure 14, this would always be the most optimal situation in terms of performance; however, realistically most of the time workloads are distributed across different hypervisor hosts for load balancing and high availability. However, one could force VMs to stay together on the same host. Due to this being uncommon case, it was left out.

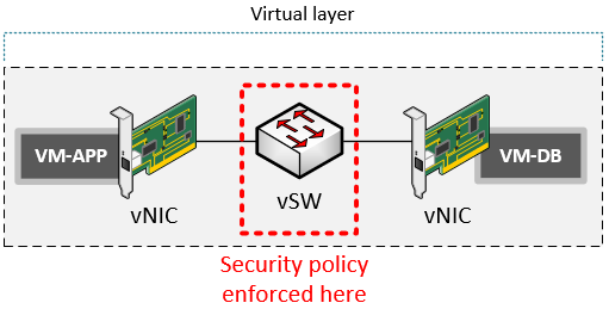


Figure 14. Layer 2 data path between VMs inside same hypervisor host

As mentioned in the Network Virtualization chapter (3.3), subnet-level virtual firewalls also have existed before microsegmentation was made possible by kernel-based virtual firewall inside virtualization hypervisor. They still serve a purpose and functionality; however, one can predict their number of implementations to decrease in infrastructure areas where a transformation to SDN takes place, as their performance has been sub-par compared to kernel-based hypervisor functions.

Figure 15 depicts a scenario where virtual firewall appliance VM is deployed on the third hypervisor host. Usually and according to best practices, virtualized networking appliances are to be deployed on separate hypervisor cluster/host other than the ones hosting general VMs. In this scenario, a traditional segmentation with VLANs is

enabled; however, physical network hops are decreased compared to scenario A. However, unnecessary hair-pinning of traffic and numerous logical hops still take place resulting usually in a performance penalty. This is a more common implementation used as a compromise to allow firewall scalability and multi-tenancy by virtualization for service providers who have not yet jumped to SDN or been unable, unwilling to utilize it for this purpose or implemented these before SDN was mature enough. It should be noted that even with microsegmentation capability on NSX, this type of virtual firewall appliance functionality is still provided and currently necessary for other layer 3+ functionality such as routing, NAT, load balancing, VPN, DHCP and DNS relay. So realistically one would not be able to jump to solely use kernel-based firewall in one's datacenter; at least yet.

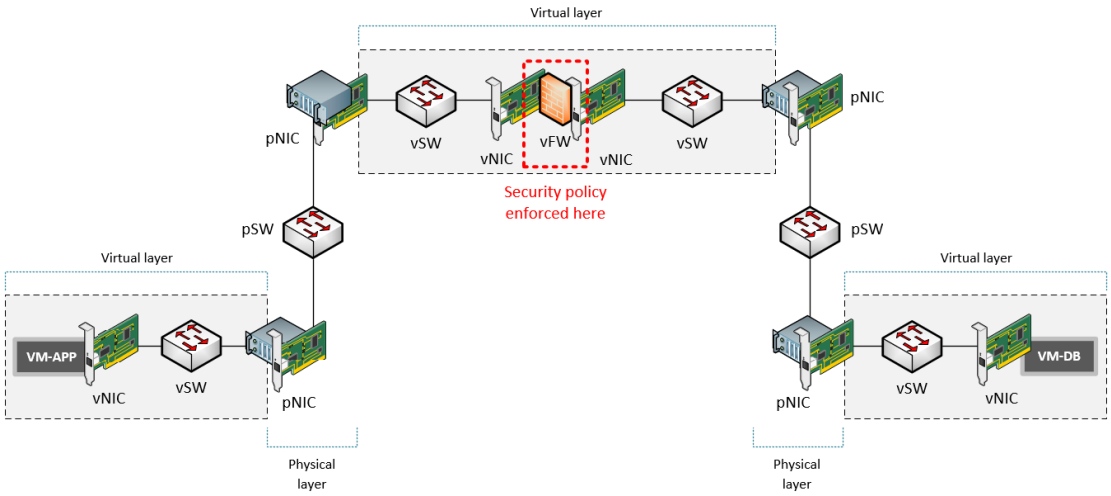


Figure 15. Layer 2 data path between VMs with virtual firewall

Virtual firewall could also be used inside the same hypervisor host as seen in Figure 16, and that would be more optimal in terms of performance; yet, realistically a very unlikely scenario.

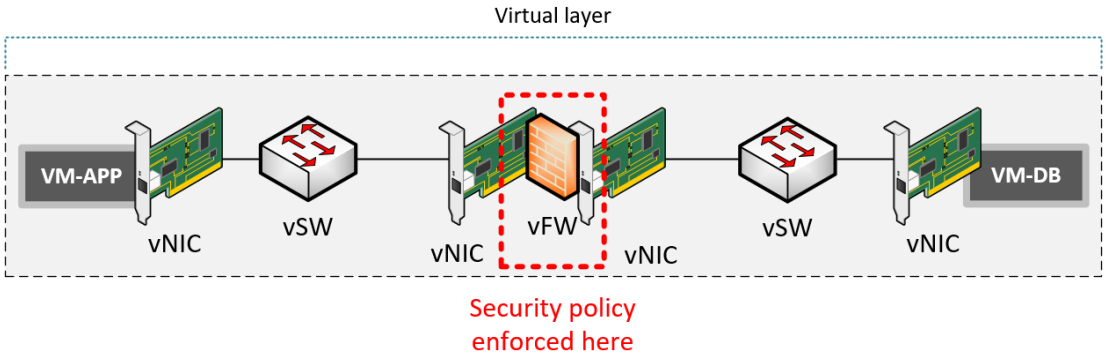


Figure 16. Layer 2 path between VMs and through virtual firewall on same host

5.3 Test methods

5.3.1 Method 1 - Security control

In order to validate segmentation security controls in place, port scanning is used to demonstrate that only explicitly allowed connections between VMs are possible. The purpose of this test was to establish in practice what has been covered so far in theory. The main purpose of segmentation after all is to provide security for protected systems. The tool for this port scan test is the network scanner and mapping software *nmap*.

Nmap is used to perform basic TCP SYN port scan on SSH (22), HTTP (80), HTTPS (443) and PostgreSQL (5432) ports. Only the latter should be open, and rest closed. The reported result is simply PASS/FAIL for this test based on if the aforementioned conditions are met thoroughly.

5.3.2 Method 2 - Network performance

To explore network performance effects of segmentation implementation, specialized performance software is used to benchmark two very important metrics affected by network design and implementation. *Qperf* tool is used to measure both TCP and UDP bandwidth and latencies between the nodes in both of the aforementioned scenarios. Each test is run sequentially five times with one minute runtime. The individual tests are as follows.

1. **TCP bandwidth -** 64 KiB message size
2. **TCP latency -** one byte message size
3. **UDP bandwidth -** 1400 byte message size
4. **UDP latency -** one byte message size

The results are reported in MB/s for bandwidth throughput and in microseconds (µs) for communication latency. Some firewall exceptions are temporarily added for this test to be completed.

5.3.3 Method 3 - Application performance

The database server will be hosting a madeup test database. Database benchmarking software *Sysbench* is run from application server to simulate different kind of workloads and to illustrate any possible application level performance differences between the two segmentation implementations.

Sysbench has several tests available for benchmarking purposes and of these five are used to intensively emulate common transaction operations (select, delete, insert, update) separately and lastly with an OLTP test profile which resembles common real-life usage. The testing tool in this method simulates a highly optimized application as there is no application processing and database is directly hit.

In relational data management Online Transaction Processing (OLTP) refers to how most businesses facilitate information processing in transactional applications such as online client-server based systems where transactions need to be atomic and consistent. OLTP applications have a high throughput and are intensive in insert/update operations. (OLTP 2019.)

Five different benchmarking profiles (SELECT, DELETE, INSERT, UPDATE INDEX, OLTP) are run for five minutes each with 32 processing threads. The target database has 24 tables with 100000 table size limit. Benchmarking statistics are set to exclude first 30 seconds of runtime.

Essential result metrics are transactions throughput per second and average latency from the whole run. Transaction throughput is an obvious performance metric which

could be affected by segmentation implementation. It is not uncommon to have internal direct network between application and database servers to optimize performance by bypassing routing and firewall processing. Latency is another metric that affects how application usage is perceived by the end-user but will be considered as secondary here. Low database latency does not really matter if throughput is catastrophically bad.

The whole test suite is run at least couple of times during different occasions in order to invalidate any skewed results due to incidental technical circumstances, such as network congestion or processing spikes, in the lab environment that might affect the results. However, results are only picked from a single complete run of all tests executed on the same occasion sequentially.

5.4 Test results

The summary of results from each test method is shown in Table 5. The actual command lines and full raw output are available in Appendixes 3-5 from each test method respectively.

Port scanning with nmap showed only database port open and rest closed in both scenarios.

Table 5. Summation of results from all test methods

Test	**Results - Scenario A**	**Results - Scenario B**
nmap (PASS/FAIL)	PASS	PASS
TCP-BW (5 runs MB/s)	115/115/115/115/115	112/112/112/112/112
TCP-LAT (5 runs avg µs)	168/181/163/172/171	166/165/174/171/172
UDP-BW (5 runs MB/s)	116/116/117/112/116	113/113/112/112/113
UDP-LAT (5 runs avg µs)	165/172/165/171/167	162/159/163/157/153
SB - SELECT (trans/s)	17198	17190
SB - DELETE (trans/s)	17010	17500
SB - INSERT (trans/s)	4779	4878
SB - UPDATE INDEX (trans/s)	15495	15816

SB - OLTP (trans/s)	623	625
SB - SELECT (min/avg/max ms)	0.32/1.86/1113.35	0.29/1.86/217.89
SB - DELETE (min/avg/max ms)	0.29/1.88/270.57	0.29/1.83/398.56
SB - INSERT (min/avg/max ms)	0.76/6.69/199.40	0.82/6.55/189.53
SB - UPDATE (min/avg/max ms)	0.31/2.06/295.74	0.28/2.02/229.42
SB - OLTP (min/avg/max ms)	15.02/15.35/416.00	13.22/51.17/346.27

Network performance tests showed near line rate transfer speeds for the bandwidth as illustrated in Figure 17.

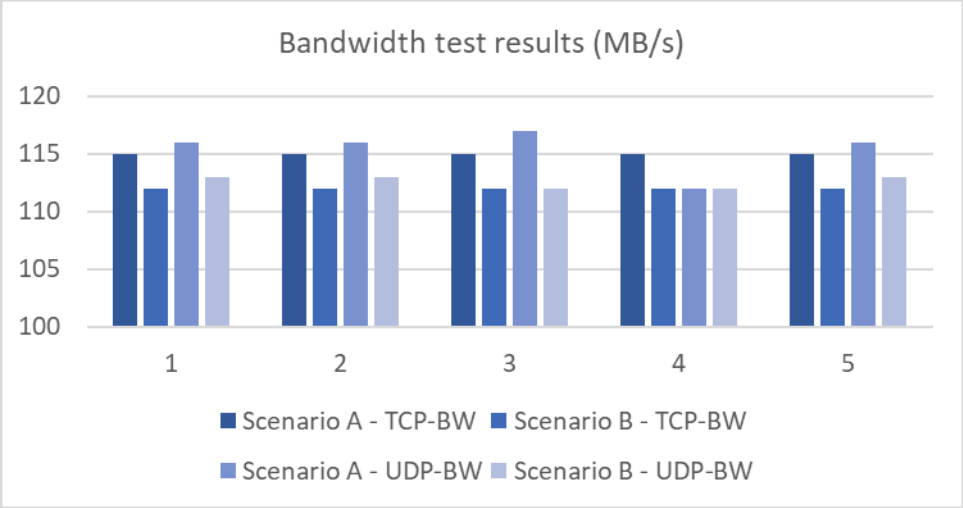


Figure 17. TCP & UDP bandwidth test results

The combined average latency ranged from 153 to 181 microseconds as seen in Figure 18.

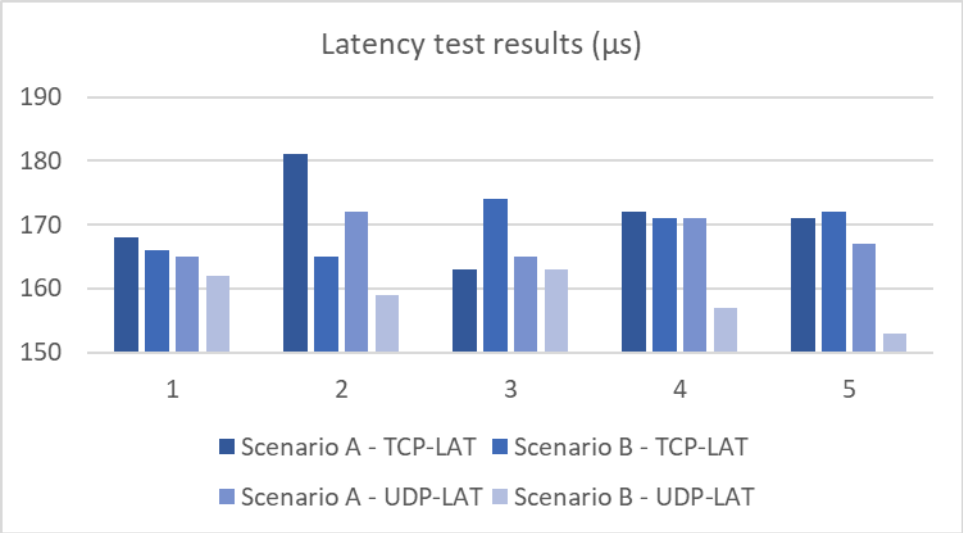


Figure 18. TCP & UDP latency test results

Benchmarking the database throughput performance over network resulted in results as charted in Figure 19.

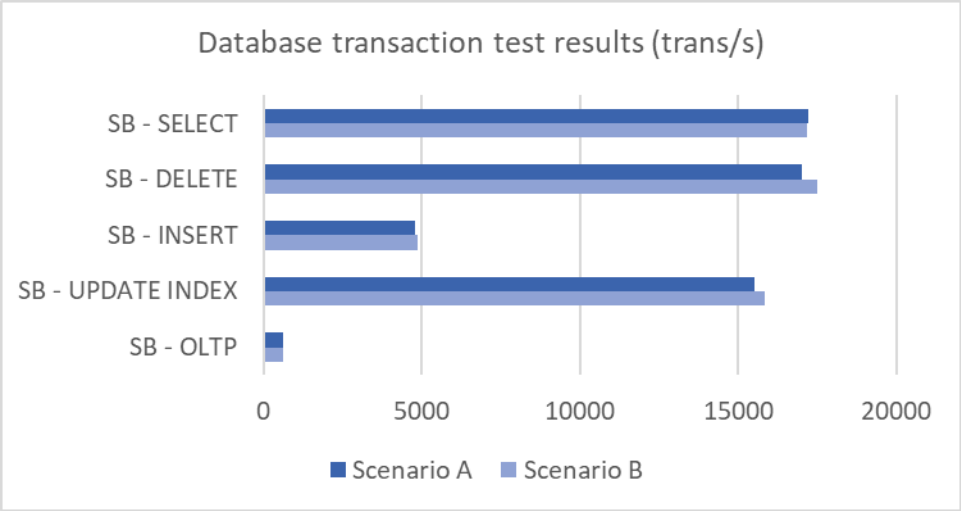


Figure 19. Database benchmarking throughput results

Latency results from database benchmark is seen in Figure 20.

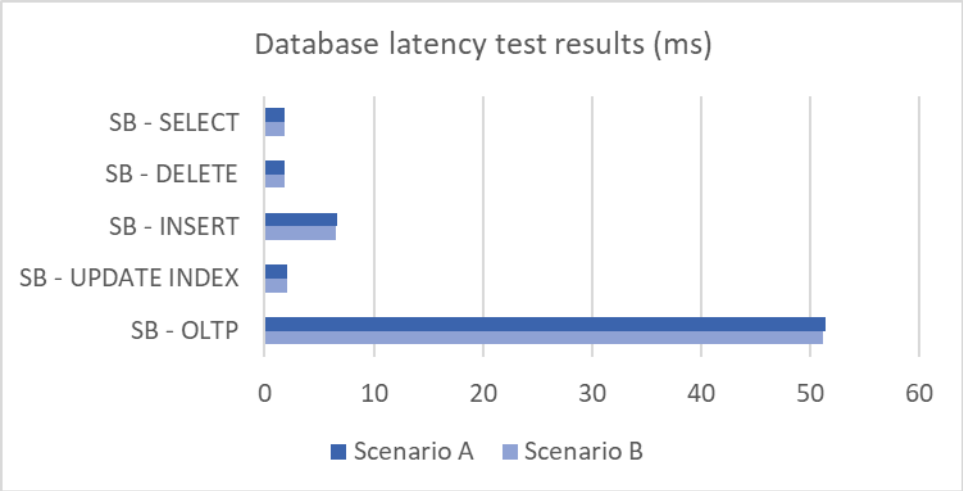


Figure 20. Database benchmark latency results

5.5 Analysis of results

It should be noted that testing was not conducted in time of network congestion and the firewall ruleset was not large like in a typical enterprise but these were not the targets of investigation in this research. Obviously multiple similar systems often exist in the shared environment, and high resource usage can cause problems related to each segmentation design. Most of the performance issues, however, raise from the capacity or processing bottlenecks common in centralized single active path designs of traditional networks.

Nmap port scanning results are as expected, and there is not much to analyze other than to determine that the policy applied is in effect and segmentation is happening.

The traditional scenario A resulted in being a slightly faster network throughput with 2.8 MB/s of difference in UDP and 3.0 MB/s in TCP. This is comparatively a rather negligible difference between the scenarios and might vary more with different kind of network usage and congestion cases. Direct hardware processing is also almost always faster than with software abstraction layers and emulation in play. Results are as expected from unoptimized sender/receiver nodes with protocol overhead.

The microsegmentation (with VXLAN logical segment) scenario B appeared to be faster only by 1.4 microseconds with TCP but made 9.2 microseconds of difference in

UDP latency test. The difference of latency between these scenarios is not remarkable for most usage scenarios but shows that shorter data path with less devices to pass through results in smaller delay in transmission.

The database throughput benchmark showed very similar results between the both scenarios, as combined transactions per second difference averaged only at around 1.4 % in favor of the microsegmentation scenario and thus should be considered rather inconsequential.

As illustrated previously on Figure 28, the reported latency of each performance test followed the same pattern as throughput with only 1.4 % of difference in combined average; however, in this case in favor of traditional segmentation scenario. This can be expected as higher processing throughput usually incurs more latency. Overall, this is also a rather trivial variation, and not much consideration should be put into the latency if it is within reasonable limits while good throughput is maintained.

The use of normalized bulk benchmark profiles does not represent actual real world usage, but acts as raw performance metric in this case. The individual firewalls under each testing scenario were also not under realistic policy configuration for throughput testing; however, firewall rule processing was not under investigation here but rather the segmentation implementation architecturally.

Exploring the quantitative results here in relation to the research question indicate that no meaningful performance loss incurred and segmentation was effective. Thus objectively operational and security effectiveness is assured from workload perspective. How the phenomenon affects processes and people, was not part of this, but will be discussed in conclusions based on theoretical basis and unstructured undocumented behavioral observations.

6 Conclusions

6.1 Outcome of results

The results portray that the microsegmentation implementation in this case does not have a meaningful effect on workload performance, and software based solution is quite on-par with hardware based appliance. The performance problems usually rise from bottlenecks in network design, to which the distributed nature of microsegmentation implemented by NSX offers software based scalability and spreading security control processing on the whole virtualization platform instead of a designated centralized point of network. The increasing distributed nature of workloads in cloud environments is better supported with similarly scalable and distributed platform security services.

Microsegmentation implemented with a hypervisor-level kernel-based firewall is a worthy option for securing workloads on virtualization platforms and clouds. It enables dynamic load distribution of firewalling, more options for policy definition, integrated into the platform, flexibility to adapt to changes, security boundaries independent of network segments and design, workload mobility and in some cases migration from traditional segmentation nondistruptively without having to change underlying infrastructure.

The test scenario for microsegmentation implementation demonstrates that to provide security into the environment and workloads, there is no absolute need for external firewall solutions. NSX and microsegmentation succesfully reduce the need for physical networking hardware and functionality provided in those thus cutting costs; however, at the same time NSX licencing itself could get pricy for some organizations. Organizations need to evaluate for themselves how they can best utilize NSX and if they can get operational benefits, needed security improvements, equipment cost and man hour savings from moving to software defined infrastructure. If the feature-set from NSX is not fully utilized, one is likely paying both network hardware vendor or virtualization vendor partly for similar features; just implemented and operated differently.

However, for smaller deployments and static environments, the benefit of microsegmentation and NSX is even more questionable from the cost perspective. Traditional segmentation design can still adequately serve the needs for many. The situation will likely change when technology advances and different microsegmentation enabling solutions become more common and accessible.

Answering directly the research question, an organization can see multiple benefits for security and operations as described in conclusions here, when NSX is used to implement microsegmentation. However, it does not come without a cost from different factors that should be considered before decision to implement is made.

6.2 Promising potential

NSX allows security policy to be defined from the same unified virtualization platform management interface and enables shifting system related policy management work from network or security administrators to virtualization or system administrators, who usually are more familiar and responsible with the systems affected by most of the policy definitions affect. This allows for more self-service operation of granular policy management while keeping big picture network zone policy in the hands of network admins.

In terms of hardware utilization, the software based approach to network services allows for better use of virtualization capacity and scalability through it. Also, without need to hair-pin traffic through a external hardware appliance, the path between endpoint systems is shorter and more optimal. Scaling out virtualization platforms with common server x86 hardware and fast network cards supporting speeds such as 40 and 100 Gbps is often more cost-effective than scaling physical high throughput firewall appliances.

Microsegmentation enabled by NSX allows creation of dynamic security policy that can be quickly adapted to new needs without changes to existing network infrastructure, e.g. segmentation is created logically in virtualized network platform, not necessarily in creation of logical network segments such as VLAN and subnet pairs that require more configuration work.

The industry has for a long time worked with the idea that network boundaries are synonym to security boundaries and it can be difficult to change this mindset of people. Change of mindset is critical in reaping the full benefits of the capabilities which these new technologies offer. The cultural shift always drags behind the technological one. With many new momentary innovations and gimmicks introduced almost on a weekly basis in the field, it's difficult for most industry professionals to keep up and to grasp true value behind things such as network virtualization - especially as it has been accustomed to characterize multitude of different networking technologies used for decades.

Microsegmentation is only one of the many features of NSX and other SDN solutions that will change how networking is carried out in the industry during the following years as the level of adoption rises and the technology advances. What SDN and similarly tilting software based solutions that can be developed more rapidly in favor of pre-defined hardware appliances allow is a game changer for the industry. The peak of global network transformation is still yet to come; however, some forward-looking organizations can already benefit from these solutions as early adopters.

6.3 Organizational readiness

Microsegmentation is not something an organization can just simply slap on the existing infrastructure and reap the benefits. The implementation work should not be underestimated if the organization plans to actually make use of the technology benefitting the business. The organization might not even be very suitable or have the readiness required to capitalize from it.

The author has introduced a few recommendations from his personal experience and acquired knowledge for organizations considering utilizing microsegmentation, VMware NSX or other SDN solutions. Firstly the author would advocate caution and suggest looking at the technical environment and how the business operates currently and in the near future. It is not definitively guaranteed that one's organization can really make the best use of these solutions even if an technical admin can see some value in some features provided, especially if one believes the

fancy marketing. The organization should also evaluate its readiness for network virtualization transformation from the technical point of view.

The following paragraphs discuss matters to consider in the current and future state.

Number and hardware configuration of virtualization hypervisors

NSX licencing is mostly based on number of the CPU sockets. There might also be options for VM-quantity based pricing model. Assuming the usual CPU based licensing, it is not very cost-efficient to licence clusters with low core counts and high socket quantities. Instead, highly consolidated and powerful hypervisor hosts would have significantly better return on investment for the organization. The price tag for licencing virtualization related software from everything on top can easily outweigh the price of the underlying hardware. Scalability is also better supported with balanced hypervisor hosts and common server hardware when dealing with SDN solutions.

Aggregate site topology and L2/L3 fabric connectivity

The relevant layout of the entire combined operational virtual platform, including all clusters, sites and clouds, will affect how well the technology in question can be utilized. NSX and other SDNs can be used to create connectivity between on-premises sites and cloud platforms through overlay networking or L2VPN-solutions. Microsegmentation in itself might not provide much value in cross-site scenarios except in special cases, but can provide global security management across managed multi-site virtual platforms.

Segmentation, multi-tenancy, compliance requirements

Business needs and requirements drive policy definition for end-user applications and systems deployed on the platform. The more granular the segmentation needs are, the more benefit can be gained from microsegmentation.

Static/dynamic nature of security state, networking, systems/apps

If the environment is very static and does not change much after the initial deployment, there is not much benefit from agility gained through the dynamic nature of SDN solution. The situation is different if adaptability is needed and the

environment is targeted with higher rate of changes. There are also cases where the environment can be static and does not need to adapt later but will need SDN to deliver some features critical for development of the business.

Will it replace any existing solution or eliminate the need for it?

One's organization might benefit from reducing external solutions or simplifying the network architecture. If the virtualization platform uses shared networking infrastructure such as switching fabric and firewalls with other separate platforms, one might not be able to cut these from the environment and could end up with redundant solutions taking unnecessary rack space and costing money. Additional compute capacity is needed when implementing software based features such as microsegmentation on the virtualization platform. Best practices with NSX dictate that separate network virtualization cluster should be implemented, resulting in more used rack space.

Will it reduce work hours needed and time to complete changes?

If one believes the marketing, then yes. In the worst case, it might have the opposite effect. It is another complex technology stack requiring e.g. separate training, know-how, maintenance. If the environment and processes can fully utilize the technology, then the organization can see benefit from it. Then it should be evaluated if the gained benefit is worth the price in the long run. Can for example the time of specialists and engineers be freed-up for other tasks after the implementation? Delegating application specific firewall management to virtualization layer can be very beneficial and provide agility to organizations.

Identity and role based access use cases

With microsegmentation and NSX, organizations can see benefits if the environment includes multi-role end-user virtual desktops or applications (eg. VDI, RDSH, Citrix). Microsegmentation can be utilized in identity firewall way to create user-specific access policies and reduce segmentation complexity provided in traditional networking segmentation. It can also reduce the number of virtual machine templates and simplify provisioning when one does not need to have differentiating pools of machines connected to different network security segments. Access to

resources can be granted when a user is logged into desktop or application and limited for the duration of session.

Organizational structure

Can the organizational boundaries be adapted to best utilize new technology in collaboration and shared goals and responsibilities? These solutions step on multiple teams; at very least networking, security and virtualization people are involved. Convergence of responsibilities and skills follows as solutions become more concentrated. Strong silos between departments ideally should disappear for optimal outcome. More agile and streamlined environment with less inherited infrastructure can profit from microsegmentation usage.

Mindset and call to action

What microsegmentation and SDN solutions offer are vastly different from the traditional type of thinking of how an infrastructure is built and security managed for systems. Infrastructure, services and applications can be built and protected in very new ways not limited by constraints of traditional hardware-centric model.

Technological debt and capability for transformation

What kind of legacy infrastructure and systems does one need to support and how easily can they be transformed into new software defined environment? Adapting to a new type of security policy management and networking infrastructure can take its toll. Systems will likely require some kind of migration projects. Migrating a large security policy from an old firewall to a new distributed firewall properly can take a great amount of work without a clear business benefit for the organization.

Early adopter's risk

The SDN solutions are still under very active development with a plenty of issues still unclear and changing. The NSX-V product investigated here has already become superseded as its vastly different brother product NSX-T has reached feature parity and will be the dominant recommended SDN product going forward in case of VMware. Existing customers of NSX-V are facing a migration project to NSX-T as support will end in a few years and feature development ceases.

Understanding of one's systems and applications

Segmentation is efficiently built when one understands the environment including connections and dependencies, and what kind of groupings or labels can be built for the creation of security policy. Labeling can utilize the existing information from CMDB, application flow mappings or other information systems.

Unfamiliarity with microsegmentation, NSX and SDN solutions makes them easy for misplaced decision making. It is a new transformative technology providing very enticing business benefits; however, with high price point for many and operational caveats not so well marketed. Even the majority of technical professionals are unaware of these concepts including how, when and where these should be appropriate to use. Proof of concept tests with these kind of technologies should be considered to be mandatory before making these leaps. Ultimately, an organization should decide if it is suitable and able to adapt to technological changes of such as dynamic and agile nature of microsegmentation or will it end up as unmanaged and unplanned implementation with an increased burden for operating the business.

6.4 Reflection and further research

The theoretical basis in this thesis was based on multiple different online sources authored by respected industry veterans and organizations. Plenty of these information sources are expected to be biased and influenced as true motivations of authors cannot be ascertained. The search for information sources was heavily geared towards technical testimonials rather than marketing materials. Businesses that benefit financially from adopting products such as NSX are obviously motivated to invest in marketing and hyping up the technology.

The credibility of what was claimed by the sources used in this research was considered to be accurate based on the experiences of the author while investigating the subject and from previous knowledge acquired professionally in the industry. However, this perceived insight by the author into the researched subject is not comprehensive and the author at this point of time has not yet acquired experience from use of microsegmentation implemented by NSX in production usage which would en-

tail more assured perspective into how it affects an organization. However, the author has long-term experience from designing and managing similar segmentation implementations in production environments where benefits of NSX microsegmentation would have been considered very advantageous.

The operational benefits would have been better brought up by exploring the technology when it was implemented into production environment and was in actual use where change management and other operational procedures would be used for some period of time. Afterwards personnel could have been interviewed with how it has affected their procedures and workflows. Organizational perspective into the subject could be retrieved from people in leadership positions.

As microsegmentation and SDN solutions enables one to build security policies very differently compared to current traditional approach, it could be worth researching more into. Especially as further advances are made in software-based workload and network security, one can expect to have even more options available on how workload security can be defined and policy strategy approached. Also similarly researching organizational suitability and benefits of microsegmentation implementations by other SDN solutions currently available would be beneficial for the industry.

The NSX-T product from VMware will likely offer more elements for future research from different viewpoints as it integrates deeper into the network infrastructure and not just virtualization layer. Also for further security research, the recently released NSX-T 3.0 includes interesting security features such as distributed IDS/IPS, microsegmentation (with DFW) for physical Linux and Windows workloads and integration with Kubernetes containers. (Mahajan 2020)

References

Andress, J. 2011. The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice. Accessed on 9 August 2019. Retrieved from https://janet.finna.fi/PrimoRecord/pci.els_bookB978-1-59749-653-7.00008-6

Balaouras, S., Cunningham, C., & Cerrato, P. 2018. Five Steps To A Zero Trust Network. Accessed on 28 August 2019. Retrieved from https://www.optiv.com/sites/default/files/2019-01/Forrester%20Five%20Steps%20To%20A%20Zero%20Trust%20Network%20Oct%202018.pdf

Blumenthal, M., & Clark, D. 2001. Rethinking the design of the Internet: The end to end arguments vs. the brave new world. Accessed on 17 February 2020. Retrieved from https://www.csd.uoc.gr/~hy558/papers/Rethinking_2001.pdf

Chandramouli, R. 2016. Secure Virtual Network Configuration for Virtual Machine (VM) Protection. Accessed on 21 September 2019. Retrieved from http://dx.doi.org/10.6028/NIST.SP.800-125B

Cisco Networking Academy's Introduction to VLANs. 2014. Accessed on 6 September 2019. Retrieved from http://www.ciscopress.com/articles/article.asp?p=2181837&seqNum=4

Cohen, A. 2017. The Truth About Micro-Segmentation (Part 2). Accessed on 9 August 2019. Retrieved from https://www.securityweek.com/truth-about-micro-segmentation-part-2

Cohen, A. 2017. The Truth About Micro-Segmentation: It's Not About the Network (Part 1). Accessed on 9 August 2019. Retrieved from https://www.securityweek.com/truth-about-micro-segmentation-its-not-about-network-part-1

Dawoud, W., Takouna, I., & Meinel C. 2010. Infrastructure as a service security: Challenges and solutions. Accessed on 9 August 2019. Retrieved from https://janet.finna.fi/PrimoRecord/pci.ieee_s5461732

Gartner Forecasts Worldwide Information Security Spending to Exceed $124 Billion in 2019. 2018. Accessed on 8 August 2019. Retrieved from https://www.gartner.com/en/newsroom/press-releases/2018-08-15-gartner-forecasts-worldwide-information-security-spending-to-exceed-124-billion-in-2019

Hagland, M. 2018. A New Era in Network Segmentation? Accessed on 27 August 2019. Retrieved from https://www.hcinnovationgroup.com/cybersecurity/article/13029865/a-new-era-in-network-segmentation

Holmes, W. 2017. VMware NSX® Micro-segmentation Day 1. Accessed on 20 June 2019. Retrieved from https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/nsx/vmware-nsx-microsegmentation.pdf

Inter-VLAN Routing. N.d. Accessed on 9 September 2019. Retrieved from https://www.ccnablog.com/inter-vlan-routing/

Internet Organised Crime Threat Assessment (IOCTA). 2018. Accessed on 8 August 2019. Retrieved from https://www.europol.europa.eu/internet-organised-crime-threat-assessment-2018

Internet Organised Crime Threat Assessment (IOCTA). 2019. Accessed on 18 February 2020. Retrieved from https://www.europol.europa.eu/activities-services/main-reports/internet-organised-crime-threat-assessment-iocta-2019

Jang-Jaccard, J., & Nepal S. 2014. A survey of emerging threats in cybersecurity. Accessed on 8 August 2019. Retrieved from https://www.sciencedirect.com/science/article/pii/S0022000014000178

Jaworski, S. 2017. Does Network Micro-segmentation Provide Additional Security? Accessed on 9 August 2019. Retrieved from https://www.sans.org/reading-room/whitepapers/networksecurity/paper/38030

Kindervag, J. 2010. Build Security Into Your Network's DNA: The Zero Trust Network Architecture. Accessed on 28 August 2019. Retrieved from http://www.virtualstarmedia.com/downloads/Forrester_zero_trust_DNA.pdf

Kirkpatrick, K. 2013. Software Defined Networking. Accessed on 6 September 2019. Retrieved from https://janet.finna.fi/PrimoRecord/pci.acm2500473

Korolov, M. 2019. What We Can Learn from the Ransomware Attack That Crippled Norsk Hydro. Accessed on 9 October 2019. Retrieved from https://www.datacenterknowledge.com/security/what-we-can-learn-ransomware-attack-crippled-norsk-hydro

Kothari, C.R. 2004. Research Methodology : Methods and Techniques. New Age International Ltd.

Kybertaistelu 2020. 2014. Accessed on 20 June 2019. Retrieved from http://www.doria.fi/handle/10024/103034

Logistics Command Finland - The Finnish Defence Forces. N.d. Accessed on 8 April 2020. Retrieved from https://puolustusvoimat.fi/en/about-us/logistics-command

Mahajan U. 2020. VMware Delivers NSX-T 3.0 with Innovations in Cloud, Security, Containers, and Operations. Accessed on 17 April 2020. Retrieved from https://blogs.vmware.com/networkvirtualization/2020/04/nsx-t-3-0.html/

Micro-Segmentation Builds Security Into Your Data Center's DNA. 2016. Accessed on 10 September 2019. Retrieved from https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/solutionbrief/partners/intel/vmware-micro-segmentation-builds-security-into-your-data-centers-white-paper.pdf

Mitchell, D., & Keegan, T. 2011. VMware vSphere for Dummies. FD; 2 edition

Morency, J., & Torti, T. 1996. Do VLANs make sense in your network? Accessed on 22 August 2019. Retrieved from https://janet.finna.fi/PrimoRecord/pci.gale_ofg17975809

Mutchler, C. 2016. Accessed on 11 September 2019. Retrieved from https://twitter.com/chrismutchler/status/800885500517023745

Online transaction processing (OLTP). 2019. Accessed on 19 February 2020. Retrieved from https://docs.microsoft.com/en-us/azure/architecture/data-guide/relational-data/online-transaction-processing

Ponemon, L. 2019. What's New in the 2019 Cost of a Data Breach Report. Accessed on 18 September 2019. Retrieved from https://securityintelligence.com/posts/whats-new-in-the-2019-cost-of-a-data-breach-report/

Portnoy, M. 2016. Virtualization Essentials. Accessed on 2 July 2019. Retrieved from https://janet.finna.fi/Record/nelli14.3710000000829230

Pujolle, G. 2015. Software Networks: Virtualization, SDN, 5G and Security. Accessed on 3 July 2019. Retrieved from https://janet.finna.fi/Record/nelli14.3710000000459379

Routio, P. 2007. Case Study. Accessed on 9 April 2020. Retrieved from http://www2.uiah.fi/projects/metodi/171.htm

Routio, P. 2007. Developing an Activity. Accessed on 9 April 2020. Retrieved from http://www2.uiah.fi/projects/metodi/120.htm#toimtutk

Salisbury, B. 2013. Inside Google's Software-Defined Network. Accessed 5 April 2020. Retrieved from https://www.networkcomputing.com/networking/inside-googles-software-defined-network

Santana, G., 2017. VMware NSX® Network Virtualization Fundamentals. Accessed on 18 February 2020. Retrieved from https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/nsx/vmware-network-virtualization-fundamentals-guide.pdf

Shackleford, D. 2019. Evolving Micro-Segmentation for Preventive Security: Adaptive Protection in a DevOps World. Accessed on 17 September 2019. Retrieved from https://www.sans.org/reading-room/whitepapers/analyst/membership/38760

Software-Defined Networking: The New Norm for Networks. 2012. Accessed on 11 September 2019. Retrieved from https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

Terranova, A. 2018. Zero Trust Security Architectures - Network Micro-Segmentation. Accessed on 9 August 2019. Retrieved from https://blogs.akamai.com/2018/09/zero-trust-security-architectures---network-micro-segmentation.html

The History of NSX and the Future of Network Virtualization. 2016. Accessed on 26 August 2019. Retrieved from https://www.vmware.com/radius/history-nsx-future-network-virtualization/

The VMware NSX Network Virtualization Platform. 2013. Accessed on 20 June 2019. Retrieved from https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/whitepaper/products/nsx/vmware-nsx-network-virtualization-platform-white-paper.pdf

VMware SDDC. N.d. Accessed on 20 June 2019. Retrieved from https://code.vmware.com/sddc-getting-started

VandenBrink, R. 2010. Layer 2 Security - Private VLANs. Accessed on 11 September 2019. Retrieved from https://isc.sans.edu/forums/diary/Layer+2+Security+Private+VLANs+the+Story+Continues/8785/

Vanveerdeghem, S. 2018. Context-Aware Micro-segmentation – an innovative approach to Application and User Identity Firewall. Accessed on 10 September 2019. Retrieved from https://blogs.vmware.com/networkvirtualization/2018/02/context-aware-micro-segmentation-innovative-approach-application-user-identity-firewall.html/

Virtual Route Forwarding Design Guide. 2008. Accessed on 6 September 2019. Retrieved from https://www.cisco.com/c/en/us/td/docs/voice_ip_comm/cucme/vrf/design/guide/vrfDesignGuide.html

Wagner, N., Sahin, C., Winterrose, M., Riordan, J., Pena, J., Hanson, D. & Streilein, W. 2016. Towards automated cyber decision support: A case study on network segmentation for security. Accessed on 9 August 2019. Retrieved from https://janet.finna.fi/PrimoRecord/pci.ieee_s7849908

Ward, R., & Beyer, B. 2014. BeyondCorp - A New Approach to Enterprise Security. Accessed on 24 August 2019. Retrieved from https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43231.pdf

Wilmington, G. 2017. VMware NSX® Micro-segmentation Day 2. Accessed on 20 June 2019. Retrieved from https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/nsx/vmware-micro-segmentation-day-2.pdf

Wilmington, G. 2019. Overcoming the Barriers to Micro-segmentation. Accessed on 17 September 2019. Retrieved from https://blogs.vmware.com/networkvirtualization/2019/10/overcoming-barriers-to-micro-segmentation.html/

Zero Trust Security. N.d. Accessed on 26 August 2019. Retrieved from https://www.akamai.com/us/en/solutions/security/zero-trust-security-model.jsp

Appendices

Appendix 1. Test scenario VM networks and IP addressing

```
[root@JKMT-APP ~]# ip -4 addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
    inet X.X.X.171/23 brd X.X.X.255 scope global noprefixroute ens192
       valid_lft forever preferred_lft forever
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
    inet 192.168.42.10/24 brd 192.168.42.255 scope global noprefixroute
ens224
       valid_lft forever preferred_lft forever
4: ens256: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
    inet 1.0.0.1/24 brd 1.0.0.255 scope global noprefixroute ens256
       valid_lft forever preferred_lft forever
[root@JKMT-APP ~]# ip route show
default via X.X.X.1 dev ens192 proto static metric 100
1.0.0.0/24 dev ens256 proto kernel scope link src 1.0.0.1 metric 102
192.168.42.0/24 dev ens224 proto kernel scope link src 192.168.42.10 metric
103
192.168.43.0/24 via 192.168.42.1 dev ens224 proto static metric 103
X.X.X.0/23 dev ens192 proto kernel scope link src X.X.X.171 metric 100


[root@JKMT-DB ~]# ip -4 addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
    inet X.X.X.172/23 brd X.X.X.255 scope global noprefixroute ens192
       valid_lft forever preferred_lft forever
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
    inet 192.168.43.10/24 brd 192.168.43.255 scope global noprefixroute
ens224
       valid_lft forever preferred_lft forever
4: ens256: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
    inet 1.0.0.2/24 brd 1.0.0.255 scope global noprefixroute ens256
       valid_lft forever preferred_lft forever
[root@JKMT-DB ~]# ip route show
default via X.X.X.1 dev ens192 proto static metric 100
1.0.0.0/24 dev ens256 proto kernel scope link src 1.0.0.2 metric 102
192.168.42.0/24 via 192.168.43.1 dev ens224 proto static metric 103
192.168.43.0/24 dev ens224 proto kernel scope link src 192.168.43.10 metric
103
X.X.X.0/23 dev ens192 proto kernel scope link src X.X.X.172 metric 100
```

Appendix 2. SRX firewall rule configuration

```
##Global address book configurations##
set security address-book global address TESTI-JKMT-DB_joku1 description
"JKMT-APP palvelin"
set security address-book global address TESTI-JKMT-DB_joku1
192.168.43.10/32
set security address-book global address TESTI-JKMT_APP-joku1 description
"JKMT-DB palvelin"
set security address-book global address TESTI-JKMT_APP-joku1
192.168.42.10/32
##Applications##
set applications application qperf term qperf-tcp destination-port 19765-
19766
set applications application qperf term qperf-tcp protocol tcp
set applications application qperf term qperf-udp destination-port 19765-
19766
set applications application qperf term qperf-udp protocol udp
##Security Firewall Policy : X-jkmt-app - X-jkmt-db##
set security policies from-zone X-jkmt-app to-zone X-jkmt-db policy TESTI-
JKMT-APP_to_DB match application postgres
set security policies from-zone X-jkmt-app to-zone X-jkmt-db policy TESTI-
JKMT-APP_to_DB match application qperf
set security policies from-zone X-jkmt-app to-zone X-jkmt-db policy TESTI-
JKMT-APP_to_DB match destination-address TESTI-JKMT-DB_joku1
set security policies from-zone X-jkmt-app to-zone X-jkmt-db policy TESTI-
JKMT-APP_to_DB match source-address TESTI-JKMT_APP-joku1
set security policies from-zone X-jkmt-app to-zone X-jkmt-db policy TESTI-
JKMT-APP_to_DB then log session-close
set security policies from-zone X-jkmt-app to-zone X-jkmt-db policy TESTI-
JKMT-APP_to_DB then log session-init
set security policies from-zone X-jkmt-app to-zone X-jkmt-db policy TESTI-
JKMT-APP_to_DB then permit
```

Appendix 3. Test run output - nmap

```
[root@JKMT-APP ~]# nmap -p 22,80,443,5432 -sS -T3 -Pn 192.168.43.10

Starting Nmap 6.40 ( http://nmap.org ) at 2020-02-13 13:09 EET
Nmap scan report for 192.168.43.10
Host is up (0.00081s latency).
PORT     STATE    SERVICE
22/tcp   filtered ssh
80/tcp   filtered http
443/tcp  filtered https
5432/tcp open     postgresql

Nmap done: 1 IP address (1 host up) scanned in 1.28 seconds
[root@JKMT-APP ~]# nmap -p 22,80,443,5432 -sS -T3 -Pn 1.0.0.2

Starting Nmap 6.40 ( http://nmap.org ) at 2020-02-13 13:13 EET
Nmap scan report for 1.0.0.2
Host is up (0.00072s latency).
PORT     STATE  SERVICE
22/tcp   closed ssh
80/tcp   closed http
443/tcp  closed https
5432/tcp open   postgresql
MAC Address: 00:50:56:B3:CF:80 (VMware)
```

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

Appendix 4. Test run output - qperf

```
[root@JKMT-APP ~]# host=192.168.43.10; for i in 1 2 3 4 5; do qperf $host -
ip 19766 -t 60 -m 1400 udp_bw; sleep 10; echo "------"; done
udp_bw:
    send_bw  =  116 MB/sec
    recv_bw  =  115 MB/sec
------
udp_bw:
    send_bw  =  115 MB/sec
    recv_bw  =  114 MB/sec
------
udp_bw:
    send_bw  =  117 MB/sec
    recv_bw  =  116 MB/sec
------
udp_bw:
    send_bw  =  116 MB/sec
    recv_bw  =  116 MB/sec
------
udp_bw:
    send_bw  =  117 MB/sec
    recv_bw  =  117 MB/sec
------
[root@JKMT-APP ~]# host=1.0.0.2; for i in 1 2 3 4 5; do qperf $host -ip
19766 -t 60 -m 1400 udp_bw; sleep 10; echo "------"; done
udp_bw:
    send_bw  =  113 MB/sec
    recv_bw  =  112 MB/sec
------
udp_bw:
    send_bw  =  113 MB/sec
    recv_bw  =  113 MB/sec
------
udp_bw:
    send_bw  =  113 MB/sec
    recv_bw  =  113 MB/sec
------
udp_bw:
    send_bw  =  113 MB/sec
    recv_bw  =  113 MB/sec
------
udp_bw:
    send_bw  =  113 MB/sec
    recv_bw  =  112 MB/sec
------

[root@JKMT-APP ~]# host=192.168.43.10; for i in 1 2 3 4 5; do qperf $host -
ip 19766 -t 60 -m 64k tcp_bw; sleep 10; echo "------"; done
tcp_bw:
    bw  =  115 MB/sec
------
tcp_bw:
    bw  =  115 MB/sec
------
tcp_bw:
    bw  =  115 MB/sec
------
tcp_bw:
```

```
    bw  =  115 MB/sec
------
tcp_bw:
    bw  =  115 MB/sec
------
[root@JKMT-APP ~]# host=1.0.0.2; for i in 1 2 3 4 5; do qperf $host -ip
19766 -t 60 -m 64k tcp_bw; sleep 10; echo "------"; done
tcp_bw:
    bw  =  112 MB/sec
------
tcp_bw:
    bw  =  112 MB/sec
------
tcp_bw:
    bw  =  112 MB/sec
------
tcp_bw:
    bw  =  112 MB/sec
------
tcp_bw:
    bw  =  112 MB/sec
------
[root@JKMT-APP ~]# host=192.168.43.10; for i in 1 2 3 4 5; do qperf $host -
ip 19766 -t 60 -m 1 tcp_lat udp_lat; sleep 10; echo "------"; done
tcp_lat:
    latency  =  168 us
udp_lat:
    latency  =  165 us
------
tcp_lat:
    latency  =  181 us
udp_lat:
    latency  =  172 us
------
tcp_lat:
    latency  =  163 us
udp_lat:
    latency  =  165 us
------
tcp_lat:
    latency  =  172 us
udp_lat:
    latency  =  171 us
------
tcp_lat:
    latency  =  171 us
udp_lat:
    latency  =  167 us
------
[root@JKMT-APP ~]# host=1.0.0.2; for i in 1 2 3 4 5; do qperf $host -ip
19766 -t 60 -m 1 tcp_lat udp_lat; sleep 10; echo "------"; done
tcp_lat:
    latency  =  166 us
udp_lat:
    latency  =  162 us
------
tcp_lat:
    latency  =  165 us
udp_lat:
    latency  =  159 us
------
```

```
tcp_lat:
    latency  =  174 us
udp_lat:
    latency  =  163 us
------
tcp_lat:
    latency  =  171 us
udp_lat:
    latency  =  157 us
------
tcp_lat:
    latency  =  172 us
udp_lat:
    latency  =  153 us
------
```

Appendix 5. Test run output - sysbench

```
[root@JKMT-APP ~]# for thost in "192.168.43.10" "1.0.0.2"; do
>         for sbtest in select delete insert update_index oltp; do
>                 echo ">>>>> RUNNING ${sbtest} TO ${thost}"
>                 sysbench --warmup-time=30 --threads=32 --db-driver=pgsql
--oltp-table-size=100000 --oltp-tables-count=24 --pgsql-host=${thost} --
pgsql-user=postgres --pgsql-password=Password1 --pgsql-db=sbtest --time=300
--report-interval=30 /usr/share/sysbench/tests/include/oltp_leg-
acy/${sbtest}.lua run
>                 sleep 60
>         done
> done
>>>>> RUNNING select TO 192.168.43.10
sysbench 1.0.9 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 32
Report intermediate results every 30 second(s)
Initializing random number generator from current time


Initializing worker threads...

Threads started!

[ 30s ] thds: 32 tps: 16584.89 qps: 16584.89 (r/w/o: 16584.89/0.00/0.00)
lat (ms,95%): 3.13 err/s: 0.00 reconn/s: 0.00
[ 60s ] thds: 32 tps: 17794.42 qps: 17794.42 (r/w/o: 17794.42/0.00/0.00)
lat (ms,95%): 3.13 err/s: 0.00 reconn/s: 0.00
[ 90s ] thds: 32 tps: 16243.58 qps: 16243.58 (r/w/o: 16243.58/0.00/0.00)
lat (ms,95%): 3.55 err/s: 0.00 reconn/s: 0.00
[ 120s ] thds: 32 tps: 17507.30 qps: 17507.30 (r/w/o: 17507.30/0.00/0.00)
lat (ms,95%): 3.19 err/s: 0.00 reconn/s: 0.00
[ 150s ] thds: 32 tps: 16544.88 qps: 16544.88 (r/w/o: 16544.88/0.00/0.00)
lat (ms,95%): 3.43 err/s: 0.00 reconn/s: 0.00
[ 180s ] thds: 32 tps: 18288.99 qps: 18288.99 (r/w/o: 18288.99/0.00/0.00)
lat (ms,95%): 2.97 err/s: 0.00 reconn/s: 0.00
[ 210s ] thds: 32 tps: 17383.74 qps: 17383.74 (r/w/o: 17383.74/0.00/0.00)
lat (ms,95%): 3.25 err/s: 0.00 reconn/s: 0.00
[ 240s ] thds: 32 tps: 16971.70 qps: 16971.70 (r/w/o: 16971.70/0.00/0.00)
lat (ms,95%): 3.30 err/s: 0.00 reconn/s: 0.00
```

```
[ 270s ] thds: 32 tps: 17724.18 qps: 17724.18 (r/w/o: 17724.18/0.00/0.00)
lat (ms,95%): 3.13 err/s: 0.00 reconn/s: 0.00
[ 300s ] thds: 32 tps: 16950.31 qps: 16950.31 (r/w/o: 16950.31/0.00/0.00)
lat (ms,95%): 3.36 err/s: 0.00 reconn/s: 0.00
SQL statistics:
    queries performed:
        read:                            5159982
        write:                           0
        other:                           0
        total:                           5159982
    transactions:                        5159982 (17198.00 per sec.)
    queries:                             5159982 (17198.00 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          300.0293s
    total number of events:              5159982

Latency (ms):
         min:                                  0.32
         avg:                                  1.86
         max:                               1113.35
         95th percentile:                      3.25
         sum:                            9587266.26

Threads fairness:
    events (avg/stddev):           161249.4375/1055.80
    execution time (avg/stddev):   299.6021/0.01

>>>>> RUNNING delete TO 192.168.43.10
sysbench 1.0.9 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 32
Report intermediate results every 30 second(s)
Initializing random number generator from current time


Initializing worker threads...

Threads started!

[ 30s ] thds: 32 tps: 15734.95 qps: 15734.95 (r/w/o: 0.00/1027.83/14707.12)
lat (ms,95%): 4.33 err/s: 0.00 reconn/s: 0.00
[ 60s ] thds: 32 tps: 17342.08 qps: 17342.08 (r/w/o: 0.00/424.41/16917.67)
lat (ms,95%): 3.62 err/s: 0.00 reconn/s: 0.00
[ 90s ] thds: 32 tps: 17099.95 qps: 17099.95 (r/w/o: 0.00/370.33/16729.62)
lat (ms,95%): 3.55 err/s: 0.00 reconn/s: 0.00
[ 120s ] thds: 32 tps: 16515.13 qps: 16515.17 (r/w/o: 0.00/322.77/16192.40)
lat (ms,95%): 3.82 err/s: 0.00 reconn/s: 0.00
[ 150s ] thds: 32 tps: 16540.83 qps: 16540.79 (r/w/o: 0.00/283.37/16257.43)
lat (ms,95%): 3.75 err/s: 0.00 reconn/s: 0.00
[ 180s ] thds: 32 tps: 17636.40 qps: 17636.40 (r/w/o: 0.00/272.49/17363.90)
lat (ms,95%): 3.43 err/s: 0.00 reconn/s: 0.00
[ 210s ] thds: 32 tps: 16578.95 qps: 16578.95 (r/w/o: 0.00/231.04/16347.91)
lat (ms,95%): 3.62 err/s: 0.00 reconn/s: 0.00
[ 240s ] thds: 32 tps: 17369.84 qps: 17369.84 (r/w/o: 0.00/225.43/17144.40)
lat (ms,95%): 3.43 err/s: 0.00 reconn/s: 0.00
[ 270s ] thds: 32 tps: 16884.67 qps: 16884.67 (r/w/o: 0.00/199.07/16685.60)
lat (ms,95%): 3.49 err/s: 0.00 reconn/s: 0.00
```

```
[ 300s ] thds: 32 tps: 18405.78 qps: 18405.78 (r/w/o: 0.00/201.90/18203.88)
lat (ms,95%): 3.07 err/s: 0.00 reconn/s: 0.00
SQL statistics:
    queries performed:
        read:                            0
        write:                           106765
        other:                           4996603
        total:                           5103368
    transactions:                        5103368 (17009.94 per sec.)
    queries:                             5103368 (17009.94 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          300.0186s
    total number of events:              5103368

Latency (ms):
         min:                                    0.29
         avg:                                    1.88
         max:                                  270.57
         95th percentile:                        3.62
         sum:                              9587507.71

Threads fairness:
    events (avg/stddev):           159480.2500/4832.51
    execution time (avg/stddev):   299.6096/0.01

>>>>> RUNNING insert TO 192.168.43.10
sysbench 1.0.9 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 32
Report intermediate results every 30 second(s)
Initializing random number generator from current time


Initializing worker threads...

Threads started!

[ 30s ] thds: 32 tps: 5035.56 qps: 5035.56 (r/w/o: 0.00/5035.56/0.00) lat
(ms,95%): 12.98 err/s: 0.00 reconn/s: 0.00
[ 60s ] thds: 32 tps: 4894.02 qps: 4894.02 (r/w/o: 0.00/4894.02/0.00) lat
(ms,95%): 13.70 err/s: 0.00 reconn/s: 0.00
[ 90s ] thds: 32 tps: 4719.94 qps: 4719.94 (r/w/o: 0.00/4719.94/0.00) lat
(ms,95%): 14.21 err/s: 0.00 reconn/s: 0.00
[ 120s ] thds: 32 tps: 4868.74 qps: 4868.74 (r/w/o: 0.00/4868.74/0.00) lat
(ms,95%): 13.95 err/s: 0.00 reconn/s: 0.00
[ 150s ] thds: 32 tps: 4728.64 qps: 4728.64 (r/w/o: 0.00/4728.64/0.00) lat
(ms,95%): 14.21 err/s: 0.00 reconn/s: 0.00
[ 180s ] thds: 32 tps: 4788.28 qps: 4788.28 (r/w/o: 0.00/4788.28/0.00) lat
(ms,95%): 13.95 err/s: 0.00 reconn/s: 0.00
[ 210s ] thds: 32 tps: 4635.15 qps: 4635.15 (r/w/o: 0.00/4635.15/0.00) lat
(ms,95%): 14.73 err/s: 0.00 reconn/s: 0.00
[ 240s ] thds: 32 tps: 4521.99 qps: 4521.99 (r/w/o: 0.00/4521.99/0.00) lat
(ms,95%): 15.00 err/s: 0.00 reconn/s: 0.00
[ 270s ] thds: 32 tps: 4958.12 qps: 4958.12 (r/w/o: 0.00/4958.12/0.00) lat
(ms,95%): 13.70 err/s: 0.00 reconn/s: 0.00
[ 300s ] thds: 32 tps: 4646.74 qps: 4646.74 (r/w/o: 0.00/4646.74/0.00) lat
(ms,95%): 14.46 err/s: 0.00 reconn/s: 0.00
```

```
SQL statistics:
    queries performed:
        read:                            0
        write:                           1433974
        other:                           0
        total:                           1433974
    transactions:                        1433974 (4778.92 per sec.)
    queries:                             1433974 (4778.92 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          300.0579s
    total number of events:              1433974

Latency (ms):
         min:                                  0.76
         avg:                                  6.69
         max:                                199.40
         95th percentile:                     13.95
         sum:                           9595109.44

Threads fairness:
    events (avg/stddev):           44811.6875/252.97
    execution time (avg/stddev):   299.8472/0.02

>>>>> RUNNING update_index TO 192.168.43.10
sysbench 1.0.9 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 32
Report intermediate results every 30 second(s)
Initializing random number generator from current time


Initializing worker threads...

Threads started!

[ 30s ] thds: 32 tps: 13780.42 qps: 13780.42 (r/w/o: 0.00/147.32/13633.11)
lat (ms,95%): 4.18 err/s: 0.00 reconn/s: 0.00
[ 60s ] thds: 32 tps: 15511.12 qps: 15511.12 (r/w/o: 0.00/164.69/15346.43)
lat (ms,95%): 3.68 err/s: 0.00 reconn/s: 0.00
[ 90s ] thds: 32 tps: 14862.87 qps: 14862.87 (r/w/o: 0.00/159.27/14703.60)
lat (ms,95%): 3.96 err/s: 0.00 reconn/s: 0.00
[ 120s ] thds: 32 tps: 16037.23 qps: 16037.27 (r/w/o: 0.00/172.01/15865.25)
lat (ms,95%): 3.49 err/s: 0.00 reconn/s: 0.00
[ 150s ] thds: 32 tps: 16240.69 qps: 16240.66 (r/w/o: 0.00/178.42/16062.24)
lat (ms,95%): 3.43 err/s: 0.00 reconn/s: 0.00
[ 180s ] thds: 32 tps: 15399.25 qps: 15399.25 (r/w/o: 0.00/162.57/15236.68)
lat (ms,95%): 3.68 err/s: 0.00 reconn/s: 0.00
[ 210s ] thds: 32 tps: 16272.49 qps: 16272.49 (r/w/o: 0.00/177.97/16094.52)
lat (ms,95%): 3.36 err/s: 0.00 reconn/s: 0.00
[ 240s ] thds: 32 tps: 14714.33 qps: 14714.33 (r/w/o: 0.00/155.70/14558.63)
lat (ms,95%): 3.96 err/s: 0.00 reconn/s: 0.00
[ 270s ] thds: 32 tps: 15703.39 qps: 15703.39 (r/w/o: 0.00/168.37/15535.02)
lat (ms,95%): 3.68 err/s: 0.00 reconn/s: 0.00
[ 300s ] thds: 32 tps: 16433.34 qps: 16433.34 (r/w/o: 0.00/174.85/16258.49)
lat (ms,95%): 3.30 err/s: 0.00 reconn/s: 0.00
SQL statistics:
    queries performed:
```

```
        read:                            0
        write:                           49837
        other:                           4598884
        total:                           4648721
    transactions:                        4648721 (15494.98 per sec.)
    queries:                             4648721 (15494.98 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          300.0129s
    total number of events:              4648721

Latency (ms):
         min:                                  0.31
         avg:                                  2.06
         max:                                295.74
         95th percentile:                      3.68
         sum:                            9588656.39

Threads fairness:
    events (avg/stddev):           145272.5312/6356.37
    execution time (avg/stddev):   299.6455/0.01

>>>>> RUNNING oltp TO 192.168.43.10
sysbench 1.0.9 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 32
Report intermediate results every 30 second(s)
Initializing random number generator from current time


Initializing worker threads...

Threads started!

[ 30s ] thds: 32 tps: 741.04 qps: 14835.77 (r/w/o:
10386.73/1363.76/3085.28) lat (ms,95%): 68.05 err/s: 0.10 reconn/s: 0.00
[ 60s ] thds: 32 tps: 679.65 qps: 13595.82 (r/w/o: 9517.75/1851.57/2226.51)
lat (ms,95%): 69.29 err/s: 0.07 reconn/s: 0.00
[ 90s ] thds: 32 tps: 660.50 qps: 13211.96 (r/w/o: 9248.70/2035.87/1927.40)
lat (ms,95%): 68.05 err/s: 0.10 reconn/s: 0.00
[ 120s ] thds: 32 tps: 602.67 qps: 12050.43 (r/w/o:
8434.88/1939.94/1675.60) lat (ms,95%): 86.00 err/s: 0.07 reconn/s: 0.00
[ 150s ] thds: 32 tps: 630.47 qps: 12613.89 (r/w/o:
8829.74/2072.64/1711.51) lat (ms,95%): 74.46 err/s: 0.17 reconn/s: 0.00
[ 180s ] thds: 32 tps: 653.22 qps: 13067.43 (r/w/o:
9148.27/2165.52/1753.65) lat (ms,95%): 66.84 err/s: 0.17 reconn/s: 0.00
[ 210s ] thds: 32 tps: 544.05 qps: 10884.43 (r/w/o:
7618.84/1810.01/1455.57) lat (ms,95%): 87.56 err/s: 0.07 reconn/s: 0.00
[ 240s ] thds: 32 tps: 576.91 qps: 11540.39 (r/w/o:
8078.69/1924.33/1537.37) lat (ms,95%): 82.96 err/s: 0.13 reconn/s: 0.00
[ 270s ] thds: 32 tps: 561.29 qps: 11223.74 (r/w/o:
7856.39/1870.83/1496.52) lat (ms,95%): 94.10 err/s: 0.00 reconn/s: 0.00
[ 300s ] thds: 32 tps: 580.50 qps: 11613.10 (r/w/o:
8129.55/1944.21/1539.34) lat (ms,95%): 84.47 err/s: 0.13 reconn/s: 0.00
SQL statistics:
    queries performed:
        read:                            2617636
        write:                           569455
```

```
        other:                          552329
        total:                          3739420
    transactions:                       186944 (623.03 per sec.)
    queries:                            3739420 (12462.32 per sec.)
    ignored errors:                     30     (0.10 per sec.)
    reconnects:                         0      (0.00 per sec.)

General statistics:
    total time:                         300.0542s
    total number of events:             186944

Latency (ms):
         min:                                  15.02
         avg:                                  51.35
         max:                                 416.00
         95th percentile:                      78.60
         sum:                            9599936.92

Threads fairness:
    events (avg/stddev):           5842.0000/29.56
    execution time (avg/stddev):   299.9980/0.02

>>>>> RUNNING select TO 1.0.0.2
sysbench 1.0.9 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 32
Report intermediate results every 30 second(s)
Initializing random number generator from current time


Initializing worker threads...

Threads started!

[ 30s ] thds: 32 tps: 16956.34 qps: 16956.34 (r/w/o: 16956.34/0.00/0.00)
lat (ms,95%): 3.25 err/s: 0.00 reconn/s: 0.00
[ 60s ] thds: 32 tps: 17348.63 qps: 17348.63 (r/w/o: 17348.63/0.00/0.00)
lat (ms,95%): 3.25 err/s: 0.00 reconn/s: 0.00
[ 90s ] thds: 32 tps: 17556.71 qps: 17556.71 (r/w/o: 17556.71/0.00/0.00)
lat (ms,95%): 3.07 err/s: 0.00 reconn/s: 0.00
[ 120s ] thds: 32 tps: 17313.49 qps: 17313.49 (r/w/o: 17313.49/0.00/0.00)
lat (ms,95%): 3.19 err/s: 0.00 reconn/s: 0.00
[ 150s ] thds: 32 tps: 15950.10 qps: 15950.10 (r/w/o: 15950.10/0.00/0.00)
lat (ms,95%): 3.62 err/s: 0.00 reconn/s: 0.00
[ 180s ] thds: 32 tps: 17903.94 qps: 17903.94 (r/w/o: 17903.94/0.00/0.00)
lat (ms,95%): 3.07 err/s: 0.00 reconn/s: 0.00
[ 210s ] thds: 32 tps: 17275.92 qps: 17275.92 (r/w/o: 17275.92/0.00/0.00)
lat (ms,95%): 3.19 err/s: 0.00 reconn/s: 0.00
[ 240s ] thds: 32 tps: 17356.44 qps: 17356.44 (r/w/o: 17356.44/0.00/0.00)
lat (ms,95%): 3.25 err/s: 0.00 reconn/s: 0.00
[ 270s ] thds: 32 tps: 17680.62 qps: 17680.62 (r/w/o: 17680.62/0.00/0.00)
lat (ms,95%): 3.19 err/s: 0.00 reconn/s: 0.00
[ 300s ] thds: 32 tps: 16564.20 qps: 16564.20 (r/w/o: 16564.20/0.00/0.00)
lat (ms,95%): 3.43 err/s: 0.00 reconn/s: 0.00
SQL statistics:
    queries performed:
        read:                           5157320
        write:                          0
        other:                          0
        total:                          5157320
```

```
    transactions:                        5157320 (17189.85 per sec.)
    queries:                             5157320 (17189.85 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          300.0173s
    total number of events:              5157320

Latency (ms):
         min:                                    0.29
         avg:                                    1.86
         max:                                  217.89
         95th percentile:                        3.25
         sum:                                9587694.19

Threads fairness:
    events (avg/stddev):           161166.2500/2076.02
    execution time (avg/stddev):   299.6154/0.01

>>>>> RUNNING delete TO 1.0.0.2
sysbench 1.0.9 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 32
Report intermediate results every 30 second(s)
Initializing random number generator from current time


Initializing worker threads...

Threads started!

[ 30s ] thds: 32 tps: 16463.40 qps: 16463.40 (r/w/o: 0.00/988.89/15474.51)
lat (ms,95%): 4.03 err/s: 0.00 reconn/s: 0.00
[ 60s ] thds: 32 tps: 14049.14 qps: 14049.14 (r/w/o: 0.00/293.44/13755.70)
lat (ms,95%): 4.33 err/s: 0.00 reconn/s: 0.00
[ 90s ] thds: 32 tps: 18693.38 qps: 18693.38 (r/w/o: 0.00/344.70/18348.68)
lat (ms,95%): 3.13 err/s: 0.00 reconn/s: 0.00
[ 120s ] thds: 32 tps: 18978.02 qps: 18978.02 (r/w/o: 0.00/303.20/18674.81)
lat (ms,95%): 2.97 err/s: 0.00 reconn/s: 0.00
[ 150s ] thds: 32 tps: 17177.35 qps: 17177.35 (r/w/o: 0.00/241.73/16935.62)
lat (ms,95%): 3.68 err/s: 0.00 reconn/s: 0.00
[ 180s ] thds: 32 tps: 17887.13 qps: 17887.13 (r/w/o: 0.00/230.70/17656.43)
lat (ms,95%): 3.30 err/s: 0.00 reconn/s: 0.00
[ 210s ] thds: 32 tps: 17679.91 qps: 17679.91 (r/w/o: 0.00/206.70/17473.21)
lat (ms,95%): 3.43 err/s: 0.00 reconn/s: 0.00
[ 240s ] thds: 32 tps: 18272.98 qps: 18272.98 (r/w/o: 0.00/194.64/18078.34)
lat (ms,95%): 3.13 err/s: 0.00 reconn/s: 0.00
[ 270s ] thds: 32 tps: 17243.33 qps: 17243.33 (r/w/o: 0.00/169.33/17074.00)
lat (ms,95%): 3.49 err/s: 0.00 reconn/s: 0.00
[ 300s ] thds: 31 tps: 18566.98 qps: 18566.98 (r/w/o: 0.00/166.80/18400.18)
lat (ms,95%): 3.07 err/s: 0.00 reconn/s: 0.00
SQL statistics:
    queries performed:
        read:                            0
        write:                           94217
        other:                           5156324
        total:                           5250541
    transactions:                        5250541 (17500.52 per sec.)
    queries:                             5250541 (17500.52 per sec.)
```

```
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          300.0181s
    total number of events:              5250541

Latency (ms):
         min:                                    0.29
         avg:                                    1.83
         max:                                  398.56
         95th percentile:                        3.49
         sum:                                9586669.22

Threads fairness:
    events (avg/stddev):           164079.4062/947.18
    execution time (avg/stddev):   299.5834/0.02

>>>>> RUNNING insert TO 1.0.0.2
sysbench 1.0.9 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 32
Report intermediate results every 30 second(s)
Initializing random number generator from current time


Initializing worker threads...

Threads started!

[ 30s ] thds: 32 tps: 4956.31 qps: 4956.31 (r/w/o: 0.00/4956.31/0.00) lat
(ms,95%): 13.46 err/s: 0.00 reconn/s: 0.00
[ 60s ] thds: 32 tps: 4872.91 qps: 4872.91 (r/w/o: 0.00/4872.91/0.00) lat
(ms,95%): 13.70 err/s: 0.00 reconn/s: 0.00
[ 90s ] thds: 32 tps: 4767.24 qps: 4767.24 (r/w/o: 0.00/4767.24/0.00) lat
(ms,95%): 14.21 err/s: 0.00 reconn/s: 0.00
[ 120s ] thds: 32 tps: 5012.59 qps: 5012.59 (r/w/o: 0.00/5012.59/0.00) lat
(ms,95%): 13.22 err/s: 0.00 reconn/s: 0.00
[ 150s ] thds: 32 tps: 4979.81 qps: 4979.81 (r/w/o: 0.00/4979.81/0.00) lat
(ms,95%): 13.46 err/s: 0.00 reconn/s: 0.00
[ 180s ] thds: 32 tps: 5008.07 qps: 5008.07 (r/w/o: 0.00/5008.07/0.00) lat
(ms,95%): 12.98 err/s: 0.00 reconn/s: 0.00
[ 210s ] thds: 32 tps: 4801.47 qps: 4801.47 (r/w/o: 0.00/4801.47/0.00) lat
(ms,95%): 13.95 err/s: 0.00 reconn/s: 0.00
[ 240s ] thds: 32 tps: 4754.89 qps: 4754.89 (r/w/o: 0.00/4754.89/0.00) lat
(ms,95%): 14.21 err/s: 0.00 reconn/s: 0.00
[ 270s ] thds: 32 tps: 4867.62 qps: 4867.62 (r/w/o: 0.00/4867.62/0.00) lat
(ms,95%): 13.70 err/s: 0.00 reconn/s: 0.00
[ 300s ] thds: 32 tps: 4770.83 qps: 4770.83 (r/w/o: 0.00/4770.83/0.00) lat
(ms,95%): 13.95 err/s: 0.00 reconn/s: 0.00
SQL statistics:
    queries performed:
        read:                            0
        write:                           1463815
        other:                           0
        total:                           1463815
    transactions:                        1463815 (4878.40 per sec.)
    queries:                             1463815 (4878.40 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)
```

```
General statistics:
    total time:                          300.0568s
    total number of events:              1463815

Latency (ms):
         min:                                  0.82
         avg:                                  6.55
         max:                                189.53
         95th percentile:                     13.70
         sum:                              9595302.67

Threads fairness:
    events (avg/stddev):           45744.2188/470.76
    execution time (avg/stddev):   299.8532/0.02

>>>>> RUNNING update_index TO 1.0.0.2
sysbench 1.0.9 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 32
Report intermediate results every 30 second(s)
Initializing random number generator from current time


Initializing worker threads...

Threads started!

[ 30s ] thds: 32 tps: 14789.51 qps: 14789.51 (r/w/o: 0.00/132.31/14657.21)
lat (ms,95%): 3.82 err/s: 0.00 reconn/s: 0.00
[ 60s ] thds: 32 tps: 16373.26 qps: 16373.26 (r/w/o: 0.00/140.70/16232.56)
lat (ms,95%): 3.30 err/s: 0.00 reconn/s: 0.00
[ 90s ] thds: 32 tps: 14310.97 qps: 14310.97 (r/w/o: 0.00/124.60/14186.37)
lat (ms,95%): 4.18 err/s: 0.00 reconn/s: 0.00
[ 120s ] thds: 32 tps: 15531.57 qps: 15531.57 (r/w/o: 0.00/133.90/15397.67)
lat (ms,95%): 3.62 err/s: 0.00 reconn/s: 0.00
[ 150s ] thds: 32 tps: 15114.49 qps: 15114.49 (r/w/o: 0.00/132.64/14981.85)
lat (ms,95%): 3.68 err/s: 0.00 reconn/s: 0.00
[ 180s ] thds: 32 tps: 16128.20 qps: 16128.20 (r/w/o: 0.00/140.93/15987.27)
lat (ms,95%): 3.43 err/s: 0.00 reconn/s: 0.00
[ 210s ] thds: 32 tps: 16432.14 qps: 16432.14 (r/w/o: 0.00/138.93/16293.21)
lat (ms,95%): 3.25 err/s: 0.00 reconn/s: 0.00
[ 240s ] thds: 32 tps: 16163.38 qps: 16163.38 (r/w/o: 0.00/140.83/16022.55)
lat (ms,95%): 3.30 err/s: 0.00 reconn/s: 0.00
[ 270s ] thds: 32 tps: 16170.07 qps: 16170.07 (r/w/o: 0.00/138.47/16031.60)
lat (ms,95%): 3.55 err/s: 0.00 reconn/s: 0.00
[ 300s ] thds: 32 tps: 17158.72 qps: 17158.72 (r/w/o: 0.00/148.73/17009.99)
lat (ms,95%): 3.02 err/s: 0.00 reconn/s: 0.00
SQL statistics:
    queries performed:
        read:                            0
        write:                           41162
        other:                           4704125
        total:                           4745287
    transactions:                        4745287 (15816.35 per sec.)
    queries:                             4745287 (15816.35 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
```

```
    total time:                          300.0201s
    total number of events:              4745287

Latency (ms):
         min:                                  0.28
         avg:                                  2.02
         max:                                229.42
         95th percentile:                      3.55
         sum:                            9588195.49

Threads fairness:
    events (avg/stddev):           148290.2188/687.80
    execution time (avg/stddev):   299.6311/0.01

>>>>> RUNNING oltp TO 1.0.0.2
sysbench 1.0.9 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 32
Report intermediate results every 30 second(s)
Initializing random number generator from current time


Initializing worker threads...

Threads started!

[ 30s ] thds: 32 tps: 727.57 qps: 14563.39 (r/w/o:
10196.01/1317.81/3049.57) lat (ms,95%): 69.29 err/s: 0.03 reconn/s: 0.00
[ 60s ] thds: 32 tps: 662.12 qps: 13245.60 (r/w/o: 9272.30/1779.54/2193.76)
lat (ms,95%): 77.19 err/s: 0.20 reconn/s: 0.00
[ 90s ] thds: 32 tps: 630.00 qps: 12602.14 (r/w/o: 8821.52/1910.37/1870.24)
lat (ms,95%): 77.19 err/s: 0.07 reconn/s: 0.00
[ 120s ] thds: 32 tps: 594.80 qps: 11898.28 (r/w/o:
8329.59/1903.23/1665.46) lat (ms,95%): 82.96 err/s: 0.07 reconn/s: 0.00
[ 150s ] thds: 32 tps: 630.50 qps: 12611.69 (r/w/o:
8828.42/2054.37/1728.90) lat (ms,95%): 78.60 err/s: 0.07 reconn/s: 0.00
[ 180s ] thds: 32 tps: 612.29 qps: 12247.97 (r/w/o:
8573.67/2028.55/1645.75) lat (ms,95%): 80.03 err/s: 0.17 reconn/s: 0.00
[ 210s ] thds: 32 tps: 605.91 qps: 12120.69 (r/w/o:
8484.77/2005.79/1630.13) lat (ms,95%): 78.60 err/s: 0.10 reconn/s: 0.00
[ 240s ] thds: 32 tps: 593.56 qps: 11871.77 (r/w/o:
8310.48/1981.65/1579.65) lat (ms,95%): 75.82 err/s: 0.00 reconn/s: 0.00
[ 270s ] thds: 32 tps: 538.57 qps: 10770.88 (r/w/o:
7538.92/1796.91/1435.04) lat (ms,95%): 97.55 err/s: 0.13 reconn/s: 0.00
[ 300s ] thds: 32 tps: 656.60 qps: 13134.34 (r/w/o:
9194.29/2190.05/1750.01) lat (ms,95%): 69.29 err/s: 0.10 reconn/s: 0.00
SQL statistics:
    queries performed:
        read:                            2626680
        write:                           569145
        other:                           556519
        total:                           3752344
    transactions:                        187592 (625.19 per sec.)
    queries:                             3752344 (12505.55 per sec.)
    ignored errors:                      28     (0.09 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          300.0504s
    total number of events:              187592
```

```
Latency (ms):
         min:                                    13.22
         avg:                                    51.17
         max:                                   346.27
         95th percentile:                        78.60
         sum:                                 9599855.42

Threads fairness:
    events (avg/stddev):           5862.2500/192.37
    execution time (avg/stddev):   299.9955/0.01
```

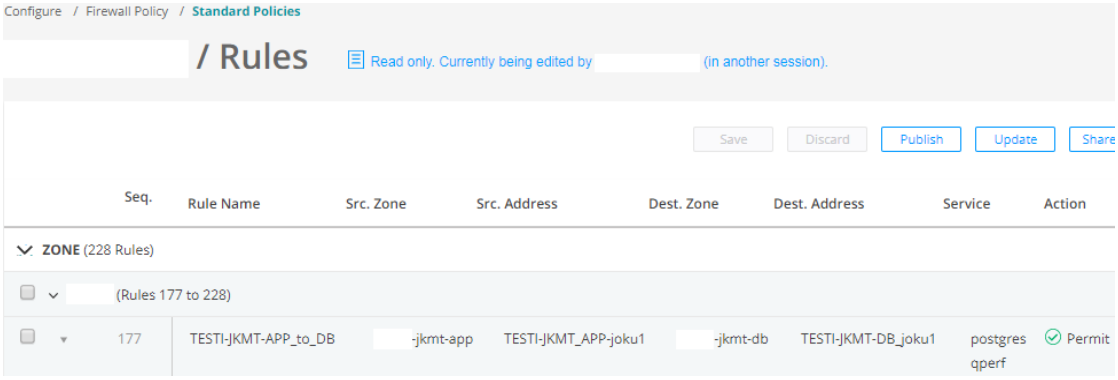
Appendix 6. Test virtual machine specifications and DRS rule

```
PS D:\Users\juha.koskinen> get-vm -Name JKMT-* |select Id,Name,`
>> PowerState,Guest,NumCPU,CoresPerSocket,MemoryMB |fl


Id             : VirtualMachine-vm-1397
Name           : JKMT-DB
PowerState     : PoweredOn
Guest          : JKMT-DB:CentOS 7 (64-bit)
NumCpu         : 2
CoresPerSocket : 1
MemoryMB       : 8192

Id             : VirtualMachine-vm-1396
Name           : JKMT-APP
PowerState     : PoweredOn
Guest          : JKMT-APP:CentOS 7 (64-bit)
NumCpu         : 2
CoresPerSocket : 1
MemoryMB       : 8192



PS D:\Users\juha.koskinen> Get-DrsRule -Cluster (get-cluster)[0] `
>> -Name JKMT* | select Name,VMIds,KeepTogether,Enabled,Type


Name          : JKMT-2TIER
VMIds         : {VirtualMachine-vm-1396, VirtualMachine-vm-1397}
KeepTogether : False
Enabled       : True
Type          : VMAntiAffinity
```

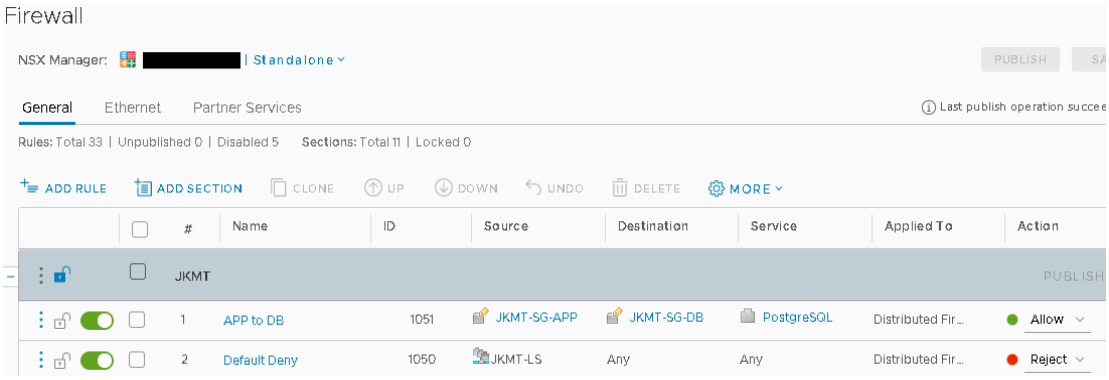
Appendix 7. Test virtual machine network cards and memberships

```
PS D:\Users\juha.koskinen> get-vm -Name JKMT-* | Get-NetworkAdapter `
>> | select Parent,Name,Type,NetworkName,MacAddress | ft

Parent   Name                    Type NetworkName  MacAddress
------   ----                    ---- -----------  ----------
JKMT-DB  Network adapter 1 Vmxnet3                 00:50:56:b3:68:c8
JKMT-DB  Network adapter 2 Vmxnet3 JKMT-DB         00:50:56:b3:22:96
JKMT-DB  Network adapter 3 Vmxnet3 JKMT-LS         00:50:56:b3:cf:80
JKMT-APP Network adapter 1 Vmxnet3                 00:50:56:b3:e5:ee
JKMT-APP Network adapter 2 Vmxnet3 JKMT-APP        00:50:56:b3:2e:8c
JKMT-APP Network adapter 3 Vmxnet3 JKMT-LS         00:50:56:b3:2e:a5
```

Appendix 8. SRX firewall rule management



Appendix 9. NSX firewall rule management



Appendix 10. NSX network trace between VMs through a logical switch