Journal of Applied Informatics and Computing (JAIC)
Vol.10, No.3, June 2026, pp. 2620~2628
e-ISSN: 2548-6861

Identification and Mitigation of Web Application Vulnerabilities in Healthcare Systems

Saeful Diyan Pratama [1]*, Aris Tri Joko Harjanto [2]*, Bambang Agus Herlambang [3]*
* Informatics, Universitas PGRI Semarang
** 22670115@upgris.ac.id [1], aristrijaka@upgris.ac.id [2], bambangherlambang@upgris.ac.id [3]

Article Info

Article history:

Received 2026-05-07
Revised 2026-05-25
Accepted 2026-06-11

Keyword:

Application Security,
Cross-Site Scripting,
Healthcare Systems,
SQL Injection,
Token Reuse.

ABSTRACT

The rapid adoption of web-based applications in healthcare systems has increased exposure to security threats, particularly at the application layer. Despite the implementation of various security mechanisms, many systems remain vulnerable due to improper input validation, weak authentication controls, and insecure database interactions. This study aims to identify, validate, and mitigate critical web application vulnerabilities in a healthcare system, focusing on nonce reuse vulnerabilities in token-based authentication mechanisms, stored cross-site scripting (XSS), and SQL injection. The research employs an empirical approach through controlled security testing, including vulnerability identification, exploitation validation, and mitigation evaluation. The results demonstrate that all identified vulnerabilities are actively exploitable, affecting authentication integrity, data confidentiality, and system reliability. Furthermore, the implementation of targeted mitigation strategies, such as token validation, input sanitization, and parameterized queries, substantially reduced the observed exploitability of the identified vulnerabilities within the tested scenarios. These findings highlight that application-layer security weaknesses remain a significant risk in healthcare systems and require systematic and integrated mitigation approaches. The study suggests that adopting secure-by-design principles and continuous security testing may improve system resilience against application-layer attacks. The implications of this research emphasize the need for proactive security practices in web-based healthcare applications to prevent exploitation and protect sensitive data from evolving cyber threats.

I. INTRODUCTION

The rapid digital transformation in the healthcare sector has significantly increased the reliance on web-based applications for various services, including registration, data management, and administrative processes. Web applications play a critical role as they facilitate direct interaction between users, servers, and databases, making them essential yet highly exposed components of modern healthcare systems [1], [2]. Vulnerabilities in web applications are frequently exploited as primary entry points for cyberattacks, particularly in mechanisms related to input validation, authentication, and session management [3], [4]. Common attacks such as cross-site scripting (XSS), SQL injection, and request manipulation can severely compromise the confidentiality, integrity, and availability of healthcare systems [5], [6].

The complexity of cybersecurity threats continues to increase alongside technological advancements and system interconnectivity. The growing number of digital components and interconnections significantly expands the attack surface and increases the likelihood of cyberattacks [7]. This indicates that highly connected systems require more robust and integrated security approaches. Furthermore, complex systems demand integrated approaches for risk identification, vulnerability assessment, and adaptive mitigation strategies to effectively reduce systemic risks [8]. Therefore, web application security in healthcare should be treated as a

dynamic process that requires continuous evaluation and improvement [9].

Despite the implementation of various security mechanisms, significant challenges remain in applying effective controls at the application level. Many systems are still vulnerable due to weak input validation, insecure token management, and insufficient protection against request manipulation [10], [11]. This condition reflects a gap between theoretical security models and real-world implementation. As a result, there is a need for evidence-based approaches that not only identify vulnerabilities but also demonstrate exploitation and evaluate the effectiveness of mitigation strategies [12].

Modern cybersecurity literature emphasizes that mitigation strategies should not be isolated but instead integrated across the system lifecycle. System security should be analyzed using a lifecycle-aware approach that systematically maps threats to defense mechanisms [13]. In addition, security practices such as risk assessment, continuous monitoring, and adherence to security standards are essential for improving system resilience against attacks [14]. These approaches highlight that mitigation should be proactive, structured, and embedded within system design rather than applied reactively [15].

Previous studies, however, present several limitations. Cybersecurity threats in digital healthcare are becoming increasingly complex due to technological integration, yet many studies remain general and lack concrete exploit-based analysis of web applications [16]. Moreover, vulnerability analysis should consider interactions between system components, but such approaches are rarely applied in practical web application security studies [17]. These limitations indicate a research gap, particularly in studies that combine real-world vulnerability identification, exploit validation, and mitigation evaluation within a single healthcare web application context [18], [19], [20].

Based on this gap, this study aims to identify and analyze security vulnerabilities in the e-STR system, focusing on nonce reuse vulnerabilities, stored cross-site scripting (XSS), and SQL injection. In this study, nonce reuse refers to the improper reuse of authentication tokens or cryptographic request values that should only be accepted once within a limited validity period. The absence of strict nonce expiration and replay validation enables attackers to reuse previously intercepted authentication requests to bypass session integrity controls. The novelty of this study lies in its empirical real-world case-based approach, which combines vulnerability identification, exploit validation, mitigation implementation, and post-mitigation verification within a healthcare web application environment. Unlike previous studies that primarily discuss web application vulnerabilities conceptually, this study provides empirical exploit validation and mitigation verification on a real-world healthcare web application. The study also analyzes the relationship between authentication flaws, client-side vulnerabilities, and database exploitation within a single operational environment. Furthermore, this study links technical vulnerabilities to their impact on confidentiality, integrity, and availability while providing practical mitigation strategies applicable to similar systems. The scope of this study is limited to application-layer vulnerabilities and relevant mitigation implementations without addressing the entire system infrastructure.

II. METHODOLOGY

A. System Environment and Study Context

This study was conducted on an anonymized web-based healthcare system referred to as the e-STR system. The system operates as a client-server web application consisting of a browser-based frontend, RESTful API services, and a relational database backend. The backend services are implemented using the Go programming language with the GORM framework for database interaction.

The analysis is limited to the application layer, focusing on authentication mechanisms, input handling, and server-side request processing. The materials analyzed in this study include HTTP request and response data, authentication tokens, user input fields, and administrative endpoints. These components were evaluated to identify potential vulnerabilities and observe system behavior under both normal and adversarial conditions.

The security assessment was performed on a real operational environment under authorized conditions provided by the system owner. Testing activities were conducted through a secured Virtual Private Network (VPN) connection to ensure controlled and legitimate access during the assessment process. This study adopts a strict black-box testing methodology, where no access to source code or internal implementation details was provided. All vulnerabilities were identified and validated solely through observable system behavior, input-output analysis, and HTTP response characteristics. All exploitation activities were performed in a controlled and authorized environment. The testing process did not involve real patient records or unauthorized access to sensitive medical data. Payload execution was restricted to validation purposes and designed to avoid service disruption or operational impact on the healthcare system. Security testing scenarios included replay attack simulation for nonce validation, persistent payload injection for stored cross-site scripting (XSS) testing, and time-based SQL injection payloads for database interaction analysis. In the mitigation phase, the use of GORM enabled the implementation of parameterized queries and prepared statements to reduce SQL injection risks.

B. Security Testing Workflow

The security testing process in this study follows a structured workflow to simulate real-world attack scenarios and evaluate system responses. The overall workflow is illustrated in Figure 1.

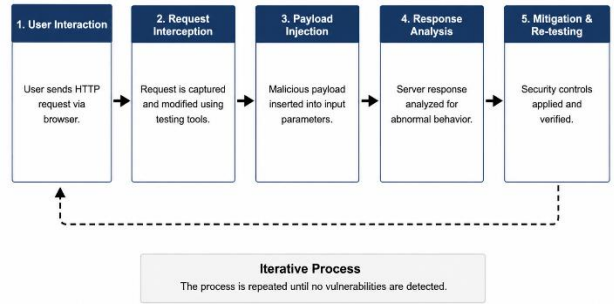

Figure 1. Security Testing Workflow

As shown in Figure 1, the process begins with user interaction with the web application through a browser, where HTTP requests are generated and sent to the server. These requests are intercepted using a testing tool to allow inspection and modification of parameters. Subsequently, crafted attack payloads are injected into selected input fields or request parameters to test for potential vulnerabilities. The server responses are then analyzed to identify abnormal behavior, such as script execution, delayed responses, or unintended data exposure. Finally, mitigation mechanisms are implemented and the same testing process is repeated to verify that the vulnerabilities were no longer exploitable within the evaluated scenarios.

C. Testing Environment and Tools

The experimental setup was designed to replicate realistic web application interactions within a controlled environment. The testing environment consists of a client-side browser interacting with the web server, where HTTP communication is intercepted and analyzed.

Security testing tools were used to support vulnerability identification and validation. The tools and their functions are summarized in Table 1.

TABLE I
SECURITY TESTING TOOLS USED IN THE STUDY

Tool	Function
Burp Suite	Intercepting and modifying HTTP requests
Web Browser	Accessing and interacting with the system
Manual Payload Testing	Crafting attack payloads (XSS, SQLi)
Logging Mechanism	Observing system responses and behavior

As shown in Table 1, Burp Suite was primarily used to intercept and manipulate HTTP requests, allowing detailed inspection of request-response behavior. Manual payload testing was applied to simulate realistic attack scenarios beyond automated detection.

D. Vulnerability Testing Procedure

The study follows a systematic vulnerability assessment procedure consisting of several main stages. The security assessment methodology follows the Penetration Testing Execution Standard (PTES), including reconnaissance, vulnerability identification, exploitation validation, mitigation verification, and reporting stages. In addition, the testing scenarios were aligned with the OWASP Web Security Testing Guide and OWASP Top 10 categories related to injection flaws and authentication weaknesses. SQLmap was also used to support automated validation of SQL injection vulnerabilities. First, reconnaissance was conducted to identify available endpoints, input parameters, and authentication mechanisms. This stage aims to map the system structure and identify potential attack surfaces.

Second, vulnerability identification was performed by injecting crafted inputs into selected parameters. Various attack vectors were tested, including script injection, database query manipulation, and token reuse scenarios.

Third, exploitation validation was carried out to confirm the existence of vulnerabilities. Indicators such as script execution, delayed responses, and repeated token acceptance were used to verify successful exploitation.

Finally, mitigation verification was performed after corrective actions were applied. The same attack scenarios were repeated to ensure that vulnerabilities were effectively mitigated and no longer exploitable.

E. Data Collection and Analysis

Data collection was performed through real-time monitoring of HTTP traffic and system responses during testing. The collected data include request parameters, server responses, execution behavior, and observable system outputs.

The analysis was conducted using an evidence-based approach, focusing on exploitability, impact, root cause, and mitigation effectiveness. The evaluation criteria are summarized in Table 2.

TABLE II
VULNERABILITY EVALUATION CRITERIA

Criteria	Description
Exploitability	Ability to execute the attack successfully
Impact	Effect on confidentiality, integrity, availability
Root Cause	Source of vulnerability within the system
Mitigation	Effectiveness of applied security controls

III. RESULT AND DISCUSSION

A. *Empirical Vulnerability Findings*

The security assessment conducted on the e-STR system revealed the presence of three critical vulnerabilities at the application layer, namely nonce reuse, stored cross-site scripting (XSS), and SQL injection. These vulnerabilities were identified and confirmed through controlled testing

scenarios, where each attack vector was systematically executed to validate exploitability.

The findings indicate that the system contains weaknesses in authentication mechanisms, input validation processes, and database query handling. Such weaknesses enable attackers to manipulate application behavior, bypass authentication controls, and execute malicious payloads [21], [22].

A summary of the identified vulnerabilities is presented in Table 3.

TABLE III
SUMMARY OF IDENTIFIED VULNERABILITIES

Vulnerability	Category	Description	Exploitation Evidence	Severity
Nonce Reuse	Authentication	Token remains valid beyond intended usage period	Previously generated hash still accepted	Critical
Stored XSS	Client-side	Malicious script stored and executed in user interface	Script rendered in browser	Critical
SQL Injection	Database	User input directly injected into SQL query without validation	Response delay using time-based payload	Critical

As shown in Table 3, all identified vulnerabilities are categorized as critical due to their high impact on system security. The nonce reuse vulnerability allows attackers to reuse previously generated authentication tokens, indicating the absence of proper nonce validation and anti-replay mechanisms. The stored XSS vulnerability enables persistent execution of malicious scripts in user browsers, which can lead to session hijacking or credential theft. Meanwhile, the SQL injection vulnerability exposes the backend database to unauthorized access and manipulation.

To provide a more detailed view of the attack surface, the identified vulnerabilities were observed across several critical endpoints. The nonce reuse vulnerability was found in the /encryption/encode and /encryption/decode endpoints, where tokens remained valid beyond their intended lifetime. The stored XSS vulnerability affected the /pengajuanbaru and /adminvalidasipemohon endpoints, allowing malicious scripts to be stored and executed in the admin interface. The SQL injection vulnerability was identified in the /adminvalidasipemohon/getregbaru endpoint, where user input was directly processed in database queries without proper validation.

Empirical evidence from the testing process confirms that these vulnerabilities are not merely theoretical but can be actively exploited in real scenarios. For instance, the nonce reuse vulnerability demonstrates that authentication tokens remain valid even after extended periods, allowing repeated use without revalidation. Similarly, the stored XSS vulnerability confirms that injected scripts are stored in the database and executed when accessed by other users. In the case of SQL injection, the use of time-based payloads resulted in measurable delays in server response, confirming that the input is directly processed by the database query [23], [24].

B. *Exploitation Analysis*

To validate the identified vulnerabilities, controlled exploitation was performed for each attack vector. This process demonstrates the practical impact of the vulnerabilities and provides empirical evidence of system weaknesses.

1) *Nonce Reuse Vulnerability*: The nonce reuse vulnerability was identified in the authentication mechanism involving encoding and decoding processes. The system failed to enforce token uniqueness and expiration, allowing previously generated tokens to remain valid beyond their intended usage period. As shown in Table 4, a previously generated token remains valid even after several days, indicating the absence of proper nonce validation and anti-replay protection.

TABLE IV
REQUEST AND RESPONSE ANALYSIS

Component	Content
Endpoint	/encryption/decode
Request	POST /encryption/decodevalue=rEteF9xDh6ez4x8HqSn0A...
Response	HTTP 200 OK"success": true"value": "C#aqw12s"
Observation	Previously issued token is still accepted, indicating a replay vulnerability

This behavior indicates a failure in enforcing token lifecycle constraints, which are fundamental in secure authentication design. The absence of anti-replay mechanisms allows attackers to reuse captured tokens, thereby bypassing authentication controls and compromising session integrity. Such vulnerabilities are particularly critical in systems that rely on token-based authentication without strict validation [25]. The vulnerability persists beyond the expected validity period, where previously generated tokens remain valid even after several days, confirming the absence of strict expiration and replay protection mechanisms.

2) *Stored Cross-Site Scripting (XSS)*: A stored XSS vulnerability was identified in administrative input fields that accept user-generated content without proper sanitization. The injected payload is stored in the database and executed when accessed by other users. As illustrated in Figure 2, a malicious script injected through the administrative interface is rendered and executed in the user interface.

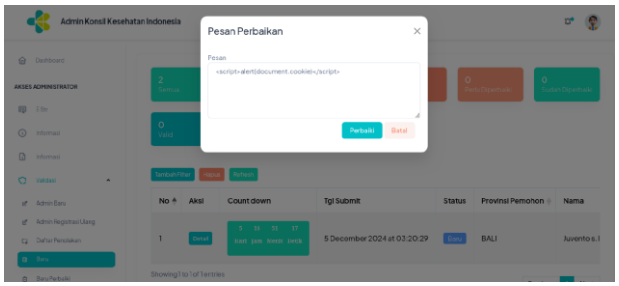

Figure 2. Persistent Stored XSS Payload Execution in Administrative Interface

This confirms the presence of a persistent XSS vulnerability. Unlike reflected XSS, stored XSS has a broader impact as the malicious payload is permanently stored and executed across multiple user sessions. This type of vulnerability is particularly critical in multi-user systems, where malicious scripts can propagate across different roles and compromise trust boundaries within the application. The root cause lies in the absence of input validation and output encoding mechanisms, allowing user-supplied data to be interpreted as executable code [26], [27].

3) *SQL Injection Vulnerability*: A time-based SQL injection vulnerability was identified due to improper handling of user input in database queries. The application directly incorporated user input into SQL statements without parameterization. As shown in Figure 3, injecting the payload SELECT PG_SLEEP(5) resulted in an approximate 6-second delay in the response time, confirming that the database executed the injected command.

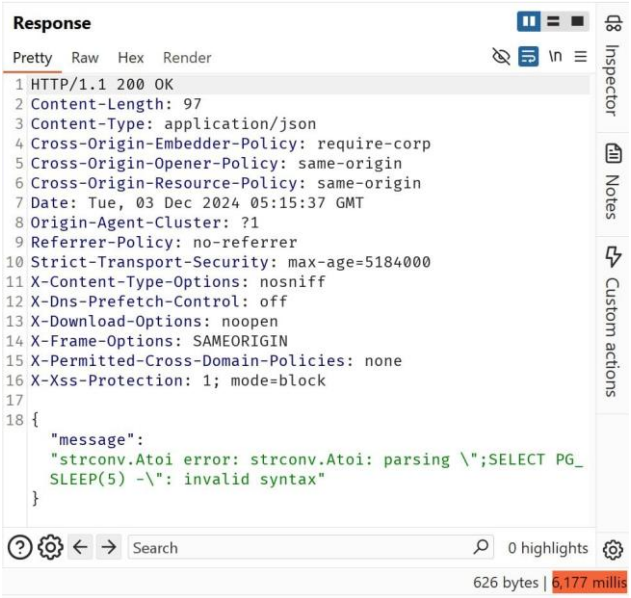

Figure 3. Time-Based SQL Injection Attack Evidence

The observed delay-based response confirms that the application directly processes user input within SQL queries, violating secure coding practices that require strict parameterization. This vulnerability allows attackers to perform database enumeration, extract sensitive data, and potentially gain full control over the database. Compared to other vulnerabilities, SQL injection poses the highest risk due to its potential impact on both data confidentiality and system integrity [28]. Further exploitation demonstrated that the attacker was able to enumerate database structures, including table names, confirming deeper database exposure beyond simple time-based validation.

C. *Security Impact Analysis*

The identified vulnerabilities have significant implications for the security of the system, particularly in relation to the core principles of confidentiality, integrity, and availability (CIA). Each vulnerability contributes differently to the overall security risk, indicating systemic weaknesses in the application-layer design.

The nonce reuse vulnerability primarily affects authentication integrity. By allowing previously generated tokens to be reused, the system fails to enforce token uniqueness and expiration constraints. This enables replay attacks, where attackers can reuse intercepted tokens to gain unauthorized access without re-authentication. Such behavior undermines trust in the authentication mechanism and may lead to unauthorized actions within the system [29].

The stored cross-site scripting (XSS) vulnerability impacts both integrity and confidentiality. Since malicious scripts are stored and executed within the application, attackers can manipulate user interactions, steal session data, or perform actions on behalf of legitimate users. This type of vulnerability is particularly critical in multi-user systems, where injected payloads can propagate across different user roles and compromise trust boundaries within the application [30].

The SQL injection vulnerability poses the most severe risk, affecting confidentiality, integrity, and potentially availability. By allowing direct manipulation of database queries, attackers can extract sensitive information, modify stored data, or disrupt database operations. In advanced exploitation scenarios, SQL injection may lead to complete database compromise or denial of service conditions [26].

In healthcare environments, these vulnerabilities may also introduce regulatory and compliance risks related to the protection of sensitive patient information. Unauthorized disclosure or manipulation of medical records may violate healthcare data protection regulations and reduce trust in digital healthcare services. Therefore, application-layer vulnerabilities should not only be viewed as technical weaknesses but also as risks affecting legal compliance, patient privacy, and organizational reputation.

The combined presence of these vulnerabilities indicates that the system lacks fundamental secure coding practices, particularly in input validation, session management, and query construction. Rather than isolated issues, these

vulnerabilities suggest systemic weaknesses that can be exploited in combination, increasing the overall attack impact.

Furthermore, the coexistence of multiple vulnerabilities within a single system significantly increases the attack surface. Attackers may chain vulnerabilities, for example, using XSS to steal tokens and then exploiting nonce reuse to maintain persistent access. This demonstrates that the security risks are not only individual but also cumulative, amplifying their potential impact [12].

These findings are consistent with common web application attack patterns, where injection-based and authentication-related vulnerabilities remain dominant threats, particularly in systems handling sensitive healthcare data [29]. Therefore, addressing these vulnerabilities requires a comprehensive and integrated security approach rather than isolated fixes.

D. *Mitigation Effectiveness Evaluation*

Following the identification and validation of vulnerabilities, mitigation strategies were implemented and evaluated through controlled re-testing. The objective of this phase is to assess whether the applied security controls effectively eliminate the identified vulnerabilities and prevent further exploitation. The mitigation results are summarized in Table 5.

TABLE V
MITIGATION RESULTS BEFORE AND AFTER IMPLEMENTATION

Vulnerability	Before Mitigation	Mitigation Applied	After Mitigation	Status
Nonce Reuse	Token reusable	Token invalidation and validation added	Token rejected after reuse	Mitigated
Stored XSS	Script executed in browser	Input sanitization and output encoding	Script rendered as plain text	Mitigated
SQL Injection	Delayed response (time-based)	Parameterized queries and input validation	No delay, payload not executed	Mitigated

As shown in Table 5, all identified vulnerabilities were effectively mitigated within the evaluated testing scenarios after implementing appropriate security controls. Each mitigation strategy directly addresses the root cause of the corresponding vulnerability.

For the nonce reuse vulnerability, the insecure token mechanism was corrected by enforcing token uniqueness and implementing proper validation checks. Tokens are no longer reusable after initial use, effectively preventing replay attacks. This demonstrates the importance of enforcing strict token lifecycle management in authentication systems.

For the stored XSS vulnerability, input sanitization and output encoding mechanisms were applied. As a result, malicious scripts are no longer executed but instead rendered as plain text. This confirms that proper handling of user input is essential to prevent client-side code execution.

For the SQL injection vulnerability, the use of parameterized queries and server-side validation successfully prevented malicious payloads from being interpreted as part of SQL commands. Re-testing using identical payloads showed no abnormal response behavior, indicating that the injection point was no longer exploitable within the evaluated testing scenarios.

However, input sanitization and parameterized queries should not be considered absolute protection mechanisms. Improper implementation, inconsistent validation across application components, or misconfigured query handling may still introduce potential bypass opportunities. Therefore, these mitigation techniques should be combined with secure coding practices, continuous security testing, and regular security reviews to maintain long-term effectiveness.

The results demonstrate that vulnerability mitigation is most effective when it directly targets the root cause rather than applying superficial fixes. This aligns with secure coding principles that emphasize input validation, proper authentication handling, and safe database interaction [22].

However, it is important to note that mitigation effectiveness depends on consistent implementation across all components of the system. Partial or inconsistent application of security controls may leave residual vulnerabilities that can still be exploited. These findings highlight that effective mitigation strategies must be systematically validated through re-testing to ensure that vulnerabilities are fully resolved and do not reappear in different forms.

E. *Quantitative Mitigation Evaluation*

To evaluate mitigation effectiveness quantitatively, repeated testing was conducted before and after mitigation implementation. Prior to mitigation, all tested attack payloads were successfully executed, resulting in an attack success rate of 100%. After mitigation, no payloads were successfully executed during repeated testing, reducing the observed attack success rate to 0% within the evaluated scenarios.

For SQL injection testing, the average response delay caused by the time-based payload decreased from approximately 6 seconds before mitigation to normal response times after parameterized queries were implemented. Additional latency introduced by mitigation mechanisms was observed to be negligible during normal system interaction.

F. *Comparison with Previous Studies*

The findings of this study are consistent with existing research on cybersecurity vulnerabilities, particularly in complex and interconnected systems. However, this study also provides additional contributions through empirical validation and real-world exploitation analysis.

Increasing system interconnectivity significantly expands the attack surface, making application-layer vulnerabilities

more exploitable [7]. The results of this study support this observation, as all identified vulnerabilities, including nonce reuse, stored XSS, and SQL injection, originate from weaknesses in user input handling and request processing mechanisms within a highly interactive system.

Effective vulnerability management requires integrated and adaptive strategies that address multiple risk factors simultaneously [8]. In this study, the coexistence of multiple vulnerabilities within a single system highlights the importance of comprehensive security approaches rather than isolated mitigation techniques. Each vulnerability affects different components of the system, reinforcing the need for a holistic security framework.

Modern security analysis should adopt a lifecycle-aware approach, mapping threats to mitigation strategies across different system stages [13]. This study aligns with that perspective by demonstrating how each identified vulnerability requires a specific mitigation mechanism, including token validation, input sanitization, and query parameterization.

Cybersecurity threats in healthcare systems are becoming increasingly complex due to digital transformation, yet many studies remain conceptual and lack empirical validation [16]. This study addresses this limitation by providing direct exploitation evidence, demonstrating that the identified vulnerabilities are actively exploitable in real-world scenarios.

Vulnerability analysis should consider interactions between system components rather than treating vulnerabilities as isolated issues [17]. The findings of this study confirm this perspective, as vulnerabilities were identified across multiple layers, including authentication mechanisms, client-side input processing, and database interaction.

Compared to previous studies, the key contribution of this research lies in its empirical approach. While prior studies primarily focus on theoretical analysis or general risk assessment, this study combines vulnerability identification, exploitation validation, and mitigation evaluation within a single real-world system. This integrated approach provides stronger evidence of security weaknesses and demonstrates practical mitigation effectiveness.

G. *Study Limitations*

Despite the significant findings, this study has several limitations that should be considered when interpreting the results.

First, the analysis is limited to a single web-based healthcare system. While the identified vulnerabilities—nonce reuse, stored XSS, and SQL injection—are representative of common application-layer weaknesses, the findings may not be directly generalizable to other systems with different architectures, technologies, or security implementations.

Second, the scope of this study is restricted to application-layer vulnerabilities. Network-level, infrastructure-level, and operating system-level security aspects were not evaluated. As a result, the overall security posture of the system may involve additional risks that are beyond the scope of this research.

Third, the testing was conducted in a controlled environment using manual and semi-automated techniques. Although this approach allows precise validation of specific vulnerabilities, it may not fully reflect real-world attack scenarios, where attackers may use more advanced tools or exploit chained vulnerabilities in complex ways.

Additionally, the study focuses on known vulnerability types and does not explore zero-day vulnerabilities or unknown attack vectors. This limitation implies that other undiscovered vulnerabilities may still exist within the system.

This study only evaluates three specific application-layer vulnerabilities and does not guarantee the absence of other security weaknesses within the system. Additional vulnerabilities related to network infrastructure, business logic, access control, or zero-day attack vectors may still exist and were not covered within the scope of this research.

Although this study recommends the adoption of secure-by-design principles, the implementation was not evaluated throughout the entire software development lifecycle (SDLC). Therefore, the effectiveness of these practices in long-term development and maintenance processes was not comprehensively assessed.

Finally, the mitigation evaluation was performed within a limited testing scope and timeframe. While the applied fixes successfully prevented the tested exploits, long-term effectiveness and resilience against evolving attack techniques were not assessed. Further longitudinal security assessments and continuous monitoring are required to determine whether the implemented mitigations remain effective against evolving attack techniques over time.

H. *Future Work*

Based on the findings and limitations of this study, several directions for future research can be proposed to enhance the security of web-based healthcare systems.

First, future studies should extend the analysis to multiple systems with different architectures and technologies. This would improve the generalizability of the findings and provide a broader understanding of application-layer vulnerabilities across various healthcare platforms.

Second, the integration of automated vulnerability detection tools and continuous security testing within the software development lifecycle is recommended. Incorporating security testing into continuous integration and continuous deployment (CI/CD) pipelines can help detect vulnerabilities at an earlier stage and reduce the risk of exploitation in production environments.

Third, future research may explore advanced attack scenarios, including vulnerability chaining and multi-stage exploitation. For example, combining stored XSS with token-related vulnerabilities could lead to more severe attack impacts, such as persistent session hijacking or privilege

escalation. Investigating such attack chains would provide deeper insight into real-world threat scenarios.

Additionally, expanding the scope to include network-level and infrastructure-level security analysis would provide a more comprehensive evaluation of system security. This includes examining firewall configurations, network segmentation, and server hardening practices.

Another promising direction is the adoption of adaptive security frameworks and real-time monitoring systems. Implementing anomaly detection and behavior-based monitoring can help identify suspicious activities and respond to threats dynamically.

Finally, future work may also investigate the application of secure-by-design principles and formal security verification methods to prevent vulnerabilities during the system development phase. This approach shifts security from reactive mitigation to proactive prevention, which is essential for protecting critical healthcare systems.

IV. CONCLUSION

This study successfully identified and validated three critical vulnerabilities in a web-based healthcare system, namely nonce reuse, stored cross-site scripting (XSS), and SQL injection, demonstrating that these weaknesses are not only present but actively exploitable in real-world scenarios. The findings highlight that inadequate input validation, improper token lifecycle management, and insecure database query construction remain fundamental issues in application-layer security. The implications of these results extend beyond the studied system, indicating that similar vulnerabilities may exist in other healthcare applications that rely on web-based architectures, thereby posing significant risks to confidentiality, integrity, and availability of sensitive data. Furthermore, the successful mitigation of these vulnerabilities confirms that implementing secure coding practices such as input sanitization, parameterized queries, and strict authentication controls can effectively reduce exploitation risks when properly applied and validated. Based on these findings, it is recommended that developers adopt secure-by-design principles, integrate continuous security testing within the development lifecycle, and perform regular vulnerability assessments to proactively identify and mitigate risks. Future research should focus on expanding empirical studies across multiple systems, exploring advanced attack scenarios such as vulnerability chaining, and developing automated and adaptive security mechanisms to enhance the resilience of web-based healthcare systems against evolving cyber threats.

REFERENCES

[1] C. M. Mejía-Granda, J. L. Fernández-Alemán, J. M. Carrillo-de-Gea, and J. A. García-Berná, "Security vulnerabilities in healthcare: an analysis of medical devices and software," *Med. Biol. Eng. Comput.*, vol. 62, no. 1, pp. 257–273, 2024.

[2] P. Ewoh and T. Vartiainen, "Vulnerability to cyberattacks and sociotechnical solutions for health care systems: systematic review," *J. Med. Internet Res.*, vol. 26, p. e46904, 2024.

[3] K. Kandasamy, S. Srinivas, K. Achuthan, and V. P. Rangan, "Digital healthcare-cyberattacks in asian organizations: an analysis of vulnerabilities, risks, nist perspectives, and recommendations," *IEEE access*, vol. 10, pp. 12345–12364, 2022.

[4] M. A. Khatun, S. F. Memon, C. Eising, and L. L. Dhirani, "Machine learning for healthcare-IoT security: A review and risk mitigation," *IEEE access*, vol. 11, pp. 145869–145896, 2023.

[5] B. O. Owolabi, "Exploring systemic vulnerabilities in healthcare digital ecosystems through risk modeling, threat intelligence, and adaptive security control mechanisms," *Int J Comput Appl Technol Res*, vol. 11, no. 12, pp. 687–699, 2022.

[6] A. C. Ikegwu, U. R. Alo, and H. F. Nweke, "Cyber threats in mobile healthcare applications: systematic review of enabling technologies, threat models, detection approaches, and future directions," *Discover Computing*, vol. 28, no. 1, p. 152, 2025.

[7] T. Fernandes, J. P. Magalhães, and W. Alves, "Cybersecurity in Smart Railways: Exploring risks, vulnerabilities and mitigation in the data communication services," *Green Energy and Intelligent Transportation*, vol. 4, no. 4, p. 100305, Aug. 2025, doi: 10.1016/j.geits.2025.100305.

[8] G. Rahman, M.-K. Jung, T.-W. Kim, and H.-H. Kwon, "Drought impact, vulnerability, risk assessment, management and mitigation under climate change: A comprehensive review," *KSCE Journal of Civil Engineering*, vol. 29, no. 1, p. 100120, Jan. 2025, doi: 10.1016/j.kscej.2024.100120.

[9] L. Nemec Zlatolas, T. Welzer, and L. Lhotska, "Data breaches in healthcare: security mechanisms for attack mitigation," *Cluster Comput.*, vol. 27, no. 7, pp. 8639–8654, 2024.

[10] İ. AVCI and E. DOĞAN, "WEB APPLICATION SECURITY: DETECTION AND MITIGATION OF VULNERABILITIES," in *10th INTERNATIONAL NEW YORK CONFERENCE ON EVOLVING TRENDS IN INTERDISCIPLINARY RESEARCH & PRACTICES*, 2024, pp. 491–508.

[11] F. F. Fadlalla and H. T. Elshoush, "Input validation vulnerabilities in web applications: Systematic review, classification, and analysis of the current state-of-the-art," *IEEE Access*, vol. 11, pp. 40128–40161, 2023.

[12] O. Erukayenure, H. A. Bashir, A. Adekunbi, S. E. Abere, O. Okpan, and A. A. Guwa, "Human factor vulnerabilities in healthcare cybersecurity: Mitigating insider threats in medical facilities," *Int. J. Sci. Res. Arch*, vol. 17, pp. 24–31, 2025.

[13] A. D. E. Berini *et al.*, "Security and privacy in LLMs: A comprehensive survey of threats and mitigation strategies," *Information Fusion*, vol. 132, p. 104241, Aug. 2026, doi: 10.1016/j.inffus.2026.104241.

[14] A. E. Hafez and M. M. Almustafa, "Detecting Security Vulnerabilities in Web Applications: A Proposed System.," *International Journal of Safety & Security Engineering*, vol. 14, no. 6, 2024.

[15] B. Riskhan, M. A. U. Sheikh, M. S. Hossain, K. Hussain, Z. Zainol, and N. Z. Jhanjh, "Major vulnerabilities of web application in real world scenarios and their prevention," in *2025 International Conference on Intelligent and Cloud Computing (ICoICC)*, IEEE, 2025, pp. 1–6.

[16] C. Mehra and A. K. Sharma, "Safeguarding the Landscape of Mental Wellness: Analyzing Cyber Threats and Mitigation Strategies in Digital Healthcare," *Procedia Comput. Sci.*, vol. 260, pp. 22–31, 2025, doi: 10.1016/j.procs.2025.03.173.

[17] M. deVries *et al.*, "A conceptual framework for identifying and managing system vulnerabilities for diversion of controlled substances in healthcare," *Research in Social and Administrative Pharmacy*, vol. 21, no. 4, pp. 228–238, Apr. 2025, doi: 10.1016/j.sapharm.2025.01.001.

[18] S. Silvestri, S. Islam, S. Papastergiou, C. Tzagkarakis, and M. Ciampi, "A machine learning approach for the NLP-based analysis of cyber threats and vulnerabilities of the healthcare ecosystem," *Sensors*, vol. 23, no. 2, p. 651, 2023.

[19] M. Almaiah, L. Saqr, L. Al-Rawwash, L. Altellawi, R. Al-Ali, and O. Almomani, "Classification of cybersecurity threats, vulnerabilities

and countermeasures in database systems," *Computers, Materials, & Continua*, vol. 81, no. 2, p. 3189, 2024.

[20] I. Bala, I. Pindoo, M. M. Mijwil, M. Abotaleb, and W. Yundong, "Ensuring security and privacy in healthcare systems: a review exploring challenges, solutions, future trends, and the practical applications of artificial intelligence," *Jordan Med. J.*, vol. 58, no. 3, 2024.

[21] H. Riggs *et al*., "Impact, vulnerabilities, and mitigation strategies for cyber-secure critical infrastructure," *Sensors*, vol. 23, no. 8, p. 4060, 2023.

[22] V. Bharathi, "Vulnerability detection in cyber-physical system using machine learning," *Scalable Computing: Practice and Experience*, vol. 25, no. 1, pp. 577–591, 2024.

[23] M. F. Hady and H. P. Pratama, "Implementing Defense-in-Depth Framework on Orange Pi NAS Using Host-Based Security and ZFS," *Journal of Applied Informatics and Computing*, vol. 10, no. 1, pp. 889–899, 2026.

[24] R. F. A. Bakar and M. Rahardi, "Analysis of Gradient Boosting Algorithms with Optuna Optimization and SHAP Interpretation for Phishing Website Detection," *Journal of Applied Informatics and Computing*, vol. 10, no. 1, pp. 664–672, 2026.

[25] N. K. Hamzidah *et al.*, "AI-YOLO Based Smart Laboratory Security for Automated Face Recognition and Suspicious Activity Detection," *Journal of Applied Informatics and Computing*, vol. 10, no. 1, pp. 440–450, 2026.

[26] C. M. Okafor *et al.*, "Mitigating cybersecurity risks in the US healthcare sector," *International Journal of Research and Scientific Innovation (IJRSI)*, vol. 10, no. 9, pp. 177–193, 2023.

[27] M. K. Hasan *et al.*, "A review on security threats, vulnerabilities, and counter measures of 5G enabled Internet-of-Medical-Things," *IET communications*, vol. 16, no. 5, pp. 421–432, 2022.

[28] S. Erniwati, B. Imran, Z. Muahidin, Z. Zaeniah, and J. Juhartini, "SemetonBug: Next-Generation Machine Learning-Powered Code Analyzer for Precision Bug Detection and Dynamic Error Localization," *Journal of Applied Informatics and Computing*, vol. 10, no. 1, pp. 224–231, 2026.

[29] A. A. Slameto and B. R. Kahmas, "Implementation of the Random Forest Algorithm for Anomaly Detection of Phishing Attacks on Computer Networks," *Journal of Applied Informatics and Computing*, vol. 10, no. 1, pp. 204–211, 2026.

[30] O. Panahi, "Secure IoT for healthcare," *European Journal of Innovative Studies and Sustainability*, vol. 1, no. 1, pp. 17–23, 2025.