

## Problema de XSS com **ElementRef** no Angular

O **ElementRef** permite acessar diretamente elementos do DOM no Angular, mas seu uso incorreto pode abrir brechas de segurança, especialmente para **ataques de Cross-Site Scripting (XSS)**.

---

### O Que é XSS?

XSS (Cross-Site Scripting) é um ataque onde um invasor insere **código JavaScript malicioso** em uma página web para roubar informações ou executar ações indesejadas.

---


## Exemplo Inseguro de XSS com **ElementRef**

Aqui está um exemplo clássico de **uso inseguro de ElementRef**, onde o usuário pode injetar código malicioso:

### Código TypeScript

```
import { Component, ElementRef } from '@angular/core';

@Component({
  selector: 'app-xss-demo',
  templateUrl: './xss-demo.component.html',
  styleUrls: ['./xss-demo.component.scss']
})
export class XssDemoComponent {
  constructor(private _elRef: ElementRef) {}

  createElement(inputText: string) {
    const divEl = document.createElement('div');
    divEl.innerHTML = inputText; //  PERIGO: Insere conteúdo sem sanitização
    this._elRef.nativeElement.appendChild(divEl);
  }
}
```

### Código HTML

```
<input placeholder="Digite algo..." #meuInput>
<button (click)="createElement(meuInput.value)">Criar Elemento</button>
```

## Como esse código pode ser explorado?

Se o usuário digitar este texto no input e clicar no botão:

```

```

### Resultado:

- O navegador tentará carregar uma imagem chamada "x", mas como ela não existe, o evento `onerror` será acionado.
- O JavaScript dentro de `onerror="alert('XSS!');"` será executado, mostrando um alerta perigoso.

💥 Isso significa que qualquer código JavaScript inserido pelo usuário será executado no navegador!

---

## ✓ Solução 1: Usar **Renderer2** para Evitar XSS

O **Renderer2** é uma API segura do Angular para manipular o DOM sem abrir brechas de segurança.

### Código Seguro com **Renderer2**

```
import { Component, ElementRef, Renderer2 } from '@angular/core';

@Component({
  selector: 'app-xss-demo',
  templateUrl: './xss-demo.component.html',
  styleUrls: ['./xss-demo.component.scss']
})
export class XssDemoComponent {
  constructor(private _elRef: ElementRef, private _renderer: Renderer2) {}

  createElement(inputText: string) {
    const divEl = this._renderer.createElement('div');
    const textNode = this._renderer.createText(inputText); // Cria um nó de texto seguro

    this._renderer.appendChild(divEl, textNode);
    this._renderer.appendChild(this._elRef.nativeElement, divEl);
  }
}
```

### O que mudou?

- ✓ Agora, o texto do usuário é tratado como simples texto e não como HTML.
  - ✓ Se o usuário tentar injetar `<script>alert('XSS!')</script>`, o navegador NÃO executará o código.
  - ✓ O Renderer2 evita que qualquer código JavaScript malicioso seja interpretado.
- 

## ✓ Solução 2: Sanitização de HTML com DomSanitizer

Se for realmente necessário permitir HTML dinâmico, utilize o **DomSanitizer** do Angular para evitar XSS.

### Código Seguro com DomSanitizer

```
import { Component } from '@angular/core';
import { DomSanitizer, SafeHtml } from '@angular/platform-browser';

@Component({
  selector: 'app-xss-demo',
  templateUrl: './xss-demo.component.html',
  styleUrls: ['./xss-demo.component.scss']
})
export class XssDemoComponent {
  safeHtml: SafeHtml = ''; // Armazena HTML sanitizado

  constructor(private _sanitizer: DomSanitizer) {}

  createElement(inputText: string) {
    this.safeHtml = this._sanitizer.bypassSecurityTrustHtml(inputText);
    // ♦ Transforma o texto digitado pelo usuário em HTML seguro
  }
}
```

### Código HTML

```
<input placeholder="Digite algo..." #meuInput>
<button (click)="createElement(meuInput.value)">Criar Elemento</button>

<div [innerHTML]="safeHtml"></div> <!-- Insere o HTML sanitizado -->
```

### O que mudou?

- ✓ Agora, o HTML do usuário passa por uma verificação de segurança antes de ser exibido.
- ✓ Se houver scripts maliciosos, eles serão removidos antes da exibição.



## Resumo: Como Evitar XSS no Angular

Método	Seguro ?	Quando Usar?
<code>innerHTML</code>	❌ Não	Nunca, pois permite execução de scripts maliciosos
<code>Renderer2</code>	✅ Sim	Quando quiser adicionar apenas texto puro
<code>DomSanitizer</code>	✅ Sim	Quando quiser permitir HTML controlado e seguro

---



## Conclusão

O `ElementRef` é uma ferramenta poderosa, mas **se usado incorretamente, pode criar vulnerabilidades sérias no seu aplicativo Angular.**

- ✓ Sempre que possível, use `Renderer2` para manipular elementos do DOM.
- ✓ Se precisar renderizar HTML, utilize `DomSanitizer` para evitar XSS.
- ✓ Evite `innerHTML` diretamente com dados vindos do usuário.

⚡ Agora você está preparado para evitar ataques XSS no Angular! 💪