HTML5 and Security

Part 4 : DOM based XSS

HTML5セキュリティ その4 : DOM based XSS編

Sep 8 2014

Yosuke HASEGAWA

#owaspjapan

自己紹介

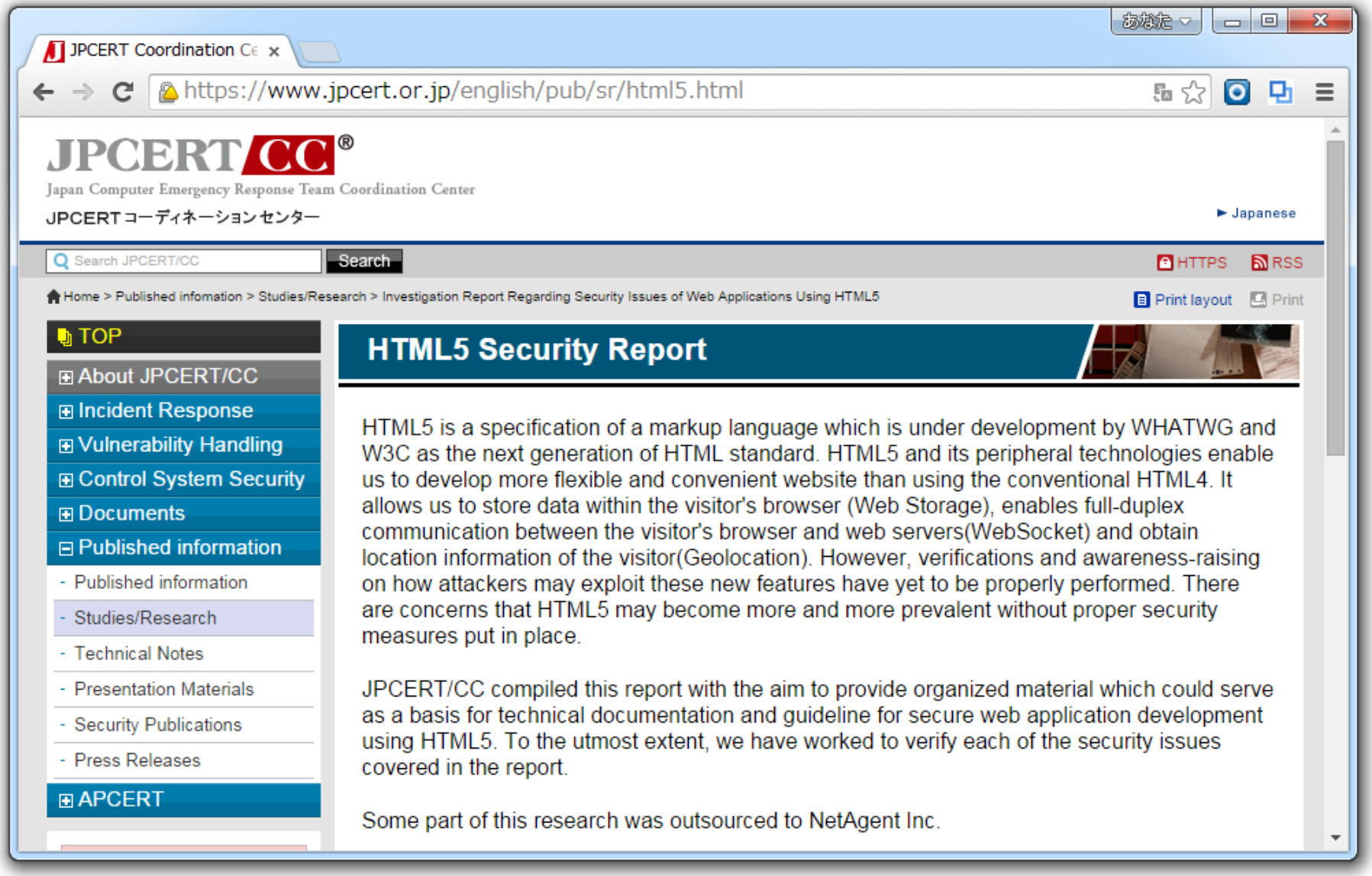
はせがわようすけ

- ネットエージェント株式会社
- 株式会社セキュアスカイ・テクノロジー 技術顧問
- http://utf-8.jp/
- OWASP Kansai Chapter Leader
- OWASP Japan Chapter Advisory Board member

@hasegawayosuke

お知らせ
Announcement

HTML5 Security Report English edition from JPCERT/CC



今日の話題
DOM based XSS
Today's topic : DOM based XSS

DOM based XSS

- JavaScriptでDOMを組み立てるときのXSS

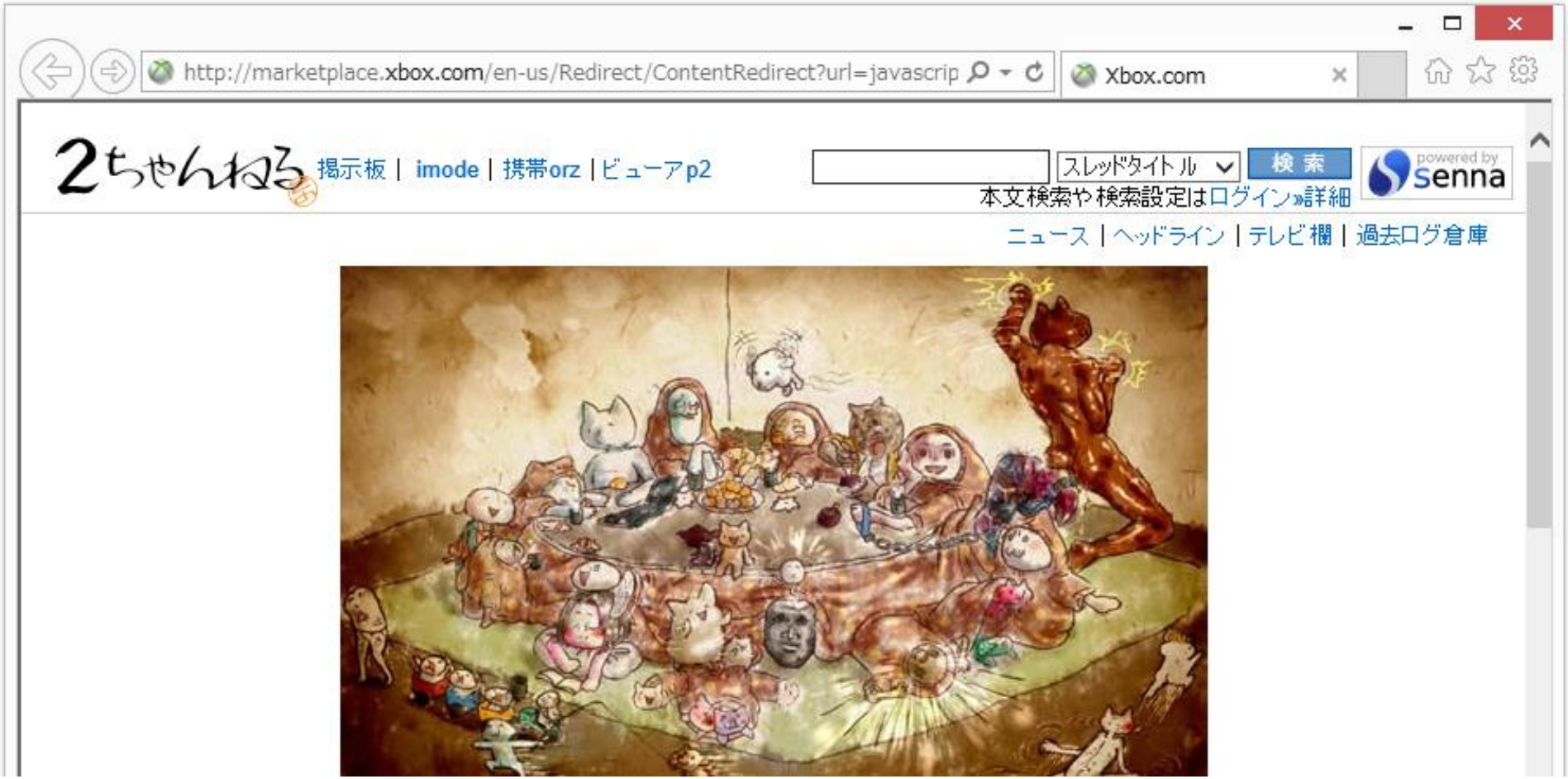
```
location.href = location.hash.substring(1);
```

```
document.write( document.referrer );
```

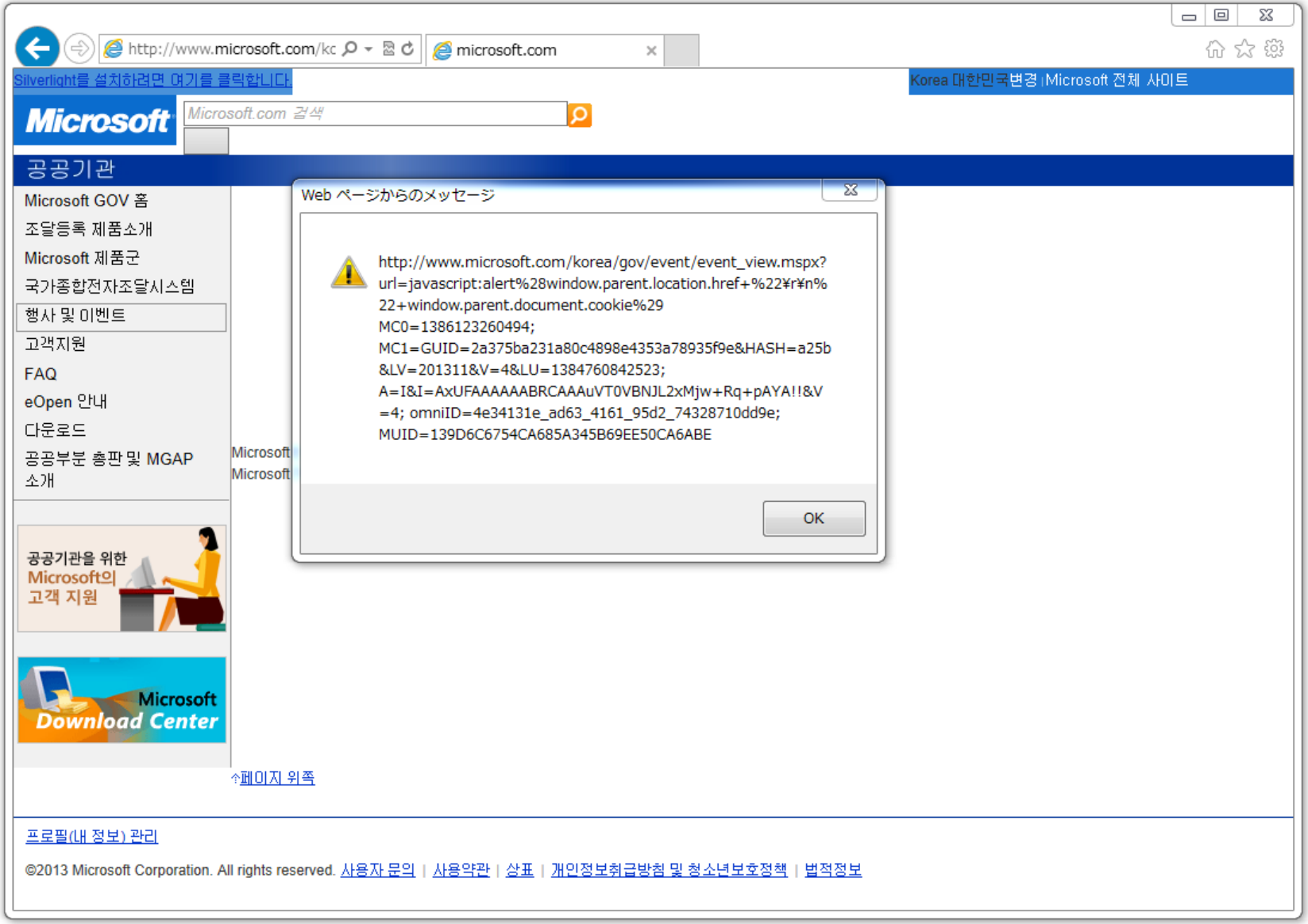
```
div.innerHTML = xhr.responseText;
```

  - サーバ側のHTML生成には問題なし
  - JavaScriptの利用に合わせて発生も増加

DOM based XSS



DOM based XSS



DOM based XSS

- ブラウザのXSSフィルタを通過することが多い
- location.hash内の実行コードはサーバ側にログが残らない

```
//http://example.jp/#<script>alert(1)</script>
div.innerHTML = location.hash.substring(1);
```

- history.pushStateでアドレスバー書き換え
  - 技術のあるユーザでもXSSに気づきにくい

DOM based XSS

- DOM based XSSは増えている
  - JavaScriptの大規模化に伴い増加
- サーバ側での対策と原則は同じ
  - HTML生成時(レンダリング時)にエスケープ
  - URL生成時はhttp(s)のみ
  - ライブラリの更新を忘れずに(jQueryとか)
  - CSSやイベントハンドラの動的生成は避ける

DOM based XSS

- HTML生成時にエスケープ

```
div.innerHTML = s.replace( /&/g, "&amp;" )
    .replace( /</g, "&lt;" )
    .replace( />/g, "&gt;" )
    .replace( /"/g, "&quot;" )
    .replace( /'/g, "&#x27;" );
```

- むしろtextNodeを使おう!

```
div.appendChild(
    document.createTextElement( s )
);
```

DOM based XSS

- URL生成時はhttp(s)のみ

```
// bad code
div.innerHTML = '<a href="' + url + '">' + url + '</a>';
```

```
if( url.match( /^https?:¥/¥// ) ){
    var elm = docuement.createElement( "a" );
    elm.appendChild( document.createTextNode( url ) );
    elm.setAttribute( "href", url );
    div.appendChild( elm );
}
```

DOM based XSS

- URL生成時はhttp(s)のみ
  - リダイレクト時はオープンリダイレクタを発生させないよう同一ホストに制限

```
var base = location.origin + "/";
if( url.substring( 0, base.length ) == base ){
    location.href = url;
}
```

ここまでDOM based XSSの基本です

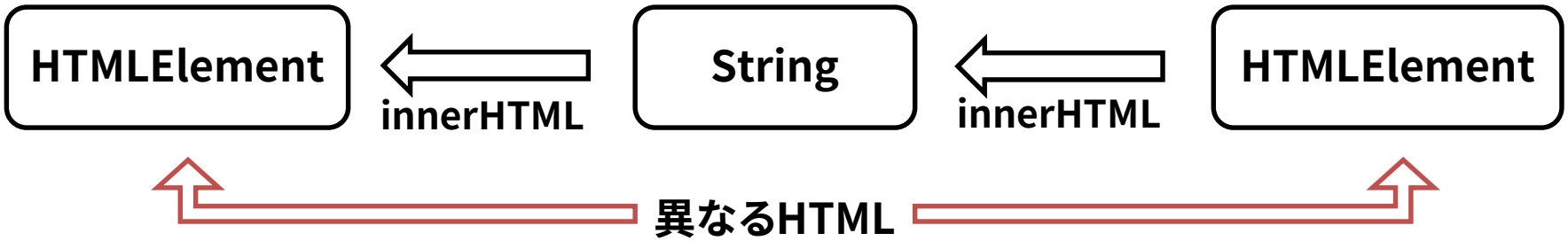
Basics of DbXSS so far

Mutation-based XSS

Mutation-based XSS : mXSS

- DOM based XSSの一種
- innerHTML / outerHTML の参照により元のDOM構造とは異なるHTML文字列が返されることによりXSSが発生

```
element1.innerHTML = element2.innerHTML;
```



Mutation-based XSS : mXSS

- DOM based XSSの一種
- innerHTML / outerHTML の参照により元のDOM構造とは異なるHTML文字列が返されることによりXSSが発生

  - The innerHTML Apocalypse – How mXSS attacks change everything we believed to know so far –
    http://www.slideshare.net/x00mario/the-innerhtml-apocalypse
  - mXSS Attacks: Attacking well-secured Web-Applications by using innerHTML Mutations
    https://cure53.de/fp170.pdf
  - mXSS – The Spanner
    http://www.thespanner.co.uk/2014/05/06/mxss/

Mutation-based XSS : mXSS

```
<!-- IE8 -->
<div id="div1">
  <input type="text" value="``onmouseover=alert(1)">
</div>
<div id="div2"></div>
....
div2.innerHTML = div1.innerHTML;
```

```
<div id="div2">
<INPUT value=``onmouseover=alert(1) type=text>
</div>
```

Mutation-based XSS : mXSS

```
<input type="text" value="``onmouseover=alert(1)">
<INPUT value=``onmouseover=alert(1) type=text>
```

```
<listing>&lt;img src=1 onerror=alert(1)&gt;</listing>
<LISTING><img src=1 onerror=alert(1)></LISTING>
```

```
<style/>&lt;/style&gt;&lt;img src=1 onerror=alert(1)&gt;</style>
<STYLE></style><img src=1 onerror=alert(1)></STYLE>
```

```
<title>&lt;img src=1 onerror=alert(1)&gt;</title>
<TITLE><img src=1 onerror=alert(1)></TITLE>
```

Mutation-based XSS : mXSS

- mXSS
  - innerHTML/outerHTMLを参照したときに本来のDOM構造とは異なる文字列が取得される
  - IE以外でも発生し得る(CDATA要素の参照など)
- 対策
  - 攻撃者がコントロール可能な要素のinnerHTML / outerHTMLを参照しない
  - 文字列ではなくDOM操作で…

ここまでのまとめ

- DOM based XSS
  - JS上で発生するXSS
- Mutation-based XSS
  - JS上でinnerHTML/outerHTMLを参照したときに発生するXSS
- 対策
  - 文字列操作ではなくDOM操作で。
    - createElement / textContent…

DOM操作めんどうくさい！

Manipulating DOM is messy!

DOM操作めんどうくさい (例)

```
[
    "<a href='http://example.jp/foo'>2014.09.08 新製品</a>",
    "<a href='http://example.jp/bar'>2014.09.01 お知らせ</a>",
    "<img src=# onerror=alert(1)>"
]
```

```
var xhr = new XMLHttpRequest();
xhr.open( "GET", "http://3rdparty.example.com/", true );
xhr.onload = function(){
    var items = JSON.parse( xhr.responseText );
    var elm = document.getElementById( "div" );
    for( var i = 0; i < items.length; i++ ){
        elm.innerHTML +=
            "<div class='item'>" + items[ i ] + "</div>";
    }
};
```

ここ、DOM操作で作るのしんどい…

DOM操作めんどうくさい

- toStaticHTML
  - IE8+、安全なHTML文字列を返す。お手軽。

```
div.innerHTML = toStaticHTML( s );
```

```
"<script>alert(1)</script>" → ""
"<img src=# onerror=alert(1)>" → "<img src=#>"
"<a href='javascript:alert(1)'>link</a>" → "<a>link</a>"
```

DOM操作めんどうくさい

IE以外はどうするか

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

ブラウザの機能を使ってHTMLをパース

- DOMParser / createHTMLDocument
  - ブラウザ内蔵のHTMLパーサ
- 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を除く

ブラウザの機能を使ってHTMLをパース

- 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

ブラウザの機能を使ってHTMLをパース

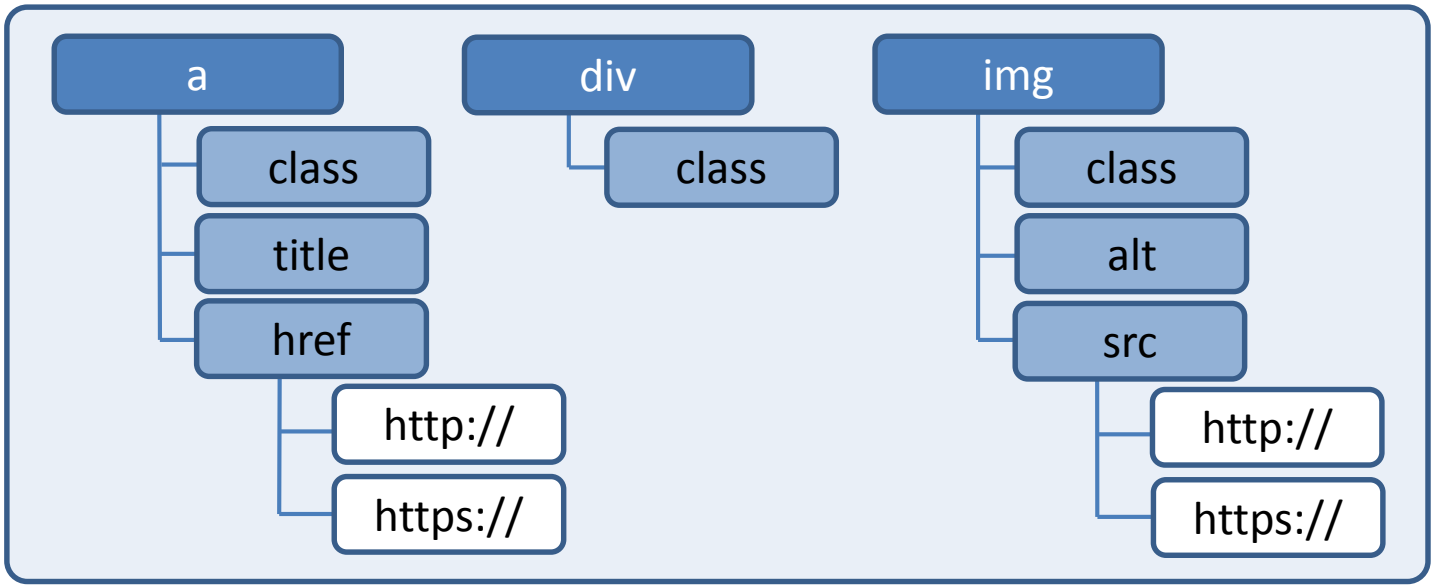
- DOMParser、createHTMLDocumentは現在表示しているdocumentに影響を与えない

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

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

ノードから必要な要素、属性を切り出す

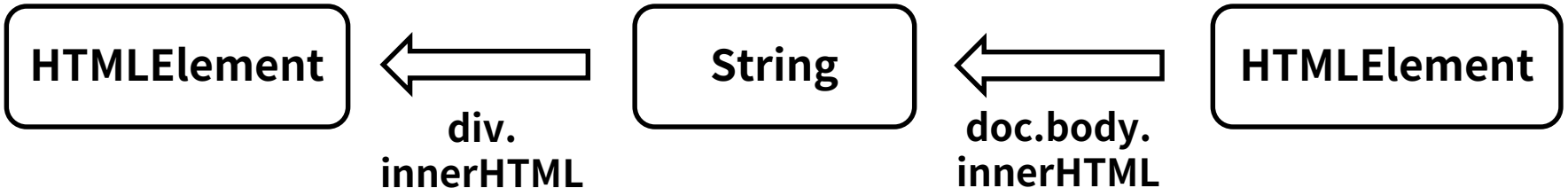
- DOMParser、createHTMLDocumentでHTMLElementを生成
- 生成されたHTMLElementの要素、属性を列挙して安全なものだけを抽出



文字列とHTMLElement

- 作成したHTMLElementを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;
```



文字列とHTMLElement

- 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" );
div.innerHTML = doc.body.innerHTML;
```

文字列とHTMLElement

- 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 Parser();
    var doc = parser.parseFromString( s, "text/html" );
    var newNode = sanitize( doc.body );
    return newNode.innerHTML;
}
```

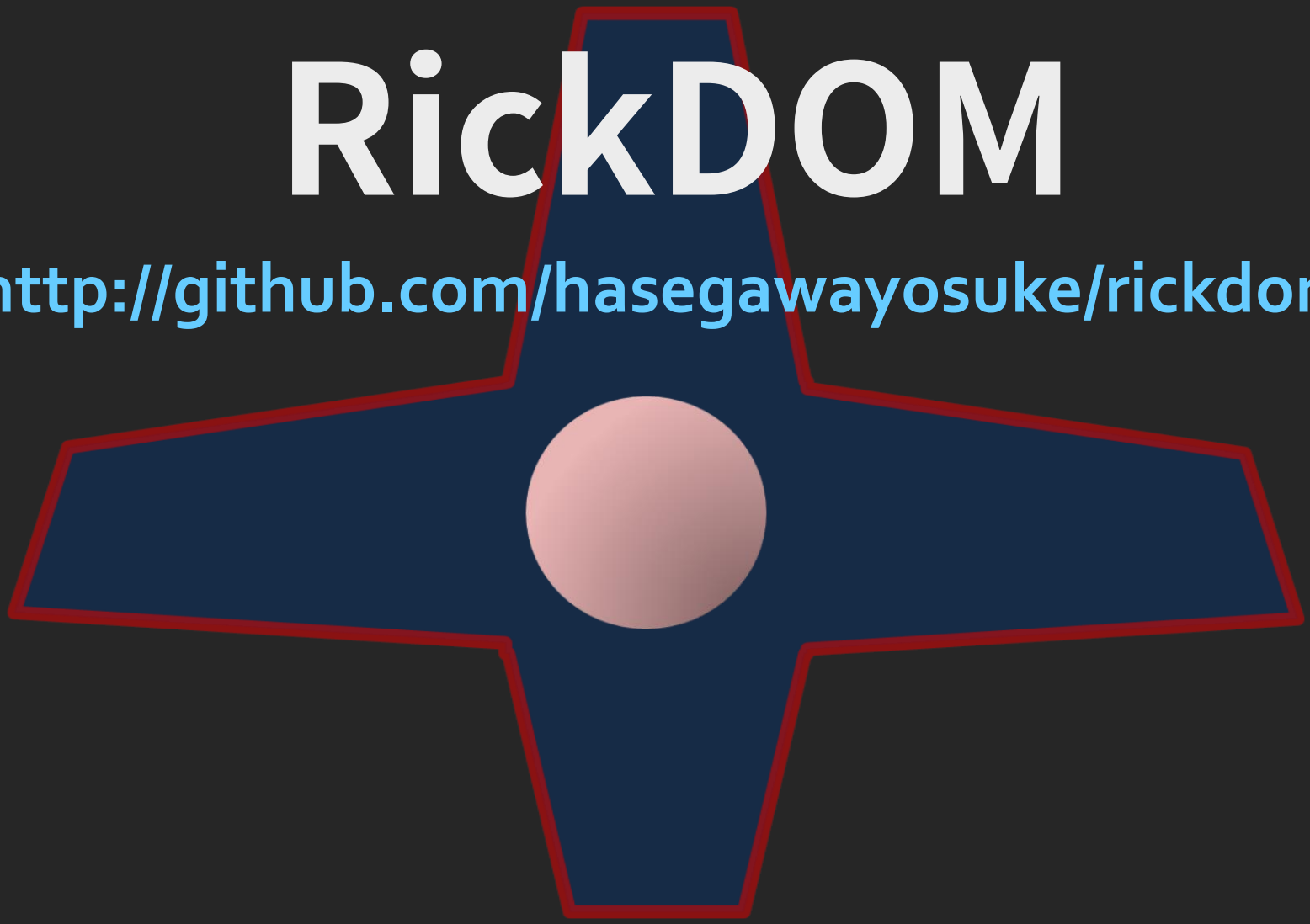
文字列とHTMLElement

- 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.childNodes;
}else{
    var parser = new Parser();
    var doc = parser.parseFromString( s, "text/html" );
    var newNode = sanitize( doc.body );
    return newNode.childNodes; // HTMLElement
}
```

RickDOM

http://github.com/hasegawayosuke/rickdom/



RickDOM

- 簡単かつ安全に使えるライブラリ

```
var rickdom = new RickDOM();
var elms = rickdom.build( "<img src=# onerror=alert(1)>" );
for( var i = 0; i < elms.length; i++ ){
    div.appendChild( elms[ i ] );
}
```

- 独自の許可ルールの設定も可能

```
var rickdom = new RickDOM();
rickdom.allowings =
  { img: { src: { pattern : "^https?:¥¥/¥¥/", flag: "i" } } };
var elms = rickdom.build( "<img src=# onerror=alert(1)>" );
```

http://github.com/hasegawayosuke/rickdom/
http://utf-8.jp/public/rickdom/

よくわかんない。不安だ。

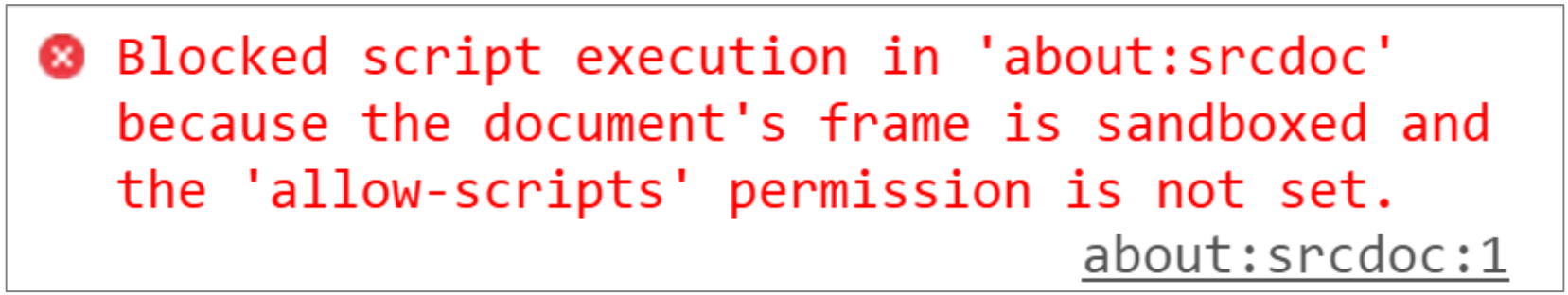
Anxious

sandboxed iframe

- sandboxなiframeを応用

```
<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(XSSを防ぐ)



sandboxed iframe

- sandboxなiframeを応用

```
<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(XSSを防ぐ)
  - seamless属性により親フレームのCSSを継承
  - コンテンツはsrcdoc属性に直接設定
    HTMLElementならcontentDocument経由で。

sandboxed iframe

- "/page" などのリンクの許可

```
<iframe id="iframe" sandbox="allow-top-navigation" seamless
  style="border-width:0px"></iframe>
...
document.getElementById( "iframe" ).srcdoc =
  '<base href="http://example.jp/" target="_parent">' +
  '<a href="/page">next</a>';
```

  - allow-top-navigationで親frame内でのページ遷移を許可

まとめ

- DOM based XSS,mXSS
  - 文字列で操作しない、DOM経由で操作
  - リンクはhttp(s)のみ
- 文字列からDOMの構築
  - DOMParser APIが便利
- sandboxed iframe
  - DOM baed XSSの予防に便利

質問タイム

Question ?

Question? 質問

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

@hasegawayosuke

http://utf-8.jp/

@hasegawayosuke