|  |  |
| --- | --- |
| Name: | Ajay Jain |
| Lab User ID: | 23SEK3324\_U33 |
| Date: | 10/01/2024 |
| Application Name: | Wrong Secrets |

**Follow the below guidelines:**





System Architecture:

(Understand the system and document the physical and logical architecture of the system, use the shapes and icons to capture the system architecture)



Wrong Secrets deployment

Master

 

Wrong secrets service

Application hosted on localhost: 8080

Worker-1



Gremlin for chaos Engineering

Worker-2

Define system’s normal behavior:

(Define the steady state of the system is defined, thereby defining some measurable outputs which can indicate the system’s normal behavior)

The Different challenges tab is responsive

We can submit the answer in the challenges

We can see the hints by clicking on the hint button

Hypothesis:

(During an experiment, we need a hypothesis for comparing to a stable control group, and the same applies here too. If there is a reasonable expectation for a particular action according to which we will change the steady state of a system, then the first thing to do is to fix the system so that we accommodate for the action that will potentially have that effect on the system. For eg: "If one of our database servers fails, our service will automatically switch to a backup server, and users will not experience any downtime or data loss.")



**Known**

In case a node fails we know that application will run on the second node but we don’t know the time taken for it to run on that node

If a node fails then we know then the replicas will ensure that the application runs perfectly on the Active Node.

**Unknown**

If we shutdown the replica nodes then we don’t know whether the pseudo primary nodes and it’s replicas are not able to take over the transactions effectively

If we shut down all the replicas then we don’t know the time taken to restore the service. But we do know that the pseudo primary and its replicas can carry out the transactions.

**Unknown**

**Known**

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

**Implementation:**

1. Firstly, Set up the Kubernetes cluster with a master and 2 worker nodes
2. After setting up the Kubernetes cluster Create a Deployment and Service and run them
3. After deploying the application make use of localhost IP with port no to view the live application
4. Upload the repository on Snyk to view the vulnerabilities in the webpage.

**Observation and Results:**

1. Upon using Snyk tool we find several Vulnerabilities. Some vulnerabilities are listed below.

**Vulnerabilities Found:**

1. **Improper Authentication:** This weakness occurs when application improperly verifies identity of a user.

**Fix:**

* Implementing strong authentication methods that feature anti-brute force and session protection mechanisms.
* Enforcing account lockouts, rate limiting, IP-based monitoring, application firewalls, and CAPTCHAs to prevent brute-force attacks.
* Enforcing a secure password policy by creating a password checker that tells users how strong their passwords are in real-time.

1. **Improper Access Control:** Improper access control is a vulnerability that occurs when a system does not properly restrict or enforce access to resources, such as files, directories, network resources, or application functions.

**Fix:**

Deny access by Default: Except for public resources.

Least Privilege: Implement least-privilege roles. Access should be granted only to those who need it.

Re-use Access Control Mechanisms: Implement access control mechanisms once and re-use them throughout the application

Record Ownership: Model access controls should enforce record ownership.

Disable Web Server Directory Listing: Ensure file metadata (e.g., .git) and backup files are not present within web roots.

Log Access Control Failures: Alert admins when

1. **Improper Input Validation:** Improper input validation is a security vulnerability that occurs when an application does not properly validate or sanitize input data before processing it.

**Fix:** apply proper input validation and sanitization techniques such as Django Validators, etc.

1. **Cross-Site Request Forgery( CSRF ):** Cross site request forgery (CSRF) is a vulnerability where an attacker performs actions while impersonating another user.

**Fix:** CSRF tokens, SameSite cookies, Renew authentication for security critical actions

