

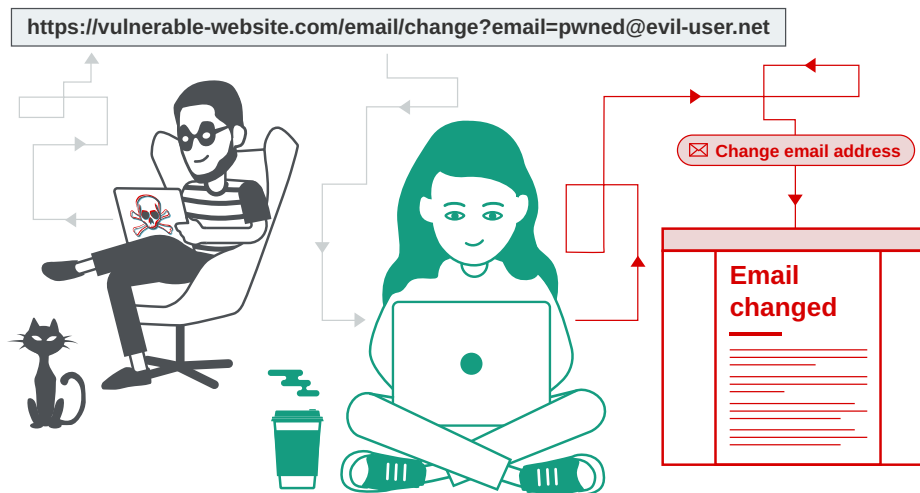
Подделка межсайтовых запросов (CSRF)



В этом разделе мы объясним, что такое подделка межсайтовых запросов, опишем некоторые примеры распространенных уязвимостей CSRF и объясним, как предотвратить атаки CSRF.

Что такое CSRF?

Подделка межсайтовых запросов (также известная как CSRF) - это уязвимость веб-безопасности, которая позволяет злоумышленнику побудить пользователей выполнить действия, которые они не собираются выполнять. Это позволяет злоумышленнику частично обойти одну и ту же политику происхождения, которая предназначена для предотвращения взаимодействия разных веб-сайтов друг с другом.



Каково влияние CSRF-атаки?

В успешной CSRF-атаке злоумышленник заставляет пользователя-жертву непреднамеренно выполнить действие. Например, это может быть изменение адреса электронной почты в учетной записи, изменение пароля или перевод денежных средств. В зависимости от характера действия злоумышленник может получить полный контроль над учетной записью пользователя. Если скомпрометированный пользователь имеет привилегированную роль в приложении, злоумышленник может получить полный контроль над всеми данными и функциями приложения.

Как работает CSRF?

Чтобы CSRF-атака была возможной, должны быть выполнены три ключевых условия:

- Соответствующее действие.** В приложении есть действие, которое злоумышленник может вызвать. Это может быть привилегированное действие (например, изменение разрешений для других пользователей) или любое действие с данными пользователя (например, изменение собственного пароля пользователя).
- Обработка сеанса на основе файлов cookie.** Выполнение действия включает в себя отправку одного или нескольких HTTP-запросов, и приложение полагается исключительно на файлы cookie сеанса для идентификации пользователя, отправившего запросы. Нет другого механизма для отслеживания сеансов или проверки запросов пользователей.
- Никаких непредсказуемых параметров запроса.** Запросы, выполняющие действие, не содержат параметров, значения которых злоумышленник не может определить или угадать. Например, когда вы заставляете пользователя изменить свой пароль, функция не будет уязвима, если злоумышленнику необходимо знать значение существующего пароля.

Например, предположим, что приложение содержит функцию, которая позволяет пользователю изменять адрес электронной почты в своей учетной записи. Когда пользователь выполняет это действие, он отправляет

Хотите отслеживать свой прогресс и получать более индивидуальный подход к обучению? (Это бесплатно!)

[Зарегистрирова](#)[Войти](#)

В этой теме

[CSRF](#)[CSRF](#)[XSS против CSRF](#)[Токены файлов cookie](#)[SameSite](#)

Все темы

[SQL инъекция](#)[XSS](#)[CSRF](#)[ClickJacking](#)[DOM основе](#)[CORS](#)[XXE](#)[SSRF](#)[запрос контрабанды](#)[Команда впрыска](#)[Серверный шаблон инъекции](#)[небезопасной десериализации](#)[обходных](#)[контроля доступа](#)[аутентификации](#)[OAuth аутентификации](#)[бизнес логики уязвимости](#)[Web кэша Отравление](#)[атак HTTP заголовков узла](#)[WebSockets](#)[Раскрытие информации](#)

Найдите
уязвимости CSRF с
помощью Bursuite

[Privacy - Terms](#)

HTTP-запрос, подобный следующему:

```
POST /email/change HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Cookie: session=yvthwsztyeQkAPzeQ5gHgTvlyxHfsAfE
```

email=wiener@normal-user.com

Это соответствует условиям, необходимым для CSRF:

- Действие по изменению адреса электронной почты в учетной записи пользователя представляет интерес для злоумышленника. После этого действия злоумышленник обычно может вызвать сброс пароля и получить полный контроль над учетной записью пользователя.
- Приложение использует файл cookie сеанса, чтобы определить, какой пользователь отправил запрос. Других токенов или механизмов для отслеживания пользовательских сессий не существует.
- Злоумышленник может легко определить значения параметров запроса, которые необходимы для выполнения действия.

При соблюдении этих условий злоумышленник может создать веб-страницу, содержащую следующий HTML-код:

```
<html>
<body>
  <form action="https://vulnerable-website.com/email/change" method="POST">
    <input type="hidden" name="email" value="pwned@evil-user.net" />
  </form>
  <script>
    document.forms[0].submit();
  </script>
</body>
</html>
```

Если пользователь-жертва посещает веб-страницу злоумышленника, произойдет следующее:

- Страница злоумышленника инициирует HTTP-запрос к уязвимому веб-сайту.
- Если пользователь вошел в систему на уязвимом веб-сайте, его браузер автоматически включит файл cookie сеанса в запрос (при условии, что файлы cookie SameSite не используются).
- Уязвимый веб-сайт обработает запрос обычным образом, будет считать его отправленным пользователем-жертвой и изменит свой адрес электронной почты.

Примечание

Хотя CSRF обычно описывается в связи с обработкой сеанса на основе файлов cookie, он также возникает в других контекстах, когда приложение автоматически добавляет некоторые учетные данные пользователя в запросы, такие как базовая аутентификация HTTP и аутентификация на основе сертификатов.

Как построить CSRF-атаку

Создание HTML, необходимого для эксплойта CSRF, вручную может быть обременительным, особенно если желаемый запрос содержит большое количество параметров или в нем есть другие особенности. Самый простой способ создать эксплойт CSRF - использовать [генератор CSRF PoC](#), встроенный в [Burp Suite Professional](#):

- Выберите запрос в любом месте Burp Suite Professional, который вы хотите протестировать или использовать.
- В контекстном меню, вызываемом правой кнопкой мыши, выберите Инструменты взаимодействия / Создать CSRF PoC.
- Burp Suite сгенерирует некоторый HTML, который вызовет выбранный запрос (за вычетом файлов cookie, которые будут автоматически добавлены браузером жертвы).
- Вы можете настроить различные параметры в генераторе CSRF PoC для точной настройки аспектов атаки. Это может потребоваться в некоторых необычных ситуациях, чтобы иметь дело с причудливыми функциями запросов.
- Скопируйте сгенерированный HTML-код на веб-страницу, просмотрите его в браузере, который вошел в систему на уязвимом веб-сайте, и проверьте, успешно ли выполнен предполагаемый запрос и выполнено ли желаемое действие.

[ПОПРОБУЙ БЕСПЛАТНО](#)

Как поставить эксплойт CSRF

Механизмы доставки для атак с подделкой межсайтовых запросов по существу такие же, как и для **отраженных XSS**. Обычно злоумышленник помещает вредоносный HTML-код на контролируемый им веб-сайт, а затем побуждает жертв посетить этот веб-сайт. Это может быть сделано путем подачи пользователю ссылки на веб-сайт по электронной почте или в сообщении в социальных сетях. Или, если атака проводится на популярный веб-сайт (например, в комментарии пользователя), они могут просто ждать, пока пользователи посетят этот веб-сайт.

Обратите внимание, что некоторые простые эксплойты CSRF используют метод GET и могут быть полностью автономными с помощью одного URL-адреса на уязвимом веб-сайте. В этой ситуации злоумышленнику может не потребоваться использовать внешний сайт, и он может напрямую передать жертвам вредоносный URL-адрес в уязвимом домене. В предыдущем примере, если запрос на изменение адреса электронной почты может быть выполнен с помощью метода GET, тогда автономная атака будет выглядеть так:

```

```

Читать далее

XSS против CSRF ⓘ

Предотвращение атак CSRF

Самый надежный способ защиты от атак CSRF - включить **токен CSRF** в соответствующие запросы. Токен должен быть:

- Непредсказуемо с высокой энтропией, как и для токенов сессий в целом.
- Привязан к сеансу пользователя.
- Строго проверяется в каждом случае перед выполнением соответствующего действия.

Читать далее

Токены CSRF ⓘ

Найдите уязвимости CSRF с помощью веб-сканера уязвимостей Burp Suite ⓘ

Дополнительная защита, которая частично эффективна против CSRF и может использоваться в сочетании с **токенами CSRF**, - это файлы cookie **SameSite**.

Распространенные уязвимости CSRF

Наиболее интересные уязвимости CSRF возникают из-за ошибок, допущенных при проверке **токенов CSRF**.

В предыдущем примере предположим, что приложение теперь включает токен CSRF в запрос на изменение пароля пользователя:

```
POST /email/change HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 68
Cookie: session=2yQIDcpia41WrAtfjPqvm9tOkDvkMvLm
```

```
csrf=WfF1szMUHhiokx9AHFply5L2xAOfjRkE&email=wiener@normal-user.com
```

Это должно предотвратить атаки CSRF, поскольку нарушает необходимые условия для уязвимости CSRF: приложение больше не полагается исключительно на файлы cookie для обработки сеанса, а запрос содержит параметр, значение которого злоумышленник не может определить. Однако существуют различные способы взлома защиты, что означает, что приложение по-прежнему уязвимо для CSRF.

Проверка токена CSRF зависит от метода запроса

Некоторые приложения правильно проверяют токен, когда запрос использует метод POST, но пропускают проверку, когда используется метод GET.

В этой ситуации злоумышленник может переключиться на метод GET, чтобы обойти проверку и выполнить CSRF-атаку:

```
GET /email/change?email=pwned@evil-user.net HTTP/1.1
Host: vulnerable-website.com
```



Cookie: session=2yQIDcpia4lWrATfjPqvm9tOkDvkMvLm

LAB CSRF, где проверка токена зависит от метода запроса ⓘ

Проверка токена CSRF зависит от наличия токена

Некоторые приложения правильно проверяют токен, если он присутствует, но пропускают проверку, если токен опущен.

В этой ситуации злоумышленник может удалить весь параметр, содержащий токен (а не только его значение), чтобы обойти проверку и выполнить CSRF-атаку:

```
POST /email/change HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Cookie: session=2yQIDcpia4lWrATfjPqvm9tOkDvkMvLm
```

email=pwned@evil-user.net

LAB CSRF, где проверка токена зависит от наличия токена ⓘ

Токен CSRF не привязан к пользовательской сессии

Некоторые приложения не проверяют, принадлежит ли токен тому же сеансу, что и пользователь, который делает запрос. Вместо этого приложение поддерживает глобальный пул выданных им токенов и принимает любой токен, который появляется в этом пуле.

В этой ситуации злоумышленник может войти в приложение, используя свою учетную запись, получить действительный токен, а затем передать этот токен пользователю-жертве в своей CSRF-атаке.

LAB CSRF, где токен не привязан к пользовательскому сеансу ⓘ

Токен CSRF привязан к несессионному cookie

В варианте предыдущей уязвимости некоторые приложения связывают токен CSRF с файлом cookie, но не с тем же файлом cookie, который используется для отслеживания сеансов. Это может легко произойти, когда приложение использует две разные структуры, одну для обработки сеанса, а другую для защиты CSRF, которые не интегрированы вместе:

```
POST /email/change HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 68
Cookie: session=pSJYSScWKpmC60LpFOAHKixuFuM4uXWF;
csrfKey=rZHCnSzEp8dbI6atzagGoSYyqJqTz5dv
```

csrf=RhV7yQDO0xcq9gLEah2WVbmuFqyOq7tY&email=wiener@normal-user.com

Эту ситуацию сложнее использовать, но она все еще уязвима. Если на веб-сайте присутствует какое-либо поведение, позволяющее злоумышленнику установить файл cookie в браузере жертвы, атака возможна. Злоумышленник может войти в приложение, используя свою учетную запись, получить действительный токен и связанный файл cookie, использовать поведение настройки файлов cookie, чтобы поместить свой файл cookie в браузер жертвы и передать свой токен жертве в своей CSRF-атаке.

LAB CSRF, где токен привязан к несессионному cookie ⓘ

Примечание

Поведение, связанное с настройкой файлов cookie, даже не обязательно должно существовать в том же веб-приложении, что и уязвимость CSRF. Любое другое приложение в том же общем домене DNS

потенциально может быть использовано для установки файлов cookie в целевом приложении, если контролируемый файл cookie имеет подходящую область действия. Например, функция настройки файлов cookie на `staging.demo.normal-website.com` может быть использован для размещения файла cookie, который отправляется `secure.normal-website.com`.


Токен CSRF просто дублируется в cookie

В еще одном варианте предыдущей уязвимости некоторые приложения не поддерживают серверную запись выданных токенов, а вместо этого дублируют каждый токен в файле cookie и параметре запроса. Когда последующий запрос подтвержден, приложение просто проверяет, соответствует ли токен, представленный в параметре запроса, значению, представленному в файле cookie. Это иногда называют защитой от CSRF с помощью «двойной отправки», и ее рекомендуют, потому что она проста в реализации и устраняет необходимость в каком-либо состоянии на стороне сервера:

```
POST /email/change HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 68
Cookie: session=1DQGdzYbOJQzLP7460tfyiv3do7MjyPw;
csrf=R8ov2YBfTYmzFyjIt8o2hKBuoIjXXVpa
```

```
csrf=R8ov2YBfTYmzFyjIt8o2hKBuoIjXXVpa&email=wiener@normal-user.com
```

В этой ситуации злоумышленник может снова выполнить CSRF-атаку, если веб-сайт содержит какие-либо функции настройки файлов cookie. Здесь злоумышленнику не нужно получать собственный токен. Они просто изобретают токен (возможно, в требуемом формате, если он проверяется), используют поведение настройки файлов cookie, чтобы поместить свой файл cookie в браузер жертвы, и передают свой токен жертве в своей CSRF-атаке.

LAB CSRF, где токен дублируется в cookie 

Защита от CSRF на основе рефереров

Помимо средств защиты, использующих токены CSRF, некоторые приложения используют протокол HTTP. `Referer` заголовок, чтобы попытаться защититься от атак CSRF, обычно путем проверки того, что запрос исходит из собственного домена приложения. Этот подход обычно менее эффективен и часто обходится без обходных путей.

Заголовок реферера


Заголовок HTTP `Referer` (который случайно неправильно написан в спецификации HTTP) является необязательным заголовком запроса, который содержит URL-адрес веб-страницы, связанной с запрашиваемым ресурсом. Обычно он автоматически добавляется браузерами, когда пользователь запрашивает HTTP-запрос, в том числе путем нажатия ссылки или отправки формы. Существуют различные методы, которые позволяют странице, на которую указывает ссылка, удерживать или изменять значение `Referer` заголовка. Часто это делается из соображений конфиденциальности.

Проверка `Referer` зависит от наличия заголовка

Некоторые приложения проверяют `Referer` заголовок, если он присутствует в запросах, но пропустить проверку, если заголовок опущен.

В этой ситуации злоумышленник может создать свой эксплойт CSRF таким образом, чтобы браузер пользователя-жертвы сбрасывал `Referer` заголовок в результирующем запросе. Есть несколько способов добиться этого, но самый простой - использовать тег `META` на HTML-странице, на которой размещается CSRF-атака:

```
<meta name="referrer" content="never">
```

LAB CSRF, где проверка `Referer` зависит от наличия заголовка 

Проверка `Referer` можно обойти

Некоторые приложения проверяют `Referer` заголовок наивным способом, который можно обойти. Например, если приложение проверяет, что домен в `Referer` начинается с ожидаемого значения, затем злоумышленник

может разместить это как поддомен своего собственного домена:
`http://vulnerable-website.com.attacker-website.com/csrf-attack`

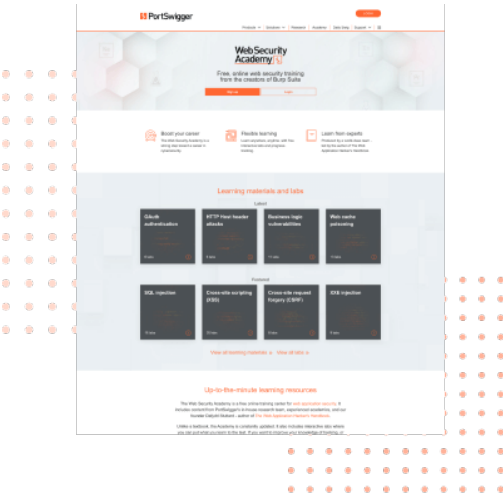
Аналогичным образом, если приложение просто проверяет, что `Referer` содержит собственное доменное имя, тогда злоумышленник может разместить необходимое значение в другом месте URL-адреса:
`http://attacker-website.com/csrf-attack?vulnerable-website.com`

Примечание
 Хотя вы можете определить это поведение с помощью Burp, вы часто обнаруживаете, что этот подход больше не работает, когда вы идете проверять свою концепцию в браузере. В попытке снизить риск утечки конфиденциальных данных таким образом, многие браузеры теперь удаляют строку запроса из `Referer` заголовка по умолчанию.
 Вы можете переопределить это поведение, убедившись, что ответ, содержащий ваш эксплойт, имеет `Referrer-Policy: unsafe-url` набор заголовков (обратите внимание, что `Referer` в этом случае написано правильно, просто для того, чтобы убедиться, что вы обращаете внимание!). Это гарантирует, что будет отправлен полный URL-адрес, включая строку запроса.

LAB

CSRF с нарушенной проверкой Referer ⓘ

Зарегистрируйтесь бесплатно, чтобы отслеживать свой прогресс в обучении



- Практикуйтесь в использовании уязвимостей на реальных целях.
- Запишите свой прогресс от ученика до эксперта.
- Посмотрите, какое ваше место в нашем Зале славы.

Уже есть аккаунт? [Авторизуйтесь здесь](#)

Люкс Burp	Уязвимости	Клиенты	Компания	Insights	PortSwigger
Сканер веб-уязвимостей Burp Suite Editions Release Notes	Межсайтовый скриптинг (XSS) межсайтовых Внедрение SQL запросов Внедрение Подделка объектов внешних XML Обход каталогов Подделка запросов на стороне сервера	Организации Тестировщики Разработчики	О PortSwigger Новости Вакансии Контакт Правовая информация Уведомление о конфиденциальности	Академии веб-безопасности Блог Исследование The Daily Swig	<div>Подписывайтесь на нас</div> <div>© 2021 PortSwigger Ltd.</div>