INTRODUCTION

In our days, more companies are adopting the DevOps paradigm. DevSecOps is a subset of the DevOps paradigm, which focuses on incorporating software security practises into the DevOps life cycle. While this improves security in projects, the usage alone of DevOps doesn't ensure the created system is secure on the sole basis that DevSecOps was used. In other words, companies are misguided to believe that they can solve all their security needs just by using DevSecOps with no additional input. The paper in question tries therefore to answer the question *Could misuse of the DevOps pipeline subject applications to malicious behaviour?*, the main goal of the paper is therefore to inform and raise awareness about the importance of security in the DevOps pipeline. To this effect the authors point out 3 main contributions they intended for the paper

- Developing a threat model of a DevOps environment utilising Strimzi application as a case study.
- Designing four attacks scenarios that demonstrate four of the threats to the DevOps environment.
- Proposing mitigation techniques for the identified threats.

RELATED WORK

Regarding related work the main focus of the referenced papers was the understanding potential entry points. To this effect the paper references 3 main sources:

Shamim et al, was referenced as a basis for how to execute security in the internet and how multiple systems such as authentication might have an effect how security.

Minna et al, which extended Shamim et al, and who outlined some specific problems regarding K8s clusters.

Karamitsos et al, discussed the influence of the business manager in deciding on accepting risks of the DevOps.

Bertucio et al, analysed the security implications of each component of the supply chain and provided real world examples of possible attack vectors for each component

CASE STUDY SETUP

To test their assertions the authors created a study setup which was used as a stand in for a normal system developed with DevOps

The system includes a simple Python-based web application that uses Apache Kafka and the Strimzi library, which serves as a secure application for later attack scenarios. The system runs on four virtual machines managed by ESXi and uses GitHub as a code repository and Jenkins for CI/CD. The application components are deployed to a Kubernetes cluster, an orchestration tool for Docker containers that allows for monitoring, management, and scaling.

The DevOps pipeline in this case study is therefore (from developer to customer)

- Submitting code to GitHub

- A web hook will trigger a Jenkins build job
- Compiles and packages everything from the Jenkins build into a docker container
- Pushes the docker container to DockerHub
- Deploys the image from DockerHub to the Kubernetes cluster

ATTACK DEMONSTRATION SCENARIOS ON THE DEVOPS CASE STUDY

In order to analyse the potential threats, the components of the environment were broken down. With this 4 main attack types were defined.

Retrieve Information in Topic

The first attack scenario involves an attacker with privileges to deploy containers to a K8s cluster. The attacker creates a custom application with a front-facing web UI to compromise data from secure applications within the SSCS that exist within the cluster. The attacker uses the custom application to siphon data from the Strimzi application and potentially gain access to secretive data.

Manipulate CI/CD by Modifying the Files

The second attack scenario involves an attacker with privileges to the Jenkins instance used in the CI/CD pipeline. The attacker modifies files during the build process before the application is packaged and deployed to a container repository. This could allow the attacker to deploy a modified version of the application that contains a malicious payload, which could potentially put multiple systems at risk if the application is widely used.

Kubernetes Expose cluster-IP to External Users

The third attack involves an attacker who gains access to the K8s cluster and manipulates networking protocols to expose the cluster IP with another ingress object such as a nodeport. The nodeport will attach an external URL that will allow external applications to contact the internal K8s application via the nodeport. The attacker can then hook into the insecure application and access confidential data.

Kubernetes hostPath Namespace Breakout

The final attack is about a hostPath namespace breakout in Kubernetes. An attacker with CRUD privileges in any namespace within the cluster deploys a pod within the allowed namespace and abuses the hostPath volume to mount an escape for privilege escalation. The attacker then accesses the node's root file system and finds the kubeconfig files to gain cluster admin privileges. With this, the attacker can target secure applications on other nodes and perform malicious actions against them, such as editing or deletion.

PROPOSED PROTECTION MECHANISMS

Taken this into consideration the authors infer that the common denominator across all of these attacks is their relation to privilege escalation. Therefore they conclude that better usage of the principle of least privilege when managing a SSCS within the utilised DevSecOps model K8s, is essential to solving these problems. For each attack the authors present some specific solutions that can solve the problems at hand.

Protection from deploying malicious application

To prevent the deployment of malicious applications and the disclosure of confidential information, the most important safeguard is to restrict user privileges. Therefore the authors suggest to establish service accounts that are linked to particular namespaces, thereby preventing users from deploying containers beyond their designated zone.

Protection from the CI/CD manipulation

To prevent unauthorised access and manipulation of the CI/CD pipeline, it is recommended to restrict access to the Jenkins instance. The authors recommend to achieve this by using a service account to trigger Jenkins jobs, while limiting other functions to admin roles only, which should only be used if necessary.

Cluster IP exposure mitigation

To prevent external exposure of the IP address of an internal K8s resource, it is important to implement service accounts within the K8s cluster. Therefore the authors advise to assign service accounts to specific resources within designated namespaces and to closely monitor the activities of users who have access to these internal resources to detect any malicious behaviour.

Host Path volume escalation mitigation

To prevent a hostPath volume namespace breakout, the authors recommend to restrict CRUD privileges to higher-level accounts. Lower-level accounts that require CRUD privileges must authenticate as a high-level user to perform their tasks.

Additionally, the authors recommend acquiring dedicated storage to avoid the need for deploying hostPath volumes. Instead, hosting the storage on another machine that is not part of the main cluster would enhance the security of the system.

CONCLUSIONS

During the course of this paper the authors analysed a case study on a DevOps pipeline and demonstrated how attackers could exploit privilege elevation to make software insecure.

Specifically, they outlined four attack scenarios that involved manipulating the K8s cluster, Jenkins instance, K8s networking, and hostPath volume, showing how some components of the pipeline can be leveraged to make software insecure and how the simple use of DevSecOps is not enough to ensure security in the software in question.

They also showed how better abidance of the principle of least privilege, by limiting admin or root access for lower level accounts, among other solutions..., can mitigate most of these attacks.

While this paper focuses on these specific scenarios, there are numerous other potential attacks at different levels of the system, and future research can include cloud-based K8s environments and other attack vectors beyond privilege escalation.