Continuous Integration and Continuous Delivery Framework for SDS

Yahuza Bello*†, Emanuel Figetakis *†, Ahmed Refaey*†, and Petros Spachos *
*University of Guelph, Ontario, Canada
†Manhattan College, New York, USA

2022 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) | 978-1-6654-8432-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/CCECE49351.2022.9918437

***Abstract*—Fast and efficient development of software drives the high demand for automation techniques, especially for cloud-based systems trying to implement Software Defined Systems (SDS). The emergence of Continuous Integration/Continuous Delivery (CI/CD) provides a set of steps for building, testing, and deployment of new software in an automated fashion. Consequently, many companies integrate CI/CD pipelines into their platform to automate the development and deployment of new software and applications. Software-Defined Perimeter (SDP) is a new approach to cyber security proposed by the Cloud Security Alliance (CSA) to dynamically secure network services. This is reached utilizing the need-to-know concept where authorization is only granted after strict user verification. SDP framework integrates with cloud-based systems seamlessly. However, the installation, configurations, and management of its components are still manual. This will require a lot of time and resources as the number of protected services increases. Therefore, this paper presents the implementation of the Continuous Integration/Continuous Delivery (CI/CD) pipeline for the open SDP project that automates the installation and deployment of its various components. Specifically, the Open SDP components (i.e., SDP controller and gateway) will be used as a use case to show the use of CI/CD and to secure applications hosted on the OpenShift environment. The OpenShift pipeline operator, based on the Tekton project was adopted as the CI/CD pipeline for this project. The Code Ready Container (CRC) was utilized as the OpenShift cluster, which is then hosted on a server running a Windows OS. Furthermore, the challenges, as well as their solutions to the Open SDP CI/CD pipeline, are presented.**

I. Introduction

The past few decades have seen a rapid increase in cloud-based systems that have a multitude of applications. These applications range from consumer base services such as storage, web hosting, email, etc. However, cloud-based systems are also finding considerable use within the commercial sector that provide these services and have their own applications. Applications can range from data collection, cloud-based virtual instances, and deployments of web-based applications. Nonetheless, cloud-based systems arrive with their own challenges for even the largest of corporations trying to implement them as well as other issues with latency, security, and quality of life.

The introduction of Software Defined Systems (SDS) has helped alleviate some of these challenges by adding software components. An example of how the SDS is assisting cloud-based systems is through the separation of different software layers through a hypervisor. This means different software can run independently of one another without compromising one another, which in turn maximizes the use of the hardware. However, even the SDS can bring some of its own challenges and the main one that is shared with traditional cloud-based systems is the implementation. The total setup of a system can be lengthy and can take up to several days, which for a large corporation means a loss of capital due to more personnel having to be dedicated to a single implementation. This can slow down the services and if a problem presents itself, it can disrupt the workflow.

Implementation of continuous integration and continuous deployment (CI/CD) pipeline could be extremely beneficial. Not only does the CI/CD pipeline help with implementation but also helps with maintenance, testing, and deployment, and allows for several people to work on the same project seamlessly [1]. The commercial use of the CI/CD pipeline has a major impact, it allows for the constant update of software and management for applications and products. Most cloud-based systems that have adopted the SDS can be managed through a CI/CD pipeline as well as be implemented faster than the traditional setup. Most of the challenges have been met from both cloud-based systems and the SDS except one, security. The threat of security is a large one especially since most of the services must be able to be accessed by clients and therefore are visible on the public internet [2]. This creates a large point of failure since anyone with the address can talk to the service hosted on the hardware. Software vulnerabilities can be found if there is an open port and with a few queries the exact hardware can be discovered. Not only does the threat to the service and data exist but also a threat to the physical hardware.

Due to the high increment in security vulnerabilities in software-defined paradigms, the traditional security solutions fail to provide the required network defense perimeters. The severity of this point of failure is great and is a major concern to any company that deals with customer data both financial and personal. Even further a company that has private software that must remain within the company, for it to be used by the employees can put it on the cloud to be accessed, but if it exists on the network then lies the chance that it can be found and stolen or replicated. However by utilizing the cloud-based applications and SDS, a security method can easily be implemented by using CI/CD pipeline, and one such security method is the Software Defined Perimeter (SDP). SDP is a zero-trust software-based security system designed to provide devices/applications with a logical defense perimeter against cyber-attacks [3][4]. This is achieved through verification and authentication of any device/application looking to gain access to the protected network services. The SDP can mitigate several different kinds of cyber-attacks but it is most advantageous to use in a cloud-based setting for the fact that it can easily mitigate large-scale Distributed Denial of Service attacks without any disruption to the network. For many companies, this is the worst kind of attack because it can stop all services that are being provided to a client. This downtime can result in a loss of millions of dollars, especially in the financial sector. Since

the nature of the SDP is zero-trust, this means it will mitigate a DDOS attack without having to receive any special instruction. By implementing this system companies can have a secure network within keeps customer data and their own software private. Another benefit of using the SDP is that it prevents an attack from the internal network. The SDP covers several points of failure from SDS while still using a cloud-based system. Evidently, SDP is a proven software-based security solution for cloud-based systems [5]–[9]. However, the installation and deployment of the SDP framework is not an easy task especially when more SDP clients and SDP gateways are required as the network expands. Therefore, in this paper, we present the implementation of the Continuous Integration/Continuous Delivery (CI/CD) pipeline for SDP on the Red Hat OpenShift Platform as a use case of SDS.

The rest of the paper is organized as follows: the next section introduces the related works, which is then followed by section III which covers the SDP as a use case for the SDS. Section IV-A covers the CI/CD principle and its automation, followed by Section IV-B which covers the Tekton CI/CD Pipeline. Section V covers the implementation of CI/CD with SDP using Openshift, followed by Section VI which covers the challenges, the proposed solution and results. Section VII concludes the paper.

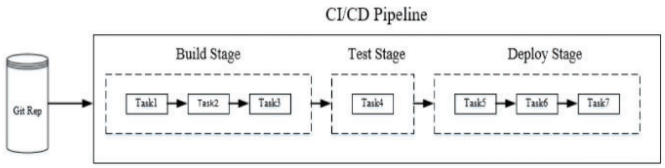

Fig. 1: CI/CD Pipeline

II. Related Work

Within the industry, there are many systems that are being created as Software Defined Systems. They are being used to get more use out of cloud-based systems and create a better application with better features. A recurring theme within the industry is using software-defined systems for networking. Introducing SDS into the network helps enforce networking rules, restrictions, and for mitigation. In [10] the authors are using a Software Defined Network (SDN) to help manage networks to create a faster incident response time in an industrial setting. The SDN allows for better control over a network with direct communication for sensors. This is one of the benefits of using an SDN, it allows for more direct communication with the hardware on the network, this can be especially useful when developing smart cities, buildings, housing, etc.

The resilience of SDP against several cyber attacks was demonstrated in several research works in the literature [5]–[9]. Some of the security vulnerabilities introduced by the abstraction of control and data layer in SDN were addressed using SDP in [5]. The authors proposed combining SDP and SDN controllers to secure the entire network against such vulnerabilities. SDP was also integrated into the Network Function Virtualization (NFV) to provide a zero-trust environment within the Network Function Virtualization Infrastructure (NFVI) [6]. The authors demonstrated how the SDP framework can be adopted to protect network services from external attacks. In a previous work [7], we proposed SDP as a security framework within Multi-access Edge Computing (MEC) by placing the SDP components at the edge of the network in order to block attacks such as DoS and port scanning attacks. Additionally, we showcase the suitability of the SDP framework (i.e., as a potential security framework for MEC-based networks) by performing an End-to-End delay analysis of the proposed SDP-MEC architecture. Most recently, we propose a vEPC-vSDP architecture that integrates virtualized versions of the SDP components with virtualized versions of Evolved Packet Core (EPC) entities to provide a zero-trust environment within the vEPC [8]. The proposed combined architecture was capable of protecting the vEPC entities from both internal and external attacks. The authors in [9] demonstrate how the SDP framework fits into today's cloud Infrastructure as a Service (IaaS) to serve as a security measure against Denial of Service (DoS) attack. The authors leverage the Amazon Web Service (AWS) platform to implement their proposal and showcase its ability to resist cloud-based attacks. In [11], the authors demonstrated how the Single Packet Authorization (SPA) method of the SDP framework is able to replace the login authentication method of the Message Queuing Telemetry Transport (MQTT).

III. Use Case of SDP as SDS

The SDP architecture consists of an SDP controller module, an SDP client module (also referred to as Initiating Host (IH)), and an SDP Accepting Host (AH) module. The SDP controller module is the brain responsible for verification and authentication of authorized hosts (i.e., initiating and accepting hosts) as well as setting up a list of authorized services they can access. The AH normally the SDP gateway enforces rules set by the SDP controller to block all devices/applications' access to the network services except the ones verified and authorized by the controller. This setup guarantees that only the authorized SDP IHs with a valid certificate have access to the protected network services.

The concept of CI/CD pipeline is well-established thanks to the rapid development of Development and Operations (DevOps) [12]. A typical CI/CD pipeline aims to assist DevOps teams to develop codes, run tests and deliver/deploy the latest version of applications in either the staging or production environment reliably without the need for expert interventions [13]. Some of the most widely used CI/CD pipeline software include Jenkins, TeamCity, Tekton, AWS CodePipeline, GitLab, and GitHub Actions [14].

Red Hat OpenShift is an open source container Platform as a Service (PaaS) that hands developers the capability to develop and deploy docker containerized applications and orchestrate them using Kubernetes orchestrator [15]. It has a built-in CI/CD pipeline based on the Tekton CI/CD project, which developers can leverage code-to-container technology easily [16]. For the purpose of this project, we opt to go with the built-in CI/CD pipeline operator of OpenShift and also adopt the OpenShift Cluster to host an application protected by SDP.

IV. CI/CD Pipeline Principle

A. CI/CD Pipeline

A typical CI/CD pipeline consists of a series of automated steps, which when followed correctly ensures a reliable delivery/deployment of new versions of applications [17]. A typical CI/CD pipeline introduces automation and monitoring of application

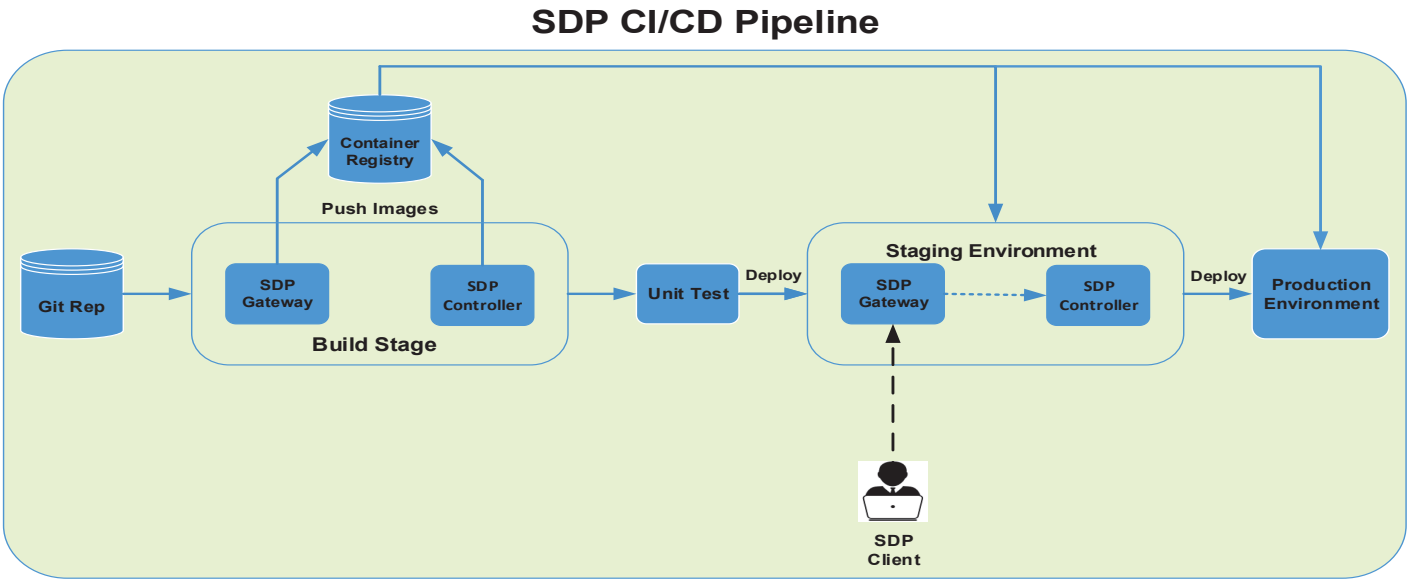

Fig. 2: Open SDP CI/CD pipeline

development, particularly the critical testing phase in the staging environment as well as application deployment. Consequently, developers eliminate the need to manually test and merging of codes with the CI/CD pipeline. Conventional CI/CD systems were designed to work with pipelines running on VMs. Recently, cloud-native CI/CD pipelines move to deploy containerized applications on platforms that provide PaaS hosting such as AWS, AZURE, and OpenShift. Some of the most widely adopted CI/CD pipeline tools are OpenShift pipeline, Jenkins, AWS CodePipeline, GitHub Actions, etc.

Figure 1 shows the elements of a typical CI/CD pipeline, which includes the following:

- Build Stage: in this stage, the pipeline fetches the source code from the desired repository and builds a container image.
- Test Stage: this stage allows for a series of tests to be conducted automatically
- Deployment Stage: in this stage, the verified application is deployed either to the staging environment for further testing or to production environments to end users.

OpenShift provides a built-in OpenShift CI/CD pipeline, which is based on the Tekton project. Leveraging the OpenShift capabilities for hosting containerized applications, the OpenShift CI/CD pipeline can be designed to fetch codes, build a container image and deploy the application on OpensShift Cluster.

B. Tekton CI/CD Pipeline

The Tekton CI/CD pipeline is a cloud-native CI/CD framework designed for Kubernetes platforms for containerized applications [18]. Leveraging the Kubernetes framework capabilities, Tekton makes it possible to deploy containerized applications across multiple cloud environments. Utilizing the Custom Resource Definitions (CRDs) of the Kubernetes framework, a Tetkon-based CI/CD pipeline can be created and run through the Kubernetes control plane.

A typical Tekton CI/CD pipeline consists of the following elements:

- Task: a reusable, loosely coupled number of steps that perform a specific task (e.g. building a container image).
- PipelineResources: are the set of objects that are going to be used as inputs to a Task and can be the output of a Task.
- Pipeline: the definition of the pipeline and the Tasks that it should perform.
- TaskRun: the execution and result of running an instance of a task.
- PipelineRun: the execution and result of running an instance of the pipeline, which includes a number of TaskRuns.

V. Open SDP CI/CD Pipeline on Openshift Cluster

OpenShift provides a developer version of its platform in the form of Code Ready Container (CRC) that can be installed locally on several OS such as Windows, Linux-based OS (only CentOS and Ubuntu are supported), and MaC OS. For the Open SDP CI/CD pipeline implementation, we opt to install the latest CRC version of the OpenShift cluster on a server with CentOS OS.

Figure 2 shows the Open SDP CI/CD pipeline implementation, which consists of the following elements:

- Build stage: in this stage, the containerized SDP controller and SDP gateway are created, which are then stored in the built-in container registry provided by the OpenShift platform. Note that the docker build strategy provided by OpenShift is used, which requires a dockerfile containing the source code required to create the containerized SDP components (i.e., SDP controller and gateway)
- Unit test stage: in this stage, a series of tests are conducted to ensure that both SDP components are running as required.
- Deployment to staging environment: in this stage, the containerized SDP controller and gateway are deployed in the staging environment where all the necessary configurations are made and tested with an SDP client.
- Deployment to production environment: in this stage, the containerized SDP components (i.e., SDP controller and SDP gateway) are deployed in the production environment to be used by clients.

As explained earlier, the OpenShift CI/CD pipeline is based on the Tekton pipeline and thus, requires CRDs files created for the pipeline, all the tasks (i.e., SDP gateway, SDP controller, unit test and deployment for the SDP gateway, and SDP controller) included in the pipeline, pipelineResources (i.e., input resource for SDP gateway and input resource for SDP controller) and pipelineRun.

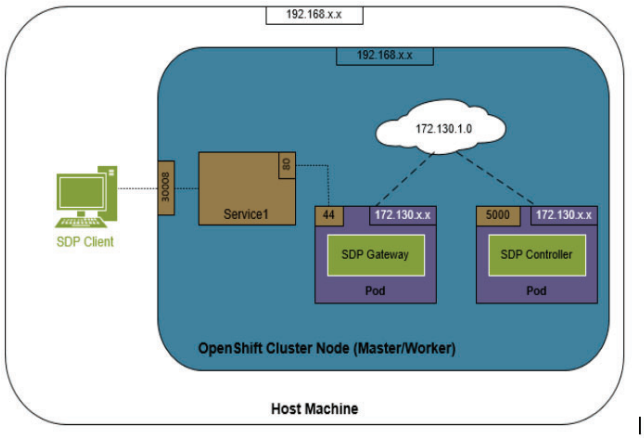

Fig. 3: Deployment of Open SDP CI/CD pipeline on OpenShift Cluster

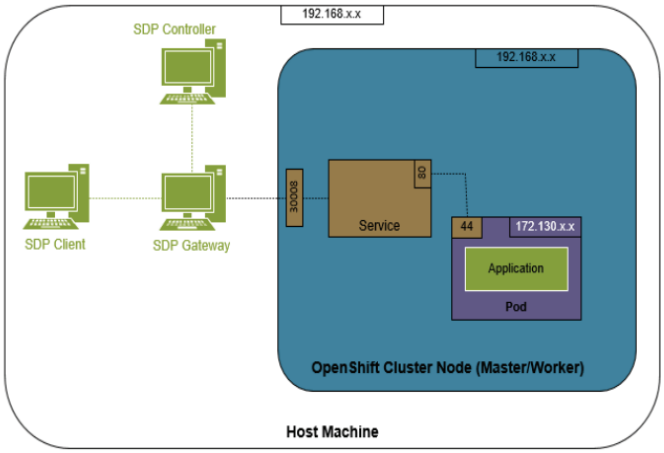

Fig. 4: Proposed Solution for Open SDP CI/CD Pipeline

The detail of the Open SDP CI/CD pipeline within the OpenShift cluster, which is a single node cluster serving as both master and worker node is shown in figure 3. Upon successful deployment of the containerized SDP components, a service (a Kubernetes object for connectivity where a port on the host server is mapped to a port on the pod hosting the SDP gateway) is created to connect the SDP client to the SDP gateway. Note that under this setup, the SDP controller is considered a protected application. Within the OpenShift cluster, an internal network is automatically created for the pods hosting the containerized SDP components as depicted in figure 3. In this way, the SDP controller and gateway can communicate as intended without further modifications. The SDP client is hosted on VM within the server hosting the OpenShift cluster and connects to the SDP gateway via the service created as explained earlier.

VI. Challenges, Solution and Results

A. Challenges

There are several challenges that arise in the Open SDP CI/CD implementation. These challenges are as follows:

- Containerizing SDP gateway: by its default setting, the SDP gateway requires access to the kernel of the host OS to function correctly when setting up the iptables' firewall policies, which makes containerizing it unsuitable as containerized applications have restrictions on kernel access by default. This raises a big issue for the correct functioning of the SDP gateway when containerized.
- SDP controller database for storing credential keys: by default, the SDP controller stores all credential keys for authentication and authorization of SDP components in a MySQL database. These credential keys are automatically generated and stored in the database when the containerized SDP controller is built through the docker-build strategy offered by OpenShift when the Open SDP CI/CD pipeline is executed. This means that any change (i.e., either by adding/removing SDP clients or SDP gateways) made that will trigger the re-execution of the pipeline will result in the generation of new credential keys for all the SDP components involved. This will prompt the re-distribution of those keys to the SDP components involved. To overcome this problem, a persistence MySQL database offered by OpenShift can be deployed separately in the CI/CD pipeline and attached to the SDP controller.
- Root privileges: the default settings on the OpenShift cluster restricts root access for any container launched in any pod. It is usually considered a bad practice to allow root access in both staging and production environments since any person that has access to the cluster can be able to make significant changes that are not allowed. Meanwhile, the configurations of the SDP components strictly require root privileges. This raises another issue that has to be addressed to fully configure the SDP components as desired.

B. Proposed Solution

To address the challenges mentioned in the previous section, we move the SDP controller and gateway to VMs and deploy a simple containerized SSH server on the OpenShift cluster to be the protected application as depicted in figure 4. The CRC for the OpenShift cluster is installed on a server with Windows OS and the SDP Controller, gateway, and client modules are installed on VMs within the same host to ease connectivity. Internal loopback adapters are used so the entire system is able to be connected. With this setup, the SDP components can be configured as desired bypassing the challenges faced when containerized. Also with this configuration, only one machine is needed, and it still provides the same security since both the gateway and controller are separate entities on the same machine. All clients that request access to the application deployed on the OpenShift cluster must be authenticated and authorized by the SDP controller.

C. Results

Figure 5 compares the time of installation for any application across three different methods of installation. The figure scales

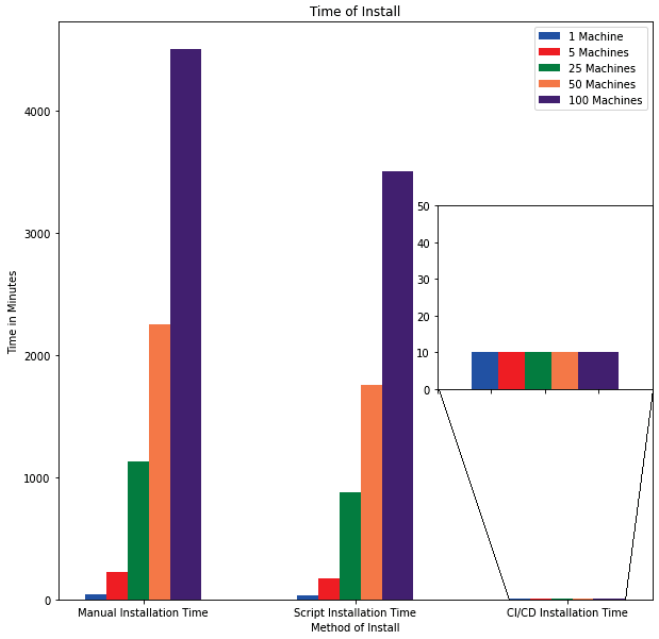

Fig. 5: Open SDP CI/CD pipeline

with the number of machines that need to have the application installed and assuming once one installation is finished the administrator would move to the next. This is neglecting the fact that several installations could happen on different machines at the same time but shows the maximum amount of time needed for all of the machines. The three methods are manual installation, script installation, and CI/CD installation. Manual installation is the longest of the three, script assists little with the installation whereas the installation on a pipeline takes little to no time. We exclude loading times but the time needed for an administrator to be attentive to the machine. This also shows the time needed for updates as well, and the reason the CI/CD Pipeline is a better method is that all installation is taken care of from the pipeline as well as updates. As soon as an update is pushed through the workflow it is downloaded onto the machines, and this all can take place simultaneously across several different machines at once.

VII. Conclusion

This paper presents an implementation of CI/CD pipeline framework for SDS. As a use case, we implemented the CI/CD pipeline for the open SDP project on the Red Hat OpenShift Platform. We adopted the Tekton-based built-in pipeline offered by OpenShift and design the CI/CD pipeline stages for the SDP project. Various challenges encountered such as container-based SDP gateway problem, persistent database problem, and root privileges problem were discussed and a solution was proposed to overcome them. The results obtained demonstrate how the CI/CD pipeline reduces the installation time compared to the manual and script installations.

The system that was developed could be implemented in large corporations to help protect data and provide a more reliable service, at a very small cost. Also, since it is able to be implemented as an SDS on a CI/CD pipeline it can be installed and managed easily. It can also prevent the recurring threat of large-scale DDOS attacks that began to appear in early 2022. We have seen examples where companies like Amazon and Github have both been attacked by different methods of DDOS attacks with a traffic volume of 2.3Tbps. They were able to successfully mitigate the attack but not prevent it. In many cases, the current methods of mitigation are being used to attack the defenders. The Connectionless Lightweight Directory Access Protocol (CLDAP) Reflection method to increase traffic, a DDOS mitigation technique, has been used to attack these large corporations. This is how SDP differs from current methods, it will not allow unauthorized packets to be sent to the network while running checks on the current network nodes. This achieves both mitigation and defense within the network if an attack would come from within. Both of these recent DDOS attacks could have been prevented if SDP was in place.

References

[1] S. Arachchi and I. Perera, "Continuous integration and continuous delivery pipeline automation for agile software project management," in *2018 Moratuwa Engineering Research Conference (MERCon)*, 2018, pp. 156–161.
[2] P. Mirdita, Z. Khaliq, A. R. Hussein, and X. Wang, "Localization for intelligent systems using unsupervised learning and prediction approaches," *IEEE Canadian Journal of Electrical and Computer Engineering*, vol. 44, no. 4, pp. 443–455, 2021.
[3] A. Refaey, S. Roy, and P. Fortier, "On the application of bp decoding to convolutional and turbo codes," in *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*, 2009, pp. 996–1001.
[4] "Sdp: The most advanced zero trust architecture," Software Defined Perimeter Working Group, May 2020. [Online]. Available: https://cloudsecurityalliance.org/artifacts/sdp-the-most-advanced-zero-trust-architecture/
[5] A. Sallam, A. Refaey, and A. Shami, "On the security of sdn: A completed secure and scalable framework using the software-defined perimeter," *IEEE Access*, vol. 7, pp. 146 577–146 587, 2019.
[6] J. Singh, A. Refaey, and A. Shami, "Multilevel security framework for nfv based on software defined perimeter (sdp)," *IEEE Network*, pp. 1–6, March 2020.
[7] J. Singh, Y. Bello, A. Refaey, A. Erbad, and A. Mohamed, "Hierarchical security paradigm for iot multi-access edge computing," *IEEE Internet of Things Journal*, pp. 1–1, 2020.
[8] Y. Bello, A. R. Hussein, M. Ulema, and J. Koilpillai, "On sustained zero trust conceptualization security for mobile core networks in 5g and beyond," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1876–1889, 2022.
[9] J. Singh, A. Refaey, and J. Koilpillai, "Adoption of the software-defined perimeter (sdp) architecture for infrastructure as a service," *Canadian Journal of Electrical and Computer Engineering*, vol. 43, no. 4, pp. 357–363, 2020.
[10] A. F. Murillo Piedrahita, V. Gaur, J. Giraldo, A. Cárdenas, and S. J. Rueda, "Leveraging software-defined networking for incident response in industrial control systems," *IEEE Software*, vol. 35, no. 1, pp. 44–50, 2018.
[11] A. Refaey, A. Sallam, and A. Shami, "On iot applications: a proposed sdp framework for mqtt," *Electronics Letters*, 09 2019.
[12] A. Alnafessah, A. U. Gias, R. Wang, L. Zhu, G. Casale, and A. Filieri, "Quality-aware devops research: Where do we stand?" *IEEE Access*, vol. 9, pp. 44 476–44 489, 2021.
[13] F. Zampetti, S. Geremia, G. Bavota, and M. Di Penta, "Ci/cd pipelines evolution and restructuring: A qualitative and quantitative study," in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2021, pp. 471–482.
[14] J. Shah, D. Dubaria, and J. Widhalm, "A survey of devops tools for networking," in *2018 9th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, 2018, pp. 185–188.
[15] *Red Hat Openshift*. [Online]. Available: https://www.redhat.com/en/technologies/cloud-computing/openshift/container-platform
[16] *Understanding OpenShift Pipelines*. [Online]. Available: https://docs.openshift.com/container-platform/4.7/cicd/pipelines/understanding-openshift-pipelines.html
[17] *What is a CI/CD pipeline?* [Online]. Available: https://www.redhat.com/en/topics/devops/what-cicd-pipeline
[18] *Welcome to Tekton*. [Online]. Available: https://tekton.dev/docs/concepts/overview/