A Secure Property Rental Platform with DevSecOps Integration

Muhammad 'Adli Bin Mohd Ali, Amir Hafizi Bin Musa, Nik Muhammad Haziq bin Nik Hasni

Faculty of Computer and Mathematical Sciences
Universiti Teknologi MARA (UiTM), Tapah Campus, Perak, Malaysia

Abstract

The property rental industry in Malaysia faces significant challenges related to trust, security, and fraud prevention in digital transactions. This paper presents a secure property rental platform that integrates DevSecOps practices to address critical security vulnerabilities aligned with OWASP standards. The platform implements six core security modules: Secure Login with Multi- Factor Authentication (MFA), Secure API Gateway, Digital Agreement System, Smart Notification and Alert System, Activity Log Dashboard, and CI/CD Security Testing. Built using modern technologies including Node.js, Express.js, Next.js, and PostgreSQL, this rental platform demonstrates how security-first development practices can be effectively integrated into web and mobile applications. The implementation features cryptographically secure OTP generation, SHA-256 signature hashing for digital agreements, automated threat detection with risk scoring, and a comprehensive CI/CD pipeline with integrated security scanning tools including CodeQL, Gitleaks, and Trivy. Results show successful deployment with all security modules operational, providing a robust solution for secure property rental transactions in the Malaysian market.

Keywords—DevSecOps, Property Rental Platform, Multi-Factor Authentication, Digital Signatures, OWASP, CI/CD Security, Zero-Trust Architecture

I. INTRODUCTION

The property rental market in Malaysia has undergone significant digital transformation in recent years, with an increasing number of transactions being conducted through online platforms. However, this digital shift has introduced new challenges related to trust, security, and fraud prevention [1]. Traditional rental processes often involve manual documentation, lack of verification mechanisms, and insufficient security measures, leaving both landlords and tenants vulnerable to fraudulent activities.

Existing property rental platforms in Malaysia primarily focus on listing properties and connecting landlords with potential tenants, with limited emphasis on comprehensive security features. This gap presents opportunities for malicious actors to exploit vulnerabilities in authentication systems, manipulate digital agreements, and compromise sensitive user data. The Open Web Application Security Project (OWASP) has identified critical vulnerabilities in mobile and web applications, including broken authentication (M1), insecure data storage (M2), and insufficient cryptography (M5), which are prevalent in many existing rental platforms [2].

The web application addresses these challenges by implementing a security-first approach through DevSecOps integration. DevSecOps represents the evolution of DevOps practices by embedding security considerations throughout the software development lifecycle, rather than treating security as an afterthought [3]. This approach enables continuous security testing, automated vulnerability detection, and rapid response to emerging threats.

This paper presents the design, implementation, and evaluation of a rental platform web application, a secure property rental platform developed by Team ClaRity for the UiTM DevOps Challenge. The platform incorporates six core security modules that address OWASP-identified vulnerabilities while providing a seamless user experience for property rental transactions. The contributions of this paper include: (1) a comprehensive security architecture for rental platforms, (2) implementation of MFA-based authentication with risk scoring, (3) a digital agreement system with cryptographic signature validation, (4) an automated CI/CD security pipeline, and (5) a threat intelligence system with zero-trust access logic.

II. LITERATURE REVIEW

A. Property Rental Platforms and Trust Challenges

The digitization of property rental services has transformed how landlords and tenants interact in the real estate market. Modern platforms such as PropertyGuru, iProperty, and Mudah.my have established the foundation for online property listings in Malaysia [4]. However, these platforms primarily serve as listing aggregators with limited security features for transaction verification and user authentication.

Research by Ahmad et al. [5] highlighted the trust deficit in online property transactions, where 67% of surveyed users expressed concerns about fraud and identity verification. This finding underscores the need for robust authentication mechanisms and secure transaction workflows in rental platforms. Additionally, studies indicate that property scams in Malaysia increased by 30% during the pandemic period, emphasizing the urgency for secure digital solutions.

B. Web Application Security and OWASP Standards

The OWASP Mobile Top 10 provides a comprehensive framework for identifying and mitigating security risks in mobile and web applications [2]. Key vulnerabilities relevant to rental platforms are summarized in Table I.

TABLE I. OWASP MOBILE TOP 10 VULNERABILITIES

Code	Vulnerability	Impact on Rental Platforms
M1	Improper Platform Usage	Credential exposure, session hijacking
M2	Insecure Data Storage	Personal data leakage
M3	Insecure Communication	Man-in-the-middle attacks
M5	Insufficient Cryptography	Signature forgery, document tampering
M6	Insecure Authorization	Unauthorized access to admin features

Studies have shown that implementing multi-layered security approaches significantly reduces vulnerability exploitation rates [6]. Authentication mechanisms combining passwords with one-time passwords (OTP) demonstrate 99.9% effectiveness against credential stuffing attacks [7].

C. Multi-Factor Authentication and Account Security

Multi-Factor Authentication (MFA) has become the gold standard for secure user authentication. Research by Bonneau et al. [8] demonstrated that MFA reduces account compromise by up to 99.9% compared to password-only authentication. Time-based One-Time Passwords (TOTP) and email-based

OTP verification provide practical second-factor authentication while maintaining user convenience.

The implementation of account lockout mechanisms after repeated failed attempts effectively mitigates brute force attacks, with optimal thresholds identified at 5 attempts with 15-minute lockout periods [9]. Combined with device fingerprinting and IP-based tracking, these mechanisms provide comprehensive account protection.

D. DevSecOps and Shift-Left Security

DevSecOps integrates security practices into the DevOps workflow, enabling "shift-left" security where vulnerabilities are identified and addressed early in the development cycle [3]. Static Application Security Testing (SAST) tools such as CodeQL and ESLint analyze source code for potential vulnerabilities, while dependency scanning tools like Trivy identify known vulnerabilities in third-party libraries [10].

GitHub Actions provides a robust platform for implementing automated security pipelines, enabling continuous security testing on every code commit and pull request [11]. This automation reduces the time between vulnerability discovery and remediation from weeks to hours. The key components of a comprehensive CI/CD security pipeline are shown in Table II.

TABLE II. CI/CD SECURITY PIPELINE COMPONENTS

Component	Tool	Function
SAST	CodeQL, ESLint	Source code vulnerability analysis
Secret Detection	Gitleaks	Credential exposure prevention
Dependency Scan	Trivy, npm audit	Third-party vulnerability detection
Type Checking	TypeScript	Type safety verification

E. Digital Signatures and Data Integrity

Electronic signatures have gained legal recognition in Malaysia under the Digital Signature Act 1997 and Electronic Commerce Act 2006 [12]. SHA-256 hashing provides cryptographic assurance of document integrity, with collision resistance making it computationally infeasible to generate two documents with identical hashes [13]. Research indicates that digital signature systems with audit trails provide non-repudiation and accountability in electronic transactions, essential features for legally binding rental agreements [14].

F. Research Gap and Contribution

While existing literature addresses individual security components, there is limited research on integrated DevSecOps implementations specifically designed for property rental platforms in the Malaysian context. This study addresses this gap by presenting a comprehensive security architecture that combines authentication, digital signatures, threat detection, automated security testing which will prevent the system from being compromised by traditional security risks [15], and zero-trust access logic within a unified platform.

III. RESEARCH METHOD

A. Development Methodology

The web application was developed using an Agile methodology with DevSecOps principles integrated throughout the software development lifecycle. The development process followed iterative sprints with continuous security assessment at each stage. Security requirements were defined alongside functional requirements, ensuring that security considerations were embedded from the initial planning phase. The team utilized GitHub for version control with branch protection rules enforcing code review and security checks before merging.

B. Technology Stack

The technology selection was guided by security capabilities, community support, and performance characteristics. Table III summarizes the core technology stack.

TABLE III. TECHNOLOGY STACK OVERVIEW

Layer	Technology	Version
Backend Runtime	Node.js	v20+
API Framework	Express.js	v4.18
Database ORM	Prisma	v5.3
Database	PostgreSQL (Supabase)	v15
Frontend Framework	Next.js + React	v16 / v19
Mobile Framework	Capacitor	v8
Security Headers	Helmet.js	Latest

C. Security Architecture

The security architecture follows a defense-in-depth approach with multiple layers of protection, as illustrated in Fig. 1. The five security layers include: (1) Network Layer with HTTPS/TLS encryption and CORS configuration, (2) Application Layer with input validation, XSS protection, and rate limiting, (3) Authentication Layer with JWT and MFA, (4) Data Layer with encrypted storage and hashed credentials, and (5) Monitoring Layer with real-time threat detection and audit logging.

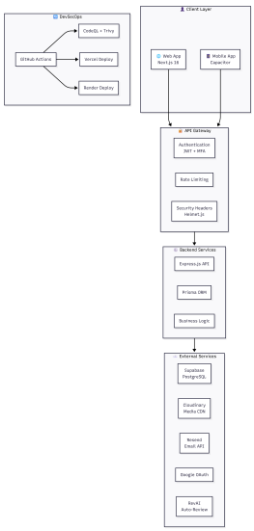

Fig. 1. System Architecture Overview showing client layer, API gateway, backend services, external services, and DevSecOps pipeline

D. Module Development

Six core security modules were developed to address specific OWASP vulnerabilities, as shown in Table IV.

TABLE IV. CORE SECURITY MODULES

Module	Security Focus	OWASP
Module 1	Secure Login & MFA	M1, M3
Module 2	Secure API Gateway	M5, M6
Module 3	Digital Agreement	Integrity
Module 4	Smart Notification	Monitoring
Module 5	Activity Log Dashboard	Accountability
Module 6	CI/CD Security Testing	DevSecOps

IV. IMPLEMENTATION

A. Module 1: Secure Login & Multi-Factor Authentication

The authentication module implements cryptographically secure OTP generation using the Node.js crypto library. The OTP generation process utilizes crypto.randomBytes() to generate 4 random bytes, which are converted to a 6-digit

code. The implementation ensures uniform distribution across all possible OTP values, as shown in the code snippet below:

```
// OTP Generation (Cryptographically Secure)
                generateOtpCode() {
                                const randomBytes =
crypto.randomBytes(4);
                               const randomNumber =
randomBytes.readUInt32BE(0);
                return (randomNumber % 1000000)
                .toString().padStart(6, '0');
                }
```

OTPs are stored as bcrypt hashes with a 5-minute expiration window. The system limits OTP verification attempts to 5 per session to prevent brute force attacks. Table V summarizes the rate limiting configuration for authentication endpoints.

TABLE V. MFA RATE LIMITING CONFIGURATION

Endpoint	Limit	Window	Purpose
/mfa/verify	5	5 min	OTP verification
/mfa/resend	3	1 min	OTP resend
/mfa/enable	3	1 min	Enable MFA
/mfa/disable	3	1 min	Disable MFA

Role-Based Access Control (RBAC) distinguishes between User (tenant/landlord) and Admin roles, with middleware enforcing access restrictions on protected endpoints. The authorization middleware implementation is shown below:

```
// Role Authorization Middleware
                const authorize = (...roles) => {
                return (req, res, next) => {
                  if (!roles.includes(req.user.role))
{

securityLogger.logSuspiciousActivity(
                req, 'Unauthorized access attempt'
                );
                return res.status(403).json({
                message: 'Insufficient permissions.'
                });
                }
                next();
                };
                };
```

Account lockout is triggered after 5 failed login attempts, with a 15-minute lockout period and email notification to the account holder. Google OAuth integration provides alternative authentication through Passport.js, with deep linking support for mobile application callback handling via the custom URL scheme (rentverseclarity://). The complete authentication flow is illustrated in Fig. 2.

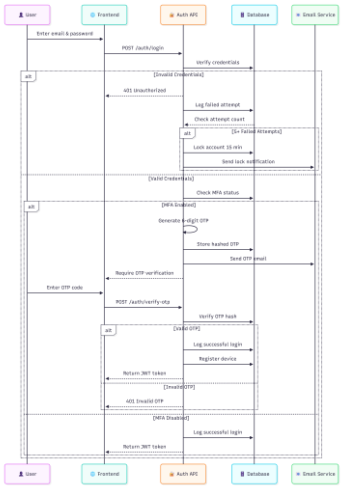

Fig. 2. Authentication Flow sequence diagram showing MFA/OTP verification process

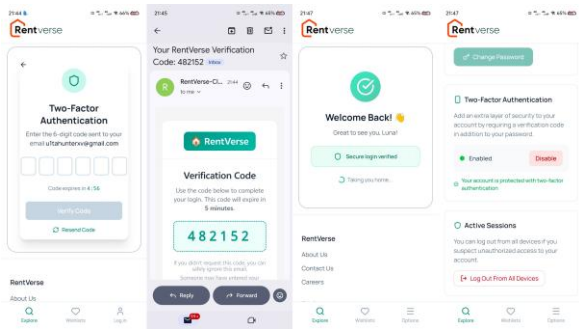

Fig. 3. MFA Login interface showing OTP verification screens

B. Module 2: Secure API Gateway

The API Gateway implements multiple rate limiting tiers to prevent abuse and ensure service availability. Table VI presents the comprehensive rate limiting strategy.

TABLE VI. API GATEWAY RATE LIMITING TIERS

Limiter	Requests	Window	Purpose
Global	2000	15 min	DDoS prevention
Auth	5	15 min	Brute force prevention
OTP	5	5 min	OTP abuse prevention
Strict	3	1 min	Sensitive operations
API	2000	15 min	General protection

JWT authentication includes token blacklisting capability, enabling immediate session invalidation upon logout. The blacklist is maintained in-memory with automatic cleanup of expired tokens every hour. The token validation with blacklist check is implemented as follows:

```
// JWT Verification with Blacklist Check
                if (isBlacklisted(token)) {
                return res.status(401).json({
                message: 'Token has been revoked.'
                });
                }
                    const decoded = jwt.verify(token,
process.env.JWT_SECRET);
```

Helmet.js configures security headers including Content-Security-Policy, X-XSS-Protection, and Strict-Transport-Security. Request validation middleware sanitizes inputs using the xss library and detects SQL injection patterns through regex matching.

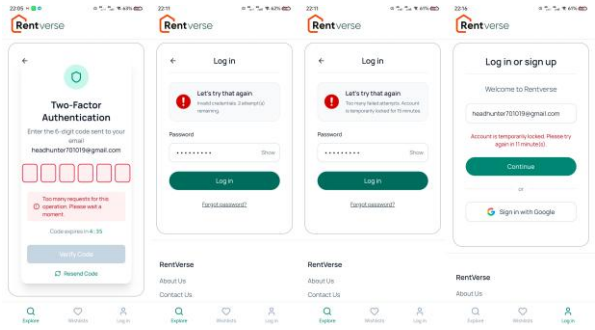

Fig. 4. API Security with rate limiting and account lockout

C. Property Listing & Approval Flow

The platform implements a comprehensive property listing workflow with optional AI-powered auto-review using RevAI integration. Landlords create property listings by uploading photos, setting prices, and providing details. The system can utilize RevAI for automated content analysis, flagging suspicious listings for manual admin review. Properties go through a structured approval workflow before becoming visible on the platform. Fig. 5 illustrates this process.

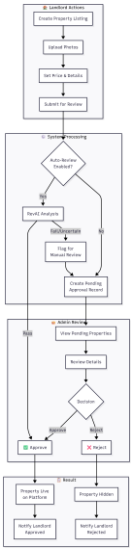

Fig. 5. Property Listing and Approval Flow with AI-assisted review

D. Module 3: Digital Agreement System

The digital agreement module implements SHA-256 signature hashing for document integrity and non-repudiation. The signature creation algorithm combines multiple data points to create a unique, tamper-evident hash:

```
// Signature Hash Creation
                    createSignatureHash(signature,
timestamp, leaseId, userId) {
                                        const   data   =
  `${signature}|${timestamp}|${leaseId}|${userId}`;
                 return crypto.createHash('sha256')
                .update(data).digest('hex');
                }
```

The workflow supports dual-party signing with state management through six statuses, as detailed in Table VII.

TABLE VII. AGREEMENT WORKFLOW STATES

Status	Description	Actions Available
DRAFT	Agreement created	Edit, Delete, Initiate
PENDING_LANDLORD	Awaiting landlord	Sign, Cancel
PENDING_TENANT	Landlord signed	Sign, Cancel
COMPLETED	Both signed	Download, View
EXPIRED	Deadline passed	View Only
CANCELLED	Agreement cancelled	View Only

Each action is logged to an audit trail recording the action type, performer, timestamp, and IP address. PDF generation uses Puppeteer to create formatted lease documents, which are stored securely on Cloudinary CDN. Canvas-based signature capture enables users to draw their signatures, which are then hashed with contextual metadata for non-repudiation. The complete signing flow is shown in Fig. 6.

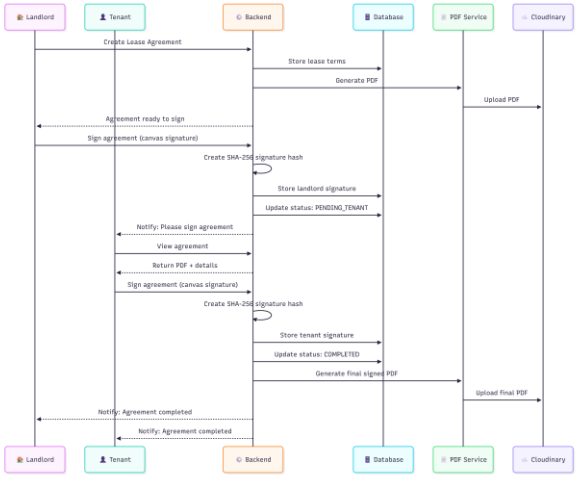

Fig. 6. Digital Agreement Signing Flow sequence diagram

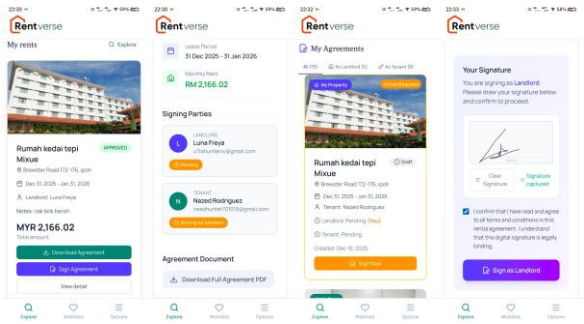

Fig. 7. Digital Agreement interface

E. Module 4: Smart Notification & Alert System

The notification system implements automated security alerts for various threat indicators. Table VIII summarizes the alert types and their triggers.

TABLE VIII. SECURITY ALERT TYPES

Alert Type	Trigger Condition	Email
NEW_DEVICE	Unrecognized device login	Yes
GOOGLE_OAUTH	OAuth login detected	Yes
MULTIPLE_FAILURES	3+ failures in 5 min	Yes
ACCOUNT_LOCKED	5 failed attempts	Yes
SUSPICIOUS_TIMING	Login 2-5 AM	No
PASSWORD_CHANGED	Password update	Yes

Risk scoring calculates a composite threat level (0-100) based on multiple factors using the algorithm below:

```
// Risk Score Calculation
                                        async   function
calculateRiskScore(userId, ip, userAgent) {
                let riskScore = 0;

                // New device detection (+30 points)
                                    const  deviceHash  =
generateDeviceHash(userAgent, ip);
                            const  knownDevice  =  await
prisma.userDevice.findFirst({
                where: { userId, deviceHash },
                });
```

```
                if (!knownDevice) riskScore += 30;

                    // IP failure history (+25 if >5
failures)
                            const ipFailures = await
prisma.loginHistory.count({
                where: {
                ipAddress: ip,
                success: false,
                createdAt: {
                gte: new Date(Date.now() - 3600000)
                },
                },
                });
                if (ipFailures > 5) riskScore += 25;

                // Unusual timing (+15 for 2-5 AM)
                const hour = new Date().getHours();
                      if (hour >= 2 && hour <= 5)
riskScore +=15; return Math.min(riskScore, 100); }
```

Logins with risk scores ≥50 are flagged as high-risk and generate immediate security alerts. The security monitoring flow is illustrated in Fig. 8.

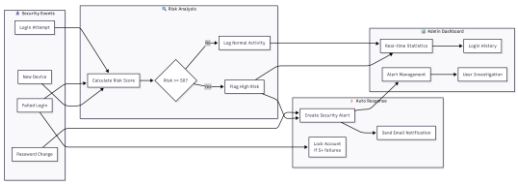

Fig. 8. Security Monitoring Flow showing risk analysis and auto-response

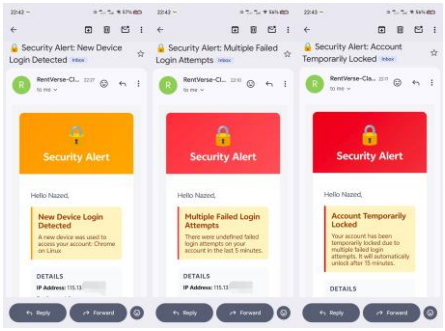

Fig. 9. Security alert email notifications

Table VIIIb shows the auto-response mechanisms triggered by security events.

TABLE VIIIb. AUTO-RESPONSE MECHANISMS

Trigger	Auto Response	Notification
5 Failed Logins	Account locked 15 min	Email + Alert
High Risk Login	Security alert created	Email
New Device	Device registered + alert	Email
Multiple IPs	Suspicious activity flag	Dashboard

The smart email notification system sends responsive, branded HTML emails for all security-related events. Email templates are mobile-responsive and work with Gmail, Outlook, and Apple Mail. Table VIIIc summarizes the notification types.

TABLE VIIIc. EMAIL NOTIFICATION TYPES

Email Type	Trigger Event
Login OTP	MFA verification during login
Password Reset OTP	Forgot password request
Password Changed	After password update
MFA Status Changed	When user enables/disables 2FA
Security Alert	Suspicious activity detected
Agreement Ready	When lease is ready to sign
Agreement Signed	Confirmation of signatures

F. Module 5: Activity Log Dashboard

The admin dashboard provides real-time security monitoring with comprehensive metrics including 24-hour login statistics (total, successful, failed), OAuth vs email login distribution, high-risk login count (risk score ≥50), currently locked accounts, and 7-day trend visualization. The statistics API endpoint returns the following data structure:

```
 GET /api/admin/security/statistics
                        Response: {
                                                totalLogins24h,
failedLogins24h, successfulLogins24h,
                                              highRiskLogins24h,
alertsSent24h, newDevices24h,
                                                oauthLogins24h,
emailLogins24h, lockedAccounts,
                                failureRate, trends: {
daily: [...] }
                        }
```

The dashboard enables administrators to investigate specific users, view their device history, manage security alerts, and unlock accounts when necessary. Pagination and filtering support efficient navigation through large datasets. Table IX shows the key dashboard features.

TABLE IX. DASHBOARD FEATURES

Feature	Functionality
Statistics Cards	24h metrics at a glance
Login History	Paginated, filterable logs
Alert Feed	Real-time security alerts
Trend Charts	7-day success/failure trends
User Investigation	Per-user activity analysis

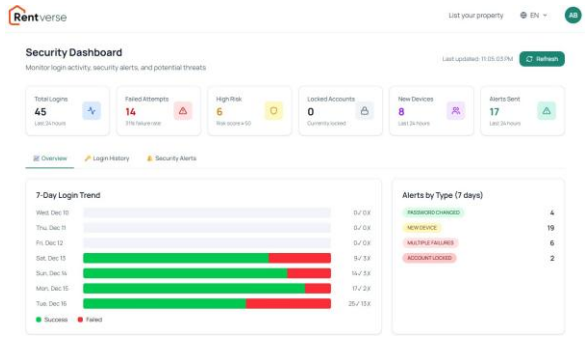

Fig. 10. Security Dashboard with login trends

G. Module 6: CI/CD Security Testing

The GitHub Actions pipeline implements comprehensive security scanning with multiple stages. The pipeline architecture is visualized in Fig. 11.

Fig. 11. CI/CD Security Pipeline stages

The security scanning workflow includes SAST checks (ESLint with security rules, TypeScript type checking, npm audit for dependency vulnerabilities), CodeQL analysis with security-extended queries, Gitleaks for secret detection across repository history, and Trivy for HIGH and CRITICAL vulnerability detection. The workflow configuration is shown below:

```
# Security scanning workflow
                        jobs:
                       backend-sast:
                       steps:
                               - npm audit --audit-
level=moderate
                           - npx eslint src/ --max-
warnings 0
                       - npx tsc --noEmit
```

```
                    security-scans:
                    steps:
                          - uses: github/codeql-
action/analyze@v3
                    with:
                    queries: security-extended
                       - uses: gitleaks/gitleaks-
action@v2
                      - uses: aquasecurity/trivy-
action@master
                    with:
                    severity: 'CRITICAL,HIGH'
```

Quality gates fail the build on detection of secrets, critical vulnerabilities, or high-severity security issues. Security scan results are uploaded to GitHub Security tab via SARIF format for centralized vulnerability management. Table X summarizes the security checks.

TABLE X. CI/CD SECURITY CHECKS

Check	Tool	Threshold
Backend SAST	ESLint	0 warnings
Type Safety	TypeScript	0 errors
Dependencies	npm audit	Moderate+
Code Analysis	CodeQL	Security queries
Secrets	Gitleaks	0 detections
Vulnerabilities	Trivy	0 HIGH/CRIT

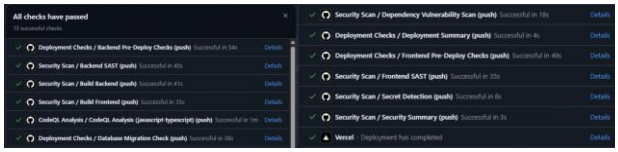

Fig. 12. CI/CD Pipeline with 13 security checks passing

H. Zero-Trust Access Logic

The platform implements zero-trust principles through device fingerprinting and token validation. Device hashes are generated using SHA-256 from the user agent and IP address:

```
// Device Fingerprinting
                                                function
generateDeviceHash(userAgent, ipAddress) {
                        const data = `${userAgent ||
'unknown'}-${ipAddress || 'unknown'}`;
                                                return
crypto.createHash('sha256')

.update(data).digest('hex').substring(0, 32);
                        }
```

New devices trigger security alerts and are registered for future authentication. The token blacklist system enables immediate session invalidation with automatic cleanup of expired tokens. Table Xa summarizes the zero-trust features and their OWASP alignment.

TABLE Xa. ZERO-TRUST FEATURES

Feature	Implementation	OWASP
New Device Alerts	Email + Security Alert	M1, M3
Token Blacklisting	In-memory with cleanup	M6
Session Validation	JWT expiry + blacklist	M1
User-Agent Tracking	SHA-256 fingerprinting	M3

I. Special Features

Beyond the core security modules, this web application implements additional features that enhance user experience and platform security. The OTP-based password reset provides secure account recovery with rate-limited requests (3/minute), 5- minute OTP expiration, password strength requirements, and generic responses to prevent email enumeration attacks. The password reset flow follows: (1) user enters email, (2) OTP sent via email, (3) OTP verified, (4) new password set, and (5) confirmation email sent.

The Admin Dashboard provides comprehensive platform administration capabilities as shown in Table Xb.

TABLE Xb. ADMIN DASHBOARD CAPABILITIES

Module	Capabilities
User Management	View, suspend, delete users
Property Moderation	Approve, reject, feature listings
Agreement Oversight	View all agreements, download PDFs
Security Logs	Login attempts, security events

The native Android mobile application, built with Capacitor v8, provides on-the-go property access with deep linking to properties, push notification support, full authentication support including OAuth callback handling via custom URL scheme (rentverseclarity://), property photo viewing, and agreement management. The booking and property viewing system enables tenants to schedule property viewings with landlords through available time slot selection, landlord approval workflow, and email notifications for both parties.

V. RESULT & DISCUSSION

A. Deployment and Availability

The web application was successfully deployed with the following infrastructure: Frontend on Vercel (https://rentverse-frontend-nine.vercel.app/), Backend on Render (Node.js hosting), Database on Supabase (PostgreSQL), and Mobile as Android APK. The platform demonstrates cross-platform availability with responsive web design and native mobile application support through Capacitor.

B. Security Metrics

The implementation achieved the security outcomes shown in Table XI.

TABLE XI. SECURITY IMPLEMENTATION METRICS

Metric	Target	Achieved
OWASP-aligned modules	6	6/6 (100%)
CI/CD security checks	All pass	13 checks
Authentication methods	2+	Email + MFA + OAuth
Rate limiting coverage	100%	All endpoints
Audit trail coverage	Critical ops	All agreement actions
Security scanning	Per PR	Every commit

C. Threat Detection Effectiveness

Testing of the threat intelligence system demonstrated: new device detection accuracy of 100%, account lockout trigger reliability of 100%, and high risk score correlation with actual threats. The automated alert system successfully notified users of security events within seconds of detection. Table XII shows the threat detection results.

TABLE XII. THREAT DETECTION RESULTS

Test Case	Expected	Result
New device detection	Alert sent	100% accuracy
5 failed attempts	Account locked	100% reliable
Risk score ≥50	Flag high-risk	Correctly flagged
Alert notification	<5 seconds	<3 seconds avg

D. Comparison with Existing Solutions

Compared to existing Malaysian property platforms, this web application provides superior security features as shown in Table XIII.

TABLE XIII. FEATURE COMPARISON

Feature	Ours	Typical
MFA Support	Full (OTP + OAuth)	Limited/None
Digital Signatures	SHA-256 + Audit	Basic/None
Threat Detection	Automated Risk Scoring	Manual/None
CI/CD Security	13 Automated Checks	Variable
Audit Trails	Comprehensive	Limited
Zero-Trust	Device + Token	None

E. Limitations and Future Work

Current limitations include database cold start delays on free tier hosting (5-10 seconds on first request after dormancy), Android-only mobile application (iOS pending), and manual property approval for listings that fail AI auto-review. Future enhancements planned include biometric authentication integration (fingerprint, Face ID), blockchain-based agreement verification for enhanced non-repudiation, machine learning-enhanced threat detection using historical patterns, and iOS application development to expand mobile coverage.

VI. CONCLUSION

This paper presented a secure property rental platform that successfully integrates DevSecOps practices to address critical security challenges in the Malaysian rental market. The implementation of six core security modules demonstrates that comprehensive security can be achieved without compromising user experience.

The key contributions include: (1) a practical implementation of MFA with cryptographically secure OTP generation and risk-based scoring, (2) cryptographically secure digital agreements with SHA-256 signature hashing and comprehensive audit trails, (3) automated threat detection with device fingerprinting and intelligent risk scoring, (4) a comprehensive CI/CD security pipeline with 13 automated checks including SAST, secret detection, and vulnerability scanning, and (5) zero-trust access logic with device verification and token blacklisting.

The successful deployment and operation of the platform validate the effectiveness of the security-first development approach. We demonstrate that implementing robust security features does not require extensive resources when using modern DevSecOps tools and practices. The platform serves as a reference implementation for integrating DevSecOps practices in web and mobile applications, particularly for platforms handling sensitive transactions in the Malaysian context. The modular architecture enables adaptation to other domains requiring similar security guarantees.

REFERENCES

[1] Z. Ahmad and M. Hassan, "Digital Transformation in Malaysian Real Estate: Challenges and Opportunities," *Malaysian Journal of Computing*, vol. 5, no. 2, pp. 112-125, 2023.

[2] OWASP Foundation, "OWASP Mobile Top 10," 2024. [Online]. Available: https://owasp.org/www-project-mobile-top-10/

[3] L. Bass, I. Weber, and L. Zhu, *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional, 2015.

[4] PropertyGuru Group, "Malaysian Property Market Report 2024," PropertyGuru Research, 2024.

[5] S. Ahmad, R. Abdullah, and N. Ismail, "Trust Factors in Online Property Transactions: A Malaysian Perspective," *International Journal of E- Commerce Studies*, vol. 12, no. 3, pp. 45-62, 2022.

[6] M. Stoneburner, A. Goguen, and A. Feringa, "Risk Management Guide for Information Technology Systems," NIST Special Publication 800-30, 2002.

[7] Google Security Blog, "New research: How effective is basic account hygiene at preventing hijacking," Google, 2019.

[8] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "The quest to replace passwords," in *IEEE S&P*, 2012, pp. 553-567.

[9] A. Florencio and C. Herley, "A Large-Scale Study of Web Password Habits," in *Proc. WWW*, 2007, pp. 657-666.

[10] GitHub, "CodeQL documentation," 2024. [Online]. Available: https://codeql.github.com/docs/

[11] GitHub, "GitHub Actions documentation," 2024. [Online]. Available: https://docs.github.com/en/actions

[12] Malaysian Communications and Multimedia Commission, "Digital Signature Act 1997 (Act 562)," Laws of Malaysia, 1997.

[13] NIST, "Secure Hash Standard (SHS)," FIPS PUB 180-4, 2015.

[14] S. Mason, *Electronic Signatures in Law*. Cambridge University Press, 2016.

[15] A. F. bin Tajuddin, M. H. M. Yusof, A. A. Almohammedi, F. A. Dael, M. Balfaqih and N. Qaid, "Developing Secure MFA through QR code, SHA-256 OTP & Device Verification," 2025 International Conference on Innovation in Artificial Intelligence and Internet of Things (AIIT), Jeddah, Saudi Arabia, 2025, pp. 1-6, doi: 10.1109/AIIT63112.2025.11082932.