Ensuring Complete User Data Privacy with LaunchDarkly

Ensuring Complete User Data Privacy With LaunchDarkly

Table of Contents

Problem

Using the private attributes feature, LaunchDarkly server-side SDKs are able to ensure that no data ever leaves your server. Since all feature flag evaluation occurs locally in memory, there is no need to send user data to LaunchDarkly in order to take full advantage of our platform.

On the client side, this poses more of a challenge because by default feature flag evaluation occurs at endpoints hosted by LaunchDarkly.

When the LaunchDarkly Javascript SDK initializes in the default state it sends a base64 encoded user object to LaunchDarkly in order to perform an evaluation.

If you set `allAttributesPrivate` no user-specific data will be sent back to LaunchDarkly in the event stream, and no user-specific data will ever be stored in LaunchDarkly, but because the evaluation occurs against the LaunchDarkly servers the base64 encoded user object appears in any request logs since it is a part of the URL.

A sample default client side request is shown below.

```
curl
'https://app.launchdarkly.com/sdk/evalx/5baa7d07a3050e21323b735d/user
s/eyJmaXJzdE5hbWUiOiJCb2IiLCJsYXN0TmFtZSI6IkxvYmxhdyIsImtleSI6ImJvYkB
leGFtcGxlLmNvbSIsImN1c3RvbSI6eyJncm91cHMiOiJiZXRhX3Rlc3RlcnMifX0' -H
'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86
Safari/537.36' -H 'X-LaunchDarkly-User-Agent: JSClient/2.9.4' -H
'Origin: null' --compressed
```

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

Benefits of Sending User Data to LaunchDarkly

Before moving on to the proposed solutions, let's discuss why you might want to send user data to LaunchDarkly in the first place.

1. Sending user data will populate the user dashboard. This is a place where you can see all of the users that have been evaluated through a LaunchDarkly SDK, along with which attributes they had in a recent evaluation. Note that none of this data is used in future evaluations since all evaluations occur based on the current user context that exists at the time of evaluation.
2. Having data in the user dashboard will create some UI enhancements such as auto-populating user data when creating targeting rules.
3. If you utilize the data export feature, you will see user information in the stream. This helps you see "who saw what, when, and why" at a granular level. In practice, this user specific data is not necessary since as a part of the data pipeline most customers will include LaunchDarkly event data with data that they already know about their customers to perform further analysis.

Proposed Solutions

We have three proposed solutions to this problem. These solutions are our recommended best practices to ensure that no private data is ever shared with LaunchDarkly. We describe the solutions in detail below.

1. Use the `REPORT` method of the Javascript SDK.
2. Evaluate against a Relay Proxy.
3. Use Bootstrapping against a Server Side SDK.

1 Use the REPORT method of the Javascript SDK

The client-side SDK can be configured to use the `REPORT` HTTP verb instead of `GET`. In this model, although the user object is sent back to LaunchDarkly as binary data in the request body, it is never stored anywhere and would not show up in any request logs since the user object is not a part of the URL. In addition, we use end to end encryption and all communication occurs over HTTPS. This implies that there is no chance of a Man in the Middle attack unless the attacker is able to somehow break TLS encryption.

This requires an additional configuration change when the SDK initializes. A sample is shown below. A full, usable, code example is shown in the Appendix of this document.

```
      var ldclient = LDClient.initialize('5baa7d07a3050e21323b735d',
user, options = {
        allAttributesPrivate: true,
        useReport: true
      })
```

A sample client side request using `REPORT` is shown below.

```
curl
'https://app.launchdarkly.com/sdk/evalx/5baa7d07a3050e21323b735d/user
' -X REPORT -H 'Origin: null' -H 'X-LaunchDarkly-User-Agent:
JSClient/2.9.4' -H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS
X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86
Safari/537.36' -H 'Content-Type: application/json' --data-binary
'{"firstName":"Bob","lastName":"Loblaw","key":"bob@example.com","cust
om":{"groups":"beta_testers"}}' --compressed
```

Please note that when using `REPORT` the SDK operates in a "polling" mode. The SDK connects to the stream when it initializes and will receive a `ping` to keep the connection alive. Whenever a `ping` occurs, the SDK will reach back out to LaunchDarkly using the same `REPORT` request and perform a re-evaluation to get most up to date flag values for that user.

2 Evaluate against a Relay Proxy

The LaunchDarkly Relay Proxy is an open source service written in Go. It can run anywhere where Go can run in binary form. We also provide a Docker container that you can use to run the relay instance.

The relay proxy provides mobile and client-side evaluation endpoints. This means that you can initialize a client-side SDK directly against the relay instead of reaching out to LaunchDarkly.

The benefit of this approach is that the relay would run entirely within your own infrastructure, so no private data would ever need to leave the network. In addition, the SDK functions in the same way that it would function in a standard environment so you can still take advantage of our streaming API and real-time updates to any flag changes.

A sample client configuration is shown below:

```
      var ldclient = LDClient.initialize('5baa7d07a3050e21323b735d',
user, options = {
        allAttributesPrivate: true,
        baseUrl: "http://localhost:8030",
        streamUrl: "http://localhost:8030"
      });
```

Note that in the example above, you would replace `http://localhost:8030` with the URL to a running relay instance.

A configuration for the relay is shown below:

```
[environment "production"]
    prefix = "ld:support-service:production"
    sdkKey = "sdk-fdsf-579f-4101-9104-fdsfsd"
    envId = "5baa7d07a3050e21323b735d"
    allowedOrigin = "http://localhost"
```

Note that in the example above, you would replace `http://localhost` with the allowed origins of your application. You can have as many of these as you need per environment.

3 Use Bootstrapping against a server-side SDK

Similar to using a relay, you can get the initial payload for users against one of your own servers instead of reaching out to LaunchDarkly. This technique is referred to as Bootstrapping.

A sample client side configuration for this is shown below.

```
var ldclient = LDClient.initialize(
  'YOUR_CLIENT_SIDE_ID',
  user,
  options = {
    bootstrap:{{ ldclient.all_flags_state(user, {client_side_only:
true}) }}
  }
);
```

In this model, the user-specific values are acquired from a server-side SDK instance when the Javascript is being served up. The only downside to this approach is that you are not able to take advantage of real-time updates since connecting to the stream would expose the user object. The only way to get updates with this approach is to "poll" for changes against your own server, or perform a full page refresh.

Appendix

Full Javascript SDK Example using the REPORT method

You can save this file as `index.html` and run it in a local browser.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>LaunchDarkly tutorial</title>
    <script
src="https://app.launchdarkly.com/snippet/ldclient.min.js"></script>
  </head>
  <body>
    <script>
      var user = {
        "firstName": "Bob",
        "lastName": "Loblaw",
        "key": "bob@example.com",
        "custom": {
          "groups": "beta_testers"
        }
      };

      var div = document.createElement("div");
      document.body.appendChild(div);

      div.appendChild(document.createTextNode('Initializing...'));

      var ldclient = LDClient.initialize('5baa7d07a3050e21323b735d',
user, options = {
        allAttributesPrivate: true,
        useReport: true
      });

      function render() {
        var shouldShow = ldclient.variation("dark-theme", false);
        var label = (shouldShow ? "Showing" : "Not showing") + " your
feature to " + user.key;
        div.replaceChild(document.createTextNode(label),
div.firstChild);
      }

      ldclient.on('ready', render);
      ldclient.on('change', render);
    </script>
  </body>
</html>
```