CAPSTONE PROJECT REPORT

TITLE:

Securing the Application Lifecycle

https://github.com/ScaleSec/vulnado.git

By:

Gowtham Attili

Mugundhan R

Securing the Application Lifecycle

Problem Statement: Security breaches have been on the rise, and "TechSolutions" doesn't want to take any chances. Your task is to review their application for vulnerabilities, secure the development lifecycle, and implement security best practices in their DevOps processes.

Dataset: A sample web application with known vulnerabilities, available on GitHub. This simulates a typical business application with potential security loopholes.

https://github.com/ScaleSec/vulnado.git

Project Steps:

STEP-1: Application Vulnerability Assessment

- Clone the vulnerable web application from github.
- Run the web application in local
- Download and install OWASP ZAP tool to check vulnerabilities of the application (https://www.zaproxy.org/download/) .
- Add the foxy proxy extension to the browser to identify the vulnerabilities of the application.
- Give the url where our application is running in the browser (https://localhost:1337).
- Do automated and manual scans for the application.
- It'll automatically generates reports,alerts and vulnerabilities.

For better understanding on the vulnerabilities and how to secure the application we did scanning to our application using Nessus tool and got report of vulnerabilities.

Valnado Scan 2

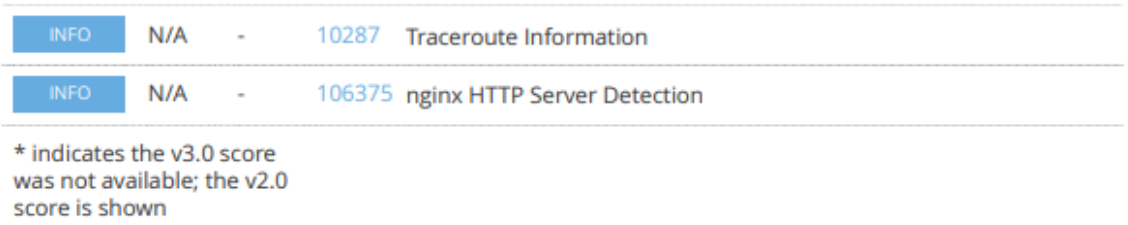
Report generated by Nessus™ Fri, 08 Sep 2023 10:18:25 India Standard Time

Fig 1: Vulnerability report by Nessus Tool

0	0	0	0	20
CRITICAL	HIGH	MEDIUM	LOW	INFO

Vulnerabilities Total: 20

SEVERITY	CVSS V3.0	VPR SCORE	PLUGIN	NAME
INFO	N/A	-	10114	ICMP Timestamp Request Remote Date Disclosure
INFO	N/A	-	166602	Asset Attribute: Fully Qualified Domain Name (FQDN)
INFO	N/A	-	45590	Common Platform Enumeration (CPE)
INFO	N/A	-	54615	Device Type
INFO	N/A	-	10107	HTTP Server Type and Version
INFO	N/A	-	12053	Host Fully Qualified Domain Name (FQDN) Resolution
INFO	N/A	-	24260	HyperText Transfer Protocol (HTTP) Information
INFO	N/A	-	11219	Nessus SYN scanner
INFO	N/A	-	19506	Nessus Scan Information
INFO	N/A	-	11936	OS Identification
INFO	N/A	-	117886	OS Security Patch Assessment Not Available
INFO	N/A	-	70657	SSH Algorithms and Languages Supported
INFO	N/A	-	10881	SSH Protocol Versions Supported
INFO	N/A	-	153588	SSH SHA-1 HMAC Algorithms Enabled
INFO	N/A	-	10267	SSH Server Type and Version Information
INFO	N/A	-	22964	Service Detection
INFO	N/A	-	25220	TCP/IP Timestamps Supported
INFO	N/A	-	110723	Target Credential Status by Authentication Protocol - No Credentials Provided



INFO	N/A	-	10287	Traceroute Information
INFO	N/A	-	106375	nginx HTTP Server Detection

* indicates the v3.0 score was not available; the v2.0 score is shown

Fig 2: These are the vulnerabilities identified by Tenable Nessus Tool.

STEP-2: Container Security:

- The Application has client as frontend , vulnado, internal-site as middleware and postgres database as backend.
- So we dockerized the application into four dockerfiles by using docker best practices.
- These are the dockerfiles created for Vulnado Application

Dockerfile for client:

```
FROM nginx:alpine
COPY . /usr/share/nginx/html
```

Dockerfile for vulnado:

```
FROM openjdk:8

RUN apt-get update && \
apt-get install build-essential maven default-jdk cowsay netcat -y &&
\
update-alternatives --config javac
COPY . .

CMD ["mvn", "spring-boot:run"]
```

Dockerfile for internal-site:

```
FROM nginx:alpine
COPY . /usr/share/nginx/html
```

Docker-compose file for vulnado application:

```
version: "3"
services:
```

```
vulnado:
build: .
ports:
- 8081:8081
links:
- db
- internal_site
environment:
- PGPASSWORD=vulnado
- PGDATABASE=vulnado
- PGHOST=db:5432
- PGUSER=postgres
depends_on:
- "db"

client:
build: client
ports:
- 1337:80

db:
image: postgres
environment:
- POSTGRES_PASSWORD=vulnado
- POSTGRES_DB=vulnado

internal_site:
build: internal_site
```

STEP-3: DevSecOps Implementation:

- Integrated OWASP Dependency check with the CI/CD pipeline to detect security vulnerabilities in dependencies.
- Integrated Snyk with the CI/CD pipeline to detect security vulnerabilities in dependencies and in our application.
- These two will generate vulnerability reports.

Steps to add owasp dependency check in jenkins pipeline :

1. Install OWASP Dependency Check plugin in the plugins section in manage jenkins.
2. Add the Owasp tool in the global tool configuration.
3. Add owasp dependency stage in the jenkins pipeline.

Owasp Dependency Check stage to be added in Jenkins Pipeline:

```
stage('OWASP Dependency-Check Vulnerabilities') {
steps {
dependencyCheck additionalArguments: '''
-o './'
-s './'
-f 'ALL'
--prettyPrint''', odcInstallation: 'owasp-zap'
dependencyCheckPublisher pattern: 'dependency-check-report.xml'
}
}
```

Steps to add Snyk security check in jenkins pipeline :

1. Install Snyk security plugin in the plugins section in manage jenkins.
2. Add the Snyk security scan in the global tool configuration.
3. Create an account in Snyk and generate API token and copy the token.
4. Go to the credential manager in jenkins and add the credentials as Snyk api token and give the credential id.
5. Add Snyk security scan stage in the jenkins pipeline using pipeline script generator.

Snyk security scan stage to be added in Jenkins Pipeline:

```
stage('Snyk Security') {
steps {
snykSecurity failOnError: false, failOnIssues: false, organisation:
'attiligowtham', projectName: 'vulnado', snykInstallation: 'snyk',
snykTokenId: 'snyk-id', targetFile: 'pom.xml'
}
}
```

EKS Cluster Setup:

Create an EKS cluster in the same instance where you set up jenkins. Following are the steps to create EKS Cluster:

- Install Kubectl

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
sudo mv kubectl /usr/local/bin
kubectl version --client
```

- Install eksctl

```
curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl
_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
```

- Install AWS CLI

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
sudo apt install unzip
sudo unzip awscliv2.zip
sudo ./aws/install
```

- Create IAM Role with following policies:
    1. EC2 full access
    2. VPC full access
    3. IAM full access
    4. Cloudformation access
    5. Administrator Access
    6. Attach it to the instance (Ec2 dashboard --> instance --> actions -->security --> modify IAM role --> select this role)

- Create EKS cluster using following command

```
eksctl create cluster --name cluster_name \
--region aws_region \
--node-type instance_type \
--nodes-min 2 \
--nodes-max 2 \
--zones availability-zone1,availability-zone2
```

- Check whether the cluster is created or not using the following command

```
kubectl get nodes -o wide
```

Istio Setup in EKS Cluster:

Download Istio using the following documentation given below.
https://istio.io/latest/docs/setup/getting-started/

Create and apply manifest files for all the services in our application in eks cluster by following commands:

- Go inside the istio folder

```
cd istio-1.19.0
```

- Create Istio ingress gateway to route traffic to our application.

```
vi gateway.yml

apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
name: java-app
spec:
selector:
istio: ingressgateway # use istio default controller
servers:
- port:
number: 80
name: http
protocol: HTTP
hosts:
- "*"
```

- Create Virtual Services to route traffic to ours destination i.e., client (frontend).

```
vi vs.yml

apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
name: java-app-vs
spec:
```

```
hosts:
- "*" # This will match any host
gateways:
- java-app # Reference to the Gateway created above
http:
- route:
- destination:
host: client-service # The name of your service in Kubernetes
port:
number: 8080 # The port your service is listening on
```

- Check whether the gateway and virtual services are created or not

```
root@ip-10-0-0-27:/home/ubuntu# kubectl get gateway
NAME       AGE
java-app   24h
root@ip-10-0-0-27:/home/ubuntu# kubectl get vs
NAME          GATEWAYS        HOSTS   AGE
java-app-vs   ["java-app"]    ["*"]   24h
```

- Create manifest files for client,vulnado,internal-site and db.

  Client.yml

```
vi client.yml

apiVersion: apps/v1
kind: Deployment
metadata:
name: client-deployment
spec:
replicas: 1
selector:
matchLabels:
app: client
template:
metadata:
labels:
app: client
spec:
containers:
- name: client
image: gowtham47/client:lastest
ports:
- containerPort: 80
---
apiVersion: v1
kind: Service
```

```
metadata:
name: client-service
spec:
selector:
app: client
ports:
- protocol: TCP
port: 80
targetPort: 80
type: LoadBalancer
```

Vulnado.yml

```
vi vulnado.yml

apiVersion: apps/v1
kind: Deployment
metadata:
name: vulnado-deployment
spec:
replicas: 1
selector:
matchLabels:
app: vulnado
template:
metadata:
labels:
app: vulnado
spec:
containers:
- name: vulnado
image: gowtham47/vulnado
ports:
- containerPort: 8081
env:
- name: PGPASSWORD
value: vulnado
- name: PGDATABASE
value: vulnado
- name: PGHOST
value: db:5432
- name: PGUSER
value: postgres
```

Internal-site.yml

```
vi internal-site.yml

apiVersion: apps/v1
kind: Deployment
metadata:
```

```
name: internal-site-deployment
spec:
replicas: 1
selector:
matchLabels:
app: internal-site
template:
metadata:
labels:
app: internal-site
spec:
containers:
- name: internal-site
image: gowtham47/internal-site
```

Db.yml

```
vi db.yml

apiVersion: apps/v1
kind: Deployment
metadata:
name: db-deployment
spec:
replicas: 1
selector:
matchLabels:
app: db
template:
metadata:
labels:
app: db
spec:
containers:
- name: db
image: gowtham47/postgres
env:
- name: POSTGRES_PASSWORD
value: vulnado
- name: POSTGRES_DB
value: vulnado
```

- Apply all the manifest files using following commands

```
kubectl apply -f client.yml
kubectl apply -f vulnado.yml
kubectl apply -f internal-site.yml
kubectl apply -f db.yml
```

- Check whether the deployments are up and running

```
root@ip-10-0-0-27:/home/ubuntu# kubectl get deployments
NAME READY UP-TO-DATE AVAILABLE AGE
client-deployment 1/1 1 1 24h
db-deployment 1/1 1 1 24h
internal-site-deployment 1/1 1 1 24h
vulnado-deployment 1/1 1 1 24h
```

Now Integrate the EKS Cluster with the Jenkins pipeline to automate the complete CI/CD pipeline.

COMPLETE CI/CD PIPELINE FOR SECURING THE APPLICATION LIFE CYCLE.

```
pipeline {
    agent any
    tools{
        jdk 'jdk'
        maven 'maven'
    }
    environment {
        aws_region = "us-west-1"   //Your aws region
        eks_cluster = "myeks"  // Your cluster
    }

    stages {
        stage('checkout') {
            steps {
                git branch: 'master', url: 'https://github.com/Gowtham-
745/vulnado.git'  //update your repo
            }
        }
                stage('OWASP Dependency-Check') {
                        steps {
                                dependencyCheck additionalArguments: '''
                                -o './'
                                -s './'
```

```
                        -f 'ALL'
                        --prettyPrint'", odcInstallation: 'owasp'
                        dependencyCheckPublisher pattern:
'dependency-check-report.xml'
                    }
                }
        stage('Clean') {
            steps {
                // Get some code from a GitHub repository
                git 'https://github.com/Gowtham-745/vulnado.git'



                // To run Maven on a Windows agent, use
                sh "mvn clean"
            }
            post {
                // If Maven was able to run the tests, even if some of the
test
                // failed, record the test results and archive the jar file.
                success {
                    echo 'Cleaning Project is Done'
                }
            }
        }

        stage('Compile') {
            steps {
                // Get some code from a GitHub repository
                git 'https://github.com/Gowtham-745/vulnado.git'
                // To run Maven on a Windows agent, use
                sh "mvn compile"
            }

            post {
                // If Maven was able to run the tests, even if some of the
test
```

```
            // failed, record the test results and archive the jar file.
            success {
              echo 'Compiling Project is Done'
            }
          }
        }



        stage('Snyk Security') {
          steps {
            snykSecurity failOnError: false, failOnIssues: false,
organisation: 'attiligowtham', projectName: 'vulnado',
snykInstallation: 'snyk', snykTokenId: 'snyk-id', targetFile:
'pom.xml'
            //bat
'C:\\Users\\attil\\AppData\\Roaming\\npm\\node_modules\\snyk\\wra
pper_dist\\snyk-win.exe test'
          }
        }



            stage('Package') {
          steps {
            // Get some code from a GitHub repository
            git 'https://github.com/Gowtham-745/vulnado.git'
            // To run Maven on a Windows agent, use
            sh "mvn -Dmaven.test.failure.ignore=true clean package"
          }

          post {
            // If Maven was able to run the tests, even if some of the
test
            // failed, record the test results and archive the jar file.
            success {
             archiveArtifacts 'target/*.jar'
```

```
            }
        }
    }

    stage("Docker Build & Push"){
        steps{
            script{
              withDockerRegistry(credentialsId: 'docker-hub',
toolName: 'docker') {
                    sh "docker tag vulnado_vulnado
gowtham47/vuln:latest "
                    sh "docker push gowtham47/vuln:latest "
                    sh "docker tag vulnado_client gowtham47/clin:latest
"
                    sh "docker push gowtham47/clin:latest "
                    sh "docker tag vulnado_internal_site
gowtham47/ins:latest "
                    sh "docker push gowtham47/ins:latest "
                    sh "docker tag postgres gowtham47/d-b:latest "
                    sh "docker push gowtham47/d-b:latest "
                }
            }
        }
    }

    stage('deploy') {
        steps {
            script {
                sh "aws eks --region $aws_region update-kubeconfig --
name $eks_cluster"
                sh "kubectl get svc"
            }
        }
    }
  }
}
```

Build the pipeline to build, check for owasp dependency check and snyk security scan and to push the images to docker hub and deploy the application in the EKS cluster.
-> To access the application through the browser use the link from the output of the pipeline from the deployment stage.
Console Output to access the application:

```
+ aws eks --region us-west-1 update-kubeconfig --name myeks
Updated context arn:aws:eks:us-west-1:670161448272:cluster/myeks in /var/lib/jenkins/.kube/config
[Pipeline] sh
+ kubectl get svc
NAME             TYPE           CLUSTER-IP     EXTERNAL-IP                                                                PORT(S)        AGE
client-service   LoadBalancer   10.100.70.65   a293a6ac28c984f7da9c8f9b5cc708c6-161162788.us-west-1.elb.amazonaws.com      80:30965/TCP   6h29m
kubernetes       ClusterIP      10.100.0.1     <none>                                                                     443/TCP        4d2h
```

Console Output of Owasp Dependency Check Phase:

Dependency-Check Results

SEVERITY DISTRIBUTION

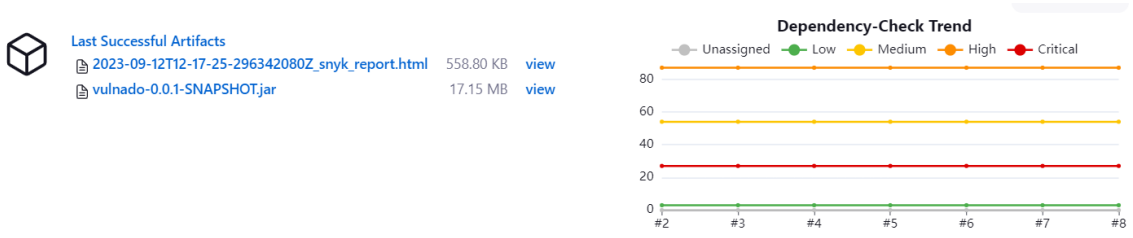
27 | 87 | 54 | 3

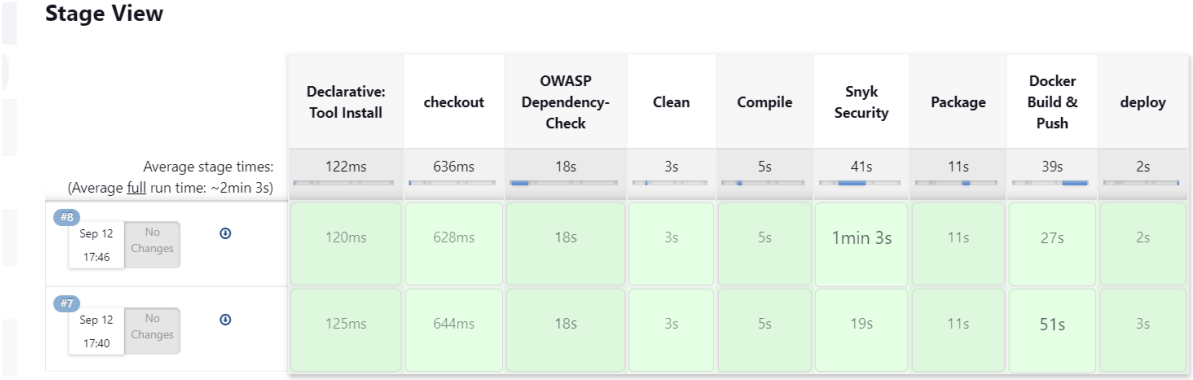
Search

File Name	Vulnerability	Severity	Weakness
vulnado-0.0.1-SNAPSHOT.jar: hibernate-validator-6.0.14.Final.jar	NVD CVE-2019-10219	Medium	CWE-79
vulnado-0.0.1-SNAPSHOT.jar: hibernate-validator-6.0.14.Final.jar	NVD CVE-2020-10693	Medium	CWE-20
vulnado-0.0.1-SNAPSHOT.jar: jackson-annotations-2.9.0.jar	NVD CVE-2018-1000873	Medium	CWE-20
vulnado-0.0.1-SNAPSHOT.jar: jackson-core-2.9.8.jar	NVD CVE-2022-45688	High	CWE-787
vulnado-0.0.1-SNAPSHOT.jar: jackson-databind-2.9.8.jar	NVD CVE-2019-12086	High	CWE-502
vulnado-0.0.1-SNAPSHOT.jar: jackson-databind-2.9.8.jar	NVD CVE-2019-12384	Medium	CWE-502
vulnado-0.0.1-SNAPSHOT.jar: jackson-databind-2.9.8.jar	NVD CVE-2019-12814	Medium	CWE-502
vulnado-0.0.1-SNAPSHOT.jar: jackson-databind-2.9.8.jar	NVD CVE-2019-14379	Critical	CWE-1321
vulnado-0.0.1-SNAPSHOT.jar: jackson-databind-2.9.8.jar	NVD CVE-2019-14439	High	CWE-502
vulnado-0.0.1-SNAPSHOT.jar: jackson-databind-2.9.8.jar	NVD CVE-2019-14540	Critical	CWE-502

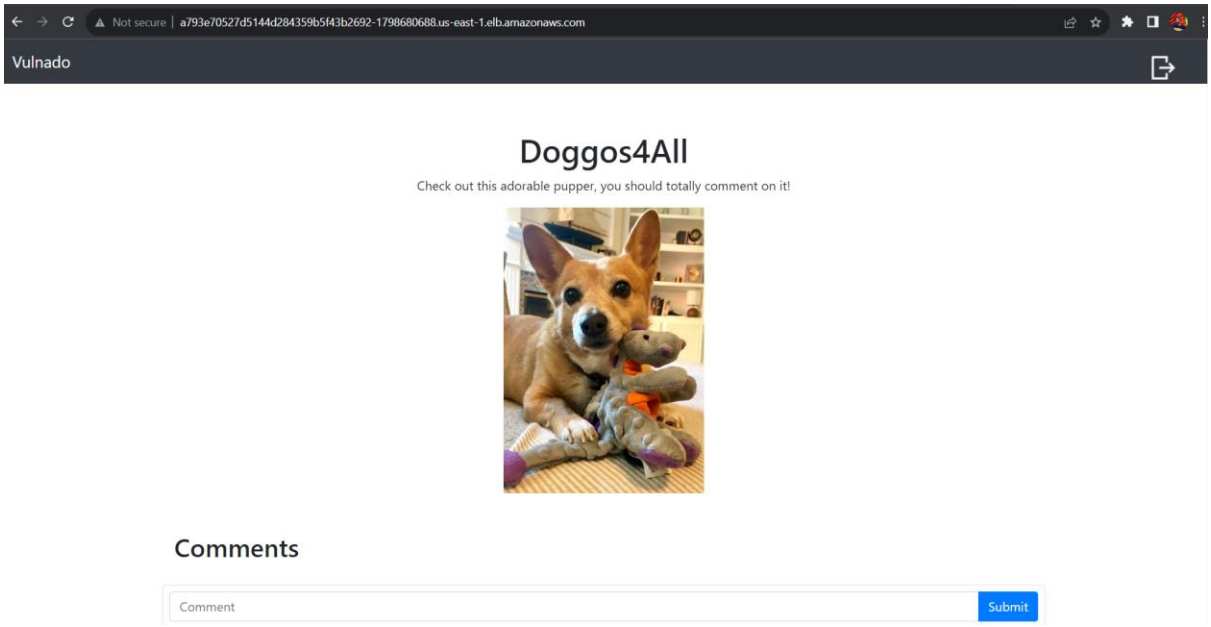
Console Output of Snyk security stage:

Build overview of pipeline:

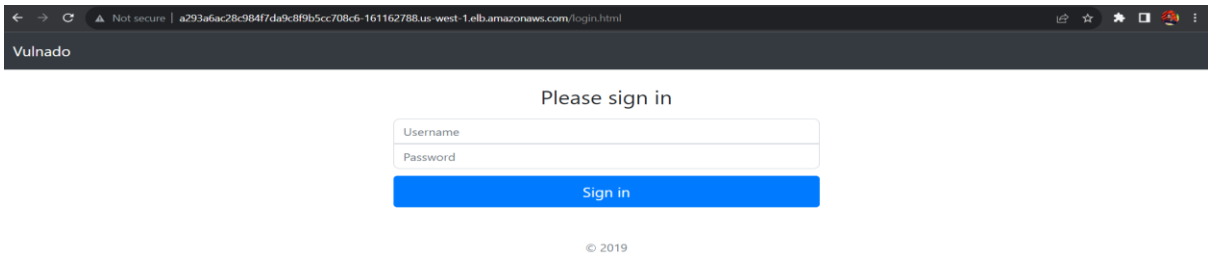




Output:



Output:



THE END