



Bug Bounty

Checklist

GOVERDHAN

Information gathering:

Enumerate subdomains and directories

Identify web technologies used (e.g., JavaScript libraries, CMS, server software)

Look for default installations, test pages, and exposed sensitive files (e.g., backup files, Git directories, logs)

Review public resources (e.g., documentation, GitHub repositories, support forums)

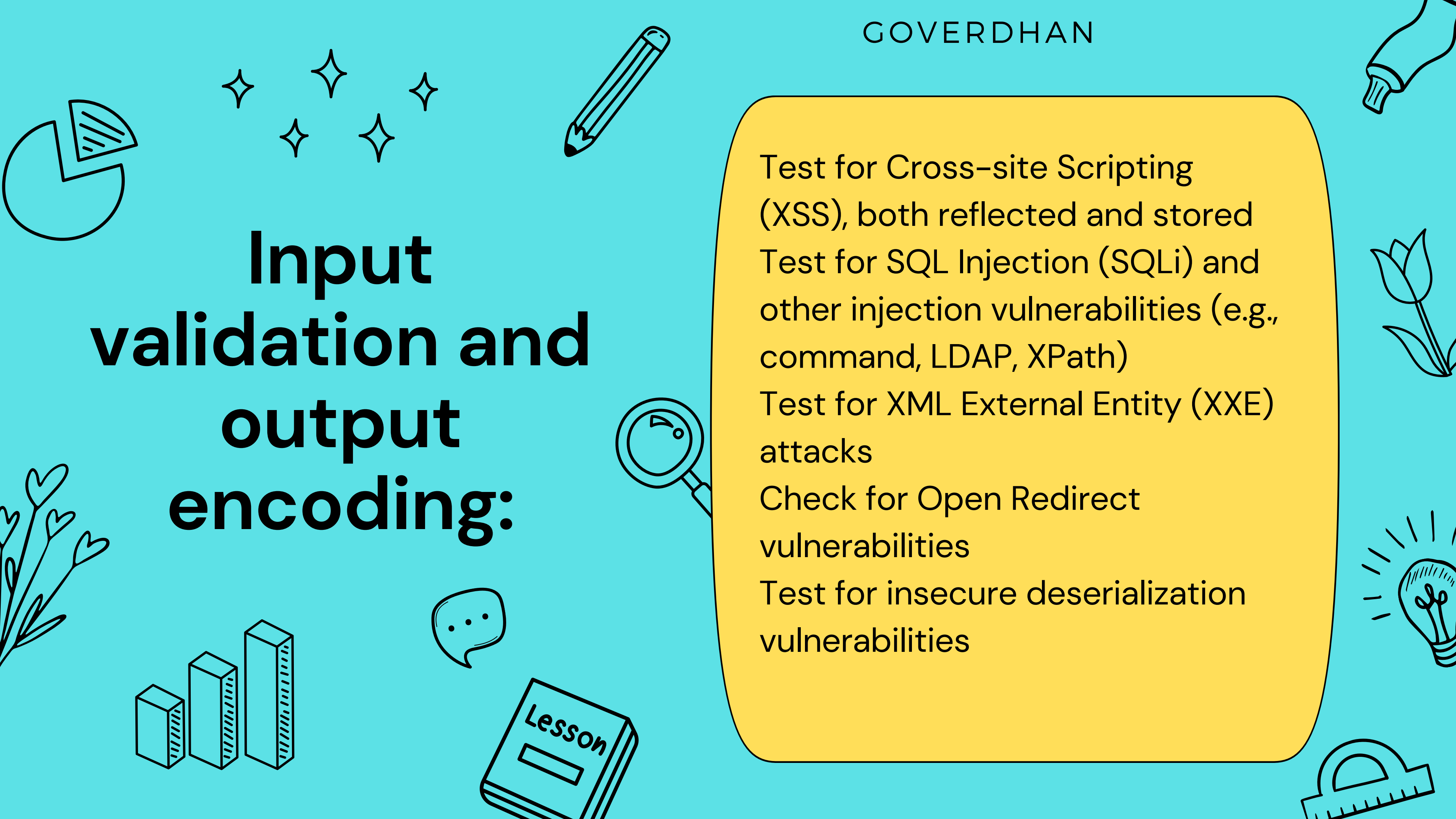
Authentication and authorization:

Test for weak credentials,
password reuse, and password
reset vulnerabilities

Check for authentication bypass
and session management issues

Test role-based access control
and vertical/horizontal privilege
escalation

Verify proper enforcement of
CORS policies



Input validation and output encoding:

Test for Cross-site Scripting (XSS), both reflected and stored
Test for SQL Injection (SQLi) and other injection vulnerabilities (e.g., command, LDAP, XPath)

Test for XML External Entity (XXE) attacks

Check for Open Redirect vulnerabilities

Test for insecure deserialization vulnerabilities

Business logic vulnerabilities:

Test for Insecure Direct Object Reference (IDOR) issues

Check for rate limiting and brute force protection

Examine multi-step processes for logic flaws (e.g., shopping carts, registration)

Test for data leakage through cache and history poisoning

Server-side vulnerabilities:

Test for server-side vulnerabilities like remote code execution (RCE), local file inclusion (LFI), and remote file inclusion (RFI)

Test for Server-Side Request Forgery (SSRF) vulnerabilities

Check for server misconfigurations, including weak TLS/SSL cipher suites, insecure HTTP headers, and exposed error messages

Test for path traversal vulnerabilities

Client-side vulnerabilities:

Test for Cross-Site Request Forgery (CSRF) vulnerabilities
Examine JavaScript code for vulnerabilities (e.g., client-side validation, DOM-based XSS)
Check for Clickjacking and other UI redressing attacks
Test for HTML5 postMessage vulnerabilities

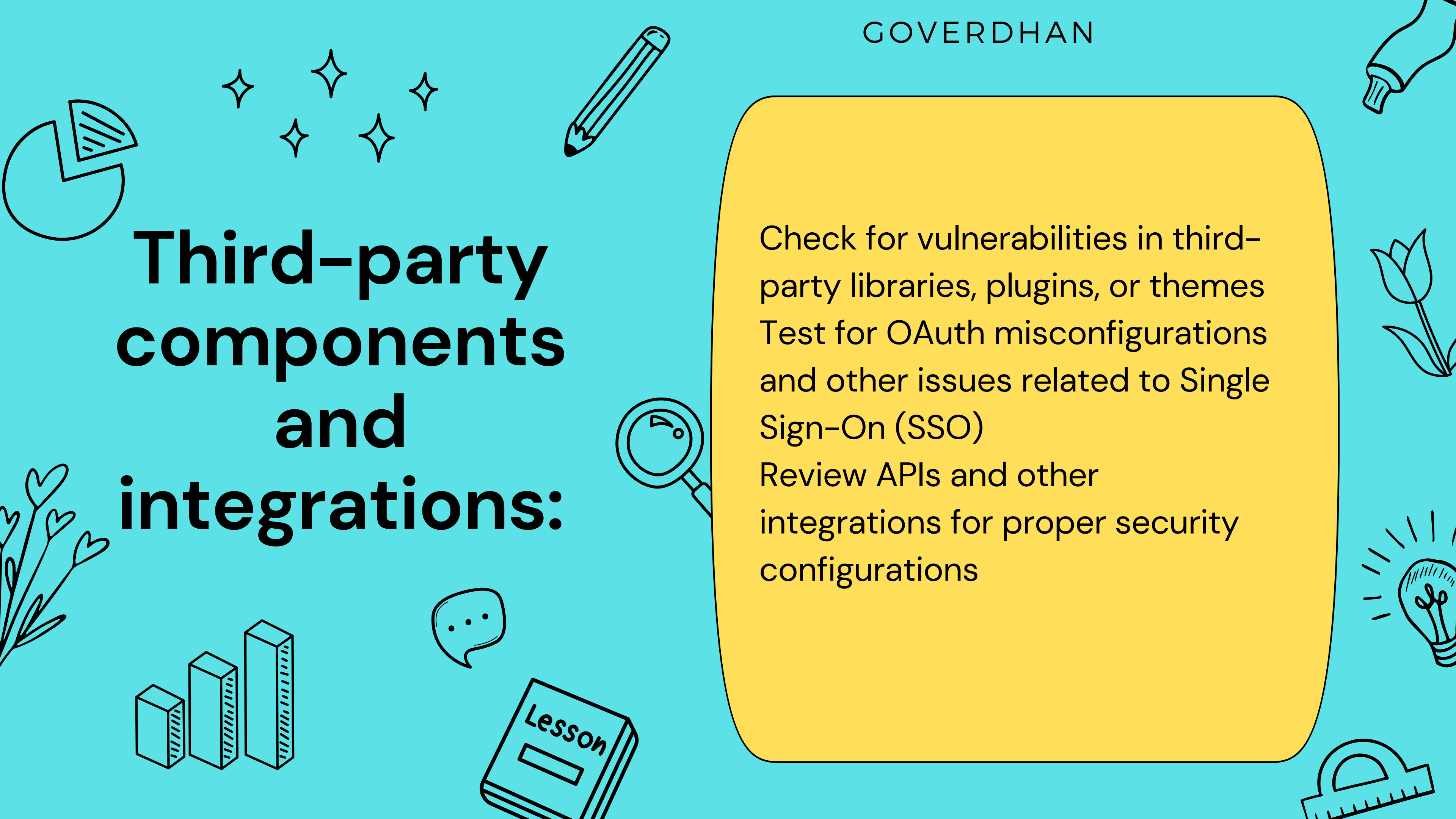
Web APIs:

Enumerate API endpoints and methods

Test for authentication, authorization, and rate limiting issues

Check for parameter tampering and injection vulnerabilities

Test for improper handling of data formats (e.g., JSON, XML)



Third-party components and integrations:

Check for vulnerabilities in third-party libraries, plugins, or themes
Test for OAuth misconfigurations and other issues related to Single Sign-On (SSO)

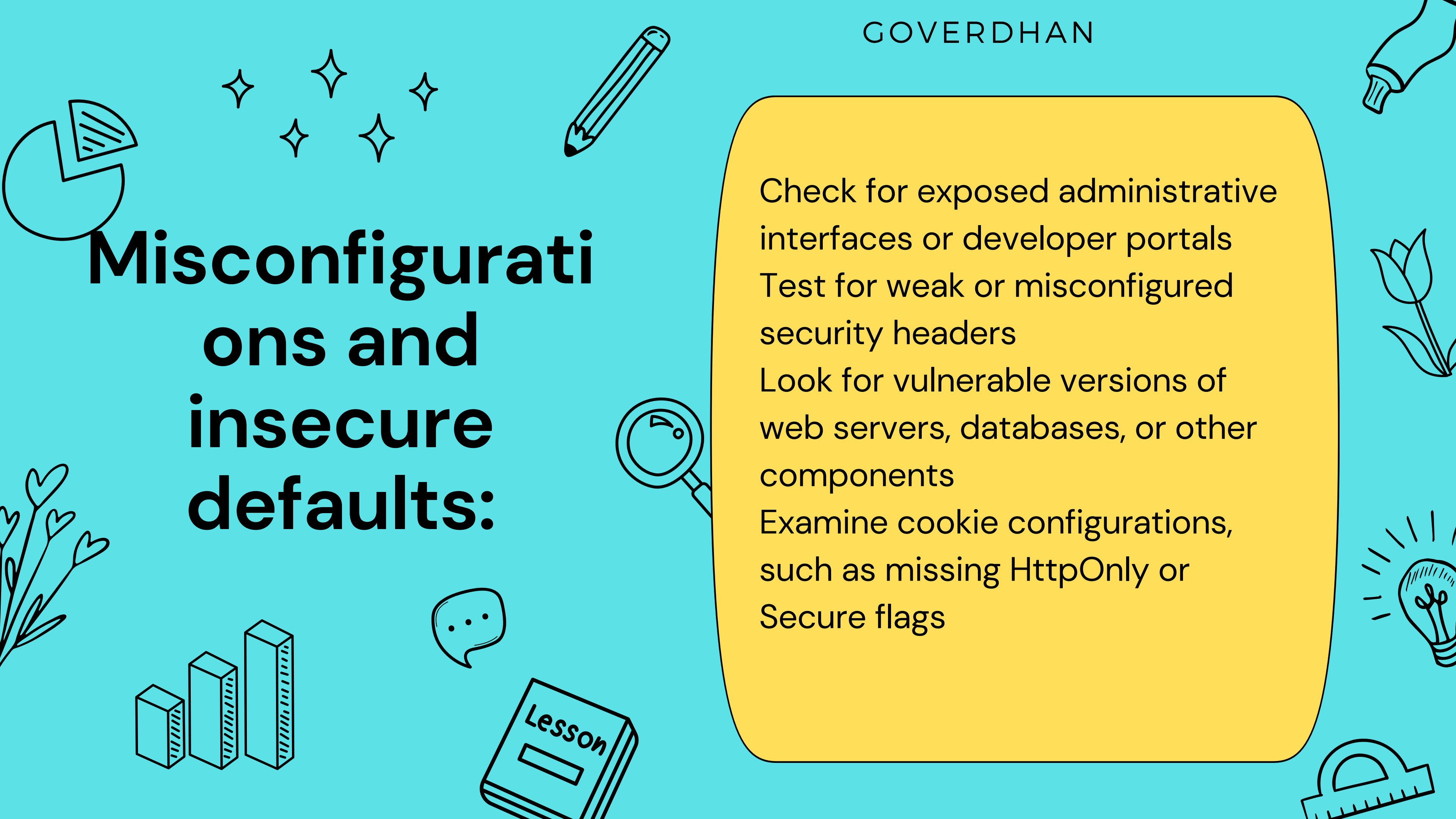
Review APIs and other integrations for proper security configurations

Mobile application components:

Test for insecure communication between mobile apps and web servers

Check for mobile-specific issues such as insecure storage of sensitive data

Assess mobile app-specific vulnerabilities, like deep linking or WebView issues



Misconfigurations and insecure defaults:

Check for exposed administrative interfaces or developer portals
Test for weak or misconfigured security headers
Look for vulnerable versions of web servers, databases, or other components
Examine cookie configurations, such as missing HttpOnly or Secure flags

Cryptography- related vulnerabilities:

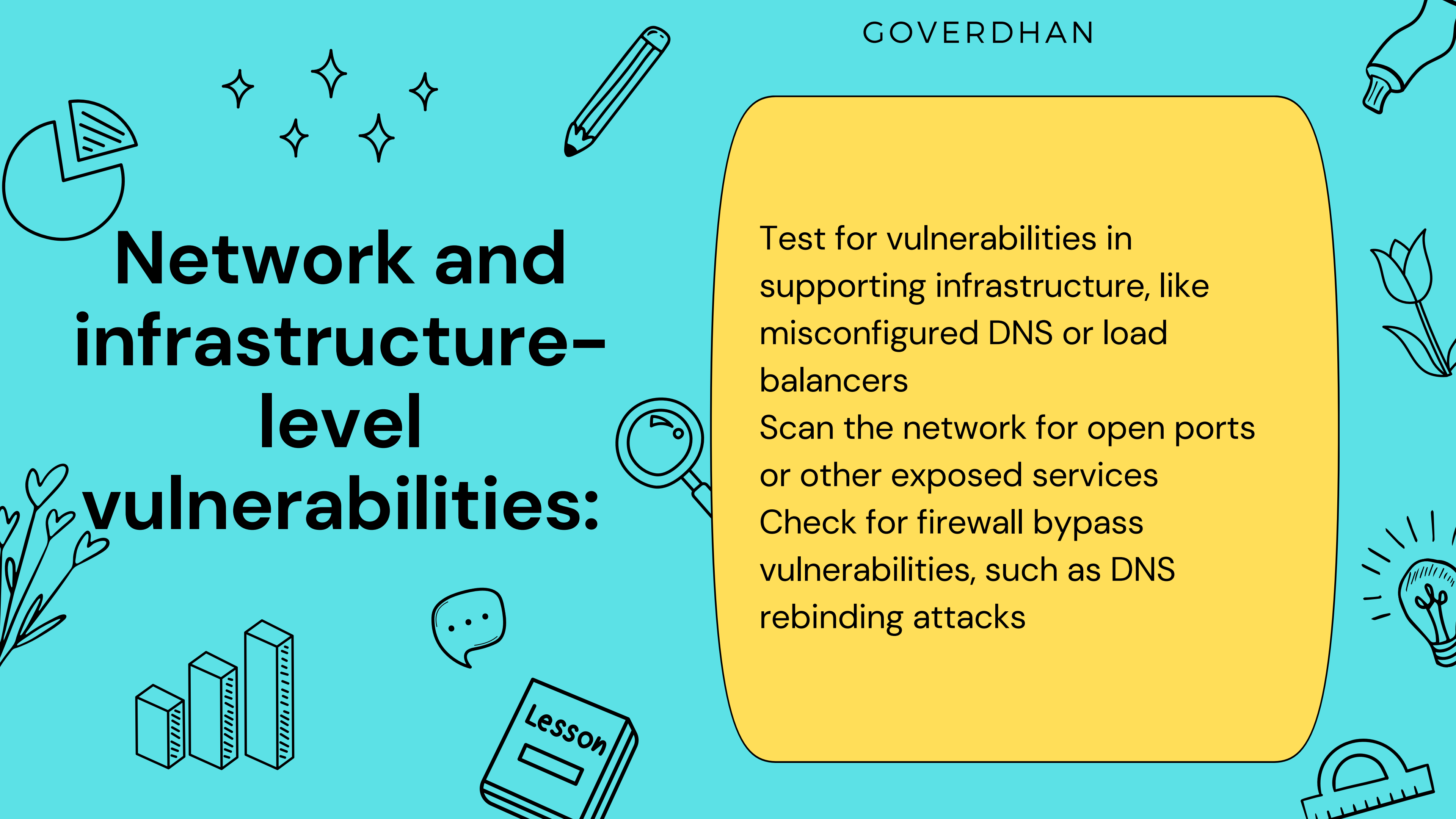
Test for weak encryption or hashing algorithms
Check for improper usage of encryption libraries
Assess the application for issues like padding oracle attacks
Look for hardcoded keys or secrets in the application's code

Data exposure and privacy issues:

Check for sensitive data leakage in HTTP responses, logs, or error messages

Test for unprotected access to user profile data or personally identifiable information (PII)

Verify the proper handling of user consent and privacy-related features

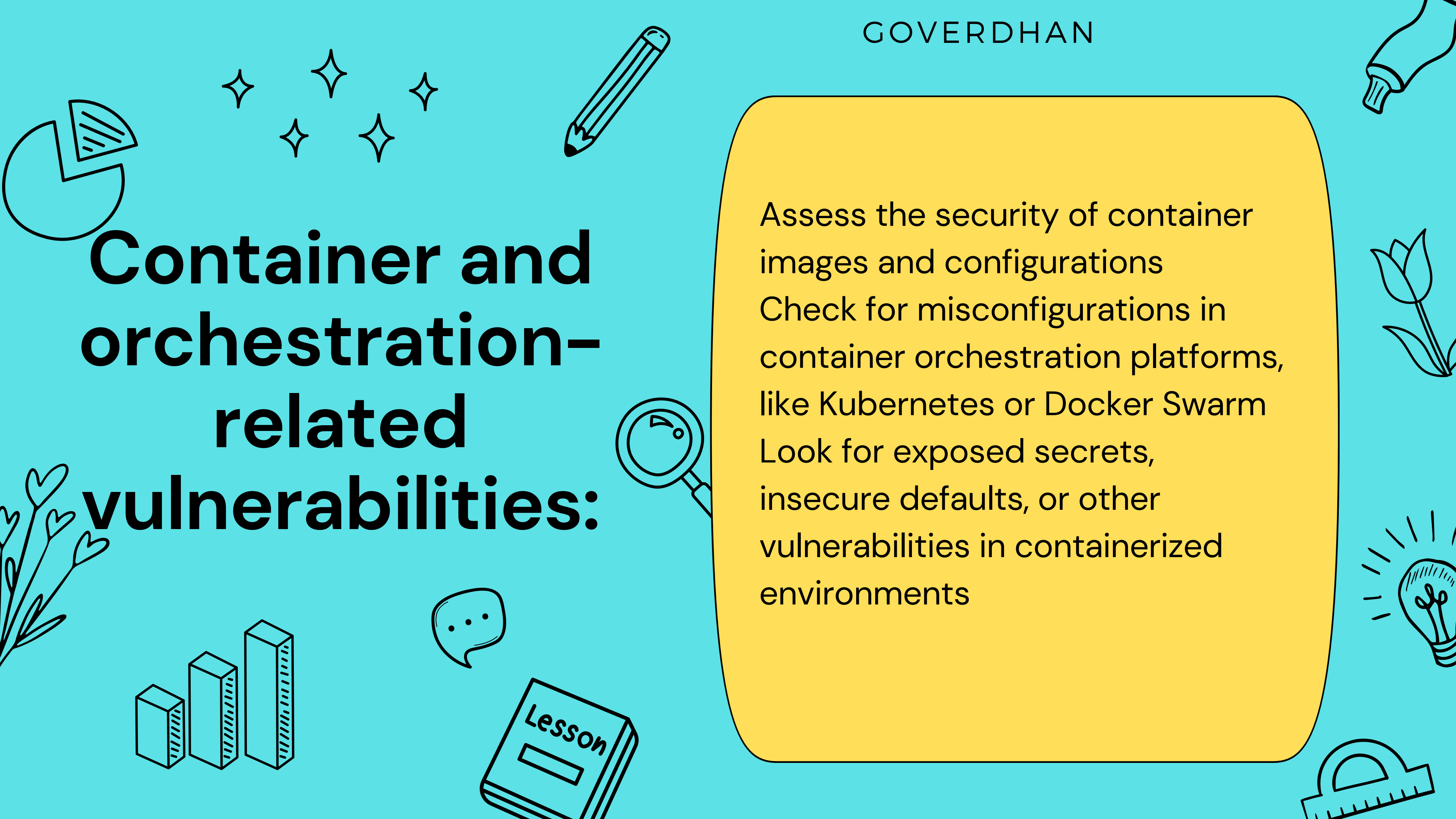


Network and infrastructure-level vulnerabilities:

Test for vulnerabilities in supporting infrastructure, like misconfigured DNS or load balancers

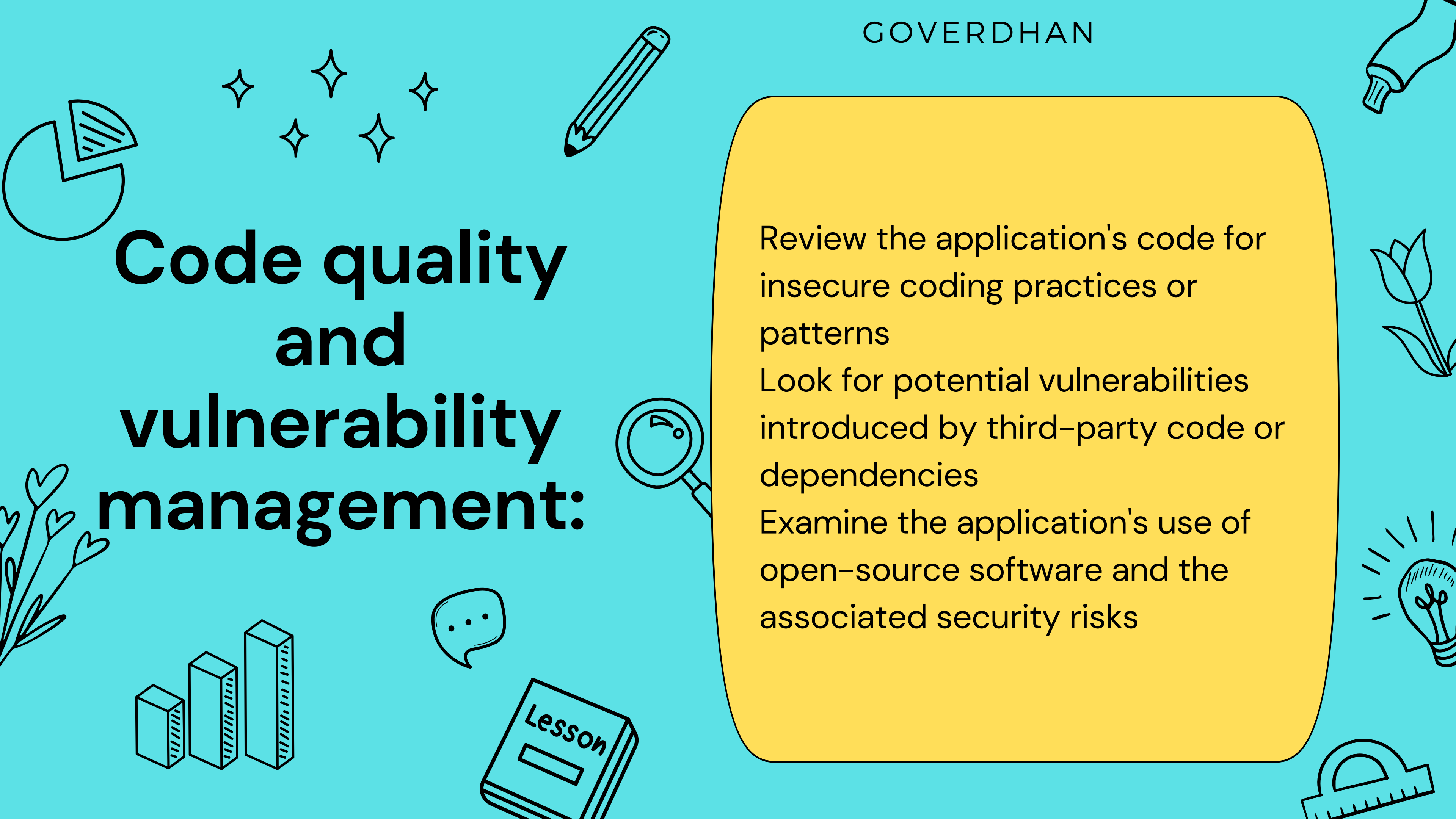
Scan the network for open ports or other exposed services

Check for firewall bypass vulnerabilities, such as DNS rebinding attacks



Container and orchestration-related vulnerabilities:

Assess the security of container images and configurations
Check for misconfigurations in container orchestration platforms, like Kubernetes or Docker Swarm
Look for exposed secrets, insecure defaults, or other vulnerabilities in containerized environments



Code quality and vulnerability management:

Review the application's code for insecure coding practices or patterns

Look for potential vulnerabilities introduced by third-party code or dependencies

Examine the application's use of open-source software and the associated security risks

Thank You

Follow for more



P_Goverdhan



goverdhankumar



pan_goverdhan