# AWS re:Inforce

JUNE 10 - 12, 2024 | PHILADELPHIA, PA

NIS305

# Secure your APIs the Well-Architected way from foundation to perimeter


**Jerry Chen**
(he/him/his)
Sr. Solutions Architect
Cloud Optimization Success
Amazon Web Services

**Frank Phillis**
(he/him/his)
Sr Solutions Architect
Security
Amazon Web Services

**Syed Shareef**
(he/him/his)
Sr. Solutions Architect
Security
Amazon Web Services

**Jorge L Gomez**
(he/him/his)
Staff Cloud Security Engineer
Twilio

# What to expect from this session

01 Common API security challenges

02 Well-Architected (WA) Framework

03 Mitigating common API security challenges

04 Twilio WAF as a service (WaaS) solution

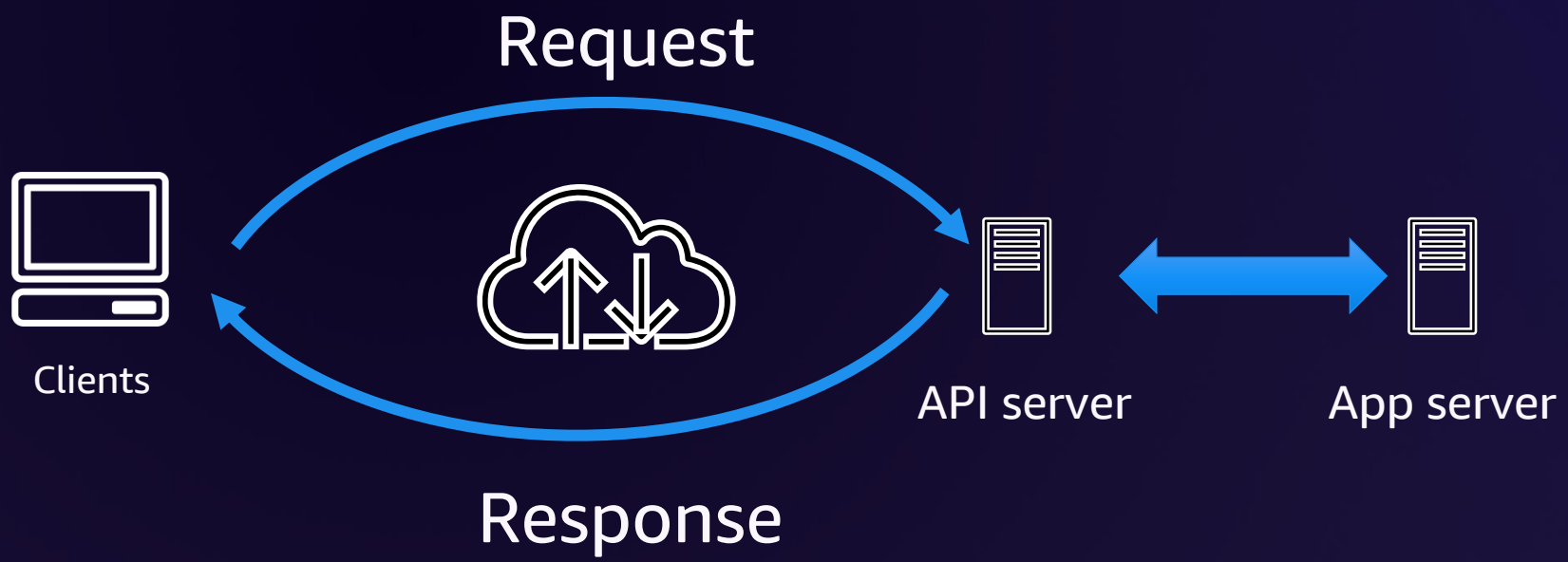
05 Call to action

# Common API security challenges

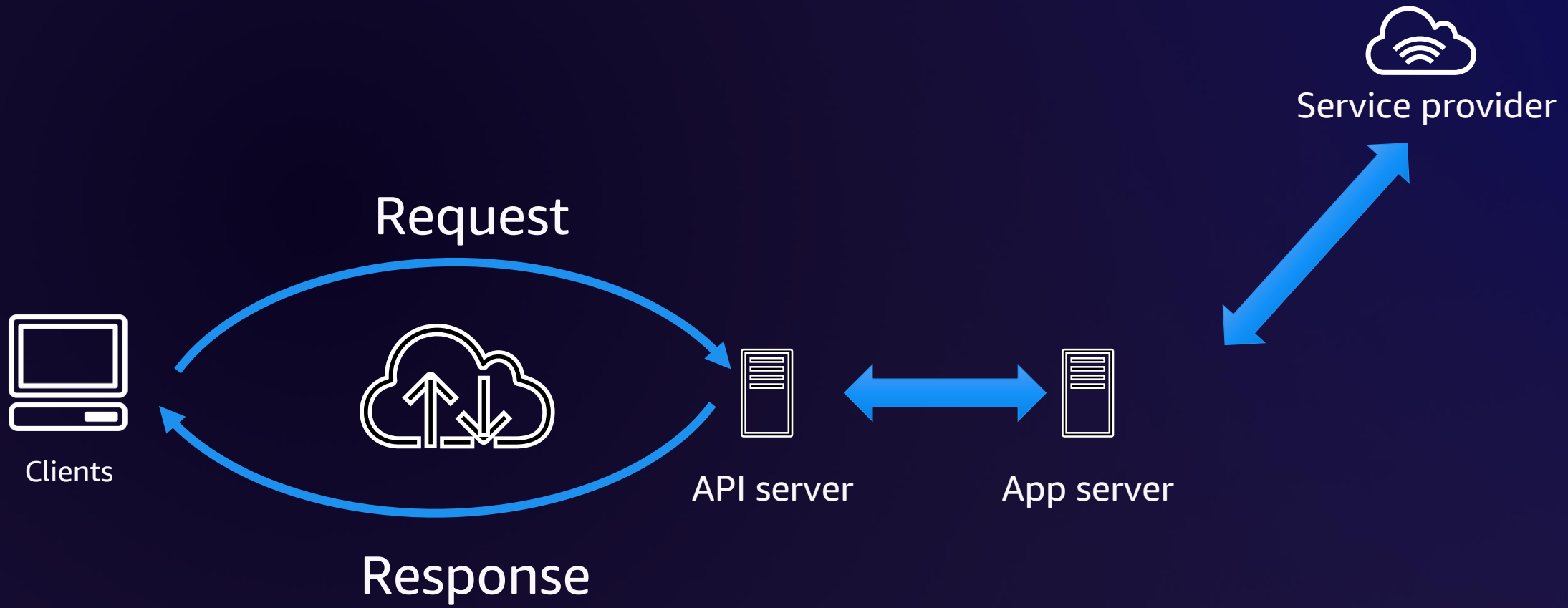
# Common API security challenges

# Common API security challenges

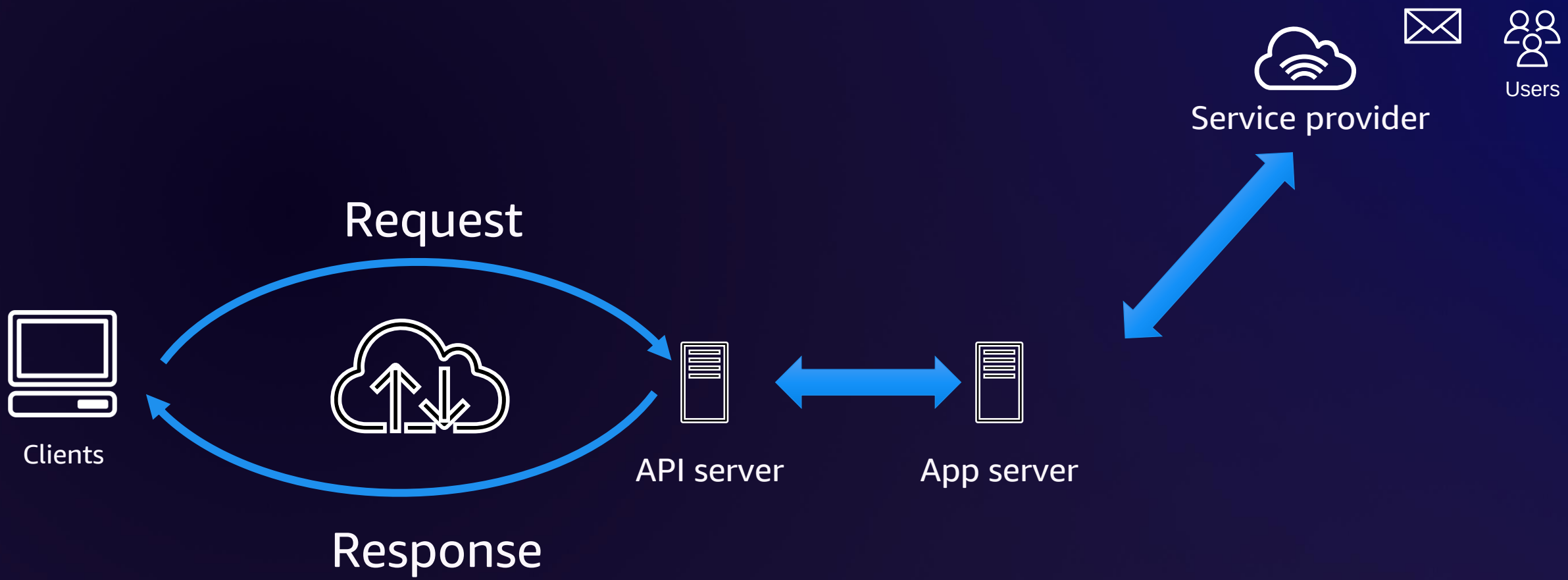
# Common API security challenges



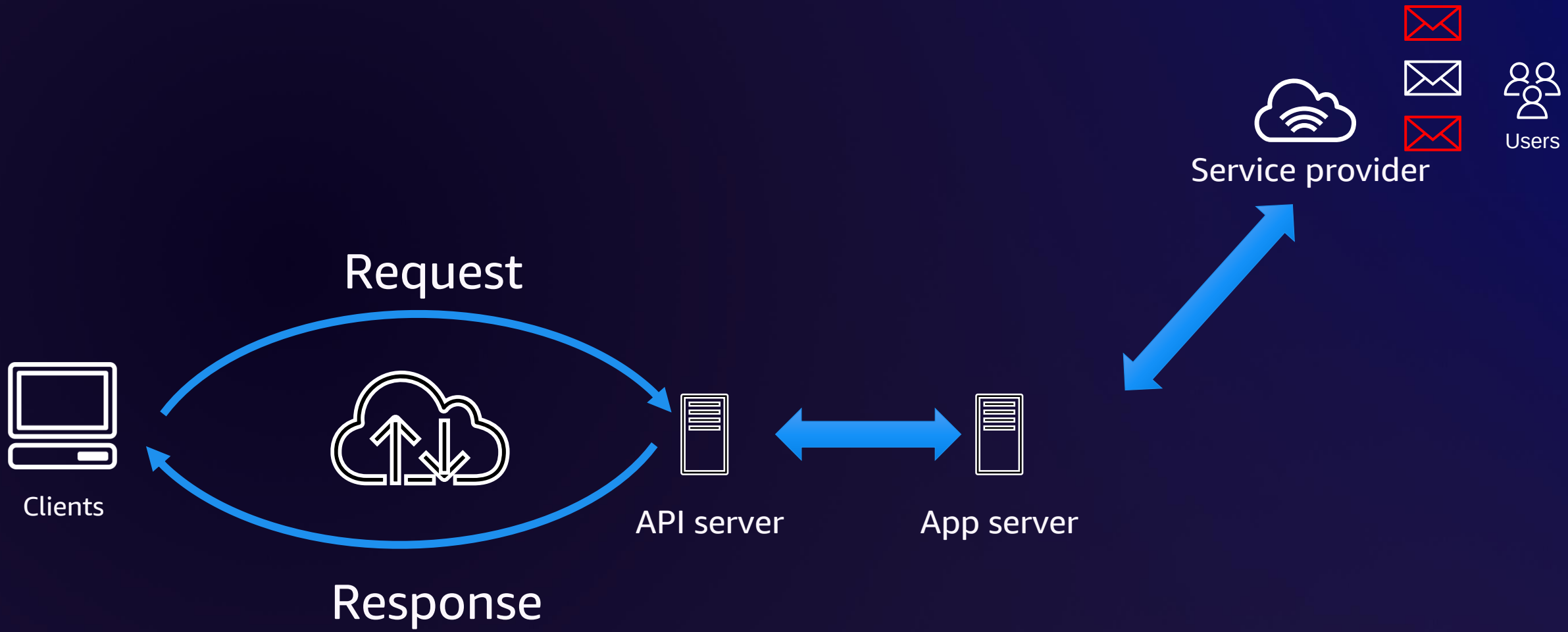
# Common API security challenges



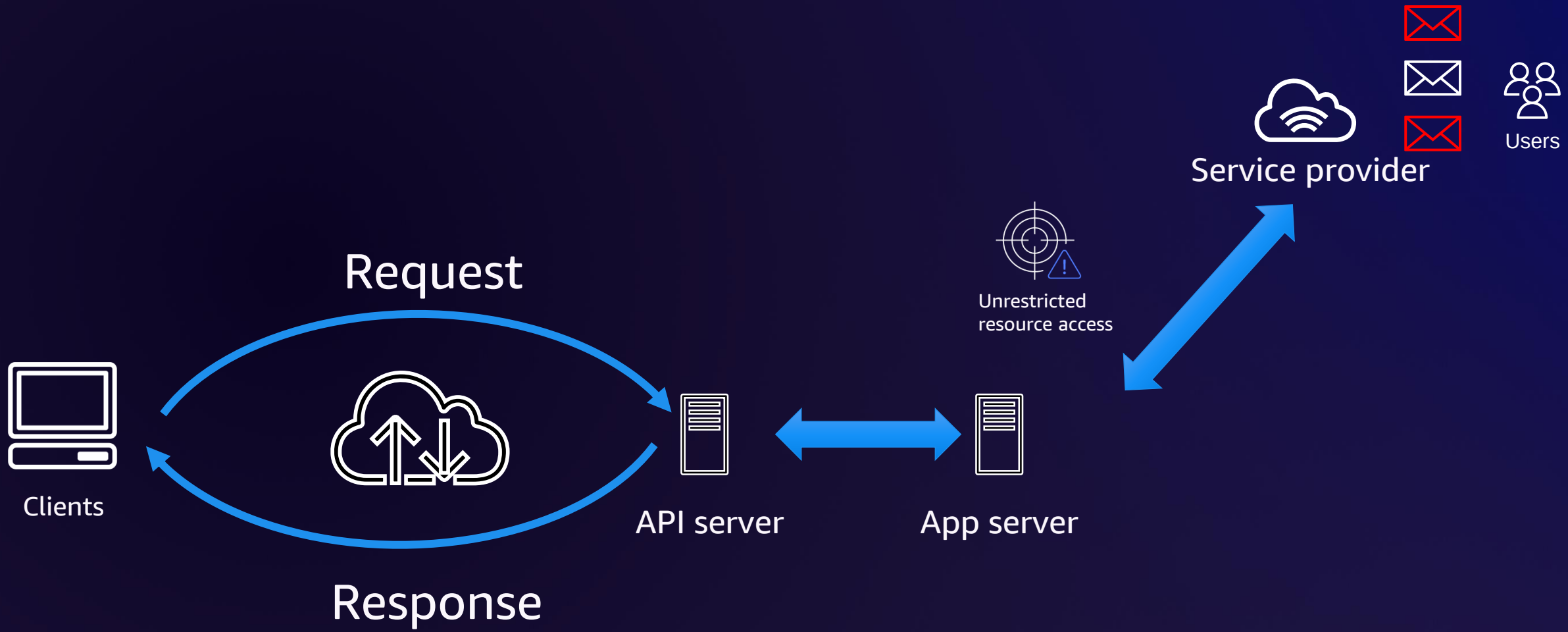
# Common API security challenges



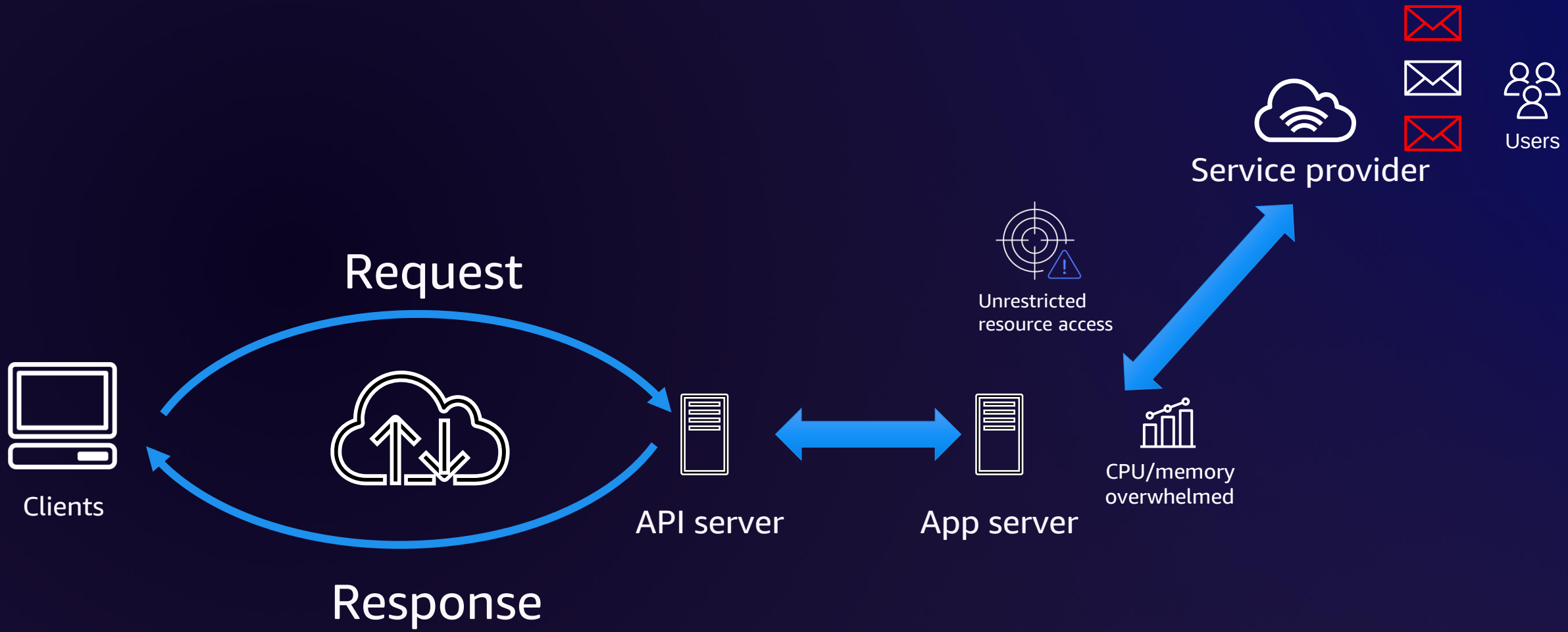
# Common API security challenges



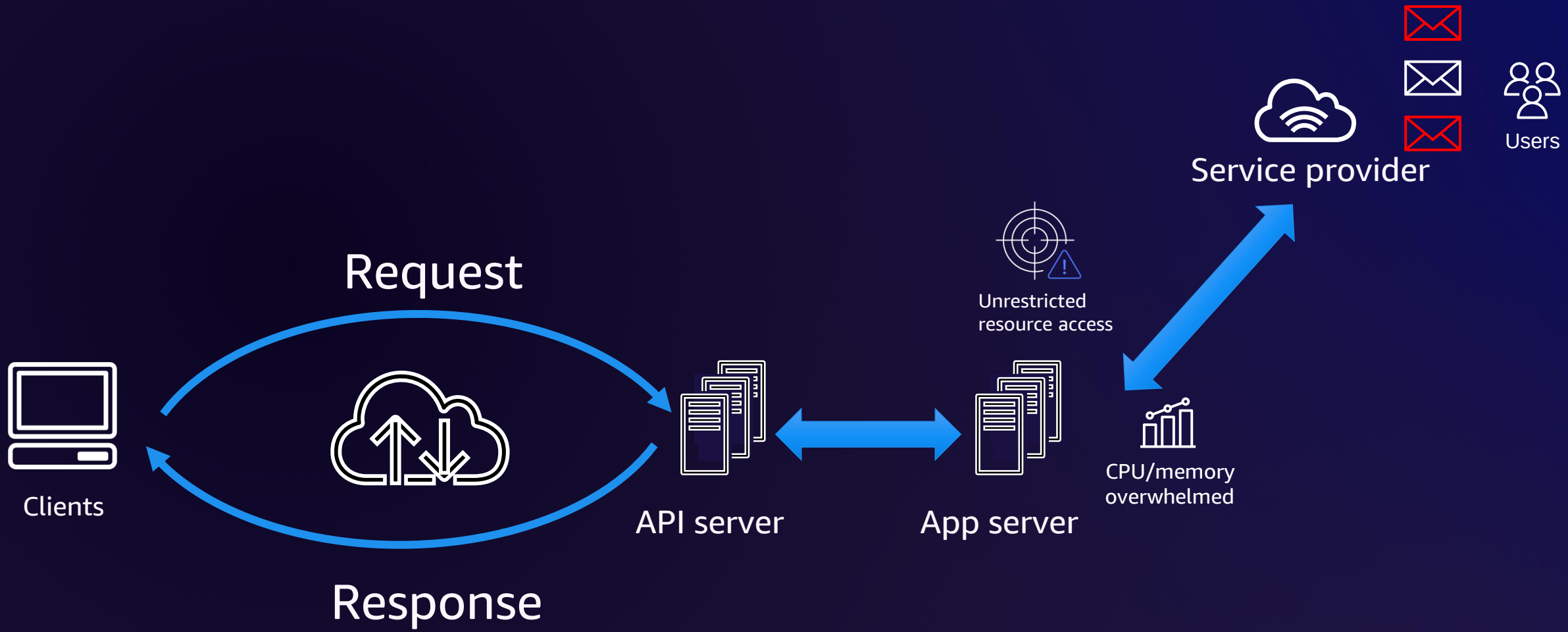
# Common API security challenges



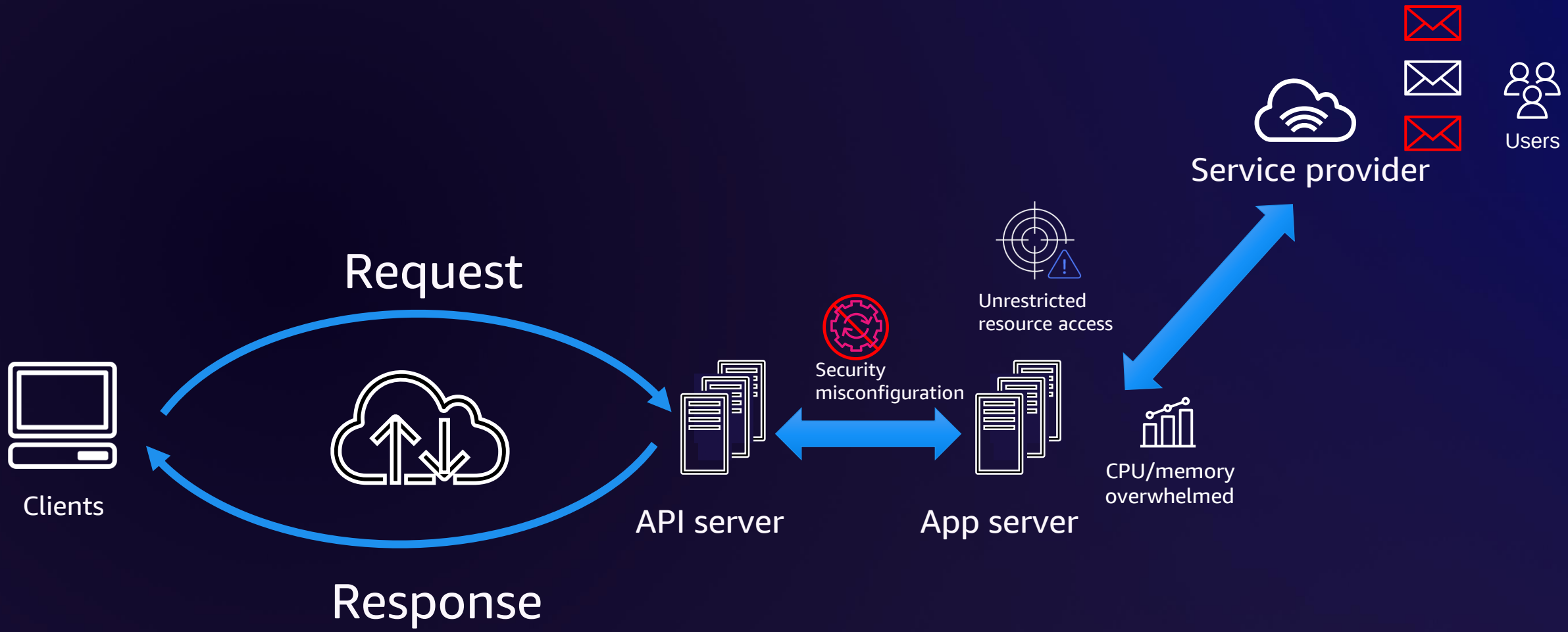
# Common API security challenges



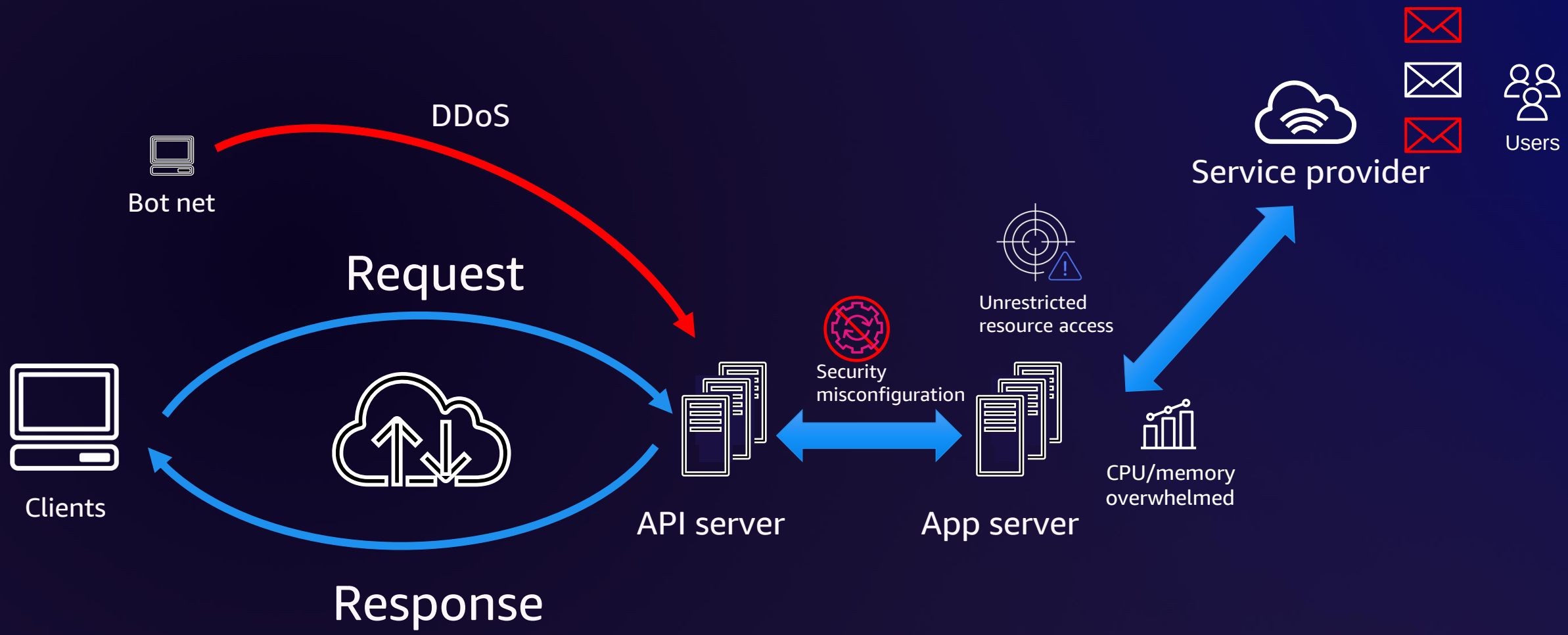
# Common API security challenges



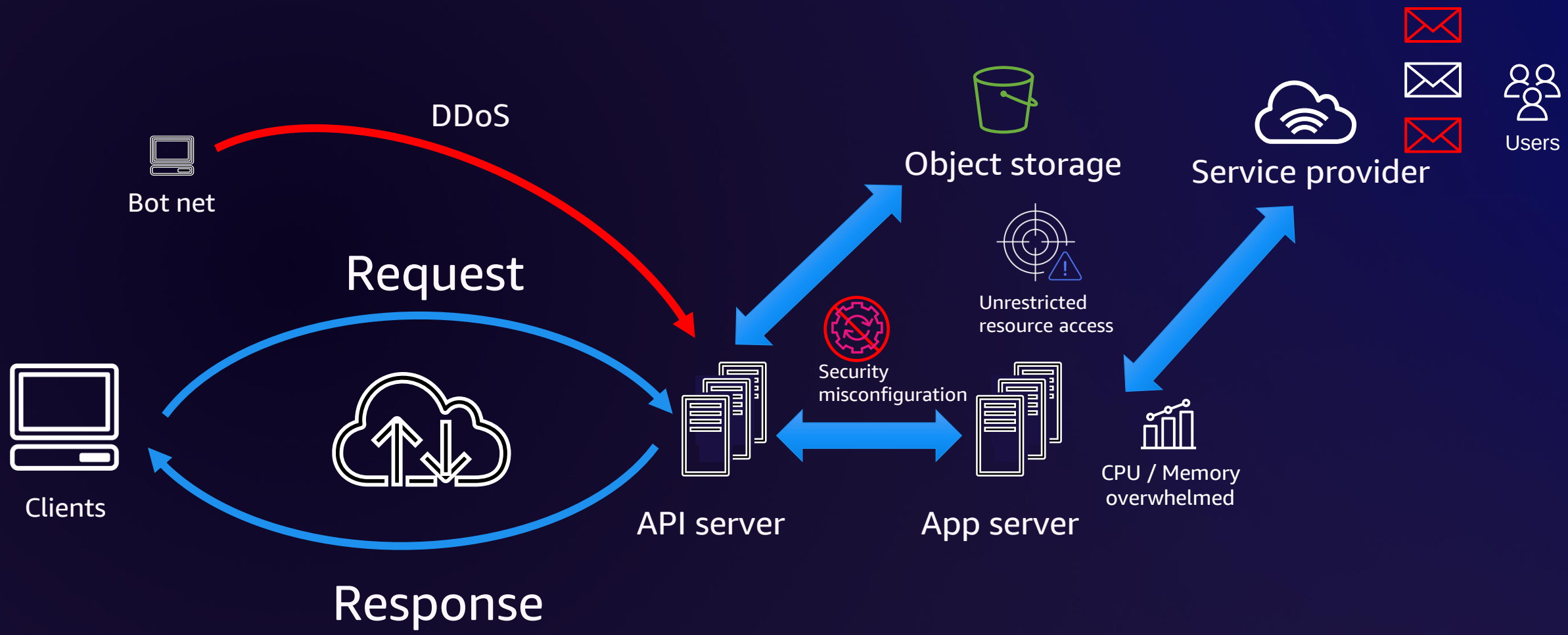
# Common API security challenges



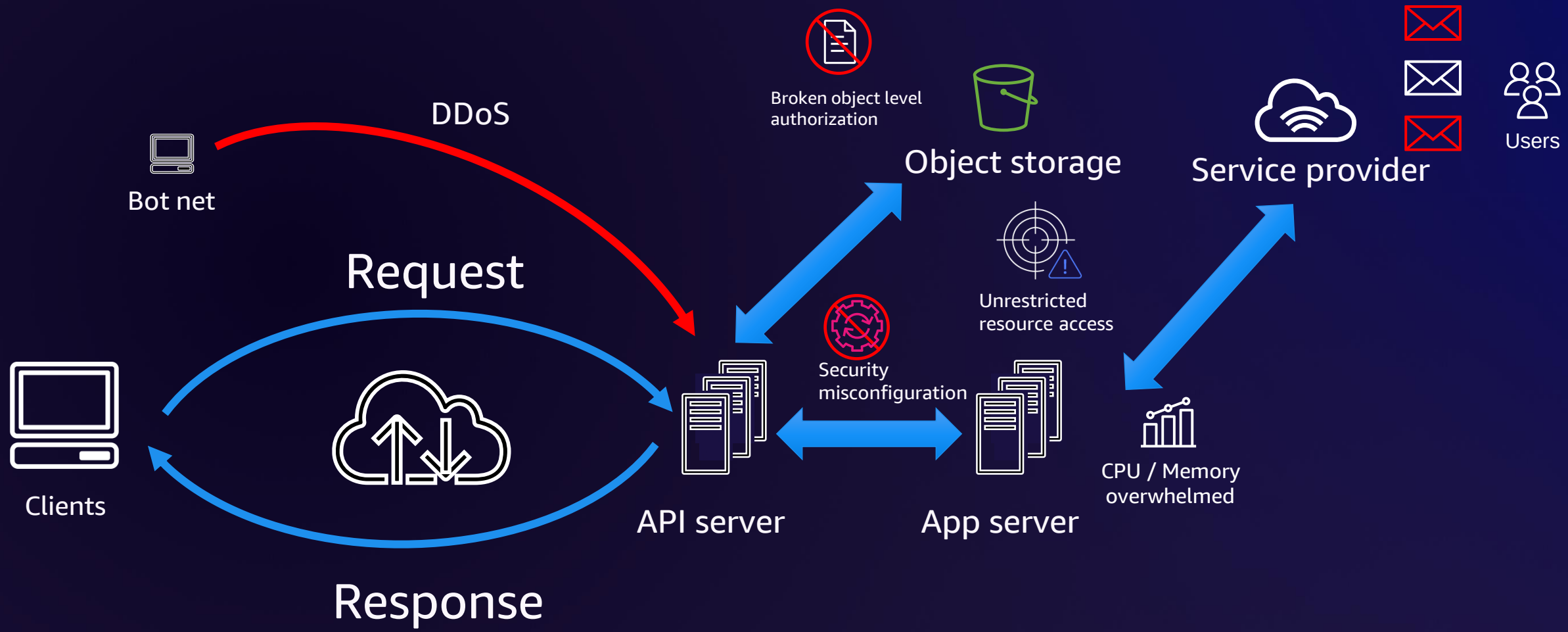
# Common API security challenges



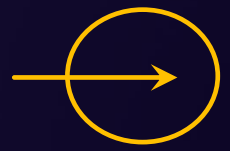
# Common API security challenges



# Common API security challenges



# Common API security challenges for enterprises



Broken function level authorization

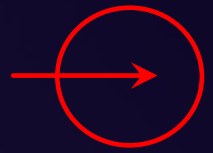
Broken object level authorization

Broken authentication

Broken object property level authorization

Security misconfiguration

OWASP Top 10

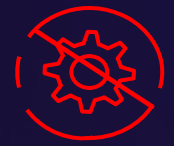


Unsafe consumption of APIs

Unrestricted access to sensitive business flows

Server side request forgery

Unrestricted resource access



Improper inventory management

https://owasp.org/API-Security/

# Well-Architected (WA) Framework

# Well-Architected Framework security pillar

# Well-Architected Framework security pillar

# Well-Architected Framework security pillar

# Well-Architected Framework security pillar


## Focus areas

# Well-Architected Framework security pillar


## Focus areas

## Best practices

[ SEC05-BP02 Control traffic at all layers ]

# Well-Architected Framework security pillar


## Focus areas

## Best practices

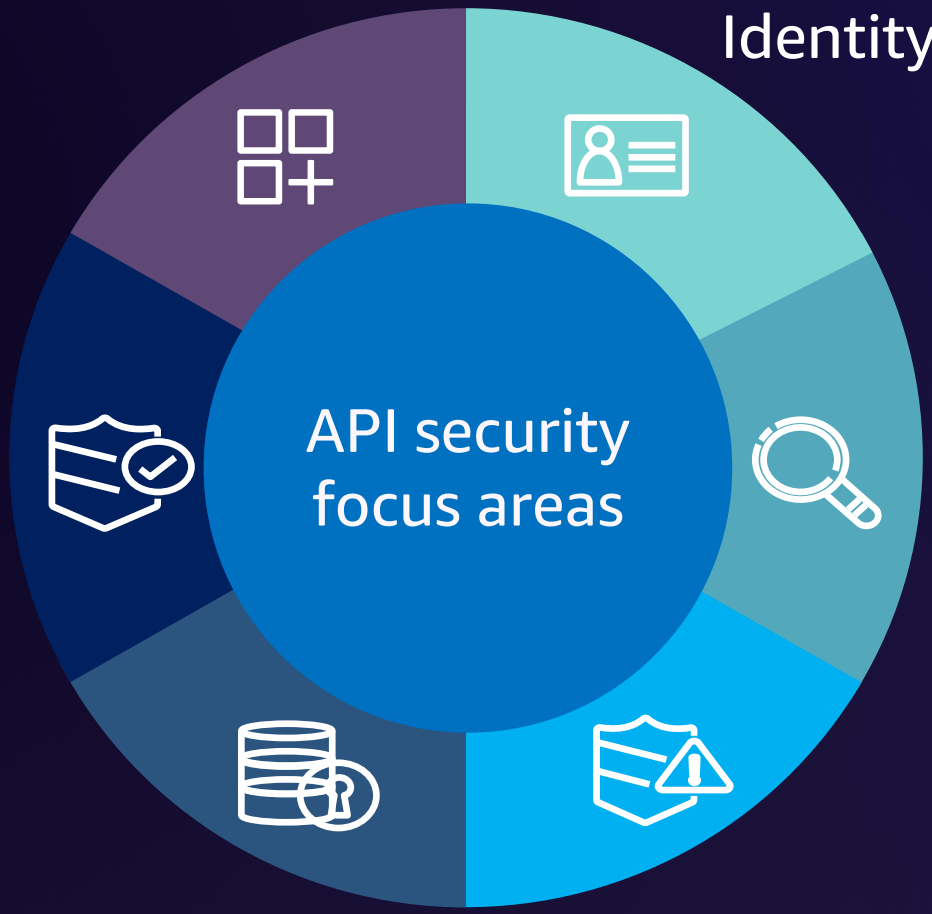
[ SEC05-BP02 Control traffic at all layers ]

## Implementation steps

1. *Amazon VPC security*
2. *AWS Web Application Firewall (AWS WAF)*
3. *Amazon Route 53*
4. *...*

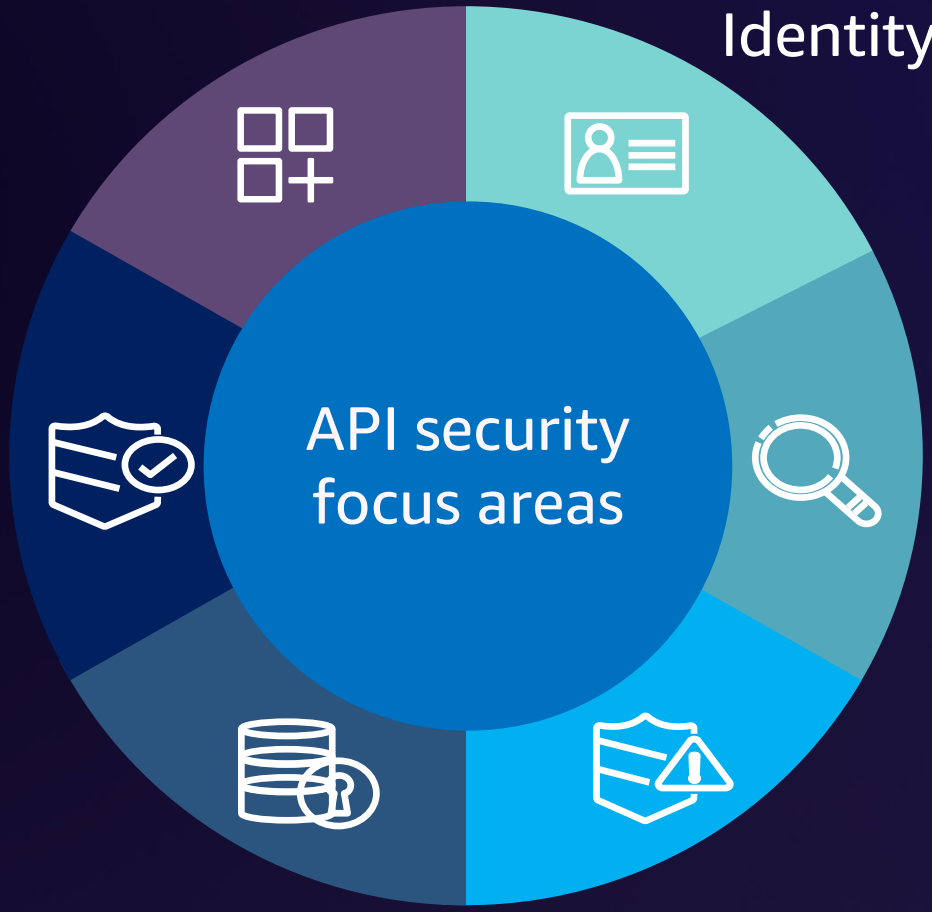
# Security foundations

# Security foundations




Identity and access management

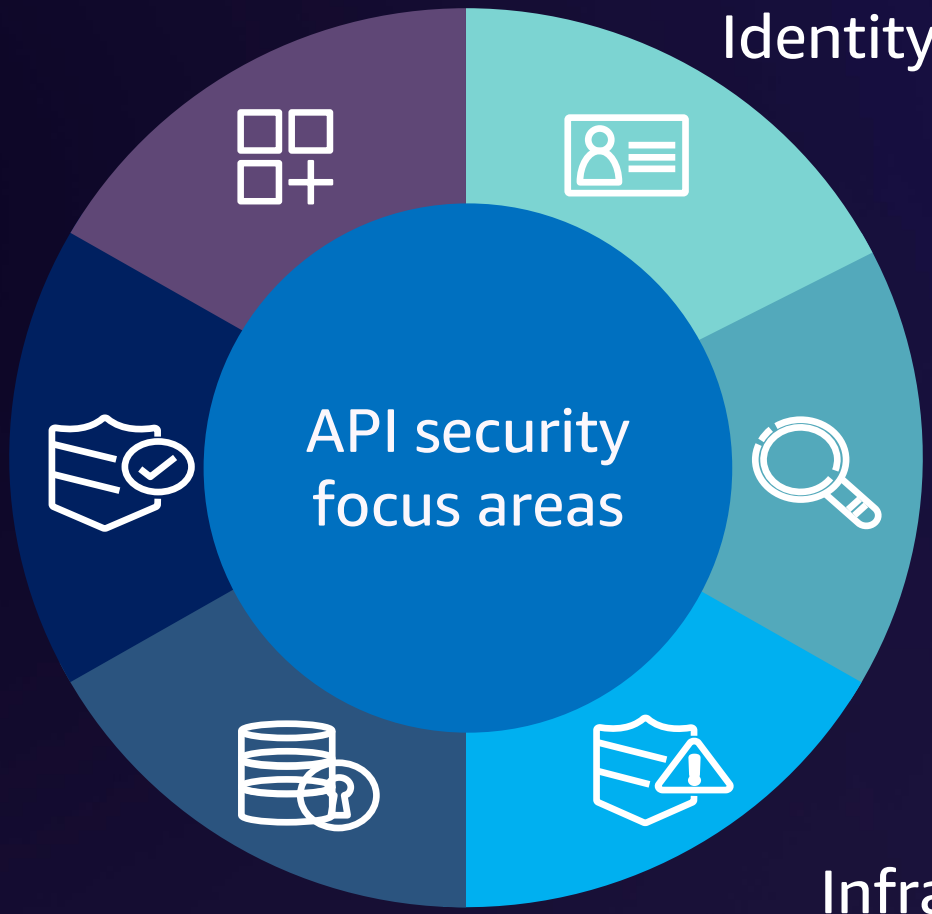
# Security foundations




Identity and access management

Detection

# Security foundations

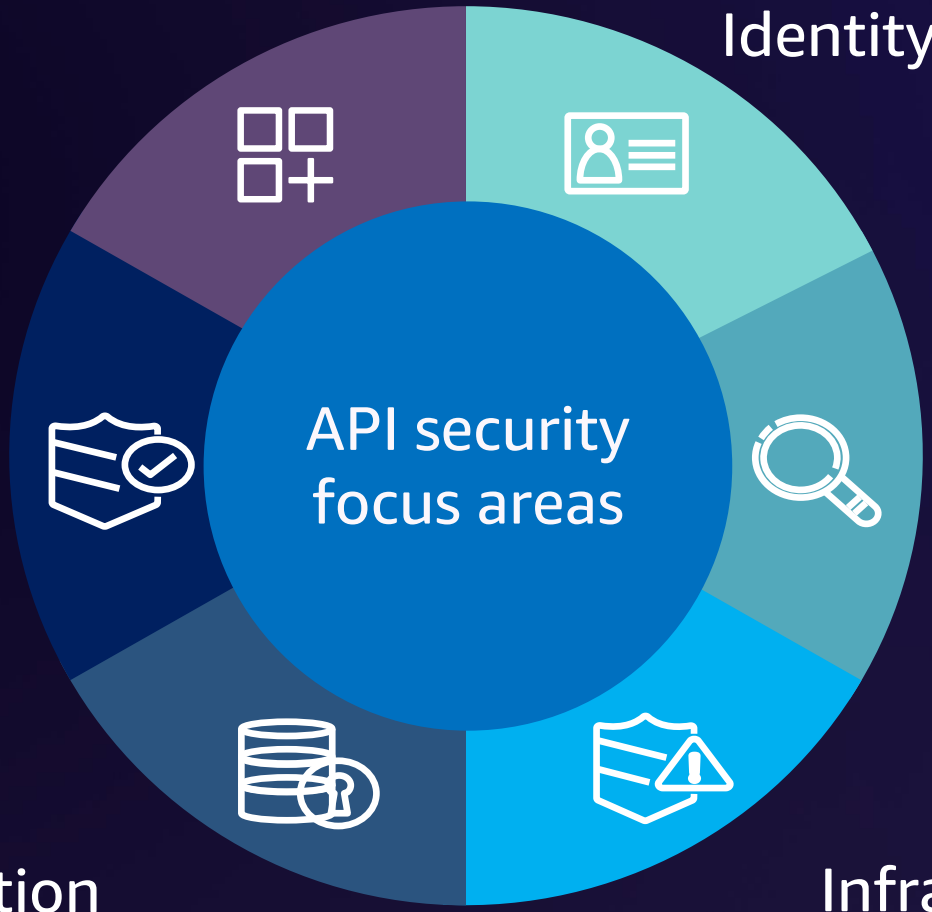



Identity and access management

Detection

Infrastructure protection

# Security foundations



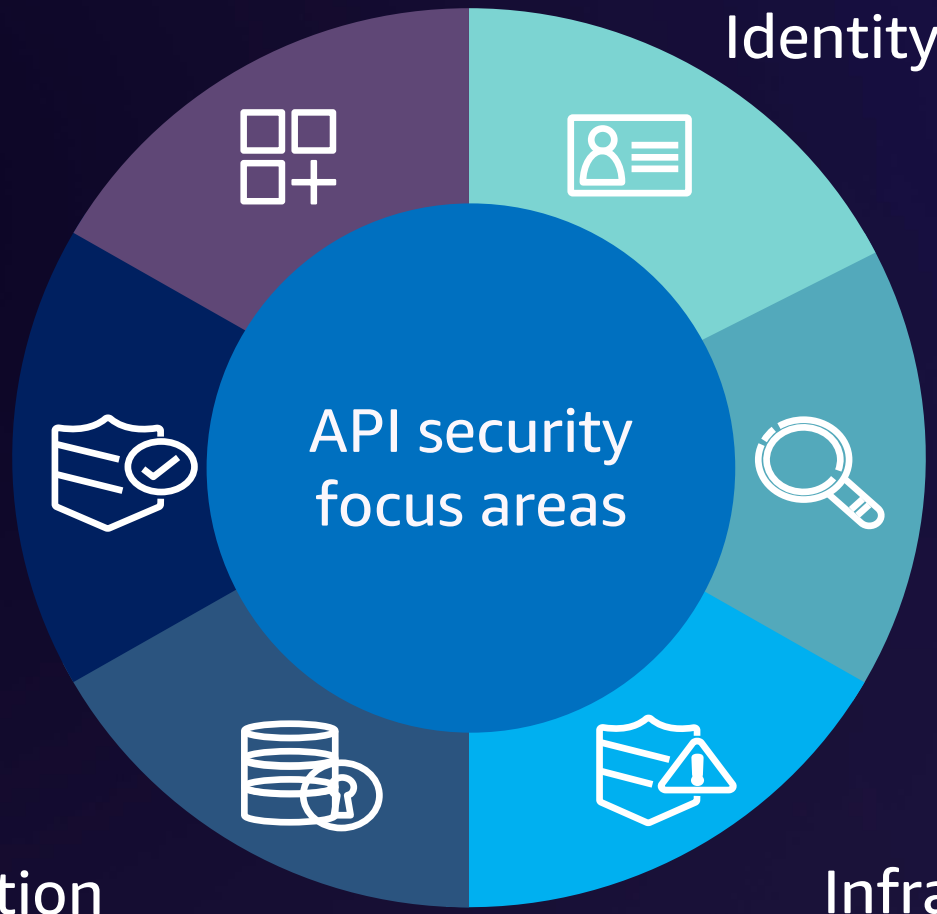

Identity and access management

Detection

Infrastructure protection

Data protection

# Security foundations




Identity and access management

Detection

Infrastructure protection

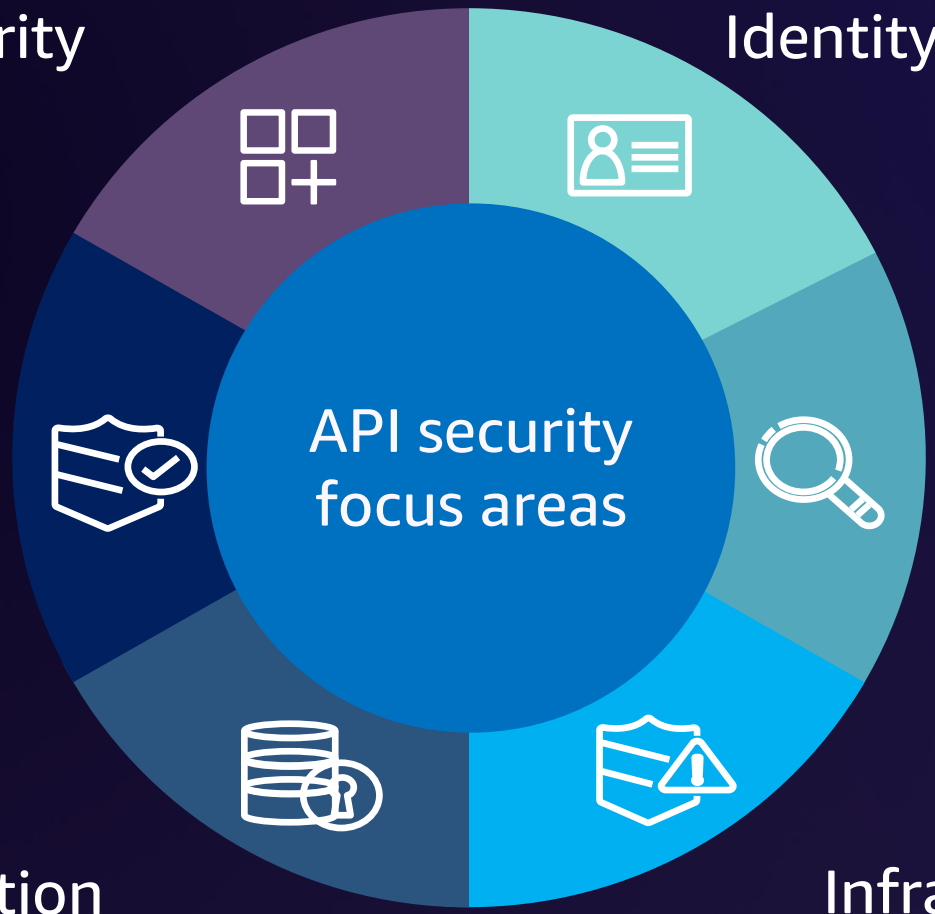
Data protection

Incident response

# Security foundations

Application security

Identity and access management




Incident response

Detection

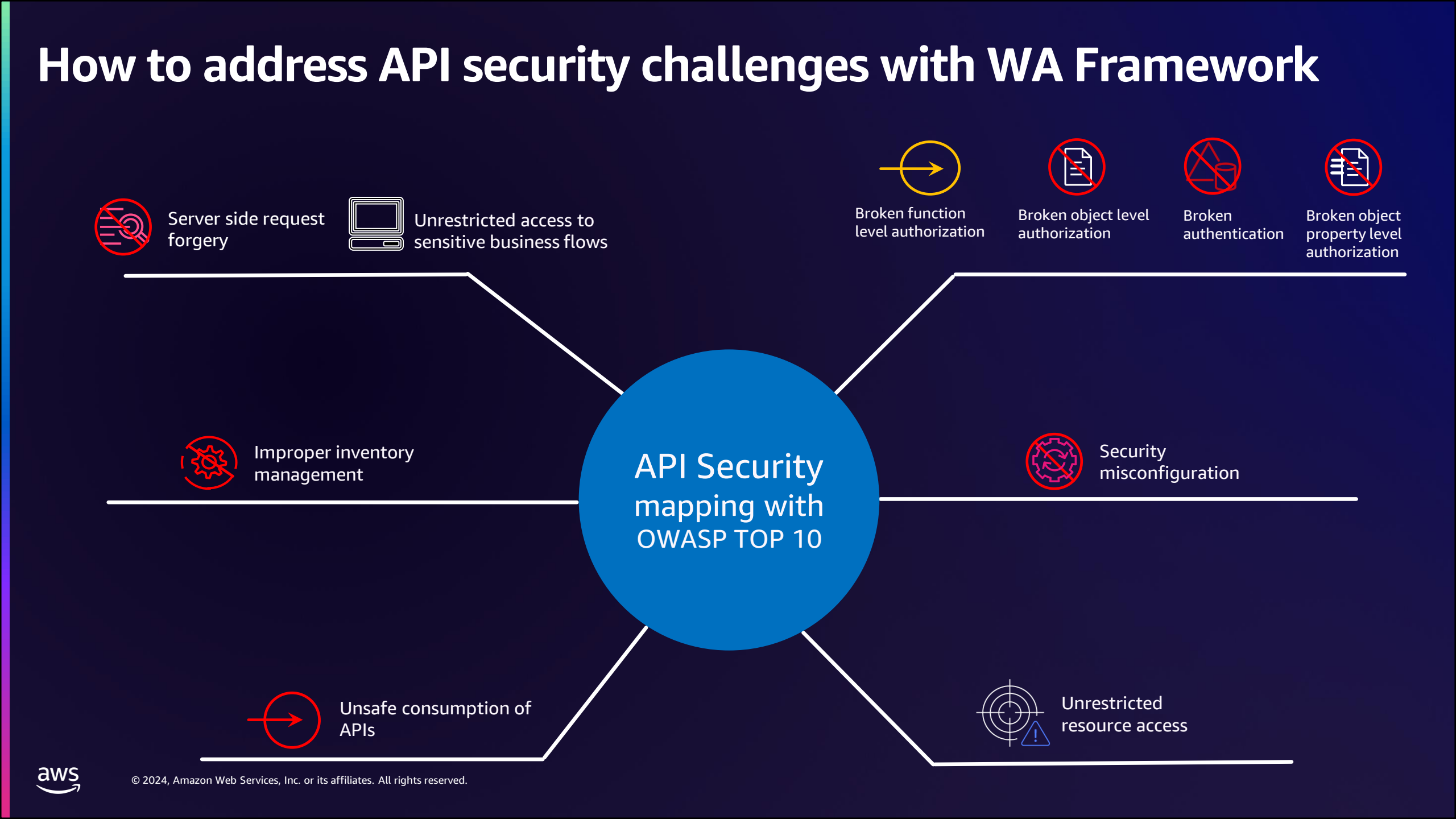
Data protection

Infrastructure protection

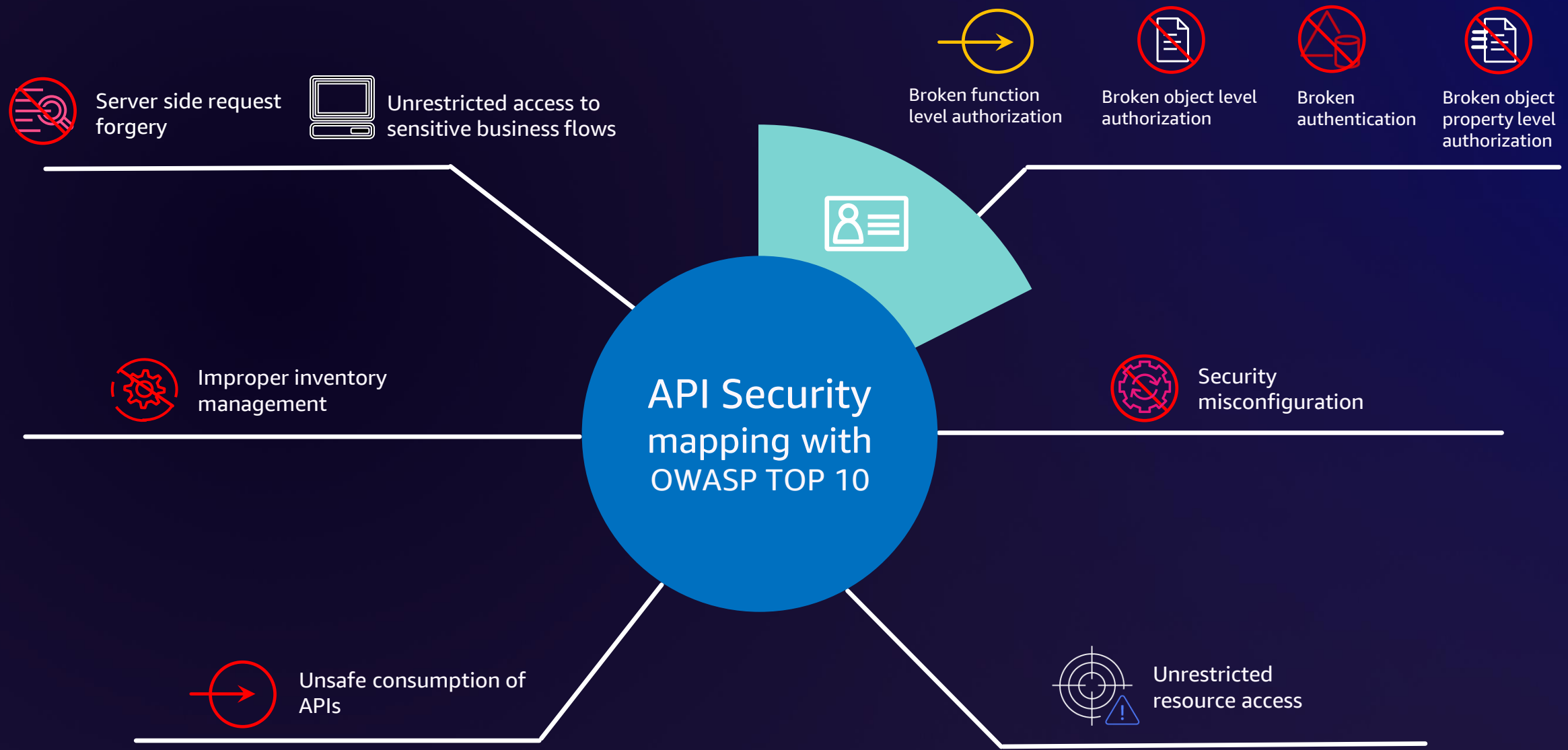
# How to address API security challenges with WA Framework

API Security
mapping with
OWASP TOP 10

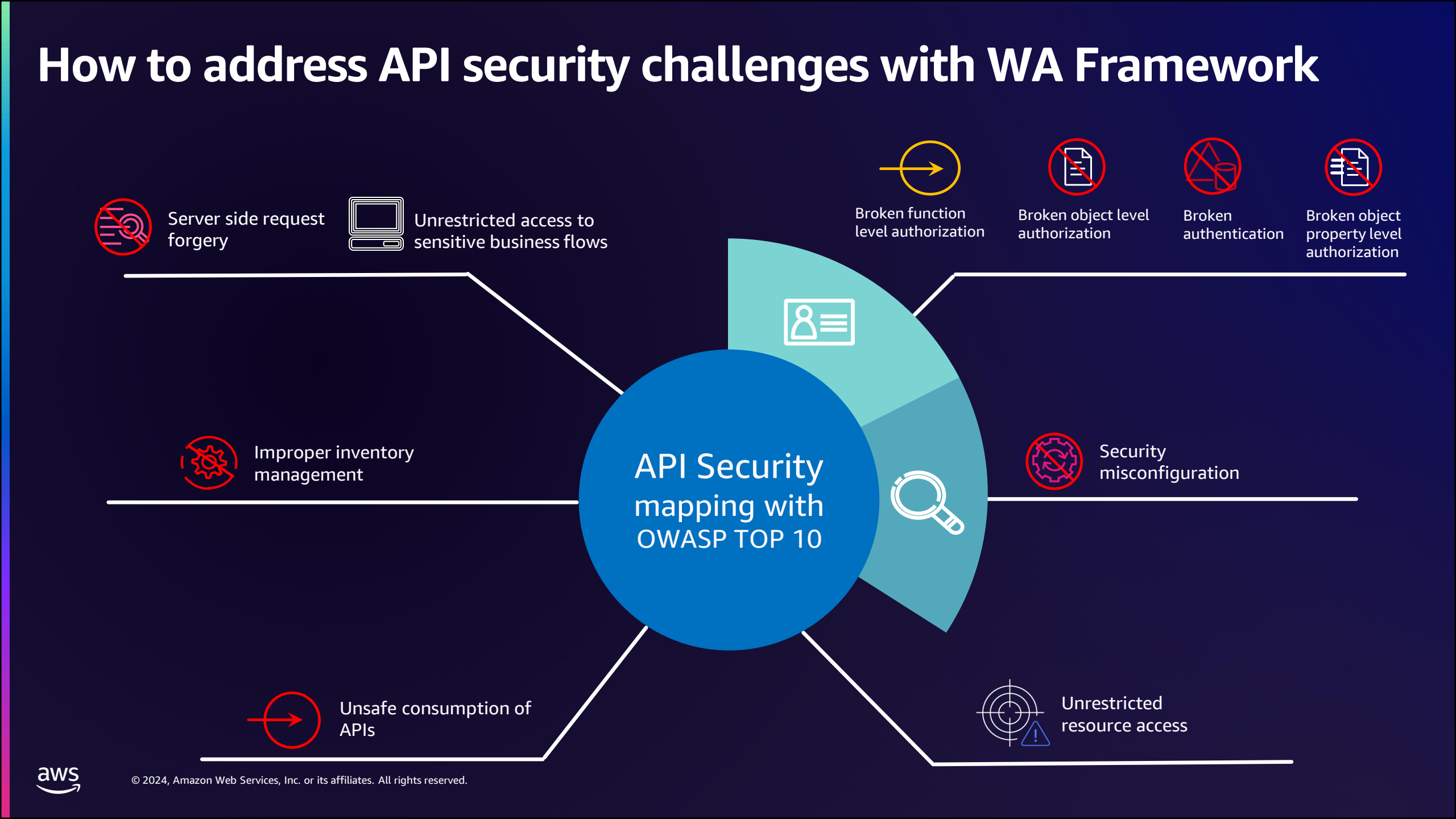
# How to address API security challenges with WA Framework



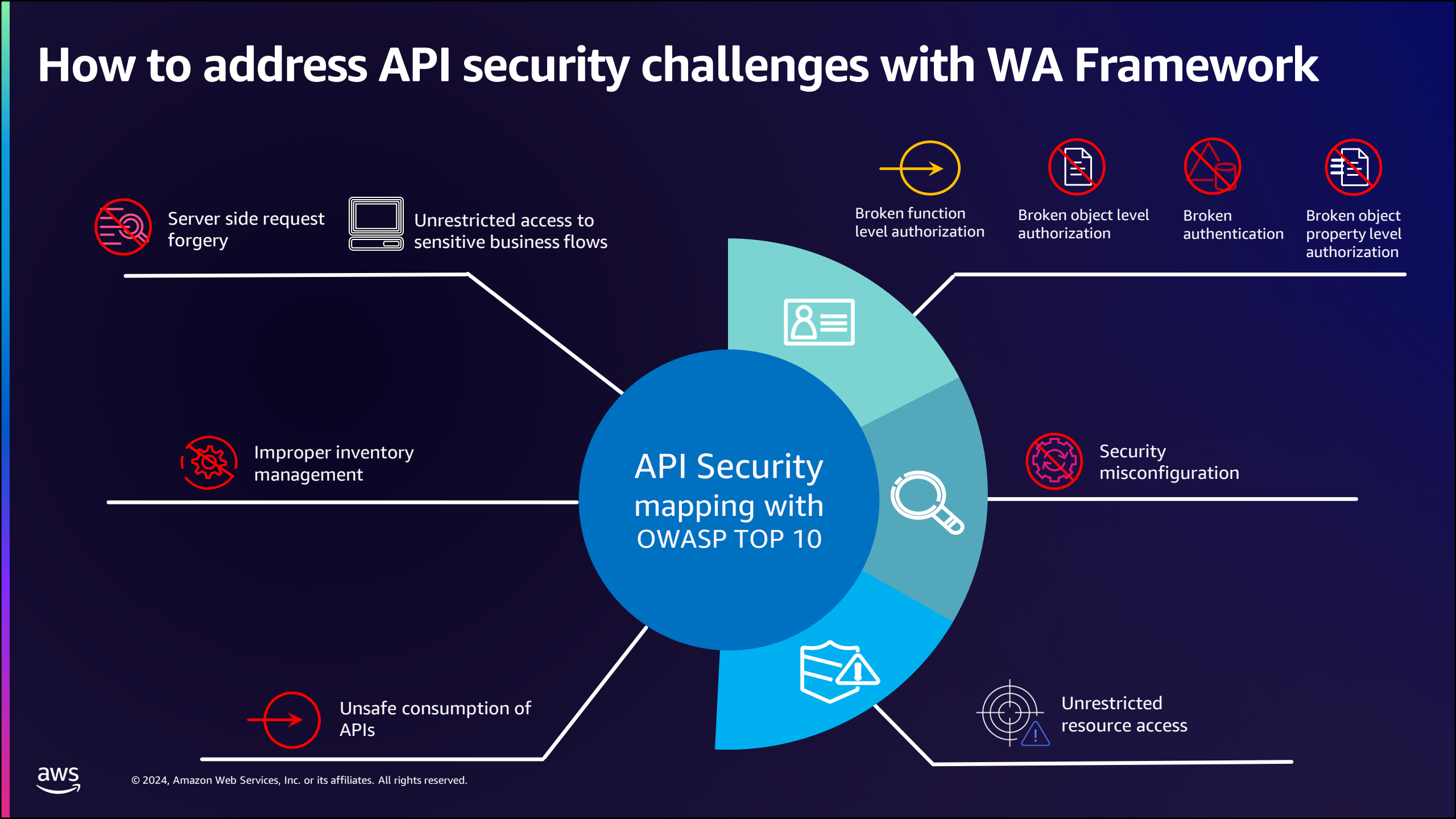
# How to address API security challenges with WA Framework



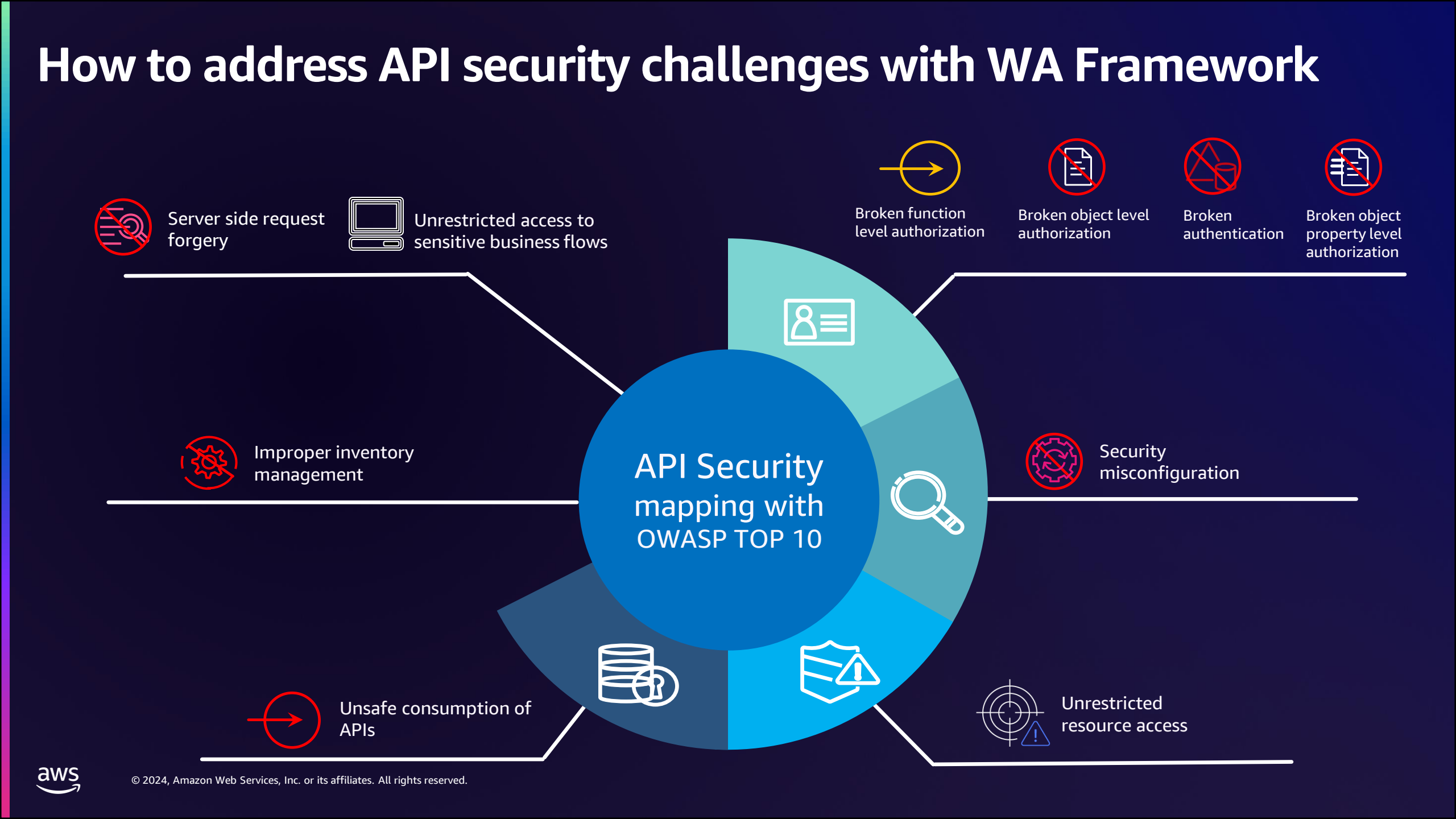
# How to address API security challenges with WA Framework



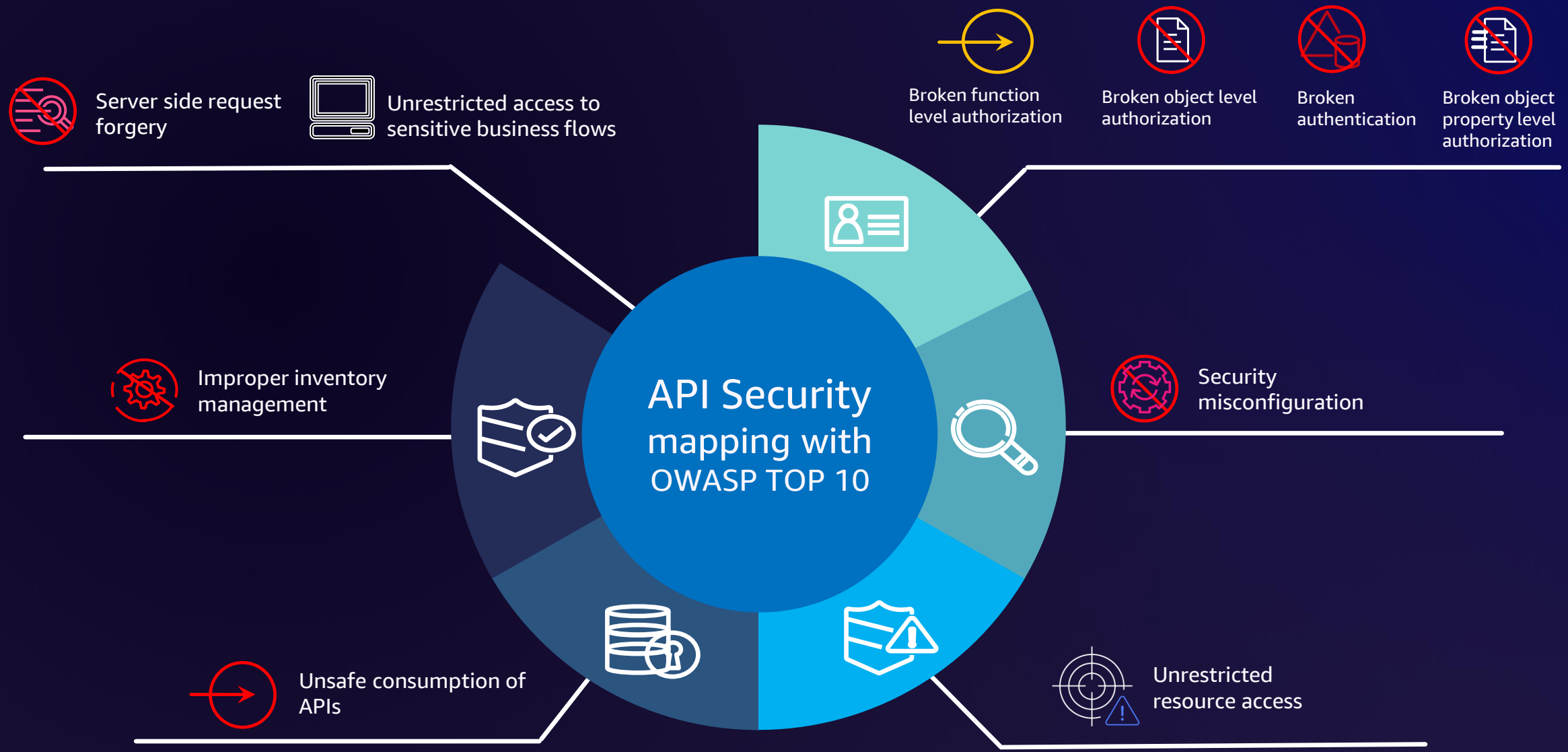
# How to address API security challenges with WA Framework



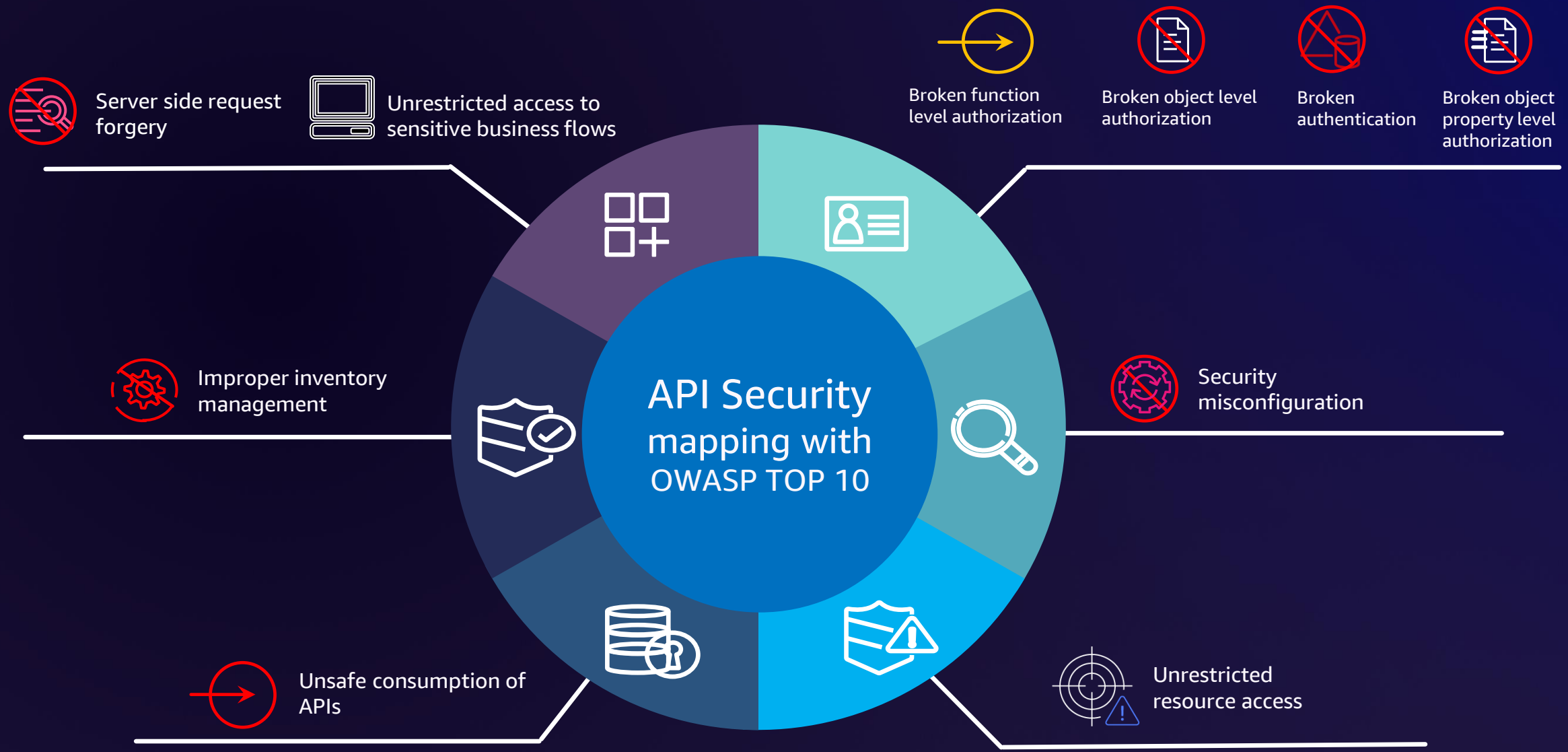
# How to address API security challenges with WA Framework



# How to address API security challenges with WA Framework

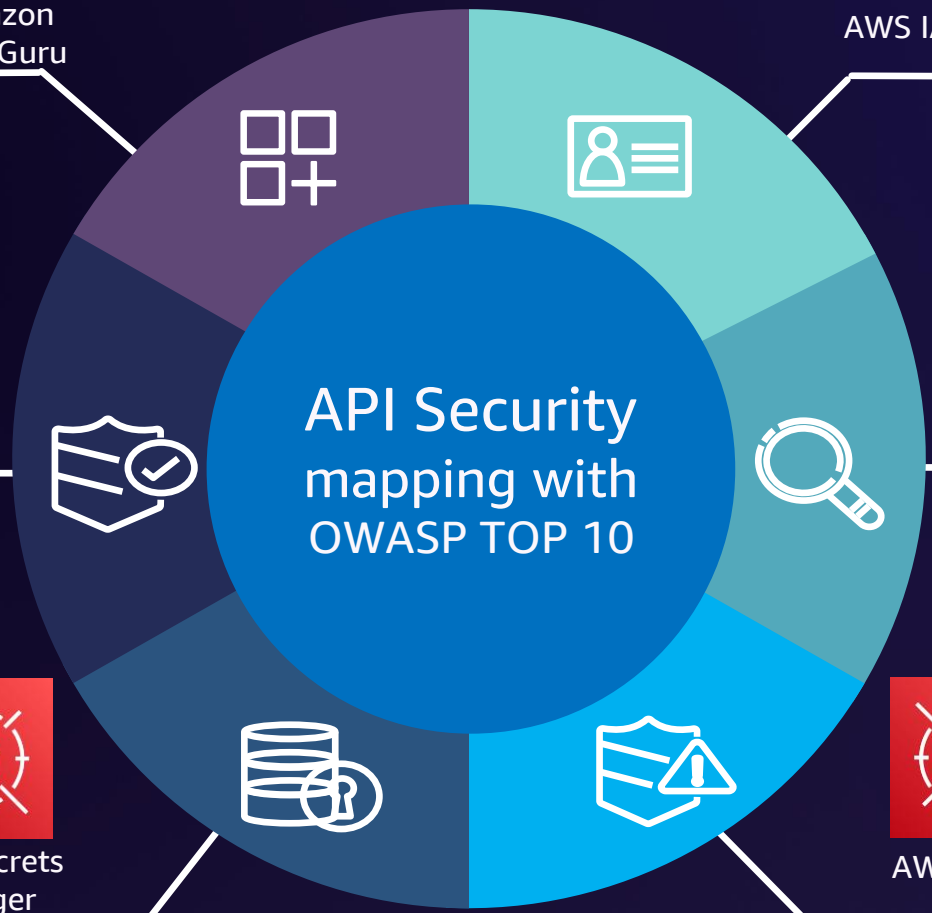


# How to address API security challenges with WA Framework



# How to address API security challenges with WA Framework

# How to address API security challenges with WA Framework




AWS CodePipeline

AWS Service Catalog

Amazon CodeGuru

AWS IAM

Amazon Cognito

AWS IAM Identity Center

AWS Organizations

Amazon Detective

AWS Fault Injection Simulator

AWS CloudTrail

Amazon CloudWatch

AWS Config

Amazon Macie

AWS KMS

AWS Secrets Manager

AWS WAF

AWS Shield

Amazon CloudFront

AWS Firewall Manager

# Mitigating common API security challenges

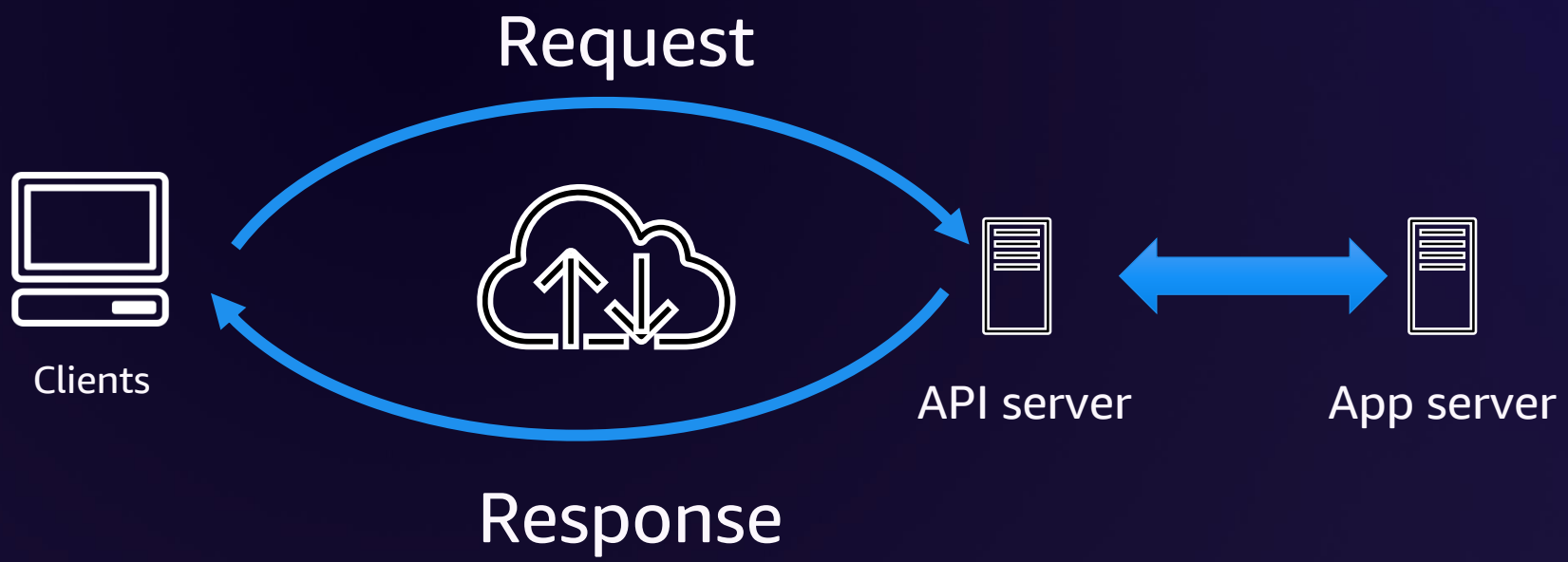
# How to address API security challenges with WA Framework

- Identity and access management
- Detection
- Infrastructure protection
- Data protection
- Incident response
- Application security

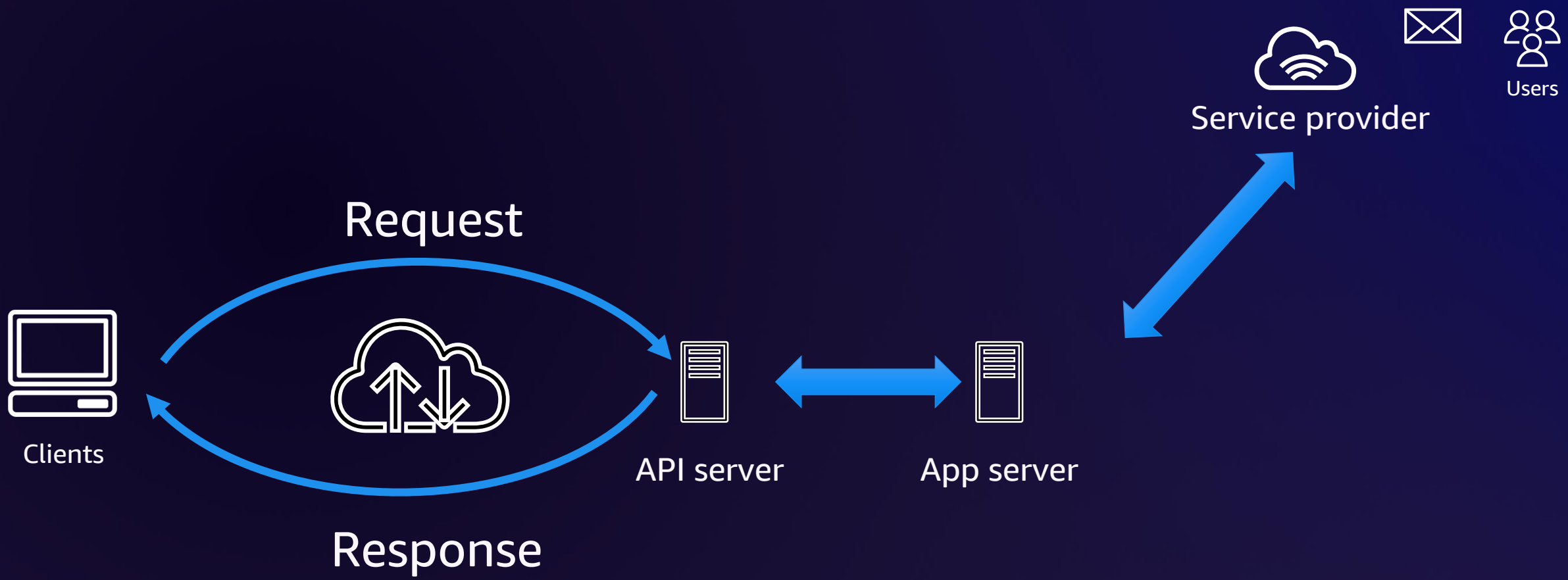
# How to address API security challenges with WA Framework

- Identity and access management
- Detection
- Infrastructure protection
- Data protection
- Incident response
- Application security

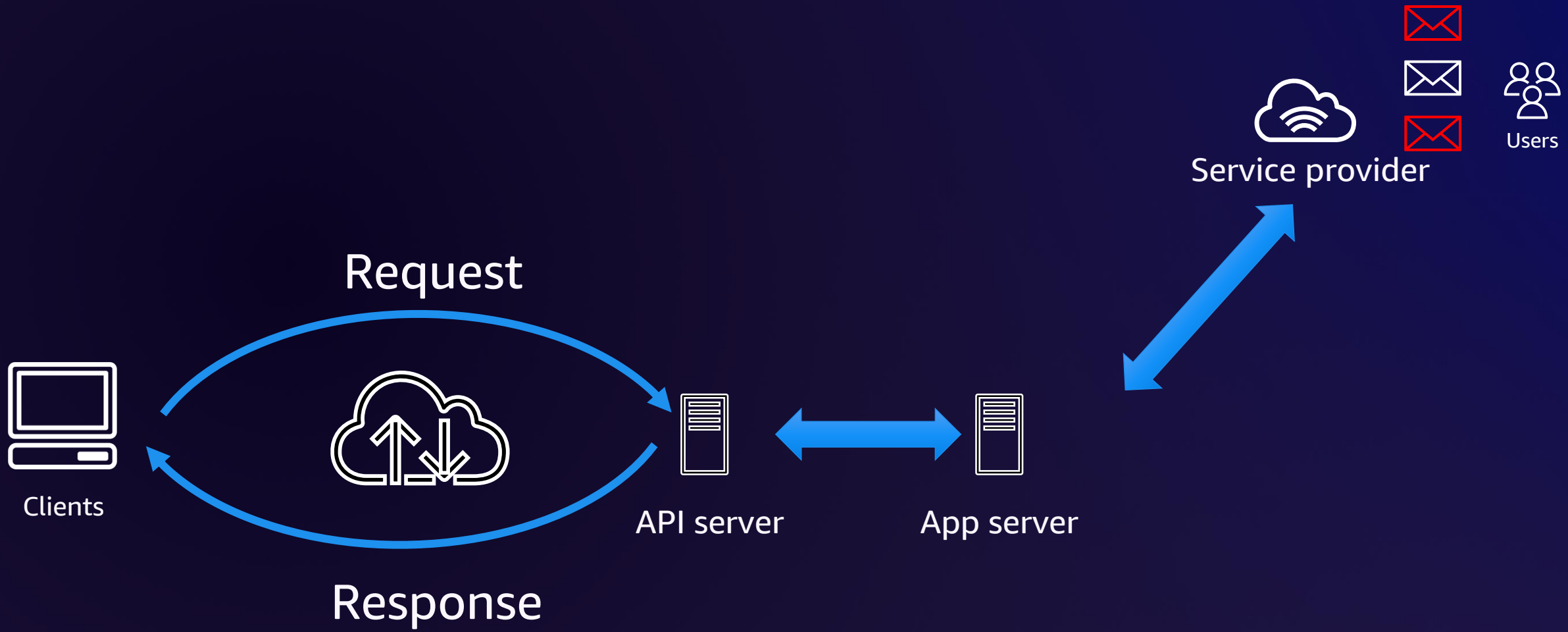
# Common API security challenges



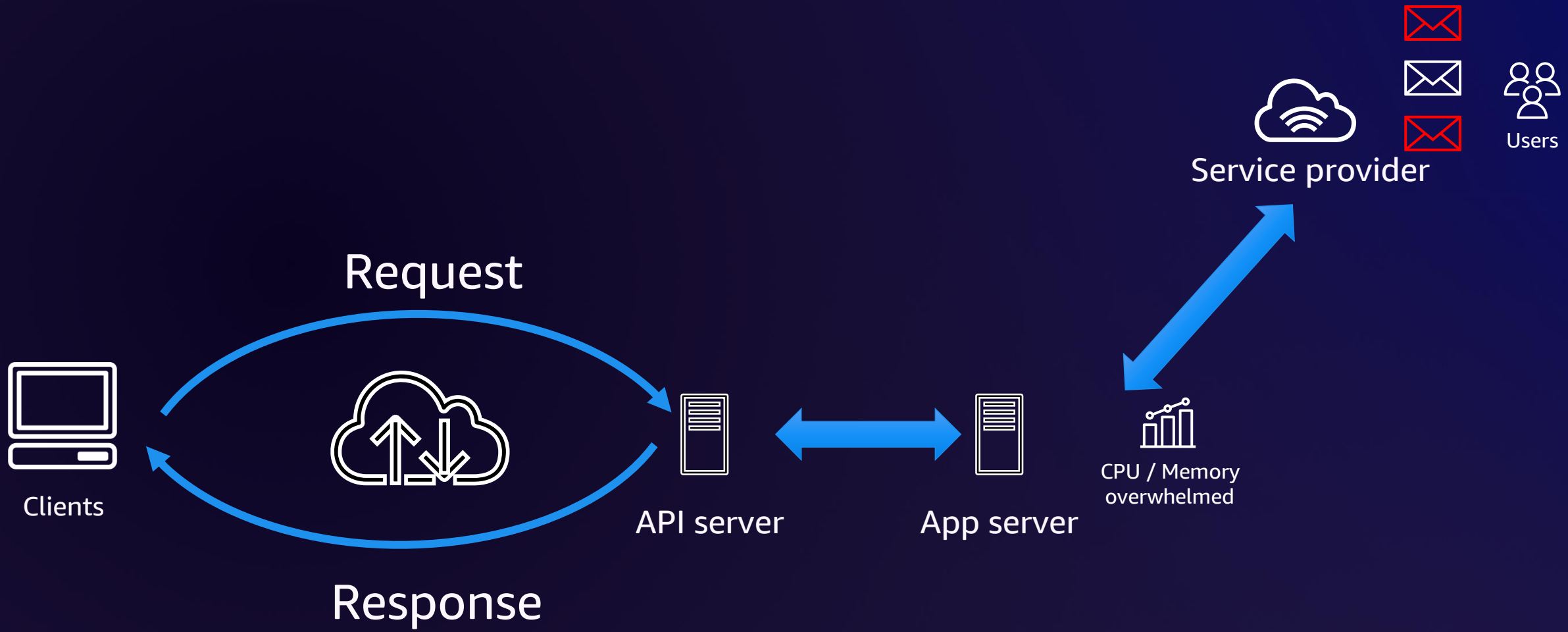
# Common API security challenges



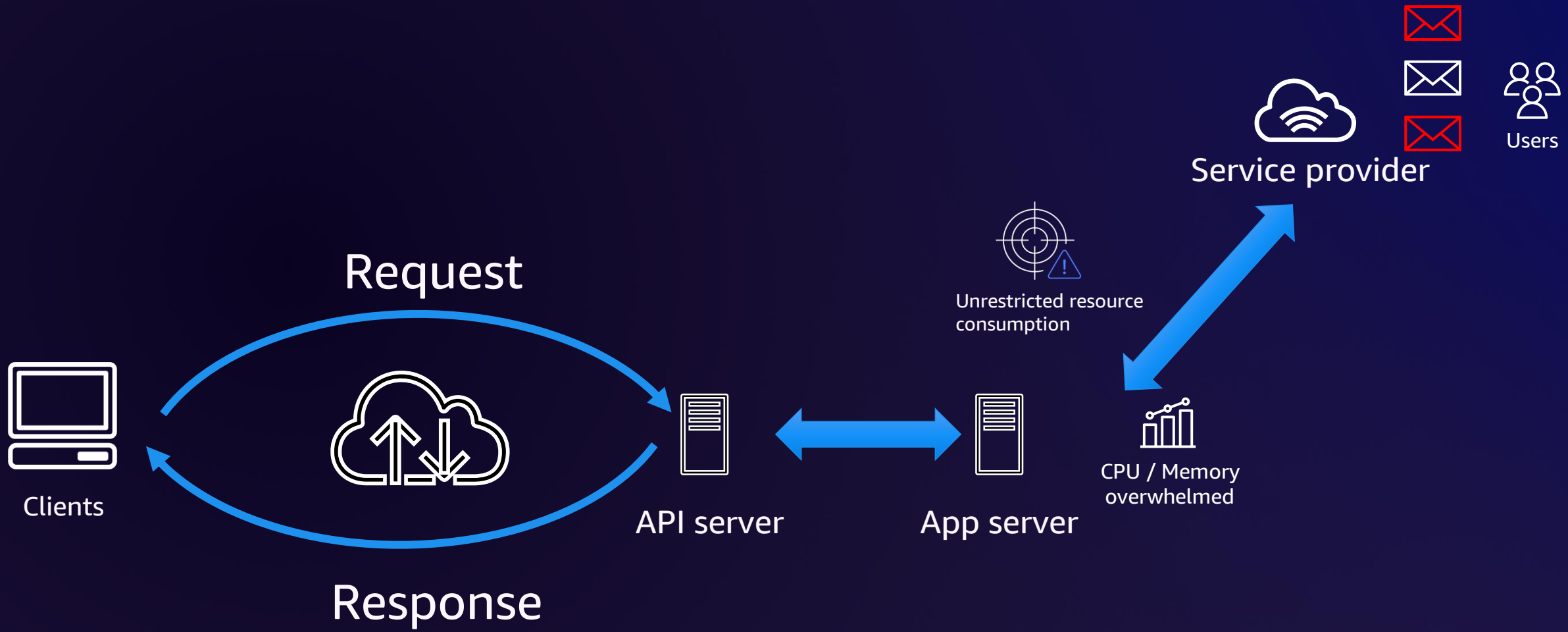
# Common API security challenges



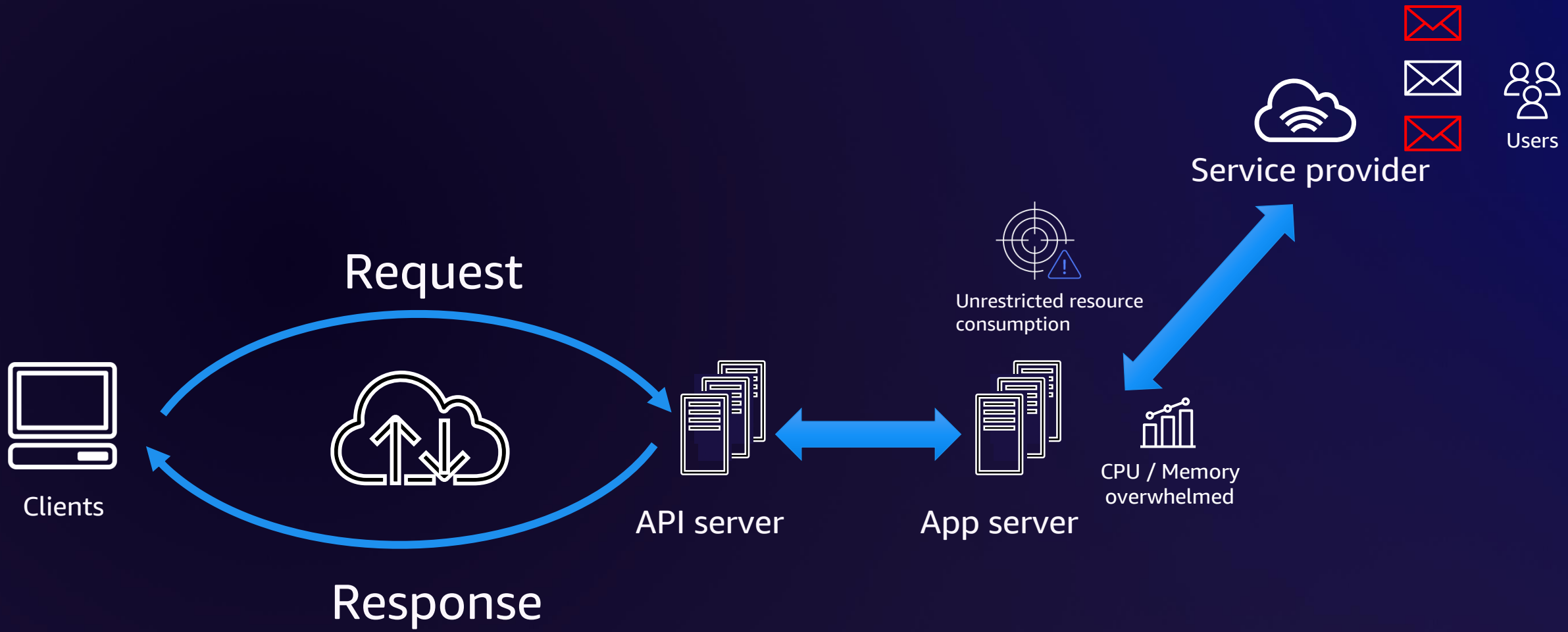
# Common API security challenges



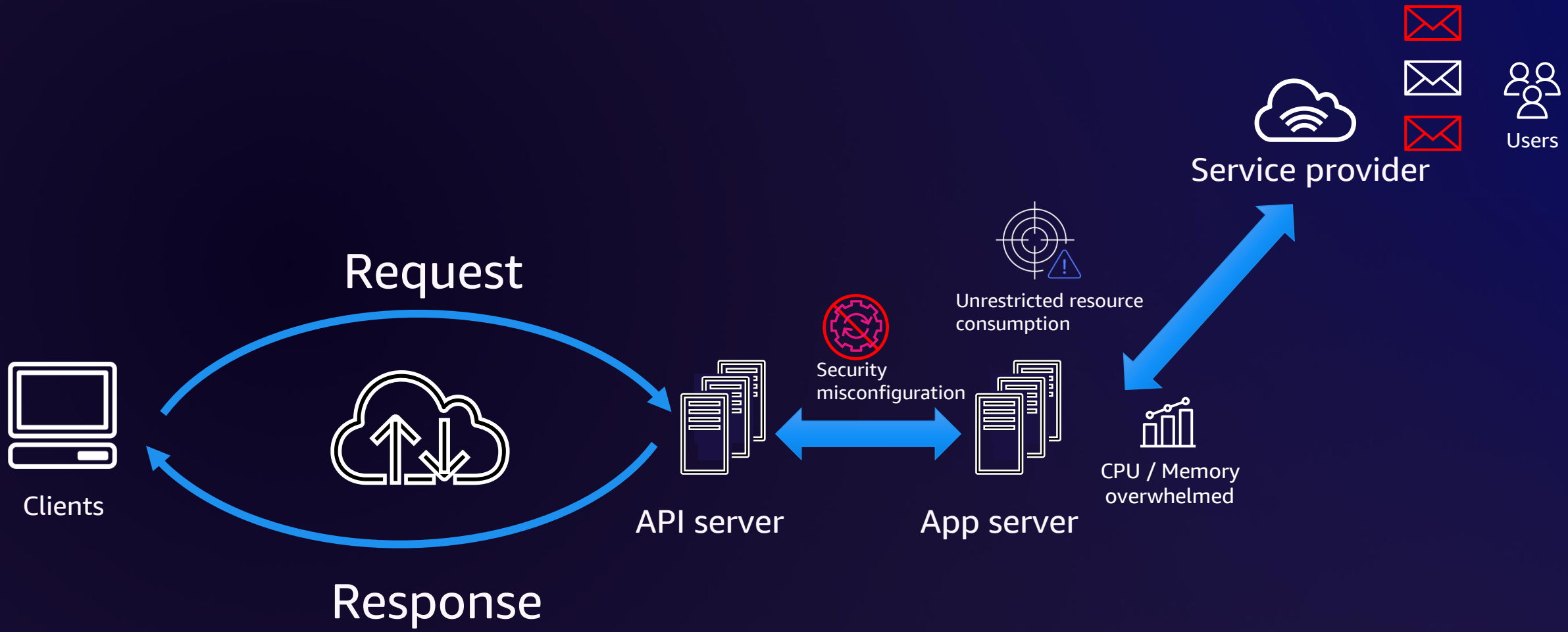
# Common API security challenges



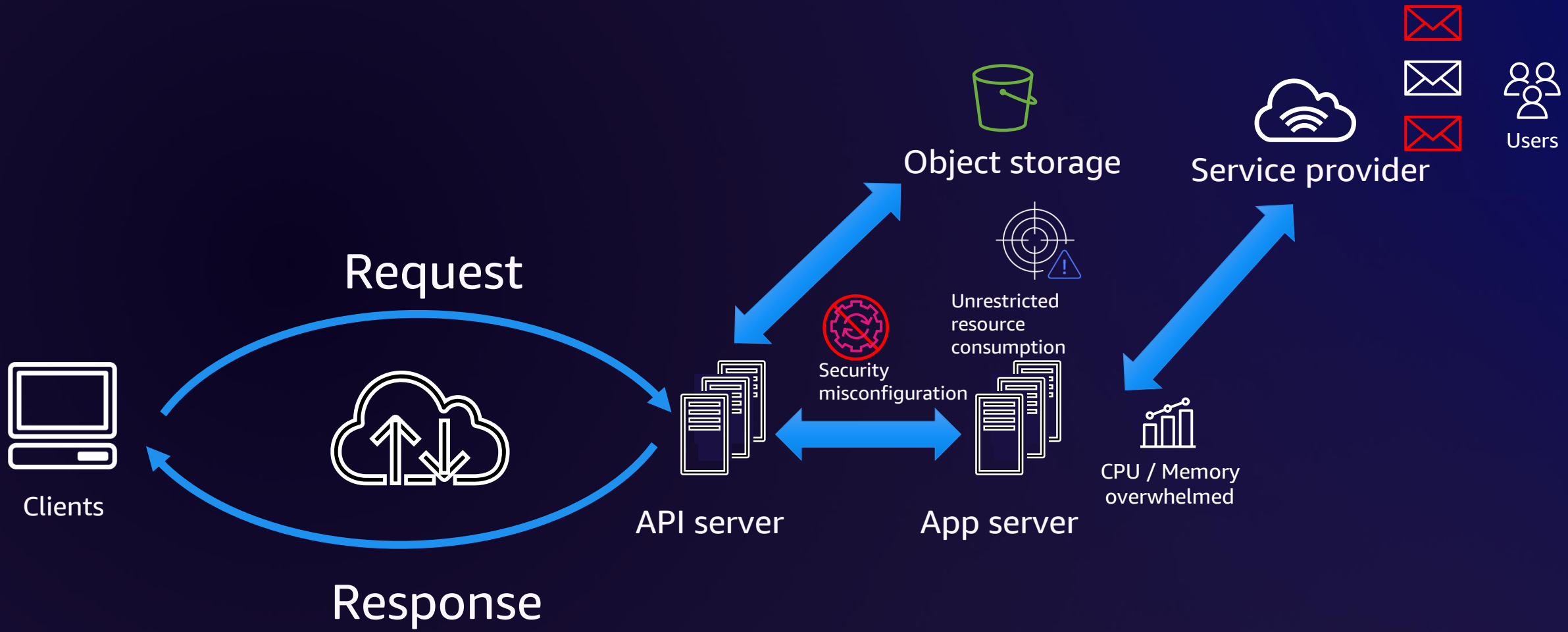
# Common API security challenges



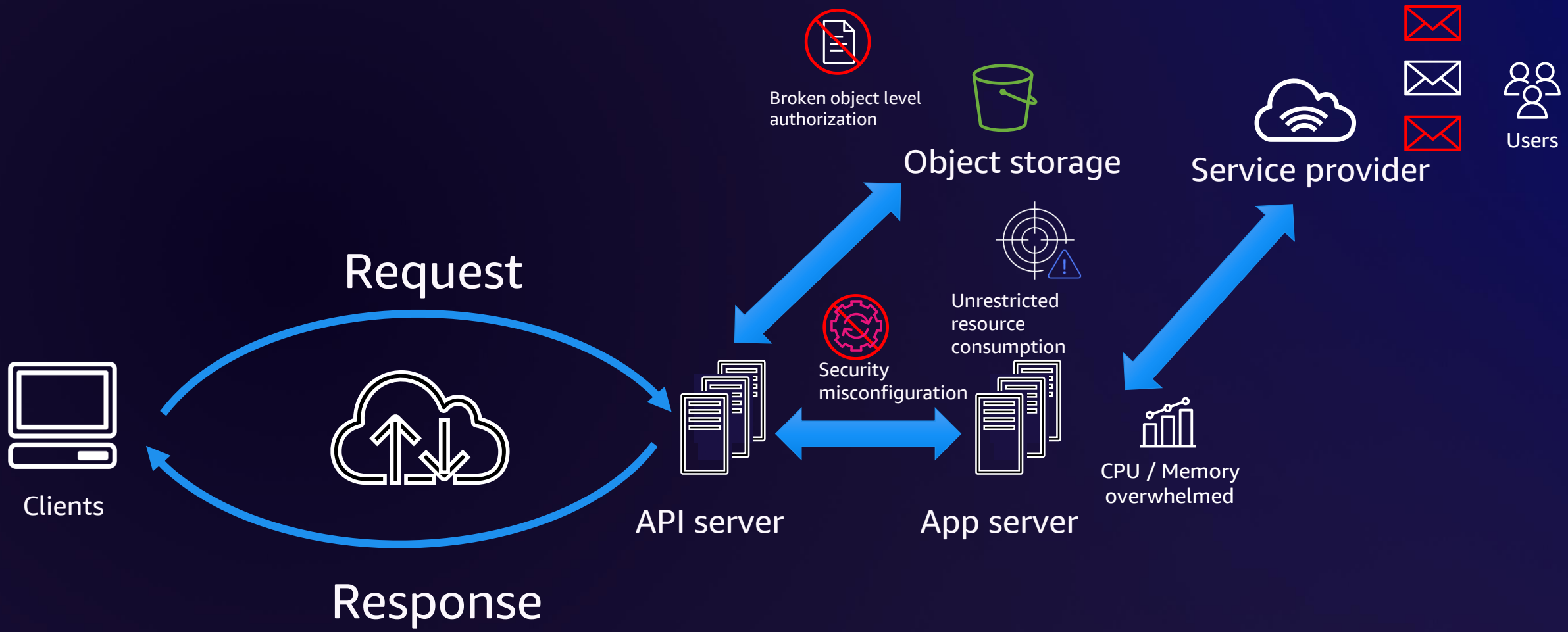
# Common API security challenges



# Common API security challenges



# Common API security challenges

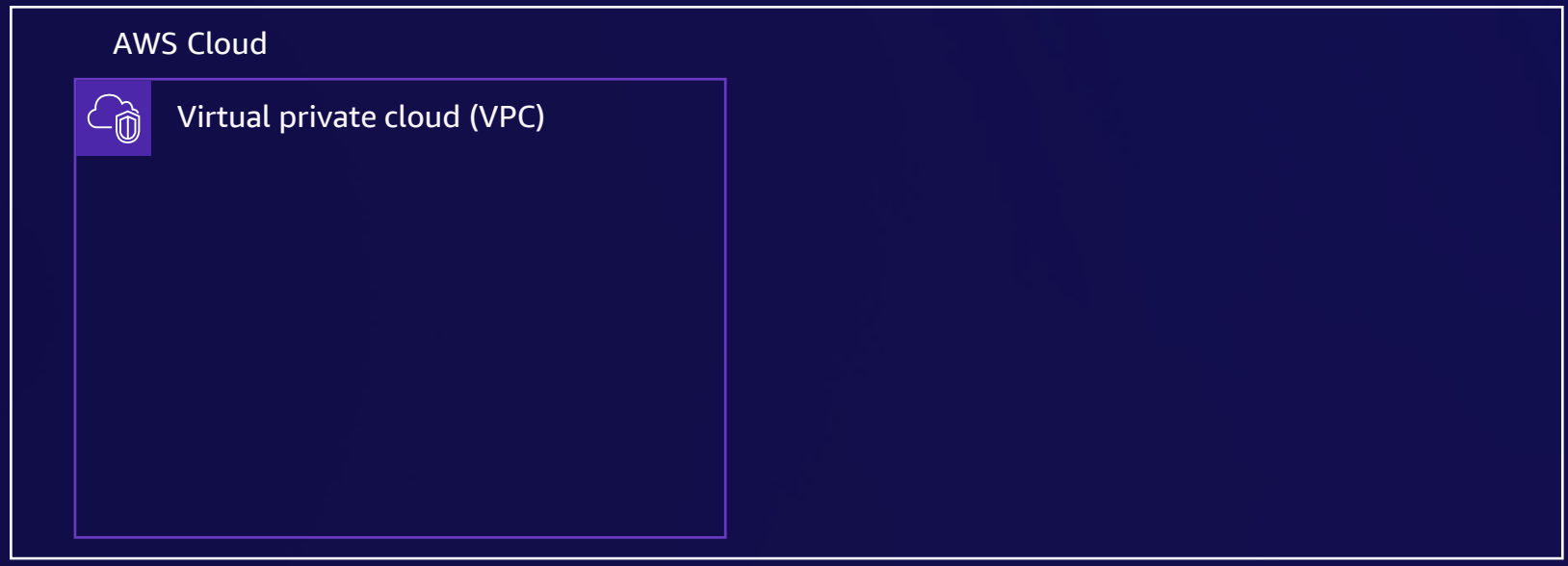


# Implement a strong identity foundation


AWS Cloud

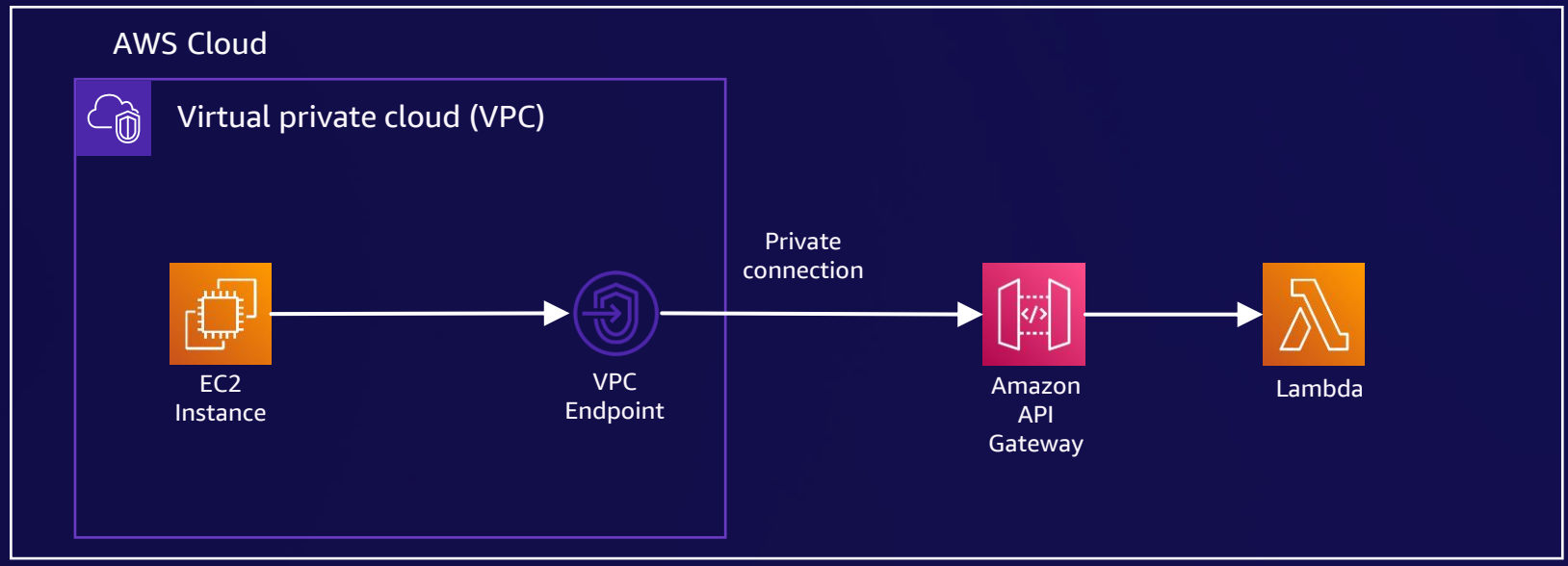
PRIVATE API ENDPOINT

# Implement a strong identity foundation

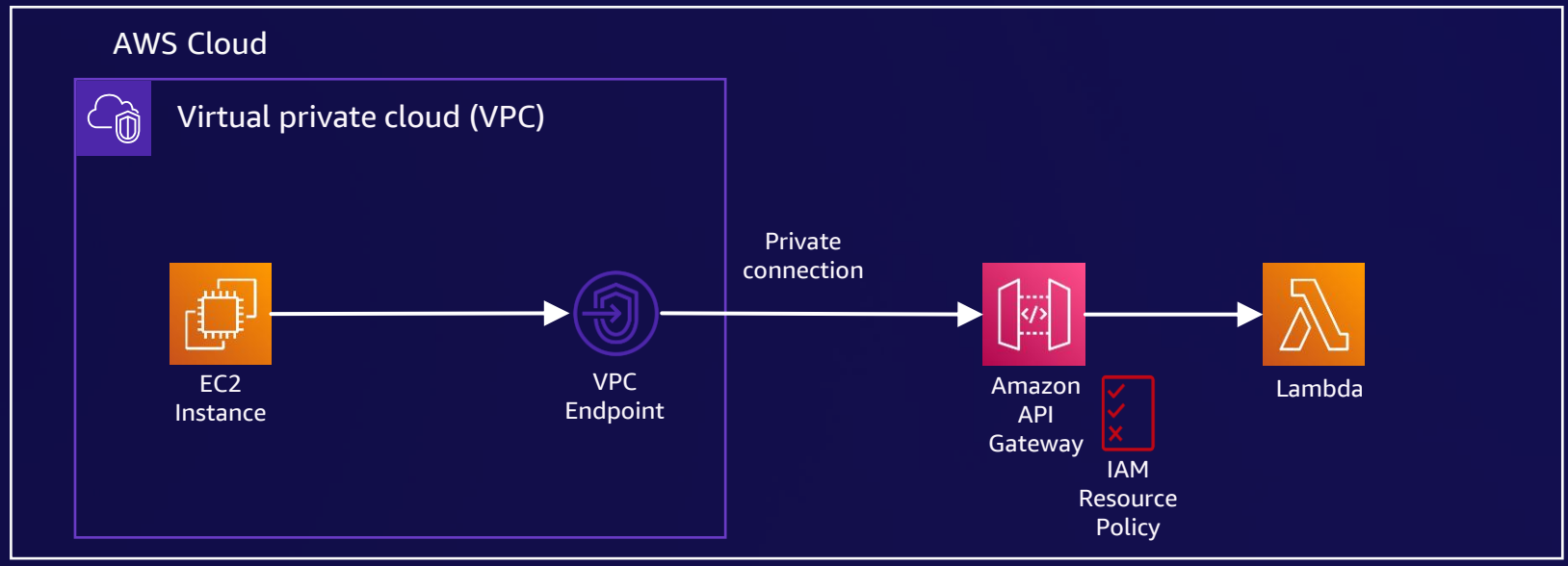

**PRIVATE API ENDPOINT**

# Implement a strong identity foundation

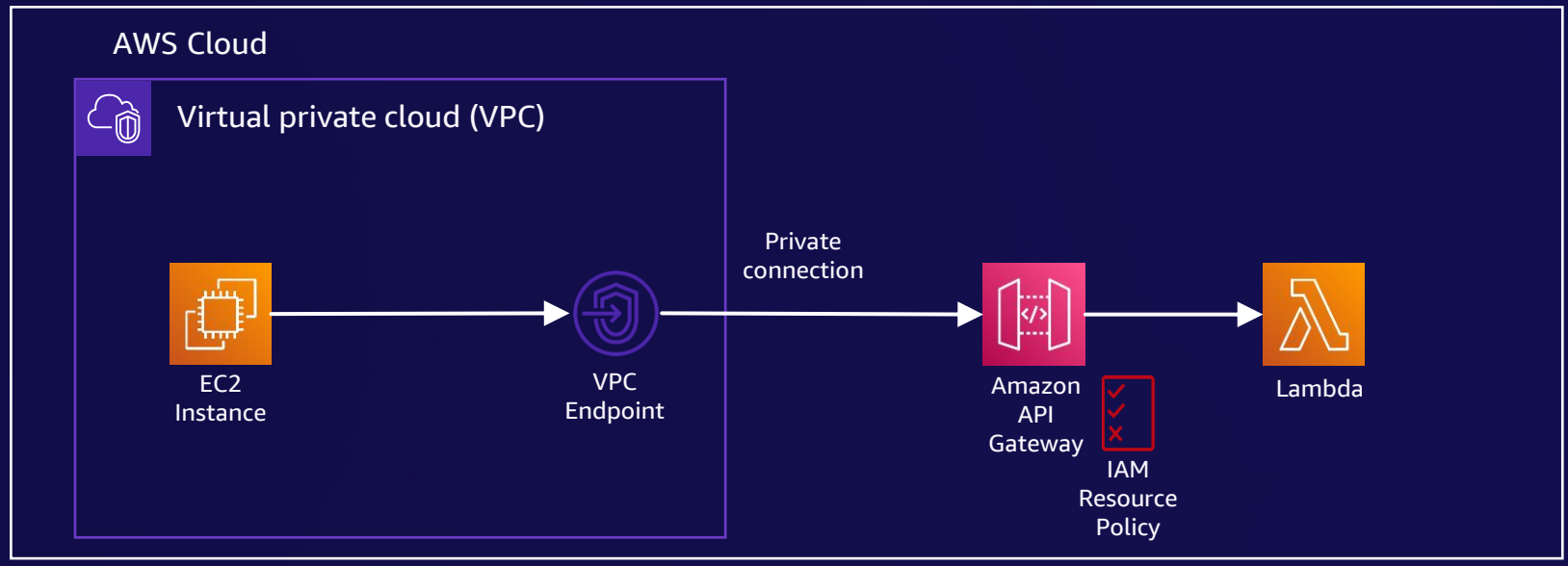

## PRIVATE API ENDPOINT

# Implement a strong identity foundation


## PRIVATE API ENDPOINT

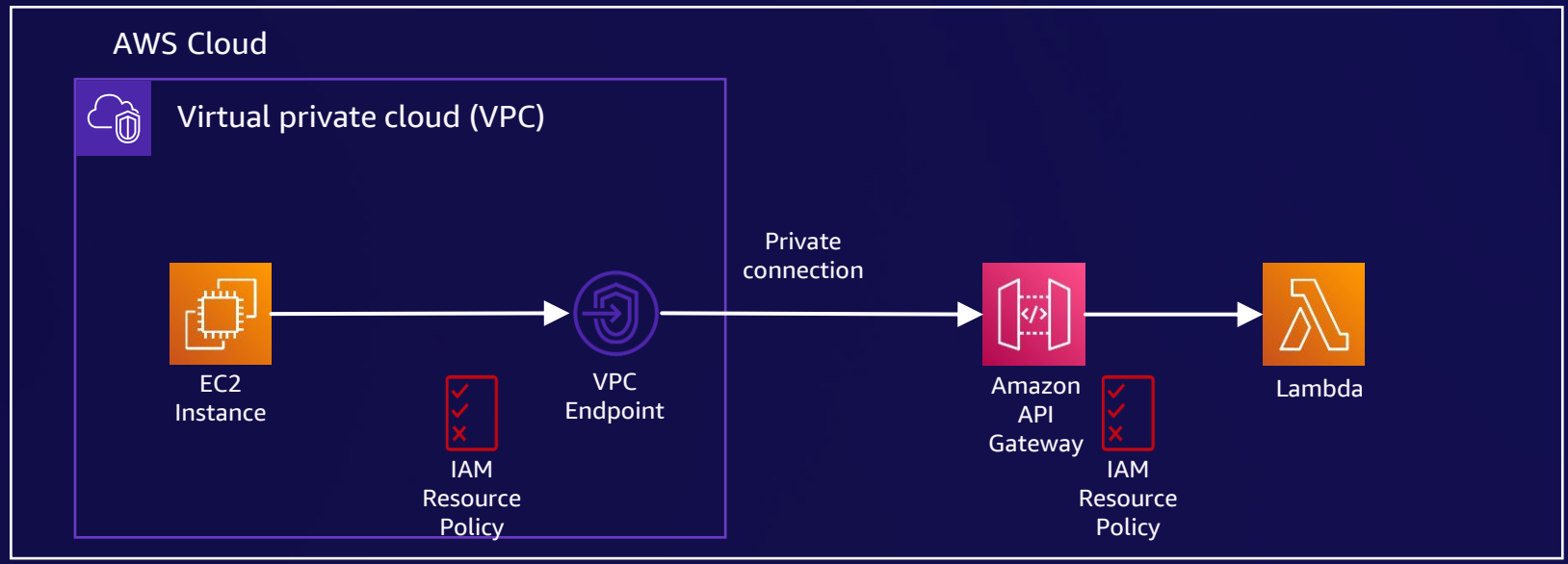
# Implement a strong identity foundation


```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "arn:aws:iam::account-id-2:role/developer",
            "Action": "execute-api:Invoke",
            "Resource": "execute-api:/dev/GET/pets"
        }
    ]
}
```

## PRIVATE API ENDPOINT

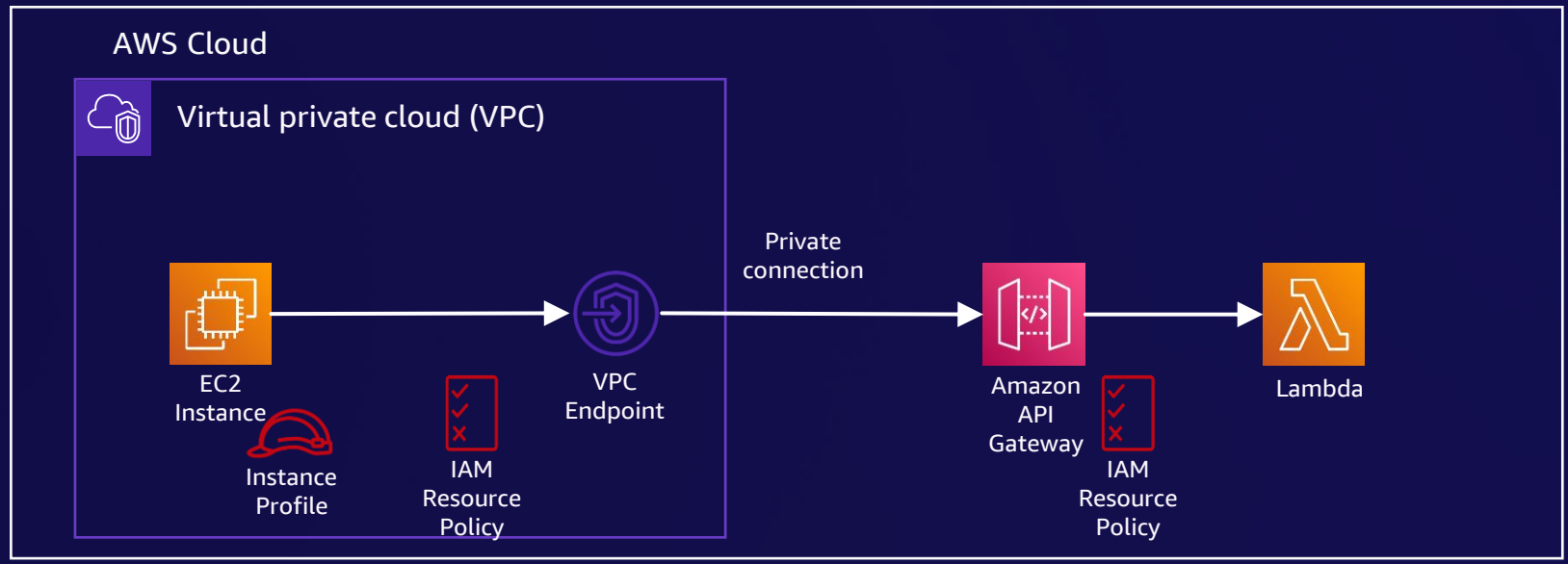
# Implement a strong identity foundation


```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "arn:aws:iam::account-id-2:role/developer",
            "Action": "execute-api:Invoke",
            "Resource": "execute-api:/dev/GET/pets"
        }
    ]
}
```

## PRIVATE API ENDPOINT

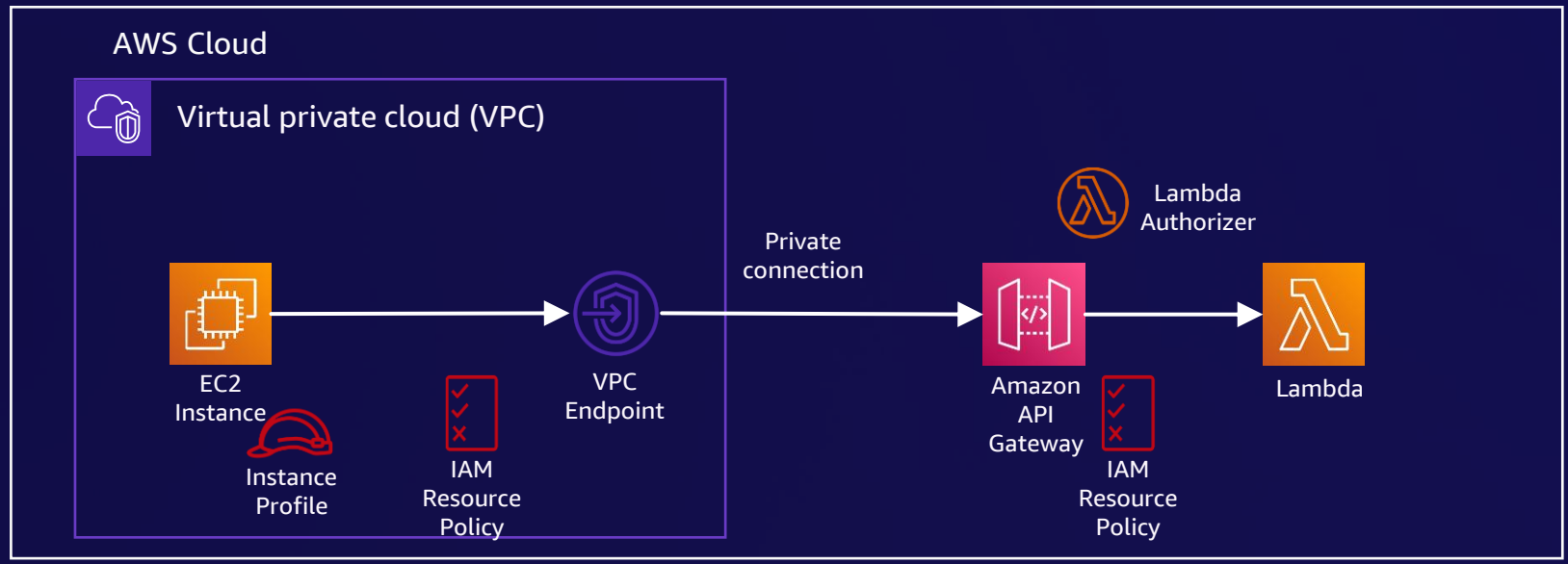
# Implement a strong identity foundation


```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "arn:aws:iam::account-id-2:role/developer",
            "Action": "execute-api:Invoke",
            "Resource": "execute-api:/dev/GET/pets"
        }
    ]
}
```

## PRIVATE API ENDPOINT

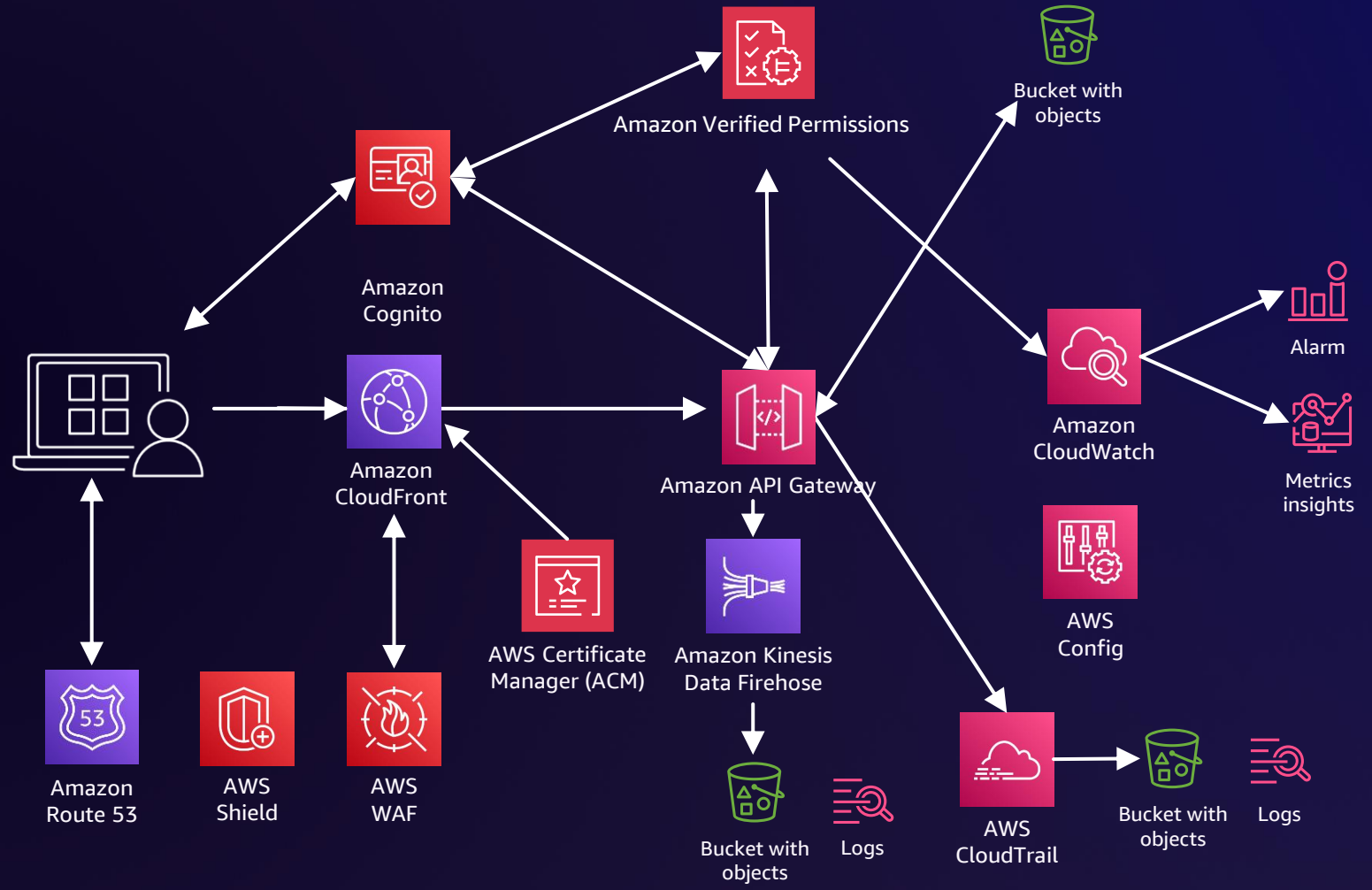
# Implement a strong identity foundation


```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "arn:aws:iam::account-id-2:role/developer",
            "Action": "execute-api:Invoke",
            "Resource": "execute-api:/dev/GET/pets"
        }
    ]
}
```

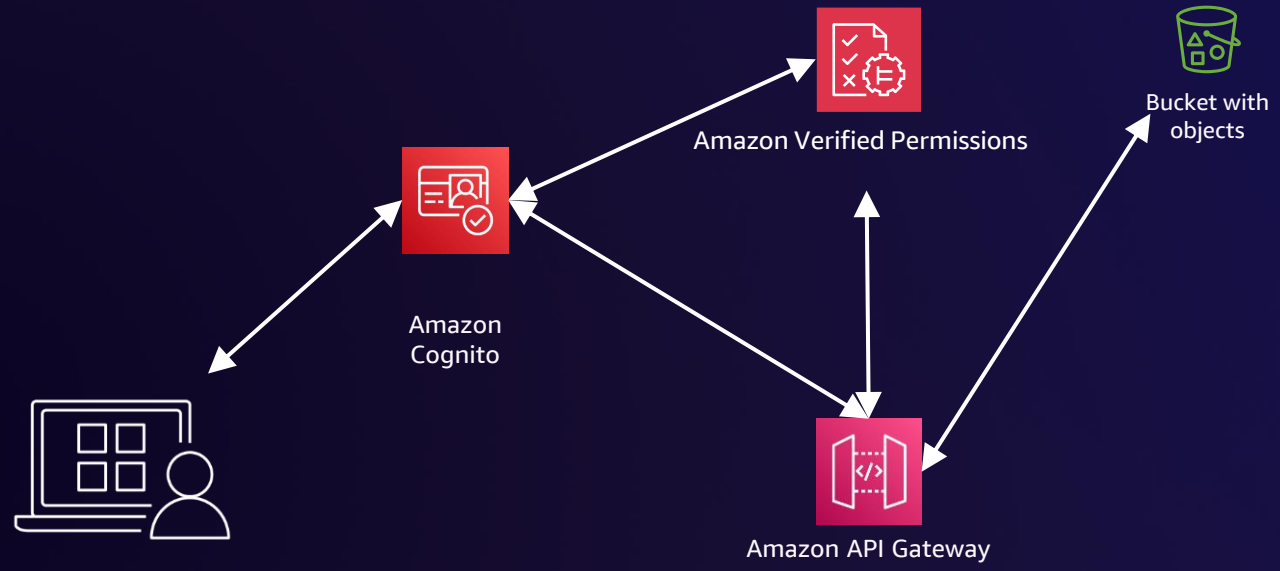
## PRIVATE API ENDPOINT

# Implement a strong identity foundation

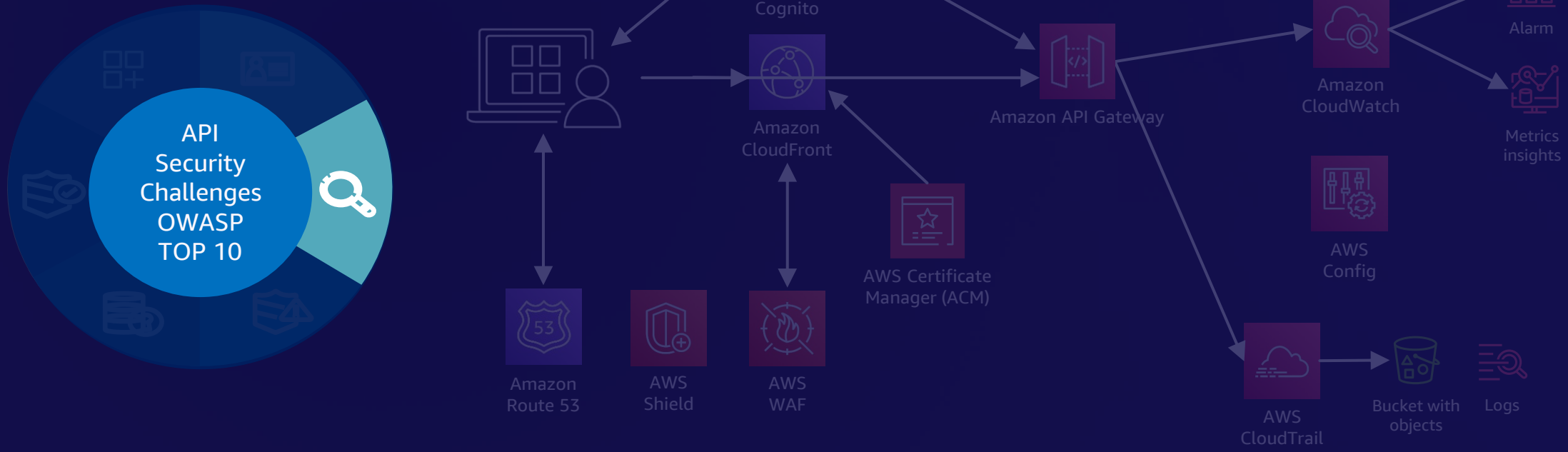

## PUBLIC API ENDPOINT

# Implement a strong identity foundation

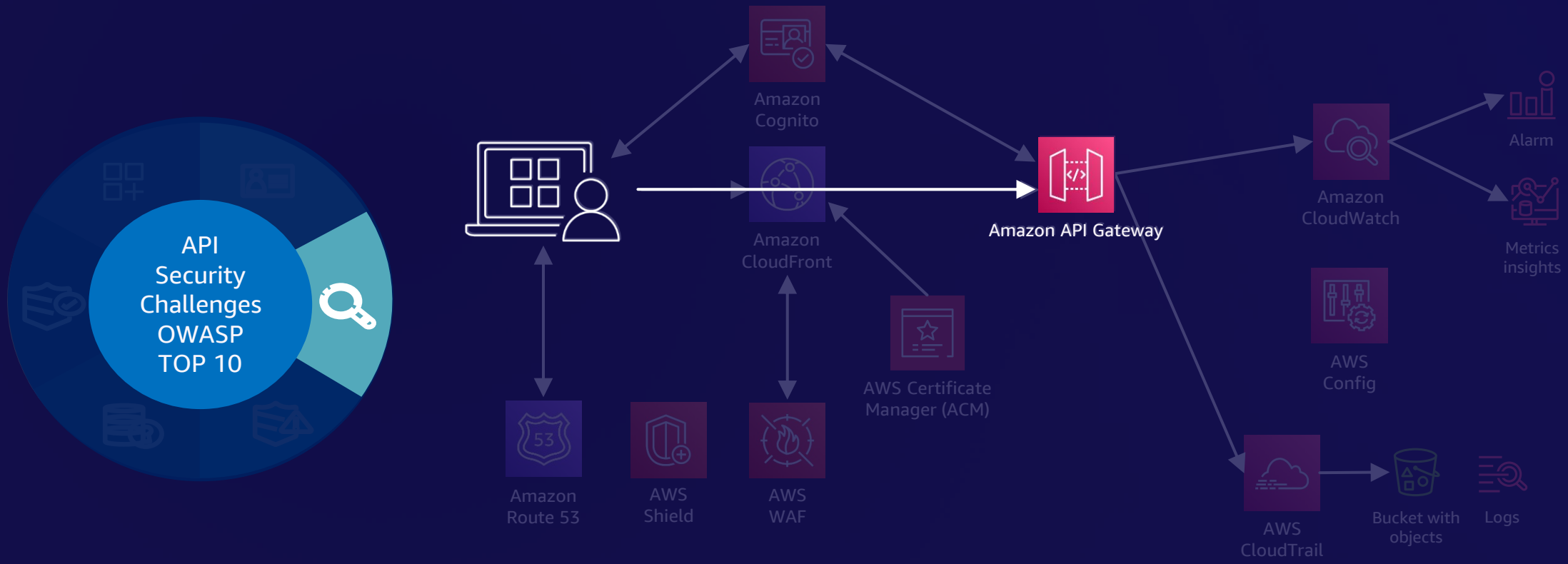

## PUBLIC API ENDPOINT

# Enable traceability



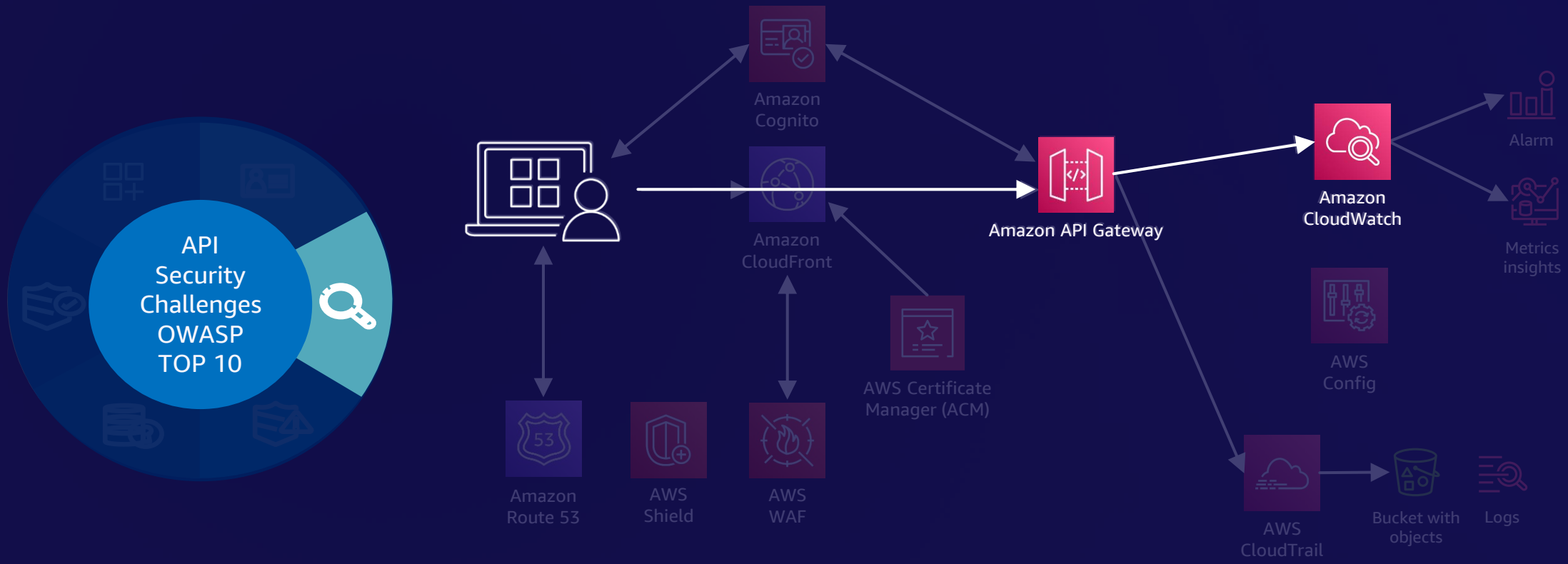

## PUBLIC API ENDPOINT

# Enable traceability

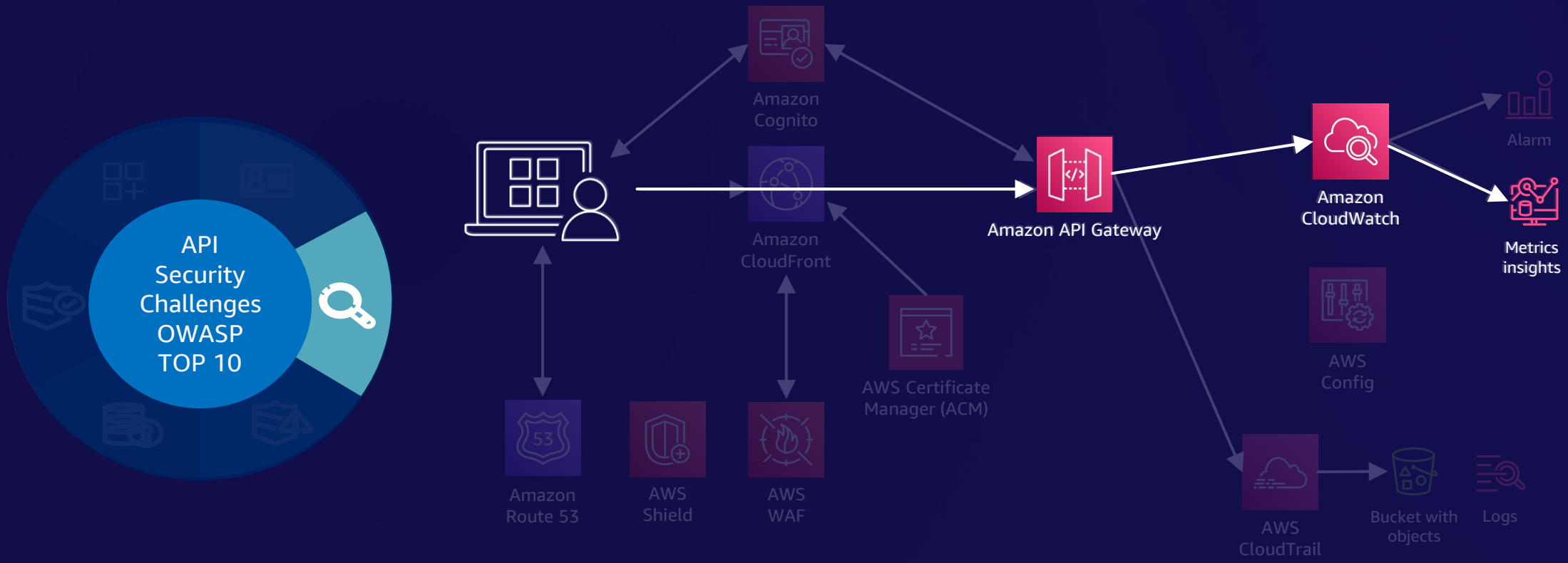



## PUBLIC API ENDPOINT

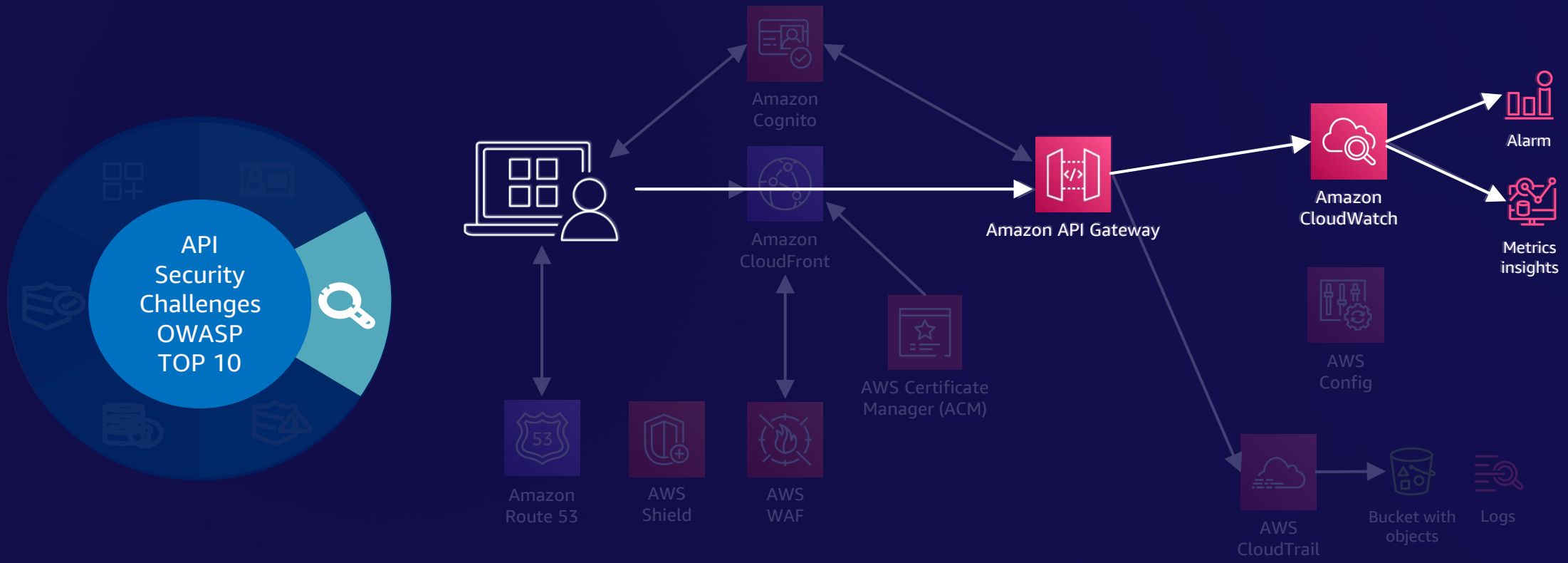
# Enable traceability



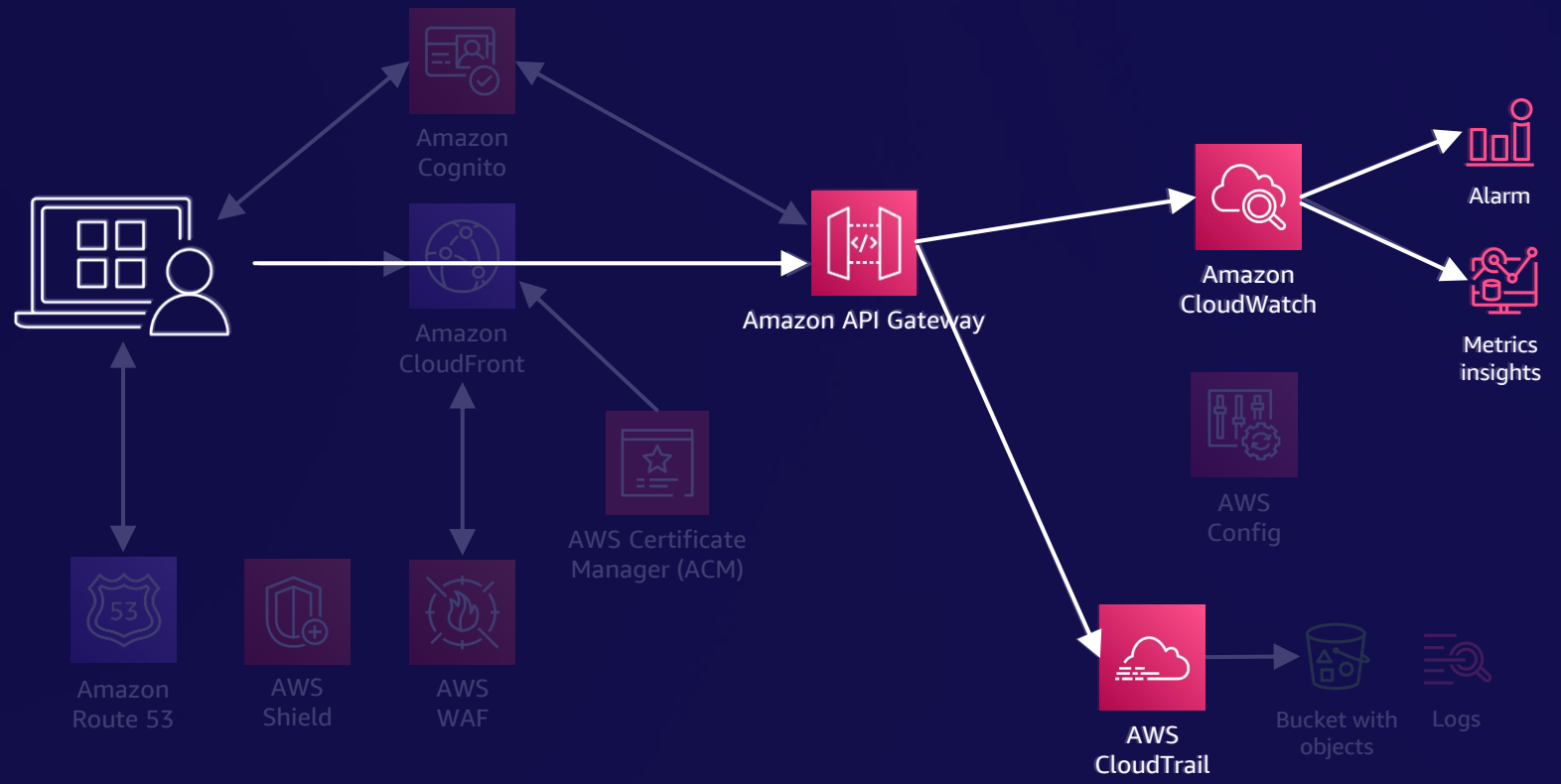
# Enable traceability



# Enable traceability

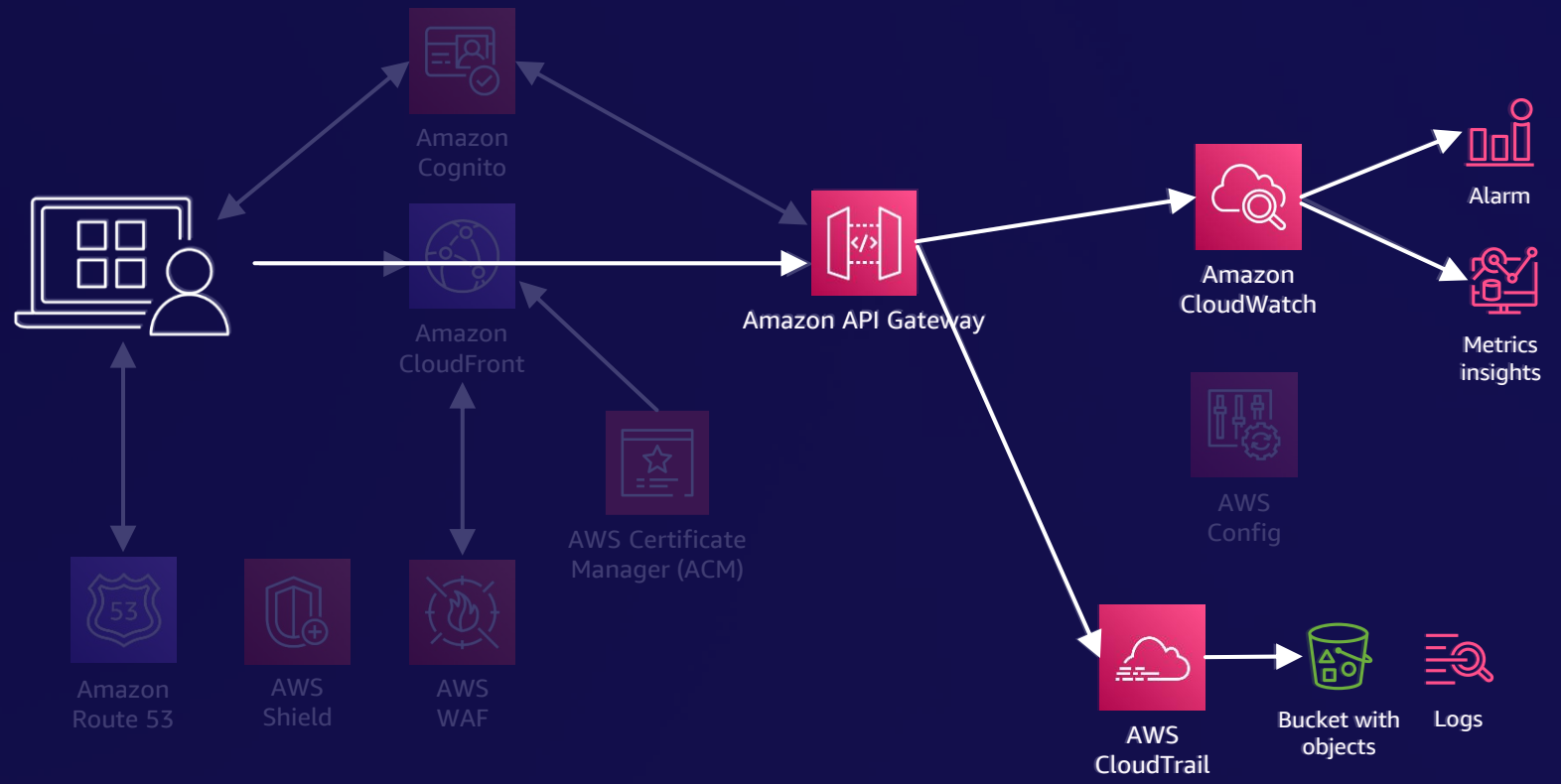


# Enable traceability

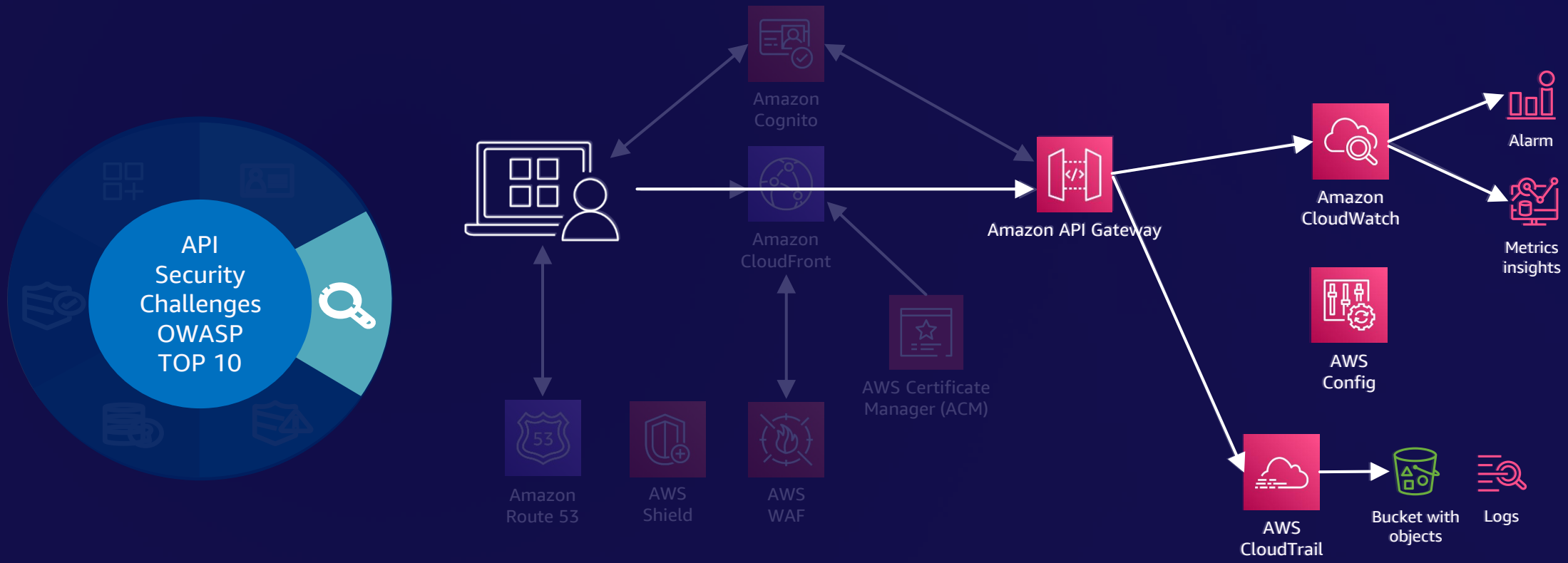

## PUBLIC API ENDPOINT

# Enable traceability

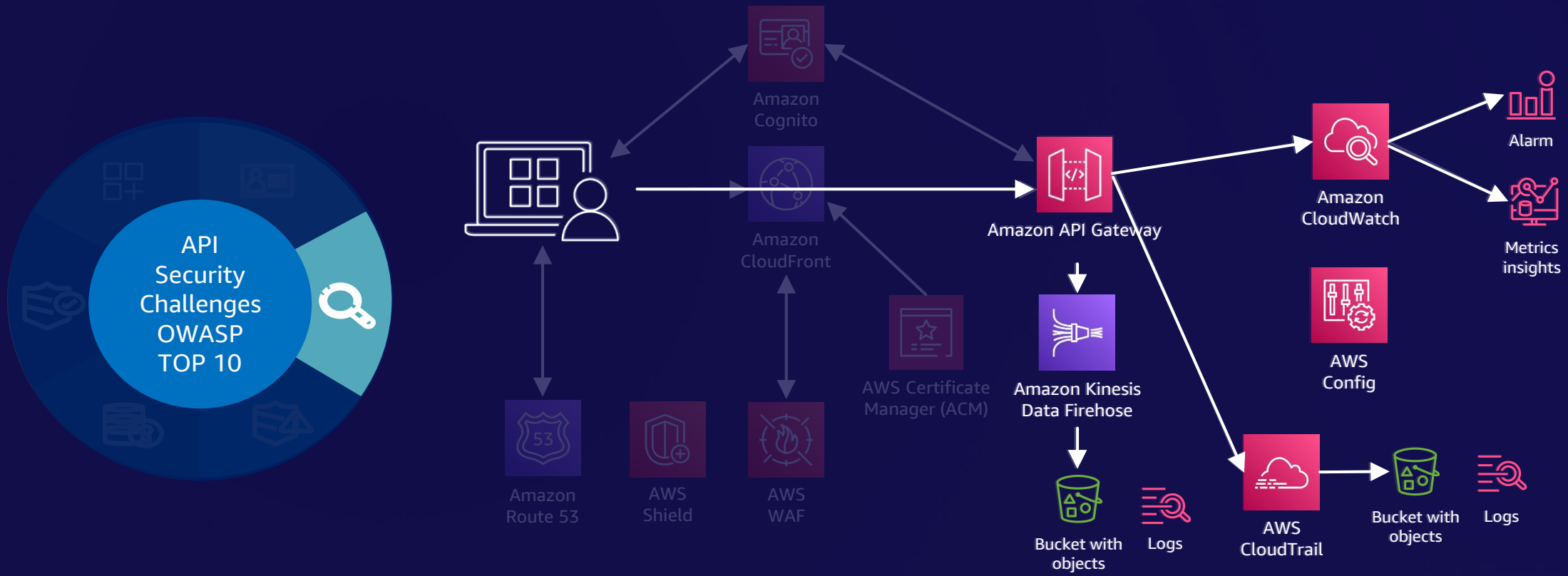




## PUBLIC API ENDPOINT

# Enable traceability

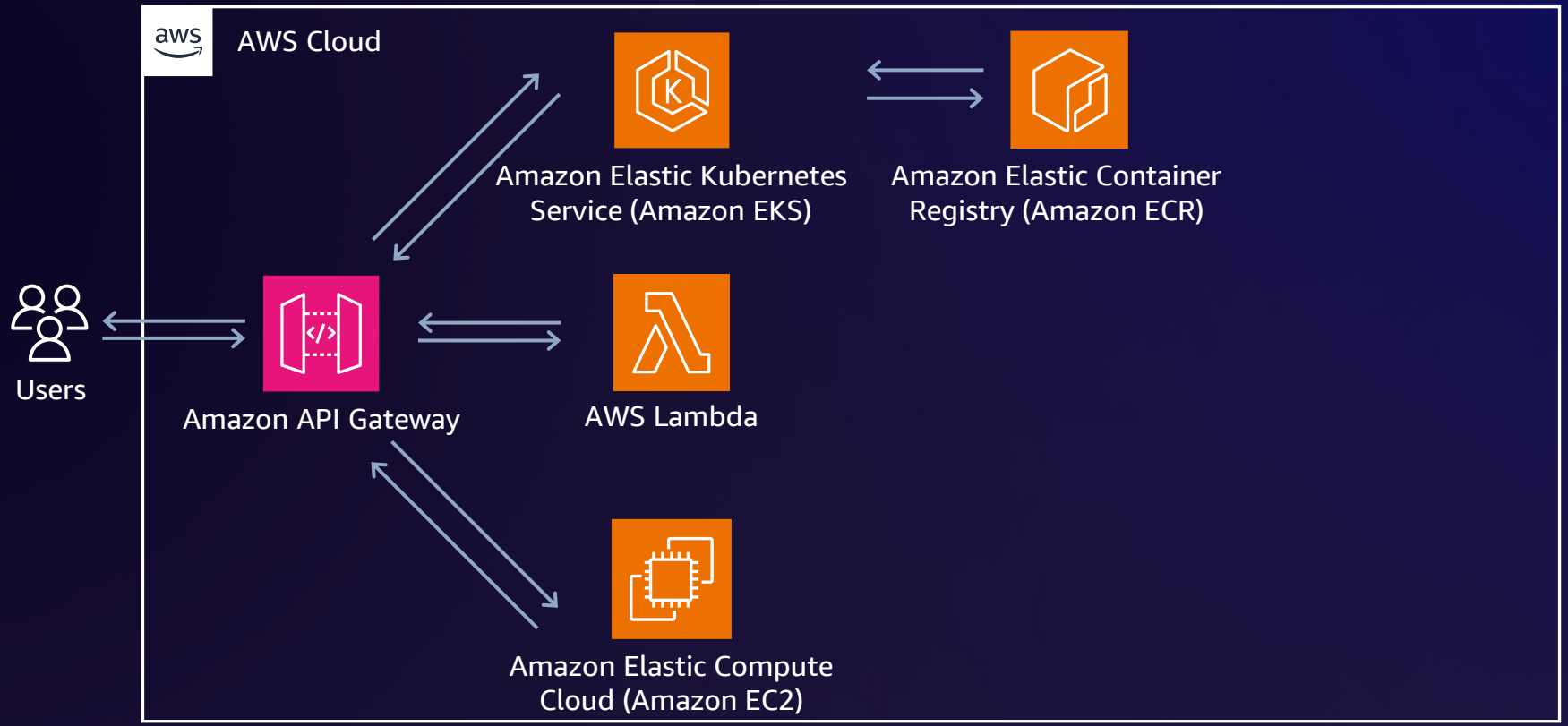


# Enable traceability

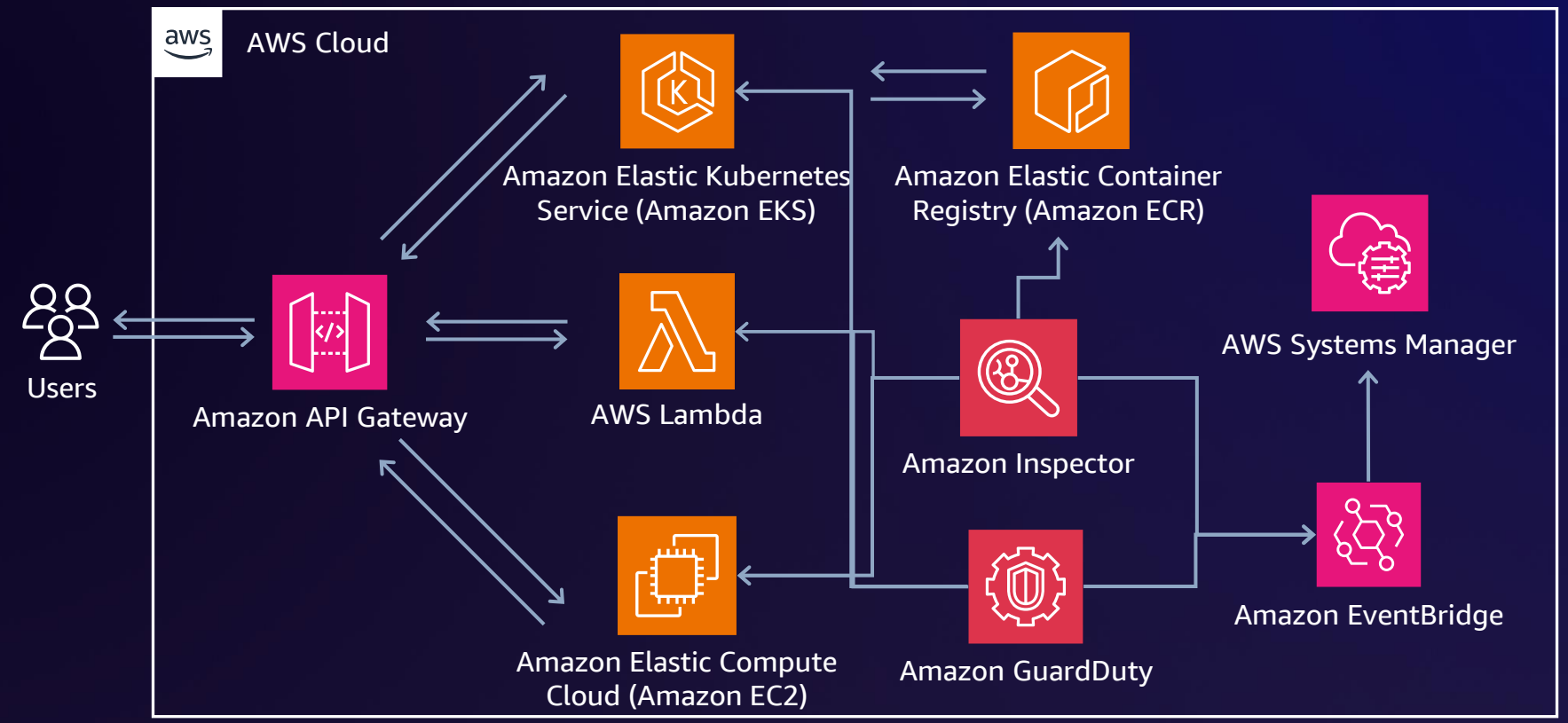



## PUBLIC API ENDPOINT

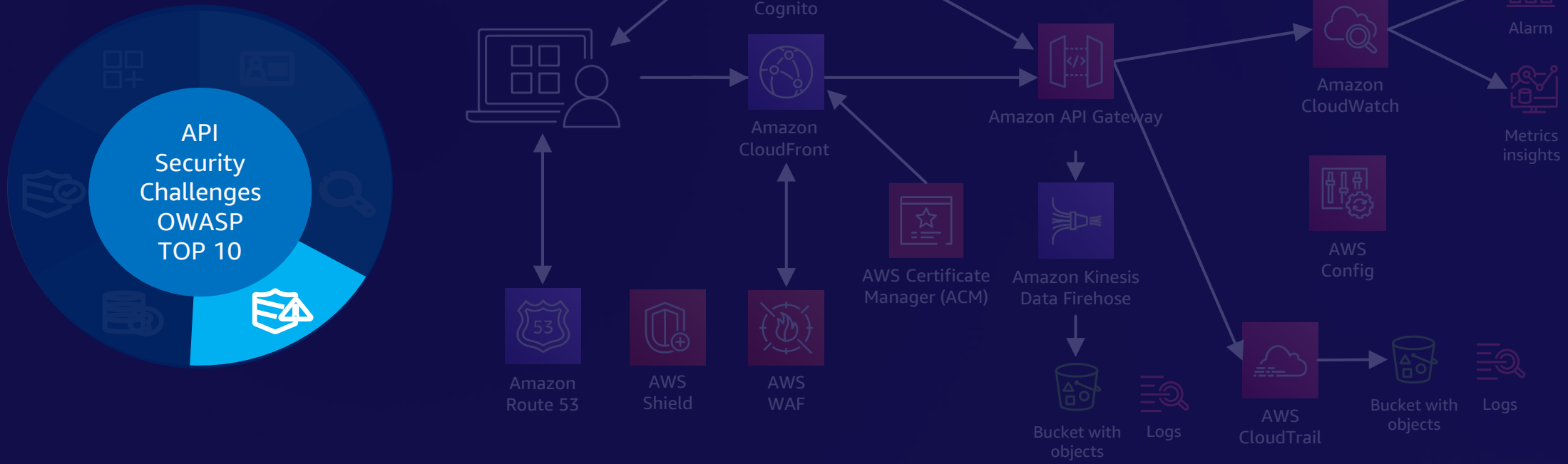
# Enable traceability

# Enable traceability

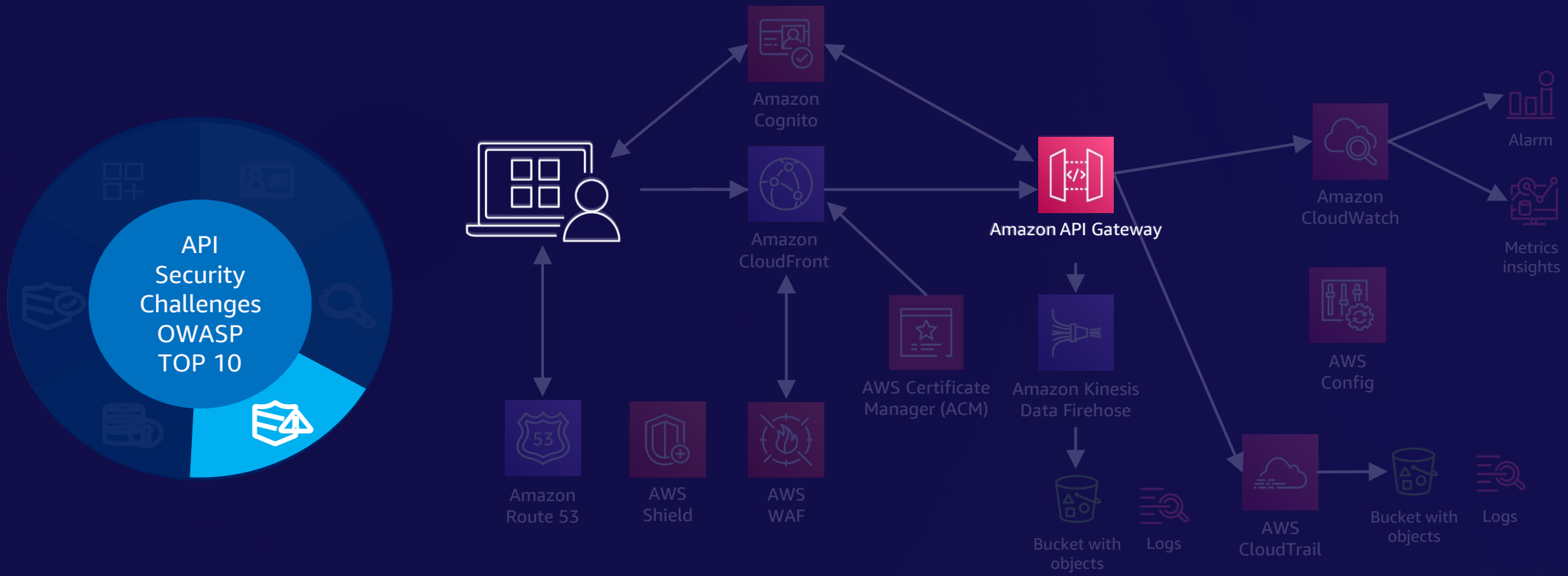



# Apply security at all layers



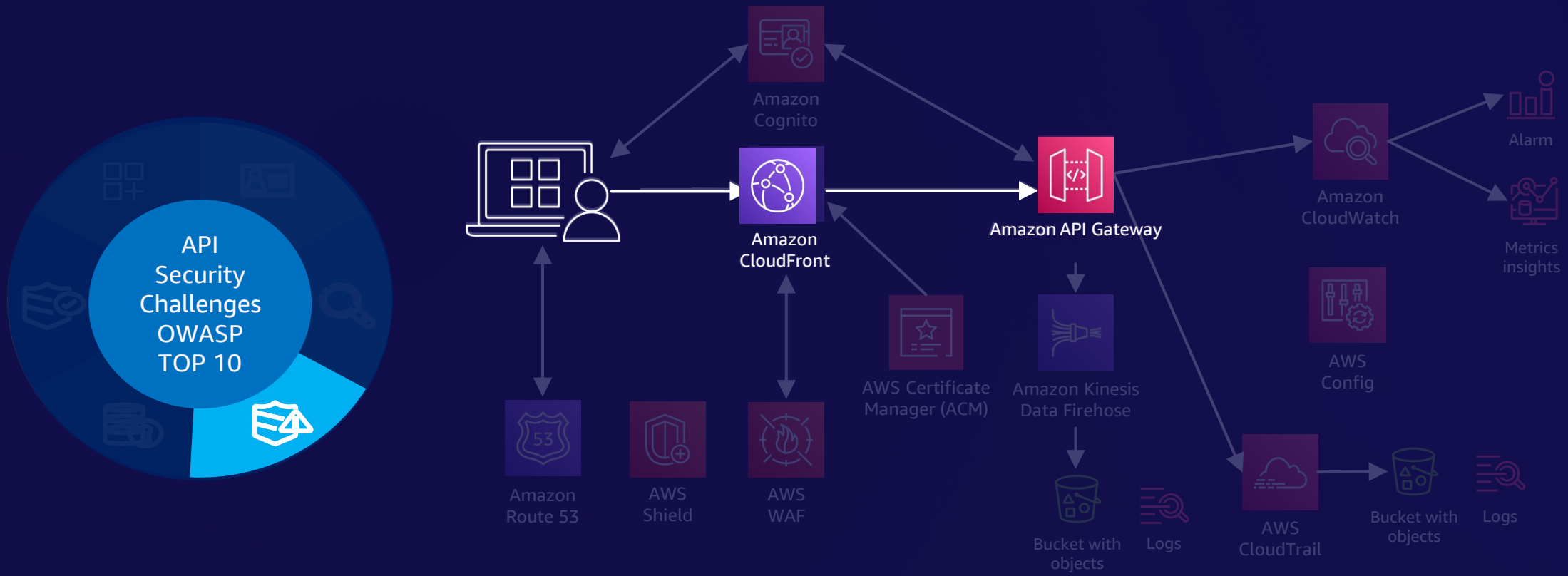

## PUBLIC API ENDPOINT

# Apply security at all layers



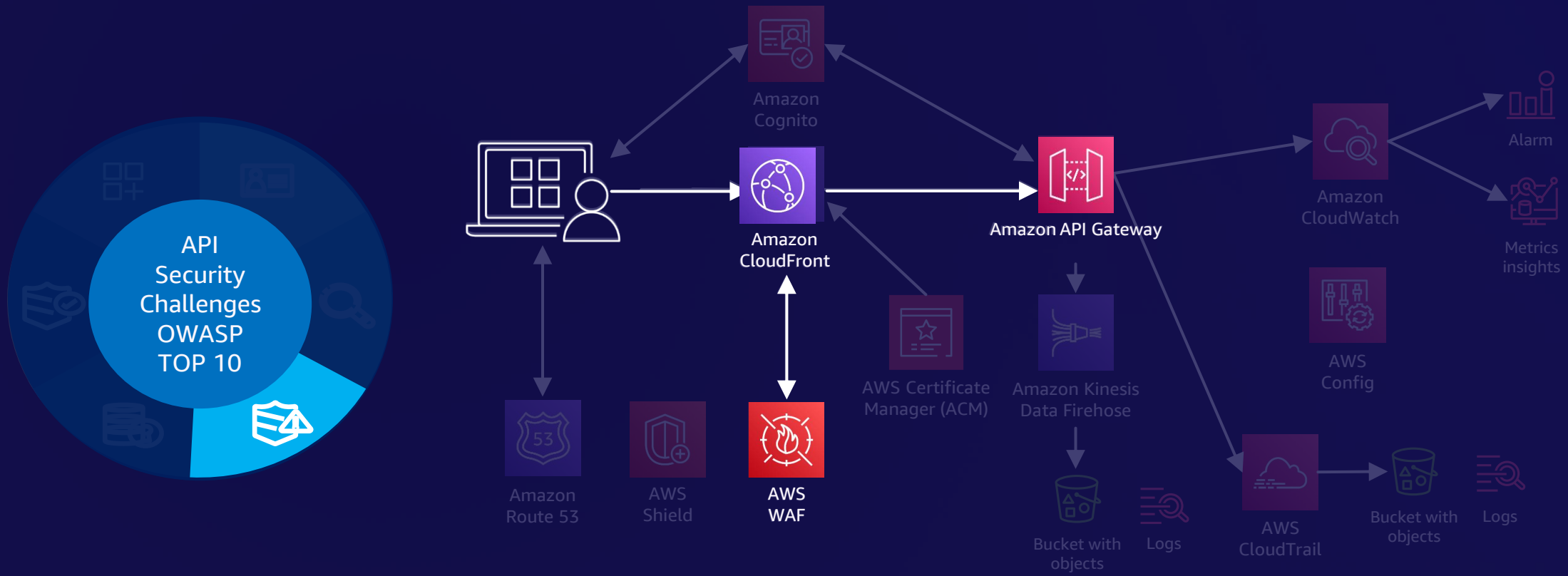

## PUBLIC API ENDPOINT

# Apply security at all layers



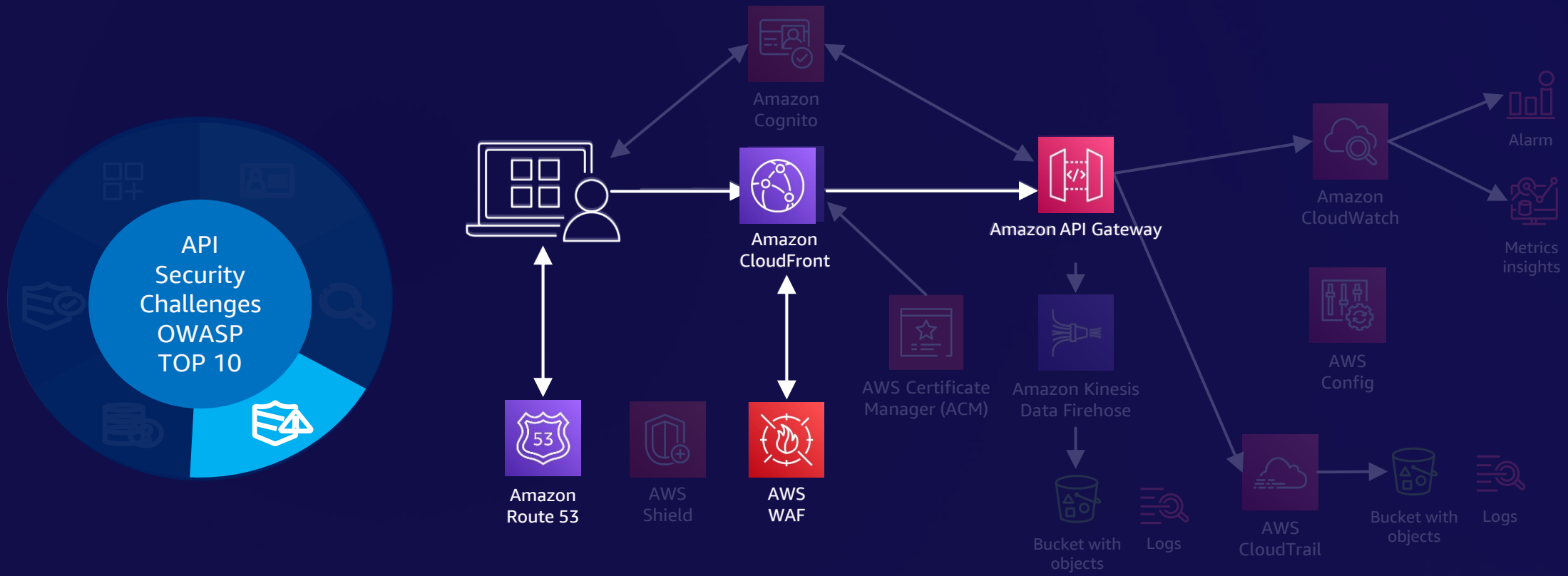

## PUBLIC API ENDPOINT

# Apply security at all layers



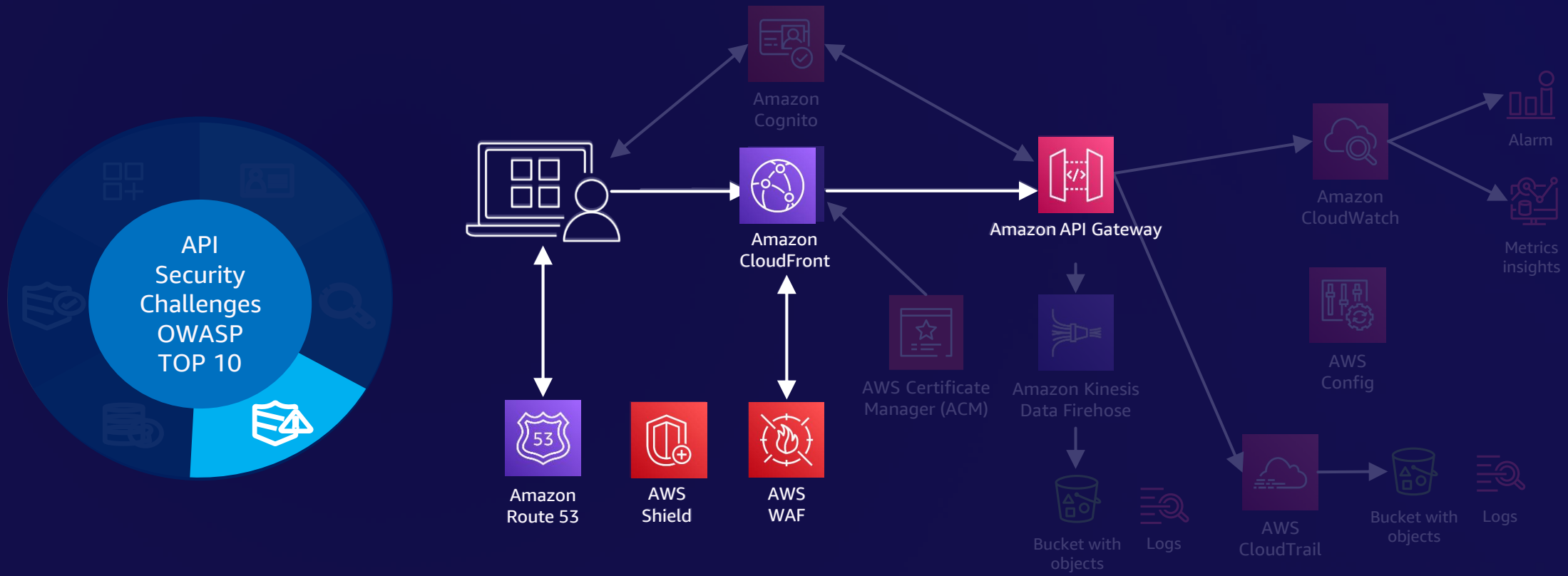

## PUBLIC API ENDPOINT

# Apply security at all layers




## PUBLIC API ENDPOINT

# Apply security at all layers




## PUBLIC API ENDPOINT

# Twilio WAF as a service (WaaS) solution

# AWS WAF (Web Application Firewall)

PROTECT YOUR WEB APPLICATIONS AND APIS AGAINST COMMON VULNERABILITIES, TARGETED ATTACKS, AND BOTNETS

## Multi-layered security controls

Protect against sophisticated attacks

## Customizable security

Powerful rule customizations

## Low operational overhead

Fully managed service with ready-to-use, built-in rules

## Frictionless set up

No application changes required

# AWS Firewall Manager

CENTRALLY CONFIGURE AND MANAGE FIREWALL POLICIES

Central management of firewalls

Configure baseline security policies

Detect non-compliance and remediate

# Amazon OpenSearch Service

SECURELY UNLOCK REAL-TIME SEARCH, MONITORING, AND ANALYSIS OF SECURITY AND OPERATIONAL DATA

Managed

Secure

Cost conscious

Observability

# About Twilio

Born in the cloud in 2008 to revolutionize communications for developers

Leading CPaaS provider, powering startups and the greater global digital economy

Generates over $4 billion in annual revenue, facilitating 1.7 trillion interactions for 306,000+ customers

Twilio has been a pioneer in the API-first approach

# Enterprise challenges managing AWS WAFs

Growing pains

Resource coverage

Poor observability

Increased time to remediate

# How Twilio manages AWS WAF globally

**01**

## Globally inherited AWS WAF rules

Core rules are applied to all resources within the organization

**02**

## Tag-driven policies

AWS FMS policies are associated with Tags for **First Rule Group**

**03**

## Security maintained safeguards

**Last rule groups** are managed by the Security Teams and serve as a safety net

First rule groups

**Managed by the Engineering Team**

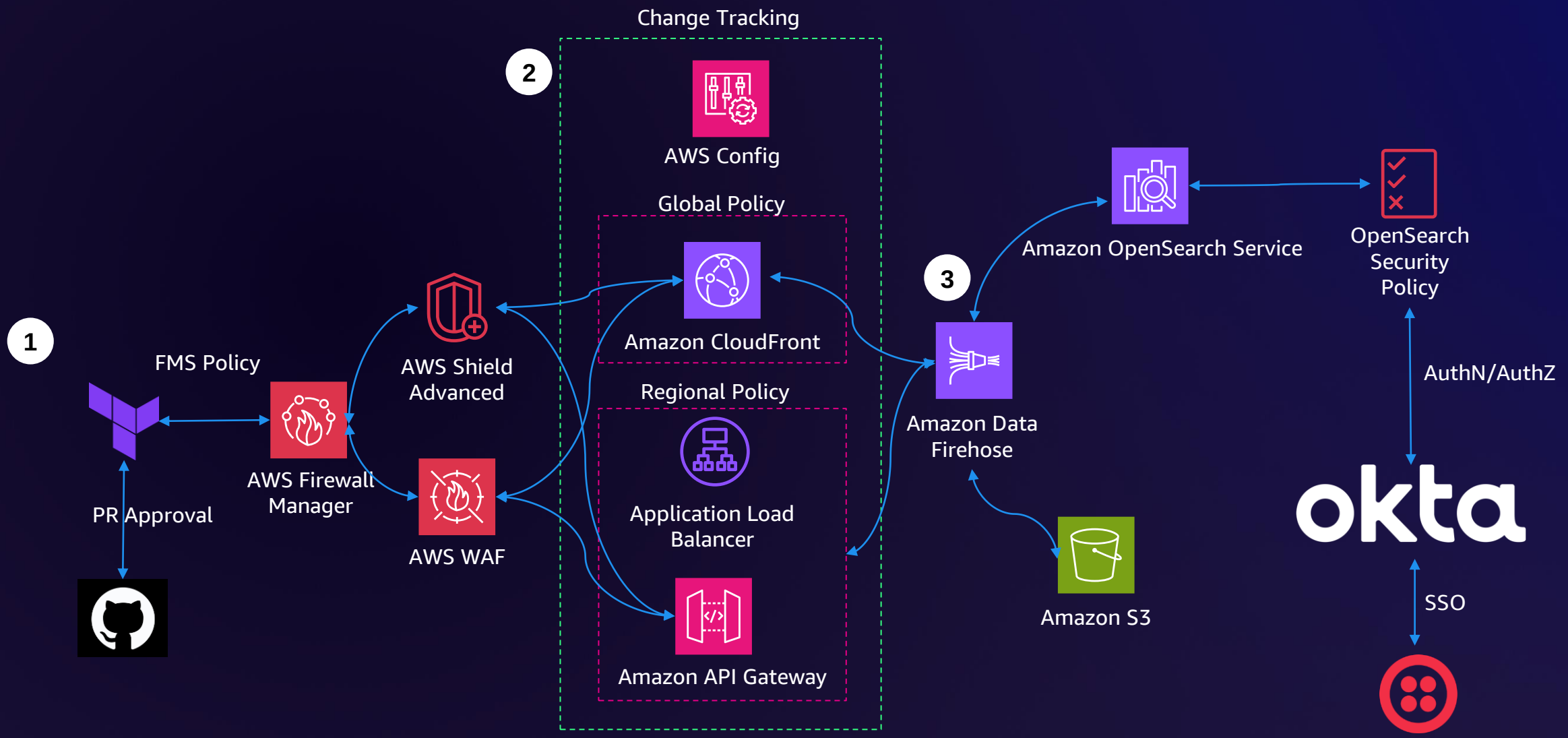
| Order | Rule group name | Capacity |
|---|---|---|
| 1 | WaaS_TWILIO-PUBLIC-SERVICES_RULE_GROUP_BLOCK | 550 |
| 2 | AWS Core rule set | 700 |
| 3 | AWS SQL database | 200 |

Last rule groups

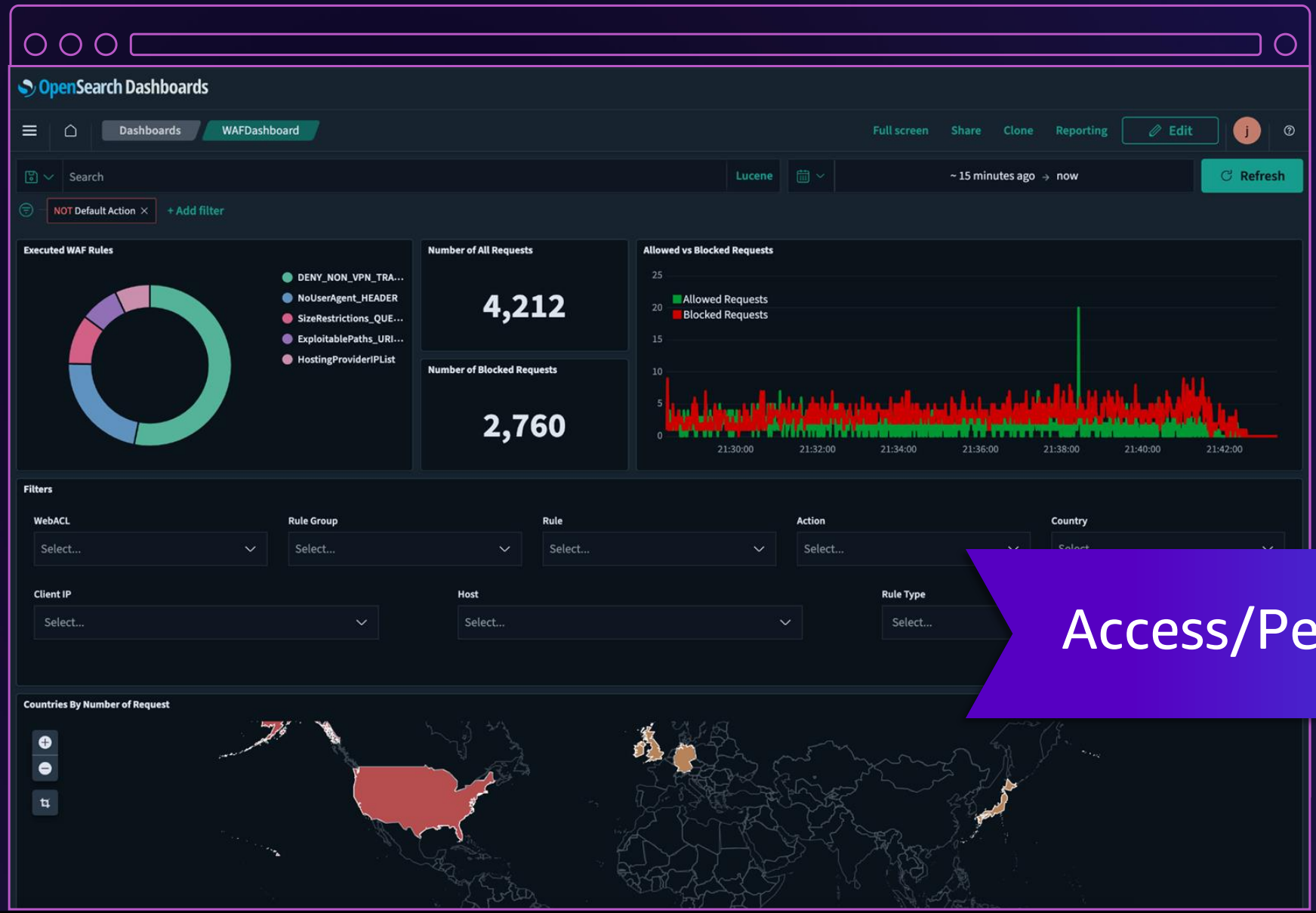
**Managed by the Security Team**

| Order | Rule group name | Capacity |
|---|---|---|
| 1 | WaaS_BASE_SECURITY_REGIONAL_RULE_GROUP | 2500 |
| 2 | AWS Amazon IP reputation list | 25 |
| 3 | AWS Anonymous IP list | 50 |
| 4 | AWS Core rule set | 700 |
| 5 | AWS Known bad inputs | 200 |

# WaaS architecture



# Using AWS OpenSearch for Visibility




Access/Permissions tied to IDP

# AWS WAF at scale for enterprise

Centralized through self-service

OpenSearch dashboards

Auto-enroll new AWS resources

Decreased time to remediate

# Key takeaways

Use a framework

Start with the developer

Automate as appropriate

Ensure observability

# Call to action

OWASP API Security Challenges

Well-Architected Security Pillar

Well-Architected Hands-on Labs