Strong Authentication In and Beyond the Browser

Lessons Learned from FIDO

Brad Hill, PayPal, brad.hill@paypal.com
Jeff Hodges, PayPal, jeff.hodges@paypal.com
Davit Baghdasaryan, NokNok Labs, davit@noknok.com
Christiaan Brand, Entersekt, christiaan@entersekt.com
Rob Philpott, RSA, robert.philpott@rsa.com

18-July-2014, for the W3C Workshop on Web Cryptography Next Steps

The original chartering of the WebCrypto Working Group emerged in part from the W3C's "Identity in the Browser" Workshop, which identified that standardized access to strong, low-level crypto abstractions was absolutely necessary for the Open Web Platform. Browser-based access to strong crypto will enable many important applications, but displacing the password[1] from its reign as the seemingly inevitable authentication credential will require a solution with reach beyond just the browser.

Recently, the FIDO Alliance (https://fidoalliance.org) has undertaken the design of two protocols to produce authentication solutions that are both stronger and easier to use than passwords, meet important requirements for both end-users and consumer application service providers, and also provide enough strength for enterprise and regulated use cases. In the process, we have gained some insight into how such systems must be integrated both into the browser and platforms beyond it.

Application Facets, Application Isolation, and Stopping In-App Phishing

A key feature of modern distributed applications is that users interact with the same data and logic across multiple modalities. They access network-based services in a browser with a web app, and they also interact with platform-native applications produced by the same service providers. They may also access the same web app in multiple browsers on the same device. This is common across a wide spectrum of application types. In FIDO, we have termed these to be "facets" of the same underlying distributed application. Like a password, a user will expect to employ any replacement credential across these different facets on the same device and, if their credential can roam from device to device, they expect to be able to employ it in the "same" app on different platforms.

Some previous approaches addressing multi-modal strong authentication involved applications delegating the user experience back to browsers. Rather than re-implementing authentication workflows, applications simply opened an embedded browser in which the user could complete

[1] I.e., user-wielded, reusable, sharable, password.

the web-based workflow. While this works for traditional operating systems where all apps share a single security principal (e.g. Microsoft Word on Windows opening an embedded Internet Explorer control to handle logon, where both run as the logged-in user) this model has serious shortcomings on new platforms that isolate apps by assigning them their security principals.

Platforms (including Android, iOS, Windows Phone and Windows 8) that provide an “app store” and notions of “app identity” often tell users that any app is “safe to try”. That is, they attempt to guarantee that a malicious app cannot interfere with the operation of other applications, and it cannot access sensitive system services without being granted special permission from the user. If strong authentication ceremonies are strictly implemented in terms of a browser, keeping these guarantees becomes difficult. Users must either endure disorienting context-switches between installed apps and a system-level browser app, or worry about “in-app phishing” because embedded browsers (aka “WebViews”) on such platforms are typically under the full control of any malicious app that instantiates them.

Our goals for a solution to these issues in FIDO included the following:

- The user should be able to utilize a registered credential across multiple facets of the same application.
- User should be protected from “app phishing” - especially for biometric-type experiences where verification data is inherently reused.
- For operating environments that provide isolation among applications, installing a malicious app should not violate established relationships among trustworthy components or compromise high-value shared state (e.g. cryptographic keys) on the system.
- Establishing and maintaining the security of relationships among trusted components must be transparent to the user. They should not have to make choices about which apps to trust or approve requests to share credentials.

Meta-goals included:

- Users should retain free choice about which web browsers they choose, and web browsers are a special-case application because they are trusted to work with facets of many different applications.
- Mechanisms for security enforcement between application facets should rely on the native security features and verifiable notions of application identity for each platform. They should not involve, e.g. bundling shared “secrets” as part of the packaged distribution from an app store.

User Experience and Authentication Modalities

A strong barrier to supplanting the password is that web sites desire to maintain strong control and branding of the authentication experience in the context of their application. Experiences like the browser HTTP Authentication dialog are considered unacceptable.

But service providers also would like to be able to accept strong authenticators, no matter how they are implemented, including new modalities they may not have seen before, so long as they can verifiably provide the necessary security guarantees.

FIDO has attempted to strike a balance between these two goals. It provides applications a means to discover information about the available authentication capabilities of a device, including icons for different authenticator modalities. This allows a website or app to provide a branded, in-context experience under its own control for much of the authentication process - and for some types of authenticators the user may never see any other UI elements. But for some types of authenticators, the UX must be handed off to an implementation provided outside the application context. e.g. a local PIN verification code to unlock a cryptographic element cannot be securely entered in the application context. Our goals for this set of issues included:

- Employ only the minimum necessary UX outside of the application-provided experience. (e.g. maybe just flash an LED on the authenticator device and provide no on-screen guidance)
- When UX is provided, it should be consistent across all facets. e.g. the experience to authenticate with a fingerprint should be the consistent across every browser on the system, when invoked from any app, or when used by the operating system to unlock the device.

Additionally, optional access to a “secure display” is a desirable feature for confirming transactions. This must also be provided outside the execution context of the application in order to be useful for non-repudiation.

Privacy

Strong user privacy guarantees are critical to consumer acceptance in the broad marketplace. This includes the ability to do things such as having multiple accounts at the same service provider (e.g. multiple email addresses with the same webmail system) without having those accounts inherently linked or collapsed by the authentication modality. Solving this for some configurations may require the user have some “account chooser” functionality implemented outside the context of the local application.

Lessons Learned

In devising an overall solution to meet these and other diverse goals we have found that, while the Open Web Platform must be a first-class citizen of any authentication ecosystem, Web APIs and the browser (or APIs directly connecting web apps to secure elements of various types) are not by themselves sufficient to meet user and service providers' expectations for a system to supplant passwords.

FIDO has found it is necessary to introduce at least one additional layer of abstraction to manage many of these concerns, mediate among various consumers of credentials, and in some limited cases communicate directly with a service provider. (e.g. to retrieve a list of authorized application facets)

Our solution exposes four basic functions that can be implemented as APIs across various platforms to expose this underlying abstraction layer, which we call the "FIDO client":

- Discovery - allow an app to determine what authentication capabilities and modalities are available on the user's device
- Registration - create a new credential and register it with an application
- Authentication - provide proof of control of (and optionally verify a transaction using) a registered credential
- Deregistration - allow a server to manage credential lifecycles transparently to the user

Graining broad adoption and availability requires standardizing these kinds of APIs, first and foremost for the Web and browsers, but also eventually for platforms like Android, iOS, and Windows Phone.

The goal of the FIDO Alliance is to produce open, unencumbered specifications, but also to submit these to recognized standards-setting organizations such as the W3C in order to foster the broadest possible adoption.

We hope that the W3C membership will consider, at such time as FIDO Alliance's membership agreement allows, chartering or accepting into the chartered scope of an existing working group, this type of approach to facilitating strong authentication.