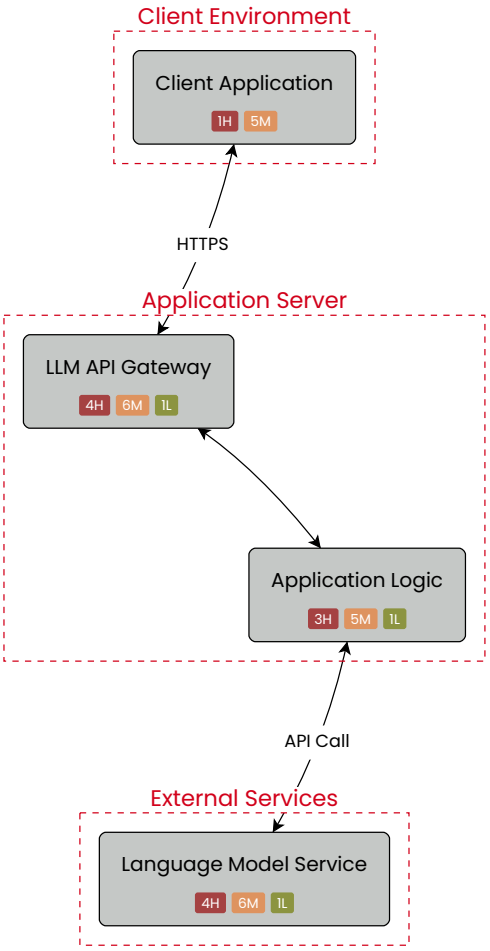
ThreatCanvas

by SecureFlag

Untitled model

11/25/2025, 2:23:09 PM

Diagram



Project settings

Project type	Web

Open risks

Threat	Node	Risk rating
Broken Access Controls	Application Logic	High
Software and Data Integrity Failures	Application Logic	High
Weak Authentication Mechanisms	Application Logic	High
Software and Data Integrity Failures	Client Application	High
Broken Access Controls	Language Model Service	High
Injection Attacks	Language Model Service	High
Software and Data Integrity Failures	Language Model Service	High
Weak Authentication Mechanisms	Language Model Service	High
Broken Access Controls	LLM API Gateway	High
Injection Attacks	LLM API Gateway	High
Software and Data Integrity Failures	LLM API Gateway	High
Weak Authentication Mechanisms	LLM API Gateway	High
Cryptographic Failures	Application Logic	Moderate
Insecure Design	Application Logic	Moderate
Man-in-the-middle Attack	Application Logic	Moderate
Security Misconfiguration	Application Logic	Moderate
Vulnerable and Outdated Component	Application Logic	Moderate
Cryptographic Failures	Client Application	Moderate
Insecure Design	Client Application	Moderate
Man-in-the-middle Attack	Client Application	Moderate
Security Misconfiguration	Client Application	Moderate
Vulnerable and Outdated Component	Client Application	Moderate
Cryptographic Failures	Language Model Service	Moderate
Insecure Design	Language Model Service	Moderate

Threat	Node	Risk rating
Man-in-the-middle Attack	Language Model Service	Moderate
Security Misconfiguration	Language Model Service	Moderate
Server-Side Request Forgery (SSRF)	Language Model Service	Moderate
Vulnerable and Outdated Component	Language Model Service	Moderate
Cryptographic Failures	LLM API Gateway	Moderate
Insecure Design	LLM API Gateway	Moderate
Man-in-the-middle Attack	LLM API Gateway	Moderate
Security Misconfiguration	LLM API Gateway	Moderate
Server-Side Request Forgery (SSRF)	LLM API Gateway	Moderate
Vulnerable and Outdated Component	LLM API Gateway	Moderate
Insufficient Logging	Application Logic	Low
Insufficient Logging	Language Model Service	Low
Insufficient Logging	LLM API Gateway	Low

Node analysis

Client Application

Component	Generic Process
Trust boundary	Client Environment

Cryptographic Failures

Risk rating	Moderate
Status	Open

Securely Store and Rotate Keys

Implemented	No

Use Modern Encryption Libraries

Implemented	No

Insecure Design

Risk rating	Moderate
Status	Open

Layered Architecture and Tenant Segregation

Implemented	No

Rigorous Testing and Resource Limits

Implemented	No

Secure Development Lifecycle

Implemented	No

Man-in-the-middle Attack

Risk rating	Moderate
Status	Open

Authentication of Client Certificate

Implemented	No

Authentication of Server Certificate

Implemented	No

Secure Connections with Strong Encryption

Implemented	No

Security Misconfiguration

Risk rating	Moderate
Status	Open

Configuration Management

Implemented	No

Secure Defaults

Implemented	No

Security Audit

Implemented	No

Software and Data Integrity Failures

Risk rating	High
Status	Open

Enable Runtime Integrity Monitoring

Implemented	No

Verify Integrity of Updates and Dependencies

Implemented	No

Vulnerable and Outdated Component

Risk rating	Moderate
Status	Open

Patch Management

Implemented	No

LLM API Gateway

Component	Generic Process
Trust boundary	Application Server

Broken Access Controls

Risk rating	High
Status	Open

Apply Least Privilege

Implemented	No

Enforce Authorization

Implemented	No

Unique User Identification

Implemented	No

Use Role-Based or Attribute-Based Controls

Implemented	No

Cryptographic Failures

Risk rating	Moderate
Status	Open

Securely Store and Rotate Keys

Implemented	No

Use Modern Encryption Libraries

Implemented	No

Injection Attacks

Risk rating	High
Status	Open

Input Sanitization

Implemented	No

Input Validation

Implemented	No

Insecure Design

Risk rating	Moderate
Status	Open

Layered Architecture and Tenant Segregation

Implemented	No

Rigorous Testing and Resource Limits

Implemented	No

Secure Development Lifecycle

Implemented	No

Insufficient Logging

Risk rating	Low
Status	Open

Logging and Monitoring

Implemented	No

Man-in-the-middle Attack

Risk rating	Moderate
Status	Open

Authentication of Client Certificate

Implemented	No

Authentication of Server Certificate

Implemented	No

Secure Connections with Strong Encryption

Implemented	No

Security Misconfiguration

Risk rating	Moderate
Status	Open

Configuration Management

Implemented	No

Secure Defaults

Implemented	No

Security Audit

Implemented	No

Server-Side Request Forgery (SSRF)

Risk rating	Moderate
Status	Open

Input Validation

Implemented	No

Software and Data Integrity Failures

Risk rating	High
Status	Open

Enable Runtime Integrity Monitoring

Implemented	No

Verify Integrity of Updates and Dependencies

Implemented	No

Vulnerable and Outdated Component

Risk rating	Moderate
Status	Open

Patch Management

Implemented	No

Weak Authentication Mechanisms

Risk rating	High
Status	Open

Enforce Authentication

Implemented	No

Mitigate Automated Attacks

Implemented	No

Multi-Factor Authentication

Implemented	No

Password Policies

Implemented	No

Update Default Credentials

Implemented	No

Application Logic

Component Generic Process
Trust boundary Application Server

Broken Access Controls

Risk rating High
Status Open

Apply Least Privilege

Implemented No

Enforce Authorization

Implemented No

Unique User Identification

Implemented No

Use Role-Based or Attribute-Based Controls

Implemented No

Cryptographic Failures

Risk rating Moderate
Status Open

Securely Store and Rotate Keys

Implemented No

Use Modern Encryption Libraries

Implemented No

Insecure Design

Risk rating Moderate
Status Open

Layered Architecture and Tenant Segregation

Implemented No

Rigorous Testing and Resource Limits

Implemented	No

Secure Development Lifecycle

Implemented	No

Insufficient Logging

Risk rating	Low
Status	Open

Logging and Monitoring

Implemented	No

Man-in-the-middle Attack

Risk rating	Moderate
Status	Open

Authentication of Client Certificate

Implemented	No

Authentication of Server Certificate

Implemented	No

Secure Connections with Strong Encryption

Implemented	No

Security Misconfiguration

Risk rating	Moderate
Status	Open

Configuration Management

Implemented	No

Secure Defaults

Implemented	No

Security Audit

Implemented	No

Software and Data Integrity Failures

Risk rating	High
Status	Open

Enable Runtime Integrity Monitoring

Implemented	No

Verify Integrity of Updates and Dependencies

Implemented	No

Vulnerable and Outdated Component

Risk rating	Moderate
Status	Open

Patch Management

Implemented	No

Weak Authentication Mechanisms

Risk rating	High
Status	Open

Enforce Authentication

Implemented	No

Mitigate Automated Attacks

Implemented	No

Multi-Factor Authentication

Implemented	No

Password Policies

Implemented	No

Update Default Credentials	
Implemented	No

Language Model Service

Component	Generic Process
Trust boundary	External Services

Broken Access Controls

Risk rating	High
Status	Open

Apply Least Privilege

Implemented	No

Enforce Authorization

Implemented	No

Unique User Identification

Implemented	No

Use Role-Based or Attribute-Based Controls

Implemented	No

Cryptographic Failures

Risk rating	Moderate
Status	Open

Securely Store and Rotate Keys

Implemented	No

Use Modern Encryption Libraries

Implemented	No

Injection Attacks

Risk rating	High
Status	Open

Input Sanitization

Implemented	No

Input Validation

Implemented	No

Insecure Design

Risk rating	Moderate
Status	Open

Layered Architecture and Tenant Segregation

Implemented	No

Rigorous Testing and Resource Limits

Implemented	No

Secure Development Lifecycle

Implemented	No

Insufficient Logging

Risk rating	Low
Status	Open

Logging and Monitoring

Implemented	No

Man-in-the-middle Attack

Risk rating	Moderate
Status	Open

Authentication of Client Certificate

Implemented	No

Authentication of Server Certificate

Implemented	No

Secure Connections with Strong Encryption

Implemented	No

Security Misconfiguration

Risk rating	Moderate
Status	Open

Configuration Management

Implemented	No

Secure Defaults

Implemented	No

Security Audit

Implemented	No

Server-Side Request Forgery (SSRF)

Risk rating	Moderate
Status	Open

Input Validation

Implemented	No

Software and Data Integrity Failures

Risk rating	High
Status	Open

Enable Runtime Integrity Monitoring

Implemented	No

Verify Integrity of Updates and Dependencies

Implemented	No

Vulnerable and Outdated Component

Risk rating	Moderate
Status	Open

Patch Management

Implemented	No

Weak Authentication Mechanisms

Risk rating	High
Status	Open

Enforce Authentication

Implemented	No

Mitigate Automated Attacks

Implemented	No

Multi-Factor Authentication

Implemented	No

Password Policies

Implemented	No

Update Default Credentials

Implemented	No

Threat reference

Broken Access Controls

The node does not perform an adequate authorization check against attackers when attempting to access data or perform actions they should not be allowed to perform.

Cryptographic Failures

The node mishandles encryption through weak algorithms, poor key management, or flawed certificate handling, resulting in unauthorized exposure or alteration of sensitive data.
Attackers can exploit these weaknesses to intercept and decrypt confidential information.

Injection Attacks

The node processes untrusted input without proper validation or sanitization. Attackers can insert malicious code or commands into system components.
This can lead to data exposure, data corruption, or full system compromise.

Insecure Design

The node's architecture and features lack robust security considerations.
Insufficient threat modeling, weak default configurations, and missing layers of defense give attackers opportunities to compromise the node's confidentiality, integrity, or availability.

Insufficient Logging

The node does not sufficiently log events such as logins, failed logins, high-value transactions, and errors.

Man-in-the-middle Attack

This node allows network traffic that is not adequately encrypted, such as unencrypted traffic, outdated TLS protocol versions, or weak cipher suites. An adversary positioned between two nodes can read and potentially manipulate transmitted information.

Security Misconfiguration

The node relies on default or improperly configured settings. Attackers can exploit these misconfigurations to gain unauthorized access, escalate privileges, or otherwise compromise the system.
Inadequate patching, incomplete hardening, or overlooked permissions create exploitable gaps that undermine the node's confidentiality, integrity, and availability.

Server-Side Request Forgery (SSRF)

The node uses untrusted input to make network requests to other nodes. Attackers can submit malicious strings to perform actions such as making a request to unintended nodes and services.

Software and Data Integrity Failures

The node does not verify the authenticity or integrity of software updates, dependencies, or critical data. Attackers can tamper with code or inject malicious alterations.
This can lead to unauthorized modifications, data corruption, and even full compromise of the node's operations.

Vulnerable and Outdated Component

The node contains vulnerable or outdated components, such as software libraries, that lack the latest security patches, exposing the system to potential exploits and breaches.

Weak Authentication Mechanisms

A node with weak authentication mechanisms, such as default passwords, weak password policies, outdated login processes, or lack of multi- factor authentication, can be exploited by attackers to gain unauthorized access, escalate privileges, or compromise sensitive data.

Control reference

Apply Least Privilege

Grant only the minimum necessary access and permissions to each user or process.
Regularly audit privileges to prevent accumulated access rights that exceed actual requirements.

Authentication of Client Certificate

When acting as the server, ensure that the node authenticates the clients' certificate.

Authentication of Server Certificate

When acting as the client, ensure that the node authenticates the server's certificate.

Configuration Management

Implement a robust configuration management process to ensure consistent settings and authorized changes. Regularly review and update configurations to address security vulnerabilities and maintain system stability. Use automated tools to enforce configuration policies and detect unauthorized changes.

Enable Runtime Integrity Monitoring

Continuously check the node's operating environment- file integrity, configurations, and running processes-for unauthorized changes. Establish alerts for any modifications to critical code or data to quickly detect and respond to breaches.

Enforce Authentication

Enforce robust authentication mechanism to access the node's resources and functionalities, such as passwords, pre-shared tokens, or digital certificates.

Enforce Authorization

Ensure that the node uses strict access policies against unauthorized access.

Input Sanitization

Check untrusted input and remove anything that might be potentially dangerous.

Input Validation

Ensure that only properly formed data is entered into the system.

Layered Architecture and Tenant Segregation

Partition the system into distinct tiers (e.g., presentation, business logic, data) and apply separate network segments based on exposure and protection requirements.
Robustly segregate tenant data and resources across all tiers, ensuring that any compromise in one tenant or layer does not spill over into another.

Logging and Monitoring

Keep detailed audit logs with timestamps for activities such as user logins, sensitive data access, access control changes, and administrative actions.

Mitigate Automated Attacks

Protect against automated attacks such as content scraping, password brute-force, or denial of service attacks.

Multi-Factor Authentication

Require the use of multiple factors to confirm the identity of someone.

Password Policies

Set and enforce secure password policies for accounts.

Patch Management

Keep software libraries, external components, and other dependencies up- to- date in an automated, risk-based, and timely manner.

Rigorous Testing and Resource Limits

Develop unit and integration tests to confirm critical flows match the threat model, and compile both use-cases and misuse-cases for each tier.
Additionally, enforce strict limits on resource consumption by user or service- covering memory, CPU, or parallel requests-to prevent denial-of-service scenarios and uphold system stability.

Secure Connections with Strong Encryption

Ensure that the node enforces secure network connections using industry- standard protocols, such as TLS, with approved versions and strong encryption mechanisms to protect data in transit from unauthorized access or exposure.

Secure Defaults

Configure the node with restrictive baseline settings. Disable unnecessary services, set strong permissions on critical files and directories, and ensure that default passwords or credentials are replaced immediately.

Secure Development Lifecycle

Collaborate with AppSec professionals throughout the design and development process. Incorporate threat modeling for critical authentication, access control, and business logic flows; integrate security requirements into user stories; and rely on a library of secure design patterns or pre-approved components.

Securely Store and Rotate Keys

Protect cryptographic keys by using a secure storage mechanism- such as a hardware security module (HSM) or a key management service- and never embed them in code or plain configuration files.
Establish a key rotation policy to regularly replace and retire keys, minimizing the impact of potential compromise over time.

Security Audit

Perform audits or scans of systems, permissions, insecure software, insecure configurations, etc. to identify potential weaknesses.

Unique User Identification

Assign a unique name and/or number for identifying and tracking user identity.

Update Default Credentials

Replace default credentials with secure, unique ones to enhance security and prevent unauthorized access.

Use Modern Encryption Libraries

Rely on well-tested, widely trusted cryptographic libraries (e.g., AES-256 for symmetric encryption, RSA-2048 for asymmetric encryption).
Keep them updated to the latest secure versions and follow recommended configurations to prevent known vulnerabilities and maintain robust encryption standards.

Use Role-Based or Attribute-Based Controls

Define clear roles with specific privileges, or use attributes (e.g., user groups, resource tags) to control access.

This structured approach helps maintain granular and easily managed authorization policies.

Verify Integrity of Updates and Dependencies

Enforce digital signatures, checksums, or similar mechanisms on software updates, libraries, and packages. Ensure that only trusted and verified dependencies are imported or installed during the build and deployment process.