Security Evaluation of Keycloak-Based Role-Based Access Control in Microservice Architectures Using the OWASP ASVS Framework

Michael Christ Kurniawan [1]*, Indra Gamayanto [2]*, Gabriello Klavin Sanyoto [3]*, Budi Widjajanto [4]*
Faculty of computer science, Information system department, Dian Nuswantoro University, Semarang
112202206752@mhs.dinus.ac.id [1], indra.gamayanto@dsn.dinus.ac.id [2], klavinsanyoto@gmail.com [3], budi.widjajanto@dsn.dinus.ac.id [4]

Article Info

Article history:

Received 2025-10-30
Revised 2025-11-25
Accepted 2025-12-20

Keyword:

Role-Based Access Control,
Data Security,
Mobile Application,
OWASP,
Security Evaluation.

ABSTRACT

The Rocket Car Wash Semarang application operates using a microservice architecture that handles sensitive information such as user identity data, transaction history, and vehicle details. As multiple services interact through authenticated API calls, strong access control is required to protect the system from unauthorized access and privilege escalation. This research evaluates the Keycloak-based Role-Based Access Control (RBAC) implementation by referencing relevant domains of the OWASP Application Security Verification Standard (ASVS) Level 2, specifically V2: Authentication, V3: Session Management, V4: Access Control, and V14: Configuration. The RBAC structure consists of three primary roles—Admin, Owner, and Customer—and the assessment examines the correctness of role–permission mapping and token-based authorization across microservices. The security evaluation was conducted through configuration auditing, API endpoint verification using Postman, JWT token validation, and automated penetration testing using OWASP Zed Attack Proxy (ZAP). The ZAP scan targeted common web vulnerabilities, particularly misconfigurations and weaknesses in HTTP security headers. The results indicate that Keycloak effectively enforces centralized authentication and authorization, with no critical issues such as Broken Access Control identified. However, several non-critical weaknesses were found, including incomplete Content Security Policy (CSP) directives and missing HSTS headers. These findings show that the RBAC implementation meets core ASVS Level 2 controls, while further improvements in security header configuration are required to enhance overall system resilience.

I. Introduction

The evolution of digital technology has spurred a major transformation in the development of modern applications based on the microservice architecture. This model offers scalability, modularity, and ease of service management, but on the other hand, it also increases security complexity, particularly in the aspect of access control between services that interact via REST APIs [1]. In this context, the Role-Based Access Control (RBAC) mechanism becomes a crucial component for restricting user access rights according to their assigned roles. The implementation of Keycloak-based RBAC as an Identity and Access Management (IAM) system has been widely adopted due to its ability to centralize authentication and authorization processes in a distributed microservice system. However, the complexity of communication between microservices often leads to inconsistencies in RBAC policies, potentially causing authorization violations (access control violations) [2].

One of the primary security threats to web-based and API applications is weak access control. According to the 2021 OWASP Top 10 report, Broken Access Control ranks first as the most critical threat in web application security, displacing vulnerabilities such as Injection and Cross-Site Scripting (XSS) that previously dominated [3], [4], [5]. This threat arises when users can access resources or data that they should not be permitted to, resulting from errors in the implementation of authorization rules. In complex microservice systems, a minor error in the authorization configuration between services can create serious exploitation opportunities against sensitive data and system integrity [6].

The OWASP Application Security Verification Standard (ASVS) provides a systematic framework for verifying web application security based on risk level. ASVS version 4.0.3 covers 14 categories and 286 verification criteria that focus on aspects of authentication, authorization, input validation, session management, and system security configuration [7]. This approach assists organizations in evaluating the extent to which security controls have been properly implemented and meet the appropriate security level, especially for applications with medium-to-high risk. Applying this standard in the context of evaluating Keycloak-based RBAC allows for a more objective analysis of the effectiveness of access control policies and the detection of configuration weaknesses in the microservice-based backend system [8].

Previous research has highlighted the importance of automated assessment of RBAC policies in microservice systems using the Systematic Architecture Reconstruction (SAR) approach and static code analysis [1]. These studies showed that inconsistencies between microservice services could lead to potential access violations that are difficult to detect manually. Meanwhile, a quantitative security evaluation model was developed based on OWASP ASVS to measure the security level of web applications in a structured and analytical manner [7]. The combination of these two approaches supports the needs of this research in evaluating the effectiveness and security of the Keycloak-based RBAC system within the microservice architecture.

By referring to the OWASP Application Security Verification Standard (ASVS), this research focuses on the security evaluation of the backend service of the Rocket Car Wash Application, which uses a microservice architecture to support authentication, transactions, and customer data management. The main objective of this study is to assess the system's compliance level with modern application security standards, particularly concerning the role-based access control (RBAC) mechanism centrally governed through Keycloak. The evaluation is directed towards identifying potential vulnerabilities such as Broken Access Control, authentication and authorization configuration errors, and the risk of access token misuse. The analysis results are expected to provide recommendations for improvements that can strengthen the security of microservice services and ensure continuous protection of user data.

Furthermore, various recent studies also indicate that the implementation of DevSecOps and automated OWASP ASVS-based verification can significantly enhance vulnerability detection compared to conventional methods [9]. Other research confirms that the integration of continuous security inspection mechanisms into the software development lifecycle is essential to ensure consistency in security policies and risk control [10]. Thus, this research is highly relevant in the context of implementing Role-Based Access Control (RBAC) using Keycloak, combined with the OWASP ASVS standard, as an effort to strengthen microservice application security in an increasingly complex digital era.

II. METHOD

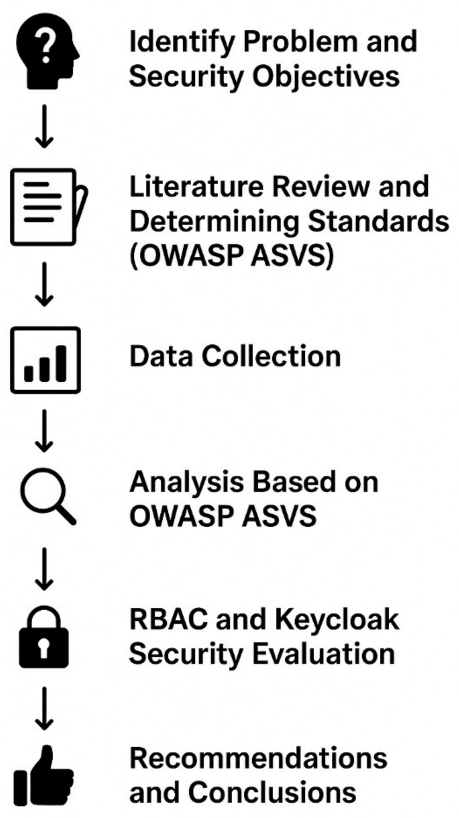

Identify Problem and Security Objectives
Literature Review and Determining Standards (OWASP ASVS)
Data Collection
Analysis Based on OWASP ASVS
RBAC and Keycloak Security Evaluation
Recommendations and Conclusions

Figure 1 Research Process

A. Problem Identification and Security Objectives

The initial stage of the research began by identifying the main problems related to data security and access control in the microservice-based system. In the context of the Rocket Car Wash Application, it was found that the system lacked a centralized access control mechanism, which potentially leads to the misuse of access rights and data leakage. Therefore, this research aims to evaluate the security of Keycloak-based Role-Based Access Control (RBAC) using the OWASP Application Security Verification Standard (ASVS) [11]. This stage also included defining the scope and security objectives to be tested, such as authentication, authorization, and user data protection.

B. Literature Study and Standard Determination (OWASP ASVS)

This stage was conducted to establish the theoretical foundation and security standard that serves as the reference for the research. The literature study encompassed discussions on Keycloak as an Identity and Access

Management (IAM) system, the implementation of Role-Based Access Control (RBAC), and the principles of modern web application security.

As the security evaluation standard, this research referred to the OWASP Application Security Verification Standard (ASVS) version 4.0.3, which is an international framework providing a systematically verifiable list of application security controls [12].

In this research, OWASP ASVS Level 2 was applied because the Rocket Car Wash application manages user data and possesses a medium risk level consistent with the characteristics of a web-based public service system. The security control categories verified included:

1) *V2 – Authentication*:

To ensure the authentication process is secure and not easily compromised.

2) *V3 – Session Management:*

To ensure user tokens and sessions are protected from theft or misuse.

3) *V4 – Access Control:*

To validate the effectiveness of RBAC in preventing privilege escalation.

4) *V14 – Configuration:*

To evaluate the security of communication headers and system configuration against network-based attacks.

C. Data Processing

In this stage, data were collected through four methods:

1) *Application Security Testing using OWASP ZAP:*

Conducted to identify potential vulnerabilities on the backend endpoints, authentication tokens, and Keycloak configuration. The Keycloak system was accessed via a Cloudflare Tunnel, which serves as an additional security layer to conceal the endpoint from direct public access [13].

2) *RBAC Mechanism Testing through Authorization Testing:*

The testing was carried out by ensuring that each role (Admin, Owner, and Customer) could only access serverless API services according to the access rights defined in Keycloak and the microservice. The testing employed role misuse and IDOR (Insecure Direct Object Reference) attempts techniques to measure the effectiveness of the RBAC-based access control [14].

3) *API Performance Testing using JMeter:*

Aimed at ensuring that the service remains responsive when token-based authorization is implemented. The testing was performed with a specific load simulation to observe performance metrics such as throughput, response time, and error rate [15].

4) *Distribution of questionnaires to prospective users:*

To gather user perceptions regarding the security of personal data in the use of the car wash service booking application, a questionnaire survey was conducted involving 103 respondents. The questions focused on users' levels of concern, awareness, and security expectations. The results of the survey were then used as input for formulating the system's security requirements, ensuring that the RBAC-based design and OWASP ASVS–compliant architecture remain aligned with user needs and trust [16].

5) *Test Environment Configuration:*

The security tests were conducted in a controlled staging environment deployed on Google Cloud Platform (GCP) using a Linux-based virtual machine, configured to closely resemble the production setup. Keycloak (version 26.3.5) and the backend microservices were accessible securely through a Cloudflare Tunnel acting as a reverse proxy, with Cloudflare providing TLS termination, WAF protection, and rate limiting. API requests were validated using JWT tokens issued by Keycloak, and the same network and security configuration was maintained consistently across OWASP ZAP, Postman, and JMeter testing. This ensures that the evaluation accurately reflects the system's operational security posture while maintaining safe isolation from real production data.

D. Analysis Based on OWASP ASVS

This stage aimed to evaluate the system's level of compliance with the security controls established in OWASP ASVS 4.0.3, specifically focusing on the aspects of authentication, authorization, input validation, and access control [17]. The findings from the testing were mapped to each verification criterion to identify areas that met the standard as well as areas that still pose potential security risks. Furthermore, the questionnaire results were used to complement the analysis by linking technical findings with the primary needs and concerns of users regarding personal data security. This approach ensured that security enhancement recommendations not only meet technical standards but also align with the trust and experience of the end-user [18].

E. Evaluation of RBAC and Keycloak Security

This stage constituted the core of the research, focusing primarily on evaluating the effectiveness of the Keycloak-based Role-Based Access Control (RBAC) in managing user access rights. The comprehensive evaluation encompassed several critical aspects: analysing the structure of user roles and permissions defined in Keycloak (admin, owner, customer); examining the JSON Web Token (JWT) authentication and authorization mechanism; and validating the security of API endpoints against these assigned user roles. Furthermore, the analysis involved correlating the findings from the ZAP test results with the ASVS controls, paying particular attention to the Broken Access Control aspect. The main objective was to assess the extent to which the RBAC implementation via Keycloak met the OWASP ASVS security standard and demonstrated its capability to minimize potential access rights violations [19].

F. Conclusion and Recommendation

This stage is the final part of the research process, aiming to formulate recommendations for security enhancement and draft conclusions based on the system evaluation results. The recommendations generated will focus on the implementation of supplementary security mechanisms that support the OWASP ASVS Level 2 standard, particularly concerning the aspects of authentication, access control, and security configuration.

The process of drafting recommendations was carried out by analyzing the technical testing results using OWASP ZAP, the evaluation of the Role-Based Access Control (RBAC) implementation in Keycloak, and the user perception survey results. From these three data sources, a comprehensive conclusion was drawn, covering: the effectiveness of the RBAC–Keycloak implementation in managing user access rights based on the Admin, Owner, and Customer roles; and the system's level of compliance with the security controls within the V2: Authentication, V3: Session Management, and V4: Access Control domains of the OWASP ASVS.

III. RESULT AND DISCUSSION

A. Initial Security Issues and Research Objectives

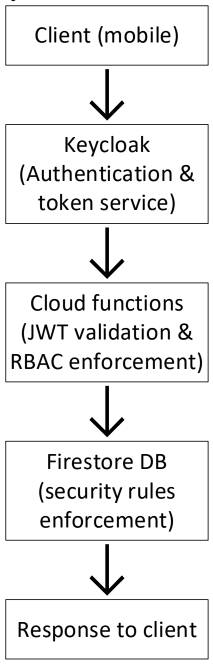

Client (mobile)
Keycloak (Authentication & token service)
Cloud functions (JWT validation & RBAC enforcement)
Firestore DB (security rules enforcement)
Response to client

Figure 2 RBAC Security

Based on the results of initial observation and analysis of the Rocket Car Wash Application system architecture, which adopts a microservice approach, several major issues related to access control security were identified. Each microservice previously managed the authentication and authorization processes separately, resulting in the absence of a centralized access control mechanism. This condition potentially leads to inconsistencies in access rights management, weaknesses in authorization tracking, and an increased risk of privilege escalation and unauthorized access to user data. Furthermore, the system had not implemented a structured security verification standard based on industry best practices, meaning previous security testing was not directed at crucial aspects such as authentication, access token protection, and the restriction of access rights according to user roles.

Based on these findings, this research set the security objectives to provide a centralized access control mechanism through the implementation of Role-Based Access Control (RBAC) using Keycloak, to ensure that authentication and authorization across every microservice are implemented according to user roles, and to evaluate the system's compliance with application security standards based on the OWASP Application Security Verification Standard (ASVS), specifically within the Authentication (V2) and Access Control (V4) categories. Additionally, this research also aims to identify necessary security improvement steps to enhance user data protection. With this identification, the direction of the security evaluation is expected to be more structured and capable of demonstrating measurable security improvements following the implementation of Keycloak-based RBAC.

B. Literature Review and Standard Determination (OWASP ASVS)

The results of the literature study conducted during the methodology phase indicate that the Rocket Car Wash Application possesses the characteristics of a web-based service application involving the management of sensitive user data, such as account information and transaction details. Based on this risk analysis, the relevant security standard to be used as a reference for evaluation is the OWASP Application Security Verification Standard (ASVS) Level 2, which is recommended for applications with a medium risk level.In this stage, the system's security requirements were mapped to the verification categories available in the OWASP ASVS. Four categories were selected due to their direct relevance to the authentication mechanism, session management, and access control within the microservice architecture.

TABLE 1
OWASP ASVS MAPPING

Kode ASVS	Verification Category	Relevance to the System
V2	Authentication	Ensures that only legitimate users can access the system through a secure authentication mechanism.
V3	Session Management	Guarantees that user sessions are protected from misuse and session hijacking.
V4	Access Control	Ensures that each user can only access data and services corresponding to their role within the system.
V14	Configuration	Assesses the security configuration, especially concerning Keycloak integration and microservice configuration.

Based on this mapping, an evaluation instrument was compiled in the form of a verification checklist, which will be used as the indicator of success for the implementation of Keycloak-based RBAC security in the system.

TABLE 2
SUMMARY OF ASVS LEVEL 2 SECURITY CONTROL EVALUATION

ASVS Code	Description	Implementation	Status
V2.1.1	Password must be validated securely	Uses OIDC with Google as the identity provider (no password stored locally)	Pass
V3.2.2	Session tokens must expire	Keycloak access tokens expire within 5–30 minutes	Pass
V4.1.3	Server-side access control enforced	Role validation implemented through Keycloak policies and backend checks	Pass
V14.2.3	CSP must be defined	CSP configuration is partially incomplete	Fail

C. Data Processing

1) Testing and Vulnerability Mapping (OWASP ZAP):

The security testing of the Rocket Car Wash application was conducted using the OWASP Zed Attack Proxy (ZAP) as an automated security scanner to identify potential vulnerabilities in the Keycloak-based authentication and authorization implementation. The testing was focused on the surface endpoints related to the login process, session management, and security configuration at the communication layer.

OWASP ZAP produced several findings with varying risk levels, primarily concerning the aspect of security header configuration and content policy. Generally, no critical vulnerabilities such as SQL Injection, Remote Code Execution, or Authentication Bypass were discovered. However, several alerts appeared related to the lack of security configurations recommended by OWASP ASVS Level 2.

The following table presents the mapping of the test results to the verification categories in OWASP ASVS v4.0, along with a risk analysis and the mitigations that have been implemented.

TABLE 3
MAPPING OF OWASP ZAP SECURITY TEST RESULTS TO OWASP ASVS

No	OWASP ZAP Finding	ASVS Category (v4.0)	Mitigation Performed
1	Absence of Anti-CSRF Tokens	V4 – Access Control	Keycloak already uses the OIDC state parameter, ensuring authentication remains protected.
2	CSP: Failure to Define Directive with No Fallback	V14 – Configuration	Cloudflare Rules: added a more restrictive default-src and script-src CSP.
3	CSP: script-src unsafe-inline	V14 – Configuration	Temporarily allowed for Keycloak UI compatibility; further mitigation planned with nonce/hash.
4	CSP: style-src unsafe-inline	V14 – Configuration	Limited only to the application's own domain; plan to migrate to safer rules.
5	Cross-Domain Misconfiguration	V14 – Configuration / V13 – API Security	Restriction of Access-Control-Allow-Origin solely to the application's official domains.
6	Cookie No HttpOnly Flag	V3 – Session Management	Keycloak utilizes header-based tokens, not cross-domain shared cookies.
7	Cookie with SameSite None	V3 – Session Management	Inherent mitigation through token-based auth using the Authorization Header.
8	Strict-Transport-Security Header Not Set	V9 – Communications	HSTS activated: max-age=31536000; includeSubDomains; preload.
9	Timestamp Disclosure – Unix	V7 – Error Handling and Logging	Low risk and only appears in internal debug logs.
10	X-Content-Type-Options Header Missing	V14 – Configuration	Added header: nosniff

Based on the mapping of the security test results using OWASP ZAP against the RBAC–Keycloak based system implementation, it was found that the majority of findings were in the low to medium risk category and did not directly impact core access control. No evidence of a loophole that would allow the Keycloak authorization process to be bypassed or manipulated was found, thus the authentication and access control mechanisms can be considered operational. The most frequent findings were related to the aspect of configuration hardening on the HTTP layer and Content Security Policy (CSP).

Several mitigation actions have been applied, though some are temporary due to compatibility constraints with the Keycloak interface theme. Although the security configuration was enhanced through Cloudflare integration and tightening security headers, OWASP ZAP still detected a number of alerts. This is attributed to several factors, including: some headers not fully meeting extremely strict security standards (e.g., inline scripts still being permitted), the capability to modify CSP being dependent on support from Keycloak and the reverse proxy, and the limitations of an automated scanner that lacks context on design-based mitigations, such as the difference between token-based authentication and cookie-based session.

2) *Source Analysis of Each Security Finding:*

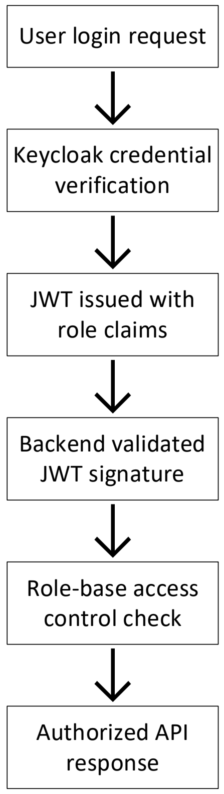
To increase the transparency of the security evaluation, each ZAP finding was analyzed to determine its origin within the system architecture. The findings were categorized based on whether the issue originated from Keycloak configuration, microservice backend code, Cloudflare / reverse proxy, or frontend/mobile client behavior. This classification is important to ensure that improvement recommendations are correctly mapped to the responsible system component.

TABLE 4
SUMMARY OF SECURITY FINDINGS AND THEIR ROOT CAUSES

Finding	Root Cause
Anti-CSRF	Not caused by Keycloak; originates from the application's UI layer, since Keycloak already uses the OIDC state parameter.
CSP warnings	Caused by Cloudflare combined with the default Keycloak theme, which includes inline scripts.
Missing security headers	Caused by the reverse proxy configuration (Cloudflare) prior to optimization.
HSTS disabled	Due to SSL/TLS configuration settings on Cloudflare.
Cookie SameSite/HttpOnly	Not applicable because the system uses JWT in the Authorization header, not cookies.
CORS misconfiguration	Originated from backend CORS settings before being restricted to official domains.

This analysis confirms that most findings were not caused by weaknesses in Keycloak itself, but rather by configuration gaps in Cloudflare, reverse-proxy headers, or UI-level CSP limitations. The RBAC and token validation mechanisms in Keycloak remained unaffected by these findings. Therefore, the identified risks are categorized as configuration-level issues rather than authorization or authentication failures.

3) *Testing and Vulnerability Mapping (Postman):*


User login request
Keycloak credential verification
JWT issued with role claims
Backend validated JWT signature
Role-base access control check
Authorized API response
Figure 3 Postman testing flow

Testing was conducted to ensure that the RBAC implementation via Keycloak effectively restricted access to endpoints based on user roles. The scenarios included role validation, authentication token validation, and session condition checks. The testing was performed using Postman on two primary endpoints: manageRole and sendNotification.

TABLE 5
POSTMAN TEST RESULTS MAPPING

Endpoint	Test Result	Conclusion
manageRole	All test scenarios were successful as expected.	Access is strictly restricted to users with the Owner role.
sendNotification	All test scenarios were successful as expected.	Notifications can only be sent by the Admin/Owner roles.

Based on the testing results across all access scenario variations, no indication of privilege escalation or access rights violations between user roles was found. Every request submitted using an invalid token was successfully rejected by the system, and all expired sessions automatically blocked access to resources. The role restriction mechanism also functioned effectively, aligning with the Role-Based Access Control (RBAC) configuration in Keycloak. Thus, the Keycloak-based RBAC implementation has functioned in accordance with the system's security requirements. All tests demonstrated that requests inconsistent with role authorization were successfully restricted, thereby confirming that access control on the API fulfills the Principle of Least Privilege and supports compliance with the OWASP ASVS V4: Access Control security standard.

4) *Testing and Vulnerability Mapping (Firestore Rules):*

Unit testing was performed on the Firestore Security Rules to ensure that access control at the database layer was consistent with the RBAC policies defined in Keycloak. A total of 15 scenarios were tested using Firebase Emulator + Jest, involving the roles of Owner, Admin, and Customer with varied write/read scenarios across different collections.

TABLE 6
MAPPING OF FIRESTORE RULES TESTING RESULTS

Component	Test Result	Conclusion
Personal User Data Access	Compliant with rules	User can only write their own data.
Business Area Access	Owner only	Management of business areas is restricted.
Category, Operational, Stock, Holiday, Voucher Access	Compliant with rules	No privilege escalation.
Notification Access	Compliant with rules	Only Admin/Owner can create notifications.
Internal System Collection Access	Compliant with rules	Ordinary users are fully denied access

The testing results showed that the system had no loopholes for Insecure Direct Object Reference (IDOR) or illegal cross-user access. All requests submitted without authentication were consistently rejected, ensuring that every interaction with resources could only be performed by verified users. Furthermore, access control at the database layer proved to be consistent with the role policies implemented via Keycloak. This security mechanism forms a layered access control system, where Keycloak restricts access rights at the API level, while Firestore Security Rules ensure no privilege escalation occurs down to the data storage layer. Accordingly, the implementation of access control in this system has met the security requirements in the OWASP ASVS V4 – Access Control domain.

5) *API Performance Testing (JMeter) Results*:

Testing 500 samples per endpoint showed that the services remained responsive with a Zero Error Rate.

TABLE 7
MAPPING OF JMETER TEST RESULTS

Endpoint	Avg Time (ms)	Throughput	Error	Evaluation
Manage Role	527	47.8/sec	0%	Remains stable despite heavy operation.
Send Notification	291	49.3/sec	0%	Normal
Midtrans Transaction	184	49.8/sec	0%	Highly responsive.
Exchange Token	90	50.8/sec	0%	Best performance.

6) *User Expectations Regarding Application Security:*

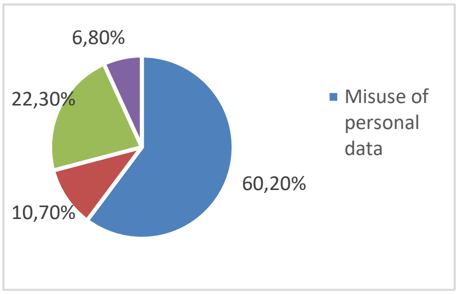


Figure 4 Analysis of User Concern Levels Regarding the Security of Car Wash Booking Applications

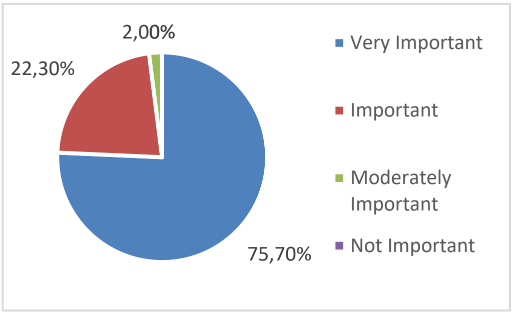


Figure 5 Perception of the Importance Level of Personal Data Security in Car Wash Scheduling Applications.

Quote:

- "I want every login process to be secure and not easily compromised by others."
- "Users must be able to control access rights and choose which data they want to share."
- "There must be transparency regarding how the application manages and protects my data."
- "If a security breach occurs, I expect immediate notification to users and swift action from the developer."

Based on the questionnaire results, respondents generally expect the application to possess a robust security system concerning authentication, personal data protection, and transaction security. Transparency in data management and a rapid response when an incident occurs are deemed essential in building user trust toward the car wash scheduling application.

D. Analysis Based on OWASP ASVS

1) *Analysis of User Perception:*

User perception analysis was conducted to understand user expectations and trust levels regarding the security of the car wash scheduling application. The questionnaire results indicated that the majority of users consider authentication security, personal data protection, and transaction security as aspects that must be prioritized in a digital service application. This reflects an increasing user awareness of the risks of cybercrime and personal data misuse.

From a technical perspective, these requirements align with the security standards recommended by the OWASP Application Security Verification Standard (ASVS), particularly in the following categories:

- V2 – Authentication: Which emphasizes the importance of strong authentication, session management, and the prevention of illegal access.
- V4 – Access Control: Which ensures that only authorized roles can access specific features or data.

This alignment demonstrates that user aspirations are not only perceptual but are also relevant to industry security standards. In other words, the implementation of a security mechanism based on Role-Based Access Control (RBAC) via Keycloak in this system is an appropriate approach to meet both user needs and modern security standards.

Furthermore, several respondents highlighted the importance of a rapid response to security incidents, such as notifications when suspicious activity or service disruption occurs. This indicates that users expect not only preventive security controls but also effective incident detection and response as part of a comprehensive security strategy. Based on these findings, it can be concluded that the enhancement of security on the backend system using RBAC and Keycloak integration not only provides technical fulfillment of the OWASP ASVS standard but also directly contributes to building user trust. Thus, the security implementation in this application not only strengthens the protection of digital assets but also supports the continued use of the application and increases user satisfaction with the provided services.

2) Correlation of Technical Findings with User Needs:

The findings from OWASP ZAP revealed several risks in the security configuration areas, such as CSP and Strict-Transport-Security, for which mitigations have been performed. Meanwhile, the RBAC–Keycloak based authentication and access control aspects have met the ASVS Level 2 standard.

TABLE 8
SURVEY RESULT

Technical Finding	Related User Concern
CSP and Security Header Risk	Protection of personal data and transactions.
Token Validation and RBAC	Account compromise, misuse of access rights.
HSTS and Secure Communication Configuration	Risk of data theft on public networks.
Transparency of Risk Mitigation	Expectation of notification during an incident.

It can be concluded that the reinforcement of security configuration and the transparency of information are crucial keys in enhancing application security.

3) Evaluation of OAuth2/OpenID Connect Integration:

The Rocket Car Wash Application uses Google Sign-In integrated with Keycloak through the OpenID Connect (OIDC) protocol. The authentication process follows the Authorization Code Flow, where Google acts as an external identity provider and Keycloak issues the final access token (JWT) consumed by the microservices. Token validation is performed by verifying the signature, issuer, audience, and expiration of the JWT. Session management is handled by Keycloak's SSO session, while the backend services validate access based on the role embedded in the token. This ensures that the RBAC mechanism remains consistent regardless of the authentication source.

Social login via Google was not the object of security testing, as OWASP ASVS focuses on application-side controls. Therefore, the evaluation only measured token validation, session management, and RBAC authorization generated by Keycloak after Google login is completed.

E. Evaluation of RBAC and Keycloak Security

This evaluation aimed to assess the effectiveness of the Role-Based Access Control (RBAC) implementation using Keycloak in securing access to backend services. Testing was performed by verifying authorization controls on every API endpoint, JWT (JSON Web Token) validation, and the consistency of access based on the user role structure (Admin, Owner, and Customer).

1) Role Structure and Permission Management:

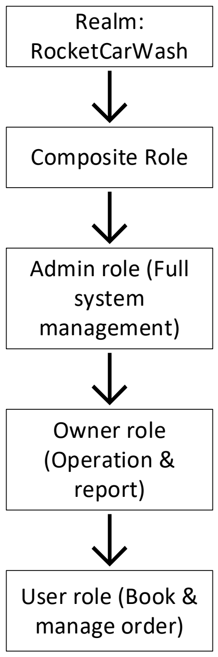
Keycloak successfully implemented the segregation of access rights based on the operational scenarios within the system. Admin and Owner roles function as full system managers, while users possess only limited access rights consistent with service needs.

TABLE 9
IMPLEMENTED CONTROLS MEETING ASVS V4: ACCESS CONTROL REQUIREMENTS

ASVS Control	System Achievement
V4.1 – Enforced server-side access control	Achieved through Keycloak policies.
V4.2 – Role-based access segmentation	The admin–owner–customer role structure functions well.
V4.3 – Prevention of privilege escalation	Data write actions are validated based on the role.

Consequently, authorization restriction is implemented server-side and does not solely rely on the mobile application side.

2) *JWT Token Validation and Authentication:*


Realm: RocketCarWash
Composite Role
Admin role (Full system management)
Owner role (Operation & report)
User role (Book & manage order)
Figure 6 RBAC model

The system ensures that service access is only granted to users with a valid and unexpired token.

TABLE 10
TOKEN VALIDATION AND AUTHENTICATION TEST RESULTS

Token Condition	System Result	ASVS Compliance
Token valid	Access granted	V2 Authentication compliance
Invalid or Modified Token	Denied	Signature validation applied
Expired Token	Denied & redirects to login	No access after session ends

This demonstrates that the system has met ASVS V2: Authentication, particularly in the aspects of token verification and session management.

3) API Endpoint Validation Based on Role:

Each endpoint was tested using various authorization scenarios to ensure that the access control mechanism functioned according to the security design. The test results showed that unauthorized users could neither access nor modify sensitive data, and no indication of privilege escalation was found in any of the tested requests. Furthermore, all write actions have been restricted only to the Admin and Owner roles, consistent with the policies established in the RBAC system. Therefore, the Business Logic Access Control mechanism is proven to function effectively in maintaining application data integrity and security.

4) Correlation with OWASP ZAP Results:

OWASP ZAP did not find Broken Access Control on the tested Keycloak endpoints. The alerts that appeared stemmed from security header configurations, not from the authorization aspect.

Thus, it can be concluded that the core security controls directly related to RBAC have successfully prevented unauthorized access, even though minor warnings exist regarding the application's security configuration.

5) Analysis and Implication:

Based on the evaluation results above, the implementation of Keycloak-based RBAC is proven effective in maintaining access rights according to each user's role, thereby preventing the occurrence of authorization misuse or privilege escalation. The system has also fulfilled the majority of security controls established in OWASP ASVS Level 2, specifically in the Authentication and Access Control domains. Furthermore, the implemented security measures are aligned with user expectations regarding personal data protection, transaction security, and the transparency of access management within the application.

TABLE 11
AREAS FOR SECURITY IMPROVEMENT

Potential Improvement	Security Impact	Recommendation
Lack of granular audit logs	Difficult to conduct incident investigation.	Integrate Keycloak Event Listener.
Multi-Factor Authentication Multi-Factor Authentication (MFA) relies on Google Account settings	MFA is available but depends on the user's Google security configuration; users without MFA enabled remain at higher risk	Encourage users to enable MFA in their Google accounts and consider enforcing MFA at the application level for critical roles.
Sub-optimal security headers	Small risk of MITM attacks still present.	Fix HSTS & CSP on the server.

IV. CONCLUSION

This research focused on evaluating the implementation of the Role-Based Access Control (RBAC) mechanism using Keycloak on a car wash service booking backend system. The evaluation was measured against the OWASP ASVS Level 2 security standard and supported by vulnerability testing via OWASP ZAP and a user perception survey. The analysis results indicate that the RBAC implementation via Keycloak proved effective in managing user access rights based on the Admin, Owner, and User roles. Testing of the API endpoints showed that the system successfully prevents illegal access and potential privilege escalation, thereby meeting the security controls in the V4: Access Control domain of OWASP ASVS. Furthermore, the authentication and authorization mechanisms are compliant with the OWASP ASVS Level 2 standard, specifically in the V2: Authentication domain. The JWT token validation functions correctly through digital signature verification, token expiration checks, and the rejection of invalid or expired tokens. Minor findings identified by OWASP ZAP, such as incomplete CSP or missing HSTS, were configuration-level issues and have been mitigated without affecting the integrity of the RBAC mechanism. Additionally, user survey results reinforce that the applied security controls align with user expectations regarding personal data protection, account safety, and transaction security.

In addition to these findings, several Keycloak-specific enhancements are recommended to ensure a more resilient RBAC implementation. The system should enable detailed audit logging through Keycloak Event Listener to improve traceability of authentication, token issuance, and role-management activities. The enforcement of PKCE within Keycloak Client Policies is also advised, particularly for mobile and public clients, to reduce the risk of authorization code interception. Furthermore, implementing refresh-token rotation, shortening token lifetime, and disabling unused features such as Direct Access Grants will minimize the possibility of token replay attacks. For microservices that handle sensitive operations, the adoption of Keycloak Authorization Services can centralize authorization logic and maintain consistent access-control enforcement across services. These improvements will strengthen the overall security posture of the system and help ensure continued compliance with modern OAuth2/OIDC and ASVS security requirements.

REFERENCES

[1] D. Das, A. Walker, V. Bushong, J. Svacina, T. Cerny, and V. Matyas, “On automated RBAC assessment by constructing a centralized perspective for microservice mesh,” *PeerJ Comput. Sci.*, vol. 7, p. e376, Feb. 2021, doi: 10.7717/peerj-cs.376.

[2] S.-F. Wen and B. Katt, “A quantitative security evaluation and analysis model for web applications based on OWASP application security verification standard,” *Comput. Secur.*, vol. 135, p. 103532, Dec. 2023, doi: 10.1016/j.cose.2023.103532.

[3] K. Abdulghaffar, N. Elmrabit, and M. Yousefi, “Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners,” *Computers*, vol. 12, no. 11, p. 235, Nov. 2023, doi: 10.3390/computers12110235.

[4] B. Ünver and R. Britto, “Automatic Detection of Security Deficiencies and Refactoring Advises for Microservices,” in *2023 IEEE/ACM International Conference on Software and System Processes (ICSSP)*, Melbourne, Australia: IEEE, May 2023, pp. 25–34. doi: 10.1109/ICSSP59042.2023.00013.

[5] “OWASP Top 10:2021.” Accessed: Oct. 23, 2025. [Online]. Available: https://owasp.org/Top10/

[6] A. Chatterjee and A. Prinz, “Applying Spring Security Framework with KeyCloak-Based OAuth2 to Protect Microservice Architecture APIs: A Case Study,” *Sensors*, vol. 22, no. 5, Art. no. 5, Jan. 2022, doi: 10.3390/s22051703.

[7] “Design and Security Evaluation of IAM Module in Microservice Architecture Using Keycloak | The Indonesian Journal of Computer Science.” Accessed: Jul. 28, 2025. [Online]. Available: https://ijcs.net/ijcs/index.php/ijcs/article/view/4854

[8] A. Venčkauskas, D. Kukta, Š. Grigaliūnas, and R. Brūzgienė, “Enhancing Microservices Security with Token-Based Access Control Method,” *Sensors*, vol. 23, no. 6, p. 3363, Mar. 2023, doi: 10.3390/s23063363.

[9] P. Billawa, A. B. Tukaram, N. E. D. Ferreyra, J.-P. Steghöfer, R. Scandariato, and G. Simhandl, “SoK: Security of Microservice Applications: A Practitioners’ Perspective on Challenges and Best Practices,” in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, Aug. 2022, pp. 1–10. doi: 10.1145/3538969.3538986.

[10] K. V. Palavesam, S. V. Arcot, M. V. Krishnamoorthy, and E. G V, “Building Automated Security Pipeline for Containerized Microservices,” *J. Adv. Math. Comput. Sci.*, vol. 40, no. 2, pp. 53–66, Feb. 2025, doi: 10.9734/jamcs/2025/v40i21969.

[11] M. S. Rahaman, S. N. Tisha, E. Song, and T. Cerny, “Access Control Design Practice and Solutions in Cloud-Native Architecture: A Systematic Mapping Study,” *Sensors*, vol. 23, no. 7, Art. no. 7, Jan. 2023, doi: 10.3390/s23073413.

[12] A. Oluwaferanmi, “Ensuring Authentication and Authorization Security Using OWASP ASVS Controls in Enterprise Applications”.

[13] S. Sutradhar, S. Karforma, R. Bose, S. Roy, S. Djebali, and D. Bhattacharyya, “Enhancing identity and access management using Hyperledger Fabric and OAuth 2.0: A block-chain-based approach for security and scalability for healthcare industry,” *Internet Things Cyber-Phys. Syst.*, vol. 4, pp. 49–67, Jan. 2024, doi: 10.1016/j.iotcps.2023.07.004.

[14] E. Amissah and F. Bentil, “Exposing Insecure Direct Object Reference (IDOR) Vulnerabilities in Academic Publication Platforms: A Case Study),” *Int. J. Eng. Res.*, vol. 13, no. 08.

[15] N. R. P. Hutasuhut, M. G. Amri, and R. F. Aji, “Security Gap in Microservices: A Systematic Literature Review,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 12, 2024, doi: 10.14569/IJACSA.2024.0151218.

[16] A. Bambhore Tukaram, S. Schneider, N. E. Díaz Ferreyra, G. Simhandl, U. Zdun, and R. Scandariato, “Towards a Security Benchmark for the Architectural Design of Microservice Applications,” in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, Vienna Austria: ACM, Aug. 2022, pp. 1–7. doi: 10.1145/3538969.3543807.

[17] M. Tahir, A. Abdullah, N. I. Udzir, and K. A. Kasmiran, “A novel approach for handling missing data to enhance network intrusion detection system,” *Cyber Secur. Appl.*, vol. 3, p. 100063, Dec. 2025, doi: 10.1016/j.csa.2024.100063.

[18] F. Di Nocera, G. Tempestini, and M. Orsini, “Usable Security: A Systematic Literature Review,” *Information*, vol. 14, no. 12, p. 641, Nov. 2023, doi: 10.3390/info14120641.

[19] W. S. Admass, Y. Y. Munaye, and A. A. Diro, “Cyber security: State of the art, challenges and future directions,” *Cyber Secur. Appl.*, vol. 2, p. 100031, 2024, doi: 10.1016/j.csa.2023.100031.