Безопасность Web приложений

Same Origin Policy

Same Origin Policy

Две страницы принадлежат одному Origin, если у них совпадают протокол, домен и порт. Доступ из одного Origin в другой - запрещен.

- Доступ к Cookie
- Доступ к LocalStorage
- Доступ к объекта DOM, вызов JavaSript

- Выполнение AJAX запросов (*)

SOP - DOM

```
<html>
    <body>
        <iframe id="tgt" src="https://another-site.ru/">
        </iframe>
    </body>
    <script>
        $('#tgt').on('load', function() {
            var innerDoc = $('#tgt')[0].contentDocument;
            var cookie = innerDoc.cookie;
            //...
        })
    </script>
```

```
</html>
```


Cross Origin Request

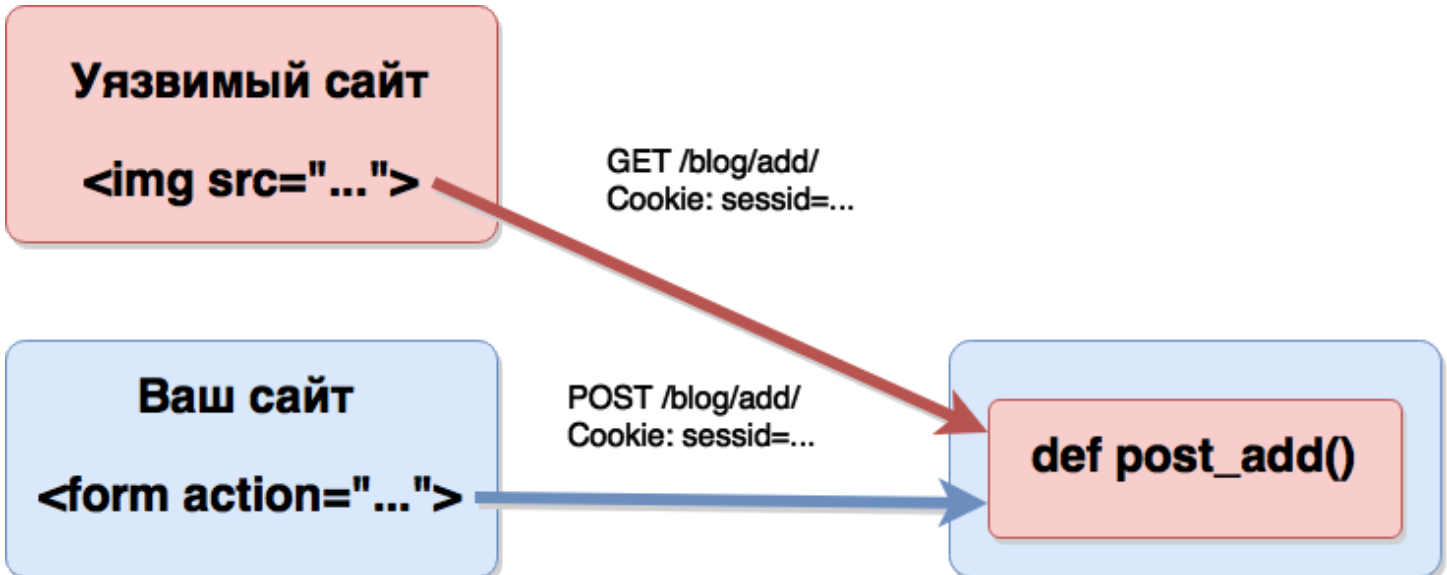
- `<img src="https://any-site.ru/logout"/>`
- `<link href="https://any-site.ru/logout"/>`
- `<script src="https://any-site.ru/logout"></script>`
- `<form action="https://any-site.ru/"></form>`
- `$.ajax({"url":"https://any-site.ru/uinfo"})`

Запрос к домену `any-site.ru` будет содержать куки!!!

Но вызывающая сторона не получит содержимое ответа.

CSRF

Cross Site Resource Forgery



Методы борьбы с CSRF

- Проверка метода `@require_POST`
- Проверка заголовка `Referer`
- Проверка CSRF-токенов
- (*) Проверка заголовка `X-Requested-With`

```
<form method="POST" action="/blog/add">
    {% csrf_token %}
</form>
```

JSONP

Запрос

```
<script src="https://partner.ru/?callback=func123">
</script>
```

Ответ

```
func123({"news": ["blablabla", "more"]})
```

Вы не контролируете КТО делает JSONP вызов, нельзя использовать для передачи личных данных пользователя.

AJAX - CORS

Запрос

```
GET /messages HTTP/1.1
Host: partner.ru
Cookie: sessid=a3b4d510c8712403a86f978cbb3
Origin: https://yoursite.ru
```

Ответ

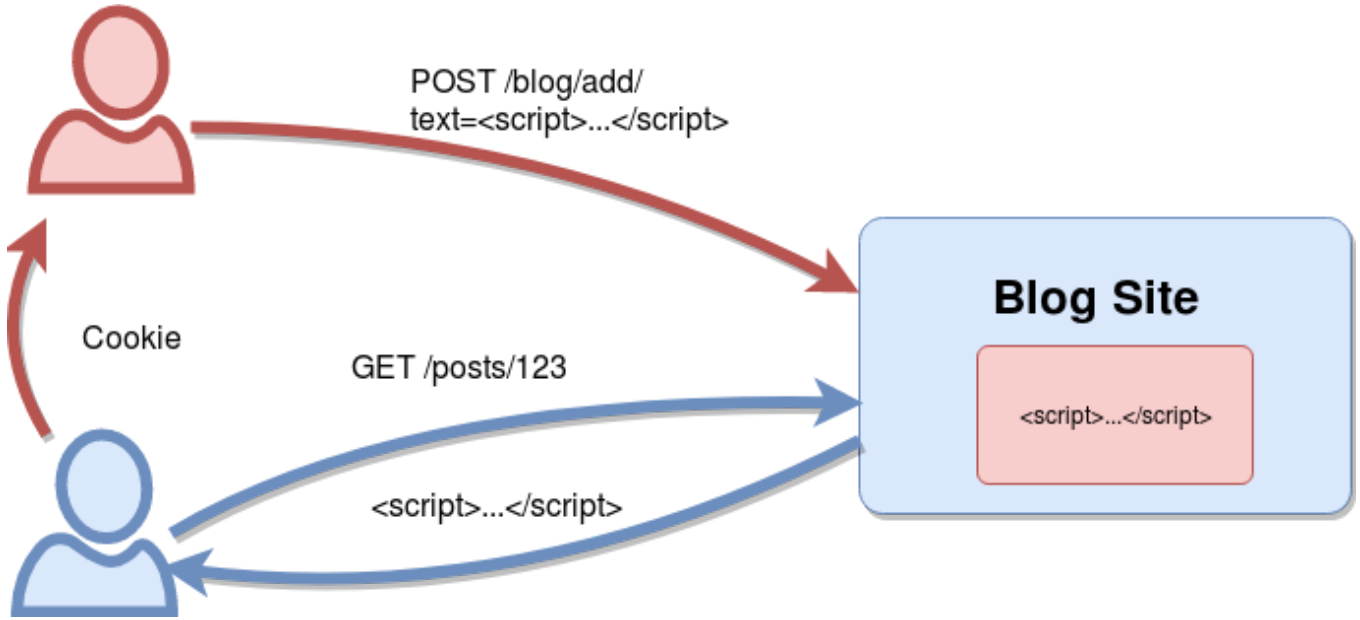
```
HTTP/1.1 200 OK
Content-Type: application/json
Access-Control-Allow-Origin: https://yoursite.ru
Access-Control-Allow-Credentials: true
```

XSS

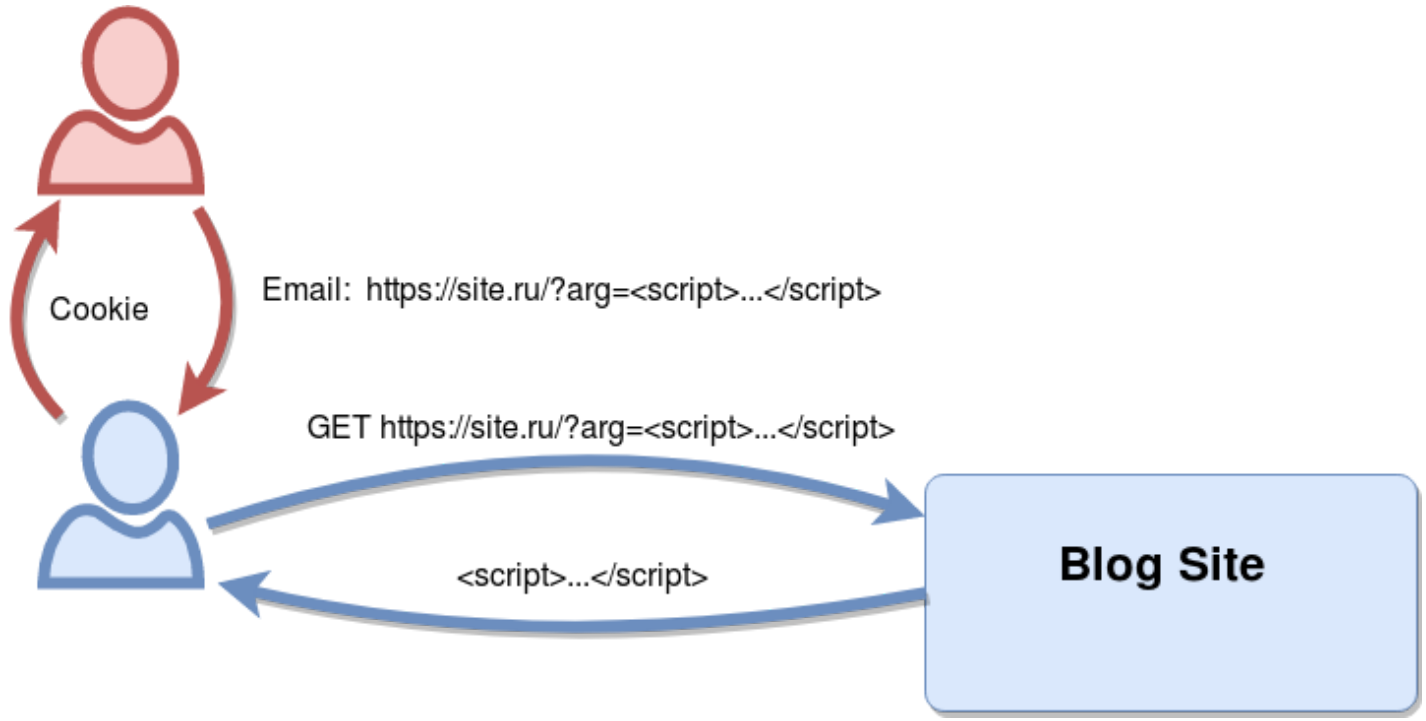
XSS - Cross Site Scripting

XSS - использование непроверенных данных в коде страницы. Позволяет злоумышленнику разместить вредоносный JavaScript код на вашей странице и выполнить его на компьютере пользователя. Злоумышленник получает доступ к данным пользователя.

Passive XSS


POST /blog/add/
text=<script>...</script>
Blog Site
<script>...</script>
Cookie
GET /posts/123
<script>...</script>

Active XSS


Cookie
Email: https://site.ru/?arg=<script>...</script>
GET https://site.ru/?arg=<script>...</script>
<script>...</script>
Blog Site

Способы борьбы

Обработка выходных данных:

- Валидация (проверка содержимого)
- Экранирование (эскейпинг) опасных символов (`< > ” ’`)
- Очистка входных данных

Экранирование

```
<h2>{{ user.nick|escapehtml }}</h2>
<p post_id="{{request.GET.post_id|escapejs}}">
    {{post.text|escapehtml|linebreaks}}
</p>
<script>
    var POST_ID = {{ request.GET.post_id|escapejs }};
    var USER_NICK = "{{ user.nick|escapejs }}";
    var USER_NICK = `{{ user.nick|escapejs }}`;
</script>
```

* Шаблонизатор - сферический в вакууме

Валидация / очистка

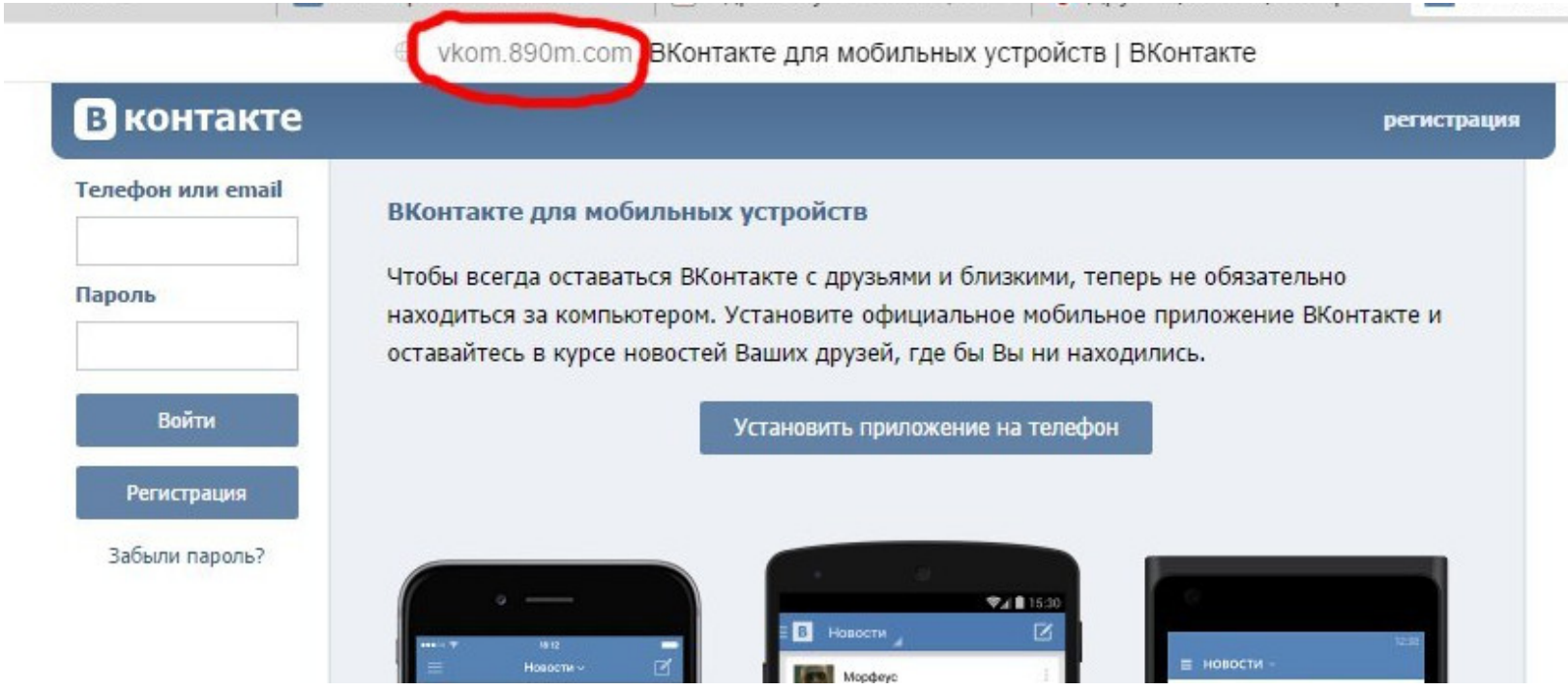
- Разрешать только простые типы int/date/text
- Не загружать и не хранить HTML вообще

```
<img src=”...” onerror=”alert(1)”/>
```

- Использовать более простые форматы: Wiki / Markdown

Fishing

Fishing


vkom.890m.com ВКонтакте для мобильных устройств | ВКонтакте
ВКонтакте
регистрация
Телефон или email
Пароль
Войти
Регистрация
Забыли пароль?
ВКонтакте для мобильных устройств
Чтобы всегда оставаться ВКонтакте с друзьями и близкими, теперь не обязательно находиться за компьютером. Установите официальное мобильное приложение ВКонтакте и оставайтесь в курсе новостей Ваших друзей, где бы Вы ни находились.
Установить приложение на телефон
Новости
Морфеус
НОВОСТИ

Open Redirect

Как отправить пользователя на фишинговую страницу ?

Сокращатели URL-ов

```
https://bit.ly/hzchtotam
```

Open redirect

```
https://site.ru/login?next=https://fake-site.ru
```

Injection

SQL Injection

```
sql = "SELECT * FROM posts WHERE id = " \
    + str(request.GET['post_id'])

sql = "SELECT * FROM posts WHERE id = {id}" \
    .format(id=request.GET['post_id'])

cursor.execute(sql)
```

Уязвимость:

```
https://site.ru/post/?post_id=1;DROP TABLE posts;
```


Как избежать?

Как избежать ?

ORM

```
Posts.objects.get(request.GET['post_id'])
```

Эскейпинг

```
cursor.execute(
    "SELECT * FROM posts WHERE id = %s",
    [ request.GET['post_id'] ])
```


Иногда это не работает

Иногда это не работает

```
SELECT * FROM posts WHERE id IN ({ids});

SELECT * FROM posts ORDER BY {order_column};

SELECT * FROM {table_name} WHERE id = {id};
```


В таком случае

Приведение типов

```
ids = [ int(i) for i in request.GET.getlist('post_id') ]
sql = "SELECT * FROM posts WHERE id IN ({ids})" \
    .format(ids=','.join(ids))
```

Проверка возможнных значений

```
order = request.GET['order']
valid = {'rating', 'added_at'}
if order in valid:
    sql = "SELECT * FROM posts ORDER BY " + order
```


Command Injection

```
month = request.GET['month']
cmd = "ls /home/backups/" + month
output = subproces.check_output(cmd, shell=True)
# ...
```

Уязвимость

```
http://site.ru/backups/?month=may;cat+/etc/passwd
```

```
http://site.ru/backups/?month=../../../../etc/passwd
```


Послушать занимательное