LaunchDarkly

LaunchDarkly Security Overview

What you need to know
to integrate LaunchDarkly
securely into your organization

LaunchDarkly Security Overview

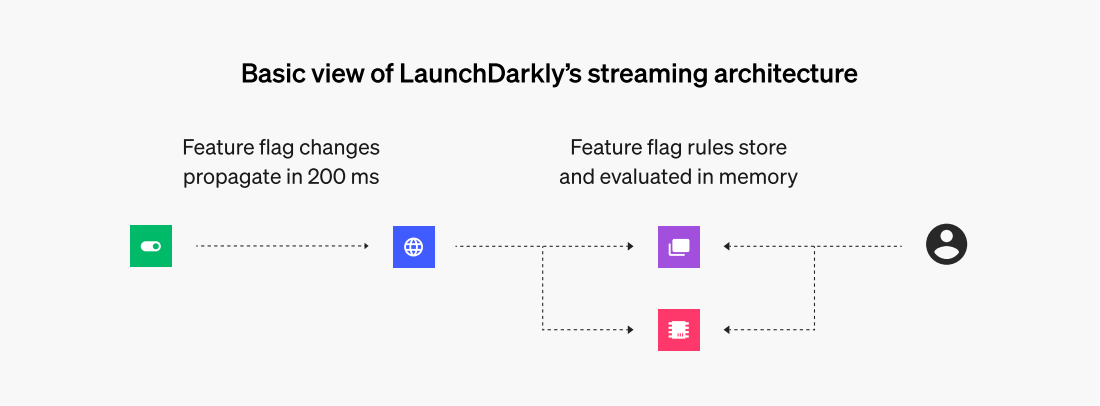
LaunchDarkly is an industry-leading SaaS product for feature management. We empower software teams to deploy code with less risk, which enables faster innovation. LaunchDarkly supports the rigorous security and privacy requirements of customers from a broad set of industries and geographies.

The paper outlines what you need to know to integrate LaunchDarkly securely into your organization. It includes:

- ✓ Architectural considerations for security
- ✓ Minimizing user data sent to LaunchDarkly
- ✓ Best practices for secure operation

Architectural considerations for security

How LaunchDarkly integrates with your application



To use LaunchDarkly, you must embed LaunchDarkly Software Development Kits (SDKs) in your applications to enable feature evaluations at runtime. Your developers wrap the code path for a feature with a call to the LaunchDarkly SDK to evaluate a feature flag. You can then manage the feature flag status and create rulesets for the status based on contextual data, like the current user, device details, or any other attributes required by your business. Flag statuses and rulesets can be changed in the LaunchDarkly UI or via a REST API. The resulting status is used to modify application behavior, like turning a feature on or off.

When a flag is updated in the UI or API, the new data about that specific change is streamed to each SDK for in-memory storage using a one-way, server-sent connection. These updates propagate to all SDKs in 200ms or less, ensuring your end-users always have the correct, up-to-date experience.

Client-side vs. server-side SDKs

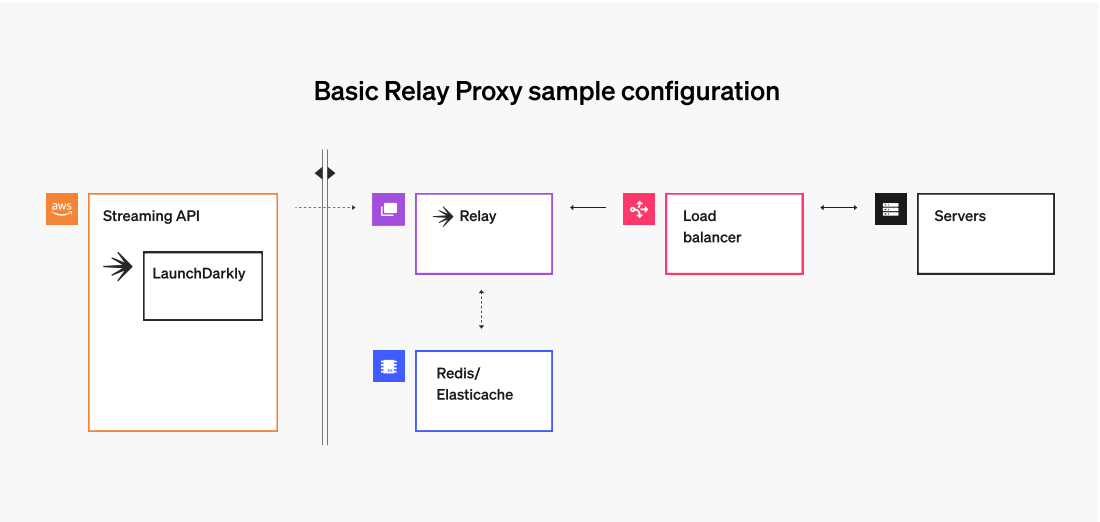
LaunchDarkly has two different SDK architectures you can use—client-side and server-side. Client-side SDKs have different security properties from server-side SDKs:

- During initialization, LaunchDarkly server-side SDKs fetch all feature flags and targeting rules from LaunchDarkly and store them locally, in memory. This means that your application can enable a feature immediately without making any external polling calls back to LaunchDarkly. For security reasons, client-side SDKs cannot download and store an entire ruleset. Client-side SDKs typically run on end users' devices, so they are technically capable of viewing and modifying all the data sent to them. Instead of storing potentially sensitive data, the client-side SDKs confirm and update flag rules by communicating with LaunchDarkly servers through streaming connections or with REST API requests.
- By default, client-side SDKs aren't authenticated. Because of this, one user could use another user's account to evaluate flags not meant for them. To authenticate user data, you can enable the SDK's secure mode, which requires you to pass a server-generated hash along with your user data. To learn more, see the secure mode documentation.
- Client-side SDKs send context data in the URL as a GET query parameter. If you are concerned about that data being stored in logs or by intermediary proxies, you can use the useReport setting to use the HTTP REPORT verb. This sends the context data in the request body, rather than in the header.
- You must enable each flag that you want client-side SDKs to be able to access using the "Make this flag available to client-side SDKs" setting or configure the flag defaults setting within your project's configuration screen.

To learn more, see the Client-side and server-side SDKs documentation.

Relay Proxy

The Relay Proxy helps minimize the number of direct connections to LaunchDarkly. Servers can connect directly to hosts in your own network, instead of connecting directly to LaunchDarkly's streaming API.



The LaunchDarkly Relay Proxy is an open source service written in Go, supported by LaunchDarkly, and available in a GitHub repository. It can run anywhere Go can run in binary form. The Relay Proxy is also provided as a Docker container and available in DockerHub.

The Relay Proxy is geared towards server-side SDKs, but also provides mobile and client-side evaluation endpoints. This means that you can initialize a client-side SDK directly against the Relay Proxy instead of connecting it directly to LaunchDarkly. This may make sense if you don't want to allow LaunchDarkly traffic from your clients, or if you want to avoid sending Private Context Attributes (see below) to LaunchDarkly.

> To learn more, see the Relay Proxy documentation or view our blog post.

Offline Mode

Enabling offline mode on the Relay Proxy lets you run without ever connecting directly to LaunchDarkly. Instead of retrieving flag and segment values from LaunchDarkly's servers, the Relay Proxy gets them from files located on your localhost or filesystem. This allows you to run your application in an isolated environment, such as a FedRAMP High, while taking advantage of LaunchDarkly's feature management solution. However, it requires that you orchestrate the transfer of flag settings to the local file system each time their value changes.

> **To learn more, see the Offline Mode documentation.**

Minimizing data sent to LaunchDarkly

Depending on your organization's security and privacy requirements, you may want to consider what data you send to LaunchDarkly.

What is Context Data?

Context data is configurable attributes your application sends to LaunchDarkly to use in configuring targeting rules for feature releases. This could be personal information about your customers or users, device details, location details, or any other configurable properties.

You configure the LaunchDarkly SDK to collect and transmit attributes about these properties to LaunchDarkly for the purpose of flag targeting. When you evaluate a feature flag in your SDK, the evaluation includes a key associated with an object. The object is the context data. It can include various key-value pairs that contain information about the context of your feature evaluation.

Context data can include personally identifiable information (PII), including names, email addresses, or other unique identifiers, depending on how you define your context. This data can be business-critical information and can present significant risk if exposed to unauthorized parties.

Every organization collects data with different risk characteristics. You may want to avoid sending sensitive user data to LaunchDarkly, although then you can't target flags based on this information. However, you can target based on information not sent to LaunchDarkly by using Private Context Attributes, described below.

Private attributes

You can use the private attribute feature to restrict the context data your service sends to LaunchDarkly while still using that data for flag targeting. You can make all attributes private, choose specific attributes to make private, or make attributes private for specific contexts.

Using the private attributes feature with server-side SDKs ensures that no data ever leaves your server. Since all feature flag evaluation occurs locally in memory, there is no need to send context data to LaunchDarkly.

For client-side SDKs, private attributes must still be sent to (but not stored by) LaunchDarkly because feature flag evaluation occurs at endpoints hosted by LaunchDarkly. You can consider using the Relay Proxy, as mentioned above, to use client SDKs and private attributes without sending the data to LaunchDarkly.

Note that when using the private attributes feature on either the server-side or client-side SDK for user contexts, the user ID will always be sent to LaunchDarkly. For this reason, we recommend the user ID be a GUID, hash, or some other non-identifiable piece of data.

For more information, see the using private context attributes documentation.

Anonymous users

Anonymous users do not register as users in your Context list, and so the usual data LaunchDarkly collects on a user isn't available for an anonymous user. You can use anonymous users to hide personally identifiable information (PII), but we recommend using private user attributes instead. You can force all users to register in LaunchDarkly as anonymous users by setting the anonymous user bit in your SDK to true when you construct the LaunchDarkly User object from the domain user object.

If you use anonymous users or private context attributes, the Context list won't populate with a complete list of users who access LaunchDarkly, and autocomplete for private attributes won't function in LaunchDarkly.

> **To learn more, see the Anonymous contexts documentation.**

Removing PII from LaunchDarkly

In the event that you need to delete PII sent to LaunchDarkly as context data, you should delete the affected data in one of two ways:

- The Context lists in the User Interface (UI)
- Calling the delete context instance or delete user API endpoints

Best practices for secure operation

Account security

LaunchDarkly has a number of features that support account security best practices. By default, you use a username and password to sign into LaunchDarkly. We support TOTP-based multi-factor authentication (MFA) and allow you to configure your account to require MFA for all new members.

LaunchDarkly can also be integrated for single sign-on (SSO) using SAML. We support ADFS, Azure, Google Apps, Okta, and OneLogin. We also support SCIM for user provisioning.

Other notable account security features include:

- You can create API access tokens to access LaunchDarkly's REST API
- You can configure the web session duration and revoke active sessions
- You can enable enhanced support to grant LaunchDarkly support teams access to your account for troubleshooting purposes
- You can connect external applications to your LaunchDarkly account using integrations, OAuth and webhooks

Authorization

LaunchDarkly uses a Role-Based Access Control (RBAC) model to control access to resources. By default, every account member in your LaunchDarkly project gets the Reader role, which can view user and flag information, including targeting data. The other built-in roles are Writer, Admin, Owner, and No Access role.

You can configure granular access by Project, Environment, and flags themselves, using custom roles and policies. To learn more, see the Custom roles documentation.

Teams

To simplify managing roles, LaunchDarkly supports the concept of teams. Teams are groups of your organization's members. Administrators can assign custom roles to a team and control permissions at the group level rather than individually assigning roles to members.

This allows you to map permissions in LaunchDarkly to your organizational structure. For example, you can give mobile flag permissions to the mobile team and desktop flag permissions to the desktop team, or give all organization members access to the staging environment, but only people on a particular team permissions to control flags on production.

> **To learn more, see the Teams documentation.**

Auditing

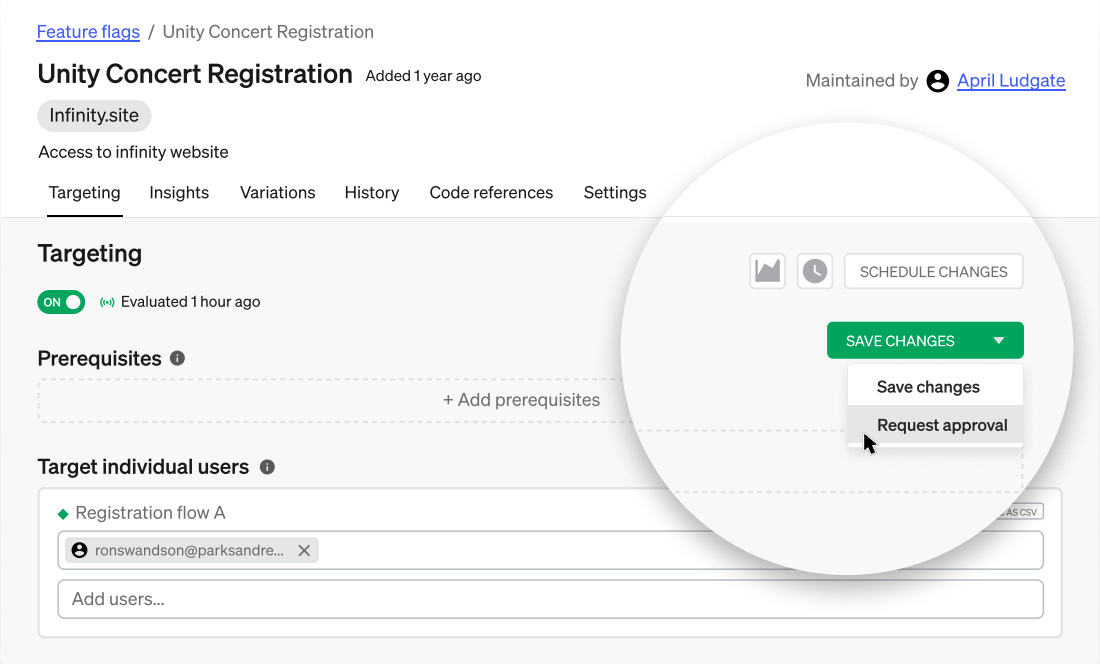
LaunchDarkly maintains audit logs of all changes to the LaunchDarkly data model. These audit logs can be viewed in the UI, exported via the API, or via integrations to Splunk, LogDNA, CloudTrail Lake, and other destinations.

You can configure granular access by Project, Environment, and flags themselves, using custom roles and policies.

> **To learn more, see the audit logs and history tabs documentation.**

Approvals

LaunchDarkly supports workflows to better map how your teams build, ship, and control software. One security-related workflow is for users to request or require approvals before making flag changes. For example, you may want to get your manager to review and sign off on the changes you are about to make in production.



LaunchDarkly also supports teams that may already have change management practices in place or use third-party tools to manage changes to their production environment for compliance purposes. Our approval workflows integrate with tools such as ServiceNow and Jira to allow users to manage requests from within them.

> To learn more, see the Approvals documentation.

Appendix

Certifications

ISO 27001 and 27701

LaunchDarkly has received certification that our Information Security Management System (ISMS) follows the ISO 27001 (Security) and 27701 (Privacy) standards. Further information can be provided with a signed NDA.

SOC 2 Type II

LaunchDarkly undergoes a regular third-party assessment of our information system controls against a set of AICPA Trust Services Criteria (TSC). A SOC 2 report can be provided upon request, with a signed NDA.

FedRAMP

LaunchDarkly's Federal instance (LaunchDarkly Federal) is listed in the FedRAMP marketplace as FedRAMP Moderate Authorized.

About LaunchDarkly

LaunchDarkly isn't just a leader in feature management — it's the first scalable feature management platform. Feature management allows development teams to innovate faster by fundamentally transforming how software is delivered to customers. With the ability to gradually release new software features to any segment of users on any platform, DevOps teams can standardize safe releases at scale, accelerate their journey to the cloud and collaborate more effectively with business teams. Today, LaunchDarkly deploys peaks of 20 trillion feature flags a day, and that number continues to grow. Founded in 2014 in Oakland, California by Edith Harbaugh and John Kodumal, LaunchDarkly has been named on the Forbes Cloud 100 list, InfoWorld's 2021 Technology of the Year list, and the Enterprise Tech 30 list. Learn more at launchdarkly.com.


LaunchDarkly