

Раздел

Фреймы и окна

Навигация по уроку

Идея

Демонстрация

Примеры слабой защиты

Заголовок X-Frame-Options

Отображение с ограниченными возможностями

Атрибут cookie: samesite


Итого

Комментарии

Поделиться



Редактировать на GitHub

 → Фреймы и окна 1-го июля 2019

# Атака типа clickjacking

Атака типа clickjacking (англ. «захват клика») позволяет вредоносной странице кликнуть по сайту-жертве от имени посетителя.

Многие сайты были взломаны подобным способом, включая Twitter, Facebook, PayPal и другие. Все они, конечно же, сейчас защищены.

## Идея

Идея этой атаки очень проста.

Вот как clickjacking-атака была проведена на Facebook:

1. Посетителя заманивают на вредоносную страницу (неважно как).
2. На странице есть ссылка, которая выглядит безобидно (например, «Разбогатеет прямо сейчас» или «Нажми здесь, это очень смешно»).
3. Поверх этой ссылки вредоносная страница размещает прозрачный `<iframe>` с `src` с сайта facebook.com таким образом, что кнопка «like» находится прямо над этой ссылкой. Обычно это делается с помощью `z-index` в CSS.
4. При попытке клика на эту ссылку посетитель на самом деле нажимает на кнопку.

## Демонстрация

Вот как выглядит вредоносная страница. Для наглядности `<iframe>` полупрозрачный (на реальных вредоносных страницах он полностью прозрачен):

```
1 <style>
2   iframe { /* ифрейм с сайта-жертвы */
3     width: 400px;
4     height: 100px;
5     position: absolute;
6     top:0; left:-20px;
7     opacity: 0.5; /* в реальности opacity:0 */
8     z-index: 1;
9   }
10 </style>
11
12 <div>Нажми, чтобы разбогатеть:</div>
13
14 <!-- Url с сайта-жертвы -->
15 <iframe src="/clickjacking/facebook.html"></iframe>
16
17 <button>Нажмите сюда!</button>
18
19 <div>...И всё будет супер (у меня, хакера)!</div>
```

Полная демонстрация атаки:

Результат    facebook.html    index.html



Нажми, чтобы разбогатеть:  
  
...И всё будет супер (у меня, хакера)!

Раздел

Фреймы и окна

Навигация по уроку

Идея

Демонстрация

Примеры слабой защиты

Заголовок X-Frame-Options

Отображение с  
ограниченными  
возможностями

Атрибут cookie: samesite

Итого

Комментарии

Поделиться



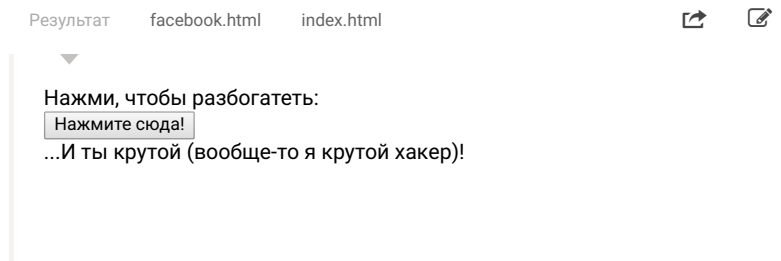
Редактировать на GitHub



Здесь у нас есть полупрозрачный `<iframe src="facebook.html">`, и в примере мы видим его висющим поверх кнопки. Клик на кнопку фактически кликает на ифрейм, но этого не видно пользователю, потому что ифрейм прозрачный.

В результате, если пользователь авторизован на сайте Facebook («Запомнить меня» обычно активировано), то он добавляет «лайк». В Twitter это будет кнопка «читать», и т.п.

Вот тот же пример, но более приближенный к реальности с `opacity: 0` для `<iframe>`:



Всё, что нам необходимо для атаки — это расположить `<iframe>` на вредоносной странице так, чтобы кнопка находилась прямо над ссылкой. Так что пользователь, кликающий по ссылке, на самом деле будет нажимать на кнопку в `<iframe>`. Обычно это можно сделать с помощью CSS-позиционирования.

#### **Clickjacking-атака для кликов мыши, а не для клавиатуры**

Эта атака срабатывает только на действия мыши (или аналогичные, вроде нажатия пальцем на мобильном устройстве).

Клавиатурный ввод гораздо сложнее перенаправить. Технически, если у нас есть текстовое поле для взлома, мы можем расположить ифрейм таким образом, чтобы текстовые поля перекрывали друг друга. Тогда посетитель при попытке сфокусироваться на текстовом поле, которое он видит на странице, фактически будет фокусироваться на текстовом поле внутри ифрейма.

Но есть одна проблема. Всё, что посетитель печатает, будет скрыто, потому что ифрейм не виден.

Обычно люди перестают печатать, когда не видят на экране новых символов.

## Примеры слабой защиты

Самым старым вариантом защиты является код JavaScript, запрещающий открытие страницы во фрейме (это называют «framebusting»).

Выглядит он вот так:

```
1 if (top !== window) {
2   top.location = window.location;
3 }
```

В этом случае, если окно обнаруживает, что оно открыто во фрейме, оно автоматически располагает себя сверху.

Этот метод не является надёжной защитой, поскольку появилось множество способов его обойти. Рассмотрим некоторые из них.

### Блокировка top-навигации

Мы можем заблокировать переход, вызванный сменой `top.location` в обработчике события `beforeunload`.

Внешняя страница (принадлежащая хакеру) устанавливает обработчик на это событие, отменяющий его, например, такой:

Раздел

Фреймы и окна

Навигация по уроку

Идея

Демонстрация

Примеры слабой защиты

Заголовок X-Frame-Options

Отображение с ограниченными возможностями

Атрибут cookie: samesite

Итого

Комментарии

Поделиться



Редактировать на GitHub



```
1 window.onbeforeunload = function() {  
2     return false;  
3 };
```

Когда `iframe` пытается изменить `top.location`, посетитель увидит сообщение с вопросом действительно ли он хочет покинуть эту страницу. В большинстве случаев посетитель ответит отрицательно, поскольку он не знает об ифрейме: всё, что он видит – это верхнюю страницу, которую нет причин покидать. Поэтому `top.location` не изменится!

В действии:

Результат    iframe.html    index.html



После клика на кнопку пользователь получает "странный" вопрос о том, хочет ли он покинуть данную страницу.

Скорее всего, он ответит "нет", и защита ифрейм будет взломана.

Добавить "защищённый" ифрейм

## Атрибут «sandbox»

Одним из действий, которые можно ограничить атрибутом `sandbox`, является навигация. Соответственно ифрейм внутри `sandbox` не изменит `top.location`.

Поэтому мы можем добавить ифрейм с `sandbox="allow-scripts allow-forms"`. Это снимет некоторые ограничения, разрешая при этом использование скриптов и форм. Но мы опускаем `allow-top-navigation`, чтобы изменение `top.location` было запрещено.

Вот код этого примера:

```
1 <iframe sandbox="allow-scripts allow-forms" src="facebo
```

Есть и другие способы обойти эту простую защиту.

## Заголовок X-Frame-Options

Заголовок `X-Frame-Options` со стороны сервера может разрешать или запрещать отображение страницы внутри фрейма.

Это должен быть именно HTTP-заголовок: браузер проигнорирует его, если найдёт в HTML-теге `<meta>`. Поэтому при `<meta http-equiv="X-Frame-Options">` ничего не произойдёт.

Заголовок может иметь 3 значения:

### DENY

Никогда не показывать страницу внутри фрейма.

### SAMEORIGIN

Разрешить открытие страницы внутри фрейма только в том случае, если родительский документ имеет тот же источник.

### ALLOW-FROM domain

Разрешить открытие страницы внутри фрейма только в том случае, если родительский документ находится на указанном в заголовке домене.

Например, Twitter использует `X-Frame-Options: SAMEORIGIN`.

Вот результат:

```
1 <iframe src="https://twitter.com"></iframe>
```

Раздел

Фреймы и окна

Навигация по уроку

Идея

Демонстрация

Примеры слабой защиты

Заголовок X-Frame-Options

Отображение с ограниченными возможностями

Атрибут cookie: samesite

Итого

Комментарии

Поделиться



Редактировать на GitHub



В зависимости от того, какой браузер вы используете, `iframe` выше либо будет пустым, либо оповестит вас о том, что его невозможно отобразить.

## Отображение с ограниченными возможностями

У заголовка `X-Frame-Options` есть побочный эффект. Другие сайты не смогут отобразить нашу страницу во фрейме, даже если у них будут на то веские причины.

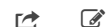
Так что есть другие решения... Например, мы можем «накрыть» страницу блоком `<div>` со стилями `height: 100%; width: 100%;`, чтобы он перехватывал все клики. Этот `<div>` будем убирать, если `window == top` или если мы поймём, что защита нам не нужна.

Примерно так:

```
1 <style>
2   #protector {
3     height: 100%;
4     width: 100%;
5     position: absolute;
6     left: 0;
7     top: 0;
8     z-index: 99999999;
9   }
10 </style>
11
12 <div id="protector">
13   <a href="/" target="_blank">Перейти к сайту</a>
14 </div>
15
16 <script>
17   // Здесь будет отображаться ошибка, если верхнее окно
18   // а здесь будет код, если всё в порядке
19   if (top.document.domain == document.domain) {
20     protector.remove();
21   }
22 </script>
```

Демонстрация:

Результат    iframe.html    index.html



Этот текст всегда виден. Но если страница была открыта внутри документа, принадлежащего другому домену, `div` поверх неё предотвратит любые действия.  
В данном случае клик не сработает

## Атрибут cookie: samesite

Атрибут `samesite` также может помочь избежать clickjacking-атаки.

Файл куки с таким атрибутом отправляется на сайт только в том случае, если он открыт напрямую, не через фрейм или каким-либо другим способом. Подробно об этом – в главе [Куки](#), [document.cookie](#).

Если сайт, такой как Facebook, при установке авторизирующей куки ставит атрибут `samesite`:



1 Set-Cookie: authorization=secret; `samesite`



... Тогда такие куки не будут отправляться, когда Facebook будет открыт в ифрейме с другого сайта. Так что атака не удастся.

Атрибут `samesite` не играет никакой роли, если куки не используются. Так что другие веб-сайты смогут отображать публичные, не требующие авторизации, страницы в ифрейме.

Однако, это даёт возможность в некоторых ситуациях осуществить `clickjacking`-атаку, например, на сайт для анонимных опросов, который предотвращает повторное голосование пользователя путём проверки IP-адреса. Он останется уязвимым к атаке, потому что не аутентифицирует пользователей с помощью куки.

## Итого

Атака `clickjacking` — это способ хитростью «заставить» пользователей кликнуть на сайте-жертве, без понимания, что происходит. Она опасна, если по клику могут быть произведены важные действия.

Хакер может разместить ссылку на свою вредоносную страницу в сообщении или найти другие способы, как заманить пользователей. Вариантов множество.

С одной стороны — эта атака «неглубокая», ведь хакер перехватывает только один клик. Но с другой стороны, если хакер знает, что после этого клика появятся другие элементы управления, то он может хитростью заставить пользователя кликнуть на них.

Этот вид атаки довольно опасен, ведь при разработке интерфейсов мы не предполагаем, что хакер может кликнуть от имени пользователя. Поэтому уязвимости могут быть обнаружены в совершенно неожиданных местах.

- Для защиты от этой атаки рекомендуется использовать `X-Frame-Options: SAMEORIGIN` на страницах или даже целиком сайтах, которые не предназначены для просмотра во фрейме.
- Или, если мы хотим разрешить отображение страницы во фрейме и при этом оставаться в безопасности, то можно использовать перекрывающий блок `<div>`.

Проводим [курсы по JavaScript и фреймворкам](#). ✕

## Комментарии

перед тем как писать...

- Раздел
- Фреймы и окна
- Навигация по уроку
- Идея
- Демонстрация
- Примеры слабой защиты
- Заголовок `X-Frame-Options`
- Отображение с ограниченными возможностями
- Атрибут cookie: `samesite`
- Итого
- Комментарии
- Поделиться
- 
- Редактировать на GitHub