Threat Analysis and Security Requirements Derivation

Management Summary

This report outlines the results of a security analysis and mitigation process applied to a system based on a detailed architecture and flow diagram. The process followed five critical steps: system review, threat analysis using STRIDE-LM, derivation of security requirements, mapping to NIST SP 800-53 controls, and identification of compliance assurance methods. The key objective was to ensure that the system's security posture is aligned with best practices and regulatory requirements, particularly in safeguarding sensitive and Personally Identifiable Information (PII) data.

The first step involved a thorough review of the provided system diagram, which included identifying all system components, data flows, and storage points. The diagram helped break down the internal structure, including user interactions, processes like Docker containers and NodeJS applications, API communications, third-party modules, and data repositories. This detailed understanding of the system architecture was foundational for conducting a comprehensive threat analysis.

Following the system review, a STRIDE-LM threat analysis was performed, which lead to the identification of 38 threats. Each identified component and data flow was evaluated for potential threats in each STRIDE-LM category, resulting in a comprehensive list of system vulnerabilities and weaknesses.

Based on the identified threats, twelve specific security requirements were derived to mitigate each risk. The security requirements focused on protecting confidentiality, integrity, and availability, with particular attention given to protecting sensitive and PII data. Requirements included:

- Implementing multi-factor authentication (MFA).
- Using strong encryption protocols (e.g., TLS 1.2+) for data in transit and at rest.
- Applying role-based access control (RBAC) to enforce least-privilege access.
- Establishing tamper-proof audit logs for non-repudiation.
- Performing regular vulnerability scans and updating software dependencies.
- Implementing rate-limiting and load balancing to mitigate DoS attacks.

Each security requirement was then mapped to relevant controls from the NIST SP 800-53 cybersecurity framework. This alignment ensures that the security requirements are not only addressing the threats but are also aligned with a globally recognized standard, reinforcing the organization's security posture.

To ensure compliance with the mapped NIST SP 800-53 controls, specific methods for continuous compliance were identified. These include:

- **Regular audits:** Periodic review of system configurations, access controls, and security logs to ensure adherence to policies.

- **Automated monitoring:** Using security monitoring tools to track anomalies in real-time and trigger alerts in case of policy violations.
- **Access controls:** Verifying that users and system components have only the necessary permissions required for their roles.
- **Encryption:** Ensuring data at rest and in transit is properly encrypted, with regular validation through network scans and key management processes.
- **Incident response plans:** Implementing robust incident detection and response capabilities to manage and mitigate security events.

These compliance methods ensure that the system remains secure over time, even as the environment changes or new threats emerge.

Background & Tasks

Derive security requirements for the system, which CI, and map these requirements to the relevant NIST SP 800-
53 controls. Explain how compliance with these controls is assured.

Tasks

1. Understand the System
    - Review the provided system diagram thoroughly.
    - Identify all components, data flows, and storage points within the system.
2. Perform STRIDE-LM Threat Analysis:
    - Use the STRIDE-LM model to identify potential threats in the system.
      STRIDE-LM stands for:
        - Spoofing
        - Tampering
        - Repudiation
        - Information Disclosure
        - Denial of Service
        - Elevation of Privilege
        - Lateral Movement
        - Man-in-the-Middle
    - Document each identified threat with a brief description.
3. Derive Security Requirements:
    - Based on the identified threats, derive security requirements to mitigate these threats.
    - Ensure that the requirements address the protection of confidential and PII data.
4. Map to NIST SP 800-53 Controls:
    - Refer to the NIST SP 800-53 framework and map each security requirement to the relevant controls.

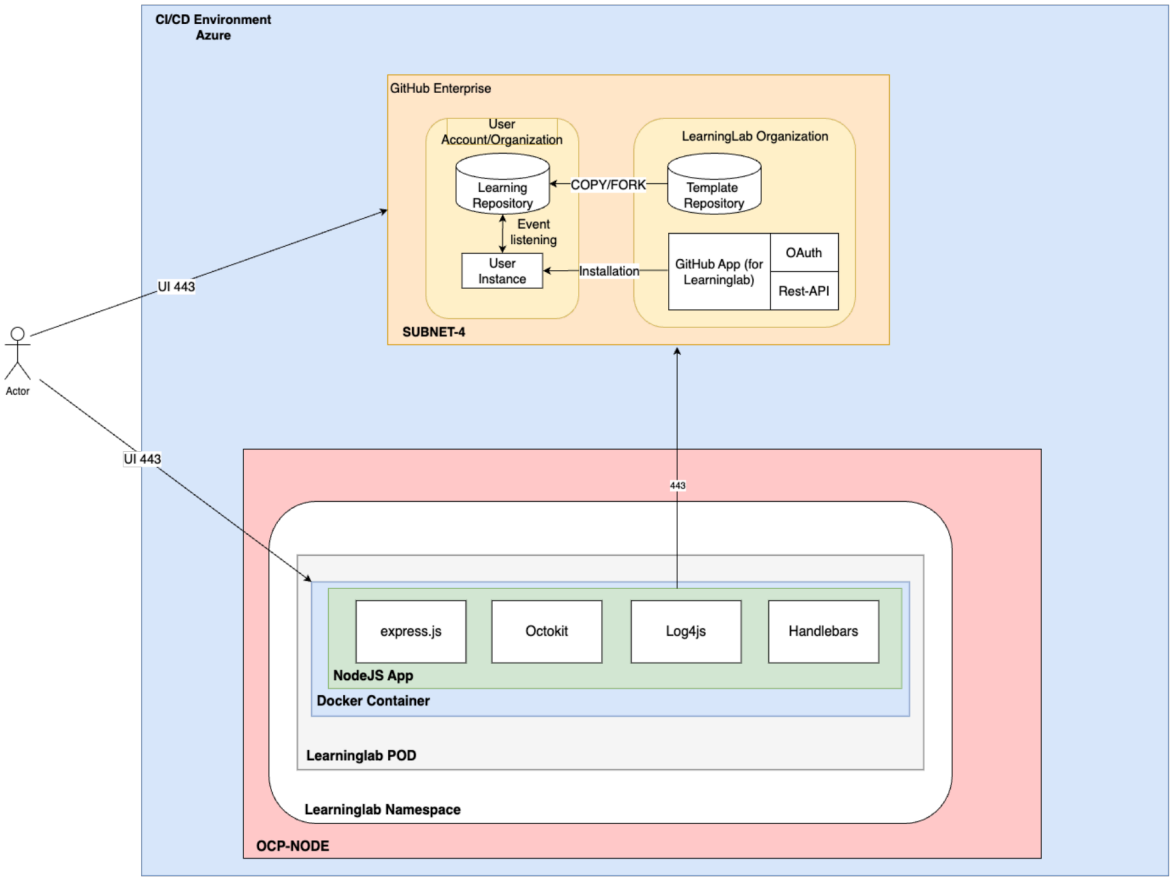
- Document the specific controls (e.g., AC-1, IA-2) that correspond to each security requirement.

5. Explain Compliance Assurance:
    - For each mapped control, explain how compliance will be assured.
    - Include methods such as regular audits, automated monitoring, access controls, encryption, and incident response plans.

Deliverables:

- A detailed report containing:
    - The system diagram with identified threats.
    - A list of derived security requirements.
    - A mapping of security requirements to NIST SP 800-53 controls.
    - An explanation of how compliance with each control will be assured.

1. Understading the system



1.1 Components and Interactions

The provided diagram describes the following components and interactions between them:

1. Actor

- **Role:** The actor (likely a user or developer) interacts with the system through a User Interface (UI) over port 443 (HTTPS communication).
- **Interaction:**
  - The actor sends/receives data from two key components: the GitHub Enterprise instance and the LearningLab environment within the OpenShift Container Platform (OCP).

2. Process GitHub Enterprise

- **Components:**
  - **User Account/Organization:** This represents the user's account or an organization account in GitHub.
  - **Learning Repository:** A GitHub repository containing learning-related materials or content, likely forked or copied from the Template Repository.
  - **User Instance:** An instance created for the user, potentially containing code, configurations, or custom settings related to the learning environment.
  - **LearningLab Organization:** A GitHub organization that holds the **Template Repository**.
  - **GitHub App (for LearningLab):** This app manages GitHub-based interactions. It includes:
    - **OAuth:** Handles authentication for the app.
    - **Rest-API:** Provides API access for programmatic operations.
- **Interactions:**
  - The **Learning Repository** is copied or forked from the **Template Repository** in the LearningLab Organization.
  - The **GitHub App** is installed to manage communication between the user and the LearningLab environment. The app listens for events and processes actions such as repository changes or user activity.

3. LearningLab Namespace (OCP-NODE)

- **Components:**
  - **LearningLab POD:** A pod running on the OpenShift (OCP) node.
  - **NodeJS App:** A Node.js-based application that is part of the LearningLab environment. It contains the following libraries or modules:
    - **express.js:** A web framework for Node.js that handles HTTP requests and routes them to different parts of the application.
    - **Octokit:** A GitHub API client for Node.js, allowing communication with GitHub services like repositories and users.
    - **Log4js:** A logging framework to monitor the application's actions and errors.

        - **Handlebars:** A templating engine used for rendering dynamic content, possibly in the UI or reports.
- **Interaction:**
    - The **NodeJS App** runs inside a Docker container, which is encapsulated within a LearningLab pod.
    - The app handles user interactions, such as repository creation and event processing, by using **Octokit** to communicate with GitHub.
    - **express.js** manages web requests, while **Log4js** logs activities, and **Handlebars** renders dynamic content.

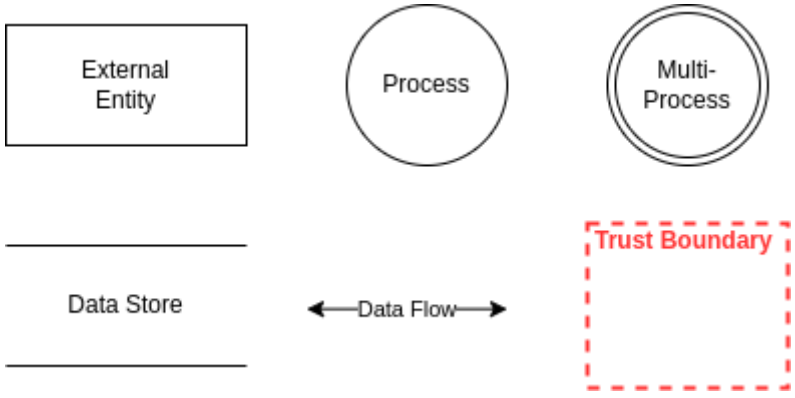
4. Communication and Flow

- The **Actor** interacts with both the **GitHub Enterprise** and the **LearningLab POD** over HTTPS (UI 443).
- The **LearningLab POD** communicates with the **GitHub Enterprise**:
    - **User Instance** listens for events (like commits, repository actions) triggered by the actor in the **Learning Repository**.
    - The **LearningLab POD** likely processes those events and actions through the **GitHub App**, which uses **OAuth** for authentication and the **Rest-API** for interactions.
    - The **Learning Repository** is either a fork or a copy of the **Template Repository** maintained by the LearningLab organization in GitHub Enterprise.

1.2. Data Flow Diagram (DFD) for STRIDE-LM Methodology

In prepararion for the STRIDE-LM Threat Analysis data flow diagrams (DFD) are derived from the provided system diagram and identified components as well as their relation to each other. The key elements of a DFD for STRIDE-LM are: external entities, processes, data stores, and data flows. The goal is to map out the flow of information to identify potential security threats using the STRIDE-LM methodology.

DFD for STRIDE-LM are created using the following six elements.



The first element, an **external entity**, represents any user, system, or organization outside the application that interacts with it. For instance, an external entity could be a user logging into a website or a third-party service exchanging data with the system. In the context of STRIDE-LM, external entities are critical points for identifying threats such as spoofing, where an attacker could

impersonate a legitimate user or system. Additionally, they raise concerns about repudiation, where users might deny their actions within the system, leading to challenges in accountability.

Next, the **data store** refers to where information is stored in the system, like databases, file systems, or any persistent storage. The data store plays a crucial role in maintaining the integrity and confidentiality of data, and it is a key focus when considering threats such as tampering and information disclosure. An attacker could tamper with the data by altering it without authorization, or they could expose sensitive data, leading to breaches in confidentiality. Moreover, denial of service can also target data stores, making them inaccessible and disrupting the system's functionality.

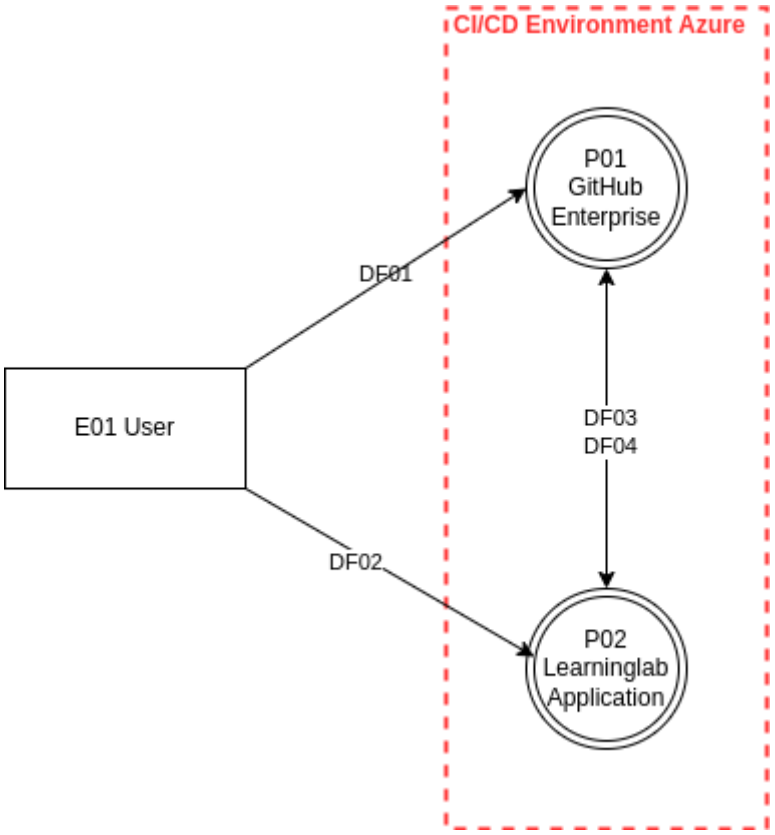
The **data flow** element illustrates how information moves between various parts of the system, whether it's between an external entity and a process, or from one process to another.

A **multi-process** (or composite process) represents a higher-level system component that encapsulates several individual processes within it. This abstraction helps organize complex systems into manageable parts, but it also opens the door to threats like elevation of privilege. An attacker might attempt to exploit vulnerabilities within one process and gain unauthorized control over others, thereby escalating their access within the system.

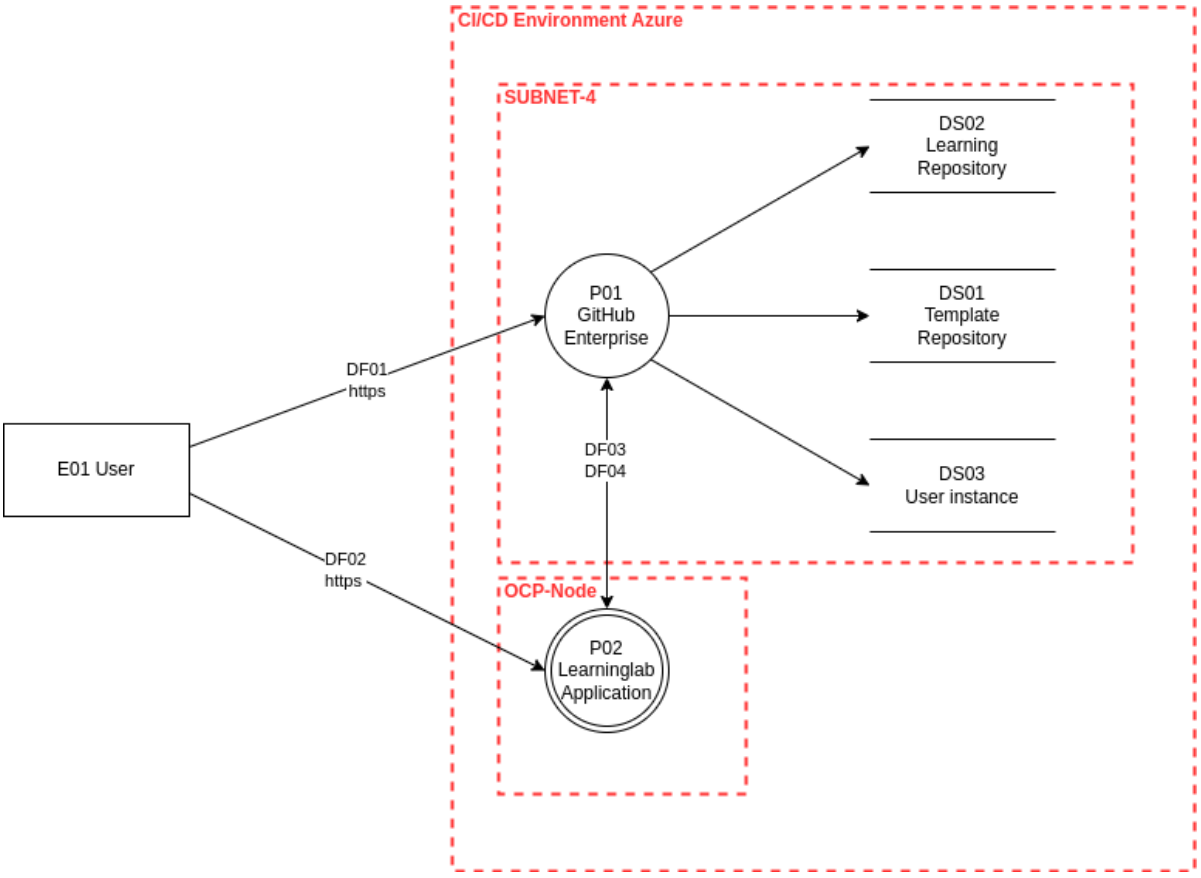
The **process** element itself signifies any operation or computation within the system that takes in data, performs some task, and outputs the result. This is where a majority of threats can occur, including tampering, where attackers manipulate the internal workings of a process, and denial of service, where they overload or disrupt processes to degrade performance. Processes are also vulnerable to elevation of privilege, where attackers exploit flaws to gain unauthorized access to sensitive parts of the system.

Finally, the **trust boundary** is a conceptual line that separates different levels of trust within the system. For example, the boundary between user input and the internal logic of an application is one such trust boundary. Data crossing this boundary must be carefully validated, as crossing from a lower-trust environment (like user inputs or external systems) into a higher-trust area can open the system to threats like spoofing, tampering, and elevation of privilege. Mismanagement at trust boundaries is a common source of vulnerabilities, as data may not be sufficiently verified before it’s processed by trusted system components.

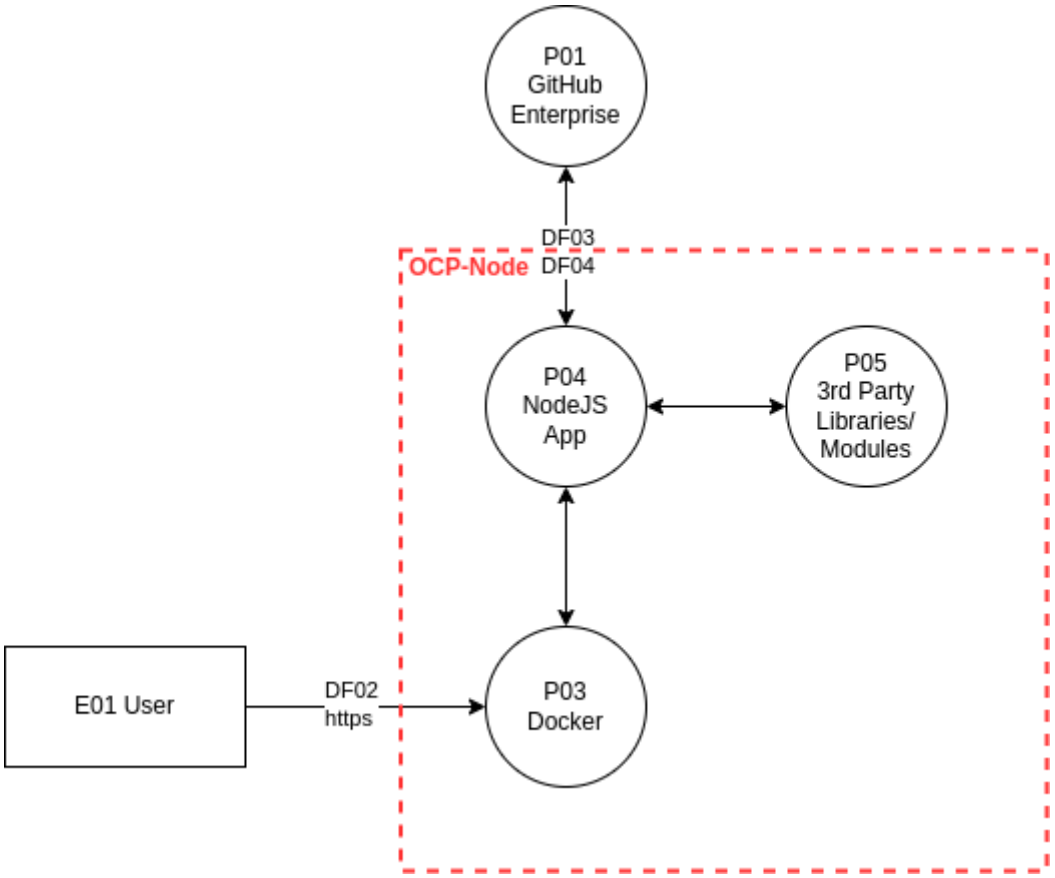
The figure below shows the DFD at context level. The actor or user interacts with two processes via HTTP.



The figure below shows a DFD at level 1.



The figure below shows a DFD the multi process P02 drilled down.



1. External Entities (Actors)

- **E01 User (Actor):** Represents the external entity interacting with the system.

2. (Multi) Processes

- **P01 GitHub Enterprise Process:**
  - Handles authentication, repository management, and event handling.
  - Key processes:
    - Authentication via **OAuth**.
    - Data flow for copying/forking the **Learning Repository**.
    - Event listening (e.g., webhooks triggered by repository changes).
- **P02 LearningLab Application Process:**
  - Manages user interactions via Node.js application (running in Docker inside an OCP pod).
  - Key processes:
    - HTTP requests handled by **express.js**.
    - API requests to GitHub via **Octokit**.
    - Logging via **Log4js**.
    - Rendering dynamic content using **Handlebars**.

3. Data Stores

- **DS01 Template Repository (GitHub):** Holds learning templates in GitHub.
- **DS02 Learning Repository (User):** Forked or copied repository specific to the user.
- **DS03 User Instance:** Represents data specific to the user stored in the system (could be code, learning materials, etc.).

4. Data Flows

- **DF01: User to GitHub Enterprise (HTTPS)**
  - User interacts with the GitHub Enterprise system to authenticate, fork repositories, and trigger events.
- **DF02: User to LearningLab App (HTTPS)**
  - User sends HTTP requests to the LearningLab application to manage learning environments.
- **DF03: LearningLab App to GitHub Enterprise (API calls)**
  - LearningLab App communicates with GitHub using Octokit for repository and user management.
- **DF04: Event Listening from GitHub Enterprise to LearningLab App**
  - GitHub sends event notifications (like repository updates) to the LearningLab App via webhooks.
- **DF05: LearningLab App to User**
  - Sends processed data (e.g., logs, dynamic content) back to the user via the application interface.

2. STRIDE-LM Threat Analysis

Applying STRIDE-LM Threat Analysis lead to the identification of the following threats for each component. Man-in-the-Middle threats are covered by the categories Spoofing, Tampering and Information Disclosure on data flows and are not categorized here as their own class, which aligns with academic and in practice threat analysis methodologies.

1. External Entities (Actors)

The external entity actor (E01) interacts with the system over HTTPS (UI 443). The main threats related to the actor involve impersonation, where an attacker could pretend to be a legitimate user to access the system, and repudiation, where a legitimate user could deny taking specific actions within the system.

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T001	Spoofing	E01	An attacker could impersonate a legitimate user (actor) and gain unauthorized access to the system.	Implement multi-factor authentication (MFA) and strong OAuth-based authentication.	High
T002	Repudiation	E01	A user might deny performing actions such as making repository changes or interacting with the LearningLab app.	Ensure detailed logging of user actions, using tamper-proof audit logs with timestamping and non-repudiation mechanisms.	Medium

2. (Multi) Processes

For the processes, particularly the GitHub Enterprise Process (P01) and LearningLab Application Process (P02), the threats mainly revolve around spoofing, tampering, information disclosure, denial of service, elevation of privilege, and lateral movement. Spoofing could allow attackers to impersonate system components or users. Tampering with data during the authentication or repository management processes could compromise the system's integrity. Unauthorized access could result in information disclosure, and system overload could lead to denial of service. Additionally, vulnerabilities in the OAuth process or within the NodeJS application could enable attackers to elevate their privileges and move laterally across system components.

P01 GitHub Enterprise Process

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T003	Spoofing	P01, E01	An attacker could impersonate the GitHub App or a user during the authentication process to gain unauthorized access.	Use OAuth with strict validation of tokens, and ensure the identity of the app and users is verified during every interaction.	High
T004	Tampering	P01	An attacker could tamper with the GitHub authentication or repository data during the copy/fork process.	Use strict access controls on repository actions and ensure integrity checks on repositories using digital signatures.	Medium
T005	Information Disclosure	P01, DS01, DS02	Sensitive repository or user data could be exposed during the OAuth authentication or fork process.	Ensure all sensitive data is encrypted during transit using TLS and apply strict role-based access control mechanisms.	High
T006	Denial of Service (DoS)	P01	The GitHub Enterprise system could be overwhelmed with too many repository creation or event requests, causing a DoS.	Implement rate-limiting, monitoring, and load balancing for requests to the GitHub Enterprise system.	Medium

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T007	Elevation of Privilege	P01	An attacker could exploit a vulnerability in the OAuth process to gain higher-level access to repositories or user data.	Use proper privilege separation, regularly update OAuth libraries, and perform security audits.	High
T008	Lateral Movement	P01	An attacker who gains access to a user's repository might use this access to pivot into other parts of the system, such as admin-level repositories.	Implement least privilege policies, network segmentation, and monitoring of unusual activity in the repositories.	High

P02 LearningLab Application Process

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T009	Spoofing	P02, E01	An attacker could impersonate the LearningLab application or API client to communicate with GitHub or users improperly.	Use strong API authentication, such as OAuth or JWT (JSON Web Tokens), to verify the identity of the application and requests.	High

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T010	Tampering	P02	An attacker could tamper with the LearningLab app, altering its behavior (e.g., modifying API calls or logs).	Secure the LearningLab POD with integrity checks (e.g., hash validation), and regularly update application libraries.	Medium
T011	Information Disclosure	P02	Sensitive data (e.g., logs or repository details) might be exposed through the LearningLab app or during logging.	Ensure logs are secured, encrypted, and access is restricted only to authorized personnel.	High
T012	Denial of Service (DoS)	P02	The LearningLab app could be overwhelmed with HTTP requests, causing it to slow down or become unavailable.	Implement rate-limiting, load balancing, and scalable infrastructure to mitigate DoS attacks.	Medium
T013	Elevation of Privilege	P02	A malicious user could exploit vulnerabilities in the NodeJS app to gain higher privileges within the system.	Regularly patch and audit the NodeJS app, using security plugins and best practices for privilege management.	High

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T014	Lateral Movement	P02	Once an attacker compromises the LearningLab application, they could use that access to move laterally to other components (e.g., Octokit or express.js).	Implement network segmentation between application components, monitor internal network traffic for abnormal behavior, and limit permissions to sensitive components.	High

P03 Docker

Threat Description:
Docker (P03) serves as a container environment running the NodeJS App (P04). Tampering, where an attacker could modify the Docker environment or the application running inside it, is a significant risk. Lateral movement could occur if the attacker gains access to the container, allowing them to spread to other components.

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T015	Tampering	P03, P04	An attacker could tamper with the Docker environment, modifying the NodeJS app or gaining control over the container.	Use secure Docker images and apply integrity checks on the container configuration. Regularly patch vulnerabilities.	Medium
T016	Lateral Movement	P03	If the Docker container is compromised, an attacker could move laterally to other containers or parts of the system.	Use proper container isolation, network segmentation, and monitoring of intra-container communications.	High

P04 NodeJS Application

Threat Description:
The NodeJS App (P04) manages user requests and interacts with Docker (P03), GitHub (P01), and third-party libraries (P05). The major threats include spoofing (impersonating the app to communicate with other components), tampering (modifying the app's code or its API interactions), and elevation of privilege (exploiting vulnerabilities in the NodeJS app).

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T017	Spoofing	P04, DF02	An attacker could impersonate the NodeJS app (P04) to send unauthorized requests to Docker or GitHub.	Use strong authentication methods for internal communications (e.g., JWT or OAuth).	High
T018	Tampering	P04, P03	An attacker could modify the NodeJS app's code or API calls, altering its behavior or causing it to send malicious data.	Secure the NodeJS app with integrity checks, regularly update the application, and implement API request validation.	Medium
T019	Information Disclosure	P04	Sensitive data handled by the NodeJS app, such as user data or API tokens, could be exposed if the app is compromised.	Encrypt sensitive data both at rest and in transit, and implement role-based access control (RBAC).	High

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T020	Elevation of Privilege	P04	An attacker could exploit vulnerabilities in the NodeJS app to gain higher-level access to other parts of the system.	Regularly patch vulnerabilities, limit access control to sensitive functions, and apply least-privilege policies.	High
T021	Tampering	P05	Vulnerabilities in third-party libraries could allow attackers gain access to sendistive information.	Apply container isolation, sandbox third-party libraries, and regularly scan for vulnerabilities in dependencies.	Medium
T022	Information Disclosure	P05	Vulnerabilities in third-party libraries could allow attackers gain access to sendistive information	Apply container isolation, sandbox third-party libraries, and regularly scan for vulnerabilities in dependencies.	High
T023	Lateral Movement	P05	Vulnerabilities in third-party libraries could allow attackers to move laterally and compromise other system components.	Apply container isolation, sandbox third-party libraries, and regularly scan for vulnerabilities in dependencies.	High

3. Data Stores

For the data stores, the primary concerns are tampering, information disclosure, and denial of service. If an attacker gains access to repositories, they could alter critical learning materials or user data. Sensitive information within repositories might be exposed to unauthorized users. Furthermore, excessive requests could cause these data stores to become unavailable, disrupting system operations.

DS01 Template Repository (GitHub)

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T024	Tampering	DS01	The template repository could be altered by unauthorized users, affecting future forks or copies.	Implement strict access control, using role-based permissions and audit trails for all repository actions.	Medium
T025	Information Disclosure	DS01	Sensitive learning materials in the template repository might be accessed or copied by unauthorized users.	Apply encryption for sensitive data and enforce access controls to limit repository visibility.	High
T026	Denial of Service (DoS)	DS01	Attackers could repeatedly request forks or access, causing a denial of service on the repository or GitHub infrastructure.	Implement rate-limiting on repository actions and monitoring for abnormal activity.	Medium

---

DS02 Learning Repository (User)

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T027	Tampering	DS02	The learning repository could be modified by unauthorized users, corrupting the learning environment for the user.	Implement strict access controls and use digital signatures to verify the integrity of the repository.	Medium

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T028	Information Disclosure	DS02	Unauthorized users could gain access to the user's learning repository, disclosing sensitive data or personal information.	Apply encryption and access controls to protect sensitive user data in the repository.	High
T029	Denial of Service (DoS)	DS02	Attackers could continuously make API requests or attempts to access the repository, causing it to become unavailable.	Implement rate-limiting and monitor API requests for abnormal activity to prevent DoS attacks.	Medium

DS03 User Instance

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T030	Tampering	DS03	The user instance could be altered, affecting the user's environment or the learning material they are working on.	Secure the user instance with proper authentication and regular checks for data integrity.	Medium
T031	Information Disclosure	DS03	Data specific to the user (e.g., learning progress or settings) might be exposed to unauthorized users.	Encrypt user-specific data and ensure proper access controls are in place.	High

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T032	Denial of Service (DoS)	DS03	Attackers could repeatedly request access to the user instance, overwhelming the system and preventing the user from accessing it.	Monitor access requests and implement rate-limiting to protect against DoS attacks.	Medium

4. Data Flows

The data flows involve communications between users and system components. The primary threats are tampering, information disclosure, and denial of service. Attackers could intercept and modify data in transit, expose sensitive data, or overwhelm the system with excessive requests, leading to service unavailability.

DF01: User to GitHub Enterprise (HTTPS)

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T033	Tampering	DF01, E01, P01	Data between the user and GitHub Enterprise could be intercepted and altered if HTTPS is misconfigured or compromised.	Use strong encryption protocols (e.g., TLS 1.2 or higher), ensure up-to-date certificates, and enforce HSTS.	High
T034	Information Disclosure	DF01, E01, P01	Sensitive data, like authentication tokens or repository content, could be exposed if HTTPS is not properly enforced.	Ensure that all data is encrypted during transmission, and implement strict HTTPS policies.	High

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T035	Denial of Service (DoS)	DF01, E01, P01	A flood of requests from a user or attacker could overwhelm the GitHub Enterprise, leading to denial of service.	Implement rate-limiting, monitor user behavior for anomalies, and apply auto-scaling where possible.	Medium

DF02: User to LearningLab App (HTTPS)

Threat ID	STRIDE-LM Category	Involved DFD Elements	Threat Description	Mitigation	Risk Rating
T036	Tampering	DF02, E01, P02	An attacker could intercept and modify data sent between the user and the LearningLab app if HTTPS is compromised.	Use strong encryption protocols (TLS 1.2+), enforce secure communication through HTTPS, and apply HSTS policies.	High
T037	Information Disclosure	DF02, E01, P02	Sensitive user data, such as learning progress or repository details, could be exposed if HTTPS is not properly enforced.	Ensure proper encryption and data protection mechanisms are in place for all data transmissions.	High
T038	Denial of Service (DoS)	DF02, E01, P02	Continuous requests from a user or attacker could overwhelm the LearningLab app, resulting in service unavailability.	Implement rate-limiting and monitoring to detect and mitigate DoS attacks.	Medium

3. Security Requirements

The table below shows a list of security requirements derived from the mitigations from the threat analysis as well as the threat IDs which they cover.

Security Requirement ID	Security Requirement	Mitigated Threats
SR001	Implement multi-factor authentication (MFA) for all users and system components.	T001, T003, T009, T017
SR002	Use strong OAuth or JWT-based authentication mechanisms for system components and API interactions.	T003, T005, T009, T017
SR003	Enforce strict access control using role-based permissions and least privilege principles.	T004, T007, T008, T013, T014, T027, T028, T030, T031
SR004	Regularly audit and update OAuth libraries, APIs, and dependencies to patch vulnerabilities.	T007, T013, T020
SR005	Implement rate-limiting to prevent denial of service attacks on APIs and system components.	T006, T012, T015, T020, T029, T032, T035, T038
SR006	Use HTTPS with strong encryption protocols (TLS 1.2+), and enforce HSTS for all communications.	T011, T012, T024, T025, T028, T036, T037
SR007	Ensure sensitive data is encrypted both at rest and in transit.	T005, T019, T024, T025, T028, T031, T037
SR008	Apply integrity checks on all system components, including Docker containers and the NodeJS app.	T004, T010, T015, T018
SR009	Implement detailed logging and tamper-proof audit trails for non-repudiation.	T002
SR010	Use network segmentation to prevent lateral movement and isolate compromised components.	T008, T014, T016, T023, T014, T023
SR011	Regularly scan for vulnerabilities in third-party libraries and sandbox their access to sensitive data.	T021, T022, T023
SR012	Implement load balancing and auto-scaling to handle excessive requests and mitigate DoS attacks.	T006, T015, T029, T032, T035, T038

4. Mapping to NIST SP 800-53 Controls

The security requirements are mapped to NIST SP 800-53 Controls in the following table.

Security Requirement ID	Security Requirement	NIST SP 800-53 Controls
SR001	Implement multi-factor authentication (MFA) for all users and system components.	IA-2: Identification and Authentication (Organizational Users) IA-2(1): Multi-factor Authentication
SR002	Use strong OAuth or JWT-based authentication mechanisms for system components and API interactions.	IA-2: Identification and Authentication (Organizational Users) IA-5: Authenticator Management
SR003	Enforce strict access control using role-based permissions and least privilege principles.	AC-2: Account Management AC-6: Least Privilege AC-3: Access Enforcement
SR004	Regularly audit and update OAuth libraries, APIs, and dependencies to patch vulnerabilities.	SI-2: Flaw Remediation SI-5: Security Alerts, Advisories, and Directives
SR005	Implement rate-limiting to prevent denial of service attacks on APIs and system components.	SC-5: Denial-of-Service Protection
SR006	Use HTTPS with strong encryption protocols (TLS 1.2+), and enforce HSTS for all communications.	SC-8: Transmission Confidentiality and Integrity SC-13: Cryptographic Protection
SR007	Ensure sensitive data is encrypted both at rest and in transit.	SC-12: Cryptographic Key Establishment and Management SC-28: Protection of Information at Rest
SR008	Apply integrity checks on all system components, including Docker containers and the NodeJS app.	SI-7: Software, Firmware, and Information Integrity SA-10: Developer Security Testing and Evaluation
SR009	Implement detailed logging and tamper-proof audit trails for non-repudiation.	AU-2: Audit Events AU-3: Content of Audit Records AU-10: Non-Repudiation

Security Requirement ID	Security Requirement	NIST SP 800-53 Controls
SR010	Use network segmentation to prevent lateral movement and isolate compromised components.	SC-7: Boundary Protection AC-4: Information Flow Enforcement
SR011	Regularly scan for vulnerabilities in third-party libraries and sandbox their access to sensitive data.	SI-2: Flaw Remediation SA-5: System, Services, and Information Integrity
SR012	Implement load balancing and auto-scaling to handle excessive requests and mitigate DoS attacks.	SC-5: Denial-of-Service Protection AU-6: Audit Review, Analysis, and Reporting

5. Compliance Assurance

Below is a table with security requirements mapped to NIST SP 800-53 controls along with an explanation of how compliance with each control will be assured:

Security Requirement ID	Security Requirement	NIST SP 800-53 Controls	Compliance Methods
SR001	Implement multi-factor authentication (MFA) for all users and system components.	IA-2: Identification and Authentication (Organizational Users) IA-2(1): Multi-factor Authentication	Implement MFA for all accounts, verify via regular access control reviews and test MFA enforcement using pen testing and audits.
SR002	Use strong OAuth or JWT-based authentication mechanisms for system components and API interactions.	IA-2: Identification and Authentication (Organizational Users) IA-5: Authenticator Management	Ensure OAuth/JWT is properly implemented by testing API authentication flows. Periodically audit tokens for expiration and renewal processes.

Security Requirement ID	Security Requirement	NIST SP 800-53 Controls	Compliance Methods
SR003	Enforce strict access control using role-based permissions and least privilege principles.	AC-2: Account Management AC-6: Least Privilege AC-3: Access Enforcement	Conduct regular access reviews to ensure roles align with least privilege. Automate RBAC audits using IAM tools and access logs.
SR004	Regularly audit and update OAuth libraries, APIs, and dependencies to patch vulnerabilities.	SI-2: Flaw Remediation SI-5: Security Alerts, Advisories, and Directives	Implement automated vulnerability scanning tools, apply patches as per advisories, and maintain an inventory of dependencies for regular updates.
SR005	Implement rate-limiting to prevent denial of service attacks on APIs and system components.	SC-5: Denial-of-Service Protection	Configure rate-limiting in API gateways and verify through stress testing and security tools simulating DoS attacks.
SR006	Use HTTPS with strong encryption protocols (TLS 1.2+), and enforce HSTS for all communications.	SC-8: Transmission Confidentiality and Integrity SC-13: Cryptographic Protection	Ensure TLS 1.2+ is enforced via configuration management tools (e.g., Terraform, Ansible). Validate through regular penetration tests and SSL/TLS audits.
SR007	Ensure sensitive data is encrypted both at rest and in transit.	SC-12: Cryptographic Key Establishment and Management SC-28: Protection of Information at Rest	Validate encryption is enabled for data at rest (e.g., database encryption) and in transit (e.g., network traffic scans). Regularly rotate encryption keys.
SR008	Apply integrity checks on all system components, including Docker containers and the NodeJS app.	SI-7: Software, Firmware, and Information Integrity SA-10: Developer Security Testing and Evaluation	Use code signing and image scanning tools to validate the integrity of application containers and system binaries. Conduct regular static and dynamic code analysis.

Security Requirement ID	Security Requirement	NIST SP 800-53 Controls	Compliance Methods
SR009	Implement detailed logging and tamper-proof audit trails for non-repudiation.	AU-2: Audit Events AU-3: Content of Audit Records AU-10: Non-Repudiation	Ensure audit logs are immutable (using append-only storage). Use automated log collection and review mechanisms, and conduct periodic compliance audits.
SR010	Use network segmentation to prevent lateral movement and isolate compromised components.	SC-7: Boundary Protection AC-4: Information Flow Enforcement	Enforce network segmentation policies (e.g., VLANs or VPCs). Conduct regular penetration tests to validate segmentation and containment of compromised components.
SR011	Regularly scan for vulnerabilities in third-party libraries and sandbox their access to sensitive data.	SI-2: Flaw Remediation SA-5: System, Services, and Information Integrity	Implement automated dependency and vulnerability scanners. Regularly sandbox third-party libraries and test using static code analysis.
SR012	Implement load balancing and auto-scaling to handle excessive requests and mitigate DoS attacks.	SC-5: Denial-of-Service Protection AU-6: Audit Review, Analysis, and Reporting	Implement load balancing and auto-scaling for critical components, and test the system's resilience under simulated DoS conditions. Regularly monitor system logs for anomalies.