文字列からHTMLを組み立てる話

Shibuya.XSS techtalk #5

Aug 7 2014

Yosuke HASEGAWA

#shibuyaxss

文字列からHTMLを組み立てる

- サーバ上でHTML断片を生成
- ブラウザ上でinnerHTMLに挿入

```
//よくある脆弱なコード例
div.innerHTML = xhr.responseText;
```

- DOM based XSSの温床

よろしい ならばサニタイズだ

文字列のサニタイズ

- サニタイズしてinnerHTMLへ

```
div.innerHTML = sanitize( xhr.responseText );
```

- IEであればtoStaticHTMLが使える

```
//IE限定
div.innerHTML =
    toStaticHTML( xhr.responseText );
```

JavaScript等を削除した「静的」な
HTML文字列を返す

IE以外はどうするのか

- toStaticHTMLもどきを作る?
- 「作るな！安全なものを探せ！」
- Google Caja

> 最後にJavaScriptでHTMLをパースする方法です。ブラウザ側での実装に依存せず，JavaScriptのみでHTMLのサニタイジングを行います。このアプローチで実績があるのは，google-cajaライブラリです。
> …(略)…
> ただし，google-cajaライブラリに含まれるHTMLサニタイザは処理速度に問題があることが知られています。…(略)…また，ブラウザ側のHTMLパーサとの非互換の問題もあるでしょう。通常は問題にならないでしょうが，文法違反のHTMLの扱いなどで，細かな差異が必ずあるからです。

第4回　危険性が理解されにくいネイティブアプリ内XSS（2）：フロントエンドWeb戦略室｜gihyo.jp … 技術評論社
http://gihyo.jp/dev/serial/01/front-end_web/000402

Google Caja

- 文字列をパースして安全なHTML文字列を組み立て
  - 遅い(らしい)
  - ブラウザのパーサと非互換(らしい)
- ブラウザと非互換なHTMLパーサはDOM XSSを生みやすい(Cajaがそう、というわけではない)
  - XSS with HTML parsing confusion (ma.la) at AppSecAPAC 2014
    https://speakerdeck.com/owaspjapan/xss-with-html-parsing-confusion-number-appsecapac2014

ブラウザ互換のパーサ?

ブラウザ互換のパーサ

- ブラウザ内蔵のパーサを使う
  - DOMParser
  - createHTMLDocument
- 現在表示しているdocumentに影響を与えずにDOMツリーを構築可能
  - Opera 12(Presto)を除く

ブラウザ互換のパーサ

- **DOMParser、createHTMLDocumentによるパースとDOMノードの構築**

> 次に，ブラウザ側の機能を使ってHTMLをパースする方法です。これはcreateHTMLDocumentを使うとよいでしょう。createHTMLDocumentはHTML5仕様で標準化されており，安定して使うことができます。IEでもIE9以降でサポートされています。HTMLパース処理を行い，新たなドキュメントを作ったら，あとはすべてのDOMノードとAttributeを列挙して，許可したタグと属性以外をすべて除去すれば完了です。

第4回　危険性が理解されにくいネイティブアプリ内XSS（2）：フロントエンドWeb戦略室｜gihyo.jp … 技術評論社
http://gihyo.jp/dev/serial/01/front-end_web/000402

ブラウザ内パーサの利用

- DOMParser

```
var s = xhr.responseText;
var parser = new DOMParser();
var doc = parser.parseFromString( s, "text/html" );
var elm = doc.body;
```

- createHTMLDocument

```
var s = xhr.responseText;
var elm =
  document.implementation.createHTMLDocument("").body;
elm.innerHTML = s;
```

- 文字列をパースしDOMノードを構築可能
- DOMノードから必要な要素、属性を切り出す

ブラウザ内パーサの利用

- DOMParser、createHTMLDocumentは現在表示しているdocumentに影響を与えない

```
var s = "<img src=# onerror=alert(1)>"; // 発火しない!!
var parser = new DOMParser();
var doc = parser.parseFromString( s, "text/html" );
```

- createContextualFragmentはブラウザによって発火する

ブラウザ互換のパーサ

- **DOMParser、createHTMLDocumentによるパースとDOMノードの構築**

> 次に，ブラウザ側の機能を使ってHTMLをパースする方法です。これはcreateHTMLDocumentを使うとよいでしょう。createHTMLDocumentはHTML5仕様で標準化されており，安定して使うことができます。IEでもIE9以降でサポートされています。HTMLパース処理を行い，新たなドキュメントを作ったら，あとはすべてのDOMノードとAttributeを列挙して，許可したタグと属性以外をすべて除去すれば完了です。

第4回　危険性が理解されにくいネイティブアプリ内XSS（2）：フロントエンドWeb戦略室｜gihyo.jp … 技術評論社
http://gihyo.jp/dev/serial/01/front-end_web/000402

＿人人人人人人人人人人人人＿
＞　すべてのDOMノードと　＜
＞　Attributeを列挙して，　＜
＞許可したタグと属性以外を＜
＞　すべて除去すれば完了　＜
￣Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y￣

言うのは簡単…

必要な要素、属性のみ切り出す

- とりあえず作ってみた
- 構築されたHTMLBodyElementから必要な要素、属性を切り出す

DEMO

http://jsbin.com/dafeh

ブラウザ内パーサの利用

- DOMParser、createHTMLDocumentはHTMLBodyElementを取得可能
- 生成したHTMLElementは文字列に変換しないこと！

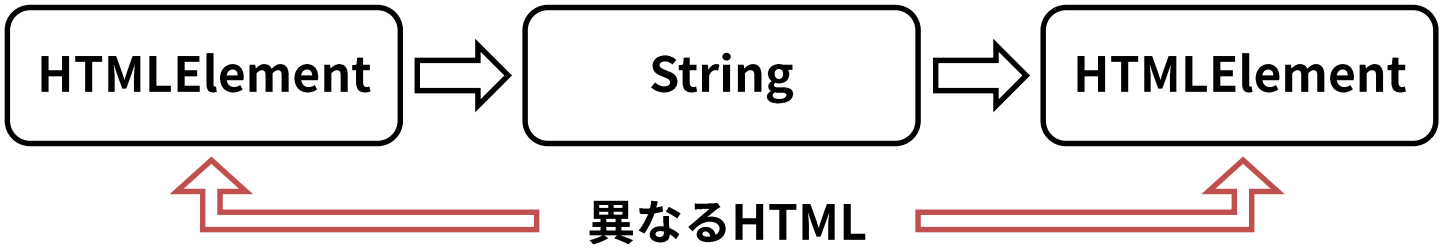
```
// bad code.
var parser = new DOMParser();
var doc = parser.parseFromString( s, "text/html" );
div.innerHTML = doc.body.innerHTML;
```

文字列に変換しないこと。なぜ？

文字列に変換しないこと

- HTMLElementから文字列への変換は mXSSを引き起こす(可能性がある)

```
// bad code. IEではmXSSとなる
var s =
  "<listing>&lt;img src=1 onerror=alert(1)&gt;</listing>";
var parser = new DOMParser();
var doc = parser.parseFromString( s, "text/html" );
sanitize( doc.body );
div.innerHTML = doc.body.innerHTML;
```



http://www.thespanner.co.uk/2014/05/06/mxss/

文字列に変換しないこと

- こういうコードはダメ

```
// bad code. toStaticHTMLﾓﾄﾞｷを作りたい
if( window.toStaticHTML ){
    return toStaticHTML( s );
}else{
    var parser = new DOMParser();
    var doc = parser.parseFromString( s, "text/html" );
    var newNode = rebuildSanitizedElement( doc.body );
    return newNode.innerHTML;
}
```

文字列に変換しないこと

- 書くならこういう感じ

```
// toStaticHTMLﾓﾄﾞｷを作りたい
if( window.toStaticHTML ){
    var div = document.createElement("div");
    div.innerHTML = toStaticHTML( s );
    return div;
}else{
    var parser = new DOMParser();
    var doc = parser.parseFromString(
        "<div>" + s + "</div>", "text/html" );
    var newNode = rebuildSanitizedElement( doc.body );
    return newNode.childeNodes[ 0 ];
}
```

- 文字列ではなくHTMLElementを返す

文字列に変換、本当にダメなの?

文字列への変換、本当にダメなの?

- mXSSは不偏的なものではない
  - 「mXSSがあるかも知れない可能性」への対応
  - 未知のmXSSを恐れても仕方ないのでは!

> さて，HTMLサニタイズ処理について書きましたが，これらは実はあまり重要ではないと考えています。それよりも高い優先度で行うべきことは，「HTMLサニタイズ処理にバグがあっても平気なようにする」ことです。

第4回　危険性が理解されにくいネイティブアプリ内XSS（3）：フロントエンドWeb戦略室｜gihyo.jp … 技術評論社
http://gihyo.jp/dev/serial/01/front-end_web/000403

＿人人人人人人人人＿
＞　バグがあっても　＜
＞　平気なようにする　＜
￣Y^Y^Y^Y^Y^Y^Y^Y^Y^Y￣

つらい…

バグがあっても平気なように

- HTML5 sandboxed iframeが使える!

```
<iframe id="iframe" sandbox seamless
  style="border-width:0px"></iframe>
...
document.getElementById("iframe").srcdoc = xhr.responseText;
```

  - sandbox属性によりJSを禁止
  - seamless属性により親のCSSを継承
  - コンテンツはsrcdocに直接設定
    HTMLElementならcontentDocument経由で。

まとめ
Conclusion

まとめ

- toStaticHTML便利(IE限定)
- DOMParser、createHTMLDocumentでHTML文字列をパースしてDOM構築可能
- HTMLElement→文字列の流れはmXSSの可能性がある
- iframe sandbox超便利

Question? 質問

hasegawa@utf-8.jp
hasegawa@netagent.co.jp

@hasegawayosuke

http://utf-8.jp/

@hasegawayosuke