Secure your APIs the AWS Well-Architected Way

Jhalak Modi
Solutions Architect

Thank You to Our Sponsors and Hosts!

AppSec NZ
appsec.org.nz

OWASP NEW ZEALAND
owasp.org.nz

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

BASTION
SECURITY GROUP

DATACOM

aws

84.

PentesterLab

plexure

VERACODE

Without them, this Conference couldn't happen.

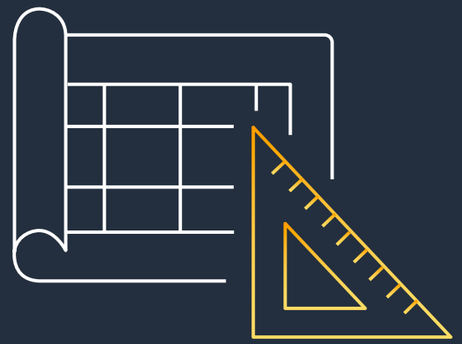
What to expect from this session:

- Introduction to AWS Well Architected Framework
- Common API security challenges
- AWS Well-Architected Framework security design principles
- How to address API security challenges with the AWS Well-Architected Framework
- Takeaways

Well-Architected Framework

What is the AWS Well-Architected Framework?

Pillars & Lenses



Design principles

Questions

Best Practices

Why use the AWS Well-Architected Framework?

Build and deploy faster

Lower or mitigate risks

Make informed decisions

Learn AWS best practices

AWS Well-Architected Framework continuous improvement

Pillars of AWS Well-Architected

Operational Excellence

Security

Reliability

Performance Efficiency

Cost Optimization

Sustainability

AWS Well-Architected Framework security pillar

Security design principle

Implementation guide

Security design principles

- Implement a strong identity foundation
- Maintain traceability
- Apply security at all layers
- Automate security best practices
- Protect data in transit and at rest
- Keep people away from data
- Prepare for security events

What is an API?

What is an API?

- Application programming language
- Connect apps together
- Interface for your code
- API is a messenger that takes the request and gives back the response

Analogy: The cooks in the kitchen are the back end, dining area for guests is the front end, and the Waiters are the APIs.

What is an API?

Example – AnyCompany

- Seems like one APP
- Multiple API are connected together
- Where the Puzzle pieces connects are the API

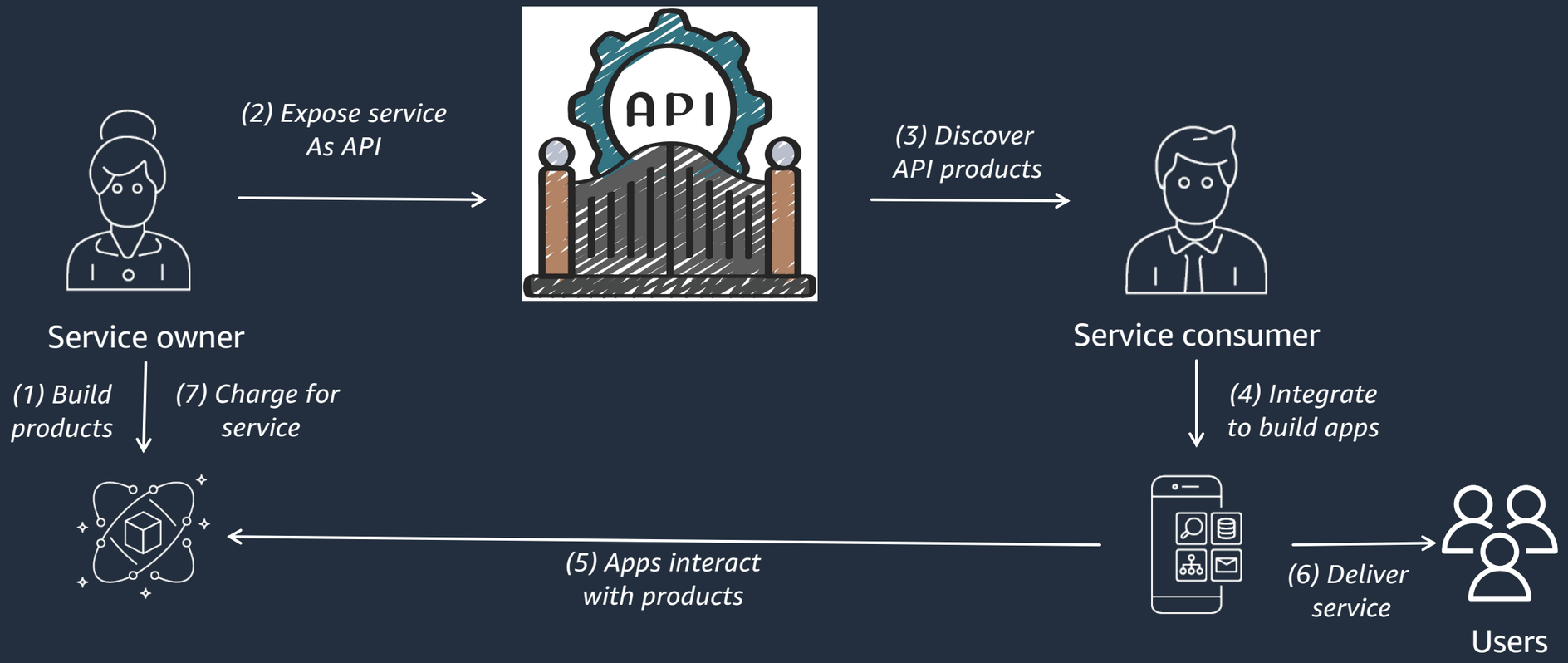
What is an API?

API definition – shape of the puzzle piece

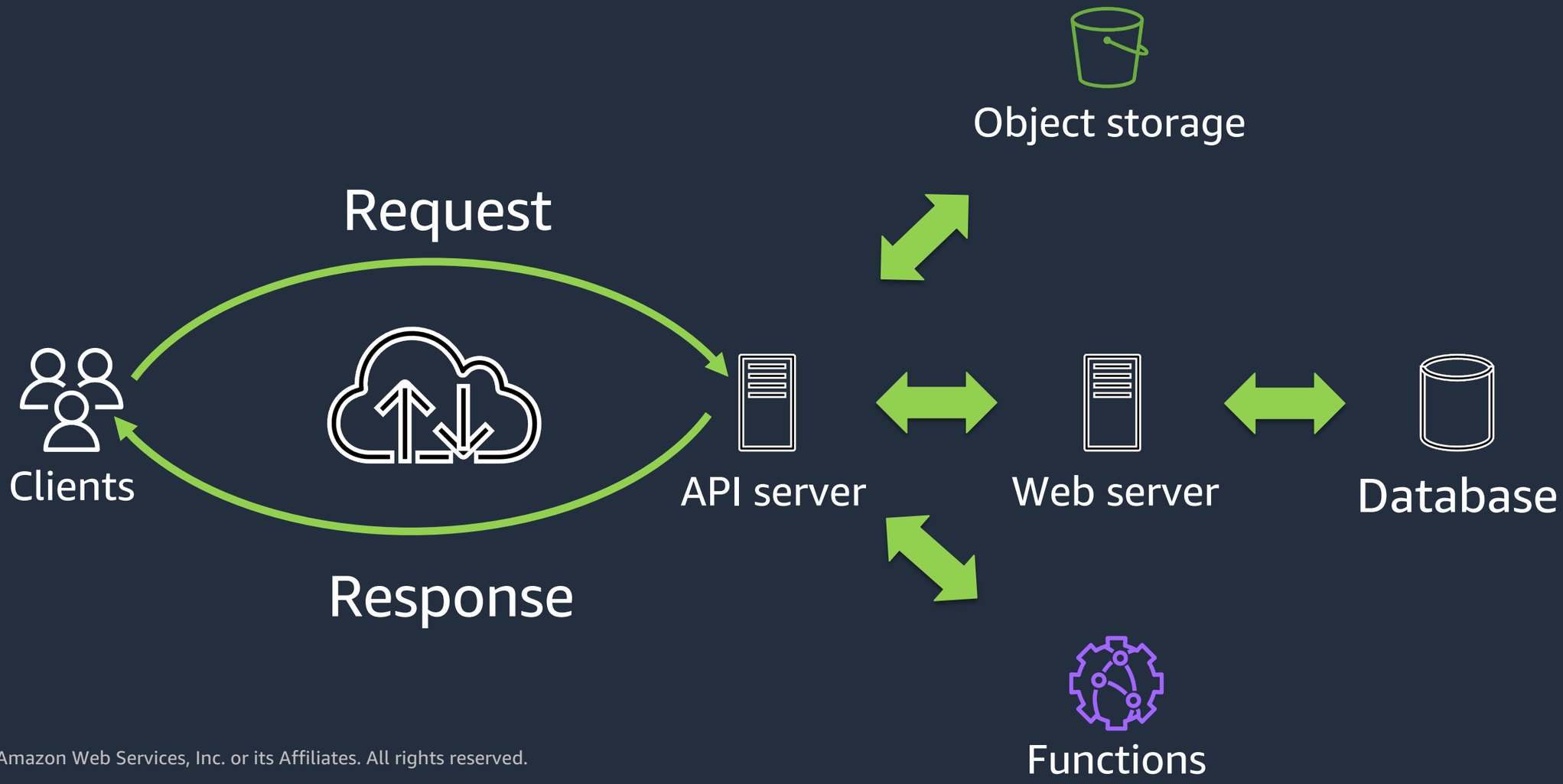
- Rating app
  - Rider
  - Driver
  - Stars

- Login app
  - Username
  - password

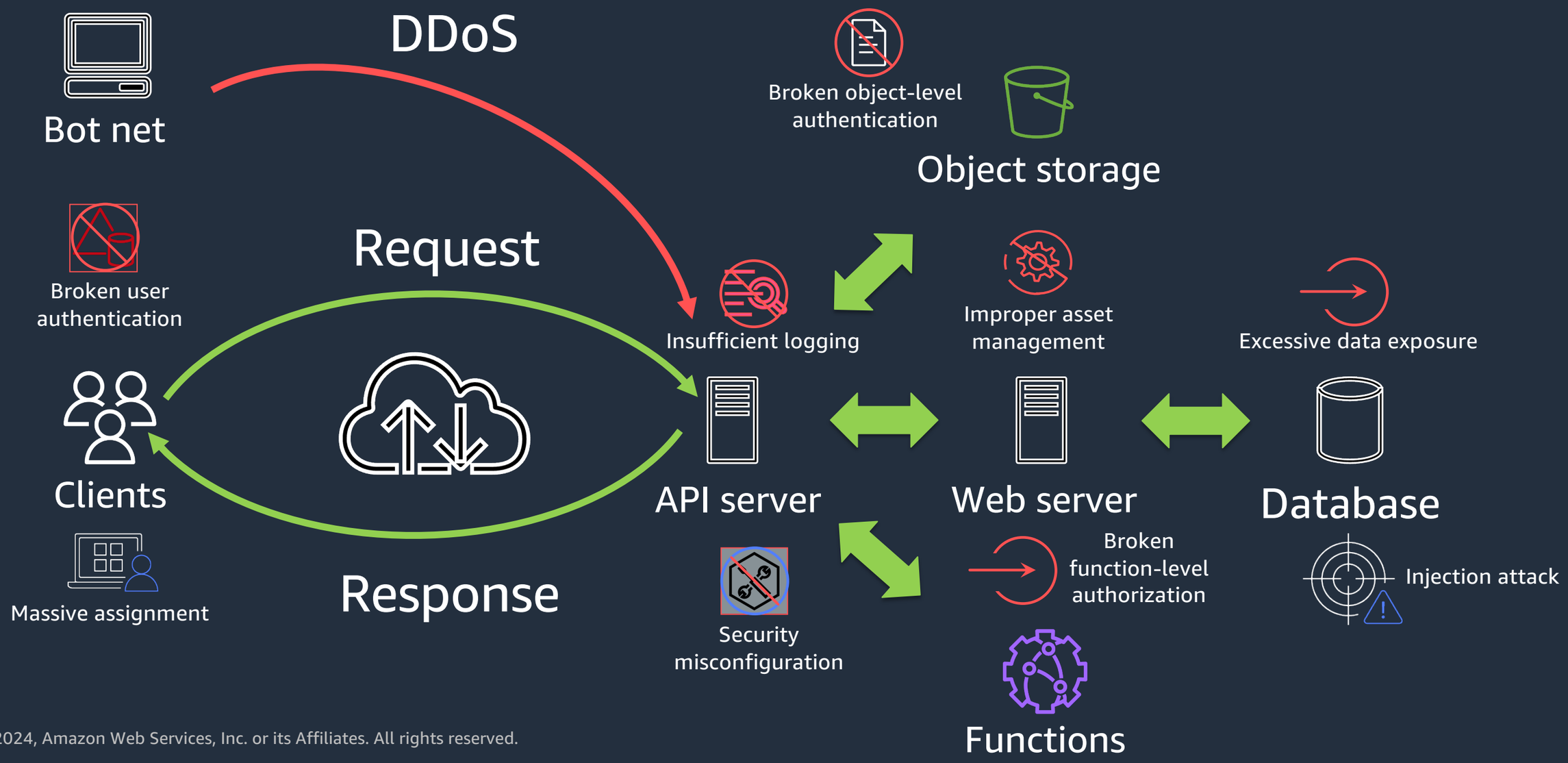
API as a product



Let’s look at real world example



Common API security challenges for enterprises



Common API security challenges for enterprises

Bot net

Broken object-level authentication

Broken user authentication

Insufficient logging

Improper asset management

Excessive data exposure

Massive assignment

Security misconfiguration

Broken function-level authorization

Injection attack

Common API security challenges for enterprises

Bot net

Broken object-level authentication

Broken user authentication

Insufficient logging

Improper asset management

Excessive data exposure

Massive assignment

Security misconfiguration

Broken function-level authorization

Injection attack

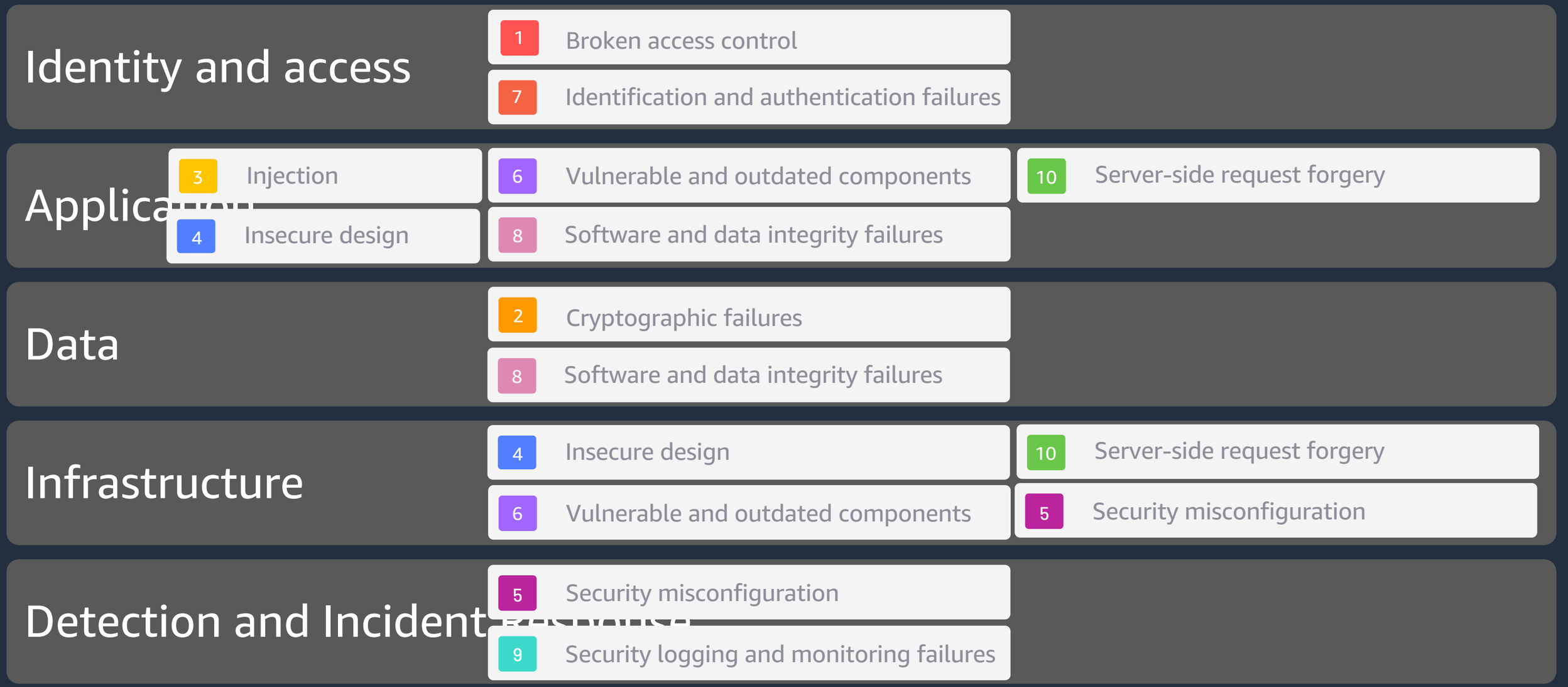
API security challenges
(OWASP Top 10)

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

OWASP top 10 mapped to security domains



Security best practices – Well Architected Way

- Use authentication and authorization mechanisms
- Application security
- Data encryption and integrity
- Infrastructure Protection and Denial of service
- Threat Detection and Incident Response

Security Best Practices

Security best practices – Well Architected Way

- Use authentication and authorization mechanisms
- Application security
- Data encryption and integrity
- Infrastructure Protection and Denial of service
- Threat Detection and Incident Response

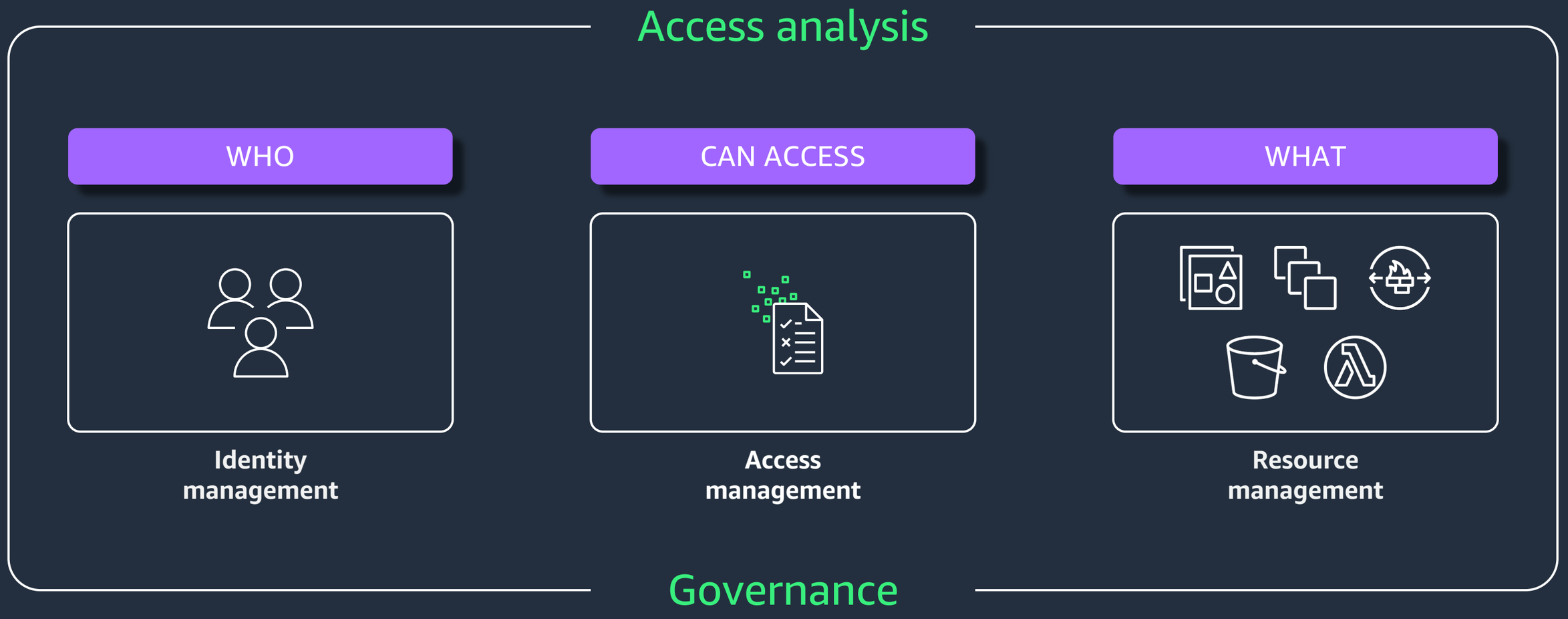
Best practice 1: Use authentication and authorization mechanisms

- Use appropriate authentication and authorization mechanisms
- Follow least-privilege model
- Take advantage of smaller, single purpose microservices (lambda functions)
- Store secrets securely
- Use multi factor authentications
- Log failed logins and delay logins

OWASP Serverless Top 10
S1:2021 Broken Access Control
S7:2021 Identification and authentication failures

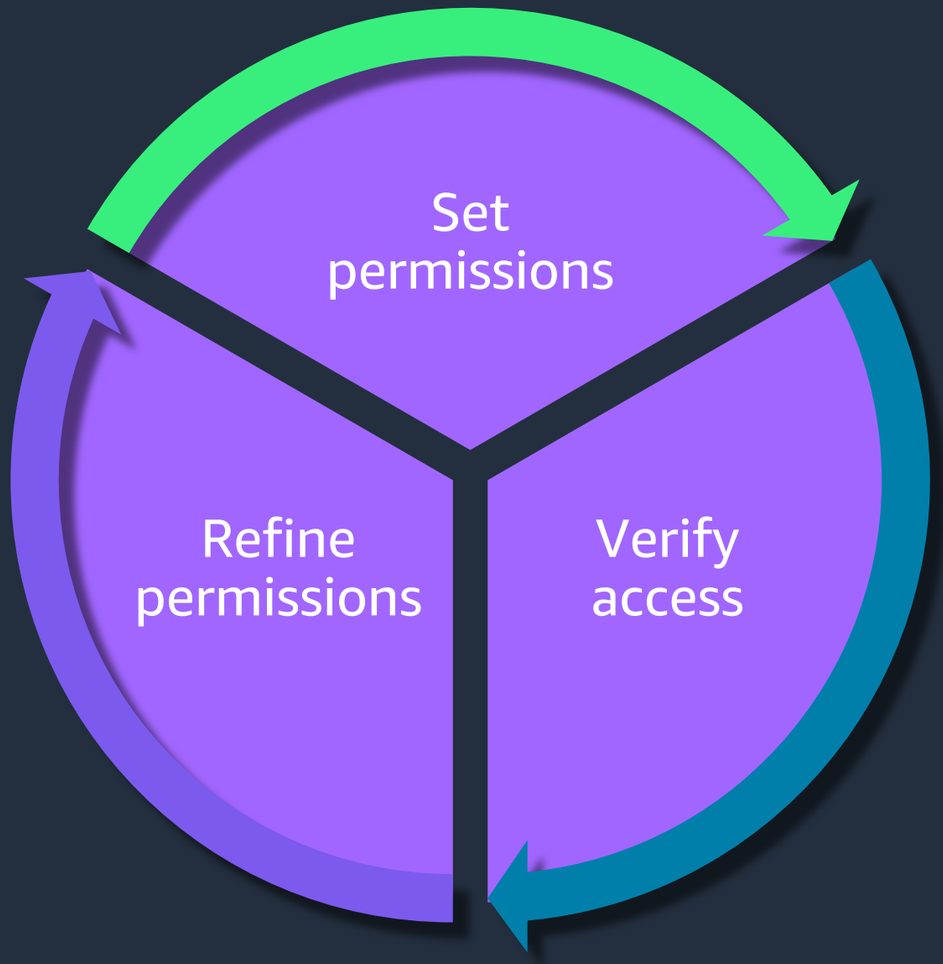
AWS Well-Architected Framework
Implement a strong identity foundation
Apply security at all layers

Securing resources with right access



Don’t set and forget

SIMPLIFY YOUR JOURNEY TO LEAST PRIVILEGE

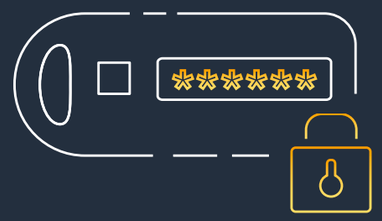


Set: the right fine-grained permissions

Verify: who can access what

Refine: excessive permissions

Manage authentication



Use strong sign-in mechanisms

Use temporary credentials

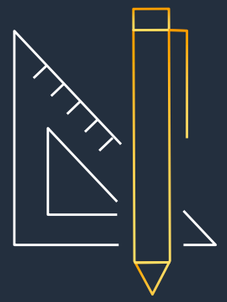
Store and use secrets securely

Relay on centralized identity provider

Audit and rotate credentials periodically

Leverage user groups and attributes

Manage permissions



Define access requirements

Grant least privilege access

Establish emergency access process

Reduce permissions continuously

Define permissions guardrails for your organization

Manage access based on lifecycle

Analyze public and cross-account access

Share resources securely within your organization

Share resources securely with a third party

Security best practices – Well Architected Way

- Use authentication and authorization mechanisms
- Application security
- Data encryption and integrity
- Infrastructure Protection and Denial of service
- Threat Detection and Incident Response

Best practice 2: Application Security

- Enable People
- Security is everyone’s job
- Compile the Technical requirements
- Collect Business requirements
- Follow secure coding practices
- Check for vulnerabilities on your dependencies and remove any unnecessary dependencies

OWASP Serverless Top 10
S3:2021 Injection
S6:2021 Vulnerable and outdated components
S8:2021 Software and data integrity failures

AWS Well-Architected Framework
Automate security best practices
Apply security at all layers

Application security

Train for application security

Automate testing throughout the development and release lifecycle

Perform regular penetration testing

Manual code reviews

Application security – cont'd

Centralize services for packages and dependencies

Deploy software programmatically

Regularly assess security properties of the pipelines

Build a program that embeds security ownership in workload teams

Security best practices – Well Architected Way

- Use authentication and authorization mechanisms
- Application security
- Data encryption and integrity
- Infrastructure Protection and Denial of service
- Threat Detection and Incident Response

Best practice 3: Data encryption and integrity

- Identify and classify sensitive data
- Protect data at rest and in transit
    - Decrypt as late as possible
    - Use TLS for transit
- Minimize storage of sensitive data to only what is necessary

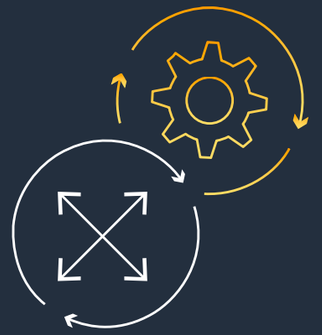
OWASP Serverless Top 10
S2:2021 Cryptographic failures
S8:2021 Software and data integrity failures

AWS Well-Architected Framework
Protect data in transit and at rest
Apply security at all layers

Data classification

Identify the data within your workload

Define data protection controls



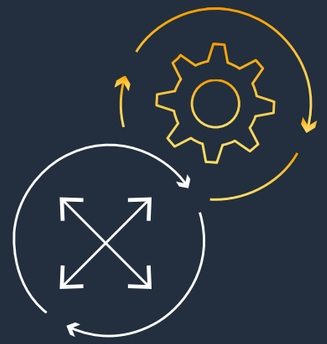
Automate identification and classification

Define data lifecycle management

Protecting data at rest

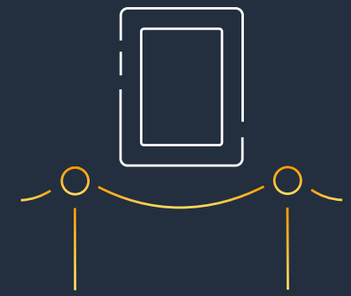
Implement secure key management

Enforce encryption at rest



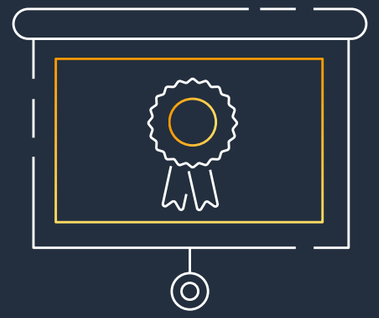
Automate data at rest protection

Enforce access control



Use mechanisms to keep people away from data

Protecting data in transit



Implement secure key and certificate management

Enforce encryption in transit

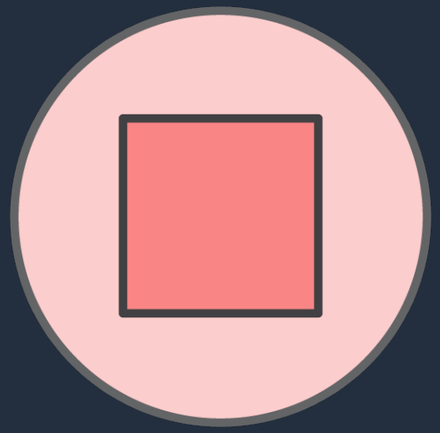
Automate detection of unintended data access

Authenticate network communications

Keep people away from data

Don’t store

Don’t grant



Encrypt

Mask

Tokenize

Isolate

Eliminate

direct access

Operations

as code

Version

control

Security best practices – Well Architected Way

- Use authentication and authorization mechanisms
- Application security
- Data encryption and integrity
- Infrastructure Protection and Denial of service
- Threat Detection and Incident Response

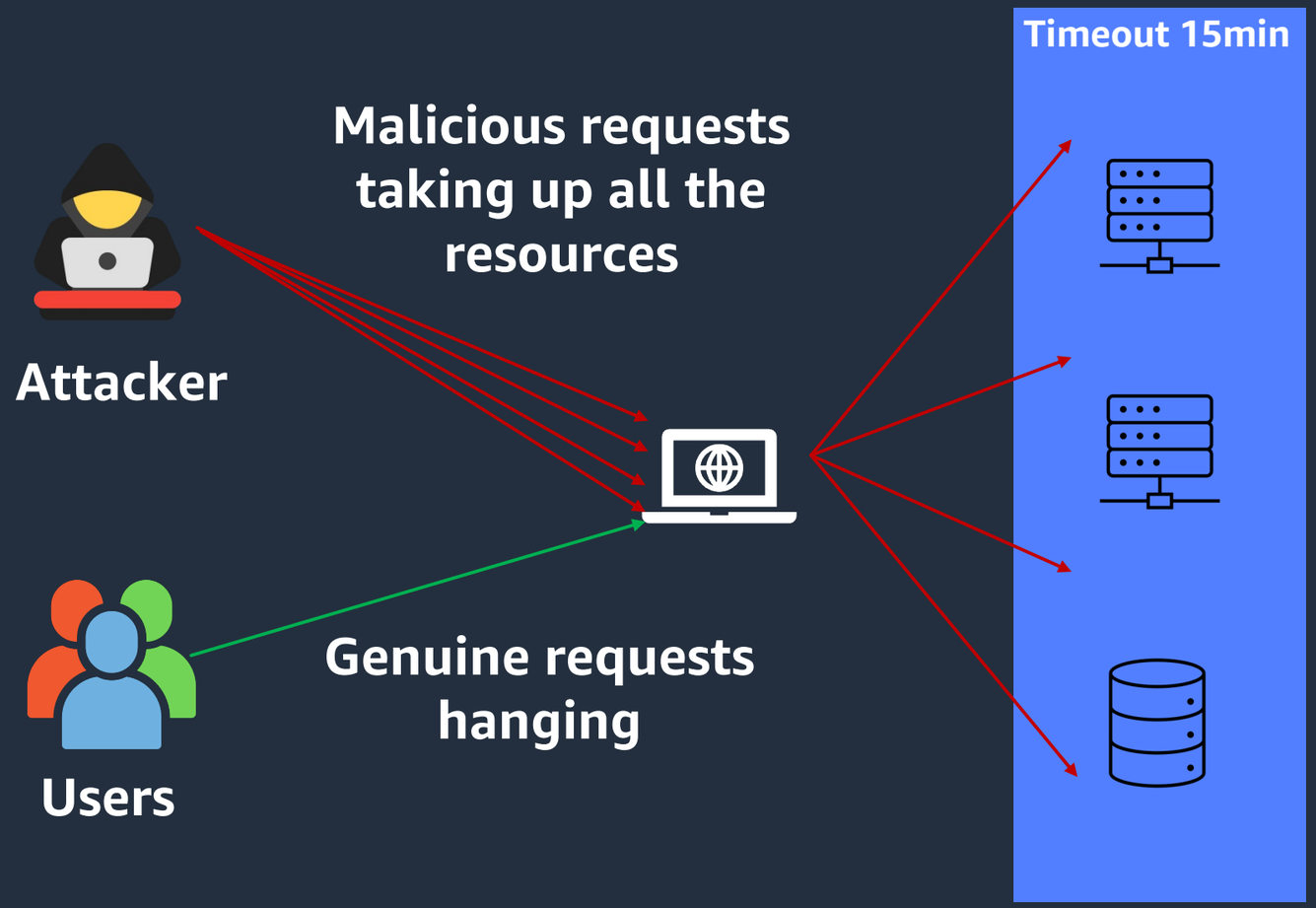
Best practice 5: Denial of Service and infrastructure protection

- DDoS protection
- Throttling/rate limiting
- Network boundaries
- Compute protection

OWASP Serverless Top 10
S4:2021 Insecure design
S5:2021 Security misconfiguration
S6: 2021 Vulnerable and outdated components

AWS Well-Architected Framework
Enable traceability
Apply security at all layers

Example: Denial of Service



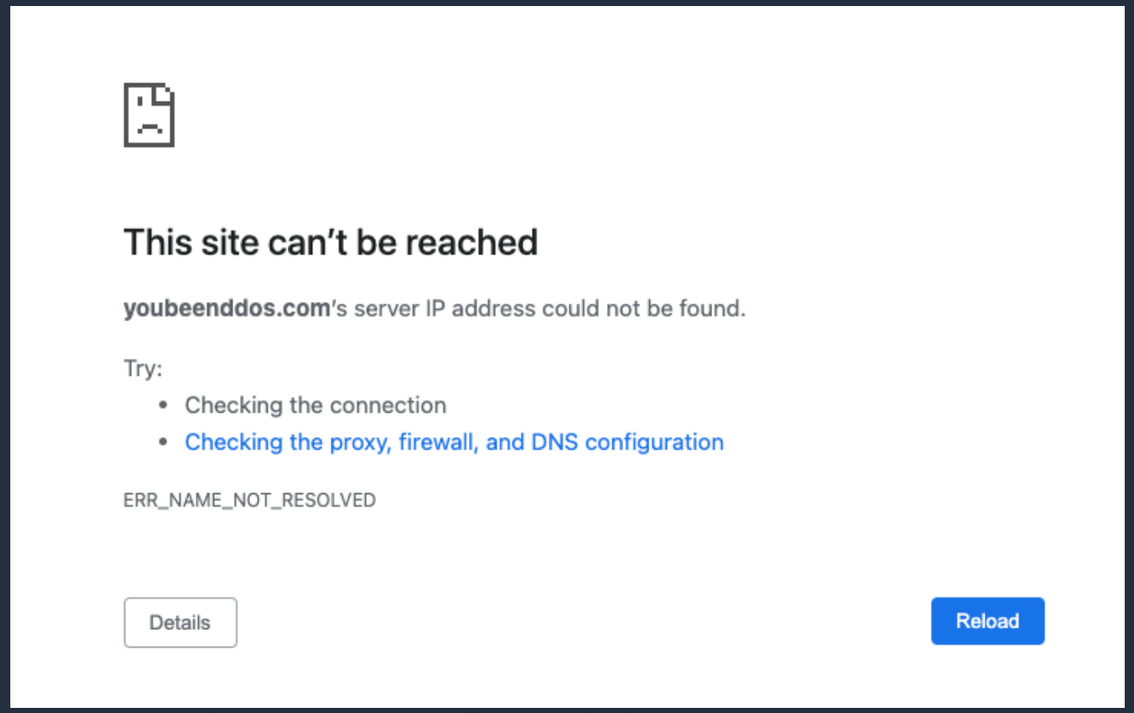
What does a DDoS attack look like?

HTTP 500 Internal Server Error

HTTP 502 Bad Gateway

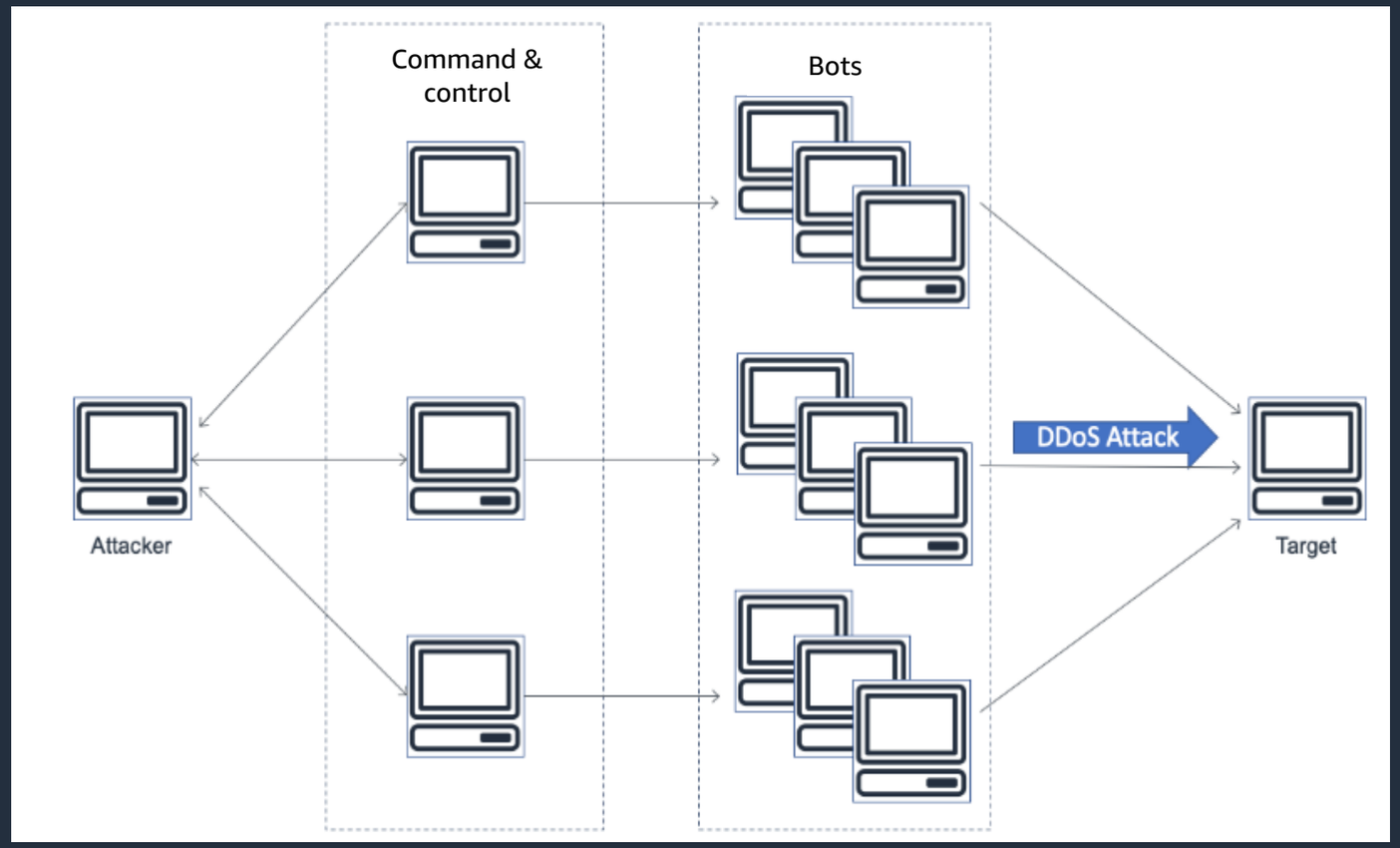
HTTP 503 Service Unavailable

HTTP 504 Gateway Timeout



Understanding distributed denial of service events

- Disrupts the availability of a targeted system, such as a website, reducing the performance for legitimate users
- How? Sends illegitimate traffic from multiple sources (distributed)
- Traffic can come from botnets (network of compromised devices) or DDoS-as-a-service
- Types include HTTP request floods, reflection attacks, and packet floods

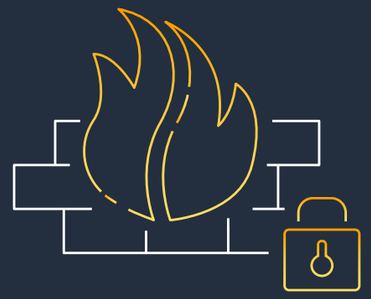


Best practices for DDoS resiliency

- CDN - Edge Locations as an entry point for your applications.
- Protect your DNS infrastructure
- Protect your Origins
- Scalable architecture

Protecting network resources

Create network layers



Control traffic at all layers

Automate network protection

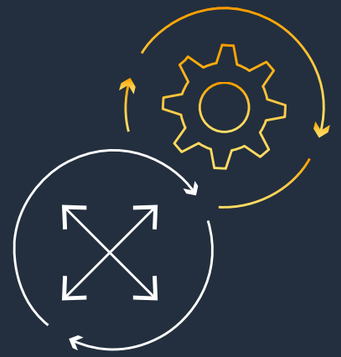
Implement inspection and protection

Protecting compute resources

Perform vulnerability management

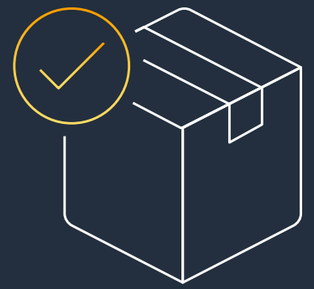
Reduce attack surface

Implement managed services



Automate compute protection

Enable people to perform actions at a distance



Validate software integrity

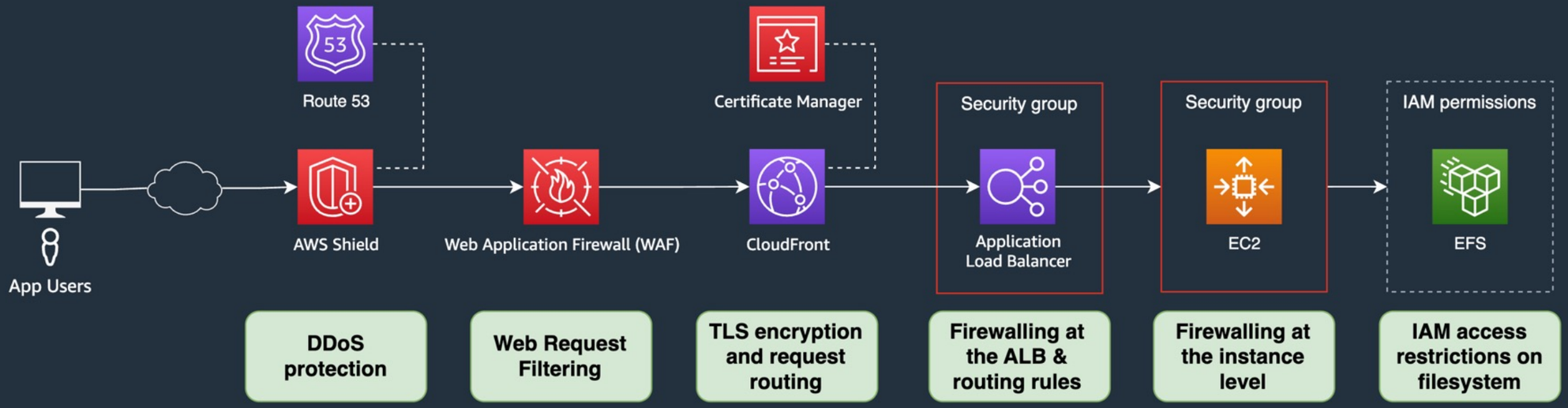
Apply security at all layers

AWS WELL-ARCHITECTED SECURITY PILLAR: DESIGN PRINCIPLES

Apply a defense in depth approach with multiple security controls

Apply security to all layers (for example, edge of network, VPC, load balancing, every instance and compute service, operating system, application, and code).

Defense in depth - **Apply security at all layers**



Security best practices – Well Architected Way

- Use authentication and authorization mechanisms
- Application security
- Data encryption and integrity
- Infrastructure Protection and Denial of service
- Threat Detection and Incident Response

Best practice 5: Threat Detection and Incident Response

- Establish a Plan
- Logging and Monitoring
- Incident Response - Prepare, Simulate, and Iterate

OWASP Serverless Top 10
S5:2021 Security misconfiguration
S9:2021 Security logging and monitoring failures

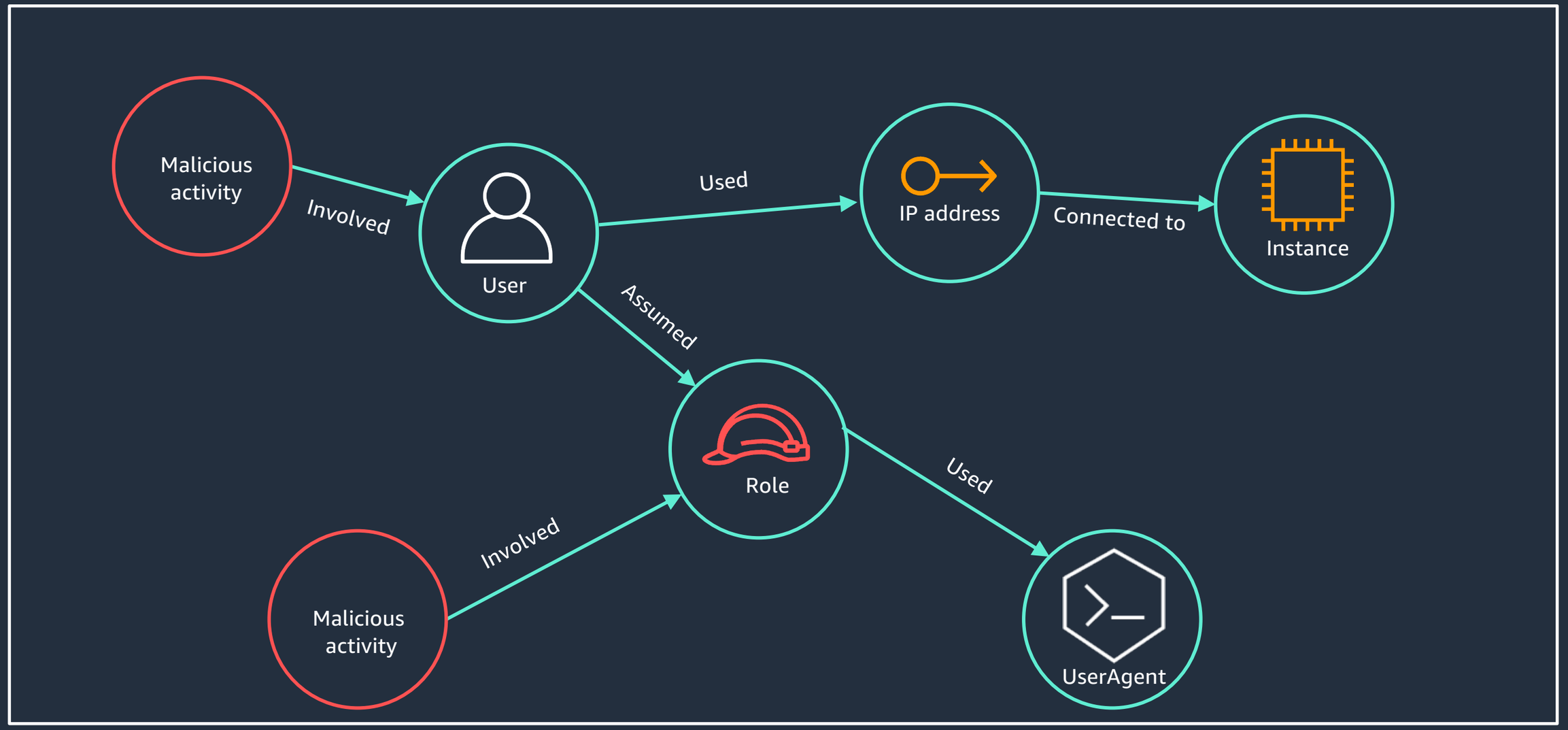
AWS Well-Architected Framework
Enable traceability
Apply security at all layers

Prepare for security events

AWS WELL-ARCHITECTED Security Pillar: Design Principles

- Prepare for an incident by having incident management and investigation policy and processes that align to your organizational requirements.
- Run incident response simulations and use tools with automation to increase your speed for detection, investigation, and recovery.

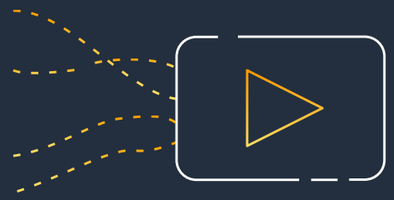
How do we streamline root cause analysis?



Logging and Monitoring

Configure service and application logging

Analyze logs, findings, and metrics centrally



Automate response to events

Implement actionable security events

Incident Response - Prepare, Simulate, and Iterate

Identify key personnel and external resources

Develop incident management plans

Prepare forensic capabilities

Develop and test security incident response playbooks

Pre-provision access

Pre-deploy tools

Run simulations

Establish a framework for learning from incidents

Takeaway

Suboptimal security

Lack of visibility

Cost and complexity with reviewing logs

Not enough people

Shortage of skilled security professionals

Prioritizing findings

Analysts cannot review every security issue

Distribution of security ownership

"We **own** the security of **what we build and run**"

Service team

Owner

Security Team

Org owner

Enabler

"We **own** the **organization's overall security –** and **enable** product teams to deliver and operate securely"*

Five Key Elements to Building a Good Security Culture:

1. **SECURITY IS EVERYONE'S JOB:** It is important for security to be a focus for all employees. Broad engagement helps establish the business's overall security posture.

2. **HONEST, NO-BLAME CULTURE:** If people are punished for raising concerns or admitting to having created a problem, they won't speak up.

3. **SECURITY FIRST APPROACH:** Engage security teams in projects early on so they can influence the success of projects, eg by performing risk and security analysis.

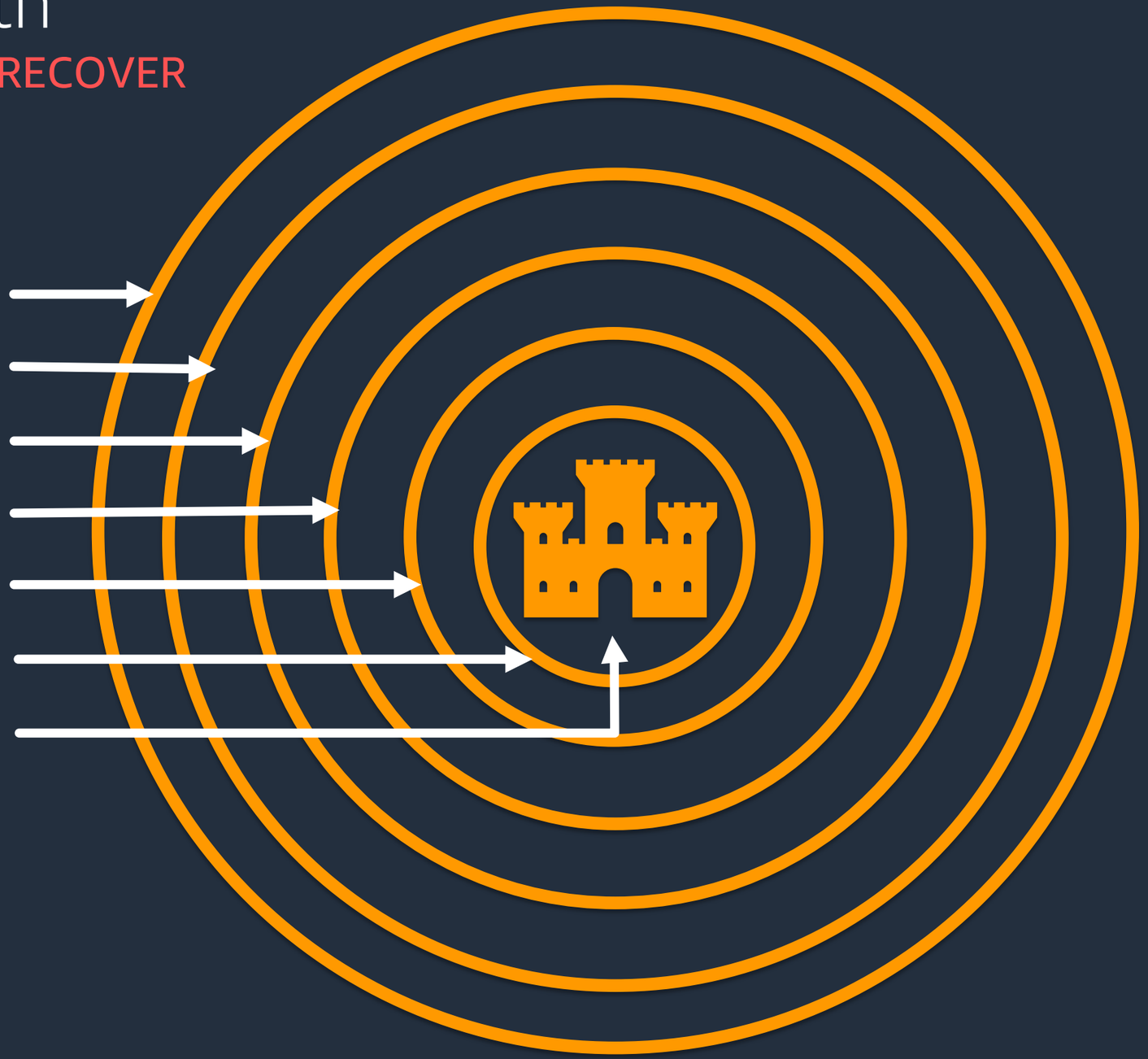
4. **SET CLEAR EXPECTATIONS:** Devise non-negotiable tenets that are applied across the organization. Consider nominating Security Guardians to aid adoption.

5. **BE ACCOUNTABLE:** A culture of transparency is critical and every employee needs to understand they are accountable for their actions.

Takeaway - Defense-in-Depth

IDENTIFY, PROTECT, DETECT, RESPOND & RECOVER

- Policies, Procedures & Awareness
- Network & Edge Protection
- Identity & Access Management
- Threat Detection & Incident Response
- Infrastructure Protection
- Application Protection
- Data Protection



“Protecting your customers should be your #1 priority. Without that, you don’t have a business. It should come before any features.”

- Dr. Werner Vogels, CTO Amazon

Questions ☺

Thank you!

Jhalak Modi

https://www.linkedin.com/in/jhalakmodi/