Bulletin of Electrical Engineering and Informatics
Vol. 12, No. 5, October 2023, pp. 3142~3152
ISSN: 2302-9285, DOI: 10.11591/eei.v12i5.4240

A case analysis for Kubernetes network security of financial service industry in Indonesia using zero trust model

Nico Surantha[1,2], Felix Ivan[1], Ritchie Chandra[1]
[1]BINUS Graduate Program-Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia
[2]Department of Electrical, Electronic, and Communication Engineering, Faculty of Engineering, Tokyo City University, Tokyo, Japan

Article Info

Article history:

Received Jun 12, 2022
Revised Dec 16, 2022
Accepted Feb 3, 2023

Keywords:

Kubernetes
Micro-services
Network security
Zero trust model

ABSTRACT

In this study, a case analysis of Kubernetes application in an enterprise providing financial services in Indonesia is presented. It is implemented to improve their digital services-based application, developed using micro-services architecture. Kubernetes, an application container technology, has been applied in the enterprise providing financial services in Indonesia to improve their digital services-based application, developed using micro-services architecture concepts. Without incorporating any additional hardware, the new technology and services have been adopted to existing virtualized resources of the enterprise. An infrastructure design for secure Kubernetes networking was built and has been studied using the data center provided by the enterprise. This study focuses on two important aspects: network and security. As a security guideline, the network recommendations from VMware, Cisco, and Forrester's zero trust model were employed to design the infrastructure and were evaluated. The proposed secure network infrastructure design is successfully applied in the container networks using the zero trust requirement, the enterprise's requirements, and constraints. And we hope this study about network design security can be used and adaptable with existing network and reduce risk and disruption to the business caused by the network.

Corresponding Author:

Ritchie Chandra
BINUS Graduate Program-Master of Computer Science, Bina Nusantara University
Jakarta, Indonesia
Email: ritchie.chandra@binus.ac.id

1. INTRODUCTION

Infrastructure is required to develop and operate an application with micro-service architecture. As an application, the micro-services are expected to work together to provide a larger function of services in which the connectivity between the services becomes critical. To ensure the communication of each service, a reliable network is needed. Security acts to protect the services by validating and mitigating these services from being compromised or attacked. Virtualization is one of the infrastructure technologies that is commonly used by enterprises to host applications, including services. Alternatively, the enterprises could be benefited by applying container technology resulting in immutable and agile services. Explain software aging in cloud computing, software aging is in the operation software systems will be accumulate errors to system failure and other hazardous consequences [1]. Software aging can be happening because memory fragmentation, resource consumption in large scale and accumulation of errors happen in system.

One of the Indonesian financial service enterprises has deployed application container technology as infrastructure technology. An IT security head explains that his enterprise as well as many other enterprises are striving to be more competitive in their field. Reliable technology infrastructure enables his enterprise to

provide significant improvement, such as shortening the product-development cycles. Moreover, troubleshooting and manual setting of the configuration changes would require less time. The respondent also focuses on important aspects of infrastructure, such as network and security. His concern raises two major questions: i) "how to run applications on Kubernetes container technology?" and ii) "how to address the network and security aspects in their highly virtualized data center?". Using hypervisors, most of the enterprises' applications are run virtually while the dynamic routing protocol is used to route the access to the applications in a physical network device by the open shortest path first (OSPF). A firewall to inspect outgoing and incoming traffic is deployed to implement security at the perimeter. In this case study, Kubernetes acted as leverage only in the farm segment of the enterprise's internal server, instead of the entire network segment.

According to Pahl [2], the container could be activated via a requirement to deploy an application on distributed cloud platforms because many enterprises have adopted Kubernetes to run their applications. Kubernetes along with Docker is a standard platform for a cloud-based service [3]. Moreover, Pahl discusses the relevance of containers with virtualization technology. The relevance is laid on the requirement of containers for advanced networking features based on network virtualization like routing [2]. To manage applications on a microservice basis, Kubernetes allows the creation of multiple Horizontal Pod Autoscaler instances adapts to a single microservice deployment, and proposes me-Kube (multi-level elastic Kubernetes), referring to a Kubernetes extension that introduces a hierarchical architecture to control the elasticity of application-based microservices [4]. Containers are characterized by having the ephemeral state and aspects of a stateful microservice, which create a more complex array previously created by Kubernetes controllers [5]. The development of microservice architecture-based cloud applications presents a variety of challenges; one of them is scalability at the container level. A study shows that an algorithm uses an agnostic approach to auto-scale microservices which are implemented in Google Kubernetes Engine [6]. Microservices in software development arise when a business department finds risk in a product and starts asking questions; an example of the migration from monolithic to microservices took place in 2010 when Netflix started using Amazon`s Web Services (AWS) to host applications and services with over 100 fine-grained services instead of web applications, also known as or war [7].

Since the new threats using techniques to bypass perimeter-based protections emerge, John Becker highlights the significance of the Forrester® zero trust model and explains the vulnerability of IP-based network perimeter [8]. An American market research company, Forrester®, proposes the zero trust model stipulating that all networks should be classified as untrusted, and all traffic must be logged and inspected without considering its location; these strategies will lead to the use of least privilege strategies and enforce strict access control [9]. Zero trust model can be used to secure and protect the big data environment as well as solve a problem in big data security [10]. A zero trust management model for internet of things (IoT) can guarantee the validation and participation of the credentials and configurations of every resource of the infrastructure in the network [11]. A study proposes a model that is applicable to protect various data adopted from the zero trust model and present new techniques based on the zero trust model to secure the data [12].

The implementation of zero trust for the energy sector could reduce the risks, secure the systems, and protect the data of zero trust security architecture [13]. The authors have developed cyber security and combined the zero trust model with cloud computing because zero trust schemes in cloud computing should be integrated and combined with practices, policies, and technologies; thus, cyber security could become workable [14]. The combination of the zero trust model and Kubernetes can protect infrastructure from attackers who threaten the data of enterprises or organizations [15].

Plenty of security aspects and container systems are still necessarily explored. This study provides a novel network design by developing secure Kubernetes based on container technology. This development is expected to serve several recommendations for the enterprise. The vendors' best practices, such as the network virtualization technology of VMware and Cisco, were reviewed and adopted as a part of our design. The zero trust model of Forrester® was also outlined as the security guideline. To validate the proposed design, a lab simulation was deployed to explore the ability of the design to fulfill the enterprises' requirements. This paper is an extended version of the work of [16].

2. PROBLEM STATEMENT AND PRELIMINARIES

2.1. Kubernetes container system

Kubernetes is used over the years to suffice the needs and the capability efficiently and using Kubernetes to expose HTTP endpoints for metrics as it is simpler and most used [17]. Proposed work to move high computational power near IoT sensors with fog serverless framework (FSF) and using Kubernetes to form an interface with Kubernetes cluster which manage a set of Docker container [18]. Each of the Kubernetes clusters has one or more worker nodes and masters [3] because the namespaces of each cluster enable the isolation of networks and environments. Meanwhile, different environments, such as development and production, can be segmented to have their network segment in traditional physical networks. In

Kubernetes, similar levels of segmentation possibly use a namespace. Kubernetes *et al.* [3] define a pod as the smallest entity in the Kubernetes cluster. The container pod is allocated with its internet protocol (IP) address, which is comparable to the endpoints, such as a real machine in the physical network environment. A network gateway or router is required for the container pods in a namespace. Thus, the external network is accessible to external networks or other namespaces. The containers run within pods in Kubernetes. Nonetheless, Kubernetes does not manage the containers within a pod because it only manages the pods. Through the master node, Kubernetes could expand or shrink the number of pods [3]. The pod's probability of compromising would increase as the number of pods increases. The zero trust model is implemented at the network level to mitigate the risk of compromising, internal attacks, and external attacks as well as ultimately avoid disruption to the business environment [9].

2.2. Zero trust model-based information security

There are no trusted and untrusted-internal or external networks in the zero trust model [9]. Inspection and logging should be performed on all traffic since the traffic is categorized as untrusted. According to Forrester®, the perimeter-based network security insufficiently protects the networks and is labeled as internal or trusted to perimeter-based protection [9]. The virtual local area network (VLAN), which performs network segmentation, is not categorized as a network security method since it cannot prevent any traffic to move between VLANs and cannot gain access to critical systems in different VLANs [19].

The architecture that deploys microservice applications employs Kubernetes in the private cloud and high availability of Kubernetes for application-based microservices [20]. As a comparison, the 2017 IBM X-Force report has also been studied. The results of investigating global threat intelligence and deep security could provide enhanced security solutions. The report states that in 2016, 58% of the attacks in the financial service sector is from insiders while 53% of these attacks are not noticed by any employees [21]. The study on Forrester® and the 2017 IBM X-Force report provide a strong motivation to more comprehensively implement the zero trust model in the network. Meanwhile, adopts the zero trust model in intranet data to secure the authentication when users want to access intranet data [22]. This method or architecture can more effectively protect network communication.

3. THE PROPOSED METHOD DESIGN OF NETWORK AND SECURITY

This study employed a top-down design approach [23] to design the network. High-level staff, such as the Head of the Department, were targeted from the financial service enterprises in Indonesia to understand the business constraints and goals. Meanwhile, the manager-level staff contributed to give knowledge on the technical goals and limitations to produce a proposed design. Then, the lab environment was built to ensure that all technical requirements had been achieved. The requirements were based on the enterprises' business requirements to collect the technical requirements. The container network connectivity from and to existing external physical networks was tested by dynamic-routing protocols to conduct route redistribution. By default, all of the traffics were considered unsecured or untrusted, unless they had been stated. In other words, the network security was configured to only allow intended traffic between containers.

In detail, the plan, design, implement, operate, and optimize (PDIOO) network life-cycle was adopted [23] to design a secure network. As shown by Figure 1, the methods were classified into three major steps: i) collecting requirements, ii) designing a network, and iii) evaluating the design. The results of this analysis are as follows.

3.1. First stage: requirement gathering

The process of requirement gathering contains several steps, such as defining business goals, translating the business goals into technical constraints and goals, and ultimately evaluating the currently existing network. The outputs of every process are outlined. The analysis focuses on both business and technical aspects. XYZ Bank's current network topology and configuration is required, to understand how the container networks shall be integrated into the existing networks.

3.1.1. Business goals

The IT security head expects shorter product development cycles because the released cycle could be adjusted to the business requirement. To mitigate problems, a new secure network design should be able to solve inconsistent policies and causes of misconfigurations. The network and security design should leverage the dedicated technology for the container system, instead of using hardware-based technology. To accommodate such a request, the containers should run with a consistent scale that is parallel to its host and prefers network virtualization.

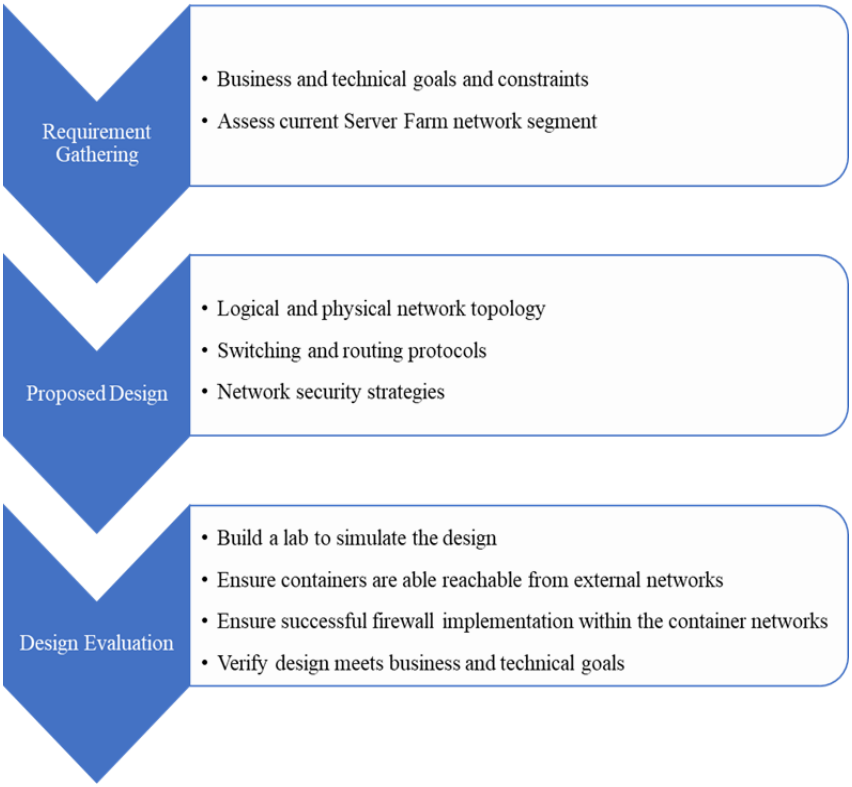

Figure 1. Research method

3.1.2. Technical goals

The technical team was interviewed to technically translate the goals by seeking their expectations and technical perspectives. The technical goals provided by the network team are i) aligning with the existing IP address schemes, ii) redistributing authentic and intended routing information through a routing protocol, and iii) preventing reconnaissance attacks (e.g., port scanning) and network-based attacks (e.g., route hijacking). Hence, internal network communications could be protected. The translation from business goals to technical goals is outlined in Table 1.

Table 1. The mapping of the business goals into technical goals

Available business goals	Translation of technical goals
Shortening life-cycles of the product development	Planning IP address for containerized application networks Ensuring the connectivity to the existing IPv4 networks
Mitigating problems due to security and network (e.g., misconfigurations and inconsistent policies)	Protecting the internal container-to-container communications Ensuring only intended and authentic-routing information advertised in the networks Implementing network security to combat network-based attacks, including reconnaissance-based and route hijacking attacks.
Modernizing container technology-specific technologies	Protecting the internal communications between containers Implementing network security to prevent network-based attacks, such as route hijacking and reconnaissance-based attacks Properly recording traffic

3.1.3. Technical constraints

The IT security head expects a conformable design with the existing data center environment of the enterprise without additional hardware. Hence, VMware's virtualization is deployed to virtualize the infrastructure, while any required virtual network components can be supported by the available hardware resources. The network team is also put in another constraint because the OSPF must be used in its networks as a dynamic routing protocol to avoid any configuration changes that might impact the existing systems; for example, changing routing protocols or migrating network gateways. All of the prerequisites for configuration changes are provided in Table 2.

Table 2. Necessary requirements prior to configuration change

Pre-requisite design decision	Realization of design
Minimal having three different port groups in the VDS in the form of overlay, management, and data	Traffic differentiation and routing among physical servers in physical networks
Reusing the existing subnets and VLAN ID of the management network	Aligning the security policy of XYZ Bank and putting the management IP address of all components under the out-of-band subnet
Provisioning a new overlay traffic network	Traversing devices and links to increase the MTU size by 1580 bytes or above ue to the additional 80 bytes of payload in the original packet by overlay
Provisioning a new network for external traffic data path	Isolating the high-security risk of rating in- and outgoing traffic of the container networks from management and overlay's network

3.2. Second stage: proposed design of network

This section discusses the network design while the next section discusses the security design. In this study, all of the required components of the design are virtual-based to avoid any additional hardware. Additional configuration changes are still required to implement the design in the existing devices.

To set up a lab simulation for designing the study, the minimum requirements defined in the Kubernetes documentation [3] are used since the size and number of workloads performing in a Kubernetes cluster are unique for each enterprise. Kubernetes pods use the application programming interface (API) server in master to worker nodes within the cluster because each Kubernetes cluster ran with one worker and one master node [3]. Configuration changes are required when installing the underlying equipment. For example, the addition of VLAN and IP addressing network and the increase of maximum transmit unit (MTU) size are needed to add the IP packet header of the overlay network since generic network virtualization encapsulation (GENEVE) [24] is used to overlay encapsulation. Table 2 summarizes the configuration changes for the design recommendations, which relate to the design of the existing logical network.

Figure 2 depicts that the setup of the logical container network is outlined and a high-level visualization of the network design is provided. Through VMware's NSX [21], each namespace in the cluster provides a logical switch, and a Tier-1 router is constructed. Before being connected to the external virtual router, the namespaces are routed and aggregated to a Tier-0 router [21]. To set up the logical network as shown in Figure 2, several steps are followed, such as making a design decision based on the IP address planning as well as routing the protocol option and security considerations. Table 3 summarizes recommendations for the logical network design.

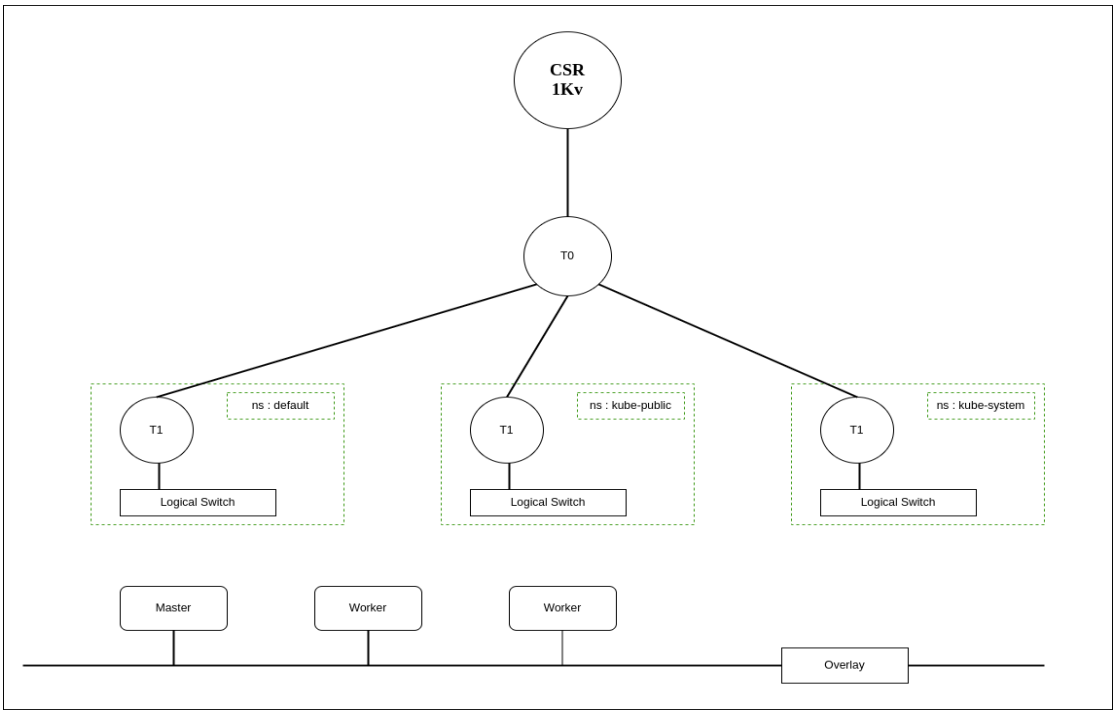

Figure 2. Proposed design of logical network

Table 3. The recommendation for logical network design

Logical network design decisions	Realization of the design
Pre-allocating the 16 prefixes of network address spaces for the container environment	256 subnets are reserved for new namespaces.
Providing at least one logical switch for each namespace [25]	Ensuring pod-to-pod network connectivity [25]
Using BGP as the protocol for dynamic routing [26]	Proving dynamic routing protocols for >100,000 endpoints to deploy large-scale data centers [26]
Enabling authentication of a routing protocol between routing peers [27]	Avoid unauthorized access or/and routing information tampers and attacks, such as route hijacking
Enabling route filtering for containers of internal networks [27]	Not advertising container networks without access from or to external networks outside the domain of BGP routing

Enterprise probably has various space availability allocations of network addresses. It is recommended to allocate a large prefix per cluster to aid the subnet assignment for each newly created namespace. Using this setup, a prefix allocation of/16 becomes a sample instead of a definitive requirement. Moreover, 256 namespaces could be served in a Kubernetes cluster, and 254 pods could be instantiated for each namespace. The modification of prefix and subnet allocations could be performed when required.

Container podscouldbe dramaticallycompared to virtual or physical machines since the container pods could be easily escalated. To accommodate the dynamic networking behavior, a scalable dynamic routing protocol is preferred. Border gateway protocol (BGP), as a proven dynamic routing protocol, is reported in IETF RFC 7938, supporting "large-scale" data centers with over 100,000 servers [26]. In the design, the container pods are assumed as endpoints, which are similar to traditional servers and have their unique IP address. Kubernetes could be considered "large-scale" since it supports >100,000 pods within a cluster. Cisco suggests that the security mechanism of BGP in an enterprise environment (e.g., filtering and routing protocol authentication) could diminish the risks of attacks from the router's perspective, such as route hijacking [27]. To reduce risks from reconnaissance attacks, the routes (i.e., internal pods or namespace networking) are not necessarily and externally advertised because they can be filtered out at the Tier-0 router.

3.3. Third stage: proposed design of network security

The zero trust model considers that all traffic is untrusted or unsecured [9], [19]. The inspection is required, and the traffic is only allowed when needed, for example when the application or service requires the traffic to function properly [9], [19]. Moreover, all traffic could be denied by default and only allow a required or legitimate traffic communication.

The network security recommendations based on the zero trust model are summarized in Table 4. To inspect network traffic, the firewall could be deployed [28]. The incoming traffic from or to external networks could be inspected and protected by the perimeter-based firewall [9]. The firewall must be able to recognize the incoming traffic from the pods to execute the internal firewall at the container pod level. To map the pods as security objects in the firewall, Kubernetes' label is utilized as an identifier [21], and the label is assigned using Kubernetes API-server to the pods after or while the pods are instantiated [3], [21]. The firewall policies should have been configured and defined before the pod labeling process to protect and inspect the incoming traffic from pods [21]. The firewall policies deny all traffic by default and only allow the required communications. The policies include routing protocol of the traffic, legitimate workload traffic or users, and the management of traffic from workloads' or users' requirements. Then, all traffic that has been analyzed is reported to an external Syslog server in order to store connectivity information.

Table 4. The recommendation for security design

Decision on security design	Realization of the design
Using zero trust as a model	Treating all traffic as untrusted or unsecured without considering its direction or location but complying with the requirement of the respondents to protect external elements (at the perimeter) and internal elements (i.e., between container pods) [9], [19]
Inspecting all traffics [19]	Attacking unauthorized personnel mitigation in accordance with the zero trust model [19]
Logging all traffic [19]	Troubleshooting, analyzing, and/or forensic testing based on the zero trust model [19]

4. RESULTS AND DISCUSSION

The conformability of the proposed design with the technical requirements has been evaluated. Table 5 shows that six requirements have been evaluated from the security or network perspectives. Therequirement analysis is done to state the technical requirements as shown in Table 1. A lab simulation was performed using a computer with a core 4 processor, x86, 500 GB disk capacity, 32 GB RAM, and a

network interface card (NIC) for network connectivity. Figure 3 shows the design of the logical network for the lab simulation.

A vSphere ESXi is installed to run all the required components on the computer. As a virtual machine in a VMware vSphere environment, a Kubernetes cluster with 2 worker nodes and 1 master node [3] is installed by following the setup installation guide. To set up the NSX environment and components, VMware's installation guide is also followed along with Kubernetes nodes. A virtual router (CSR1Kv) is incorporated into the environment to simulate the layers of three devices, which are the enterprise's checkpoint firewall appliances. To evaluate the design, the configurations following the proposed and recommended design are made. The design was evaluated using the performed connectivity tests and the design verifications. The results and the technical requirements are finally mapped, as presented in Table 1. The design evaluation is summarized in Table 5.

Table 5. Summary of the design evaluation

Design category	Technical specification	Method	Status (successful or failed)	Description
Network	Planning IP address to apply container networks	Producing several new namespaces in Kubernetes clusters [21]	Successful	Large network address space should be synchronized with the number of namespaces, and every new namespace is supplied with a /24 network.
Network	Securing connectivity to the existing IPv4 networks	Redistributing BGP routes into OSPF networks	Successful	BGP to OSPF redistributions are needed at the existing layers of three devices that become checkpoints of firewall appliances because the container networks use BGP.
Network	Ensuring only intended and authentic routing information advertised in networks	Conducting BGP authentication and filtering out internal routes from being externally advertised [27]	Successful	The BGP authentications should be configurated at the checkpoint firewall appliances. The internal route of namespaces is only added manually into the lists of prefixes by denying the policy at the container's perimeter (i.e., Tier-0) router and restricting it from being broadcasted to the external networks.
Security	Preventing penetration from external routes to the environment of the application container network	Nmap port scanning of the container pods from the external network [29]	Successful	A perimeter-based firewall is required to inspect outgoing and incoming traffic from the container networks' external routes.
Security	Considering all of the traffic as unsecured and only allowing required container communications	Testing ICMP between container pods in the same namespace and network [21]	Successful	Allocating Kubernetes labels to every instantiated pod is compulsory to apply the firewall policy. Moreover, proper configuration and documentation are needed for the firewall policy.
Security	Properly logging and recording traffic	Verifying the logging of inspected traffic to a Syslog serve	Successful	External or third-party Syslog servers should store the firewall's logged traffic inspection logs.

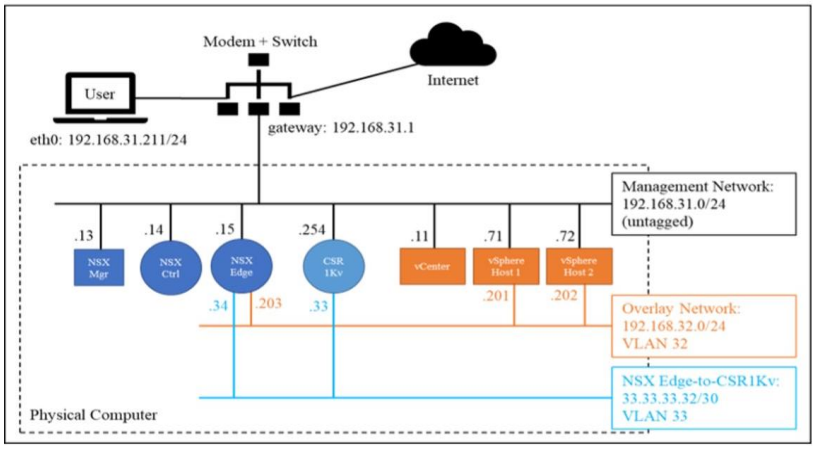

Figure 3. Design of logical network for lab environment

The first step of proposed design evaluations is to ensure the advertised information in the networks is only for the intended routing and authentic information. A new namespace is created to verify whether a subnet is allocated from the pre-allocated prefix and justified its external and internal connectivity. The external connectivity is verified using route redistribution, which refers to the virtual router of this setup and is performed at the external router peer. Alternatively, it can be done through the checkpoint firewalls in the enterprise's environment. Moreover, BGP message digest 5 (MD5) authentication is deployed to confirm the authenticity of the BGP peers and filter out certain internal namespace networks. Therefore, the BGP MD5 authentication is not acknowledged from the virtual router's routing table. Figure 4 shows the BGP routing table from the virtual router before the traffic filter.

The routing information from its peer is listed. The namespace's subnet contains the 10.1.6.0/24 network in the Kubernetes cluster. Figure 4 presents that the route could be filtered out by using and mapping the prefix lists to the outgoing interface at the BGP configuration of the Tier-0 (T0) router, supposing that the subnet is only internal and does not need external connectivity [27]. The 10.1.6.0/24 route would be excluded at the T0 router after the prefix lists are mapped to the router. Figure 5 shows that the 10.1.6.0/24 could not be found in the virtual router's routing table. To sum up, the external BGP authentication and filtration of the "internal" routes are successfully advertised.

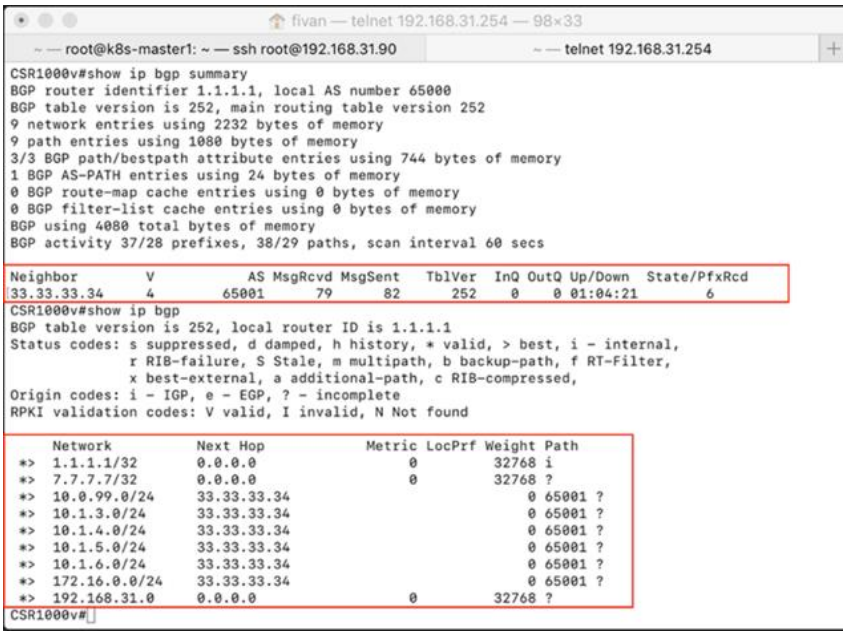

Figure 4. Screenshot of BGP routing table from virtual router prior to traffic filter

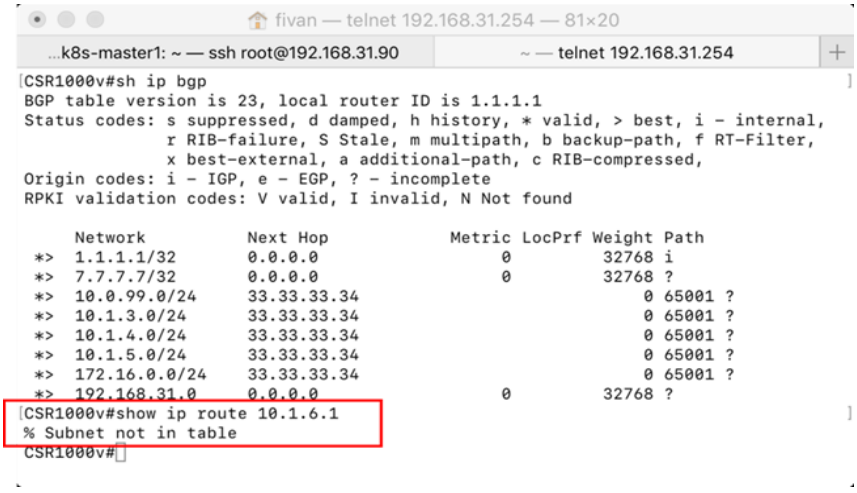

Figure 5. Screenshot of BGP routing table from route after traffic filter

The second step is the penetration test of the design from the external route to the application of the container network environment. To simulate a reconnaissance attack on the container pods, a port scanning is performed with the Nmap tool, initiated from the outside of Kubernetes' network. The perimeter firewall policy is configured according to the zero-trust model. All of the traffic is denied by default and allows only web traffic, internet control message protocol (ICMP), and BGP communication traffic for testing purposes.

Figure 6 signifies that the Nmap scan result is shown in four pods: 1 pod runs MYSQL service with the IP address of 10.1.5.1, and 3 pods run the NGINX service with TCP 80 port opened. The port of MYSQL pod is opened at TCP 3306. Nonetheless, it results in closed and unidentified status. Since the TCP 3306 port communication or the MYSQL service is not allowed by the policy, such a testing result is expected. Meanwhile, the MYSQL pod in the actual scenario would not be attained or allowed from any external networks because the MYSQL pod should be for the internal route only. Moreover, the MYSQL pod is reachable because ICMP communication traffic from the external route to the pod's network is only allowed for testing purposes.

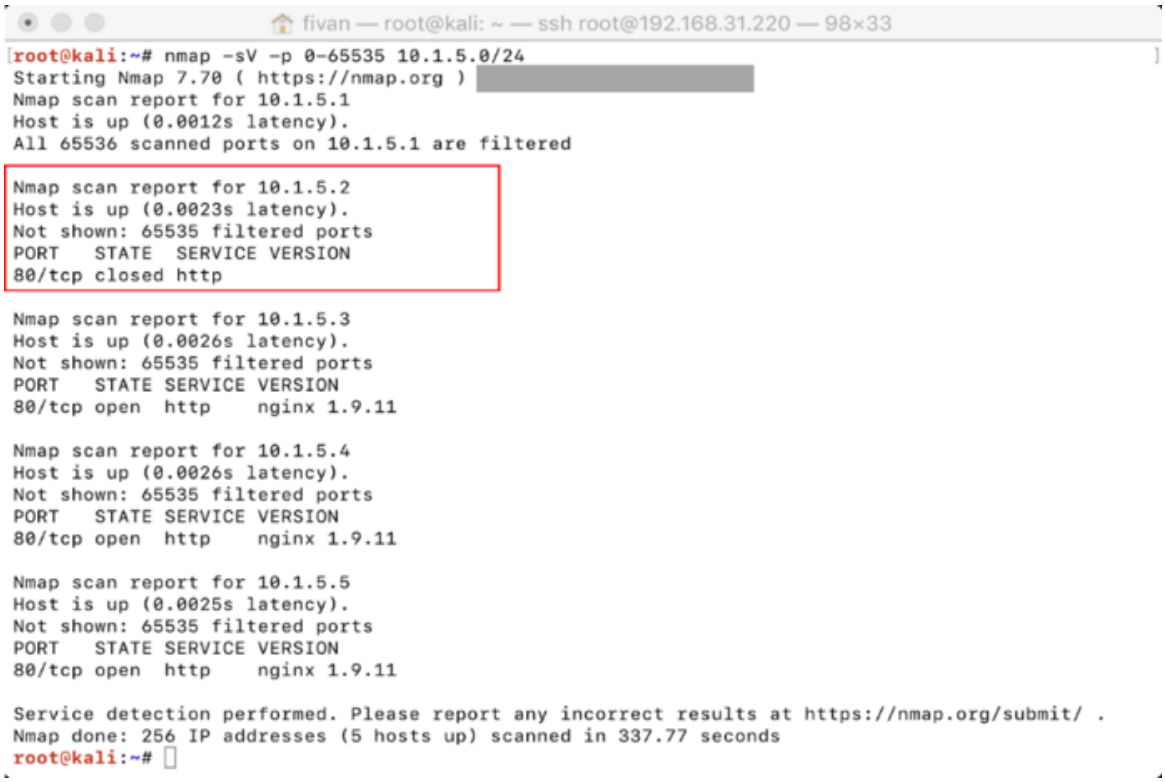

Figure 6. Screenshot of port scanning from external to container network

The third step is to evaluate the conformability of the proposed design to the other four technical requirements. Table 5 summarizes all of the evaluation results, which provide a better perspective of this particular work. To evaluate the internal network security, the label assignment to the pods verifies the container's pod-to-pod communication. Although the two pods are in the same network segment and namespace, the ICMP test traffic can be inspected and denied due to undefined communication in the distributed internal firewall's policy. The default firewall policy is configured to deny all traffic in the zero trust policy. In other words, communication would not be permitted without permission from the firewall policy. The traffic inspection was implemented by Kubernetes label that identifies the pods of incoming traffic to logical switch in the NSX networking. The policy only allows a legitimate pod-to-pod communication traffic so that the built-in internal firewall is identified and inspected. By default, pods could not communicate with another pod since they are not allowed and denied. Then, the logs of the inspected traffic are forwarded to an external Syslog server, and the ejected logs by the firewall are displayed.

5. CONCLUSION

A simulation of the design was built to evaluate the secure network design and validate network security following the zero trust model. To ensure proper implementation of firewall policy at the perimeter, penetration testing of network security, such as external network scanning and route hijacking, was conducted. The results show that the network is not penetrated. To examine the internal security, the internal traffic was identified using Kubernetes' labels, which allow the internal security to flow in the environment. The default has restricted non-legitimate or unidentified traffic. All of the traffic is examined by the firewall so that the traffics are logged to an external Syslog server to store the connectivity information. Then, the traffics could be repurposed for troubleshooting, forensics, and analysis. For future work, several aspects could be considered to improve the design. First, the role-based access controls could be implemented to provide strict control and the least privilege for authorized users. Second, identity management systems could be applied by unauthorized users and the next possible security-related users and would prevent unauthorized access.

REFERENCES

[1] S. P and N. G. Cholli, "An analysis of software aging in cloud environment," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 6, pp. 5985-5991, Dec. 2020, doi: 10.11591/ijece.v10i6.pp5985-5991.

[2] C. Pahl, "Containerization and the PaaS Cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, May 2015, doi: 10.1109/mcc.2015.51.

[3] K. Hightower, B. Burns, and J. Beda, *Kubernetes: Up and Running: Dive into the Future of Infrastructure*, 1st edition. Sebastopol, CA: O'Reilly Media, 2017.

[4] F. Rossi, V. Cardellini, and F. L. Presti, "Hierarchical Scaling of Microservices in Kubernetes," in *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pp. 28-37, Aug. 2020, doi: 10.1109/acsos49614.2020.00023.

[5] L. A. Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Microservice Based Architecture: Towards High-Availability for Stateful Applications with Kubernetes," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*, pp. 176-185, Jul. 2019, doi: 10.1109/qrs.2019.00034.

[6] A. A. Khaleq and I. Ra, "Agnostic Approach for Microservices Autoscaling in Cloud Applications," in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 1411-1415, Dec. 2019, doi: 10.1109/csci49370.2019.00264.

[7] N. Mateus-Coelho, M. Cruz-Cunha, and L. G. Ferreira, "Security in Microservices Architectures," *Procedia Computer Science*, vol. 181, pp. 1225–1236, 2021, doi: 10.1016/j.procs.2021.01.320.

[8] J. Becker, "Trust No One: A Gap Analysis of Moving IP-Based Network Perimeters to A Zero Trust Network Architecture | SANS Institute," *Global Information Assurance Certification Paper*, pp. 1–23, 2017.

[9] J. Kindervag, "No More Chewy Centers: The Zero Trust Model of Information Security," *Forrester*, 2016. [Online]. Available: https://www.forrester.com/report/No-More-Chewy-Centers-The-Zero-Trust-Model-Of-Information-Security/RES56682

[10] Y. Tao, Z. Lei, and P. Ruxiang, "Fine-Grained Big Data Security Method Based on Zero Trust Model," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 1040-1045, Dec. 2018, doi: 10.1109/padsw.2018.8644614.

[11] M. Samaniego and R. Deters, "Zero-Trust Hierarchical Management in IoT," in *2018 IEEE International Congress on Internet of Things (ICIOT)*, Jul. 2018, doi: 10.1109/iciot.2018.00019.

[12] I. Ahmed, T. Nahar, S. S. Urmi, and K. A. Taher, "Protection of Sensitive Data in Zero Trust Model," in *Proceedings of the International Conference on Computing Advancements*, Jan. 2020, doi: 10.1145/3377049.3377114.

[13] A. Alagappan, S. K. Venkatachary, and L. J. B. Andrews, "Augmenting Zero Trust Network Architecture to enhance security in virtual power plants," *Energy Reports*, vol. 8, pp. 1309–1320, Nov. 2022, doi: 10.1016/j.egyr.2021.11.272.

[14] S. Mehraj and M. T. Banday, "Establishing a Zero Trust Strategy in Cloud Computing Environment," in *2020 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1-5, Jan. 2020, doi: 10.1109/iccci48352.2020.9104214.

[15] D. D'Silva and D. D. Ambawade, "Building A Zero Trust Architecture Using Kubernetes," in *2021 6th International Conference for Convergence in Technology (I2CT)*, Apr. 2021, doi: 10.1109/i2ct51068.2021.9418203.

[16] N. Surantha and F. Ivan, "Secure Kubernetes Networking Design Based on Zero Trust Model: A Case Study of Financial Service Enterprise in Indonesia," in *Innovative Mobile and Internet Services in Ubiquitous Computing*, Springer International Publishing, 2019, pp. 348–361, doi: 10.1007/978-3-030-22263-5_34.

[17] P. S. S. P, S. V. Soudri, R. K. P, and S. Bailuguttu, "Enhancement of observability using Kubernetes operator," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 25, no. 1, pp. 496-503, Jan. 2022, doi: 10.11591/ijeecs.v25.i1.pp496-503.

[18] M. Elkholy and M. A. Marzok, "Light weight serverless computing at fog nodes for internet of things systems," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 26, no. 1, pp. 394-403, Apr. 2022, doi: 10.11591/ijeecs.v26.i1.pp394-403.

[19] V. Sucasas *et al.*, "A privacy-enhanced OAuth 2.0 based protocol for Smart City mobile applications," *Computers &amp$\mathsemicolon$ Security*, vol. 74, pp. 258–274, May 2018, doi: 10.1016/j.cose.2018.01.014.

[20] L. A. Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 970-973, Jul. 2018. doi: 10.1109/cloud.2018.00148.

[21] "IBM X-Force Threat Intelligence Index 2017," *Security Intelligence*, Mar. 29, 2017. [Online]. Available: https://securityintelligence.com/ibm-x-force-threat-intelligence-index-2017/ (accessed Mar. 09, 2023).

[22] T. Chuan, Y. Lv, Z. Qi, L. Xie, and W. Guo, "An Implementation Method of Zero-trust Architecture," *Journal of Physics: Conference Series*, vol. 1651, no. 1, pp. 1-6, Nov. 2020, doi: 10.1088/1742-6596/1651/1/012010.

[23] P. Oppenheimer, *Top-Down Network Design, 3rd Edition*, 3rd ed. Cisco Press, 2010. [Online]. Available: http://www.teraits.com/pitagoras/marcio/gpi/b_POppenheimer_TopDownNetworkDesign_3rd_ed.pdf Accessed: Mar. 09, 2023.

[24] J. Gross, I. Ganga, and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation," RFC Editor, Nov. 2020. doi: 10.17487/rfc8926.
[25] A. Gangil, J. Shen, D. Han, S. Orlando, S. Chaitanya, Y. Fauser, G. Kotton, Translating PAAS/CAAS abstractions to logical network topologies, US20180375728, 2018. [Online]. Available: https://patentscope.wipo.int/search/en/detail.jsf?docId=US235228914&tab=NATIONALBIBLIO. accessed: Mar 30, 2023.
[26] P. Lapukhov, A. Premji, and J. Mitchell, "Use of BGP for Routing in Large-Scale Data Centers," RFC Editor, Aug. 2016, doi: 10.17487/rfc7938.
[27 J. Stuppi, G. Schudel, T. Sammut, Protecting Border Gateway Protocol for the Enterprise, Cisco Secure (n.d.). [Online]. Available: https://tools.cisco.com/security/center/resources/protecting_border_gateway_protocol. accessed: Mar 30, 2023.
[28] M. E. Whitman and H. J. Mattord, *Principles of Information Security*, 4th edition. Boston, MA: Cengage Learning, 2011.
[29] P. Engebretson, *The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy*, 2nd edition. Amsterdam Boston: Syngress, 2013.

BIOGRAPHIES OF AUTHORS

Nico Surantha (Member, IEEE) received the B.Eng. and M.Eng. degrees from the Institut Teknologi Bandung, Indonesia, in 2007 and 2009, respectively, and the Ph.D. degree from the Kyushu Institute of Technology, Japan, in 2013. He currently works as a full-time Lecturer with the Department of Electrical, Electronic and Communication Engineering, Tokyo City University, Japan. He is also a lecturer at the Department of Computer Science, BINUS Graduate Program, Bina Nusantara University, Indonesia. His research interests include ubiquitous computing, intelligent systems, the internet of things, health monitoring, and system on chip. He can be contacted at email: nico.surantha@binus.ac.id.

Felix Ivan received the bachelor's degree from the Department of Computer Science, Bina Nusantara University, Jakarta, in 2010. Received Master degree with the Department of Computer Science, BINUS Graduate Program—Master of Computer Science, Jakarta, in 2016. His research interests include cloud computing, network security and network design. He can be contacted at email: felix.ivan@binus.ac.id.

Ritchie Chandra received the bachelor's degree from the Department of Computer Science, Bina Nusantara University, Jakarta, in 2018. He is currently pursuing the degree with the Computer Science Department, BINUS Graduate Program—Master of Computer Science. His research interests include health monitoring and the internet of things. He can be contacted at email: ritchie.chandra@binus.ac.id.