Answer Key

VIT® Vellore Institute of Technology (Deemed to be University under section 3 of UGC Act, 1956)

REG.NO.:
SLOT: B1+TB1

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
FALL SEMESTER 2025-2026

Programme Name & Branch : **B.Tech CSE (Core & Specialization)**
Course Code and Course Name : **BCSE320L – Web Application Security**
Faculty Name(s) : **Dr. Bhulakshmi Bonthu, Dr. Mohankumar B, Dr. Samriddhi Sarkar**
Class Number(s) : **VL2025260101452, 1456, 1457**
Date of Examination : **18.08.2025**
Exam Duration : **90 minutes** **Maximum Marks: 50**

General instruction(s):

- Answer All Questions
- M - Max mark; CO – Course Outcome; BL – Blooms Taxonomy Level
- Course Outcomes
  - CO1: Understand security challenges and the need for Authentication and Authorization in web based systems and applications.
  - CO2: Familiarize the Application Programming Interface analysis and vulnerability management of securing a web-based system.

Q. No	Question	M	CO	BL
1.	Explain the purpose of authentication and authorization systems in a large retail website that uses third-party services. How can these systems become vulnerable to unauthorized access? **Answer:** What authentication and authorization do ***(4 Marks)*** • **Authentication:** Proves who the user or system is. On a retail site this covers shoppers, admins, customer-support agents, mobile apps, and service accounts from payment, shipping, search, analytics, and marketing partners. • **Authorization:** Decides what that authenticated party can do. Examples: shoppers can view their orders but not others'; support agents can issue refunds; payment gateways can charge but not read support notes. Why this matters in retail • Protects PII, cardholder data, order history, loyalty points, and coupons. • Prevents fraud (account takeovers, refund abuse, gift-card drains). • Separates duties for internal teams and third parties. **How these systems become vulnerable *(4 Marks)*** **Authentication pitfalls** • Weak or reused credentials • Credential-stuffing exposure • Session flaws • JWT issues • OAuth/OpenID Connect mistakes • Password reset/2FA logic bugs **Authorization pitfalls**	10	CO1	BL2

	• Broken Object-Level Authorization • Broken function-level authorization • Over-permissive scopes/roles • Inconsistent checks across duplicate endpoints			
2.	Analyze how the adoption of REST APIs and Single Page Applications (SPAs) has changed the structure of modern web applications compared to legacy architectures. Illustrate your answer with examples, and evaluate how these changes influence an attacker's web reconnaissance techniques. **Answer:** **Working (3 Marks):** A request is sent from client to server in the form of a web URL as HTTP GET or POST or PUT or DELETE request. After that, a response comes back from the server in the form of a resource which can be anything like HTML, XML, Image, or JSON. But now JSON is the most popular format being used in Web Services.  REST stands for Representational State Transfer, which is a fancy way of defining an API that has a few unique traits: • It must be separate from the client • It must be stateless • It must be easily cacheable • Each endpoint should define a specific object or method **Legacy (server-rendered) (2 Marks)** • Pages built on the server per request: /cart.php?action=add&id=123. • State stored server side via session cookies. • Navigation maps closely to server routes and directories. • Small number of JSON endpoints if any; most data arrives inside HTML. **Modern (SPA front end + REST API back end) (3 Marks)** • SPA serves static assets from CDN or app server: index.html, JS bundles, CSS. • Browser renders views and calls a **stateless REST API**: /api/v1/orders, /api/v1/users/42. • Authentication uses tokens (often JWT) and fine-grained scopes. • Cross origin is common: app.example.com talks to api.example.com via CORS. • Services split by function with versioned routes: /api/v1, /api/v2.	10	CO1	BL4

	Example (2 Marks) Legacy checkout: POST /checkout.php returns an HTML receipt and sets a PHP session. SPA + REST checkout: SPA calls POST https://api.example.com/v1/orders, receives JSON, updates the view client side. Session is a bearer token in the Authorization header.			
3.	Explain how brute-forcing subdomains and checking search engine caches can reveal unintended security issues in a web application that uses multiple subdomains for different business functions. Use examples to support your explanation. **Answer:** **Why multiple subdomains matter (2 Marks)** Modern web applications often separate business functions into different subdomains: • shop.example.com for e-commerce • blog.example.com for marketing • admin.example.com for staff dashboards • api.example.com for REST services • dev.example.com or staging.example.com for testing Each subdomain may run on a different stack, have unique authentication, and sometimes be less secure than the primary domain. Attackers target these weaker links to pivot into critical systems. **Brute-forcing subdomains (5 Marks)** Attackers use automated tools (like **Sublist3r, Amass, Gobuster**) to guess or enumerate hidden subdomains. **Example 1: Forgotten staging site** • Brute-forcing reveals staging.example.com. • This staging site has debug mode enabled and default credentials (admin/admin). • The attacker exploits it to learn internal API endpoints, then reuses them on the production system. **Example 2: Misconfigured admin portal** • Tool finds admin.example.com. • The login page is not linked anywhere on the main site. • Weak or missing access controls let an attacker attempt brute-force login or exploit an outdated framework. This shows how **security-by-obscurity** (hiding subdomains) often fails once brute-forcing is applied. **Search engine caches as reconnaissance tools (3 Marks)** Search engines like Google or Bing often index forgotten or unlinked resources. Attackers can use operators like site:example.com or cached pages to discover exposed content. **Example 1: Cached sensitive documents** • Google cache shows an old PDF at files.example.com/employee_salaries.pdf even after it was removed.	10	CO2	BL2

	• Although the live link returns 404, the cached copy is still accessible, leaking confidential data. **Example 2: Exposed API docs** • docs.example.com/swagger.json was indexed. • Cached version reveals **all available API endpoints**, even if access was later restricted. • Attackers use this map for targeted exploitation. **Example 3: Old admin pages** • admin.example.com/backup/login.php is no longer linked but appears in Bing's cache. • The cached content shows the login structure and parameter names, useful for password spraying or SQL injection.			
4.	Describe the concept of endpoint discovery in web APIs and explain how attackers might use automated tools to uncover hidden or undocumented endpoints in a web application. **Answer:** **Concept of Endpoint Discovery: (5 Marks)** Endpoint discovery in web APIs is the process of identifying all available API endpoints that a web application exposes. Each endpoint represents a specific resource or functionality (e.g., /users, /login, /orders). While some endpoints are documented and publicly accessible for developers, others may be hidden, legacy, or undocumented, yet still remain active on the server. Discovering these endpoints is critical because undocumented ones can expose sensitive functionality, unvalidated inputs, or administrative operations that attackers can exploit **Use of Automated Tools by Attackers: (5 Marks)** Attackers often rely on automated tools and scripts to uncover hidden endpoints. These tools perform actions such as: • **Brute-Force Enumeration:** Sending a large number of requests with common endpoint names (like /admin, /config, /backup) from pre-built wordlists. • **Fuzzing:** Injecting random or structured inputs to see how the server responds, which can reveal valid but undocumented endpoints. • **Traffic Inspection:** Monitoring API calls made by web or mobile applications through network analysis tools (like Burp Suite or Postman) to capture all requests. • **Search Engine Dorking:** Using cached or indexed API endpoints exposed unintentionally to the public. **Example:** Suppose a retail website has documented endpoints like /products and /checkout. Using automated discovery, an attacker might also find an undocumented endpoint /getAllUsers that exposes sensitive customer	10	CO2	BL2

	data without authentication. This would significantly increase the risk of data breaches.			
5.	A healthcare web application integrates multiple technologies at different layers of the application stack. Discuss the importance of detecting both client-side and server-side frameworks in building a comprehensive security strategy. **Answer:** **Importance of Detecting Client-Side and Server-Side Frameworks: (3 Marks)** A healthcare web application typically handles highly sensitive patient information, which makes security a top priority. Since such applications integrate multiple technologies at different layers, understanding and detecting both client-side and server-side frameworks is crucial for developing a comprehensive security strategy. **1. Client-Side Frameworks: (3 Marks)** Client-side technologies (like Angular, React, or Vue.js) manage user interactions, rendering, and data exchange with the backend. Detecting these frameworks is important because: • It helps identify **framework-specific vulnerabilities**, such as DOM-based XSS in Angular or React. • It enables proper configuration of **Content Security Policies (CSPs)** to prevent script injection attacks. • It allows security teams to evaluate how data is validated and sanitized before being sent to the server. **2. Server-Side Frameworks: (3 Marks)** Server-side frameworks (such as Django, Express.js, Spring, or ASP.NET) handle core business logic, authentication, and database interactions. Detecting these frameworks is vital because: • It helps locate **misconfigurations** (e.g., default admin endpoints, error pages) that could be exploited. • It allows for the identification of **injection risks** (SQLi, XXE) depending on how the framework processes user input. • It provides insights into **session management** and **authentication mechanisms**, which are critical in healthcare systems. **3. Integrated Security Strategy: (1 Mark)** By detecting both client-side and server-side frameworks, security teams can: • Map the **attack surface more accurately**. • Apply **framework-specific patches and updates** quickly. • Build **layered defenses** that address both front-end and back-end risks, ensuring end-to-end protection of patient data.	10	CO2	BL4

****