

# Contents

<b>Authentication Security Architecture</b>	<b>2</b>
Table of Contents . . . . .	2
Architecture Overview . . . . .	2
Security Principles . . . . .	3
Identity Management . . . . .	3
User Pool Separation . . . . .	3
Tenant Isolation . . . . .	4
Authentication Flows . . . . .	4
Password Authentication Flow . . . . .	4
Security Controls in Flow . . . . .	4
Token Security . . . . .	5
Token Types . . . . .	5
JWT Structure . . . . .	5
Token Validation . . . . .	6
Token Storage Best Practices . . . . .	6
Multi-Factor Authentication . . . . .	6
TOTP Implementation . . . . .	6
MFA Enrollment Security . . . . .	6
Backup Codes . . . . .	7
Session Management . . . . .	7
Session Properties . . . . .	7
Session Storage . . . . .	7
Session Termination Events . . . . .	8
Enterprise SSO Security . . . . .	8
SAML 2.0 Security . . . . .	8
OIDC Security . . . . .	8
IdP Trust Model . . . . .	8
OAuth 2.0 Security . . . . .	9
Authorization Server Security . . . . .	9
Token Security . . . . .	9
Consent Management . . . . .	9
Threat Mitigation . . . . .	9
Threat Model . . . . .	9
Rate Limiting . . . . .	10
Anomaly Detection . . . . .	10
Audit and Compliance . . . . .	10
Authentication Events Logged . . . . .	10
Log Storage . . . . .	11
Compliance Frameworks . . . . .	11
Cryptographic Standards . . . . .	11
Algorithms in Use . . . . .	11
Key Management . . . . .	11
Security Recommendations . . . . .	12
For Tenant Administrators . . . . .	12
For Platform Administrators . . . . .	12
For Developers . . . . .	12

## Authentication Security Architecture

**Version:** 5.52.29 | **Last Updated:** January 25, 2026 | **Audience:** Security Teams, Architects, Compliance Officers

This document describes RADIANT's authentication security architecture, including threat models, security controls, compliance considerations, and implementation details.

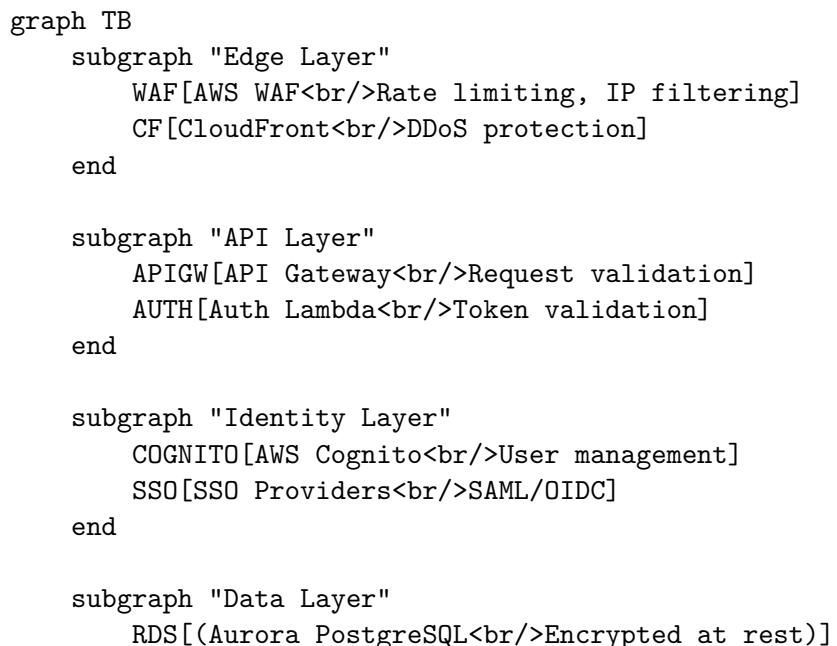
---

### Table of Contents

1. Architecture Overview
  2. Identity Management
  3. Authentication Flows
  4. Token Security
  5. Multi-Factor Authentication
  6. Session Management
  7. Enterprise SSO Security
  8. OAuth 2.0 Security
  9. Threat Mitigation
  10. Audit and Compliance
  11. Cryptographic Standards
- 

### Architecture Overview

RADIANT implements a defense-in-depth authentication architecture with multiple security layers:



```

SECRETS [Secrets Manager<br/>Credential storage]
end

CF --> WAF
WAF --> APIGW
APIGW --> AUTH
AUTH --> COGNITO
COGNITO --> SSO
AUTH --> RDS
AUTH --> SECRETS

style WAF fill:#fffcdd2
style COGNITO fill:#c8e6c9
style RDS fill:#bbdefb

```

## Security Principles

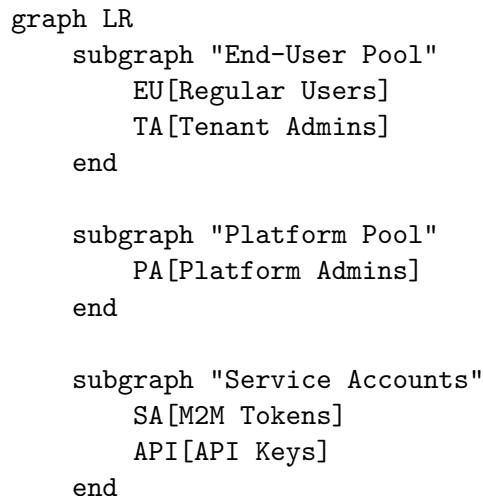
Principle	Implementation
<b>Defense in Depth</b>	Multiple security layers, no single point of failure
<b>Least Privilege</b>	Minimal permissions by default, explicit grants
<b>Zero Trust</b>	Verify every request, assume breach
<b>Fail Secure</b>	Default to deny on errors
<b>Audit Everything</b>	Complete authentication event logging

---

## Identity Management

### User Pool Separation

RADIANT maintains separate identity pools for security isolation:



Pool	Purpose	Isolation
<b>End-User</b>	Think Tank, Curator, Tenant Admin	Per-tenant RLS
<b>Platform</b>	RADIANT Admin	Separate Cognito pool
<b>Service</b>	API integrations	Token-based, no user context

## Tenant Isolation

Row-Level Security (RLS) ensures complete tenant data isolation:

```
-- All queries automatically filtered by tenant
ALTER TABLE users ENABLE ROW LEVEL SECURITY;
```

```
CREATE POLICY tenant_isolation ON users
    USING (tenant_id = current_setting('app.current_tenant_id')::uuid);
```

---

## Authentication Flows

### Password Authentication Flow

```
sequenceDiagram
    participant Client
    participant API as API Gateway
    participant Lambda
    participant Cognito
    participant DB as Database

    Client->>API: POST /auth/signin {email, password}
    API->>Lambda: Validate request
    Lambda->>Lambda: Rate limit check
    Lambda->>Cognito: InitiateAuth
    Cognito->>Cognito: Verify password (SRP)

    alt MFA Required
        Cognito-->>Lambda: MFA_REQUIRED challenge
        Lambda-->>Client: {challenge: "MFA_REQUIRED"}
        Client->>API: POST /auth/mfa {code}
        API->>Lambda: Verify MFA
        Lambda->>Cognito: RespondToAuthChallenge
    end

    Cognito-->>Lambda: Tokens
    Lambda->>DB: Create session record
    Lambda->>DB: Log authentication event
    Lambda-->>Client: {access_token, refresh_token}
```

### Security Controls in Flow

Step	Control	Purpose
Request	TLS 1.3	Encryption in transit
Request	Rate limiting	Brute force prevention
Request	CAPTCHA (optional)	Bot prevention
Validation	Input sanitization	Injection prevention
Auth	SRP protocol	Zero-knowledge password
MFA	TOTP verification	Second factor
Response	Secure cookies	XSS/CSRF protection

---

## Token Security

### Token Types

Token	Type	Lifetime	Storage
<b>Access Token</b>	JWT	1 hour	Memory only
<b>Refresh Token</b>	Opaque	30 days	Secure storage
<b>ID Token</b>	JWT	1 hour	Memory only
<b>API Key</b>	Opaque	Configurable	Server-side

### JWT Structure

```
{
  "header": {
    "alg": "RS256",
    "typ": "JWT",
    "kid": "key-id-from-jwks"
  },
  "payload": {
    "sub": "user-uuid",
    "iss": "https://cognito-idp.region.amazonaws.com/pool-id",
    "aud": "client-id",
    "exp": 1706234567,
    "iat": 1706230967,
    "auth_time": 1706230967,
    "token_use": "access",
    "scope": "openid profile",
    "tenant_id": "tenant-uuid",
    "roles": ["member"]
  },
  "signature": "..."
}
```

## Token Validation

```
flowchart TD
    A[Receive Token] --> B{Signature Valid?}
    B -->|No| X[Reject]
    B -->|Yes| C{Expired?}
    C -->|Yes| X
    C -->|No| D{Issuer Valid?}
    D -->|No| X
    D -->|Yes| E{Audience Valid?}
    E -->|No| X
    E -->|Yes| F{Revoked?}
    F -->|Yes| X
    F -->|No| G[Accept]
```

## Token Storage Best Practices

Client Type	Access Token	Refresh Token
<b>SPA</b>	Memory variable	httpOnly cookie or memory
<b>Mobile</b>	Secure enclave	Keychain/Keystore
<b>Server</b>	Memory	Encrypted database

## Multi-Factor Authentication

### TOTP Implementation

Parameter	Value
<b>Algorithm</b>	SHA-1 (RFC 6238 compliant)
<b>Digits</b>	6
<b>Period</b>	30 seconds
<b>Clock tolerance</b>	±1 period

## MFA Enrollment Security

```
sequenceDiagram
    participant User
    participant App
    participant Backend
    participant Cognito

    User->>App: Click "Enable MFA"
    App->>Backend: Request MFA setup
    Backend->>Cognito: AssociateSoftwareToken
    Cognito-->>Backend: Secret key
    Backend-->>App: QR code (not stored)
```

```

App->>User: Display QR code
User->>User: Scan with authenticator
User->>App: Enter verification code
App->>Backend: Verify code
Backend->>Cognito: VerifySoftwareToken
Cognito-->>Backend: Success
Backend->>Backend: Mark MFA enabled
Backend-->>App: MFA enabled

```

## Backup Codes

Property	Value
<b>Count</b>	10 codes
<b>Format</b>	8 alphanumeric characters
<b>Storage</b>	Hashed (bcrypt)
<b>Usage</b>	Single use, then invalidated

---

## Session Management

### Session Properties

Property	Value	Rationale
<b>Session ID</b>	UUID v4	Unpredictable
<b>Binding</b>	User + Device fingerprint	Prevent session hijacking
<b>Inactivity timeout</b>	Configurable (default 7 days)	Balance security/UX
<b>Absolute timeout</b>	Configurable (default 30 days)	Force re-authentication
<b>Concurrent limit</b>	Configurable (default 5)	Detect sharing/theft

### Session Storage

```

CREATE TABLE user_sessions (
    id UUID PRIMARY KEY,
    user_id UUID REFERENCES users(id),
    tenant_id UUID REFERENCES tenants(id),
    device_fingerprint TEXT,
    ip_address INET,
    user_agent TEXT,
    created_at TIMESTAMPTZ,
    last_activity_at TIMESTAMPTZ,
    expires_at TIMESTAMPTZ,
)

```

```

    revoked_at TIMESTAMPTZ,
    revocation_reason TEXT
);

-- Index for cleanup and validation
CREATE INDEX idx_sessions_expiry ON user_sessions(expires_at) WHERE revoked_at IS NULL;
CREATE INDEX idx_sessions_user ON user_sessions(user_id, tenant_id) WHERE revoked_at IS NULL;

```

## Session Termination Events

Event	Action
User logout	Revoke current session
Password change	Revoke all sessions
MFA reset	Revoke all sessions
Admin action	Revoke specified sessions
Suspicious activity	Revoke all sessions

## Enterprise SSO Security

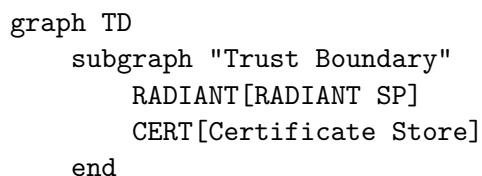
### SAML 2.0 Security

Control	Implementation
<b>Signature validation</b>	RSA-SHA256 required
<b>Assertion encryption</b>	AES-256 supported
<b>Replay prevention</b>	Assertion ID tracking, 5-min validity
<b>Audience restriction</b>	Enforce SP entity ID
<b>Time validation</b>	Clock skew 5 minutes

## OIDC Security

Control	Implementation
<b>PKCE</b>	Required for all flows
<b>State parameter</b>	Required, validated
<b>Nonce</b>	Required for implicit flows
<b>Token binding</b>	Client ID validation

## IdP Trust Model



```

subgraph "External"
    IDP[Identity Provider]
end

IDP -->|SAML Assertion| RADIANT
CERT -->|Verify Signature| RADIANT

Note1[Certificates managed<br/>per tenant]
Note2[Automatic expiration<br/>alerts]

```

---

## OAuth 2.0 Security

### Authorization Server Security

Control	Implementation
<b>PKCE</b>	Required (S256 only)
<b>Redirect URI</b>	Exact match validation
<b>State parameter</b>	Required, cryptographic
<b>Client authentication</b>	Secret for confidential clients
<b>Scope limitation</b>	Explicit consent required

### Token Security

Token Type	Security Measures
<b>Authorization Code</b>	Single-use, 10-min expiry, bound to client
<b>Access Token</b>	JWT signed RS256, 1-hour expiry
<b>Refresh Token</b>	Rotation on use, revocable

### Consent Management

```

CREATE TABLE oauth_consents (
    id UUID PRIMARY KEY,
    user_id UUID REFERENCES users(id),
    client_id TEXT NOT NULL,
    scopes TEXT[] NOT NULL,
    granted_at TIMESTAMPTZ DEFAULT NOW(),
    revoked_at TIMESTAMPTZ,
    UNIQUE(user_id, client_id)
);

```

---

### Threat Mitigation

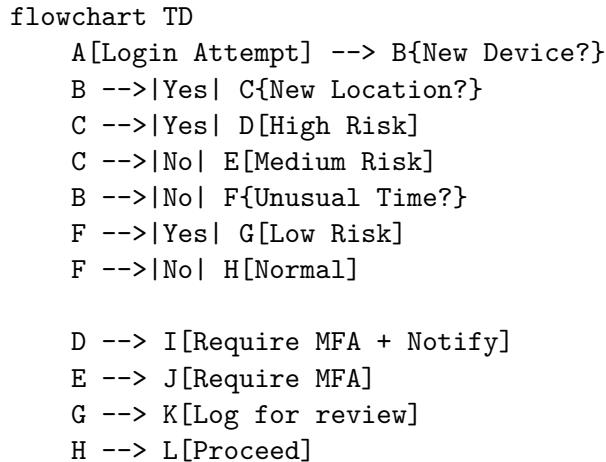
#### Threat Model

Threat	Mitigation
<b>Credential stuffing</b>	Rate limiting, CAPTCHA, breach monitoring
<b>Brute force</b>	Account lockout, progressive delays
<b>Session hijacking</b>	Secure cookies, device binding, IP validation
<b>Token theft</b>	Short expiry, secure storage, rotation
<b>Phishing</b>	MFA, passkeys, security training
<b>Man-in-the-middle</b>	TLS 1.3, certificate pinning (mobile)
<b>XSS</b>	CSP headers, httpOnly cookies, sanitization
<b>CSRF</b>	SameSite cookies, CSRF tokens

## Rate Limiting

Endpoint	Limit	Window	Action
/auth/signin	10	1 minute	Block IP
/auth/signup	5	1 minute	Block IP
/auth/password-reset	3	1 hour	Block email
/auth/mfa/verify	5	1 minute	Lock account

## Anomaly Detection




---

## Audit and Compliance

### Authentication Events Logged

Event	Data Captured
auth.attempt	User, IP, device, timestamp, method
auth.success	User, IP, device, session ID, MFA used
auth.failure	User (if known), IP, device, reason
auth.mfa.setup	User, method, timestamp
auth.mfa.verify	User, method, success/failure

Event	Data Captured
<code>auth.logout</code>	User, session ID, reason
<code>auth.session.revoke</code>	User, session ID, admin (if applicable)
<code>auth.password.change</code>	User, IP, timestamp
<code>auth.password.reset</code>	User, IP, timestamp

## Log Storage

Requirement	Implementation
<b>Retention</b>	90 days hot, 7 years archive
<b>Encryption</b>	AES-256 at rest
<b>Integrity</b>	Hash chain, tamper detection
<b>Access</b>	Role-based, audited

## Compliance Frameworks

Framework	Authentication Requirements	Status
<b>SOC 2</b>	Access controls, MFA, logging	Compliant
<b>GDPR</b>	Consent, data minimization	Compliant
<b>HIPAA</b>	Access controls, audit trails	Ready
<b>ISO 27001</b>	ISMS controls	Aligned

## Cryptographic Standards

### Algorithms in Use

Purpose	Algorithm	Key Size
<b>Password hashing</b>	Argon2id	Memory: 64MB, Time: 3
<b>Token signing</b>	RS256 (RSA-SHA256)	2048-bit
<b>Session IDs</b>	CSPRNG (UUID v4)	128-bit
<b>MFA secrets</b>	HMAC-SHA1	160-bit
<b>Backup codes</b>	CSPRNG	64-bit
<b>TLS</b>	TLS 1.3	ECDHE+AES-256-GCM

## Key Management

Key Type	Rotation	Storage
<b>Cognito signing keys</b>	Automatic (AWS managed)	AWS KMS
<b>SSO certificates</b>	Annual (manual)	Secrets Manager
<b>API secrets</b>	On compromise	Secrets Manager

Key Type	Rotation	Storage
<b>Encryption keys</b>	Annual (automatic)	AWS KMS

---

## Security Recommendations

### For Tenant Administrators

1. **Enable MFA for all admins** (enforced by default)
2. **Configure SSO** for enterprise identity management
3. **Review audit logs** regularly for anomalies
4. **Set appropriate session timeouts** for your security requirements
5. **Use IP restrictions** for admin access when possible

### For Platform Administrators

1. **Monitor rate limiting effectiveness**
2. **Review failed authentication patterns**
3. **Keep SSO certificates updated** (30-day alerts)
4. **Perform regular access reviews**
5. **Test incident response procedures**

### For Developers

1. **Never log sensitive data** (passwords, tokens)
  2. **Validate all tokens server-side**
  3. **Use secure token storage patterns**
  4. **Implement proper error handling** (no information leakage)
  5. **Follow OWASP guidelines**
- 

## Related Documentation

- [Authentication Overview](#)
- [Platform Admin Guide](#)
- [OAuth Developer Guide](#)
- [API Reference](#)
- [Compliance Documentation](#)