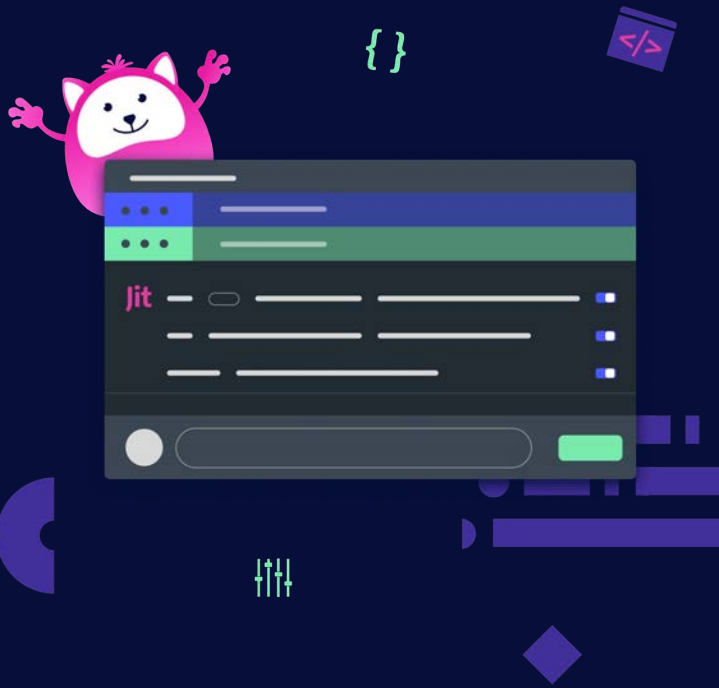
OWASP Serverless Top 10

As Code



Shlomi Kushchi, System Architect at Jit

March, 2023

Who am I?

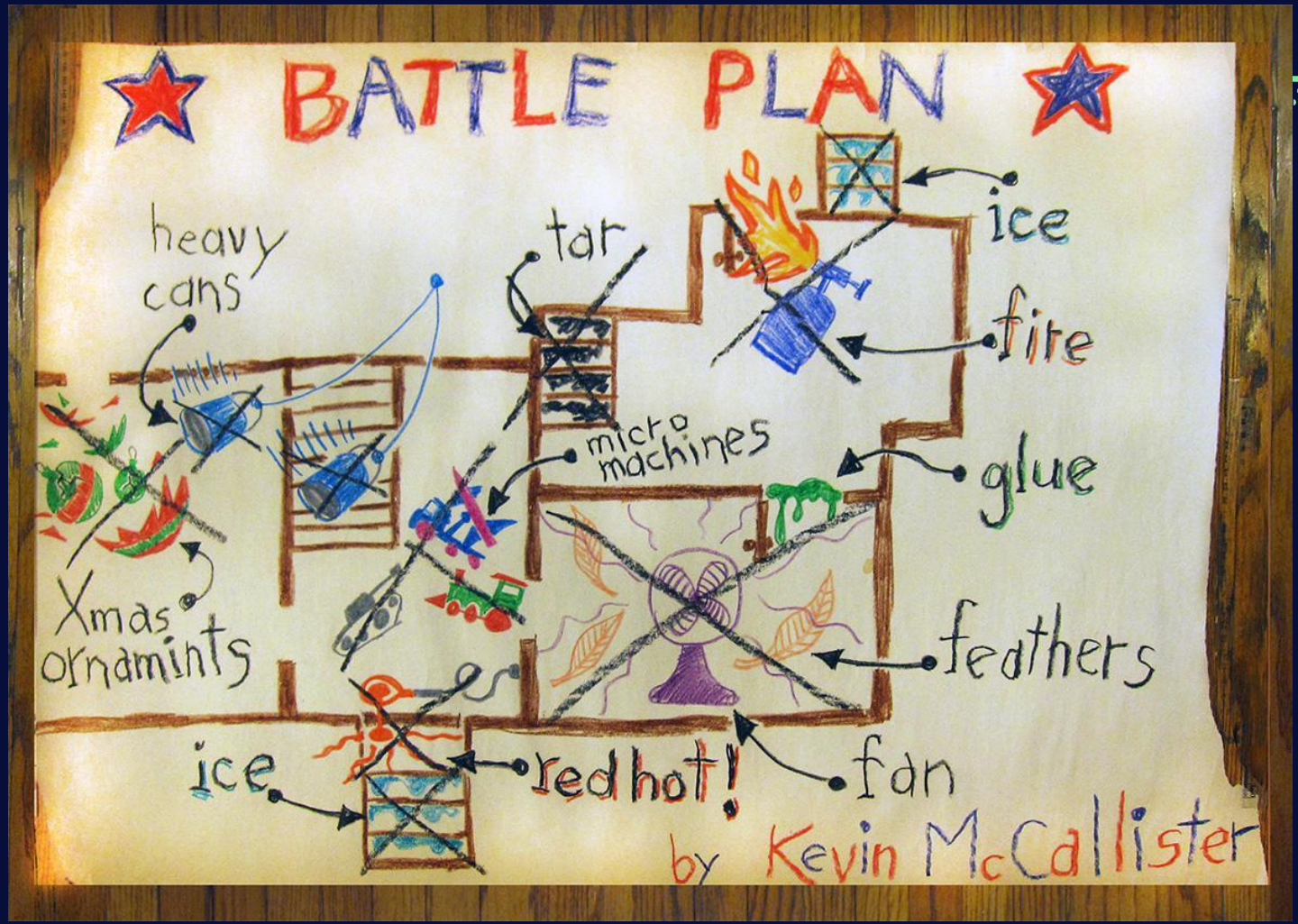
Shlomi Kushchi,

System Architect at Jit

Engineer, 8200 veteran, Cloud Enthusiast

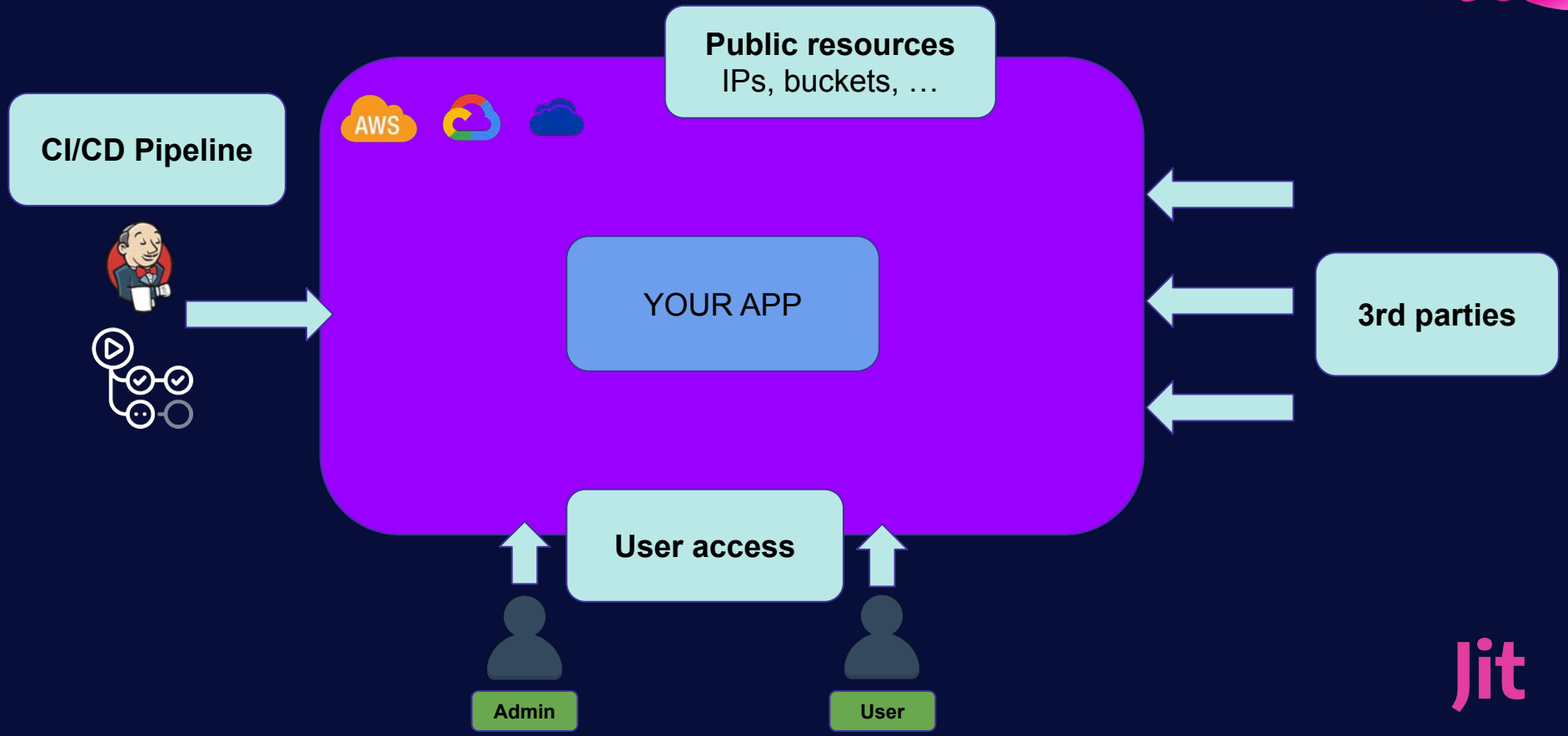
Jit

Jit


BATTLE PLAN
heavy cans
tar
ice
fire
micro machines
glue
Xmas ornaments
feathers
ice
red hot!
fan
by Kevin McCallister

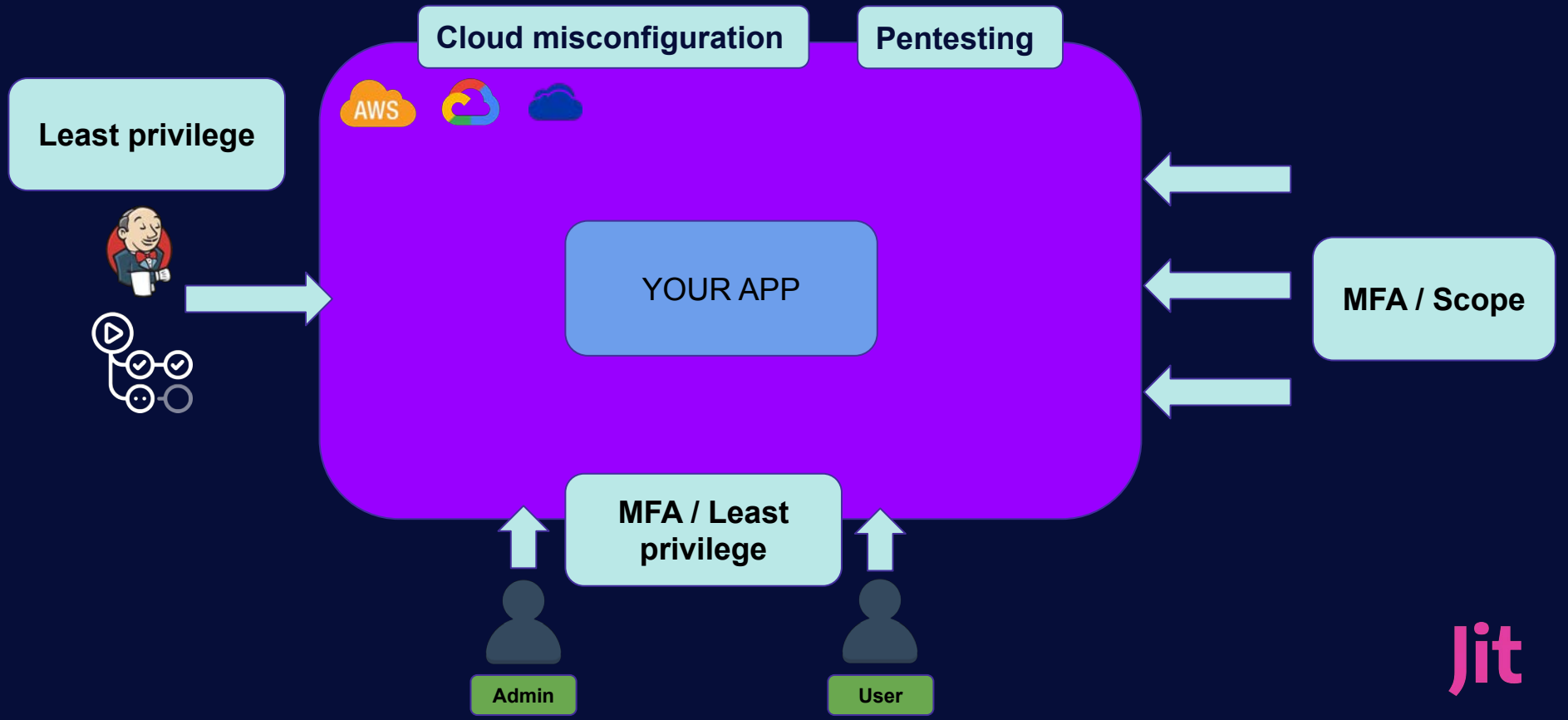
Jit

Protecting your perimeter



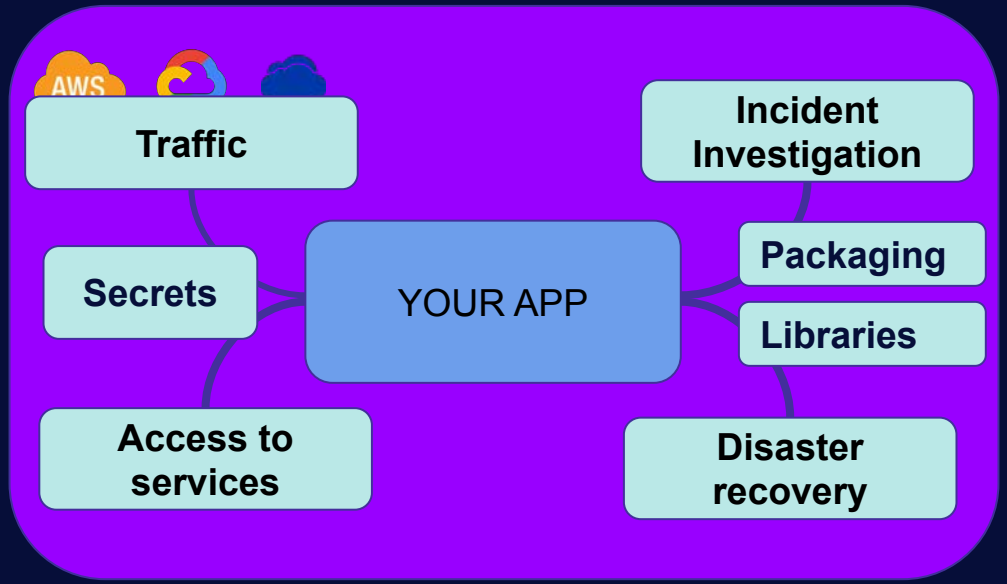
Jit

Controls to protect your perimeter



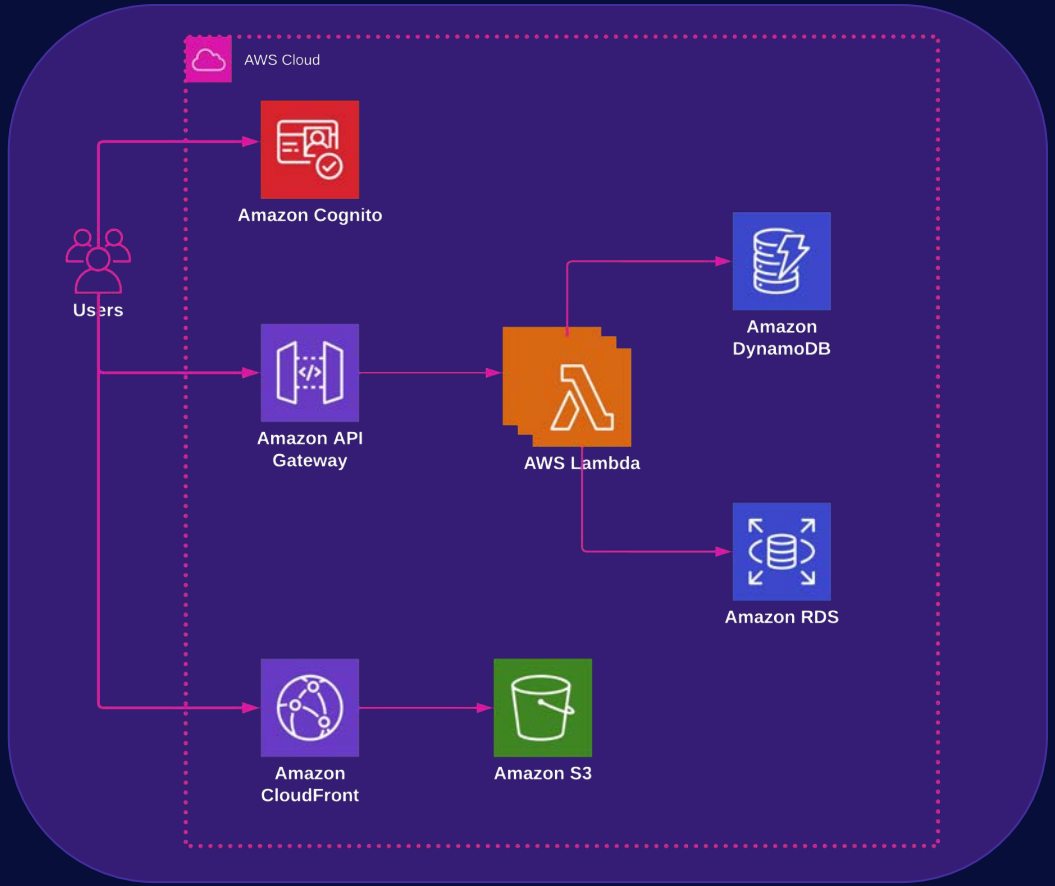
Jit

Controls to protect your app



Jit

Example of Serverless Architecture (AWS)



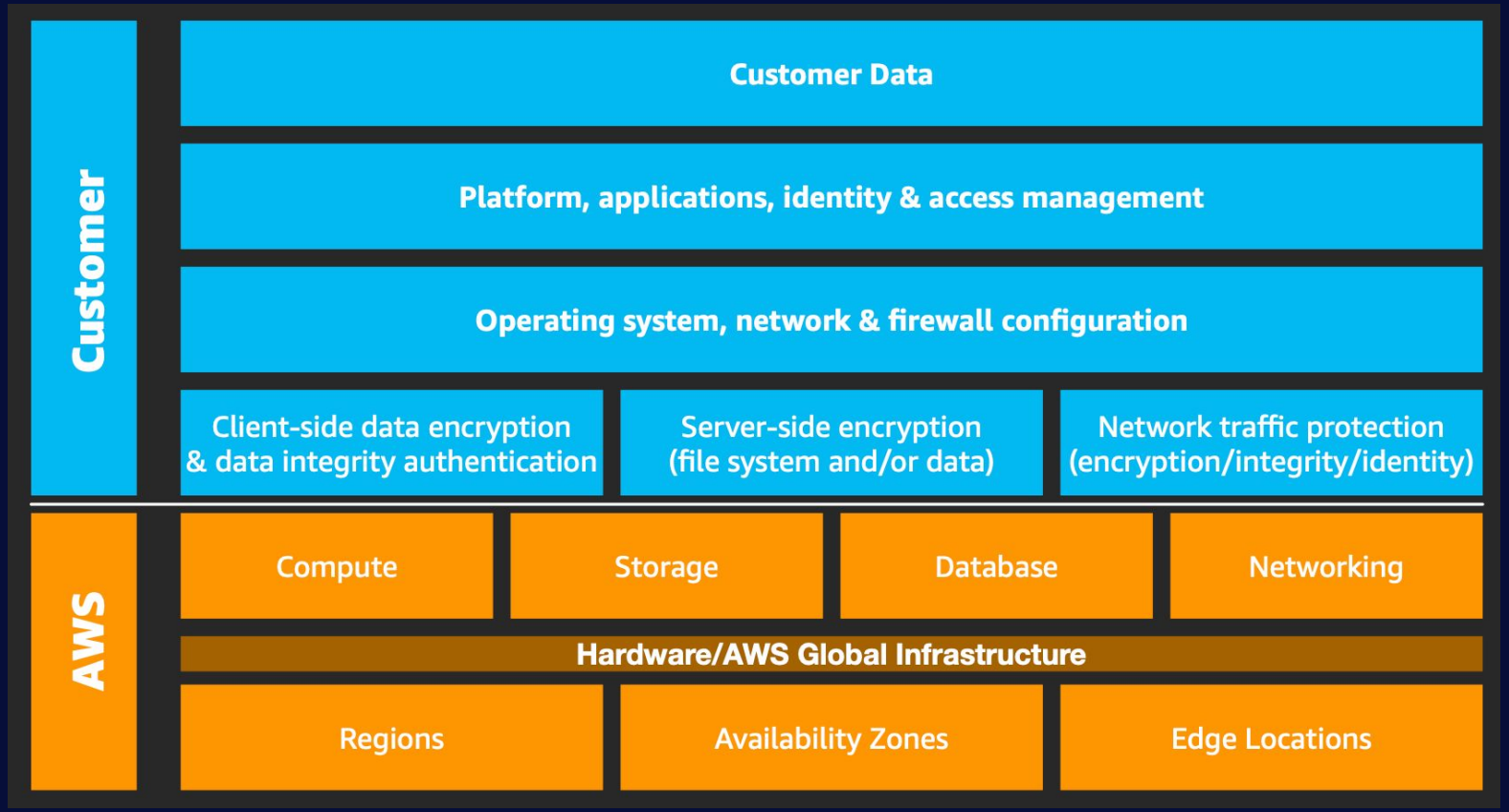
Jit

Differences between classic and serverless

- ➔ Different Shared Responsibility Model
- ➔ Serverless Functions are ephemeral
- ➔ Increased attack surface
- ➔ More fine-grained control

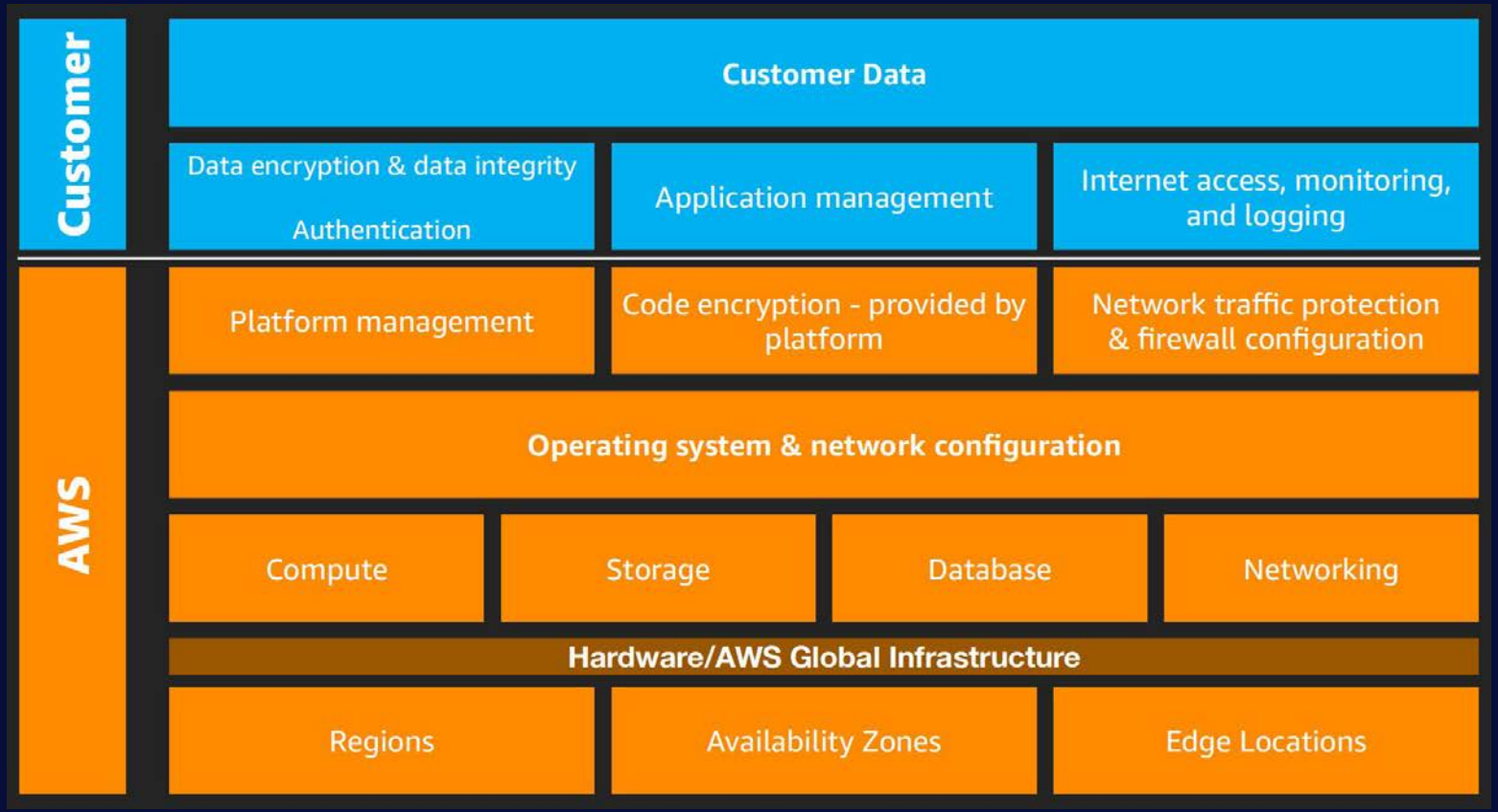
Jit

Shared Responsibility Model (classic)



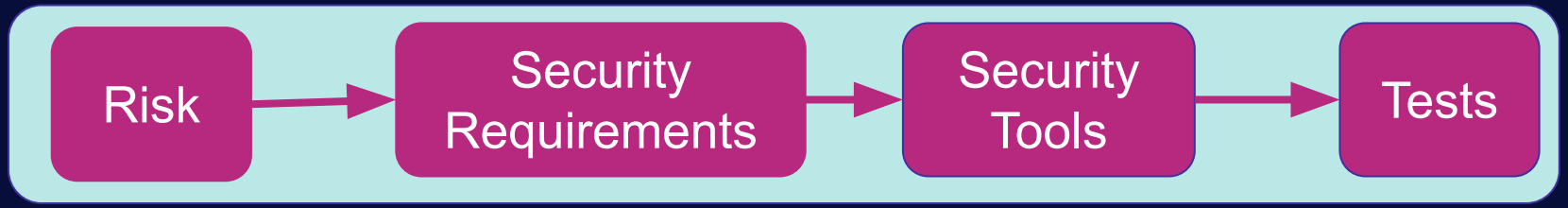
Jit

Shared Responsibility Model (serverless)



Jit

Security Plan Translation



Example: **Injection** **Validate data input** **OWASP ZAP**

Jit

What is the OWASP Foundation?

Who is the OWASP® Foundation?

The Open Web Application Security Project® (OWASP) is a nonprofit foundation that works to improve the security of software. Through community-led open-source software projects, hundreds of local chapters worldwide, tens of thousands of members, and leading educational and training conferences, the OWASP Foundation is the source for developers and technologists to secure the web.

- Tools and Resources
- Community and Networking
- Education & Training

For nearly two decades corporations, foundations, developers, and volunteers have supported the OWASP Foundation and its work. Donate, Join, or become a Corporate Member today.

Jit

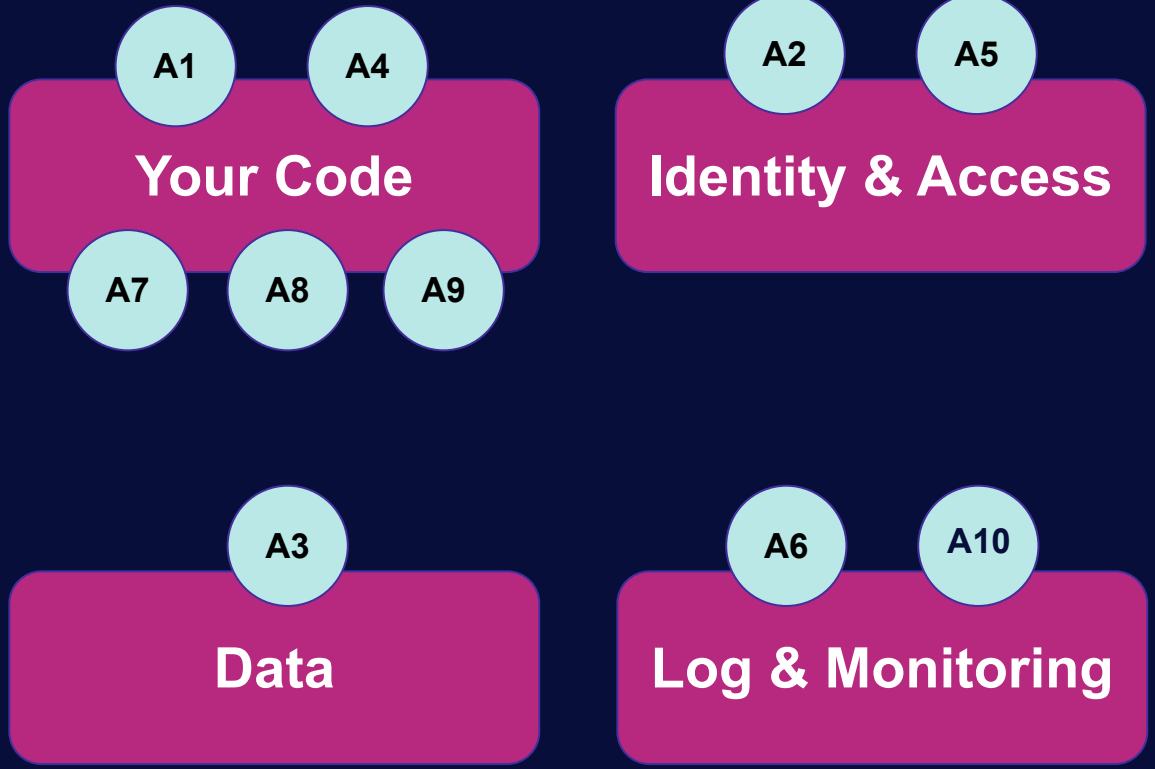
OWASP Serverless Top 10 risks

	Security Domain	Risk Description
A1:2017	Code	Injection
A2:2017	Identity & Access	Broken authentication
A3:2017	Data	Sensitive data exposure
A4:2017	Code	XML external entities (XXE)
A5:2017	Identity & Access	Broken access control
A6:2017	Logging & Monitoring	Security misconfiguration
A7:2017	Code	Cross-site scripting (XSS)
A8:2017	Code	Insecure deserialization
A9:2017	Code	Using components with known vulnerabilities
A10:2017	Logging & Monitoring	Insufficient logging & monitoring

Source: https://owasp.org/www-project-serverless-top-10/

Jit

OWASP Serverless Top 10: overview



Jit

A1:2017 – Injection

Your Code

Risk

Lack of input validation can lead to exploits like SQL injection

Security requirements

- ➔ Validate your data input
- ➔ Ensure your functions are running with least privilege
- ➔ Ensure you monitor your functions at runtime

Possible controls to remediate

- ➔ Check you are not vulnerable to traditional injection attacks
- ➔ Check your functions don’t have a wide privilege scope

Jit

A9:2017 – Using components with known vulnerabilities

Your Code

Risk

Data leakage, account compromise, ...

Security requirements

➔ Ensure you don't use dependencies with known vulnerabilities

Possible controls to remediate

➔ Vulnerable libraries scanner

Jit

A1:2017 – Broken Authentication

Identity & Access

Risk

Data leakage, break flow execution

Security requirements

- ➔ Make sure you don't have unauthorized endpoints
- ➔ Ensure you use some known IdP for identity management for user login
- ➔ Ensure your infra uses a central authentication method for inter-services
- ➔ Lambda and services should require authentication
- ➔ Verify that you don't have unmanaged public resources that do not require auth

Possible controls to remediate

- ➔ Tool to check for runtime misconfiguration

Jit

A5:2017 – Broken Access Control

Identity & Access

Risk

Data leakage from cloud storage or database

Security requirements

➔ Ensure your functions are running with least privilege

Possible controls to remediate

➔ Check for least privilege IAM roles

Jit

A3:2017 – Sensitive data exposure

Your data

Risk

Data leakage

Security requirements

- ➔ Make sure your sensitive data is not accessible through public resources
- ➔ Ensure you use encryption at rest where you store your sensitive data
- ➔ Ensure you use encryption in transit for inbound traffic

Possible controls to remediate

- ➔ Check for hard-coded secrets
- ➔ Check that traffic is encrypted at rest/in transit

Jit

A6:2017 – Security Misconfiguration

Logging & Monitoring

Risk

Information leakage, DDos / Denial of Wallet

Security requirements

➔ Ensure your functions are configured properly, i.e. max. concurrency,avg. timeouts...

Possible controls to remediate

➔ Check that the functions are configured correctly

Jit

OWASP ZAP

Web Security tool

Features

- Will discover several types of vulnerabilities, i.e. SQL injections
- Works for URLs and API endpoints using Swagger/OpenAPI
- Will only work for API Gateway endpoints

Project: https://owasp.org/www-project-zap/

Jit

OWASP dependency-check

Vulnerable libraries detection

Features

- Detect publicly disclosed vulnerabilities contained within a project's dependencies
- Uses the NVD (National Vulnerability Database)

Project: https://owasp.org/www-project-dependency-check/

Jit

Gitleaks

Secret Detection

Features

- Detects multiple types of secrets: API keys, tokens, ...
- Search in git history
- Easily integrated in CI/CD

Project: https://github.com/zricethezav/gitleaks

Jit

Prowler

Runtime infrastructure misconfiguration

Features

- ~200 checks
- Multiple output formats
- Support for AWS organization

Project: https://github.com/prowler-cloud/prowler

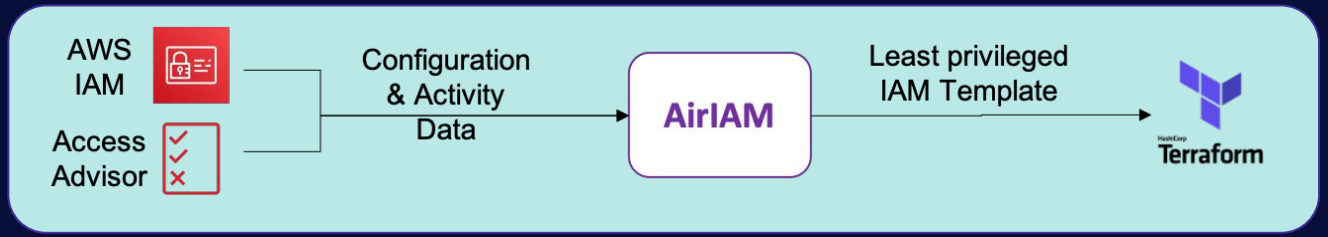
Jit

AirIAM (BridgeCrew)

Least Privilege IAM

Features

- Scans existing IAM usage patterns & detects unused IAM resources using native AWS + Access Advisor
- Migrates IAM configuration in Terraform plan



Project: https://github.com/bridgecrewio/AirIAM

Jit

A1:2017 – Injection

Your Code

Risk

Lack of input validation can lead to exploits like SQL injection

Security requirements

- ➔ Validate your data input
- ➔ Ensure your functions are running with least privilege
- ➔ Ensure you monitor your functions at runtime

Possible OSS tools to remediate

- ➔ detect OWASP ZAP
- ➔ prevent AirIAM (by Bridgecrew)

Jit

A4:2017 – XML external entities (XXE)

Your Code

Risk

Exploit XXE attacks leveraging old XML processors

Security requirements

➔ Ensure you are protected against XXE attacks

Possible OSS tools to remediate

➔ detect OWASP ZAP

➔ detect OWASP dependency-check

Jit

A7:2017 – Cross-site Scripting (XSS)

Your Code

Risk

Generate untrusted input from the backend using user data

Security requirements

➔ Ensure you are protected against XSS attacks

Possible OSS tools to remediate

➔ detect OWASP ZAP

Jit

A8:2017 – Insecure Deserialization

Your Code

Risk
Exploit library

Security requirements
➔ Ensure you are protected against insecure deserialization

Possible OSS tools to remediate
➔ detect OWASP ZAP
➔ detect OWASP dependency-check

Jit

A9:2017 – Using components with known vulnerabilities

Your Code

Risk

Data leakage, account compromise, ...

Security requirements

➔ Ensure you don't use dependencies with known vulnerabilities

Possible OSS tools to remediate

➔ detect OWASP dependency-check

Jit

A1:2017 – Broken Authentication

Identity & Access

Risk

Data leakage, break flow execution

Security requirements

- ➔ Make sure you don't have unauthorized endpoints
- ➔ Ensure you use some known IdP for identity management for user login
- ➔ Ensure your infra uses a central authentication method for inter-services
- ➔ Lambda and services should require authentication
- ➔ Verify that you don't have unmanaged public resources that do not require auth

Possible OSS tools to remediate

- ➔ detect OWASP ZAP
- ➔ detect Prowler

Jit

A5:2017 – Broken Access Control

Identity & Access

Risk

Data leakage from cloud storage or database

Security requirements

➔ Ensure your functions are running with least privilege

Possible OSS tools to remediate

➔ prevent AirIAM

Jit

A3:2017 – Sensitive data exposure

Your data

Risk

Data leakage

Security requirements

- ➔ Make sure your sensitive data is not accessible through public resources
- ➔ Ensure you use encryption at rest where you store your sensitive data
- ➔ Ensure you use encryption in transit for inbound traffic

Possible controls to remediate

- ➔ detect Gitleaks
- ➔ detect Prowler
- ➔ detect OWASP ZAP

Jit

A6:2017 – Security Misconfiguration

Logging & Monitoring

Risk

Information leakage, DDos / Denial of Wallet

Security requirements

➔ Ensure your functions are configured properly, i.e. max. concurrency,avg. timeouts...

Possible OSS tools to remediate

➔ detect Prowler

➔ detect Custom tool

Jit

A10:2017 – Insufficient Logs and Monitoring

Logging & Monitoring

Risk

Letting attackers go unnoticed

Security requirements

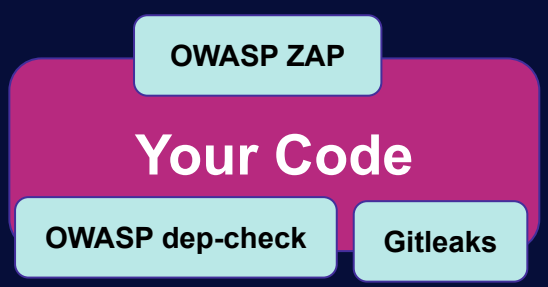
- ➔ 1. Ensure you have enough logging for your infrastructure services to investigate possible security issues
- ➔ 2. Ensure you have enough logging in your application to investigate possible security issues and that you log admin events in your application (login, privilege grant, invite user, config change, delete data)

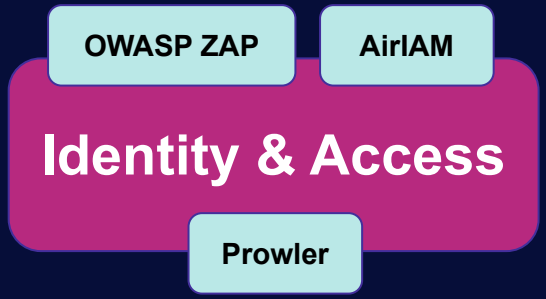
Possible OSS tools to remediate

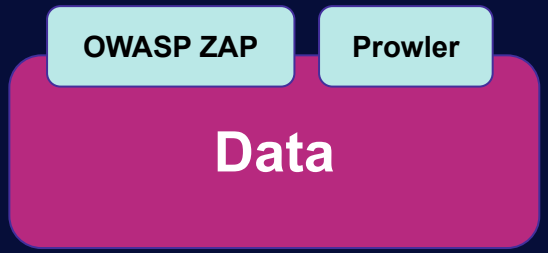
- ➔ detect Prowler

Jit

OWASP Serverless Top 10: OSS tools



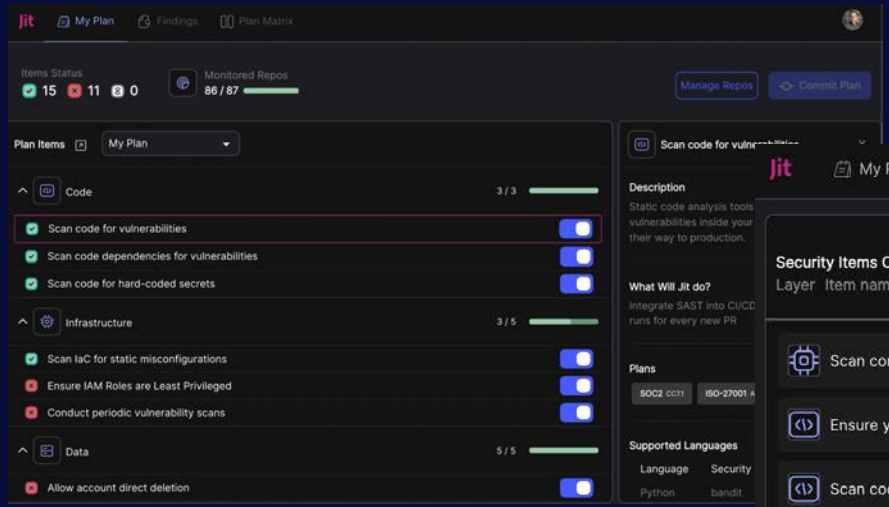




Jit

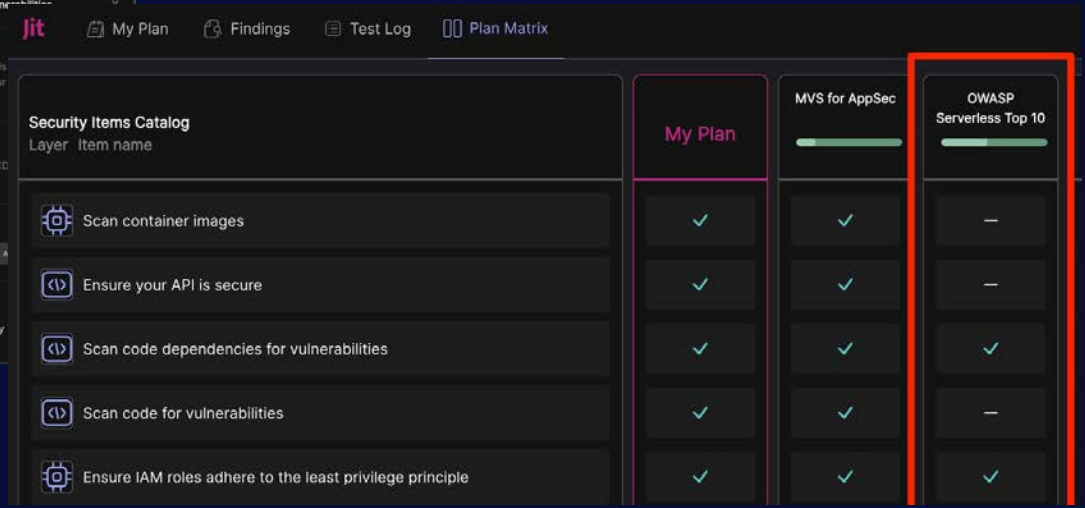
Taking it to the next level: Jit

Automating security plans using OSS security tools orchestration



Automating OWASP plans

Customized plan



Jit

Taking it to the next level: Jit

Plan codification

```yaml
name: OWASP Serverless Top 10
description: Minimum Viable Security plan for your serverless applications. It provides an automatic and continuous
level: beginner
author: OWASP
version: 0.1
is_enabled: true
references:
  - https://owasp.org/www-project-serverless-top-10/
tags:
  - serverless
  - owasp
owners:
  default:
# ------------ List of plan items ----------
items:
  - name: Run a Web Application Scanner
    uses: jitsecurity-controls/jit-plans/items/runtime/item-web-app-scanner.yml@latest

  - name: Ensure IAM Roles are Least Privileged
    uses: jitsecurity-controls/jit-plans/items/infrastructure/item-least-privileged-iam.yml@latest

  - name: Check for runtime infrastructure misconfiguration
    uses: jitsecurity-controls/jit-plans/items/infrastructure/item-runtime-misconfiguration-detection.yml@latest

  - name: Check for vulnerable dependencies
    uses: jitsecurity-controls/jit-plans/items/code/item-dependency-check.yml@latest

  - name: Ensure your serverless functions are properly configured
    uses: jitsecurity-controls/jit-plans/items/infrastructure/item-check-functions-config.yml@latest

  - name: Ensure your logs are shipped to a central system
    uses: jitsecurity-controls/jit-plans/items/operations/item-central-log-shipping.yml@latest

  - name: Add logging capabilities for your app
    uses: jitsecurity-controls/jit-plans/items/operations/item-logging.yml@latest
```

Jit

Your next step on the security journey

Jit

Jit

Thank you

Intrigued? Try our **app** at **jit.io**

Inspired? Join us! **We are hiring!**

Questions? Contact me at **shlomi@jit.io**