

---

# **Security Strategies in Web Applications and Social Networking**

## **Lesson 9**

### **Mitigating Web Application Vulnerabilities**

# Learning Objective

- Describe the attributes and qualities of secure coding practices.

# Key Concepts

- Common Web application platforms and code bases
- Secure coding technologies and practices
- Weaknesses in coding platforms
- Software configuration management and revision-level tracking
- Best practices for mitigating Web application vulnerabilities

# Secure Coding

- Authentication
- Data input validation
- Session management
- Incorporate security in each phase of application development!

# Mitigating Vulnerabilities with Policies

- A well-written security policy addresses:
  - Data security in storage and in transit
  - Asset inventory and management
  - End-user security
  - Physical security
  - Access control mechanisms
  - Incident management and reporting
  - Fault-tolerant measures
  - Non-compliance consequences

<https://www.sans.org/top25-software-errors>

Insecure Interaction Between Components

Risky Resource Management  
Porous Defenses

# Mitigating Vulnerabilities with Policies (Continued)

- When developing policies, ask:
  - What assets are you trying to protect?
  - What are the common vulnerabilities and threats?
  - What are the mitigation strategies?
  - When should the security policy be reviewed and updated?

# Mitigating Vulnerabilities with Policies (Continued)

- Create:
  - Authentication policy
  - Access privilege policy
  - Disclosure of confidential data policy
  - Incident response policy/plan

# Implementing Secure Coding Best Practices

<https://www.sans.org/security-resources/policies>

Validate input

Heed compiler warnings

Plan and design for security policies

Keep the coding simple

Deny access by default



# Implementing Secure Coding Best Practices (Continued)

Layered security  
e.g. SSL, HTTPS

Use the principle of least privilege

Sanitize data sent to other systems

Use layered security

Use effective QA techniques

Adopt a secure coding standard

# HTML Security

Encrypt  
HTML code

Keep the  
code clean

Monitor  
HTML code

Use input  
validation

Validate  
URLs

Establish trust boundaries  
If things coming from untrusted src, refuse them

# JavaScript Security

Untrusted code, run in a sandbox, isolated env don't let it affect our env  
Sandbox security

Prefer to have obviously no flaws than no obvious flaws

Avoid duplication

Restrict privileges

Establish trust boundaries

Contain sensitive data

Avoid dynamic SQL

Take care of interpreting untrusted code

# CGI Form and SQL Database Access Security

CGI

Intermediary b/t client and any request to server  
Std, written in any language

"In computing, Common Gateway Interface offers a standard protocol for web servers to execute programs that execute like console applications running on a server that generates web pages dynamically. Such programs are known as CGI scripts or simply as CGIs"



Limit user access to the database

Validate input

Limit error messages

Log and audit access

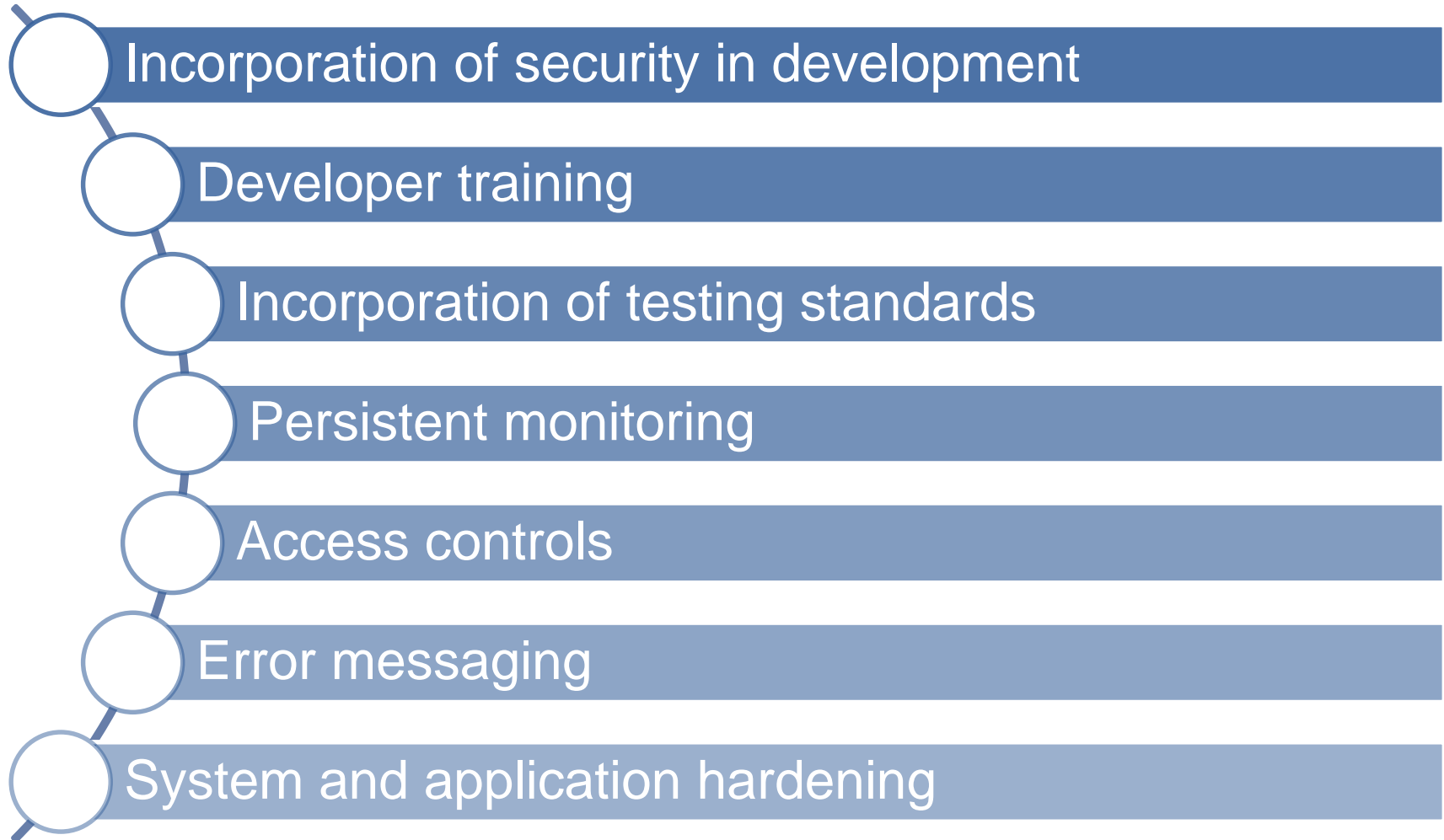
Use encryption protocols

Restrict physical access to database servers

# Revision-Level Tracking

- Prevent unauthorized changes
- Gives you greater control
- Eases management
- Helps assure quality control

# Best Practices



# Summary

- Common Web application platforms and code bases
- Secure coding technologies and practices
- Weaknesses in coding platforms
- Software configuration management and revision-level tracking
- Best practices for mitigating Web application vulnerabilities