

Еще один блог веб-разработчика

Заметки на полях о программировании и веб-разработке

info@webdevblog.ru

Разработка современных веб сервисов на Django и Vue.js

Следить за блогом:



ОБ АВТОРЕ

Веб безопасность

Уязвимость Open Redirect

11.09.2020 Editorial Team 0 Комментариев 3 021 просмотров

Spread the love

0

Поделиться

Перевод статьи: [s0cket7](#) — [Open Redirect Vulnerability](#)

TL;DR

Этот пост рассказывает об уязвимостях Open/Unvalidated Redirects и Forwards. Вы узнаете, что они из себя представляют, как их найти, как их использовать и как их исправить.

Что стоит отметить

- Требуются базовые знания о HTTP.
- Если у вас есть знания в области

Поиск



Свежие записи

[Разрешения в Django Rest Framework](#)

[Докеризация Django с Postgres, Unicorn, и Traefik](#)

[Валидация полей Json в моделях Django](#)

[Спецификация ECMAScript 2021 для JavaScript приближается к финишу](#)

[5 советов по повышению производительности Vue](#)

Рубрики

[CSS](#) (3)

[JavaScript](#) (171)

программирования, это тоже поможет, потому что мы рассмотрим некоторый код в этом посте.

Введение

Что такое перенаправление (Redirects)?

Перенаправление означает, что веб-сайт может перенаправить запрос ресурсов на другой URL/конечную точку. Предположим, что вы отправляете запрос на **apple.com**, а **apple.com** может перенаправить вас на другой сайт (**new-apple.com**), так что вы окажетесь на **new-apple.com**, даже если исходный запрос был сделан для **apple.com**. Это называется «перенаправление». Существуют различные типы перенаправлений в HTTP.

Стандартные коды статусов перенаправлений — 3xx

- [300 Multiple Choices](#) множество выбора
- [301 Moved Permanently](#) перемещено навсегда
- [302 Found](#) найдено
- [303 See Other](#) смотреть другое
- [304 Not Modified](#) не изменялось
- [305 Use Proxy](#) использовать прокси
- [307 Temporary Redirect](#) временное перенаправление
- [308 Permanent Redirect](#) постоянное

[Python](#) (117)

[Алгоритмы](#) (38)

[Базы данных](#) (5)

[Веб безопасность](#) (16)

[Другие](#) (16)

[Новости](#) (6)

[Метки](#)

[asyncio](#) [axios](#) [celery](#)

[Composition](#) [Composition](#)

[API](#) [Dataclasses](#) [django](#)

[docker](#) [DOM](#) [DRF](#) [FSM](#) [gensim](#)

[GraphQL](#) [iterators](#) [jest](#) [lerna](#)

[logging](#) [metaclass](#) [MongoDB](#) [news](#)

[nginx](#) [nlp](#) [nuxt.js](#) [patterns](#)

[pipenv](#) [Proxy](#) [RASA](#) [Redis](#) [solid](#)

[sort](#) [spread](#) [tailwindcss](#) [TDD](#)

[threading](#) [uwsgi](#) [vue-router](#)

[vue.js](#) [vuex](#) [webpack](#)

[wordpress](#) [Алгоритмы](#) [ООП](#)

[Собеседования](#)

[Тестирование](#) [чатботы](#)

Популярные статьи

[Jupyter Notebook для начинающих: учебник](#) - 136 673 Просмотры

[Как перебрать словарь в Python](#) - 135 443 Просмотры

[Как использовать в Python](#)

перенаправление

Перенаправление может происходить на стороне сервера (**Server-Side**) или на стороне клиента (**Client-Side**).

Server-Side : Запрос на перенаправление отправляется на сервер, затем сервер уведомляет браузер о перенаправлении на URL-адрес, указанный в ответе.

Client-Side : Браузер получает уведомление о перенаправлении на указанный URL-адрес напрямую, без вмешательства сервера.

Что такое Open Redirects?

Open redirect (Открытое перенаправление) это то, что написано в названии: Открытое (**Open**) перенаправление (**Redirects**) на любой веб-сайт.

Почему это проблема?

Ну, это плохо с самого начала, подумайте об этом на мгновение, что если `apple.com`, **ДОВЕРЕННЫЙ** веб-сайт, позволяет вам перенаправить на любой другой веб-сайт. Тогда злонамеренный пользователь может просто перенаправить `apple.com` на `attacker.com` , и люди все время будут думать что они находятся на `apple.com` , полагая, что ему доверяют, но на самом деле это не так. Поэтому разрешать перенаправления на любой веб-сайт без проверки или без надлежащего уведомления пользователя — это плохо.

[лямбда-функции](#) - 104 736
Просмотры

[Использование middleware во Vue](#) - 102 176
Просмотры

[Введение в потоки в Python](#) - 95 610
Просмотры

[Разбираемся с настройкой Webpack для любого проекта](#) - 89 369
Просмотры

[Создание Django API используя Django Rest Framework часть 1](#) - 83 485
Просмотры

[Что такое CORS](#) - 76 837
Просмотры

[Логирование в Python](#) - 68 408
Просмотры

[Обзор Async IO в Python 3.7](#) - 53 520
Просмотры



[Новости из мира Python](#)

Объяснение

Допустим, есть «хорошо известный» веб-сайт — `https://example.com/`. И давайте предположим, что есть ссылка, как

```
https://example.com  
/signup?redirectUrl=https://example.com  
/login
```

Эта ссылка на страницу регистрации, после регистрации вы будете перенаправлены на `https://example.com/login`, который указан в URL-адресе в GET параметре `redirectUrl`.

Что произойдет, если мы изменим `example.com/login` на `attacker.com`?

```
https://example.com  
/signup?redirectUrl=https:  
//attacker.com/
```

Посещая этот URL, если после регистрации мы будем перенаправлены на `attacker.com`, это означает, что у нас есть открытая уязвимость перенаправления. Это классический открытый редирект и часто используется для фишинга.

Почему это происходит?

Это происходит из-за недостаточной проверки перенаправления в серверной части, что

означает, что сервер неправильно проверяет, находится ли URL-адрес перенаправления в своем белом списке или нет. Вот несколько примеров уязвимого кода

PHP (Server-Side)

```
<?php
    $url_to_redirect =
$_GET['redirect_url'];
    header('Location: ' .
$url_to_redirect);
    die();
```

Здесь код php слепо захватывает URL-адрес из параметра `redirect_url` и перенаправляет на этот URL-адрес, используя HTTP-заголовок `Location`.

Java (Server-Side)

```
response.sendRedirect(request.getParameter("u"));
```

Здесь jsp-страница берет URL-адрес из параметра `u` и слепо перенаправляет его на указанный URL-адрес.

Javascript (Client-Side)

```
window.location.href =
```

```
"https://attacker.com";
```

Мы можем назначить строку URL для `location.href` объекта `window`. Это приведет к перенаправлению. Если там нет проверок, значит, это уязвимость.

HTML (Client-Side)

```
<meta http-equiv="refresh" content="0;
URL='http://attacker.com/'" />
```

Метатеги HTML могут обновлять сайт с заданным URL-адресом в качестве содержимого (`content`), а также вы можете указать время задержки перед обновлением.

Как обнаружить эту уязвимость?

- Посетите каждую конечную точку цели, чтобы найти параметры «`redirect`».
- Просмотрите истории прокси. Обязательно используйте фильтры.
- Простой перебор (Bruteforcing) тоже может помочь найти уязвимость
- Вы можете раскрыть многие конечные точки, просто прочитав код JavaScript.
- Google — твой друг, поищите в поисковике, пример запроса: `inurl:redirectUrl=http site:target.com`

- Изучите и проанализируйте целевое приложение на наличие потребности перенаправления , например, перенаправление на панель мониторинга после входа в систему или что-то в этом роде.

Немного магии

- Тест на базовую модификацию URL как `target.com/?redirect_url=https://attacker.com.`
- Попробуйте с двойными косыми чертами `target.com//attacker.com.`
- Попробуйте `target.com/@attacker.com.` В этом случае интерпретация будет такой: `target.com` — это имя пользователя, а `attacker.com` — это домен.
- Тест на интерпритацию javascript `javascript:confirm(1).`
- Попробуйте `target.com/?image_url=attacker.com/.jpg,` если загружаются ресурсы изображений.
- Попробуйте IP-адрес вместо имени домена.
- Вы можете пойти дальше с точки зрения представления IP в десятичном, шестнадцатеричном или восьмеричном виде.
- Вы также можете попробовать `target.com/?redirect_url=target.com.attacker.com` , чтобы обойти слабые реализации регулярных выражений.

- Китайский разделитель 。 в виде точки
— `https://attacker%E3%80%82com`.
- Тест на обратную строку
`unicode("\u202e") target.com@%E2%80`
`%AE@attacker.com`.
- Без слешей `https:attacker.com`.
- Обратные слешы `http:/\`
`/\attacker.com` или `https:/\attacker.com`.
- Разные домены `redirect_url=.jp` В
результате перенаправляется
на `target.com.jp` который отличаются от
`target.com`.
- Попробуйте юникод безумие (включая
смайлики) `tAرget.com` или `Aтtacker.com('A'`
это `"\uD835\uDC00")`.

Использование уязвимости

Фишинг

Предположим, что целью является `example.com`.
Он имеет страницу восстановления пароля по
адресу `example.com/forgot-password`. Вы вводите
адрес электронной почты и нажимаете кнопку
« **Forgot Password** », и вам будет отправлено
письмо со ссылкой для сброса пароля, и эта
ссылка может выглядеть следующим образом


```
https://example.com/reset-password/some-  
random-token?redirect=https:  
//example.com/login
```

Если мы вмешаемся в параметр `redirect` и изменим его на

```
https://example.com/reset-password/some-  
random-token?redirect=https:  
//attacker.com/login
```

Это перенаправляет пользователя на злую страницу входа в систему, если исходная и пользователь могут быть фишинговыми.

Сцепление с SSRF

Если вы ничего не знаете о SSRF, вы можете поискать об этом в Google. В любом случае, допустим, у нас есть цель — `example.com`, и вы обнаружили ошибку SSRF на `https://example.com/?get-image=https://images.example.com/cat.jpg`. По сути, мы хотим заставить сервер сделать запрос от нашего имени. Мы можем изменить значение параметра `get-image`, и сервер сделает запрос к этой конечной точке. Но в этом случае он ограничивает запросы своим собственным (под) доменом (ами), например `images.example.com`. Это означает, что вы можете сделать запрос на `*.example.com` (Любой поддомен) в качестве сервера и не можете получить доступ ни к чему

вне этой области. Допустим, вы также обнаружили ошибку Open Redirect на `https://test.example.com/redirect_url=https://www.example.com/`, теперь вы можете связать их вместе, чтобы заставить SSRF работать для любого домена, например

Сначала мы используем уязвимость Open Redirect, изменяя параметр `redirect_url`

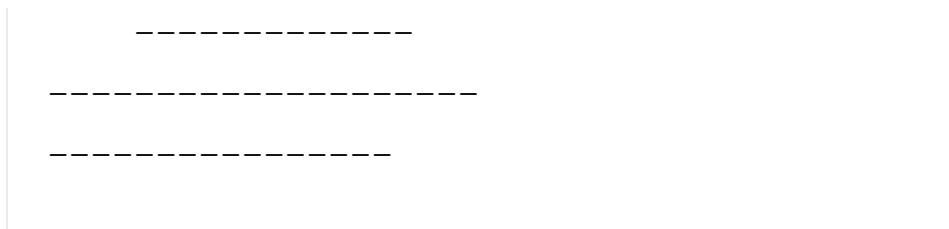
```
https://test.example.com  
/?redirect_url=https://www.attacker.com/
```

Затем мы можем использовать это с **SSRF**, добавив открытый URL-адрес **Open Redirect** сверху в качестве параметра `get-image`.

```
https://example.com/?get-image=https:  
//test.example.com/?redirect_url=https:  
//www.attacker.com/
```

Теперь это заставит сервер сделать запрос на один из его собственных поддоменов (`test.example.com`), а затем он будет перенаправлен на `attacker.com`, чтобы сработала ошибка SSRF.

```
-----  
-----  
-----  
==> | get-image | ==> | test.example.com  
| ==> { REDIRECTS } ==> | attacker.com |
```



Подобно этому, открытые перенаправления могут использоваться по-разному, я не могу обсуждать все их, это зависит только от вашей цели.



“Вы ограничены только вашим воображением.”

Защита от уязвимости

- Используйте перенаправления, только если нет другого варианта.
- Если вы хотите их использовать, убедитесь, что вы правильно проверяете домены через белый список.
- Вы также можете использовать hmas, если хотите, но с этим довольно легко справиться (атаки с расширением длины — [length extension attacks](#)), так что не усложняйте вещи, используйте правильный белый список.

Немного полезных ссылок

Список всех уникальных отчетов об Open Redirect Reports на HackerOne, которые я смог найти на

первых 20 страницах результатов Google.

- [\[Report-226408\] Open Redirect on Shopify](#)
- [\[Report-211213\] Open Redirect on Nextcloud](#)
- [\[Report-246897\] Open Redirect on Twitter](#)
- [\[Report-103772\] Open Redirect on Shopify](#)
- [\[Report-309058\] Open Redirect on WordPress](#)
- [\[Report-260744\] Open Redirect and XSS on Twitter](#)
- [\[Report-320376\] Open Redirect on HackerOne](#)
- [\[Report-111968\] Interstitial redirect bypass / Open Redirect on HackerOne Zendesk Session](#)
- [\[Report-244721\] Open Redirect on Mail.Ru](#)
- [\[Report-236599\] Open Redirect on ExpressionEngine](#)
- [\[Report-299403\] Open Redirect on HackerOne](#)
- [\[Report-239503\] Open Redirect & Information Disclosure on HackerOne](#)
- [\[Report-210875\] Open Redirect via Host Header](#)
- [\[Report-119236\] Open Redirect on Uber](#)
- [\[Report-126203\] Open Redirect on Uber](#)
- [\[Report-28865\] Open Redirect Filter bypass on HackerOne](#)
- [\[Report-144525\] Open Redirect bypass on New Relic](#)
- [\[Report-104087\] Open Redirect bypass using svg on Slack](#)
- [\[Report-179568\] Open Redirect via window.opener on Open-Xchange](#)
- [Open Redirect to RCE on Google Hangouts Electron app & RCE Tweet](#)

Шпаргалки

Этот список может быть не уникальным, очистите дубликаты и отсортируйте их.

- [EdOverflow Open Redirect CheatSheet](#)
- [cujanovic Open Redirect Payloads](#)
- [fuzzdb Redirect Url Templates](#)
- [ak1t4 Open Redirect Payloads](#)
- [random robbie Open Redirect Payloads](#)
- [OWASP Unvalidated Redirects and Forwards Cheat Sheet](#)


Ресурсы

- [Open Redirect by zseano](#)
- [Open Redirect by SI9INT](#)
- [Using Burp to Test for Open Redirections](#)
- [Detectify Unvalidated Redirects and Forwards](#)
- [Open redirects that matter](#)
- [Client Side URL Redirections](#)
- [Fun With Redirects 2010](#)

Вот и все, ребята. Спасибо за чтение. Хорошего дня 😊

– s0cket7

Была ли вам полезна эта статья?



 [1 / 5]

Spread the love

0
Поделиться

← [Объяснение @classmethod и @staticmethod в Python](#)

[Методы уклонения от Web Application Firewall \(WAF\) →](#)

 Подписаться 

Соединить с

[авторизуйтесь](#)



B *I* U    “ ” </>  {} [+]



0 КОММЕНТАРИЙ



Copyright © 2019-2021 [Еще один блог веб-разработчика](#).

Перепечатывание материалов сайта с указанием первоисточника приветствуется.

Если вы считаете что публикация какой то статьи в этом блоге нарушает чьи-то авторские права, [напишите нам об этом](#).

If you think that the publication of any article on this blog violates someone's copyright, [email to us about it](#)

