Apollo.ai Zero-Knowledge Architecture

Overview

Apollo.ai implements a **true zero-knowledge architecture** where the server never has access to your sensitive data in plaintext. This document explains how this works and why it’s important for enterprise security.

What is Zero-Knowledge Architecture?

In a zero-knowledge system:

1. **Client-Side Encryption**: All sensitive data is encrypted in your browser before being sent to the server
2. **User-Controlled Keys**: Encryption keys are derived from your password and exist only in your browser’s memory
3. **Server Blindness**: The server only stores encrypted blobs and can never decrypt your data
4. **No Trust Required**: Even if the server is compromised, your data remains secure

Security Principles

Apollo.ai follows security best practices from:
- Google’s “Building Secure & Reliable Systems”
- OWASP Security Guidelines
- “The Pragmatic Programmer” design principles
- “Clean Code” architecture patterns

Key Principles Implemented:

1. **Least Privilege**: Server has minimal access - only encrypted data
2. **Defense in Depth**: Multiple layers of encryption and security
3. **Zero Trust**: Network position grants no special access
4. **Separation of Duties**: Encryption happens client-side, storage server-side

How It Works

1. Key Derivation

When you log in:

```
User Password
    ↓
PBKDF2 (600,000 iterations) + Salt
    ↓
256-bit AES-GCM Encryption Key
    ↓
Stored in Browser Memory ONLY
```

Important:
- Your password is NEVER sent to the server in plaintext
- The encryption key is NEVER sent to the server
- The server only stores a hashed version of your password for authentication

2. Database Credentials Encryption

When you add a database connection:

```
┌─────────────┐
│  Browser    │
│             │
│ 1. Enter    │
│ DB Creds    │
│             │
│ 2. Encrypt  │
│ with User   │
│ Key         │
│             │
│ 3. Send     │
│ Encrypted   │
└─────────────┘
      ↓
      ↓ (Encrypted Blob)
      ↓
┌─────────────┐
│  Server     │
│             │
│ 1. Store    │
│ Encrypted   │
│ Blob        │
│             │
│ 2. Cannot   │
│ Decrypt!    │
└─────────────┘
```

Server stores:
- Encrypted connection string
- Public metadata (connection name, type)
- Your unique salt (not secret)

Server NEVER sees:
- Your database password
- Your database username
- Your database host

3. Query Execution

When you run a query:

```
┌─────────────────────────────────┐
│         Browser                 │
│                                 │
│  1. User enters natural query   │
│     "Show me top customers"     │
│                                 │
│  2. Send to server (plaintext)  │
│     [This is safe - it's just   │
│      a natural language query]  │
└─────────────────────────────────┘
          ↓
          ↓
┌─────────────────────────────────┐
│         Server                  │
│                                 │
│  1. Convert to SQL using NLP    │
│  2. Retrieve encrypted DB creds │
│  3. Send back to browser        │
│     [Server cannot decrypt!]    │
└─────────────────────────────────┘
          ↓
          ↓
┌─────────────────────────────────┐
│         Browser                 │
│                                 │
│  1. Decrypt DB credentials      │
│  2. Execute query DIRECTLY to   │
│     your database               │
│  3. Encrypt results             │
│  4. Send encrypted results to   │
│     server for storage          │
└─────────────────────────────────┘
```

Important:
- The server helps translate natural language to SQL
- But the server NEVER connects to your database
- Your browser connects directly to your database
- Results are encrypted before being stored

4. Data Storage

All sensitive data is encrypted:

```
┌─────────────────────────────┐
│   What Server Stores        │
├─────────────────────────────┤
│                             │
│  ✓ Encrypted DB credentials │
│  ✓ Encrypted query results  │
│  ✓ User salt (not secret)   │
│  ✓ Query metadata           │
│                             │
│  ✗ Your password            │
│  ✗ Encryption keys          │
│  ✗ DB credentials (plain)   │
│  ✗ Query results (plain)    │
└─────────────────────────────┘
```

Security Guarantees

What is Protected

✅ **Database credentials** - Encrypted client-side, server cannot access
✅ **Query results containing PII** - Encrypted before storage
✅ **Sensitive connection information** - Encrypted end-to-end
✅ **User data privacy** - Server operators cannot view your data

What is NOT Protected

⚠️ **Natural language queries** - These are sent in plaintext (necessary for NLP)
⚠️ **Query metadata** - Timestamps, database names (needed for audit)
⚠️ **Public information** - Connection names, database types

Threat Model

Apollo.ai protects against:

✅ Protected Threats

1. **Compromised Server** - Even if server is hacked, encrypted data is useless
2. **Malicious Administrator** - Server admins cannot access your sensitive data
3. **Database Breach** - Stored encrypted data cannot be decrypted
4. **Man-in-the-Middle** - All sensitive data encrypted in transit
5. **Insider Threat** - Apollo.ai staff cannot access your data

⚠️ Not Protected (By Design)

1. **Compromised User Device** - If your computer is hacked, attacker may access keys in memory
2. **Weak Password** - Your encryption key is only as strong as your password
3. **Keylogger** - Password capture before encryption occurs
4. **Browser Extension Malware** - Malicious extensions can access browser memory

Best Practices

For Maximum Security:

1. **Use a Strong Password**
   - Minimum 16 characters
   - Mix of letters, numbers, symbols
   - Use a password manager

2. **Keep Your Device Secure**
   - Use full-disk encryption
   - Keep OS and browser updated
   - Use antivirus software

3. **Network Security**
   - Use VPN when on public WiFi
   - Ensure HTTPS is always used
   - Enable firewall

4. **Regular Security Audits**
   - Review audit logs regularly
   - Monitor for unusual activity
   - Rotate database credentials periodically

Compliance & Audit

Audit Logging

Apollo.ai maintains comprehensive audit logs:

- **What is logged**: User actions, timestamps, IP addresses, success/failure
- **What is NOT logged**: Encrypted data, encryption keys, plaintext credentials
- **Log storage**: Encrypted at rest, tamper-evident
- **Retention**: Configurable per compliance requirements

Compliance Standards

Zero-knowledge architecture helps with:

- **GDPR**: Data minimization, privacy by design
- **HIPAA**: PHI protection, access controls
- **SOC 2**: Security controls, access logging
- **ISO 27001**: Information security management

Technical Details

Encryption Algorithms

- **Key Derivation**: PBKDF2-HMAC-SHA256 (600,000 iterations)
- **Symmetric Encryption**: AES-256-GCM (authenticated encryption)
- **Hashing**: SHA-256
- **Random Number Generation**: Cryptographically secure (Web Crypto API)

Why These Choices?

1. **PBKDF2**: OWASP recommended, widely supported, FIPS 140-2 compliant
2. **AES-GCM**: Authenticated encryption prevents tampering, NIST approved
3. **SHA-256**: Industry standard, quantum-resistant for hashing
4. **600K iterations**: OWASP 2024 minimum recommendation

Limitations & Tradeoffs

Performance

- Client-side encryption adds ~10-50ms per operation
- Key derivation at login takes ~500ms (intentionally slow for security)
- Acceptable tradeoff for zero-knowledge security

User Experience

- Must enter password to derive encryption key
- Cannot recover data if password is forgotten

- Session expires require re-entering password

Functionality

- Server cannot perform server-side analytics on encrypted data
- Cannot implement server-side data deduplication
- Limited server-side search capabilities on encrypted content

FAQs

Q: What if I forget my password?

A: If you forget your password, **your encrypted data cannot be recovered**. This is an inherent property of zero-knowledge architecture. We recommend:
- Using a password manager
- Maintaining database backups independently
- Documenting recovery procedures

Q: Can Apollo.ai staff access my data?

A: No. Apollo.ai staff and administrators **cannot access your encrypted data** even if they wanted to. The encryption keys exist only in your browser's memory and are derived from your password, which we never store in plaintext.

Q: Is this really secure?

A: Yes, assuming:
1. You use a strong password
2. Your device is not compromised
3. You follow security best practices

Zero-knowledge is a proven security model used by password managers (1Password, Bitwarden), secure messaging apps (Signal), and encrypted cloud storage providers.

Q: Why not encrypt the natural language queries too?

A: Natural language queries need to be processed by the NLP engine on the server to convert them to SQL. However, these queries are typically generic (e.g., "show top customers") and don't contain credentials or sensitive personal data. The resulting data is what we encrypt.

Q: What happens if the server is compromised?

A: If the server is compromised:
- ✅ Your encrypted data remains secure
- ✅ Your database credentials cannot be accessed
- ⚠️ Attacker could modify the application code
- ⚠️ Attacker could log natural language queries

We implement code integrity checks and monitoring to detect tampering.

Q: Can I audit the security?

A: Yes! Apollo.ai:
- Open security architecture documentation (this file)
- Comprehensive audit logging

- Regular security assessments
- Third-party penetration testing

Future Enhancements

Planned improvements:

1. **Homomorphic Encryption**: Enable server-side computation on encrypted data
2. **Zero-Knowledge Proofs**: Prove data properties without revealing data
3. **Hardware Security Modules**: Enhanced key protection
4. **Multi-Party Computation**: Distributed trust model

Support

For security questions or concerns:
- Review audit logs in Apollo.ai dashboard
- Consult your security team
- Contact Apollo.ai security team

Last Updated: November 2025
Version: 1.0
Security Review: Compliant with OWASP, NIST, and Google SRE best practices