Implementation Guide

Flag sensitive transactions for step-up authentication

Background

Once a user has signed in, there may be parts of your platform you still want to safeguard with additional layers of security and access control.

Sensitive resources are often safeguarded with step-up authentication.

Step-up authentication involves invoking an additional authentication factor, where you, the developer, have flagged it necessary to do so.

Preventing unauthorized visibility or transaction activity for granular use cases often requires complex logic in order to work in harmony with your existing identity solution.

Actions with Okta CIC makes it easy for organizations to implement step-up authentication, and we have the template to prove it.

As part of our extensibility framework, Actions are a drag-and-drop pro-code/no-code logic that you can customize for your own applications and integrations that start with Identity.

Actions lets you add code to vital points in the authentication pipeline with just javascript — and 2M+ npm modules at your disposal.

Actions templates teach you how to harness the power of Actions, and get to market faster than the competition, addressing common use cases that are vital for organizations today.

Learn how step-up authentication works, with an Actions template and a web application

Step-up authentication is all about understanding the authentication context of the user.

Assuming you have one-click enabled MFA, let's walk through what this template teaches us about step-up authentication.

```
exports.onExecutePostLogin = async (event, api) => {
// confirm if MFA is enabled. For more information on client config,
refer to
// https://auth0.com/docs/secure/multi-factor-authentication/
step-up-authentication/configure-step-up-authentication-for-web-
apps#configure-app
const isMfa = event.transaction?.acr_values.includes(
'http://schemas.openid.net/pape/policies/2007/06/multi-factor'
);

let authMethods = [];
if (event.authentication && Array.isArray(event.authentication.
methods)) {
authMethods = event.authentication.methods;
}

if (isMfa && !authMethods.some((method) => method.name === 'mfa')) {
api.multifactor.enable('any', { allowRememberBrowser: false });
}
};
```

There are two states to consider when building for a step-up context: what method was used to authenticate with the identity provider before, and what a service request should require to authenticate now.

`event.authentication`	What authentication signals were obtained during login.
`event.authentication.methods`	What method was used at login.
In our action template, we represent the initial login context by `authMethods`.	

Once we understand what, if any, MFA method was used to login, then we can build context to step-up authentication by understanding what is required by the current transaction.

`event.transaction`	This property represents forthcoming details about your current transaction.
`acr_values`	Authentication context class values or `acr_values` tells us what specific business rules are necessary in the authentication request in order to allow access to a given resource.

`isMfa` looks into the request cut from your web app service, and sees what rules are necessary to be satisfied. In your application, this request would look something like:

```
https://{yourDomain}/authorize?
        audience=https://{yourDomain}/userinfo&
        scope=openid&
        response_type=code&
        client_id={yourClientId}&
        redirect_uri={https://yourApp/callback}&
        state={yourOpaqueValue}&
        acr_values=http://schemas.openid.netpape/policies/
        2007/06/multi-factor
```

If no MFA was used at login, and our `acr_values` indicates we need MFA, then your Action will request MFA from your user in order for your web app to access the information on their behalf.

When the user authenticates with MFA, a new ID token is issued, and they will not be asked to provide MFA again during their session.

To see your Action in, well, action, deploy your Action, then drag-and-drop your Action into the **Login** trigger flow, and hit **Apply**.

This is just one way to implement step-up authentication with Auth0.

You can even create Actions in combination with permissions to require the extra authentication step before, say, proceeding with a high-value transaction, or any sensitive task in-platform.

Summary

This template illustrates how to use the api interface with event context data to implement basic step-up authentication.

You can combine even more information about the user, session, and custom claims to create a sophisticated security posture that ensures peace of mind for both your admins and your consumers.

Actions templates show you that with a few lines of code, you can customize Actions functions to do big things with less.

Check out our other implementation guides:

- Flag sensitive transactions for step-up
- Collect FPD post-reg and post-login
- Prompt with personalized recommendations
- Customize UI for accessibility