

OWASP Security Fundamentals

Simple 20 MCQ Questions on OWASP Basics, Secret Manager, Threat Modeling, SonarQube, and Static Security App Testing for Beginners

Master the fundamentals of application security through practical multiple-choice questions covering essential topics every security-conscious developer needs to know. This comprehensive assessment covers OWASP principles, secret management best practices, threat modeling methodologies, SonarQube analysis, and static application security testing.

Each question includes scenario-based examples to help you apply theoretical knowledge to real-world security challenges. Whether you're starting your cybersecurity journey or reinforcing foundational concepts, these questions will test your understanding of critical security practices that protect modern applications from common vulnerabilities and threats.

Assessment Structure

Question Format

Each question presents a security scenario or concept with four possible answers, testing practical application of security principles rather than memorization.

Coverage Areas

Questions span five critical security domains: OWASP fundamentals, secret management, threat modeling, SonarQube analysis, and static security testing.

Correct Answers

Each question includes the correct answer with explanations to reinforce learning and provide immediate feedback on security best practices.

The following table structure presents each question with four multiple-choice options (A, B, C, D) and identifies the correct answer. This format enables easy integration into learning management systems or conversion to various assessment formats while maintaining clarity and educational value.

Question 1: OWASP Foundation

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
What does OWASP stand for?	Open Web Application Security Project	Online Web Application Security Program	Open World Application Security Protocol	Organization for Web Application Security and Protection	Open Web Application Security Project



Learning Point: OWASP (Open Web Application Security Project) is a nonprofit foundation dedicated to improving software security through community-driven projects, standards, and educational resources. Understanding this foundational organization is crucial for anyone working in application security.

Question 2: SQL Injection Vulnerability

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
Which OWASP Top 10 vulnerability involves attackers injecting malicious SQL code?	Cross-Site Scripting (XSS)	SQL Injection	Broken Authentication	Security Misconfiguration	SQL Injection

SQL Injection remains one of the most critical and common vulnerabilities in web applications. This attack occurs when malicious SQL code is inserted into application queries, potentially allowing attackers to view, modify, or delete database information. Modern applications should always use parameterized queries and input validation to prevent these attacks.

SQL Injection attacks can compromise entire databases, making proper input sanitization and parameterized queries essential security controls.

Question 3: Secret Manager Purpose

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
What is the main purpose of a Secret Manager in application security?	Store and manage sensitive credentials securely	Monitor network traffic	Scan code for bugs	Manage user permissions	Store and manage sensitive credentials securely



Secure Storage

Secret Managers provide encrypted storage for sensitive data like API keys, passwords, and certificates, eliminating hardcoded secrets in source code.



Automatic Rotation

Many secret managers automatically rotate credentials on schedule, reducing the risk of compromised long-term secrets being exploited.



Access Control

Fine-grained permissions ensure only authorized applications and users can access specific secrets, implementing principle of least privilege.

Question 4: Threat Modeling Process

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
In threat modeling, what is the first step?	Identify potential threats	Implement security controls	Test the application	Deploy the application	Identify potential threats

O1

Identify Threats

Begin by systematically identifying potential security threats that could affect your application or system.

O3

Design Controls

Develop appropriate security controls and countermeasures to mitigate the highest priority threats.

O2

Assess Impact

Evaluate the potential damage and likelihood of each identified threat to prioritize security efforts effectively.

O4

Validate & Monitor

Test implemented controls and establish ongoing monitoring to ensure continued protection against evolving threats.

Question 5: Static Application Security Testing

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
Which tool is commonly used for static application security testing (SAST)?	SonarQube	OWASP ZAP	Burp Suite	Wireshark	SonarQube

SonarQube excels as a static application security testing (SAST) tool because it analyzes source code without executing it, identifying security vulnerabilities, code quality issues, and technical debt early in the development lifecycle. Unlike dynamic testing tools that require running applications, SAST tools like SonarQube can detect problems during the coding phase.

This early detection capability allows developers to fix security issues before they reach production, significantly reducing remediation costs and improving overall application security posture. SonarQube integrates seamlessly into CI/CD pipelines, providing continuous security feedback throughout the development process.



Tool Categories

- **SAST:** SonarQube - analyzes source code
- **DAST:** OWASP ZAP - tests running applications
- **Manual Testing:** Burp Suite - interactive security testing
- **Network Analysis:** Wireshark - packet inspection

Question 6: Cross-Site Scripting Vulnerability

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
What type of vulnerability does Cross-Site Scripting (XSS) exploit?	Server misconfiguration	Injection of malicious scripts into web pages	Weak password policies	Unencrypted data storage	Injection of malicious scripts into web pages

1

Reflected XSS

Malicious script reflected back from server in response to user input, typically through URL parameters or form data.

2

Stored XSS

Malicious script permanently stored on target server (database, message forum, comment field) and served to other users.

3

DOM-based XSS

Client-side script execution where malicious payload modifies DOM environment in victim's browser without server interaction.

Question 7: Hardcoded Password Scenario

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
Scenario: You find hardcoded passwords in your source code repository. What is the best practice to fix this?	Leave them as is	Use a Secret Manager to store credentials securely	Encrypt passwords in the code	Remove passwords entirely	Use a Secret Manager to store credentials securely

Critical Security Issue: Hardcoded credentials in source code represent one of the most dangerous security anti-patterns, as they're visible to anyone with repository access and cannot be easily rotated.

Immediate Actions

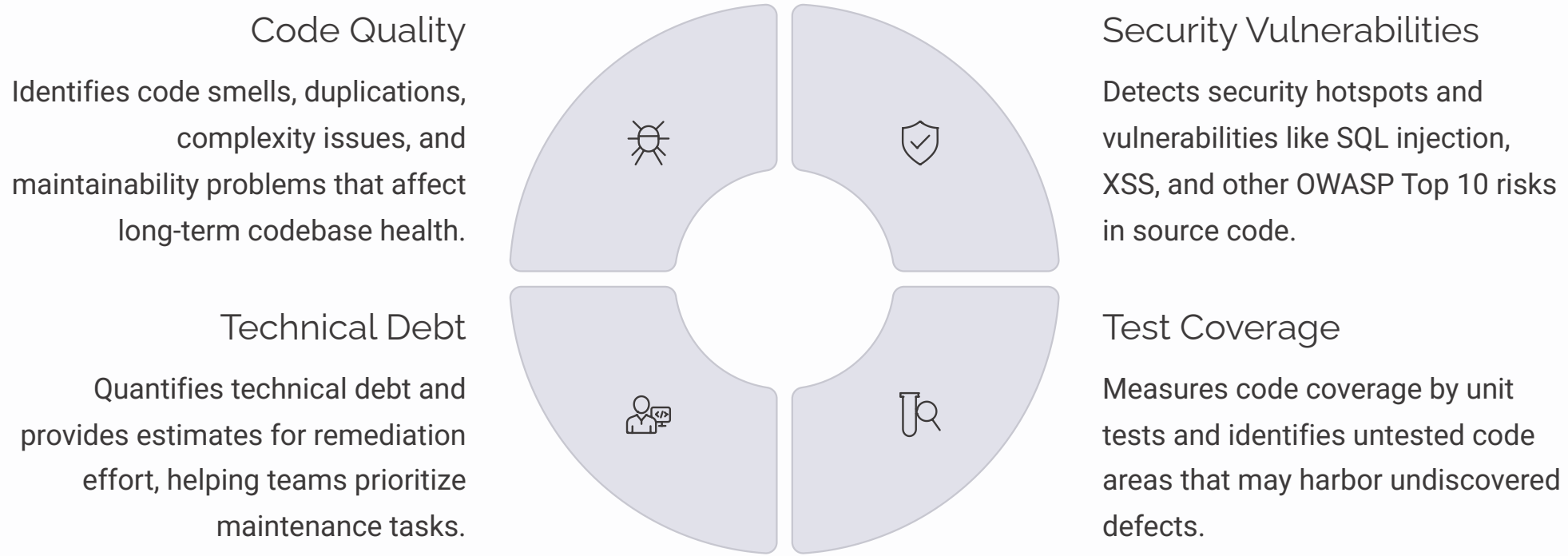
- Remove hardcoded credentials from code
- Revoke and rotate exposed credentials
- Scan commit history for other secrets

Long-term Solution

- Implement Secret Manager (AWS Secrets Manager, HashiCorp Vault)
- Use environment variables for configuration
- Implement automated secret scanning in CI/CD

Question 8: SonarQube Analysis Focus

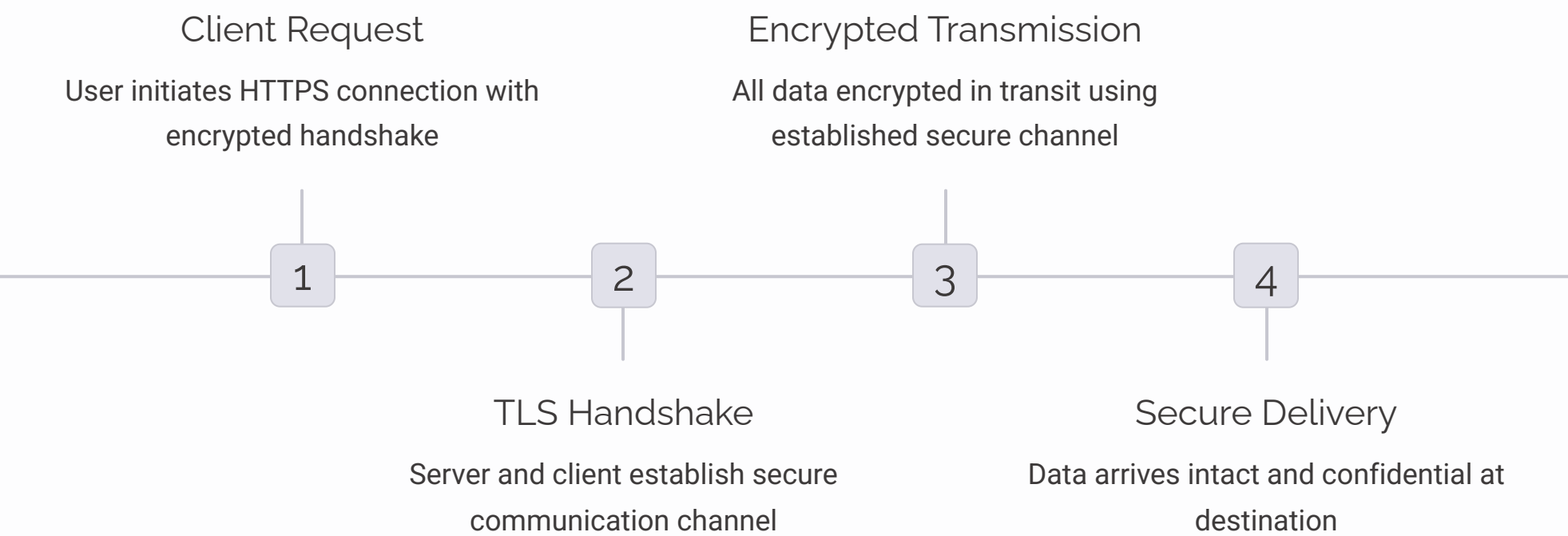
Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
What does SonarQube primarily analyze?	Network traffic	Source code quality and security vulnerabilities	User behavior	Server logs	Source code quality and security vulnerabilities




Question 9: Data in Transit Protection

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
Scenario: During threat modeling, you identify that an attacker could intercept data in transit. What security control should you recommend?	Use HTTPS/TLS encryption	Disable logging	Use weak passwords	Remove authentication	Use HTTPS/TLS encryption

When threat modeling reveals risks to data in transit, implementing proper encryption becomes critical. HTTPS/TLS provides end-to-end encryption that protects data as it travels between client and server, preventing eavesdropping, tampering, and man-in-the-middle attacks.



 Never transmit sensitive data over unencrypted HTTP connections in production environments. Always enforce HTTPS for authentication, payment processing, and personal data handling.

Question 10: SAST Benefits

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
What is the benefit of Static Application Security Testing (SAST)?	Detects vulnerabilities early in the development lifecycle	Tests application performance	Monitors live network traffic	Encrypts data at rest	Detects vulnerabilities early in the development lifecycle

Early Detection Advantages

Static Application Security Testing (SAST) analyzes source code before application deployment, enabling security teams and developers to identify and remediate vulnerabilities during the development phase rather than in production.

This early detection approach significantly reduces remediation costs, as fixing security issues in development is exponentially cheaper than addressing them after deployment. Studies show that fixing a bug in production can cost 100 times more than fixing it during development.

100X

Cost Reduction

Production fixes cost 100x more than development fixes

80%

Faster Resolution

Earlier detection leads to 80% faster vulnerability resolution

90%

Coverage Improvement

SAST can analyze 90% of application codebase automatically

Question 11: Insecure Direct Object References

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
Which OWASP vulnerability involves improper validation of user input leading to unauthorized access?	Broken Authentication	Security Misconfiguration	Insecure Direct Object References (IDOR)	Sensitive Data Exposure	Insecure Direct Object References (IDOR)

Insecure Direct Object References (IDOR) occur when applications provide direct access to objects based on user-supplied input without proper authorization checks. This vulnerability allows attackers to bypass authorization and access data belonging to other users by manipulating object references.

Vulnerable Example

```
GET /account?id=12345
```

Attacker changes ID to access other accounts: `/account?id=67890`

Secure Implementation

```
if (account.owner !=  
    currentUser) {  
    return UNAUTHORIZED;  
}
```

Always verify user authorization before granting object access

Prevention Methods

- Use indirect references or UUIDs
- Implement proper access controls
- Validate user permissions per request

Question 12: Secret Exposure in Logs

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
Scenario: You want to ensure your application secrets are not exposed in logs. What should you do?	Log all data including secrets	Mask or exclude secrets from logs	Store secrets in plain text	Share secrets via email	Mask or exclude secrets from logs

1

Identify Sensitive Data

Catalog all sensitive information including passwords, API keys, tokens, and personal data that should never appear in logs.

2

Implement Log Filtering

Configure logging frameworks to automatically detect and mask sensitive patterns before writing to log files.

3

Regular Audits

Periodically review log files to ensure no sensitive data is accidentally being logged by new code changes.

Log files are often stored for extended periods, backed up to multiple locations, and accessed by various team members. Ensuring secrets never enter logs prevents accidental exposure through log aggregation systems, backup files, or unauthorized access to log storage.

Question 13: Threat Modeling Goals

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
What is the main goal of threat modeling?	To identify, prioritize, and mitigate security risks	To write more code	To increase application speed	To reduce server costs	To identify, prioritize, and mitigate security risks

Threat modeling is a systematic approach to identifying, understanding, and addressing security threats in applications and systems. By proactively analyzing potential attack vectors and vulnerabilities, organizations can implement appropriate security controls before threats materialize into actual security incidents.



Question 14: SonarQube Developer Benefits

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
SonarQube can help developers by:	Automatically fixing all bugs	Highlighting code smells and security issues	Managing user accounts	Encrypting source code	Highlighting code smells and security issues

SonarQube serves as a developer's quality assurance partner by continuously analyzing code and providing actionable feedback on both security vulnerabilities and code quality issues. Rather than automatically fixing problems, it educates developers by highlighting issues and explaining why they matter.

This educational approach helps developers learn security best practices and improve code quality over time. SonarQube integrates directly into IDEs and CI/CD pipelines, providing immediate feedback during development rather than waiting for formal security reviews or production incidents.

The tool's ability to track technical debt and provide remediation guidance helps development teams make informed decisions about code maintenance priorities and resource allocation.

Code Quality Metrics

Complexity, duplication, maintainability index

Security Hotspots

Potential vulnerabilities requiring manual review

Technical Debt

Quantified remediation effort estimates

Coverage Gaps

Untested code areas needing attention

Question 15: Buffer Overflow Response

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
Scenario: Your static code analysis tool flags a potential buffer overflow. What should you do?	Ignore it	Investigate and fix the vulnerability	Delete the code	Disable the tool	Investigate and fix the vulnerability

Buffer overflow vulnerabilities represent serious security risks that can lead to arbitrary code execution, system crashes, or data corruption. When static analysis tools flag potential buffer overflows, immediate investigation is crucial because these vulnerabilities are frequently exploited by attackers to gain system control.

O1

Analyze the Alert

Review the specific code location and understand why the tool flagged this as a potential buffer overflow vulnerability.

O2

Verify the Issue

Manually inspect the code to confirm whether the buffer overflow risk is legitimate or a false positive.

O3

Implement Fix

Apply appropriate bounds checking, use safe string functions, or implement proper input validation to eliminate the vulnerability.

O4

Test and Validate

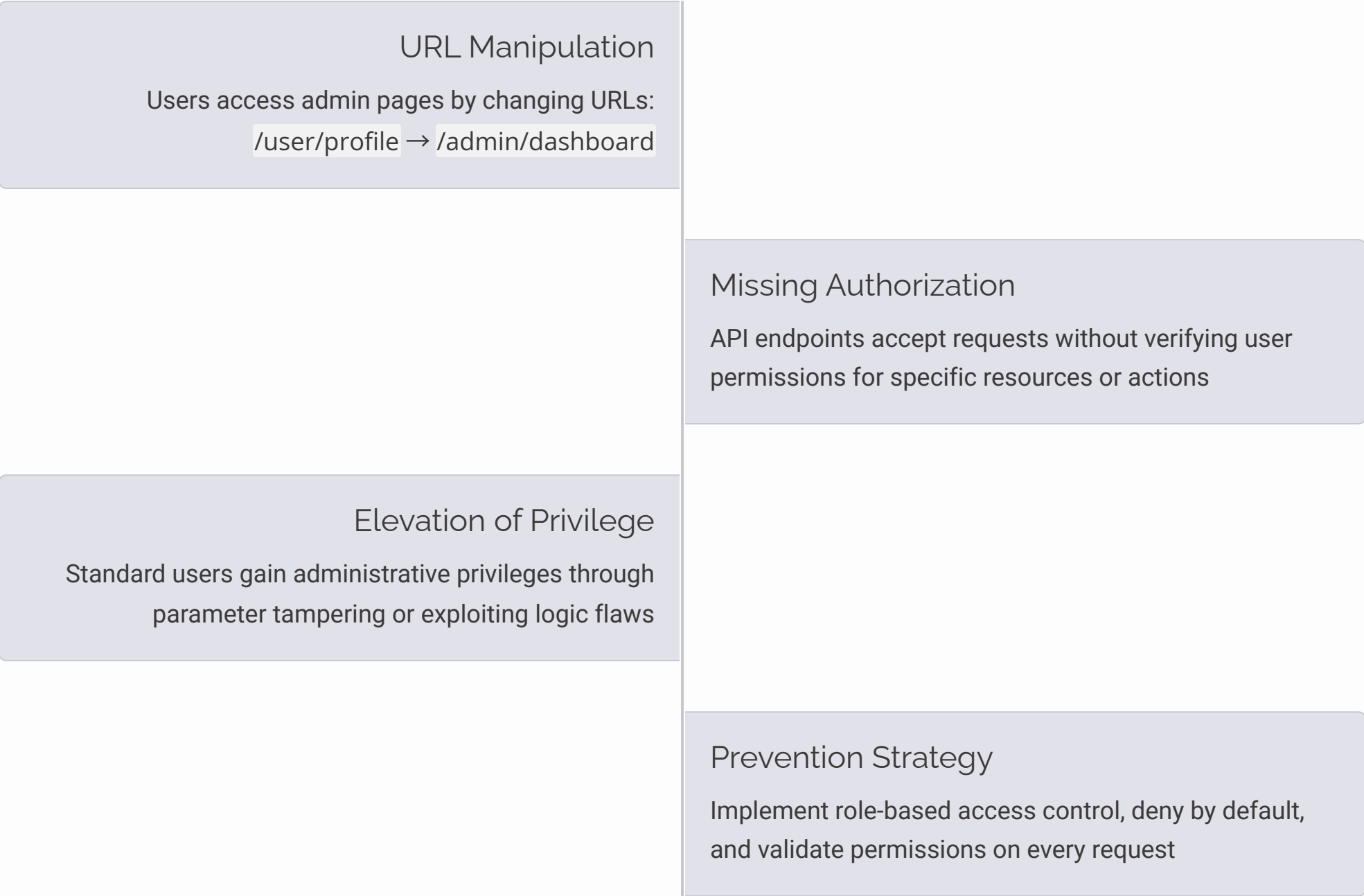
Thoroughly test the fix to ensure it resolves the vulnerability without breaking functionality, then re-scan to confirm resolution.

⊗ Never ignore security alerts from static analysis tools, even if they seem minor. Buffer overflows have been responsible for many high-profile security breaches and can provide attackers with complete system control.

Question 16: Broken Access Control

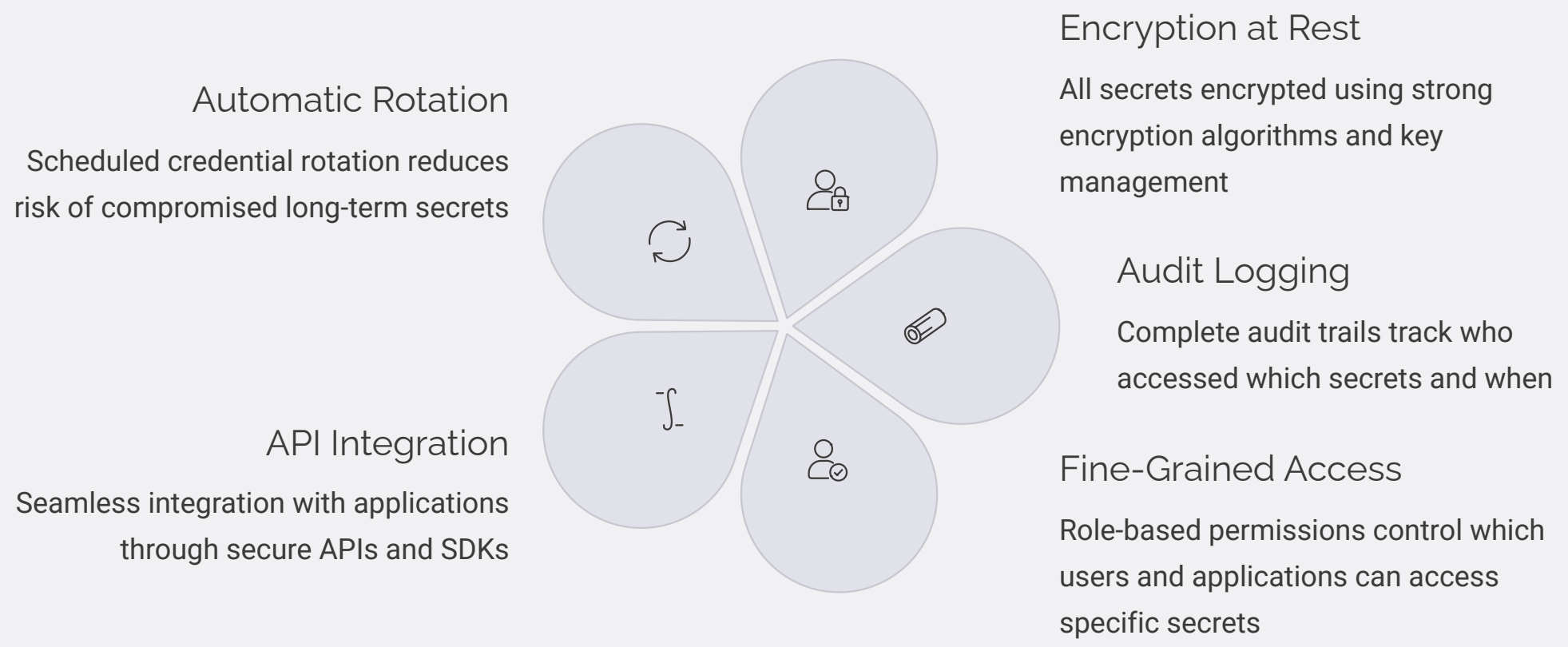
Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
Which OWASP vulnerability is caused by failing to restrict URL access properly?	Cross-Site Request Forgery (CSRF)	Broken Access Control	Injection	Security Misconfiguration	Broken Access Control

Broken Access Control represents the #1 vulnerability in the OWASP Top 10, occurring when applications fail to properly enforce restrictions on authenticated users. This vulnerability allows users to access functionality or data they shouldn't have permission to view or modify.



Question 17: Secret Manager Features

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
What is a common feature of Secret Managers like AWS Secrets Manager or HashiCorp Vault?	Automatic rotation of secrets	Public sharing of secrets	Storing secrets in plain text	No access control	Automatic rotation of secrets

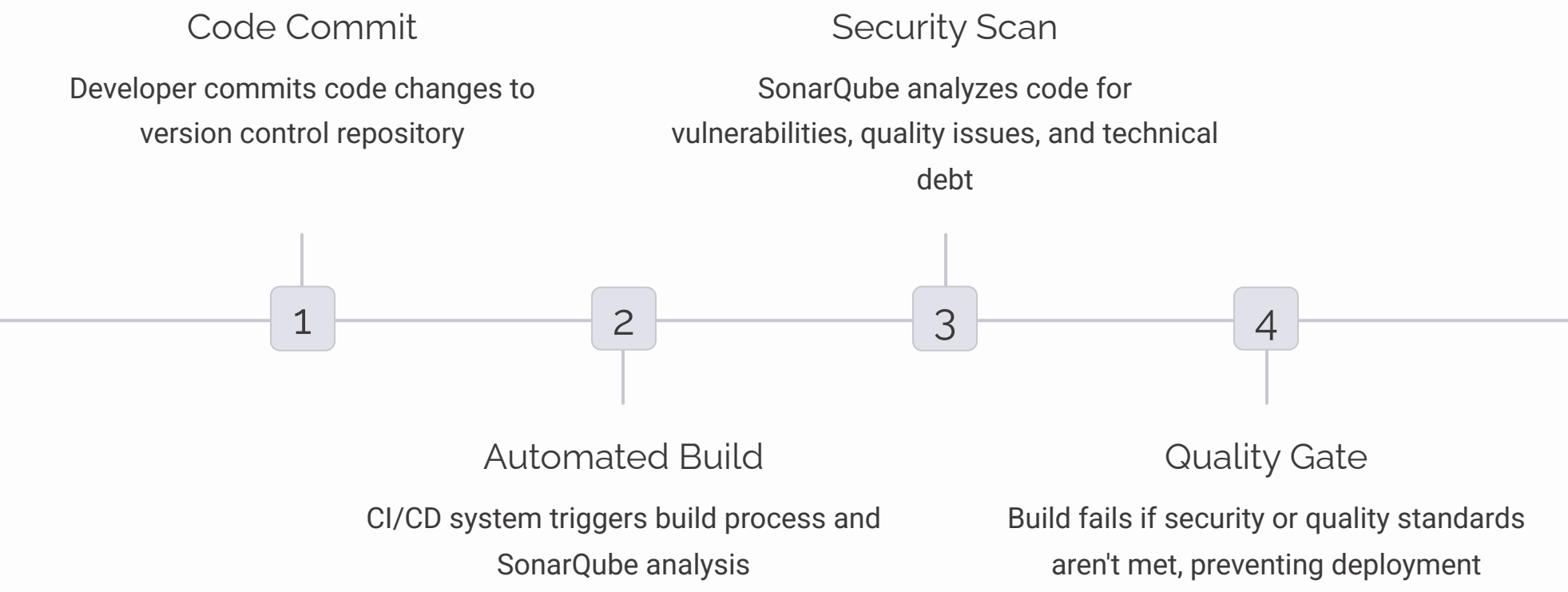


Automatic secret rotation is particularly valuable because it limits the window of opportunity for attackers who may have obtained credentials. By regularly changing passwords, API keys, and certificates, organizations reduce the risk of credential-based attacks even when secrets are occasionally compromised.

Question 18: CI/CD Security Integration

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
Scenario: You want to integrate security testing into your CI/CD pipeline. Which tool would be best suited for static code analysis?	SonarQube	OWASP ZAP	Metasploit	Nmap	SonarQube

Integrating security testing into CI/CD pipelines enables "shift-left" security practices, where security considerations move earlier in the development lifecycle. SonarQube excels in this environment because it can automatically analyze code with every commit, providing immediate feedback without requiring running applications.



Static Analysis Tools (SAST)

- **SonarQube**: Comprehensive code quality and security analysis
- **Checkmarx**: Enterprise static security testing
- **Veracode**: Cloud-based static analysis platform

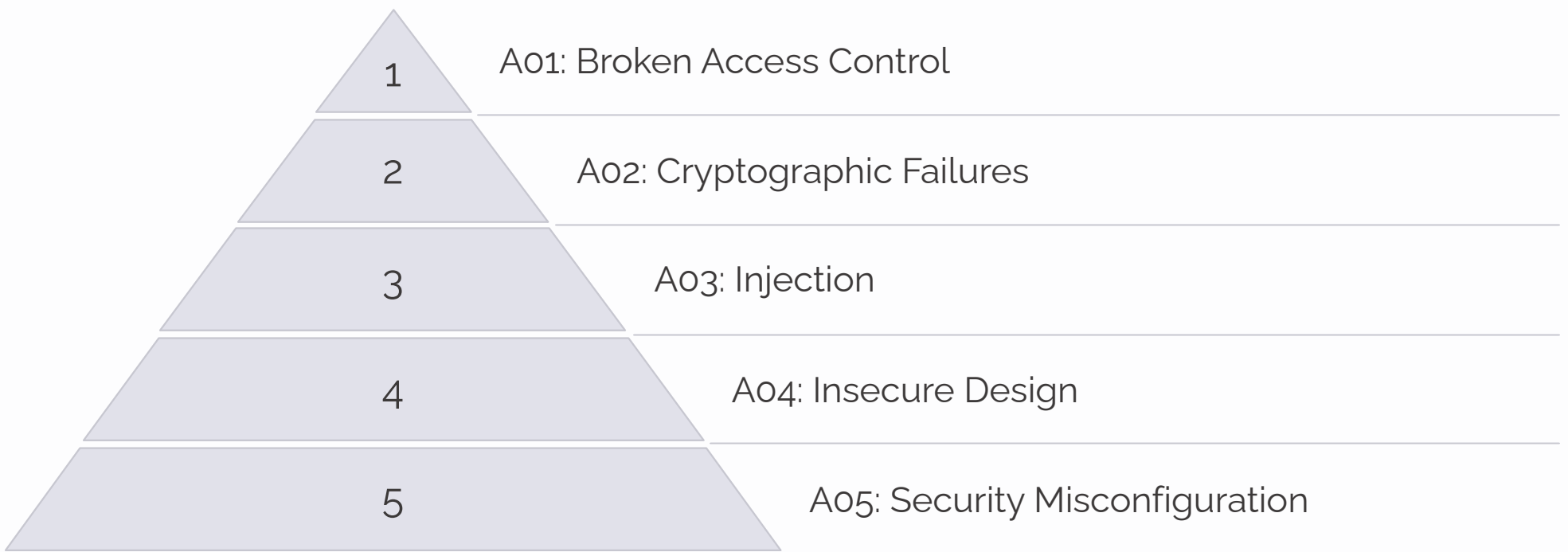
Dynamic Analysis Tools (DAST)

- **OWASP ZAP**: Web application security testing
- **Burp Suite**: Professional web vulnerability scanner
- **Nessus**: Network vulnerability assessment

Question 19: OWASP Top 10 Purpose

Question	Choice A	Choice B	Choice C	Choice D	Correct Answer
What does the OWASP Top 10 list represent?	The top 10 programming languages	The top 10 web application security risks	The top 10 software companies	The top 10 network protocols	The top 10 web application security risks

The OWASP Top 10 represents the most critical and prevalent web application security risks based on comprehensive data analysis, industry surveys, and security expert consensus. Updated every few years, this authoritative list helps organizations prioritize their security efforts on the vulnerabilities that pose the greatest real-world threats.



Understanding and addressing these top risks provides organizations with the most effective security investment, as these vulnerabilities are actively exploited by attackers and represent the majority of successful web application attacks. The list serves as a foundation for security training, secure development practices, and vulnerability assessment priorities.

✔ **Impact:** Organizations that systematically address OWASP Top 10 vulnerabilities typically reduce their web application security risk by 80-90%, making this a highly effective security framework for resource allocation and training focus.