Prompting the user is security failure

Adam Barth
Google, Inc

Introduction

When adding new features to the web platform, we must be mindful of the security consequences of letting arbitrary web sites use those features. A common approach to restricting access to dangerous features, such as the ability to read from the user's microphone and video camera, is to ask the user whether the access should be allowed. In this paper, I argue that prompting the user with security question is security failure because users are annoyed in non-attack scenarios (and become complacent) and, in non-attack scenarios, user grant the attacker access anyway.

There are a number of web platform features that prompt the user with security decisions. By studying whether these prompts are successful at achieving their security goals, we gain insight into how to design better authorization interfaces. Broadly speaking, security prompts fall into two categories: modal and non-modal. Modal security prompts interrupt the user's current task and require the user to make a security decision before they can complete their task. Non-modal prompts, by contrast, let the user complete his or her primary task without making a security decision.

Modal prompts

Browser contain a number of modal security prompts, some of which are more successful than others. In this section, I discuss three well-known prompts: certificate errors, mixed content, and file upload.

Certificate error. One of the most well-known modal security prompts is the HTTPS certificate error prompt. When the user attempts to navigate to a web site that uses HTTPS, the site presents the browser with a certificate binding the web site's identity to some cryptographic material. If the site presents an invalid certificate, the browser will not complete the connection and, instead, will prompt the user. The vast majority of times the mixed content prompt is shown, there is no attack underway. Instead, the server is simply misconfigured, and users proceed with the connection approximately 80% of the time.

Mixed content. Prior to version 9, Internet Explorer used a modal prompt to ask the user whether a secure site (e.g., an HTTPS site) should be allowed to include insecure resources (e.g., a script over HTTP). The mixed content prompt suspends the rendering of the web page, and the user must either allow or deny access to the resource before interacting further with the browser. These so-called "mixed content" vulnerabilities are relatively common on the web, and users become habituated to clicking through these errors. Worse, denying the load often leaves the web site unusable.

Instead of using modal prompts, other browsers use a non-modal notification of mixed content, usually by degrading the lock icon in the primary security user interface. In the current beta release of Internet Explorer 9, the browser no longer shows the modal prompt. Instead, “passive” resources (e.g., images) are allowed with a non-modal notification and “active” resources (e.g., scripts) are blocked. This new design is vast security improvement over the previous design because the browser makes a more accurate risk management decision for the user than the user would likely have made for themselves.

File upload. The file upload control is one of the most successful security prompts in the web platform. In fact, many users do not even realize they are making a security decision when using the file upload control. Instead of granting web sites read access to the user’s files system, web sites are allowed to read files under two conditions:

1. The user selects the file with a file-picker dialog.
2. The user drags and drops the file onto the web page.

In both cases, the web site is allowed access only to the particular file the user designates. Moreover, there is no separate “authorization” step. The act of designating which file the web site ought to operate on is often required for non-security reasons.

Non-modal prompts

Over time, browsers have been shifting towards using non-modal security prompts. Users often click through modal security prompts because the prompts train them that they need to “click OK” to accomplish their primary task.

Downloads. The authorization experience for executing a downloaded file is one of the most powerful security prompts in browsers because executing a downloaded file grants the author of that file unrestricted access to the user’s computer. Traditionally, browsers have used modal modal security prompts to control saving and executing downloads. More recently, browsers have shifted to non-modal security prompt. For example, Safari saves downloads automatically but does not execute them. Google Chrome places downloads on a “download bar” at the bottom of the browser window.

Browser also harden their download authorization prompts against clickjacking attacks. Because the browser itself displays the download authorization prompt, the browser can avoid web content drawing on top of the prompt, but attackers have managed to mount timing-based clickjacking attacks. For example, the Google Chrome download bar requires two clicks in spatially separated regions of the screen in order to actually execute a downloaded file. These mechanisms are cumbersome for users but necessary to prevent attackers from tricking users into dismissing the security prompts unintentionally.

Geolocation. Recently, browsers have added support for sharing geolocation information with web sites. When a web page invokes the geolocation API, the browser prompts the user to authorize the web site. In Mobile Safari, the prompt is modal and blocks access to the web site until the user dismisses the prompt. Desktop browsers typically use a non-modal “infobar” security prompt. These infobars have a number of security issues:

1. Attackers can display fake infobars that are indistinguishable from infobars displayed by the browser because infobars are displayed in the content area. There are been examples of legitimate web sites (such as The Huffington Post) spoofing infobars.
2. Attackers can use timing-based clickjacking attacks to cause the user to click on the infobar unintentionally. If the user dismisses the security prompt unintentionally, the prompt provides little security benefit.

Infobars are not a scalable solution for security prompts because the more web platform features use infobars to grant permissions, the more users will habitually dismiss the infobars without reading them.

Pop-up blocker. Most modern browsers include a pop-up blocker to prevent web sites from opening unwanted windows. In most cases, browsers are able to correctly determine when the user wishes to grant the web site the ability to open a new window by determining whether the web site opened in the window in response to a “user gesture,” effectively rate-limiting the creation of new windows.

In some cases, however, the user wishes a web site to open a window without a user gesture. Most browsers use a non-modal prompt to let the user authorize the web page to circumvent the pop-up blocker. This prompt is successful because in the vast majority of cases, the user never needs to interact with the prompt (unlike the geolocation prompt which usually requires user interaction). In the rare cases when the browser has the wrong default for a particular web site, the user can interact with the prompt and correct the error.

Two ways forward

A systemic problem with security prompts is that they stand between the user and their primary task. When interacting with a web site, users are not usually focused on managing security. Instead, they are intent on some other task, such as finding a hip restaurant or booking a flight. Security prompts are must successful in situations where the browser can make a reasonable guess as to the correct answer. In the extreme, the browser is able make the correct decision in all cases and can avoid prompting the user entirely. These considerations point towards two directions for improving the user experience of browser authorization.

Better defaults. The most successful of the existing security prompts leverage additional context to make better default decisions. For example, file uploads and the pop-up blocker security prompts leverage information about recent user interaction to make better authorization decisions. For certificate errors, the Strict-Transport-Security header gives browsers more context to make better authorization decisions. In designing new web platform features, we should consider what additional contextual information browser can use to make better authorization decisions.

Pre-authorization. Rather than prompting the user for security during their primary task, we should consider authorization experiences that occur before the user is fixated on a task that requires the authorization. Prompting for security earlier also lets up batch up multiple permissions into a single authorization experience, reducing authorization fatigue. One promising direction for improving security prompts is the notion of an installable web application. By prompting for security at “install-time,” the user has more context for making an authorization decision. For example, if the user installs the web application from a gallery, the gallery can provide third-party rates and reviews as well as inform the user about how many other users use the web application. The user can then see the full list of privileges requested by the application and make a decision about whether to install it. If the user later decides to revoke the privileges, the user can simply uninstall the web application.

Conclusion

Real-time communication requires granting web sites access to the user’s microphone and video camera. Because these privileges seem too powerful to grant to arbitrary web sites, browsers will need to implement some kind of authorization experience before exposing these features. Rather than simply prompting the user for authorization, we should think more carefully about how to design a better authorization experience.