The Fast and Furious
Integration Challenge

Dear Candidate,

The following exercise, AKA “home test”, is a simple integration exercise of an HTML page(s) with BioCatch JS, followed by some API requests.

The first task is the challenge itself, and it is followed by a few “bonus questions” for extra credit. It is up to you if/what to accomplish.

What will you need?

- 2-3 hours
- A simple web app OR a few HTML pages and Postman – up to you

We appreciate your time!

If you find yourself struggling with this exercise or spending more than just a few hours, don’t hesitate to contact us.

It does not necessarily mean you have failed; it only means we don't want you to waste efforts on "wrong directions" while trying to resolve.

Good Luck!

BioCatch Solutions Engineering Team

Introduction

Many BioCatch customers use our account takeover fraud detection. This solution typically includes two key components a customer is required to integrate:

1. **A client-side agent** to capture the user's behavior, device, and network data, e.g., a JS file for a website.

   This data is being sent to BioCatch every 5 seconds, typically, and each data packet is called "WUP", which stands for what's up…

   We aggregate WUPs together into a "session", based on a CSID (customer session ID) which is unique for the specific period of the user's interaction – typically, a few minutes (e.g. from login until logged out).

2. **A server-to-server 'Scoring API'** which allows the customer retrieving our risk assessment and perform other actions.

   For example, an "init" API request will associate the client-data collected with a unique user ID (UUID), based on the CSID value which must be identical on the front- and back-ends. This allows us to build "use profiles", with their session's history.

   The "getScore" API request will provide back our risk "score" and related insights. This allows our customers to make decisions (e.g. approve/decline a money transaction).

This exercise's goal is to simulate a typical "online banking" session with BioCatch integrated, i.e. client-data collection and Scoring API.

For example: Homepage → Login → Account Overview → Make Payments → Logout.

The Challenge

As explained in the introduction, the goal is to create a little website, include our JS file and functions, and trigger API requests.

Use simple HTML, CSS and JS (or your preferred framework, like ReactJS, AngularJS etc.) to create a few pages which will mimic the user's journey. The website does *not* need to have a backend to authenticate a user, database etc. Just a non-functional demo.

In the appropriate events, trigger the API requests to BioCatch.

Here's a dummy JS SDK.

Once imported in to an HTML page, it starts capturing the user's device and behavioral data (e.g. mouse movements, keyboard events etc.) and sending it to BioCatch "wupservers".

1. **Import the JS.**
   It should be loaded in all pages/screens.

2. **Set the CSID.**
   Make sure it's consistent throughout the session, until a new one should be set (e.g. on logout/login page). For example:
   ```
   cdApi.setCustomerSessionId("jgh875wdwlv0skue63");
   ```

3. **Set the a context name for each page/screen in the app.**
   For example:
   ```
   cdApi.changeContext("login_screen");
   ```

Use your browser's Developer Tools to see what happens in the Console / Network / etc.
Note - this will not work without a webserver – either use a localhost or upload the page(s) to a webserver to view online.

See the next page for the last item – the Scoring API.

4. Trigger the relevant API requests when the user clicks certain buttons, e.g., "init" when clicking the *Login* button, then "getScore" when clicking a *Make Payment* button.

   You can either trigger these API requests from the client-side, or use Postman.

An example to API *init* request:

URL: https://hooks.zapier.com/hooks/catch/1888053/bgwofce/
method: POST
payload/body:

```
{
"customerId": "dummy",
"action": "init",
"customerSessionId": "WHAT_YOU_HAVE_SET_IN_THE_CLIENT_SIDE",
"activityType": "LOGIN",
"uuid": "USE_SOME_RANDOM_STRING",
"brand": "SD",
"solution": "ATO",
"iam": "YOUR_EMAIL_ADDRESS_OR_FULL_NAME"
}
```

To trigger the *getScore* (must only happen after *init*) – replace the action's value above.

You may also want to set "MAKE_PAYMENT" or similar as the activity type…

In this test, a successful response would have HTTP status code 200.

(Sorry, you're not going to get BioCatch risk scores today:)

Ready to share the results?

- If available online, send us the link to view the web page/app (please restrict the access or delete after we review).
- Alternatively, send it as an attachment, compressed with Zip or similar, next to a some screenshots/video explaining how to load and "what's inside", i.e. the JS import, some WUPs' payloads, API requests & responses etc.

Thank You and Good Luck!

Bonus Questions

For extra credit, you may answer one or more of following:

1: SPAs

Many javascript frameworks enable the creation of single-page applications (SPA) where the HTML contents do not reload often in the browser, but the *view* shown to the user does change.

How would you implement our JS functions (e.g. *changeContext* or *setCustomerSessionId*) when there is no true reloading of the DOM but just dynamic changes of views?

2: CSPs

Banks websites can block specific resources from running in their web apps by using a Content Security Policy (CSP).

Based on what you've learned about our JS so far, write a CSP set (directives) that would allow the BioCatch JS to work in Chrome browser (assuming downloaded from the provided URL).

3: Domain names

What problems can arise when loading a JS from a different domain name than the bank's website? How could this be remedied?

4: iFrames

When an iFrame is on a page, we can usually collect the data. However, if the iFrame is from a different domain, we cannot collect data within the iFrame if the JS is only incorporated on the parent page.

What kind of architecture would you propose to integrate this?

5: In a production bank application, where do we want to trigger the API request, from the backend or from the frontend?