ISSN 2347 - 3983

Volume 8. No. 3, March 2020

International Journal of Emerging Trends in Engineering Research

Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter44832020.pdf

https://doi.org/10.30534/ijeter/2020/44832020

# Securing web application by using qualitative research methods for detection of vulnerabilities in any application of DevSecOps

**B.S.Vishnu Vardhan Reddy[1],Burla Kumara Swamy[2], Savaram Pavan Siva Sai[3], Dr.K.V.D.Kiran[4]**

Department of Computer Science & Engineering, IV/IV B.Tech CSE Students[1,2,3], Professor[4]

K L E F Deemed to be University, Vaddeswaram, Guntur Dt, Andhra Pradesh, Indiavishnu.boreddy111@gmail.com, 160030196@kluniversity.in, 160031250@kluniversity.in,kiran_cse@kluniversity.in

## ABSTRACT

These days an ever increasing number of organizations execute the DevSecOps approach. It was created to empower increasingly proficient joint effort between development (dev) and operations (ops) and security(Sec) teams. An attacker can access individual information if there are vulnerabilities in applications. Pipelines can comprise out of different online applications Continuous Delivery (CD) tools. The initial segment of this area considers papers which have managed making sure about web applications. The subsequent part is about papers which have occupied with making sure about pipelines. One model is that the system can be examined by running infused pernicious unit tests. This can negatively affect the picture of the organization which works and uses CD pipelines. In this manner, the inquiry emerges which vulnerabilities are available in CD pipelines and how they can be detected. One focal point of the paper is to discover which devices are accessible to recognize vulnerabilities in CD pipelines. The hypothetical discoveries of this exploration are stretched out by a viable case study. Papers containing security techniques and security tools that broaden the DevSecOps approach complete this methodology

**Key words:** Continuous Delivery ,Development, Operation, Pipelines, Security, vulnerabilities, web based applications.

## 1. INTRODUCTION

According to the agile manifesto rule, it is "The highest priority is to fulfill the client through right on time and continuous delivery of valuable software. By applying the continuous delivery (CD) approach organizations can send application changes and highlights to the client quickly and dependably.

In Cater's interview Francois Raynaud mentioned that the use of security tools during the deployment process is necessary to add security to the software.

Rahman and Williams discovered that the automation of activities, for example, monitoring, testing, and code review add security practices to the DevOps procedure and can positively affect the security of the framework. A further consequence of their exploration is that the decision of deployment tools and software measurements affect the security of the framework. Furthermore, Rahman and Williams exhibited that in eight assessed organizations numerous DevOps security activities are acted in a non-automated way. Security prerequisites investigation, performing security arrangements or information approval are three activities of that rundown.

Jim Bird described in his book various prospects of how to add security practices to DevOps tools and to the development procedure. He referenced that security tests and practices can be added to each phase of the pipeline. He suggested that before the source code is checked in threat modeling or peer code review ought to be performed. SAST tools ought to be executed in the commit stage. In the acceptance stage tools, for example, Puppet, Chef or Docker ought to be utilized to automate the configuration management. This prompts greater security all the while. In this stage, tests ought to be performed, for example, fluffing or DAST tests. Moreover, automated security attacks (penetration testing) can be performed to detect further vulnerabilities. In the production stage, he recommended doing monitoring and automated setup to identify vulnerabilities. These practices and test techniques are fundamentally used to secure the source code and the development process. The point is to program safely and discover the vulnerabilities as early as possible. Also, Jim Bird referenced strategies for securing the software supply chain. An enormous amount of applications are open source or third-party components. In this way, it is important to detect the dependencies between the used applications. The issue is that regular vulnerabilities are accounted for in open source software. In the event that third party components are utilized, at that point the application is reliant on these components. On the off chance that such parts have vulnerabilities, at that point this application has them as well. If an organization utilizes Docker, Jim Bird recommended doing reliance checks in Docker images. For making sure about CD pipelines Jim Bird referenced that it is important to survey documents and shows, similar to Puppet and Docker records. Also, he brought up that investigating the source code or different records is important to discover mystery certifications. In his eyes, one solution is to diminish the attack surface. This implies unused segments which have realized vulnerabilities ought to be expelled. As he would see it, further viewpoints, for example, observing of delicate information, logs and situations (e.g., production, deployment, and testing environment) are fundamental. To summarize, Jim Bird's book records and notices techniques which can be utilized to identify the vulnerabilities of a CD pipeline. The book gives a overview of tools which can be coordinated into the CD pipelines.

Kuusela discovered in his paper how to coordinate accessible software security tools into a CI process. In view of the literature and the documentation of the tools, he recognized attributes to choose whether the tools are reasonable for the CI process. The tools he has found are constrained to open source/free software which should be easy to integrate and the tools results should be understandable. He has done four case studies in which various tools have been tested. The choice which tools ought to be coordinated is made by the team members of the investigated projects. The tested software were web applications. The accompanying tools were assessed: Brakeman, FindSecurityBugs, OWASP dependency checks, Version Maven Plugin and Retrie.js. In the case studies, just static examination tools and dependency checks were coordinated and tested. Kuusela found that these two techniques can be handily coordinated into the CI procedure. Furthermore, the vulnerabilities distinguished by the tools can be handily dispensed with or moderated.

## 1.1 Why Security is required for web applications

A CD pipeline comprises of tools that are in most cases web based applications (e.g., Jenkins, Bitbucket, JFrog Artifactory). Deepa and Thilagam made an assortment of known methodologies that recognize vulnerabilities in web applications. In addition, they recorded scientific methods and how developers can prevent vulnerabilities in web applications. They referenced that vulnerabilities are available in all phases of the software development lifecycle (SDL), so it is significant that they are examined in each phase of the lifecycle. To secure web applications, the initial step for developers is to create secure program code. Developers should follow rules and use, for instance, programming languages which naturally do datatype checking and memory management (garbage collection).

Vulnerabilities happen for the most part through errors in the source code. Lee et al. found that vulnerabilities in web applications and servers can be distinguished with the fluffing strategy. To utilize this strategy abuse cases which are made out of known vulnerabilities must be created. It is examined whether an issue happens in the software when an input with this generated cases is made.

The OWASP Top 10 list 2017 proposed that SQL injections are the vulnerability which happens in the most applications. several tools has been created to detect SQL injections in web applications. Huang et al. developed a tool called WebSSARI which recognizes vulnerabilities through static analysis strategies on the source code and by runtime assessments. There exist other detection tools for instance Sania by Kosuga et al. or then again the framework WAVES of Hung et al. These tools distinguish the vulnerabilities with various methodologies (syntactic, semantic analysis or black-box approach). Each SQL query is parsed into a tree by Sina. For each query a tree must be generated and stored. Each input query generates a new tree. An attack can be recognized if the new produced tree and the stored tree show contrasts. The framework WAVES attempts to recognize SQL injections and Cross-Site-Scripting (XSS) with the black-box approach. This tool detects vulnerabilities in applications and includes an error scanner. These two standards are essential parts of increasing the security of web applications.

## 1.2 Securing Continuity Deployment pipelines

"A key challenge in a continuous deployment is simply the security of the pipeline". A few people have created approaches for securing the pipelines. Bass et al., Ullah et al., and Rimba et al. created approaches in type of strategies to increase the security level of continuous deployment pipelines. Bass et al. mapped out an engineering procedure for increasing the security of a pipeline. For their case, four stages must be attempted to increase the security level of the pipeline. In the initial step, the assortment of the security requirements must be finished. The following step is to "distinguish the trustworthy and untrustworthy and segments of the pipeline".

Bass et al. referenced that the detection is complex because of the variety of tools which are utilized in a pipeline. Each tool Has its own vulnerabilities.

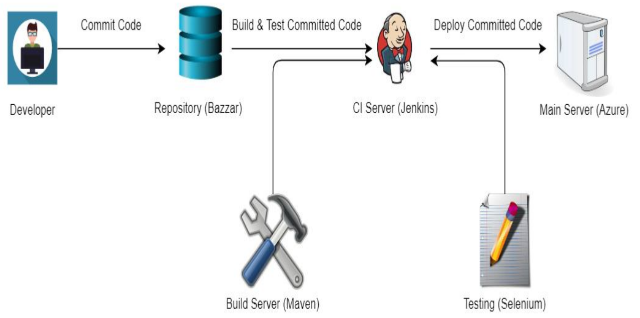


**Figure 1:** Continuous deployment pipeline used in the paper

In this work, the figure 1 investigated pipeline tools are Chef, Jenkins, Docker, Bazzar, and AZURE. The focus in the work of Bass et al. is only set on Jenkins. The result of their work is that they give untrustworthy parts lower rights. Through the confinement of access and permissions they attempt to make the pipeline secure. Subsequently, the attacker can just access the trusted components. These components should be able to prevent the attacks.

The five tactics are:

1. Securing repository through controlled access

2. Securing connection to the main server through use of private key over Secure Shell (SSH).

3. Using roles on the main server to control access

4. Setting up the CI server to start up a Virtual Machine (VM) with a clean state
5. Using Jenkins roles plug-in

The structure of the continuous deployment pipelines is outlined in Figure 1. The continuous deployment pipelines comprise of the following parts: Bazzar, Jenkins, Selenium and Maven. All components are facilitated on an AZURE server aside from the Bazzar repository. Ullah et al. gave no reason why they picked this exceptional structure of continuous deployment pipeline. The considered security parts of this work are access control and visualization. The assessment of their executed strategies were finished with two analysis techniques. The first is a qualitative analysis of continuous deployment pipeline. With the goal structuring notification (GSN) they discovered that their strategies improve the security level of the continuous deployment pipeline. The quantitative analysis which was led with the tools OWASP Scan and OWASP Zed Attack Proxy Scanner gave the outcomes which exhibited that the secure continuous deployment pipeline has less vulnerabilities than the non-secure pipeline. The results of the work additionally show that the Bazzar repository and CI Server (Jenkins) without these actualized strategies have a larger number of vulnerabilities than the primary server (AZURE).

Rather than Lipke, the point of this paper is additionally to look for tools that can detect vulnerabilities in the CD pipeline structure. The referenced papers show that the threat modeling approach is utilized to identify threats and vulnerabilities in critical systems with the objective to ensure the system security. Hence, in this paper threat modeling approach isn't being assessed however the methodology is effectively utilized to detect the threats and vulnerabilities in CD pipelines.

## 2. QUALITATIVE RESEARCH METHOD

So as to get the information on the organization's developers, a survey was led in form of an in-depth online interview. In the initial part of the following section, it is depicted what the qualitative research strategy is and why this technique is picked to gain the information of the developer If a theme isn't known then a subjective methodology can assist with getting fundamental information about it. Mack et al. recommended the qualitative research strategy to comprehend the present issue or the context of a topic. The strategy acquires the experiences and opinions of a sample of the population. Furthermore, Mack et al. referenced that the gained information is utilized to depict the point or issue and not to anticipate or measure the information. They found that the most well-known technique to do subjective research is the in-depth interview. An interview comprises predominantly of open questions, this implies the members answer them with their own words and not just with yes and no. It is important to define the questions so that the members can't reply with yes or no. As per Mack et al., the researchers increase an overview of this technique and more profound understanding into the subject and through the open-ended questions it doesn't limit the participant's perspective.

Toward the start of the research of this paper topic, the in-depth interview technique should assist with getting the information on developers of a software organization. Because of the way that the employees have brief period to spare, the interview is structured in type of an online survey.

### 2.1 Methodology of Survey design of Qualitative research methods

To gain the information of the employees two distinct surveys were planned. With the primary survey the knowledge on an example of employees is acquired. The subsequent survey is organized as follows. The initial part of this survey contains indistinguishable questions as the first survey. The subsequent part incorporates a particular question regarding the CD pipelines which are team members of the projects using the investigated CD pipelines of the case study. The Appendix A incorporates the total questionnaires of both surveys.

Nine inquiries are the equivalent in the two surveys. The initial four questions are about different aspects of CD pipelines (security goals, security attributes, attack scenarios). The following five questions are utilized to gain the profiles of the participants.

The first four questions are:

1. In your opinion which security objectives should be pursued to CD pipelines? Please do not focus on a specific used pipeline. Think in general.
2. In your opinion which security attribute is the most important one in respect to CD pipelines (artifacts, files, scripts, connections, ...)? Order the following security attributes (confidentiality, integrity, availability, authorization, authentication, nonrepudiation) according to their importance. The attribute on top is the most important one for you.
3. In your opinion what are possible attack scenarios for the pipeline you use? Against which attacks would you like to protect your pipeline?
4. Which security objectives are pursued in your project in respect to CD pipelines? Which are implemented? The first question should give an overview about the employee’s thinking in regard to the security of CD pipelines. The gained data of the second question should help to delimit the subject because it reflects the interest and the thought necessity of the employees. The questions in the second part are mostly multiple-choice questions.
5. How many years of experience in software development do you approximately have?
6. Which tools do you know and/or use? Response options: (DevOps tools) Jenkins; Kubernetes; TeamCity; Spinnaker; Travis; GoCD; Concourse CI; JFrog Artifactory; (static analysis tools) PMD; Checkstyle; FindBugs; FindBugs Security; (security tools) OWASP ZAP; BDD Security; JFrog Xray; Security Monkey; Black Duck; Snyk
7. In which role do you interact with your CD pipeline? Response options: user (committing code to the project, usage of the CD pipeline); installation and operation of the pipeline; configuration of the pipeline; other

8. In your opinion how important is the topic security vulnerabilities in CD pipelines? Response options: 1; 2; 3; 4; 5 (1: not important, 5: very important)
9. How often do you deal with security in your development process? Response options: Never; only occasionally; quite often; most of the time; no answer

The fifth, seventh and ninth questions are posed to discover how familiar employees are with security and how frequently they come into contact with the CD pipeline. The sixth question shows to the researcher how much information the employees have in various DevSecOps tool classifications. Since there is an immense number of DevSecOps tools, the selection of tools is made for those which are utilized and known by each employee of the organization or that are known to the researcher so far in time. The static analysis tools are also asked because the organization utilizes these tools in every project to identify mistakes in the source code. Question eight shows to the researcher what priority the security has in their thinking. The extra question of the subsequent part is given below:

10. In the next step think about the security of the [...] CD pipeline. In your opinion how secure is this pipeline? Response opinion: 1; 2; 3; 4; 5 (1: means CD pipeline is insecure, 5: means CD pipeline is secure (pipeline has no vulnerabilities))

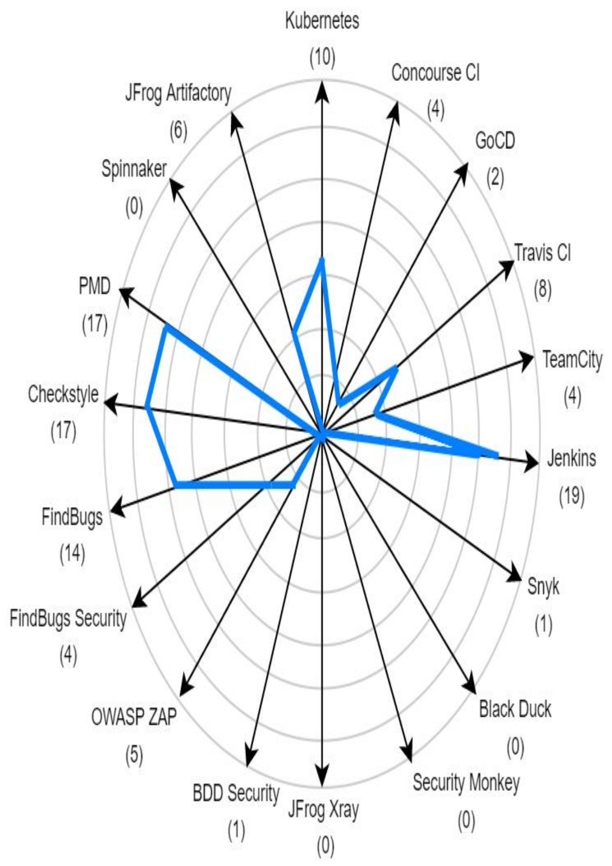
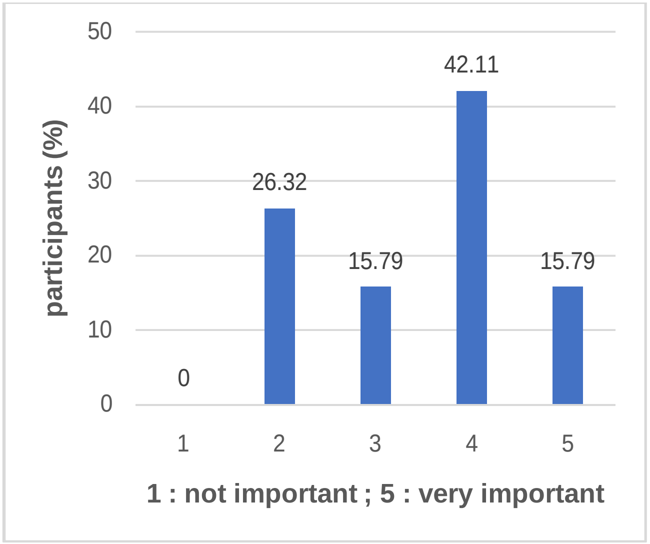
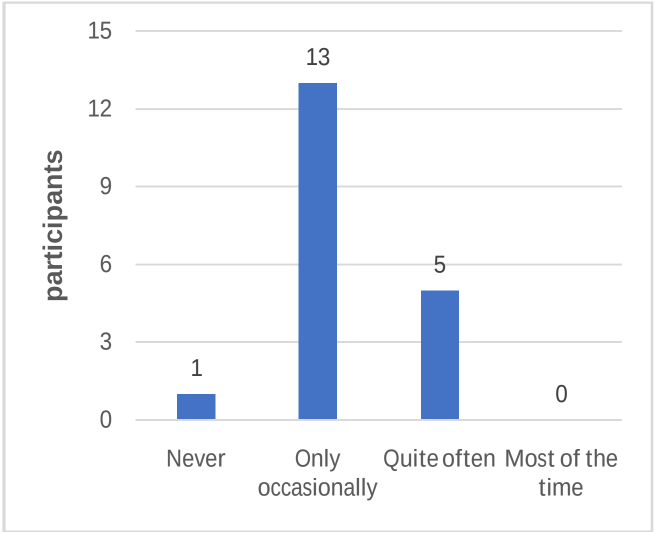


**Figure 2:** Tools which are known and/or used

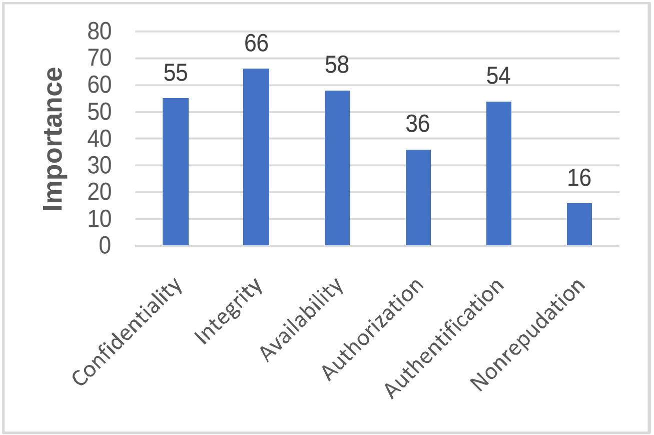
**The role of the developers if they interact with a CD pipeline.**



**Graph 1:** The importance of the topic in an industrial company.



**Graph 2:** The developer's security involvement frequency during the development process



**Graph 3:** Assessment of the six security attributes through developers of a selected company

**Security objectives in industrial projects**

• Requiring authentication and authorization
• Securing credentials and hide critical data.
• Review the process
• No information should be included in the source code of applications
• Implemented access control (not all team members have administrator rights)
• Keep the pipeline components and software up to date

In the figure 2 , It tends to be seen that in the industrial projects authentication and authorization approaches are implemented. furthermore, securing sensitive data and access rights adds to the security of the pipeline. Two participants referenced that they have too less or none security destinations. If a project does not pursue security goals, it can't be ensured to the client that the software will be deployed safely. The results of these four questions help to discover the vulnerabilities in CD pipelines and in the further course to research the pipelines in the case study. A non-recognition of these referenced security targets (results of survey question 4) prompts vulnerabilities and open attack entry points into the CD pipeline. In the case study, it is important to check which sorts of the security goals are kept in the investigated CD pipelines with Graph1 ,2 and 3

## 3. VULNERABILITIES IN CD PIPELINES

Before vulnerabilities in CD pipelines can be recognized, two requirements must be defined. These essentials, a generalized pipeline and potential kinds of threat classifications, are depicted in the initial part of this section. After this, the STRIDE technique is then applied to the generalized pipeline to detect threats and vulnerabilities. The main component which is utilized in both investigated CD pipelines in the case study is Jenkins. Subsequently, the following section of this chapter describes the vulnerabilities in Jenkins which can influence CD pipelines' security. The last section summarizes the results of the survey and of this chapter.

### 3.1. Prerequisites for vulnerability detection

The initial phase in the threat modeling approach is the deterioration of the pipeline in its components and data flows. For the detection of vulnerabilities in CD pipelines, it is important to know which parts and data flows are available. To detect vulnerabilities, a generalized pipeline is required. Subsequently, in the initial part of this section, such a generalized CD pipeline is described.
Vulnerabilities can be derived from threats. So as to detect vulnerabilities, it is important to discover who has an enthusiasm for attacking the pipeline. The following stage is to distinguish the dangers of the CD pipeline. Moreover, a distinction is made among external and internal threats which are described in the subsequent part.

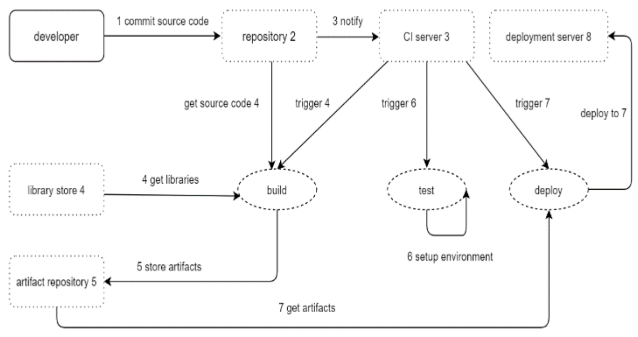


**Figure 3:** Components and data flows of a generalized CD pipeline

**Generalized CD pipeline**

Figure 3 outlines the key components and data flows of a generalized CD pipeline. This pipeline is a generalized mixed version of the named pipeline towards the starting and the investigated CD pipelines.
The components of the pipeline in Figure 2 are the source code repository, the CI server, the library store, the artifact repository and the deployment server. The pipeline's procedures are as per the following: First, a developer submits code changes into the repository. From that point onward, the CI Server is told about this changes. Therefore, the build step is triggered and the external libraries are downloaded from the library store. From that point forward, the code is assembled. The following step is that the built artifact is put away in the artifact repository. The sixth step is that the tests are activated and environment for the test session is set up. If the tests are effectively executed and no mistakes have happened the deployment step is activated. Right now, the artifact is downloaded from the artifact repository and is deployed on a deployment server. If the artifact is accessible on the deployment server, at that point the client can install the artifact and make use of it.

### 3.2. Detection of vulnerabilities in Jenkins

Jenkins is the CI server utilized in both investigated CD pipelines. If the right conditions exist, the accompanying threats could happen: A difference in the Jenkins setup without warning on the Jenkins UI and without Jenkins approval, a difference in security properties without notice on the Jenkins UI and an removal of documents with Jenkins.

### 3.3 Change Jenkins configuration file without notification and Jenkins authorization

The vulnerability is to change the setup document in the Docker environment with no warning on the Jenkins UI. The vulnerability is that the setup file of Jenkins can be changed with no approval and warning on the UI. The consequence of the exploit is that the Jenkins URL is open for all users who knows the URL. Moreover, all users have administrator rights and can do anything they need.

The precondition for the exploit is that you approach the framework where the Docker container is running. With the primary command the Docker name is discovered. The second to fourth commands help the individual to discover the way where the config.xml is put away in the Docker. The following step is to duplicate the insecure config.xml (Appendix A) from outside into the configuration directory of the Docker.

The path to the Docker here is: jenkins-ace:/var/jenkins_home/config.xml. The initial part is the Docker name (here: jenkins-ace ) and the subsequent part is the path to the config.xml in the Docker. After that the Docker container must be restarted.

### 3.4. Remove relevant files with Jenkins

Jenkins grants the execution of discretionary shell commands. The result of this is it is conceivable to irreversibly evacuate folders or files. If the required plugin isn't installed there exists no notice which shows changes on pipeline jobs. Listing 5.2 lists commands which must be executed to remove a folder. The required steps for this action is that you need to install Jenkins in a Docker container and make a pipeline in Jenkins. While making a project, select "execute shell" and insert the following lines of code (see Listing 5.2) into the field. Besides, an folder with the name "removeFolder" must be made in the Docker container under the path /jenkins_home/. From that point onward, the pipeline must be saved. In the following step the pipeline must be built and the "removeFolder" is erased. The first and third command just showcase the directory context when the folder erasure. Accordingly, every folder or pertinent document (e.g., Jenkins config record) can be erased. This can damage the CD pipeline or it is conceivable that the CI server crashes because of this attack.

### 3.5. Summary of vulnerabilities in CD pipelines

The results of the survey and this chapter summarize that CD pipelines may have the following vulnerabilities:
• Internal employees (human errors)
• Unencrypted connections between CD pipeline components
• Insecure environment of the CD pipeline components
• None or few access restrictions
• Use of vulnerable versions of the CD pipeline components
• Vulnerable CD pipeline configurations
• Vulnerable code commits, CD pipeline scripts, Docker images/containers, artifacts
• No review of changes on the CD pipeline

## 4.CONCLUSION

The objective of this paper was to find which vulnerabilities exist with respect to CD pipelines. For the detection of vulnerabilities the qualitative analysis technique for the threat modeling approach was executed on a generalized CD pipeline. It was found that CD pipelines have a few vulnerabilities. If the association between two CD pipeline components is decoded, for instance, an attacker would be able to incorporate malicious code into the CD pipeline. Since just two industrially utilized CD pipelines of a chosen organization were examined, further CD pipelines of different organizations could be tested to recognize extra vulnerabilities in existing CD pipelines. A case study with a few participants from various organizations could affirm that the CD pipelines utilized by most organizations have vulnerabilities where the overall risk level is among medium and conceivably high.

## REFERENCES

[1] M. Shahin, M. Ali Babar, L. Zhu. “Continuous Integration, Delivery And Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices “In: IEEE Access. Vol. 5. 2017, pp. 3909–3943
https://doi.org/10.1109/ACCESS.2017.2685629
[2] P. Rimba, L. Zhu, L. Bass, I. Kuz, S. Reeves.“Composing Patterns to Construct Secure Systems.” In: Proceedings of the 2015 11th European Dependable Computing Conference (EDCC). 2015, pp. 213–224
https://doi.org/10.1109/EDCC.2015.12
[3] K.V.D.KIRAN,” Integrated Distributed Architecture to Integrate Wireless Sensor Networks (WSN) with Grid for Healthcare, ”International Journal of Bio-Science and Bio-Technology”, Vol.7, No.3 (2015), pp.243-250, ISSN: 2233-7849 IJBSBT.
https://doi.org/10.14257/ijbsbt.2015.7.3.26
[4] K.V.D.KIRAN,”A Critical study of information security risk assessment using fuzzy and entropy methodologies,” International Journal on Computers and Communications”, Pages: 17-22,Vol1,Isuue1,Dec-,12, ISSN: 2319 – 8869.
[5] K.V.D.KIRAN,”A tool for analyzing & mitigating application vulnerabilities in any web application, Journal of Advanced Research in Dynamical and Control Systems, Volume 12, Issue 2, 2020, Pages 59-68.
[6] K.V.D.KIRAN,” “Literature Review on Risk Literature Review on Risk and their Components” International Journal for Research in Emerging Science and Technology (IJREST) “,Volume-1, Issue-6,
November 2014”, (e-ISSN 2349-7610).
[7] K.V.D.KIRAN,” Performance Analysis of Layered Architecture to Integrate Mobile Devices and Grid computing with a resource scheduling algorithm”, IEEE CS’07, SIVAKASI, TAMIL NADU, India.
[8] F. Ullah, A. J. Raft, M. Shahin, M. Zahedi, M. Ali Babar. “Security Support in Continuous Deployment Pipeline.” In: Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering. SCITEPRESS - Science and Technology Publications, 2017, pp. 57–68.
https://doi.org/10.5220/0006318200570068

[9] Adele Mailangkay, Eko Indrajit, Raymond Kosala and Acep Hidayat, Analysis of the factors that affecting intention to use Tourism Online Booking,IJATCSE,2019 https://doi.org/10.30534/ijatcse/2019/04862019

[10]. Multiple Level Information Security Using Image Steganography and Authentication, Marilou O. Espina, Arnel C. Fajardo, Bobby D. Gerardo and Ruji P. Medina. IJATCSE,2019

[11]. Security Mechanisms leveraged to overcome the effects of Big Data characteristics, P Amarendra Reddy and O Ramesh, IJATCSE,2019.