Dynamic security model for container orchestration platform

R.G.K.P.Kulathunga

2020

Dynamic security model for container orchestration platform

A dissertation submitted for the Degree of Master of Science in Computer Science

R.G.K.P.Kulathunga

University of Colombo School of Computing

2020

Declaration Page

The thesis is my original work and has not been submitted previously at this or any other university/Institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student name : R.G.K.P. Kulathunga

Registration number : 2017/MCS/047

Index number : 17440471

……..

Signature Date:

This is to certify that this thesis is based on the work of Mr. / Ms. under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by :

Supervisor name : Dr Kasun de Zoysa

……

Signature Date: 2020/11/20

Abstract

With the development of science and technology, people and organizations use widely spread software applications and huge databases to fulfill their tasks. Those applications and databases connect with and store more sensitive and personal details belonging to the domain they are address with. Therefore, a security system is more important in these kinds of situations. When a third-party user accesses the applications or databases, the security system plays a major role in order to verify the security.

When consider the security and privacy of application data, the Intrusion Detection System (IDS) comes into the play. This is a device or software application that uses to monitor a network or systems malicious activity or policy violations.

Currently, IDS is deployed in the container orchestration platform as the centralized component that monitor the whole traffic that enter to the system. But there are many problems that can occur in this existing method. This uses a central point to define the whole security and if the IDS down, it will affect the security of the entire application. In other words, this can be named as single point of failure. Moreover, the performance of the IDS can be affected with the usage of centralized mechanism. This centralized mechanism will lead the application to execute each and every rule set defined for every application type without depend on a specific type that application belongs to. This accumulate more processing power and decrease the performance. Other than that, this can only monitor the traffic when moving to the system and will not be able to monitor the traffic that moving into the namespaces. If the namespaces are compromised this is not being able to address that one. So, this approach can only monitor the one place of the traffic flow and will not be able to detect malicious events occurring at different places at the same time.

This research is focused on introducing a new decentralized model to deploy IDS in a microservice application for performance improvements. The solution is capable of defining separate rule sets for each namespace dynamically, and they are only responsible to monitor the application related to defined namespace only.

Because Kubernetes is one of the most well received container orchestration platform for run the containers like docker, the Kubernetes container orchestration platform was used to do the experiment. In order to maintain an uninterrupted service, the Azure Kubernetes Cluster (AKS) was used.

After deploying the sample containerized web application, Prometheus is used to save the metrics of data received under CPU usage, Memory usage and network latency categories. Then, the Grafana GUI applications are used to obtain the graphs and visualize the obtained results.

According to the output, the previous security model Memory usage was 280MB, but with the new security model Memory usage is only up to 93MB. And, the previous security model CPU usage was increased up to 6.0 but with the new security model CPU usage is increased only up to 3.6.

Moreover, it can be identified that, there is a performance improvement in the new security model rather than using the old approaches.

Acknowledgements

I extremely express my sincere gratitude to my supervisor Dr Kasun de Zoysa and Mr. K.M.Thilakarathne of University of Colombo School of computing for their stimulating guidance, encouragement, useful suggestions and supervision throughout the course of present work.

I also wish to extend my thanks to all the people including lectures and my colleagues for their insightful comments and constructive suggestions to improve the quality of this project work. Last but not the least I express my sincere thanks to all the staff of University of Colombo school of computing who have patiently extended all sorts of help to accomplish this research work.

Thank you.

R,G.K.P.Kulathunga

17440471(2017/MCS/047)

Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Introduction

There are so many security issues occur in the computer systems and there are some methods to prevent these security issues like firewall, Virtual Private Network (VPN), Intrusion Detection System (IDS), etc. IDS can be used to monitor the traffic comes to the applications. The performance of the IDS is different when it applies to the microservices rather than in the monolithic architecture. This research is focused on introducing the new model to deploy IDS in a microservice application for performance improvement.

Intrusion detection system can be identified as firewall security mechanism. The firewall shields the enterprise software from malicious Internet attacks, IDS identifies whether somebody is attempting to access through the firewall or can breach the firewall security, attempts to get to any system in the organization, and cautions the system administrator if there is any undesirable action in the firewall.

Therefore, IDS is a security system that screens network traffic as well as computer systems and attempts to analyze this traffic for potential unfriendly attacks beginning from outside the organization and inappropriate utilization of the system or assaults from inside the organization.

Without real-time detection capabilities, attackers and intruders can lurk inside containers in many forms such as Trojans, malware, ransom, encryption codes. In extreme scenarios spoiling and infiltrating data is possible. Therefore, IDS is a needful element in container environments and quantifying the performance of IDS on container orchestration platform is essential in order to guarantee the seamless operations.

Microservices have become widely popular in recent years, alongside the spread of DevOps practices and container technologies, such as Kubernetes and Docker [1]. We can observe a significant increase in the use of the architectural style of microservices since 2014 [2].

Microservices are autonomous components that isolate fine-grained business capabilities. In addition, a microservice generally operates on its own process and communicates using standardized interfaces and light protocols [3]. In practice, microservices are widely used by large web companies, such as Netflix, LinkedIn and Amazon, which may be motivated by the benefits that microservices bring. For example, the reduced time to bring a new feature into service [3].

The use of microservices has many advantages, such as technological diversity in a single system, better scalability, increased productivity and ease of deployment [4]. Therefore, these advantages can improve the maintainability of the software [3].

The goal of microservices is to use autonomous units isolated from each other and to coordinate them in an infrastructure distributed by a light container technology, such as Docker. Usually, the adoption of this architectural model also implies the adoption of an agile practice, like DevOps, which reduces the time between the implementation of a change in the system and the transfer of this change to the environment of production [3].

It is the fact that, a distributed system is required to work with microservices. The components of a distributed system are in different networked computers or also known as nodes and communicate their actions by passing messages. According to Coulouris distributed system is “a system where the hardware and software components have been installed in geographically dispersed computers that coordinate and collaborate their actions by passing messages between them [5]. Tanenbaum and Van Steen have defined a distributed system as “a collection of systems that appears to the users as a single system”. From Tanenbaum’s definition, it can be conceived that a distributed system refers to a software system rather than the hardware that are involved in creating the system [5].

Container technologies such as Docker and Rocket are examples of application containers, designed to package, isolate, and run applications. This technology provides a faster and better way to deploy and run applications.

Recently, industry adoption of Docker containers has increased to simplify software deployment. Almost in parallel, container orchestration middleware such as Docker Swarm, Kubernetes, Mesos, and OpenShift 3.0 are emerging to support automated container deployment, scaling, and management.

Docker Swarm (Swarm), Kubernetes, Mesos, and OpenShift 3.x are the most popular container orchestration systems for running production-level services.

Kubernetes is one of the most well-received container orchestration platforms for run the containers like docker, rkt etc. Load Balancing and horizontal scalability of containers are some of its features that makes it a reliable and robust solution in multiple domains such as microservices and internet of things.

1.2 Problem Definition

Current deployment method of IDS in centralized way

Currently IDS is deployed in the container orchestration platform as the centralized way as below.

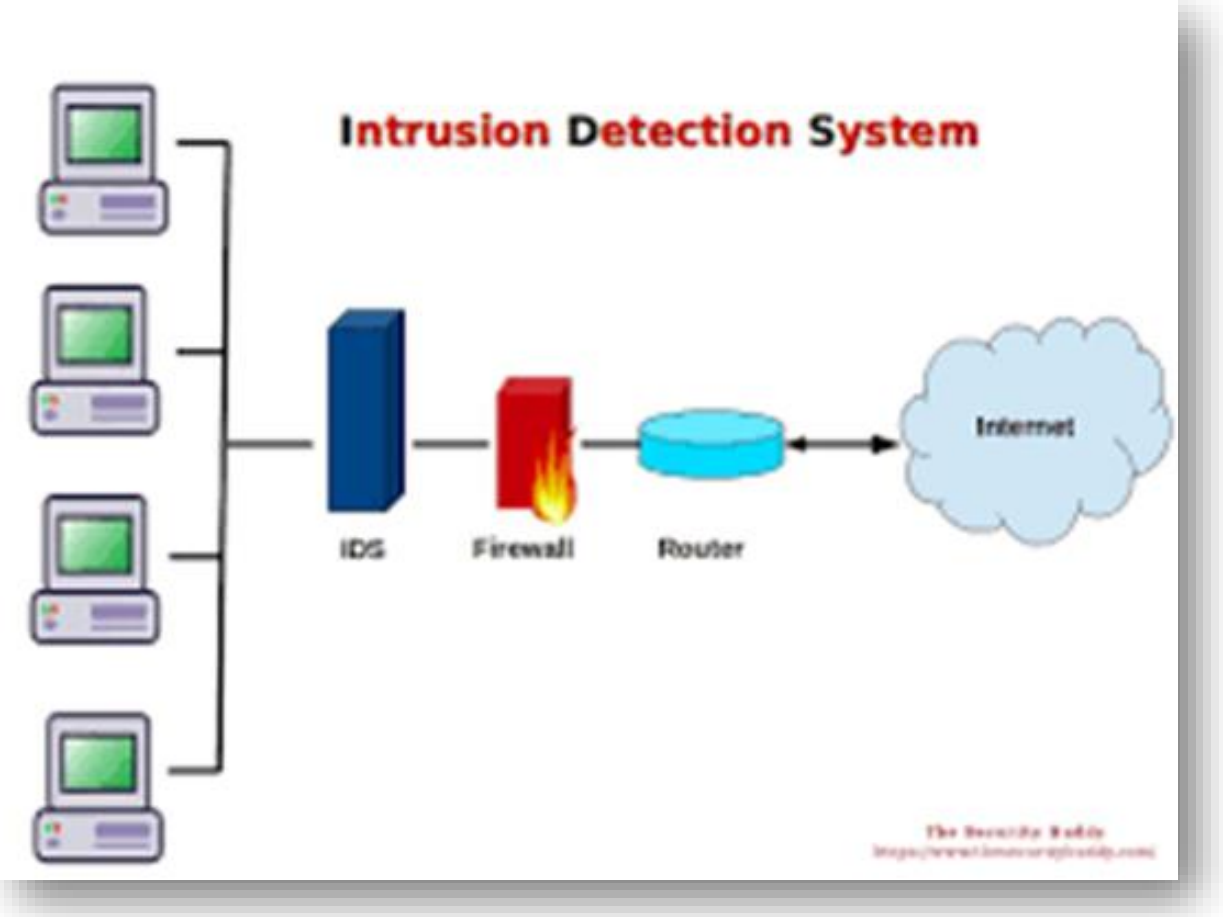


Figure 1.1: IDS in centralized way

Below problems are identified since this has been deployed following a centralized approach.

1. Reduction of performance:

Performance of the IDS will be reduced when it deploys in a centralized way. Because, then it needs to monitor the whole traffic that moving to the system and need to apply all the defined rules for all applications without depend on the specific application type.

2. Not able to detect malicious events occurring at different places at the same time:

This can only monitor the traffic when moving to the system. But it is not able to monitor the traffic that moving into the namespaces. If the namespaces are compromised this is not being able to address that one. So, this approach can only monitor the one place of the traffic flow.

3. Single point of failure:

If the IDS is down, then the whole security of the system is in critical situation.

1.3 Motivation

Computer security systems are more essential for the protection of computing systems and the data that store or access. So computer security systems help people for their critical business processes, inventions, researches, jobs, education etc. Most of this information are personal and sensitive information in day to day life.

The curiosity towards hacking and security systems made me motivate towards this research. Since I need to introduce more efficient and reliable security model using what I have studied, I have decided to create a dynamically scalable security model for containerized deployment platform as a solution for the problem that I have discussed above.

1.4 Research contribution

Container orchestration platforms are used to deploy different services and applications. However, IDSs are not fine-grained to provide specific application-based intrusion detection. Hence, the IDS performance get lagged with respect to memory and CPU as network traffic needs to go through irrelevant signature matching in IDSs. Our main goal of this project to find a solution to improve

the network performance by incorporating an application specific IDS solution for containerized environments.

1.5 Goal and Objectives

The Goal of this study is to introduce a new security model for microservices running platform to overcome the issues in existing security models.

When consider the goal mentioned above, the following are the objectives that need to be fulfilled.

- Improve performance of the IDS
- Security expertise have to focus on specific domain areas only
- Easy to analyze the impact of the specific application area (whether a database application, web application or so on)
- To improve the availability of the system, by using a distributed approach
- Improve the Maintainability of IDS
- Improve the flexibility and scalability of IDS

1.6 Scope

The scope of this research is to Extend the capabilities of Kubernetes container orchestration platform to categorize applications based on their service.

All Microservices that use for the experiment are deployed in the docker containers that provide benefits such as modularity, scalability, distributed etc.

Modularizing Falco IDS to cater Intrusion Detection requirements based on the service provided by the particular container.

Chapter 2: Background/Literature Review and Research Gap

The feature Cgroup was introduced to Linux Operating System Kernel from its Version 2.6.24 in 2007. Which paved the way for Linux Containers [2].Linux namespaces, introduced in the Linux Kernel version 2.4.19, while similar to cgroups that came after it, is different and complementary to cgroups.

At a high level, Linux containers such as Docker, LXC can isolate the application running within the container using the Linux kernel features namespaces and control groups [1]. They are lightweight virtual machines that uses the host Linux OS Kernel and shares the resources, tools, dependencies, application code and settings to function. Control groups and security profiles can apply containers to minimize the attack surface. But since they are using same OS host kernel there will be a chance of compromising the applications that is not possible in the virtual machines.

Docker is a popular containerization engine(hot-scalable) which wraps an application with its dependencies that include all to run code, runtime, system tools, system libraries that can be installed on a VM [1]. This interprets that docker the container does not depend on its run-time environment. Microservice application are container-based service has the below advantages over macro-service (VM based services).

- ✓ Reduce Complexity
- ✓ Scalability
- ✓ Easily deployment
- ✓ Improve the flexibility
- ✓ Enhanced Reliability [6] – Hot scale research paper

Platforms such as Tutum[9], Kubernetes[10], Nirmata[11] provide the platform for run the containerized microservice applications. They have the ability of scale the microservice applications.

Currently many popular companies such as Google, AWS, and Facebook have been using containers for more than a decade. When adopting container technologies to the enterprise level while developing, delivering and deploying many vulnerabilities have been discovered.

Microservices that deploy in the containers have a lot of security issues. One of them is when the application is running inside the container, it needs to interact with the Linux kernel which act as the host and if it is not limited, the container can be compromised. Other than that, the attacker can lurk in to the other containers in the separate namespaces and can compromise the whole container orchestration platform[3]. There are some methods to solve this issue and OS-level virtualization provides the Linux kernel security profiles such as SE Linux, App Armor to minimize the issue[4].

In SE Linux model, the kernel manages and enforces all the access controls over objects, not over their owners. Such as; write policies for enforcement, multi-level security enforcement, multi category security enforcement, etc. Other than that, in SE Linux model everything is controlled by labels. In there, every file/directory, process and system objects has a label. These solutions can fulfill the static security of the container orchestration platform.

Intrusion Detection system is the perfect solution for the network-based security aspect. It is running in a Linux microservices application container environment. Running in a Linux microservices application container environment is much different than running in a monolithic application environment. IDS running in the docker container and how the performance has been affected with the different security perspective web, database is systematically measured [1]. According to that, IDS running in the containers are effective than applying them in traditional networks. But, IDS is using the same centralized deployment approach that is used previously, the limitation of this analysis is that measures were taken only on the IDS container without considering the impact on orchestration platform. Further, another analysis has measured the performance of the IDS based on the functionality, strategy and deployment model as an application without considering the containerized IDS using auto scaling features.

In Network intrusion detection system following major limitations are identified [5].

1. Latency – Need to inspect and blocking action on each network packet

2. Resource Consumption – Usually consume significant resources for some rules
3. Inflexible network configurations – NIPDS are static and not able to automatically reconfigure the networking system and pointed only for specific traffic

There are more deployment design patterns can be used in container based distributed applications other than the centralized approach that previously described. These patterns can be applied according to the scenario to get better performances of them. They are; single container design pattern, sidecar design pattern, ambassador design pattern, adapter design pattern, etc. In single container design pattern, it needs to use to when the container has single responsibility, but if the system needs to fulfill more than one responsibility it needs to use the sidecar design pattern. But according to our scenario IDS needs to work as a proxy for each application pods. It transfers the responsibility to distribute the network load, retries, or monitoring etc. So best design patter for our scenario is the ambassador design pattern [6].

Scalability is the main aspects when working with the micorservice applications. There are different aspects when working with the elasticity such as definition, metrics, tools. Cloud computing provides the capabilites to scale the computing resources up or down without service interruption. This will provide the scalbility with the different metrics, resource availablility, start up time, etc [4].

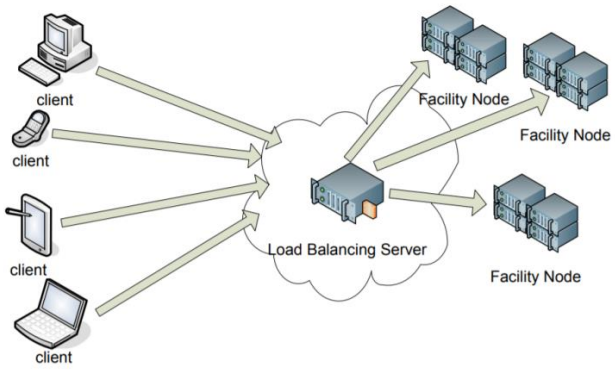


Figure 2.1: Scalability of the cloud

Container based operating system virtualization scalability has a high perfomance alternative to the hypervisors [7]. Linux virtual server provides the better isolation and the superior system efficiency than the hypervisor technology, such as Xen and VM ware. So deploying the IDS in the container based is more scalable than running it in the VM.

Existing network intrusion detection system are monolithic and centralized; therefore, they have limited scalability and responsiveness. So, there are solutions like distributed packet processing and software defined networking control to provide an efficient IDS. But still there are limitation such as;

- SDN based system is the amount of work IDS need to do to check the fields of every packet for all of the applications.
- Bottleneck of the system

Following are the issues identified relevant to the research area

1. Micro services that deploy in containers have security issues
2. IDS performance
3. Scalability of cloud computing
4. Design Patterns in distributed system
5. IDS deploy as the Distributed System

Problem	**Solution**
Performance of the intrusion detection when running in conventional networks as compared to container networks.	IDS running in the containers are effective than the similarly run in the traditional networks[2].
When containers are communicating with the Linux Kernel each one should be isolated.	Linux kernel security profiles introduced such as SElinux and App armor[8].
System is getting out of resources from DOS attack	Use the SELinux model, in contrast, the kernel manages and enforces all of the access controls over objects, not over their owners[8].
Different aspects of elasticity, such as definition, metrics, tools and existing solutions.	In cloud computing provides the capability to scale computing resources up or down without service interruption[9].
Container-based Operating System Virtualization Scalability than the High-performance Alternative to Hypervisors	Linux-Vserver provides better isolation and the superior system efficiency than the hypervisors technology Xen and VMware[9].
Good deployment methodologies for container-based distributed systems	These patterns can be applied according to the scenarios to get the better performance. Single container design pattern, sidecar design pattern, ambassador design pattern, adapter design pattern, etc[6].
Existing NIDPS are monolithic/centralized, and hence they are very limited in terms of scalability and responsiveness	Distributed packet processing, and centralized Software Defined Networking (SDN) control, to provide an efficient and extensible NIDPS[10].

Table 2.1: Summary of literature review

Chapter 3: Research Methodology

The constructive approach has been used when completing the research. First, the real-world problem has been identified and extract the experiments and strategies followed by previous researches from the literature review in order to meet the objectives of the research. Then, by studying the existing solutions further to find out the new solution.

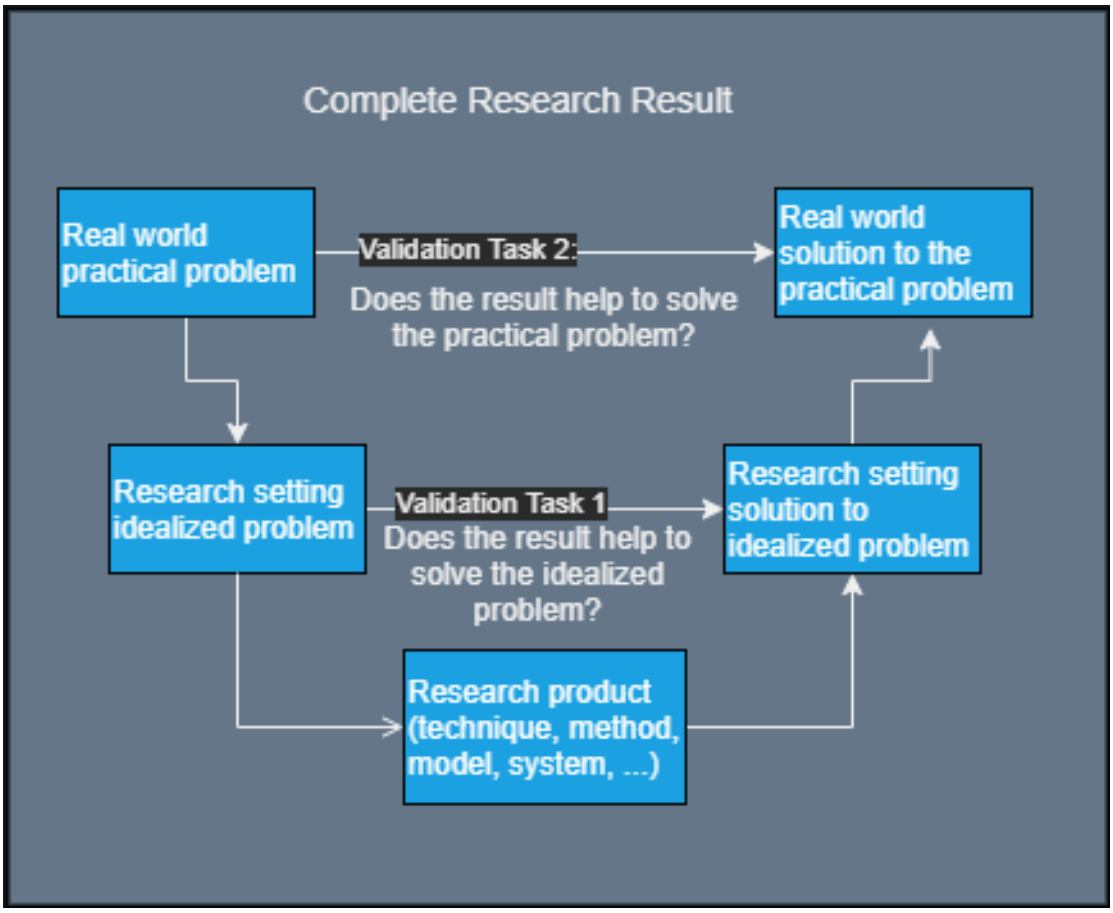


Figure 3.1: constructive approach of research

3.1 Problem Analysis

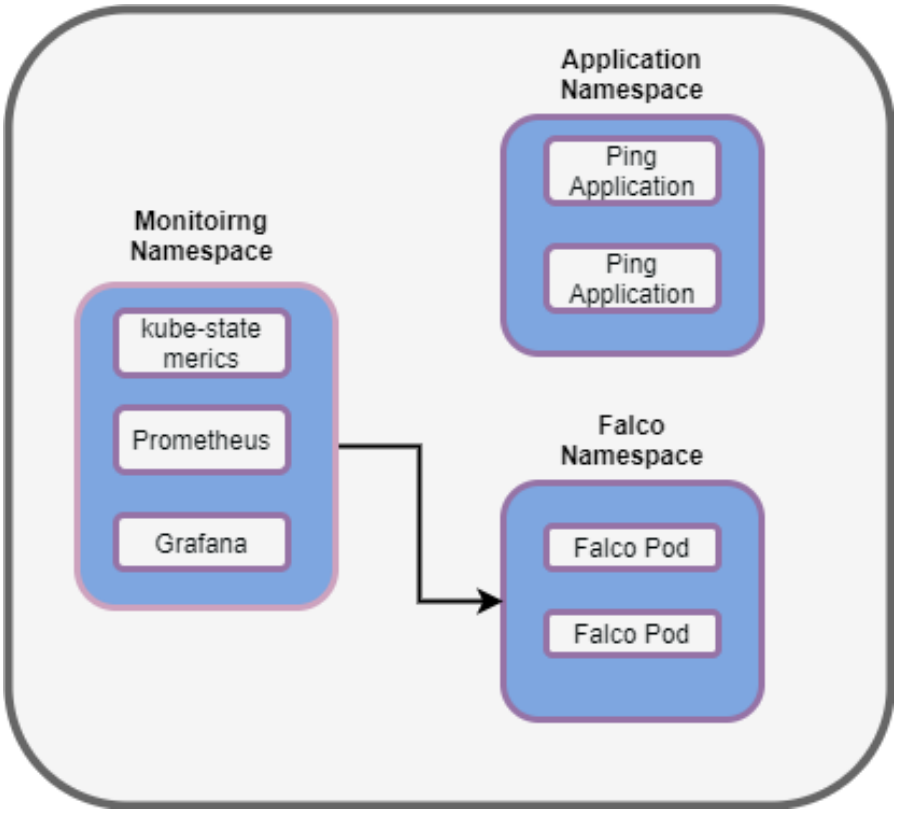


Figure 3.2: Problem Analysis Setup

We have started to evolve the Falco IDS after done some initial investigation of analyzing the CPU usage of the Falco application respect to the sample ping application. The Falco IDS is defined with a sample rule set with respect to the sample ping application we are using in the analysis. The obtained results, which led to initiate the research are mentioned below.

The above Figure 3.2 Ping Application deployed into the application namespace and Falco application was installed into the Falco namespace for monitor the traffic flowing to the system. Then the behavior of the CPU and memory usage of the Falco pod was measured using the Grafana dashboard deployed inside the Monitoring namespace. This was the sample test did to analyze the problem.

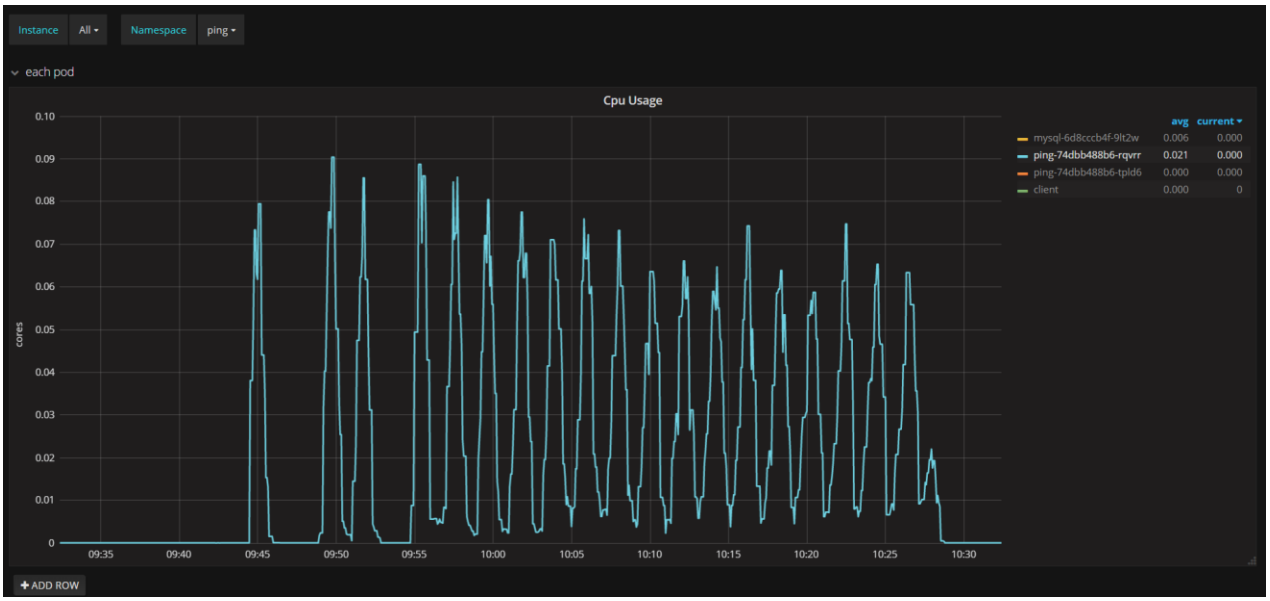


Figure 3.3: ping application CPU usage

The above Figure 3.3 describes how the CPU usage of the sample ping application changed with the increasing load on the same ping application.

Figure 3.4: Falco CPU usage

The above Figure 3.4 describes how the CPU usage of Falco IDS changed with the increasing load on the ping application.

According to the Figure 3.4 CPU usage of the Falco IDS increases with the traffic flow of the sample ping application.

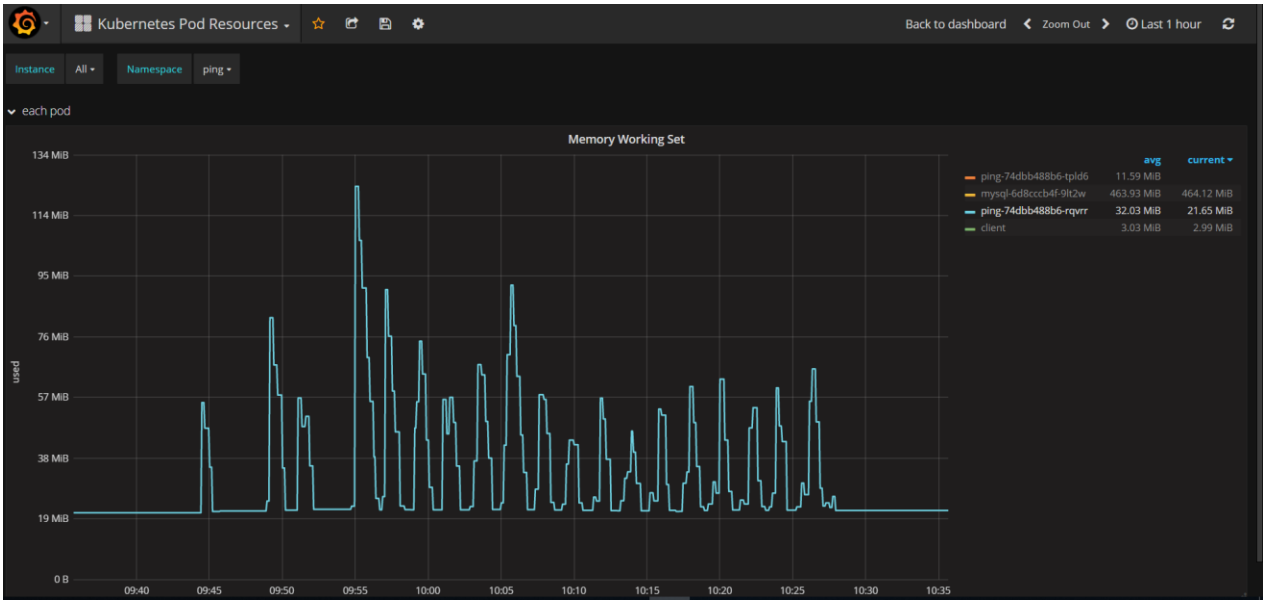


Figure 3.5: Ping application memory usage

The above Figure 3.5 represents how the memory usage of the ping application is increasing respect to the load applied on the sample ping application.

Figure 3.6: Falco memory usage

The above Figure 3.6 describes how the memory usage of Falco IDS changed with the increasing load on the ping application.

According to the Figure 3.6 memory usage of the Falco IDS increases with the traffic flow of the sample ping application.

Accordingly, the CPU usage and memory usage of sample ping application and the Falco IDS is checked against the traffic flow to the ping application and noticed that both memory usage and the CPU usage are increases with the traffic flow.

3.1.1 Research Question

Based on the facts identified from the above experiment, the following are identified as the research questions which we are going to find the solution.

- How the performance of IDS affects the system?
- Is there a way to improve the performance of the IDS in centralized environment?
- What is the best way to improve the performance in centralized environment?

Other than the above, the following can be identified as the concerns of the IDS in centralized environment.

- Is there a performance improvement in terms of memory and CPU consumption of modularized IDSs in the new approach?

- Are the performance improvements gain through new IDS approach integrated to orchestration platform useful for domain experts?

3.2. Proposing Solution

There are number of good Intrusion detection systems which can be used for this research. “OSSEC(Open Source Host Based Intrusion Detection System Security)” is for host based intrusion detection system, “Snort” for the network intrusion prevention and network intrusion detection system and “Strace” is for the monitor linux kernel system calls are some of intrusion detection systems that are focused on specific tasks .

But Falco Intrusion detection system is used here because it provides the combine task of snort, ossec and strace systems.

Falco is the CNCF included project that can be used for the container security[10] and it supports the monitoring of the activities inside a running container. Further, Sysdig falco facilitates capturing all the container host to container and container to container system calls and it defines the highly granular rules in a standard format.

Change the deployment method of Falco IDS in several times by improving the new component created for Falco IDS and accordingly CPU and memory were analyzed. Then, compared this new approach result with the old approach to analyze the performance improvement.

Old approach Falco was installed using the pre-defined rule set and that Rules are defined in the “/etc/falco/falco_rules.yaml” file.

```
- macro: container
  condition: container.id != host

- macro: spawned_process
  condition: evt.type = execve and evt.dir=<

- rule: run_shell_in_container
  desc: a shell was spawned by a non-shell program in a container. Container entry
  condition: container and proc.name = bash and spawned_process and proc.pname exi
  output: "Shell spawned in a container other than entrypoint (user=%user.name con
  priority: WARNING
```

Figure 3.7: Sample Falco Rule

Web application rule set was used for the experimental setup of the previous approach. Then, the load test was done for the web application and the memory and CPU metrics of the Falco IDS were gathered for evaluation.

New component falco-operator was installed when setting up the experiment for the new approach and it will extend the Kubernetes API to create the FalcoRule dynamically.

Those rules can be created and applied for the namespaces in the runtime of the Falco IDS. The similar load test procedure was used here as previous approach.

At the both experiments, Variables of number of pods, ruleset and load tests were remained constant.

3.2.1. Creating the environment

Kubernetes container orchestration platform was used to do the experiment. Because Kubernetes is one of the most well received container orchestration platform for run the containers like docker, rkt etc. Load Balancing and horizontal scalability of containers are some of its features that makes it a reliable and robust solution in multiple domains such as microservices and internet of things.

Azure Kubernetes Cluster (AKS) was used because on-prem Kubernetes cluster needs to maintain. If the cluster is getting down during the experiment because of any reason that can be affected to result of the experiment. AKS cluster is maintaining by the Azure. Therefore, it is reliable.

Sample containerized web application deployed. Then to save the metrics used the Prometheus database and visualized deployed the Grafana applications.

3.3 Evaluation

From the metrics gathered by the experimental results memory and CPU with the new approach has been reduced compare to the old approach. From the survey selected the convenient sample that has the good idea about the IDS usage.

3.3.1. Result Analysis

Analysis from the results gathered new security model of the IDS has been improved the performance than the old approach.

A summary of the obtained results is as follows.

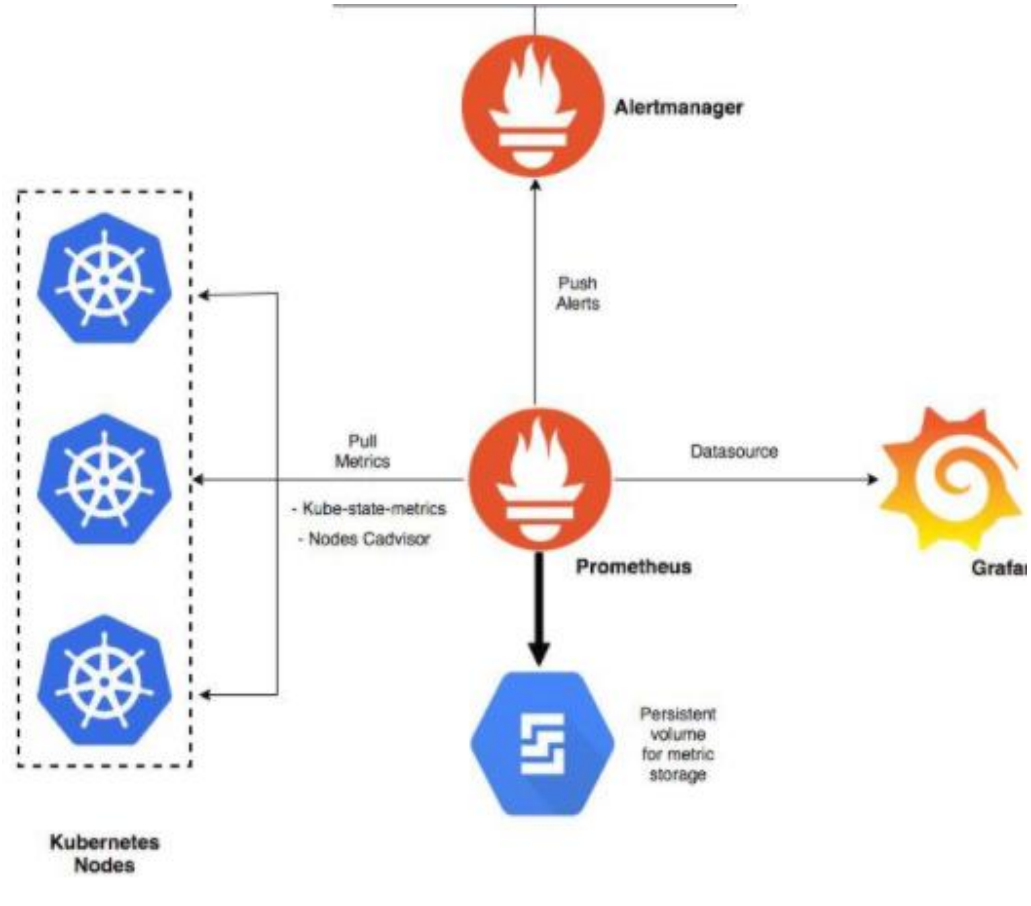


Figure 3.8: Model low-level design

Gather the data set of CPU and memory of a microservice application which is deployed inside a container orchestration platform that is not straight forward.

Therefore, the kube-state-metrics application was installed and used to gather the data set.

Cadvisor was gathered the metrics from the Kubernetes API server that exposing the metrics from the Kubernetes application pod level. It installed default with the Kubernetes. Kube-state-metric application gather the metrics from the Cadvisor that installed default with the Kubernetes. Prometheus is gathering all the CPU, memory of the applications from the kube-state-metrics. The CPU and memory metrics of the load testing web applications and Falco intrusion detection system was gathered by the experiment.

3.3.2. Survey

A survey is done to make sure the objectives of the research is expected. This survey is done using a convenient sample. The sample is selected from the expertise in the IT industry related to security. From the result of the survey proved that below expected objectives has been achieved.

- Security expertise have to focus on specific domain areas only
- Easy to analyze the impact of the specific application area (whether a database application, web application or so on)
- To improve the availability of the system, by using a distributed approach
- Improve the Maintainability of IDS
- Improve the flexibility and scalability of IDS

Chapter 4: Proposed Solution

Lot of companies are using different methodologies to secure their application from hackers. IDS is providing runtime security for the applications. Some IDS have minimal performance and some of them have reduced the performance due to the deployment model.

Currently, IDS is deployed in the container orchestration platform as the centralized way. This IDS security system checks all the traffic that comes from an outer environment. It checks all the rules although it is necessary or not. As an example, if we use a web-based application, it is sufficient to check rules for web-based applications. But this IDS checks all the applications such as web application, database application, multimedia application etc.

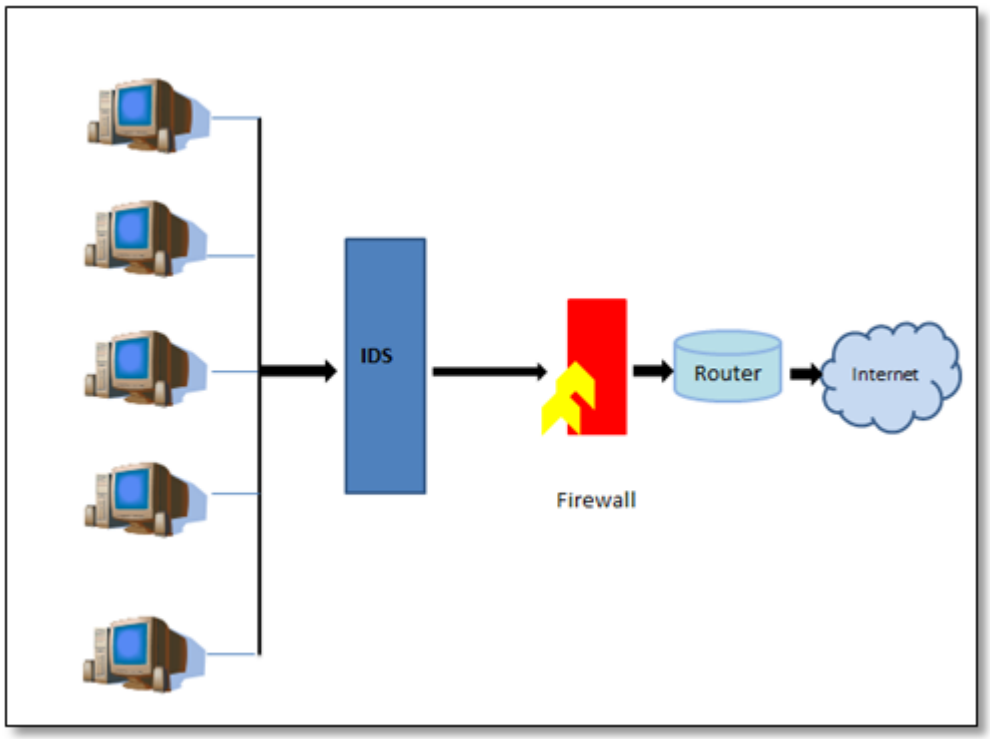


Figure 4.1: IDS in the centralized way

Above Figure 4.1 presents the high-level overview of the approach used in earlier systems, which is identified through the conducted literature review. Accordingly, the traffic is flowing from the

internet to the application through firewall. Then the IDS sits between the firewall and the application/system.

In this approach, IDS includes rules respect to all the applications in the system and whole traffic needs to monitor with each rule defined for each application.

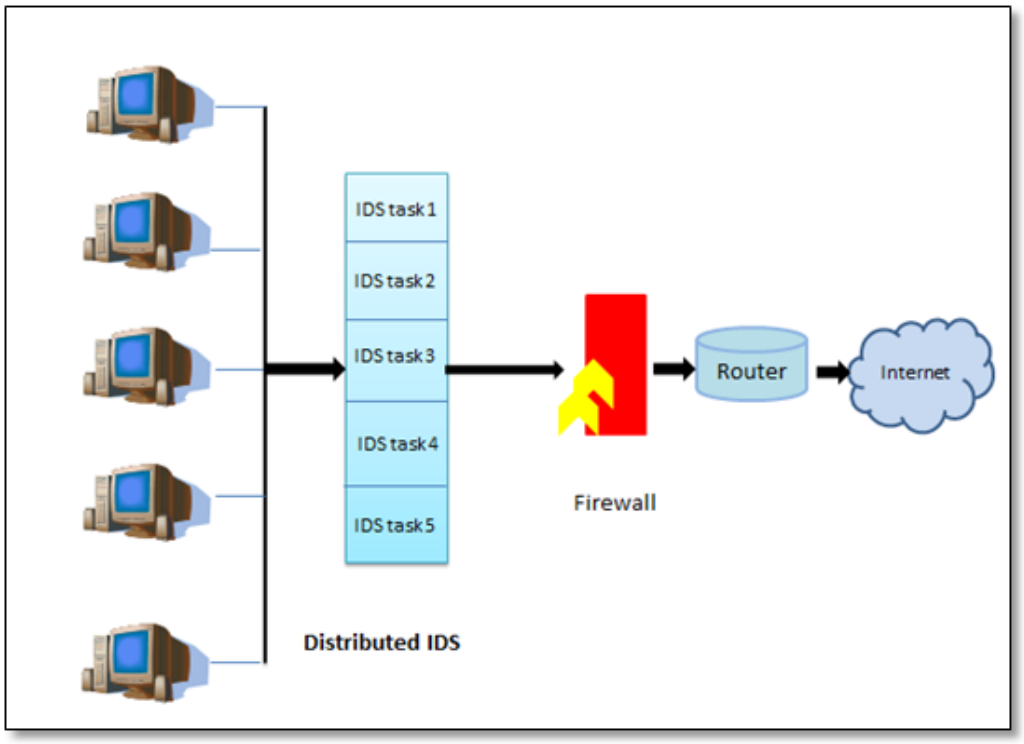


Figure 4.2: Deploy the IDS new model

The above Figure 4.2 represents the proposed model IDS. It monitors the traffic when they are moving to the namespaces database, web etc. It doesn't monitor the whole traffic respect to the all rules. IDS rules deployed as the distributed way according to the category of the applications.

Based on the findings from the literature review and the experiment we have proposed the below solution.

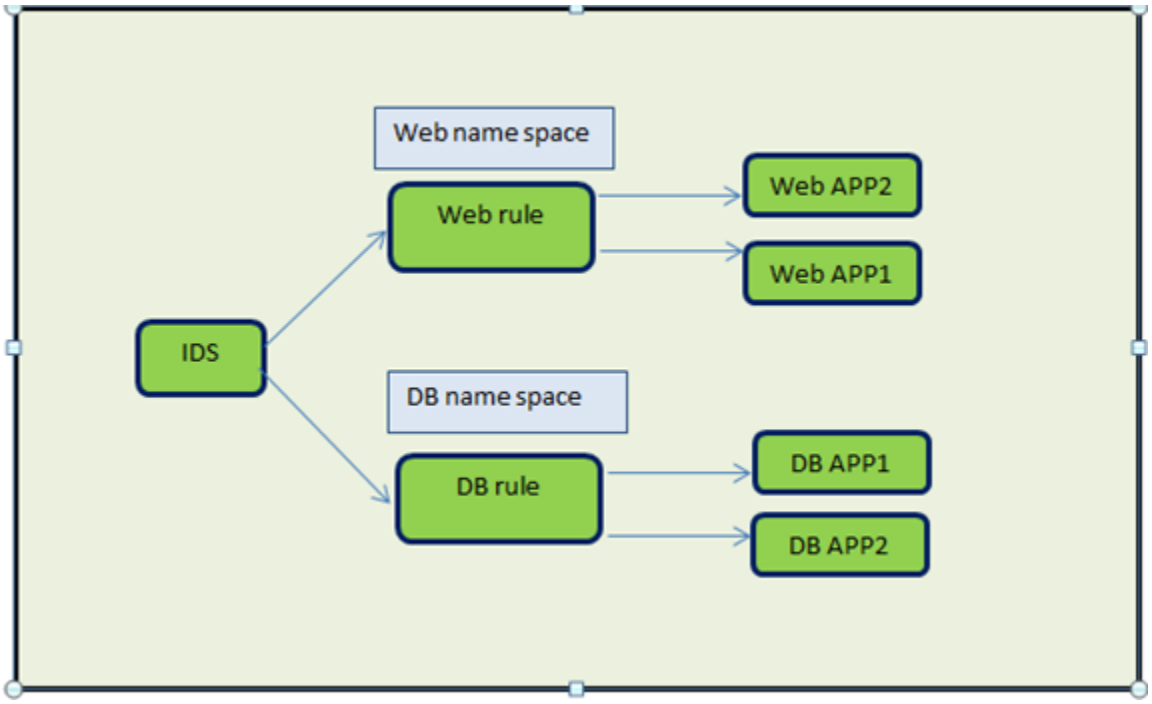


Figure 4.3: Model low-level design

We propose a new method for deploy security applications for container orchestration platform.

The approach is to design a security model based on decentralizing the traffic monitoring for intrusions according to the application categories in relevant namespaces. Namespaces can vary how the security engineer or person who is responsible define the whole system.

For example, if the system is defined as namespaces such as web, databases, storage, etc. Web namespace includes web applications Nginx, php, etc. The database includes MongoDB, Redis, Elasticsearch, etc.

The proposing solution can define the separate rule sets for each namespace. They are only responsible to monitor the traffic moving to the relevant namespace.

The new approach to design the model based on the idea that has been divided into two parts. Falco security application that define the ruleset and Falco operator application that is watching the creation of new rule and force it to only monitor the relevant namespace.

2. Differentiate the proposed solution with the existing solution

In the approaches used before, there is only one part of the application. Falco security application rules can be only defined for the whole system. So, Falco application should monitor whole traffic respect to all the rules that moving the system. As an instance, Falco application needs to apply the database rule for web application traffic as well.

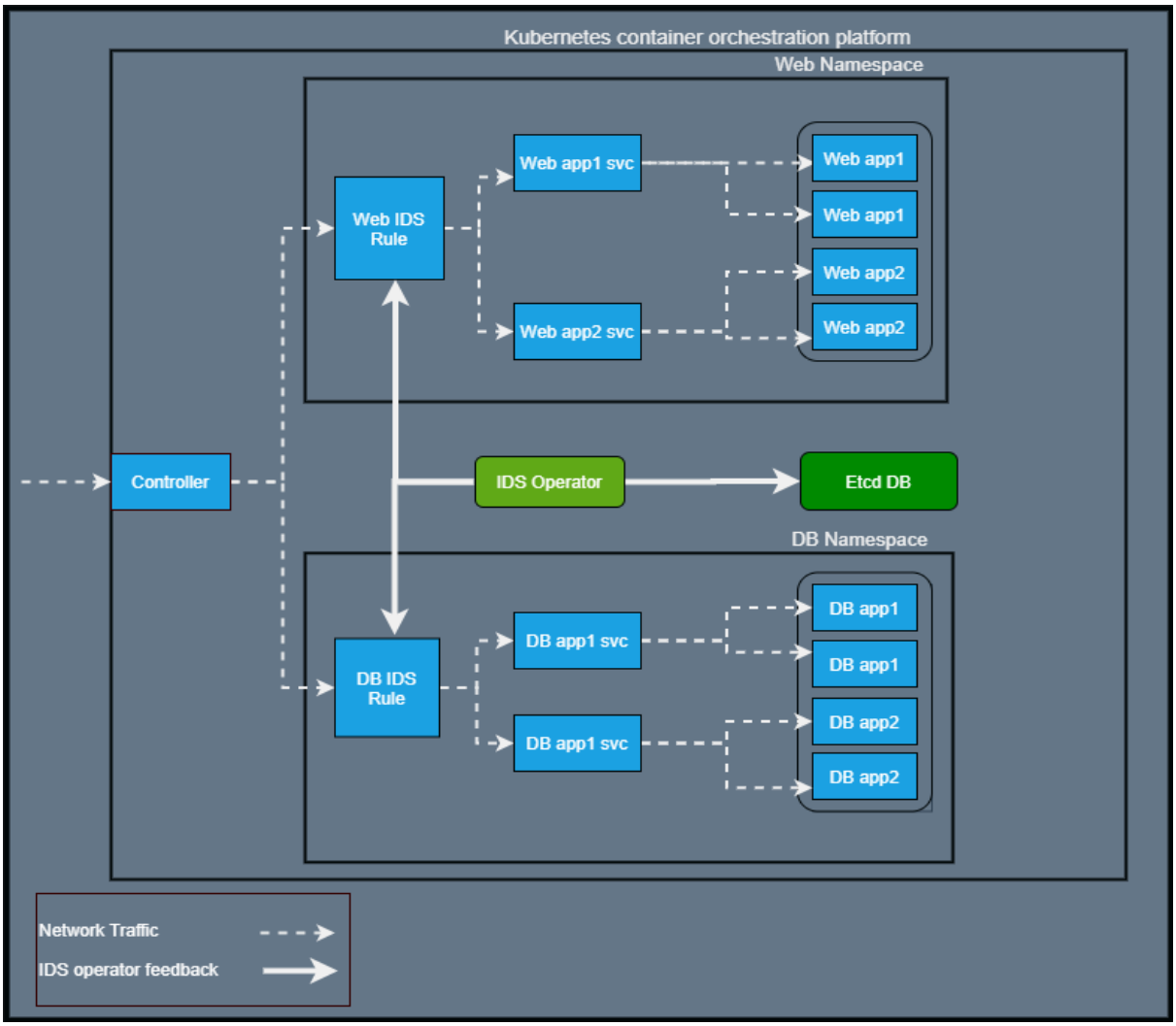


Figure 4.4: Design Diagram

The proposing solution is to introduce the new component IDS-operator. When the IDS operator installs in the Kubernetes container orchestration platform, it will extend the Kubernetes API. Likewise, a new API is introduced as FalcoRule. Using that API can create the rules for specific namespaces, such as; web application, DB application, etc.

Traffic will flow from internet to the system and controller will forward the traffic to relevant namespace. IDS operator will listen to IDS behavior. When traffic flow through each namespace, the IDS operator will feed the IDS to apply the rules for the traffic that moving to the relevant namespace.

As an instance when the traffic moving to the web namespace IDS operator feeds the IDS to check the below rule.

```
apiVersion: "kakulk.falco.io/v1alpha1"
kind: "FalcoRule"
metadata:
  name: "bash"
  namespace: "web"
spec:
  rule: shell_in_container
  desc: notice shell activity within a container
  condition: container.id != host and proc.name = bash
  output: shell in a container
```

Figure 4.5: Sample Falco rule for web namespace

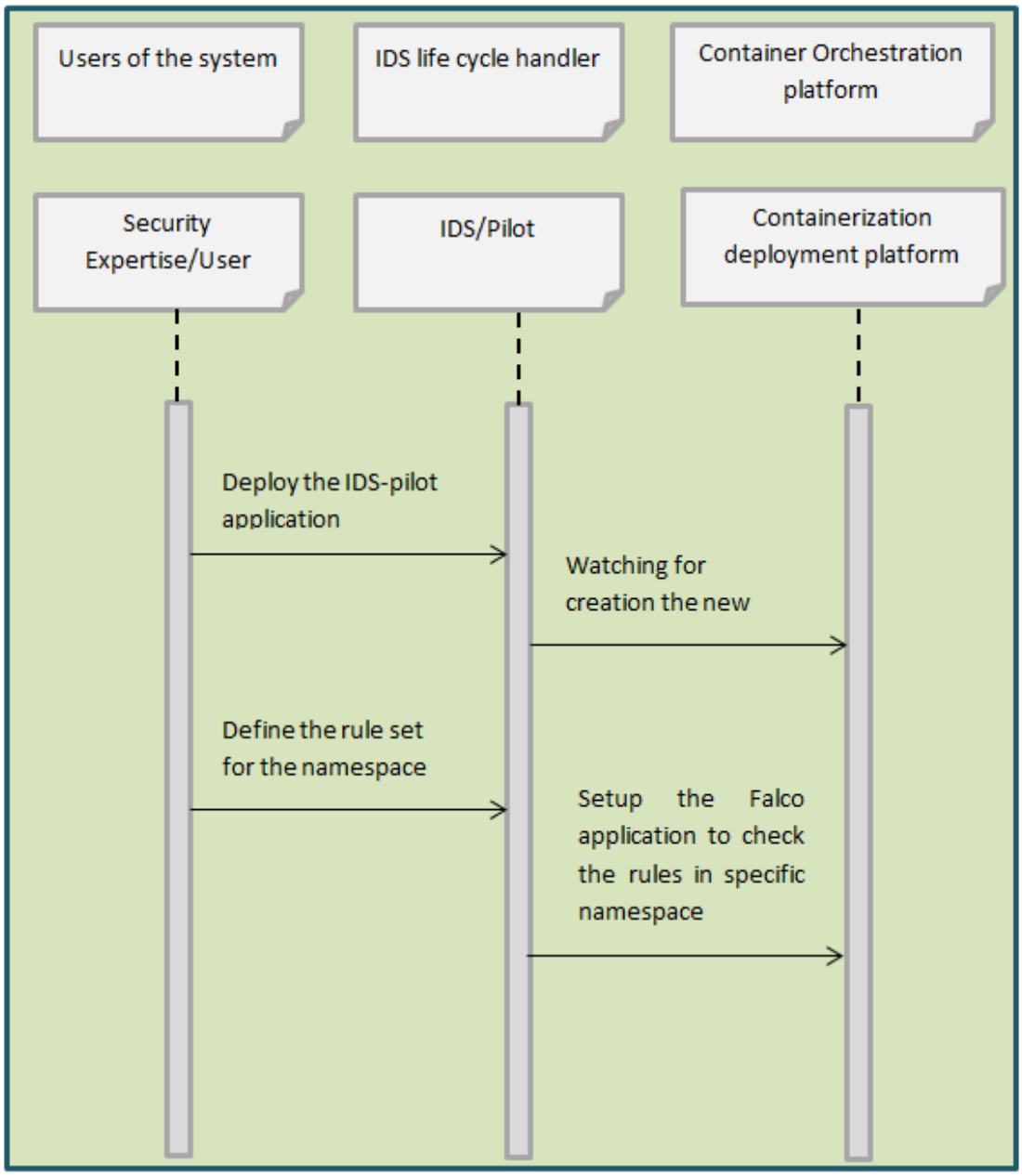


Figure 4.6: Sequence Diagram

The above Figure 4.5 includes the new proposed solution application flow to identify usage of the new approach.

Security of the system is defined by the security experts. At that time, he/she can define the applications category of the whole system and separate with each other. As an instance web application into the one namespace and DB application into the DB namespace.

So, each namespace web, DB can define the security separately because of they are isolated with each other. DB ruleset can define by the database security expert and web database security can define by the web application security expertise. They are not conflict with each other.

After installation of the Falco-ids-operator it will watch the creation of the Falco security rules and do the needful to monitor the traffic across namespaces.

Falco is a good Intrusion detection system that most companies currently use[10]. It has the capability to detect the security vulnerability of the applications in run time that deploy in the container orchestration platform like Kubernetes, Mesos etc. This can detect anomalous activity in hosts and containers. Rules can be defined for securing the database, web application etc. and rules built using tcpdump packet capture like syntax.
Currently, Falco is deployed in a centralized way in the container orchestration platform as previously described. All the rules need to be written separately for different application types, such as; database applications and web applications. In the existing systems, As an example database, web etc. In existing systems, Falco IDS can't define the rules according to their category.

In the proposed system, Falco IDS can be deployed according to the category of applications. This will increase the performance by not checking all the rules at once.

As the solution for the identified problem, a new component called falco-operator has been developed. There are a variety of components in Kubernetes deployment, services etc. This one is created as a new kind ids-operator to create new rules according to the application category.

Ids-operator component will monitor the traffic by listening to Kubernetes API. For an example, when the new rules are created for the category 'database', it will automatically apply if the database category uses the term of Kubernetes namespace. The namespace of the Kubernetes helps to separate the application sets from each other according to their characteristics.

Further, the ids-operator guides Falco applications to monitor the traffic when traffic only moves to the relevant namespaces, such as; database, web, etc.

The ids-operator is using the capability of Kubernetes container orchestration platform deployed in Microsoft Azure AKS. AKS is easily managing and deploying containerized applications. So that, it will reduce the effect of the performance in the application by container orchestration platform parameters with the use of added firewall capability using Network Security Group (NSG). NSG is working as a firewall for the system and that prevents hackers from hacking into the system. This system created as the virtual private server; therefore, this could not be accessed from the public and the load should be provided only for testing for the research purpose.

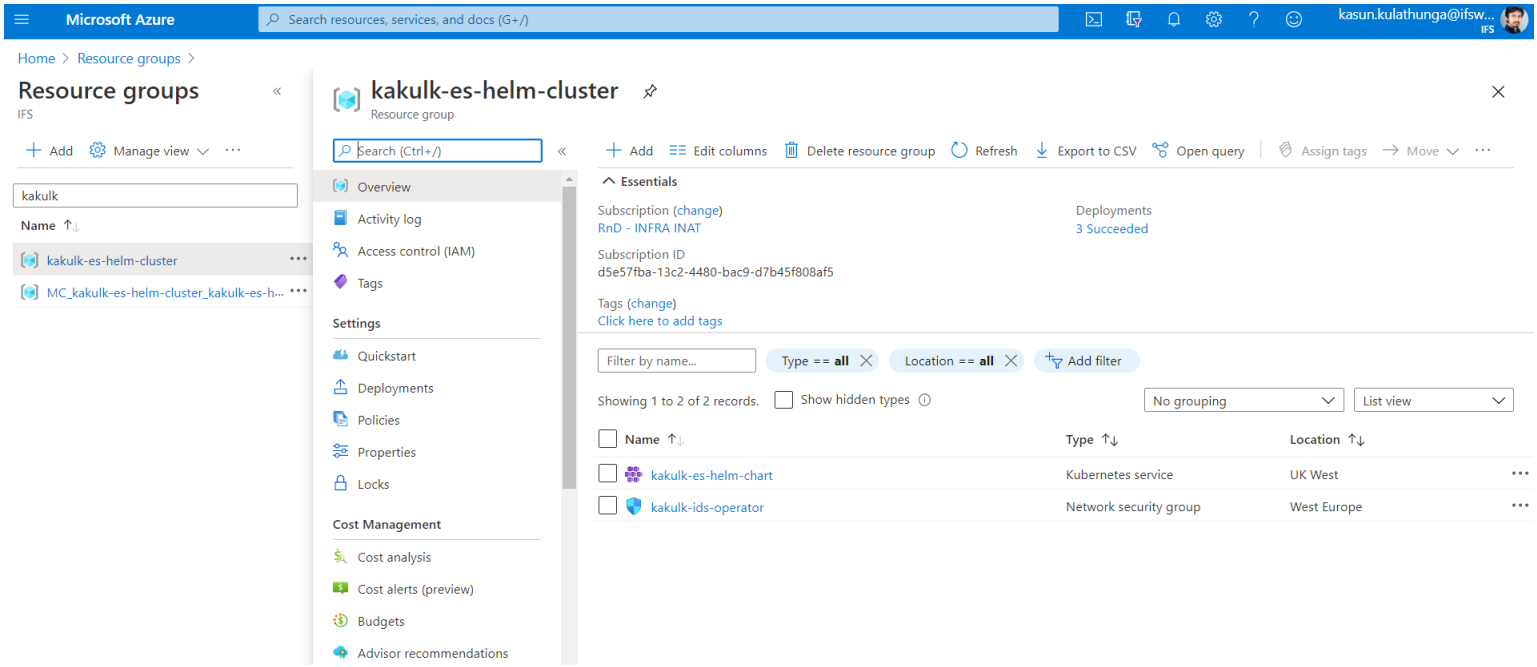


Figure 4.7: Research setup in Azure cloud

The Falco IDS has the rules to monitor the traffic and it has different rule sets for different application types, such as; database, web etc. So, the security expertise can focus on specific domain areas and it will avoid unnecessary checks on the data packets, and it will improve the performance.

Cloud-Native Security Hub is providing the sample rules that can be applied for different kinds of applications. Sample rules can be found below.

```
# Considering any inbound network connection suspect

- rule: Unexpected inbound connection php_fpm

desc: Detect any inbound connection arriving at php_fpm

condition: inbound and evt.rawres >= 0 and app_php_fpm

output: Unexpected inbound connection arriving at php_fpm (command=%proc.cmdline
pid=%proc.pid connection=%fd.name user=%user.name %container.info
image=%container.image)

priority: NOTICE
```

Figure 4.8: Sample Falco Rule (PHP application rule)

Analyze the performance of the system, and generate the evaluation results under CPU, memory and network latency categories. The Prometheus, Grafana is used to monitor the metrics changes with the time and to generate the evaluation result.

The Kube-state-metrics service is used to gather the metrics. It is a simple service that listens to the Kubernetes API server and generates metrics in the objects. Prometheus is collecting data from the kube-state-metrics service and can show in the Grafana.

The IDS-operator is used to extend the API in Kubernetes. It uses the operator SDK, a software development kit that can be used to extend the API in Kubernetes using the Helm package manager underlying GO language.

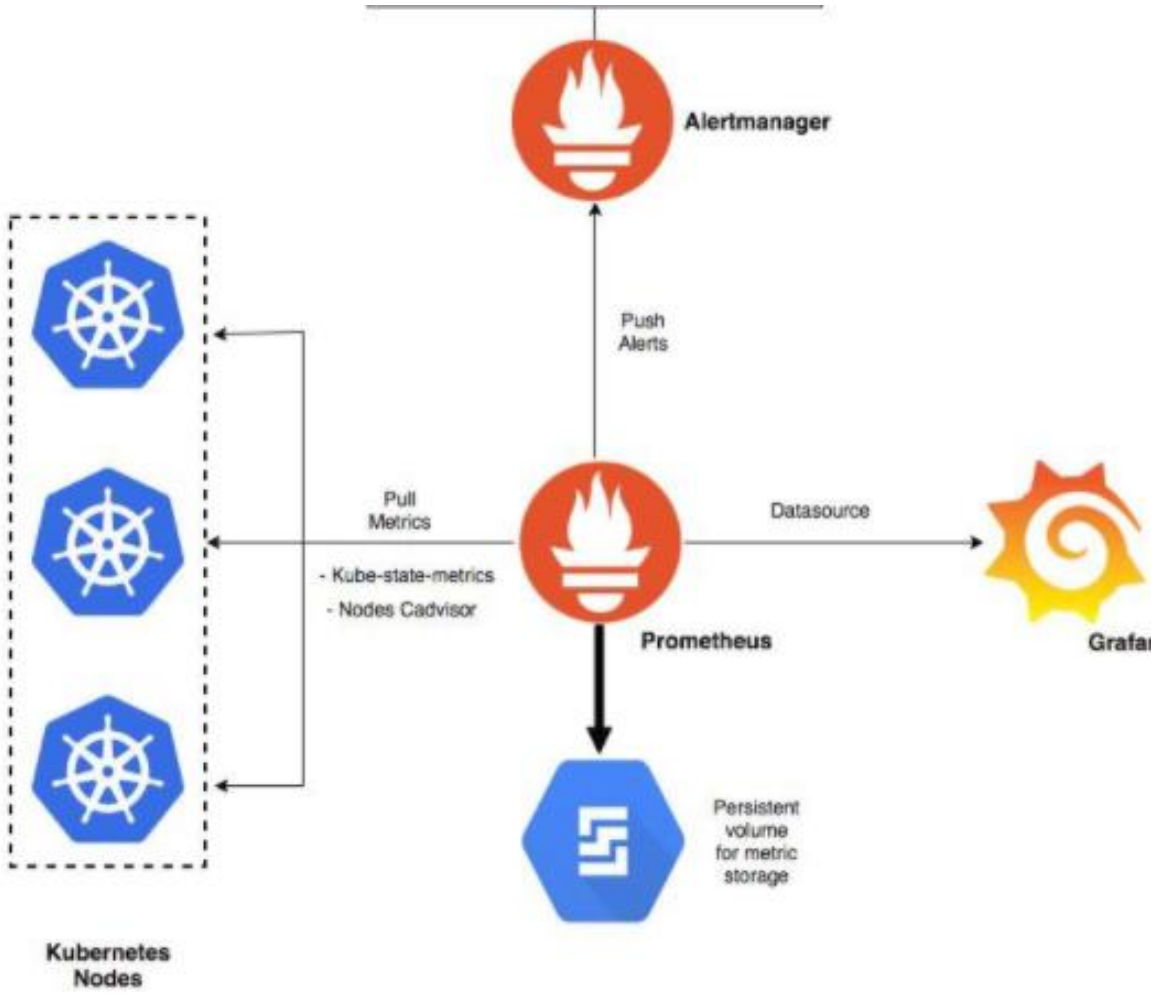


Figure 4.9: Monitoring solution to check the performance

IDS-operator provides an API to create the ids-operator component using Kubernetes commands. After IDS-operator integrates to the Kubernetes API. Any developer can create the ids-operator component in the Kubernetes cluster. That ids-operator is watching and monitoring of rule creation. If we create the rule for the web application namespace ids-operators component, set up the rules in the Falco IDS applications to monitor the web application rules when traffic moves to any application in web namespaces.

Chapter 5: Evaluation and Results

This research is carried out to develop a new approach to run the Intrusion Detection System in container orchestration platform. Hence this approach and the concept has to be evaluated with the respective audience. Because the project evaluation is a process used to determine whether the design and delivery of a project were effective and whether the proposed outcomes were met. Therefore, the opinions of IT professionals Software Engineers, Software Architects, Security Experts, Team Leads were considered for the evaluation criteria since they have a good understanding of the security aspects of Software.

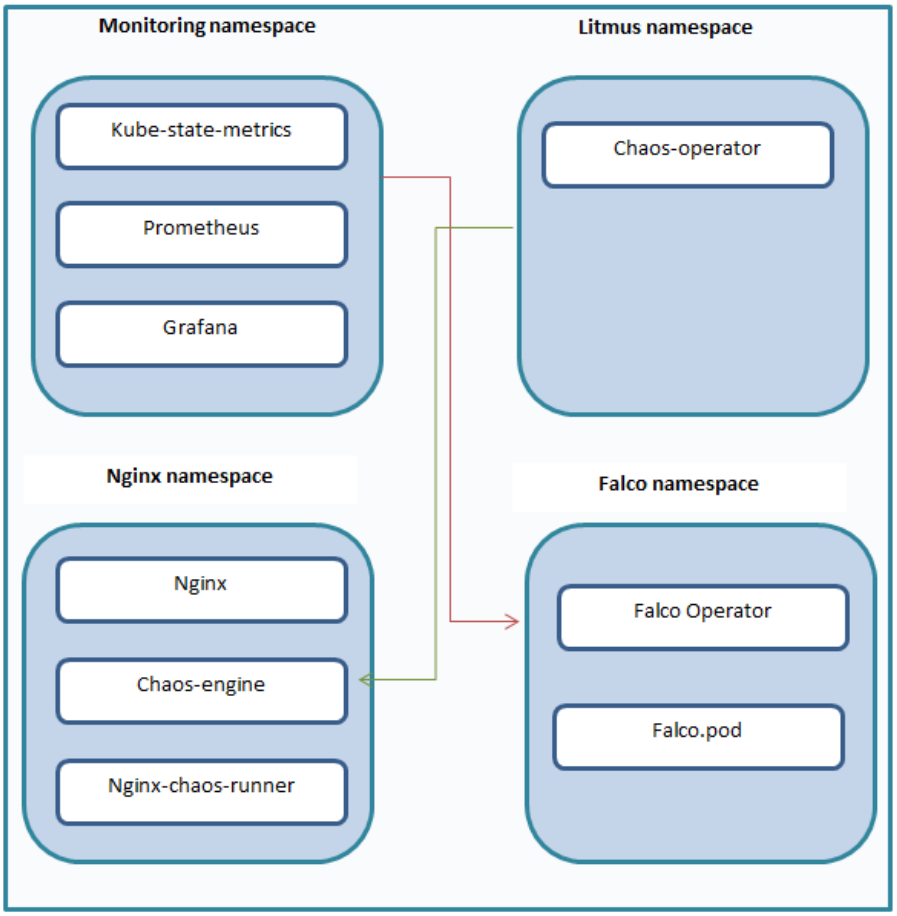


Figure 5.1: Evaluation Setup

According to the above Figure 5.1 the evaluation setup included with 4 namespaces and they are mentioned below with a simple definition.

1. Nginx Namespace – Install web applications in this setup Nginx
2. Falco Namespace – Falco IDS application
3. Litmus Namespace – Load Test application
4. Monitoring Namespace – Output the monitoring metrics memory/CPU of the applications

For the performance evaluation of the new model did some comparison with the previous approach using the same setup.

First, set up the container orchestration platform in Azure Kubernetes Cluster inside the virtual private server to prevent from accessing by public. As the firewall using the Network Security Group (NSG) in the Azure cloud to minimize the vulnerabilities by closing the unnecessary ports.

Then, run the applications in the platform. The web applications were deployed to the web namespace ‘ns’ and the database applications were deployed to the namespace ‘db’. Then, apply the rules for both web and database applications. For the web applications used the PHP-FPM Falco rules set and for database application used the Redis Falco rule.

Chaos litmus application used to perform the load testing on the applications. By using this application, the load applying on the application CPU and memory can be changed.
The kube-state-metrics application is deployed in the container orchestration platform to collect the CPU, memory and network metrics from the applications and they are stored in the Prometheus time-series database and generate the graphs by using the Grafana GUI application which is connected to Prometheus database.

In the new model, the deployment rules are created using the Falco Rule component API that created for the Kubernetes. When deploying Falco-IDS, the rules are set up automatically for each namespace in Falco applications.

5.1 Evaluation Result

Evaluation is done respect to the memory usage and CPU usage with the both setups with previous model and new model of the IDS deployment.

Figure	Security Model	Load Test Application	No of web pods	Load Test Metrics	Applied Rules	CPU Usage	Memory Usage(Maximum)
Figure 5.2	Previous	Web	2	Memory	Web/DB	X	280 MB
Figure 5.3	Previous	Web	2	CPU	Web/DB	6.0	X
Figure 5.4	New	Web	2	Memory	Web/DB	X	93 MB
Figure 5.5	New	Web	2	CPU	Web/DB	3.6	X

Table 5.2: Evaluation Setup

According to the output, the previous security model Memory usage was 280MB, but with the new security model Memory usage is only up to 93MB. And, the previous security model CPU usage was increased up to 6.0 but with the new security model CPU usage is increased only up to 3.6.

Moreover, it can be identified that, there is a performance improvement in the new security model rather than using the old approaches.

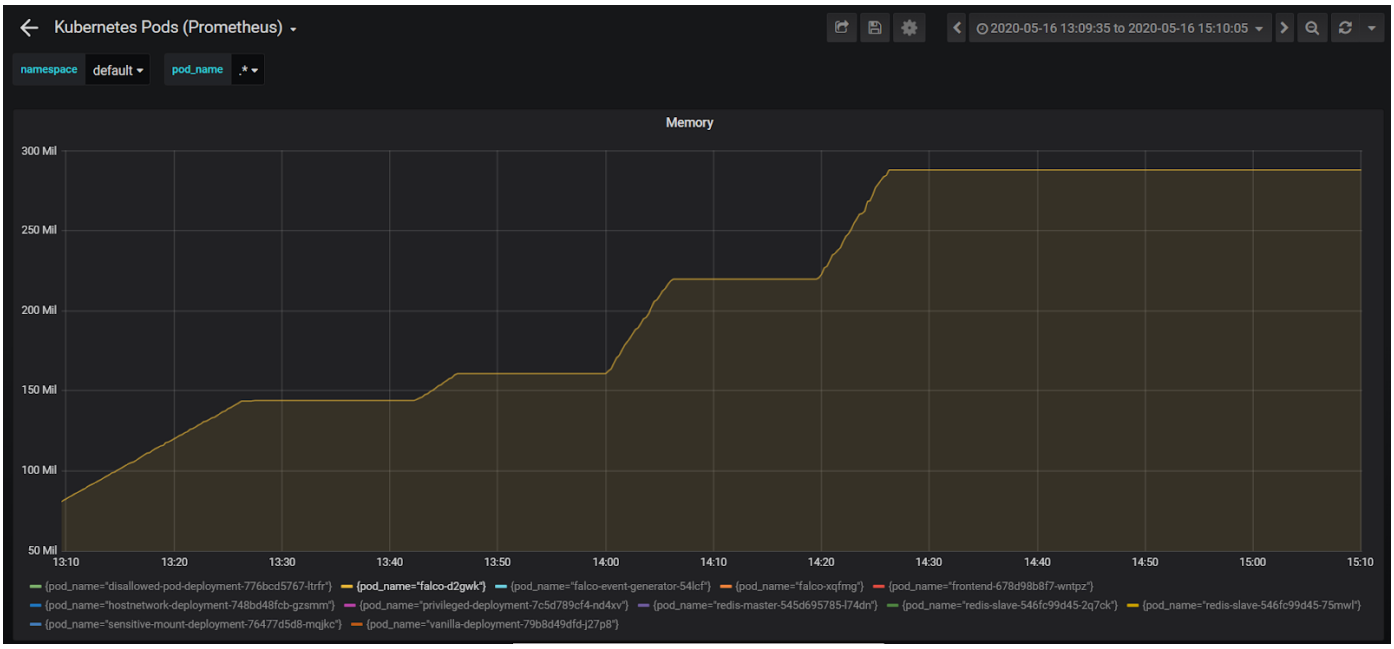


Figure 5.2: Previous security model memory usage of Falco application pods

The above Figure 5.2 represents the memory usage of the Falco applications deployed in the previous model. When do the load test for the web application memory and number of pods are using 2 in all scenarios. Memory is increased up to the 280MB in this case. Because Falco application apply web, db both rules for traffic flowing to the web application.

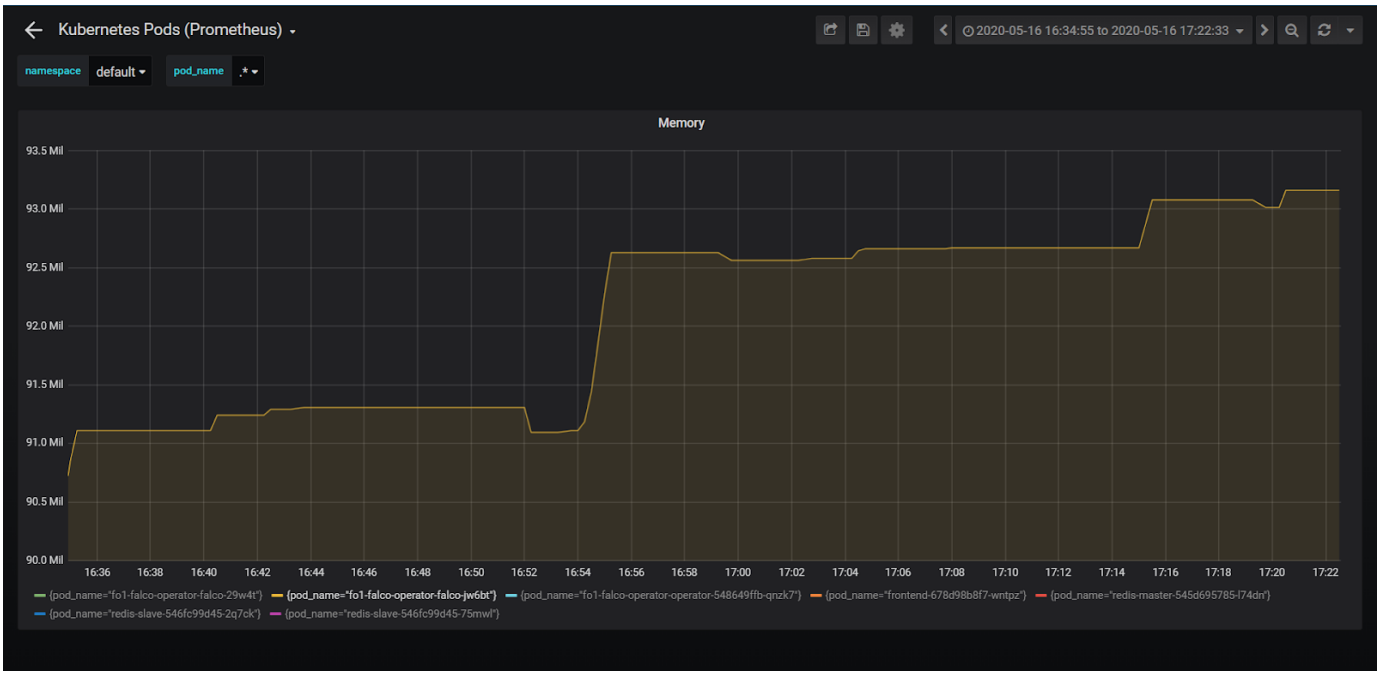


Figure 5.3: New security model memory Usage of Falco application pods

This above Figure 5.3 represents the memory usage of the Falco applications deployed in the new model. When do the load test for the web application memory and number of pods are using 2 in all scenarios. Memory is increased only up to the 93MB in this case. Because with new approach Falco application does not apply the DB rules for traffic flowing to the web application.

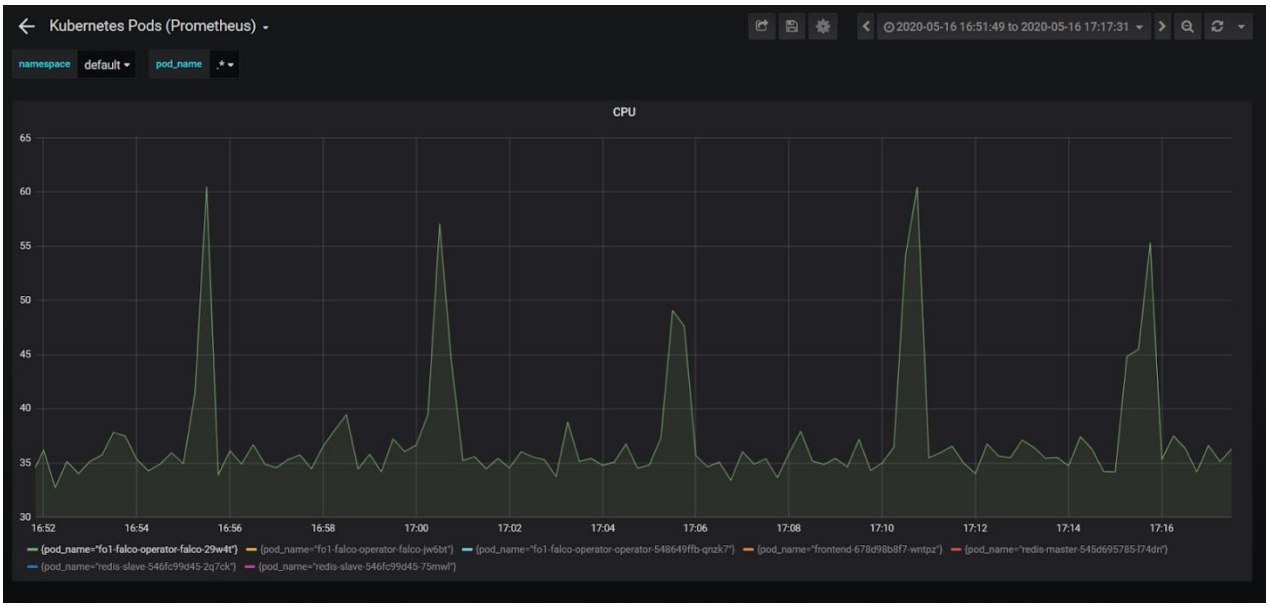


Figure 5.4: Previous security model CPU usage of Falco application pods

This above Figure 5.4 represents the CPU usage of the Falco applications deployed in the previous model. When do the load test for the web application CPU and number of pods are using 2 in all scenarios. CPU is increased up to the 6.0 in this case. Because Falco application apply web, DB both rules for traffic flowing to the web application.

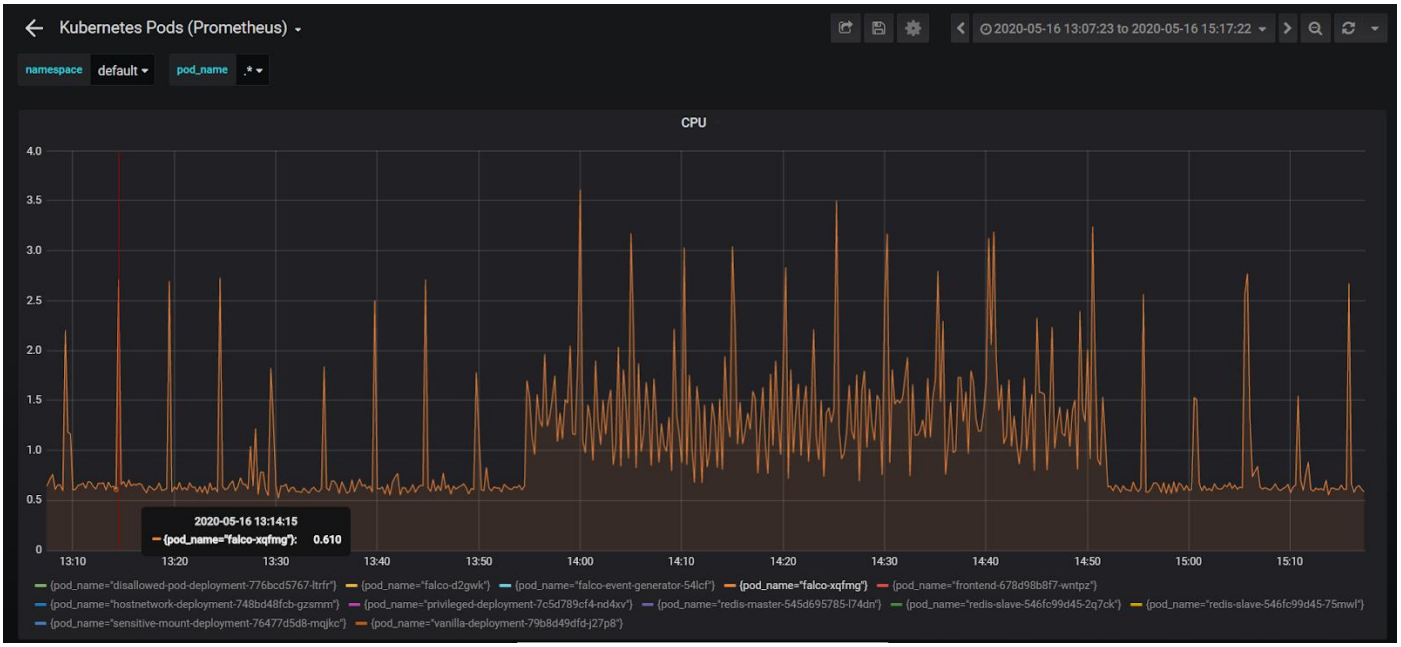


Figure 5.5: New security model CPU usage of Falco application pods

This above Figure 5.5 represents the CPU usage of the Falco applications deployed in the previous model. When do the load test for the web application CPU and number of pods are using 2 in all scenarios. CPU is increased up only to the 3.5 in this case. Because with new approach Falco application doesn't apply the db rules for traffic flowing to the web application.

5.2 Survey

The survey has been done for the IT professionals who are working in different companies with IT security expertise. There are 5 questions included in the survey and each question has 4 answers. These 5 main questions have counted for the final results and each question can get a maximum of 4 marks. Here is the mark categorization for the 4 answers.

Answer	Mark
Strongly Yes	4
Yes	3
Maybe	2
No	1

According to the results obtained from the survey, the detailed summary of each question can be listed down as below. According to the summary of each question, we can clearly get an idea about the result of the survey.

The output extracted from the all results obtained from the survey are stated below.

1. Easy to analyze the impact of the specific area
2. Security expertise need to only focus only on the specific area
3. No single point of failure in this new model and it improves the availability of the system
4. New solution will improve the flexibility and scalability of the system
5. New solution will improve the maintainability of the system

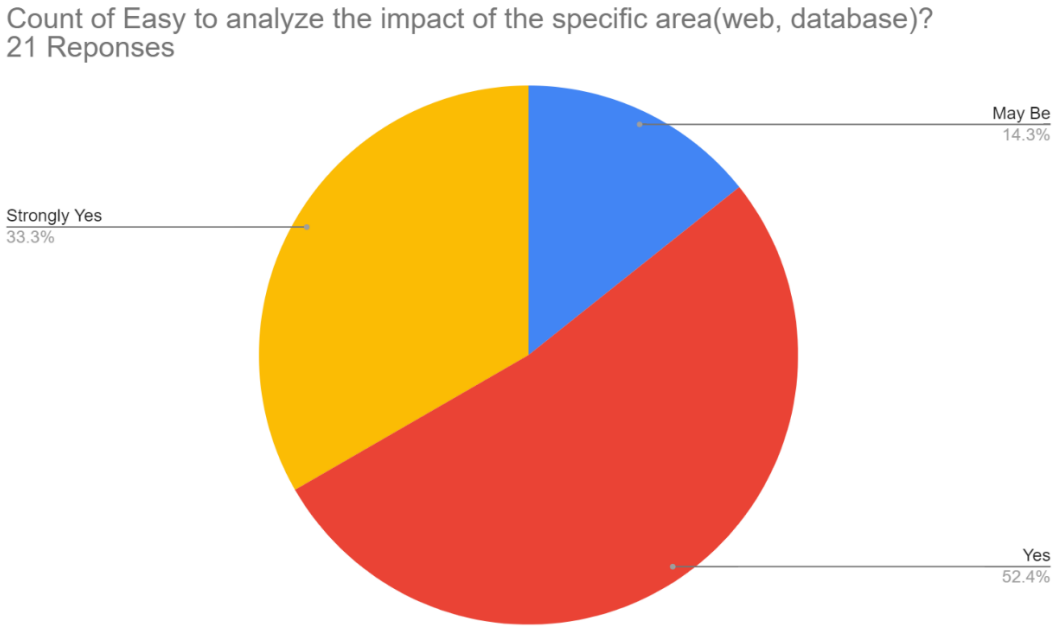


Figure 5.6: Easy to analyze the impact of area

This above Figure 5.6 proves that users are mostly agree new system is easy to analyze the impact of the area with the system is categorized into the separate security areas web, database etc.

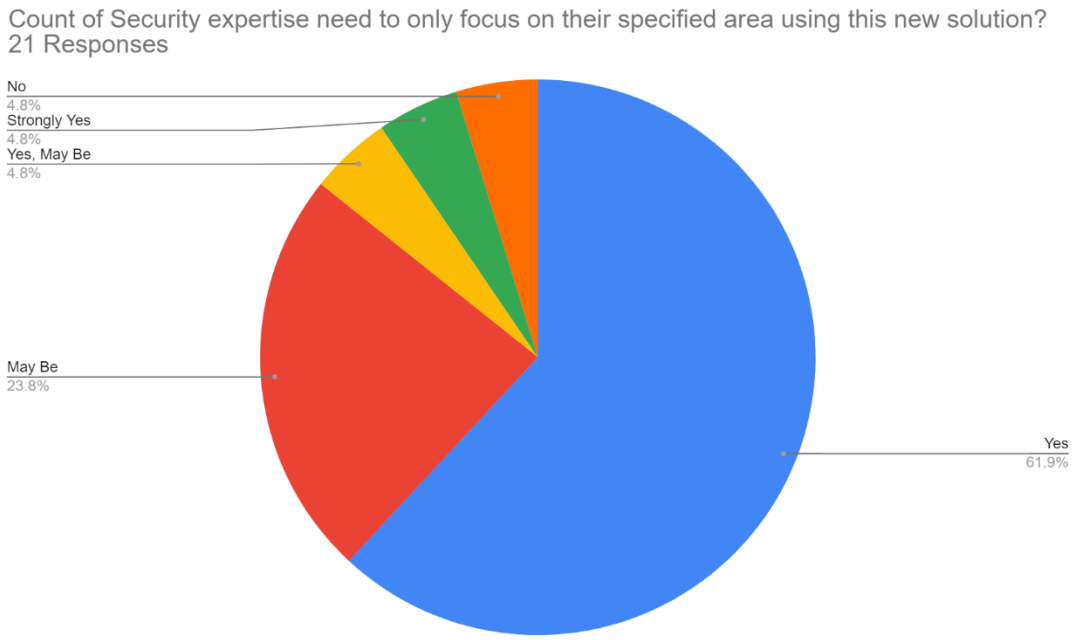


Figure 5.7: Security expertise needs to only focus on specific area

This above Figure 5.7 most users are agreed upon this. So, this represents the security experts can focus on specific areas when defining the system. That will improve the strength of the security of each section the system web, database etc.

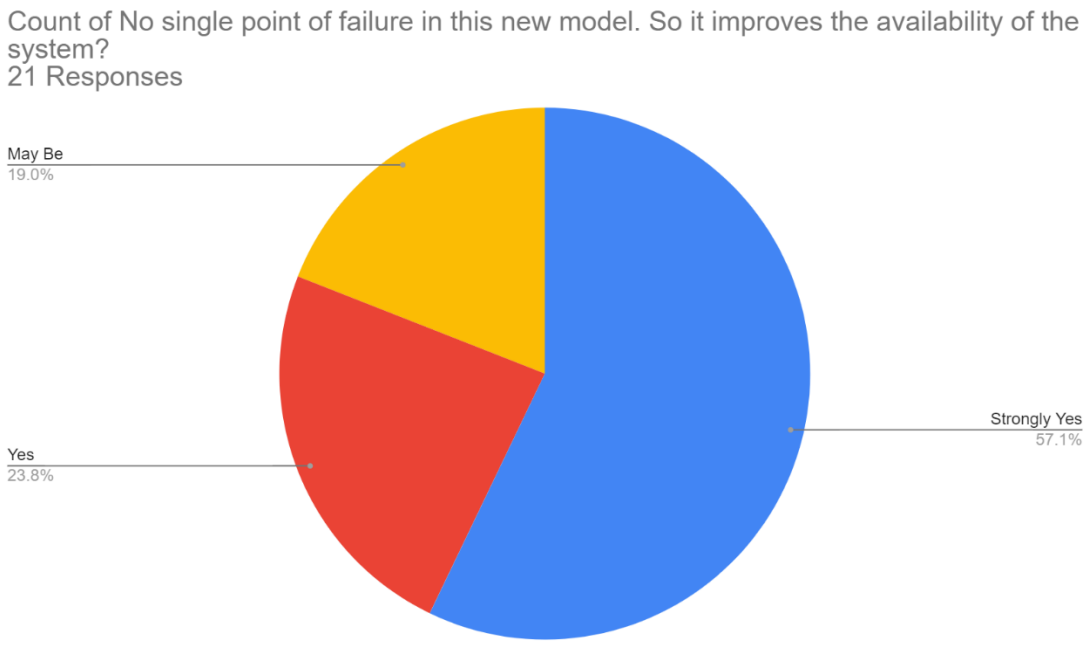


Figure 5.8: No single point of failure in this new model and improved the availability

This above Figure 5.8 represents with new approach no single point of failure in this model and can mostly agree on the availability will be improve with the new model.

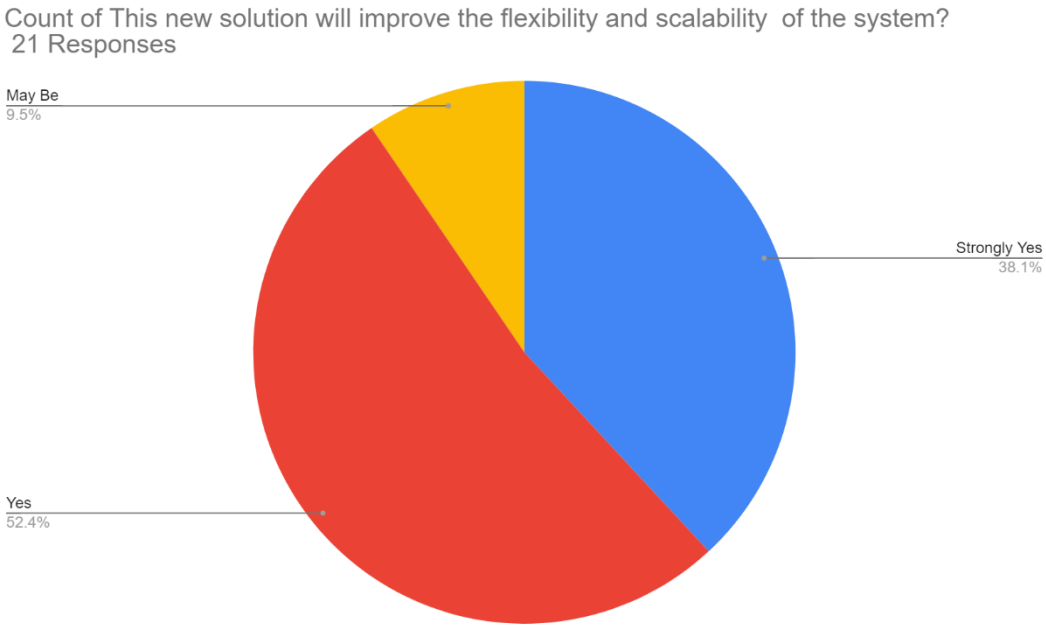


Figure 5.9: Improved the flexibility and scalability of the system

This above Figure 5.9 represents the flexibility and scalability of the system will be improve the new approach

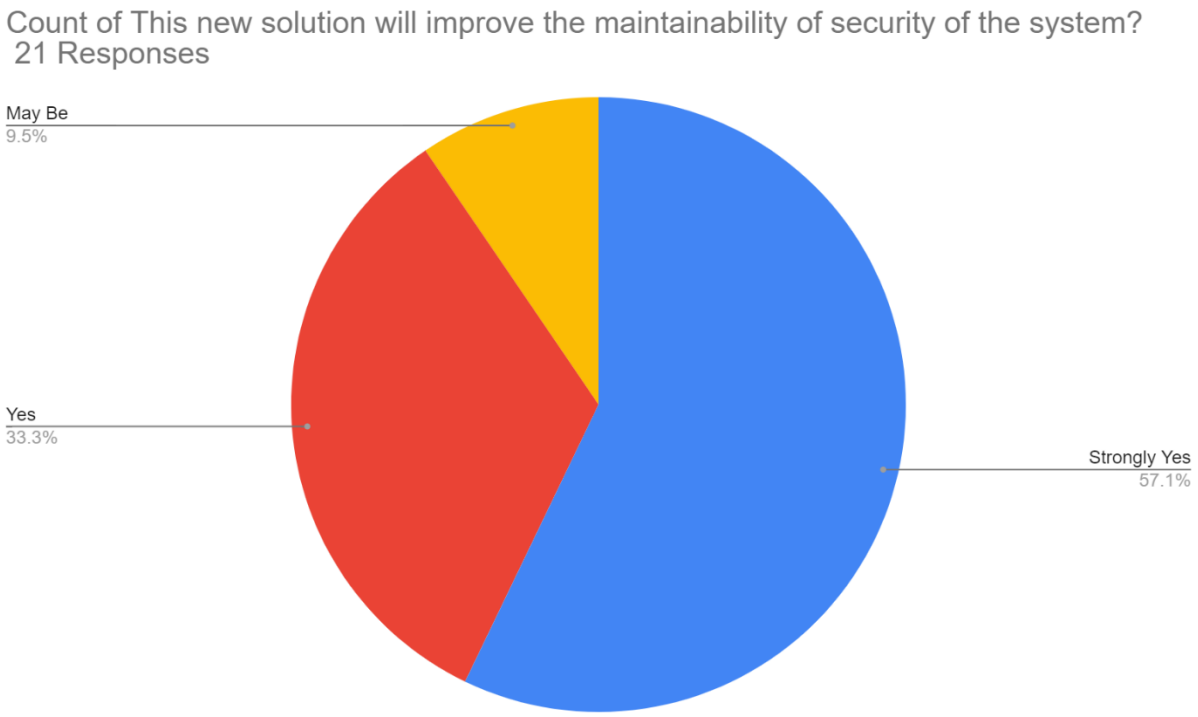


Figure 5.10: Improve the maintainability of the system

This above Figure 5.10 represents the maintainability of the system will be improve the new approach.

From the output of the survey is clear that with the new approach objectives of the research were accomplished.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

Based on the results gathered from the experiment and survey, the CPU usage and the memory usage of the new security model is decreased by a considerable amount when comparing with the security models introduced and used before.

Therefore, the objective of the research, the performance improvement of the IDS deployed in the container orchestration platform is accomplished. As well, able to achieve the below goals by analyzing the outcome of the survey.

1. Security expertise must focus on specific domain areas only
2. Easy to analyze the impact of the specific area (such as database, web application)
3. No Single point of failure in this model. So, it improves the availability of the system
4. Improve the Maintainability
5. Flexibility and scalability of the system

As the outcome of this research, a new model is introduced to achieve the security in microservice world and that will pave a path to introduce new deployment strategies match with the containerization. Now a days, most of the IT applications are moving to the microservice architecture that paves a path to the DevOps concepts with Docker and Kubernetes technologies. In new world of containerization, security aspects of the applications need to move into a different path by using the scalability and model of deployment. So, the compatible deployment pattern and the capability of the containerized world have been used in order to improve the performance of the proposed solution to achieve the security.

6.2 Limitation

This research is only conducted for Docker containers and Kubernetes container orchestration platform.

6.3 Future Work

IDS security model can be deploy in the application level for future improvements. If the IDS move to the application level that will improve the flexibility, scalability and availability of the IDS. Scalability of the IDS can be improved according to CPU, memory, latency metrics of the application. That should be different from one application to another as their application type, such as; database, web, storage, etc. Rules can be defined according to the application and it can be automatically deployed when the application is ready to start.

References

[1] D. Santoro, D. Zozin, D. Pizzolli, F. De Pellegrini, and S. Cretti, "Foggy: A Platform for Workload Orchestration in a Fog Computing Environment," *2017 IEEE Int. Conf. Cloud Comput. Technol. Sci. CloudCom*, pp. 231–234, Dec. 2017.

[2] E. F. Coutinho, F. R. de Carvalho Sousa, P. A. L. Rego, D. G. Gomes, and J. N. de Souza, "Elasticity in cloud computing: a survey," *Ann. Telecommun. - Ann. Télécommunications*, vol. 70, no. 7–8, pp. 289–309, Aug. 2015.

[3] K. A. Scarfone and P. M. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-94, 2007.

[4] M. Shelar, S. Sane, and V. Kharat, "Enhancing Performance of Applications in Cloud using Hybrid Scaling Technique," *Int. J. Comput. Appl.*, vol. 143, no. 2, pp. 43–48, Jun. 2016.

[5] A. Hickman, "Container Intrusions: Assessing the Efficacy of Intrusion Detection and Analysis Methods for Linux Container Environments," p. 32, 2018.

[6] "login_1410_02-bottomley.pdf." .

[7] "Enterprise Container Platform," *Docker*. [Online]. Available: https://www.docker.com/. [Accessed: 25-Nov-2019].

[8] S. Soltesz and M. E. Fiuczynski, "Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors," p. 13.

[9] B. Burns and D. Oppenheimer, "Design patterns for container-based distributed systems," p. 6.

[10] "Production-Grade Container Orchestration - Kubernetes." [Online]. Available: https://kubernetes.io/. [Accessed: 25-Nov-2019].

[11] "Kubernetes Deployment, Enterprise Kubernetes, Containerization." .

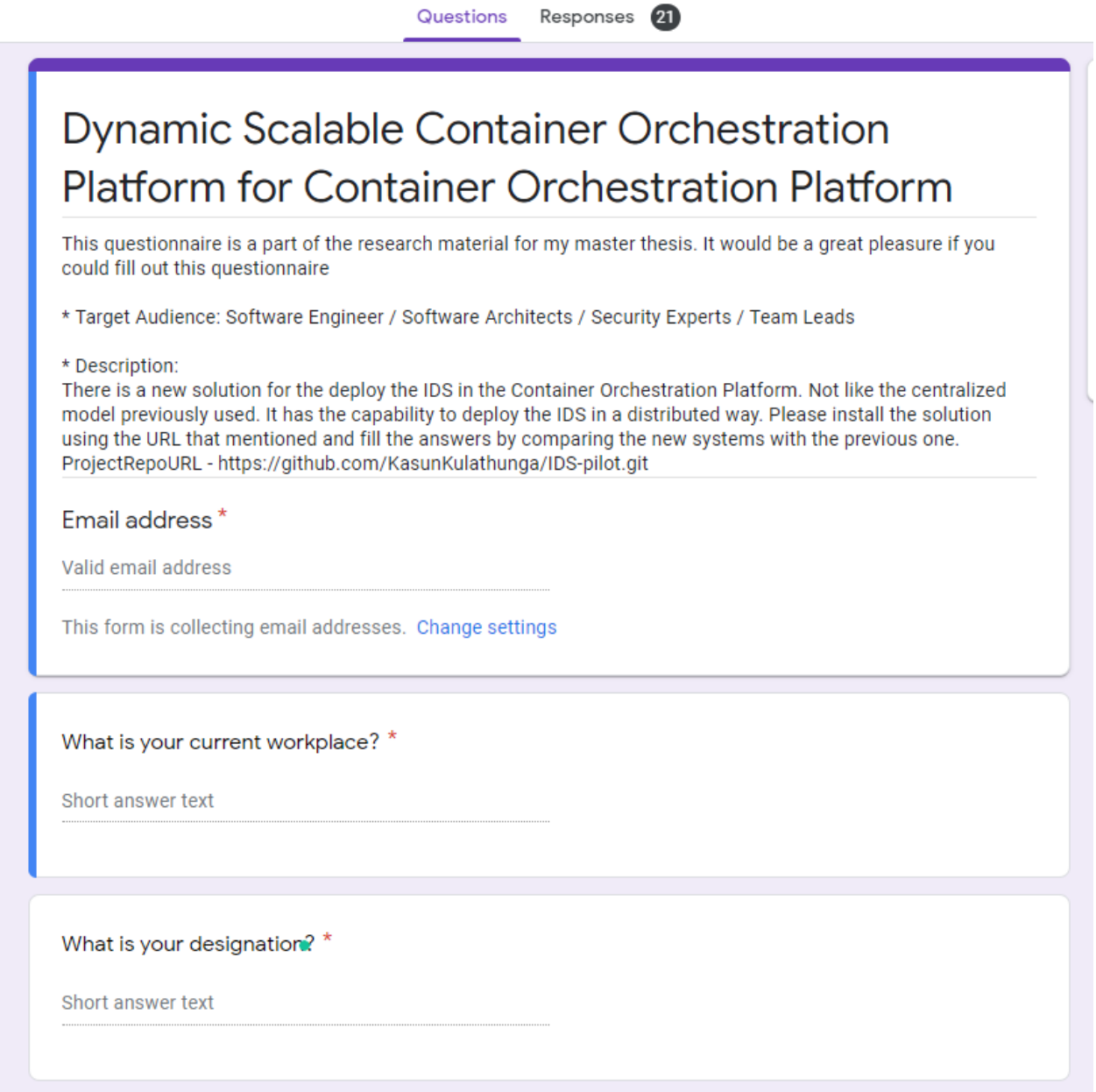
[12] A. S. Abed, T. C. Clancy, and D. S. Levy, "Applying Bag of System Calls for Anomalous Behavior Detection of Applications in Linux Containers," *2015 IEEE Globecom Workshop GC Wkshps*, pp. 1–5, Dec. 2015.

[13] T. Xing, D. Huang, L. Xu, C.-J. Chung, and P. Khatkar, "SnortFlow: A OpenFlow-Based Intrusion Prevention System in Cloud Environment," in *2013 Second GENI Research and Educational Experiment Workshop*, Salt Lake, UT, USA, 2013, pp. 89–92.

[14] T. Xing, Z. Xiong, D. Huang, and D. Medhi, "SDNIPS: Enabling Software-Defined Networking based intrusion prevention system in clouds," in *10th International Conference on Network and Service Management (CNSM) and Workshop*, Rio de Janeiro, Brazil, 2014, pp. 308–311.

Appendices

Survey



This new solution will improve the maintainability of security of the system? *

- [ ] Strongly Yes
- [ ] Yes
- [ ] May Be
- [ ] No

This new solution will improve the flexibility and scalability of the system? *

- ( ) Strongly Yes
- ( ) Yes
- ( ) May Be
- ( ) No

Security expertise need to only focus on their specified area using this new solution? *

- [ ] Strongly Yes
- [ ] Yes
- [ ] May Be
- [ ] No

Easy to analyze the impact of the specific area(web, database)? *

- [ ] Strongly Yes
- [ ] Yes
- [ ] May Be
- [ ] No

No single point of failure in this new model. So it improves the availability of the system? *

- [ ] Strongly Yes
- [ ] Yes
- [ ] May Be
- [ ] No

Timestamp	Email Address	What is your current workplace?	What is your designation?	This new solution will improve the maintainability of security of the system?	This new solution will improve the flexibility and scalability of the system?	Security expertise need to only focus on their specified area using this new solution?	Easy to analyze the impact of the specific area(web, database)?	No single point of failure in this new model.So it improves the availability of the system?
5/15/2020 11:28	chintaka.wijetunga@yahoo.com	IFS	Senior Software Engineer	Strongly Yes	Strongly Yes	Yes	May Be	Strongly Yes
5/15/2020 11:32	heshanmarasinghe@gmail.com	Allion Technologies Pvt LTD	Security Testing Specialist	Strongly Yes	Yes	Yes	Yes	Strongly Yes
5/15/2020 11:36	nimaliw25@gmail.com	Peoples Bank	Network Administrator	Yes	May Be	May Be	Yes	Strongly Yes
5/15/2020 12:05	asiri.kusal@gmail.com	Dialog	Senior software engineer	Strongly Yes	Yes	May Be	Strongly Yes	Strongly Yes
5/15/2020 12:35	ravi.wewliyadda@gmail.com	Epic lanka pvt Ltd	Software engineer	May Be	Strongly Yes	Yes	Strongly Yes	Strongly Yes
5/15/2020 15:31	sameera.madushanka@ifsworld.com	IFS	Senior Systems Engineer	Strongly Yes	Strongly Yes	May Be	Yes	Strongly Yes
5/15/2020 17:58	minudika001@gmail.com	Capgemini Singapore Pte Ltd	Senior Full stack Engineer	Strongly Yes	Strongly Yes	Yes	Strongly Yes	Strongly Yes
5/15/2020 18:47	prageeth.apr@gmail.com	PaperTrl	Systems Engineer	Strongly Yes	Strongly Yes	Yes	Yes	Strongly Yes
5/15/2020 20:02	prageeth2007@gmail.com	Dialog Axiata	Senior QA Engineer	Strongly Yes	Yes	May Be	Strongly Yes	Strongly Yes

5/15/2020 20:02	prageeth2007@gmail.com	Dialog Axiata	Senior QA Engineer	Strongly Yes	Yes	May Be	Strongly Yes	Strongly Yes
5/15/2020 22:47	induisaranie1994@gmail.com	Epic Lanka Technologies	Quality Assurance Engineer	Strongly Yes	Yes	May Be	May Be	Strongly Yes
5/16/2020 17:08	thilinabalasooriya1@gmail.com	WSO2	System engineer	Yes	Yes	Yes	Yes	Yes
5/16/2020 17:09	rthissa91@gmail.com	Inova It System (pvt) ltd	Senior software engineer	Yes	Yes	Yes	Strongly Yes	May Be
5/16/2020 17:11	darshanashan@gmail.com	IT Solutions	Software Engineer	Yes	May Be	Yes	Yes	May Be
5/16/2020 17:32	sasaniandra@gmail.com	hSenid mobile solutions	senior software engineer	Yes	Strongly Yes	Yes	Yes	Yes
5/16/2020 17:48	samushika.chammi@gmail.com	Virtusa	Software Engineer	Yes	Yes	Yes	Yes	May Be
5/16/2020 18:07	Krismilaabey@gmail.com	Epic lanka	Software engineer	Strongly Yes	Yes	Yes	Strongly Yes	Yes
5/16/2020 18:17	theekshana2@gmail.com	keeneye solutions	Solution Architecture	Strongly Yes	Yes	Yes, May Be	Yes	Yes
5/16/2020 18:18	sandaninpu@gmail.com	99X technology	Software engineer	Yes	Yes	Yes	Yes	Yes
5/16/2020 18:29	udawatta.pubudu@gmail.com	Genesis Software	ATL	Strongly Yes	Strongly Yes	Yes	Yes	Strongly Yes
5/16/2020 18:32	pramithaab@gmail.com	Unicorn solutions	Software engineer	Strongly Yes	Strongly Yes	Strongly Yes	Strongly Yes	Strongly Yes
5/16/2020 18:34	harunishara123@gmail.com	InSync Information Technologies	Network Automation Engineer	May Be	Yes	No	May Be	May Be

Figure 4: feedback results

Sample Data Set

Table 2 Previous security model memory evaluation results

Series	Time	Value (Byte)
{pod_name="falco-d2gwk"}	2020-05-16T13:09:30+05:30	80474112
{pod_name="falco-d2gwk"}	2020-05-16T13:09:45+05:30	81690624
{pod_name="falco-d2gwk"}	2020-05-16T13:10:00+05:30	82632704
{pod_name="falco-d2gwk"}	2020-05-16T13:10:15+05:30	83697664
{pod_name="falco-d2gwk"}	2020-05-16T13:10:30+05:30	84516864
{pod_name="falco-d2gwk"}	2020-05-16T13:10:45+05:30	85577728
{pod_name="falco-d2gwk"}	2020-05-16T13:11:00+05:30	86425600
{pod_name="falco-d2gwk"}	2020-05-16T13:11:15+05:30	87359488
{pod_name="falco-d2gwk"}	2020-05-16T13:11:30+05:30	88281088
{pod_name="falco-d2gwk"}	2020-05-16T13:11:45+05:30	89104384
{pod_name="falco-d2gwk"}	2020-05-16T13:12:00+05:30	90390528
{pod_name="falco-d2gwk"}	2020-05-16T13:12:15+05:30	91275264
{pod_name="falco-d2gwk"}	2020-05-16T13:12:30+05:30	92086272
{pod_name="falco-d2gwk"}	2020-05-16T13:12:45+05:30	92897280
{pod_name="falco-d2gwk"}	2020-05-16T13:13:00+05:30	93900800
{pod_name="falco-d2gwk"}	2020-05-16T13:13:15+05:30	94707712
{pod_name="falco-d2gwk"}	2020-05-16T13:13:30+05:30	95838208
{pod_name="falco-d2gwk"}	2020-05-16T13:13:45+05:30	96804864
{pod_name="falco-d2gwk"}	2020-05-16T13:14:00+05:30	97681408

Table 3 Previous security model CPU evaluation results

Series	Time	Value
{pod_name="falco-d2gwk"}	2020-05-16T13:07:15+05:30	0.744305566
{pod_name="falco-d2gwk"}	2020-05-16T13:07:30+05:30	0.785075973
{pod_name="falco-d2gwk"}	2020-05-16T13:07:45+05:30	0.716502128
{pod_name="falco-d2gwk"}	2020-05-16T13:08:00+05:30	1.107246258
{pod_name="falco-d2gwk"}	2020-05-16T13:08:15+05:30	1.98157146
{pod_name="falco-d2gwk"}	2020-05-16T13:08:30+05:30	0.753293558
{pod_name="falco-d2gwk"}	2020-05-16T13:08:45+05:30	0.75488978
{pod_name="falco-d2gwk"}	2020-05-16T13:09:00+05:30	0.750753126
{pod_name="falco-d2gwk"}	2020-05-16T13:09:15+05:30	0.724851516
{pod_name="falco-d2gwk"}	2020-05-16T13:09:30+05:30	0.766547355
{pod_name="falco-d2gwk"}	2020-05-16T13:09:45+05:30	0.743863202
{pod_name="falco-d2gwk"}	2020-05-16T13:10:00+05:30	0.728313307
{pod_name="falco-d2gwk"}	2020-05-16T13:10:15+05:30	0.745508841
{pod_name="falco-d2gwk"}	2020-05-16T13:10:30+05:30	0.829524722
{pod_name="falco-d2gwk"}	2020-05-16T13:10:45+05:30	0.74088622
{pod_name="falco-d2gwk"}	2020-05-16T13:11:00+05:30	0.764260186
{pod_name="falco-d2gwk"}	2020-05-16T13:11:15+05:30	0.699229879
{pod_name="falco-d2gwk"}	2020-05-16T13:11:30+05:30	0.805247479
{pod_name="falco-d2gwk"}	2020-05-16T13:11:45+05:30	0.67386475
{pod_name="falco-d2gwk"}	2020-05-16T13:12:00+05:30	0.757419552
{pod_name="falco-d2gwk"}	2020-05-16T13:12:15+05:30	0.746782925

Table 4 New security model memory evaluation results

Series	Time	Value (Byte)
{pod_name="falco-d2gwk"}	2020-05-16T13:09:30+05:30	80474112
{pod_name="falco-d2gwk"}	2020-05-16T13:09:45+05:30	81690624
{pod_name="falco-d2gwk"}	2020-05-16T13:10:00+05:30	82632704
{pod_name="falco-d2gwk"}	2020-05-16T13:10:15+05:30	83697664
{pod_name="falco-d2gwk"}	2020-05-16T13:10:30+05:30	84516864
{pod_name="falco-d2gwk"}	2020-05-16T13:10:45+05:30	85577728
{pod_name="falco-d2gwk"}	2020-05-16T13:11:00+05:30	86425600
{pod_name="falco-d2gwk"}	2020-05-16T13:11:15+05:30	87359488
{pod_name="falco-d2gwk"}	2020-05-16T13:11:30+05:30	88281088
{pod_name="falco-d2gwk"}	2020-05-16T13:11:45+05:30	89104384
{pod_name="falco-d2gwk"}	2020-05-16T13:12:00+05:30	90390528
{pod_name="falco-d2gwk"}	2020-05-16T13:12:15+05:30	91275264
{pod_name="falco-d2gwk"}	2020-05-16T13:12:30+05:30	92086272
{pod_name="falco-d2gwk"}	2020-05-16T13:12:45+05:30	92897280
{pod_name="falco-d2gwk"}	2020-05-16T13:13:00+05:30	93900800
{pod_name="falco-d2gwk"}	2020-05-16T13:13:15+05:30	94707712
{pod_name="falco-d2gwk"}	2020-05-16T13:13:30+05:30	95838208
{pod_name="falco-d2gwk"}	2020-05-16T13:13:45+05:30	96804864
{pod_name="falco-d2gwk"}	2020-05-16T13:14:00+05:30	97681408
{pod_name="falco-d2gwk"}	2020-05-16T13:14:15+05:30	98926592
{pod_name="falco-d2gwk"}	2020-05-16T13:14:30+05:30	99344384
{pod_name="falco-d2gwk"}	2020-05-16T13:14:45+05:30	100597760

Table 5 New security model CPU evaluation results

Series	Time	Value (Cores)
{pod_name="falco-d2gwk"}	2020-05-16T13:07:15+05:30	0.744305566
{pod_name="falco-d2gwk"}	2020-05-16T13:07:30+05:30	0.785075973
{pod_name="falco-d2gwk"}	2020-05-16T13:07:45+05:30	0.716502128
{pod_name="falco-d2gwk"}	2020-05-16T13:08:00+05:30	1.107246258
{pod_name="falco-d2gwk"}	2020-05-16T13:08:15+05:30	1.98157146
{pod_name="falco-d2gwk"}	2020-05-16T13:08:30+05:30	0.753293558
{pod_name="falco-d2gwk"}	2020-05-16T13:08:45+05:30	0.75488978
{pod_name="falco-d2gwk"}	2020-05-16T13:09:00+05:30	0.750753126
{pod_name="falco-d2gwk"}	2020-05-16T13:09:15+05:30	0.724851516
{pod_name="falco-d2gwk"}	2020-05-16T13:09:30+05:30	0.766547355
{pod_name="falco-d2gwk"}	2020-05-16T13:09:45+05:30	0.743863202
{pod_name="falco-d2gwk"}	2020-05-16T13:10:00+05:30	0.728313307
{pod_name="falco-d2gwk"}	2020-05-16T13:10:15+05:30	0.745508841
{pod_name="falco-d2gwk"}	2020-05-16T13:10:30+05:30	0.829524722
{pod_name="falco-d2gwk"}	2020-05-16T13:10:45+05:30	0.74088622
{pod_name="falco-d2gwk"}	2020-05-16T13:11:00+05:30	0.764260186
{pod_name="falco-d2gwk"}	2020-05-16T13:11:15+05:30	0.699229879
{pod_name="falco-d2gwk"}	2020-05-16T13:11:30+05:30	0.805247479
{pod_name="falco-d2gwk"}	2020-05-16T13:11:45+05:30	0.67386475
{pod_name="falco-d2gwk"}	2020-05-16T13:12:00+05:30	0.757419552
{pod_name="falco-d2gwk"}	2020-05-16T13:12:15+05:30	0.746782925
{pod_name="falco-d2gwk"}	2020-05-16T13:12:30+05:30	0.784317139