

# Атакуем крупные порталы и современные технологии

Сергей Белов,  
Аудитор ИБ, Digital Security

## # whoami

- Работаю в Digital Security
  - Аудитор ИБ
  - ZeroNights
- Поиск уязвимостей (Yandex, Mail.ru, VK.com, Google...)
- Пишу на Хабрахабр, журнал «Хакер» - BeLove
- Спикер – OWASP RU/Europe, BlackHat 2014, Hack in Paris 2014, ZeroNights, CodeFest, РИТ++ и другие
- Специализируюсь на веб-технологиях

## О чем говорим следующие 30 минут?

- Уязвимости на крупных порталах и в современных технологиях (и подходах)
- Технологии, которые должны были защитить от этого (но не смогли)

XSS (межсайтовый скриптинг) – атакующий может внедрить свой JS код и обойти SOP

## XSS

### Пример:

1) Пользователь ввёл данные, отправил форму

```
<script>alert(document.domain)</script>
```

2) В ответе должно быть

```
&lt;script&gt;alert(1)&lt;/script&gt;
```

3) Но отдаётся внедренный HTML/JS

```
<script>alert(document.domain)</script>
```

**\$5000 - за XSS google.com**

## XSS

Казалось бы, защита:

- Преобразуем HTML сущности и не даём внедрять тэги пользователю

Проблема:

- Случаев, где можно допустить уязвимость, намного больше

## XSS

Случай #1 – Возможность подстановки URL

1) Ожидалось <http://example.com>

2) Подставляется `javascript:alert(1);`

```
<a href="javascript:alert(1)">my home page</a>
```

## XSS

### Случай #2 – Внедрение без HTML тэгов

```
<script>  
  var id = {{USER_ID}}; // some code using id, say:  
  // alert("Your user ID is: " + id);  
</script>
```



## XSS

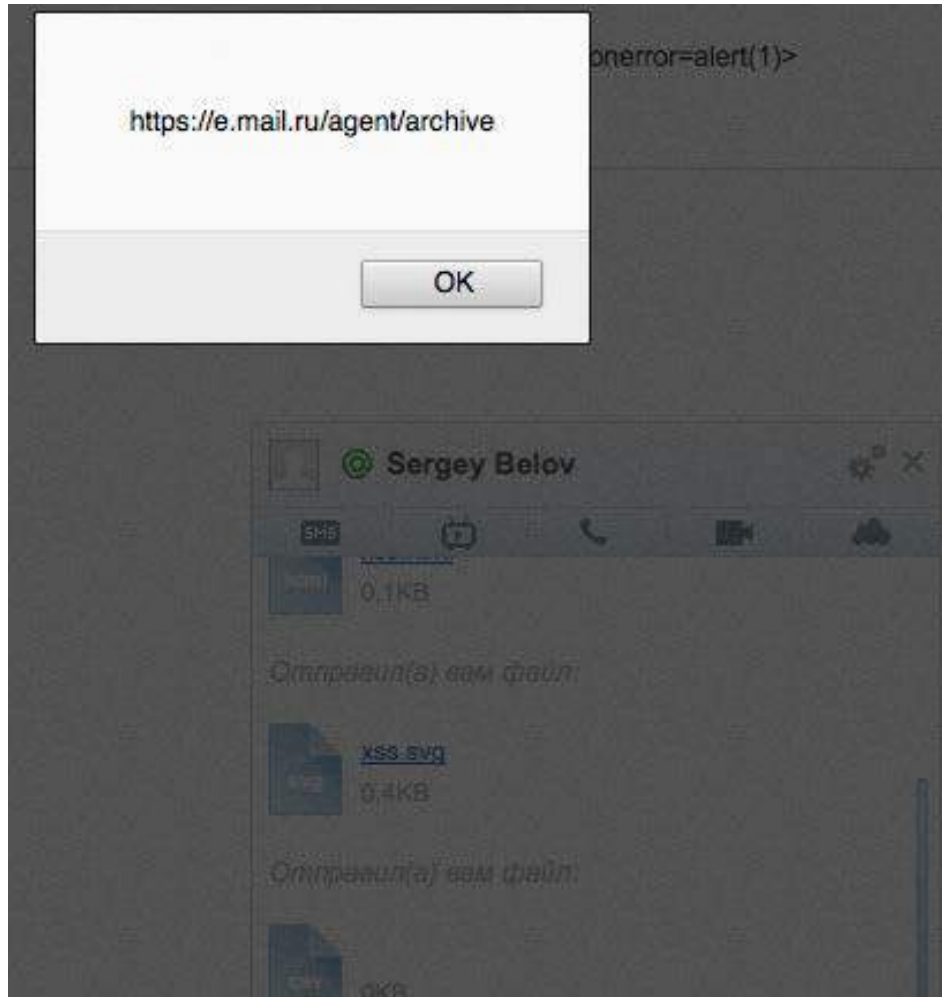
### Случай #3 – XSS в имени файла

Реальный случай из жизни: XSS в почте Mail.ru

- 1) Создаем локальный файл `<img src=x onerror=alert(1)>.jpg`
- 2) Отправляем пользователю в агент (на сайте)
- 3) При получении выводится имя файла -> XSS

*<https://hackerone.com/reports/54723>*

# XSS



## XSS

### Automatic Context-Aware Escaping in Template Systems

- За разработчика определяем место подстановки данных от пользователя
- Применяем нужный фильтр

### Почитать и попробовать

- <https://googleonlinesecurity.blogspot.ru/2009/03/reducing-xss-by-way-of-automatic.html>
- <http://webblaze.cs.berkeley.edu/papers/csas-ccs11.pdf>
- [http://google-ctemplate.googlecode.com/svn/trunk/doc/auto\\_escape.html](http://google-ctemplate.googlecode.com/svn/trunk/doc/auto_escape.html)

Исключает ли это XSS атаки?  
Нет!

## DOM XSS

Задача: отрендерить без шаблонизатора на сервере

```
document.write("Site is at: " + document.location.href);
```

[http://victim.com/action#<script>alert\('xss'\)</script>](http://victim.com/action#<script>alert('xss')</script>)

*Есть попытки решения - <https://github.com/cure53/DOMPurify>*

## XSS via HOST Header

### XSS без шаблонизатора

#### Request

Raw Params Headers Hex

```
GET /cse/tools/create_onthefly HTTP/1.1
Host: www.google.com:443</textarea><script>alert(1)</script>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:35.0)
Gecko/20100101 Firefox/35.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

#### Response

Raw Headers Hex HTML Render

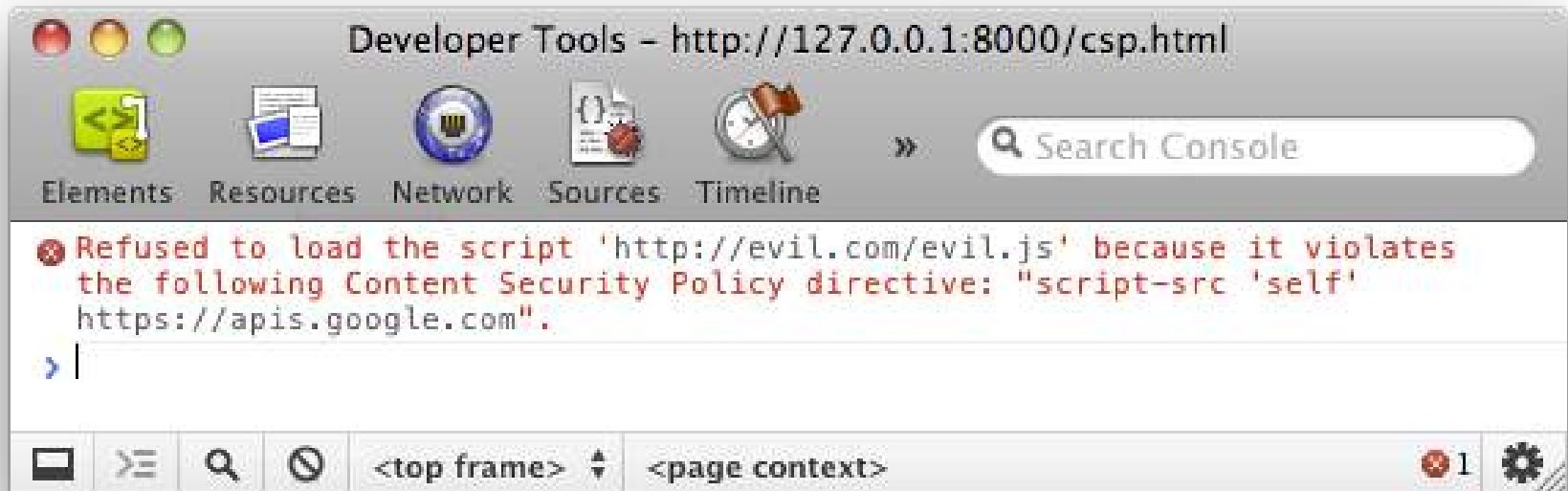
```
<textarea cols="75" rows="8" readonly="readonly" wrap="off">
<!-- Use of this code assumes agreement with the Google Custom
Search Terms of Service. -->
<!-- The terms of service are available at
http://www.google.com:443</textarea><script>alert(1)</script>/cs
e/docs/tos.html -->
<form name="cse" id="searchbox_demo"
action="http://www.google.com/cse">
<input type="hidden" name="cref" value="" />
<input type="hidden" name="ie" value="utf-8" />
<input type="hidden" name="hl" value="" />
<input name="q" type="text" size="40" />
<input type="submit" name="sa" value="Search" />
</form>
<script type="text/javascript"
src="https://www.google.com:443/</textarea><script>alert(1)</scri
pt>/cse/tools/onthefly?form=searchbox_demo&lang="></script>
</textarea><br />
<span class="gray">Use of this code assumes agreement with
the Google Custom Search <a
href="/cse/docs/tos.html"
```

<http://blog.bentkowski.info/2015/04/xss-via-host-header-cse.html>

# Content Security Policy

## CSP

X-Content-Security-Policy: script-src js.example.com





## CSP

### Проблемы:

- Избавиться от inline js очень сложно (но возможно)
- Вайтлистинг доменов – не самая тривиальная задача

## CSP

### Обход CSP:

- Загрузка файлов на доменов из whitelist (в сообщениях, почте и т.п.)
- Использование файлов с разрешенных доменов, например Google CDN

## CSP

Конкурс от @cure53 – обход CSP, условия:

- Для JS разрешен только `ajax.googleapis.com`
- Есть inline внедрение

Решение:

- Подключение уязвимой версии AngularJS

```
ng-app"ng-csp ng-click=$event.view.alert(1337)>  
<script src=  
//ajax.googleapis.com/ajax/libs/angularjs/1.0.8/  
angular.js>  
</script>
```

# OAuth

## OAuth

**OAuth** — открытый протокол авторизации, который позволяет предоставить третьей стороне ограниченный доступ к защищенным ресурсам пользователя без необходимости передавать ей (третьей стороне) логин и пароль.


OAuth

# Атакуем Digital Ocean

Реализация OAuth: Ruby on Rails + Doorkeeper

# OAuth

[Go back to Apps & API Page](#)

 Your application's callback URL. Read our OAuth documentation for more information

Register Application

# OAuth

## Authorize Application

Hey habr would like permission to access your account: **sergeybelove@gmail.com**

### Review Permissions

- Read

Authorize Application

Deny

**Hey habr**

Yo

[Visit application website](#)



## OAuth

Вектор:

- Создаем свое приложение в панели
- Через CSRF атаку форсируем установку приложения
- Похищаем токен полного доступа к аккаунту

*Статус: исправлено*

<http://habrahabr.ru/company/dsec/blog/246025/>

## OAuth

Находят ли тоже самое в кастомных решениях? Да!

- Microsoft Hotmail
- \$24 000
- Абсолютно аналогичный вектор

*<https://www.synack.com/labs/blog/how-i-hacked-hotmail/>*

Open Source – не гарантирует безопасность

# Обфускация

## Обфускация

**SAP  
AFARIA**



## Обфускация

### **Пример из нашей практики**

Мобильное приложение SAP Afaria – MDM  
решение для компаний

ah  
ai  
aj  
ak  
al  
am  
an  
ao  
ap  
aq  
ar  
as  
at  
au  
av  
aw  
ax  
ay  
az  
b  
ba  
bb  
bc  
bd  
be  
bf  
bg  
bh  
bi  
bj  
bk  
bl  
bm  
bn  
bo  
bp  
bq  
br  
bs  
bt  
bu  
bv  
bw  
bx  
by  
bz  
c  
ca  
cb  
cc  
cd  
ce

```
t.class m.class d.class c.class ax.class bb.class bf.class bi.class bk.class x
package com.Android.Afaria.applist;

import android.app.Notification;

public class bk
    extends Handler
    implements ao
{
    private static int a = -805306368;
    private int b;
    private g c;
    private boolean d;
    private DownloadService e;
    private File f;
    private Notification g;
    private int h;
    private RemoteViews i;
    private long j;
    private long k;
    private String l;
    private String m;
    private int n;
    private int o;

    public bk(DownloadService paramDownloadService, g paramw)
    {
        this.e = paramDownloadService;
        this.c = paramw;
        this.b = 0;
        this.n = 0;
        this.m = "";
        this.l = "";
        this.o = paramw.D;
    }

    private void a(int paramInt)
    {
        Message.obtain(this, paramInt).sendToTarget();
    }

    /* Error */
    private static String b(int paramInt)
    {
        // Byte code:
        // 0: iload_0
        // 1: sipush 1024
        // 4: if_icmpge +30 -> 34
        // 7: new 75 java/lang/StringBuilder
        // 10: dup
        // 11: invokespecial 76 java/lang/StringBuilder:<init> ()V
        // 14: iload_0
        // 15: invokestatic 81 java/lang/String:valueOf (I)Ljava/lang/String;
        // 18: invokevirtual 85 java/lang/StringBuilder:append (Ljava/lang/String;)Ljava/lang/StringBuilder;
        // 21: ldc 87
        // 23: invokevirtual 85 java/lang/StringBuilder:append (Ljava/lang/String;)Ljava/lang/StringBuilder;
        // 26: invokevirtual 91 java/lang/StringBuilder:toString ()Ljava/lang/String;
        // 28: areturn
    }
}
```

## Обфускация

Перехватывает SMS сообщения

```
@#!Afaria64aACAhntVzjTIjHDMGql8ldvc/8U6IlIoPU7aAOT8=$\ $CMD:USER  
LOCK
```

Выполняет нужную команду.

Структура

```
@#!Afaria+base64(sha256(<ClientID >  
+<ClientID>+<TransmitterID>+$\ $CMD:  
+<CMD_NAME> ) )+$\ $CMD:+ <CMD_NAME>
```



## Обфускация

```
private static boolean a(final String s, final String s2) {
    while (true) {
        try {
            c.a(t.a);
            final String upperCase = c.c("LastSessionGUID", "").toUpperCase(Locale.US);
            final String upperCase2 = c.c("ClientGUID", "").toUpperCase(Locale.US);
            final String c = com.Android.Afaria.core.c.c("TransmitterID", "");
            final int length = upperCase2.length();
            boolean b = false;
            if (length != 0) {
                final int length2 = upperCase.length();
                b = false;
                if (length2 != 0) {
                    final int length3 = c.length();
                    b = false;
                    if (length3 != 0) {
                        final String d = d(upperCase + upperCase2 + c + s2);
                        final String d2 = d(upperCase2 + upperCase2 + c + s2);
                        com.Android.Afaria.tools.b.b("Command", "SMS Hash: " + s);
                        com.Android.Afaria.tools.b.b("Command", "Calculated Hash: " + d);
                        com.Android.Afaria.tools.b.b("Command", "Ignore lastSessionGUID calculated Hash: " + d2);
                        if (d2.compareTo(s) == 0 || d.compareTo(s) == 0) {
                            com.Android.Afaria.tools.b.b("Command", "Hashes match!");
                            b = true;
                        }
                    }
                    else {
                        com.Android.Afaria.tools.b.b("Command", "Hashes do not match!");
                        b = false;
                    }
                }
            }
        }
    }
}
```

## Обфускация

- `TransmitterID` МОЖНО ПОЛУЧИТЬ АНОНИМНО
- `ClientID` основан на IMEI  
IMEI?
  - Information Disclosure
  - XSS
  - bruteforce
  - ...

# Обфускация



IMSI	SESSIONS	NETWORK
eli2	2013-10-08 10:16:39	358867042495036
425089109373513	2013-10-08 10:00:11	012403007261896
eli	2013-10-08 09:59:20	013032005418064
demo	2013-10-03 12:53:40	352926023488000
test	2013-10-03 12:46:33	352924024951985
425020172168013	2013-10-03 12:39:04	352993056959711
demo		demo

Обфускация != безопасность

## Финальные мысли:

- 1) Современные технологии могут сократить количество уязвимостей
- 2) Нужно их правильное применение
- 3) Обязательно есть места, о которых никто не думал. И их нужно найти!

# Проверяйте безопасность!

Digital Security в Москве: (495) 223-07-86  
Digital Security в Санкт-Петербурге: (812) 703-15-47

[sbelov@dsec.ru](mailto:sbelov@dsec.ru)  
[www.dsec.ru](http://www.dsec.ru)