websec	A. Barth
Internet-Draft	Google, Inc.
Intended status: Informational	February 2011
Expires: August 03, 2011	

Principles of the Same-Origin Policy
draft-abarth-principles-of-origin-00

Abstract

The security model of the web platform has evolved over time to meet the needs of new applications and to correct earlier mistakes. Although web security has evolved largely organically, the security model has converged towards a handful of key concepts. This document presents those concepts and provides advice to designers of new pieces of the web platform.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The security model of the web platform has evolved over time to meet the needs of new applications and to correct earlier mistakes. Although web security has evolved largely organically, the security model has converged towards a handful of key concepts. This document presents those concepts and provides advice to designers of new pieces of the web platform.

2. Trust

```
<script src="https://example.com/library.js"></script>
```

On the web, trust is specified by URL. For example, HTML documents designate which script to run with a URL:
When a user agent process this element, the user agent will fetch the script at that URL and execute the script with the privileges of the document. In this way, the document grants all the privileges it has to the resource specified by the URL. In essence, the document declares that it trusts the integrity of information retrieved from that URL.

```
<form method="POST" action="https://example.com/login">
 ... <input type="password"> ...
</form>
```

In addition to importing libraries from URLs, user agents also let principals export data to URLs. For example, consider the HTML form element:
When the user enters his or her password and submits the form, the password is sent to the network endpoint designated by the URL. In this way, the document exports its secret data to that URL. In essence, the document declares that it trusts the confidentiality of information sent to that URL.

2.1. Pitfalls

When designing new pieces of the web platform, make sure that important trust distinctions are visible in URLs. For example, if both TLS and non-TLS protected resources used the "http" URL scheme, a document would be unable to specify that it wished to retrieve a script over TLS. By using the "https" URL scheme, documents are able to indicate that they wish to interact with resources that are protected from active network attackers.

3. Origin

In principle, user agents could treat every URL as a separate principal and isolate each document from every other URL unless the document explicitly indicated that it trusted that URL. Unfortunately, this design is cumbersome for developers because web applications often consist of a number of resource acting in concert.
As an approximation, user agents group URLs together into protection domains called origins. In particular, two URLs are part of the same origin (i.e., represent the same principal) if they have the same scheme, host, and port.
Q: Why not just use the host?
A: Including the scheme in the origin tuple is essential for security. If user agents did not include the scheme, there would be no isolation between documents from http://example.com and https://example.com because the two have the same host. However, without this isolation, an active network attacker could corrupt the document retrieved from http://example.com and then use that document as a stepping stone to compromise the confidentiality and integrity of the document retrieved from https://example.com, bypassing the protections afforded by TLS.
Q: Why use the fully qualified host name instead of just the "top-level" domain?
A: Although the DNS has hierarchical delegation, the trust relationships between host names vary by deployment. For example, at

many educational institutions, students can host content at https://example.edu/~student/, but that does not mean a document authored by a student should be part of the same origin (i.e., represent the same principal) as a web application for managing grades hosted at https://grades.example.edu/.

4. Authority

Although user agents group URLs into origins, not every resource in an origin carries the same authority. For example, an image is passive content and, therefore, carries no authority, meaning the image has no access to the objects and resources available to its origin. By contrast, an HTML document contains active content and, therefore, carries the full authority of its origin. The user agent determines how much authority to grant a resource by examining its MIME type. For example, resources with a MIME type of image/png are treated as images and resources with a MIME type of text/html are treated as HTML documents.

When hosting untrusted content (such as user-generated content), web applications can limit that content's authority by restricting its MIME type. For example, serving user-generated content as image/png is less risky than serving user-generated content as text/html. Of course many web applications incorporate untrusted content in their HTML documents. If not done carefully, these applications risk leaking their origin's authority to the untrusted content, a vulnerability commonly known as cross-site scripting.

4.1. Pitfalls

When designing new pieces of the web platform, be careful not to grant authority to resources irrespective of MIME type. Many web applications server untrusted content with restricted MIME types. A new web platform feature that grants authority to these pieces of content risks introducing vulnerabilities into existing applications. Instead, prefer to grant authority to MIME types that already possess the origin's full authority or to new MIME types designed specifically to carry the new authority.

5. Policy

Generally speaking, user agents isolate different origins and permit controlled communication between origins. The details of how user agents provide isolation and communication vary depending on several factors.

5.1. Object Access

Most objects (also known as application programming interfaces or APIs) exposed by the user agent respect the same-origin policy. Specifically, a script running on behalf of one document can access objects

associated with another document if, and only if, the two documents belong to the same origin, e.g., were retrieved from URLs with the same scheme, host, and port.
There are some exceptions to this general policy. For example, some parts of HTML's Location interface are available across origins (e.g., to allow for navigating other browsing contexts). As another sample, HTML's postMessage interface is visible across origins to facilitate cross-origin communication. Exposing objects to foreign origins is dangerous and should be done only with great care because doing so exposes these objects to potential attackers.

5.2. Network Access

Access to network resources varies depending on whether the resources are in the same origin as the document attempting to access them. Generally, reading information from another origin is forbidden. However, a document is permitted use some kinds of resources retrieved from other origins. For example, a document is permitted to execute script, render images, and apply style sheets from any origin. Likewise, a document can display a document from another origin in a frame.
Generally, sending information to another origin is permitted. However, sending information over the network in arbitrary formats is dangerous. For this reason, user agents restrict documents to sending information using particular protocols, such as in an HTTP request without custom headers.

5.3. Pitfalls

Whenever user agents allow documents from one origin to interact with resources in another origin, they invite security issues. For example, the ability to display images from another origin leaks their height and width across origins. Similarly, the ability to send network requests to another origin gives rise to cross-site request forgery vulnerabilities. However, user agents tolerate these risks when the value proposition is high enough. For example, a user agent that blocked cross-origin network requests would prevent its users from following hyperlinks, a core component of the web architecture.
When designing new pieces of the web platform, it can be tempting to grant a privilege to one document but to withhold that privilege from another document in the same origin. However, withholding privileges in this way is ineffective because the document without the privilege can usually obtain the privilege anyway because user agents do not isolate documents within an origin. Instead, new pieces of the platform should grant or withhold privileges from origins as a whole (rather than discriminating between individual documents within an origin).

6. Conclusion

The web security model designates trust relationships with URLs. URLs are grouped together into origins, which represent principals. Some resources in an origin (active content) are granted the origin’s full authority, whereas other resources in the origin (passive content) are not granted the origin’s authority. Documents that carry their origin’s authority are granted access to objects and network resources within their own origin. These documents are also granted limited access to objects and network resources of other origins, but these cross-origin privileges must be carefully designed to avoid introducing security vulnerabilities into the web platform.

7. References

Appendix A. Acknowledgements

We would like to thank Ian Hickson, Collin Jackson, Thomas Roessler, Jesse Ruderman, and Daniel Veditz for their valuable feedback on this document.

Author's Address

Adam Barth Barth Google, Inc. EMail: ietf@adambarth.com URI: http://www.adambarth.com/