Behavioral Biometrics Authentication API Guide

Introduction

Use this guide to configure the SecureAuth Authentication API to use Behavioral Biometrics which analyzes an end-user’s keystroke and mouse movement behavior during login / page interaction to create a profile that is then compared to subsequent login attempts using the configured authentication workflow.

Once a profile is trained, the end-user enters the required information into the fields, and SecureAuth IdP can determine whether this behavior is consistent with that of the mature profile.

Authentication API Behavioral Biometrics Workflow

The Authentication API uses two (2) endpoints and three (3) requests for Behavioral Biometrics.

1. The first endpoint is a simple GET request to acquire the JavaScript source file required to create the behavior profile.

   Once the JavaScript source file is obtained, it must be added to the webpage(s) in order to collect the relevant data.

   As the end-user interfaces with the page(s) and form(s), SecureAuth IdP generates a profile containing particular keystroke and mouse movement behavior, actual values, and other components.

2. The second endpoint can be called either via
   - a POST request to post the end-user's behavior profile to enable the comparison

     The profile created by the JavaScript is then posted to the second endpoint for either training or authentication purposes.

     - or -

   - a PUT request to reset the end-user's behavior profile or a field in the profile to retrain keystrokes and mouse movements.

     If the profile is still being trained, then the data is used solely to mature the profile, rather than to validate the end-user's identity.

     Using the PUT endpoint, the end-user can reset the entire, trained profile, or specific fields in the trained profile.

   When a profile is trained, that behavior is expected by SecureAuth IdP; so if a field value changes (such as password, address, phone number, etc.), then resetting the profile is necessary to ensure the assessment of correct values.

   Once a profile is reset, then the end-user undergoes the same prior training procedure until the new values are matured and the profile can once more be used in authentication.
Profile Training and Score Definitions / Descriptions

Profile Training

Profile training takes ten (10) sessions, which can be completed in days, weeks, or months, depending on the frequency of user access to the form.

Once a profile has matured, it is ready for comparison to securely authenticate the user.

After the profile is posted to the API endpoint, SecureAuth IdP responds with an Accuracy Score and a Confidence Score.

Accuracy Score and Confidence Score

The Accuracy Score reveals how closely the current behavior resembles that of the trained profile, and the Confidence Score reveals the confidence of SecureAuth IdP in the provided score.

To increase accuracy and confidence, it is recommended to utilize numerous, static fields with six (6) or more characters per field.

Having a profile that is rich in behavior produces the best results and raises the confidence level of the identity verification.

Behavioral Biometrics Best Practices

To achieve the best Behavioral Biometrics results, SecureAuth recommends the following:

- Use at least six (6) characters for text fields
- After resetting a password, the profile (or profile field) should be reset to ensure accurate data collection
- The more data that is captured in the application result in more accuracy and confidence in the scores
- Since profile training takes ten (10) sessions, applications that are accessed more frequently yield faster training

Prerequisites

1. Complete the steps in the Authentication API Guide
2. Complete the Behavioral Biometrics configuration steps in the SecureAuth IdP Web Admin

LDAP directory integration is the only supported data store for the Behavioral Biometrics API

Refer to Behavioral Biometrics Guide for additional information

SecureAuth IdP Configuration Steps
Behavioral Biometrics Configuration Steps
1. In the Profile Fields section, map the Behavior Biometrics Profile Property to a directory attribute field, e.g. comment.

2. Check Writable.
Endpoints

Endpoints for /behavebio enable the use of SecureAuth IdP’s behavioral biometrics which trains the end-user profile for authentication by the sole individual based on unique keystrokes and mouse movements.

The /behavebio/js endpoint uses the GET method to retrieve the JavaScript reference that is required to gather and analyze an end-user's behavioral biometric profile.

Using the JavaScript reference, most of the data required for the POST and PUT endpoints are provided, but the remaining information (user ID, host address, and user agent) must be supplied by the application.

The /behavebio endpoint uses the POST method to collect and create the end-user's behavioral biometric profile which is then analyzed against subsequent profile information posted to the endpoint.

The API gathers the end-user keystroke, text input, and other factors while the application's provided fields are filled out (e.g. username, password, home address, phone number, etc.)

After these fields are filled out ten (10) times, the profile is considered to have been trained, and the API can then return a score based on the comparison of the "normal" behavior and the current behavior.

The /behavebio endpoint uses the PUT method to reset the end-user's profile to enable retraining – this is especially useful after the end-user changes a password.

Resetting a profile can involve completely resetting the profile, or resetting specific fields from which behavior profile information is collected.

GET

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/api/v1/behavebio/js</td>
<td><a href="https://secureauth.company.com/secureauth2/api/v1/behavebio/js">https://secureauth.company.com/secureauth2/api/v1/behavebio/js</a></td>
</tr>
</tbody>
</table>

Response to Request Example

```json
{
    "src": "https://SecureAuthIdPFQDN/SecureAuthIdPRealm/assets/scripts/api/behaveBio.obf.js?ver=9.0.0.22"
}
```
The code example is as follows:

```html
<form>
  ...  
  <input id="password" class="form-control" type="password" name="loginForm:Password" autocomplete="off" value="" placeholder="Password">
  <input type="hidden" name="behavio_hidden" id="behavio_hidden">
  <input id="textbox" class="form-control" type="text" name="loginForm:NotMonitored" autocomplete="off" value="" placeholder="Ignored">
  ...  
</form>

<script>
$(document).ready(function() {
    secureAuth.api.behaveBio({
        anonymousByName: [],
        anonymousById: [],
        anonymousByType: ['password'],
        behavioHiddenId: ['behavio_hidden'],
        ignoreFields: ['loginForm:NotMonitored'],
        haveMouse: false
    });
});
</script>
```
Code Explanation

Form

- `<input type="hidden" name="behavio_hidden" id="behavio_hidden">` – Required anywhere in the `<form>` to create the behaviorProfile, with any preferred name and id values
- `<input id="password" class="form-control" type="password" name="loginForm:Password" autocomplete="off" value="" placeholder="Password">` – Example of input field with necessary components id, type, and name to identify anonymoustext fields
- `<input id="textbox" class="form-control" type="text" name="loginForm:NotMonitored" autocomplete="off" value="" placeholder="Not for Profile">` – Example of input field with necessary components id, type, and name to identify ignored fields

Script

- Full script required in app code
- `anonymousByName` – Identifies anonymoustext fields by name (in `<input>`) 
- `anonymousById` – Identifies anonymoustext fields by id (in `<input>`) 
- `anonymousByType` – Identifies anonymoustext fields by type (in `<input>`, as shown in code example) 
- `behavioHiddenId` – Calls to `<input type="hidden" ...>` line (required in `<form>`) with the same id value 
- `ignoreFields` – Identifies fields to not include in the behaviorProfile by name (in `<input>`, as shown in code example) 
- `haveMouse` – Monitors mouse movements as well as keystroke behavior (true or false)

POST

/behavio

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/api/v1/behavio</td>
<td><a href="https://secureauth.company.com/secureauth2/api/v1/behavio">https://secureauth.company.com/secureauth2/api/v1/behavio</a></td>
</tr>
</tbody>
</table>
### Definitions

- **userID**: End-user's username
- **behaviorProfile**: End-user's behavioral biometric profile, generated by the JavaScript reference acquired in the `ehavebio/js` endpoint
- **hostAddress**: End-user's IP address
- **userAgent**: End-user's user agent (computer / browser information)
- **TotalScore**: Similarities between the presented behavior and the stored profile
- **TotalConfidence**: Level of assurance by the API that the score is accurate
- **Device**: Type of device the end-user employs
- **Results**: Breakdown of specific field scores

### POST Endpoint JSON Parameters and Response Examples

<table>
<thead>
<tr>
<th>JSON Parameters</th>
<th>Success Response</th>
<th>Failure / Error Response</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{</td>
<td>Untrained Profile:</td>
<td></td>
</tr>
<tr>
<td>&quot;userId&quot;:&quot;&lt;USERNAME&gt;&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;behaviorProfile&quot;:'&lt;BEHAVIORAL BIOMETRIC PROFILE&gt;'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;hostAddress&quot;:&quot;&lt;IP ADDRESS&gt;&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;userAgent&quot;:&quot;&lt;USER AGENT&gt;&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;userId&quot;:&quot;jsmith&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;behaviorProfile&quot;:'{&quot;m&quot;,&quot;n&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;mozPay&quot;:null,&quot;mozContacts&quot;:{},&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mozApps&quot;:{},&quot;doNotTrack&quot;:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;unspecified&quot;,&quot;battery&quot;:{},&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>oscpu&quot;:&quot;Intel+Mac+OS+X+10.10&quot;,&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vendor&quot;:&quot;,&quot;vendorSub&quot;:&quot;&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;productSub&quot;:&quot;20100101&quot;,&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cookieEnabled&quot;:true,&quot;buildID&quot;:&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20160210153822&quot;,&quot;mediaDevices&quot;:[],&quot;serviceWorker&quot;:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{&quot;controller&quot;:null,&quot;geolocation&quot;:{},&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>appCodeName&quot;:&quot;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Trained Profile:

```
{
  "status": "invalid",
  "message": "Request validation failed with: behaviorProfile was not present."
}
```

```
{
  "status": "invalid",
  "message": "Request validation failed with: hostAddress was not present."
}
```
PUT

/behavbio

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT</td>
<td>/api/v1/behavbio</td>
<td><a href="https://secureauth.company.com/secureauth2/api/v1/behavbio">https://secureauth.company.com/secureauth2/api/v1/behavbio</a></td>
</tr>
</tbody>
</table>

Definitions

- **fieldName**: Name of field to reset (unique to application); or set to ALL for global reset
- **fieldType**: Type of field, either **regulartext** (actual values stored in profile) or **anonymoustext** (no actual values stored in profile, e.g. password entries); or set to ALL for global reset
- **deviceType**: Type of device used by user (Desktop or Mobile); or set to ALL for global reset
## PUT Endpoint JSON Parameters and Response Examples

<table>
<thead>
<tr>
<th>JSON Parameters</th>
<th>Success Response</th>
<th>Failure / Error Response</th>
</tr>
</thead>
</table>
| `{  
  "userId": "<USERNAME>",
  "fieldName": "<FIELD TO RESET>",
  "fieldType": "<TYPE OF FIELD>",
  "deviceType": "<TYPE OF DEVICE>"
} | `{  
  "status": "valid",
  "message": "Reset sent to data store"
} | `{  
  "status": "invalid",
  "message": "Request validation failed with: userId was not present."
} |

### Specific Field Reset Example:

```
{
  userId:"jsmith",
  fieldName:"UserID",
  fieldType:"regulartext",
  deviceType:"DESKTOP"
}
```

### Global Reset Example:

```
{
  userId:"jsmith",
  fieldName:"ALL",
  fieldType:"ALL",
  deviceType:"ALL"
}
```