HTML5のセキュリティ もうちょい詳しく

HTML5セキュリティ その3：JavaScript API

Jun 6 2014

Yosuke HASEGAWA

#owaspkansai

自己紹介

はせがわようすけ

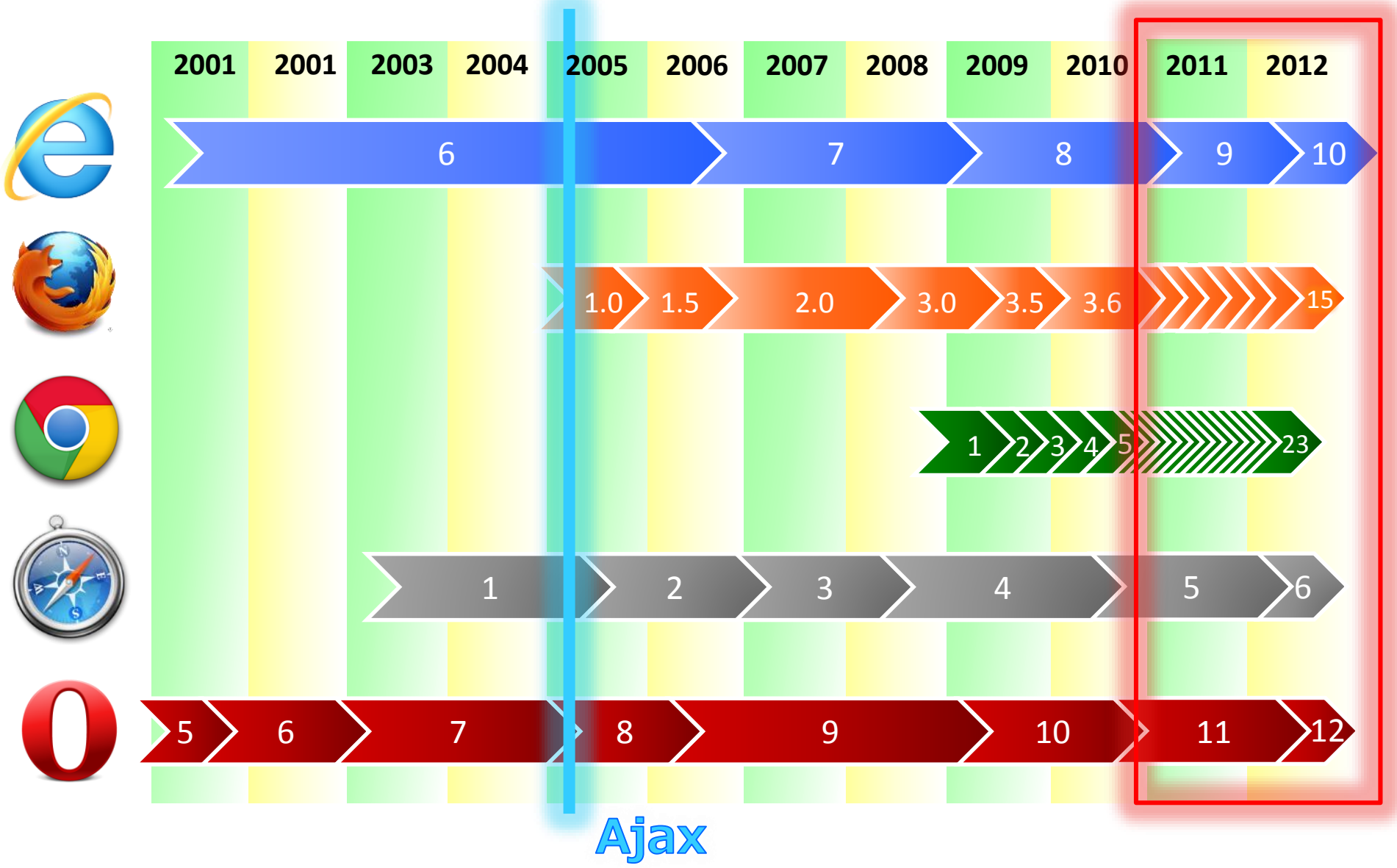
- ネットエージェント株式会社
- 株式会社セキュアスカイ・テクノロジー 技術顧問
- http://utf-8.jp/
- OWASP Kansai Chapter Leader
- OWASP Japan Chapter Advisory Board member

@hasegawayosuke

HTML5時代のWebアプリ

- 次々とリリースされるブラウザ
  - 多数の新しい要素と属性
    - canvas, video, audio, input…
  - 多数の新しいAPI
    - Web Sockets, Web Storage, XHR Lv.2…
  - 最適化されたJavaScriptエンジン
  - 高速化された描画エンジン
- どのブラウザにどの機能が実装されているのか把握できない

次々リリースされるブラウザ



HTML5の新機能

- **マルチメディアのサポート**
  `<video> <audio> <canvas>...`
- **文書構造を表す要素**
  <section> <header> <footer> ...
- **フォームの拡張**
  <input type="email"> ...
- **JavaScript API**
  Web Workers, WebSocket, File...
- **その他…**

HTML5時代のWebアプリ

- HTML5時代のブラウザ
  - 高速化、高機能化
- 実行コードのブラウザ上へのシフト
  - ネイティブアプリからWebアプリへ
  - サーバ側で実行されていた処理がブラウザのJavaScript上へ
- 攻撃もクライアントサイドへシフト
  - JavaScript上の問題点の増加
  - XSSやCSRFなどの比重が増加

Webの技術 楽しいですよね！


クロスサイトスクリプティング
強制ブラウズ
書式文字列攻撃
リモートファイルインクルード
SQLインジェクション
LDAPインジェクション
パストラバーサル
バッファオーバーフロー
CSRF
セッションハイジャック
Webの技術 楽しいですよね！
OSコマンドインジェクション
セッション固定攻撃
オープンリダイレクタ
DoS
HTTPレスポンス分割
メモリリーク
XPathインジェクション
HTTPヘッダインジェクション

HTML5を使った攻撃

- 攻撃側こそ新しいWebの技術をもっとも活用できる
  - クロスブラウザ対応不要！
  - 誰に遠慮することもなく使いたい技術を選んで使える！
  - 多少不安定な技術でもかまわない！

HTML5で増加する脅威

- 攻撃もクライアントサイドへシフト
- JavaScriptを通じた攻撃の比重が増加
- XSSのリスクも増加

> 多くの点から見て、XSS 脆弱性の危険性はバッファ オーバーフローに匹敵します。

セキュリティに関するブリーフィング：Web に対する SDL の適用
http://msdn.microsoft.com/ja-jp/magazine/cc794277.aspx

HTML5で増加する脅威

- XSS
  - HTML5の新要素によるXSS
  - JSコード量の増加 – DOM Based XSS
  - AjaxデータによるXSS
- CSRF
  - XMLHttpRequestで攻撃者有利
- オープンリダイレクタ
  - JavaScriptによるリダイレクトの増加
- その他
  - Ajaxデータからの情報漏えい
  - APIの使い方の問題
    - WebSocket、Web Storage、Web Workers…

今日のはなし

- JavaScript API使用上の注意点
  - WebSocket
  - Web Storage
  - Web Workers

WebSocket

WebSocket

- JavaScriptにおける双方向通信機能

```
var ws = new WebSocket( "ws://example.jp/" );
ws.onopen = function( evt ){
    console.log( "connected" );
};
ws.onmessage = function( evt ){
    console.log( "received:" + evt.data );
};

ws.send( "hello" );
```

WebSocket

- 重要な情報はTLS(wss://)を使うこと
  - httpに対するhttpsと同様

```
// no TLS
var ws = new WebSocket( "ws://example.jp/" );
```

```
// over TLS
var ws = new WebSocket( "wss://example.jp/" );
```

WebSocket

- CookieはHttp/Httpsと共有される
  - http://example.jp/で発行されたCookieは
    - https://example.jp/
    - ws://example.jp:8080/websocket
    - wss://example.jp:8081/websocket

    などで共有される

    (secure属性がない場合)

WebSocket

- Cookieはhttp/httpsと共有される

```
GET /index.html HTTP/1.1
Host: example.jp
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Tue, 1 Jan 2013 09:0
Content-Length: 2524
Content-Type: text/html; c
Set-Cookie: session=12AFE9
....
```

```
GET /websocket HTTP/1.1
Upgrade: websocket
Connection: Upgrade
Host: example.jp
Origin: http://example.jp
Sec-WebSocket-Key: mU6OBz5GKwUgZqbj2OtWfQ==
Sec-WebSocket-Version: 13
Sec-WebSocket-Protocol: chat, superchat
Cookie: session=12AFE9BD34E5A202

HTTP/1.1 101 Switching Protocols
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Accept: IsCRPjZ0Vshy2opkKOsG2UF74eA=
Sec-WebSocket-Protocol: chat
....
```

WebSocket

- secure属性付きのCookieはhttpsとwssで共有される
  - https://example.jp/index.html
  - wss://example.jp:8081/websocket

Web Storage

Web Storage

- JavaScriptでデータを保存する機構
  - Cookieより大容量
  - Cookieと違って自動送信されない
  - JavaScript上で明示的な読み書きが必要

```
sessionStorage.setItem( "foo", "abcdefg" );
var value = sesseionStorage.getItem( "foo" );
```

  - localStorage - 永続的に保持
  - sessionStorage - セッション間だけ保持

Web Storage

- localStorage
  - 明示的に削除しない限りデータを保持
  - 原則、オリジン単位でデータを保持
  - IE8ではhttpとhttpsで共有される
  - Safariではプライベートブラウズ時に読み書きできない

```
sessionStorage.setItem( "foo", "data" ); // 例外発生
alert( sesseionStorage.getItem( "foo" ) );
```

Web Storage

- sessionStorage
  - ブラウザが開かれている間データを保持
  - 新しいタブでは異なるセッション
  - 同一オリジンのiframe、frameではセッションを共有(IE8,9を除く)
  - IE8ではhttpとhttpsで共有される

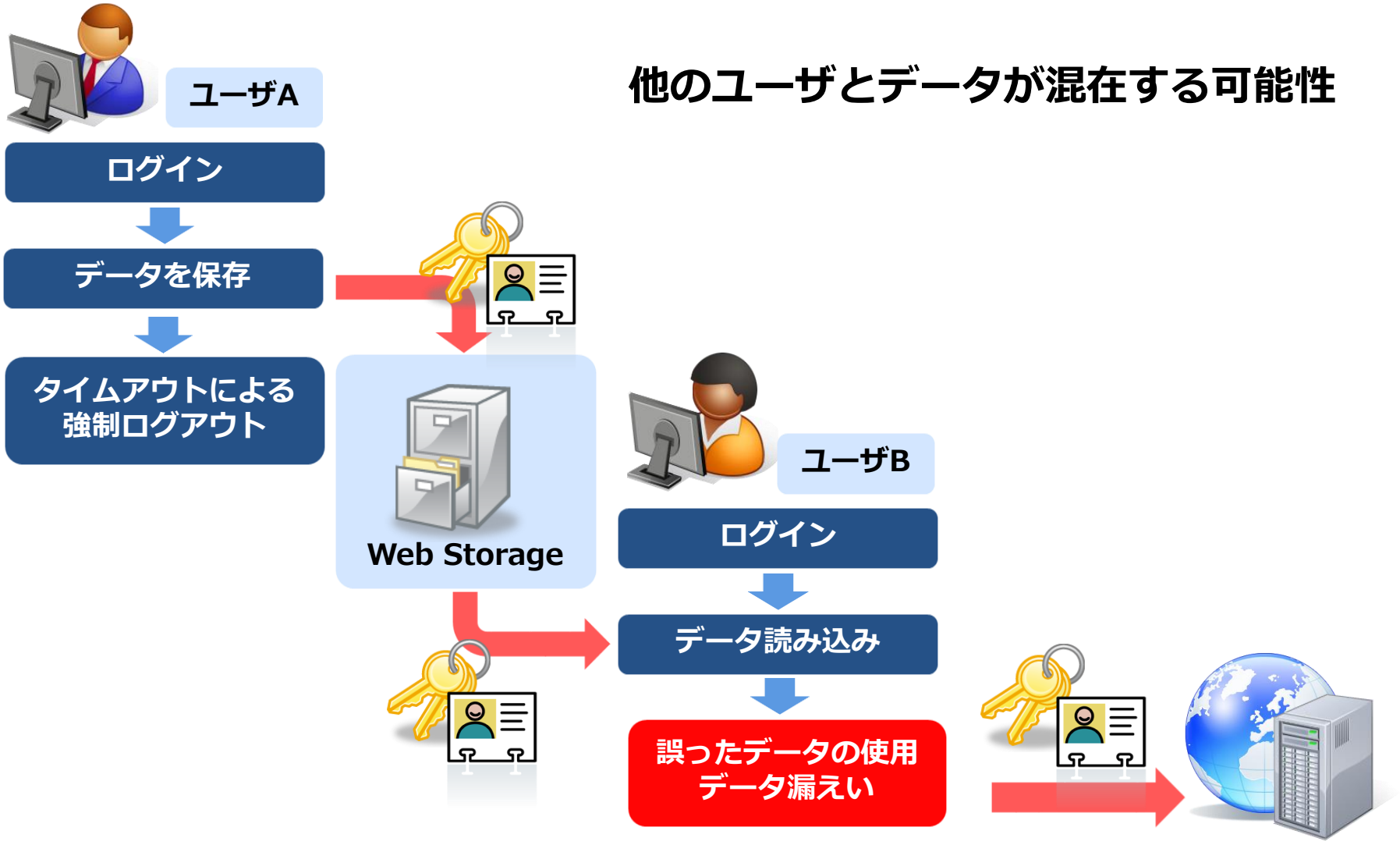
Web Storage

- Cookieと違い、常にJavaScriptから読み書き可能
  - Cookie – httponlyによってJSからアクセス不可
  - XSSが発生したときに盗み見を防ぐことができない
  - パスワードや生年月日などの重要情報をWeb Storageに保存しないこと

Web Storage

- Webアプリケーションの「セッション」と異なるデータ保持期間
  - Webアプリケーション: セッション = ログインからログアウトまで
  - sessionStorage – (原則)タブを閉じるまで
  - localStorage – 無期限
- ログインしたユーザに紐づくデータの保存期間に注意

Web Storage



他のユーザとデータが混在する可能性

Web Storage

- 対策
  - ユーザIDと合わせて保存する

```
sessionStorage.setItem( userid + "-foo", "data" );
```

  - 未ログインでのページロード時やログイン処理時、ログアウト処理時にstorage内のデータを削除する
    (ログアウト処理は確実に走るとは限らない)

Web Workers

Web Workers

- JavaScriptによるバックグラウンド処理機構
  - 多量の計算などの重い処理をUIをブロックせず実行

```
var worker = new Worker( "background-task.js" );
worker.onmessage = function( event ){
    alert( "completed:" + event.data );
};
```

```
// background-task.js
for( var i = 0, sum = 0; i < 1000; i++ ){
    sum += i;
}
postMessage( sum );
```

Web Workers

- 外部からのコードが実行されないよう注意

```
// 脆弱なコード
// http://example.jp/#data:text/javascript,onmessage=...
var src = location.hash.substring(1);
var worker = new Worker( src );
```

  - 攻撃者が用意したスクリプトがWorkerとして実行される
  - ただしDOM操作できないので脅威は低い

Web Workers

- importScriptsも同様

```
//脆弱なコード。importScriptsに任意のURIがわたる
var src = location.hash.substring(1);
var worker = new Worker( 'worker.js' );
worker.postMessage( src );
```

```
// worker.js
onmessage = function( evt ){
    if( evt.data ) importScripts( evt.data );
}
```

まとめ
Conclusion

まとめ

- WebSocket
  - 重要情報はTLSで保護されたwss:スキームで
  - Cookieの共有に注意
- Web Storage
  - 重要情報は保存しない
  - 複数ユーザの意図しないデータ共有に注意
- Web Workers
  - 外部由来のコードをworker srcとして使用しない

参考文献

HTML5 調査報告 from JPCERT/CC

参考文献

- http://utf-8.jp/
  - HTML5セキュリティ その1:
    基礎編、XSS編

    http://utf-8.jp/public/20130613/owasp.pptx
  - HTML5セキュリティ その2:
    オープンリダイレクト、CSRF

    http://utf-8.jp/public/20131114/owasp.pptx

質問タイム

Question ?

Question? 質問

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

@hasegawayosuke

http://utf-8.jp/

@hasegawayosuke