Политика одного и того же происхождения

Политика одного **и того же источника** — это критически важный механизм безопасности, который ограничивает взаимодействие документа или сценария, загруженного из одного источника , с ресурсом из другого источника.

Это помогает изолировать потенциально вредоносные документы, уменьшая возможные векторы атак. Например, он не позволяет вредоносному веб-сайту в Интернете запускать JS в браузере для чтения данных из сторонней службы веб-почты (в которую входит пользователь) или внутренней сети компании (которая защищена от прямого доступа злоумышленника с помощью не имея публичного IP-адреса) и передавая эти данные злоумышленнику.

Определение происхождения

Два URL-адреса имеют одинаковое происхождение, если протокол , порт (если указан) и хост одинаковы для обоих. Вы можете увидеть это как «кортеж схемы/хоста/порта» или просто «кортеж». («Кортеж» — это набор элементов, которые вместе составляют целое — общая форма для двойного, тройного, четверного, пятерного и т. д.)

В следующей таблице приведены примеры сравнения происхождения с URL-адресом `http://store.company.com/dir/page.html`:

URL-адрес	Исход	Причина
http://store.company.com/dir2/other.html	То же происхождение	Только путь отличается
http://store.company.com/dir/inner/another.html	То же происхождение	Только путь отличается
https://store.company.com/page.html	Отказ	Другой протокол
http://store.company.com:81/dir/page.html	Отказ	Другой порт (http://по умолчанию это порт 80)
http://news.company.com/dir/page.html	Отказ	Другой хост

Унаследованное происхождение

Скрипты, выполняемые на страницах с URL-адресом `about:blank` или `javascript:`, наследуют источник документа, содержащего этот URL-адрес, поскольку эти типы URL-адресов не содержат информации об исходном сервере.

Например, `about:blank` часто используется как URL-адрес новых пустых всплывающих окон, в которые родительский скрипт записывает контент (например, через механизм `Window.open()`). Если это всплывающее окно также содержит JavaScript, этот сценарий унаследует то же происхождение, что и сценарий, который его создал.

`data:` URL-адреса получают новый, пустой контекст безопасности.

Происхождение файлов

Современные браузеры обычно рассматривают происхождение файлов, загружаемых с использованием `file:///` схемы, как непрозрачное происхождение . Это означает, что если файл включает в себя другие файлы из той же папки (скажем), не предполагается, что они происходят из того же источника и могут вызвать ошибки CORS.

Обратите внимание, что в спецификации URL указано, что происхождение файлов зависит от реализации, и некоторые браузеры могут рассматривать файлы в одном и том же каталоге или подкаталоге как имеющие одинаковое происхождение, даже если это имеет последствия для безопасности .

Изменение происхождения

Предупреждение. Описанный здесь подход (с использованием document.domainустановщика) устарел, поскольку он подрывает защиту безопасности, обеспечиваемую той же политикой происхождения, и усложняет модель происхождения в браузерах, что приводит к проблемам совместимости и ошибкам безопасности.

Страница может изменить свое происхождение с некоторыми ограничениями. Сценарий может установить значение document.domain для своего текущего домена или супердомена своего текущего домена. Если установлен супердомен текущего домена, более короткий супердомен используется для проверок того же происхождения.

Например, предположим, что сценарий из документа at `http://store.company.com/dir/other.html` выполняет следующее:

JS

```
document.domain = "company.com";
```

После этого страница может пройти проверку того же происхождения с помощью http://company.com/dir/page.html(при условии, что `http://company.com/dir/page.html` для нее установлено `document.domain` значение «`company.com`», чтобы указать, что она желает это разрешить - см. document.domainдополнительную информацию). Однако не `company.com` удалось установить , поскольку это не супердомен `.document.domainothercompany.comcompany.com`

Номер порта проверяется браузером отдельно. Любой вызов `document.domain`, включая `document.domain = document.domain`, приводит к перезаписи номера порта на `null`. Следовательно, невозможно поговорить `company.com:8080`, `company.com` только установив `document.domain = «company.com»` первое. Он должен быть установлен в обоих, чтобы оба номера портов были `null`.

Механизм имеет некоторые ограничения. Например, он выдаст «SecurityError» DOMException, если включен или документ находится в изолированной программной среде , и изменение источника таким образом не влияет на проверки происхождения, используемые многими веб-API (например , , ,). Более полный список случаев сбоя можно найти в Document.domain > Отказы. document-domain Permissions-Policy <iframe> localStorage indexedDB BroadcastChannel SharedWorker

Примечание. При использовании `document.domain` субдомена для доступа к своему родительскому элементу необходимо установить `document.domain` одно и то же значение как в родительском домене, так и в субдомене. Это необходимо, даже если при этом родительский домен возвращается к исходному значению. Невыполнение этого требования может привести к ошибкам разрешения.

Доступ к сети из разных источников

Политика одного и того же источника контролирует взаимодействие между двумя разными источниками, например, когда вы используете XMLHttpRequest или <img> элемент. Эти взаимодействия обычно делятся на три категории:

- Запись между источниками обычно разрешена. Примерами являются ссылки, перенаправления и отправки форм. Некоторые HTTP-запросы требуют предварительной проверки.
- Обычно допускается встраивание перекрестного происхождения . (Примеры перечислены ниже.)
- Чтение из перекрестного источника обычно запрещено, но доступ для чтения часто теряется из-за внедрения. Например, вы можете прочитать размеры встроенного изображения, действия встроенного скрипта или доступность встроенного ресурса.

Вот несколько примеров ресурсов, которые могут быть встроены из разных источников:

- JavaScript с `<script src=«…»></script>`. Подробности синтаксических ошибок доступны только для сценариев того же происхождения.
- CSS применяется с `<link rel=«stylesheet» href=«…»>`. Из-за смягченных правил синтаксиса CSS для CSS с перекрестным происхождением требуется правильный `Content-Type` заголовок. Браузеры блокируют загрузку таблицы стилей, если это загрузка из разных источников, где тип MIME неверен и ресурс не начинается с допустимой конструкции CSS.
- Изображения, отображаемые <img>.
- СМИ играют <video> и <audio>.
- Внешние ресурсы, встроенные в <object> и <embed>.
- Шрифты, примененные с помощью @font-face. Некоторые браузеры допускают использование шрифтов из разных источников, другие требуют использования шрифтов одного и того же происхождения.
- Все, что встроено в <iframe>. Сайты могут использовать X-Frame-Options заголовок для предотвращения фреймов из разных источников.

Как разрешить доступ из разных источников

Используйте CORS, чтобы разрешить доступ из разных источников. CORS — это часть HTTP, которая позволяет серверам указывать любые другие хосты, с которых браузер должен разрешать загрузку контента.

Как заблокировать доступ из разных источников

- Чтобы предотвратить запись из разных источников, проверьте неугаданный токен в запросе, известный как токен межсайтовой подделки запроса (CSRF) . Вы должны запретить чтение страниц из перекрестного источника, которым требуется этот токен.
- Чтобы предотвратить чтение ресурса из перекрестного источника, убедитесь, что он не является встраиваемым. Часто бывает необходимо предотвратить внедрение, поскольку при внедрении ресурса всегда происходит утечка некоторой информации о нем.
- Чтобы предотвратить встраивание из разных источников, убедитесь, что ваш ресурс не может быть интерпретирован как один из встраиваемых форматов, перечисленных выше. Браузеры могут не учитывать Content-Typeзаголовок. Например, если вы укажете `<script>` тег на HTML-документ, браузер попытается проанализировать HTML как JavaScript. Если ваш ресурс не является точкой входа на ваш сайт, вы также можете использовать токен CSRF для предотвращения внедрения.

Доступ к API сценариев из разных источников

API JavaScript iframe.contentWindow, такие как window.parent, window.open, и window.openerпозволяют документам напрямую ссылаться друг на друга. Когда два документа имеют разное происхождение, эти ссылки обеспечивают очень ограниченный доступ к объектам Windowи Location, как описано в следующих двух разделах.

Для связи между документами из разных источников используйте window.postMessage.

Спецификация: HTML Living Standard § Объекты перекрестного происхождения .

Window

Разрешен следующий доступ из разных источников к этим Windowсвойствам:

Методы

- window.blur
- window.close
- window.focus
- window.postMessage

Атрибуты

window.closed	Только чтение.
window.frames	Только чтение.
window.length	Только чтение.
window.location	Читай пиши.
window.opener	Только чтение.
window.parent	Только чтение.
window.self	Только чтение.
window.top	Только чтение.
window.window	Только чтение.

Некоторые браузеры разрешают доступ к большему количеству свойств, чем указано выше.

Расположение

Разрешен следующий доступ к Location свойствам из разных источников:

Методы

- location.replace

Атрибуты	
location.href	Только запись.

Некоторые браузеры разрешают доступ к большему количеству свойств, чем указано выше.

Доступ к хранилищу данных из разных источников

Доступ к данным, хранящимся в браузере, таким как веб-хранилище и indexedDB, разделен по источнику. Каждый источник получает свое собственное отдельное хранилище, и JavaScript в одном источнике не может читать или записывать в хранилище, принадлежащее другому источнику.

Файлы cookie используют отдельное определение происхождения. Страница может установить файл cookie для своего собственного домена или любого родительского домена, если родительский домен не является общедоступным суффиксом. Firefox и Chrome используют список общедоступных суффиксов , чтобы определить, является ли домен общедоступным суффиксом. Когда вы устанавливаете файл cookie, вы можете ограничить его доступность с помощью флагов Domain, Path, Secureи HttpOnly. Когда вы читаете файл cookie, вы не можете видеть, где он был установлен. Даже если вы используете только безопасные соединения https, любой файл cookie, который вы видите, мог быть установлен с использованием небезопасного соединения.