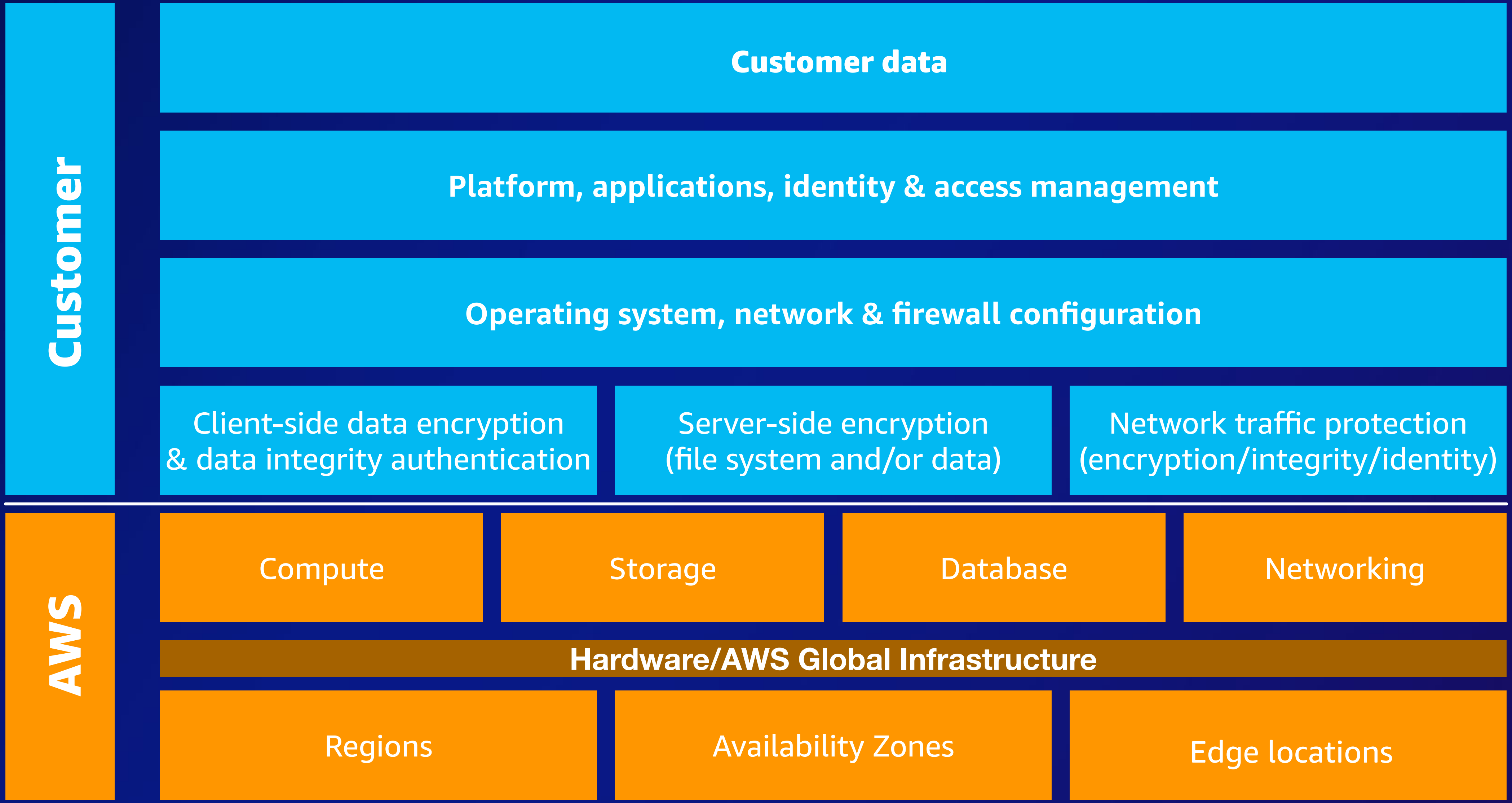
SEC301

Defending your Serverless App against the OWASP top 10

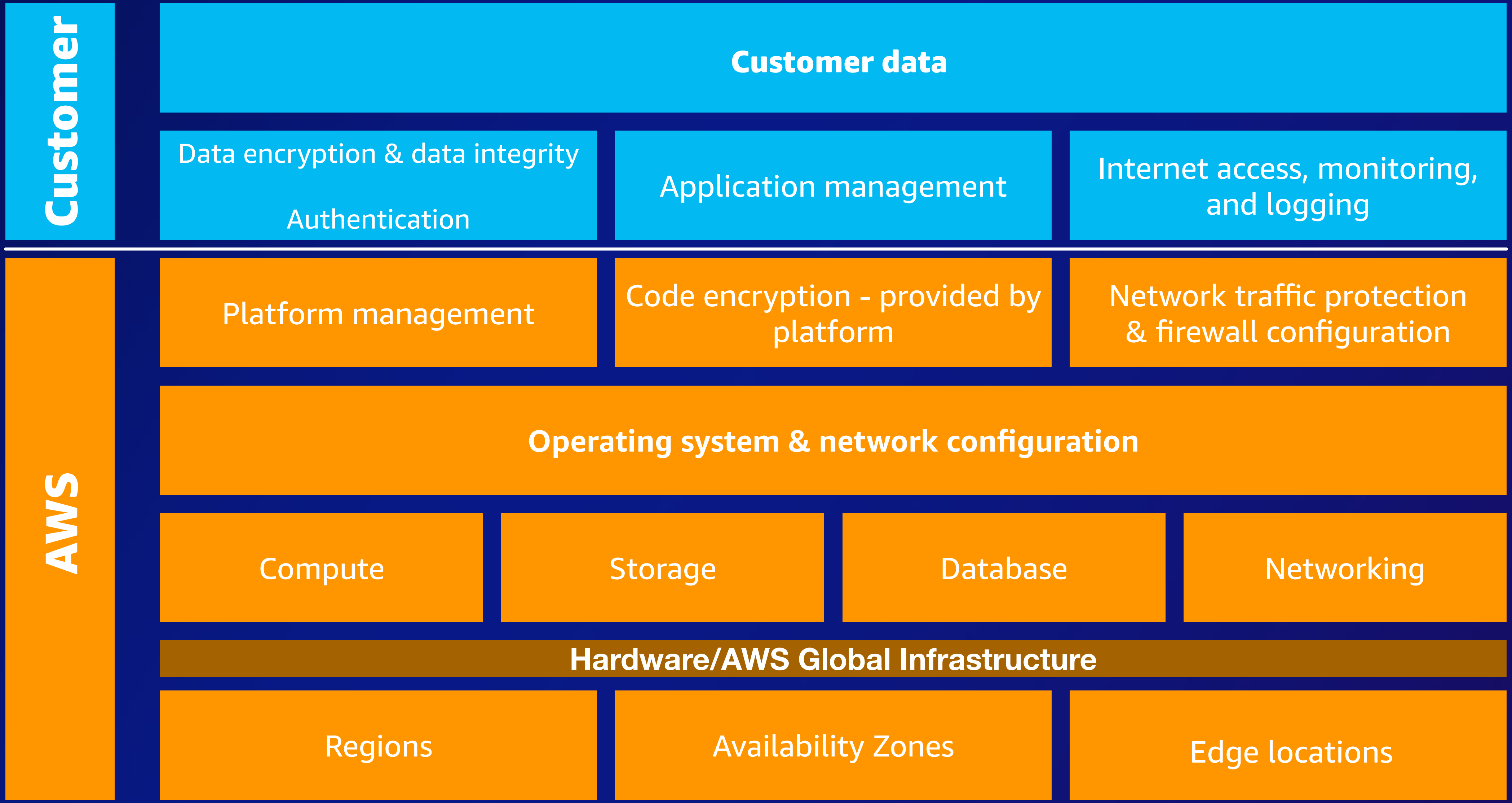
Boaz Ziniman
Principal Developer Advocate
Amazon Web Services

David Melamed
CTO and Co-Founder
Jit

Shared responsibility model: “Serverful"

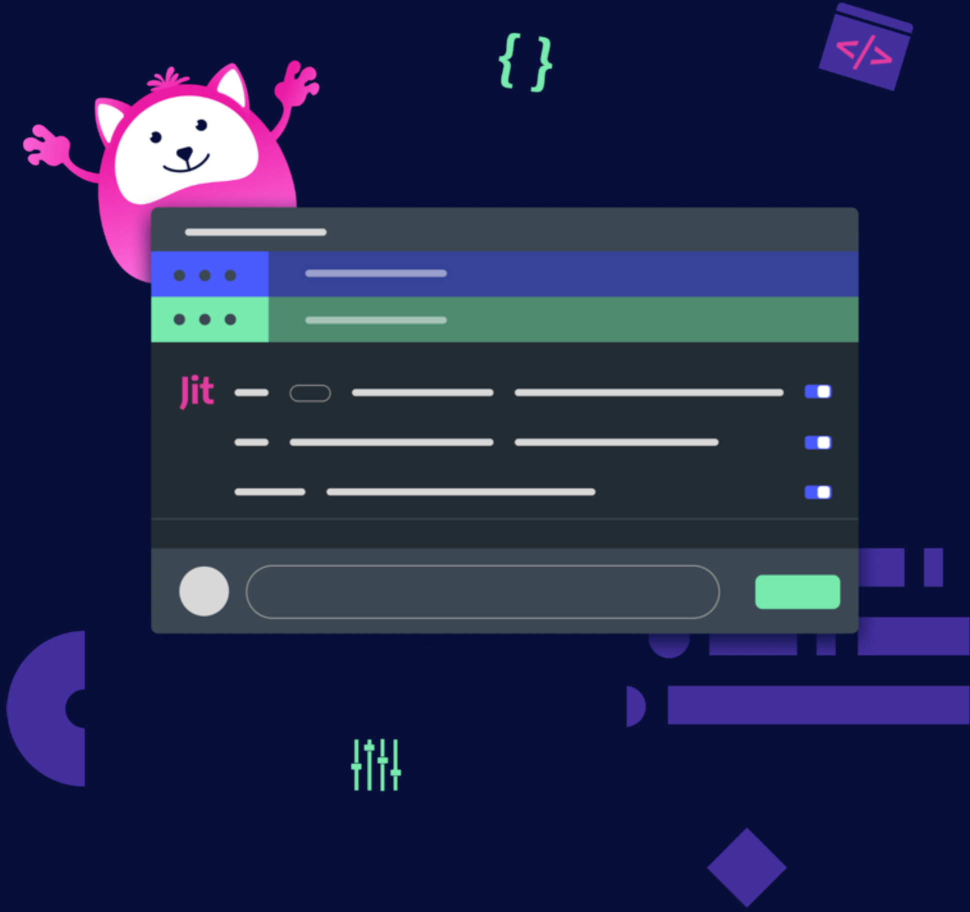

Customer
Customer data
Platform, applications, identity & access management
Operating system, network & firewall configuration
Client-side data encryption & data integrity authentication
Server-side encryption (file system and/or data)
Network traffic protection (encryption/integrity/identity)
AWS
Compute
Storage
Database
Networking
Hardware/AWS Global Infrastructure
Regions
Availability Zones
Edge locations

Shared responsibility model: Serverless


Customer
Customer data
Data encryption & data integrity
Authentication
Application management
Internet access, monitoring, and logging
AWS
Platform management
Code encryption - provided by platform
Network traffic protection & firewall configuration
Operating system & network configuration
Compute
Storage
Database
Networking
Hardware/AWS Global Infrastructure
Regions
Availability Zones
Edge locations

David Melamed

CTO & Co-Founder



About me

David Melamed
CTO & Co-Founder

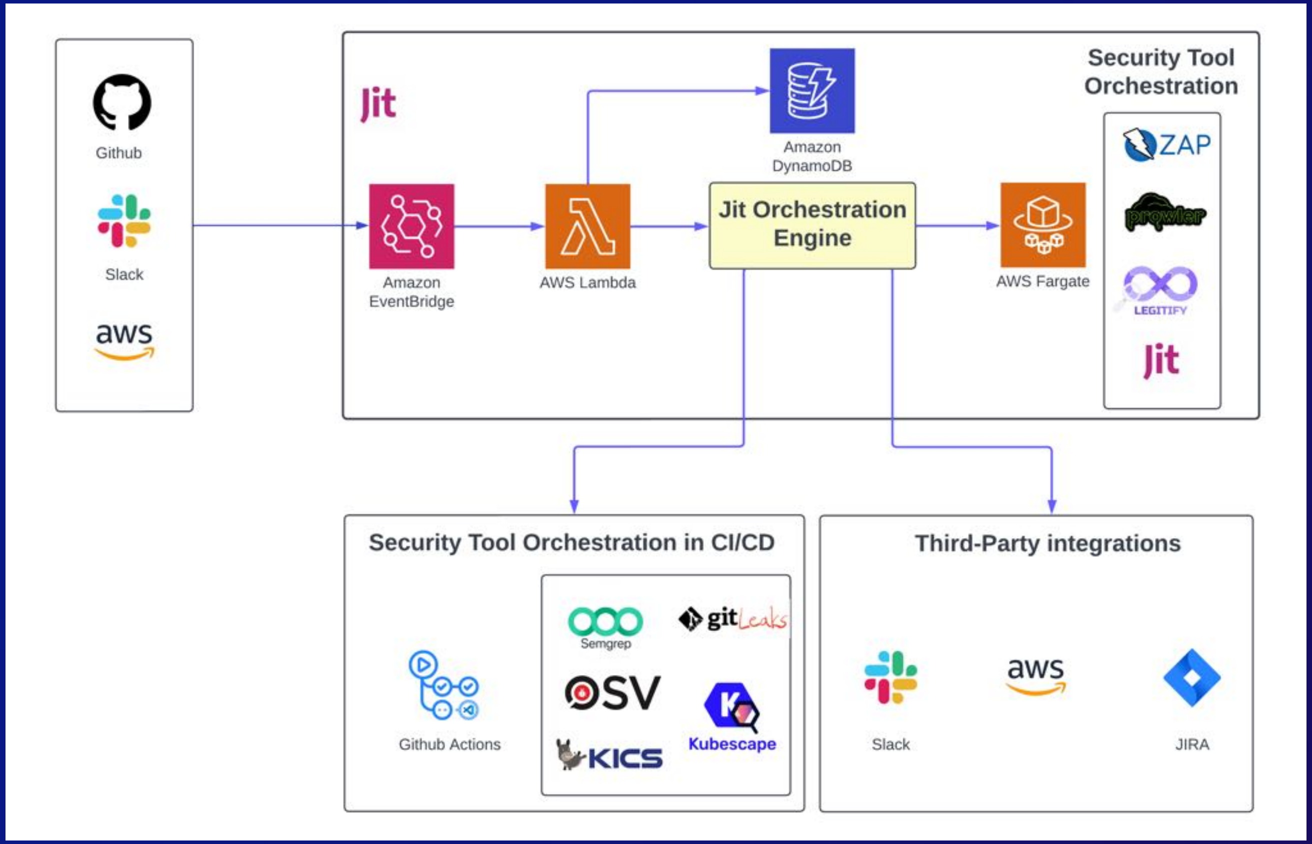
Tech & Automation addict

About Jit

Accelerate
Product Security Programs

Built a DevSecOps Platform packaging OSS tools across AppSec, Infra, CI/CD, Runtime

Self-service, Easy Onboarding & Great Dev Experience



100% built on Serverless

OWASP Top Ten

THE MOST CRITICAL SECURITY RISKS TO WEB APPLICATIONS.

“Globally recognized by developers as the first step towards more secure coding“

https://www.owasp.org

OWASP top 10 web application security risks

2021

1 Broken access control

2 Cryptographic failures

3 Injection

4 Insecure design

5 Security misconfiguration

6 Vulnerable and outdated components

7 Identification and authentication failures

8 Software and data integrity failures

9 Security logging and monitoring failures

10 Server-side request forgery

https://www.owasp.org

OWASP top 10 mapped to security domains

Identity and access	1	Broken access control
	7	Identification and authentication failures
Code	3	Injection
	4	Insecure design
	6	Vulnerable and outdated components
	8	Software and data integrity failures
	10	Server-side request forgery
Data	2	Cryptographic failures
	8	Software and data integrity failures
Infrastructure	4	Insecure design
	5	Security misconfiguration
	6	Vulnerable and outdated components
	10	Server-side request forgery
Logging and monitoring	5	Security misconfiguration
	9	Security logging and monitoring failures

OWASP top 10 web application security risks

2021

1. Broken access control
2. Cryptographic failures
3. Injection
4. Insecure design
5. Security misconfiguration
6. Vulnerable and outdated components
7. Identification and authentication failures
8. Software and data integrity failures
9. Security logging and monitoring failures
10. Server-side request forgery

https://www.owasp.org

OWASP top 10 web application security risks

1. Broken access control
2. Cryptographic failures
3. Injection
4. Insecure design
5. Security misconfiguration
6. Vulnerable and outdated components
7. Identification and authentication failures
8. Software and data integrity failures
9. Security logging and monitoring failures
10. Server-side request forgery

https://www.owasp.org

Broken Access Control

Access to unauthorized resources or data within the system

Broken Access Control

RISKS

Modification or deletion of data

Execution of unauthorized functions

Data leakage from cloud storage or database

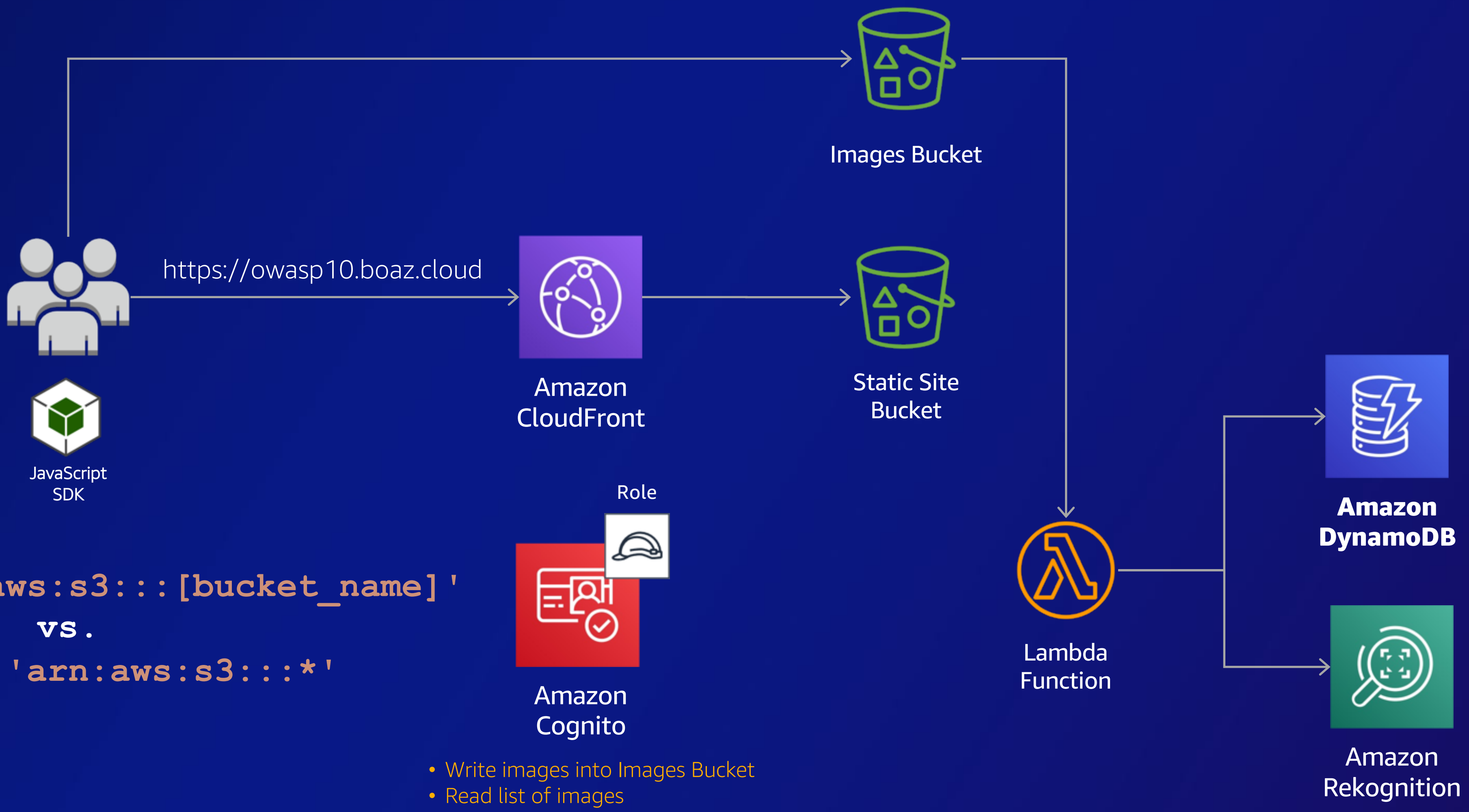
Security requirements:

- Strong authentication
- Ensure your functions are running with least privilege
- Regular access logs audit

Demo

https://owasp10.boaz.cloud

Serverless Website – owasp10.boaz.cloud



- Write images into Images Bucket
- Read list of images

```
Resource: 'arn:aws:s3:::[bucket_name]'
                  vs.
      Resource: 'arn:aws:s3:::*'
```

Broken Access Control

DETECTION & REMEDIATION

- AWS IAM Access Analyzer
- AWS GuardDuty for AWS Lambda (new lambda support!)
- OSS tools
  - Pmapper
  - CloudTracker
  - Police Sentry
  - Repokid

Event Injection

Injection of input data to lead a target system into performing unintended actions

Event Injection

RISKS

Resource exhaustion

Privilege escalation

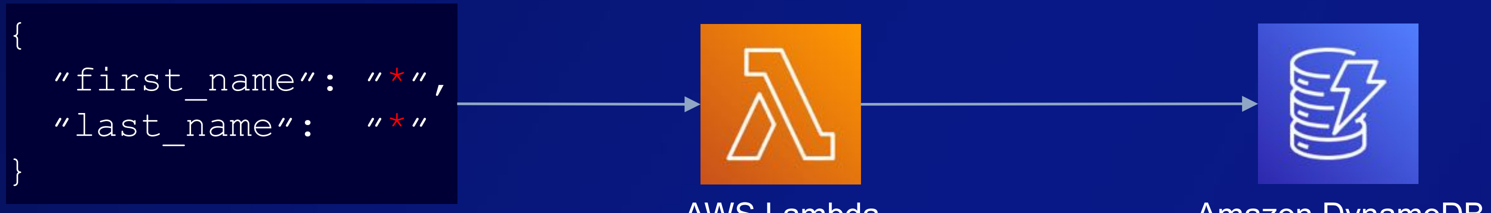
Execution of unauthorized code
(to steal data or take control)

Data leakage

Security requirements:

- Validate & sanitize data input in your function handler
- Ensure your functions are running with least privilege
- Monitor functions at runtime

Example: DynamoDB noSQL injection


```
{
  "first_name": "*",
  "last_name":  "*"
}
```
Malicious user input

Lambda function scaning the table to retrieve some records

```
...
fname = input['first_name']
lname = input['last_name']
client.scan(TableName = 'dynamo-user', Select = 'ALL_ATTRIBUTES',
        ScanFilter = {'first_name': {"AttributeValueList": [{"S": fname}],
        "ComparisonOperator": "GT"},
        'last_name': {"AttributeValueList": [{"S": lname}],
        "ComparisonOperator": "GT"}})
...
```

Event Injection

DETECTION & REMEDIATION

- AWS IAM Access Analyzer
- AWS CloudWatch Logs
- AWS CloudTrail
- If your function is connected to an API Gateway, use DAST tools (i.e. OWASP ZAP)

Security Misconfiguration

Exposing sensitive data through misconfigured services

Security Misconfiguration

RISKS

Unauthorized access

Denial-of-service attacks & Denial of Wallet

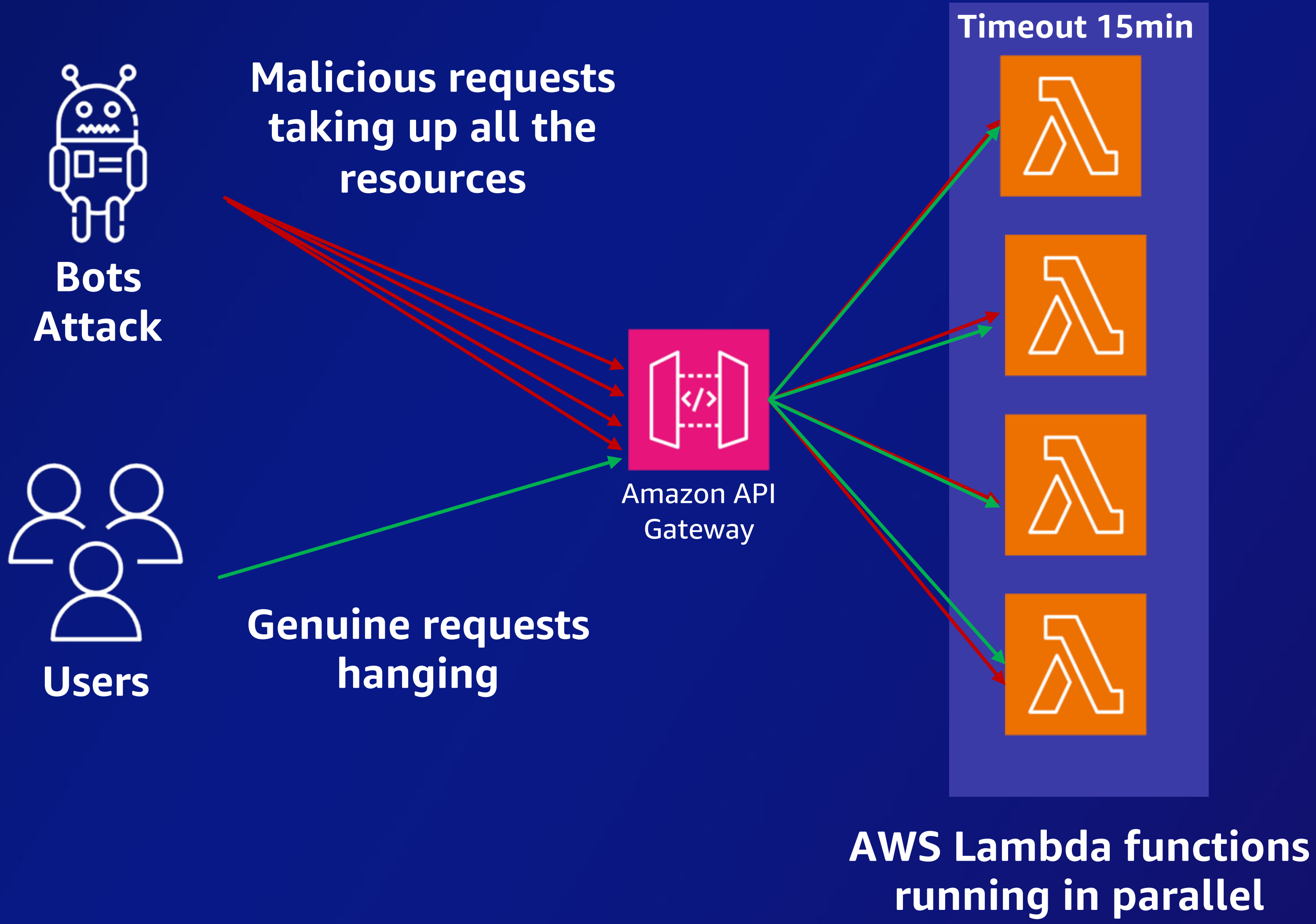
Malware & ransomware attacks

Data leakage

Security requirements:

- Proper functions configuration:
  - Max concurrency
  - Timeout
  - IAM Roles

Example: Denial of Service


Bots Attack
Malicious requests taking up all the resources
Users
Genuine requests hanging
Amazon API Gateway
Timeout 15min
AWS Lambda functions running in parallel

Demo

Security Misconfiguration

DETECTION & REMEDIATION

- Amazon Inspector
- AWS GuardDuty for AWS Lambda (new lambda support!)
- AWS Security Hub
- Logging (Amazon CloudWatch) & Tracing (AWS X-Ray)
- OSS tools: KICS (IaC), Prowler

Vulnerable and outdated components

Allowing malicious code to sneak into your environment

Vulnerable and outdated components

RISKS

Denial-of-service attacks

Malware & ransomware attacks

Data breaches

Security requirements:

- Don’t use dependencies with known vulnerabilities

Vulnerable and outdated components

DETECTION & REMEDIATION

- Some OSS SCA tools
  - OSV-scanner
  - npm audit
  - Nancy
  - OWASP dependency-check

Security logging and monitoring failures

Detecting anomalous behavior

Security logging and monitoring failures

RISKS

Undetected unauthorized access

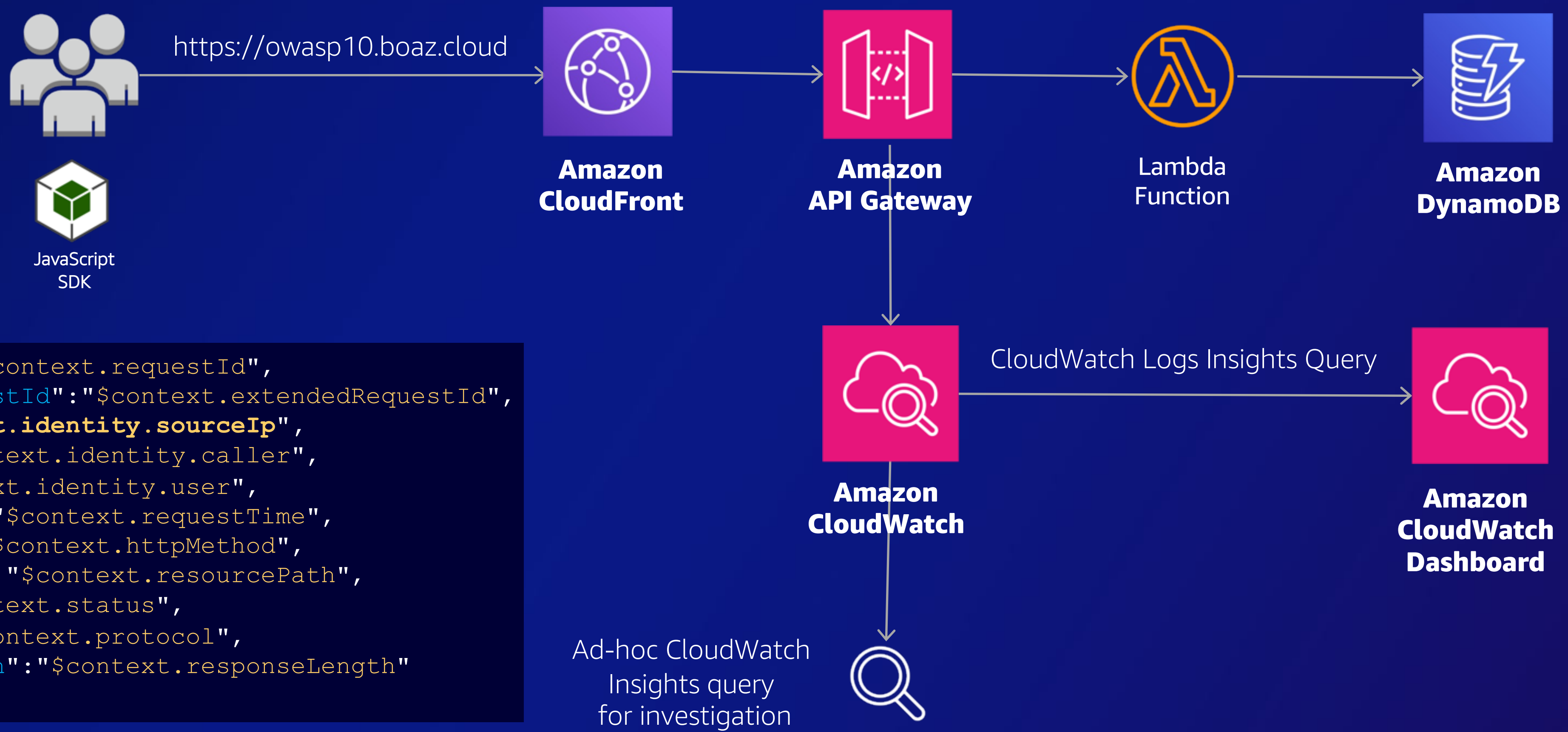
Delayed detection and response

Data loss

Security requirements:

- Ensure you have enough & effective logging for your app services, i.e. all admin events
- Ensure you have logs for infrastructure to investigate issues
- Having a basic incident response process

Example of attacker detection using log insights


https://owasp10.boaz.cloud
JavaScript SDK
Amazon CloudFront
Amazon API Gateway
Lambda Function
Amazon DynamoDB
Amazon CloudWatch
CloudWatch Logs Insights Query
Amazon CloudWatch Dashboard
Ad-hoc CloudWatch Insights query for investigation

```
{  "requestId":"$context.requestId",
   "extendedRequestId":"$context.extendedRequestId",
   "ip": "$context.identity.sourceIp",
   "caller":"$context.identity.caller",
   "user":"$context.identity.user",
   "requestTime":"$context.requestTime",
   "httpMethod":"$context.httpMethod",
   "resourcePath":"$context.resourcePath",
   "status":"$context.status",
   "protocol":"$context.protocol",
   "responseLength":"$context.responseLength"
}
```

Demo

Security logging and monitoring failures

DETECTION & REMEDIATION

Application

- Add application logging for system & admin events

Infrastructure

- Amazon Detective
- Amazon CloudWatch Logs
- AWS CloudTrail

How to secure your Serverless App

From theory to practice

How to secure your serverless application

Start with a security framework with prescriptive controls

- OWASP Serverless Top 10
- AWS Startup Security Baseline (SSB)
- CIS AWS Foundations Benchmark (orchestrated by SecurityHub)
- AWS Well-Architected Framework

Continuously monitor your application & environment

Jit: Productizing Security Engineering

Library of plans

Minimum Viable | OWASP Top 10 | AWS SSB | AWS FTR

Security plans as-code

Orchestrate all tools

Pre-package the best controls (OSS, cloud-native, commercial*)

Runs in the user GitHub env

AppSec

- Static Code Analysis (**SAST**) **Semgrep, GoSec**
- Dependency check (**SCA**) **npm-audit**, **Nancy**, **OSV-scanner**
- **Secrets** detection **GitLeaks**
- **Dockerfile** scanning **Trivy**

CI/CD Security

- GitHub security **Legitify**
- GitHub **Branch Protection** **Jit-custom**
- GitHub 2FA **Jit-custom**

Cloud Security

- Cloud IaC security **KICS**
- Multi-cloud runtime security **Prowler**
- K8s IaC Security **Kubescape**
- AWS 2FA **Jit-custom**

Runtime

- Web App Scanning **ZAP**
- API security **ZAP**

Want to know more?

or visit us: www.jit.io

Thank you!

Please complete the session survey in the mobile app

Boaz Ziniman
Principal Developer Advocate
Amazon Web Services

@ziniman
bziniman

David Melamed
CTO and Co-Founder
Jit

@dvdmelamed
mlmd