InsightBridge

JWT Validation & Enforcement Gateway

v4.5.0 **Production Ready** **Enterprise Grade**

Complete Documentation | January 13, 2026

📋 Table of Contents

1. Project Overview

InsightBridge v4.5 is an enterprise-grade JWT validation gateway that makes ALLOW/DENY/MONITOR decisions based on multiple security factors:

- **JWT Token Validation** - Signature verification, expiration checks, format validation
- **Rate Limiting** - Token bucket algorithm to prevent abuse
- **Replay Prevention** - JTI tracking to prevent token reuse
- **Trusted Score Enforcement** - Receiver-controlled decision making (never trusts JWT payload)

> **Core Principle:** Never trust untrusted input (JWT payload) for security decisions. Always validate using receiver-controlled trusted sources.

Use Cases

- API Gateway - Central JWT validation for microservices
- Authentication Proxy - Intermediary between clients and services
- Security Enforcement - Rate limiting and fraud detection
- Access Control - Fine-grained decision making based on trust scores

2. Key Features

Core Capabilities

✅ RS256 JWT Validation
RSA-2048 asymmetric signing with full validation

✅ Rate Limiting
Token bucket algorithm with configurable burst size

✅ Replay Detection
In-memory JTI tracking with TTL-based cleanup

✅ Score-Based Decisions
Trusted source enforcement with configurable thresholds

✅ Fail-Closed Design
All errors result in DENY for maximum security

✅ RESTful API
FastAPI with automatic OpenAPI documentation

✅ Health Monitoring
Real-time system status and metrics

✅ Request Tracking
X-Request-ID correlation and telemetry logging

3. Architecture & Project Structure

Directory Layout

```
insightbridge4.5/ ├── app/ # Main application │ ├── main.py # FastAPI entry point │ ├── config.py # Configuration (Pydantic Settings) │ ├── models.py # Request/response models │ ├── api/ # API endpoints │ ├── core/ # Business logic │ ├── middleware/ # Middleware components │ ├── persistence/ # Data layer │ └── telemetry/ # Monitoring & logging ├── tests/ # Test suite │ ├── unit/ # Unit tests │ ├── integration/ # Integration tests │ └── chaos/ # Chaos/stress tests ├── scripts/ # Utility scripts │ └── generate_test_jwts.py # Token generation ├── keys/ # Cryptographic keys ├── docs/ # Documentation ├── .env.example # Configuration template ├── requirements.txt # Dependencies └── README.md # This documentation
```

Core Components

Component	Purpose	Key File
JWT Validator	Validates token signature, expiration, format	core/jwt_validator.py
Decision Engine	Makes ALLOW/DENY/MONITOR decisions	core/decision_engine.py
Rate Limiter	Token bucket rate limiting	core/rate_limiter.py
Replay Cache	Prevents duplicate token usage	core/replay_cache.py
Score Provider	Retrieves trusted scores	core/score_provider.py
Telemetry Logger	Structured event logging	telemetry/logger.py

4. Installation & Setup

Prerequisites

- Python 3.11 or higher
- pip (Python package manager)
- Virtual environment (recommended)
- Git (for cloning)

Step-by-Step Installation

1. Clone or Extract Project

```
cd insightbridge4.5
```

2. Create Virtual Environment

```
python -m venv .venv

# Activate (Windows)
.\.venv\Scripts\Activate.ps1

# Activate (macOS/Linux)
source .venv/bin/activate
```

3. Install Dependencies

```
pip install -r requirements.txt
```

4. Generate JWT Keys

```
python scripts/generate_test_jwts.py
```

5. Configure Environment

```
cp .env.example .env
# Edit .env with your values
```

6. Run Server

```
python -m uvicorn app.main:app --host 127.0.0.1 --port 8000 --reload
```

7. Verify Installation

```
curl http://localhost:8000/health
```

5. API Endpoints

Core Validation Endpoint

POST **/validate**

Validate JWT token and get enforcement decision

Request Body:

```
{"token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9..."}
```

Response:

```
{
  "decision": "ALLOW",
  "reason": null,
  "request_id": "550e8400-e29b-41d4-a716-446655440000",
  "timestamp": "2026-01-13T10:30:00",
  "score": 95
}
```

Decision Values:

- **ALLOW** - Request permitted (score ≥ 70)
- **MONITOR** - Request monitored (50 ≤ score < 70)
- **DENY** - Request denied (score < 50)

Health & Monitoring Endpoints

GET **/health** - System health status

GET **/metrics** - Aggregated metrics summary

GET **/status** - Detailed application status

GET **/docs** - Interactive API documentation (Swagger UI)

6. Configuration Guide

Environment Variables

Variable	Default	Description
`ENVIRONMENT`	development	Environment: development, staging, production
`SECRET_KEY`	(required)	Application secret key (generate with secrets module)
`JWT_ALGORITHM`	RS256	JWT signing algorithm (RS256, HS256, etc.)
`JWT_PUBLIC_KEY_PATH`	./keys/public_key.pem	Path to RSA public key
`JWT_EXPIRATION_HOURS`	1	Token expiration time in hours
`RATE_LIMIT_REQUESTS_PER_MINUTE`	100	Rate limit threshold
`RATE_LIMIT_BURST_SIZE`	120	Maximum burst size for rate limiting

Generating Secure Configuration

```
# Generate SECRET_KEY
python -c "import secrets; print(secrets.token_urlsafe(32))"

# Output example:
# Drmhze6EPcv0fN_81Bj-nA_7d4hdsI2w3w6dH8twMVw

# Add to .env
SECRET_KEY=Drmhze6EPcv0fN_81Bj-nA_7d4hdsI2w3w6dH8twMVw
```

7. Security Implementation

JWT Validation Process

1. **Signature Verification** - RSA-2048 public key verification
2. **Expiration Check** - Validates `exp` claim with 30-second clock drift tolerance
3. **Not-Before Check** - Validates `nbf` claim to prevent premature use
4. **Format Validation** - Ensures required fields (`jti` , `sub`) are present
5. **Malformed Detection** - Rejects invalid or corrupted tokens

Replay Attack Prevention

- Tracks JWT ID (`jti`) claims in memory
- Rejects duplicate `jti` values
- Automatic TTL-based cleanup of expired entries

Rate Limiting

- **Algorithm:** Token bucket
- **Default:** 100 requests per minute
- **Burst Size:** 120 requests maximum
- **Configuration:** Per-environment via environment variables

Fail-Closed Design

> ⚠️ **Important:** All errors result in DENY
>
> - JWT validation error → DENY
> - Score retrieval error → DENY (score = 0)
> - Rate limit exceeded → DENY
> - Replay detected → DENY
> - Server error → DENY

Key Management

- **Algorithm:** RSA-2048 (2048-bit keys)
- **Rotation:** Every 12-24 months recommended
- **Storage:** Secure file system with restricted permissions
- **Backup:** Keep secure backup of private key

8. Score-Based Decision Engine

Score Thresholds

Score Range	Decision	Action
≥ 70	**ALLOW**	Request is permitted
50-69	**MONITOR**	Request allowed but logged
< 50	**DENY**	Request is blocked

Score Validation (Score > 9)

The system includes validation ensuring scores above 9 are properly handled:

- Scores from 10-49 → DENY (below threshold)
- Scores from 50-69 → MONITOR (monitoring range)
- Scores 70+ → ALLOW (trusted range)

Score Provider Implementation

Scores are retrieved from **trusted receiver-controlled sources**, never from JWT payload:

- **Database** - Query user reputation from database
- **Redis Cache** - Fast in-memory cache with database fallback
- **External API** - Call scoring service with proper authentication

> **Key Principle:** Never trust scores embedded in JWT tokens. Always fetch from internal trusted source.

9. Production Deployment

Pre-Deployment Checklist

- ☐ Generate new RSA key pair
- ☐ Set `ENVIRONMENT=production`
- ☐ Generate strong `SECRET_KEY`
- ☐ Configure database connection string
- ☐ Enable HTTPS/TLS
- ☐ Set `DEBUG_MODE=false`
- ☐ Configure rate limiting appropriately
- ☐ Setup monitoring and alerting
- ☐ Run full test suite
- ☐ Review security configuration

Docker Deployment

```
docker build -t insightbridge:4.5.0 .
docker run -p 8000:8000 \
  -e ENVIRONMENT=production \
  -e SECRET_KEY=your-secret-key \
  -e DATABASE_URL=postgresql+asyncpg://... \
  insightbridge:4.5.0
```

Performance Tuning

- **Workers:** Use multiple Uvicorn workers for concurrency
- **Database Pooling:** Configure SQLAlchemy connection pool
- **Redis Caching:** Enable Redis for distributed replay cache
- **Load Balancer:** Use load balancer for horizontal scaling

10. Testing & Validation

Running Tests

```
# All tests
pytest -v

# Unit tests
pytest tests/unit/ -v

# Integration tests
pytest tests/integration/ -v

# With coverage
pytest --cov=app tests/ -v
```

Generating Test Tokens

```
# Complete test suite
python scripts/generate_test_jwts.py

# Specific token types
python scripts/generate_test_jwts.py --type valid --user-id alice
python scripts/generate_test_jwts.py --type expired
python scripts/generate_test_jwts.py --type custom --exp-hours 24
```

Manual Testing with curl

```
# Get test token
TOKEN=$(cat test_tokens.txt | grep -A1 "Valid Token" | tail -1)

# Test validation
curl -X POST http://localhost:8000/validate \
  -H "Content-Type: application/json" \
  -d "{\"token\": \"$TOKEN\"}"

# Expected response:
# {"decision": "ALLOW", "score": 95, ...}
```

Test Coverage

- **JWT Validation:** Valid, expired, future, malformed tokens
- **Rate Limiting:** Single requests, burst, exhaustion
- **Replay Detection:** Duplicate JTI, different tokens
- **Score Thresholds:** High (ALLOW), medium (MONITOR), low (DENY)
- **Error Handling:** All failure modes

InsightBridge v4.5.0 - Complete Documentation

Generated: January 13, 2026 | Status: Production Ready ✅

For the latest updates, visit the project repository or documentation folder