Ошибки путаницы привилегий: подделка межсайтовых запросов в веб-приложениях

дисциплина: Основы информационной безопасности

Подъярова Ксения Витальевна

Содержание

Список иллюстраций

Список таблиц

1 Введение

В мире кибербезопасности, где приложения становятся всё более сложными и взаимосвязанными, важно уделять внимание безопасности данных и защите от различных угроз. Одна из самых распространенных и опасных угроз для веб-приложений - подделка межсайтовых запросов (CSRF).

Важность темы: Уязвимости CSRF могут привести к серьезным последствиям, включая кражу данных, несанкционированные действия и повреждение системы.

2 Определение

Подделка межсайтовых запросов CSRF (Cross-Site Request Forgery) — уязвимость веб безопасности позволяющая злоумышленникам побуждать пользователей выполнять действия, которые они не намерены выполнять. Это позволяет злоумышленнику частично обойти политику одинакового источника (same-origin policy), предназначенную для предотвращения взаимодействия различных веб сайтов друг с другом.

Представьте, что вы вошли в свой онлайн-банкинг и оставили открытой вкладку. Злоумышленник может создать ссылку, которая выглядит безобидной, например, ссылку на сайт с новостями. Когда вы кликнете по ссылке, она автоматически выполнит запрос на перевод средств с вашего банковского счета, оставаясь незаметной для вас.

3 Виды CSRF

Существует несколько видов атак CSRF:

- **Классический CSRF:** Злоумышленник использует незащищенные формы и ссылки для отправки запросов.
- **CSRF с использованием JavaScript:** Злоумышленник использует JavaScript-код для отправки запросов в фоновом режиме.
- **CSRF с использованием HTTP-заголовков:** Злоумышленник может использовать HTTP-заголовки, чтобы выполнить запрос.

4 Как работает CSRF

Чтобы CSRF атака была возможно, должны быть соблюдены три условия:

- **Активное действие**. В приложении есть действие для вызова которого у злоумышленника есть причина. Это может быть привилегированное действие (например, изменение разрешений для других пользователей) или любое действие с данными пользователя (например, изменение собственного пароля пользователя).
- **Обработка сеансов на основе файлов cookie**. Выполнение действия включает в себя отправку одного или нескольких HTTP-запросов, и приложение полагается исключительно на файлы cookie сессии для идентификации пользователя, отправившего запрос. Другого механизма для отслеживания сеансов или проверки пользовательских запросов не существует.
- **Нет непредсказуемых параметров запроса**. Запросы, выполняющие действие, не содержат параметров, значения которых злоумышленник не может определить или угадать. Например, когда пользователь изменяет свой пароль, функция неуязвима, если злоумышленнику необходимо знать значение существующего пароля.

CSRF могут быть подвергнуты веб-приложения использующие cookies, браузерную аутентификацию или клиентские сертификаты авторизации. По сути, CSRF подвержены все веб-приложения, которые автоматически добавляют аутентификационные данные пользователя к запросу.

Либо нужно начать с того, что злоумышленник обманом заставит жертву загрузить или отправить информацию в веб-приложение. Это может произойти несколькими способами – например, через фишинговую ссылку.

Эксплойт может быть замаскирован под обычную ссылку или скрыт в теге изображения.

Вот пример атаки через обычную ссылку:

```
<a href=“вредоносная ссылка”>Unsubscribe here</a>
```

Или через тэг изображения:

```
<img src=“вредоносная ссылка” width=“0” height=“0” border=“0”>
```

5 Как работает CSRF

Предположим, приложение содержит функцию, позволяющую пользователю изменить адрес электронной почты в своей учётной записи. Когда пользователь выполняет это действие, он делает HTTP-запрос, подобный этому:

```
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 атаки:

- Действие по изменению адреса электронной почты в учётной записи пользователя представляет интерес для злоумышленника. После этого действия злоумышленник, как правило, сможет инициировать сброс пароля и получить полный контроль над учётной записью пользователя.
- Приложение использует сессию cookie для определения какой пользователь послал запрос. Других токенов или механизмов для отслеживания сеансов пользователей не существует.
- Злоумышленник может легко определить значения параметров запроса, необходимые для выполнения действия.

С учётом этих условий злоумышленник может создать веб-страницу содержащую следующий HTML-код:

```html
<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, происходит следующее:

- Страница злоумышленника инициирует HTTP-запрос к уязвимому веб-сайту.
- Если пользователь авторизовался на уязвимом веб-сайте, его браузер автоматически включит файл сессии cookie в запрос (при условии, что не используются [SameSite cookie]
- Уязвимый веб сайт обработает запрос обычным образом, расценит его как выполненный пользователем-жертвой и изменит его адрес электронной почты.

6 Как доставляют CSRF эксплойт

Механизмы доставки для атак с подделкой межсайтовых запросов практически такие же, как и для отражённых XSS. Как правило, злоумышленник размещает вредоносный HTML-код на контролируемом им веб-сайте, а затем побуждает жертву посетить этот веб-сайт. Это можно сделать предоставив пользователю ссылку на веб-сайт по электронной почте или в сообщении соцсети. Или, если атака размещена на популярном веб-сайте (например, в комментарии пользователя), они могут просто подождать, пока пользователи не посетят веб-сайт.

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

```
<img src="https://vulnerable-website.com/email/change?email=pwned@evil-user.net">
```

7 Последствия CSRF-атак

- Неавторизованные действия - изменение паролей, перевод средств, изменение настроек учетной записи.
- Ущерб репутации - психологический эффект для пользователей и компании.

8 Основные способы защиты от CSRF атак

В настоящее время для успешного поиска и использования CSRF-уязвимостей требуется обход мер по борьбе с CSRF, развёрнутых целевым веб-сайтом, браузером жертвы или и тем, и другим. Наиболее распространены следующие средства защиты:

- **CSRF токен** (Генерация уникального токена для каждой сессии и встраивание его в формы) — уникальное, секретное и непредсказуемое значение генерируемое приложением на стороне сервера и передаваемое клиенту. При попытке выполнить конфиденциальное действие, такое как отправка формы, клиент должен включить в запрос правильный CSRF токен. Из-за этого злоумышленнику очень сложно составить валидный запрос от имени жертвы.
- **SameSite cookie** (Правильная настройка полей CORS и использование куков с флагом `SameSite`.) — механизм безопасности браузера, который определяет, когда cookie веб-сайта включаются в запросы исходящие с других веб-сайтов. Поскольку для запросов на выполнение конфиденциальных действий обычно требуется сессия cookie с проверкой подлинности, соответствующие ограничения `SameSite` могут помешать злоумышленнику инициировать эти действия на других сайтах. С 2021 года Chrome по умолчанию применяет ограничения `Lex SameSite`. Поскольку это предлагаемый стандарт, мы ожидаем, что другие основные браузеры примут это поведе-

ние в будущем.

- **Проверка на основе Referer** (Проверка заголовка `Referer` на наличие допустимого источника.) — некоторые приложения используют HTTP-заголовок `Referer` для защиты от CSRF атак. Обычно путём проверки того, что запрос исходит оз собственного домена приложения. Как правило, это менее эффективно, чем проверка CSRF токена.

8.1 Anti-CSRF токены

Токены (или synchronizer token) – это способ защиты со стороны сервера. Сервер генерирует случайный уникальный токен для браузера пользователя и проверяет его для каждого запроса.

Токен находится в скрытом поле, должен быть непредсказуемым случайным числом и иметь небольшое время жизни, без возможности переиспользования.

Токен должен удовлетворять следующим условиям:

- быть уникальным в пределах каждой операции;
- использоваться один раз;
- иметь размер устойчивый к подбору;
- генерироваться криптографически стойким генератором псевдослучайных чисел;
- иметь ограниченное время жизни.

8.2 Использование двух токенов

Смысл этого метода в том, что используются два токена: первый сохраняется в cookies, а второй — в одном из параметров ответа.

В таком случае сервер, получая один из небезопасных запросов, должен проверить оба токена.

8.3 Использование флага Same-Site в cookies

Этот флаг помечает куки для определенного домена.

Таким образом проверяется источник запроса, и его не получится выполнить с мошеннического сайта.

Этот флаг поддерживает большинство браузеров. Его стоит использовать как часть общей стратегии защиты от CSRF атак.

9 Заключение

Ошибки путаницы привилегий, такие как CSRF, являются серьезной угрозой для безопасности веб-приложений. Важно уделять внимание защите от этой атаки и использовать соответствующие меры безопасности для предотвращения ее возникновения.