June 2020

Standardized Flow-control Signals to Support Chaining of Identity Providers

Guibin Kong

Steven Soneff

Standardized Flow-control Signals to Support Chaining of Identity Providers

ABSTRACT

Many online services, such as websites or apps, facilitate user sign-up or sign-in via different identity providers (IDPs). When a service permits the use of multiple IDPs, displaying the UIs of all of these at the same time is not suitable from a user experience point of view. In such cases, each of the available IDP options needs to be shown one by one. However, owing to various issues regarding how libraries and/or Application Programming Interfaces (APIs) of different IDPs handle the authentication action flow, it is difficult for online services to integrate with multiple IDPs in such a chained manner. To tackle this problem, the techniques described in this disclosure provide a chainable API to facilitate standardized communication between an online service that requests user authentication and an IDP that holds the user's credentials.

KEYWORDS

- Identity provider
- User authentication
- Login
- User credentials
- Authentication protocol
- Authentication flow
- Chained authentication
- Flow-control signals
- OAuth 2.0

BACKGROUND

Many online services, such as websites or apps, facilitate user sign-up or sign-in via different identity providers (IDPs). For example, the IDPs are third parties with whom users may already have an account with corresponding login credentials. Typically, each IDP uses its own user interface (UI) to enable end users to enter their credentials and choose appropriate actions, such as granting specific permissions to a website or app based on the selected credentials.

When a service permits the use of multiple IDPs, displaying the UIs of all of these at the same time is not suitable from a user experience point of view. In such cases, each of the available IDP options needs to be shown one by one. However, handling such chained flow of credential retrieval poses several problems:

1. Notifications from the libraries and/or application programming interfaces (APIs) do not always pass information to the original service at the most appropriate times. For example, a response from the library and/or API is not always assured if no UI can be displayed (e.g., due to no credentials being available) or if a user wishes to skip the currently shown IDP. In such cases, without an explicit indication from the library and/or API, the requesting app doesn't know the UI status, and cannot call the next IDP immediately or stop the chained credential retrieval flow altogether if the user does not wish to try any of the offered IDPs.
2. Even when the libraries and/or APIs return the authentication flow-control signals (e.g., in the form of error messages), the content and formats of these messages, such as data structures, field names, data types, etc., can vary widely across different IDPs, thus requiring the requesting service or app to handle each case separately to extract the flow-control signals. Further, the use of error messages results in the flow-control signals being mixed with other errors, such as those related to the specific authentication protocol. As a result, the

requesting service can encounter cases where protocol related errors need to be translated into flow-control signals.

Owing to these problems, it is difficult for online services to integrate multiple IDPs.

DESCRIPTION

This disclosure describes techniques to provide a chainable API to facilitate standardized communication between an online service that requests user authentication and an IDP that holds the user's credentials. The standardized communication involves IDPs using a separate channel to communicate the status of their authentication user interfaces to the requesting service by using predefined flow-control signals that are defined in a standard format that is independent of any specific authentication protocol. The techniques are implemented with specific user permission.

The predefined flow-control signals include notifications of the following specific moments within an authentication interaction with an IDP:

- **Display**: indicates whether the authentication UI is displayed to the user.
- **Skip**: indicates that the user wishes to skip the currently shown IDP.
- **Done**: indicates that the user successfully authenticated with the IDP, subsequently closing the authentication UI.
- **Abort**: indicates that the user does not wish to use the chained credential retrieval flow for authentication.

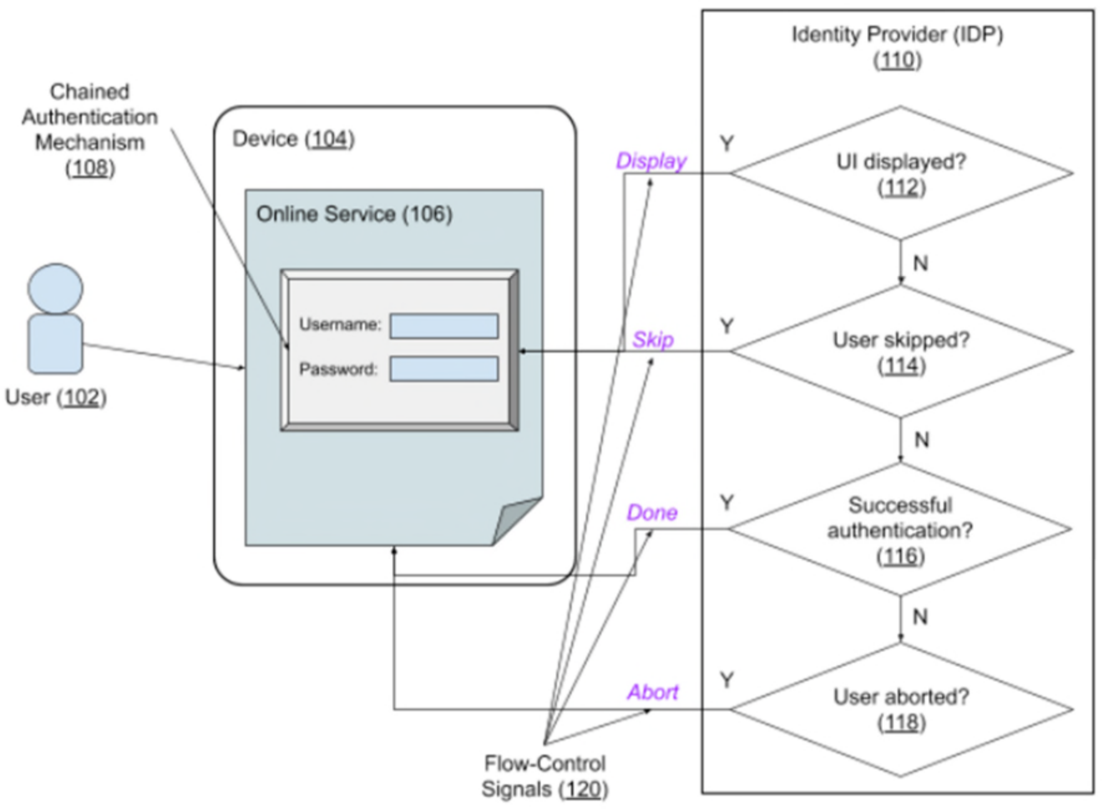

Fig. 1: Chaining identity providers via authentication flow-control signals

Fig. 1 shows an operational implementation of the techniques described in this disclosure. A user (102) accesses an online service (106) via a device (104). The service provides a chained authentication mechanism (108) that allows users a choice of IDPs, presented one after another.

The currently shown IDP (110) checks various aspects of the authentication UI: whether the UI is displayed (112), if the user skipped (114), if authentication was successful (116), or if the user aborted chained authentication (118). As applicable, the checks generate corresponding flow-control signals (120): display, skip, done, and abort, respectively.

The display and skip signals are relayed to the requesting entity and enable determination of whether to move to the next IDP in the chain. The done and abort signals are relayed to the online service to signal the end of the chained authentication process since the user credentials were verified successfully or the user chose to not use the process, respectively.

If a user's credentials are not available for a given IDP, the authentication UI is not displayed and the display signal is set to indicate that the user was not shown the UI. Similarly, the signal to skip a given IDP can be generated to indicate that the user does not wish to authenticate via that IDP or the user failed to authenticate with the IDP. Each signal can include the type of the signal along with other relevant information such as a Boolean value to indicate display status for the display signal, a string to indicate an optional reason for the skip and abort signals, etc.

When a skip signal is received, the chained credential retrieval flow proceeds with the next IDP in the authentication chain, if any. If all IDP options in the chain are exhausted, the chained flow is stopped. Similarly, the chained flow is stopped when a done or abort signal is received because the user has successfully authenticated or chosen not to use the chained credential retrieval flow, respectively. Since both done and abort signals necessitate the end of the chained credential retrieval flow, they can be combined into a single signal type with a reason field used in the case of abort to distinguish between the two.

A dedicated callback function can provide the separate channel to receive the flow-control signals. For example, an IDP may include an API of the form: theIDP.retrieveCredential(responseCallback). The functionality described in this disclosure can be achieved by extending the API with the addition of another callback parameter: theIDP.retrieveCredential(responseCallback, uiMomentCallback). The first callback (responseCallback) parameter is used to process credential type related responses as usual. The added callback (uiMomentCallback) parameter is employed to process flow-control signals that support the chained credential retrieval flow.

Alternatively, or in addition, the flow-control signals can be communicated via an event-driven model. In such implementations, whenever an IDP generates a flow-control signal corresponding to any of the moments described above, a uiMoment event is triggered. The API can support handling the event with appropriate functions via commands such as: addEventListener(uiMoment, uiMomentListener).

Support for the described techniques can be added by minor modifications that extend current authentication libraries and/or APIs used for credential management and identity provision. Since the described flow-control signals are independent of the specifics of any authentication protocol, the functionality can support the inclusion of any IDP, thus making the system broad in scope and easily extensible by adding IDPs. As long as the authentication response and flow-control signals are sent via separate channels, the implementation of the described functionality can use any available channel, supporting a variety of platforms that can differ in terms of channel availability. Overall, implementation of the techniques described in this disclosure enables online services or apps to provide multiple IDPs for user authentication in a seamless chained manner that provides a smooth UX.

CONCLUSION

The techniques described in this disclosure provide a chainable API to facilitate standardized communication between an online service that requests user authentication and an IDP that holds the user's credentials. The standardized user-permitted communication involves IDPs using a separate channel to communicate the status of their authentication UIs to the requesting service by using predefined flow-control signals that are defined in a standard format independent of any specific authentication protocol. A dedicated callback function can provide the separate channel used to receive the flow-control signals. Alternatively, or in addition, the flow-control signals can be communicated via an event-driven model. The described flow-control signals are independent of the specifics of any authentication protocol.

REFERENCES

1. How to implement identity federation
2. AWSCredentialsProviderChain (AWS SDK for Java - 1.11.779)
3. Credential Management Level 1, W3C Working Draft
4. One-tap sign-up and auto sign-in on websites