# タイトル未定

– #ssmpjp (ハセガワナイト) –

はせがわようすけ

# 自己紹介

はせがわようすけ

- OWASP Kansai チャプターリーダー
- OWASP Japan アドバイザリボードメンバー
- 株式会社セキュアスカイ・テクノロジー 常勤技術顧問
- CODE BLUE Security Conference Review board member
- セキュリティキャンプ講師 (Webクラス/高レイヤートラック)
- http://utf-8.jp/
  - jjencodeとかaaencodeとか

# 今日の話

- その1 - Edgeで電卓起動
- その2 - Electronで電卓起動
- その3 - X-Content-Type-Optionsのバイパス
- その4 - 実践FiddlerScript

その1

# Edgeで電卓起動

# Edgeで電卓起動

- Edge / IE11 on Windows 10ではデフォルトで calclator:プロトコルハンドラが有効

```
<a href="calculator:">Click here</a>
```

# Edgeで電卓起動

おもしろURLハンドラ

- 設定画面の表示

```
<a href="ms-settings:">設定画面</a>
<a href="ms-settings:network-proxy">プロキシ設定画面</a>
<a href="ms-settings:privacy-microphone">マイク設定画面</a>
```

https://msdn.microsoft.com/ja-jp/library/windows/apps/xaml/dn741261.aspx

- スタートメニュー(Cortana)の表示

```
<a href="ms-cortana:">Cortana</a>
```

- Edgeで開く(IE11から)

```
<a href="microsoft-edge:http://utf-8.jp/">Edgeで開く</a>
```

これ以外にもたくさん

その2

# Electronで電卓起動

ページ削除

その3

# X-Content-Type-Optionsのバイパス

# X-Content-Type-Options

- レスポンスヘッダに付与

```
Content-Type: text/plain; charset=utf-8
X-Content-Type-Options: nosniff

これはテキストファイルです。
<script>alert(1)</script>
```

- IE8+でContent-Typeを厳格に扱う
  - text/html以外をHTMLとして扱うことがなくなる
  - <script src>でJS,VBS以外をスクリプトとして扱わなくなる
  - text/css以外をスタイルシートとして扱わなくなる

# X-Content-Type-Options

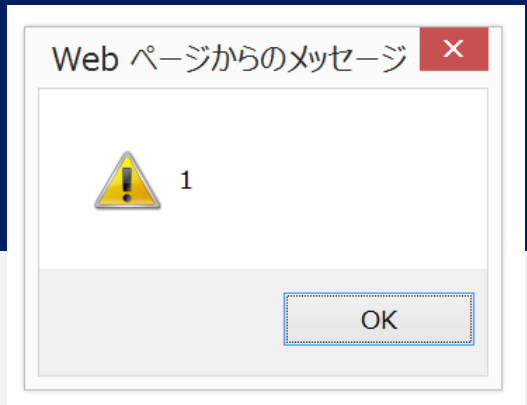
- <script src>でJS、VBS以外のContent-Typeのものはスクリプトとして扱わなくなる

```
Content-Type: text/html
X-Content-Type-Options: nosniff

//<script src=#></script>
alert(1);
```

# X-Content-Type-Optionsのバイパス

- Content-Typeヘッダが存在しない場合や空の場合にはX-C-T-O:nosniffが機能しない

```
Content-Type:
X-Content-Type-Options: nosniff

//<script src=#></script>
alert(1);
```




- Content-Typeヘッダ、ちゃんとつけよう!
  - 通常つけてるから問題ない

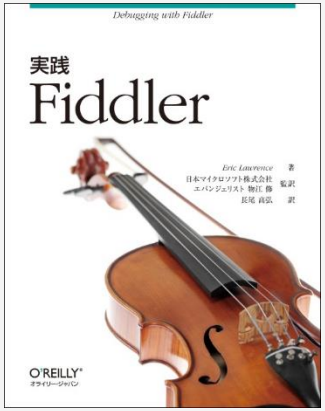
その4

# 実践FiddlerScript

# そもそもFiddlerって何?

Fiddler http://www.telerik.com/fiddler

- デバッグ用ローカルproxyツール
- 元MicrosoftのEric Lawrence作
- スクリプトによる強力なカスタマイズ機能




実践 Fiddler
ISBN978-4-87311-616-7
Eric Lawrence 著、日本マイクロソフト株式会社
エバンジェリスト 物江 修 監訳、長尾 高弘 訳
http://www.oreilly.co.jp/books/9784873116167/

# FiddlerScript何それ

- Fiddlerの挙動やUIをカスタマイズできるスクリプト
  - http://docs.telerik.com/fiddler/KnowledgeBase/FiddlerScript/
  - JScript.NET (JavaScriptでもJScriptでもありません)
- できること
  - リクエストやレスポンスの書き換え
  - HTTPヘッダの追加、削除
  - 表示色の変更、メニュー項目の追加
  - その他いろいろ

# FiddlerScriptのデバッグ

- ブレークポイントを置いてステップ実行、みたいなのは簡単にはできない(たぶん)
- できなくはない(かも知れない)
  - Visual StudioからFiddlerプロセスをアタッチし
  - FiddlerScript内からSystem.Diagnostics.Debugger.Break()を呼び出す
- 出来る方法知ってる人がいたら教えて!

# FiddlerScriptのデバッグ

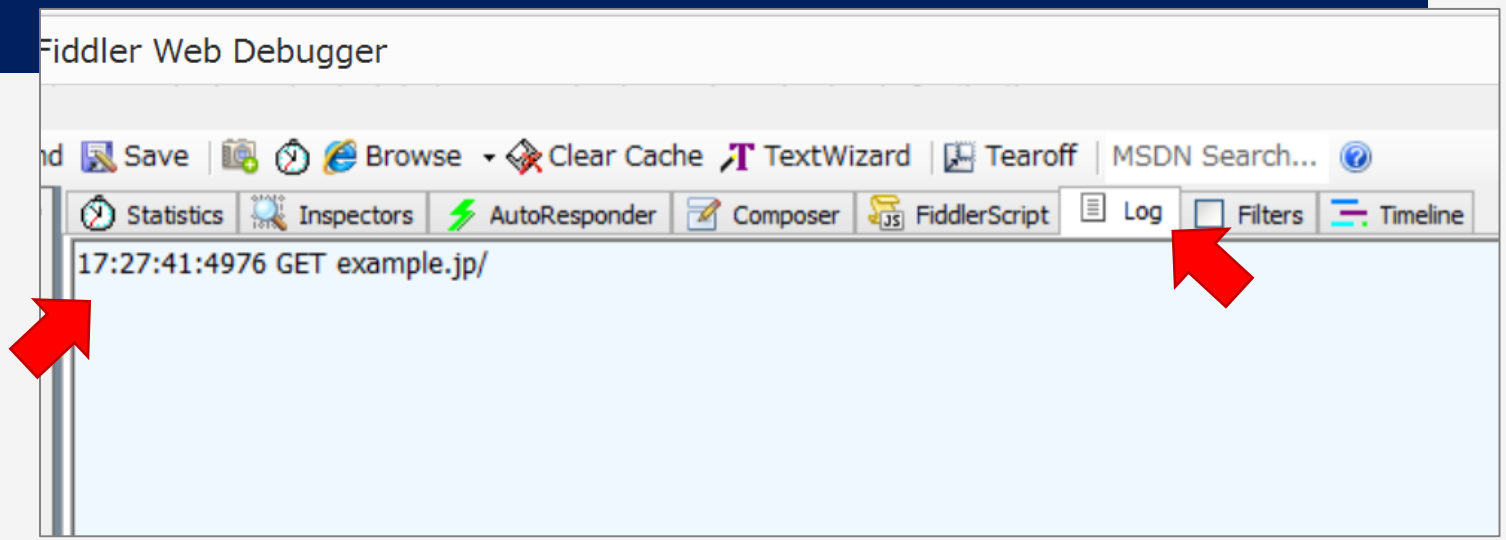
基本的にはprintfデバッグあるのみ！

- FiddlerObject.log
  - ログメッセージの記録
- FiddlerApplication.Log.LogFormat
  - フォーマット付きログ
- FiddlerObject.statusText
  - ステータスバーに1行表示
- FiddlerObject.alert
  - みんな大好き alert

# FiddlerScriptのデバッグ

- FiddlerObject.log( msgText )
  - Logペインに記録される

```
static function OnBeforeRequest(oSession: Session) {
  FiddlerObject.log(
      oSession.RequestMethod + " " + oSession.url
  );
  ....
}
```



# FiddlerScriptのデバッグ

- FiddlerApplication.Log.LogFormat ( FormatText, arguments )

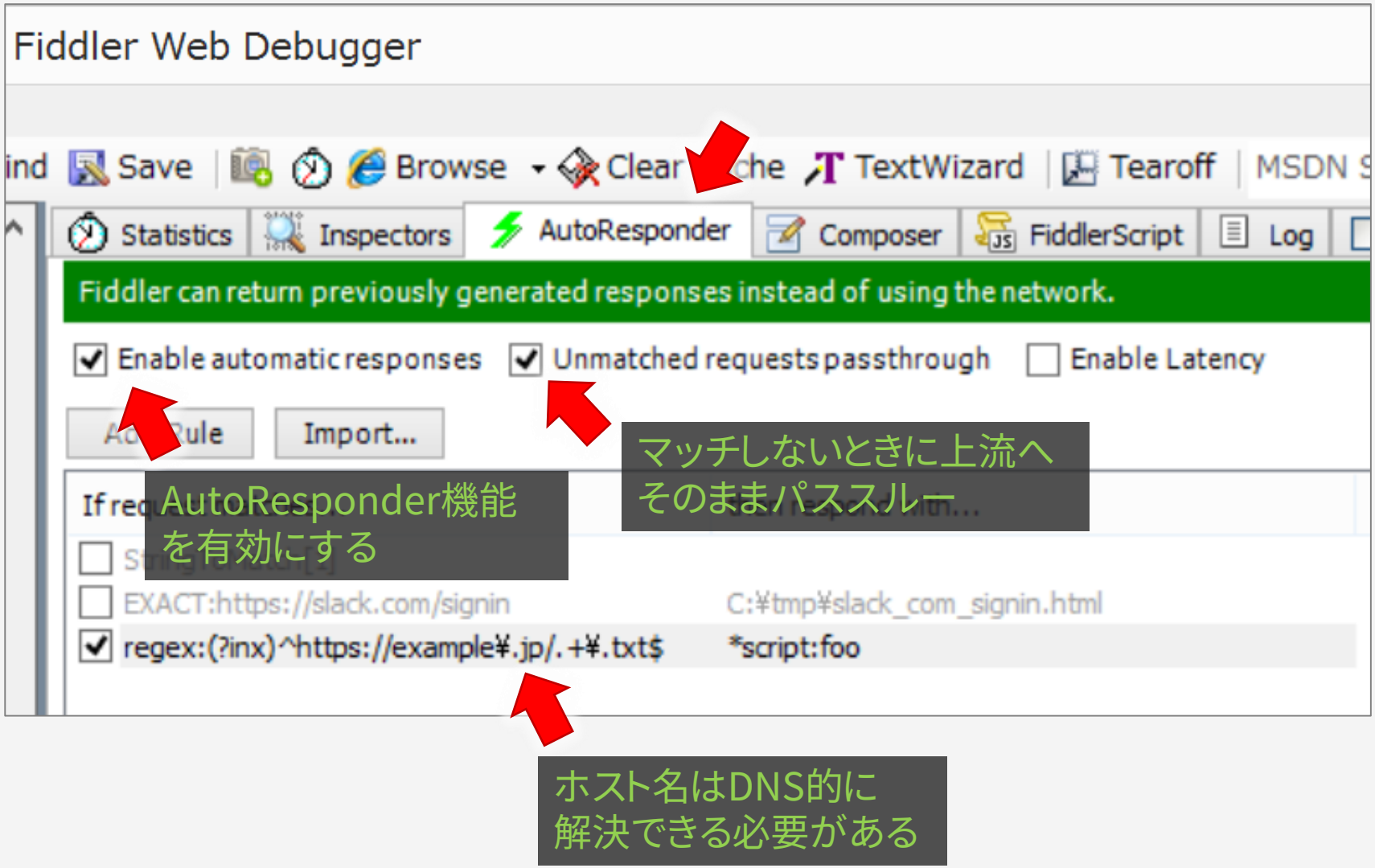
```
static function OnBeforeRequest(oSession: Session) {
    FiddlerApplication.Log.LogFormat(
        "p1:{0}, p2:{1}", param1, param2
    );
    ....
}
```

# AutoResponder機能

AutoResponder - Fiddlerの強力な機能のひとつ

- 指定された条件に一致したリクエストのときに
  - URLが指定されたものと一致
  - URLが正規表現で指定されたものにマッチ
  - 指定されたリクエストヘッダを含む
    など
- レスポンスを生成して返す
  - 事前に指定されたステータスやファイル
  - 他のサーバへリダイレクト
  - レスポンスヘッダの追加
  - スクリプトの実行

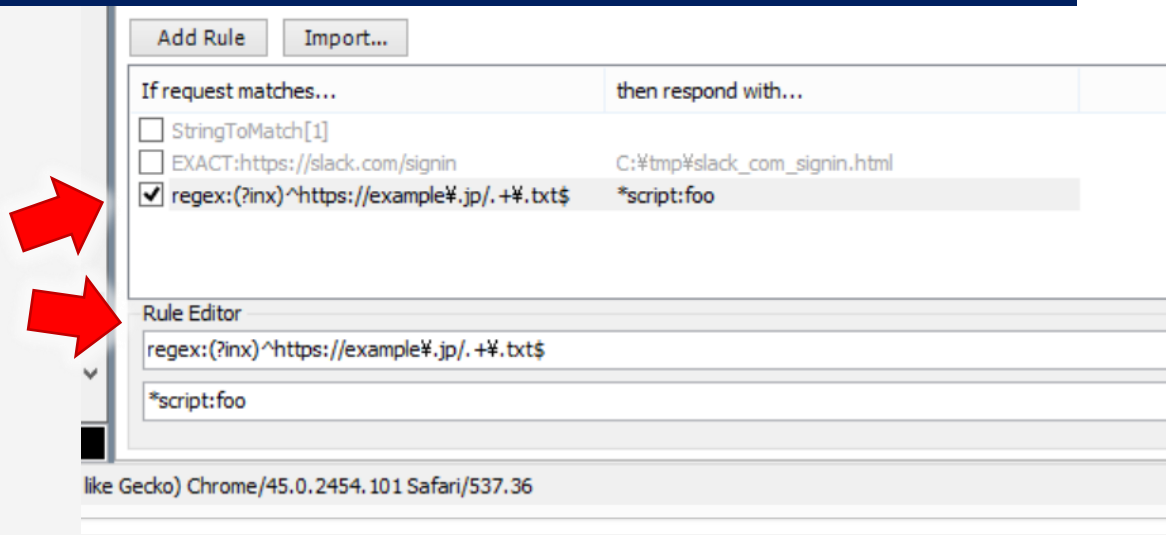
# AutoResponder機能



# AutoResponderでスクリプト実行

```
public static function foo( oSession: Session ){
    oSession["ui-backcolor"] = "red";
    oSession.utilCreateResponseAndBypassServer();
    oSession.oResponse.headers.Add( "Content-Type", "text/plain" );
    oSession.ResponseBody =
        System.IO.File.ReadAllBytes("C:/tmp/File.txt");
}
```

https://example.jp/*.txtの
ときには関数fooを実行



# AutoResponderでスクリプト実行

- FiddlerScript内でリクエストに応じた応答

  つまり

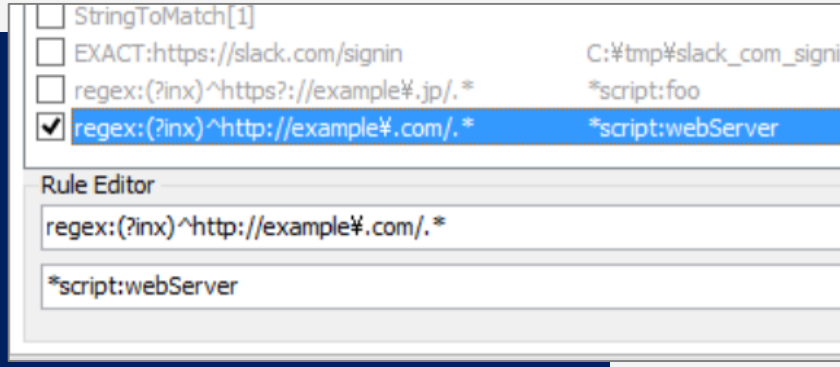
  FiddlerScriptだけでWebアプリ書けるんじゃね？

Web Apps on FiddlerScript

# DEMO

# FiddlerScriptでWeb Server

```
public static function webServer( oSession: Session ){
    var documentRoot:String = "c:/tmp/htdocs";
    var path = oSession.PathAndQuery.replace( /¥?.*/g, "" );
    var realPath = documentRoot + path;
    var contentTypes = {
        "txt" : "text/plain",
        "html" : "text/html",
        "js" : "application/javascript",
        "css" : "text/css"
    };
    oSession.utilCreateResponseAndBypassServer();

    if( path.indexOf( "../" ) >= 0 || path.indexOf( "..¥¥" ) >= 0 ){
        oSession.oResponse.headers.Add( "Content-Type", "text/html;charset=utf-8" );
        oSession.oResponse.headers.HTTPResponseCode = 500;
        oSession.oResponse.headers.HTTPResponseStatus = "500 Internal Server Error";
    }else if( System.IO.File.Exists( realPath ) ){
        var contentType = "application/octet-stream";
        var ext = ( /¥.(.+)$/.exec( path ) || [] )[ 1 ];
        if( ext && contentTypes[ ext ] ) contentType = contentTypes[ ext ];
        oSession.oResponse.headers.Add( "Content-Type", contentType );
        oSession.ResponseBody = System.IO.File.ReadAllBytes( realPath );
    }else{
        oSession.oResponse.headers.Add( "Content-Type", "text/html;charset=utf-8" );
        oSession.oResponse.headers.HTTPResponseCode = 404;
        oSession.oResponse.headers.HTTPResponseStatus = "404 Not Found";
    }
}
```




「../」が含まれるときは500を応答

ファイルが実際に存在するとき

デフォルトのC-Tは
application/octet-stream

ファイルを読み込んで返す

ファイルが存在しないときは404

# FiddlerScriptで計算結果を返す

```
public static function calc( oSession: Session ){
    var buf = new System.Text.StringBuilder();
    var q = oSession.PathAndQuery.replace( /[^￥?]*￥?/, "" );
    var template = System.IO.File.ReadAllText( "C:/tmp/template.html" );
    var ans = "invalid";
    if( /^[￥d￥(￥)￥+￥-￥/￥*￥s]+$/.test( q ) ){
        try{
            ans = eval( q );
        }catch( e ){
        }
    }
    buf.Append(
        template.replace( /%(exp|ans)%/g, function( s, p ){
            if( p === "exp" ){
                return htmlEscape( q );
            }else if( p === "ans" ){
                return htmlEscape( ans );
            }else{
                return "";
            }
        } )
    );
    oSession["ui-backcolor"] = "yellow";
    oSession.utilCreateResponseAndBypassServer();
    oSession.oResponse.headers.Add( "Content-Type", "text/html;charset=utf-8" );
    oSession.ResponseBody = System.Text.Encoding.UTF8.GetBytes( buf );
}
```

テンプレート読み込み

URLクエリが数値と演算子のみならevalで計算

テンプレートの展開

Fiddlerの該当行を黄色背景に

結果のHTMLを応答

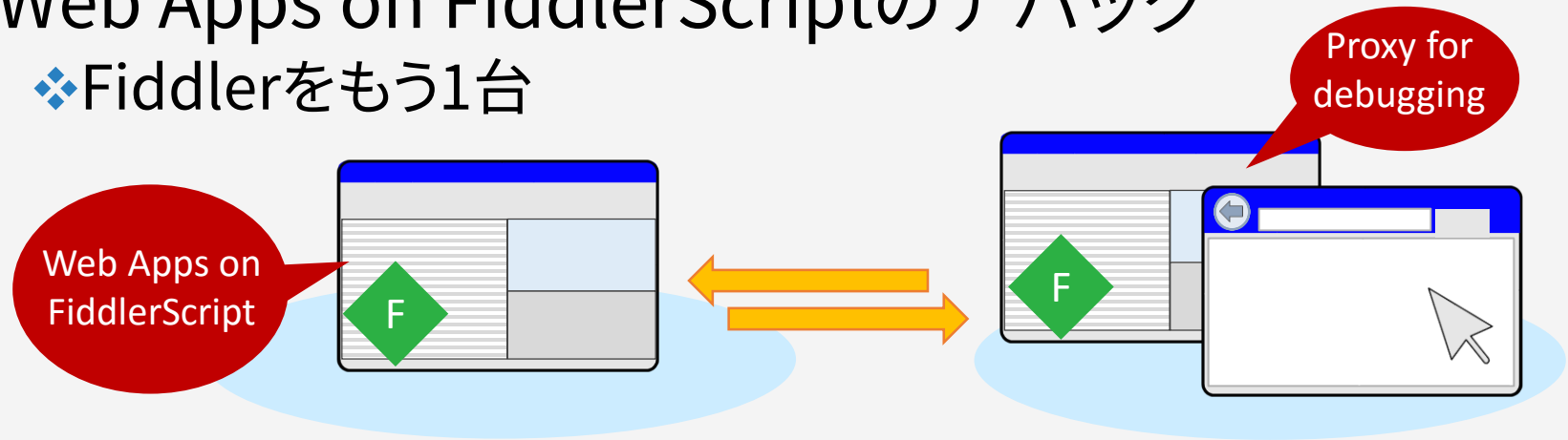
# Web Apps on FiddlerScript

- FiddlerScript内にWebアプリを書くことで
  - Webサーバ不要
  - .NETな機能を使い放題
  - JavaScriptぽさも使い放題(?)

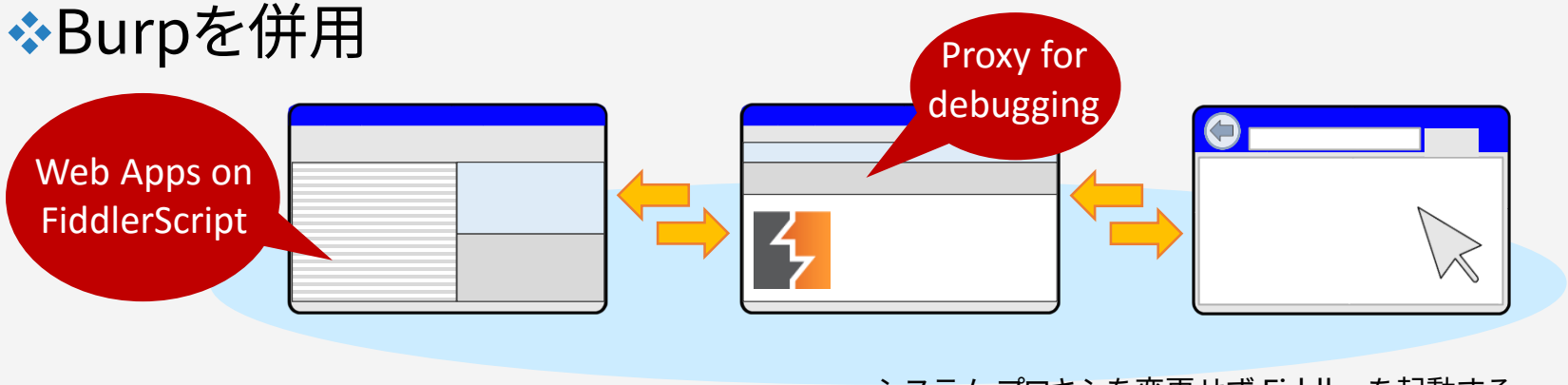
  - アプリごとにCustomRules.jsを手動で置き換え
  - 遅い
  - デバッグしにくい

# Web Apps on FiddlerScript

- Web Apps on FiddlerScriptのデバッグ
  - Fiddlerをもう1台




  - Burpを併用




システム プロキシを変更せず Fiddler を起動する
https://hebikuzure.wordpress.com/2012/07/23/

# Question?

hasegawa@securesky-tech.com

@hasegawayosuke

http://utf-8.jp/