Assignment No-2

(1) Describe how GitLab CI can be integrated with SAST tools like SonarQube?

Ans Integrating GitLab CI with Static Application Security Testing (SAST) tools like SonarQube involves several steps to ensure that code quality and security vulnerabilities are assessed during the continuous integration process.

(1) Setup SonarQube:- Install SonarQube: Download and install SonarQube on a server. You can use Docker for easy deployment.

```
docker run -d --name sonarqube -p 9000:9000 sonarqube
```

Create a project:→ Login in to SonarQube and create a new project. Note down the project key and token.

(2) Configure GitLab CI/CD:→ 1) Add a GitLab CI configuration:- Create or edit your .gitlab-ci.yml file in your repository.

ii) Define the Build Job: Add a job for building your application. This job should run before the SAST job.

```
stages:
  - build
  - test
  → sonarqube.
```

Step3 Integrate sonarqube in Gitlab CI

1) Add SonarQube Job → Create a job in your .gitlab-ci.yml file to run sonarQube analysis.

```
Sonarqube:
    Stage: sonarqube.
    image: sonarsource/sonar-scanner-client:latest
```

Add SonarQube Configuration file :→ Optimally, create a sonar-project.properties file in your repository to define project setting which can be referenced in your job.

Step4 Run CI/CD pipeline

1) Push Changes: push changes to your Gitlab repository. The CI/CD pipe should trigger automatically.

2) monitor the pipeline:→ monitor the Jobs in the Gitlab CI/CD pipeline. The sonarQube Job will execute and send the analysis result to your sonarQube server.

(2) Compare the capabilities of OWASP ZAP with other web application security testing tools. what sets it apart?

Ans Comparision with other tools.

feature	OWASP ZAP	Burp Suite	Acunetix	Netsparker	Nikto
cost	free & open source	paid (community edition free)	paid	paid	free and open-source.
User interface	User-friendly GUI	comprehensive GUI	User-friendly GUI	User friendly GUI	Basic CLI/GUI
Active scanning	Yes	Yes	Yes	Yes	Limited active scanning.
API support	Yes (REST API)	Yes (Extensive API)	Yes (API available)	Yes (REST API)	Basic API.
Plugin Architecture	Highly extensible with plugin	plugin support	limited custom plugin	plugin architecture	Limited extensibility
Reporting	Customizable reports	Detailed reports	Advanced reporting	Detailed reports	Basic reports.

The key feature that set OWASP ZAP apart from other web application security.

(I) Open source & free.
(II) User-friendly Interface.
(III) Strong community & support
(IV) Comprehensive API
(V) Flexibility in scanning
(VI) Extensible plugin Architecture.

(3) How does Nagios monitor multiple servers in a network?

Ans Nagios is a powerfull open-source monitoring tools that helps organizations monitor various components of their IT infrastructure, including server, application and Network devices.

(i) Architecture overview:→

- Nagios core:→ This is the central server component that performs the monitoring. It collects data, generates alerts, and provides a web interface for viewing status and reports.

(ii) monitoring mechanisms:→ @ Active Checks:→ polling: Nagios actively polls each server at regular intervals. It sends requests to the monitored services (eg. HTTP, FTP, SSH) & receives responses.

plugins: Nagios uses plugins to perform the checks. Each check is executed by calling the corresponding plugins scripts that returns the services status.

(iii) Service and Host monitoring:→ Services: Nagios monitors various services on each server (eg. CPU usage, memory usage, disk space, network connectivity) by defining specific checks for each service.

(4) Notification & Alerts :- Thresholds : Administrators set thresholds for different metrics (eg CPU load > 90% triggers a warning). If the thresholds are exceeded, Nagios generates alerts.

(4) How does snort identify suspicious network activity and potential security threats?

Ans Snort is an open-source network intrusion detection and prevention system (NIDS/NIPS) that identifies suspicious network activity and potential security threats through several key mechanisms.

(1) Packet Sniffing :- Snort captures and analyzes network packet in real-time it can run in various modes, including packet logger mode

(2) Signature-Based Detection :- Snort uses predefined rules (signature) to identify known threats. Each rules defines a specific pattern or behaviour that indicates malicious activity, such as scanning for open ports, specific pattern or behaviour that indicates malicious activity.

(3) Protocol Analysis :→ Snort can analyze the structure of different network protocols (like TCP/IP, UDP, Icmp) to detect anomalies or violation of protocol standards, which may indicate suspicious activity.

(4) Anomaly Detection :- In addition to signature based detection, Snort can implement anomaly detection techniques. This involves creating a baseline of normal network behaviour and then identifying deviations from this baseline, which may indicate potential threats.

(5) Logging and Alerts :→ When Snort detects suspicious activity, it logs the events and generates alerts.

(5) Explain how AWS Lambda differs from traditional computing models.

Ans AWS lambda is a serverless computing service that significantly differs from traditional computing models in several key ways.

1) No server management (serverless Architecture) → AWS lambda: You don't need to manage servers. AWS handles the underlying infrastructure, scaling, and availability automatically

Traditional computing:- You need to provision, configure and manage physical or virtual servers yourself, including task like installing software, managing storage.

2) Event - Driven Execution:- Aws lambda:- code execution is event-driven, meaning it is triggered by specific effects like HTTP request, file uploads to Amazon S3, or database changes.

Traditional computing:- Application typically run continuously, even when no specific event are occuring.

3) Auto scaling:- Aws lambda: it automatically scales to handle the number of incoming requests by running multiple copies of the function in parallel, without any need for user intervention.

Traditional computing:- scaling is manual or semi-automated. you must set up auto-scalling policies, load balancers, or provision more servers manually to handle increased traffic.

(6) Create a lambda function that interacts with an Aws service (eg. S3, DynamoDB) & returns a response.

Ans Here's an example of an AWS lambda function that interacts with an AWS-service- in this case, Amazon S3. The function will list all the object in a specific S3 bucket and return them as a response.

Prerequisite: Ensure you have created an S3 bucket
Grant your lambda function permission to interact with S3 via the AWS identity & Access management (IAM) role.

Creating a Lambda function in AWS involves several steps, from setting up your AWS environment to writing & deploying function.

Lambda function

```
Const AWS = require('aws-sdk');
Const S3 = new AWS.S3();
exports.handler = async(event) => {
    Const bucketName = "bucket-name";
    try {
        Const params = {
            Bucket: bucketName
        };
        Const data = await S3.listObjectsV2(params).
                                promise();
        Const objectKeys = data.Contents.map(item =>
                                item.key);
        return{
            status code: 200
        };
```

```
}
  Catch (error) {
        (onsole.error ('Error listining objects:', error)

return {
      statusCode: 500,
      body: JSON.Stringify ({
        message: "Error
        error: error.message
      }),
    };
  }
```

permissions: Ensure the Lambda function has the necessary permissions to access the S3 bucket

- Attach the Amazon S3 ReadOnly Access policy or create a custom policy with the following permission.

```
{ "Effect": "Allow",
  "Action": [
        "S3: List Bucket"
    ],
  "Resource": [
        arn: aws: s3:: bucket-name
    ]
}
```

Environment Variables:- You can optionally store the bucket name in an environment variable to make the function more dynamic.