



Systematic Vulnerability Assessment and Security Analysis of Web Applications using Burp Suite

Tanishk Kaushik¹

¹ Scholar, Integrated B.Tech.-M.Tech.(Computer Science Engineering), School of Information, Communication and Technology (SOICT), Gautam Buddha University, Opposite Yamuna Expressway Greater Noida, Gautam Budh Nagar, Uttar Pradesh – 201312, INDIA.

Article Info

Article History:

Published: 28 Sept 2026

Publication Issue:

Volume 3, Issue 9
September-2026

Page Number:

537-550

Corresponding Author:

Tanishk Kaushik

Abstract:

Cybersecurity has become one of the most significant concerns in modern computing due to the exponential rise in sophisticated web-based attacks. Web applications routinely process sensitive user information, financial data, and classified business records. Any vulnerability within these applications can lead to unauthorized access, catastrophic data theft, and severe organizational compromise. This comprehensive study focuses on identifying, analyzing, and validating critical web application vulnerabilities using Burp Suite Community Edition. It bridges theoretical cybersecurity concepts with practical, real-world penetration testing methodologies. The practical implementation involves a deep-dive security assessment of a live enterprise portal. The assessment successfully uncovered critical and high-severity flaws, including Unauthenticated API Access, Authentication Bypasses via response manipulation, Insecure Direct Object References (IDOR), Server-Side Request Forgery (SSRF), and Cross-Site Scripting (XSS). By leveraging manual testing techniques alongside Burp Suite's powerful proxy and fuzzing capabilities, this study highlights the critical need for secure coding practices, strict server-side authorization, and continuous manual penetration testing to supplement automated security tools in the Software Development Life Cycle (SDLC).

Keywords: Comprehensive Study, Web Application, Vulnerability, Burp Suite, Assessment, Impact analysis

1. Introduction

1.1 Background of the Study

The rapid growth of the Internet has fundamentally transformed the way organizations, businesses, educational institutions, and governments deliver services. Most modern applications are now web-based, allowing users to access services such as online banking, e-commerce, healthcare, and cloud computing from anywhere in the world. While this digital transformation has improved accessibility and efficiency, it has dramatically increased the attack surface available to cybercriminals [1-7].

According to industry security reports, web application attacks remain among the leading causes of data breaches worldwide. Many successful attacks occur because organizations fail to identify vulnerabilities during software development or before deploying applications into production environments. Web application vulnerabilities often arise due to insecure coding practices, improper authentication mechanisms, weak authorization controls, insufficient input validation, and server misconfigurations.

1.2 Problem Statement

Web applications are increasingly targeted because they frequently process confidential information and are accessible continuously over the Internet. Many applications contain deep-seated security weaknesses resulting from outdated software components, weak authentication mechanisms, and a heavy reliance on client-side controls. Without regular and rigorous security assessments, these vulnerabilities remain undetected until actively exploited by threat actors, leading to unauthorized access, identity theft, data leakage, and service disruption [8-13].

Therefore, there is a critical need for an effective methodology to identify and analyze web application vulnerabilities before deployment. Automated tools alone are insufficient for identifying complex business logic flaws. This necessitates the use of advanced intercepting proxies to perform manual and semi-automated vulnerability assessments.

1.3 Aim and Objectives

The primary aim of this project is to perform a comprehensive vulnerability analysis of web applications using Burp Suite and identify potential security weaknesses that could be exploited by attackers. The specific objectives are:

- 1.3.1 To study the fundamental concepts of cybersecurity, the CIA triad, and web application security architectures.
- 1.3.2 To understand the HTTP and HTTPS communication protocols and how they can be manipulated.
- 1.3.3 To extensively explore the architecture and working principles of Burp Suite and its various modules.
- 1.3.4 To configure Burp Suite as an intercepting proxy in a customized Linux environment.
- 1.3.5 To perform practical, manual security testing on a live enterprise application to identify OWASP Top 10 vulnerabilities.
- 1.3.6 To analyze the impact and severity of discovered vulnerabilities and recommend robust, industry-standard mitigation strategies.

1.4 Scope and Limitations

The scope of this project is focused on the vulnerability assessment of web applications using Burp Suite in an authorized testing environment. It covers configuration, traffic interception, analysis of HTTP requests/responses, and manual testing for common vulnerabilities (SQLi, XSS, SSRF, IDOR, Broken Authentication). The project does not include unauthorized penetration testing of un-consenting real-world systems, denial-of-service attacks intended to take down infrastructure, or malware development.

2. Literature review

2.1 The Cybersecurity Paradigm

Cybersecurity refers to the practice of protecting computer systems, networks, software, and digital information from unauthorized access, cyberattacks, and data breaches. It involves implementing technologies, processes, and security policies that ensure the confidentiality, integrity, and availability (CIA) of information [8-12].

2.2 The CIA Triad

The CIA Triad is the fundamental principle of information security:

- **Confidentiality:** Ensures that sensitive information is accessible only to authorized users. Security mechanisms such as encryption, authentication, and access control help maintain confidentiality.
- **Integrity:** Ensures that data remains accurate, complete, and unaltered during storage or transmission. Hashing algorithms and database constraints maintain integrity.
- **Availability:** Ensures that systems and data remain accessible whenever authorized users require them, utilizing redundant servers and disaster recovery plans.

2.3 Web Application Security Threats

Modern web applications face numerous security threats. Understanding these attack vectors is essential for effective vulnerability assessment [3,4,6,8].

2.3.1 Injection Flaws: Occur when untrusted data is sent to an interpreter as part of a command or query. SQL Injection (SQLi) allows attackers to modify database queries, potentially leading to complete database compromise.

2.3.2 Cross-Site Scripting (XSS): Occurs when an application includes untrusted data in a web page without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser, enabling session hijacking and website defacement.

2.3.3 Broken Access Control: Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality or view sensitive files, often manifesting as Insecure Direct Object References (IDOR).

2.3.4 Server-Side Request Forgery (SSRF): Occurs when 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.

2.4 Vulnerability Assessment vs. Penetration Testing

A Vulnerability Assessment (VA) is the systematic process of identifying, classifying, and prioritizing security weaknesses in a system. It is primarily detection-focused. Penetration Testing (PT), on the other hand, is a controlled and authorized attempt to exploit vulnerabilities to determine their real-world impact. This project encompasses elements of both, identifying flaws and manually validating their exploitability [14-16].

2.5 Referenced Standards

This study aligns its methodology and findings with globally recognized cybersecurity standards:

2.5.1 OWASP Top 10 (2021): The Open Worldwide Application Security Project's consensus on the most critical web application security risks [17].

2.5.2 NIST Cybersecurity Framework (CSF 2.0): The newly updated policy framework providing computer security guidance for organizations to assess and improve their ability to prevent, detect, and respond to cyber-attacks [18-21].

2.5.3 CWE (Common Weakness Enumeration): A catalog of common software and hardware security weaknesses used to classify the root causes of vulnerabilities [22-23].

3. In-Depth study of Burp suite

3.1 Basic of Burp Suite

Burp Suite, developed by PortSwigger, is an integrated platform for testing the security of web applications. It is widely considered the industry standard tool for penetration testers, security researchers, and bug bounty hunters. Burp Suite operates as an intercepting proxy, positioning itself in the middle of the network stream between the user's web browser and the target web server. This Man-in-the-Middle (MitM) position allows security analysts to capture, inspect, modify, and replay HTTP and HTTPS traffic in real-time [24-26].

3.2 Core Modules and Architecture

3.2.1 Burp Proxy

The Proxy module is the heart of Burp Suite. It operates as a local web proxy server (typically binding to 127.0.0.1:8080). By configuring the web browser to route all traffic through this proxy, every request and response is routed through Burp Suite. To intercept encrypted HTTPS traffic, Burp Suite dynamically generates a unique CA (Certificate Authority) certificate. Once this CA certificate is installed and trusted in the browser, Burp can seamlessly decrypt, inspect, and re-encrypt SSL/TLS traffic without triggering browser security warnings. The Proxy's "Intercept" feature allows testers to pause a request, manually edit headers, cookies, or POST bodies, and then forward the manipulated request to the server [25-26].

3.2.2 Target and Site Map

The Target module is utilized for application mapping and reconnaissance. As traffic flows through the proxy, Burp Suite passively builds a hierarchical Site Map of the target application. This tree structure displays discovered directories, files, API endpoints, and parameters. The Site Map is crucial for defining the "Scope" of an assessment, ensuring that testing tools only interact with authorized domains and ignoring out-of-scope background noise (like analytics or third-party APIs) [26].

3.2.3 Burp Repeater

Repeater is an indispensable tool for manual vulnerability verification. Analysts can send an interesting HTTP request from the Proxy or Site Map directly to Repeater. Within Repeater, the request can be modified and reissued repeatedly without needing to interact with the browser. This iterative testing process is essential for refining exploit payloads (such as tweaking an SQL injection string or adjusting an XSS payload to bypass a Web Application Firewall) and instantly analyzing the server's response [24].

3.2.4 Burp Intruder

Intruder is a highly customizable automation tool used to perform customized brute-force and fuzzing attacks. Analysts define payload positions within an HTTP request (e.g., surrounding a username or an ID parameter) and supply a list of payloads. Intruder supports four primary attack types [26]:

- 3.2.4.1 **Sniper:** Uses a single set of payloads. It iterates through the payload positions one by one, inserting the payload into one position at a time while leaving others at their original values. Ideal for fuzzing parameters for XSS or SQLi.
- 3.2.4.2 **Battering Ram:** Uses a single set of payloads. It places the same payload into all defined payload positions simultaneously. Useful when the same value needs to be injected in multiple places (e.g., matching a username and a customized header).
- 3.2.4.3 **Pitchfork:** Uses multiple payloads sets (one for each position). It iterates through the sets simultaneously (Payload 1 from Set 1 goes into Position 1, Payload 1 from Set 2 goes into Position 2). Perfect for testing pairs of data, like usernames and passwords for credential stuffing.
- 3.2.4.4 **Cluster Bomb:** Uses multiple payload sets and iterates through all possible permutations. If Set A has 10 payloads and Set B has 10 payloads, Cluster Bomb will generate 100 requests. It is the most thorough but time-consuming attack type.

3.2.5 Decoder and Comparer

Decoder: Provides manual and smart decoding/encoding of data. Web applications frequently encode data (URL encoding, Base64, HTML entity encoding, Hex) during transmission. Decoder allows analysts to quickly translate these formats to understand the underlying data or to encode payloads to bypass basic input filters.

Comparer: A visual diff tool. It is used to compare two distinct HTTP requests or responses side-by-side to highlight minute differences. This is particularly useful when analyzing subtle changes in server responses during authorization testing or blind SQL injection.

3.2.6 Sequencer

Sequencer is an advanced tool designed to evaluate the quality of randomness in application tokens. Session identifiers, anti-CSRF tokens, and password reset links must be cryptographically secure and unpredictable. Sequencer collects thousands of tokens from the live application and subjects them to rigorous statistical tests (character-level and bit-level entropy analysis) to determine if the tokens can be predicted or brute-forced [26].

4. Research Methodology

4.1 Hardware and Software Environment Setup

A specialized, controlled testing environment was established to conduct the vulnerability assessment reliably. The hardware utilized was an Asus TUF laptop system, providing adequate computational resources for handling large proxy state files and multi-threaded fuzzing operations [25-26].

The operating system environment was built upon Arch Linux, chosen for its minimalistic nature and deep system-level control. The graphical environment utilized the Niri Wayland compositor, providing an efficient, keyboard-centric tiling interface that optimized workflow across multiple terminal windows, code editors, and the Burp Suite interface. Mozilla Firefox was configured as the dedicated testing browser, equipped with proxy extensions (like FoxyProxy) to seamlessly toggle traffic routing to the Burp Suite listener (127.0.0.1:8080) [24].

4.2 Assessment Workflow Phases

The practical assessment was executed following a rigorous, phase-based methodology [17, 26]:

- 4.2.1 **Reconnaissance and Configuration:** The target application's scope was explicitly defined within Burp Suite. Firefox was configured to route traffic, and the Burp CA certificate was installed to facilitate HTTPS interception.
- 4.2.2 **Application Mapping (Spidering):** The application was manually navigated comprehensively. Every link was clicked, forms were submitted with dummy data, and all user roles were explored. This allowed Burp Suite's Target module to build an exhaustive Site Map.
- 4.2.3 **Traffic Interception and Analysis:** The Proxy module was utilized to pause and inspect the structure of HTTP requests. Special attention was paid to session cookies, JSON Web Tokens (JWTs), hidden form fields, API endpoints, and non-standard HTTP headers.
- 4.2.4 **Vulnerability Identification:** Based on the traffic analysis, specific endpoints were targeted for common vulnerabilities. Input fields were tested for injection flaws. URL parameters utilizing sequential numbers were flagged for IDOR testing. Authentication workflows were scrutinized for logic bypasses.
- 4.2.5 **Exploitation and Manual Validation:** Discovered weaknesses were manually validated using Burp Repeater to confirm exploitability and rule out false positives. Intruder was utilized to test rate limiting and perform localized fuzzing.
- 4.2.6 **Documentation and Risk Assessment:** Identified vulnerability was verified and documented with proof-of-concept steps, categorized by CWE identifier, and assigned a severity rating based on its potential business impact.

5. Implementation and Vulnerability analysis

Implementation and Vulnerability analysis presents the detailed findings of the vulnerability assessment performed on the enterprise job portal infrastructure [17, 26]. The assessment revealed several critical flaws in API security, access control, and input validation and summarized in the Table 1 and Figure 1.

Table 1: Summary of the assessment with critical flaws

S. No.	Category (Number)	Description	Figure
1	Critical (01)	Unauthenticated Access to Life ray JSON Web Services (JSONWS) API	1a
2	High (04)	OTP Verification Response Manipulation (Authentication Bypass)	1b
		Insecure Direct Object Reference (IDOR) in Registrations	1c
		Server-Side Request Forgery (SSRF)	1d
		Stored Cross-Site Scripting (XSS)	1e
3	Medium (01)	Insufficient Rate Limiting & Resource Exhaustion	1f



Figure 1: Assessment with images of critical flaws

Detected vulnerabilities were categorized and analyzed individually on the basis of their severity. The assessment of vulnerabilities was done and case-wise has presented with case number with title, category, CWE-ID, observation, impact and remediation as described below:

Case number – 1

Title - Unauthenticated Access to Life ray JSON Web Services (JSONWS) API

Category - Critical

CWE-ID: CWE-306 (Missing Authentication), CWE-862 (Missing Authorization)

Observation: During the reconnaissance phase, directory brute-forcing and manual inspection revealed that the Life ray JSON Web Services (JSONWS) interface, located at /api/jsonws, was fully accessible to unauthenticated visitors. The JSONWS endpoint acts as a service catalog, publicly disclosing hundreds of internal API methods intended for backend communication.

Exploitation: By crafting direct HTTP GET/POST requests to the exposed endpoints (e.g. curl -k "https://target/api/jsonws/role/get-user-roles?userId=20101"), the server processed the requests and returned sensitive backend data without requiring any session cookies or authorization headers. Highly privileged administrative methods; including, add-user, delete-user and check-permission were exposed.

Impact: A malicious actor could exploit this misconfiguration to enumeration all applications users, retrieve sensitive organizational structures, discover, administrative accounts, and potentially execute unauthorized actions, leading to complete portal compromise.

Remediation: JSONWS endpoints must be disabled in production environments if not strictly required. If required, strict role-based authorization filters must be applied to all /api/* routes at the web server and application levels.

Case number – 2

Title - OTP Verification Response Manipulation (Authentication Bypass)

Category – High

CWE-ID: CWE-345 (Insufficient Verification of Data), CWE-287, CWE-302

Observation: The application's user registration workflow requires mobile number verification via an OTP (One Time Password). However, the application architecture fundamentally trusts client-side state variables. When an incorrect OTP is submitted, the server returns a JSON response: {"mobileVerified":false}.

Exploitation: Using Burp Suite's Proxy, the HTTP response from the server was intercepted before it reached the browser. The response body was manually modified from false to {"nextButton":true,"mobileVerificationMessage":"Mobile Verification Successful", "mobileVerified":true}. The client-side JavaScript processed this manipulated response, assumed the OTP was correct, and unlocked the next phase of the registration workflow.

Impact: This logic flaw completely circumvents the identity validation controls. Attackers can register thousands of fraudulent accounts using unverified, spoofed, or non-existent mobile numbers.

Remediation: Authentication and verification decisions must occur entirely on the server. Upon successful OTP validation, the server should issue a secure, signed cryptographic token (or update the server-side session state), which must be validated during the final registration submission.

Case number – 3

Title - Insecure Direct Object Reference (IDOR) in Registrations

Category – High

CWE-ID: CWE-639 (Authorization Bypass Through User-Controlled Key)

Observation: The portal's Domain Expert and Employer Registration modules allow users to view their application statuses. The URLs to access these records utilize predictable, sequential numeric identifiers, such as ...&conId=2701 and ...&empID=2004.

Exploitation: By capturing the request in Burp Repeater and manipulating the numeric identifier (e.g., changing 2701 to 2702, 2703 etc.), the server returned the registration records of other organizations. The application enforced access control only by hiding the links in the UI, completely failing to perform server-side ownership checks on the requested object IDs.

Impact: Total compromise of data confidentiality. Attackers can systematically scrape the database, accessing highly sensitive PII, business registration numbers, uploaded legal documents, and contact information of all registered entities.

Remediation: Implement strict object-level authorization (BOLA/IDOR protection). The server must verify that the currently authenticated user session has explicit permissions to view the requested empId. Furthermore, predictable numeric IDs should be replaced with randomized UUIDs (Universally Unique Identifiers).

Case number – 4

Title - Server-Side Request Forgery (SSRF)

Category – High

CWE-ID: CWE-918 (Server-Side Request Forgery)

Observation: The Marketplace App Manager component contains an endpoint (/o/marketplace-app-manager-web/icon.jsp)) that accepts a user-supplied URL via the iconURL parameter to fetch and display an image.

Exploitation: A custom Burp Collaborator payload (e.g., <https://attacker-controlled.oastify.com>) was submitted to the iconURL parameter. Monitoring the Collaborator client revealed that the backend application server initiated outbound DNS resolution and HTTP GET requests directly to the attacker's domain.

Impact: SSRF allows an attacker to abuse the server's network positioning. It can be weaponized to bypass external firewalls, port-scan internal networks, access internal APIs (like Elasticsearch or Redis), and potentially extract highly sensitive temporary credentials from Cloud Metadata endpoints (e.g., AWS 169.254.169.254).

Remediation: Implement a strict, server-side allow-list of permitted domains for URL fetching. Ensure the fetching logic explicitly blocks resolution of internal IP ranges (10.0.0.0/8, 127.0.0.1, 169.254.169.254) and disable support for dangerous URI schemes (file://, dict://).

Case number – 5

Title - Stored Cross-Site Scripting (XSS)

Category – High

CWE-ID: CWE-79 (Improper Neutralization of Input)

Observation: The application fails to adequately sanitize user input within the 'User Profile Name' and 'Job Title' creation forms. The server stores the raw input in the database and reflects it without HTML entity encoding.

Exploitation: An attacker registered an account and updated their profile name to include a malicious payload: `<a onmouseover="alert(document.cookie)" style=display:block>Hover Me</a>`. Whenever an administrator or another user navigated to a page displaying this profile, the browser interpreted the payload as executable code rather than text data.

Impact: Execution of arbitrary JavaScript within the security context of the victim's session. This leads to session cookie theft (if HttpOnly is missing), unauthorized actions performed on behalf of the victim (CSRF via XSS), and persistent credential harvesting.

Remediation: Adopt a defense-in-depth approach. Implement strict server-side input validation. Crucially, apply context-aware output encoding before rendering any user-supplied data in the DOM. Deploy a robust Content Security Policy (CSP) header to restrict inline script execution

Case number – 6

Title - Insufficient Rate Limiting & Resource Exhaustion

Category – Medium

CWE-ID: CWE-307, CWE-400

Observation: Critical endpoints, including the Forgot Password (OTP generation) and public registration APIs, lacked adequate request throttling. Using Burp Suite's Turbo Intruder, hundreds of OTP requests were fired within seconds. The server processed all requests without triggering CAPTCHA, account lockouts, or IP bans.

Impact: This facilitates SMS/Email bombing attacks, exhausting financial resources tied to SMS gateways, and enables automated mass-account creation by botnets.

Remediation: Implement IP-based and user-centric rate limiting (e.g., max 3 OTP requests per minute). Introduce progressive delays or CAPTCHA challenges after consecutive requests.

6. Validated security controls (Pass cases)

A comprehensive vulnerability assessment must also document the security mechanisms that function correctly. Testing verified that the application possesses strong defenses against several critical attack vectors, establishing a baseline of security in specific areas [17, 26].

6.1 Control Assessed – I is User Enumeration Protection and its methodology and validation Observation. Burp Intruder was utilized to submit a dictionary of both valid and non-existent email addresses to the login endpoint. The application returned identical HTTP status codes (HTTP 302/403), consistent error messages ("Authentication failed"), and uniform response times. The application successfully obfuscates whether an account exists, mitigating reconnaissance and targeted brute-force attacks.

6.2 Control Assessed – II is Brute-Force & Credential Stuffing Prevention and its methodology and validation Observation. Automated high-frequency login attempts using Burp Intruder were executed. The application accurately tracked consecutive failed authentication attempts. Upon reaching a predefined threshold, the account was locked, and subsequent requests returned HTTP 403 Forbidden responses. This effectively neutralizes automated dictionary attacks.

6.3 Control Assessed – III is Session Management & Entropy and its methodology and validation Observation. Session tokens (JSESSIONID) were analyzed using Burp Sequencer, analyzing 10,000 tokens for statistical randomness. The tokens exhibited high entropy, preventing session prediction. Furthermore, the application successfully invalidates the session token on the server side upon user logout, preventing session reuse. New tokens are properly issued upon authentication, neutralizing Session Fixation vulnerabilities.

6.4 Control Assessed – IV is Input Validation (Buffer Overflow Prevention) and its methodology and validation Observation. The application was subjected to boundary testing. Excessively large strings (10,000+ characters) and malformed payloads were submitted to job posting and profile fields. The application safely truncated the data or rejected the input with handled exceptions. No memory corruption, application crashes, or stack/heap overflow conditions were observed.

6.4 Control Assessed – V is Transport Layer Security (TLS/ HTTPS) and its methodology and validation Observation. All communication, particularly authentication requests and session tokens, is strictly transmitted over encrypted

HTTPS channels (TLS 1.3). Interception of plaintext credentials is prevented. Furthermore, session cookies are appropriately hardened with Secure, HttpOnly, and SameSite=Strict attributes, drastically reducing the impact of client-side script interception (XSS).

7. Strategic recommendations

To establish a resilient security posture and mitigate the risks identified in this report, the following strategic remediation must be prioritized by the development and infrastructure teams:

1. **Implement Strict Server-Side Authorization:** The presence of IDOR and exposed APIs highlights a reliance on UI-level hiding. The application must enforce comprehensive Role-Based Access Control (RBAC) on the backend. Every API endpoint must cryptographically verify that the authenticated user possesses the requisite permissions before executing actions or returning data. Replace sequential identifiers (IDs) with unpredictable UUIDs to prevent enumeration.
2. **Adopt Secure Input / Output Handling:** Do not rely on client-side validation. All user input must be validated against strict allow-lists on the server side. To neutralize XSS vectors, all dynamic content must undergo context-aware HTML entity encoding before being rendered in the browser. Deploying a strict Content Security Policy (CSP) is highly recommended.
3. **Secure State Management:** Critical workflows, such as OTP validation and password resets, must tie verification state to secure, short-lived server-side session variables. The application must completely distrust Boolean validation states (e.g., "mobileVerified":true) provided by client-side JSON requests.
4. **Infrastructure and Network Hardening:** Minimize the external attack surface. Disable the Life ray JSONWS API in production if it is not utilized. Restrict public access to backend services; only Ports 80 and 443 should be exposed. Administrative interfaces (SSH) and databases (MySQL) must reside in private subnets, accessible only via secure VPNs or Bastion hosts.

8. Conclusion

This project successfully demonstrated the practical application of Burp Suite Community Edition in conducting a rigorous, manual vulnerability assessment of a live enterprise web application. By establishing an intercepting proxy environment and deeply analyzing HTTP traffic, the assessment uncovered several high-impact vulnerabilities including Unauthenticated API Access, Authentication Bypasses via response manipulation, IDOR, SSRF, and XSS [17, 26].

Crucially, these findings highlight the limitations of automated security scanners, which frequently fail to comprehend complex business logic and authorization boundaries. The discovery of the OTP manipulation bypass and IDOR flaws emphasizes that while modern web frameworks provide a baseline of security, improper architectural implementation and an over-reliance on client-side trust inevitably introduce severe risks [27-29].

In conclusion, robust web application security cannot be achieved as an afterthought. It requires a "Security-by-Design" approach, integrating threat modeling, secure coding standards (referencing the OWASP Top 10), and continuous manual penetration testing throughout the Software Development Life Cycle (SDLC) to protect sensitive data and maintain organizational integrity against an evolving landscape of cyber threats.

References

1. Ross, D.'S., Dhesi, Singh, A., Kate, P., Andrew, C., Sophie, S., Watkins, N. A., & Murphy, M. F. (2023). Comparing transfusion practice at multiple hospitals using electronically collected and analysed data. *Transfusion Medicine*, 33(6), 453-458.
2. Mallipeddi, R. R., Demirezen, E. M., Kumar, S., Gopal, R. D. (2024). Dynamics of competition on openness strategies and software maintenance. *MIS Quarterly*, 48 (3), 1239.
3. Lucas, H. C. Jr. (2005). Information technology: Strategic decision making for Managers. New York: John Wiley and Sons.
4. Senn, J. A. (2004). Information technology: Principle, practices, opportunities (3rd ed.). Upper Saddle River, NJ: Pearson/Prentice Hall.
5. Grover, V., & Kohli, R. (2013). Revealing your hand: Caveats in implementing digital business strategy. *MIS Quarterly*, 37, 655–662. Retrieved November 21, 2013, from EBSCO Online Database Business Source Complete. <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=87371464>
6. Junglas, I., & Harris, J. (2013). The promise of consumer technologies in emerging markets. *Communications of the ACM*, 56, 84–90. Retrieved November 21, 2013, from EBSCO Online Database Business Source Complete. <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=87500283>
7. Leonardi, P. M. (2013). When does technology use enable network change in organizations? A comparative study of feature use and shared affordances. *MIS Quarterly*, 37, 749–775. Retrieved November 21, 2013, from EBSCO Online Database Business Source Complete. <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=89477337>
8. Wasyihun, S. A., Yirga, Y. M., & Abebe, A. D. (2024). Cyber security: State of the art, challenges and future directions. *Cyber Security and Applications*. 2 (100031), 1-9.
9. Reddy, G. N., & Reddy, G. J. U. (2014). A Study Of Cyber Security Challenges And Its Emerging Trends On Latest Technologies. *International Journal of Engineering and Technology – UK*, 4 (1). January 2014. <https://arxiv.org/pdf/1402.1842v1>.
10. Ali, A., & Muhammad, A. (2026). Cyber security essential for social workers: Mitigating AI-driven social engineering through survey-based behavioral interventions. *Kashf Journal of Multidisciplinary Research*. 3(6), 55 – 62.
11. Sahu, M., & Gupta, G. P. (2025). Cyber Attack Detection using Stacking Ensemble Model for Internet of Vehicle. In January 2025 Fourth International Conference on Power, Control and Computing Technologies (ICPC2T) (pp. 35-39). IEEE. <https://doi.org/10.1109/ICPC2T63847.2025.10958735>.
12. Korium, M. S., Saber, M., Beattie, A., Narayanan, A., Sahoo, S., & Nardelli, P. H. (2024). Intrusion detection system for cyberattacks in the Internet of Vehicles environment. *Ad Hoc Networks*, 153, 103330. <https://doi.org/10.1016/j.adhoc.2023.103330>.
13. Noluntu, M. (2024). An Analysis of Cybersecurity Architectures. Proceedings of the 19th International Conference on Cyber Warfare and Security, ICCWS-2024 (pp. 200 – 207).
14. Kalogeraki, E. M., & Polemi, N. (2024). A taxonomy for cybersecurity standards. *Journal of Surveillance, Security and Safety*, 5, 95-115.
15. Prasad, K. S., Lydia, E. L., Rajesh, M. V., Radhika, K., Ramesh, J. V. N., Neelima, N. & Pokuri, S. R. (2024). Augmenting cybersecurity through attention based stacked autoencoder with optimization algorithm for detection and mitigation of attacks on IoT assisted networks, *Scientific Reports*, 14(1), 30833.
16. Hermann, D., & Pridöhl, H. (2020). Basic Concepts and Models of Cybersecurity. In Christen, M., Gordijn, B., & Loi, M. (Eds.), *The Ethics of Cybersecurity* (pp. 11–44). Springer International Publishing. DOI: 10.1007/978-3-030-29053-5_2.
17. OWASP Top 10 (2021): The Open Worldwide Application Security Project's list of the 10 most critical web application security risks. Available at: <https://owasp.org/www-project-top-ten/>

18. Olaes, T. (2025). What is NIST Cybersecurity Framework (CSF) 2.0? | Balbix.” Accessed: Jun. 25, 2025. Online Available: <https://www.balbix.com/insights/nist-cybersecurity-framework/>.
19. Möller, D. P. F. (2023). NIST Cybersecurity Framework and MITRE Cybersecurity Criteria, in Guide to Cybersecurity in Digital Transformation, pp. 231–271. doi: 10.1007/978-3-031-26845-8_5.
20. Muñoz, A., Palomino, A. & Wong, L. (2023). Cybersecurity framework for SMEs in Peru based on ISO/IEC 27001 and CSF NIST controls, in 2023 18th Iberian Conference on Information Systems and Technologies (CISTI), IEEE, Jun. 2023, pp. 1–7. doi: 10.23919/CISTI58278.2023.10211874.
21. The NIST Cybersecurity Framework (CSF) 2.0 NIST CSWP in 29 February 26, 2024. Available free of charge from: <https://doi.org/10.6028/NIST.CSWP.29>.
22. CWE (Common Weakness Enumeration): A community-developed list of common software and hardware weakness types. Available at: <https://cwe.mitre.org/>
23. CVE (Common Vulnerabilities and Exposures): The reference method for publicly known information security vulnerabilities and exposures. Available at: <https://www.cve.org/>
24. Rahalkar, S. (2021). A Complete Guide to Burp Suite: Learn to Detect Application Vulnerabilities in January 2021, ISBN: 978-1-4842-6401-0. DOI:10.1007/978-1-4842-6402-7.
25. Sankaran, V.N. & Fernandes, J.B. (2024). Advanced Vulnerability Detection and Exploitation in Web Applications Using Burp Suite. *International Journal of Modern Computation Information and Communication Technology*, 7(2), 67-74.
26. Gowtham, N. & Latha, D.S. (2026). Vulnhawk: Design and development of a lightweight web application penetration testing tool with native AI integrated aligned with OWASP, *International Journal of Information Security (IJIS)*, 5 (1), 26-41,
27. Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). Zero Trust Architecture, NIST Special Publication 800-207, National Institute of Standards and Technology, U.S. Department of Commerce, Walter Copan, NIST Director and Under Secretary of Commerce for Standards and Technology , Available free of charge from: <https://doi.org/10.6028/NIST.SP.800-207>.
28. Almheiri, S. J., Shah, A. A., Abbas, S., Ahmad, M., & Khan, M. A. (2025). Smart sustainable cyber security: modelling an interpretable and transparent threat detection with explainable artificial intelligence, *Discover Sustainability*, 6:442-461. Available on <https://doi.org/10.1007/s43621-025-01280-z>
29. Paul, E. B., Scarfone, K. & Souppaya, M. (2008). Cyber security metrics and measures, National Institute of Standards and Technology, Gaithersburg, Maryland, pp 1-8; Handbook of Science and Technology for Homeland Security, Vol. 5, Edited by John G. Voeller Copyright 2008 John Wiley & Sons, Inc.