Adding Third-Party Authentication to Open edX: A Case Study

John Cox
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043 USA
johncox@google.com

Pavel Simakov
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043 USA
psimakov@google.com

Abstract
In this document, we describe the third-party authentication system we added to Open edX. With this system, Open edX administrators can allow their users to sign in with a large array of external authentication providers. We outline the features and advantages of the system, describe how it can be extended and customized, and highlight reusable design principles that can be applied to other authentication implementations in online education.

Author Keywords
API design; authentication; authorization; education; identity; privacy; security; testing.

L@S 2015, Mar 14-18, 2015, Vancouver, BC, Canada
ACM 978-1-4503-3411-2/15/03.
http://dx.doi.org/10.1145/2724660.2728675

ACM Classification Keywords
D.2.13 [Reusable Software]: Reusable libraries; K.6.5 [Security and Protection]: Authentication.

Introduction
User authentication is the process whereby a user of a computer system proves their identity to that computer system so they can then be granted the ability to make certain protected actions within that system. It is a difficult problem because maximizing the security an authentication system delivers in practical terms requires balancing two factors: first, the strength of the

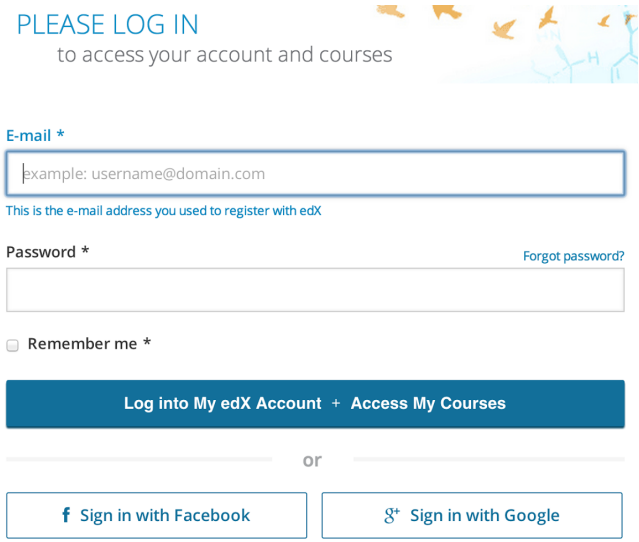

Figure 1. edx.org sign-in page with third-party authentication.

security provided by the system itself, often through strict adherence to ever-evolving cryptological standards; and second, user convenience.

These factors are often in tension: user convenience is maximized by not having an authentication step at all, and security is maximized often by creating systems that are very difficult for users to satisfy. If a system is too hard for users to use, they will opt either not to use the system at all, or they will take steps that increase their convenience but decrease the effective security of the system, such as constructing weak passwords, re-using passwords between systems, and so on.

Finding the balance between these factors, in addition to implementing authentication correctly in the first place, is very difficult. Many authors of computer systems lack the expertise or the time necessary to do this, so they may deliver authentication implementations in their products that are not secure.

Authentication in the online education space is further complicated by the need to integrate with legacy authentication systems at educational institutions. Moving off these systems is impractical, so new online education tools must be able to interoperate with them.

Below we outline an approach we took to creating a secure, configurable, extensible authentication solution for use in the online education space. We cover the features of our implementation, and explain the design principles that can be applied to implementations beyond our own.

Background

Open edX [1] is an open-source, freely-available platform for authoring and delivering educational content at scale. Until 2014, it provided only first-party authentication. "First-party authentication" means the system that requires user authentication is the same as the system that provides user authentication. First-party authentication systems are both common and simple:

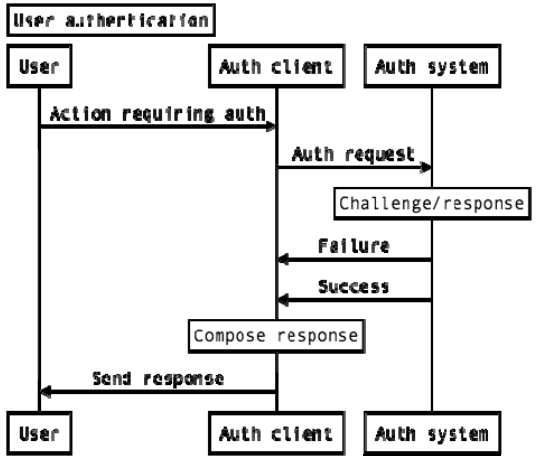

Figure 2. In a first-party authentication system, when a user takes an action that requires authentication, they are first taken to the authentication system component. In this component they are challenged to provide their identity, and they provide a response to that challenge. If they fail the challenge, the system composes a response notifying them of failure. If they succeed, the system composes a success response, which is often the resource they requested with their initial action.

Users of a first-party authentication system will have a user account with that system. That account will store an identifier for that user, their credentials (such as a nonreversible hash of their password), and other user data.

This is distinct from third-party authentication, where the system that requires user authentication (hereafter the "authentication consumer") are different from the system that users authenticate with (hereafter the "authentication provider"). The steps the user goes through under third-party authentication are the same as in Figure 2 above, except they undergo challenge/response with the authentication provider. Consequently, the authentication consumer does not need to store or validate user credentials.

This has two advantages. First, it is more convenient for the user because they do not have an additional set of credentials to manage. Second, the authentication implementation of the provider system is often more secure than authentication systems written by the authors of the consumer system, since authentication is the provider system authors' core competency.

Implementation details

At the core of our implementation [2] is python-social-auth [3], an open-source library that supports third-party authentication, written in the Python programming language. We picked python-social-auth because it supports three open authentication protocols (OpenID, OAuth 1, and OAuth 2 [4]) and over 60 authentication providers [5], is also written in Python, and provides good abstractions for the two biggest extension points we identified when gathering requirements.

First, by using an abstraction called the authentication pipeline, which comes from python-social-auth, future implementers can hook into the authentication flow at any point and insert custom code. This is done by exposing the behavior of the authentication code as a re-entrant stack of function calls via a Python API, each representing a conceptual step in the authentication process. This set of steps can be extended and re-ordered, and the pipeline as a whole can be paused and resumed on an ad-hoc basis by custom authentication code in the containing system.

Second, python-social-auth provides abstractions for additional authentication protocols or providers, also via Python APIs. These are vital because many organizations in the educational space have vast pre-existing computing infrastructure, including custom single-sign on (SSO) systems for user authentication. These systems may speak a host of different protocols, of which Central Authentication Service (CAS) [6] and Shibboleth [7] are the most common. Switching away from their legacy authentication systems is both impractical and undesirable for these organizations.

We wrapped python-social-auth in a thin layer that manages configuration details for a given deployment. This is where, for example, any deployment of Open edX selects the set of providers it will use from the full set of available providers. We then refactored the Open edX codebase to optionally use python-social-auth for authentication actions alongside the existing first-party authentication implementation. We adapted the Open edX user interface to account for a small number of new user actions, like managing associations between an existing Open edX account and any number existing provider accounts. Once these associations are

established, users can sign in to their Open edX account with any of the associated provider accounts. Users may revoke an association at any time.

Finally, we provided an extensive set of tests [8]. A major feature of our tests is a comprehensive suite that is executed once per known authentication provider. This makes it easier to test new providers against the system as a whole with a minimal investment of new code: provider developers extend one class with a few dozen lines of provider-specific detail, and the full suite is executed against that provider. This limits the knowledge of the underlying Open edX system that a provider author needs to have in order to write a new provider, and at the same minimizes the risk that they will miss important edge cases during development.

Conclusions

The approach outlined above proved successful. This third-party authentication system has been live on edx.org, the largest deployment of Open edX, since September of 2014. Members of the Open edX community have used the abstractions detailed above to author and deploy customized versions of Open edX with additional authentication providers and customized user authentication flows.

While the system has been extended beyond its initial implementation to account for new requirements, real-world use found no gaps in the overall design. We therefore consider this design approach successful and recommend it to other developers in the online education space who require integrations with third-party authentication providers.

Acknowledgements

We are very grateful to our partners at edX who worked with us on the development of this feature. As well, we'd like to extend our thanks to the university and industry staff we interviewed while gathering requirements. Finally, we would like to thank Matías Aguirre and the other authors of python-social-auth. The library is excellent, and most of the ideas in this paper are very strongly influenced by the design choices and abstractions it established.

References

[1] Open edX. http://code.edx.org.

[2] Open edX third_party_auth module. https://github.com/edx/edx-platform/tree/master/common/djangoapps/third_party_auth.

[3] Python Social Auth. https://github.com/omab/python-social-auth.

[4] Python Social Auth documentation. http://psa.matiasaguirre.net/docs/backends/implementation.html.

[5] Python Social Auth documentation. http://psa.matiasaguirre.net/docs/backends/index.html#social-backends.

[6] CAS Protocol 3.0 Specification. https://github.com/Jasig/cas/blob/master/cas-server-documentation/protocol/CAS-Protocol-Specification.md.

[7] Shibboleth. https://shibboleth.net/.

[8] Open edX third_party_auth module tests. https://github.com/edx/edx-platform/tree/master/common/djangoapps/third_party_auth/tests.