



RESEARCH ARTICLE

Gap Audit and ISO/IEC 27001:2022 Control Implementation for Improving the Control Conformity of a Web-Based Local Archive Digitization System

M. Ilham Zidni Syauquillah¹, Moh. Sukron^{1,*}, Ahmad Halimi¹

¹Department of Information Technology, Faculty of Engineering, Universitas Nurul Jadid, Probolinggo, Indonesia

*Corresponding author: sukron@unuja.ac.id

ARTICLE INFO

Article history:

Received: 02 November 2026

Revised: 18 November 2026

Accepted: 30 November 2026

Published: 16 December 2026

Keywords:

Information security audit

Annex A.8 technological controls

Penetration testing

Static application security testing

Laravel

Risk assessment

ABSTRACT

Local governments increasingly rely on web-based systems to manage digital archives, yet such systems are often deployed without a structured security evaluation. This study audits a Laravel-based local archive digitization system at the Library and Archives Office of Probolinggo City, Indonesia, before wider operational use. A mixed-methods case study proceeded through baseline assessment, risk analysis, remediation, and re-evaluation with identical instruments. The baseline combined a gap analysis against 13 technological controls of ISO/IEC 27001:2022 Annex A.8, grey-box penetration testing (manual testing and OWASP ZAP), and static analysis of source code and dependencies. Baseline conformity was 50% (13 of 26 points), and eight findings were identified: two high-risk, five medium-risk, and one low-risk. The most critical were archive files reachable by URL without authentication (risk score 25) and inconsistent authorization on data-management functions (risk score 16). Remediation comprised private storage with signed URLs, policy-based authorization, threat monitoring with automatic blocking and alerts, field-level audit logging, soft deletion with restore, and security headers. After remediation, conformity rose to 96.15% (25 of 26 points), twelve controls were fully conformant, dynamic testing reported no medium or higher alerts, and an archivist's verification matched the expected result in 33 of 34 feature scenarios. The only residual risk concerns development-branch dependencies. Conformity measures the implementation of selected application-level controls, not resistance to real attacks, and scoring was performed by the study team in an isolated environment. Within that scope, the closed-loop audit gives a repeatable account of control improvement before deployment.

I. INTRODUCTION

Digital transformation has led local governments to adopt web-based information systems for administration and internal services, including the management of archives and digital documents. These systems improve efficiency, data integration, and user access, but they also increase exposure to unauthorized access, misuse of user privileges, and disruptions to system reliability [1]. The exposure is visible at the national level: Indonesia's National Cyber and Crypto Agency (BSSN) recorded roughly 3.64 billion traffic anomalies between January and July 2025 across sectors that include government and public services [2],[3],[4]. Not every anomaly is a direct attack on a web application, and the figure does not describe the security condition of web-based archive systems in local government specifically [5]. Nevertheless, it shows how exposed public-sector information systems are to cyber threats.

In practice, information systems in government are often built to satisfy functional requirements and operational continuity. Security receives less attention once a system is in routine use, and periodic security evaluation tends to be neglected even as requirements and usage complexity grow, so vulnerabilities remain undetected and endanger data protection and the continuity of administrative services [6]. Information security rests on protecting confidentiality, integrity, and availability [7], and web-based systems face attacks such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF) [8], [9]. Modern web frameworks ship with basic security features, but security is frequently limited to application-level techniques without structured information security management and continuous evaluation. The resulting mechanisms do not necessarily match the operational context and risk level of the system, which is a concern for digital archives that hold sensitive records [10]. Archives also serve as administrative and legal evidence, so loss or alteration of their content directly affects the validity of an organization's information [11].

This study examines such a system: a web-based local archive digitization system (*Sistem Alih Media Arsip Lokal*) developed by the Library and Archives Office (*Dinas Perpustakaan dan Kearsipan*) of Probolinggo City, Indonesia, to manage digital archives internally. The system is built with the Laravel framework and uses authentication as its basic

safeguard, but it had never been evaluated against a structured information security standard, so the conformity of its security controls was unknown. It was still operated in a local environment and was not yet exposed to public networks, which made a preventive, pre-deployment evaluation both possible and timely.

Security evaluations of web-based information systems have taken several approaches. One line of work examines the built-in security features of application frameworks and finds that proper configuration of a framework such as Laravel improves protection [10]. Another applies ISO/IEC 27001 to evaluate information systems and reports weaknesses in several implemented controls [5]; a gap analysis based on ISO/IEC 27001:2022 on an academic information system reported control conformity of about 68–70% [12]. Studies on the design of an information security management system (ISMS) under ISO/IEC 27001:2022 [13] and on risk-based information security management [14] show that the standard can guide governance in government agencies. Most of these studies stop at analysis or design, without implementing corrective measures and re-evaluating the improved system. In addition, the combination of an ISO/IEC 27001 control assessment, penetration testing, and static application security testing (SAST) is rarely reported in the studies reviewed here.

This paper therefore performs a closed-loop security evaluation. A security audit is a systematic assessment of how effectively implemented controls protect a system, and it matters most for running systems whose condition changes over time [15]. The audit here consists of a baseline gap analysis against ISO/IEC 27001:2022, risk-based prioritization, implementation of countermeasures, and re-evaluation with the same instruments. It addresses three research questions: (RQ1) Which security control gaps exist in the system when compared with ISO/IEC 27001:2022? (RQ2) How can the identified gaps be closed through the design and implementation of security improvements? (RQ3) To what extent do the evaluated controls conform after remediation, and what remains to be addressed before wider operational use?

The main contributions of this paper are as follows:

- A gap audit of a Laravel-based archive digitization system against 13 technological controls of ISO/IEC 27001:2022 Annex A.8, triangulated with grey-box penetration testing (manual testing and OWASP ZAP) and static analysis of source code and dependencies (Vanguard).
- A likelihood–impact risk assessment that converts eight verified findings into a prioritized remediation sequence, with false positives and accepted risks explicitly screened out.
- A set of implemented countermeasures mapped to the audited controls: private storage with signed URLs, policy-based authorization, threat monitoring with automatic blocking and alerting, field-level audit logging, soft deletion with restore and retention, and a content security policy with security headers.
- A before-and-after evaluation using identical instruments, showing control conformity rising from 50% to 96.15% and no medium-or-higher alerts remaining in dynamic testing, together with an explicit statement of what this indicator does and does not measure.

The remainder of this paper is organized as follows. Section II reviews related work, Section III describes the materials and methods, Section IV presents the results, Section V discusses them, and Section VI concludes.

II. RELATED WORK

Related studies fall into three groups: framework-level security analysis, standard-based security evaluation, and ISMS design and risk management. Table 1 summarizes the studies most relevant to this work.

Table 1. Summary of related studies and the gap addressed by this study

Ref.	Focus	Limitation	Difference in this study
[10]	Security features of Laravel against SQL injection and XSS	No standard-based audit; conformity to a structured framework not measured	Audit and control implementation based on ISO/IEC 27001:2022
[5]	Data security evaluation with ISO/IEC 27001	Evaluation only; no remediation	Adds implementation of remediation
[12]	Gap analysis of an academic information system with ISO/IEC 27001:2022	No implementation of recommendations and no re-test	Implements the audit results and re-evaluates the system
[13]	ISMS design for a data center based on ISO/IEC 27001:2022	Design stage only; no technical testing of a running web application	Implements and evaluates controls directly on an operational web application
[14]	ISO/IEC 27001:2022 in organization-level risk management	No evaluation or testing at the web-application level	Extends to control implementation and direct security testing of a web application

Husain et al. [10] examined the security mechanisms of Laravel by testing its built-in features against common attacks such as SQL injection and XSS. Laravel’s defaults proved reasonably strong, but additional configuration was needed to make protection effective. The study did not use a standard-based audit and therefore could not measure conformity with a structured security framework.

Standard-based evaluations provide that structure. Siregar and Mardiah [5] evaluated data security in an organizational information system using ISO/IEC 27001 and found a medium level of implementation, with weaknesses in access

management and data backup. Izzani et al. [12] applied gap analysis based on ISO/IEC 27001:2022 to an academic information system and reported conformity of 68–70%. Both stopped at evaluation, so neither shows whether the recommended measures actually improve security. Gap analysis itself identifies the difference between the actual state of a system and the state expected by a standard, and its results guide prioritization and mitigation [16].

At the governance level, Alberto and Karyati [13] designed an ISMS under ISO/IEC 27001:2022 for a government data center, which yielded a structured security design but no implementation or testing on a running application. Arumdiya and Rudianto [14] emphasized the risk-based approach to selecting controls at the organizational level, without direct evaluation of web applications. Risk-based selection of controls is a core principle of the standard [17].

Technical testing methods complement these evaluations. Penetration testing simulates attacks to find vulnerabilities that other methods miss [18], OWASP ZAP supports dynamic testing of web applications [19], and SAST analyzes source code for weaknesses referenced to the OWASP Top 10 and the Common Weakness Enumeration (CWE) [20]. Threat detection mechanisms have also been proposed for web attacks [9] and for Laravel-based systems [8]. To our knowledge, none of the studies reviewed here combines a standard-based control assessment with dynamic and static testing, implements the resulting remediation, and re-evaluates the system. This study addresses that gap.

III. MATERIALS AND METHODS

A. Research Object and Design

The object of this study is the web-based local archive digitization system used by the Library and Archives Office of Probolinggo City. The system supports two functions. The first is internal archive management by two roles: users enter archive records, either manually or by importing a Microsoft Excel file, and each new record stays in a pending state until an administrator verifies it and accepts or rejects it. The second is a public search portal, which exposes only archives that a user marked as public and that an administrator verified; non-public archives are available to internal users only. At the time of the study the system had been developed and tested in a local environment as a pre-implementation stage before production deployment.

The study is a mixed-methods evaluative case study. The qualitative component is primary: it assesses security through direct observation of the system and analysis of how ISO/IEC 27001:2022 controls are applied. The quantitative component supports risk analysis by scoring the likelihood and impact of each vulnerability; these scores are indicative and are not used for statistical testing. The research proceeded through six stages (Fig. 1): preparation and initial observation, baseline evaluation, risk analysis and prioritization, implementation of improvements, re-evaluation with the same methods, and comparative analysis.

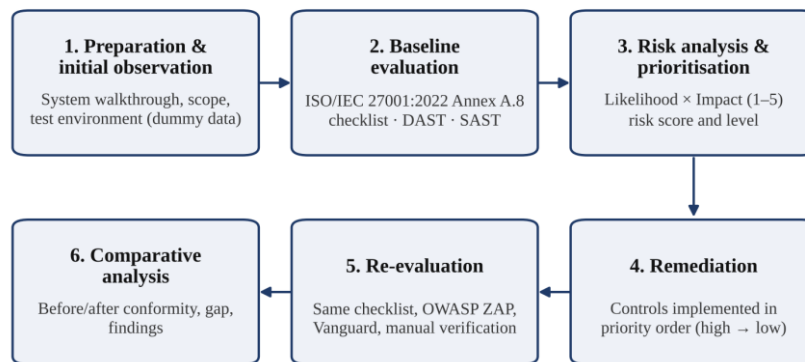


Fig. 1. Research workflow of the security audit, remediation, and re-evaluation.

B. Test Environment and Data

Testing was carried out in a local environment so that it would not disturb the real operational system. The system ran on Laragon as the local development environment, with PHP, MySQL, and Nginx, and Laravel ran in development mode. The environment was isolated from public networks to keep all testing activity controlled. All tests used dummy data that simulate real archives, so no operational data were exposed or altered. User roles were simulated according to their access rights in order to evaluate the access controls under different usage scenarios. Data were collected from direct observation of the system, the ISO/IEC 27001:2022 checklist, and the results of penetration testing and SAST. The versions of Laravel, PHP, MySQL, Nginx, OWASP ZAP, and Vanguard, and the ZAP scan mode, scope, and authentication context, were not recorded during the study, which limits exact reproduction.

C. Control Selection and Conformity Scoring

ISO/IEC 27001:2022 [21] specifies requirements for an ISMS using a risk-based approach and includes Annex A, a catalogue of information security controls [22]. This study does not assess full compliance with the standard. Selecting the subset of controls relevant to the system is an accepted way to apply it [23]. Controls were therefore chosen from

Annex A.8 (technological controls) for their relevance to a web-based archive application and to the test methods used: authentication, access control, input validation, secure configuration, logging, cryptography, and secure coding. Thirteen controls were selected (Table 2). Controls in A.5 (organizational), A.6 (people), and A.7 (physical) were excluded because they lie outside the technical scope of the application and require organizational policy and infrastructure access that was not available to the researchers. Control intent follows the implementation guidance of ISO/IEC 27002:2022 [24]. Of the 34 controls in A.8, 21 were not assessed. Most of them concern endpoint devices, capacity, malware protection, backup, redundancy, network security and segregation, and the separation of environments and change management, which need infrastructure or process evidence. A few that bear on the application itself were also left out: A.8.4 (access to source code), A.8.9 (configuration management), A.8.25 (secure development life cycle), and A.8.29 (security testing in development and acceptance). The selection therefore favors controls that the available test methods can evidence, and weaknesses in the unassessed controls would not appear in the results.

Table 2. ISO/IEC 27001:2022 Annex A.8 controls evaluated and their main evidence sources

Control	Name	Main evidence and test scenario
A.8.2	Privileged access rights	Manual testing: admin pages accessed with a regular user account; URL parameter manipulation
A.8.3	Information access restriction	Manual testing: URL parameter manipulation; direct access to archive files
A.8.5	Secure authentication	Manual testing: valid/invalid credentials, limited repeated attempts, access without session or token, session timeout and reuse after logout; OWASP ZAP
A.8.8	Management of technical vulnerabilities	OWASP ZAP scanning; review of vulnerability documentation
A.8.10	Information deletion	Manual testing: delete an archive, then check access to the deleted data
A.8.11	Data masking	Vanguard: password hashing and protection of sensitive data
A.8.12	Data leakage prevention	Manual testing: upload of files with invalid extensions; direct access to archive files
A.8.15	Logging	Manual testing: failed logins and unauthorized access, then log inspection
A.8.16	Monitoring activities	Manual testing: failed logins and unauthorized access, then monitoring inspection
A.8.24	Use of cryptography	Vanguard: password hashing and protection of sensitive data
A.8.26	Application security requirements	Vanguard scan of dependencies and source code
A.8.27	Secure system architecture	OWASP ZAP: error pages and response headers (disclosure of server or framework information)
A.8.28	Secure coding	Manual input-validation tests (special characters, SQL injection and XSS payloads); OWASP ZAP; Vanguard

Each control was rated with a checklist that has three categories: conformant (score 2), meaning the control is fully implemented and functions as intended; partially conformant (score 1), meaning it is only partly implemented or has remaining deficiencies; and not conformant (score 0), meaning it is not implemented or no evidence of implementation exists. The status of each control was determined from system observation, configuration analysis, penetration test results, SAST results, and implementation evidence. The conformity level and the security gap are computed as

$$\text{Conformity} = \left(\frac{S_{\text{actual}}}{S_{\text{max}}} \right) \times 100\% \quad (1)$$

$$\text{Gap} = 100\% - \text{Conformity} \quad (2)$$

where S_{actual} is the sum of the scores of all evaluated controls and S_{max} is the number of evaluated controls multiplied by the maximum score of 2, i.e., 26 for 13 controls. The gap is the difference between the achieved conformity and the ideal value of 100%. The score is presence-based: it records whether a control is implemented and functions as intended, and it weights all controls equally. It is therefore an indicator of control implementation, not a measure of resistance to attack. The checklist was designed and scored by the study team, and the same team scored the system again after remediation. A separate verification form completed by an archivist of the office (Section IV-F) offers a partial external check of the implemented features, but it is not an independent re-scoring of the checklist. The records of the study do not identify the individual scorer of each control, and no second assessor is reported.

D. Security Testing

Penetration testing was grey-box and non-destructive. The testers had access to the application, its structure, and its source code, but not to the server infrastructure or system-level configuration. This is more effective than black-box testing while still reflecting a realistic attack. The scope was limited to validating vulnerabilities, without exploiting real data or performing advanced attacks. The approach follows the planning, discovery, and attack phases described in NIST SP 800-115 [25], applied in a restricted, non-destructive form. Manual testing was the main method and simulated an unauthorized user through input-validation tests, direct access to resources, and parameter manipulation. OWASP ZAP,

an open-source dynamic application security testing (DAST) tool [19], [26], complemented the manual work with automated scanning.

Static analysis used Vanguard [27] to scan the source code and dependencies against rules based on the OWASP Top 10 and CWE. It covers coding practices, dependency usage, and potential misconfiguration. Eleven test scenarios mapped the controls to these methods (Table 2). After remediation, a separate verification form with 34 scenarios covering the implemented features was completed on 6 July 2026 by an archivist of the Library and Archives Office, who is not an author of this paper; each scenario states the feature, the test action, and the expected result, and the tester marked whether the observed result matched. Findings were recorded with their type, location, description, evidence, severity, and recommendation. Severity of dynamic-test findings was referenced to the Common Vulnerability Scoring System (CVSS) categories low, medium, high, and critical for technical description only; the risk level used for prioritization was calculated separately, as described next.

E. Risk Analysis

Findings from the checklist, OWASP ZAP, Vanguard, and manual verification were screened first. Findings verified as false positives, or without significant security impact, were not carried into risk assessment. For each remaining finding, likelihood (L) and impact (I) were rated on a scale from 1 (very low) to 5 (very high), and the risk score is

$$\text{Risk Score} = \text{Likelihood} \times \text{Impact} \quad (3)$$

Scores of 1–5 are classified as low (acceptable risk), 6–12 as medium (staged remediation), and 13–25 as high (immediate remediation). Remediation followed descending risk order.

F. Validity and Reliability

Validity was supported by triangulating three complementary methods—the standard-based checklist, dynamic testing, and static analysis—and by using recognized references (ISO/IEC 27001:2022, OWASP Top 10, CWE, and CVSS). Reliability was supported by documented test procedures, a fixed tool configuration throughout the study, and operational definitions for the checklist categories and risk levels. After remediation, the same checklist, tools, and manual procedures were repeated. Three threats to validity remain. The scoring is presence-based and was performed by the study team on both occasions. The set of 13 controls was chosen for testability. The evaluation took place in an isolated, development-mode environment with dummy data. Section V returns to these points.

IV. RESULTS

A. Baseline Conformity of ISO/IEC 27001:2022 Controls

Table 3 gives the status of the 13 controls at baseline and after remediation. At baseline, four controls were conformant (A.8.5, A.8.11, A.8.24, and A.8.27), five were partially conformant (A.8.2, A.8.10, A.8.15, A.8.26, and A.8.28), and four were not conformant (A.8.3, A.8.8, A.8.12, and A.8.16). The actual score was 13 of a maximum of 26, so the conformity was $(13/26) \times 100\% = 50\%$ and the gap was 50%. Each control was rated through 53 indicators in total; Supplementary Table S1 lists every indicator with its evidence. At the indicator level, 26 were conformant, 11 partially conformant, and 16 not conformant at baseline, and after remediation 52 were conformant and 1 partially conformant.

Table 3. Status of the evaluated controls at baseline and after remediation

Control	Baseline	After remediation
A.8.2 Privileged access rights	Partially conformant	Conformant
A.8.3 Information access restriction	Not conformant	Conformant
A.8.5 Secure authentication	Conformant	Conformant
A.8.8 Management of technical vulnerabilities	Not conformant	Partially conformant
A.8.10 Information deletion	Partially conformant	Conformant
A.8.11 Data masking	Conformant	Conformant
A.8.12 Data leakage prevention	Not conformant	Conformant
A.8.15 Logging	Partially conformant	Conformant
A.8.16 Monitoring activities	Not conformant	Conformant
A.8.24 Use of cryptography	Conformant	Conformant
A.8.26 Application security requirements	Partially conformant	Conformant
A.8.27 Secure system architecture	Conformant	Conformant
A.8.28 Secure coding	Partially conformant	Conformant

Score: conformant = 2, partially conformant = 1, not conformant = 0. Baseline total = 13/26; after remediation = 25/26.

The baseline statuses had the following reasons. A.8.3 and A.8.12 were not conformant because archive files could be opened directly through their URL without authentication or authorization, and archives that had not yet been approved could be accessed by other users, which creates a risk of broken access control and information leakage. A.8.8 was not conformant because no vulnerability assessment, documentation of identified vulnerabilities, dependency checks, or follow-up process existed before the study. A.8.16 was not conformant because the system had no mechanism to detect suspicious activity, record threats, notify administrators, or monitor attack patterns.

Among the partially conformant controls, A.8.2 and A.8.28 reflect CRUD (create, read, update, delete) functions that did not apply authorization or Laravel Policy consistently, even though the application used Laravel's built-in security features and no hard-coded credentials or other insecure coding practices were found. A.8.10 reflects hard deletion without soft delete or restore. A.8.15 reflects logs that captured authentication and CRUD activity, timestamps, user identity, IP address, and user agent, but not the data values before and after an update. A.8.26 reflects the absence of a Content Security Policy (CSP) header, although input validation, file validation, and partial authorization were in place.

B. Baseline Technical Testing

OWASP ZAP produced three alerts: CSP Header Not Set (medium), Big Redirect Detected (low), and Cookie No HttpOnly Flag on XSRF-TOKEN (low). Manual verification showed that the redirect alert came from Laravel's default redirect mechanism and exposed no sensitive information, so it was classified as a false positive. The cookie alert concerns the XSRF-TOKEN that Laravel uses for CSRF protection; this cookie is designed to be readable by JavaScript and holds no authentication or session information, so the alert was classified as accepted risk (by design). Only the CSP alert proceeded to risk analysis.

Vanguard reported CRUD functions without consistent authorization checks or Laravel Policy, which informed the assessment of A.8.2 and A.8.28. It also flagged file-upload processing in several modules. Verification of the upload helper showed that the application validates file type, size, MIME type, and file signature and stores files outside the public directory, so these findings were false positives. Finally, some packages in the development environment (*require-dev*) used development branches (dev-master and dev-main). Vanguard labeled this dependency finding and the authorization and upload findings as high severity; the contextual risk assessment of Section IV-C rated the dependency finding as low because the packages are not used in production. The packages need periodic monitoring, and they informed the assessment of A.8.8. Notably, the highest-risk issue, direct access to archive files, was not among the ZAP alerts; it was identified through the ISO/IEC 27001 checklist evaluation, with manual testing used to verify it.

C. Risk Assessment

Eight findings, consolidated from the checklist and the technical tests, were assessed (Table 4). Two were high risk, five were medium risk, and one was low risk.

Table 4. Risk assessment of the baseline findings

ID	Finding	Controls	L	I	Score	Level	Status after remediation
F1	Archive files accessible by URL without authentication and authorization	A.8.3, A.8.12	5	5	25	High	Not reproduced in re-test
F2	CRUD functions without consistent authorization check or Laravel Policy	A.8.2, A.8.28	4	4	16	High	Not reproduced in re-test
F3	No security monitoring, detection, or follow-up of incidents	A.8.16	3	4	12	Medium	Not reproduced in re-test
F4	Audit log lacks data-change details and security activity	A.8.15	3	4	12	Medium	Not reproduced in re-test
F5	No soft delete or restore; deletion not governed by a retention policy	A.8.10	3	4	12	Medium	Not reproduced in re-test
F6	No documented vulnerability assessment or formal vulnerability management	A.8.8	2	3	6	Medium	Assessment documented
F7	Content Security Policy header not applied	A.8.26	2	3	6	Medium	Alert no longer raised
F8	Development-branch dependencies (dev-master, dev-main) in the development environment	A.8.8	2	2	4	Low	Residual (accepted)

L = likelihood, *I* = impact (scale 1–5); score = $L \times I$. Post-remediation likelihood and impact were not re-rated; status reflects the re-test and the re-scored checklist.

The highest score (25) belongs to F1: because archive files could be opened if their URL was known, an unauthorized party could access archive documents, which threatens confidentiality. F2 scored 16 because CRUD functions are used directly in archive management, so inconsistent access control could let users read, modify, or delete data without proper rights, affecting confidentiality and integrity. The medium findings (F3–F7) do not directly cause data leakage, but they hinder detection, investigation, and handling of incidents and raise the chance that an attack succeeds if another vulnerability exists. F8 is low risk because the dependencies are in the development environment and are not used in production. Remediation followed this order, from F1 to F8.

D. Remediation Implementation

Table 5 maps each finding to the implemented measure; the measures are described below.

Table 5. Implemented security improvements mapped to findings and ISO/IEC 27001:2022 controls

ID	Control	Implemented improvement
F1	A.8.3, A.8.12	Private storage for all archive documents; temporary signed URLs with limited validity; non-public archives accessible only to authenticated users with matching rights
F2	A.8.2, A.8.28	Laravel Policy on archive-management modules; authorization check before every CRUD operation; review of role-based access rights for least privilege
F3	A.8.16	Security monitoring module: dashboard, threat log, IP and country monitoring, automatic blacklist, Telegram notification, IP whitelist, security settings
F4	A.8.15	Laravel Auditing on the main archive models (old and new values); existing activity log retained; threat and blacklist logs added
F5	A.8.10	Soft delete for all archives; restore by administrators only; administrator hard delete; scheduled purge of trash after the retention period
F6	A.8.8	Vulnerability assessment performed and documented; re-test with OWASP ZAP and Vanguard; triage of every finding
F7	A.8.26	CSP, X-Frame-Options, Permissions-Policy, and Cross-Origin-Resource-Policy headers through middleware
F8	A.8.8	Recorded as residual risk and monitored periodically (external packages cannot be modified directly)

Archive file access (F1). All archive documents were moved to private storage, so they can no longer be requested through a public URL. Access uses temporary signed URLs with a limited validity period; altering any URL parameter invalidates the digital signature, and an invalid or expired URL is rejected. Archives with non-public status can be opened only by authenticated users with appropriate rights, so access no longer depends on knowing a URL.

Authorization (F2). Laravel Policy was implemented in the archive-management modules that lacked adequate authorization. Every create, read, update, and delete operation now performs an authorization check before it runs, and role-based access rights were reviewed to apply the principle of least privilege.

Security monitoring (F3). A security monitoring module was developed. It consists of a security dashboard, a threat log, monitoring of source IP addresses and countries, automatic blacklisting, Telegram notifications, an IP whitelist, and adjustable security settings. A security middleware runs on every incoming request (Fig. 2). It validates the host, checks the IP blacklist and whitelist, detects bot activity and access to sensitive endpoints, and records threat activity before the request is processed further. When the number of threat events from one IP address exceeds a configured threshold, the system blocks that address automatically and notifies the administrator through Telegram. The whitelist prevents legitimate administrators and internal devices from being blocked.

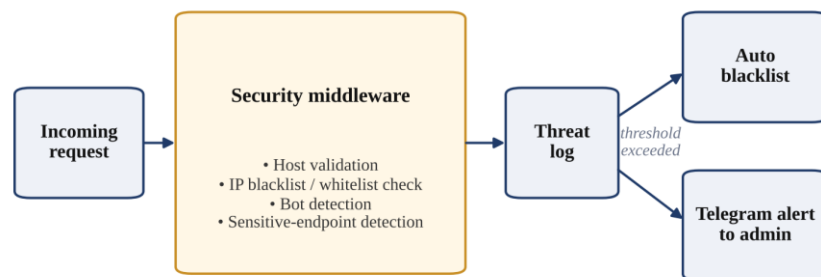


Fig. 2. Request-processing flow of the security monitoring module.

Audit logging (F4). The Laravel Auditing package was added to the main archive models so that the system stores old and new values, the changed attributes, the time of the change, and the user who made it. The existing activity log was retained for operational events such as login, logout, import, and export, and the threat and blacklist logs of the monitoring module complete the audit trail.

Deletion and recovery (F5). Hard delete was replaced by soft delete for all archive data, so deleted records stay recoverable. Only administrators can restore archives from the trash list or delete them permanently, and a scheduler purges archives that exceed the retention period configured by the administrator.

Security headers (F7). A CSP and further HTTP headers, namely X-Frame-Options, Permissions-Policy, and Cross-Origin-Resource-Policy, were applied through application middleware so that they apply to all pages. The CSP restricts the resources the browser may load, X-Frame-Options prevents clickjacking, Permissions-Policy limits access to browser

features such as camera, microphone, and geolocation, and Cross-Origin-Resource-Policy restricts cross-origin access to resources. Web-server-level settings such as TLS and HTTPS were outside the scope of this implementation.

Vulnerability management (F6, F8). A vulnerability assessment was performed, and its results were documented, including the identified vulnerabilities, their analysis, the evaluation of test results, and the follow-up for each finding. The system was then re-tested with OWASP ZAP and Vanguard. Findings verified as false positives were not remediated. The development-branch dependencies (F8) belong to external packages that developers cannot modify directly and that are not used in production, so they were recorded as residual risk.

E. Re-Evaluation and Before–After Comparison

After remediation, the same checklist was applied again (Table 3). Twelve controls were conformant, and one, A.8.8, was partially conformant because development-branch dependencies remained in the development environment. The actual score was 25 of 26, so conformity was $(25/26) \times 100\% = 96.15\%$ and the gap was $100\% - 96.15\% = 3.85\%$. Table 6 and Fig. 3 compare the two states.

Table 6. Comparison of security indicators before and after remediation

Indicator	Baseline	After remediation
Actual score	13/26	25/26
Conformity level	50%	96.15%
Security gap	50%	3.85%
Conformant controls	4	12
Partially conformant controls	5	1
Not conformant controls	4	0
OWASP ZAP alerts	3 (1 medium, 2 low)	2 (low)
Findings assessed for risk	8 (2 high, 5 medium, 1 low)	1 residual (low)

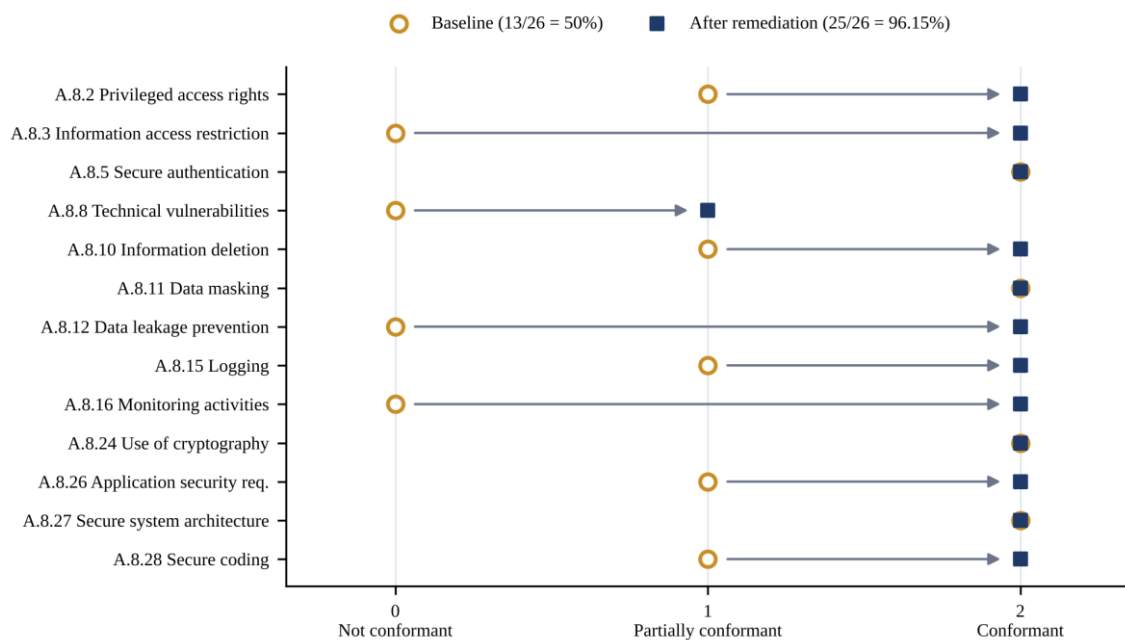


Fig. 3. Conformity score of each evaluated control at baseline and after remediation (0 = not conformant, 1 = partially conformant, 2 = conformant).

In the re-test, OWASP ZAP reported two low-risk alerts, Big Redirect Detected and Cookie No HttpOnly Flag, while the CSP alert no longer appeared. The redirect alert arises because the access control sends unauthenticated users to the login page, and the verification showed that this redirect exposes no sensitive information. The cookie alert again concerns the XSRF-TOKEN, which is readable by JavaScript by design. No medium, high, or critical vulnerability was found. Vanguard found no critical vulnerability that required direct changes to the source code. The only remaining item was the development-dependency finding, which was unchanged and kept as residual risk. Manual analysis of the source code and dependencies confirmed that no new finding required mitigation in the production environment. The monitoring module now records threats, displays them on a dashboard, blocks offending IP addresses automatically, and notifies the administrator when thresholds are reached.

F. Post-Implementation Verification of the Implemented Features

Table 7 summarizes the 34-scenario verification form. The tester marked 33 scenarios as matching the expected result. One scenario (no. 20, mass assignment: submitting the parameter role=admin should leave the user's role unchanged) carries an unclear mark on the form and is therefore treated as unconfirmed. The two scenarios on the monitoring module (nos. 27 and 28) cover a simulated brute-force login, sensitive-path scanning, or bot request, which were recorded in the threat log according to threat type, and a simulated threat that reached the threshold, which triggered an automatic blacklist and a Telegram notification. The final scenario (no. 34) confirmed that all existing features still worked normally after the changes. The form records one pass or fail mark per scenario; it does not report detection rates, false positives, or time to alert.

Table 7. Summary of the post-implementation verification form (34 scenarios)

Feature tested	Scenarios	Examples of test action and expected result	Marked as matching
Login	1–5	Five repeated failed logins return HTTP 429; brute force from one IP is rate-limited; unregistered e-mail does not reveal whether an account exists	5/5
Logout, forgot and reset password	6–9	Session removed on logout; repeated reset requests rate-limited; invalid reset token denied	4/4
Authorization	10–14	A regular user or staff member opening an admin page, or editing or deleting another user's data, is denied (403); a guest is redirected to login	5/5
Private storage and signed URLs	15–18	Direct file URL without authentication fails; valid signed URL works; altered or expired signed URL is denied	4/4
IDOR protection	19	Changing the ID of another user's archive is denied	1/1
Mass assignment	20	Submitting role=admin leaves the role unchanged	Unclear mark
Activity log	21–26	Login, create, update, soft delete, log viewing, and hard delete are recorded	6/6
Threat monitoring and response	27–28	Simulated brute force, path scanning, or bot request is recorded in the threat log; reaching the threshold triggers auto-blacklist and Telegram notification	2/2
Soft delete, trash, and retention	29–32	Deleted archive moves to trash; restore works; administrator hard delete removes it; retention scheduler removes expired archives	4/4
Content security policy	33	CSP header applied when the application is opened in a browser	1/1
Regression	34	All existing features work normally after the changes	1/1

Source: verification form completed by the tester on 6 July 2026.

V. DISCUSSION

The baseline conformity of 50% shows that using a modern framework is not in itself evidence of control conformity. The controls that were already conformant (secure authentication, data masking, cryptography, and secure architecture) are areas where Laravel provides sound defaults, such as password hashing and CSRF protection. The gaps concentrated in controls that the framework does not supply on its own: protection of stored files against direct access, consistent per-function authorization, monitoring and response, detailed audit trails, deletion and recovery policy, and browser-side hardening headers. This pattern agrees with the observation that framework features need additional configuration to protect an application effectively [10]. It suggests treating framework defaults as the starting point of a control assessment rather than its outcome.

The verification form adds a second kind of evidence to the checklist score: for 33 of 34 feature scenarios, the observed behavior matched the expected one, including the denial of direct file access, altered or expired signed URLs, and cross-user data access. This supports the claim that the main measures function as designed in the test environment, with the limits given below.

Both high-risk findings concern access control, the category with the most direct consequences for the confidentiality of archives. The highest-scoring finding (25) was a direct-file-URL exposure that OWASP ZAP did not report: ZAP raised only three alerts at baseline, of which one was actionable. This illustrates the value of combining a control-based checklist with manual testing and static analysis, because, in this case, the automated scanner reported configuration-level issues but did not reveal the authorization and file-exposure defects. Conversely, the automated outputs needed human triage. Several alerts were false positives or by-design behavior, and the screening step kept remediation effort on findings that mattered.

Earlier ISO/IEC 27001:2022 gap analyses, such as the 68–70% conformity reported for an academic information system [12], are not directly comparable with the 50% baseline here, because the studies assess different systems, control subsets, and scoring schemes. The methodological difference is more important. Where earlier studies stop at identifying gaps [5], [12], this study implemented countermeasures and re-measured, which indicates that a defined set of application-level measures raised the conformity of the evaluated controls and reduced the gap from 50% to 3.85%. In contrast to design-oriented work [13], [14], the controls were validated on a running web application. Regarding RQ3, the evaluated controls are almost fully conformant after remediation. This describes the implementation of those controls and not the

system's resistance to real attacks, and it does not extend to infrastructure-level protection, so wider use should first pass the validation steps listed in the limitations and future work.

For the Library and Archives Office, the results provide a documented baseline, a remediation record, and operational mechanisms (alerting, audit trails, and recoverable deletion) that support incident investigation before wider deployment. For practitioners with Laravel-based systems of a similar kind, Table 5 offers a mapping from control gaps to concrete measures that can serve as a starting point, although it was tested on one case only. The remaining partial conformity of A.8.8 shows that dependency governance needs an ongoing process rather than a one-time fix, and documenting the management of object storage (access rights, retention, backup, and restore) would help sustain the implemented controls.

Several measures need operational care beyond the technical implementation. The scheduled purge of the trash must follow the retention schedule that applies to the archives, because archives serve as administrative and legal evidence [11] and a misconfigured retention period could destroy records that must be kept. The signing key of the temporary URLs must be protected and rotated, since its compromise would defeat the access control built on it. The Telegram channel sends security metadata, such as IP addresses and event types, to a third-party service, which should be reviewed against the data-protection rules that apply to the office, and administrators need a defined procedure for responding to alerts so that alert fatigue does not reduce their value.

This study has several limitations. First, it covers a subset of the technological controls in Annex A.8 and excludes organizational, people, and physical controls as well as infrastructure and web-server settings such as TLS and HTTPS, so the conformity level is an application-level indicator and not a measure of readiness for ISMS certification. Second, the scoring scheme weights all controls equally on a three-level scale, and the likelihood and impact values are indicative judgments. No independent assessor or inter-rater agreement is reported, and the same team scored the system before and after its own remediation, so part of the improvement reflects the checklist asking for exactly the features that were then built; scoring bias cannot be excluded. Third, it is a single case evaluated in a local, isolated environment with dummy data and in development mode. The monitoring module was verified with two simulated-threat scenarios that each recorded a pass, but the detection thresholds, the automatic blocking, and the alerting were not validated against real attack traffic or quantified (detection rate, false positives, time to alert), and the study did not measure the performance overhead of the added middleware and audit logging. The verification form was completed by one archivist of the office rather than by an independent security assessor, one scenario (mass assignment) was not clearly confirmed, and the form was signed after, not before, the authors' own testing. The residual risk of the development-branch dependencies was judged low because they are not used in production, but this was not verified by inspecting production build configurations. Finally, the re-evaluation used the same checklist and tools as the baseline, which supports comparability but may share the blind spots of those instruments. An alternative yardstick such as the OWASP ASVS [28] was not applied.

VI. CONCLUSION

This study audited and improved the security of a web-based local archive digitization system using ISO/IEC 27001:2022 Annex A.8 controls, penetration testing, and static analysis. At baseline, 4 of 13 controls were conformant, 5 were partially conformant, and 4 were not conformant (13 of 26 points, 50% conformity). Eight findings were identified, two of high risk, five of medium risk, and one of low risk. The most critical were direct access to archive files by URL (risk score 25) and inconsistent authorization on CRUD functions (risk score 16).

Implementing private storage with signed URLs, Laravel Policy authorization, threat monitoring with automatic blocking and Telegram alerts, field-level audit logging, soft deletion with restore and retention-based purge, and a content security policy with HTTP security headers raised conformity to 96.15% (25 of 26 points) and reduced the gap from 50% to 3.85%. After remediation, OWASP ZAP reported no medium or higher alerts, and the only remaining finding was the development-branch dependencies, which were recorded as residual risk. The results show that the evaluated application-level controls are now almost fully implemented. They do not demonstrate resistance to real attacks, which still has to be validated before wider operational use.

Future work should extend the evaluation to more ISO/IEC 27001:2022 controls and to other standards, include infrastructure-level controls such as TLS and server configuration, quantify the monitoring module with scripted attack simulations (detection rate, false positives, and time to alert) and real traffic, confirm the mass-assignment scenario, add an independent assessor, apply a second static analysis tool and an alternative checklist such as the OWASP ASVS [28], verify that development-branch dependencies cannot enter production builds, and establish a periodic dependency review. Documenting the management of object storage, including access rights, retention, backup, and restore, is also recommended so that the implemented controls remain maintainable.

ACKNOWLEDGMENT

The authors thank the Library and Archives Office (*Dinas Perpustakaan dan Kearsipan*) of Probolinggo City for the opportunity to evaluate the system.

AUTHOR CONTRIBUTIONS

M. Ilham Zidni Syauquillah: Conceptualization, Methodology, Software, Investigation, Formal analysis, Writing – original draft. Moh. Sukron: Supervision, Writing – review & editing. Ahmad Halimi: Supervision, Writing – review & editing.

DATA AVAILABILITY

The evaluation used simulated (dummy) archive data; operational archive data of the Library and Archives Office cannot be shared for confidentiality reasons.

DECLARATION OF COMPETING INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper, with the following disclosure: Ahmad Halimi is the developer of Vanguard, the static analysis tool used in this study.

REFERENCES

- [1] M. S. Sahira, R. Indriati, and A. Ristyawan, "Analisis risiko website sistem keamanan informasi menggunakan metode FMEA dan framework ISO/IEC 27002:2022" [Risk analysis of a website information security system using the FMEA method and the ISO/IEC 27002:2022 framework], *JSITIK: J. Sist. Inf. Teknol. Inf. Komput.*, vol. 3, no. 2, pp. 128–138, 2025, doi: 10.53624/jsitik.v3i2.722.
- [2] D. Dhanya, "BSSN catat 3,64 miliar serangan siber di Indonesia setengah tahun ini," *Tempo.co*, Aug. 8, 2025. Accessed: Feb. 10, 2026. [Online]. Available: <https://www.tempo.co/digital/bssn-catat-3-64-miliar-serangan-siber-di-indonesia-setengah-tahun-ini-2056396>
- [3] K. K. Kalbar, "Kemenkum RI gelar sosialisasi SPBE: Waspada 3,64 miliar serangan siber di 2025," *Kemenkum Kalbar*, Aug. 28, 2025. Accessed: Feb. 10, 2026. [Online]. Available: <https://kalbar.kemenkum.go.id/berita-utama/kemenkum-ri-gelar-sosialisasi-spbe-waspada-3-64-miliar-serangan-siber-di-2025>
- [4] F. Nurhakim, "BSSN catat 3,64 miliar serangan siber selama Januari–Juli 2025," *Bloomberg Technoz*, Oct. 29, 2025. Accessed: Feb. 10, 2026. [Online]. Available: <https://www.bloombergtechnoz.com/detail-news/88672/bssn-catat-3-64-miliar-serangan-siber-selama-januari-juli-2025>
- [5] M. N. H. Siregar and Mardiah, "Analisis keamanan data pada sistem informasi menggunakan metode ISO/IEC 27001" [Data security analysis in an information system using the ISO/IEC 27001 method], *J. Ilmu Komput. Tek. Inform.*, vol. 1, no. 2, pp. 58–64, 2025, doi: 10.64803/juikti.v1i2.52.
- [6] A. M. Rizky, M. P. Pratiwi, A. Chairunnisa, I. A. Aiko, and A. Ariesmansyah, "E-Government: Meningkatkan efisiensi dan efektivitas pelayanan publik di Indonesia" [E-Government: Improving the efficiency and effectiveness of public services in Indonesia], *Innovative: J. Soc. Sci. Res.*, vol. 5, no. 1, pp. 2070–2089, 2025, doi: 10.31004/innovative.v5i1.17827.
- [7] R. Ramadhan, N. Widiyasono, and A. Rahmatullah, "Evaluasi keamanan informasi sistem informasi manajemen RSUD KHZ Musthafa menggunakan Indeks KAMI 5.0 berbasis ISO/IEC 27001:2022" [Information security evaluation of the management information system of RSUD KHZ Musthafa using KAMI Index 5.0 based on ISO/IEC 27001:2022], *JoMMIT: J. Multi Media dan IT*, vol. 9, no. 1, pp. 80–86, 2025, doi: 10.46961/jommit.v9i1.1706.
- [8] A. Halimi, F. Nur Fajri, and F. Rizal, "Pengembangan sistem identifikasi ancaman keamanan pada sistem penerimaan mahasiswa baru dengan framework Laravel" [Development of a security threat identification system for a new student admission system using the Laravel framework], *COREAI: J. Kecerdasan Buatan, Komputasi dan Teknol. Inf.*, vol. 6, no. 1, pp. 29–36, 2025, doi: 10.33650/coreai.v6i1.11537.
- [9] T. Rahmawati, N. Karna, S. Y. Shin, and M. A. P. Putra, "Enhancing network security through real-time threat detection with intrusion prevention system (case study on web attack)," *J. Ilm. Tek. Elektro Komput. dan Inform.*, vol. 10, no. 4, pp. 1004–1020, 2024, doi: 10.26555/jiteki.v10i4.30380.
- [10] S. M. Husain, L. Azhari, M. L. Aksani, and S. A. Saputra, "Analisis dan implementasi fitur keamanan aplikasi pada framework Laravel" [Analysis and implementation of application security features in the Laravel framework], *JIKA (J. Inform.)*, vol. 8, no. 3, pp. 281–287, 2024, doi: 10.31000/jika.v8i3.11198.
- [11] I. Nurkarimah, "Pengelolaan arsip digital: Tantangan dan strategi di era transformasi digital" [Digital archive management: Challenges and strategies in the era of digital transformation], *Maliki Interdiscip. J.*, vol. 3, pp. 1625–1629, 2025.
- [12] M. Izzani, N. Nellatul, A. Ismatul, and A. Hamdani, "Evaluation of the implementation of information security at SIAKAD Ibrahimy University based on ISO/IEC 27001:2022," *J. Ilm. Sist. Inf.*, vol. 5, no. 1, pp. 175–185, 2026, doi: 10.51903/d3bktm29.
- [13] J. Alberto and C. M. Karyati, "Perancangan sistem manajemen keamanan informasi (SMKI) berdasarkan ISO 27001:2022 (studi kasus Data Center Dinas Komunikasi dan Informatika Kota Tangerang Selatan)" [Design of an information security management system based on ISO 27001:2022 (case study of the Data Center of the Communication and Informatics Office of South Tangerang City)], *J. Ilm. Komput.*, vol. 22, no. 4, pp. 493–504, 2023, doi: 10.32409/jikstik.22.4.3447.
- [14] F. C. Arumdiya and C. Rudianto, "Implementasi ISO 27001:2022 dalam manajemen risiko keamanan informasi" [Implementation of ISO 27001:2022 in information security risk management], *J. Petisi (Pendidikan Teknol. Inf.)*, vol. 6, no. 2, pp. 68–81, 2025, doi: 10.36232/jurnalpetisi.v6i2.2012.
- [15] A. Halimi, A. Tholib, and M. A. Yaqin, "Optimasi keamanan data penerimaan mahasiswa menggunakan AES-256, SHA-256, dan Base64" [Optimizing student admission data security using AES-256, SHA-256, and Base64], *JUSTIFY: J. Sist. Inf. Ibrahimy*, vol. 3, no. 1, pp. 38–45, 2024, doi: 10.35316/justify.v3i1.5107.
- [16] D. Zalukhu, V. Yasin, and A. B. Yulianto, "Evaluasi keamanan sistem informasi akademik di STMIK Jayakarta berdasarkan ISO/IEC 27001:2022" [Security evaluation of the academic information system at STMIK Jayakarta based on ISO/IEC 27001:2022], *IJIS – Indones. J. Inf. Syst.*, vol. 10, no. 2, pp. 121–131, 2025, doi: 10.36549/ijis.v10i2.412.
- [17] A. Kusnandar, A. F. Rochim, and V. Gunawan, "Pengukuran tingkat risiko dan keamanan informasi menggunakan metode FMEA berbasis ISO/IEC 27001 pada instansi XYZ untuk keamanan sistem informasi" [Measuring information security risk levels using an ISO/IEC 27001-based FMEA method at institution XYZ for information system security], *J. Sist. Inf. Bisnis*, vol. 14, no. 4, pp. 375–384, 2024, doi: 10.21456/vol14iss4pp375-384.

- [18] W. Linggih Jaelani, Yanto, and F. Khoirunnisa, "Penetration testing website dengan metode black box testing untuk meningkatkan keamanan website pada instansi" [Website penetration testing with the black-box method to improve the security of an agency's website], *Naratif: J. Nas. Riset, Aplikasi dan Tek. Inform.*, vol. 5, no. 1, pp. 1–8, 2023, doi: 10.53580/naratif.v5i1.180.
- [19] S. H. Dewi, R. Albar, D. R. Yusian, and M. B. Wibawa, "Analisis pengujian keamanan aplikasi web Universitas Ubudiyah Indonesia menggunakan Open Worldwide Application Security Project Zed Attack Proxy (OWASP ZAP)" [Security testing analysis of the Universitas Ubudiyah Indonesia web application using the Open Worldwide Application Security Project Zed Attack Proxy (OWASP ZAP)], *J. Inform. Comput. Sci.*, vol. 11, no. 2, pp. 166–172, Oct. 2025.
- [20] I. G. S. Rahayuda and N. P. L. Santiari, "Evaluasi keamanan OTP Firebase pada aplikasi Android: Perbandingan SAST dan IAST dalam identifikasi kerentanan" [Security evaluation of Firebase OTP in Android applications: Comparing SAST and IAST for vulnerability identification], *JISKA (J. Inform. Sunan Kalijaga)*, vol. 11, no. 1, pp. 13–31, 2026, doi: 10.14421/jiska.4909.
- [21] *Information Security, Cybersecurity and Privacy Protection — Information Security Management Systems — Requirements*, ISO/IEC 27001:2022, International Organization for Standardization, Geneva, Switzerland, 2022.
- [22] S. Sakti, F. W. Listyanto, R. I. Damayanti, and M. A. Ningtyas, "Manajemen risiko keamanan informasi menggunakan standar ISO 27001:2022 studi kasus Komisi Pemilihan Umum" [Information security risk management using the ISO 27001:2022 standard: A case study of the General Elections Commission], *J. Manaj. Risiko*, vol. 5, no. 2, pp. 124–136, 2024, doi: 10.33541/mr.v5i2.5894.
- [23] B. S. Wibowo and R. F. Aji, "A comprehensive case study implementation recommendations 11 new information security controls ISO 27001:2022 at XYZ HealthTech company," *Indones. J. Comput. Sci.*, vol. 13, no. 4, 2024, doi: 10.33022/ijcs.v13i4.4166.
- [24] *Information Security, Cybersecurity and Privacy Protection — Information Security Controls*, ISO/IEC 27002:2022, International Organization for Standardization, Geneva, Switzerland, 2022.
- [25] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh, *Technical Guide to Information Security Testing and Assessment*, NIST Special Publication 800-115, National Institute of Standards and Technology, Gaithersburg, MD, USA, Sep. 2008.
- [26] OWASP Foundation, "Source code analysis tools." OWASP Community. Accessed: Jul. 1, 2026. [Online]. Available: https://owasp.org/www-community/Source_Code_Analysis_Tools
- [27] A. Halimi, "Vanguard," GitHub repository, 2026. Accessed: Jun. 26, 2026. [Online]. Available: <https://github.com/haliminurja/vanguard>
- [28] OWASP Foundation, *Application Security Verification Standard (ASVS)*, version 5.0.0, 2025.