UDACITY

Return to Classroom

DISCUSS ON STUDENT HUB

Hardened Microservices Environment

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Nicely done! You accurately completed the project. All the files were clearly labeled and align with the spec. Your diagram accurately describes the microservices environment. Your threat model accurately captures the threat surface for docker and Kubernetes. You hardened the docker daemon and RKE cluster successfully. You fixed the vulnerable libs in the vuln_app and fixed the XSS code aw. You got Falco running and sent events to Grafana. Nice job! I hope you enjoyed the project. I tried to make it fun and actionable.

Additional lecture

The following links might be useful, have a look at these in your free time.

- What is microservices architecture?
- Advantages and Disadvantages of Microservices Architecture
- Microservice and Container Security: 10 Best PracticeIt

Threat Model the Microservices Environment

✓

The screenshot security_architecture_design.png contains a line and box diagram showing the relationship

between the 4 core components.

✓

Submission should include 5 concrete attack surface areas for the Docker environment and 5 concrete attack surface areas for the Kubernetes control plane, etcd, and worker.

In the explanation, associate each attack surface area to at least one pillar of the STRIDE model, which includes

- Spoofing
- Tampering
- Repudiation
- Information Disclosure
- Denial of Service
- Elevation of Privilege

There can be multiple attack surface areas associated with one pillar, but the attack surface areas have to be distinct.

Harden the Microservices Environment

✓

The screenshots for Docker-bench initial run and re-run contain the result summary and all failed Docker-bench findings.

The threat_modeling_template.txt file should contain the 3 failed Docker-bench findings you want to harden. These findings need to be reflected in the screenshot and should be related to the 5 Docker attack surfaces from Step 1.

The screenshots for the Docker-bench re-run show that the 3 findings you identified as needing to be fixed now pass upon the Docker-bench re-run.

✓

The screenshots for the Kube-bench initial run and re-run contain the result summary and all failed Kube-bench findings.

The number of failed findings in the Kube-bench re-run is less than that in the Kube-bench initial run.

The `1.1.12` Kube-bench check no longer fails in the Kube-bench re-run after baseline hardening.

The Kubernetes-specific test plan consists of at least 200 words and minimally describes

- How you will test the changes
- How you will ensure the changes don't negatively affect your cluster

Harden and Deploy the Flask App

Harden and Deploy the Flask App

The submission includes the `app.py` file with changes that fix the Cross-Site Scripting vulnerability.

Good job here. You were able to remediate the XSS vulnerability

```
setup_jinja(app, loader=PackageLoader('sqli', 'templates'),
                context_processors=[csrf_processor, auth_user_processor],
                autoescape=True)
```

If you had more time, could you identify and fix the other vulnerabilities?

The screenshot grype_app_out_of_box.png contains the Grype output.

The screenshot grype_app_hardended.png shows 0 findings for vulnerable libraries.

Implement Grafana and run-time monitoring

The screenshot kube_pods_screenshot.png shows Falco and falco-exporter pods running.
The screenshot falco_alert_screenshot.png from Falco pod logs or from the falco-exporter metrics page shows that Falco is generating security events.

In the screenshot falco_grafana_screenshot.png, the Falco Grafana panel should display graphs indicating the sensitive file being read.

Incident Response

In the incident_response_report.txt, provide an incident response report that answers all these questions (at least two sentences for Questions 2-6):

1. Incident commander name
2. Executive summary of what happened
3. Summary of what was affected
4. Summary of the impact
5. Summary of the remediation steps

6. Summary of lessons learned

The report should be free of grammar and spelling mistakes.

- Executive summary of what happened ✅
- Summary of what was affected ✅
- Summary of the impact ✅
- Summary of the remediation steps ✅
- Summary of lessons learned ✅

Here is a suggestion to make your incident response report even better:

Summary of lessons learned: Crypto mining pods can trivially be run using commonly available Docker images. In admission to controller based restrictions, it's important to protect access to the administrative terminal that can run privileged kubectl commands.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START