# LP Threat Modeling Outline

**Use the LP Threat Modeling Template to fill out the following information**

1. **Define the Scope and Application Objectives**
    - Define the Scope
        - To what extent the threat model will cover
        - A threat model scope defines what is relevant, within a particular view
            - There can be multiple scopes, or views, that pertain to a single application
            - Typically there should be two scopes: Application and Infrastructure
                - The application scope is the use case / application's functionality
                - The infrastructure scope consists of how the application is compiled, tested, and stood up
    - Define the Purpose of the Application
        - What functions the application provides
    - Define Business Objectives of the Application
        - How this application benefits the business
    - Define the Application's Security Tier
        - Tiers range from 1 to 3, with Tier 1 being the minimum that all software must adhere to
        - This tiering system is derrived from OWASP's Application Security Verification Standard v3.0.1, pages 8-12
            - Reference the ASVS for more detail on the following tiers and a checklist of requirements for each level
        - Tier 1 applies to general software
            - Coincides with ASVS Level 1
            - The software must adequately defend against application security vulnerabilities which are easy to discover
                - Refer to OWASP Top 10 and other similar checklists

        - Tier 2 applies to applications that contain sensitive data or controls
            - Coincides with ASVS Level 2
            - The software must have effective security and monitoring controls in place, which are used within the application
            - The software must adequately defend against most risks associated with software to date
        - Tier 3 applies to applications that perform critical functions, where failure could significantly impact the organization's operations and/or survivability
            - Coincides with ASVS Level 3
            - The software must adequately defend against advanced application security vulnerabilities
            - The software must demonstrate principles of good security design
    - Define Compliance Requirements
2. **Decompose the Application**
    - Draw/Update Control Flow Diagrams using the Diagram Syntax
        - Application Threat Model Diagram (Example ATMD)
            - Diagram of how the app works in production
            - Use case diagram / data flow diagram
        - Infrastructure Threat Model Diagram (Example ITMD)
            - Diagram of the system infrastructure and how the app is deployed
    - List user roles and their permissions
        - This is a list of what actions each role *should* be able to do
3. **Identify Assets**
    - Identify Assets
        - Items/areas of interest to an attacker
        - The things that this application needs to secure
    - Identify External Dependencies
        - external entities that would keep the application from functioning properly if removed
4. **Identify Threats and Analyize Risk**
    i. Create a Threat Traceability Matrix
        - The simplest way to create this is to use Excel and save the matrix as a CSV file, then throw the CSV into a markdown table generator

        - Look at each asset individually while considering which categories of STRIDE may apply
            - Spoofing
            - Tampering
            - Repudiation
            - Information Disclosure
            - Denial of Service
            - Elevation of Privilege
        - Play the Elevation of Privilege card game
        - Each threat should be listed as a new row in the Threat Matrix, which has the following columns
            - STRIDE Category
            - Interaction (Connection details between the user and the application using the Interaction Syntax)
            - Exploit Description (The type of attack performed by the malicious user)
            - Attack Surface (All of the different points where an attacker could get in and get data out)
            - Impact (What is the impact if this is exploited)
            - Mitigation (How can we stop or prevent this threat)
    ii. Reorder the threat traceability matrix from highest to lowest risk
    iii. Security Tier 1 applications
    - Rank threats relative to each other to get a comparative list
    iv. Security Tier 2 applications
    - Rank threats numerically with the 5x5 Risk Matrix
    - Likelihood Ranking Definitions
    - **Rare**: it may not happen in a lifetime
    - **Unlikely**: this will occur every so often
    - **Moderate**: it will happen regularly
    - **Likely**: this will be a frequent problem
    - **Almost Certain**: it will happen constantly
    - Consequence Ranking Definitions
    - **Insignificant**: slight, possibly unnoticeable impact. Business unaffected, maybe a phone call is needed
    - **Minor**: temporary loss of service that does not affect the customer. Business experiences a hiccup in operations
    - **Significant**: loss of sensitive data or service for an extended

amount of time, possible financial loss
- **Major**: major loss of sensitive data or an interuption of critical services, business will experience financial loss
- **Severe**: complete destruction or theft of data and rendering a service unrecoverable. Extreme impact to the business and it's survivability

v. Security Tier 3 applications
- Rank threats using the CVSS
- Online CVSS Calculator

# Assessing the Threat Model

**How good is the threat model?**

How to assess the threat model's effectiveness based on the Application's Security Tier:
*Tier 1 Application*
*- General reflection on the artifacts generated*
Tier 2 Application
- General reflection like in Tier 1 with additional self-performed penetration tests
* Tier 3 Application
- Professional penetration test

# Threat Modeling Syntax Information

### Interaction Syntax

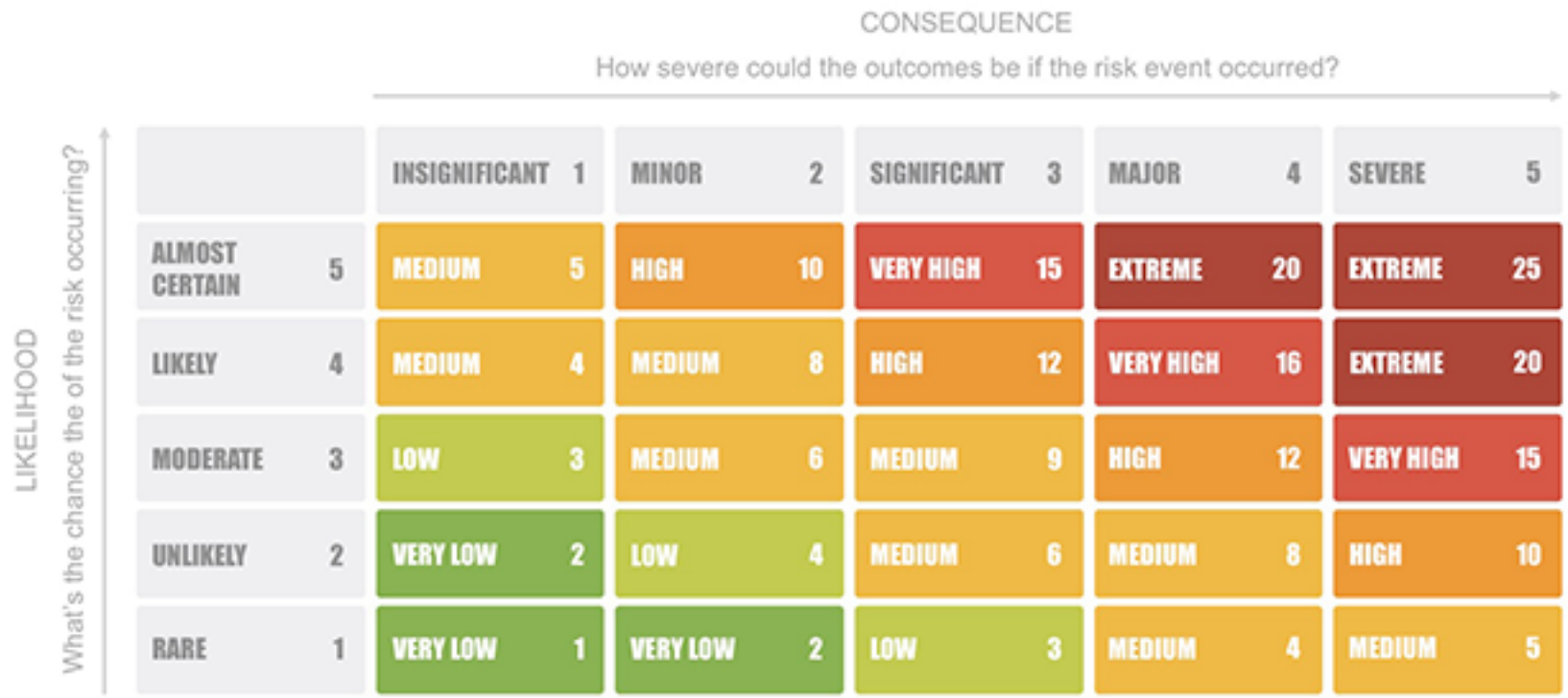
- <-->
  - designates a physical communication connection
- <-(Service Here)->
  - You may add detail as to the service used with parenthesis inside of the connection arrows
  - **Ex:** User <-(nginx)-> Web App
- Node [Action]
  - Use square brackets to specify specific actions or routes used on a node for that connection
  - **Ex:** User <--> Rails API [Log in] <--> Credential Database
- Node [Action (Detail)]
  - Use parenthesis inside of the square brackets to provide further detail
  - **Ex:** Admin User <--> Web App [Admin Console (Manage Project Access)] <--> Database
- Server One/Server Two
  - Use forward slash to list multiple nodes using the same connection route
  - **Ex:** Demo/Staging/Production Servers <--> GitLab [Project Repo]

### Diagram Syntax

- Each diagram must contain a key that holds all symbol information
- Colors are optional, but meaningful color differences should be defined by the diagram key
- Solid rectangles with sharp corners
  - Designates object grouping or server boundaries
  - Must be named with a label in a corner
- Solid rectangles with rounded corners
  - Designates an action or service
  - Must be named with a label in the center
- Solid outlined cloud
  - Represents a group of servers
  - Must be named with a label in the center
- Dashed rectangles
  - Used to designate network or authentication boundaries
  - Surrounds actions or objects that require a certain privilege
  - Must be named with a label in the corner
- Closed and filled in arrows
  - Represents the initial direction of data flow in a connection
- Open/hollow arrows
  - Represents the return direction of data flow, in connections that return a response
    - This does not include acknowledgement packets or their equivalent

# Risk Matrix 5x5

Enter subtitle information text



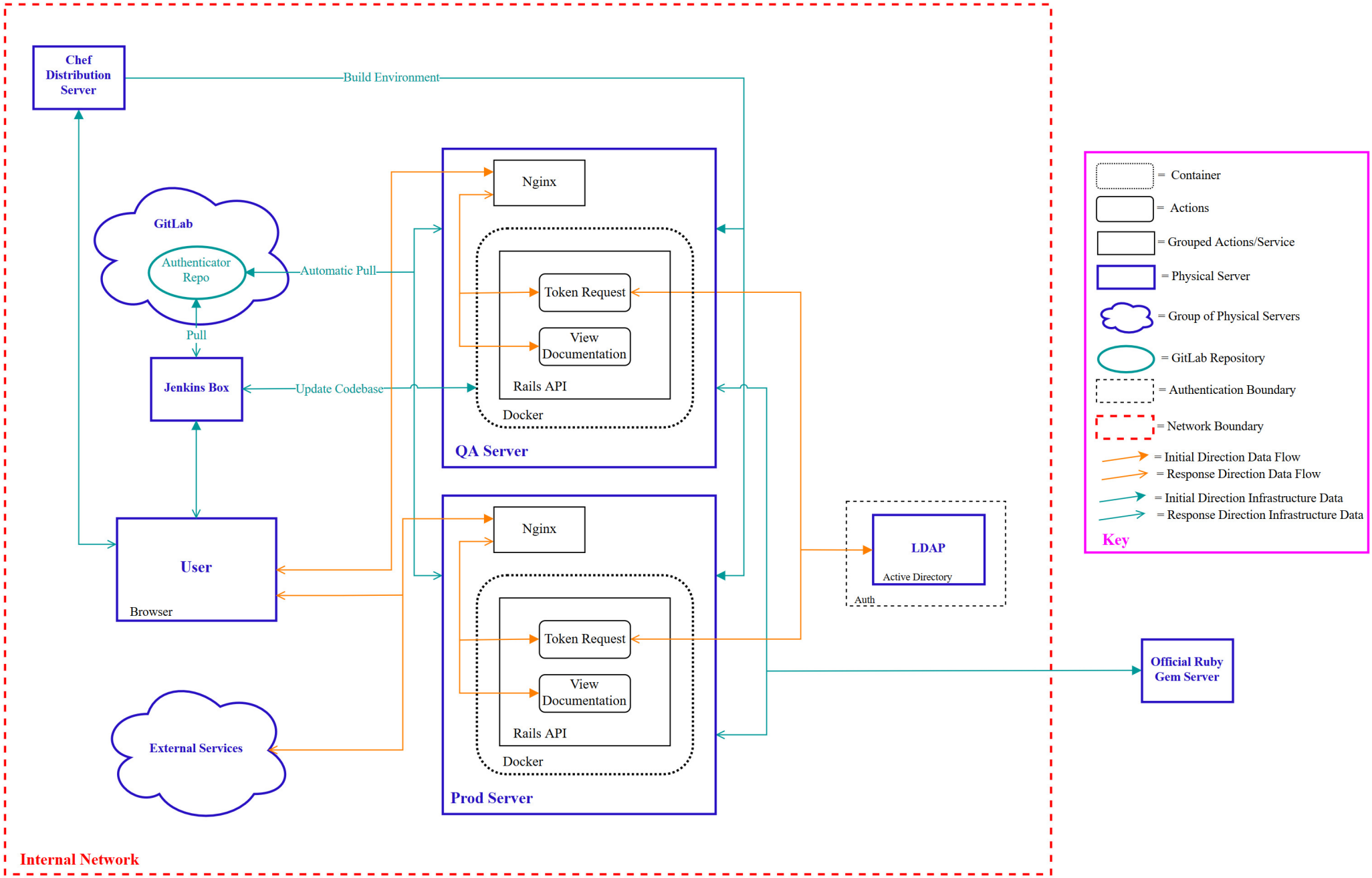

| | INSIGNIFICANT 1 | MINOR 2 | SIGNIFICANT 3 | MAJOR 4 | SEVERE 5 |
|---|---|---|---|---|---|
| ALMOST CERTAIN 5 | MEDIUM 5 | HIGH 10 | VERY HIGH 15 | EXTREME 20 | EXTREME 25 |
| LIKELY 4 | MEDIUM 4 | MEDIUM 8 | HIGH 12 | VERY HIGH 16 | EXTREME 20 |
| MODERATE 3 | LOW 3 | MEDIUM 6 | MEDIUM 9 | HIGH 12 | VERY HIGH 15 |
| UNLIKELY 2 | VERY LOW 2 | LOW 4 | MEDIUM 6 | MEDIUM 8 | HIGH 10 |
| RARE 1 | VERY LOW 1 | VERY LOW 2 | LOW 3 | MEDIUM 4 | MEDIUM 5 |

# Threat Model Information

**Application:** Authenticator

**Version:** 1.0.0
**Commit SHA** e8102fb5

1. **Define the Scope and Application Objectives**
    - Define the Scope of Threat Model
        - Production Application in its current state
        - Application deployment cycle and infrastructure
    - Define the Purpose of the Application
        - The application serves as an authentication API for internal applications
    - Define the Business Objectives of the Application
        - Create a standard and secure way internal applications can authenticate using Active Directory
    - Define the Application's Security Tier
        - Tier 2
    - Compliance Requirements
        - None
2. **Decompose the Application**
    - Draw/Update Control Flow Diagrams
        - Threat Model Diagram
            - Includes both the Application and Infrastructure Diagram
    - Identify roles and permissions
        - Unathenticated user
            - Can send requests and receive tokens
3. **Identify Assets**
    - Assets (items/areas of interest to an attacker)
        - LDAP (AD) credentials
        - Rails API
        - nginx
    - External dependencies
        - Active Directory
4. **Identify Threats and Analyize Risk**


Chef Distribution Server
Build Environment
GitLab
Authenticator Repo
Automatic Pull
Pull
Jenkins Box
Update Codebase
Nginx
Token Request
View Documentation
Rails API
Docker
QA Server
User
Browser
Nginx
Token Request
View Documentation
Rails API
Docker
Prod Server
External Services
LDAP
Active Directory
Auth
Official Ruby Gem Server
Internal Network
= Container
= Actions
= Grouped Actions/Service
= Physical Server
= Group of Physical Servers
= GitLab Repository
= Authentication Boundary
= Network Boundary
= Initial Direction Data Flow
= Response Direction Data Flow
= Initial Direction Infrastructure Data
= Response Direction Infrastructure Data
Key

| STRIDE | Interaction | Exploitation Description | Attack Surface | Impact | Mitigation | 5x5 Ranking |
|---|---|---|---|---|---|---|
| Spoofing & Elevation of Privilege | Authenticator Server <--> An attacker <--> Active Directory | "Since Authenticator doesn't validate the Active Directory server, an attacker could spoof Active Directory and place himself as a man in the middle of the connection. This would give him access to any credentials sent over the wire, in plain text" | The unvalidated connection between Authenticator and Active Directory | Unauthorized access to any resources the credentials have privileges to | Validate the certificate for Active Directory | Very High 15 |
| Tampering & Spoofing | Any user <--> Rails API [Token Request] <--> Active Directory | Input is not sanitized allowing any user to insert LDAP special characters and perform limited LDAP injection | Rails API [Token Request] | Information disclosure or unauthorized account access | Sanitize input before querying the LDAP server | High 12 |
| Elevation of Privilege | Any user on the internal network <--> Rails API Service | "Any user can send as many api requests as they want, meaning that they can brute force passwords" | "Attack surface is port 80, 443, 1337" | Potential leak of user credentials | Limit the number of consecutive attemps to authenticate | High 12 |
| Information Disclosure | An attacker <--> Authenticator Server | An attacker could gain access to the Authenticator Server which would allow them to view the secret used for JWT login token encryption. This would allow privileged access to any application that uses Authenticator during the login process | The Authenticator Server or the project repository | Unauthorized access to confidential company information and impersonation of users | Purge secrets from the repository and use a secure way to store them | High 10 |
| Information Disclosure | An unauthenticated user <--> Rails API [Log in] | The attacker could use the time difference between a valid user log in and invalid user login to brute force usernames | Rails API login route | Information disclosure and ability to enumerate user accounts | On authentication make sure that both valid and invalid log in attempts take the same amount of time to complete | Medium 9 |
| Repudiation | Any interaction with Authenticator | No logging. Lack of repudiation and no way to track authentication attempts | All user interactions | Difficult to detect malicious actions | Implement a logging system | Medium 6 |
| Repudiation | Any user on the internal network <-(Direct)-> Rails API Service | Direct connection to web API to bypass nginx and HTTPS encryption | Attack surface is port 1337 | Loss of encryption between user and API | Close port 1337 | Low 4 |
| Information Disclosure & Elevation of Privileges | N/A (Architectural Problem) | Authenticator uses symmetric encryption for the JWT authentication tokens | Any server or storage device that houses the symmetric key | Unauthorized access to confidential company information and impersonation of users | Asymmetric encryption should be implemented | Low 3 |
| Denial of Service | Any user on the internal network <--> Rails API Service | Denial of service by flooding authentication requests | "Attack surface is port 80, 443, 1337" | Major services that depend on Authenticator for authentication could not be used | Limit the number of consecutive attemps to authenticate and close port 1337 | Low 3 |
| Denial of Service | User (Spoofing a Target) <--> Rails API Service | Reflective DoS attack by spoofing the senders IP address | Attack surface is port 1337 | Unathorized use of resources and slowing down authrization for real users | Close port 1337 | Very Low 2 |