Applied research document

Dimitar Prisadnikov

S3-ITS-CB05

Contents

Introduction

This applied research focuses on finding the available approached when it comes to implementing login page with Auth0. It also aim to outline the differences between the available options in terms of security, UI and UX. The STAR method is used to describe the situation, task, action and results for each sub question.

Questions

Main question

What login implementation is more suitable for a web applications using Auth0 and how it affects the security and customizability?

Sub questions and research methods

- What login implementation are available with Auth0?
    - Literature study
    - Community research
- Which option (and why) is preferred by other companies/developers?
    - Available product analysis
    - Community research
- How each of the possible implementation affect the security aspect of the application?
    - Literature study
    - Available product analysis
    - Community research
- How each of the possible implementation affect the user experience?
    - Literature study
    - Community research
- What options are available to customize the UI?
    - Literature study
    - Community research

Research

What login implementation are available with Auth0?

Situation: When designing authentication experience with Auth0, the developer(s) have to choose one of the available login solutions.

Task: The goal of my research is to find what options are available when it comes to implementing login with Auth0.

Action: I made sure to get to know the main differences between the two by reading the official documentation provided by Auth0. In addition, I browsed StackOverflow in order to find out more about the available options from the perspective of the development community.

Result: Based on my research, I found out there are two login implementations available – Centralized Universal login and Embedded Login. [1] With the centralized universal login, users are redirected to a central domain (Auth0 page which handles the credentials) where they complete the sign in/up process and if successful they go back to the application. [2] [3]On the other hand, the embedded login provides the ability to implement the same functionality in the app directly which

eliminates the need of redirecting to a central domain. In this case, the credentials are handled by the application and sent to the authentication provider for authentication. [1] [3]

Which option (and why) is preferred by other companies/developers?

Situation: Embedded or universal sign in/up is not a choice only developers using Auth0 has to make. There are many auth products/flows/technologies that require companies and developers to select one or the other.

Task: My aim is to find which of the two is preferred by other developers and companies and why that is the case.

Action: I spent time reading on StackOverflow, the official Auth0 forums and blogs to find out which option and why is preferred over the other. In addition, I browsed existing products implanting Auth0 authentication.

Result: Google used to provide two ways to implement OAuth 2.0 SSO which were quite similar to the Auth0 ones. However, in September 2021 they retired the embedded one due to security concerns. Their embedded view allowed “man in the middle” attacks where nefarious developers could intercept and alter communications between Google and its users. Right now Google only support a centralized login solution which means the credentials handling happens on a centralized location controlled by them (that’s why when you sign in/up with your Google account it pops up in a new window). [4] When it comes to the community opinion on the topic, it turns out most people agree that the universal login is more secure, easier to debug and helps to understand the auth flow better, however, it is more or less restricted in terms of UI/UX. Although, it comes with plenty of options to tweak and adjust the UI/UX, it is nowhere even close to the embedded login which is pretty much just HTML and CSS. [1] [5]

How each of the possible implementation affect the security aspect of the application?

Situation: When it comes to credentials handling and security, the two login options are quite different to a point where one might be picked over the other solely due to its security. Although no approach can be 100% secure, still choosing the more secure one is always important.

Task: My task is to compare how the two login implementation compare security-wise.

Action: I will conduct my research by reading the official Auth0 documentation. In addition, I will try to find forums where developers have posted their experience, preferences and suggestions about each approach in terms of security. Lastly, I will include how other companies handle this.

Result: Reading the official Auth0 documentation suggest that the centralized universal login is more secure due to number of reasons. The main arguments in favor of it are the cookies being from the same origin and eliminating cross-origin requests. [1] [5] [6]Collecting user credentials in an application served from one origin (for example, React.js client) and then sending them to another origin (for example, the auth server) can present certain security vulnerabilities such as “man in the middle” attacks. Universal Login does not send information between origins, thereby negating cross-origin concerns With embedded login not only there is cross-origin authentication but also the app has access to both the authorization grant and the user's authentication credentials. As a consequence, this data is left vulnerable to recording or malicious use. Even if the app is trusted, allowing it to access the authorization grant as well as the user's full credentials is unnecessary according to Auth0. [7]

How each of the possible implementation affect the user experience?

Situation: When it comes to UX, the two options are rather similar but by no means identical. They come with differences that could have impact on the UX.

Task: My task is to compare the two UX when each login approach is used.

Action: I will be searching for answers by reading the Auth0 documentation. Also, I will check what the community thinks.

Result: When using the centralized universal login, by default, Auth0 always redirect the user to a new page. The page itself, based on the setup, could either have Auth0 domain or a custom one to match the application one. [2] The first one, might make people feel like the landed on a foreign web page and be confused by it. This could be avoid by using custom domains which allows the Auth0 page to have the same domain as your application, however, this is not a free feature. The community opinion is that the centralized universal login does not really have significant negative impact on the UX when set up properly (custom domain, custom UI templates, etc.). [7] [8]Lastly, the universal login is easier to implement and maintain since there is almost no code related and everything is done through the Auth0 admin panel. By avoiding the need to code all auth features from scratch, there is also lower chance of introducing a bug affecting the UX negatively. Regarding the embedded login, it could result in better UX since the user never have to leave the application, however, this is something that vary from application to application and if not done properly it could turn out to be worse than the default Auth0 UX.

What options are available to customize the UI?

Situation: Both login approaches can be customized to help with the company branding, however, the handle the UI customization in significantly different manner.

Task: My tasks is to find how and what can be customized on the sign in/up page for each approach.

Action: In order to learn more about the topic, I will be reading the official Auth0 documentation, blogs posts and community posts.

Result: The embedded login is the option that provides unlimited customizability since it allows developers to use HTML, CSS, JS and all types of front-end frameworks and libraries in order to create the design they are aiming for. [2] [9] On the other hand, the universal login offers can also be customized with CSS however it is nowhere even close to the flexibility of the other approach. With an centralized universal login page, the developer can create custom (CSS) templates for their login page, however, this is a paid features. If they decide to not go with custom templates, the branding possibilities are really limited (mainly colors and logos).

Conclusion

The two available login page approaches – centralized universal login and embedded login, are quite different and have their advantages and disadvantages. The universal login is more secure, easier to implement and maintain but comes with limited UI and UX possibilities. On the other hand, there is the embedded login that seems to be less and less preferred by companies due to it being significantly less secure and more prone to issue. Still, it could be the better solution in some cases where the UI and UX is number one priority and can not be satisfies with just CSS.

Recommendation

My recommendation is to go with the centralized universal login approach unless the UI/UX is more important than the security and the team behind the project can spend a lot of time and resources developing their own login solution.

References

[1] Auth0, "Centralized Universal Login vs. Embedded Login," Auth0, [Online]. Available: https://auth0.com/docs/authenticate/login/universal-vs-embedded-login. [Accessed 2022].

[2] Auth0, "Customize New Universal Login Pages," Auth0, [Online]. Available: https://auth0.com/docs/customize/universal-login-pages/universal-login-page-templates. [Accessed 2022].

[3] Auth0, "Authentication Provider Best Practices: Universal Login," Auth0, 12 12 2017. [Online]. Available: https://auth0.com/blog/authentication-provider-best-practices-centralized-login/. [Accessed 2022].

[4] Google, "Upcoming security changes to Google's OAuth 2.0 authorization endpoint in embedded webviews," Google, [Online]. Available: https://developers.googleblog.com/2021/06/upcoming-security-changes-to-googles-oauth-2.0-authorization-endpoint.html. [Accessed 2022].

[5] dan.woda, "Embedded login is wrong and obsolete?," 14 1 2022. [Online]. Available: https://community.auth0.com/t/embedded-login-is-wrong-and-obsolete/76444/3. [Accessed 2022].

[6] H. Peregrino, "Auth0 embedded login flow," 2021 7 15. [Online]. Available: https://stackoverflow.com/questions/68325925/auth0-embedded-login-flow/68396311#68396311. [Accessed 2022].

[7] jmangelo, "Why is it frowned upon to use an embedded Lock login on an "untrusted client"?," 5 5 2017. [Online]. Available: https://community.auth0.com/t/why-is-it-frowned-upon-to-use-an-embedded-lock-login-on-an-untrusted-client/6496/2. [Accessed 2022].

[8] Auth0, "Introducing the New Universal Login Experience," Auth0, 2019 6 19. [Online]. Available: https://auth0.com/blog/introducing-the-new-auth0-universal-login-experience/. [Accessed 2022].

[9] O. B. David, "A Guide for an Awesome Custom Auth0 Universal login," 31 7 2022. [Online]. Available: https://www.permit.io/blog/custom-auth0-universal-login. [Accessed 2022].