Integrating the NIST 800-30 Risk Management Framework with Penetration Testing: A Case Study of Web-Based Training Information Systems in a Cybersecurity Company

Yohanes Dewantara Marpaung[1], Halim Budi Santoso[2*], Erick Kurniawan[3], Gabriel Indra Widi Tamtama[4] , Abdul Karim[5]

[1-3]Information Systems Department, Universitas Kristen Duta Wacana, Yogyakarta

[4]Institute of Service Science, College of Technology Management, National Tsing Hua University, Hsinchu City, Taiwan

[5]Cerebrovascular Disease Research Center and Department of Artificial Intelligence Convergence, Hallym University, Chuncheon, Gangwon, South Korea

E-mail: yohanes.dewantara@si.ukdw.ac.id[1], hbudi@staff.ukdw.ac.id [2], erick@staff.ukdw.ac.id[3], gabrielindra.giwt@iss.nthu.edu.tw [4], abdulkarim@korea.ac.kr [5]

Abstract. Web application security has emerged as a critical concern due to the increasing frequency of cyber threats targeting internet-based information systems. Web developers must prioritize risk management, particularly by incorporating risk identification within the information security management system. Companies specializing in information security management must exercise caution when deploying web-based applications, as information security significantly impacts a company's reputation. However, previous research has often inadequately addressed this issue. Therefore, this study explores the integration of a risk management framework with penetration testing. The combination of penetration testing with NIST SP 800-30 is anticipated to yield a comprehensive risk assessment. Penetration testing employed a grey-box methodology, guided by OWASP WSTG v4.2 and supplemented by CVSS v3.1 for severity measurement. Subsequently, NIST SP 800-30 was utilized as the risk management framework. Grey-box testing was conducted on a newly deployed web-based training management system. Consequently, four vulnerabilities were identified and verified through Proof of Concept: SQL Injection (High), Blind Stored XSS (Critical), Unrestricted file upload enabling remote code execution (Critical), and Brute Force Attack (High). A thorough risk identification and mitigation process was then undertaken for these vulnerabilities. This study also offers improvement suggestions to assist in mitigating the identified vulnerabilities. Based on the resulting risk profile, the study recommends prioritizing remediation of high-risk vulnerabilities, minimizing unnecessary information exposure, enhancing authentication and session controls where weaknesses are detected, and conducting verification testing following corrective actions. The proposed integration enables organizations to transform technical penetration-testing findings into prioritized risk-treatment decisions based on the likelihood of exploitation, the potential impact on organizational assets, and the urgency of remediation.

Keywords: penetration testing; web application security; risk management; OWASP WSTG; NIST SP 800-30

1. Introduction

In contemporary information system development, web application security has emerged as a critical concern, given that numerous organizational activities now rely on web-based applications for managing user data, operational records, learning materials, reports, and internal workflows. The increasing complexity of web applications concomitantly enlarges the attack surface that unauthorized entities may

exploit [1, 2]. Common vulnerabilities, including injection, weak authentication, insecure session management, security misconfiguration, and unrestricted file upload, remain highly pertinent as they can compromise the confidentiality, integrity, and availability of information systems [3-5]. In the context of web applications processing sensitive data, these vulnerabilities may result in data leakage, account takeover, service disruption, or unauthorized server access [1].

A web security company bears a heightened responsibility to conduct systematic security testing on its internal applications. Such a company not only provides security services to external clients but also relies on internal systems for technical operations, analyst training, documentation, and knowledge management [2, 6]. Consequently, a web security company should robustly implement the prevailing standards of information security management systems, adhering to international standards such as ISO-27001 or COBIT 2019 [7, 8]. Furthermore, in this era of digital transformation, information serves as a critical source for decision-making, influencing both individual and organizational behavior [5, 9-11]. If its internal web application is vulnerable, the risk extends beyond data exposure, potentially impacting on professional credibility, service quality, secure coding awareness, and the company's capacity to demonstrate sound internal security practices [5, 10]. Therefore, penetration testing should not be regarded solely as a service offered to clients but also as an internal assurance mechanism to assess whether the company's own systems follow the security standards it advocates.

One of the prevailing methodologies for identifying vulnerabilities in web applications is penetration testing. This approach is extensively employed to assess web application security, as it enables testers to emulate attacker behavior and ascertain whether a vulnerability can be exploited. Priambodo, Rifansyah and Hasbi [12] demonstrated that relying solely on automated tools for vulnerability assessment is inadequate, as the findings from scanners require validation through penetration testing. Their study utilized OWASP Top 10, OWASP ZAP, Vega, and WSTG v4.2 to authenticate web application vulnerabilities and mitigate false positives. Tinambunan, Junaidi and Rizki [13] further illustrated that penetration testing grounded in the OWASP Top 10 framework can effectively identify significant vulnerabilities in academic information systems, including those related to access control, injection, security misconfiguration, and vulnerable components. Rozali and Sinaga [14] also showed that penetration testing on a school website can facilitate the identification of security alerts through information gathering, vulnerability analysis, and reporting. These studies collectively affirm that OWASP-based penetration testing is efficacious in identifying and validating technical vulnerabilities.

However, penetration testing results often remain technical in nature. A penetration test can prove that SQL Injection, Cross-Site Scripting, file upload abuse, or brute force attack is possible [2, 14, 15], but the output may still be limited to a list of vulnerabilities, severity labels, screenshots, and technical recommendations. Without risk management, these findings may lack business impact context, remediation priority, risk ownership, and residual risk estimation [16, 17]. This limitation is important for a web security company because technical findings must be translated into actionable risk decisions [16]. The company must know which vulnerability should be handled first, which control should be applied, who should be responsible, and how much risk remains after remediation.

Risk management frameworks effectively address limitations by enabling organizations to assess vulnerabilities based on likelihood, impact, and control effectiveness. The NIST SP 800-30 framework offers a structured methodology for identifying threats, vulnerabilities, likelihood, impact, risk levels, and control recommendations. Syafitri [18] investigation of NIST SP 800-30 to an academic information system demonstrated its capability to systematically classify risks into high, medium, and low categories. Similarly, Pambudi and Ramli [8] employed NIST SP 800-30 to design information security risk management for Supervision Management Information System, highlighting the framework's utility in focusing on specific infrastructure boundaries, providing detailed assessment steps, and supporting control recommendations. These studies underscore the robustness of NIST SP 800-30 as a structured risk assessment framework, although they primarily rely on interviews, observations, document reviews, and organizational risk analysis rather than direct evidence of exploitability from penetration testing.

Numerous studies, as shown in Table 1, have endeavoured to establish a connection between technical testing and risk assessment. For instance, the integration of OWASP-based vulnerability assessment, penetration testing, and CVSS has been employed to evaluate web application security risks and facilitate a more quantifiable risk evaluation [12, 17]. The current manuscript also acknowledges that while

penetration testing can technically identify and substantiate vulnerabilities, without a risk management framework, the findings remain merely a list of technical deficiencies lacking business impact context and clear mitigation priorities [17, 19]. This suggests that the primary concern is not solely the identification of vulnerabilities, but also the transformation of verified technical findings into risk scores, treatment plans, and projections of residual risk. Table 1 shows the major findings of prior research in risk management or penetration testing.

Table 1. Summary of Extant Literature and Major Findings

Literature	Major Findings
Syafitri [18]	Applied NIST SP 800-30 to assess information security risks in an academic information system. The study identified one high risk, five medium risks, and 52 low risks, showing that NIST SP 800-30 can classify information security risks systematically.
Wichmann, Groddeck and Federrath [6]	Proposed FileUploadChecker to detect, reject, or sanitize malicious uploaded files at the request level. The study shows that file upload functions require validation of file metadata, extension, MIME type, file signature, malicious content, and server-side execution behavior.
Priambodo, Rifansyah and Hasbi [12]	Conducted web penetration testing using OWASP Top 10, OWASP ZAP, Vega, and OWASP WSTG v4.2. The study found nine vulnerability types and emphasized that automated vulnerability scanning requires penetration testing validation to reduce false positives.
Pambudi and Ramli [8]	Designed information security risk management using NIST SP 800-30 for SIMWAS. The study identified 17 risks, consisting of four low risks, eight moderate risks, and five high risks. The study also stated that NIST SP 800-30 is useful because it provides detailed assessment steps and broad control recommendations.
Ajufo and Qutieshat [5]	Reviewed human factors in cybersecurity and identified social engineering, weak security culture, risky password practices, stress, burnout, and security fatigue as key factors behind cyber incidents. The study supports the need to include authentication risk and user-related controls in security assessment.
Lina [15]	Demonstrated brute force testing using Burp Suite. The study found that brute force attacks can succeed when a login system does not implement CAPTCHA, sufficient login attempt limitation, account lockout, or Two-Factor Authentication.
Rozali and Sinaga [14]	Conducted penetration testing on a school website using Whois, Nmap, and OWASP ZAP. The study produced technical security recommendations, but the output was mainly limited to website security diagnosis and did not integrate structured risk scoring.
Tinambunan, Junaidi and Rizki [13]	Tested an academic information system using penetration testing based on OWASP Top 10. The study found 23 vulnerabilities, with 20 mapped to OWASP Top 10 categories, confirming that OWASP-based testing is useful for identifying major web application weaknesses.
Prasetyo, Salsabila and Retnani [20]	Applied ISO 31000 and Bow Tie Analysis in a higher education IT context. The study identified 21 risks and used Bow Tie Analysis to define preventive and mitigative actions for priority risks. It strengthens the importance of risk treatment after risk identification.
Egbedion [3]	Discussed the role of vulnerability management and penetration testing in security-informed IT project planning. The study emphasized that both practices help identify risks early, reduce project disruption, support regulatory compliance, and improve cybersecurity resilience.
Raihan, Prabowo and Oktaria [21]	Integrated OWASP, VAPT, CVSS, and NIST SP 800-30 Revision 1 for web application risk evaluation. The study found 16 vulnerabilities and showed that CVSS can support impact evaluation within NIST-based risk assessment.
El Marzak, Chahid, Faris and Mansouri [16]	Developed a unified cyber risk management framework by integrating ISO/IEC 27005 and NIST SP 800-30 using UML, RDF, and OWL. The study highlights the importance of interoperability, traceability, and dynamic risk recalculation when new vulnerabilities are reported.
Monageng and Esiefarienrhe [22]	Developed E-RAF by integrating Scrum and a CASE tool to automate risk assessment using a probability-impact matrix, risk register, dashboard, and mitigation tracking. The study shows that risk assessment should support visibility, prioritization, and continuous monitoring

The studies summarized in Table 1 reveal that prior research has explored web penetration testing, OWASP-based vulnerability validation, specific vulnerability mitigation, and NIST-based risk assessment. Nevertheless, many penetration testing studies primarily report on vulnerability categories, technical severity, and remediation suggestions, while numerous NIST-based risk assessment studies depend on interviews, observations, document reviews, or general control analysis. Although recent investigations have commenced integrating technical testing with risk assessment, there remains a necessity for an evidence-based workflow that translates verified Proof-of-Concept exploitation from grey-box penetration testing into CVSS severity scoring, NIST SP 800-30 likelihood-impact assessment, prioritized Risk Treatment Plans, and residual risk projection [17]. This gap is particularly pertinent for internal web applications utilized by a web security company, where security testing must support both operational protection and cybersecurity training.

Recognizing this critical gap, this study aims to explore the integration of penetration testing within a risk management framework utilizing NIST 800-30. By employing an intern management web application from a web security company, this research seeks to address the existing gap in literature. The selected case is appropriate as the intern management web application is utilized for managing intern participant data, assignments, learning materials, company-related materials, and activity records. Additionally, the application serves as a controlled learning environment for cybersecurity training. Given its dual role in operational data management and security training, the application necessitates testing through a method that is technically verifiable, risk-oriented, and conducive to enhancing secure coding practices. A grey-box testing approach is deemed suitable, as it involves the tester having partial access as an authenticated non-administrator user, without access to the source code, database structure, or backend configuration. This scenario accurately reflects a realistic threat from an authenticated internal user.

This study examines four categories of vulnerabilities: SQL Injection, Blind Stored Cross-Site Scripting (XSS), Unrestricted File Upload leading to Remote Code Execution (RCE), and Brute Force Attack. These categories were chosen due to their prevalence and significant impact on web applications [6, 15]. The novelty of this research lies in transforming verified Proof-of-Concept exploitation into a structured risk scoring system, a prioritized Risk Treatment Plan, and a projection of residual risk. The research not only investigates the existence of vulnerabilities but also assesses their severity, likelihood of exploitation, potential impact, prioritization of controls, and changes in residual risk post-remediation. Consequently, this study aims to address the following questions: *(1) What vulnerabilities are present in the intern management web application of a web security company? (2) What is the severity level of each vulnerability based on CVSS v3.1 and Proof of Concept evidence? (3) How can the integration of penetration testing and NIST SP 800-30 yield a comprehensive risk assessment for the application?* The objective of this research is to identify, exploit, measure, and evaluate web application vulnerabilities through an integrated approach that links technical evidence with risk-based decision-making. This study is anticipated to offer practical value to web security companies, junior security analysts, intern training programs, and software developers who require a structured reference for implementing secure coding and risk-based remediation within the Software Development Lifecycle.

2. Research Methodology

To answer the research questions, we employed the following methodology, as illustrated in Figure 1. The methodology includes: (1) Research scope and grey-box testing setup; (2) Penetration testing based on OWASP WSTG v4.2; (3) OWASP WSTG Result Analysis; (4) Severity assessment of the internship web application; (5) Risk Management Evaluation Based on NIST SP 800-30; (6) Defining Control Action Plan (Risk Treatment Plan). The process begins by defining the research scope, testing context, target system, authenticated user access, testing boundaries, ethical rules, and controlled testing environment.

The grey-box approach was selected because the tester had partial access to the internship management web application as an authenticated user, but did not have access to the source code, database structure, administrator credentials, or backend configuration [23]. This approach was chosen because it simulates threats from authenticated internal users, which represents the most realistic threat scenario for organizational web applications [23]. Testing tools used included Burp Suite (intercepting

proxy), Nmap (port scanning), Dirsearch (directory enumeration), SQLMap (SQL injection exploitation), and Wappalyzer (technology fingerprinting). This setting represents a realistic internal threat scenario in which a user with limited privileges attempts to identify and exploit weaknesses in the system

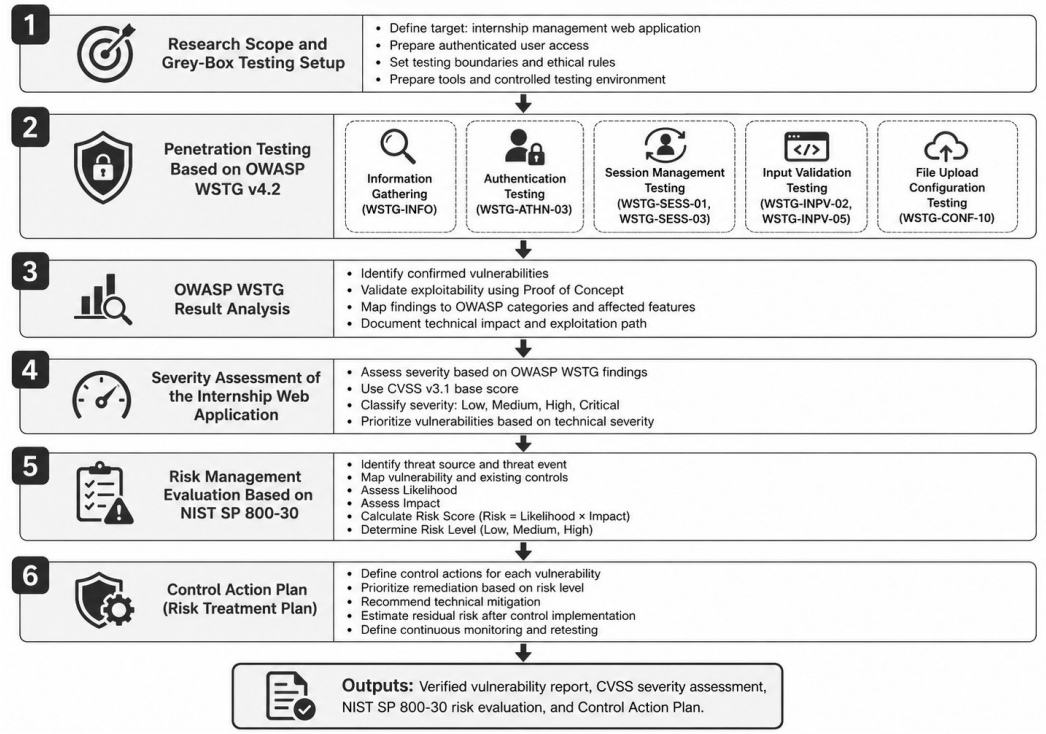

Figure 1. Research Methodology

Following the defining scope of research, the researchers conducted technical testing using OWASP WSTG v4.2 as the main penetration testing guide. The testing scope covered six phases: (1) Planning and Scope Definition, (2) Information Gathering (WSTG-INFO-01 to WSTG-INFO-10), (3) Authentication Testing specifically Brute Force (WSTG-ATHN-03), (4) Session Management Testing (WSTG-SESS-01, WSTG-SESS-03), (5) Input Validation Testing including XSS and SQL Injection (WSTG-INPV-02, WSTG-INPV-05), and (6) File Upload Configuration Testing (WSTG-CONF-10). Each identified vulnerability was documented with a Proof of Concept to demonstrate real exploitability. Information gathering was conducted within the testing scope. The target scope included domain, sub-domain, and application module, while excluded services remained outside the assessment scope. The information-gathering phase combined passive and active reconnaissance. Passive reconnaissance involved reviewing publicly accessible information, including domain and DNS records, indexed application pages, digital certificate information, robots.txt, sitemap files, publicly exposed documents, and other technical information available without directly interacting with protected application functions. Active reconnaissance was subsequently conducted. This process identified the reachable hosts, open services, web-server characteristics, application technologies, accessible directories, forms, request parameters, cookies, authentication endpoints, and other potential application entry points.

Using this approach, the research team can collect and identify some vulnerabilities, including open ports, hidden directories, exposed files, and the technology stack used by the application. Each observation was recorded in a testing worksheet containing the date and time of collection, the tool or technique used, the affected asset, the relevant request and response evidence, and its potential security relevance. The resulting attack-surface inventory was then used to determine the test cases applied during the vulnerability assessment, authentication testing, session-management testing, and risk-analysis phases. Authentication testing focused on weak lockout mechanisms and brute force resistance, while session management testing examined whether session handling could be abused through XSS-based session hijacking. Input validation testing was used to identify SQL Injection and Stored XSS, while file upload configuration testing evaluated whether the application properly restricted uploaded files.

After the OWASP WSTG testing phase, the findings were analyzed to determine which vulnerabilities were technically valid and exploitable [2, 12, 13, 21]. Each confirmed vulnerability was verified through a Proof of Concept to demonstrate its real impact on the internship management web application. The analysis included identifying the affected endpoint, vulnerable parameter, attack path, exploitation result, and technical impact. In the main research, four verified vulnerabilities were identified: SQL Injection, Blind Stored XSS leading to session hijacking, Unrestricted File Upload leading to Remote Code Execution, and Brute Force Attack on the login form. The verified findings were then assessed using CVSS v3.1 to determine their technical severity. CVSS was used to assign a base score and severity category by considering exploitability and impact metrics such as attack vector, attack complexity, privileges required, user interaction, scope, confidentiality, integrity, and availability.

The CVSS assessment, shown in Table 2, helped classify the vulnerabilities into High and Critical categories and provided a quantitative basis for prioritizing the findings before they were evaluated from a risk management perspective. Each identified vulnerability was assigned a CVSS Base Score using CVSS v3.1 to objectively and quantitatively measure severity, across three main components: Ease of Exploitation, Impact, and Scope (see Figure 2).

The Ease of Exploitation metric describes how easily a vulnerability can be exploited. It includes the attack vector (AV), attack complexity (AC), required access privileges (PR), and user interaction (UI). For example, vulnerabilities that can be exploited remotely over a network, have low attack complexity, and do not require user interaction will generally receive a higher ease-of-exploitation score. The Impact metric describes the consequences of successful exploitation on Confidentiality (C), Integrity (I), and Availability (A) [24, 25]. This metric helps determine whether vulnerability could expose sensitive data, modify system resources, or compromise system availability.

The Scope metric connects exploitability and impact by identifying whether the exploited vulnerability affects only the vulnerable component or can extend its impact to other components. This metric is important for vulnerabilities such as Stored XSS or Remote Code Execution, where a weakness in one application feature may lead to broader consequences, such as session hijacking, administrator account takeover, or server-level compromise. In this study, the Base Metric Group (as illustrated in Figure 2) supports the severity assessment stage by providing a standardized way to evaluate vulnerabilities found through OWASP WSTG v4.2 testing, such as SQL Injection, Blind Stored XSS, Unrestricted File Upload, and Brute Force Attack.

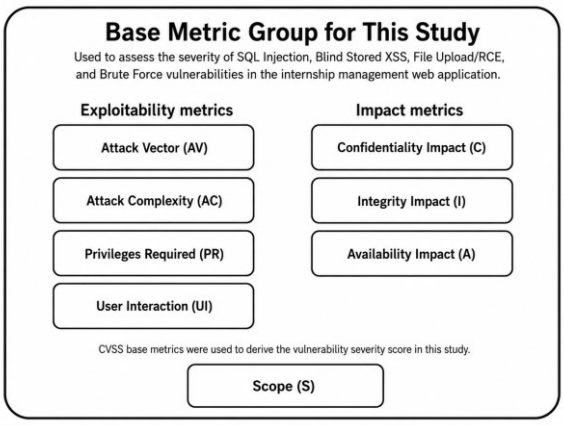

Figure 2. Base Metric Group

Table 2. CVSS Score

Rating	CVSS Score
None	0.0
Low	0.1 – 3.9
Medium	4.0 – 6.9
High	7.0 – 8.9
Critical	9.0 – 10.0

Finally, the study applied NIST SP 800-30 to transform technical vulnerability findings into risk-based decisions. Each vulnerability was evaluated based on threat source, threat event, existing controls, likelihood, and impact. The risk score was calculated using the likelihood and impact matrix, then categorized into low, medium, or high risk. Based on the risk level, a Control Action Plan, also referred to as a Risk Treatment Plan, was developed for each vulnerability. This plan included technical remediation actions, remediation priorities, expected residual risk after control implementation, and recommendations for continuous monitoring and retesting. Through this process, the methodology did not stop at vulnerability discovery but converted penetration testing evidence into structured risk management outputs.

Penetration testing findings were analyzed using the NIST SP 800-30 framework through four steps: (1) Identification of threats and vulnerabilities based on technical testing results; (2) Assessment of Likelihood (L) and Impact (I) on a scale of 1 (Low), 2 (Moderate), 3 (High) with CVSS scores as an additional reference for Impact; (3) Calculation of Risk Score using the equation Risk=L×I with a range of 1–9; and (4) Development of a Risk Treatment Plan (RTP) based on risk priority along with residual risk projections after controls are applied [18].

3. Results and Analysis

This section presents the results of the grey-box penetration testing and the subsequent risk assessment. The analysis follows the integrated workflow described in the methodology, beginning with information gathering, followed by vulnerability validation using OWASP WSTG v4.2, severity assessment using CVSS v3.1, risk assessment using NIST SP 800-30, and the development of a Risk Treatment Plan. This structure ensures that the findings are evaluated not only as technical vulnerabilities but also as security risks requiring prioritized mitigation.

3.1. Information Gathering Results

The information-gathering phase was conducted to map the attack surface of the web application before vulnerability validation and exploitation. Following the Information Gathering category of the OWASP Web Security Testing Guide, the assessment used Nmap for network and service discovery, Dirsearch for directory and file enumeration, and Wappalyzer for technology fingerprinting. The purpose of this phase was to identify exposed services, accessible resources, and underlying technologies that could guide the selection of subsequent penetration-testing scenarios.

The Nmap scan identified several open TCP ports, including port 22 for SSH, port 80 for HTTP, port 81 for the primary web application, and additional services on ports 5000, 5001, and 5003. Port 81 was reported as running Nginx version 1.24.0. This result was manually validated by accessing the corresponding address and port through a web browser. The endpoint displayed the login page, confirming that port 81 hosted the target application and should serve as the primary endpoint for further testing. The presence of several externally accessible services also indicates that the necessity, exposure, and access restrictions of each service should be reviewed to reduce the overall attack surface.

Dirsearch identified several accessible paths, including /.git/, /.git/config, /.git/HEAD, /uploads/, /pages/, and /README.md. The requests to /.git/config and /.git/HEAD returned HTTP status code 200. These findings were manually verified by accessing the files directly. The returned responses contained valid Git repository metadata rather than generic error pages, confirming that the .git directory was exposed. This exposure was considered critical because it could allow an attacker to reconstruct the repository, retrieve source code, examine application configurations, or discover sensitive development information. Dirsearch also detected the /uploads/ directory, although the initial request returned HTTP status code 500. Its existence and security relevance were subsequently confirmed during the exploitation phase, where the application accepted an uploaded PHP web shell and stored it in the directory. Accessing the uploaded file resulted in remote code execution, demonstrating that /uploads/ was not merely a false-positive enumeration result but an operational file-storage location with insufficient security controls.

Technology fingerprinting using Wappalyzer indicated that the application used Nginx, native PHP, Bootstrap, jQuery, and a MySQL 5.7 or later database environment. These results informed the

subsequent focus on server-side input validation, SQL query handling, authentication, session management, and file-upload security. The database fingerprint was further corroborated during the SQL injection exploitation described in Subsection 4.2.1. SQLMap independently identified MySQL as the back-end database management system, supporting the technology profile obtained during the initial information-gathering phase.

Accordingly, the information-gathering results were validated through two complementary procedures. First, each relevant result was manually verified by directly accessing the identified service, endpoint, directory, or file and examining the returned content. Second, selected findings were corroborated during subsequent testing when they successfully supported vulnerability exploitation. For example, the MySQL fingerprint was confirmed through SQL injection testing, while the existence and security relevance of the /uploads/ directory were demonstrated through unrestricted file upload and remote code execution. This validation process distinguishes confirmed attack-surface information from unverified scanner output and establishes a traceable connection between reconnaissance findings and the vulnerabilities reported in the following sections.

3.2. Testing Implementation and Vulnerability Analysis

The penetration testing process identified four verified vulnerabilities. Each finding was validated through Proof of Concept (PoC) to confirm exploitability and reduce the possibility of false positives. Unlike assessments that rely primarily on automated scanner outputs, this study combines automated vulnerability detection with manual verification to distinguish reproducible security weaknesses from potential false positives. The validated findings identify weaknesses across several security domains, including input validation, session management, file-upload handling, and authentication controls. The occurrence of vulnerabilities across these interconnected domains indicates a systemic problem in secure coding, application configuration, and server-side control implementation rather than a set of unrelated programming errors. This cross-domain analysis adds value by showing that remediation should not address each vulnerability independently but should include broader improvements in secure development practices, configuration management, authentication design, and security verification procedures.

3.2.1. SQL Injection in the False Positive Alert Feature.

The vulnerability resulted from inadequate server-side validation and the absence of parameterized queries. Beyond unauthorized data access, SQL Injection can facilitate data manipulation and support further attacks through credential extraction. The exposure of MD5 hashes increases the likelihood of offline password cracking. SQL Injection was identified in the POST /pages/alert_form.php endpoint through the alert_title, rule_id, and reason parameters. The application incorporated user input directly into SQL queries without prepared statements or parameter binding. Using SQLMap, the tester successfully enumerated the webapp_magang database, identified six tables, and extracted administrator credentials, including MD5 password hashes.

Directory enumeration identified several accessible paths, including /.git/, /.git/config, /uploads/, /pages/, and /README.md. The exposed /.git/ directory was the most critical finding because it could enable repository reconstruction or source-code disclosure. Technology fingerprinting further revealed the use of Nginx, PHP Native, Bootstrap, jQuery, and MySQL, indicating that testing should focus on input validation, session management, SQL query handling, and file upload security. Because exploitation was successful using publicly available tools and resulted in sensitive data exposure, this vulnerability was classified as a high-impact threat. The finding reinforces the importance of secure database interaction mechanisms and direct validation of suspected vulnerabilities.

3.2.2. Blind Stored XSS - Account Takeover Administrator.

A Blind Stored Cross-Site Scripting (XSS) vulnerability has been detected in the Full Name field of the Edit Profile feature. The injected JavaScript payload is stored in the database and is executed in the administrator's browser each time the dashboard is accessed. By employing an external XSS listener, the PHPSESSID cookie of the administrator was successfully exfiltrated, leading to a complete account

takeover without requiring knowledge of the password. Applications that do not implement output encoding and the HTTPONLY flag on session cookies are demonstrably vulnerable to session hijacking via XSS. This scenario is particularly hazardous due to the passive nature of the attack, wherein the attacker merely awaits the administrator's access to the dashboard. Table 3 provides more details on the session management testing using XSS.

Table 3. Vulnerability Assessment Results Using XSS (Cross-Site Scripting)

No.	Attribute	Detail
1	Endpoint	POST /pages/edit_profile.php
2	Vulnerable Parameters	Full name (User name column)
3	Attack Type	Blind Stored XSS → Session Hijacking → Account Takeover
4	CVSS v3.1 score	9.0 (Critical) - CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:H/I:H/A:H
5	WSTG References	WSTG – INPV – 02 – Testing for stored XSS | WSTG – SESS – 03
6	OWASP Top 10	A03:2021 – Injection (OWASP Foundation)

3.2.3. Unrestricted File Upload - Remote Code Execution.

Unrestricted File Upload was found in the profile photo upload feature at the POST /pages/update_profile.php endpoint. There is no extension, MIME type, or file content validation on the server side. By renaming the file from shell_rce.jpeg to shell_rce.php while keeping the Content-Type as image/jpeg using Burp Suite Intercept, a PHP web shell was successfully uploaded. Accessing the file via browser confirmed Remote Code Execution, allowing the attacker to execute arbitrary operating system commands. Client-side extension validation alone is insufficient; server-side MIME type validation based on magic bytes is a minimum mandatory control. The contribution of this finding lies in showing how the absence of layered upload controls can convert a file-upload weakness into remote code execution within the evaluated application. Although Wichmann, Groddeck and Federrath [6] also identified risks associated with inadequate file validation, the present study provides application-specific evidence that remediation must combine server-side extension allowlisting, content inspection using file signatures, randomized file naming, storage outside the web root, restricted execution permissions, and post-upload security verification.

3.2.4. Brute Force Attack of the Form Login.

The login form located at the POST /pages/login.php endpoint lacks a mechanism to limit authentication attempts. Utilizing Burp Suite Intruder with the Cluster Bomb type, 100 credential combinations (comprising 10 emails and 10 passwords) were transmitted without any requests being blocked or delayed. The distinction between HTTP response codes 302 (indicating success) and 200 (indicating failure) facilitates the automatic validation of attack outcomes. The absence of rate limiting is the primary factor rendering the login susceptible to credential stuffing and brute force attacks. This finding is similar to the prior study finding [15]. Table 4 refers to the vulnerability assessment result using brute force attack for authentication management (on login page).

Table 4. Vulnerability Assessment Results Using Brute Force Attack

No.	Attribute	Detail
1	Endpoint	POST /pages/login.php
2	Vulnerable Parameters	Email, password
3	Attack Type	Credential Brute Force – Cluster Bomb Attack
4	Number of Tested Request	100 combination (10 email x 10 passwords

No.	Attribute	Detail
5	CVSS v3.1 Score	7.5 (High) CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
6	WSTG References	WSTG – ATHN – 03 – Testing for Weak Lock Out Mechanism

3.3. Assessment Using CVSS v3.1

Each identified vulnerability was assessed using the Common Vulnerability Scoring System (CVSS) version 3.1 to quantify severity consistently and objectively [12]. The assessment results are presented in Table 5.

Table 5. Vulnerability Assessment Results Using CVSS v3.1

No.	Vulnerability	CVSS Score	Severity	Vector String
1	SQL Injection	8.8	High	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
2	Blind Stored XSS	9.0	Critical	CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:H/I:H/A:H
3	File Upload / RCE	9.9	Critical	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H
4	Brute Force Attack	7.5	High	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Two vulnerabilities have been classified as Critical: Unrestricted File Upload/Remote Code Execution (9.9) and Blind Stored Cross-Site Scripting (9.0). Additionally, two vulnerabilities have been identified as High: SQL Injection (8.8) and Brute Force Attack (7.5). The absence of vulnerabilities with Medium or Low severity underscores the necessity for immediate remediation of all identified vulnerabilities within this application.

3.4. Risk Assessment Based on NIST SP 800-30

Following the identification and exploitation of all vulnerabilities through technical testing, a risk assessment was conducted utilizing the NIST SP 800-30 framework, employing a three-level quantitative scale (1=Low, 2=Moderate, 3=High). The CVSS scores were considered in determining the Impact value, thereby enhancing the accuracy of the risk assessment [21]. Table 6 presents the results of the risk score assessment.

Table 6. Risk Assessment Based on NIST SP 800-30

No.	Vulnerability	Likelihood Level		Impact Level		Risk Level	
		Description	Score	Description	Score	Score	Description
1	SQL Injection	High	3	High	3	9	High
2	Blind Stored XSS	High	3	High	3	9	High
3	File Upload / RCE	High	3	High	3	9	High
4	Brute Force Attack	High	3	Moderate	2	6	Moderate

The vulnerabilities of SQL Injection, Blind Stored XSS, and File Upload/RCE have been assigned a Likelihood rating of High (3) due to their potential for remote exploitation by authenticated users utilizing freely accessible tools. These vulnerabilities also received an Impact rating of High (3), which similar to their Critical and High CVSS scores, as the resultant damage is notably severe, encompassing sensitive data leakage, administrator account takeover, and complete server control. In contrast, Brute Force has been assigned an Impact rating of Moderate (2) because its consequences are confined to account compromise without directly resulting in RCE.

3.5. Risk Treatment Plan

Following the results of the risk assessment, a Risk Treatment Plan (RTP) was formulated, prioritizing actions based on the highest levels of risk. The technical remediation recommendations for each identified vulnerability are as follows: (1) SQL Injection: implement prepared statements using PDO/mysqli_prepare, enforce whitelist input validation, apply the principle of least privilege to the database account, and deploy a Web Application Firewall (WAF) with SQL injection rules; (2) Blind Stored Cross-Site Scripting (XSS): implement output encoding using htmlspecialchars(), add a Content Security Policy header, enable HttpOnly and Secure flags on session cookies, and deploy a WAF with XSS rules; (3) Unrestricted File Upload: enforce server-side whitelist extension validation, validate MIME types using magic bytes, configure php_flag engine off in the /uploads/ directory, and rename files with random hashes; (4) Brute Force Attack: implement Nginx rate limiting (maximum of 5 attempts per minute per IP), enforce a 15-minute account lockout after 5 failed attempts, integrate Google reCAPTCHA v3, and implement Multi-Factor Authentication (MFA). Table 7 shows the risk treatment plan for the following risk management result.

Table 7. Risk Treatment Plan

No.	Vulnerabilities	Risk Level	Recommended Control	Expected Effect	Residual Risk
1	SQL Injection	High	Implement prepared statements using PDO or mysqli_prepare, enforce whitelist input validation, apply least privilege to the database account, and deploy WAF rules for SQL Injection	Prevent query manipulation and reduce database exposure	Low
2	Blind Stored XSS	High	Implement output encoding using htmlspecialchars(), apply Content Security Policy, enable HttpOnly and Secure cookie flags, and deploy WAF rules for XSS	Prevent script execution and reduce session theft risk	Low
3	File Upload / RCE	High	Enforce server-side whitelist extension validation, validate MIME type using magic bytes, rename files using random hashes, and configure the upload directory as non-executable	Prevent malicious file upload and server-side code execution	Low
4	Brute Force Attack	Moderate	Implement Nginx rate limiting, temporary account lockout, CAPTCHA, and Multi-Factor Authentication	Reduce automated credential guessing and credential stuffing risk	Low
5	SQL Injection	High	Implement prepared statements using PDO or mysqli_prepare, enforce whitelist input validation, apply least privilege to the database account, and deploy WAF rules for SQL Injection	Prevent query manipulation and reduce database exposure	Low

3.6. Continuous Monitoring and Residual Risk

After controls are applied, continuous monitoring is conducted to ensure their effectiveness in accordance with the fourth stage of NIST SP 800-30 (Maintain the Assessment). Table 8 presents the projected residual risk status after all controls are fully implemented.

Table 8. Residual Risk Status After Controls Are Applied

No.	Risk Statement	Controls Applied	Likelihood	Impact	Residual Risk Status
1	Database leakage via SQL Injection	Parameterized query, WAF, least privilege DB	1	3	Low
2	Account takeover via Stored XSS	Output encoding, CSP header, HttpOnly cookie	1	2	Low
3	RCE via File Upload	Extension whitelist, MIME validation, non-executable directories	1	3	Low
4	Brute Force on the login form	Rate limiting, account lockout, CAPTCHA, MFA	1	2	Low

All residual scores are situated within the range of 2 to 3, categorizing them as Low on the employed 1-9 scale. This outcome suggests that the implemented controls successfully diminished the Likelihood from 3 to 1, while the Impact score remained constant, as the potential damage in the event of a successful exploit cannot be entirely eradicated. Consequently, this residual risk is below the established risk tolerance threshold and is considered acceptable.

3.7. Summary and Overall Findings Discussion

The analysis identifies a recurring root cause across the findings: inadequate server-side input validation, output encoding, and sanitization. SQL injections occurred because the application processed untrusted input within database queries, whereas blind stored cross-site scripting occurred because stored user-controlled content was later rendered without adequate contextual output encoding. These findings indicate that secure coding controls were not applied consistently across the application and place the identified weaknesses within the OWASP Top 10 A03:2021 Injection category.

The risk becomes more significant when the vulnerabilities are considered as a combined attack path rather than as isolated findings. Unrestricted file upload leading to remote code execution received the highest CVSS score of 9.9 because it enabled arbitrary command execution on the server. Blind stored cross-site scripting, with a CVSS score of 9.0, could compromise a privileged user session, while SQL injection, scored at 8.8, exposed sensitive database contents. Together, these weaknesses create a potential attack chain in which an attacker can obtain application data, compromise privileged access, and subsequently execute commands on the underlying server. The added value of this analysis lies in demonstrating how vulnerabilities originating from different application components can interact and produce a level of organizational risk that is greater than the impact suggested by evaluating each weakness separately. This application-specific evidence extends the observation of Raihan, Prabowo and Oktaria [21] regarding the risk escalation produced by combined web-application vulnerabilities. Table 9 summarizes all vulnerabilities that were identified, validated, and exploited during the assessment.

Table 9. Summary of All Vulnerability Findings

No.	Vulnerability	CVSS	Severity	Risk Level (NIST)	WSTG Phase	Likelihood	Impact	Status
1	SQL Injection (Time-Based Blind)	8.8	High	High (9)	WSTG-INPV-05	1	3	Low
2	Blind Stored XSS + Session Hijacking	9.0	Critical	High (9)	WSTG-INPV-02, WSTG-SESS-03	1	2	Low
3	Unrestricted File Upload	9.9	Critical	High (9)	WSTG-	1	3	Low

No.	Vulnerability	CVSS	Severity	Risk Level (NIST)	WSTG Phase	Likelihood	Impact	Status
	+ RCE				CONF-10			
4	Brute Force Attack on Login	7.5	High	Moderate (6)	WSTG-ATHN-03	1	2	Low

A key finding is that penetration testing alone is insufficient for security decisions unless integrated into a risk management framework. OWASP WSTG-based testing identifies technical vulnerabilities, as shown by Priambodo, Rifansyah and Hasbi [12], Tinambunan, Junaidi and Rizki [13], Rozali and Sinaga [14]. However, it mainly checks exploitability, not prioritization, responsibility, or residual risk. This study found all four vulnerabilities exploitable, but risk implications varied. For example, a Brute Force Attack had High CVSS severity but a Moderate NIST risk level due to its narrower impact compared to Remote Code Execution.

Integrating NIST SP 800-30 into penetration testing provides the analytical structure needed to translate technical evidence into likelihood, impact, risk level, and treatment priority. While OWASP WSTG guides the security-testing process and CVSS quantifies the technical severity of identified vulnerabilities, NIST SP 800-30 contextualizes these findings according to the affected assets, relevant threat sources and events, existing controls, and potential organizational consequences. Previous studies [8, 18] have demonstrated the usefulness of NIST SP 800-30 for structured threat identification and risk estimation. Raihan, Prabowo and Oktaria [21] further showed that combining OWASP, vulnerability assessment and penetration testing, CVSS, and NIST SP 800-30 can produce a more systematic and quantifiable risk evaluation. In addition, El Marzak, Chahid, Faris and Mansouri [16], Monageng and Esiefarienrhe [22] emphasize that security assessment should extend beyond vulnerability identification by considering mitigation status, continuous monitoring, and residual risk.

The principal contribution of this study is the transformation of penetration-testing results from a list of technical vulnerabilities into a structured basis for risk-based security planning. The process begins with OWASP WSTG-based testing, which produces reproducible proof-of-concept evidence for each identified vulnerability. CVSS is then used to estimate technical severity, while NIST SP 800-30 extends the analysis by considering the affected asset, relevant threat scenario, likelihood of exploitation, existing controls, and potential organizational impact. Through these connected stages, each validated vulnerability is converted into a Risk Treatment Plan containing its risk level, remediation priority, recommended control, responsible party, monitoring requirement, and projected residual risk.

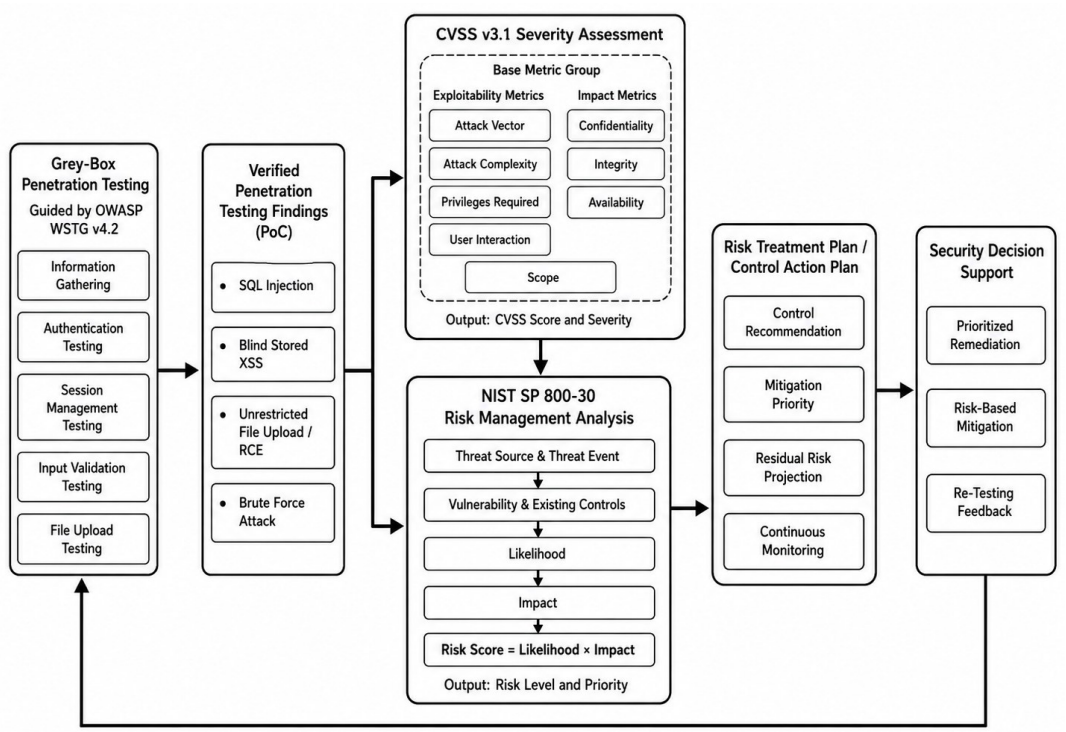

Figure 3. Integrating penetration Testing into Risk Management Framework NIST 800-30

Figure 3 illustrates the transformation of penetration testing results into structured risk management outputs. Overall, the process commences with grey-box testing, guided by OWASP WSTG v4.2, encompassing areas such as information gathering, authentication, session management, input validation, and file upload testing. The process continues with the definition of the authorized testing scope and attack-surface identification. Potential vulnerabilities are then tested and technically validated before being mapped to affected assets and relevant threat scenarios. Likelihood and impact analysis produces the risk level, treatment priority, recommended corrective action, and subsequent verification-testing requirement. Verified findings, including SQL Injection, Blind Stored XSS, Unrestricted File Upload/RCE, and Brute Force Attack, serve as inputs for the subsequent assessment stage. Each finding is evaluated using the CVSS v3.1 Base Metric Group, which measures exploitability and impact. The CVSS score provides a technical severity value that supports risk analysis.

Lastly, the result of the penetration testing will become the input for the NIST SP 800-30 risk management analysis. The integration should mapping vulnerabilities to threat sources, events, existing controls, likelihood, and impact. The risk score, calculated using a likelihood and impact matrix, yields a risk level and remediation priority. This evaluation forms the basis for a Risk Treatment Plan or Control Action Plan, which includes control recommendations, mitigation priorities, residual risk projection, and continuous monitoring. The final output supports security decision-making by linking technical evidence with risk-based mitigation planning and retesting feedback.

The integration of penetration testing to the Risk Management framework changes the role of penetration testing from a primarily diagnostic activity into a decision-support process. Developers can use the results to identify the controls that require correction, security analysts can monitor whether the implemented treatments reduce exposure, and decision-makers can prioritize resources according to organizational risk rather than technical severity alone. The assessment therefore continues beyond vulnerability discovery by establishing what should be remediate first, who should be responsible, how the treatment should be verified, and whether the remaining risk can be accepted. In this way, the integration of OWASP WSTG, CVSS, and NIST SP 800-30 provides a traceable connection between technical evidence, remediation decisions, and ongoing risk management.

4. Conclusion

This study examined three research questions about the security of an intern management web application used by a web security company. First, grey-box penetration testing found four vulnerabilities: Time-Based Blind SQL Injection in the False Positive Alert, Blind Stored XSS in Edit Profile, Unrestricted File Upload leading to Remote Code Execution in profile photo upload, and Brute Force Attack on the login form. These suggest weaknesses in input validation, session management, file upload control, and authentication protection, reflecting broader server-side security and coding issues.

Second, CVSS v3.1 severity assessment showed all vulnerabilities need urgent remediation. Unrestricted File Upload/RCE scored 9.9 (Critical), Blind Stored XSS 9.0 (Critical), SQL Injection 8.8 (High), and Brute Force Attack 7.5 (High). Proof-of-Concept evidence, including database enumeration and administrator credential extraction, confirmed technical exploitability. Lastly, integrating penetration testing with NIST SP 800-30 provided a comprehensive risk assessment, transforming technical exploitability into risk-based decision-making. NIST-based assessment rated SQL Injection, Blind Stored XSS, and File Upload/RCE as High risks (score 9), and Brute Force Attack as Moderate (score 6). This shows technical severity and organizational risk differ. By combining Proof-of-Concept evidence, CVSS scores, likelihood, and impact, the study developed a Risk Treatment Plan prioritizing remediation and projecting residual risk post-control. Proposed controls reduced all residual risks to Low, demonstrating the integrated approach's effectiveness in actionable security improvement.

This study contributes to both research and practice. Research-wise, it offers a workflow linking OWASP WSTG-based penetration testing, CVSS v3.1 severity assessment, and NIST SP 800-30 risk management into one model. This extends prior studies focusing on vulnerability categories and recommendations by adding risk scoring, treatment planning, and residual risk projection. Practically, it provides a reference for web security companies, analysts, interns, and developers to convert technical findings into prioritized mitigation decisions. It also supports secure coding practices within the Software Development Lifecycle, especially for cybersecurity training and internal management applications.

This study has two limitations. First, the assessment was on a single intern management web application in a controlled environment, so findings may not represent all web applications or production environments. Future studies should apply the framework to multiple applications, organizational contexts, and complex environments to evaluate generalizability. Second, the testing scope focused on four vulnerability categories: SQL Injection, Blind Stored XSS, Unrestricted File Upload/RCE, and Brute Force Attack. Future research should extend to other risks, such as Broken Access Control, Insecure Direct Object References, Cross-Site Request Forgery, API security weaknesses, and insecure deserialization. Future studies should also validate proposed controls through post-remediation retesting, continuous monitoring data, and comparison with black-box, white-box, or automated vulnerability assessment approaches.

5. References

[1] Safitra, M.F., Lubis, M., and Widjajarto, A.: ‘Security vulnerability analysis using penetration testing execution standard (PTES): case study of government's website’, in Editor (Ed.)^(Eds.): ‘Book Security vulnerability analysis using penetration testing execution standard (PTES): case study of government's website’ (2023, edn.), pp. 139-145

[2] Wibowo, R.M., and Sulaksono, A.: ‘Web vulnerability through Cross Site Scripting (XSS) detection with OWASP security shepherd’, Indonesian Journal of Information Systems, 2021, 3, (2), pp. 149-159

[3] Egbedion, G.E.: ‘Impact of vulnerability management and penetration testing on security-informed IT project planning and implementation’, Journal of Multidisciplinary Engineering Science and Technology (JMEST), 2024

[4] Isnaini, K., Asyari, M.H., Amrillah, S.F., and Suhartono, D.: ‘Vulnerability Assessment and Penetration Testing on Student Service Center System’, ILKOM Jurnal Ilmiah, 2024, 16, (2), pp. 161-171

[5] Ajufo, G., and Qutieshat, A.: ‘An examination of the human factors in cybersecurity: Future direction for Nigerian banks’, Indonesian Journal of Information Systems, 2023, 6, (1), pp. 1-16

[6] Wichmann, P., Groddeck, A., and Federrath, H.: 'Fileuploadchecker: detecting and sanitizing malicious file uploads in web applications at the request level', in Editor (Ed.)^(Eds.): 'Book Fileuploadchecker: detecting and sanitizing malicious file uploads in web applications at the request level' (2022, edn.), pp. 1-10
[7] Lubis, M., Luthfi, M.I., Saedudin, R.R., Muttaqin, A.N., and Lubis, A.R.: 'The Integration of ISO 27005 and NIST SP 800-30 for Security Operation Center (SOC) Framework Effectiveness in the Non-Bank Financial Industry', Computers, 2026, 15, (1), pp. 60
[8] Pambudi, R.D., and Ramli, K.: 'Information Security Risk Management Design of Supervision Management Information System At Xyz Ministry Using Nist Sp 800-30', Jurnal Teknik Informatika (Jutif), 2023, 4, (3), pp. 591-599
[9] Wang, J.-C., Hong, Y.-J., Sanjaya, E., and Santoso, H.B.: 'Engaging or Silent? Social Media Commenting on Controversial versus Uncontroversial Issues', Pacific Asia Journal of the Association for Information Systems, 2024, 17, (2), pp. 1-7
[10] Wang, J.-C., Hung, Y.-C., and Santoso, H.B.: 'How ESL Devices Transform into Connected Label Solutions: A Perspective of Actor Interaction and Information Rebundling', NTU Management Review, 2024, 34, (3), pp. 229-286
[11] Windasari, N.A., and Santoso, H.B.: 'Multichannel Retailing in Beauty Product: Understanding Customer Purchase Decisions between Offline Stores, Websites, and Augmented Reality', Jurnal Sistem Informasi, 2022, 18, (2), pp. 50-67
[12] Priambodo, D.F., Rifansyah, A.D., and Hasbi, M.: 'Web XYZ Penetration Testing using OWASP Risk Rating [*Penetration testing Web XYZ berdasarkan OWASP risk rating*]', Teknika, 2023, 12, (1), pp. 33-46
[13] Tinambunan, F., Junaidi, A., and Rizki, A.M.: 'Academic Information Testing of the University X using Penetration Testing based on Owasp Top 10 [*Pengujian Sistem Informasi Akademik Universitas X Melalui Pendekatan Penetration Testing Berdasarkan Owasp Top 10*]', JATI (Jurnal Mahasiswa Teknik Informatika), 2024, 8, (1), pp. 1062-1069
[14] Rozali, M., and Sinaga, M.D.: 'Web Security Diagnosis using Penetration Testing of a School Website [*Diagnosis Keamanan Web Menggunakan Metode Uji Penetrasi Website Sekolah*]', Jurnal Info Digit (JID), 2024, 2, (1), pp. 246-262
[15] Lina, I.M.: 'Anticipate password security with burp suite using the brute force attack method', Jurnal E-Komtek, 2023, 7, (1), pp. 118-127
[16] El Marzak, Y., Chahid, A., Faris, S., and Mansouri, K.: 'Developing a Unified Cyber Risk Management Framework Using Semantic Technologies and Structured Modeling Approaches', Engineering, Technology & Applied Science Research, 2026, 16, (1), pp. 31043-31051
[17] Mızrak, F.: 'Integrating cybersecurity risk management into strategic management: a comprehensive literature review', Research Journal of Business and Management, 2023, 10, (3), pp. 98-108
[18] Syafitri, W.: 'Information Security Risk Assessment using NIST 800-30 (Case Study: Academic Information Systes University XYZ) [*Penilaian Risiko Keamanan Informasi Menggunakan Metode NIST 800-30 (Studi Kasus: Sistem Informasi Akademik Universitas XYZ)*]', Jurnal CoreIT, 2016, 2, (2), pp. 8-13
[19] Kurniawan, D.K.C., and Sihotang, J.I.: 'NIST 800-30 Risk Assessment Audit for Academic Excellence: A Roadmap for Strengthening Cybersecurity at Universitas Advent Indonesia', TeIKa, 2025, 15, (2), pp. 63-77
[20] Prasetyo, B., Salsabila, A., and Retnani, W.E.Y.: 'IT Risk Management Analysis Based on ISO 31000 and Bow Tie Analysis (BTA) in Higher Education Institution', Indonesian Journal of Information Systems, 2024, 7, (1), pp. 84-96
[21] Raihan, S.D., Prabowo, S., and Oktaria, D.: 'Security Vulnerable Evaluation of a Web Application with Vulnerability Assessment and Penetration Testing using OWASP and NIST SP 800-30 Revision 1 [*Evaluasi Risiko Celah Keamanan Pada Aplikasi Web Dengan Penilaian Kerentanan dan Pengujian Penetrasi Menggunakan Metode OWASP dan NIST SP 800-30 Revisi 1*]', LOGIC: Jurnal Penelitian Informatika, 2024, 2, (2)

[22] Monageng, T., and Esiefarienrhe, B.M.: 'Analysis of Risk Assessment Framework Using Agile Methodology and Computer-Aided Software Engineering Tools', Indonesian Journal of Information Systems, 2026, 8, (2), pp. 157-174
[23] Acharya, S., and Pandya, V.: 'Bridge between black Box and white Box–gray Box testing technique', International Journal of Electronics and Computer Science Engineering, 2012, 2, (1), pp. 175-185
[24] Muna, S.R., Santoso, H.B., and Siang, J.J.: 'Internal Compliance Audit of the Information Security Management System in a Cybersecurity Company based on ISO/IEC 27001', Sistemasi: Jurnal Sistem Informasi, 2026, 15, (6), pp. 2267-2278
[25] Dewanti, A.C., Santoso, H.B., and Siang, J.J.: 'ISO 27001 Annex A Compliance Analysis of a Cybersecurity Company [*Analisis Kepatuhan ISO 27001 Annex A pada Perusahaan Keamanan Siber*]', Jutisi: Jurnal Ilmiah Teknik Informatika dan Sistem Informasi, 2026, 15, (3), pp. 996-1007