International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Oracle APEX Security: Best Practices for Robust Protection

Ashraf Syed

maverick.ashraf@gmail.com

Abstract

Oracle Application Express (APEX) has emerged as a powerful rapid application development tool, enabling organizations to deploy robust web applications quickly [1]. However, as applications grow in scale and complexity, security becomes paramount. This paper comprehensively reviews Oracle APEX security, discussing best practices, practical tips, and advanced techniques for securing APEX applications. It covers essential aspects such as configuring secure authentication mechanisms, implementing robust authorization controls, managing session state effectively, preventing common vulnerabilities like SQL injection and cross-site scripting (XSS), and securing file uploads. Additionally, the article emphasizes the importance of regular security assessments, staying updated with security trends, and leveraging Oracle APEX's built-in security features. By adhering to these practices, developers, and administrators can significantly bolster the security posture of their Oracle APEX applications, ensuring resilience against evolving threats and compliance with organizational security standards.

Keywords: Oracle APEX, web application security, best practices, authentication, authorization, session management, SQL injection, cross-site scripting, Error Handling, Data Encryption, Monitoring

I. INTRODUCTION

The security of a web application is not merely a technical concern, it is a critical business imperative. A breach can lead to significant financial losses, irreparable damage to an organization's reputation, and the erosion of customer trust. In today's digital landscape, robust security measures are essential to protect sensitive data, ensure operational continuity, and comply with increasingly stringent regulatory requirements. Organizations that fail to invest in proper security risk not only experience operational disruption but also long-term harm to their competitive standing and brand equity.

Beyond these immediate risks, the rapid pace of digital transformation has significantly increased the reliance on web applications for core business functions. Oracle APEX, renowned for its rapid development capabilities and seamless integration with Oracle databases, offers organizations the ability to deploy data-driven solutions quickly [2]. However, this accelerated pace can sometimes lead to oversights in security considerations, leaving applications vulnerable to exploitation. It is crucial for developers and IT security professionals to balance agility with a rigorous security framework, ensuring that speed does not come at the expense of safety. With the growing adoption of Oracle APEX in enterprise environments,

it is essential to adopt a proactive security posture to mitigate potential threats and vulnerabilities. Security is not something that can be rushed on to the application just before it gets promoted to production. It should be considered as early in the design process as possible [3]. To ensure that this happens, it's a good idea to have a security plan when building applications. The plan should be broad enough to meet the requirements yet brief enough to remain practical [3].

This paper aims to equip developers with comprehensive best practices, tips, and tricks tailored specifically for fortifying Oracle APEX applications. By synthesizing best practices from diverse sources, we provide actionable recommendations that span the entire application lifecycle, from design and development to deployment and ongoing maintenance. This article encompasses the critical aspects of authentication, input validation, session management, error handling, and data encryption. The recommendations are based on established research, industry guidelines, and official Oracle documentation, ensuring that the advice is both theoretically sound and practically applicable. The goal is to empower organizations to build and maintain secure web applications that not only protect sensitive data but also support sustainable business growth.

Furthermore, the contributions presented here highlight the importance of a proactive and continuous security strategy. By integrating security best practices into every stage of the development process, organizations can preempt potential vulnerabilities and rapidly respond to emerging threats [4]. In an increasingly sophisticated era of cyber-attacks, adopting a comprehensive security approach is essential for maintaining business resilience and ensuring long-term success [5].

II. BACKGROUND AND RELATED WORK

A. Web Application Security

Web application security has long been an active research area, given the critical role that web applications play in modern business and daily life. Pioneering works such as those by Viega and McGraw [6] and Howard and LeBlanc [7] have laid the foundation for secure software development practices. Researchers have continuously analyzed vulnerabilities such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and improper error handling, recommending robust mitigation strategies. The OWASP Top 10 [8] remains a seminal resource that outlines the most critical security risks and provides detailed recommendations for developers and security professionals.

The field of web application security has been rigorously studied over the years, focusing on mitigating vulnerabilities inherent in online systems. Foundational research has systematically addressed issues such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and deficiencies in session management. Early works emphasized the significance of secure coding practices and robust architectural designs to counter these threats [9]. Additionally, established industry frameworks like the OWASP Top 10 continue to serve as essential references for identifying and mitigating the most critical web application vulnerabilities.

B. Oracle APEX Overview in the Context of Security

Oracle APEX is an enterprise-grade rapid application development (RAD) tool that integrates tightly with Oracle databases [2]. Its ease of use and powerful features have made it popular in organizations that require quick deployment of data-driven applications. However, the features that make APEX attractive can also introduce security challenges. For example, while efficient, the declarative development approach

may obscure security pitfalls for inexperienced developers. Recent research and industry experts has highlighted the need for tailored security frameworks that address the unique characteristics of APEX [10]. Oracle Application Express (APEX) is widely recognized for its ability to expedite the development of database-centric web applications. Despite its ease of use, APEX's unique abstractions and declarative nature can obscure potential security risks. Recent investigations into Oracle APEX have highlighted challenges specific to the platform, such as ensuring effective session state management, mitigating risks from dynamic SQL, and maintaining rigorous input sanitization practices. The technical documentation provided by Oracle and targeted research studies underscores the necessity of implementing tailored security measures withinthe APEX ecosystem to address these platform-specific vulnerabilities [11].

C. Research Trends and Industry Guidelines

In recent years, several studies have investigated the security of database-centric web applications, particularly those built on platforms like Oracle APEX. Industry guidelines, such as those provided by Oracle Corporation [11], further complement academic insights by offering practical recommendations tailored to the APEX environment.

Contemporary research emphasizes the integration of security into every phase of the application development lifecycle. Recent studies in reputable journals and technical surveys have demonstrated that adopting multi-layered security strategies—combining secure design principles, proactive vulnerability testing, and continuous monitoring—is essential for modern web applications [12]. In parallel, evolving industry guidelines now advocate for advanced approaches, such as AI-driven security assessments and adaptive authentication mechanisms, to respond to the increasingly sophisticated threat landscape. These collective insights drive home the point that robust security is achieved not merely by applying isolated measures but through a cohesive strategy that evolves alongside emerging cyber threats.

The evolving landscape of web application security has prompted significant research attention toward the unique challenges presented by Oracle APEX. Recent studies have focused on identifying inherent vulnerabilities within the APEX framework, such as issues related to session state management, dynamic error handling, and third-party plugins and integrations [13]. Researchers are adapting traditional security methodologies to suit the rapid application development model that Oracle APEX embodies, emphasizing early vulnerability detection and remediation [13].

On the industry side, Oracle has established comprehensive guidelines that reflect the latest threat intelligence and mitigation strategies. The Oracle APEX Security Guide and Oracle APEX Developer Guide provide detailed recommendations for securing applications, covering aspects such as session state protection, secure coding practices for PL/SQL, and the use of built-in APEX features like dynamic actions and declarative authentication [11]. These documents underscore the importance of configuring not just the application code but also the supporting infrastructure—ranging from database hardening to network security measures—in a cohesive security strategy.

Furthermore, the convergence of academic research and industry standards has spurred the development of innovative security frameworks. These frameworks emphasize continuous monitoring and real-time threat detection, enabling APEX applications to adapt to emerging security challenges dynamically. Integrated solutions now leverage machine learning and adaptive security measures to predict potential vulnerabilities and preemptively counter attacks. This proactive approach is critical in an environment where cyber threats are increasingly sophisticated and persistent.

In summary, both the research community and industry practitioners are driving the evolution of Oracle APEX security. A more robust and resilient security ecosystem is emerging by synthesizing cutting-edge research with practical, regularly updated industry guidelines. This collaborative effort is essential for ensuring that Oracle APEX applications remain secure amid the rapidly changing landscape of cyber threats, thereby protecting sensitive data and maintaining business continuity.

III. ORACLE APEX OVERVIEW

A. Architectural Framework

Oracle APEX is a browser-based development environment that leverages Oracle databases for data management and storage [1]. Its architecture comprises a web listener, an application server, and the Oracle Database. The APEX engine processes requests and generates HTML and JavaScript content dynamically [14]. This multi-tiered architecture necessitates a comprehensive security strategy addressing each layer's vulnerabilities. Critical components include:

I. APEXENGINE:

This engine generates web pages, processes PL/SQL code, and renders dynamic content [14]. It abstracts much of the complexity of direct database interactions, enabling developers to focus on business logic. Manages the execution of PL/SQL code and serves as the interface between the user and the database.

II. ORACLEDATABASE:

The Oracle Database serves as the foundation, storing application data, business logic, and metadata [14]. Its robust security features and high availability options are critical in ensuring application resilience.

III. WEBLISTENER:

Typically implemented using Oracle REST Data Services (ORDS) or Oracle HTTP Server, the web listener facilitates communication between client requests and the APEX engine. It manages HTTP requests, ensuring efficient data transmission and adherence to web protocols [14].

B. Development Environment and Lifecycle

The rapid application development cycle offered by Oracle APEX accelerates the deployment of web applications [2]. However, the development speed may inadvertently lead to shortcuts in security considerations. Key stages in the development lifecycle include:

- Design: Emphasis is placed on rapid prototyping and iterative development, where security requirements should be defined from the outset [3].
- Implementation:Developers build pages, reports, and PL/SQL processes, often reusing components to expedite development. This phase demands strict adherence to secure coding practices [3].
- Testing: Security testing, including vulnerability scanning and penetration testing, should be integrated into the overall testing strategy [4].
- Deployment: Secure configurations, such as encryption and access controls, must be enforced during deployment.
- Maintenance: Continuous monitoring and regular upgrades of the APEX version are crucial for addressing emerging security threats.

IV. SECURITY THREAT LANDSCAPE

Just like any other web application, poorly built Oracle APEX applications are susceptible to multiple security vulnerabilities. Key vulnerabilities include:

A. URL Tampering

URL tampering is a critical security threat in Oracle APEX applications, as it allows unauthorized users to manipulate URL parameters to gain unintended access to application pages and data. The APEX URL follows a standardized "f?p" syntax, where parameters such as Application ID, Page ID, Session ID, and Item Values are passed in a colon-delimited format [3]. Due to this predictable structure, attackers can experiment with modifying URL values to bypass authentication controls, access restricted pages, or manipulate application behavior. For example, changing the Page ID in a URL could allow an attacker to navigate to an unauthorized page, while altering Item Values could lead to unauthorized data modifications.

B. SQL Injection

SQL Injection remains one of the most critical threats to Oracle APEX applications. It occurs when user input is improperly sanitized and directly embedded into SQL queries, allowing attackers to inject malicious commands that can violate the syntactic nature of the SQL/XML queries, thereby compromising the database and gaining unauthorized access to data [15]. Exploitation techniques may involve:

- Retrieving unauthorized data using UNION-based attacks.
- Modifying or deleting database records through DML injection.
- Bypassing authentication by injecting always-true conditions in login forms [11].

C. Cross-Site Scripting (XSS)

XSS attacks arise when an application improperly handles user input, allowing attackers to inject malicious scripts that execute in the browser of unsuspecting users. It can occur when the application parses/executes user inputs without proper sanitization [15]. These attacks can be classified into:

- Stored XSS: Malicious scripts are stored in the database and executed when users view infected pages.
- Reflected XSS: Malicious scripts are included in a URL or input field and executed when victims interact with the link.
- DOM-based XSS: The attack manipulates the Document Object Model (DOM) to execute malicious code dynamically.

XSS attacks can lead to session hijacking, credential theft, and unauthorized actions performed on behalf of victims.

D. Cross-Site Request Forgery (CSRF)

CSRF occurs when an attacker tricks an authenticated user into executing unwanted actions on an application without their consent. Attackers achieve this by embedding malicious requests within images, scripts, or iframes, causing:

- Unauthorized transactions in financial applications.
- Data modification without user awareness.
- Session manipulation leads to privilege escalation.

Preventing CSRF requires implementing anti-CSRF tokens, enforcing same-origin policies, and restricting cross-origin requests.

E. Broken Authentication and Session Management

Poor authentication and session management practices expose Oracle APEX applications to session hijacking and credential-based attacks. Common issues include:

- Insecure password storage (e.g., storing passwords in plaintext) [16].
- Weak session handling mechanisms, such as predictable session IDs.
- Lack of multi-factor authentication (MFA) increases exposure to brute-force attacks [16].

Mitigation strategies involve enforcing strong password policies, utilizing secure session management techniques, and integrating MFA for critical operations [16].

F. Insecure Direct Object References (IDOR)

IDOR vulnerabilities occur when an application exposes internal objects, such as user IDs or file references, without proper access control [11]. Attackers exploit IDOR to:

- Gain unauthorized access to confidential records.
- Modify data belonging to other users.
- Escalate privileges beyond intended roles.

IDOR prevention involves implementing role-based access control (RBAC), using indirect references, and enforcing server-side authorization checks.

G. Attack Surface Analysis

The attack surface of an Oracle APEX application comprises all potential entry points that an attacker could exploit. Reducing the attack surface minimizes security risks and improves resilience against cyber threats [3].

I. User Input Forms

Formsserve as the primary interface for data input in APEX applications. Attackers can exploit poorly validated forms to launch SQL injection, XSS, and IDORattacks**.**

II. APEX APIs and Web Services

Exposed APIs pose a significant security risk if not adequately protected, as they can serve as entry points for various attacks. Common threats include broken authentication, which allows unauthorized users to gain access; unrestricted access to sensitive endpoints, potentially leading to data leaks; and inadequate rate limiting, which makes APIs vulnerable to brute-force attacks.

III. Session Tokens and Cookies

Poor session management significantly heightens the risk of various security threats, including session fixation attacks, where attackers manipulate and reuse existing session tokens to gain unauthorized access. Additionally, session hijacking poses a serious concern, as attackers can steal session cookies through cross-site scripting (XSS) or network sniffing, compromising user accounts. Another critical issue is improper session expiration, which allows inactive sessions to remain active indefinitely, increasing the likelihood of exploitation by malicious actors. Implementing robust session management practices is essential to mitigate these risks and enhance the overall security of web applications.

IV. Error Messages

Verbose error messages can inadvertently expose sensitive application details, including the database structure, such as table and column names, which can aid attackers in crafting targeted SQL injection attacks [17]. Additionally, revealing the underlying technology stack, such as the Oracle database version, provides valuable information that could be exploited using known vulnerabilities. Furthermore, exposing internal file paths and configurations can assist attackers in identifying system architecture weaknesses, increasing the risk of unauthorized access and exploitation.

Developers can significantly reduce the likelihood of successful cyberattacks by minimizing the attack surface through secure configurations and controlled exposure.

V. SECURITY BEST PRACTICES

Security in Oracle APEX applications requires a proactive approach, combining robust authentication, authorization, input validation, secure session management, and continuous monitoring. With little extra effort, many security threats can be mitigated with a good understanding of vulnerabilities and proper remediation techniques. However, security measures are often overlooked due to time constraints and inadequate knowledge of evolving threats. The following sections detail the best practices for securing Oracle APEX applications.

A. Authentication and Authorization

Authentication and authorization mechanisms play a fundamental role in securing Oracle APEX applications. Strong authentication ensures that only legitimate users can access the system, while robust authorization policies dictate the extent of access granted to these users.

I. Centralized Authentication Services

Centralized authentication provides a single, secure point of user validation. APEX offers various built-in authentication schemes such as Database Authentication, LDAP (Lightweight Directory Access Protocol), Social Sign-In (OAuth, OpenID), and Oracle Single Sign-On (SSO) [3]. Organizations could integrate APEX applications with enterprise authentication solutions such as Oracle Identity Cloud Service (IDCS), Microsoft Azure Active Directory, or Active Directory Federation Services (ADFS) to enforce uniform security policies across applications.

Centralized authentication eliminates the need to manually manage individual APEX accounts, reducing administrative overhead and improving security. Moreover, it enhances auditing capabilities by ensuring that authentication logs are maintained at an enterprise level, making it easier to track login attempts and suspicious activities.

FIGURE 1. APEX Authentication flow [18].

II. Multi-Factor Authentication (MFA)

MFA adds an extra layer of security by requiring multiple verification factors before granting access to users [19]. This can include:

- Something the user knows (PIN, Secure Word)
- Something the user has (smartphone, security token, one-time password)
- Something the user is (biometric authentication such as fingerprint or facial recognition) [16]

Oracle APEX supports MFA by integrating with external identity providers such as Google Authenticator, Microsoft Authenticator, or Duo Security. Implementing MFA helps mitigate credential theft and unauthorized access [16].

III. Login Attempt Limitation

Brute force attacks involve repeated login attempts using different password combinations [20]. To mitigate this risk, APEX allows login throttling, which can be configured in the Security Attributes section of Shared Components. Enforcing a policy that locks accounts after a predefined number of failed login attempts adds an additional security barrier against unauthorized access.

IV. Role-Based Access Control (RBAC)

RBAC ensures that users can only access features and data relevant to their role. Oracle APEX facilitates RBAC through the following:

- Defining roles and privileges in the database.
- Assigning authorization schemes at different levels within the application, including pages, regions, items, and components.
- Implementing dynamic role-based menus ensures users only see the functionalities available to their role.

To restrict access, select the Authorization scheme attribute at various application levels as and when required, such as item, region, and page.

RBAC prevents unauthorized data exposure and enforces the Principle of Least Privilege (PoLP) by restricting access to necessary functionalities only [19].

FIGURE 2. APEX Authorization flow [18].

V. Session State Protection (SSP)

Session State Protection (SSP) is a built-in APEX feature that prevents URL tampering using cryptographic checksums for parameterized URLs [21]. Enabling SSP ensures that attackers cannot manipulate session state variables in URLs to gain unauthorized access to application data.

To enable SSP:

- "Navigate to Shared Components → Security Attributes"[11]
- Set Session State Protection to Enabled
- Configure session-based items with Restricted - Must not be set from the browser.

VI. Session Timeout Policies

To minimize the risk of session hijacking, enforce strict session timeout policies, including:

- Idle session timeout: Automatically logging out inactive users.
- Absolute session timeout: Setting a maximum session duration irrespective of activity.
- Logout enforcement: Invalidating session cookies and clearing authentication tokens upon logout.

B. Input Validation and SQL Injection Prevention

Unchecked user input is a primary attack vector for SQL injection and other exploits. Ensuring proper input validation helps prevent malicious data from affecting application integrity.

I. Use of Bind Variables

Bind variables prevent SQL injection by ensuring that user input is treated as data rather than executable code [15]. Instead of dynamically concatenating SQL statements, always use bind variables, using bind variables not only enhances security but also improves query performance by allowing SQL statements to be cached efficiently.

II. Sanitization and Validation

Both client-side and server-side Input validation should be enforced to prevent tampering. Best practices include:

- Enforcing whitelist validation, accepting only expected characters, and rejecting dangerous ones.
- Restricting input lengths to prevent buffer overflow attacks.
- Using the apex_escape package to sanitize inputs before rendering them in HTML or JavaScript [19].

III. Stored Procedures and PL/SQL Packages

Encapsulating SQL queries inside stored procedures and PL/SQL packages reduces direct SQL execution, limiting the attack surface. This approach allows developers to implement access control at the database level.

IV. Regular Expression Validation

Regular expressions (RegEx) are useful for ensuring that input follows a predefined structure. For instance:

- Email validation:^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}$
- Phone number validation: ^[0-9]{10}$

Applying RegEx prevents malformed input from being processed, thereby improving data integrity and security.

C. Cross-Site Scripting (XSS) Prevention

XSS vulnerabilities arise when attackers inject malicious scripts into web pages viewed by other users [15]. Oracle APEX applications should implement the following measures to prevent XSS attacks:

I. Output Encoding

Always encode user-generated content to prevent XSS before displaying it in the browser. Use APEX functions such as:

- apex_escape.html(p_string) for encoding HTML output.
- apex_escape.javascript(p_string)for encoding JavaScript output.
- apex_escape.css(p_string)for encoding CSS values.

II. Content Security Policy (CSP)

A strict CSP prevents unauthorized scripts from executing. Define CSP rules in the web server configuration or within the application's HTTP response headers:

Content-Security-Policy: default-src 'self'; script-src 'self';

III. Sanitize HTML Content

If the application permits rich text input, sanitize the content to strip harmful tags and attributes using apex_javascript.escape or external libraries such as DOMPurify.

IV. Avoid Inline JavaScript

Refrain from embedding user-generated content directly into <script> tagsinstead, load scripts from secure external files.

D. Session Management and CSRF Protection

Effective session management is crucial for maintaining user authentication integrity, preventing session hijacking, and ensuring that user data remains secure. Furthermore, Cross-Site Request Forgery (CSRF) attacks exploit the trust of authenticated sessions, making it imperative to incorporate CSRF protection mechanisms [11].

I. Secure Session Tokens

To implement secure session tokens in Oracle APEX:

- Use built-in session state management for authentication.
- Store session identifiers in secure, HTTP-only cookies rather than local storage.
- Regenerate session tokens after authentication to mitigate session fixation attacks.

II. HTTP-Only and Secure Cookies

Cookies play a vital role in session management but can be a security risk if not properly configured. Use the following best practices:

- Set the HTTP-Only Flag: This prevents JavaScript from accessing cookies, mitigating XSS-related session hijacking.
- Set the Secure Flag: This ensures that cookies are only transmitted over HTTPS connections.

- Use the SameSite Attribute: The SameSite attribute prevents cookies from being sent with cross-site requests, reducing CSRF risks.

Example of a secure cookie configuration:

Set-Cookie: session_id=xyz123; HttpOnly; Secure; SameSite=Strict

Oracle APEX allows developers to configure these attributes under Shared Components → Security Attributes.

III. Cross-Site Request Forgery (CSRF) Protection

CSRF attacks trick authenticated users into unknowingly executing unauthorized actions on a trusted website. Attackers achieve this by embedding malicious requests in web pages or emails that exploit the user's active session.

A. CSRF Token Implementation

CSRF tokens provide an additional layer of security by requiring a unique, per-session token for each state-changing request (e.g., form submissions and data modifications). These tokens should:

- Be generated per session and stored securely.
- Be included as a hidden field in forms.
- Be validated on the server before processing the request.

Example of a CSRF token implementation in Oracle APEX:

htp.p('<input type= "hidden" name= "csrf_token" value="' || :APP_SESSION || '">');

After submitting the form, the server should verify that the CSRF token matches the expected value before executing any sensitive operations.

B. Using APEX Built-in CSRF Protection

Oracle APEX provides built-in CSRF protection mechanisms:

- Enable Session State Protection: Navigate to Shared Components → Security Attributes and enable Session State Protection.
- Use APEX's Automatic CSRF Protection for Forms: Set the Form Source Type to PL/SQL Process with Automatic Row Processing, which includes CSRF token validation by default.

IV. Session Timeout Policies

Proper session timeout policies reduce the risk of session hijacking by limiting the duration of user sessions [3].

- Idle Timeout: Automatically logs users out after a period of inactivity.
- Absolute Timeout: Terminates the session after a fixed duration, regardless of activity.
- Logout Enforcement: Ensures sessions are invalidated on logout by clearing session cookies and authentication tokens.

In Oracle APEX, session timeouts can be configured under:

- Shared Components → Security Attributes → Session Timeout settings [11].
- Application Session Idle Timeout (configured in APEX authentication schemes).

By enforcing strict session management policies, developers can significantly reduce risks related to session hijacking, CSRF attacks, and unauthorized access.

E. Error Handling and Logging

Error handling and logging are essential for identifying security threats, diagnosing system failures, and preventing sensitive information leakage. Proper error-handling mechanisms ensure that applications do not expose internal system details to attackers.

I. Generic Error Messages

Detailed error messages containing system information, database queries, or stack traces can be exploited by attackers to gain insights into the application's internal workings [17]. Instead, applications should:

- Display generic error messages to users.
- Log detailed error information internally for debugging purposes.

Example of a secure error message implementation (User-Facing)

An unexpected error occurred. Please contact the application support team.

Insecure Error Message (Exposes System Details)

ORA-01756: quoted string not properly terminated.

In Oracle APEX, developers can configure custom error messages using Shared Components → Messages and override default error messages to prevent sensitive data exposure.

II. Centralized Logging

A robust logging system allows administrators to detect security incidents, monitor application usage, and conduct forensic analysis in the event of an attack.

Best practices for centralized logging in Oracle APEX:

- Use Database-Level Logging (DBMS_APPLICATION_INFO, DBMS_ALERT).
- Integrate with Oracle Unified Auditing to track login attempts and suspicious activities.
- Store logs securely in a write-only format to prevent tampering.

Example of inserting a log entry in Oracle APEX:

```
INSERT INTO security_logs (log_time, user_id, action)
VALUES (SYSDATE,:APP_USER, 'Unauthorized Access Attempt');
```

III. Log Sanitization

- To prevent sensitive data exposure, logs should:
- Exclude confidential information such as passwords, authentication tokens, or user session identifiers.
- Mask sensitive user input (e.g., credit card numbers, SSNs).
- Use a dedicated logging table with restricted access.

Example of a sanitized log entry:

[2025-03-25 12:45:00] [User: admin] [Action: Login Failed] [IP: 192.168.1.10]

Instead of logging raw SQL errors, use a sanitized version:

[ERROR] Database operation failed—contact support.

IV. Alerting Mechanisms

Real-time alerts help administrators respond quickly to security threats. Implement:

- Automated notifications using DBMS_ALERT or APEX_MAIL for critical events.
- Threshold-based alerting (e.g., lock an account after multiple failed logins).

- Integration with SIEM (Security Information and Event Management) tools for centralized security monitoring.

Example: Trigger an email alert for failed logins.

```
BEGIN
  APEX_MAIL.SEND(
    p_to => 'admin@company.com',
    p_from => no_reply_apex@company.com',
    p_subj => 'Security Alert: Maximum failed login attempts reached',
    p_body => 'Maximumfailed login attempts reached for user:' || :APP_USER ||'. The user account has
    been locked.'
  );
END;
```

V. Incident Response Planning

Every application should have a Security Incident Response Plan (SIRP) outlining:

- Detection and Classification: Identifying the nature of the security incident.
- Containment: Limiting the impact of the breach.
- Eradication: Removing threats and patching vulnerabilities.
- Recovery: Restoring normal operations.
- Post-Incident Analysis: Learning from the incident to prevent recurrence.

F. Secure Coding Practices

Secure coding is a fundamental requirement for developing robust Oracle APEX applications. Adhering to secure coding standards minimizes vulnerabilities and ensures resilience against evolving cyber threats. The following best practices should be followed:

I. Code Reviews and Security Assessments

- Conduct peer code reviews regularly to identify potential security flaws.
- Use automated static code analysis tools to detect insecure coding patterns.
- Employ security-focused testing, including penetration and fuzz testing, to simulate attack scenarios and evaluate the application's resilience.

II. Use of Built-In Security Functions

- Oracle APEX provides several security functions that should be leveraged instead of writing custom security logic.
- Encode user input using apex_escape.html, apex_escape.jsonbefore rendering it in the UI.
- Utilizeapex_session API to manage session security and prevent unauthorized access.
- Use Oracle Virtual Private Database (VPD) to enforce fine-grained access control at the database level.

III. Avoid Hardcoded Credentials

- Never store database credentials, API keys, or passwords in the application code.
- Use Application Express Credential Store or Oracle Wallet to securely manage credentials.

- Implement environment variables for sensitive configurations instead of embedding them in source code.

IV. SECURE ERROR HANDLING

- Implement a centralized error-handling mechanism to prevent exposing stack traces to end users.
- Instead of detailed database errors, use generic error messages such as "An unexpected error has occurred. Please contact the administrator."
- Log errors securely on the server side without revealing sensitive details in the front end.

V. MINIMIZE THE USE OF DYNAMIC SQL

- Prefer bind variables over string concatenation in SQL queries to prevent SQL injection attacks.
- When dynamic SQL is necessary, use DBMS_ASSERT to validate table and column names before execution.

VI. REGULAR SECURITY TRAINING

- Conduct regular security awareness training for developers to stay updated on emerging threats and mitigation techniques.
- Maintain an internal secure coding standard document for development teams to follow.

G. Data Encryption and Secure Transmission

Encryption is crucial in securing sensitive data, both at rest and in transit. Oracle APEX applications must leverage encryption mechanisms to prevent unauthorized access.

I. TRANSPARENT DATA ENCRYPTION (TDE) FOR DATA AT REST

- Oracle TDE encrypts data stored in database tables, backups, and redo logs.
- Use AES-256 encryption to protect sensitive database columns.
- Ensure that encryption keys are stored securely using Oracle Key Vault or Hardware Security Modules (HSMs).

II. SSL/TLS FOR SECURE DATA TRANSMISSION

- Enable SSL/TLS for all communication between APEX applications and users to prevent man-in-the-middle attacks [21].
- Obtain SSL/TLS certificates from a trusted Certificate Authority (CA) and configure them in Oracle REST Data Services (ORDS).
- Disable weak cryptographic protocols such as TLS 1.0 and TLS 1.1, enforcing TLS 1.2 or higher.

III. ENCRYPTION OF SENSITIVE DATA IN TRANSIT AND STORAGE

- Encrypt user passwords using bcrypt or PBKDF2 instead of storing them in plaintext.
- Use the DBMS_CRYPTO package to encrypt sensitive data, such as credit card numbers, before storing it in the database [11].
- Ensure APEX session variables that contain confidential information are encrypted before being stored in cookies or local storage.

IV. End-to-End Encryption for Highly Sensitive Applications

- Implement end-to-end encryption (E2EE) for applications handling financial transactions or healthcare records.
- Use public key infrastructure (PKI) for strong authentication and encryption of confidential data.

H. Auditing, Monitoring, and Vulnerability Assessment

Security is an ongoing process that requires continuous monitoring, logging, and vulnerability assessments. Oracle APEX applications should implement robust auditing and incident response mechanisms.

I. Real-Time Security Monitoring

- Deploy Security Information and Event Management (SIEM) solutions like Splunk or Oracle Audit Vault to monitor system activity.
- Enable Oracle Database Auditing to track changes in critical database objects and user actions.
- Set up real-time alerts for suspicious activities such as multiple failed login attempts, unauthorized data access, and privilege escalation.

II. Regular Security Audits and Compliance Checks

- Conduct periodic security audits to identify misconfigurations and security gaps in APEX applications.
- Ensure compliance with industry standards like ISO 27001, NIST, and GDPR for handling sensitive data.
- Review APEX security attributes (Session Timeout, Session State Protection, Access Control) as part of security audits.

III. Automated Vulnerability Scanning

- Use automated tools like Oracle Data Safe, OWASP ZAP, or SQLMap to detect vulnerabilities such as SQL Injection, XSS, and CSRF.
- Perform penetration testing using ethical hacking methodologies to uncover security flaws.
- Maintain a vulnerability remediation plan to address identified security issues promptly.

IV. Incident Response Plan

- Develop a well-documented incident response plan (IRP) outlining steps to follow in case of a security breach.
- Conduct simulated security drills to evaluate the effectiveness of the response plan.
- Ensure that logs and audit trails are retained securely to aid forensic investigations in the event of an attack.

Implementing these practices can help Oracle APEX applications achieve high-security standards, ensuring data integrity, confidentiality, and resilience against evolving threats.

I. Continuous Security Testing and Code Analysis

Automated security testing and static code analysis tools are indispensable in identifying vulnerabilities before they reach production:

I. Static Application Security Testing (SAST):

Integrate SAST tools into the development pipeline to analyze PL/SQL code and APEX components for common vulnerabilities. This helps in catching errors early in the development cycle [8].

II. Dynamic Application Security Testing (DAST):

Employ DAST tools to simulate attacks on running applications. Regular penetration testing can reveal exploitable weaknesses that may not be apparent during code reviews.

III. DevSecOps Practices:

Incorporate security into the DevOps pipeline, ensuring that automated security tests are part of every build and deployment. This continuous integration of security reduces the time to detect and remediate vulnerabilities [8].

VI. ADVANCED TIPS AND TRICKS FOR ENHANCED SECURITY

Beyond foundational best practices, advanced techniques can further harden Oracle APEX applications against sophisticated threats. This section presents a series of advanced tips and tricks designed to enhance security in production environments.

A. Instance Hardening and Security Configuration

Oracle APEX operates within an Oracle Database instance, and securing the instance itself is a crucial step toward application security.

I. Instance Settings

Configuring instance settings correctly is vital to reducing the attack surface. APEX supports multiple modes of operation: Development Mode (which allows application builders to create and modify applications) and Runtime Mode (which restricts all development activities and allows only application execution). In a production environment, it is recommended to run the instance in Runtime Mode to prevent unauthorized modifications. This can be enforced by executing the script apxdevrm.sql during installation [3].
Additionally, APEX instance settings should be reviewed periodically using the APEX_INSTANCE_ADMIN package to ensure no unauthorized changes have been made.

II. Security Configuration

APEX provides several configurable security parameters that should be adjusted based on the application's needs:

- Restrict Public File Uploads: Set "Allow Public File Uploads" to "No" unless absolutely required. This prevents unauthorized users from uploading malicious files [3].
- Restrict Access by IP Address: If an application is intended for a limited set of users (e.g., an internal corporate application), IP whitelisting can be configured to allow access only from specific ranges [3].
- Enforce a Strong Password Policy: Set password complexity requirements, including minimum length (at least 12 characters), inclusion of uppercase and lowercase letters, numbers, and special characters.

- Limit Application Session Length: Configure session timeouts aggressively to reduce exposure to session hijacking.

B. Secure Execution of PL/SQL Code and Dynamic Actions

Oracle APEX allows developers to execute PL/SQL processes dynamically, which, if improperly configured, can be exploited for privilege escalation or unauthorized data access.

I. Use Invoker's Rights (AUTHID CURRENT_USER)

By default, PL/SQL procedures execute with definer's rights, meaning they run with the privileges of the schema that owns them. However, this can lead to security risks if a lower-privileged user is able to execute a high-privilege procedure.
To mitigate this, define procedures that should execute with invoker's rights using the following syntax:

```
CREATEOR REPLACE PROCEDURE my_app_proc
AUTHID CURRENT_USERAS
BEGIN
-- Procedure Code here
END my_app_proc;
```

This ensures that the procedure runs with the privileges of the user calling it, preventing privilege escalation.

II. Use Dynamic Actions Securely

Dynamic Actions allow developers to execute JavaScript, PL/SQL, or AJAX requests based on user interactions. However, they can become a security risk if improperly configured.

- Avoid using unrestricted PL/SQL blocks in Dynamic Actions: Always validate user permissions before executing sensitive logic.
- Restrict access to RESTful Services: Ensure that any RESTful Web Services called via Dynamic Actions require authentication and authorization.

III. Restricting APEX Web Services and RESTful APIs

Oracle APEX provides RESTful Web Services that can expose application data. If not properly secured, these services can be exploited by attackers.

- Disable Anonymous Access: Ensure that all RESTful services require authentication.
- Restrict HTTP Methods: Only allow GET, POST, PUT, or DELETE where necessary.
- Use OAuth2 or API Keys: Secure APIs using OAuth2-based authentication or API keys.
- Limit Query Results: Avoid exposing large datasets through API calls by implementing pagination and access control mechanisms.

C. Database Security Enhancements

Securing the Oracle Database that underpins an APEX application is just as important as securing the application itself.

Oracle VPD (Virtual Private Database)enforcesrow-level security**,** ensuring that users only see data they are authorized to access.

Example policy:

```
BEGIN
  DBMS_RLS.ADD_POLICY (
    object_schema   =>'HR',
    object_name     =>'EMPLOYEES',
    policy_name     =>'EMPLOYEE_ROW_LEVEL_SECURITY',
    function_schema =>'HR',
    policy_function =>'employee_filter_function',
    statement_types =>'SELECT'
  );
END;
/
```

This restricts users from querying data outside of their allowed scope.

D. Secure Handling of APEX Collections

APEX Collections provide temporary storage within a user session but can be exploited if not properly handled.

- Avoid using APEX Collections to store sensitive data.
- Validate collection data before processing: Implement proper input sanitization before inserting or retrieving values.
- Restrict access to collection manipulation functions: Ensure only authorized users can execute procedures that modify collections.

E. Preventing Clickjacking Attacks

Clickjacking is a technique where an attacker embeds a legitimate APEX application inside an invisible iframe to trick users into performing unintended actions.

Set the X-Frame-Options header in the web server configuration to prevent the application from being loaded inside an iframe:

```
X-Frame-Options:DENY
```

Alternatively, if embedding is required for certain domains, set:

```
X-Frame-Options:SAMEORIGIN
```

F. Preventing Content Injection and MIME-Type Sniffing

Attackers may attempt **content injection** attacks by tricking browsers into interpreting non-script files (like images) as executable code.

Prevent MIME-type sniffing by adding the following header:
X-Content-Type-Options: nosniff

This ensures that files are interpreted strictly based on their declared content type.

G. Avoid security questions which have limited domain responses.

Security questions are a common authentication mechanism used in applications with custom authentication, especially when retrieving or resetting a forgotten password. However, they often introduce security vulnerabilities, especially when the possible responses fall within a limited domain. Attackers can exploit such weaknesses through social engineering, dictionary attacks, or simple guesswork.

- Predictability: Questions like *"What is your favorite color?"* or *"What is your birth month?"* have a small set of possible answers, making them easy to guess.
- Publicly Available Information: Questions such as *"What city were you born in?"* or *"What was your first pet's name?"* may be discoverable through social media or public records.
- Brute Force Attacks: Attackers can systematically attempt common answers, especially when the system does not enforce strong answer complexity.
- Reusability Risk: Users often reuse the same answers across multiple applications, making them vulnerable if another system is breached.

Therefore, the above vulnerabilities can be easily mitigated by following the steps below:
- Avoid Security Questions Entirely: Instead of security questions, use multi-factor authentication (MFA) or one-time passcodes (OTP) via email or SMS.
- Use Dynamic or User-Defined Questions: If security questions must be used, allow users to create their own questions while enforcing uniqueness.
- Require Complex Answers: Enforce policies that require longer, alphanumeric responses to mitigate brute-force attacks.
- Monitor for Unusual Login Activity: Implement logging and alert mechanisms in APEX to detect multiple failed attempts at answering security questions.

By avoiding limited-domain security questions and adopting stronger authentication mechanisms, Oracle APEX applications can significantly reduce the risk of unauthorized access.

H. Always Add Server-Side Validations to Prevent Malicious Attacks

Client-side validations and controls (such as disabling buttons, hiding items, or using JavaScript validations) enhance user experience but do not provide strong security. Attackers can bypass these controls by manipulating the client-side behavior using browser developer tools or intercepting requests. To ensure robust security, always enforce server-side validations to prevent unauthorized actions.

- Re-Enabling Disabled Buttons or Items: Attackers can use browser developer tools (e.g., Chrome DevTools) to modify the DOM and re-enable buttons or input fields that were disabled by JavaScript.
- Tampering with Hidden Items: Hidden form elements can be modified on the client side before submission, allowing attackers to alter values undetected.

- Bypassing JavaScript Validations: JavaScript-based checks (e.g., format validation, required fields) can be disabled or overridden using the browser console.
- Direct HTTP Requests: Attackers can construct custom HTTP requests (using tools like Postman or Burp Suite) to submit unauthorized or modified data, bypassing front-end restrictions.

Therefore, the above vulnerabilities can be easily mitigated by following the steps below:

- Always Validate on the Server Side: Implement PL/SQL validations in Processing → Validations or within the application logic in BEFORE INSERT/UPDATE triggers to enforce business rules.
- Verify Authorization at the Database Level: Ensure that only authorized users can perform critical operations by using database roles, policies, and row-level security (VPD/RLS).
- Use APEX Page Processing Conditions: Restrict processing logic (e.g., button actions, DML processes) by setting conditions such as current user roles, item values, or request sources.

By enforcing strict server-side validations, Oracle APEX applications can effectively mitigate security threats and prevent malicious users from manipulating client-side controls to exploit the system.

I. Religiously Use APEX Advisor for Security and Performance Audits

Oracle APEX provides APEX Advisor, a built-in tool designed to analyze applications for security risks, performance issues, accessibility concerns, and best practices compliance. Using APEX Advisor regularly is crucial to maintaining a secure, efficient, and well-structured APEX application.

APEX Advisor is a predefined program to analyze the application code for errors and bad practices. It runs pre-written automated tests to check programming errors, security issues, warnings, performance, usability, etc. [10].

- Detect Security Vulnerabilities:

1. Identifies pages or components that lack proper authorization or session state protection.
2. Flags security risks such as unprotected page items**,** exposed sensitive data,orweak authentication settings**.**

- Prevent SQL Injection and XSS Attacks:

1. Warns about unescaped SQL or JavaScript in dynamic content regions.
2. Ensures that queries use bind variables to prevent SQL injection.

- Optimize Performance:

1. Detects slow-performing SQL queries and suggests improvements.
2. Highlights unnecessary computations or unindexed columns that might impact page load speed.

- Ensure Application Integrity:

1. Identifies missing validations and recommends adding server-side validations (essential for preventing client-side bypass attacks).

2. Checks for broken links, unused components, and misconfigured settings.

- Improve Maintainability and Best Practices Compliance:

1. Ensures that naming conventions, dependencies, and coding standards are followed.
2. Helps keep the application clean and organized by identifying obsolete or redundant components.

It is good practice to run APEX Advisor before deploying an update to catch potential security or performance issues. Even minor warnings can expose vulnerabilities. Always review and fix flagged issues. APEX Advisor findings can be combined with APEX Debugging, Application Logging, and Database Audits to get deeper insights into security and performance.

VII. CHALLENGES AND FUTURE RESEARCH DIRECTIONS

A. Balancing Usability and Security

One of the persistent challenges in securing Oracle APEX applications is striking the right balance between robust security measures and user-friendly design. Stringent security controls can sometimes hinder usability, leading to friction in user experience. Future research should focus on adaptive security models that dynamically adjust security requirements based on user behavior and context, thus preserving usability without compromising on protection.

B. Evolving Threat Landscape

Cyber threats are continually evolving, with attackers devising novel techniques to bypassconventional security measures [22]. While the best practices outlined in this paper address many common vulnerabilities, new attack vectors—such as advanced social engineering and sophisticated injection techniques—demand ongoing research. Academic studies and industry collaborations are needed to develop next-generation security frameworks specifically tailored to Oracle APEX and similar RAD platforms.

C. Integration with Cloud Security Frameworks

As more organizations deploy Oracle APEX applications in cloud environments, integrating application security with broader cloud security frameworks becomes increasingly important. Research into the seamless integration of Oracle APEX security mechanisms with cloud-native tools such as automated scaling, container orchestration security, and cloud access security brokers (CASBs) is a promising direction for future work.

D. Automation and AI-Driven Security

The rapid pace of application development necessitates automation in security testing and threat detection. Future research should explore the application of artificial intelligence (AI) and machine learning (ML) to detect anomalous patterns [22] in Oracle APEX applications, automate vulnerability assessments, and predict potential security breaches [23]. Integrating AI-driven insights into the DevSecOps pipeline could significantly enhance proactive security measures [23].

E. Enhancing Developer Training and Awareness

Despite the availability of security guidelines and tools, human factors remain a critical vulnerability. Continuous training, automated code review tools, and enhanced documentation are required to keep developers abreast of the latest security threats and countermeasures. Future studies could investigate the efficacy of various training methodologies and their impact on the secure development of Oracle APEX applications.

VIII. CONCLUSION

Oracle APEX provides a powerful and flexible platform for rapid application development; however, its widespread use necessitates a rigorous approach to security [3]. This paper has presented a detailed discussion of Oracle APEX security, covering a range of topics from authentication and input validation to advanced techniques such as adaptive security models and AI-driven testing. By synthesizing academic research, industry best practices, and official Oracle guidelines, this paper has outlined a multi-layered security strategy that addresses the complex threat landscape associated with web applications.

The best practices discussed such as implementing multi-factor authentication, employing parameterized queries, enforcing robust session management, and integrating continuous monitoring form the backbone of a secure Oracle APEX application. Features such as session state protection, prevention against SQL injection and cross-site scripting, and the enforcement of robust password policies further bolster application security. Additionally, implementing granular authorization controls, including access control lists and row-level security, allows for precise management of user permissions. Regular security audits and adherence to best practices are essential to identify and mitigate potential vulnerabilities, ensuring that APEX applications remain secure and trustworthy [3][22]. Moreover, advanced tips and tricks, including leveraging built-in APEX security features and integrating with enterprise identity solutions, further enhance the resilience of applications against sophisticated attacks.

As cyber threats continue to evolve, the security measures outlined in this paper must be revisited and updated regularly. Future research should focus on addressing emerging vulnerabilities, integrating cloud-native security practices, and leveraging AI and ML to automate and enhance security testing [21]. Ultimately, a proactive and comprehensive security posture is essential for organizations deploying Oracle APEX applications in today's complex and dynamic threat environment.

ACKNOWLEDGMENT

The author would also like to disclose the use of the Grammarly (AI) tool solely for editing and grammar enhancements.

REFERENCES

[1] Oracle Corporation"Oracle APEX", [Online]. Available: https://apex.oracle.com/en/[Accessed: Feb. 15, 2025].

[2] Oracle Corporation, "20x Faster, 100x Less Code," *Oracle APEX*, 2023. [Online]. Available: https://apex.oracle.com/en/platform/low-code/intro/[Accessed:Feb. 13, 2025].

[3] S. Spendolini, *Expert Oracle Application Express Security*. Apress, 2013.

[4] Y. M. Mothanna, W. Elmedany, M. Hammad, R. Ksantini, and M. S. Sharif, "Adopting security practices in software development process: Security testing framework for sustainable smart cities," Computers & Security, vol. 144, p. 103985, Jul. 2024, doi: 10.1016/j.cose.2024.103985.

[5] M. F. Safitra, M. Lubis, and H. Fakhrurroja, "Counterattacking Cyber Threats: A Framework for the Future of Cybersecurity," Sustainability, vol. 15, no. 18, p. 13369, Sep. 2023, doi: 10.3390/su151813369.
[6] J. Viega and G. McGraw, *Building Secure Software: How to Avoid Security Problems the Right Way*, Addison-Wesley, 2001.
[7] M. Howard and D. LeBlanc, *Writing Secure Code*, Microsoft Press, 2003.
[8] OWASP Foundation, "OWASP Top 10 – 2021: The Ten Most Critical Web Application Security Risks," 2021. [Online]. Available: https://owasp.org/Top10. [Accessed: Mar. 7, 2025].
[9] B. Chess and G. McGraw, "Static Analysis for Security," *IEEE Software*, vol. 23, no. 5, pp. 78–83, Sept.-Oct. 2006.
[10] DSP "APEX Security Considerations." Jun. 2024. [Online]. Available: https://content.dsp.co.uk/apex/apex-security-considerations-part-1 [Accessed: Jan. 25, 2025].
[11] Oracle Corporation, "Oracle APEX Security Guide," Oracle Corporation, 2023. [Online]. Available: https://docs.oracle.com/en/database/oracle/apex/. [Accessed: Jan 27, 2025].
[12] R. A. Khan, S. U. Khan, M. Alzahrani, and M. Ilyas, "Security Assurance Model of Software Development for Global Software Development Vendors," IEEE Access, vol. 10, p. 58458, Jan. 2022, doi: 10.1109/access.2022.3178301.
[13] Ramsha, "Best Security Practices for Oracle Apex Applications," DanyTech Cloud. 2024. [Online]. Available: https://www.danytechcloud.com/security-practices-for-oracle-apex-applications/ [Accessed: Feb. 10, 2025].
[14] Oracle Corporation, "Oracle Application Express Installation Guide," 2023. [Online]. Available: https://docs.oracle.com/en/database/oracle/apex/23.2/htmig/index.html. [Accessed: Mar. 29, 2025].
[15] G. Deepa and P. S. Thilagam, "Securing web applications from injection and logic vulnerabilities: Approaches and challenges," Feb. 28, 2016, Elsevier BV. doi: 10.1016/j.infsof.2016.02.005.
[16] M. Papathanasaki, Λ. Μαγλαράς, and N. Ayres, "Modern Authentication Methods: A Comprehensive Survey," Jun. 01, 2022, IntechOpen. doi: 10.5772/acrt.08.
[17] WSTG - Stable," OWASP Foundation. [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/08-Testing_for_Error_Handling/01-Testing_For_Improper_Error_Handling [Accessed: Mar. 15, 2025].
[18] D. Devanaboyina, "Securing Oracle APEX: A Deep Dive into Authentication and Authorization Strategies" Micracle Software Systems, [Online], Available: https://blog.miraclesoft.com/securing-oracle-apex-a-deep-dive-into-authentication-and-authorization-strategies/ [Accessed: Mar. 15, 2025].
[19] Ashraf Syed, "Oracle APEX Development Best Practices: Ensuring Scalability and Maintainability", International Journal of Leading Research Publication, vol. 5, no. 4, pp. 1–12, Apr. 2024, doi: 10.5281/zenodo.15026525.
[20] W. Ali, "Top Security Best Practices in Oracle APEX Applications," MaxAPEX.[Online]. Available: https://www.maxapex.com/blogs/security-best-practices-in-oracle-apex/ [Accessed: Mar. 29, 2025].
[21] S. Jegan, "Oracle APEX Security Features," Doyensys Blog. [Online]. Available: https://doyensys.com/blogs/oracle-apex-security-features/ [Accessed: Mar. 29, 2025].

[22] M. Roshanaei, M. R. Khan, and N. N. Sylvester, "Navigating AI Cybersecurity: Evolving Landscape and Challenges," *Journal of Intelligent Learning Systems and Applications*, vol. 16, no. 03, pp. 155–174, 2024, doi: 10.4236/jilsa.2024.163010.

[23] M. C. Fu, J. Pasuksmit, and C. Tantithamthavorn, "AI for DevSecOps: A Landscape and Future Opportunities," ACM Transactions on Software Engineering and Methodology, Jan. 2025, doi: 10.1145/3712190.