World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(REVIEW ARTICLE)

Check for updates

Securing API Ecosystems in Digital Banking Transformation

Ashish Hota *

Department of Business Administration – IT Management, Western Governors University, United States of America.

World Journal of Advanced Engineering Technology and Sciences, 2022, 07(02), 371-378

Publication history: Received on 28 September 2022; revised on 19 November 2022; accepted on 28 November 2022

Article DOI: https://doi.org/10.30574/wjaets.2022.7.2.0126

Abstract

Modern banking—including open banking and digital car-loan platforms—relies on interconnected APIs across banks, fintech's, identity providers, credit bureaus, dealerships, and customers. Such ecosystems enable innovation (e.g., real-time financial data sharing, streamlined loan origination), but also expand exposure to threats like broken authentication, authorization misconfigurations (e.g., IDOR), injection attacks, data leakage, replay attacks, DoS, and more.

Profiles emerging threats across open banking and digital car-loan APIs.

Presents technical mitigations using OAuth 2.0, OpenID Connect, PKCE, and API Gateways.

Offers a refined secure architecture combining gateways, JWT handling, MTLS, RBAC/ABAC, WAFs, encryption, and monitoring.

Demonstrates how to secure a car-loan API flow—from login to loan issuance—with NFT-style nonces, token binding, scope enforcement, and logging.

Reviews operations practices: DevSecOps, auditing, incident response, and regulatory compliance.

Explores future innovations: DPoP (proof-of-possession), OAuth 2.1 updates, token binding, AI-driven threat detection, SSI, and standards-based API governance.

Keywords- API Security; Open Banking; OAuth 2.0; OpenID Connect (OIDC); API Gateway; Digital Transformation; Car Loan API Security; API Threat Landscape; Token Binding; PKCE; Mutual TLS; Secure API Architecture; API Vulnerability Mitigation; Zero Trust API Security; DevSecOps for APIs

1. Introduction

The financial services industry is undergoing unprecedented digital transformation driven by consumer demand, regulatory changes, and technological innovation. At the heart of this transformation is the Application Programming Interface (API), which acts as a connective tissue between banks, fintech companies, third-party service providers (TPPs), and end users. APIs enable real-time data sharing, seamless user experiences, and the integration of diverse financial services. However, this interconnectedness introduces new cybersecurity challenges.

* Corresponding author: Ashish Hota

1.1. The Rise of Open Banking and API-Driven Finance:

Open Banking refers to the practice of banks securely sharing customer financial data with licensed third parties via standardized APIs, under explicit user consent. Originating in the UK with the Open Banking Standard (2016), it gained global momentum with regulations like:

- **PSD2 (Payment Services Directive 2)** – Mandated in the EU (2018) requiring secure API access to account information and payment initiation services.
- **Australia's Consumer Data Right (CDR)** – Extended open banking principles to broader consumer data (2019).
- **Brazil's Open Banking Implementation Guidelines** – Emphasizing API standardization and security (2020).
- **U.S. Open Finance initiatives** – Market-led but increasingly regulated under CFPB proposals.

Parallel to open banking, **digital car-loan platforms** have evolved to offer end-to-end online experiences—allowing customers to apply for loans, upload documentation, undergo credit checks, and receive approval—all via API interactions between banks, credit bureaus, and automotive dealerships.

This evolution marks a shift from monolithic banking systems to API-centric microservices architectures, characterized by high interconnectivity and dynamic data exchange.

1.2. API Ecosystems: The New Attack Surface

While APIs offer scalability and agility, they expand the attack surface exponentially. Unlike traditional web applications, APIs expose business logic directly, making them prime targets for:

- **Authentication & Authorization Attacks**: Exploiting weak OAuth flows or misconfigured permissions.
- **Data Leakage**: Through verbose API responses or improper input/output validation.
- **Injection Attacks**: Leveraging unescaped parameters in API requests (SQLi, XSS).
- **Distributed Denial of Service (DDoS)**: Via high-volume API calls or bot networks.

A Salt Security report (2021) indicated that 90% of organizations experienced API security incidents, with 20% suffering data breaches due to API vulnerabilities. Gartner projected that by 2022, API abuses would move from infrequent to the most frequent attack vector, leading to data breaches for enterprise web applications (Gartner, 2020).

1.3. Regulatory and Privacy Considerations

API security is not just about technical resilience—it is crucial for regulatory compliance

- **GDPR (General Data Protection Regulation)**: Requires data protection by design, impacting API logging and PII handling.
- **GLBA (Gramm-Leach-Bliley Act)**: In the U.S., governs how financial institutions handle private customer information.
- **PCI DSS (Payment Card Industry Data Security Standard)**: Impacts APIs dealing with payment processing.

These mandates reinforce the need for tokenization, end-to-end encryption (E2EE), and fine-grained access control in API design.

1.4. API Security in Digital Car Loan Platforms

Digital car loan platforms involve multiple API interactions:

- **Authentication**: Customer logs in using OAuth 2.0 with PKCE.
- **Loan Application Submission**: POST request carrying PII and financial data.
- **Credit Bureau API Call**: Retrieves FICO/credit scores (often via partner OAuth).
- **Loan Offer Retrieval**: GET /loan/offer/{customerId}.
- **Document Uploads**: Multipart/form-data APIs.

Each stage poses unique security risks:

Table 1 Security Threats and Mitigations Across Car-Loan API Stages

Stage	Example Threat	Mitigation
Authentication	OAuth token interception	PKCE, TLS 1.3, MTLS
Loan Application	Parameter tampering (loanAmount)	JSON schema validation, ABAC
Credit Bureau Call	Replay attacks	Short-lived tokens, nonce, jti tracking
Offer Retrieval	IDOR/BOLA	Subject binding in JWT, RBAC
Document Upload	Malicious payload injection	Antivirus scanning, content-type checks

1.5. Key Security Challenges

1.5.1. Authentication & Authorization Gaps

OAuth 2.0 and OpenID Connect (OIDC) are widely adopted but often poorly implemented. Common mistakes include:

- Not using PKCE for public clients.
- Storing tokens insecurely on mobile devices.
- Excessively broad scopes.

1.6. Excessive Data Exposure

APIs often return verbose responses (over-fetching), exposing PII or internal data structures.

Example:

json

```
{
  "customerId": "12345",
  "name": "John Doe",
  "ssn": "123-45-6789",
  "loanAmountApproved": 25000,
  "internalRiskScore": 42
}
```

Mitigation: Response filtering (projection), encrypting sensitive fields.

- **Inadequate Monitoring -** API traffic lacks visibility without proper logging and analytics, hampering anomaly detection.
- **Rapid Scaling -** In auto finance, APIs handle sudden spikes (e.g., end-of-quarter dealer sales). Without throttling and rate-limiting, systems become susceptible to DDoS.

Research Objective and Scope

This journal aims to:

- **Identify**: Common API security threats in open banking and car-loan APIs.
- **Analyze**: How standards like OAuth 2.0, OIDC, PKCE, and API gateways counteract threats.

- **Design**: A secure API ecosystem architecture.
- **Validate**: The proposed model through a car-loan API case study.
- **Forecast**: Future trends such as DPoP, token binding, and AI-driven API security.

Table 2 Regulatory Requirements Impacting API Security

Regulation	Impact on APIs
GDPR	Secure data storage, minimal data in API responses
PSD2	Strong Customer Authentication (SCA), secure tokens
GLBA	Access control for financial information APIs
PCI DSS	Encryption and masking of payment-related API data

1.7. Why API Security Requires Defense-in-Depth

Defense-in-depth combines client-side protections (PKCE, secure token storage), gateway-level controls (rate limiting, WAF, JWT validation), and service-layer hardening (ABAC, encrypted storage). This layered approach mitigates both technical (e.g., replay attacks) and business logic vulnerabilities (e.g., loanAmount manipulation).

Key Principle: *"Trust no single layer—assume breach and validate at each hop."*

1.8. Contribution of This Journal

- Proposes a holistic security model combining OAuth 2.0, OIDC, API Gateway Hardening, and DevSecOps practices.
- Offers technical depth suitable for practitioners and researchers.
- Includes real-world case study from the digital car-loan domain.

2. APIS in digital banking and car-loan platforms

2.1. Ecosystem Participants

Table 3 API Ecosystem Roles and Security Responsibilities in Digital Car Loan Platforms

Role	API Integration	Security Responsibilities
Customer (Web/Mobile)	OAuth login, loan apps	Secure token storage (e.g., Keychain, Keystore)
Identity Provider	OIDC login, token issuance	PKCE, JWK rotation, nonce, redirect URI vetting
API Gateway	JWT validation, routing	Rate limiting, WAF, MTLS, RBAC enforcement
Bank Core Systems	Balance, account, ledger	RBAC, audit logging, database encryption
Credit Bureau	Credit scoring interface	Client credentials OAuth, TLS
Dealership Systems	Loan quote, doc uplink	Service tokens, ID matching
Microservices	Loan underwriting, scoring	Input validation, ABAC, custom logging

2.2. Typical API Flows

- **Authentication**: OIDC auth code + PKCE flow
- **Bank Data API**: scope=accounts.read
- **Loan Application**: POST /loan/apply requiring scope=loan.apply
- **Credit Bureau Call**: OAuth 2 client_credentials
- **Loan Finalization**: Confirm and disbursement with scope=loan.confirm

2.3. API Varieties & Security Needs

Table 4 API Types, Access Models, and Corresponding Security Focus

API Type	Access Model	Security Focus
Public CMP	OIDC + PKCE	User consent, short-lived tokens
Partner APIs	Opaque tokens	Client-auth coupled with JWT translation
Internal APIs	MTLS + JWT	High-sensitivity, service-to-service

3. Threat Landscape for API Ecosystems

3.1. API Threat Taxonomy

Table 5 Common API Security Threats and Mitigation Layers in Digital Banking

Threat	Description	Mitigation Layers
Broken AuthN	Stolen tokens, flawed login flows	OIDC, PKCE, token expiry, nonce, secure storage
IDOR / BOLA (OWASP A1)	Unauthorized object reference via IDs	JWT subject enforcement, backend ABAC, OAS ESS schema authlete.com+8developer.constantcontact.com+8Auth0+8arXiv+3Escape+3Business Insider+3developers.arcgis.com+1developer.constantcontact.com+1Business InsiderLinkedInarXiv
Excessive Data Exposure (A3)	Over-sharing in responses	Response filtering, mapping, versioning
Injection (A5)	SQLi, XSS via unchecked inputs	Schema validation (JSON schema), WAF, parameter parsing
Broken Function Auth (A5)	Privilege bypass via unsecured endpoints	Gateway-level scope/RBAC, internal ABAC
Misconfiguration (A7)	Weak TLS, open CORS, debug endpoints	Hardened configs, no default, gateway TLS enforcement EscapeABA Banking Journal
Token Replay / MITM	Reuse of intercepted codes/tokens	PKCE, token binding, short TTL, nonce
DoS / Bot Abuse (A6)	High requests or automated scraping	Rate limiting, CAPTCHA, anomaly detection
Insufficient Logging	Lack of traceability for forensic or compliance needs	Centralized SIEM, correlation with jti, sub, scopes

3.2. Car-Loan API: Attack Scenarios

- Maliciously altering loanId or customerId (IDOR) → Mitigation: subject-binding + request path claims
- PII exposure due to unfiltered API responses or logs → Masking, selective JSON projection
- Replay or double loan submission with same token → Nonce, jti tracking
- Bot-driven mass quote extraction → Abusive pattern detection + throttling

4. Security Standards: oauth 2.0 and openid Connect

4.1. OAuth 2.0 + PKCE Deep Dive

OAuth relies on bearer tokens issued via grant flows. Critical for public clients (web/SPAs) is PKCE (RFC 7636, 2015) backstage.forgerock.com+8IETF Datatracker+8Auth0+8. Sequence:

- Client generates random code_verifier (~128 bits) and code_challenge = BASE64URL(SHA256(verifier))
- Sends challenge to /authorize
- On callback, sends code + verifier to /token
- Server verifies SHA256(hash(verifier)) == code_challenge stored
  - Rejects mismatches (code injection mitigation)
  - Helps prevent interception and CSRF attacks Auth0+3authlete.com+3developers.arcgis.com+3Auth0+2developers.arcgis.com+2developer.constantcontact.com+2developers.arcgis.com+2developer.constantcontact.com+2Auth0+2Microsoft Learn

Additionally

- Redirect URI whitelisting prevents open redirect attacks (RFC 6819)
- Nonce prevents replay
- TLS required always; code flows must be confidential Curity

4.2. OpenID Connect (OIDC)

OIDC adds an ID Token (JWT) carrying claims (sub, nonce, iat, exp, etc.) for identity assertions. Servers must:

- Validate sig via JWKS and issuer
- Confirm nonce
- Enforce exp, iat, aud, sub

Misconfigured clients or dynamic registration may lead to reliances on malicious redirection endpoints or claims manipulation Curity.

4.3. Token Binding and DPoP

Token Binding binds tokens to TLS or cryptographic keys; DPoP binds access tokens to client-held proof-of-possession keys. OAuth 2.1 drafts index DPoP as a recommended extension to prevent replay in public clients. These reduce impact even if tokens are leaked.

5. API gateway as a defense mechanism

Gateways centralize controls, reducing duplicated effort.

5.1. Functions Table

Table 6 API Gateway Security Functions and Implementation Strategies

Function	Purpose	Implementation Examples
Authentication	JWT introspection or validation	JWKS key set, OIDC signature verification
Authorization	Coarse RBAC/scope enforcement	sub=customerId, scope=loan.apply filters
Token Translation	Opaque-to-JWT translation	Phantom token / token exchange via gateway LinkedIn+2Curity+2F5, Inc.+2
Rate Limiting	Prevent DoS and abuse	Rate limits per client, path, IP
Input Validation / WAF	Mitigate injection, invalid inputs	JSON schema validation, OWASP rules Escape
MTLS & TLS enforcement	Secure transport channel	Mutual TLS upstream, strong ciphers
Logging & Monitoring	Audit, compliance, anomaly detection	SIEM, trace correlation (jti, sub)
Threat Protections	Bot detection, IP reputation, blocking	Adaptive WAF, blacklists

Best practices: use always-on gateway + central OAuth server CurityABA Banking Journal.

5.2. Gateway Enforcement Sequence

Client -> Gateway: POST /loan/apply

- Gateway
  - Validates TLS (PFS, no weak ciphers)
  - Validates JWT (sig, exp, aud, scope)
  - Checks rate-limits
  - Scans request JSON via WAF/JSON schema
  - strips extraneous fields like `customerId`
  - Adds `X-Loan-Requester: sub`
  - Forwards to loan-service via MTLS
- Loan-service
  - - Re-validates `sub`, enforces ABAC
  - - Validates JSON schema again
  - - Executes business logic
  - - Returns JSON (filtered w/ projection)
- Gateway:
  - - Logs `sub`, `jti`, path, response code
  - - Sends to SIEM/ELK

6. Proposed secure architecture

6.1. High-Level Diagram

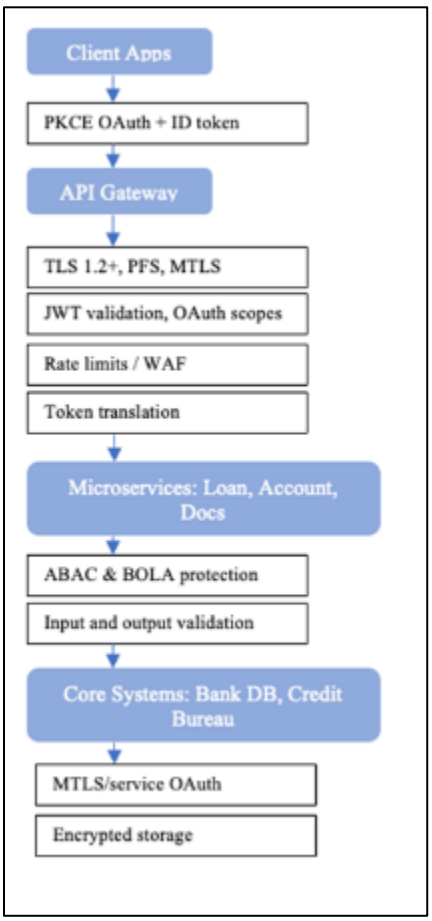

Figure 1 High-Level Secured Architecture Diagram

6.2. Mixed Table: Threat → Mitigation

Table 7 Threat Mitigation Mapping Across API Gateway and Service Layers

Threat	Gateway Role	Service Role
Token Replay	PKCE, nonce, DPoP	jti check, TTL enforcement
IDOR	Strip IDs, enforce sub on path	RBAC/ABAC schema checks
Injection	JSON schema validation	ORM parameter binding
Data Exposure	Response projection	Filtering, PII redaction
DoS/Bot	Request throttles, IP blocking	CAPTCHA, risk scoring

7. Conclusion

The study comprehensively examined the evolving security challenges in API ecosystems within digital banking, particularly in open banking and car-loan platforms, and proposed a robust, layered defense model incorporating OAuth 2.0, OpenID Connect, PKCE, and secure API gateways. By aligning threat models with mitigation strategies such as token binding, schema validation, and identity enforcement, it highlights the importance of designing secure, compliant, and resilient APIs. This research provides a practical framework for financial institutions to proactively secure their API-driven services, ultimately fostering greater trust, data privacy, and innovation—and contributes to a safer digital financial infrastructure for society while guiding future advancements in secure API design.

References

[1] Hardt, D. *The OAuth 2.0 Authorization Framework (RFC 6749)*, IETF, 2012.

[2] Lodderstedt, T. et al., *OAuth 2.0 Threat Model and Security Considerations (RFC 6819)*, IETF, 2013.

[3] Jones, M. et al., *Proof Key for Code Exchange (PKCE) for OAuth 2.0 (RFC 7636)*, IETF, 2015.

[4] OWASP Foundation. *API Security Top 10*, 2019.

[5] Hussain, H., et al., *Enterprise API Security and GDPR Compliance: Design Perspective*, IEEE, 2019.

[6] Gartner. *API Security Best Practices*, 2020.

[7] Campbell, B. et al., OAuth 2.0 Mutual TLS Client Authentication and Certificate-Bound Access Tokens, IETF, 2020.

[8] Fett, D., Küsters, R., & Schmitz, G., A Comprehensive Formal Security Analysis of OAuth 2.0, IEEE S&P, 2016.

[9] Hussain, H., Noye, D., & Sharieh, A., Enterprise API Security and GDPR Compliance: Design and Implementation Perspective, IEEE, 2019.

[10] Balebako, R., et al., The Security of Modern API Ecosystems, IEEE Security & Privacy, 2018.

[11] Hang, Z., et al., Mitigating API Security Risks in Cloud Environments, IEEE Cloud Computing, 2018.