Application Security

Cybersecurity Bootcamp 2024

Disclaimer

This learning material is addressed and used only for Cybersecurity Bootcamp 2024 and should not be used or relied upon for any other purposes. Our learning material is not to be disseminated to or used by any third party in whole or in part without prior consent and permission from Kasikorn Technology Group Secretariat Company Limited (KBTGSec). Accordingly, we will not accept or take any responsibility or liability for any party or any person, whether or not such material is shown, disseminated, obtained, or possessed to such party or person since such material is only for educational purposes. We reserve all of our rights, including but not limited to intellectual property rights in our learning material, such as presentations, spreadsheets, system techniques, ideas, concepts, information, forms, electronic tools, forming parts of the materials, etc. © 2024 KASIKORN Business-Technology Group (KBTG) All rights reserved."

Application security

1 : Overview

2 : Application Security Protection

3 : Threat Modeling

4 : OWASP

4.1 : OWASP - Web security

4.2 : OWASP - API Security

4.3 : OWASP - Mobile security

4.4 : OWASP - Proactive Control

1 : Overview

1 : Overview

2 : Application Security Protection

3 : Threat Modeling

4 : OWASP

4.1 : OWASP - Web security

4.2 : OWASP - API Security

4.3 : OWASP - Mobile security

4.4 : OWASP - Proactive Control

Application Security

Objective

- To understand the importance of Application Security.
- To learn about prevention and detection methods.
- To gain insights into strategic approaches for securing applications.
- To identify and address existing threats and vulnerabilities.

Application Security

Application security refers to the measures and practices implemented to **protect** software applications from **threats** and **vulnerabilities** throughout their lifecycle. It encompasses various processes, technologies, and techniques aimed at ensuring that applications are secure, resilient, and able to withstand cyber attacks.

Why Do We Need Application Security?

Why Do We Need Application Security?

How to determine sensitive data - NIST

Confidentiality

- Passwords
- Data encryption
- Two-factor authentication
- Security tokens
- Best practices like hard-copy only storage and using disconnected storage media

Integrity

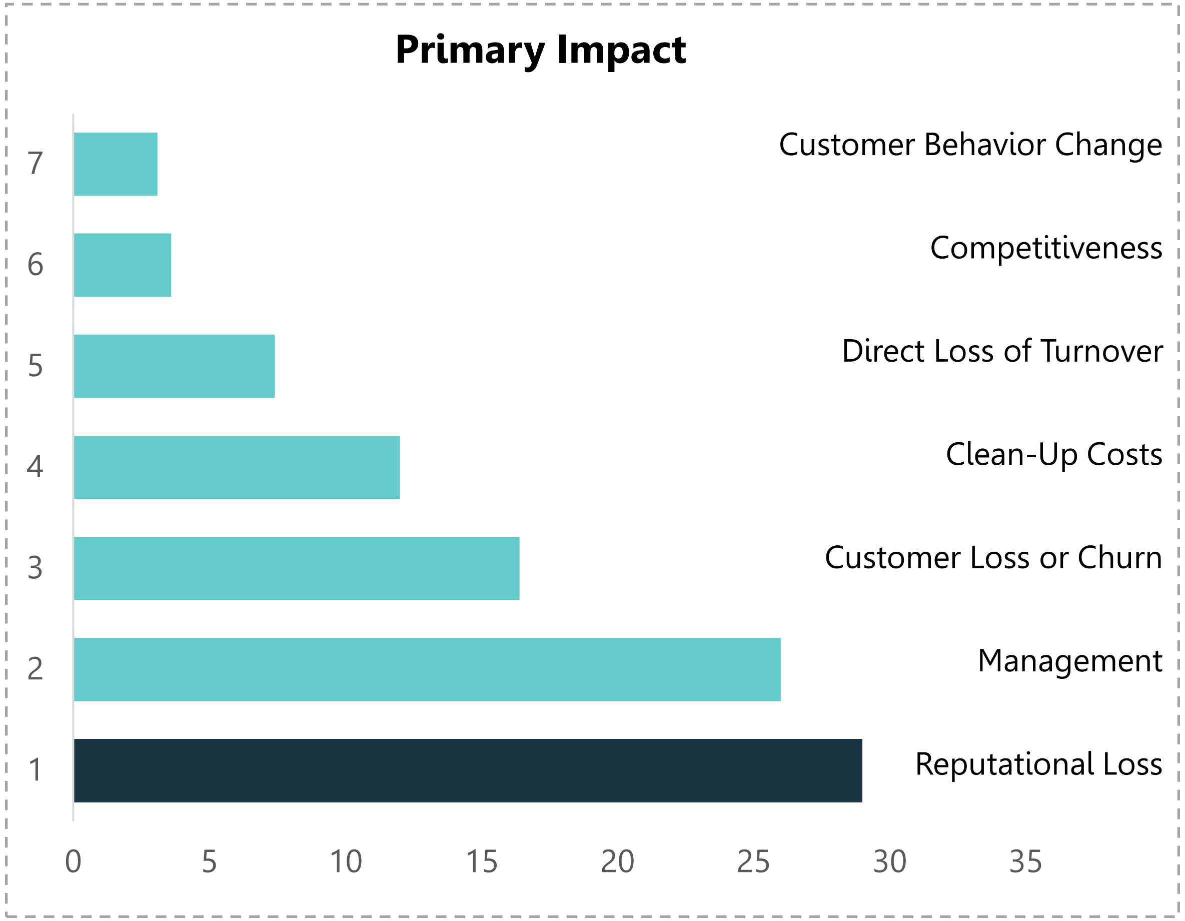
- Using audit logs
- Enacting user access controls and file permissions – including for databases
- Maintaining file backups and storage redundancies

Availability

- Proper hardware maintenance
- Keeping on top of software updates and security patches
- Having a disaster recovery plan
- Guarding against data loss in case of natural or man-made calamities

1. Protecting Sensitive Data
2. Preventing Cyber Attacks
3. Maintaining Trust and Reputation
4. Compliance Requirements
5. Reducing Business Risks
6. Enabling Digital Transformation

Why Do We Need Application Security?



1 Protecting Sensitive Data

2 Preventing Cyber Attacks

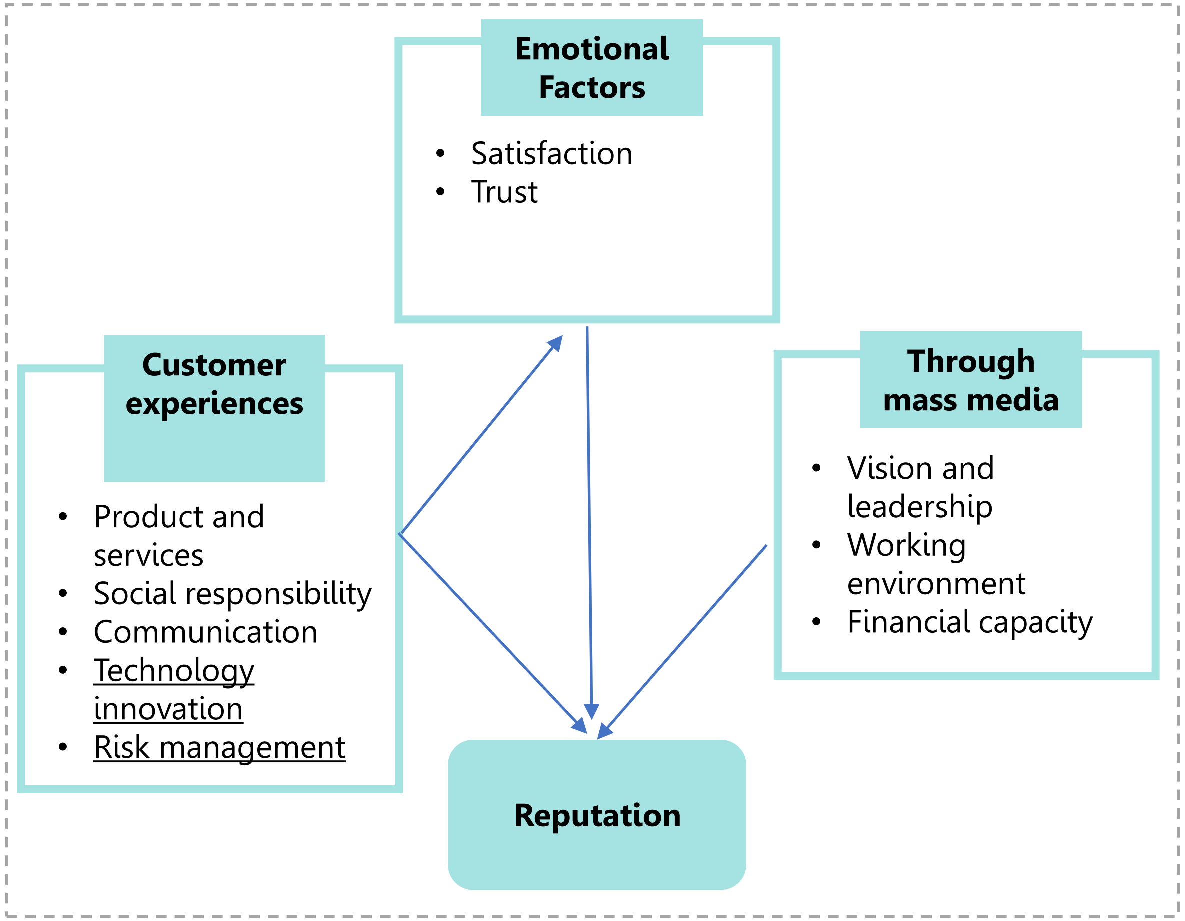
3 Maintaining Trust and Reputation

4 Compliance Requirements

5 Reducing Business Risks

6 Enabling Digital Transformation

Why Do We Need Application Security?



Emotional Factors

- Satisfaction
- Trust

Customer experiences

- Product and services
- Social responsibility
- Communication
- Technology innovation
- Risk management

Through mass media

- Vision and leadership
- Working environment
- Financial capacity

Reputation

1 Protecting Sensitive Data

2 Preventing Cyber Attacks

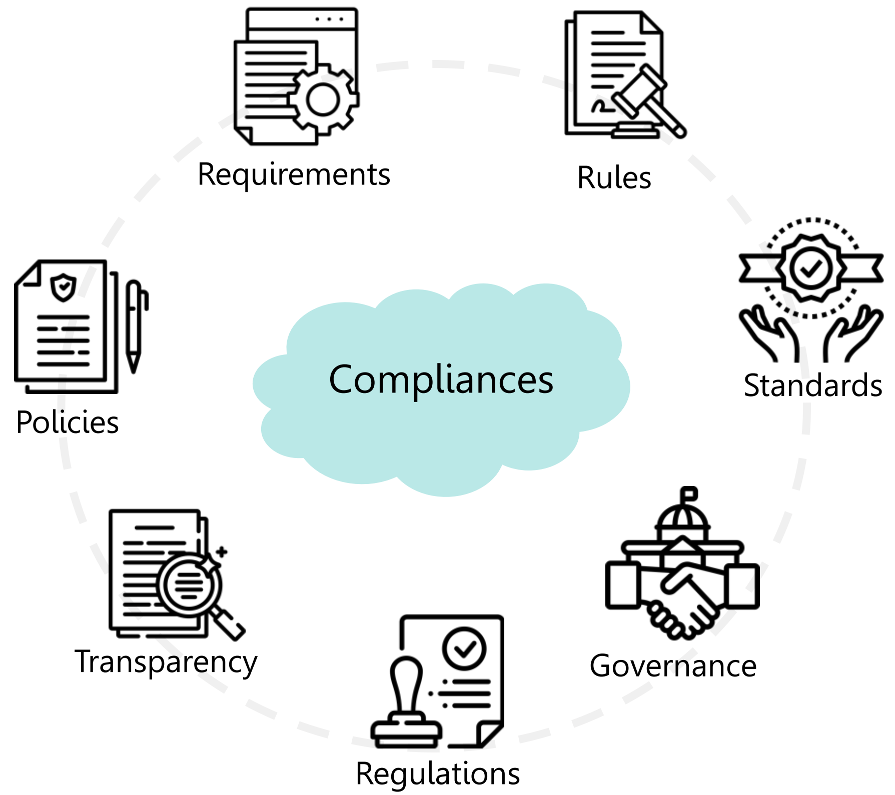
3 Maintaining Trust and Reputation

4 Compliance Requirements

5 Reducing Business Risks

6 Enabling Digital Transformation

Why Do We Need Application Security?



1 Protecting Sensitive Data

2 Preventing Cyber Attacks

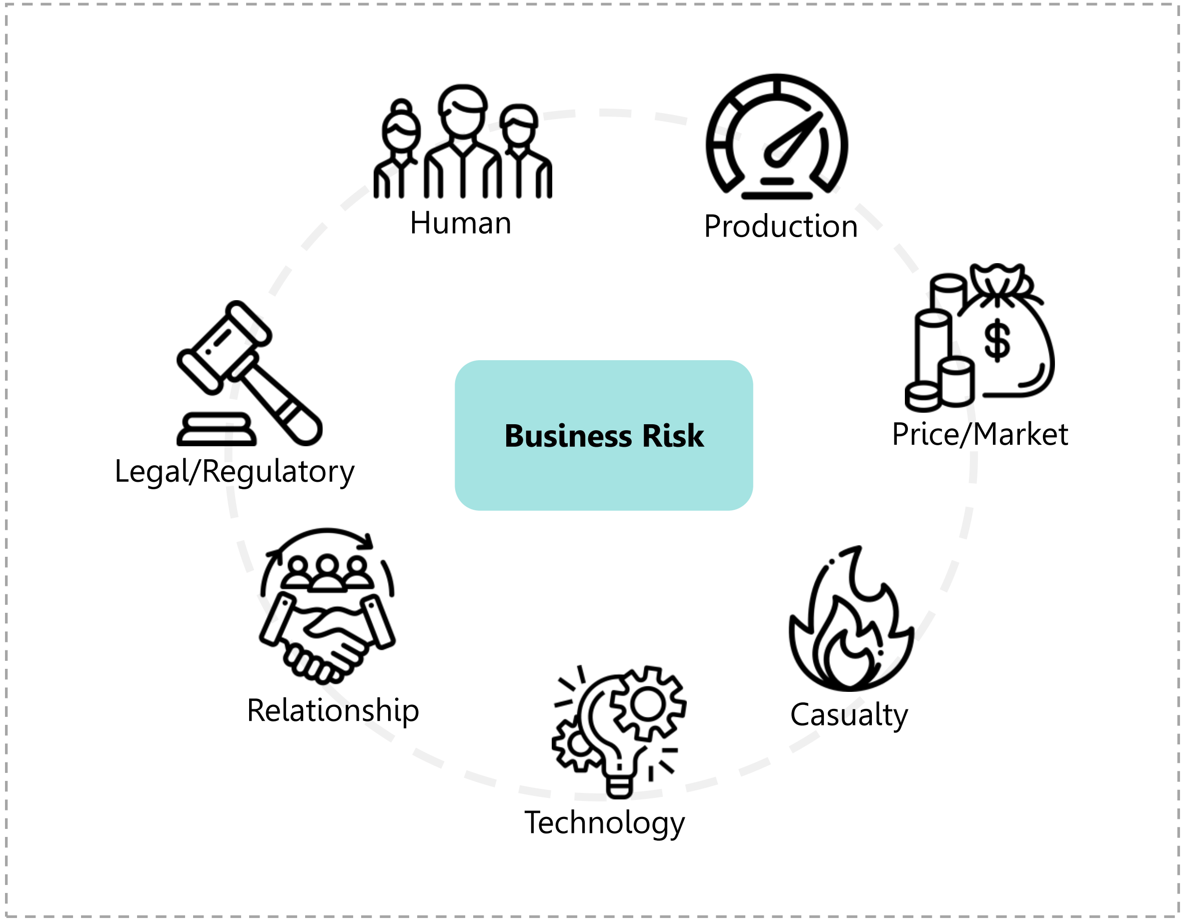
3 Maintaining Trust and Reputation

4 Compliance Requirements

5 Reducing Business Risks

6 Enabling Digital Transformation

Why Do We Need Application Security?



1 Protecting Sensitive Data

2 Preventing Cyber Attacks

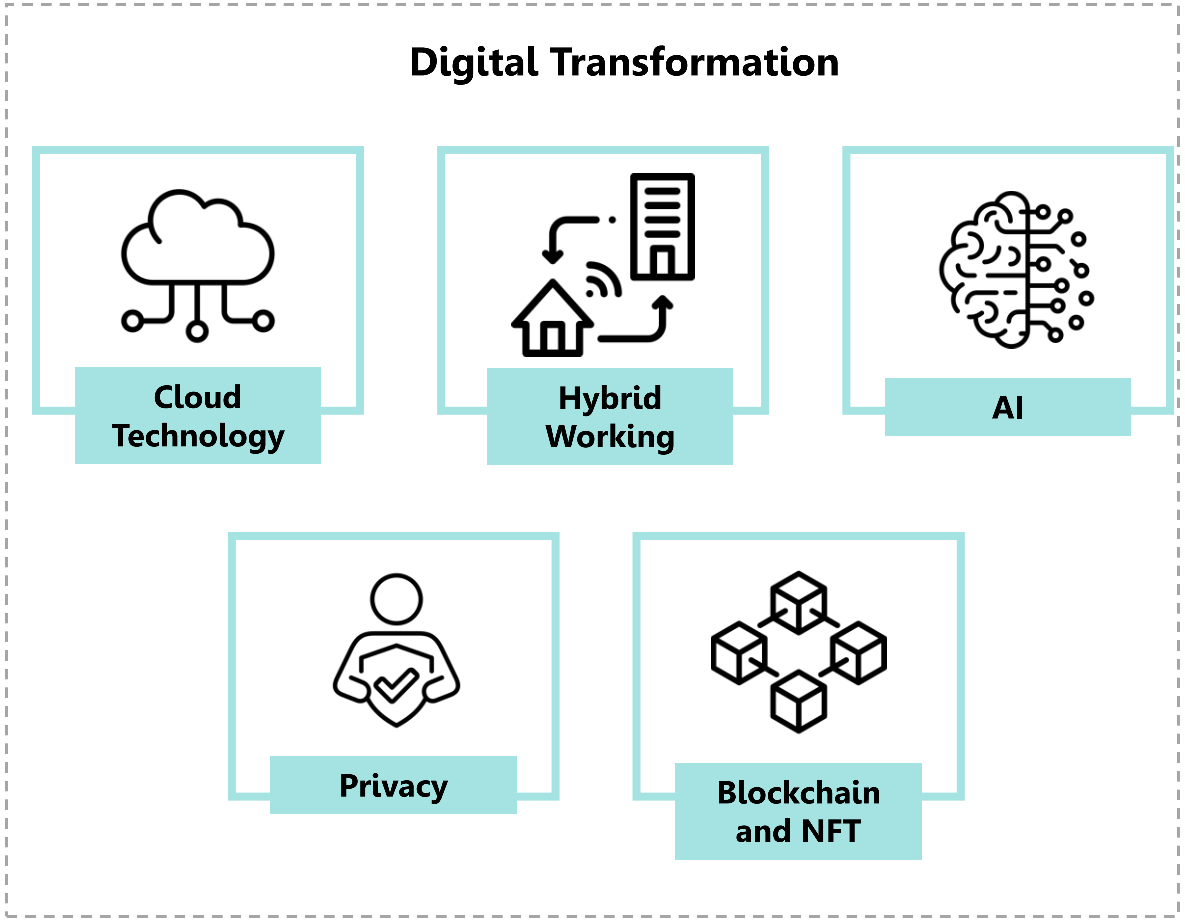
3 Maintaining Trust and Reputation

4 Compliance Requirements

5 Reducing Business Risks

6 Enabling Digital Transformation

Why Do We Need Application Security?



Digital Transformation

Cloud Technology

Hybrid Working

AI

Privacy

Blockchain and NFT

1 Protecting Sensitive Data

2 Preventing Cyber Attacks

3 Maintaining Trust and Reputation

4 Compliance Requirements

5 Reducing Business Risks

6 Enabling Digital Transformation

2 : Application Security Protection

1 : Overview

2 : Application Security Protection

3 : Threat Modeling

4 : OWASP

4.1 : OWASP - Web security

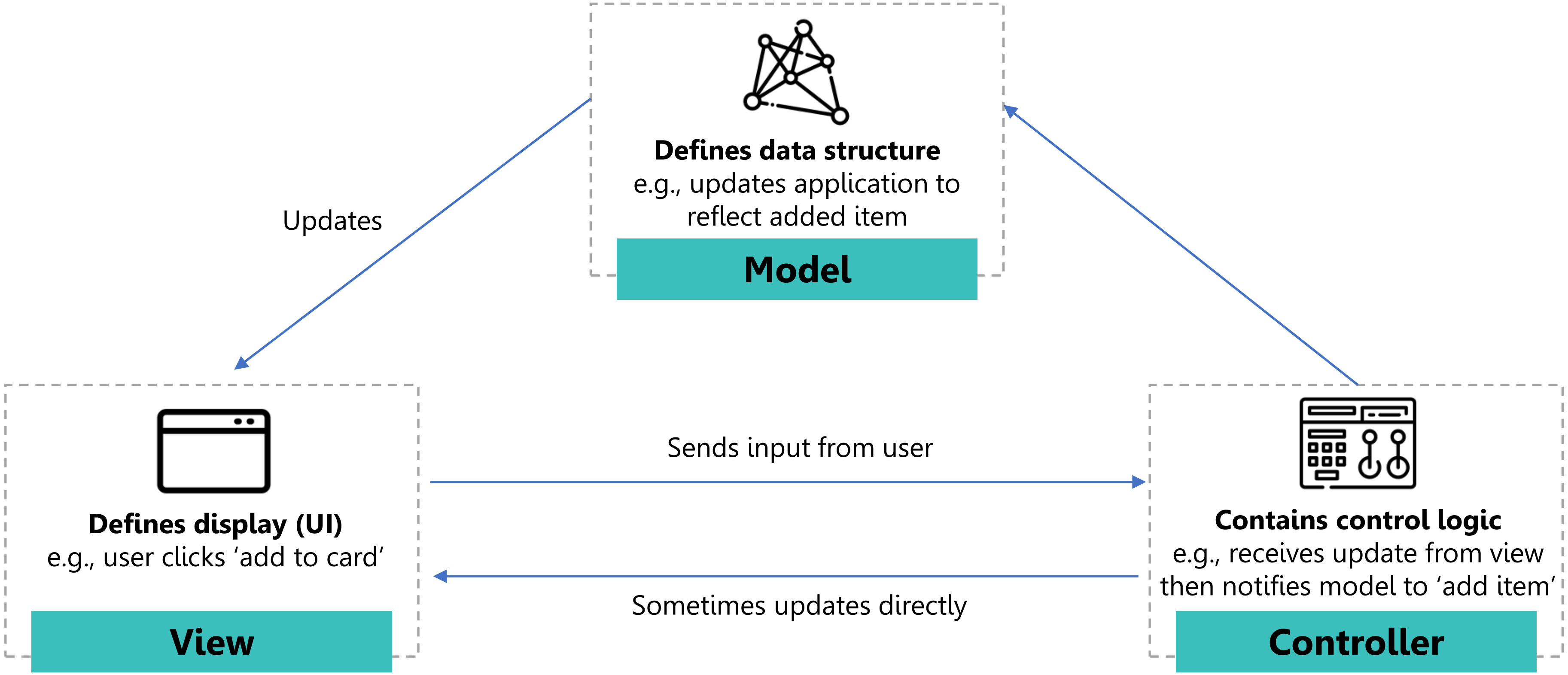
4.2 : OWASP - API Security

4.3 : OWASP - Mobile security

4.4 : OWASP - Proactive Control

Application Security Protection

Application Model - MVC



Defines data structure
e.g., updates application to reflect added item

Model

Updates

Defines display (UI)
e.g., user clicks 'add to card'

View

Sends input from user

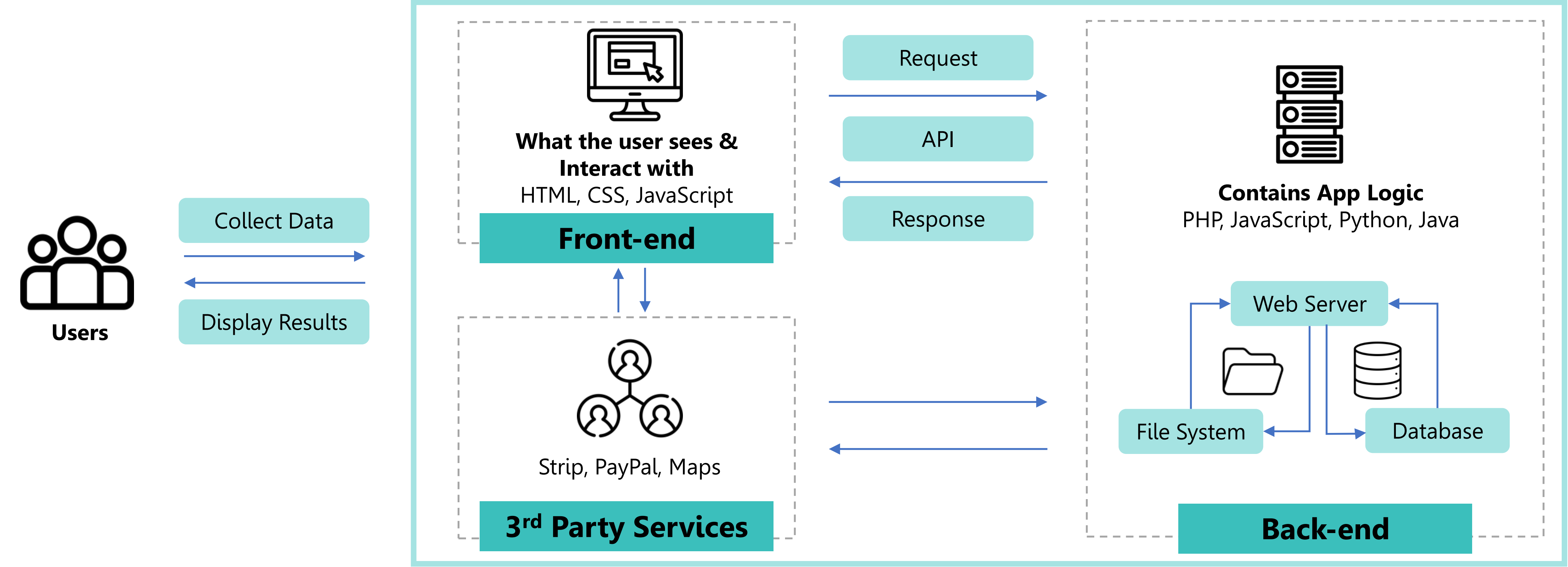
Sometimes updates directly

Contains control logic
e.g., receives update from view then notifies model to 'add item'

Controller

Application Security Protection

Web Application Architecture



Application Security Protection

Layer of protection

01 Network Layer

- firewall
- IDPS

02 Web Application Layer

- WAF
- Secure Coding
- Authentication and Authorization
- Session management

03 Data Layer

- Data encryption
- Database encryption

04 Infrastructure Layer

- Access control
- Patch management

05 Monitoring and Logging

- Logging and auditing
- SIEM

06 User Education

- User training

07 3rd Party

- 3rd Party risk assessment

08 Incident response

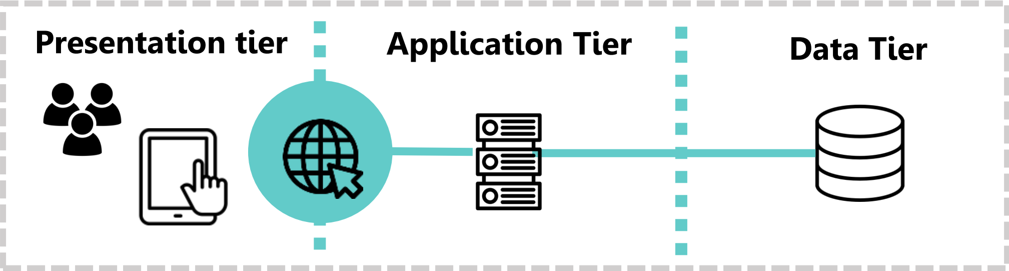
- Incident response plan

Application Security Protection

01 Network Layer Key Concerns

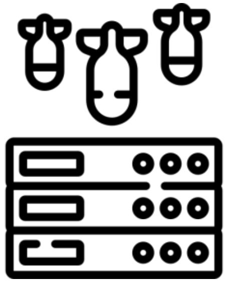
A. Firewall

Definition: Firewalls are security devices that monitor and control incoming and outgoing network traffic based on predetermined security rules.



B. Intrusion Detection and Prevention Systems (IDPS):

Definition: IDPS are security appliances or software that monitor network and/or system activities for malicious or suspicious behavior.



Application Security Protection

02 Web Application Layer Key Concerns

A. Web Application Firewall (WAF)

Definition: WAF is a security solution designed to protect web applications from various online threats.

B. Secure Coding Practices

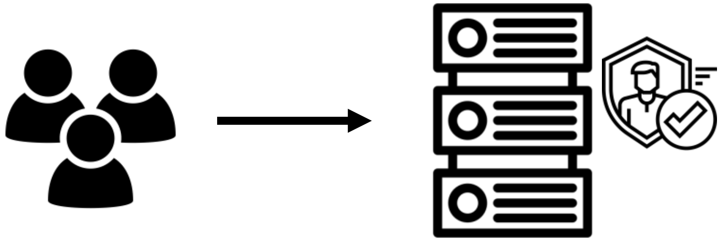
Definition : Secure coding practices involve following guidelines and best practices during the software development process to create applications that are resistant to security threats

Application Security Protection

02 Web Application Layer Key Concerns

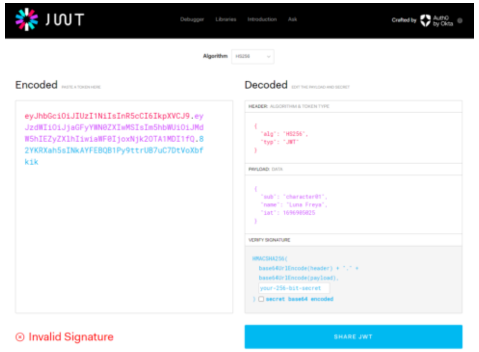
C. Authentication and Authorization

Definition: Authentication is the process of verifying the identity of users, ensuring they are who they claim to be. Authorization involves granting or denying access to resources based on the authenticated user's permissions.



D. Session Management

Definition: Session management refers to the secure handling of user sessions within a web application.



Algorithm HS256

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiJjaGFyYWN0ZXIwMSIsIm5hbWUiOiJMd
W5hIEZyZXlhIiwiaWF0IjoxNjk2OTA1MDI1fQ.8
2YKRXah5sINkAYFEBQB1Py9ttrUB7uC7DtVoXbf
kik
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "character01",
  "name": "Luna Freya",
  "iat": 1696905025
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) □ secret base64 encoded
```

⊗ Invalid Signature

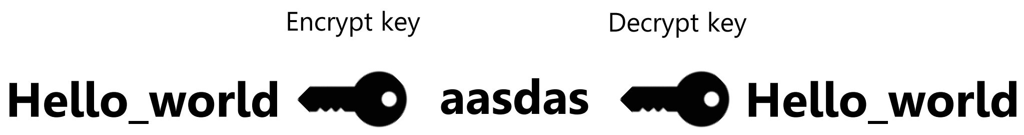
02

Application Security Protection

03 Data Layer Key Concerns

A. Data Encryption

Definition: Data encryption is the process of converting data into a secure format to prevent unauthorized access.



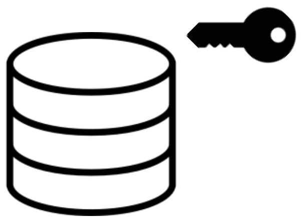
Encrypt key

Decrypt key

Hello_world → aasdas → Hello_world

B. Database Security

Definition: Database security involves implementing measures to protect databases from unauthorized access, manipulation, or disclosure..



Application Security Protection

04 Infrastructure Layer Key Concerns

A. Access Controls

Definition: Access controls restrict user access to systems, networks, and data based on their roles and responsibilities.

B. Patch Management

Definition: Patch management involves regularly applying updates, patches, and fixes to software, operating systems, and other IT infrastructure components.

Application Security Protection

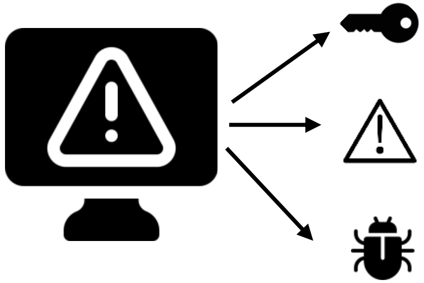
05 Monitoring and Logging Key Concerns

A. Logging and Auditing

Definition: Logging and auditing involve the systematic recording of security-relevant events and activities within a system.

B. Security Information and Event Management (SIEM)

Definition: SIEM is a comprehensive solution that integrates security information management (SIM) and security event management (SEM).

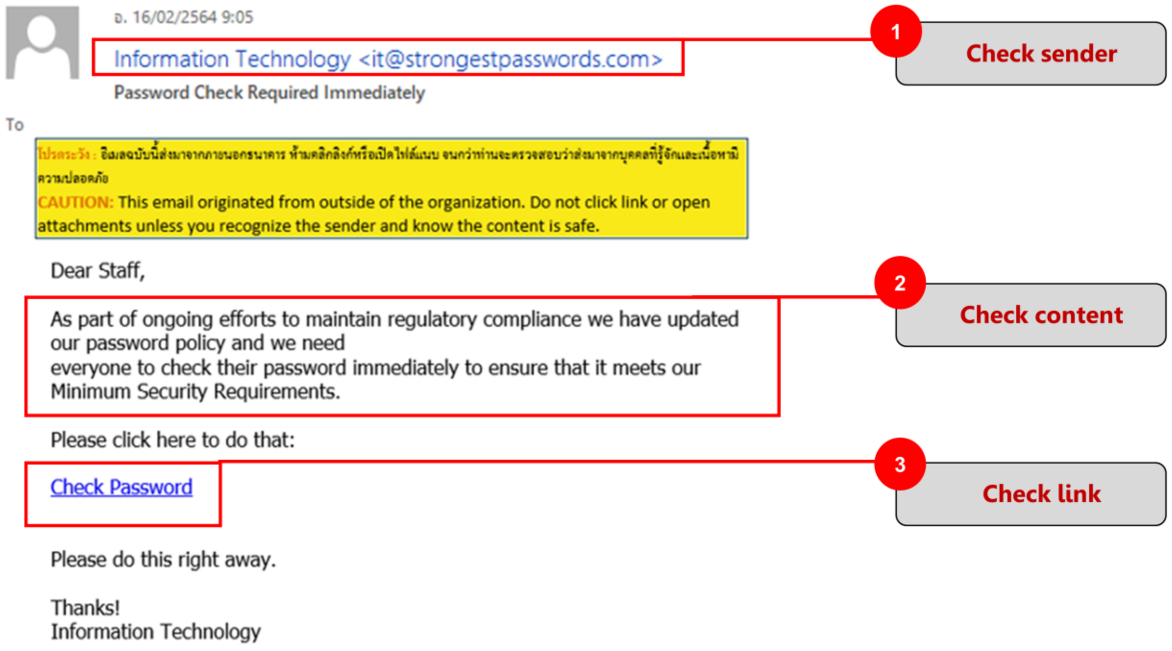


Application Security Protection

06 User Education and Awareness Key Concerns

A. User Training

Definition: User training focuses on educating individuals within an organization about security best practices



อ. 16/02/2564 9:05

Information Technology <it@strongestpasswords.com>

Password Check Required Immediately

To

โปรดระวัง : อีเมลฉบับนี้ส่งมาจากภายนอกธนาคาร ห้ามคลิกลิงก์หรือเปิดไฟล์แนบ จนกว่าท่านจะตรวจสอบว่าส่งมาจากบุคคลที่รู้จักและเนื้อหามีความปลอดภัย

CAUTION: This email originated from outside of the organization. Do not click link or open attachments unless you recognize the sender and know the content is safe.

Dear Staff,

As part of ongoing efforts to maintain regulatory compliance we have updated our password policy and we need
everyone to check their password immediately to ensure that it meets our Minimum Security Requirements.

Please click here to do that:

Check Password

Please do this right away.

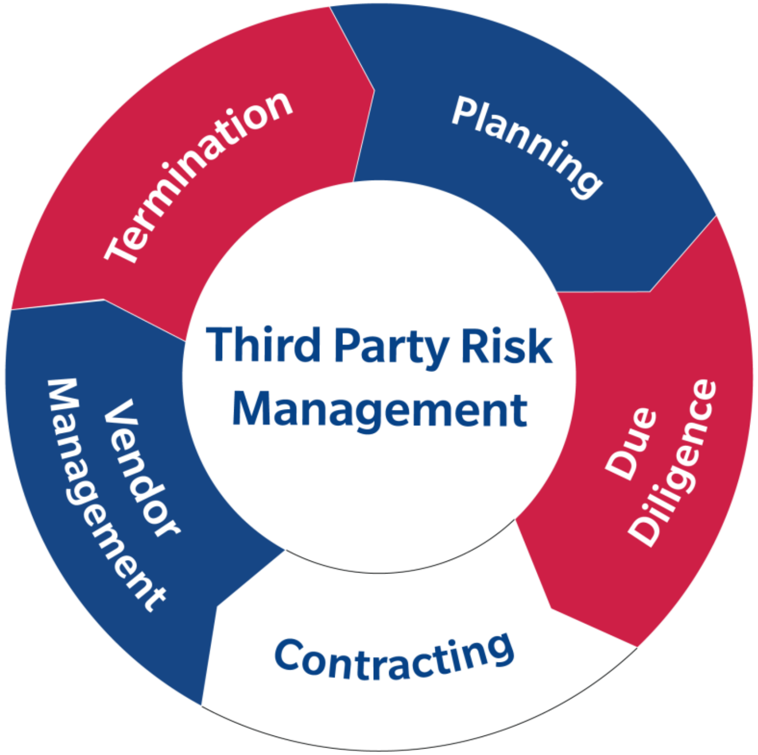
Thanks!
Information Technology

Application Security Protection

07 Third-Party and Supply Chain Security Key Concerns

A. Third-Party Risk Management

Definition: Third-party risk management involves assessing and mitigating the security risks associated with external vendors, suppliers, or service providers.

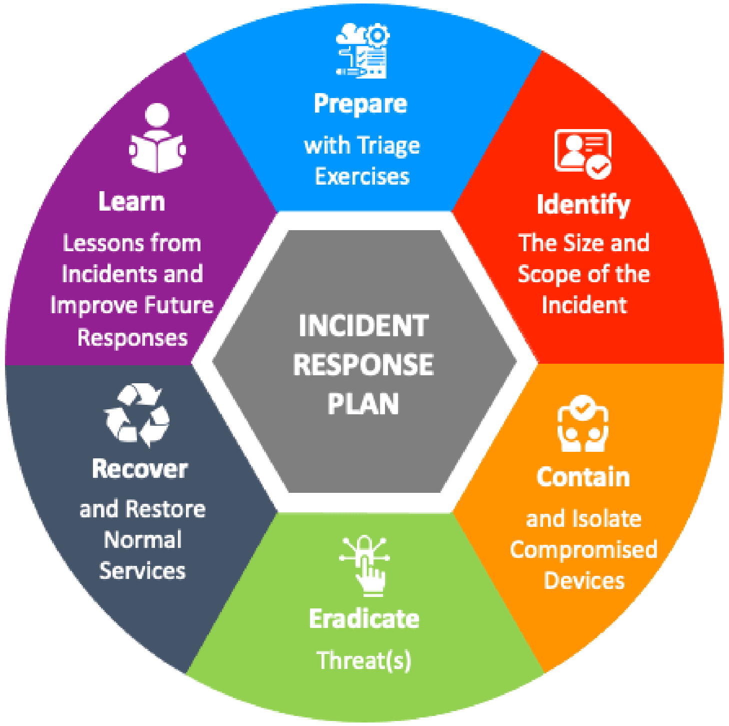


Application Security Protection

08 Incident Response Key Concerns

A. Incident Response Plan

Definition: An incident response plan is a documented set of procedures for identifying, responding to, and recovering from security incidents.



INCIDENT RESPONSE PLAN

Prepare with Triage Exercises

Identify The Size and Scope of the Incident

Contain and Isolate Compromised Devices

Eradicate Threat(s)

Recover and Restore Normal Services

Learn Lessons from Incidents and Improve Future Responses

Key take away 1

What is the purpose of using prevention and detection methods?

3 : Threat Modeling

1 : Overview

2 : Application Security Protection

3 : Threat Modeling

4 : OWASP

4.1 : OWASP - Web security

4.2 : OWASP - API Security

4.3 : OWASP - Mobile security

4.4 : OWASP - Proactive Control

Threat Modeling

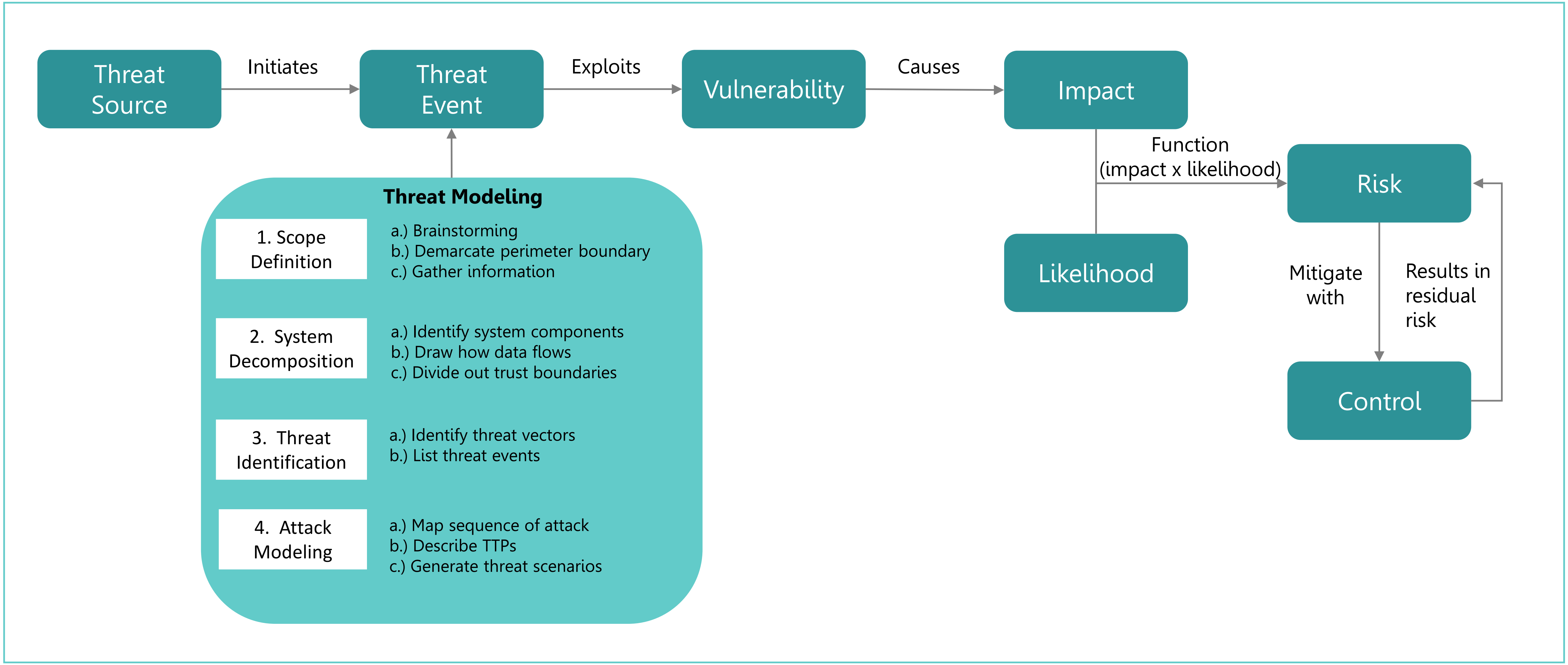
What is Threat Modeling?

- Threat modeling is a process in security architecture and design.
- Identifying and analyzing potential threats to a system or application.
- Organizations can implement effective security measures to mitigate risks and protect against attacks.

Why threat modeling is important?

- Outlining the concern you have as it pertains to a specific system, application, or process
- Making a list outlining the assumptions regarding the threat, which need to be verified as conditions change
- A concrete list of threats
- A list of remediation and elimination steps
- A way to make sure the methods of dealing with the threats are successful and still valid as the threat landscape changes

Threat Modeling fits into Risk Assessment



Threat Modeling

1. Scope Definition
   - a.) Brainstorming
   - b.) Demarcate perimeter boundary
   - c.) Gather information
2. System Decomposition
   - a.) Identify system components
   - b.) Draw how data flows
   - c.) Divide out trust boundaries
3. Threat Identification
   - a.) Identify threat vectors
   - b.) List threat events
4. Attack Modeling
   - a.) Map sequence of attack
   - b.) Describe TTPs
   - c.) Generate threat scenarios

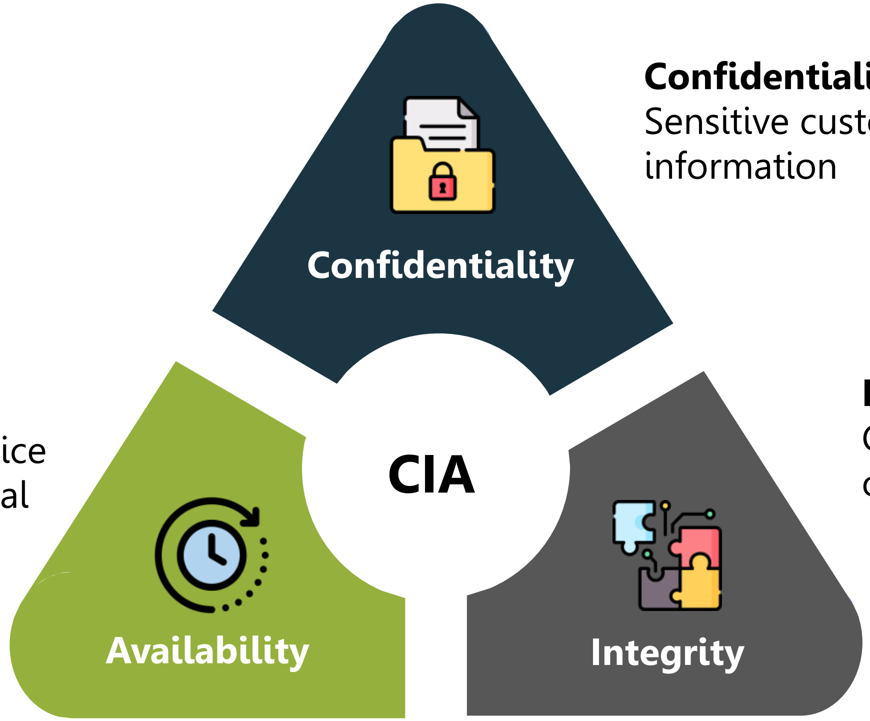
https://www.csa.gov.sg/docs/default-source/csa/documents/legislation_supplementary_references/guide-to-cyber-threat-modelling.pdf?sfvrsn=a367caff_0

Threat modeling methods and tools

Threat Modeling

CIA method

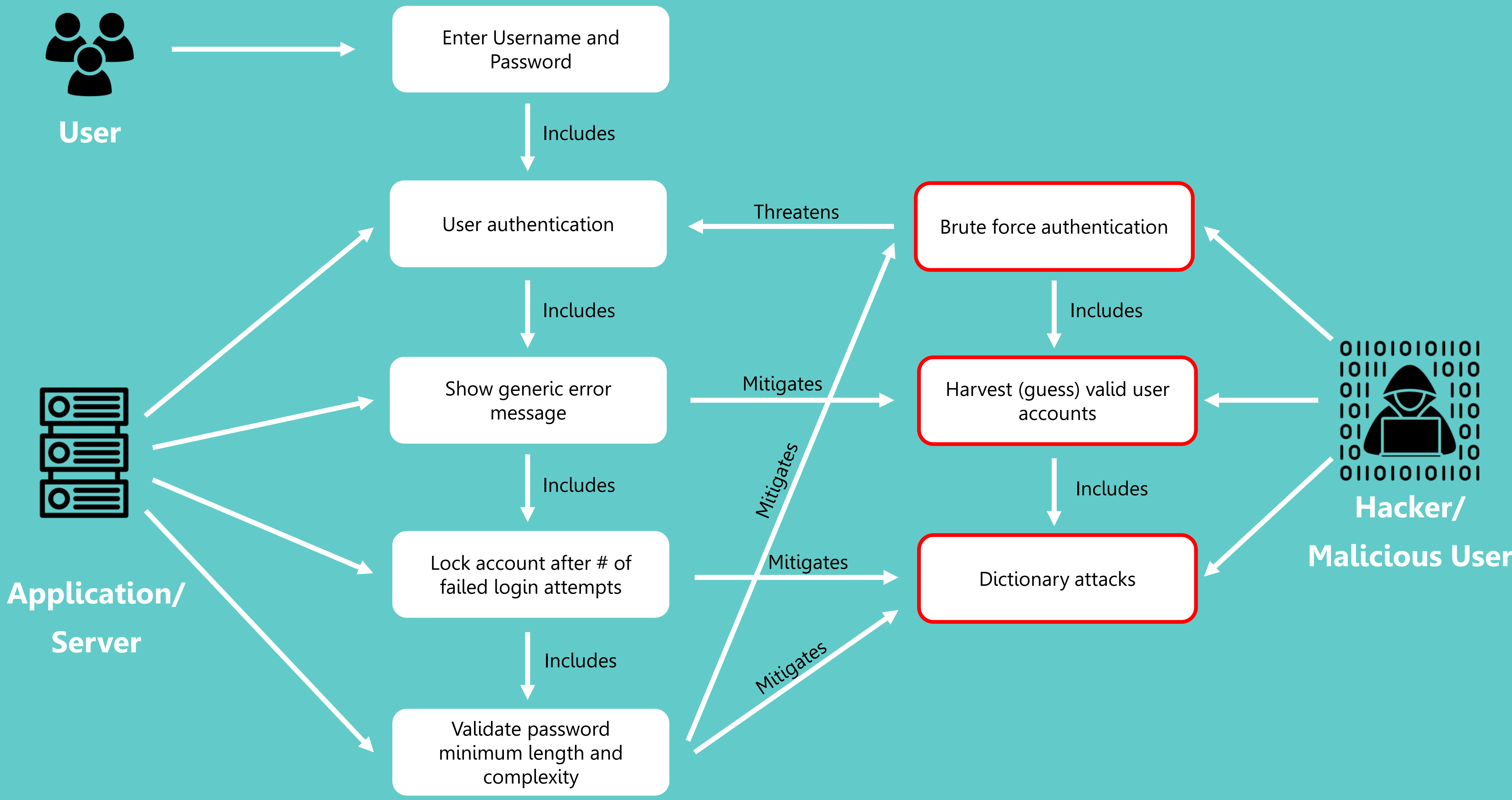
As a starting point, use the CIA (confidentiality, integrity, availability) method to define what needs protecting in the organization.



Confidentiality: Sensitive customer information

Integrity: Company operational or proprietary data

Availability: Reliability of a service such as a web portal



User

Enter Username and Password

Includes

User authentication

Threatens

Brute force authentication

Includes

Show generic error message

Mitigates

Harvest (guess) valid user accounts

Includes

Mitigates

Lock account after # of failed login attempts

Mitigates

Dictionary attacks

Includes

Mitigates

Validate password minimum length and complexity

Application/ Server

Hacker/ Malicious User

LINDDUN

Persona non grata

VAST

TRIKE

PASTA

STRIDE

Model

● ● ● ● ● ●

LINDDUN

Persona non grata

VAST

TRIKE

PASTA

STRIDE

is a model for identifying computer security threats developed by Praerit Garg and Loren Kohnfelder at Microsoft. It provides a mnemonic for security threats in six categories.

STRIDE

LINDDUN

Persona non grata

VAST

TRIKE

PASTA

Threat	Desired property	Threat Definition
Spoofing	Authenticity	Pretending to be something or someone other than yourself
Tampering	Integrity	Modifying something on disk, network, memory, or elsewhere
Repudiation	Non-repudiability	Claiming that you didn't do something or were not responsible; can be honest or false
Information disclosure	Confidentiality	Providing information to someone not authorized to access it
Denial of service	Availability	Exhausting resources needed to provide service
Elevation of privilege	Authorization	Allowing someone to do something they are not authorized to do

STRIDE

PASTA

PASTA (process for attack simulation and threat analysis) is a framework designed to elevate threat modeling to the strategic level, with input from all stakeholders, not just IT or security teams. PASTA is a seven-step process that begins with defining objectives and scope. It includes vulnerability checks, weakness analysis, and attack modeling, and ends with risk and impact analysis expressed through scoring

LINDDUN

Persona non grata

VAST

TRIKE

PASTA

STRIDE

LINDDUN

Persona non grata

VAST

TRIKE

KBTG
KASIKORN
BUSINESS-TECHNOLOGY GROUP

Stages of PASTA

1. Define the Objectives
2. Define the Technical Scope
3. Decompose the Application
4. Analyze the Threats
5. Vulnerability Analysis
6. Attack Analysis
7. Risk and Impact Analysis

PASTA

STRIDE

LINDDUN

Persona non grata

VAST

TRIKE

Stages of PASTA

1. **Define the Objectives**
2. Define the Technical Scope
3. Decompose the Application
4. Analyze the Threats
5. Vulnerability Analysis
6. Attack Analysis
7. Risk and Impact Analysis

PASTA

STRIDE

LINDDUN

Persona non grata

VAST

TRIKE

Stages of PASTA

1. Define the Objectives
2. **Define the Technical Scope**
3. Decompose the Application
4. Analyze the Threats
5. Vulnerability Analysis
6. Attack Analysis
7. Risk and Impact Analysis

PASTA

STRIDE

Stages of PASTA

1. Define the Objectives
2. Define the Technical Scope
3. **Decompose the Application**
4. Analyze the Threats
5. Vulnerability Analysis
6. Attack Analysis
7. Risk and Impact Analysis

LINDDUN

Persona non grata

VAST

TRIKE

Stages of PASTA

1. Define the Objectives
2. Define the Technical Scope
3. Decompose the Application
4. **Analyze the Threats**
5. Vulnerability Analysis
6. Attack Analysis
7. Risk and Impact Analysis

PASTA

STRIDE

LINDDUN
Persona non grata
VAST
TRIKE
PASTA
STRIDE

Stages of PASTA

1. Define the Objectives
2. Define the Technical Scope
3. Decompose the Application
4. Analyze the Threats
5. **Vulnerability Analysis**
6. Attack Analysis
7. Risk and Impact Analysis

Stages of PASTA

1. Define the Objectives
2. Define the Technical Scope
3. Decompose the Application
4. Analyze the Threats
5. Vulnerability Analysis
6. **Attack Analysis**
7. Risk and Impact Analysis

LINDDUN

Persona non grata

VAST

TRIKE

PASTA

STRIDE

LINDDUN

Persona non grata

VAST

TRIKE

Stages of PASTA

1. Define the Objectives
2. Define the Technical Scope
3. Decompose the Application
4. Analyze the Threats
5. Vulnerability Analysis
6. Attack Analysis
7. **Risk and Impact Analysis**

PASTA

STRIDE

LINDDUN
Persona non grata
VAST

Trike

An open-source tool available as a spreadsheet template or stand-alone program, Trike consists of a matrix combining assets, actors, actions, and rules. When parameters and data are entered in this matrix, the program produces a score-based analysis of risks and probabilities.

TRIKE
PASTA
STRIDE

LINDDUN
Persona non grata
VAST

Trike

Actor\Asset	Data	Computing systems
External	disallowed	disallowed
User	CRU	disallowed
Internal	CRU	CRU
Admin	CRUD	CRUD

Step 2 : allowed action, disallowed action

Step 3 : creating, reading, updating, และ deleting

TRIKE
PASTA
STRIDE

LINDDUN

Persona non grata

VAST

VAST (visual, agile, and simple threat) modeling consists of methods and processes that can be easily scaled and adapted to any scope or part of an organization. The results produce benchmarks that can be used to make reliable comparisons and measurements of effective risk across a whole organization.

VAST

TRIKE

PASTA

STRIDE

LINDDUN

Persona non grata

This method is similar to criminal profiling in law enforcement. To anticipate attacks in more detail, brainstorming exercises are performed to create a detailed picture of a hypothetical attacker, including their psychology, motivations, goals, and capabilities.

Persona non grata

VAST

TRIKE

PASTA

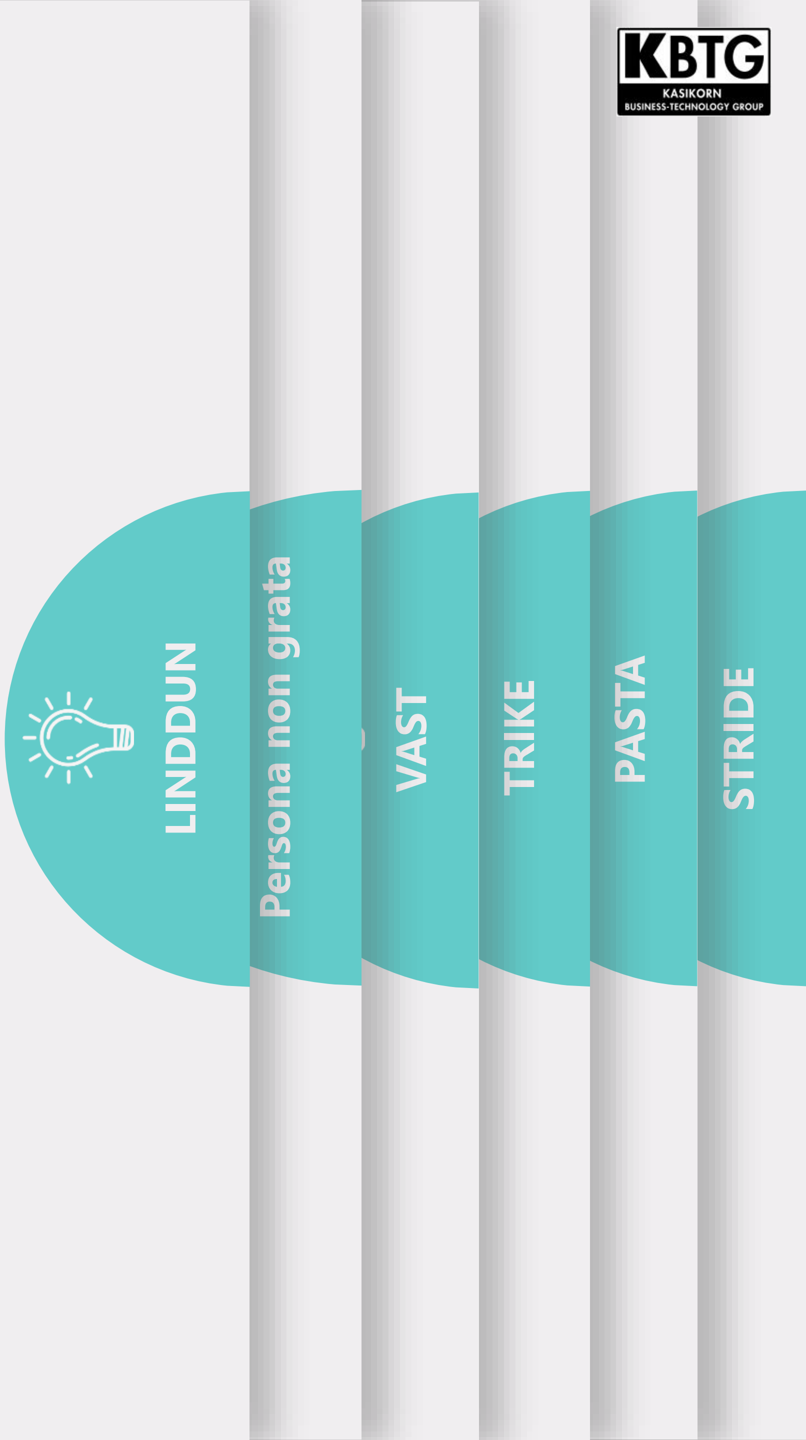
STRIDE

LINDDUN

The LINDDUN framework focuses on analysis of privacy threats, based on the categories that form its acronym: linkability, identifiability, non-repudiation, detectability, disclosure of information, unawareness, and non-compliance. It uses threat trees to help users choose the relevant privacy controls to apply.

LINDDUN

- Linkability
- Identifiability
- Non-repudiation
- Detectability
- Disclosure of information
- Unawareness
- Noncompliance



LINDDUN

- **Linkability**
- Identifiability
- Non-repudiation
- Detectability
- Disclosure of information
- Unawareness
- Noncompliance

LINDDUN

Persona non grata

VAST

TRIKE

PASTA

STRIDE

LINDDUN

- Linkability
- **Identifiability**
- Non-repudiation
- Detectability
- Disclosure of information
- Unawareness
- Noncompliance

LINDDUN

- Linkability
- Identifiability
- **Non-repudiation**
- Detectability
- Disclosure of information
- Unawareness
- Noncompliance

LINDDUN

- Linkability
- Identifiability
- Non-repudiation
- **Detectability**
- Disclosure of information
- Unawareness
- Noncompliance

LINDDUN

- Linkability
- Identifiability
- Non-repudiation
- Detectability
- **Disclosure of information**
- Unawareness
- Noncompliance

LINDDUN

Persona non grata

VAST

TRIKE

PASTA

STRIDE

LINDDUN

- Linkability
- Identifiability
- Non-repudiation
- Detectability
- Disclosure of information
- **Unawareness**
- Noncompliance

LINDDUN

Persona non grata

VAST

TRIKE

PASTA

STRIDE

LINDDUN

- Linkability
- Identifiability
- Non-repudiation
- Detectability
- Disclosure of information
- Unawareness
- **Noncompliance**

LINDDUN

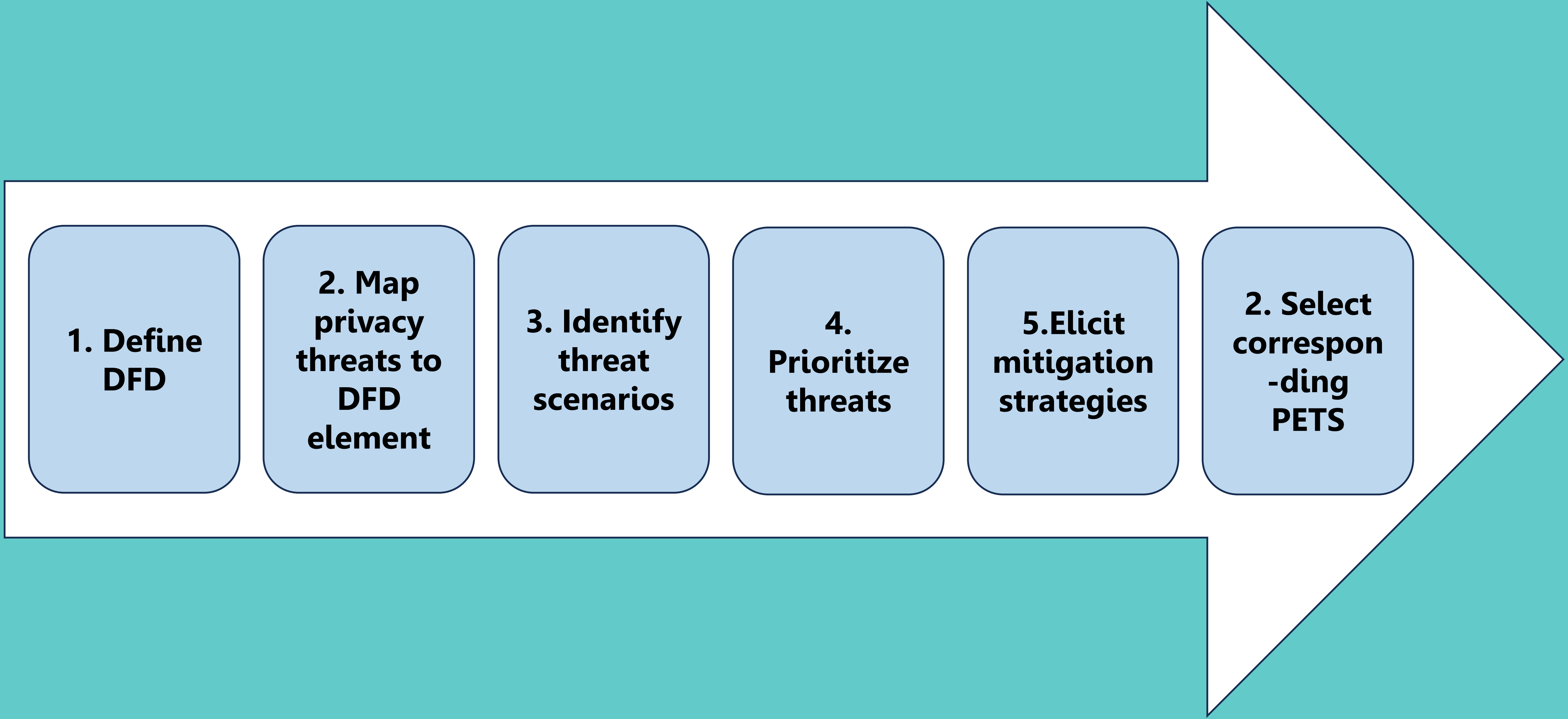
Persona non grata

VAST

TRIKE

PASTA

STRIDE



1. Define DFD

2. Map privacy threats to DFD element

3. Identify threat scenarios

4. Prioritize threats

5.Elicit mitigation strategies

2. Select correspon -ding PETS

Key take away 2

What is the benefits of Threat modeling?

4 : OWASP

1 : Overview

2 : Application Security Protection

3 : Threat Modeling

4 : OWASP

4.1 : OWASP - Web security

4.2 : OWASP - API Security

4.3 : OWASP - Mobile security

4.4 : OWASP - Proactive Control

OWASP

What is OWASP ?

nonprofit organization known as the Open Web Application Security Project, is committed to enhancing software security. It offers resources, tools, and guidelines to assist organizations in creating, deploying, and maintaining secure web applications and services.

OWASP Project

- Top Ten Project
- Projects and Tools
- Guides and Documentation
- Training and Events
- Web Application Security Testing
- Community and Chapters
- Industry Standards
- Vulnerability Database

4.1 : OWASP Web App Security

1 : Overview

2 : Application Security Protection

3 : Threat Modeling

4 : OWASP

4.1 : OWASP – Web security

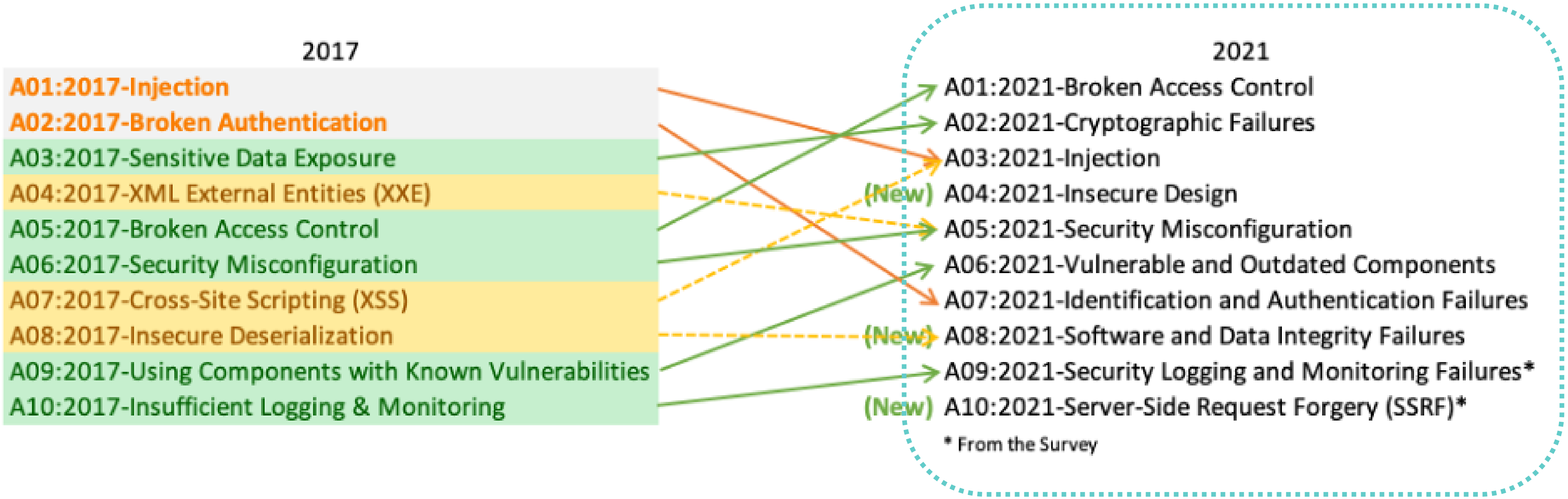
4.2 : OWASP - API Security

4.3 : OWASP - Mobile security

4.4 : OWASP - Proactive Control

OWASP Top 10 Web Application Security

OWASP Top 10 Web Application Security



2017

A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring

2021

A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*

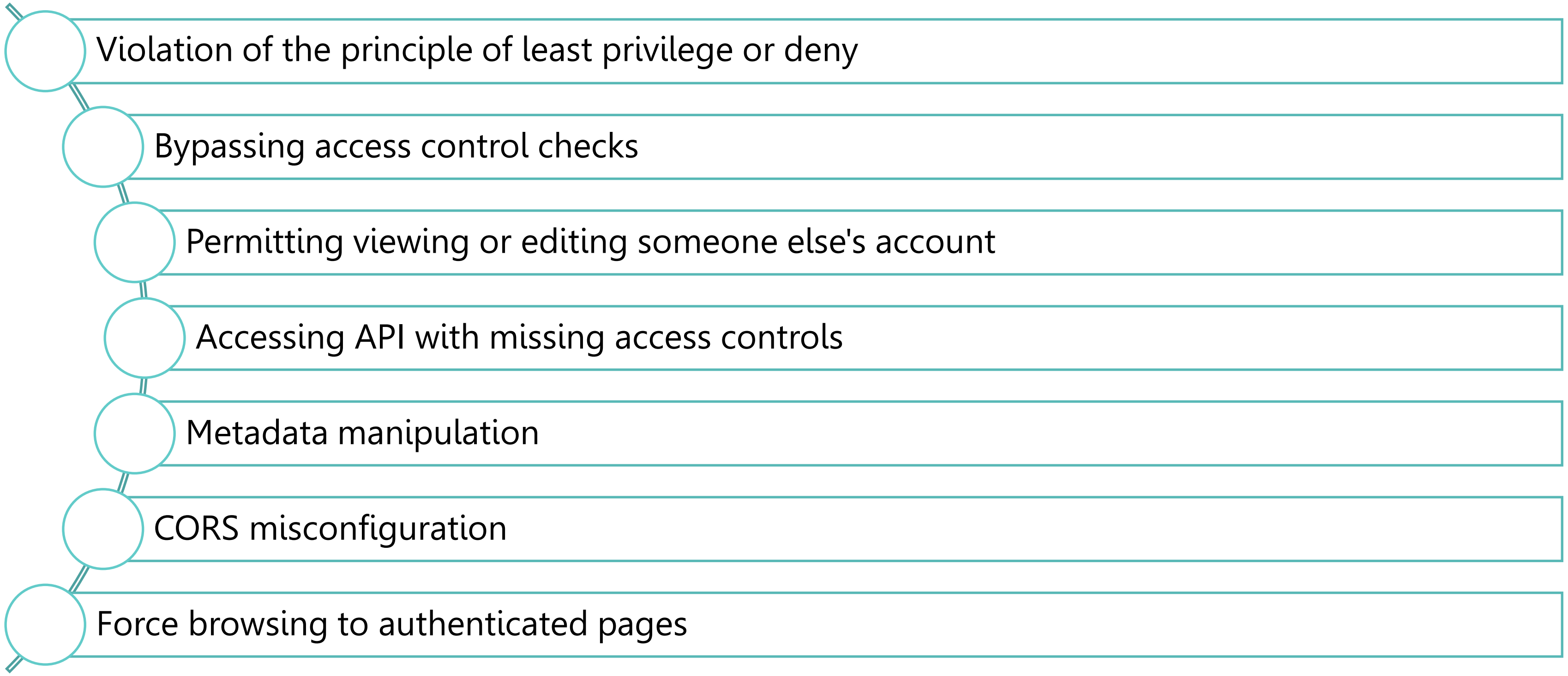
* From the Survey

https://owasp.org/www-project-top-ten/

OWASP Top 10 Web Application Security

1. Broken Access Control

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.

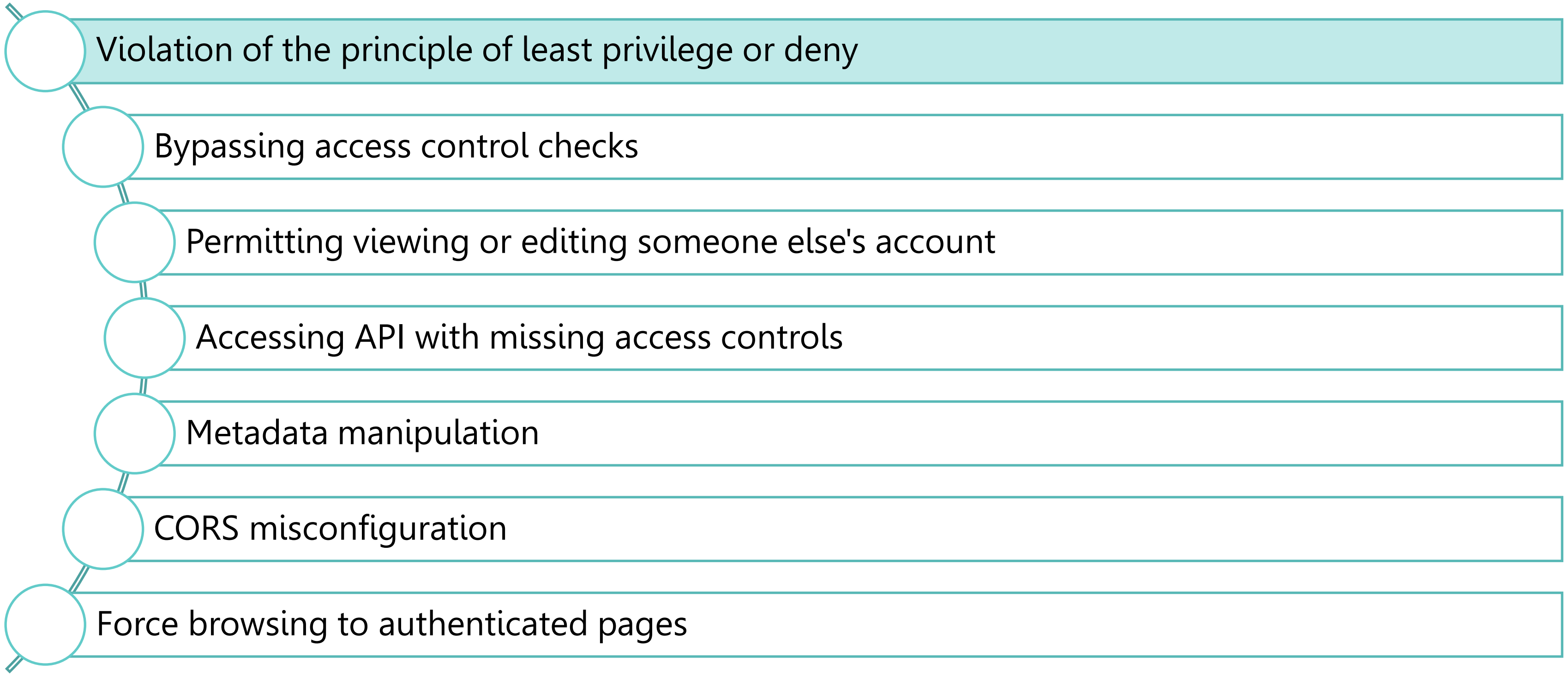


- Violation of the principle of least privilege or deny
- Bypassing access control checks
- Permitting viewing or editing someone else's account
- Accessing API with missing access controls
- Metadata manipulation
- CORS misconfiguration
- Force browsing to authenticated pages

OWASP Top 10 Web Application Security

1. Broken Access Control

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.

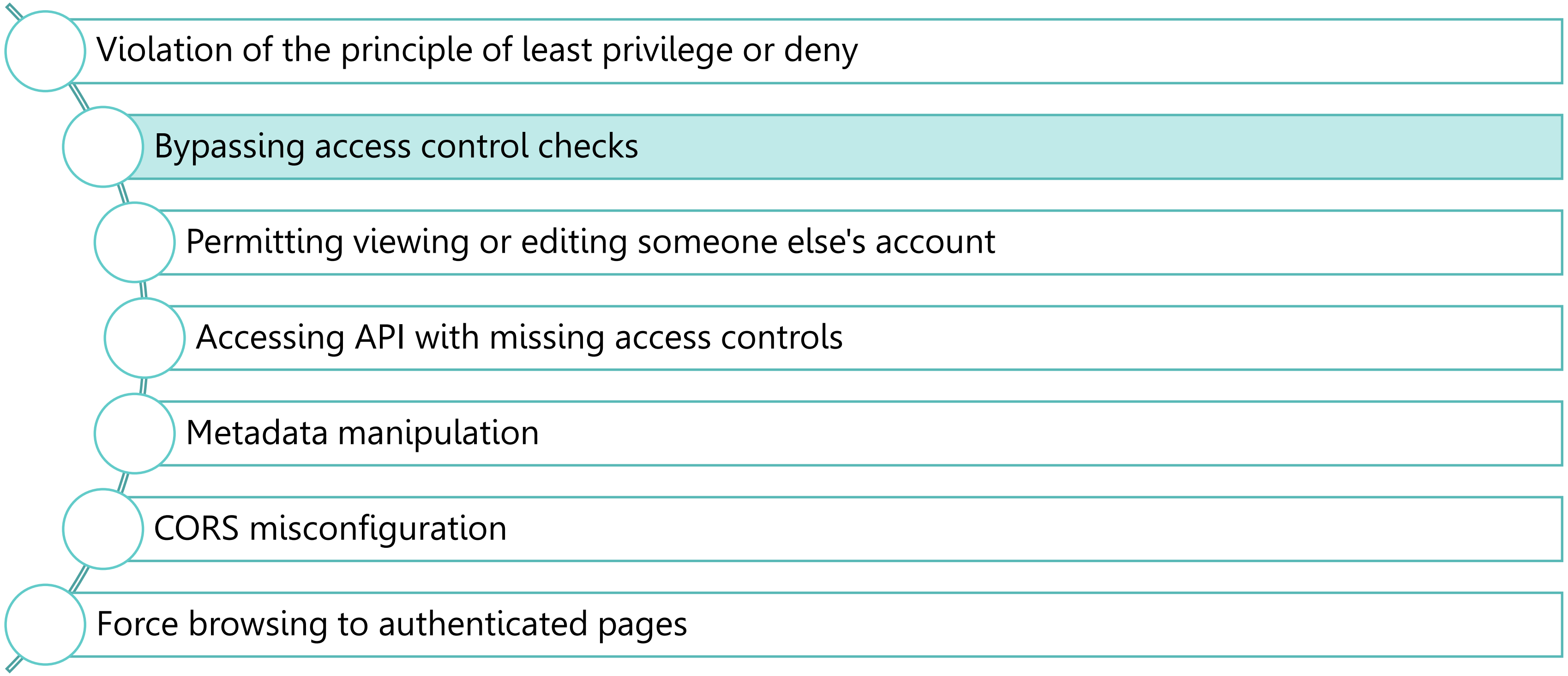


- Violation of the principle of least privilege or deny
- Bypassing access control checks
- Permitting viewing or editing someone else's account
- Accessing API with missing access controls
- Metadata manipulation
- CORS misconfiguration
- Force browsing to authenticated pages

OWASP Top 10 Web Application Security

1. Broken Access Control

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.



- Violation of the principle of least privilege or deny
- Bypassing access control checks
- Permitting viewing or editing someone else's account
- Accessing API with missing access controls
- Metadata manipulation
- CORS misconfiguration
- Force browsing to authenticated pages

OWASP Top 10 Web Application Security

1. Broken Access Control

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.

- Violation of the principle of least privilege or deny
- Bypassing access control checks
- Permitting viewing or editing someone else's account
- Accessing API with missing access controls
- Metadata manipulation
- CORS misconfiguration
- Force browsing to authenticated pages

OWASP Top 10 Web Application Security

1. Broken Access Control

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.

- Violation of the principle of least privilege or deny
- Bypassing access control checks
- Permitting viewing or editing someone else's account
- Accessing API with missing access controls
- Metadata manipulation
- CORS misconfiguration
- Force browsing to authenticated pages

OWASP Top 10 Web Application Security

1. Broken Access Control

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.

- Violation of the principle of least privilege or deny
- Bypassing access control checks
- Permitting viewing or editing someone else's account
- Accessing API with missing access controls
- Metadata manipulation
- CORS misconfiguration
- Force browsing to authenticated pages

OWASP Top 10 Web Application Security

1. Broken Access Control

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.

- Violation of the principle of least privilege or deny
- Bypassing access control checks
- Permitting viewing or editing someone else's account
- Accessing API with missing access controls
- Metadata manipulation
- CORS misconfiguration
- Force browsing to authenticated pages

OWASP Top 10 Web Application Security

1. Broken Access Control

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.

- Violation of the principle of least privilege or deny
- Bypassing access control checks
- Permitting viewing or editing someone else's account
- Accessing API with missing access controls
- Metadata manipulation
- CORS misconfiguration
- Force browsing to authenticated pages

OWASP Top 10 Web Application Security

2. Cryptographic Failures

The first thing is to determine the protection needs of data in transit and at rest. For example, passwords, credit card numbers, health records, personal information, and business secrets require extra protection, mainly if that data falls under privacy laws, e.g., EU's General Data Protection Regulation (GDPR), or regulations, e.g., financial data protection such as PCI Data Security Standard (PCI DSS).

3. Injection

An application is vulnerable to attack when:

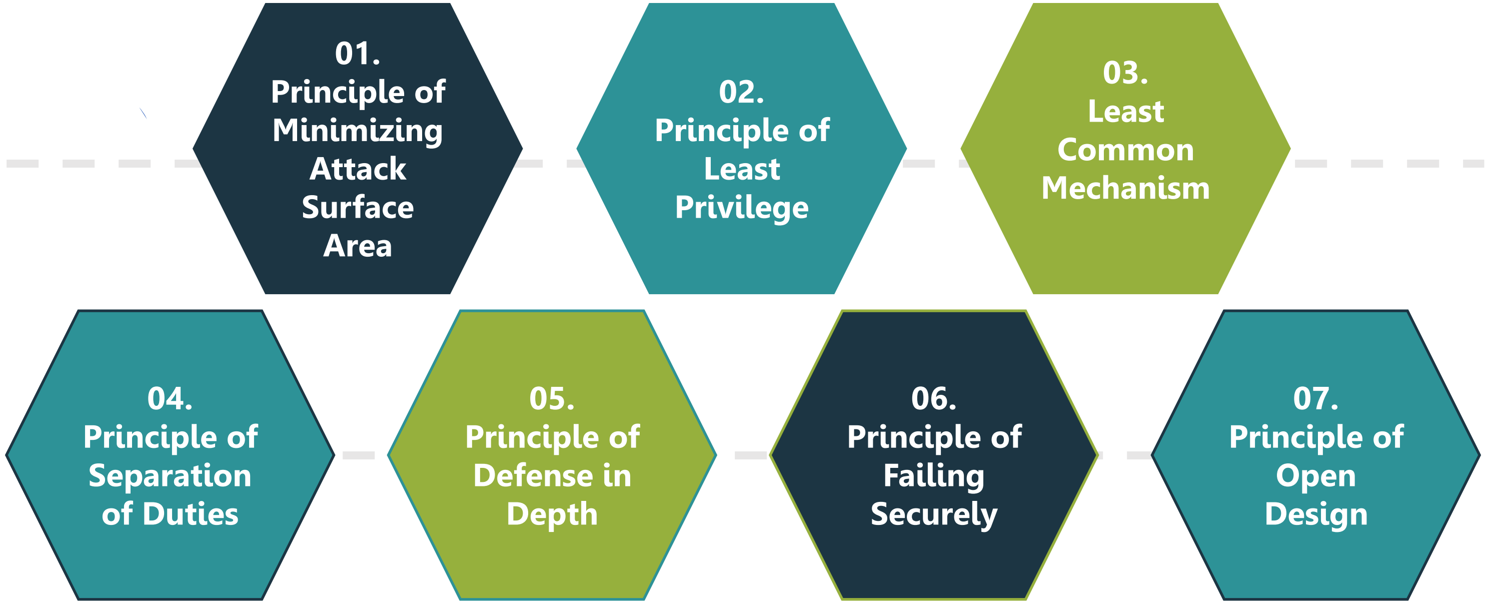
- User-supplied data is not validated, filtered, or sanitized by the application.
- Dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter.
- Hostile data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records.
- Hostile data is directly used or concatenated. The SQL or command contains the structure and malicious data in dynamic queries, commands, or stored procedures.

OWASP Top 10 Web Application Security

4. Insecure Design

Insecure design is a broad category representing different weaknesses, expressed as “missing or ineffective control design. An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks.

5. Secure Design


01. Principle of Minimizing Attack Surface Area
02. Principle of Least Privilege
03. Least Common Mechanism
04. Principle of Separation of Duties
05. Principle of Defense in Depth
06. Principle of Failing Securely
07. Principle of Open Design

OWASP Top 10 Web Application Security

5. Security Misconfiguration

The application might be vulnerable if the application is:

- Missing appropriate security hardening across any part of the application stack or improperly configured permissions on cloud services.
- Unnecessary features are enabled or installed
- Default accounts and their passwords are still enabled and unchanged.
- Error handling reveals stack traces or other overly informative error messages to users.
- For upgraded systems, the latest security features are disabled or not configured securely.
- The security settings in the application servers, application frameworks, libraries, databases, etc., are not set to secure values.
- The server does not send security headers or directives, or they are not set to secure values

OWASP Top 10 Web Application Security

5. Security Misconfiguration

The application might be vulnerable if the application is:

- Missing appropriate security hardening across any part of the application stack or improperly configured permissions on cloud services.
- Unnecessary features are enabled or installed
- Default accounts and their passwords are still enabled and unchanged.
- Error handling reveals stack traces or other overly informative error messages to users.
- For upgraded systems, the latest security features are disabled or not configured securely.
- The security settings in the application servers, application frameworks, libraries, databases, etc., are not set to secure values.
- The server does not send security headers or directives, or they are not set to secure values

OWASP Top 10 Web Application Security

5. Security Misconfiguration

The application might be vulnerable if the application is:

- Missing appropriate security hardening across any part of the application stack or improperly configured permissions on cloud services.
- Unnecessary features are enabled or installed
- Default accounts and their passwords are still enabled and unchanged.
- Error handling reveals stack traces or other overly informative error messages to users.
- For upgraded systems, the latest security features are disabled or not configured securely.
- The security settings in the application servers, application frameworks, libraries, databases, etc., are not set to secure values.
- The server does not send security headers or directives, or they are not set to secure values

OWASP Top 10 Web Application Security

5. Security Misconfiguration

The application might be vulnerable if the application is:

- Missing appropriate security hardening across any part of the application stack or improperly configured permissions on cloud services.
- Unnecessary features are enabled or installed
- Default accounts and their passwords are still enabled and unchanged.
- Error handling reveals stack traces or other overly informative error messages to users.
- For upgraded systems, the latest security features are disabled or not configured securely.
- The security settings in the application servers, application frameworks, libraries, databases, etc., are not set to secure values.
- The server does not send security headers or directives, or they are not set to secure values

OWASP Top 10 Web Application Security

5. Security Misconfiguration

The application might be vulnerable if the application is:

- Missing appropriate security hardening across any part of the application stack or improperly configured permissions on cloud services.
- Unnecessary features are enabled or installed
- Default accounts and their passwords are still enabled and unchanged.
- Error handling reveals stack traces or other overly informative error messages to users.
- For upgraded systems, the latest security features are disabled or not configured securely.
- The security settings in the application servers, application frameworks, libraries, databases, etc., are not set to secure values.
- The server does not send security headers or directives, or they are not set to secure values

OWASP Top 10 Web Application Security

5. Security Misconfiguration

The application might be vulnerable if the application is:

- Missing appropriate security hardening across any part of the application stack or improperly configured permissions on cloud services.
- Unnecessary features are enabled or installed
- Default accounts and their passwords are still enabled and unchanged.
- Error handling reveals stack traces or other overly informative error messages to users.
- For upgraded systems, the latest security features are disabled or not configured securely.
- The security settings in the application servers, application frameworks, libraries, databases, etc., are not set to secure values.
- The server does not send security headers or directives, or they are not set to secure values

OWASP Top 10 Web Application Security

5. Security Misconfiguration

The application might be vulnerable if the application is:

- Missing appropriate security hardening across any part of the application stack or improperly configured permissions on cloud services.
- Unnecessary features are enabled or installed
- Default accounts and their passwords are still enabled and unchanged.
- Error handling reveals stack traces or other overly informative error messages to users.
- For upgraded systems, the latest security features are disabled or not configured securely.
- The security settings in the application servers, application frameworks, libraries, databases, etc., are not set to secure values.
- The server does not send security headers or directives, or they are not set to secure values

OWASP Top 10 Web Application Security

5. Security Misconfiguration

The application might be vulnerable if the application is:

- Missing appropriate security hardening across any part of the application stack or improperly configured permissions on cloud services.
- Unnecessary features are enabled or installed
- Default accounts and their passwords are still enabled and unchanged.
- Error handling reveals stack traces or other overly informative error messages to users.
- For upgraded systems, the latest security features are disabled or not configured securely.
- The security settings in the application servers, application frameworks, libraries, databases, etc., are not set to secure values.
- The server does not send security headers or directives, or they are not set to secure values

OWASP Top 10 Web Application Security

6. Vulnerable and Outdated Components

Components with known vulnerabilities, such as CVEs, should be identified and patched, whereas stale or malicious components should be evaluated for viability and the risk they may introduce.

7. Identification and Authentication Failures

Confirmation of the user's identity, authentication, and session management is critical to protect against authentication-related attacks.

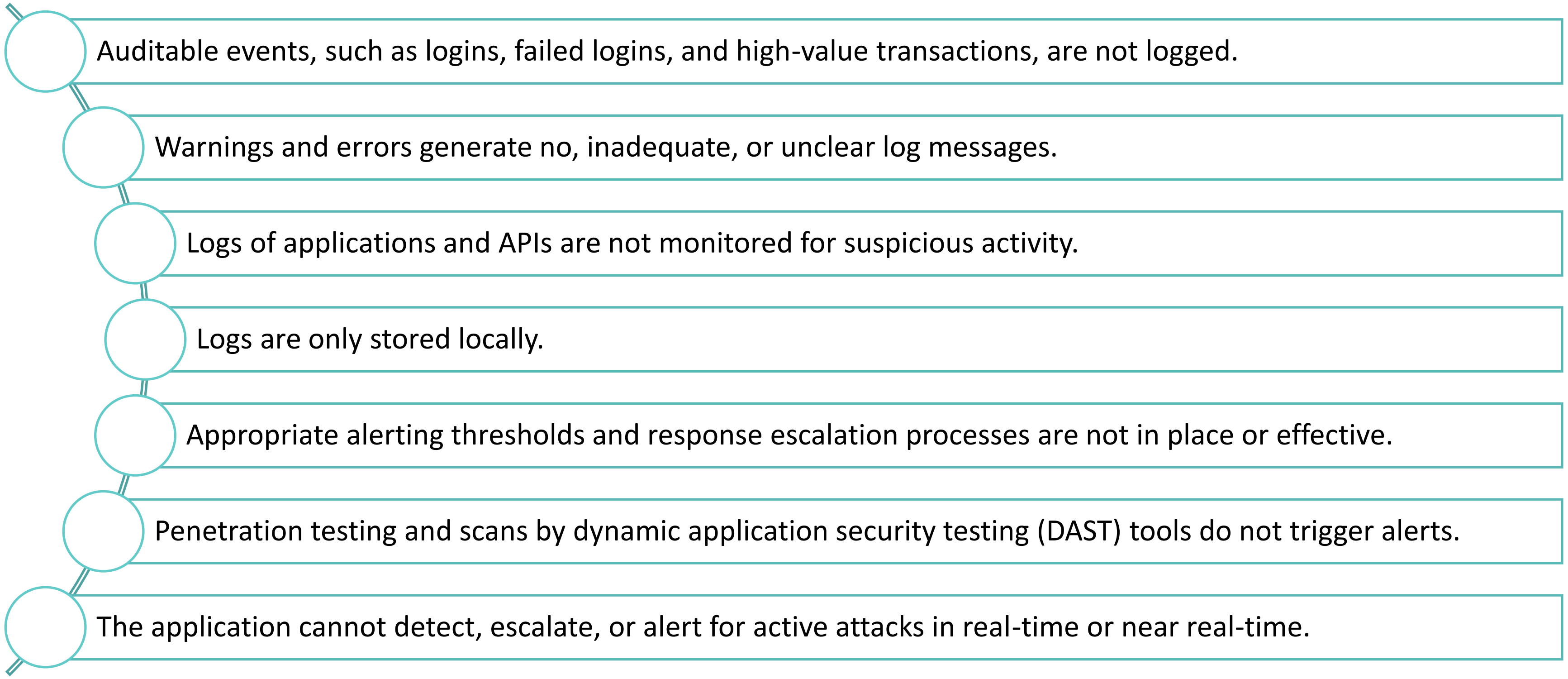
8. Software and Data Integrity Failures

Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs). An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise.

OWASP Top 10 Web Application Security

9. Security Logging and Monitoring Failures

Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time



- Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.
- Penetration testing and scans by dynamic application security testing (DAST) tools do not trigger alerts.
- The application cannot detect, escalate, or alert for active attacks in real-time or near real-time.

OWASP Top 10 Web Application Security

9. Security Logging and Monitoring Failures

Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time

- Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.
- Penetration testing and scans by dynamic application security testing (DAST) tools do not trigger alerts.
- The application cannot detect, escalate, or alert for active attacks in real-time or near real-time.

OWASP Top 10 Web Application Security

9. Security Logging and Monitoring Failures

Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time

- Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.
- Penetration testing and scans by dynamic application security testing (DAST) tools do not trigger alerts.
- The application cannot detect, escalate, or alert for active attacks in real-time or near real-time.

OWASP Top 10 Web Application Security

9. Security Logging and Monitoring Failures

Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time

- Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.
- Penetration testing and scans by dynamic application security testing (DAST) tools do not trigger alerts.
- The application cannot detect, escalate, or alert for active attacks in real-time or near real-time.

OWASP Top 10 Web Application Security

9. Security Logging and Monitoring Failures

Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time

- Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.
- Penetration testing and scans by dynamic application security testing (DAST) tools do not trigger alerts.
- The application cannot detect, escalate, or alert for active attacks in real-time or near real-time.

OWASP Top 10 Web Application Security

9. Security Logging and Monitoring Failures

Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time

- Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.
- Penetration testing and scans by dynamic application security testing (DAST) tools do not trigger alerts.
- The application cannot detect, escalate, or alert for active attacks in real-time or near real-time.

OWASP Top 10 Web Application Security

9. Security Logging and Monitoring Failures

Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time

- Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.
- Penetration testing and scans by dynamic application security testing (DAST) tools do not trigger alerts.
- The application cannot detect, escalate, or alert for active attacks in real-time or near real-time.

OWASP Top 10 Web Application Security

9. Security Logging and Monitoring Failures

Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time

- Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.
- Penetration testing and scans by dynamic application security testing (DAST) tools do not trigger alerts.
- The application cannot detect, escalate, or alert for active attacks in real-time or near real-time.

OWASP Top 10 Web Application Security

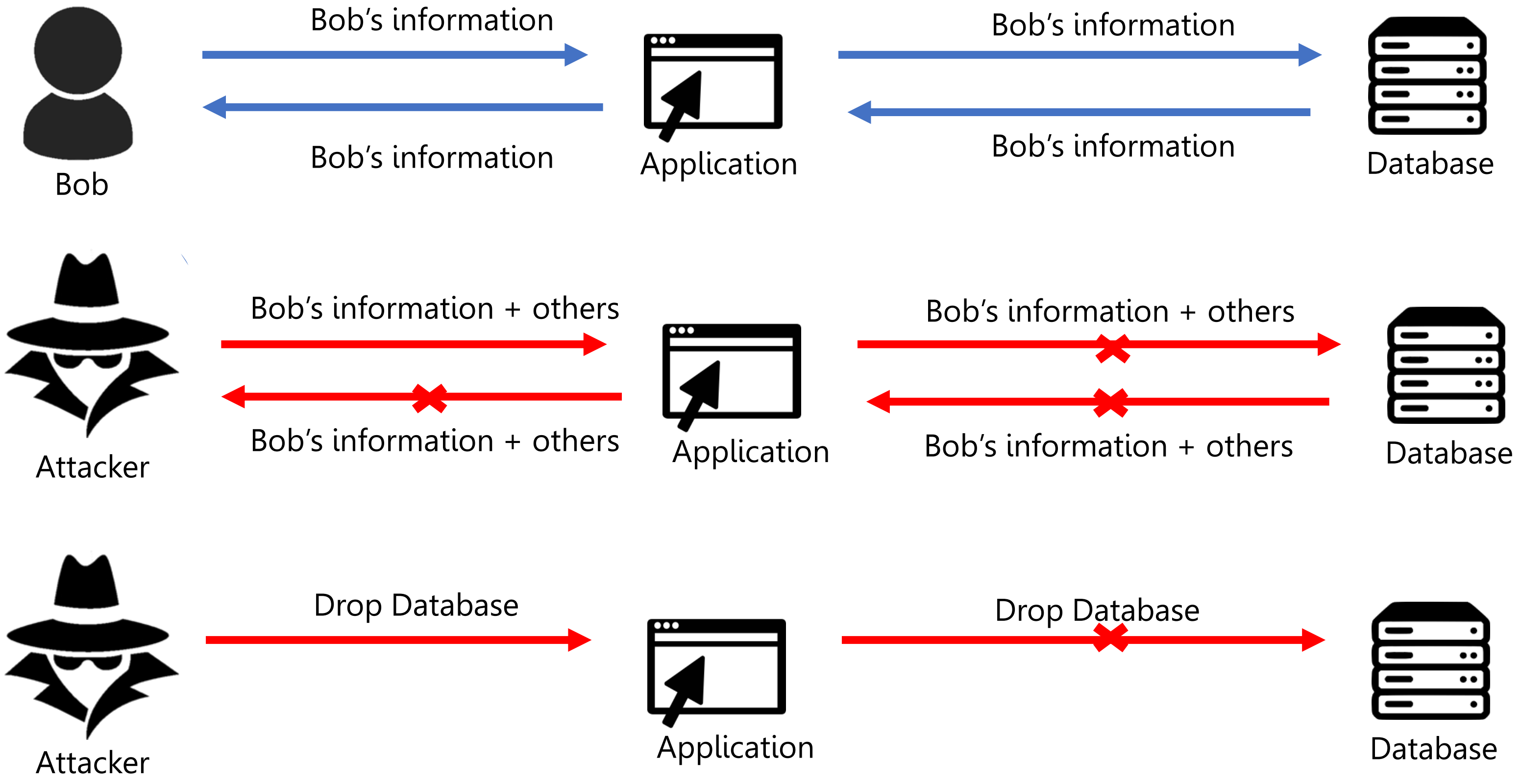
10. Server-Side Request Forgery

SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).

Example

OWASP Top 10 Web Application Security

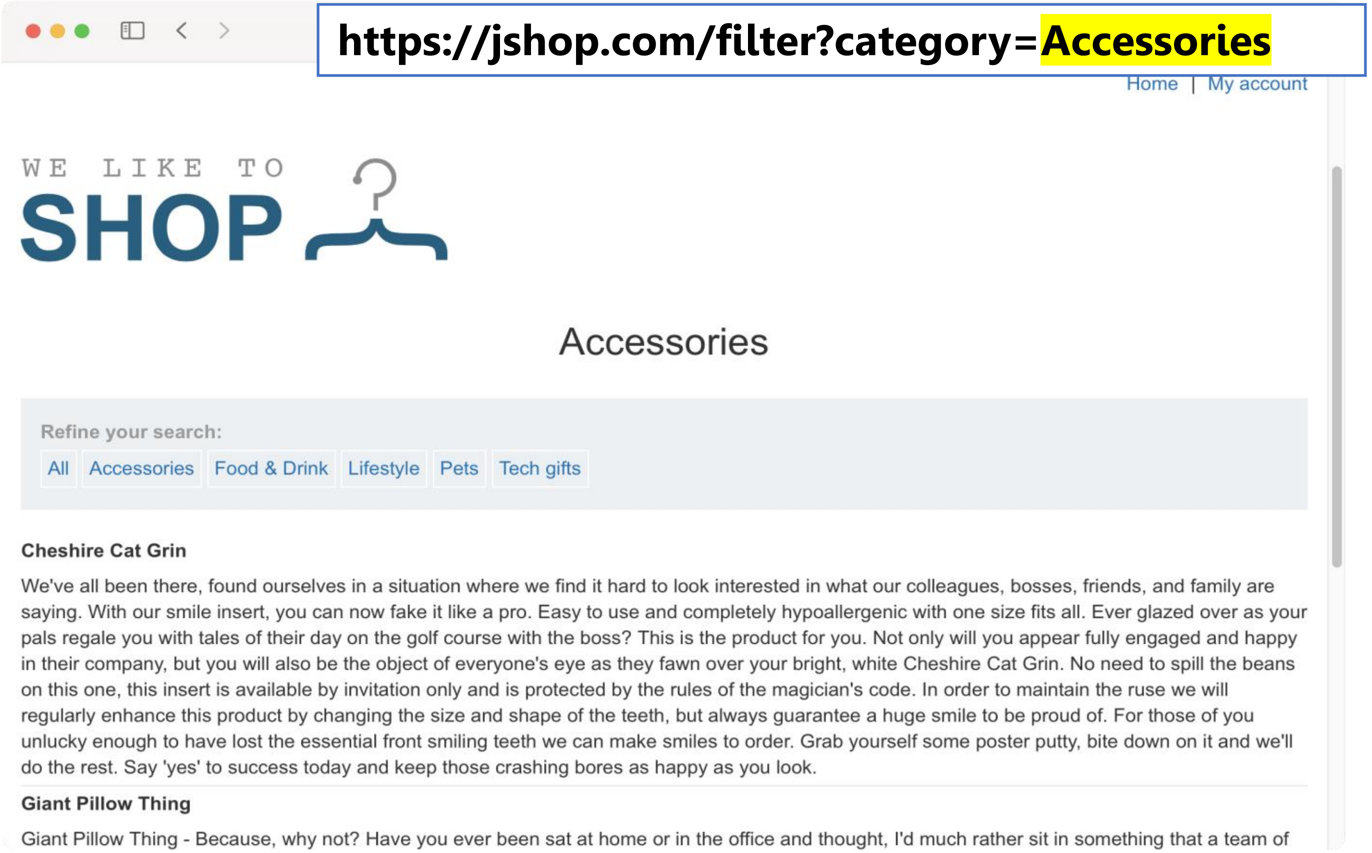
Examples – 3 Injection



OWASP Top 10 Web Application Security

Examples – 3 Injection

https://jshop.com/filter?category=Accessories



Home | My account

WE LIKE TO SHOP

Accessories

Refine your search:

All | Accessories | Food & Drink | Lifestyle | Pets | Tech gifts

Cheshire Cat Grin

We've all been there, found ourselves in a situation where we find it hard to look interested in what our colleagues, bosses, friends, and family are saying. With our smile insert, you can now fake it like a pro. Easy to use and completely hypoallergenic with one size fits all. Ever glazed over as your pals regale you with tales of their day on the golf course with the boss? This is the product for you. Not only will you appear fully engaged and happy in their company, but you will also be the object of everyone's eye as they fawn over your bright, white Cheshire Cat Grin. No need to spill the beans on this one, this insert is available by invitation only and is protected by the rules of the magician's code. In order to maintain the ruse we will regularly enhance this product by changing the size and shape of the teeth, but always guarantee a huge smile to be proud of. For those of you unlucky enough to have lost the essential front smiling teeth we can make smiles to order. Grab yourself some poster putty, bite down on it and we'll do the rest. Say 'yes' to success today and keep those crashing bores as happy as you look.

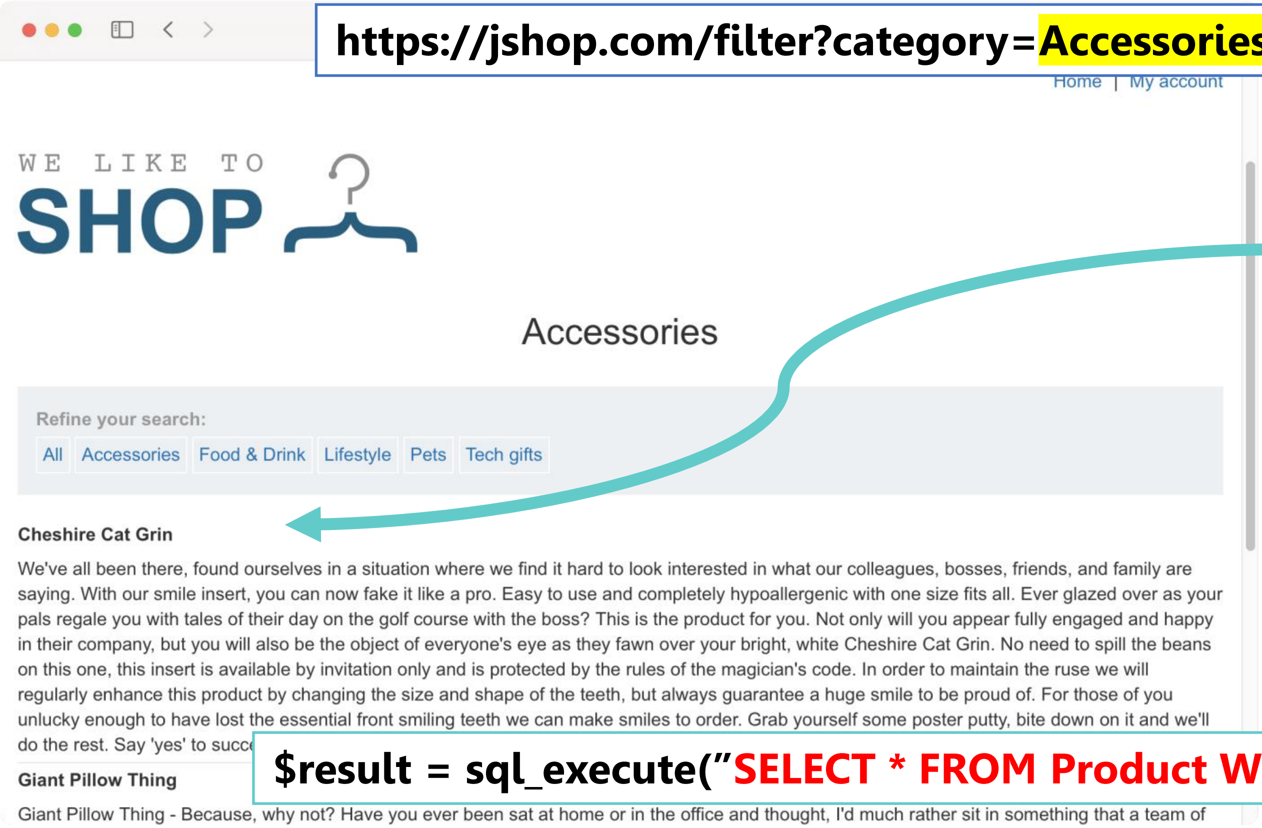
Giant Pillow Thing

Giant Pillow Thing - Because, why not? Have you ever been sat at home or in the office and thought, I'd much rather sit in something that a team of

OWASP Top 10 Web Application Security

Examples – 3 Injection

https://jshop.com/filter?category=Accessories



Home | My account

Accessories

Refine your search:

All | Accessories | Food & Drink | Lifestyle | Pets | Tech gifts

Cheshire Cat Grin

We've all been there, found ourselves in a situation where we find it hard to look interested in what our colleagues, bosses, friends, and family are saying. With our smile insert, you can now fake it like a pro. Easy to use and completely hypoallergenic with one size fits all. Ever glazed over as your pals regale you with tales of their day on the golf course with the boss? This is the product for you. Not only will you appear fully engaged and happy in their company, but you will also be the object of everyone's eye as they fawn over your bright, white Cheshire Cat Grin. No need to spill the beans on this one, this insert is available by invitation only and is protected by the rules of the magician's code. In order to maintain the ruse we will regularly enhance this product by changing the size and shape of the teeth, but always guarantee a huge smile to be proud of. For those of you unlucky enough to have lost the essential front smiling teeth we can make smiles to order. Grab yourself some poster putty, bite down on it and we'll do the rest. Say 'yes' to succ

Giant Pillow Thing

Giant Pillow Thing - Because, why not? Have you ever been sat at home or in the office and thought, I'd much rather sit in something that a team of

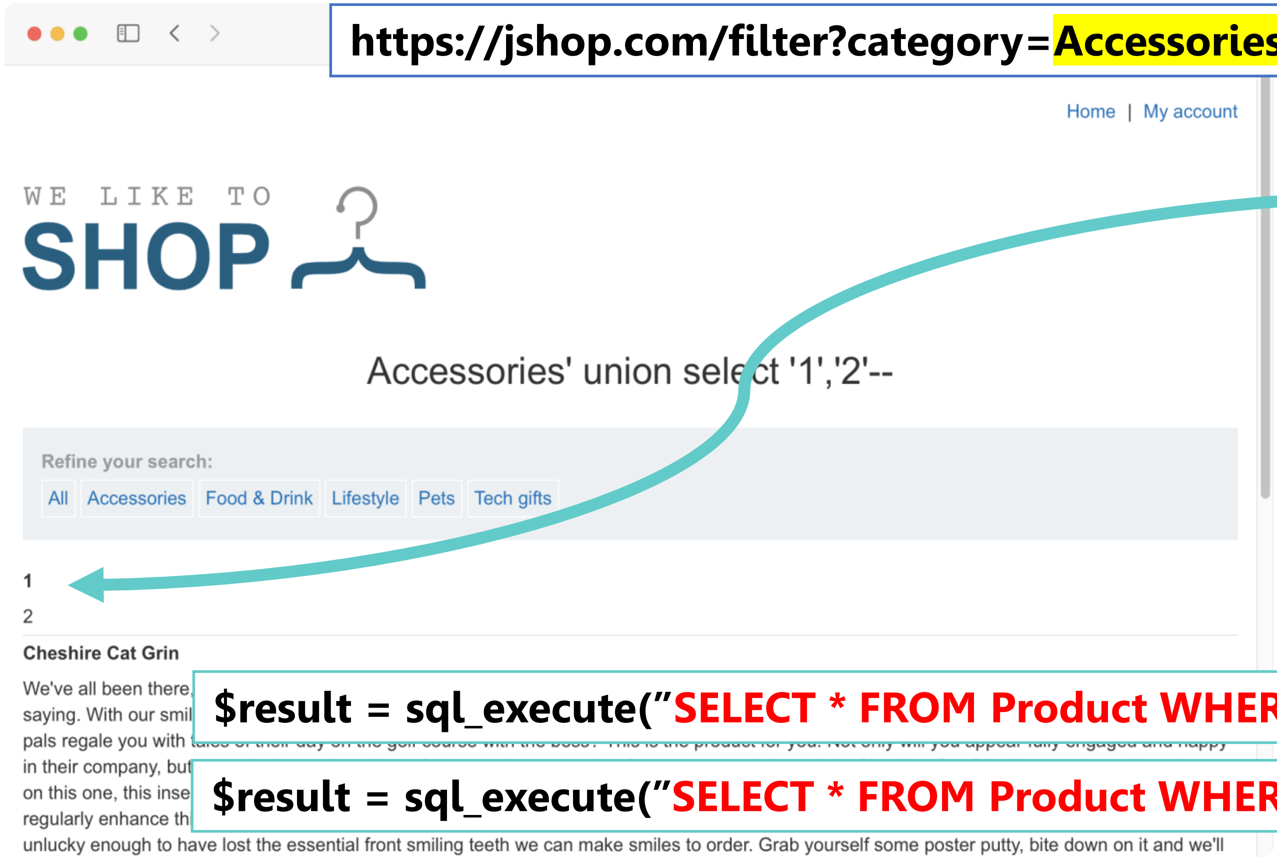
Category	Name	Description
Accessories	Cheshire Cat Grin	We've all been there, found ourselves …
Accessories	Giant Pillow Thing	Giant Pillow Thing – Because, why not …

```
$result = sql_execute(“SELECT * FROM Product WHERE Category =‘ ” + $_GET[“category”] + ” ‘ ”);
```

OWASP Top 10 Web Application Security

Examples – 3 Injection

https://jshop.com/filter?category=Accessories' union select '1','2



Accessories' union select '1','2'--

Refine your search:

All | Accessories | Food & Drink | Lifestyle | Pets | Tech gifts

1

2

Cheshire Cat Grin

We've all been there,
saying. With our smil
pals regale you with
in their company, but
on this one, this inse
regularly enhance th
unlucky enough to have lost the essential front smiling teeth we can make smiles to order. Grab yourself some poster putty, bite down on it and we'll

Name	Description
1	2
Cheshire Cat Grin	We've all been there, found ourselves …
Giant Pillow Thing	Giant Pillow Thing – Because, why not …

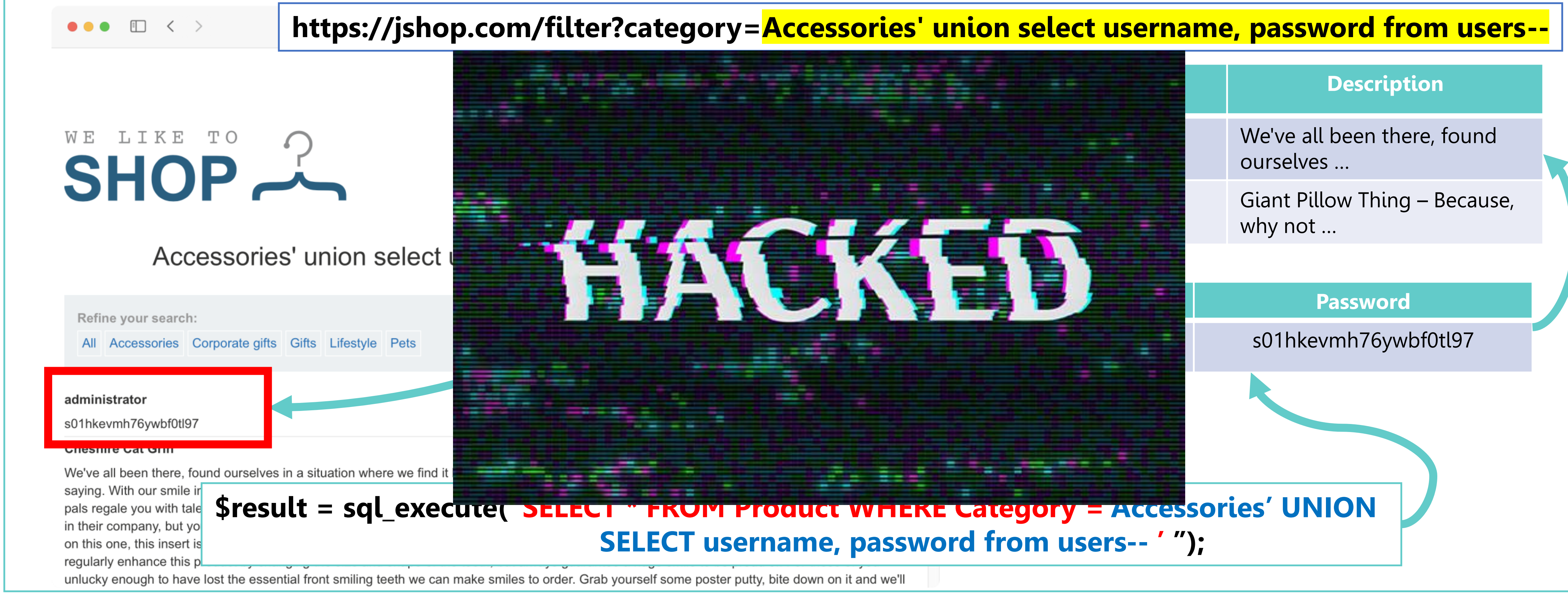
```
$result = sql_execute(”SELECT * FROM Product WHERE Category =‘ ” + $_GET[“category”] + ” ‘ ”);
```

```
$result = sql_execute(”SELECT * FROM Product WHERE Category =‘Accessories’ UNION SELECT ‘1’,’2’ ”);
```

OWASP Top 10 Web Application Security

Examples – 3 Injection

https://jshop.com/filter?category=**Accessories' union select username, password from users--**



WE LIKE TO SHOP

Accessories' union select u

Refine your search:

All | Accessories | Corporate gifts | Gifts | Lifestyle | Pets

administrator
s01hkevmh76ywbf0tl97

Cheshire Cat Grin

We've all been there, found ourselves in a situation where we find it
saying. With our smile ir
pals regale you with tale
in their company, but yo
on this one, this insert is
regularly enhance this p
unlucky enough to have lost the essential front smiling teeth we can make smiles to order. Grab yourself some poster putty, bite down on it and we'll

Description
We've all been there, found ourselves ...
Giant Pillow Thing – Because, why not ...

Password
s01hkevmh76ywbf0tl97

```
$result = sql_execute(‘SELECT * FROM Product WHERE Category = Accessories’ UNION
SELECT username, password from users-- ’ ”);
```

OWASP Top 10 Web Application Security

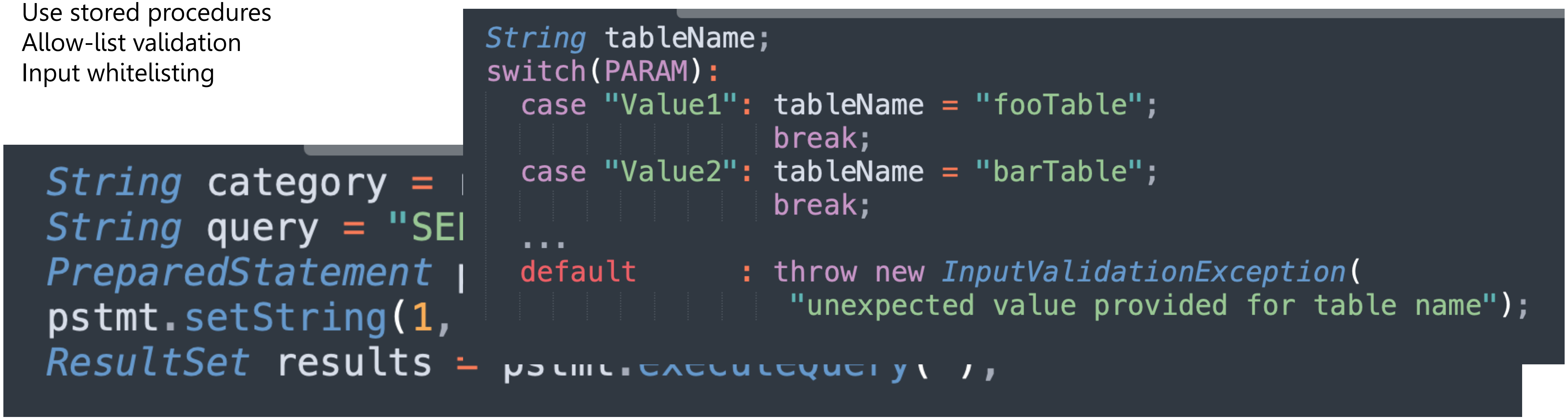
Examples – 3 Injection

BAD Mitigation

- Input blacklist

GOOD Remediation

- Use prepared statements (with parameterized queries)
- Use stored procedures
- Allow-list validation
- Input whitelisting

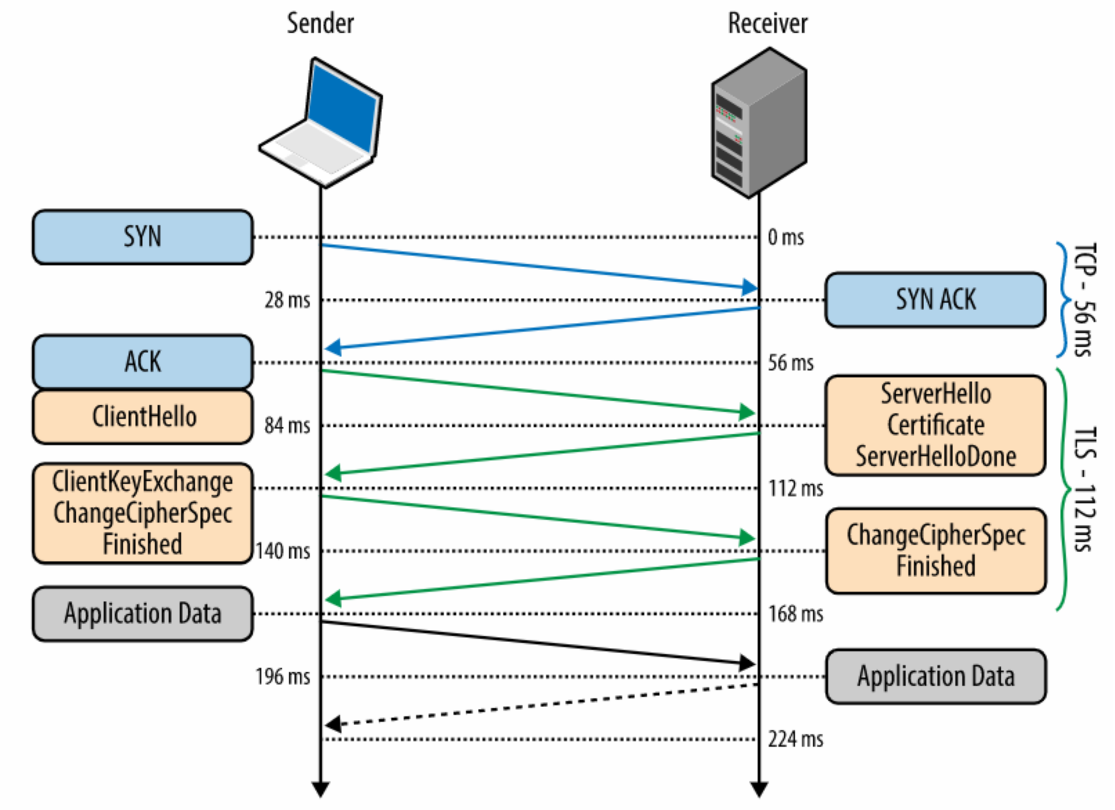


```
String tableName;
switch(PARAM):
  case "Value1": tableName = "fooTable";
                 break;
  case "Value2": tableName = "barTable";
                 break;
  ...
  default      : throw new InputValidationException(
                  "unexpected value provided for table name");
```

```
String category =
String query = "SE
PreparedStatement
pstmt.setString(1,
ResultSet results = pstmt.executeQuery( );
```

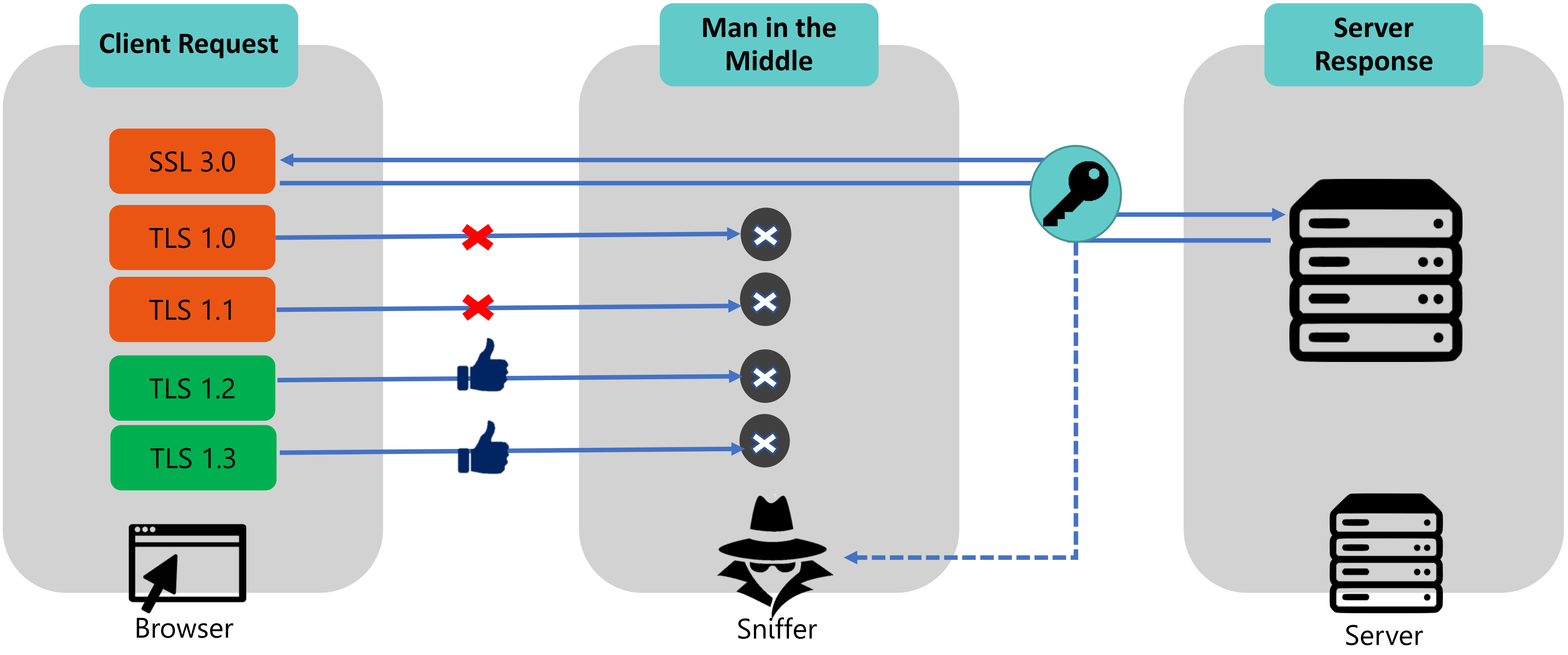
OWASP Top 10 Web Application Security

Examples – 5 Security Misconfiguration - TLS

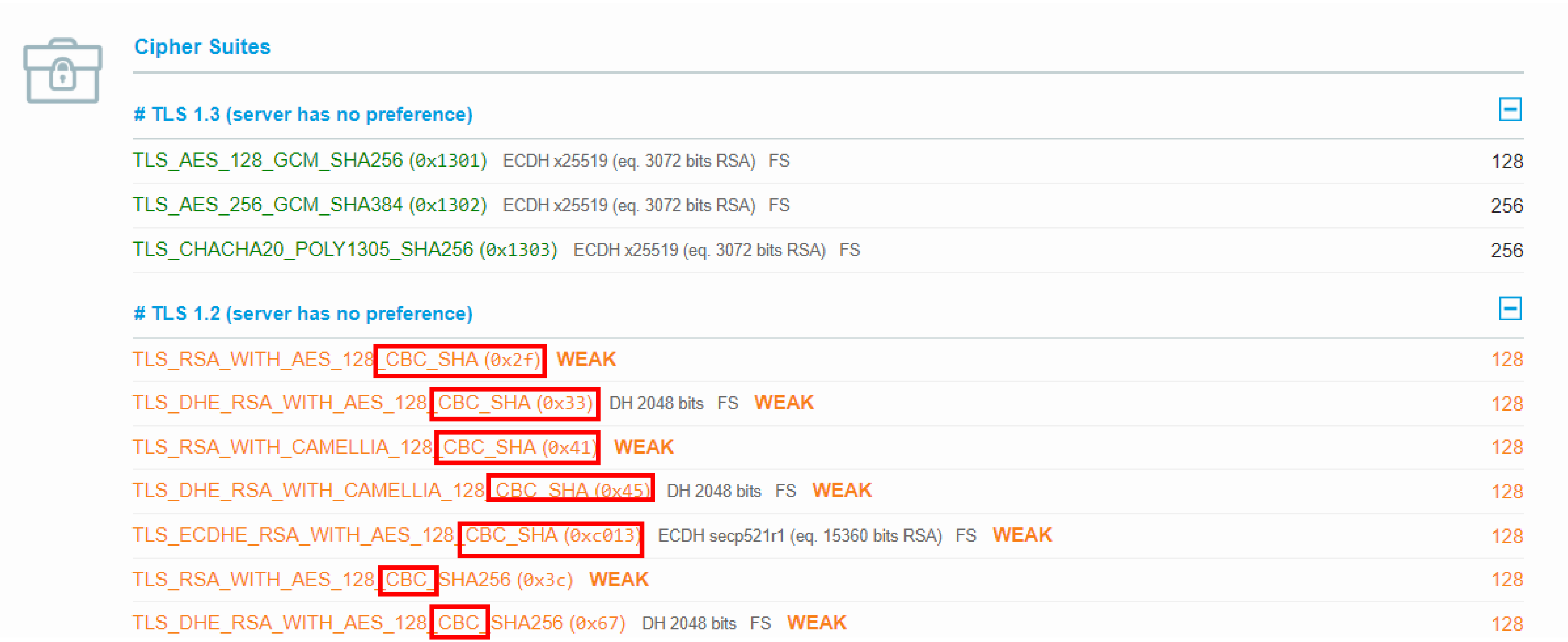


OWASP Top 10 Web Application Security

Examples – 3 Security Misconfiguration - TLS



Examples – 5 Security Misconfiguration - TLS



Cipher Suites

TLS 1.3 (server has no preference)

```
TLS_AES_128_GCM_SHA256 (0x1301)   ECDH x25519 (eq. 3072 bits RSA)   FS          128
TLS_AES_256_GCM_SHA384 (0x1302)   ECDH x25519 (eq. 3072 bits RSA)   FS          256
TLS_CHACHA20_POLY1305_SHA256 (0x1303)   ECDH x25519 (eq. 3072 bits RSA)   FS    256
```

TLS 1.2 (server has no preference)

```
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)   WEAK                                                  128
TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33)   DH 2048 bits   FS   WEAK                          128
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (0x41)   WEAK                                             128
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA (0x45)   DH 2048 bits   FS   WEAK                     128
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)   ECDH secp521r1 (eq. 15360 bits RSA)   FS   WEAK   128
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)   WEAK                                               128
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x67)   DH 2048 bits   FS   WEAK                       128
```

OWASP Top 10 Web Application Security

Examples – 5 Security Misconfiguration – Security Header

Required Headers

1. Strict-Transport-Security
2. X-Content-Type-Options
3. Cache-Control
4. Set-Cookie (Secure, HTTPOnly)
5. Expires
6. X-Frame-Options

Header	Required HTTP 1.1 (HTTPS)	Required HTTP 1.1 (non-HTTPS)	Required HTTP 1.0 (HTTPS)	Required HTTP 1.0 (non-HTTPS)
HTTP Strict-Transport-Security	TRUE		TRUE	
X-Content-Type-Options	TRUE	TRUE	TRUE	TRUE
Cache-Control	TRUE	TRUE		
Set-Cookie		TRUE		TRUE
Expires			TRUE	TRUE
X-Frame-Options			TRUE	TRUE

OWASP Top 10 Web Application Security

Examples – 5 Security Misconfiguration – Security Header

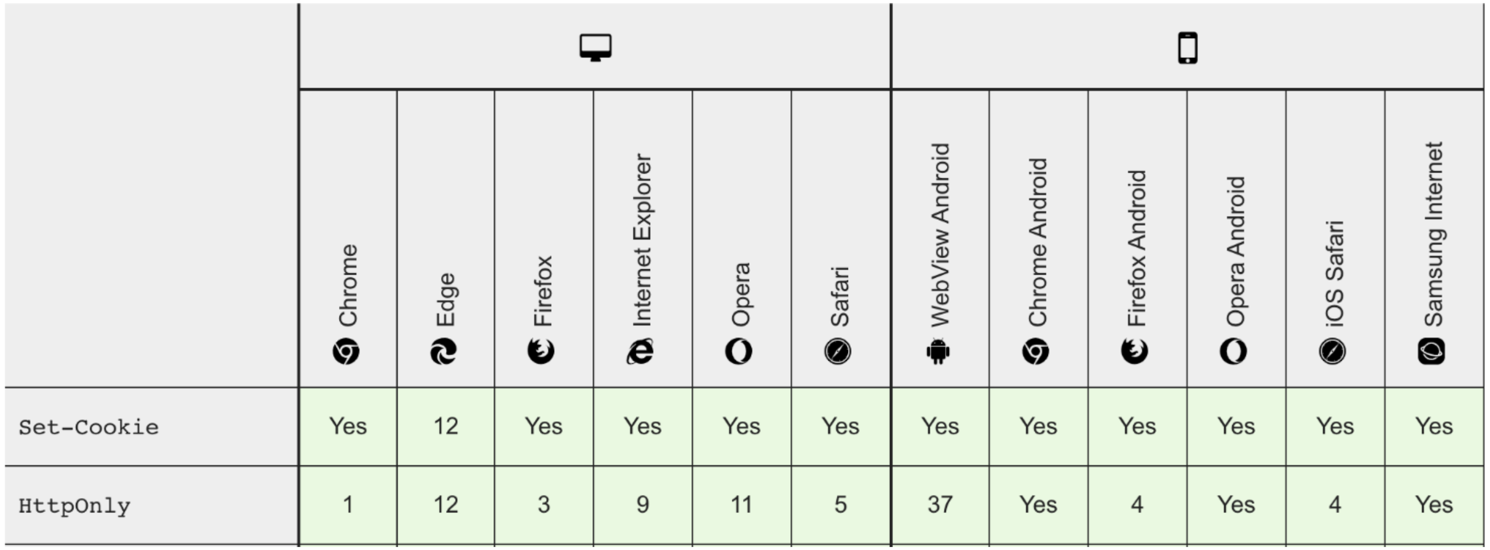
BAD Mitigation

- Configure HTTP Header at the gateway

GOOD Remediation

- Configure HTTP Header at your server

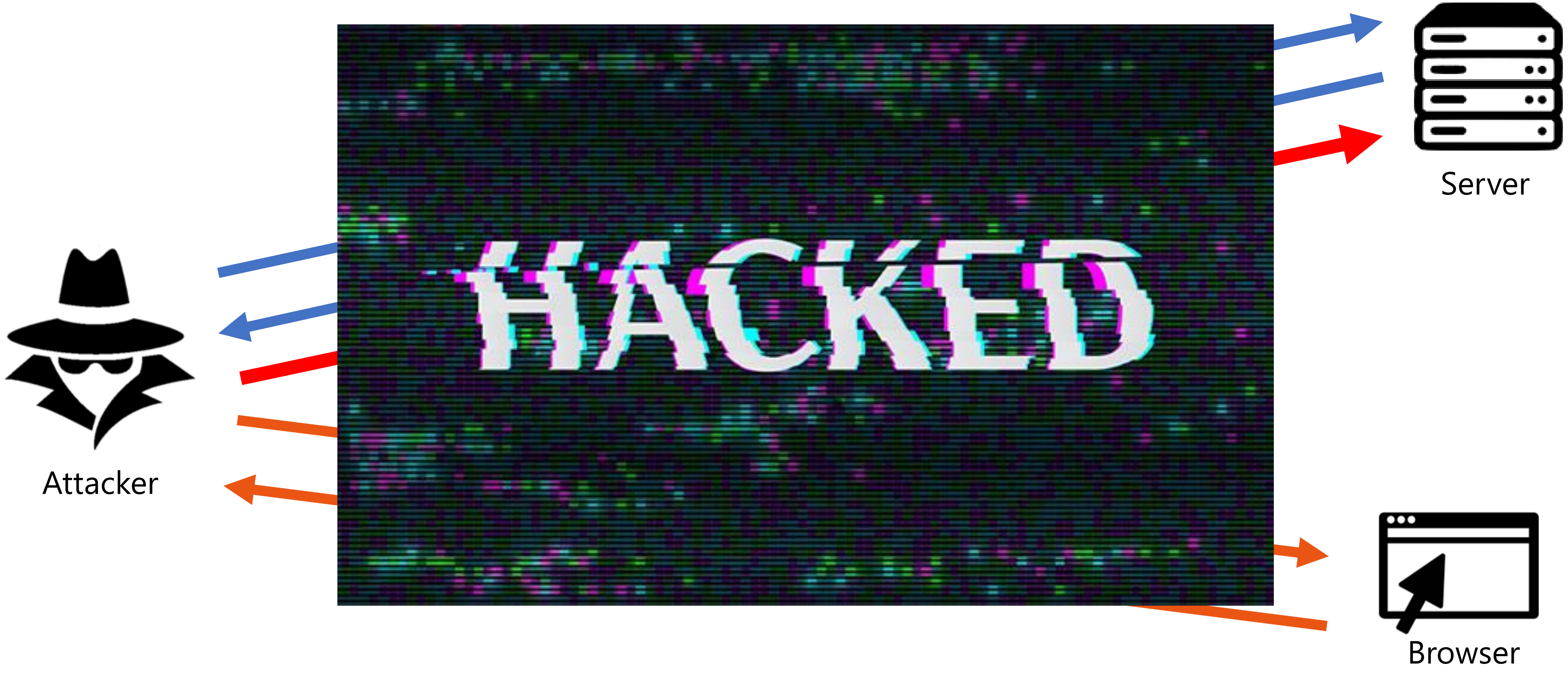
Check browser compatibility



	🖥						📱					
	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	WebView Android	Chrome Android	Firefox Android	Opera Android	iOS Safari	Samsung Internet
`Set-Cookie`	Yes	12	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
`HttpOnly`	1	12	3	9	11	5	37	Yes	4	Yes	4	Yes

OWASP Top 10 Web Application Security

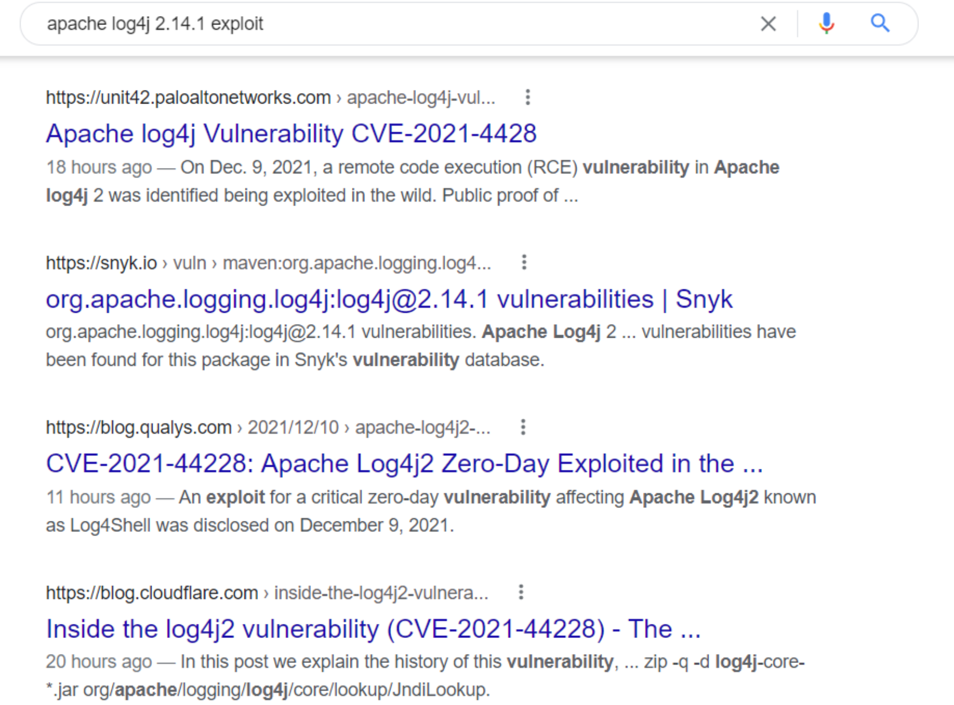
Examples – 6 Vulnerable and Outdated Components



OWASP Top 10 Web Application Security

Examples – 6 Vulnerable and Outdated Components

CVE = Common Vulnerabilities and Exposures
A system provides a reference-method for publicly known information-security vulnerabilities and exposures.



apache log4j 2.14.1 exploit

https://unit42.paloaltonetworks.com › apache-log4j-vul...
Apache log4j Vulnerability CVE-2021-4428
18 hours ago — On Dec. 9, 2021, a remote code execution (RCE) **vulnerability** in **Apache log4j** 2 was identified being exploited in the wild. Public proof of ...

https://snyk.io › vuln › maven:org.apache.logging.log4...
org.apache.logging.log4j:log4j@2.14.1 vulnerabilities | Snyk
org.apache.logging.log4j:log4j@2.14.1 vulnerabilities. **Apache Log4j** 2 ... vulnerabilities have been found for this package in Snyk's **vulnerability** database.

https://blog.qualys.com › 2021/12/10 › apache-log4j2-...
CVE-2021-44228: Apache Log4j2 Zero-Day Exploited in the ...
11 hours ago — An **exploit** for a critical zero-day **vulnerability** affecting **Apache Log4j2** known as Log4Shell was disclosed on December 9, 2021.

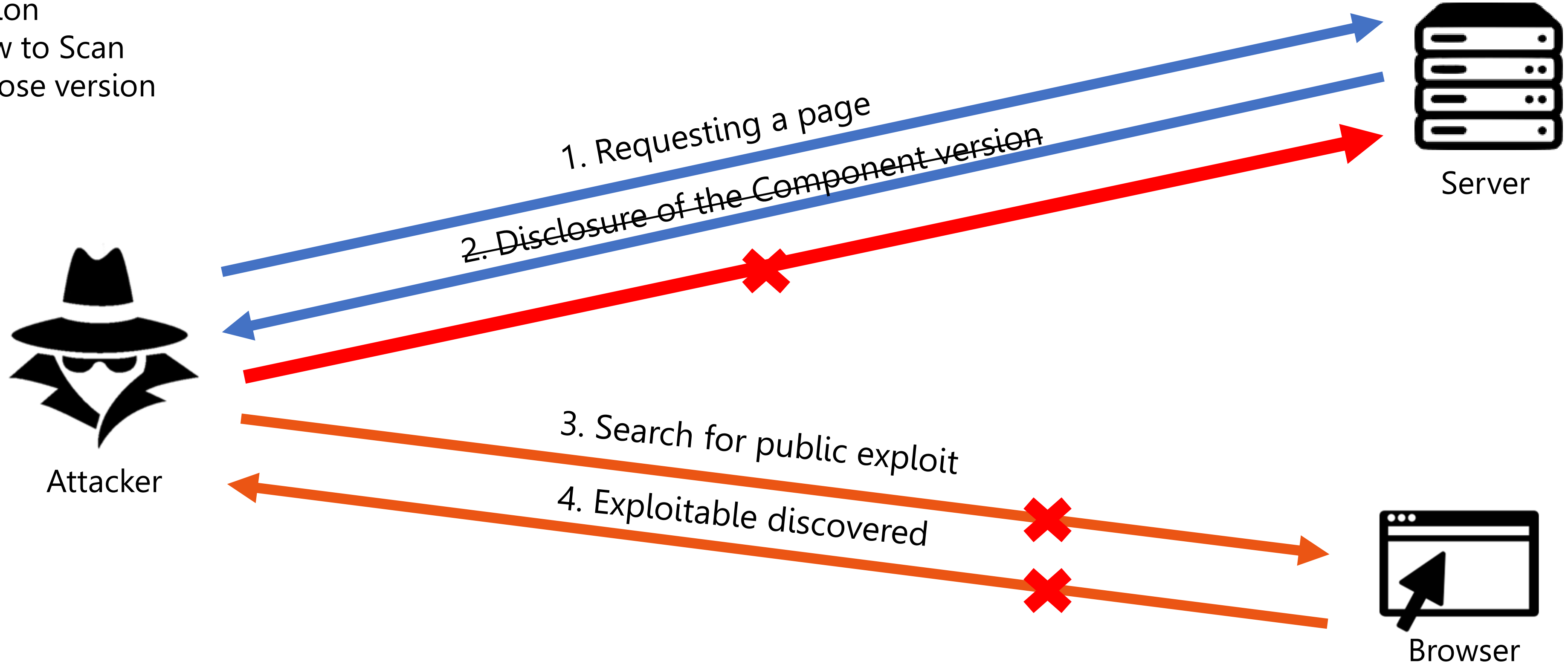
https://blog.cloudflare.com › inside-the-log4j2-vulnera...
Inside the log4j2 vulnerability (CVE-2021-44228) - The ...
20 hours ago — In this post we explain the history of this **vulnerability**, ... zip -q -d **log4j**-core-*.jar org/**apache**/logging/**log4j**/core/lookup/JndiLookup.

OWASP Top 10 Web Application Security

Examples – 6 Vulnerable and Outdated Components

BAD Mitigation

- Not Allow to Scan
- Not disclose version

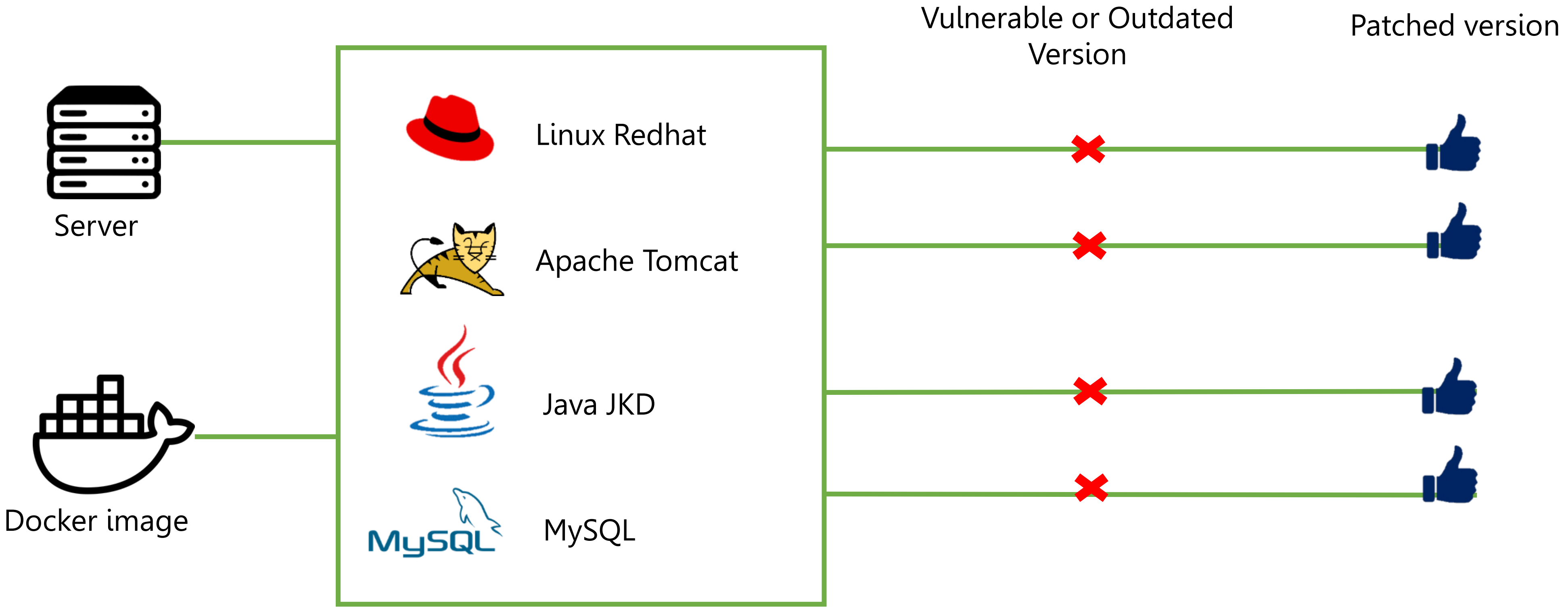

1. Requesting a page
~~2. Disclosure of the Component version~~
Server
Attacker
3. Search for public exploit
4. Exploitable discovered
Browser

OWASP Top 10 Web Application Security

Examples – 6 Vulnerable and Outdated Components

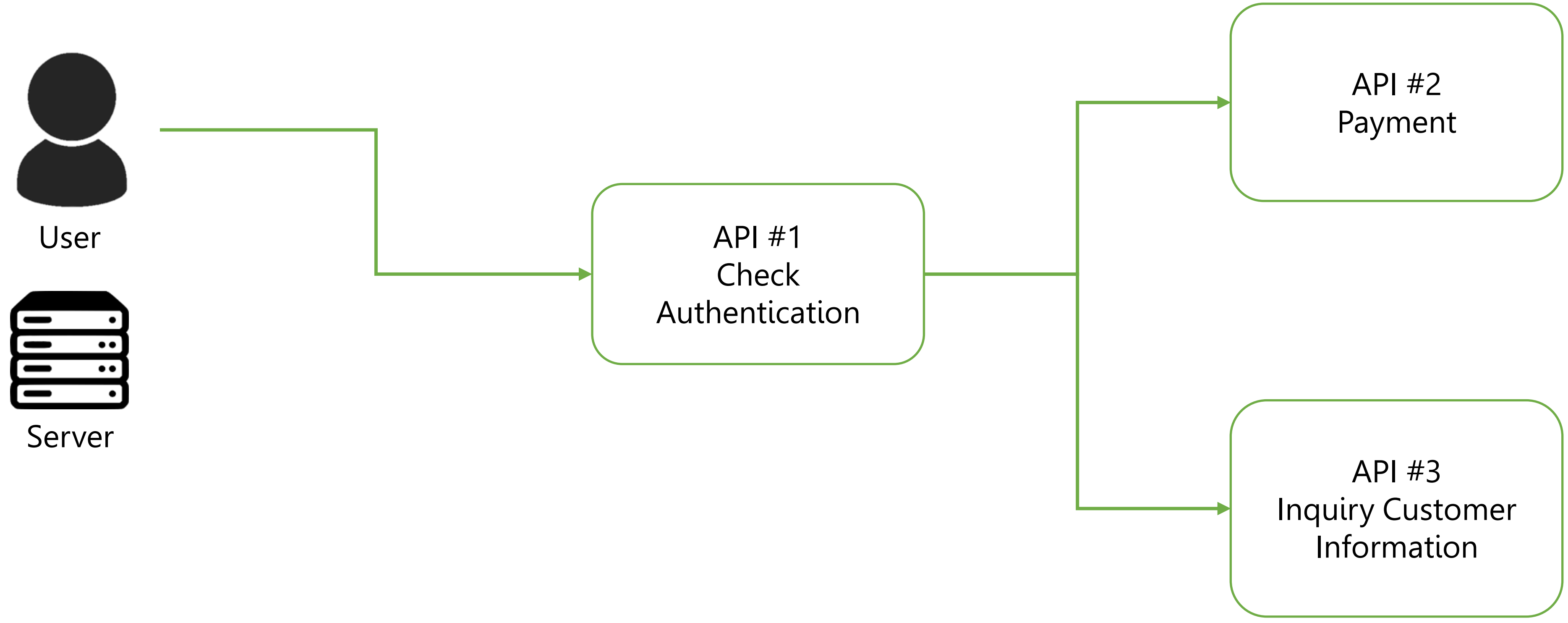
Good Remediation

- Check if there is any outdated component
- Patch !



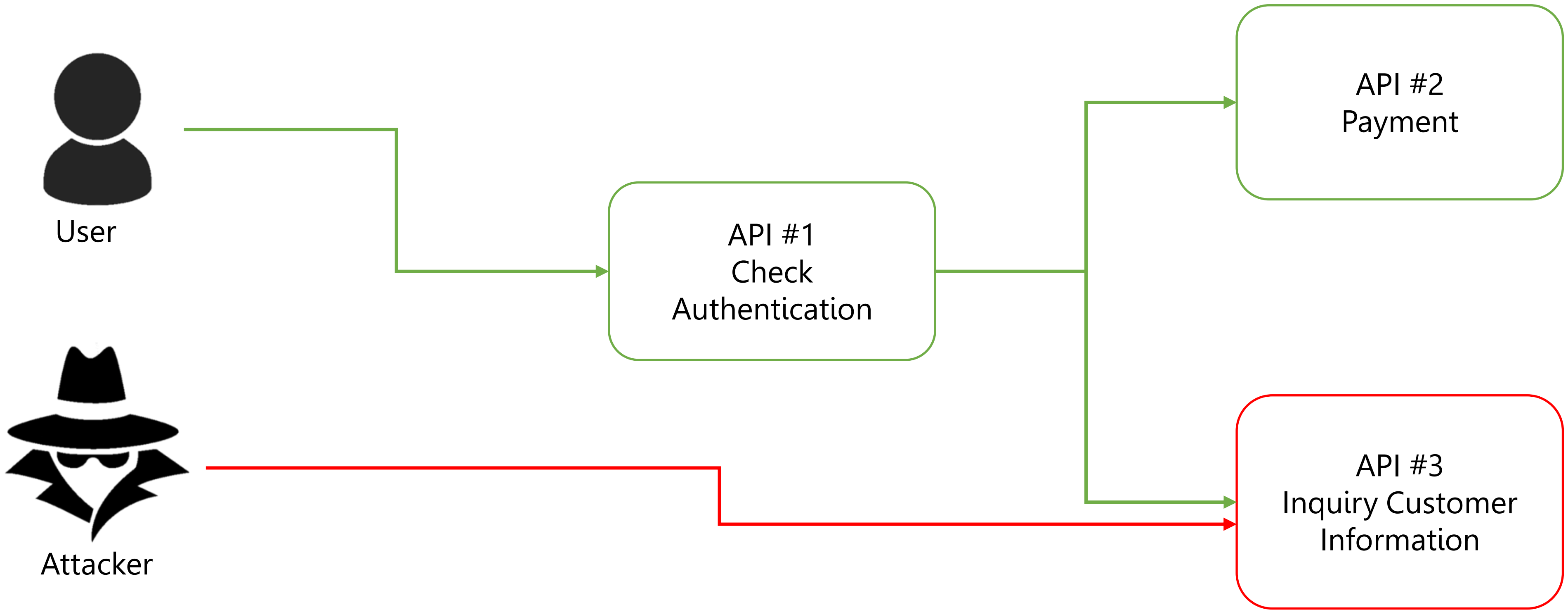
OWASP Top 10 Web Application Security

Examples – 7 Identification and Authentication Failures


User
Server
API #1
Check
Authentication
API #2
Payment
API #3
Inquiry Customer
Information

OWASP Top 10 Web Application Security

Examples – 7 Identification and Authentication Failures

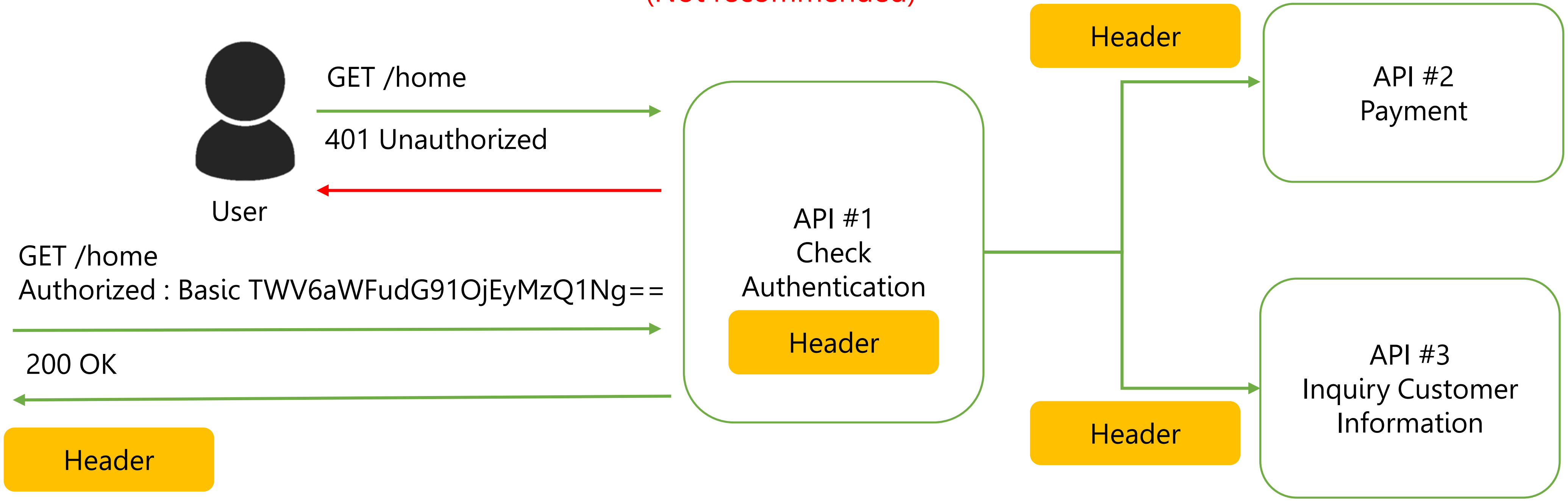


OWASP Top 10 Web Application Security

Examples – 7 Identification and Authentication Failures

Basic Authentication with username/password

(Not recommended)

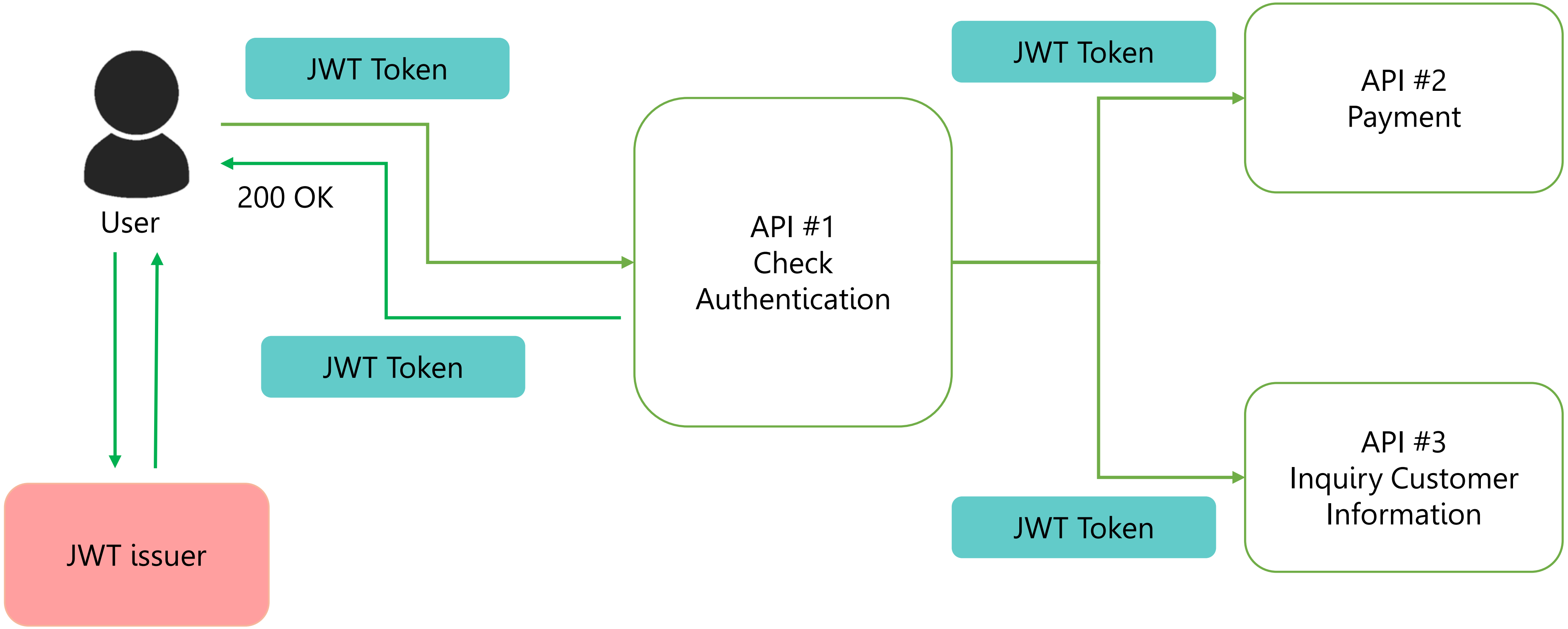


TWV6aWFudG91OjEyMzQ1Ng== was Base64(Username:Password)

OWASP Top 10 Web Application Security

Examples – 7 Identification and Authentication Failures

Json Web Token (JWT)



Key take away 3

What do we gain from understanding the various OWASP Top Ten web application?

4.2 : OWASP API Security

1 : Overview

2 : Application Security Protection

3 : Threat Modeling

4 : OWASP

4.1 : OWASP - Web security

4.2 : OWASP - API Security

4.3 : OWASP - Mobile security

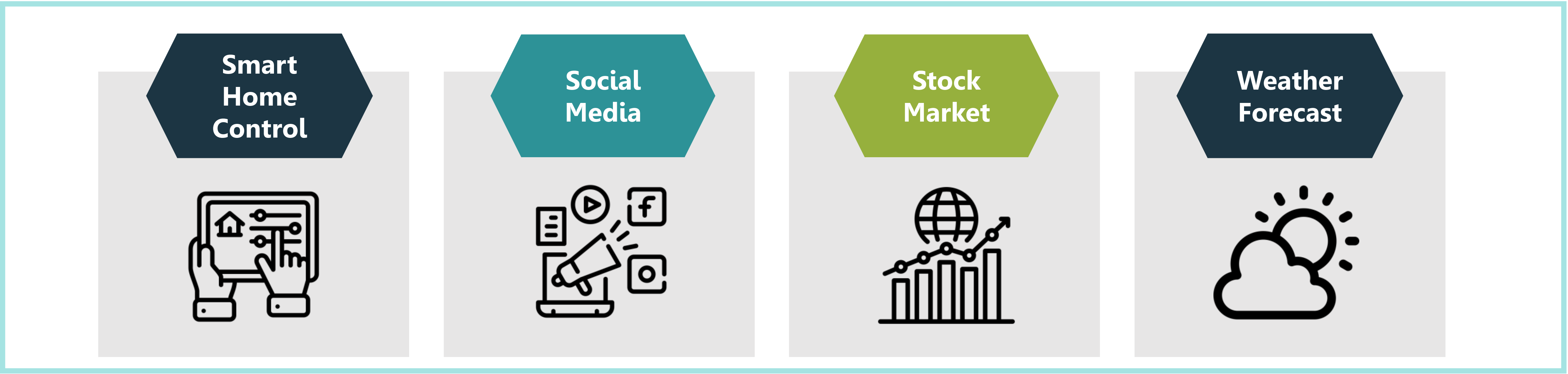
4.4 : OWASP - Proactive Control

OWASP Top 10 API Security

Understanding APIs

“An Application Programming Interface (API) is an interface or communication protocol between a client and a server intended to simplify the building of client-side software. it has been described as a “contract” between the client and the server, such that if the client makes a request in a specific format, it will always get a response in a specific format or initiate a defined action.”

APIs in Daily Life



Smart Home Control

Social Media

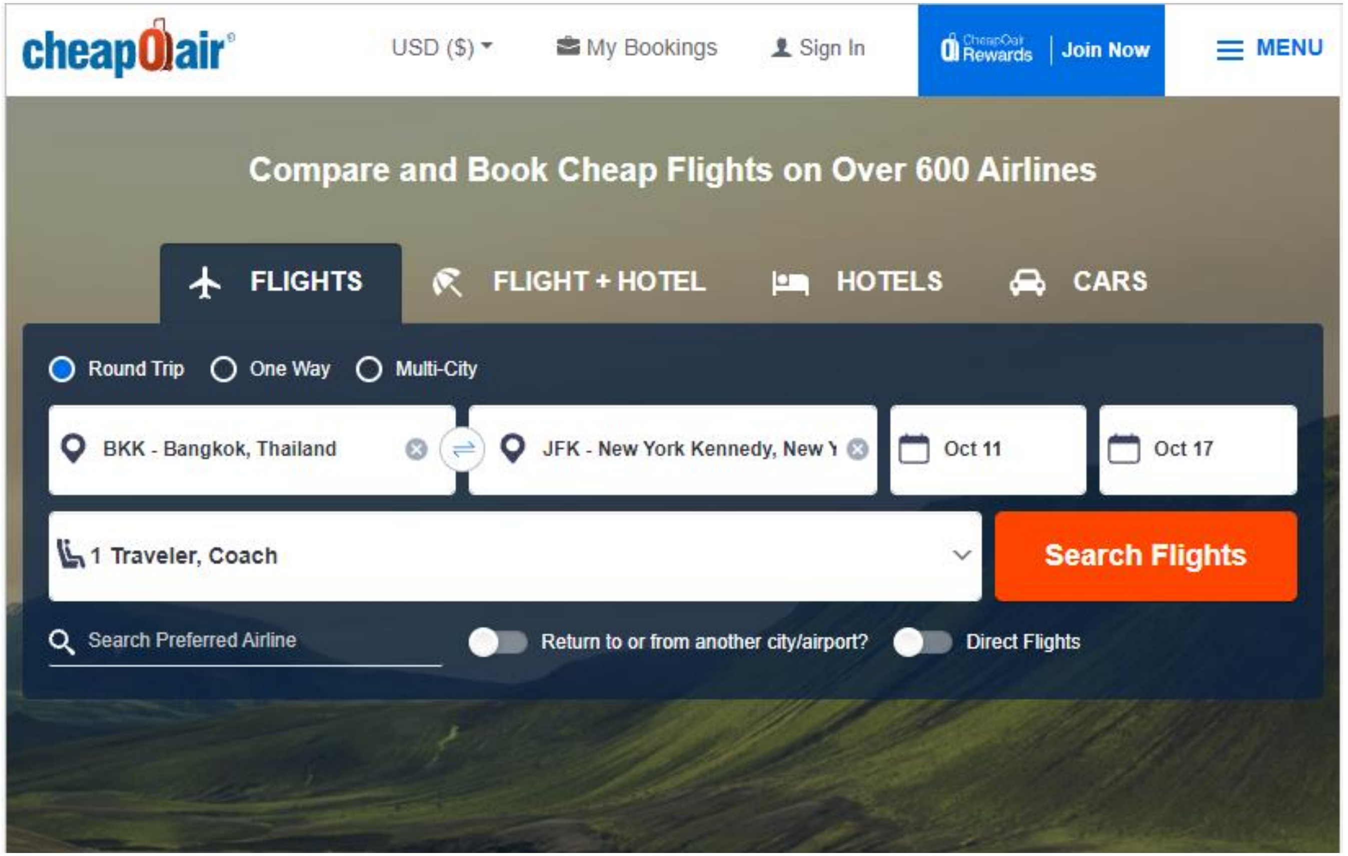
Stock Market

Weather Forecast

Ref: OWASP API Security Project

OWASP Top 10 API Security

APIs in Daily Life – User Interface



Compare and Book Cheap Flights on Over 600 Airlines

○ Round Trip (selected) ○ One Way ○ Multi-City

BKK - Bangkok, Thailand

JFK - New York Kennedy, New Y

Oct 11

Oct 17

1 Traveler, Coach

Search Preferred Airline

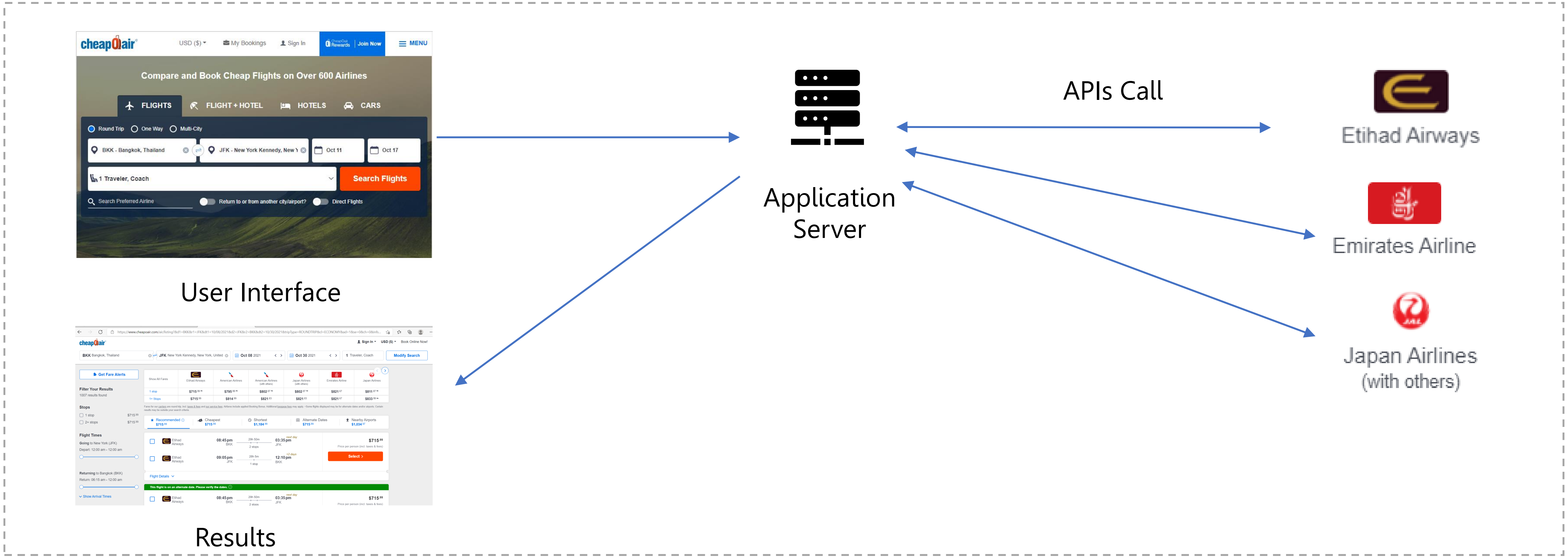
Return to or from another city/airport?

Direct Flights

OWASP Top 10 API Security

APIs in Daily Life – APIs Call

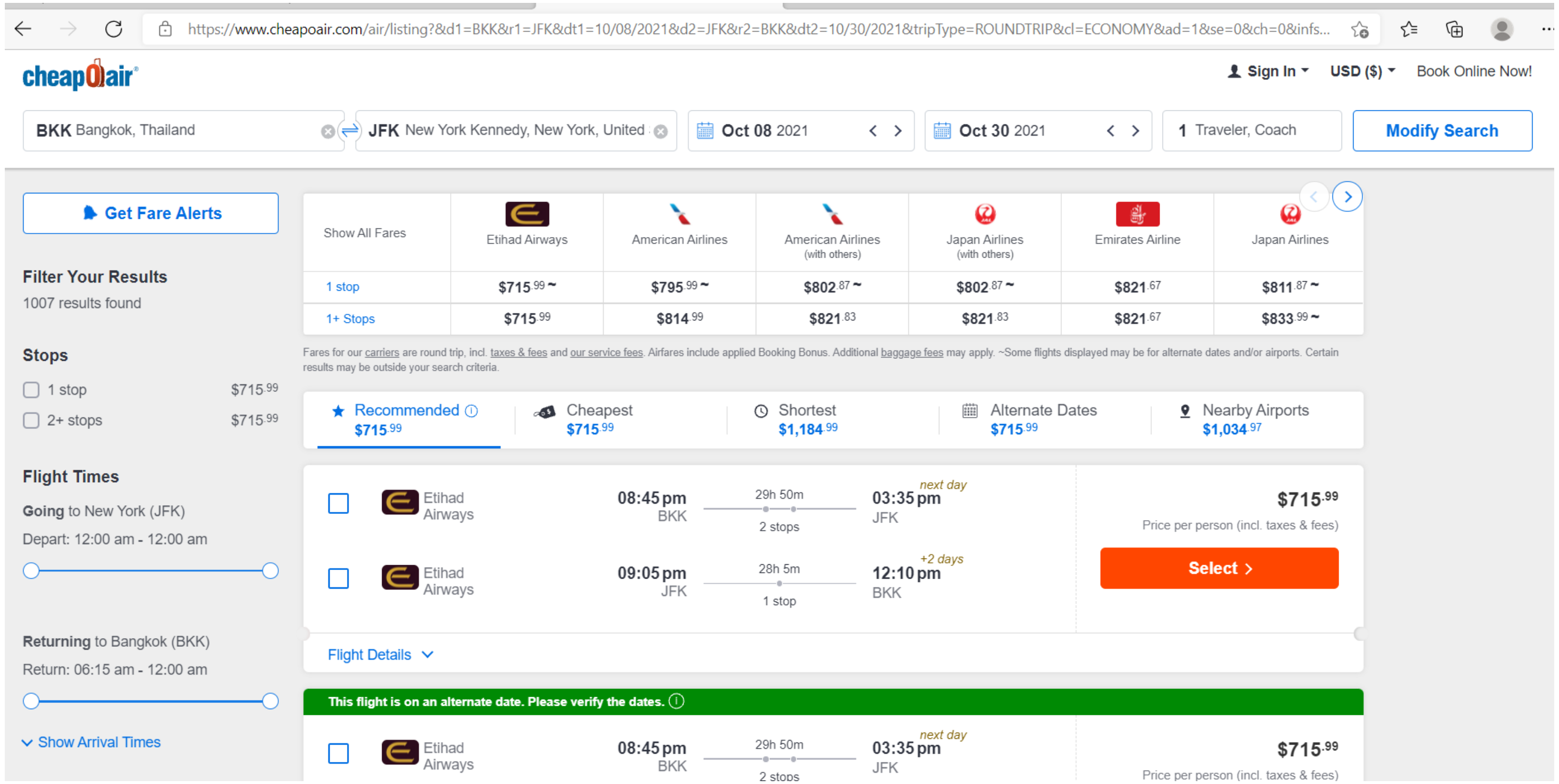
API Call is the request message to another service via Application Interface



4.2

OWASP Top 10 API Security

APIs in Daily Life – Results from APIs call



https://www.cheapoair.com/air/listing?&d1=BKK&r1=JFK&dt1=10/08/2021&d2=JFK&r2=BKK&dt2=10/30/2021&tripType=ROUNDTRIP&cl=ECONOMY&ad=1&se=0&ch=0&infs...

cheapOair

Sign In | USD ($) | Book Online Now!

BKK Bangkok, Thailand | JFK New York Kennedy, New York, United | Oct 08 2021 | Oct 30 2021 | 1 Traveler, Coach | Modify Search

Get Fare Alerts

Filter Your Results

1007 results found

Stops

□ 1 stop	$715 99
□ 2+ stops	$715 99

Flight Times

Going to New York (JFK)

Depart: 12:00 am - 12:00 am

Returning to Bangkok (BKK)

Return: 06:15 am - 12:00 am

Show Arrival Times

Show All Fares	Etihad Airways	American Airlines	American Airlines (with others)	Japan Airlines (with others)	Emirates Airline	Japan Airlines
1 stop	$715 99 ~	$795 99 ~	$802 87 ~	$802 87 ~	$821 67	$811 87 ~
1+ Stops	$715 99	$814 99	$821 83	$821 83	$821 67	$833 99 ~

Fares for our carriers are round trip, incl. taxes & fees and our service fees. Airfares include applied Booking Bonus. Additional baggage fees may apply. ~Some flights displayed may be for alternate dates and/or airports. Certain results may be outside your search criteria.

Recommended	Cheapest	Shortest	Alternate Dates	Nearby Airports
$715 99	$715 99	$1,184 99	$715 99	$1,034 97

□	Etihad Airways	08:45 pm BKK	29h 50m 2 stops	next day 03:35 pm JFK	$715 99 Price per person (incl. taxes & fees)
□	Etihad Airways	09:05 pm JFK	28h 5m 1 stop	+2 days 12:10 pm BKK	Select >

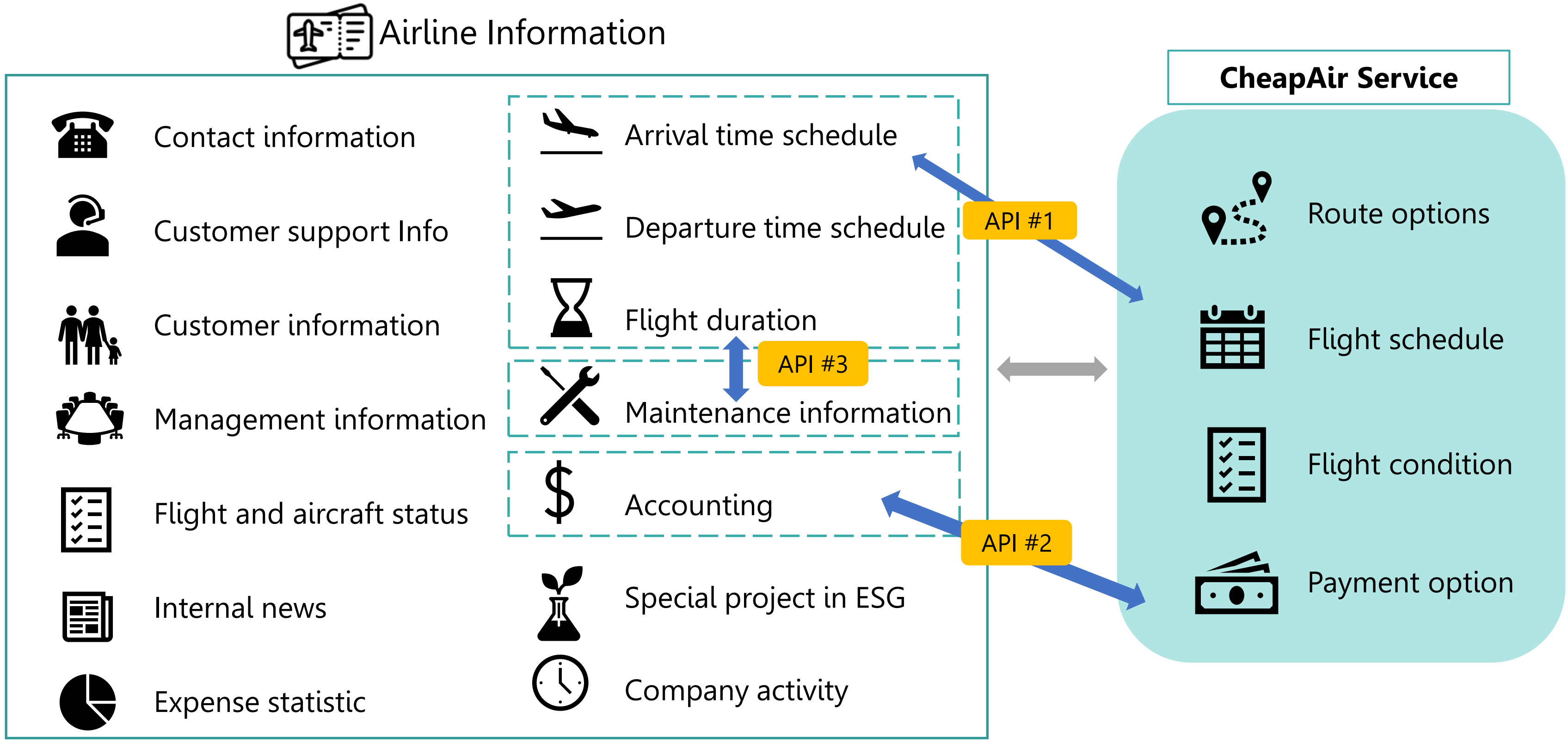
Flight Details

This flight is on an alternate date. Please verify the dates.

□	Etihad Airways	08:45 pm BKK	29h 50m 2 stops	next day 03:35 pm JFK	$715 99 Price per person (incl. taxes & fees)

OWASP Top 10 API Security

APIs in Daily Life – Required Information and Overall Information



API Security Foundations

What is API Security ?

API security involves protecting the integrity of APIs, ensuring that they are not vulnerable to cyber threats or unauthorized access, and maintaining data privacy and confidentiality.

Important of API Security

API security is crucial for safeguarding sensitive data, preventing unauthorized access, and maintaining the trust of users and stakeholders. Without proper security measures, API are vulnerable to data breaches and cyber attacks.

Common Threats

APIs are susceptible to a range of threats, including injection attacks, broken authentication, excessive data exposure, and insufficient logging and monitoring. Understanding these threats is essential for effective API security.

OWASP Top 10 API Security 2023

API 01
Broken Object Level Authorization

API 02
Broken Authentication

API 03
Broken Object Property Level Authorization

API 04
Unrestricted Resource Consumption

API 05
Broken Function Level Authorization

API 06
Unrestricted Access to Sensitive Business Flow

API 07
Server-Side Request Forgery

API 08
Security Misconfiguration

API 09
Improper Inventory Management

API 10
Unsafe Consumption of APIs

OWASP Top 10 API Application Security

Authentication and Authorization

API1:2023 Broken Object Level Authorization

- APIs tend to expose endpoints that handle object identifiers, creating a wide attack surface of Object Level Access Control issues

API2:2023 Broken Authentication

- Authentication mechanisms are often implemented incorrectly, allowing attackers to compromise authentication tokens or to exploit implementation flaws to assume other user's identities temporarily or permanently

API3:2023 Broken Object Property Level Authorization

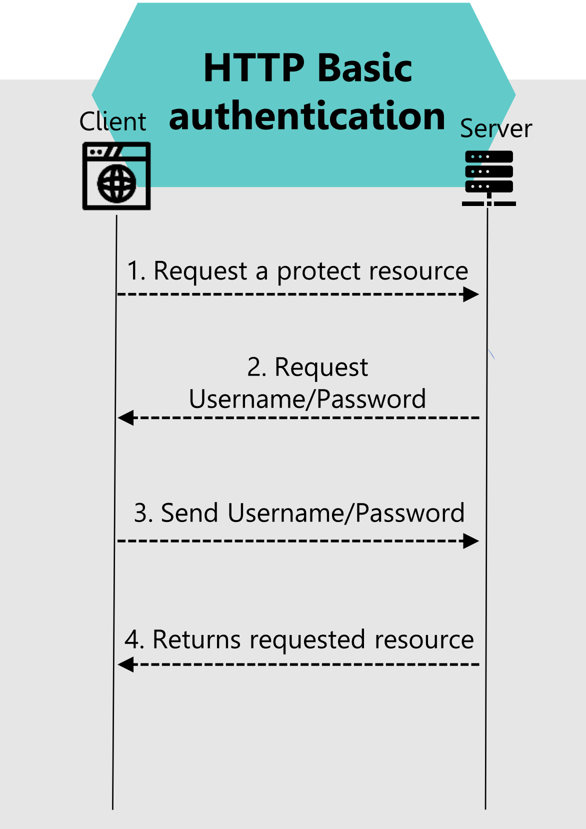
- Lack of or improper authorization validation at the object property level. This leads to information exposure or manipulation by unauthorized parties

API5:2023 Broken Function Level Authorization

- Complex access control policies with different hierarchies, groups, and roles, and an unclear separation between administrative and regular functions, tend to lead to authorization flaws

OWASP Top 10 API Application Security

Authentication and Authorization

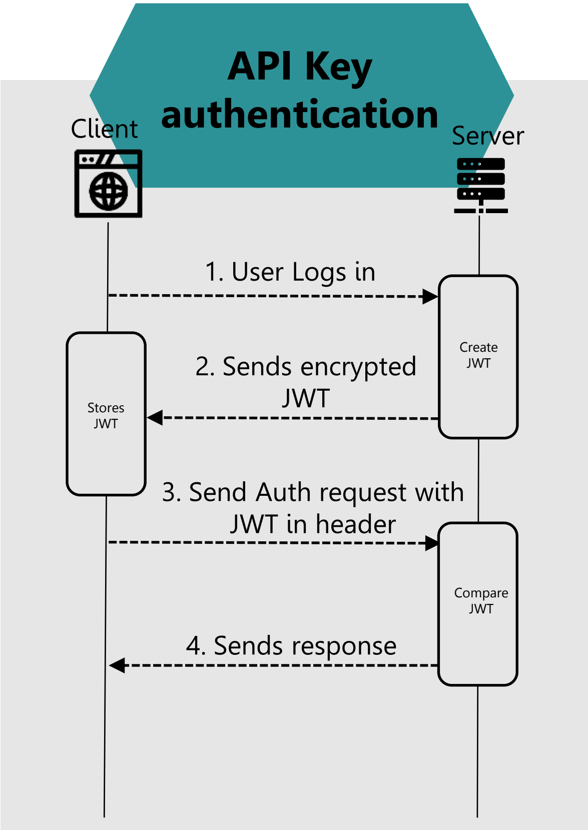


HTTP Basic authentication

Client

Server

1. Request a protect resource
2. Request Username/Password
3. Send Username/Password
4. Returns requested resource



API Key authentication

Client

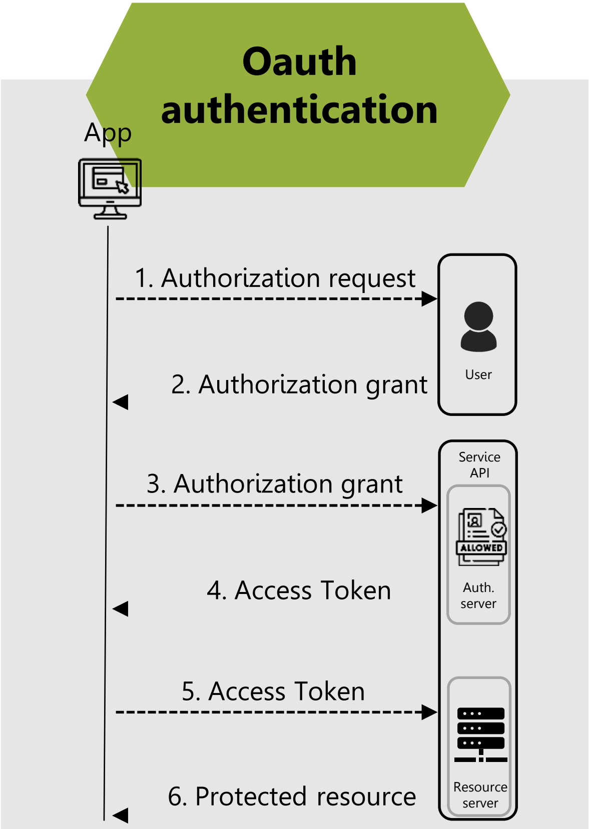
Server

1. User Logs in
2. Sends encrypted JWT
3. Send Auth request with JWT in header
4. Sends response

Create JWT

Stores JWT

Compare JWT



Oauth authentication

App

1. Authorization request
2. Authorization grant
3. Authorization grant
4. Access Token
5. Access Token
6. Protected resource

User

Service API

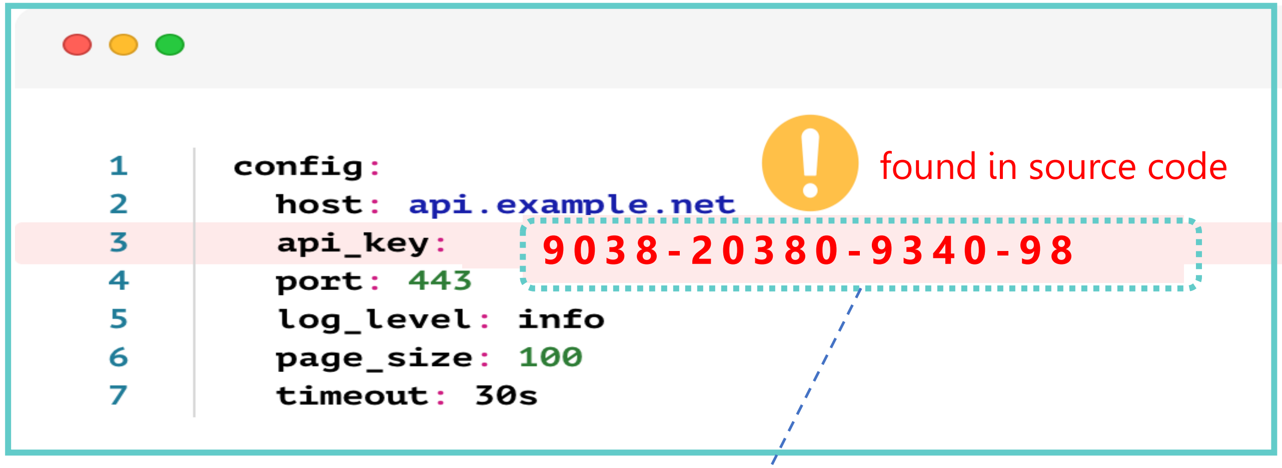
ALLOWED

Auth. server

Resource server

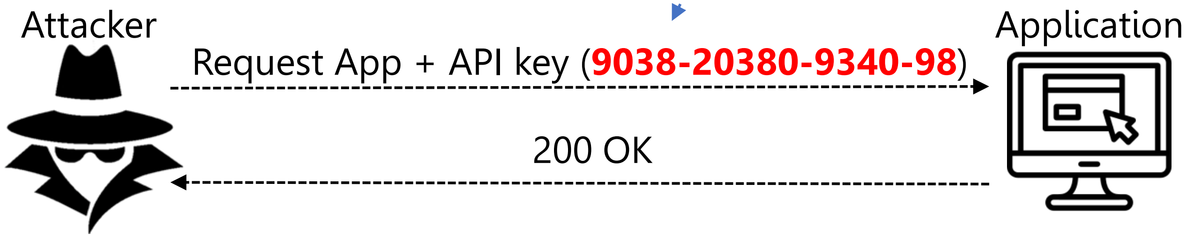
OWASP Top 10 API Application Security

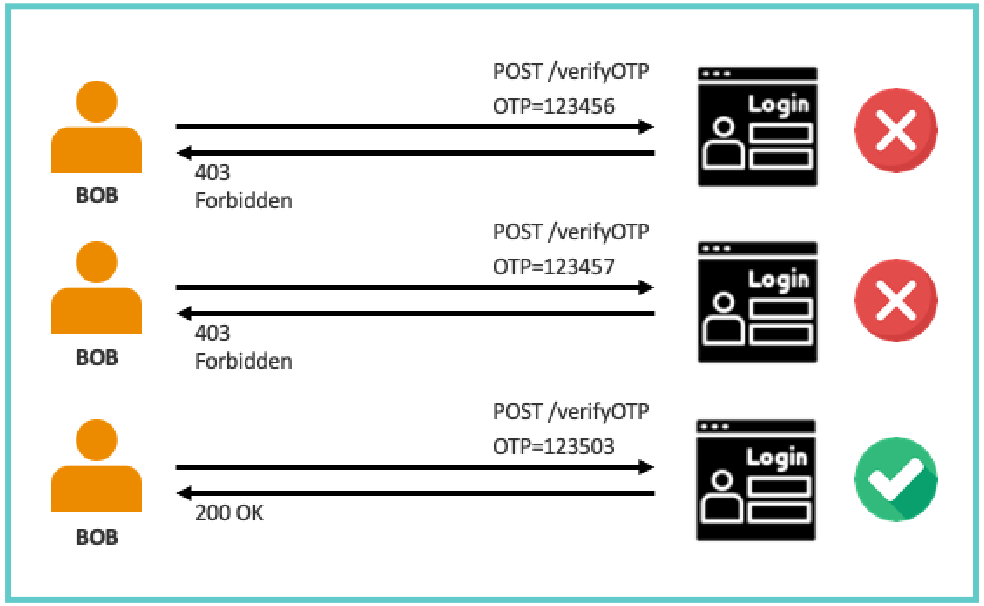
Example Broken Authentication



```
config:
  host: api.example.net
  api_key: 9038-20380-9340-98
  port: 443
  log_level: info
  page_size: 100
  timeout: 30s
```

found in source code

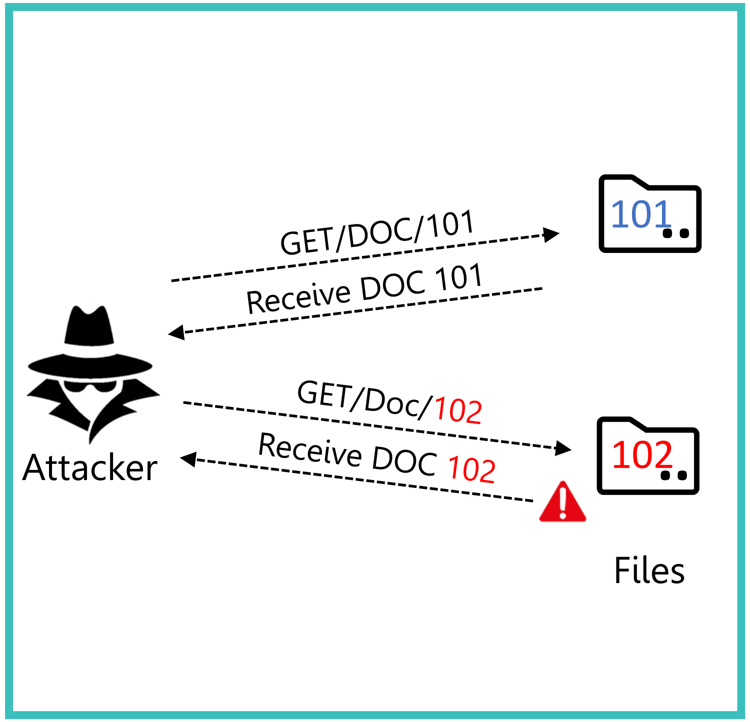




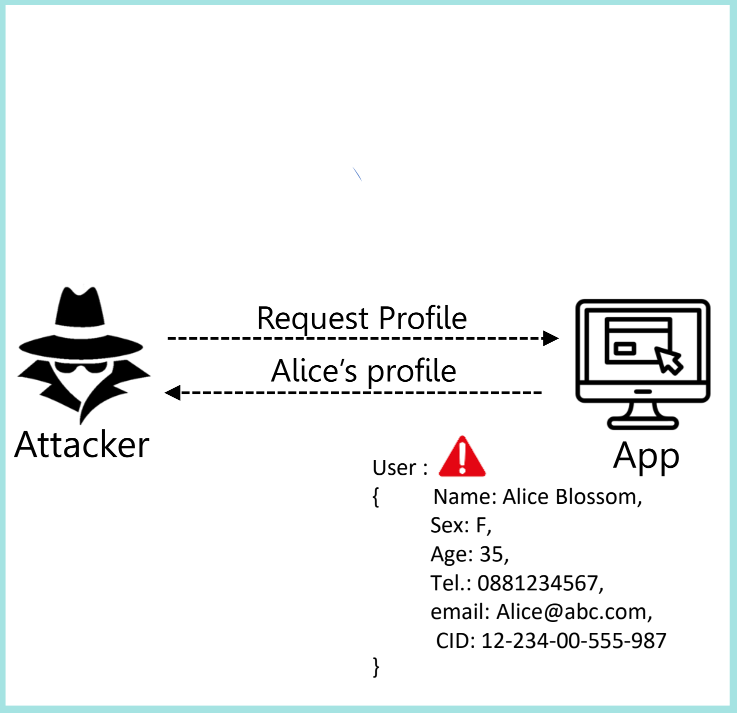
API2:2023 Broken Authentication

OWASP Top 10 API Application Security

Example Broken Authorization

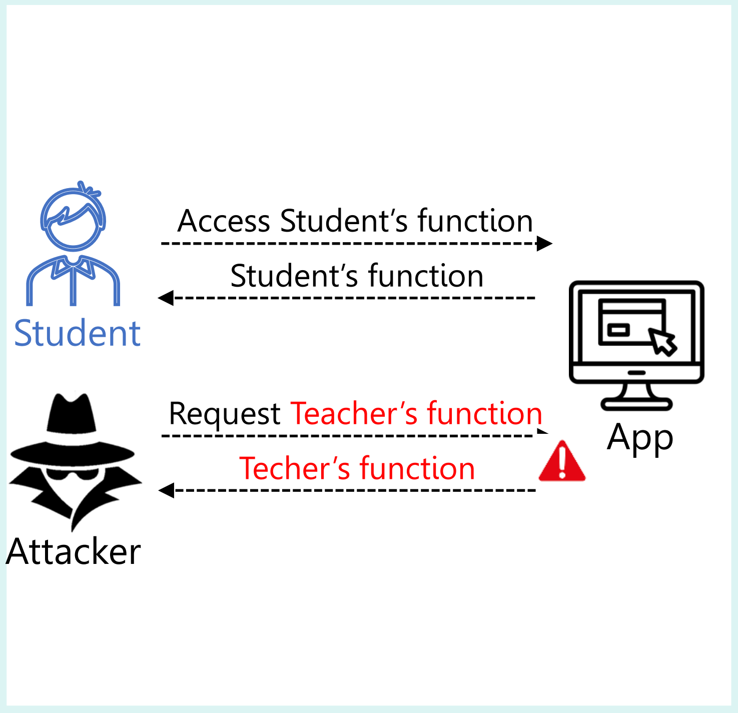


API1:2023 Broken Object Level Authorization



```
User :
{       Name: Alice Blossom,
        Sex: F,
        Age: 35,
        Tel.: 0881234567,
        email: Alice@abc.com,
         CID: 12-234-00-555-987
}
```

API3:2023 Broken Object Property Level Authorization

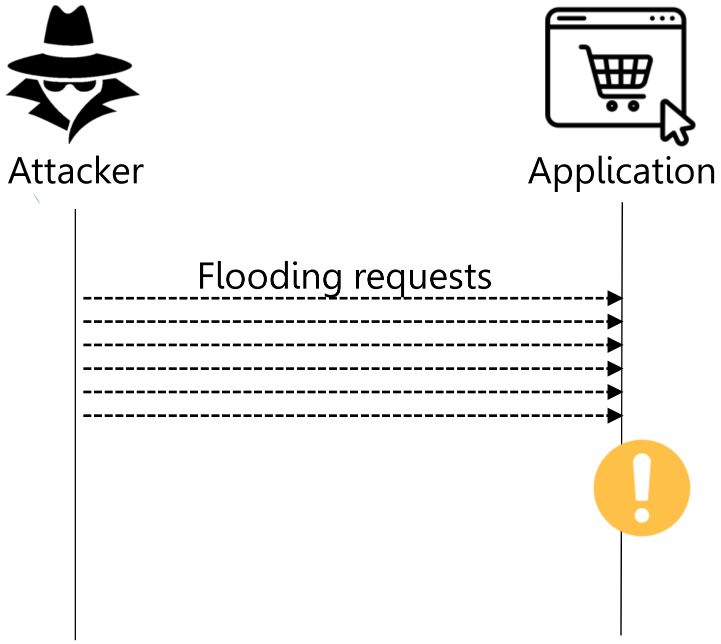


API5:2023 Broken Function Level Authorization

OWASP Top 10 API Application Security

API4. Unrestricted Resource Consumption

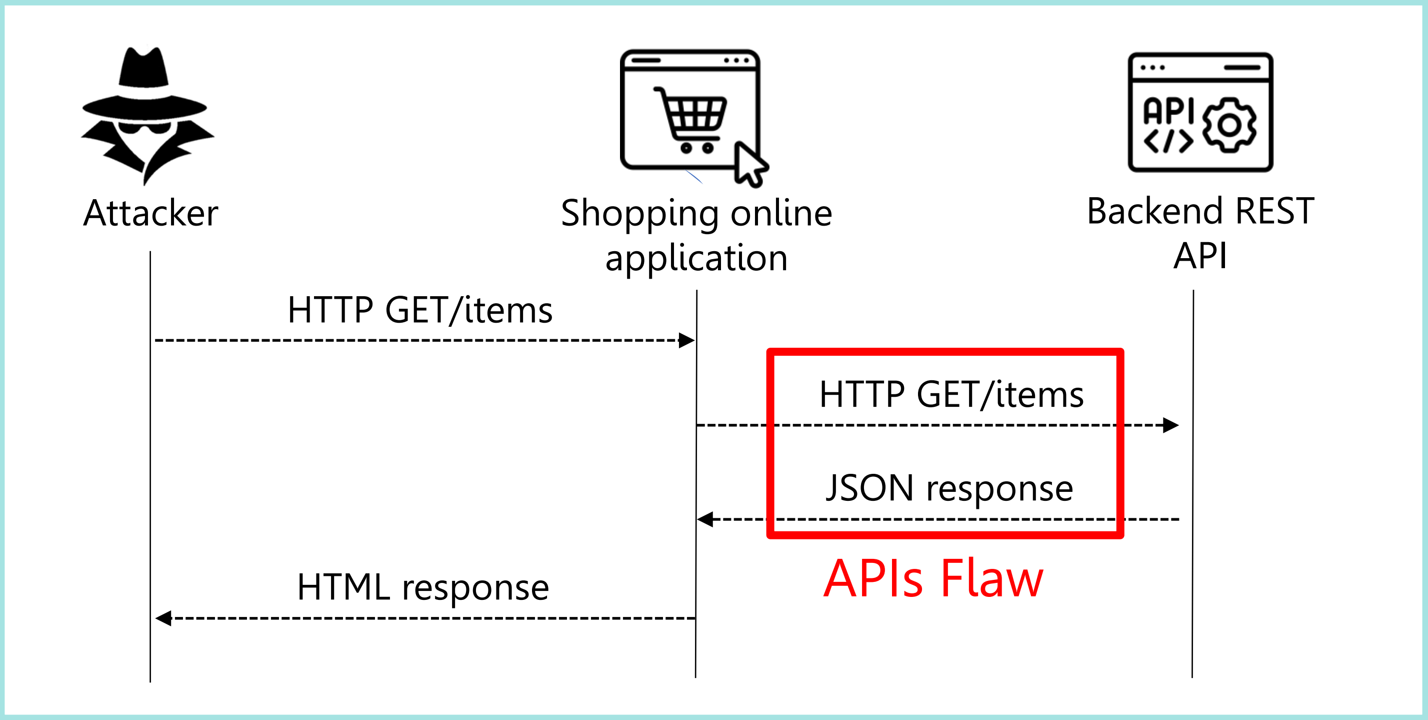
Satisfying API requests requires resources such as network bandwidth, CPU, memory, and storage. Other resources such as emails/SMS/phone calls or biometrics validation are made available by service providers via API integrations and paid for per request. Successful attacks can lead to Denial of Service or an increase of operational costs.



OWASP Top 10 API Application Security

API6. Unrestricted Access to Sensitive Business Flows

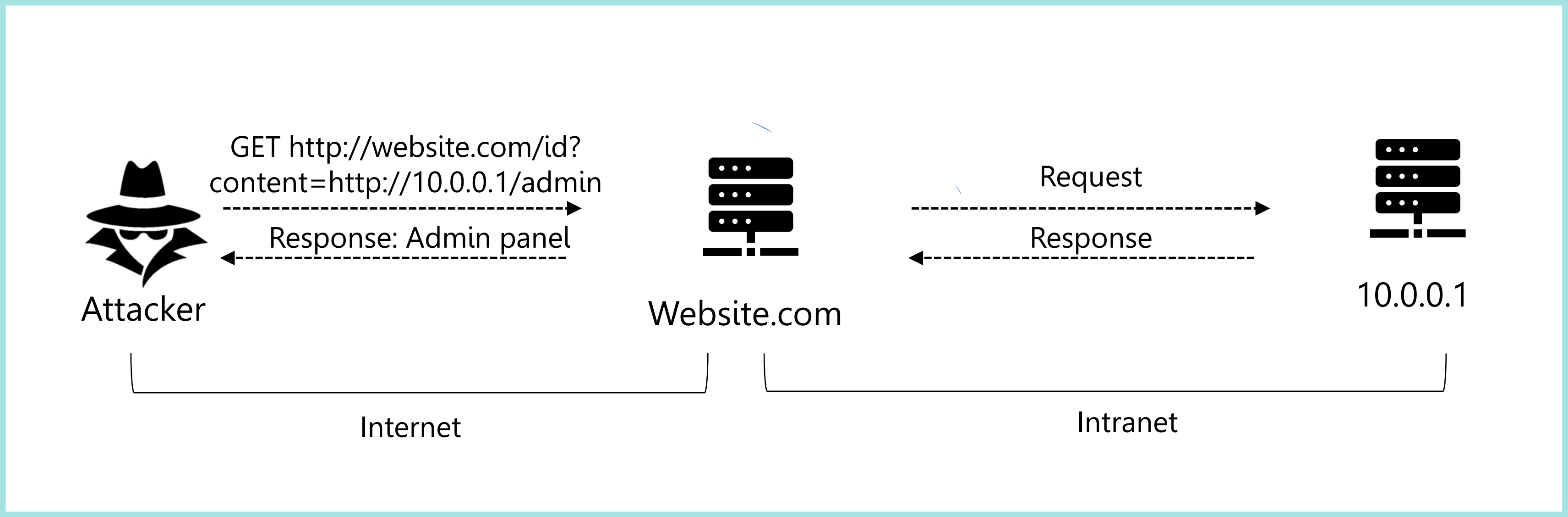
APIs vulnerable to this risk expose a business flow - such as buying a ticket, or posting a comment - without compensating for how the functionality could harm the business if used excessively in an automated manner. This doesn’t necessarily come from implementation bugs.



OWASP Top 10 API Application Security

API7. Server Side Request Forgery

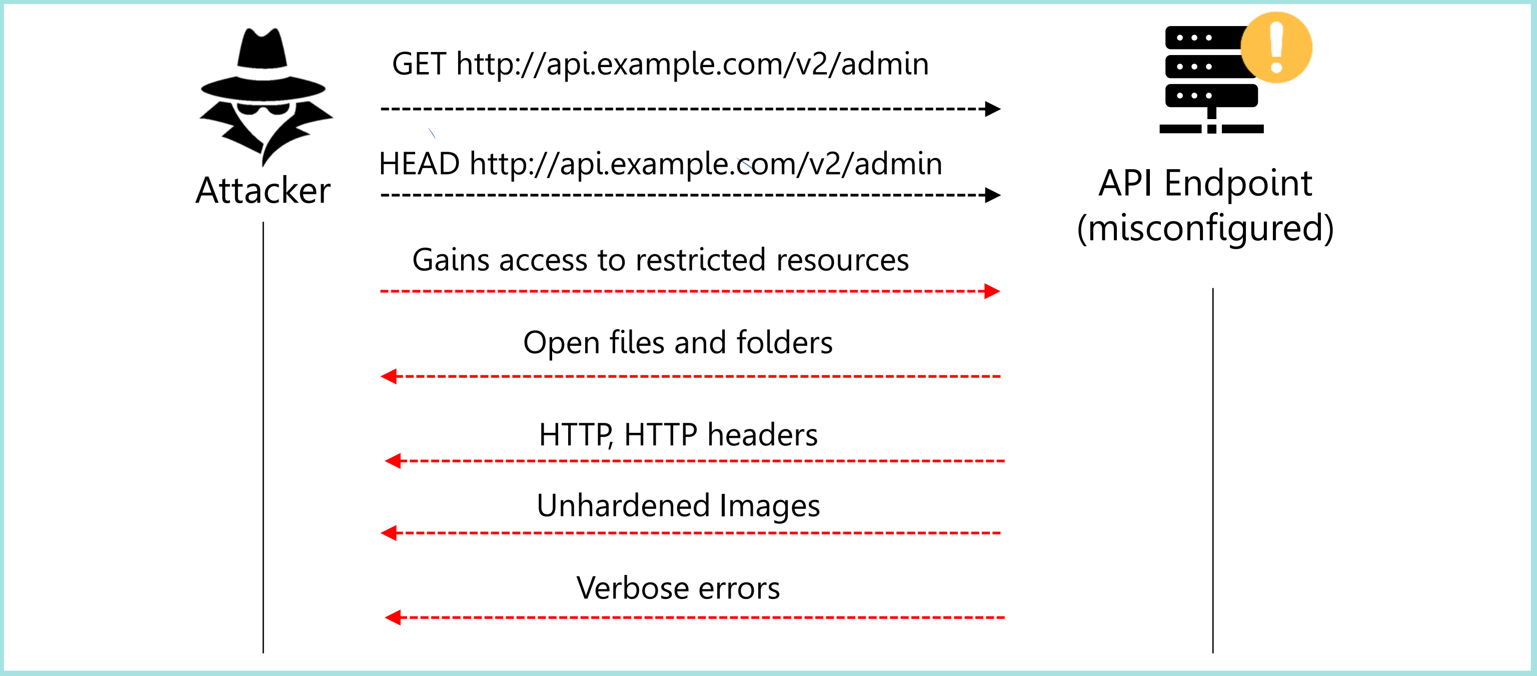
Server-Side Request Forgery (SSRF) flaws can occur when an API is fetching a remote resource without validating the user-supplied URI. This enables an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall or a VPN.



OWASP Top 10 API Application Security

API8. Security Misconfiguration

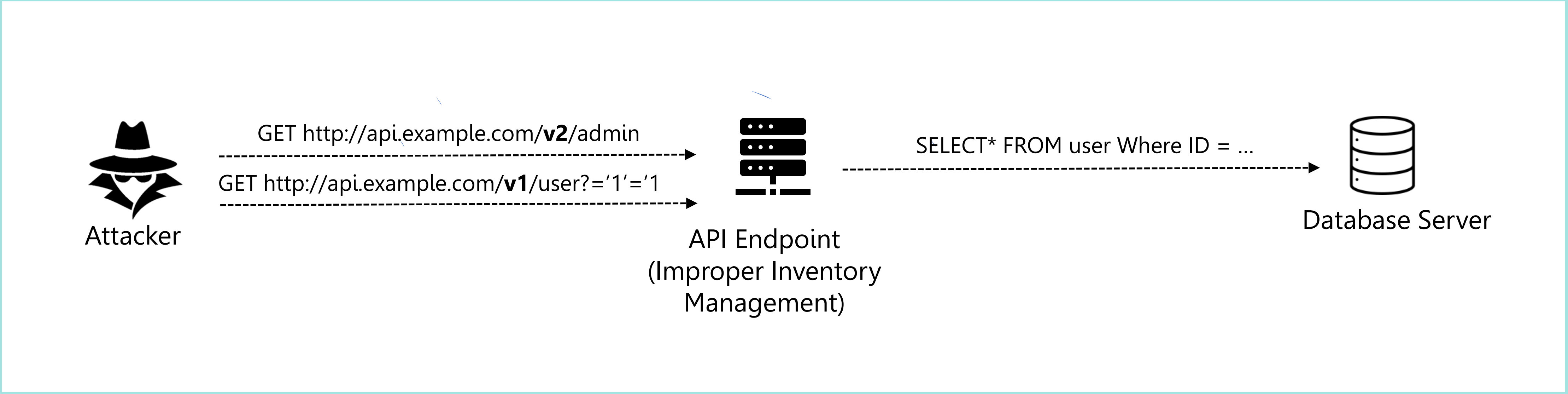
APIs and the systems supporting them typically contain complex configurations, meant to make the APIs more customizable. Software and DevOps engineers can miss these configurations, or don't follow security best practices when it comes to configuration, opening the door for different types of attacks


Attacker
API Endpoint (misconfigured)
GET http://api.example.com/v2/admin
HEAD http://api.example.com/v2/admin
Gains access to restricted resources
Open files and folders
HTTP, HTTP headers
Unhardened Images
Verbose errors

OWASP Top 10 API Application Security

API9. Improper Inventory Management

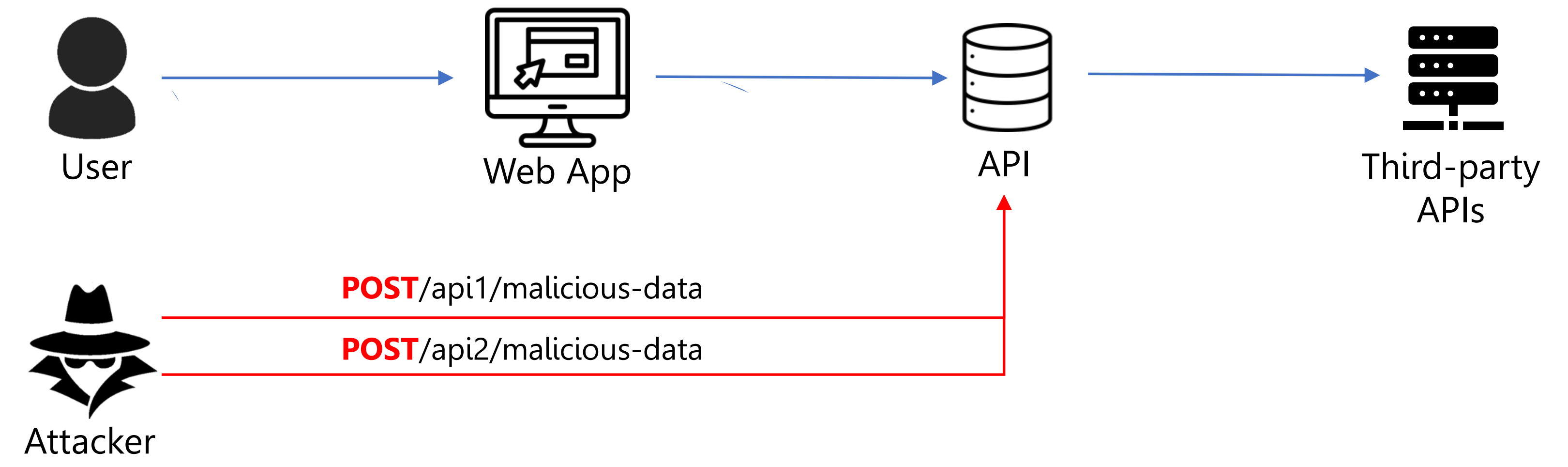
APIs tend to expose more endpoints than traditional web applications, making proper and updated documentation highly important. A proper inventory of hosts and deployed API versions also are important to mitigate issues such as deprecated API versions and exposed debug endpoints.


GET http://api.example.com/**v2**/admin
GET http://api.example.com/**v1**/user?=‘1’=‘1
Attacker
API Endpoint
(Improper Inventory
Management)
SELECT* FROM user Where ID = ...
Database Server

OWASP Top 10 API Application Security

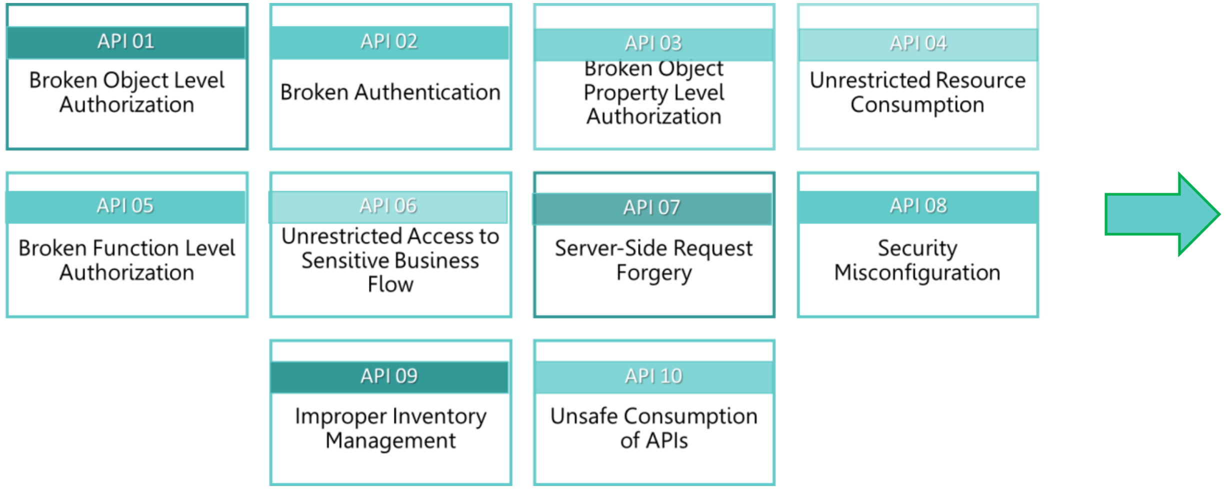
API10. Unsafe Consumption of APIs

Developers tend to trust data received from third-party APIs more than user input, and so tend to adopt weaker security standards. In order to compromise APIs, attackers go after integrated third-party services instead of trying to compromise the target API directly.



API Security Keys Takeaway

API Security Keys Takeaway



API 01
Broken Object Level Authorization

API 02
Broken Authentication

API 03
Broken Object Property Level Authorization

API 04
Unrestricted Resource Consumption

API 05
Broken Function Level Authorization

API 06
Unrestricted Access to Sensitive Business Flow

API 07
Server-Side Request Forgery

API 08
Security Misconfiguration

API 09
Improper Inventory Management

API 10
Unsafe Consumption of APIs

To reduce the risk of cybersecurity attacks, You should be aware of the following five main points:

- Authentication and Authorization
- Input Validation
- Data Protection
- API Rate Limiting
- Error Handling and Logging

4.3 : OWASP Mobile security

1 : Overview

2 : Application Security Protection

3 : Threat Modeling

4 : OWASP

4.1 : OWASP - Web security

4.2 : OWASP - API Security

4.3 : OWASP - Mobile security

4.4 : OWASP - Proactive Control

OWASP Top 10 Mobile Application Security

Number of mobile apps analyzed by industry in 2023

	Technology ★	Financial service ★	Retail	Healthcare ⚠	Total
IOS apps	857	55	220	332	1,464
Android apps	355	69	252	277	953
Total mobile apps	1,212	124	472	609	2,417

Source: Coalfire 5th Annual Penetration Risk Report

4.3

OWASP Top 10 Mobile Application Security

Privacy and security issues found in mobile app by industry in 2023

	Technology	Financial service	Retail	Healthcare	Total
Security Vulnerabilities	99%	100%	100%	98%	99%
Privacy Issues	79%	48%	73%	73%	68%

99% of high-tech apps have at least 1 or more security risks

86% of high-tech apps use dangerous permissions

42% of high-tech apps use weak cryptography

Source: Coalfire 5th Annual Penetration Risk Report

OWASP Top 10 Mobile Application Security

Risks and vulnerabilities found in mobile apps by industry in 2023

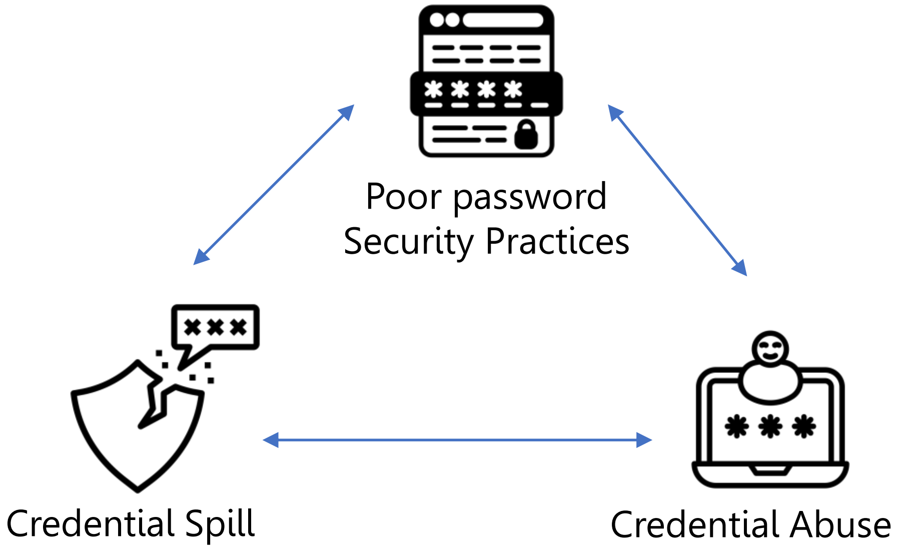
Item	Technology	Financial service	Retail	Healthcare	AVERAGE
Insecure network communication issues	**56%**	35%	48%	51%	**48%**
Insecure storage issues	32%	**48%**	**48%**	44%	**43%**
Weak cryptography issues	67%	83%	**88%**	69%	**77%**
Vulnerable outdate libraries found	40%	**60%**	**60%**	48%	**52%**
OpenSSL (number of vulnerable mobile apps)	9	2	**11**	5	**7**
Insecure code / debug issues	39%	55%	**63%**	39%	**49%**
Insecure code / permissions issues	28%	**54%**	53%	44%	**45%**
Lack of anti-tampering / resiliency	47%	**58%**	**58%**	48%	**53%**

Source: Coalfire 5th Annual Penetration Risk Report

OWASP Top 10 Mobile Application Security

1. Improper Credential Usage

- **Hardcoded Credentials** - If the mobile app contains hardcoded credentials within the app's source code or any configuration files, this is a clear indicator of vulnerability.
- **Insecure Credential Transmission** - If credentials are transmitted without encryption or through insecure channels, this could indicate a vulnerability.
- **Insecure Credential Storage** - If the mobile app stores user credentials on the device in an insecure manner, this could represent a vulnerability.
- **Weak User Authentication** - If user authentication relies on weak protocols or allows for easy bypassing, this could be a sign of vulnerability.



OWASP Top 10 Mobile Application Security

2. Inadequate Supply Chain Security

An attacker can manipulate application functionality by exploiting vulnerabilities in the mobile app supply chain.

This can lead to unauthorized data access or manipulation, denial of service, or complete takeover of the mobile app or device.

- **Lack of Security in Third-Party Components:** Third-party components, such as libraries or frameworks, can contain vulnerabilities that can be exploited by attackers. If the mobile application developer does not vet the third-party components properly or keep them updated, the application can be vulnerable to attacks.

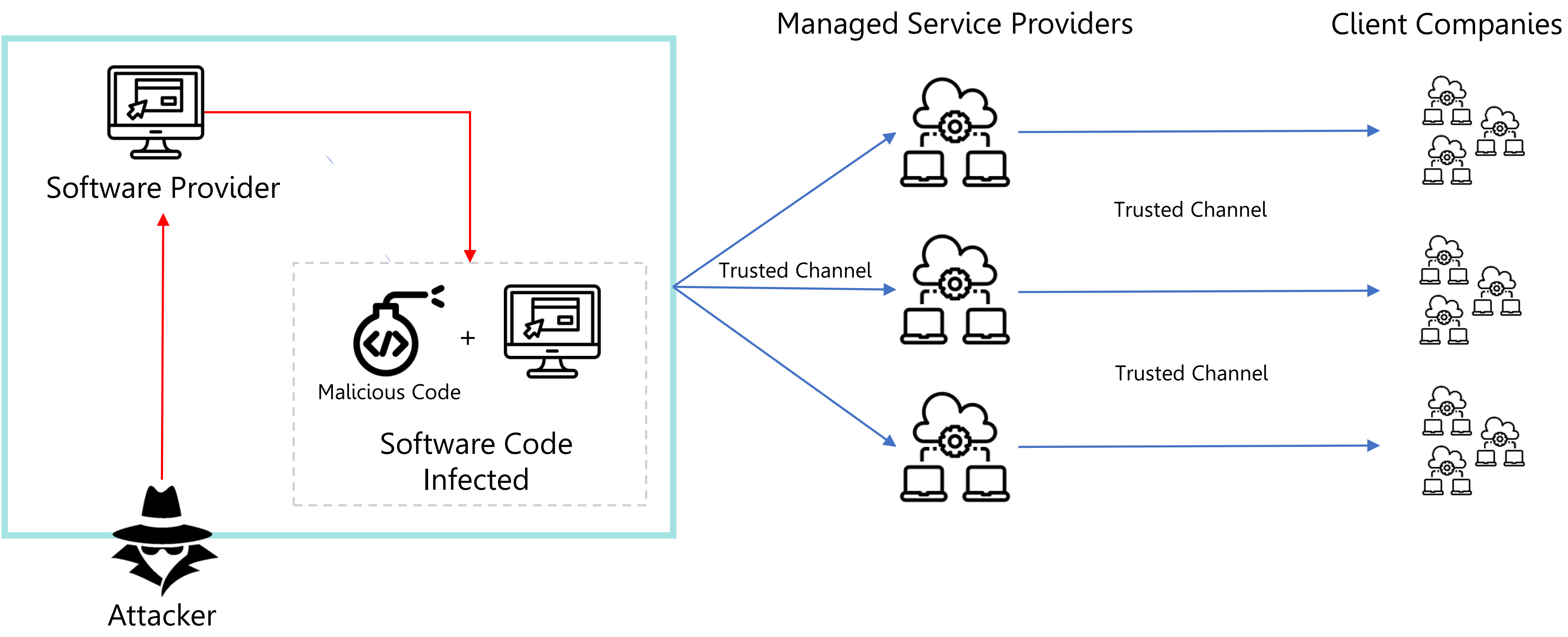
- **Malicious Insider Threats:** Malicious insiders, such as a rogue developer or a supplier, can introduce vulnerabilities into the mobile application intentionally. This can occur if the developer does not implement adequate security controls and monitoring of the supply chain process.

- **Inadequate Testing and Validation:** If the mobile application developer does not test the application thoroughly, it can be vulnerable to attacks. The developer may also fail to validate the security of the supply chain process, leading to vulnerabilities in the application.

- **Lack of Security Awareness:** If the mobile application developer does not have adequate security awareness, they may not implement the necessary security controls to prevent supply chain attacks.

OWASP Top 10 Mobile Application Security

2. Inadequate Supply Chain Security



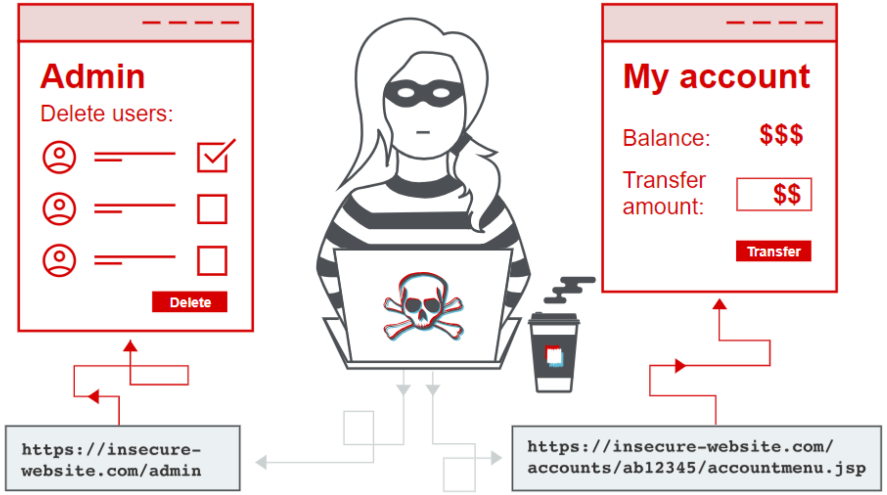
OWASP Top 10 Mobile Application Security

3. Insecure Authentication/Authorization

- **Presence of Insecure Direct Object Reference (IDOR) vulnerabilities** - Noticing an IDOR vulnerability may suggest that the code isn't conducting a proper authorization check.

- **Hidden Endpoints** - Developers might neglect authorization checks on backend hidden functionality, assuming that the hidden functionality will only be accessed by a user with the appropriate role.

- **User Role or Permission Transmissions** - Should the mobile app transmit the user's roles or permissions to a backend system as part of a request, this could signal insecure authorization.

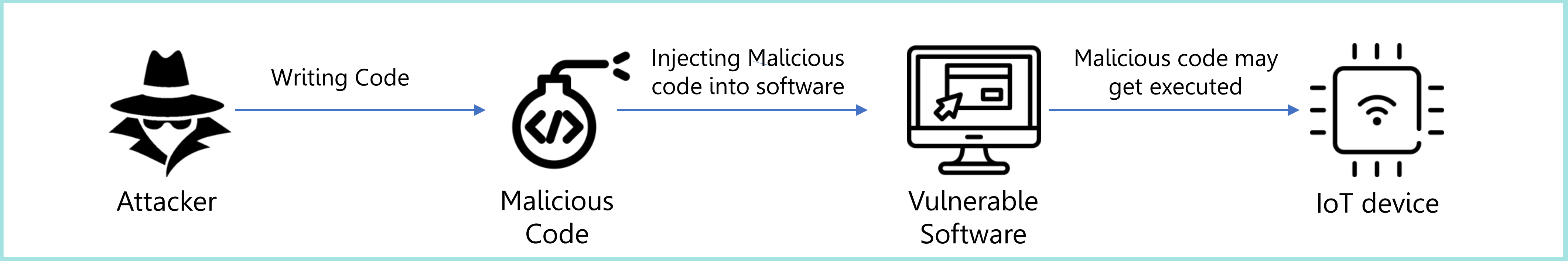

Admin
Delete users:
☑
□
□
Delete
My account
Balance: $$$
Transfer amount: $$
Transfer
https://insecure-website.com/admin
https://insecure-website.com/accounts/ab12345/accountmenu.jsp

OWASP Top 10 Mobile Application Security

4. Insufficient Input/Output Validation

Insufficient validation and sanitization of data from external sources, such as user inputs or network data, in a mobile application can introduce severe security vulnerabilities. Mobile apps that fail to properly validate and sanitize such data are at risk of being exploited through attacks specific to mobile environments, including SQL injection, Command Injection, and cross-site scripting (XSS) attacks.

Inadequate output validation can result in data corruption or presentation vulnerabilities, allowing malicious actors to inject malicious code or manipulate sensitive information displayed to users.


Attacker
Writing Code
Malicious Code
Injecting Malicious code into software
Vulnerable Software
Malicious code may get executed
IoT device

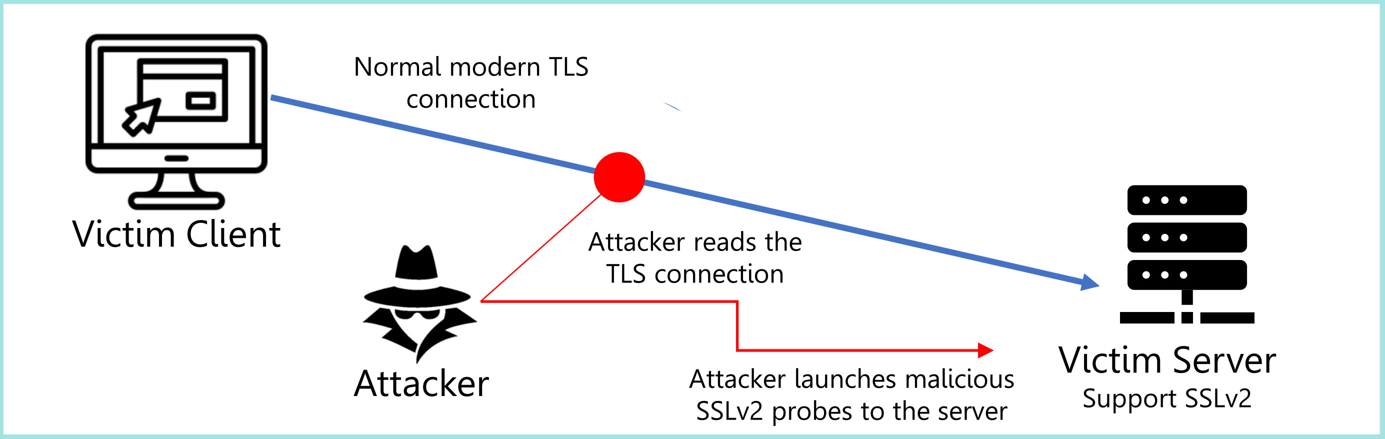
OWASP Top 10 Mobile Application Security

5. Insecure Communication

When the data transmission takes place, it typically goes through the mobile device’s carrier network and the internet, a threat agent listening on the wire can intercept and modify the data if it transmitted in plaintext or using a deprecated encryption protocol. Threat agents might have different motives such as stealing sensitive information, conducting espionage, identity theft and more.

An adversary that shares your local network (compromised or monitored Wi-Fi);

Rogue carrier or network devices (routers, cell towers, proxy’s, etc); or Malware on your mobile device.

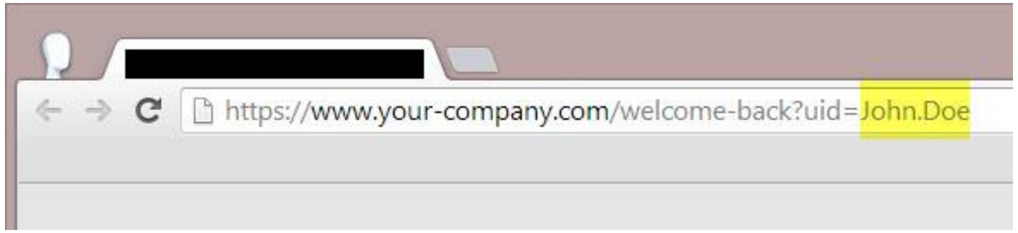

Victim Client
Normal modern TLS connection
Attacker reads the TLS connection
Attacker
Attacker launches malicious SSLv2 probes to the server
Victim Server
Support SSLv2

OWASP Top 10 Mobile Application Security

6. Inadequate Privacy Controls

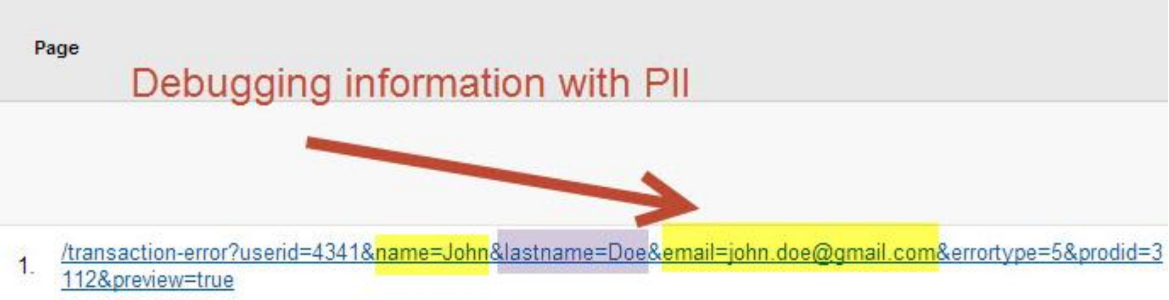
Privacy controls are concerned with protecting Personally Identifiable Information (PII), e.g., names and addresses, credit card information, e-mail and IP addresses, information about health, religion, sexuality and political opinions.

- Insecure data storage and communication
- Data access with insecure authentication and authorization
- Insider attacks on the app's sandbox



[redacted]

https://www.your-company.com/welcome-back?uid=John.Doe



Page

Debugging information with PII

1. /transaction-error?userid=4341&name=John&lastname=Doe&email=john.doe@gmail.com&errortype=5&prodid=3112&preview=true

OWASP Top 10 Mobile Application Security

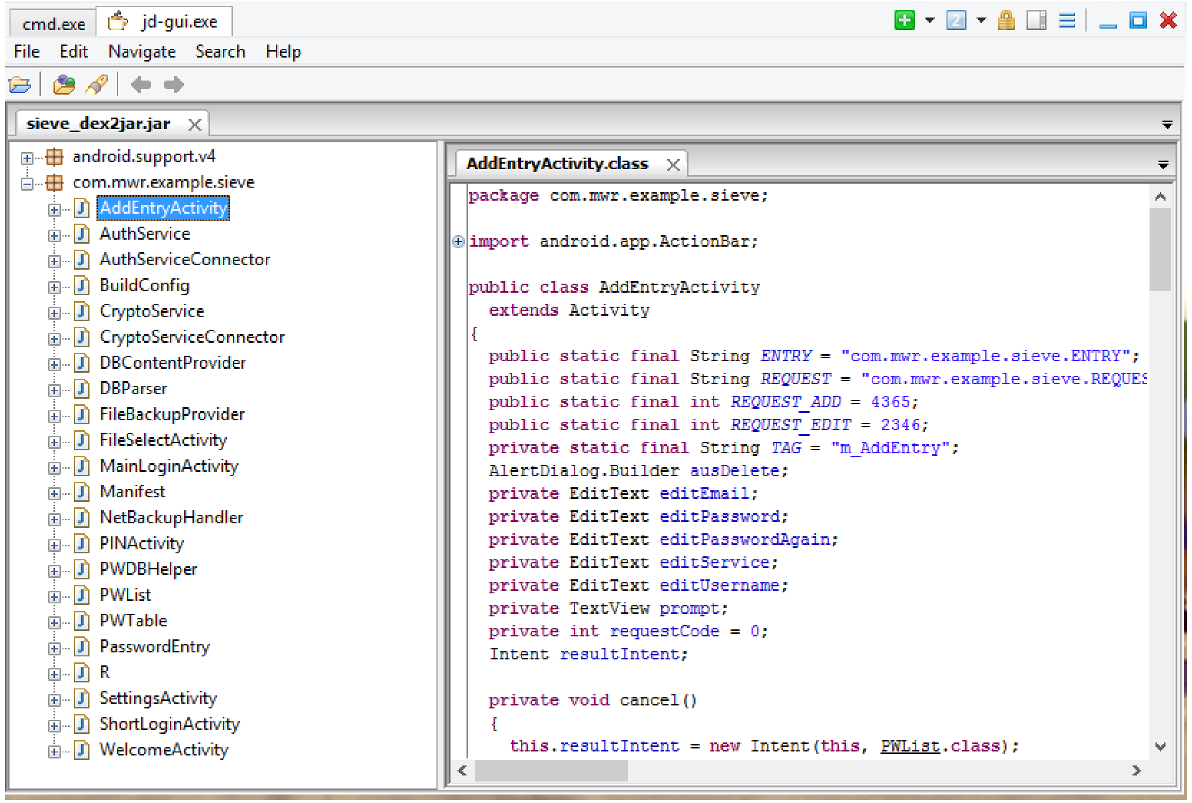
7. Insufficient Binary Protection

The binary could contain valuable secrets, such as commercial API keys or hardcoded cryptographic secrets that an attacker could misuse. In addition, the code in the binary could be valuable on its own, for example, because it contains critical business logic or pre-trained AI models. Some attackers might also not target the app itself but use it to explore potential weaknesses of the corresponding backend to prepare for an attack.

- collecting information
- manipulate app binaries to access paid features for free or to bypass other security checks
- modified to contain malicious code and be distributed via third-party app stores

OWASP Top 10 Mobile Application Security

7. Insufficient Binary Protection

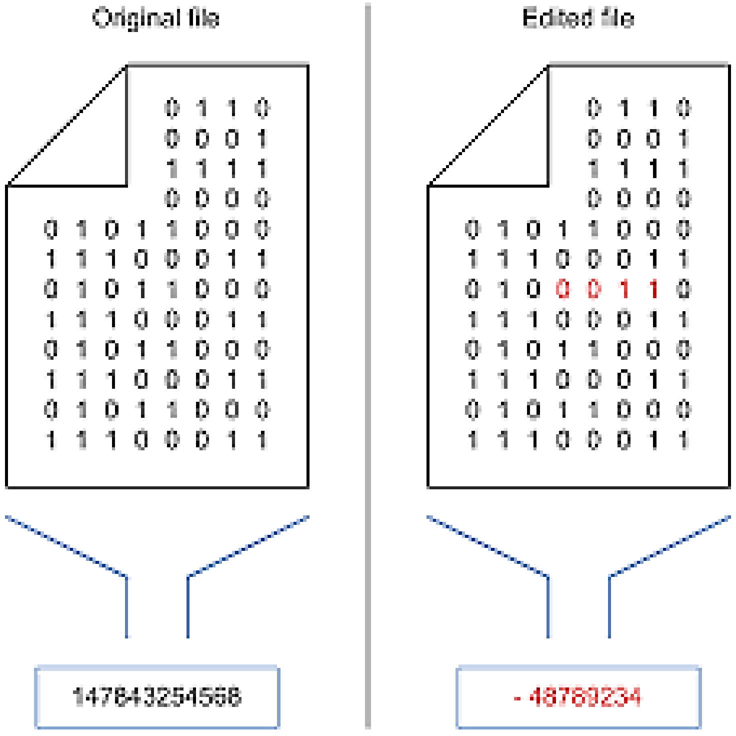


```
package com.mwr.example.sieve;

import android.app.ActionBar;

public class AddEntryActivity
  extends Activity
{
  public static final String ENTRY = "com.mwr.example.sieve.ENTRY";
  public static final String REQUEST = "com.mwr.example.sieve.REQUES
  public static final int REQUEST_ADD = 4365;
  public static final int REQUEST_EDIT = 2346;
  private static final String TAG = "m_AddEntry";
  AlertDialog.Builder ausDelete;
  private EditText editEmail;
  private EditText editPassword;
  private EditText editPasswordAgain;
  private EditText editService;
  private EditText editUsername;
  private TextView prompt;
  private int requestCode = 0;
  Intent resultIntent;

  private void cancel()
  {
    this.resultIntent = new Intent(this, PWList.class);
```


Original file

```
        0 1 1 0
        0 0 0 1
        1 1 1 1
        0 0 0 0
0 1 0 1 1 0 0 0
1 1 1 0 0 0 1 1
0 1 0 1 1 0 0 0
1 1 1 0 0 0 1 1
0 1 0 1 1 0 0 0
1 1 1 0 0 0 1 1
0 1 0 1 1 0 0 0
1 1 1 0 0 0 1 1
```

147843254568

Edited file

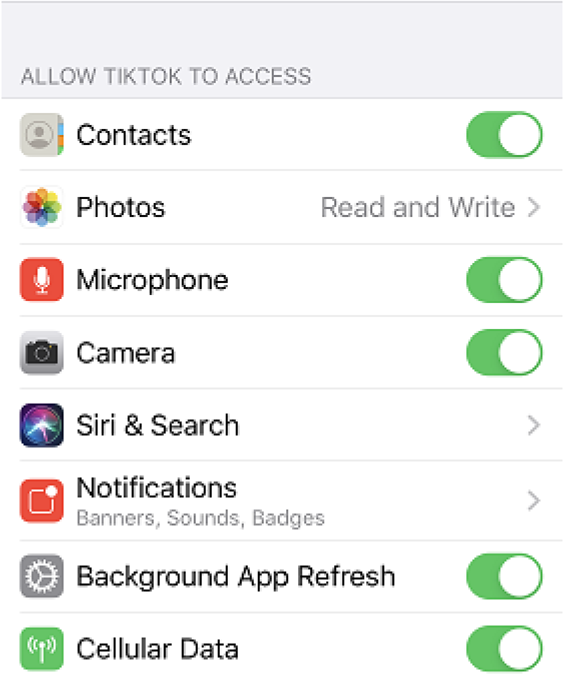
```
        0 1 1 0
        0 0 0 1
        1 1 1 1
        0 0 0 0
0 1 0 1 1 0 0 0
1 1 1 0 0 0 1 1
0 1 0 0 0 1 1 0
1 1 1 0 0 0 1 1
0 1 0 1 1 0 0 0
1 1 1 0 0 0 1 1
0 1 0 1 1 0 0 0
1 1 1 0 0 0 1 1
```

- 48789234

OWASP Top 10 Mobile Application Security

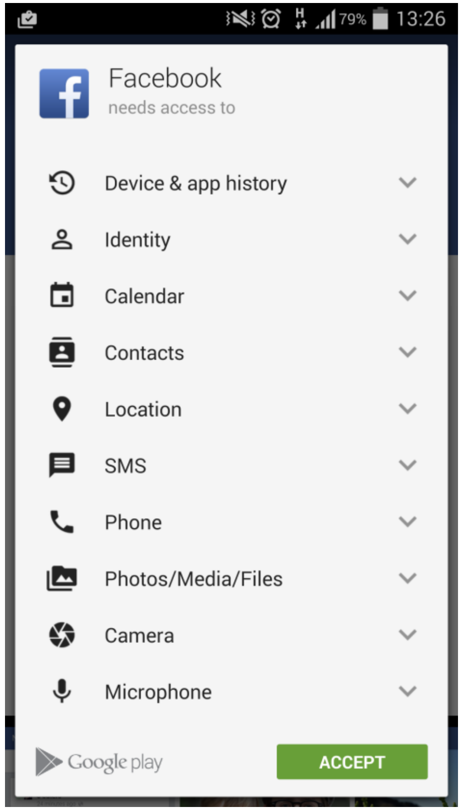
8. Security Misconfiguration

Security misconfiguration in mobile apps refers to the improper configuration of security settings, permissions, and controls that can lead to vulnerabilities and unauthorized access.



ALLOW TIKTOK TO ACCESS

- Contacts ●
- Photos — Read and Write >
- Microphone ●
- Camera ●
- Siri & Search >
- Notifications — Banners, Sounds, Badges >
- Background App Refresh ●
- Cellular Data ●



Facebook
needs access to

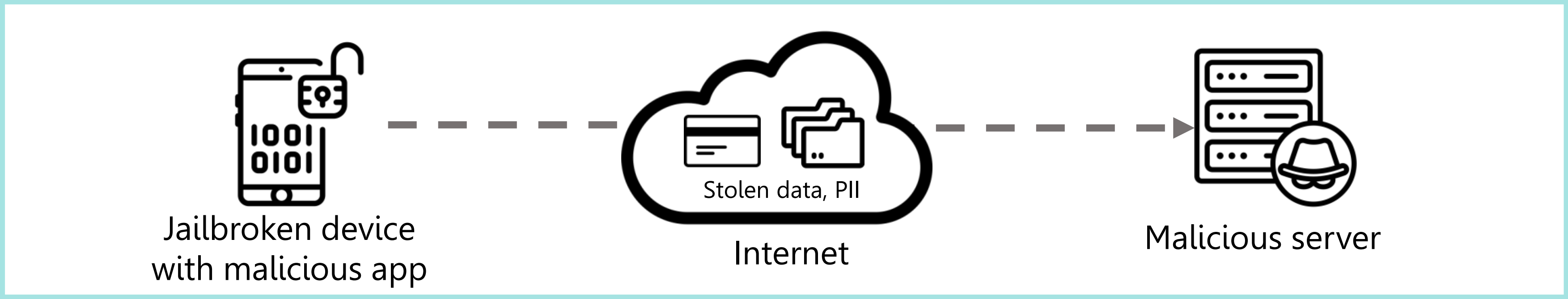
- Device & app history
- Identity
- Calendar
- Contacts
- Location
- SMS
- Phone
- Photos/Media/Files
- Camera
- Microphone

OWASP Top 10 Mobile Application Security

9. Insecure Data Storage

Insecure data storage in a mobile application can attract various threat agents who aim to exploit the vulnerabilities and gain unauthorized access to sensitive information.

These threat agents exploit vulnerabilities like weak encryption, insufficient data protection, insecure data storage mechanisms, and improper handling of user credentials. It is crucial for mobile app developers and organizations to implement strong security measures, such as robust encryption, secure data storage practices, and adherence to best practices for mobile application security, to mitigate the risks associated with insecure data storage.

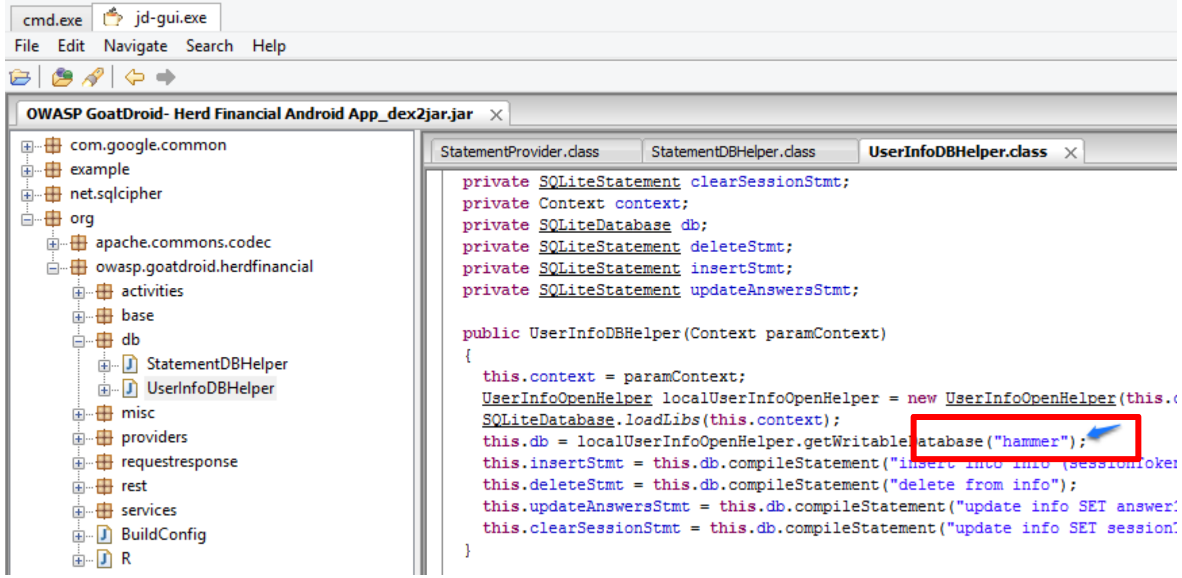


OWASP Top 10 Mobile Application Security

10. Insufficient Cryptography

Threat agents who exploit insecure cryptography in mobile applications can undermine the confidentiality, integrity, and authenticity of sensitive information.

These threat agents include attackers who target cryptographic algorithms or implementations to decrypt sensitive data, malicious insiders who manipulate cryptographic processes or leak encryption keys, cybercriminals who exploit weak encryption to steal valuable data or conduct financial fraud, and attackers who leverage vulnerabilities in cryptographic protocols or libraries.



```
private SQLiteStatement clearSessionStmt;
private Context context;
private SQLiteDatabase db;
private SQLiteStatement deleteStmt;
private SQLiteStatement insertStmt;
private SQLiteStatement updateAnswersStmt;

public UserInfoDBHelper(Context paramContext)
{
  this.context = paramContext;
  UserInfoOpenHelper localUserInfoOpenHelper = new UserInfoOpenHelper(this.c
  SQLiteDatabase.loadLibs(this.context);
  this.db = localUserInfoOpenHelper.getWritableDatabase("hammer");
  this.insertStmt = this.db.compileStatement("insert into info (sessionToke
  this.deleteStmt = this.db.compileStatement("delete from info");
  this.updateAnswersStmt = this.db.compileStatement("update info SET answer
  this.clearSessionStmt = this.db.compileStatement("update info SET session
}
```

Mobile Security Keys Takeaway

Mobile Security Keys Takeaway

- Proper Authentication and Authorization
- Secure communication
- Secure Data Storage
- Components without known vulnerabilities
- Preventing Reverse engineering

4.4 : OWASP Proactive Control

1 : Overview

2 : Application Security Protection

3 : Threat Modeling

4 : OWASP

4.1 : OWASP - Web security

4.2 : OWASP - API Security

4.3 : OWASP - Mobile security

4.4 : OWASP - Proactive Control

OWASP Proactive Control

OWASP Proactive Control

The goal of the **OWASP Top 10 Proactive Controls project** (OPC) is to raise awareness about application security by describing the most important areas of concern that software developers must be aware of.

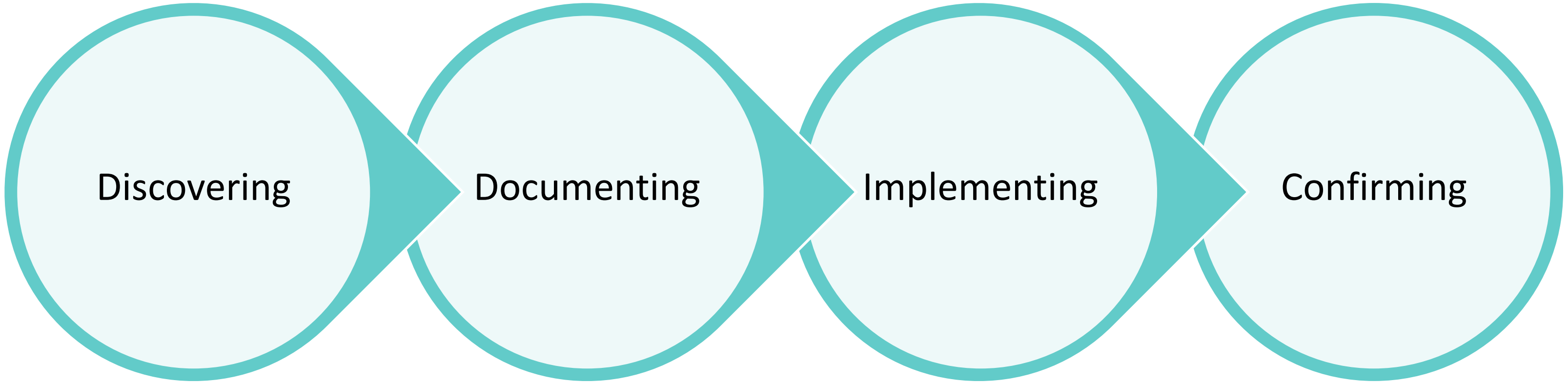
- C1: Define Security Requirements
- C2: Leverage Security Frameworks and Libraries
- C3: Secure Database Access
- C4: Encode and Escape Data
- C5: Validate All Inputs
- C6: Implement Digital Identity
- C7: Enforce Access Controls
- C8: Protect Data Everywhere
- C9: Implement Security Logging and Monitoring
- C10: Handle All Errors and Exceptions

OWASP Proactive Control

C1. Define Security Requirements

A security requirement is a statement of needed security functionality that ensures one of many different security properties of software is being satisfied. Security requirements are derived from industry standards, applicable laws, and a history of past vulnerabilities. Security requirements define new features or additions to existing features to solve a specific security problem or eliminate a potential vulnerability.

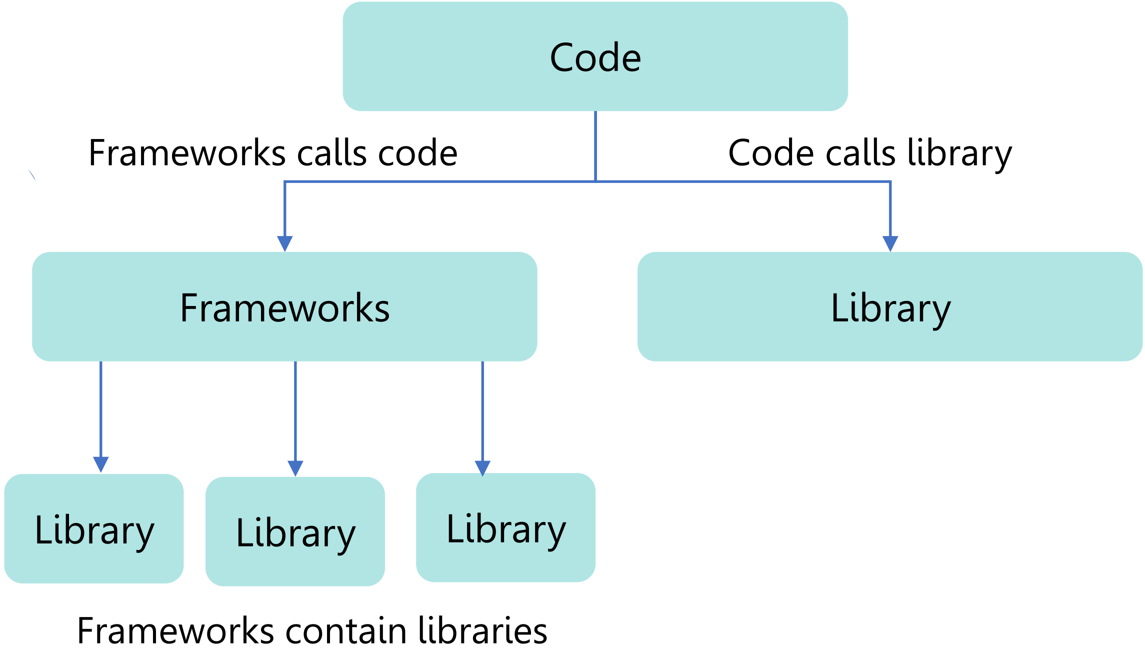
Successful use of security requirements involves four steps. The process includes discovering / selecting, documenting, implementing, and then confirming correct implementation of new security features and functionality within an application.



OWASP Proactive Control

C2. Leverage Security Frameworks and Libraries

Secure coding libraries and software frameworks with embedded security help software developers guard against security-related design and implementation flaws. A developer writing an application from scratch might not have sufficient knowledge, time, or budget to properly implement or maintain security features. Leveraging security frameworks helps accomplish security goals more efficiently and accurately.


Code

Frameworks calls code

Code calls library

Frameworks

Library

Library

Library

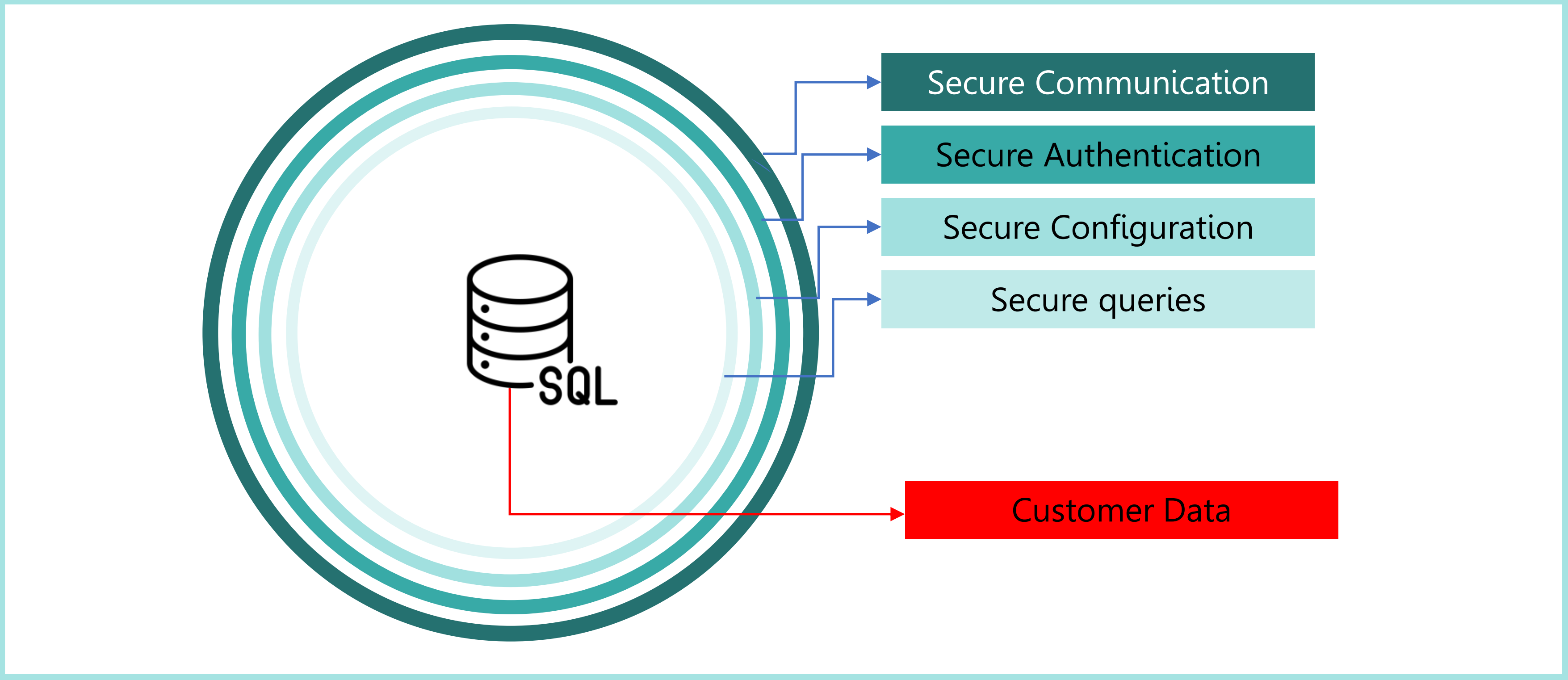
Library

Frameworks contain libraries

OWASP Proactive Control

C3. Secure Database Access

This section describes secure access to all data stores, including both relational databases and NoSQL databases. Some areas to consider:



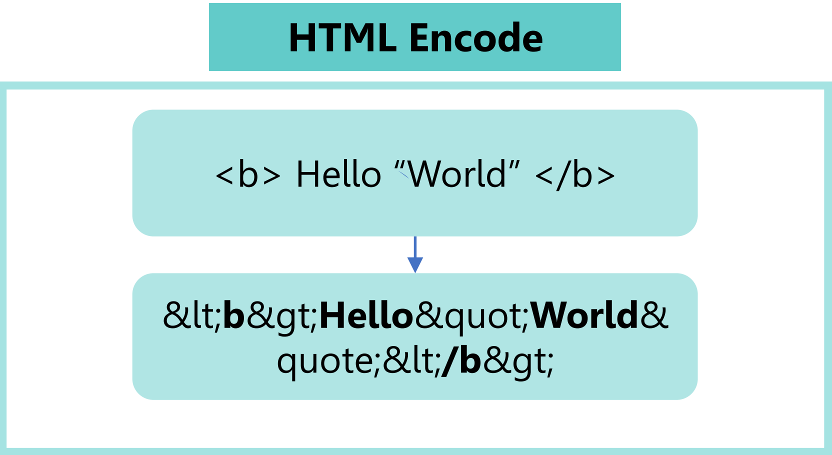
OWASP Proactive Control

C4. Encode and Escape Data

Encoding (commonly called “Output Encoding”) involves translating special characters into some different but equivalent form that is no longer dangerous in the target interpreter,

Escaping involves adding a special character before the character/string to avoid it being misinterpreted, for example, adding a \ character before a " (double quote) character so that it is interpreted as text and not as closing a string.

>	Greater than	&gt;	&#62;
&	Amperand	&apm;	&#38;
€	Cent	&cent;	&#162;
£	Pound	&pound;	&#163;
¥	yen	&yen;	&#165;



HTML Encode

<b> Hello “World” </b>

&lt;**b**&gt;**Hello**&quot;**World**&quote;&lt;**/b**&gt;

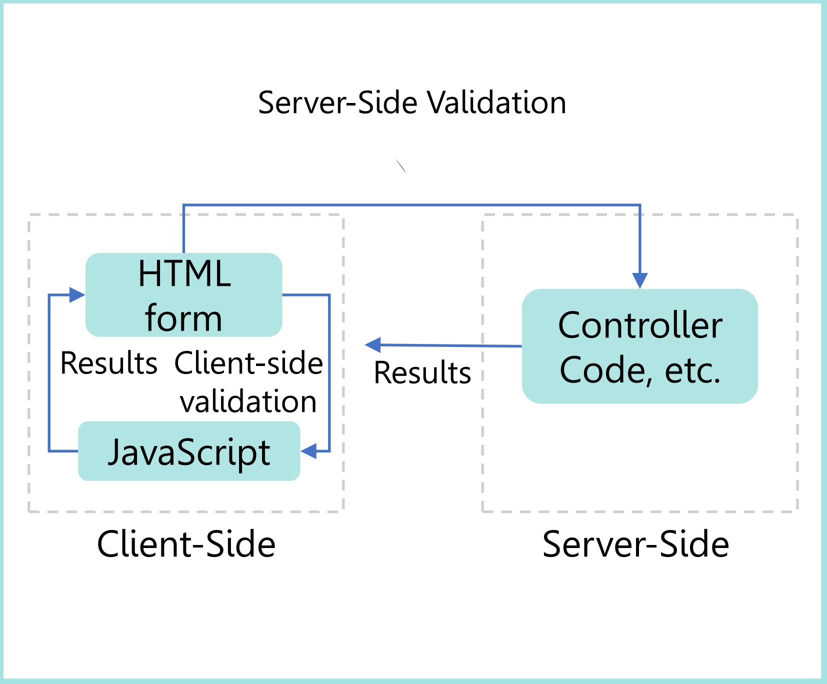
OWASP Proactive Control

C5. Validate All Inputs

Input validation is a programming technique that ensures only properly formatted data may enter a software system component.

Input validation can be implemented using any programming technique that allows effective enforcement of syntactic and semantic correctness, for example:

- Data type
- Data format
- Minimum and maximum length for the data
- Allowed set of characters to be accepted
- Input validation must always be done on the server-side for security.



OWASP Proactive Control

C6. Implement Digital Identity

Digital Identity is the unique representation of a user (or other subject) as they engage in an online transaction.

Authentication is the process of verifying that an individual or entity is who they claim to be.

Session management is a process by which a server maintains the state of the users authentication so that the user may continue to use the system without re-authenticating.

OWASP Proactive Control

C6. Implement Digital Identity

Maturity Level 1	Maturity Level 2	Maturity Level 3
Password + Voice Password + SMS	MS authenticator passwordless Password + Hardware tokens OTP Password + authenticator number match Password + software tokens OTP	Certificate based authorization FIDO2 security key Windows Hello
+ Any method in Maturity Levels 2 & 3	+ Any method in Maturity Levels 3	

OWASP Proactive Control

C7. Enforce Access Controls

Access Control (or Authorization) is the process of granting or denying *specific requests* from a user, program, or process. Access control also involves the act of *granting and revoking those privileges*.

Access Control Design Principles

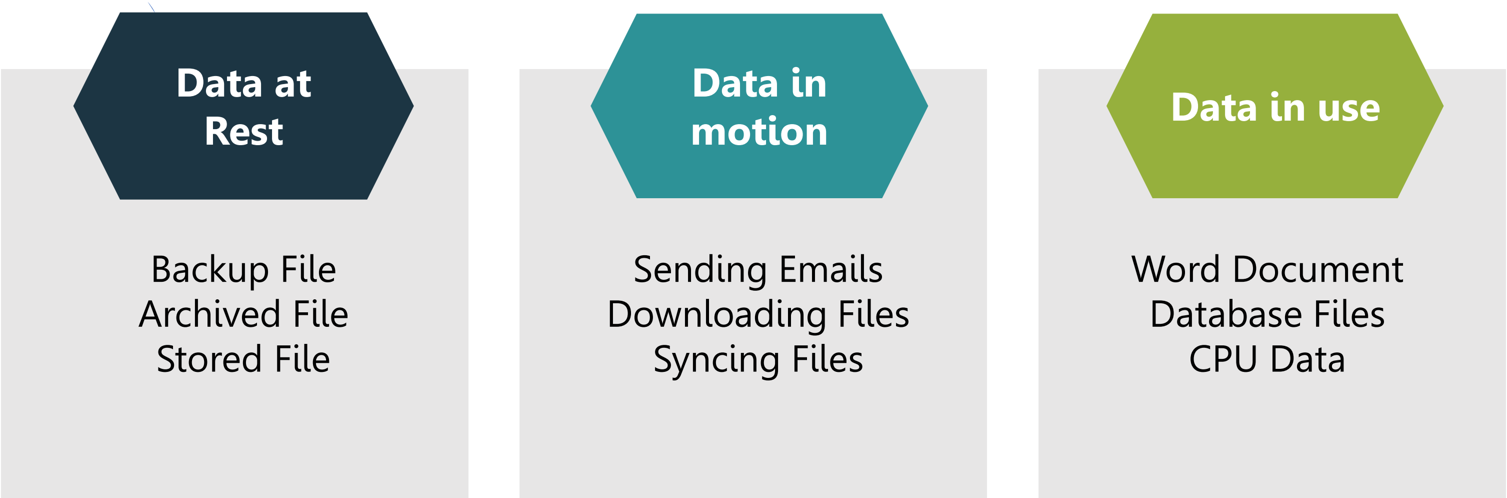
- **Design Access Control Thoroughly Up Front**
- **Force All Requests to Go Through Access Control Checks**
- **Deny by Default**
- **Principle of Least Privilege**
- **Don’t Hardcode Roles**
- **Log All Access Control Events**

OWASP Proactive Control

C8. Protect Data Everywhere

Sensitive data such as passwords, credit card numbers, health records, personal information and business secrets require extra protection, particularly if that data falls under privacy laws (EU's General Data Protection Regulation GDPR), financial data protection rules such as PCI Data Security Standard (PCI DSS) or other regulations.

Attackers can steal data from web and webservice applications in a number of ways. For example, if sensitive information in sent over the internet without communications security, then an attacker on a shared wireless connection could see and steal another user's data. Also, an attacker could use SQL Injection to steal passwords and other credentials from an applications database and expose that information to the public.



Data at Rest

Backup File
Archived File
Stored File

Data in motion

Sending Emails
Downloading Files
Syncing Files

Data in use

Word Document
Database Files
CPU Data

OWASP Proactive Control

C9. Implement Security Logging and Monitoring

Logging is a concept that most developers already use for debugging and diagnostic purposes. Security logging is an equally basic concept: to log security information during the runtime operation of an application. Monitoring is the live review of application and security logs using various forms of automation. The same tools and patterns can be used for operations, debugging and security purposes.

Secure Logging Design

- Validate any dangerous characters
- Do not log sensitive information
- Protect log integrity
- Setup permission of log files
- Centralized monitoring.

OWASP Proactive Control

C10. Handle All Errors and Exceptions

Exception handling is a programming concept that allows an application to respond to different error states (like network down, or database connection failed, etc) in various ways. Handling exceptions and errors correctly is critical to making your code reliable and secure.

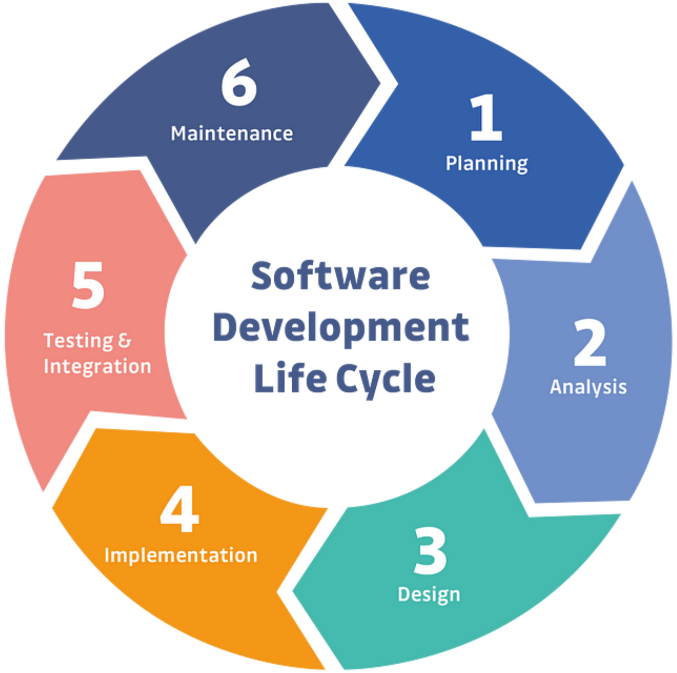
Recommendation for Handling Errors and Exceptions

- Centralized Manage Exceptions
- No Leak Critical Data
- Enough Information
- Carefully test and verify

Security Proactive Controls Keys Takeaway

Security Proactive Controls Keys Takeaway

- Implement Strong Authentication
- Enforce Principle of Least Privilege
- Secure Data Everywhere
- Validate all data before processing
- Enough information for handling errors and logging



Summary of Application security

Summary of this topic

What is the importance of Application Security?	What are the prevention and detection methods?	What strategic approaches can be employed for securing applications?	What methods can be used to identify and address existing threats and vulnerabilities?
Protect software applications from threats and vulnerabilities	firewall, DDos, WAF, Secure Coding, Authentication, Authorization, etc.	STRIDE, PASTA, TRIKE, VAST, Persona non grata, LINDDUN, etc.	OWASP and CVE