

Privilege Escalation Attack Scenarios on the DevOps Pipeline Within a Kubernetes Environment

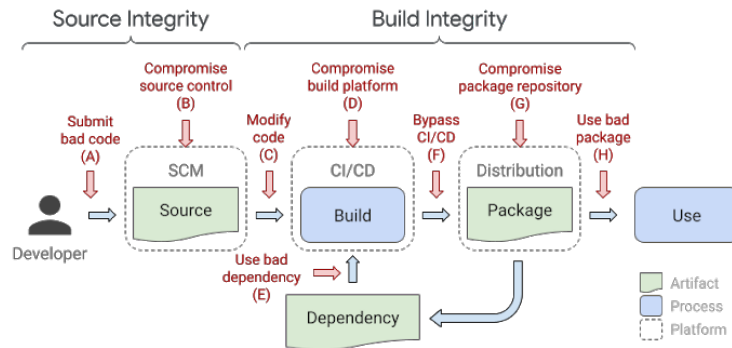
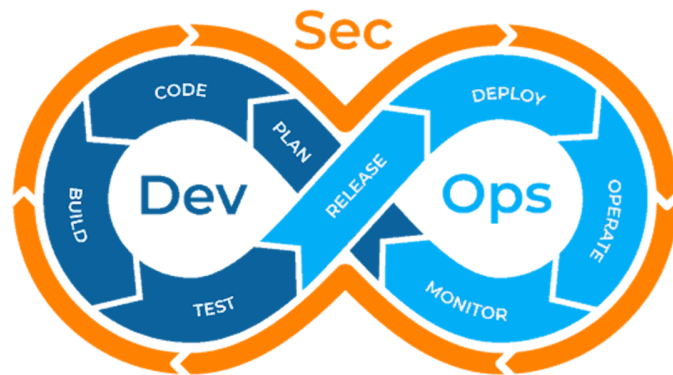
Nicholas Pecka, Lofti ben Othmane, Altaz Valani
May 19, 2022

Background

- DevSecOps
 - Garner quotes 20-50% market penetration
 - Mainstream adoption within 2-5 years
 - Growing space
- Microservice Architecture
 - Kubernetes (K8s)
- Software Supply Chain System (SSCS)
- Companies adopt DevSecOps models believing it solves their security issues without additional input



kubernetes



The Problem

- Attackers exploit the software supply chain putting companies at risk based on their dependencies of the supply chain
 - Introducing risk by placing their SSCS into this new “secure” environment

Motivation :

- Worked on SSCS while employed at Garmin Cyber Security
- Strimzi is meant to utilize K8s

Secure software	Secure software	INSECURE software
Concealed Environment	Contained Environment	External Traffic
Internal Customer Base	Testing w/out external interference	External Customer Base
Known System Interaction	Decline change w/out risk	Unknown System Interaction
Dev. env.	Testing env.	Prod. env.



Research Goal

Research question: Could the misuse of the DevOps Pipeline subject applications to malicious behaviors?

Impact: Companies being misled into thinking they've solved their security issues.
Raise awareness about the need to develop secure systems and harden them

Research Environment

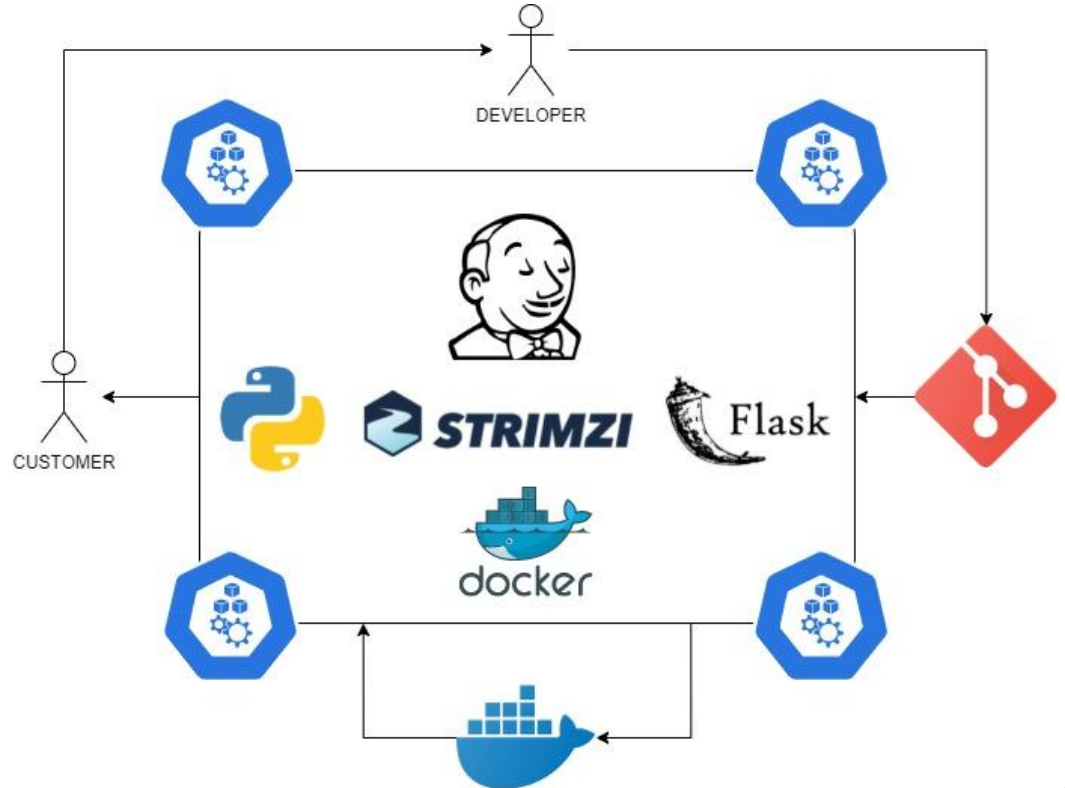
Environment Setup

- Provisioning Bare Metal Server with ESXi
- Configuring K8s cluster on ESXi
- Setting/Configuring applications to create a test SSCS
- Creation of Custom Python Application utilizing Flask

Research Environment

Components Involved

- Git
- ESXI (Bare Metal Server)
- K8s (Rancher Flavor)
- DockerHub
- Jenkins
- Strimzi
- Custom Python application utilizing Flask



Threat Modeling & Assessment

Main steps

1. Identified the component
2. Identify abuses the DevOps pipeline
3. Assess potential weaknesses

Example of Threats on the Software Supply Chain System

System	Potential Flaw	Threat Description	Risk Rating	Mitigation Plan
Git	Account Compromise	Potential privileged account compromise	L	Enable 2 factor auth
Jenkins	Control over CI/CD	Account can be used to alter build configs	M	Enable 2 factor auth
Docker	Docker Pull	Potential infected container	M	Implement vulnerability scanning for containers, Pull from trusted sources
Kubernetes	DDOS	Front end apps externally exposed	M	Proper security checks (ex. check in place to prevent multiple auth)
Strimzi	Network Policy	Default pods can access listeners	L	Configure proper network policy
Custom App	Non sanitized fields	Potential for malicious input	L	Patch to allow sanitized

Attack Scenarios - Outcome of Threat Modeling

1. Retrieve information in a Topic
2. Manipulate CI/CD to modify files
3. Kubernetes expose clusterIP to external traffic
4. Kubernetes Namespace Breakout

Attack 1- Retrieve Information from a Topic

Prerequisite

- User has access to deploy application to K8s cluster
- User is able to communicate with Strimzi

Outcome

- Attacker acquires data managed by Strimzi

Attack 2 - Manipulate CI/CD

Prerequisite

- Attacker has privileges to modify pipeline files within Jenkins

Outcome

- Attacker can modify the build files to insert malicious code before application is packaged
- Container application becomes malicious for all users that have access to pull the container from DockerHub

Attack 3 - Kubernetes Expose Internal clusterIP to External

Prerequisite

- Attacker has privileges to modify networking within K8s cluster

Outcome

- Previously internal IP's become accessible by external sources
- Internal application loses the security bubble established by K8s DNS and is exposed to traditional DNS layer

Attack 4 - Kubernetes Namespace Breakout

- User breaks out of segmented zone obtaining cluster level admin privilege
- Shows how K8s environment makes test application (Strimzi) susceptible to potential attacks
- Gather evidence to support the claim outlined in the goal

Mitigations of the Attack Scenarios

- Principle of Least Privilege
- Implement Gating Mechanism
 - Implement break glass system where users must request further privileges
 - Ensures proper auditing
 - Limiting Monitor Scope
- Host storage on dedicated storage servers
 - Hosting data on the same server as cluster enables possibility of hostPath escalation
 - Move away from single point of failure

Contributions

1. Developed a threat model of a SSCS environment utilizing Strimzi as test application
2. Designed four attacks scenarios that demonstrate four of the threats to the SSCS within a DevSecOps model (K8s)

Available at: <https://github.com/npecka/InsiderAttacksOnTheDevOpsPipeline>

3. Proposed mitigation techniques for the identified threats

⇒ DevSecOps model weaknesses could create insecure Software Supply Chain Systems

Conclusions

- It is not sufficient to assume “secure” software when introducing DevSecOps
- DevSecOps supports developing secure software but attackers may exploit the supply chain to introduce new weaknesses
 - a. Privilege escalation on supporting infrastructure (K8s)
 - b. Rapidly built applications
 - c. Growing pipeline without proper checks

⇒ Companies need to be aware of and plan to mitigate the various weaknesses that come with adopting a new architecture and model

Thank you

Questions?
