WordPress SSO Integration - Solution Design Document

Version: 1.00
Updated: 27/09/2024
Author: Danilo Ercoli

Introduction

This document outlines the design for integrating the client's custom Single Sign-On (SSO) system with a new WordPress site. The goal is to provide a seamless authentication experience by redirecting users to the client's centralized authentication system and managing user accounts within WordPress accordingly. This integration ensures that all user management and authentication processes are consistent across the client's various systems, including their CRM, DAM, and marketing tools.

Objectives

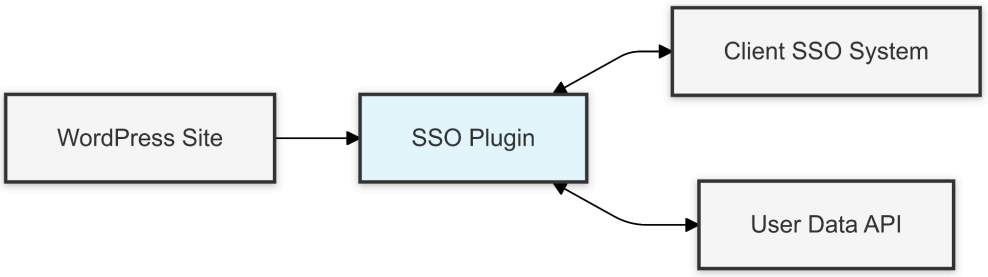
- Redirect WordPress login attempts to the client's centralized login page.
- Authenticate users returning from the centralized login with a JWT token.
- Create or update WordPress user accounts based on data from the "User Data" REST endpoint.
- Dynamically display "Login," "My Account," and "Logout" links within WordPress content.
- Support both staging and production environments with configurable settings.
- Ensure synchronization of user account details upon login if updates are detected.

Scope of Work

The solution involves developing a custom WordPress plugin that will manage the authentication flow, user account management, and dynamic rendering of authentication-related links within the site's content. The plugin must be designed to be user-friendly for both administrators and content editors, integrating smoothly with WordPress's full-site editing capabilities.

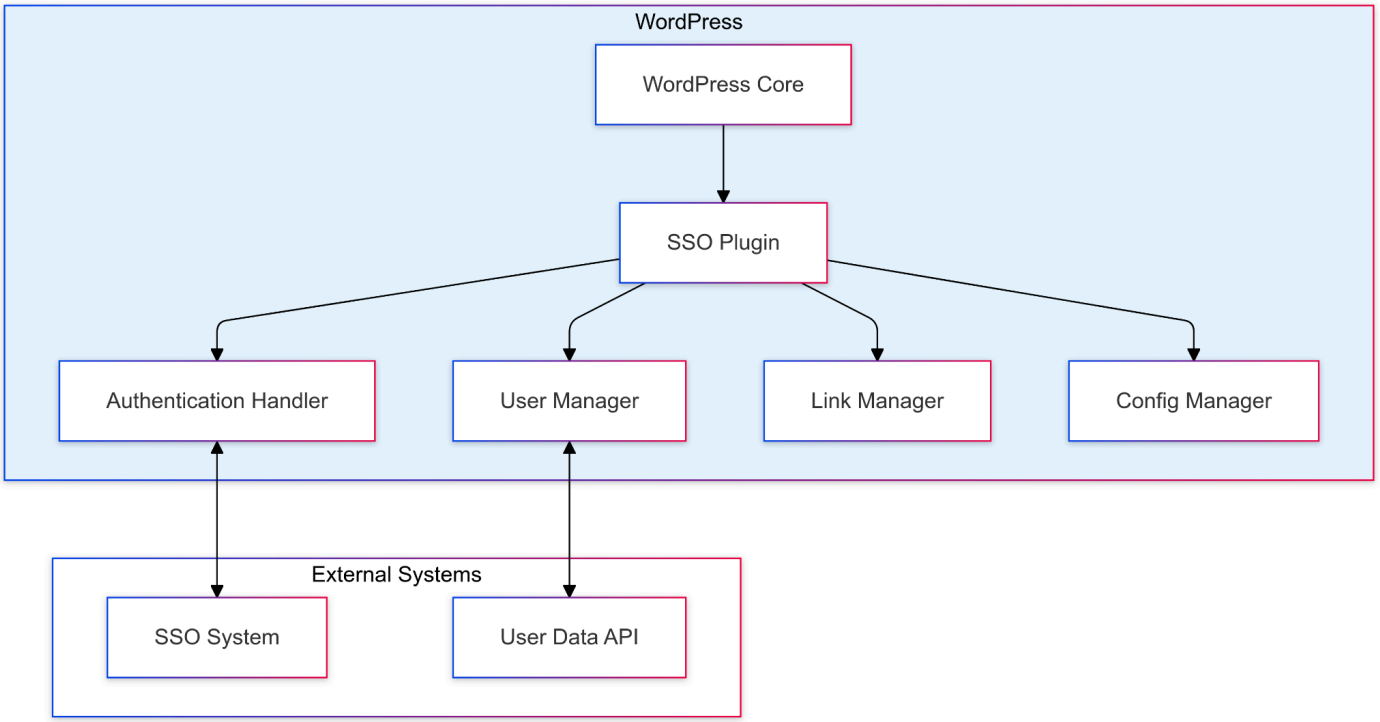
Solution Overview

System Context



Core Components

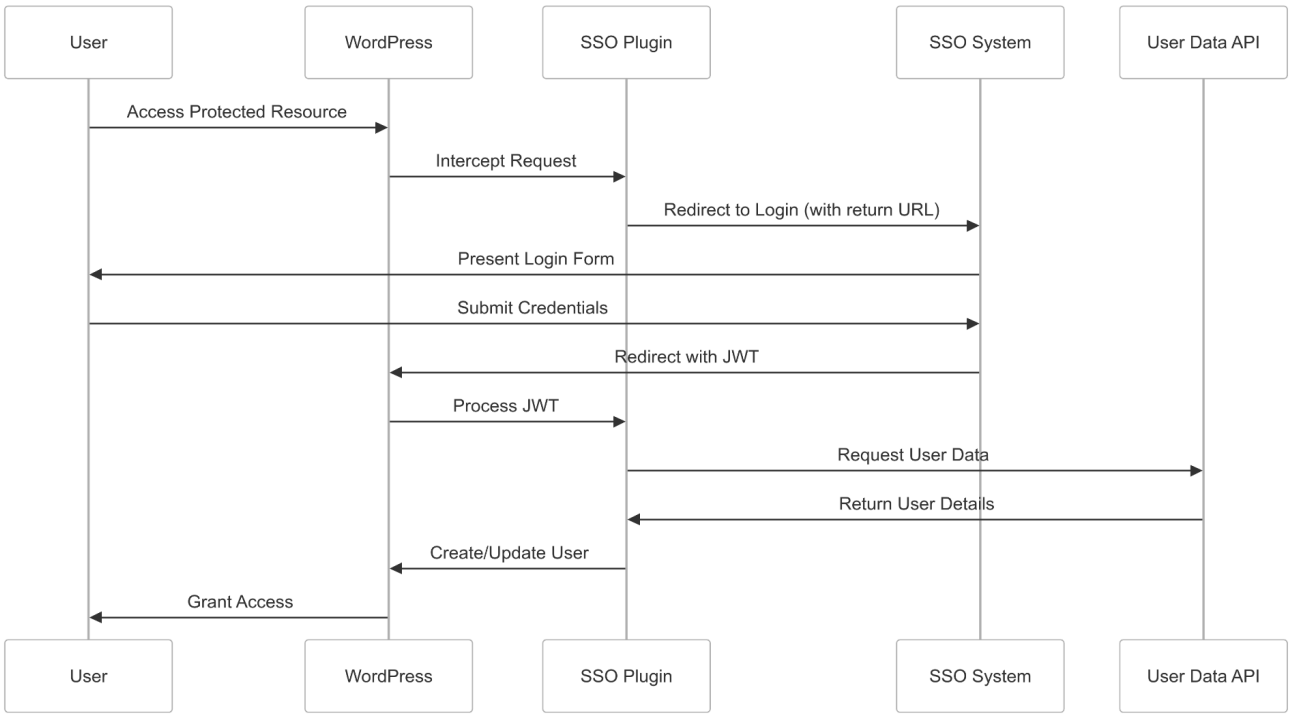
- **WordPress Site:** The main application requiring SSO integration.
- **SSO Plugin:** Custom WordPress plugin managing authentication flows.
- **Client SSO System:** Centralized authentication service.
- **User Data API:** External service providing user information.



Authentication Flow and Redirection

When a user attempts to log into the WordPress site, whether by accessing the login page directly or by trying to access a protected resource, the custom plugin will intercept this action. Instead of presenting the standard WordPress login form, the plugin will redirect the user to the client's centralized login page. This redirection is crucial for maintaining a unified authentication experience across all of the client's systems.

The redirection to the centralized login page will include the URL of the WordPress site as a parameter, typically referred to as `return_url` or a similar identifier. This parameter ensures that after the user successfully authenticates with the SSO system, they can be redirected back to the appropriate location on the WordPress site



Post-Login Process and JWT Handling

Upon successful authentication at the centralized login page, the SSO system will redirect the user back to the WordPress site. The redirection URL will include a JWT token as a parameter. This token is crucial for verifying the user's identity and initiating their session within WordPress.

The custom plugin will capture this JWT token from the URL and proceed to validate it. The validation process involves checking the token's signature using the HS256 algorithm and the shared secret specific to the environment (staging or production). This step is vital for ensuring the token's authenticity and preventing any tampering or forgery.

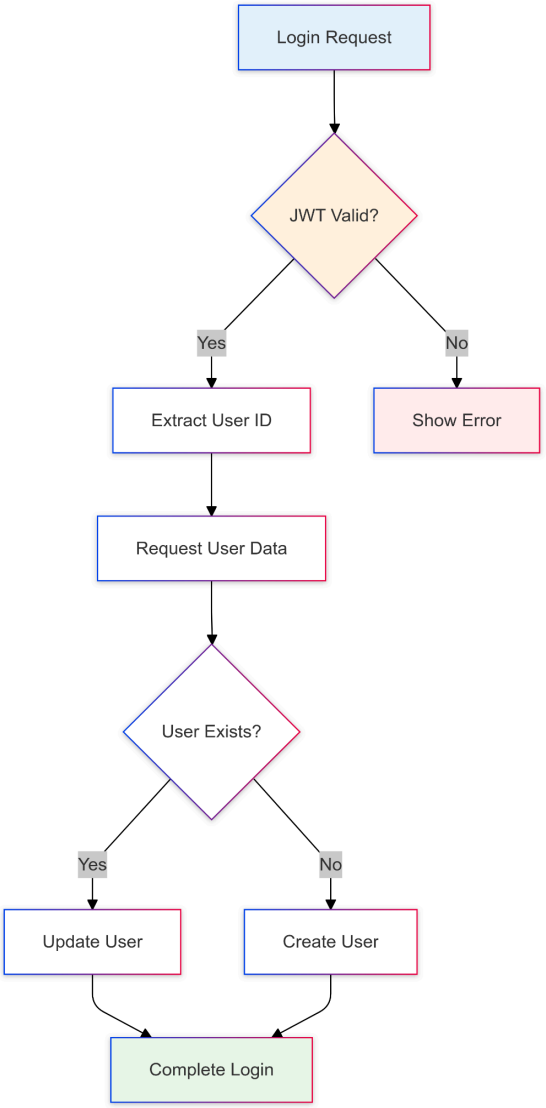
If the JWT validation fails, the plugin will display an error message to the user, indicating that authentication has failed. The user will not be logged into WordPress, and they may be prompted to retry the login process or contact support if the issue persists.

Retrieval of User Data and Account Management

Assuming the JWT is valid, the plugin will extract the "subject" claim from the token's payload, which contains the unique identifier (ID) of the user who has logged in. However, the JWT does not provide any additional user information, such as name, email, or role. To obtain this information, the plugin must communicate with the client's "User Data" REST endpoint.

The plugin will send a POST request to the "User Data" endpoint, including the JWT in the body of the request. Authentication for this request is handled via Bearer Authentication using an API key provided by the client. This API key is a sensitive piece of information and must be stored securely within the plugin's configuration.

Upon sending the request, the plugin expects to receive a response containing the user's data, including their name, email, role, and a "last-updated" timestamp indicating when the user's details were last modified in the SSO system.



Error Handling During Data Retrieval

If the request to the "User Data" endpoint fails—for example, due to network issues, invalid authentication, or an error response from the endpoint—the plugin will handle this gracefully. An error message will be displayed to the user, explaining that their account information could not be retrieved. The user will not be logged into WordPress, and the plugin will not proceed with account creation or updates.

Creation and Synchronization of WordPress User Accounts

With the user data successfully retrieved from the "User Data" endpoint, the plugin will proceed to create or update the corresponding WordPress user account.

New User Accounts

If the user does not already exist within the WordPress site's user database (determined by a unique identifier such as email or a custom user meta field storing the SSO user ID), the plugin will create a new WordPress user account. The user's name, email, and role will be set based on the data provided by the "User Data" endpoint.

Existing User Accounts

If the user already exists in WordPress, the plugin will compare the "last-updated" timestamp from the "User Data" endpoint with a stored timestamp on the WordPress user account. If the endpoint's timestamp is more recent, it indicates that the user's details have been updated in the SSO system since their last login. The plugin will then update the WordPress user account accordingly, synchronizing fields such as name, email, and role.

This synchronization ensures that any changes made to the user's account in the SSO system—such as a name change or role update—are reflected in WordPress upon their next login. However, it is important to note that this synchronization only occurs during the login process and is not performed in real-time while the user is logged into WordPress

Role Mapping and Assignment

The user's role within WordPress is determined by the "role" field provided by the "User Data" endpoint. This field contains an integer value between 1 and 3, representing different levels of access:

- A value of 1 corresponds to the WordPress role of "author."
- A value of 2 corresponds to "editor."
- A value of 3 corresponds to "administrator."

The plugin will map these integer values to the appropriate WordPress roles during account creation or update. It is critical to validate this mapping to prevent unauthorized access levels. For instance, the plugin must ensure that only the values 1, 2, or 3 are accepted and that any unexpected values are handled appropriately, possibly by assigning a default role or preventing login.

Session Management and Authentication Persistence

Once the user's WordPress account has been created or updated, the plugin will log the user into WordPress using the standard authentication mechanisms provided by the platform. From this point forward, the user's session is managed by WordPress, and the SSO system is not involved until the user logs out and logs back in.

The JWT token is not stored or used beyond the initial authentication process. This approach maintains security by not passing the JWT between pages or using it for ongoing authentication, reducing the risk of token exposure.

Logout Process

When a user decides to log out of the WordPress site, the plugin will handle this action by logging the user out of WordPress as usual. The plugin may also optionally redirect the user to the centralized logout page provided by the client's SSO system. This ensures that the user's session is terminated both on the WordPress site and within the SSO system, providing a complete logout experience.

Dynamic Rendering of Authentication Links

To enhance the user experience and provide easy access to authentication-related pages, the plugin will offer dynamic rendering of "Login," "My Account," and "Logout" links within the WordPress site's content.

Given that the site utilizes WordPress's full-site editing capabilities, the plugin will provide custom Gutenberg blocks that editors can place anywhere within the site's content. These blocks are designed to adapt their display based on the user's authentication state:

- **For Logged-in Users:** The blocks will display "My Account" and "Logout" links. The "My Account" link directs users to the client's centralized "My Account" page, allowing them to manage their account details. The "Logout" link logs the user out of WordPress (and optionally the SSO system) and redirects them appropriately.
- **For Guests (Not Logged-in Users):** The blocks will display a "Login" link that redirects users to the client's centralized login page.

This dynamic behavior ensures that users always see the most relevant options based on their current authentication state, enhancing usability and navigation within the site.

Environment Configuration and Management

The plugin must support both staging and production environments, each with its own set of configurations. This includes different URLs for the "Login," "Register," "My Account," and potentially "Logout" pages, as well as different base URLs and API keys for the "User Data" REST endpoint and different shared secrets for JWT decoding.

To manage these configurations, the plugin will include a settings page within the WordPress admin dashboard. Administrators can select the active environment (staging or production) and input the necessary parameters for each environment. Sensitive information, such as API keys and shared secrets, will be stored securely using WordPress's options API and will not be exposed in the codebase or to unauthorized users.

The settings page will provide a user-friendly interface, guiding administrators through the configuration process and ensuring that all required fields are completed. Validation checks will alert administrators if any configurations are missing or invalid, preventing potential authentication issues.

Error Handling and User Feedback

A critical aspect of the solution is providing clear and user-friendly error handling. When issues arise during the authentication process, the plugin must inform the user without exposing sensitive system details.

JWT Validation Failure

If the JWT token fails validation—due to an incorrect signature, expiration, or other issues—the plugin will display an error message indicating that authentication has failed. The message should be concise and suggest that the user retry the login process. For example:

"Authentication failed. Please try logging in again. If the problem persists, contact support."

User Data Retrieval Failure

When the plugin cannot retrieve user data from the "User Data" endpoint—perhaps due to network errors, invalid authentication, or server issues—it will inform the user that their account information could not be retrieved. The user will not be logged in, and the message may suggest trying again later. For example:

"Unable to retrieve your account information at this time. Please try again later. If the issue continues, please contact support."

General Error Handling

All errors encountered during the authentication process will be logged for administrative review. Logs will include sufficient details for troubleshooting but will avoid recording sensitive information like JWT tokens or API keys. Administrators can access these logs to identify and resolve underlying issues.

Limitations and Assumptions

Dependencies on external systems

The solution assumes that the client's SSO system and "User Data" endpoint are reliable and available. In cases where these systems are down, users will experience authentication issues, and contingency plans may need to be considered.

Synchronization Only on Login

User data synchronization occurs only during the login process. If a user's details are updated in the SSO system while they are logged into WordPress, those changes will not be reflected until the user logs out and logs back in.

Future Enhancements

Several enhancements could be considered for future development:

Real-Time User Data Synchronization

Implementing background processes or cron jobs to periodically synchronize user data from the SSO system could ensure that updates are reflected in WordPress without requiring the user to log out and back in.

- Email address changed in SSO causing potential conflicts with existing WordPress users.
- Handling merge scenarios when duplicate accounts are detected.

SSO Logout Synchronization

Extending the logout functionality to log users out of the SSO system when they log out of WordPress would provide a more seamless experience and enhance security.