

IE8 以降の「F12 開発者ツール」を使った DOM Based XSS の検査法について

NTT コミュニケーションズ株式会社
ソリューションサービス部
第四エンジニアリング部門
セキュリティオペレーション担当

2011 年 09 月 05 日

Ver. 1.0



1. 調査概要.....	3
2. F12 開発者ツール.....	3
2.1. F12 開発者ツール	3
3. F12 開発者ツールを使って DOM BASED XSSを確認する	5
3.1. サンプルHTML.....	5
3.2. HTMLロード直後のスクリプト	6
3.3. ユーザイベント駆動型のスクリプト	12
4. まとめ.....	19
5. 検証作業者	19
6. 参考	19
7. 履歴.....	19
8. 最新版の公開URL.....	20
9. 本レポートに関する問合せ先.....	20

1. 調査概要

Microsoft Internet Explorer(以下 IE)の version8 以降には、Web サイトの開発者向けに「開発者ツール」または「F12 開発者ツール」という機能が同梱されている。本文書では、この機能を使い、DOM Based XSS を検査する初歩的方法について、記述する。

2. F12 開発者ツール

2.1. F12 開発者ツール

「F12 開発ツール」とは、IEに対して「F12」を押す事で起動するデバッグ・ツールである(図 2.1-1～図 2.1-3)。

このような開発ツールによって Web アプリケーションのクライアント側スクリプトや Web ページのデザインのデバッグ環境は、大幅に改善されるだろう。

さて、DOM Based XSS は、Web ブラウザ上の DOM に対しての XSS(Cross-Site Scripting) 攻撃である。一般的な Web アプリケーションの診断手順とは、異なる箇所が発現するため、DOM Based XSS の診断作業には、このような Web ブラウザ上のデバッグ・ツールは欠かせない。



図 2.1-1: 「F12」キー、または「ツール」→「開発者ツール」で起動できる

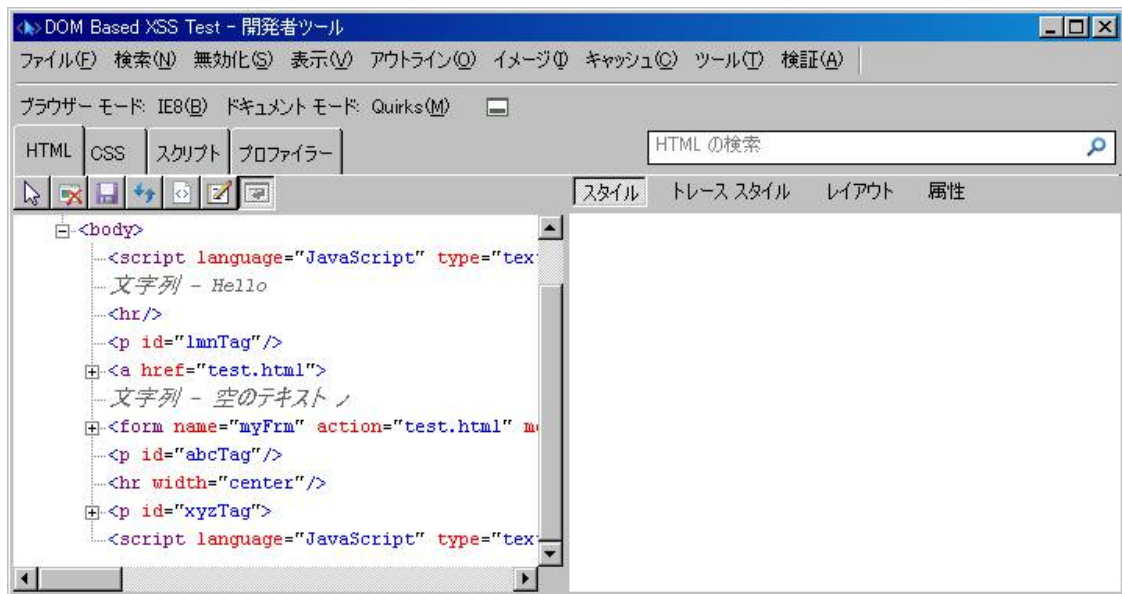


図 2.1-2: IE8 の「F12 開発者ツール」。

「CTRL」 + 「P」 キーで、図 2.1-3 のような画面になる

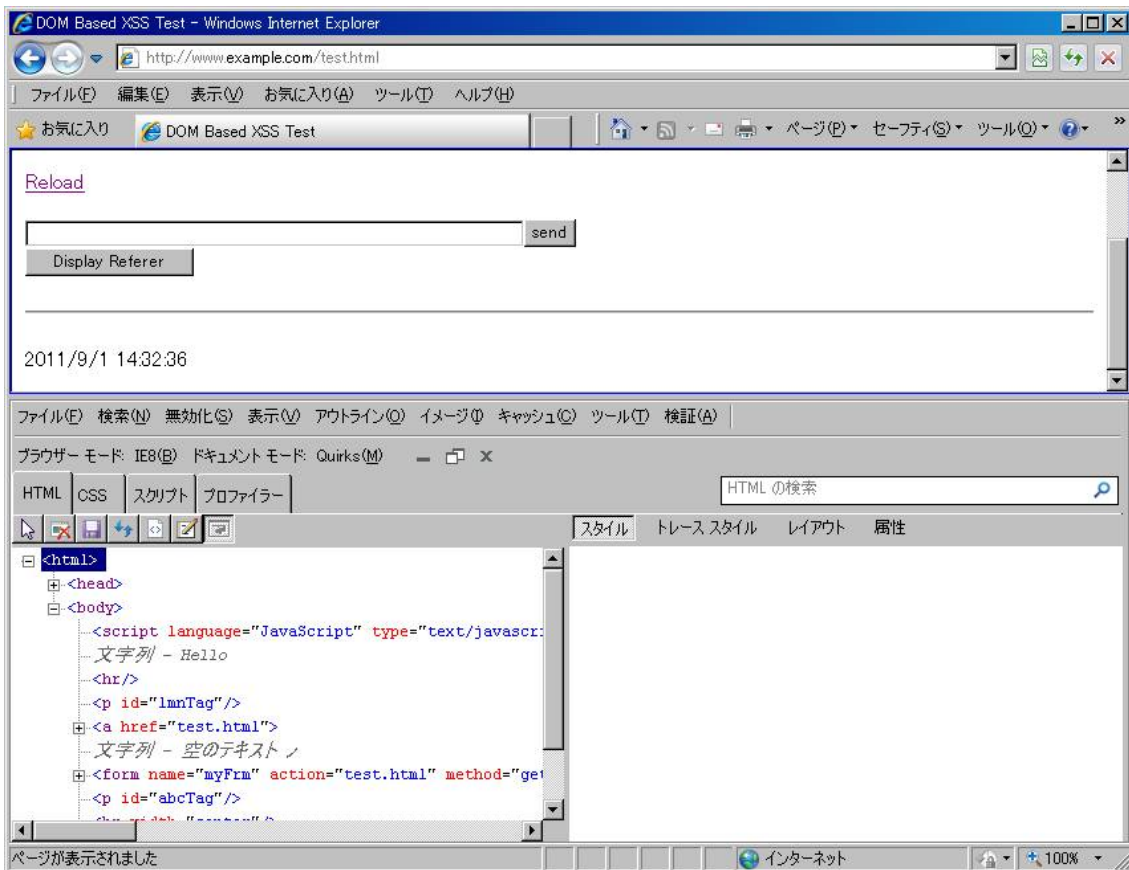


図 2.1-3: IE8 の「F12 開発者ツール」。(IE と一体化されている)

「CTRL」 + 「P」 キーで、図 2.1-2 のような画面になる

3. F12 開発者ツールを使って DOM Based XSSを確認する

3.1. サンプルHTML

今回は、非常にシンプルな「<http://www.example.com/test.html>」(図 3.1-1)を対象にして、DOM Based XSS の確認していく。

シンプルとはいえ、setTimeout()関数を使い、スクリプトをバックグラウンドで実行している部分も確認できる。

```

<html>
<head><title>DOM Based XSS Test</title></head>
<body>
<script language="JavaScript" type="text/javascript">
  <!--
    document.write("Hello");
  // -->
</script>
<hr>
<p id="lmnTag"></p>
<A HREF="test.html">Reload</A>
<form name="myFrm" METHOD="get" ACTION="test.html">
  <input type="text" name="myText" size="72"><input type="submit" name="send" value="send"><br>
  <input type="button" value="Display Referer" onClick="test()">
</form>
<p id="abcTag"></p>
<hr width="center">
<p id="xyzTag"></p>
<script language="JavaScript" type="text/javascript">
  var Obj1 = document.getElementById("lmnTag");
  Obj1.innerHTML = document.referrer;

  function test() {
    var Obj = document.getElementById("abcTag");
    Obj.innerHTML = document.referrer;
  }

  function myNow() {
    var myObj = document.getElementById("xyzTag");
    var myDate = new Date();
    var str = "" + myDate.getYear() + "/" + (myDate.getMonth()+1) + "/" + myDate.getDate();
    str += " " + myDate.getHours() + ":" + myDate.getMinutes() + ":" + myDate.getSeconds();
    myObj.innerHTML = str;
  }

  function myTimer() {
    myNow();
    setTimeout('myTimer()', 1000);
  }

  setTimeout('myTimer()', 1000);
</script>
</body>
</html>

```

図 3.1-1: 「<http://www.example.com/test.html>」

まあ、この程度なら「F12 開発者ツール」を使うまでもないかもしれない

```
<html>
<head><title>Post Page</title>
</head>
<body>
  <form action="test.html" method="get">
    <input type="text" name="myText" size="72">
    <input type="submit" name="send" value="send">
  </form>
</body>
</html>
```

図 3.1-2: 「http://www.example.com/test2.html」

test.html ヘデータを送信する入力画面

3.2. HTMLロード直後のスクリプト

今回のサンプル HTML は、HTML ロード時に実行されるスクリプトで「Referer」の値を画面に書き出していることが確認できる。

この節では、HTML ロード時に実行されるスクリプト中に含まれる DOM Based XSS について考察する。

手順は、以下の通りとなる。

ひとまず該当ページにアクセスし、「F12 開発者ツール」を起動し、コードのなるべく先頭部分にブレークポイントを設定し、「デバッグ開始」を押下する(図 3.2-1)。

一つ前のページへ戻り、再度アクセスする(図 3.2-2→図 3.2-3)。

図 3.2-3で、ブレークポイントのところで処理が停止するので、ステップ実行などでコード解析を行い、XSSが発現する可能性のある部分を探し出す(図 3.2-4)。

図 3.2-4には「innerHTML」というDOM Based XSSを誘発させやすいプロパティ名を確認したので、そこにブレークポイントを設定する(図 3.2-5)。

再度、一つ前の画面に戻り、図 3.2-5ではRefererの値がそのままinnerHTMLの値にコピーされているようなので、この一つ前の画面からRefererにXSSコードを仕込んでみる(図 3.2-6)。

つまり、図 3.2-6では、アドレス欄に直接クエリ文字列を与えて「test2.html」を呼び出した後で、「test2.html」から「test.html」へ遷移する(「send」ボタン)。

その後は、図 3.2-7～図 3.2-8のようにブレークポイントのところで停止する。「F12 開発者ツール」では実行中のスクリプトが扱っているDOMの状態を細やかに確認することができる。

最終的には、図 3.2-9のように、XSSを発現させることができた。

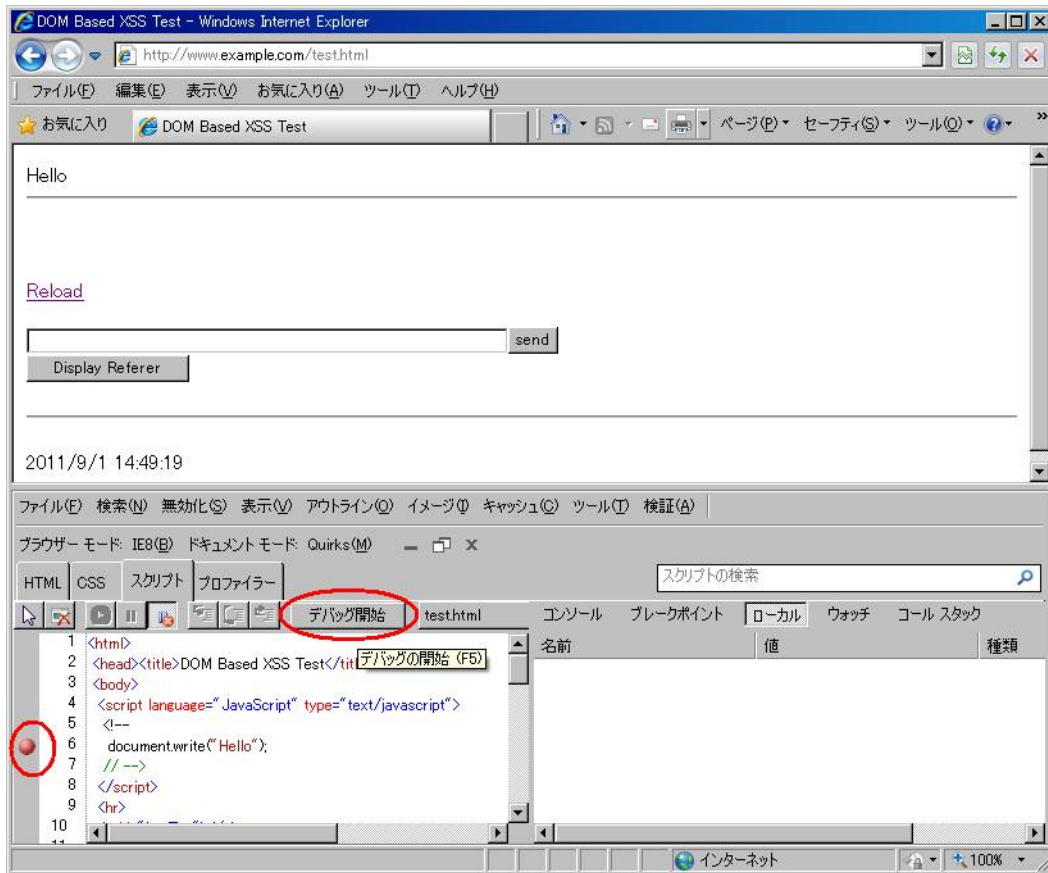


図 3.2-1: とりあえず該当ページにアクセスし、「F12 開発者ツール」を起動する。

なるべく先頭部分にブレークポイントを設定する

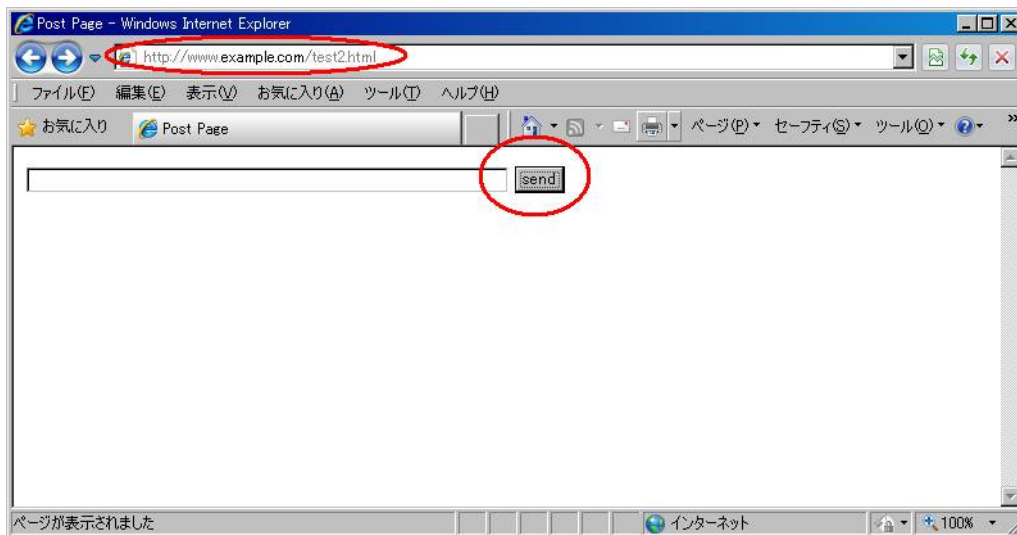


図 3.2-2: 図 3.2-1後、すばやく一つ前の画面に戻り、再度アクセスする

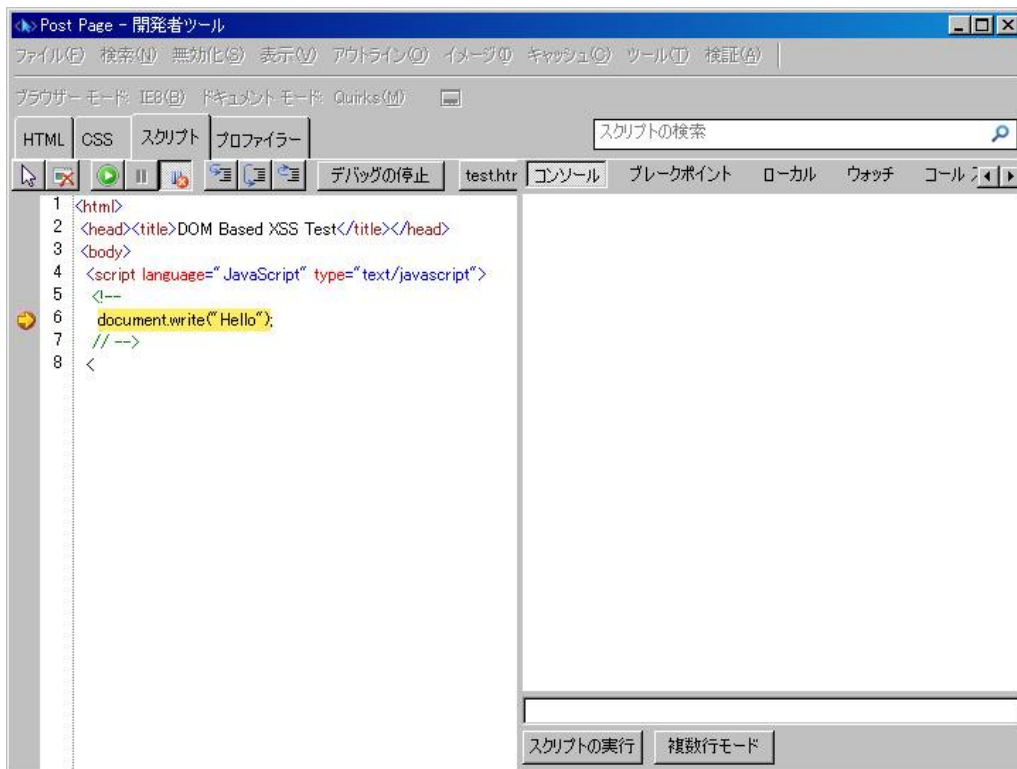


図 3.2-3 : 図 3.2-2後の画面。このように、ロード直後に実行されるスクリプトにもブレークポイントは設定できる

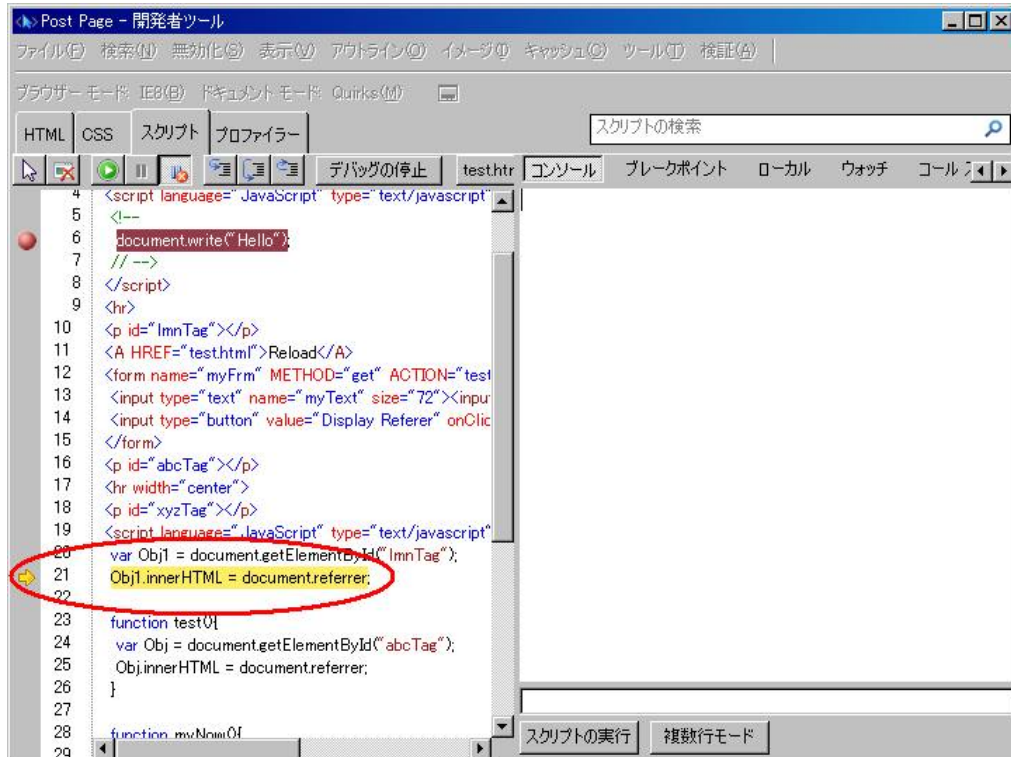


図 3.2-4 : 図 3.2-3後、ステップ実行したところで、注目部分にブレークポイントを設定する。

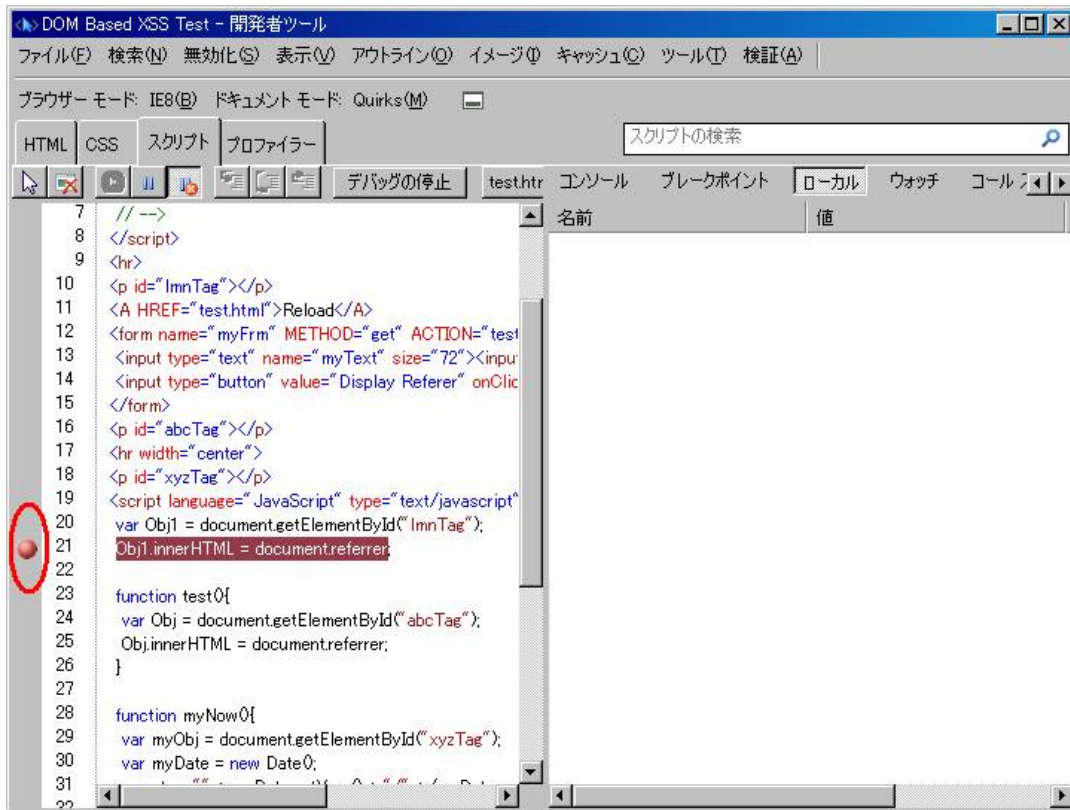


図 3.2-5 : 図 3.2-4からのつながりで、ここにブレークポイントを設定する

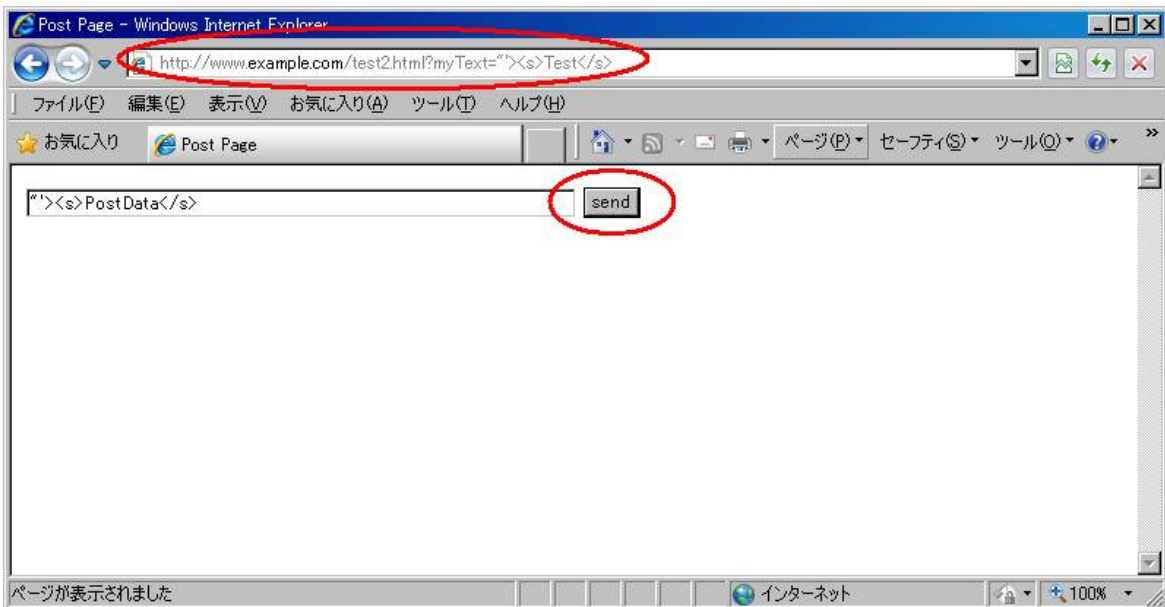


図 3.2-6 : 図 3.2-5後、再度、一つ前の画面に戻る。

図 3.2-5では、Refererの値にXSS攻撃コードを仕込めそうなので、実際に仕込んでみる
(今回は、診断作業として脆弱性の有無を確認できる程度のコード)

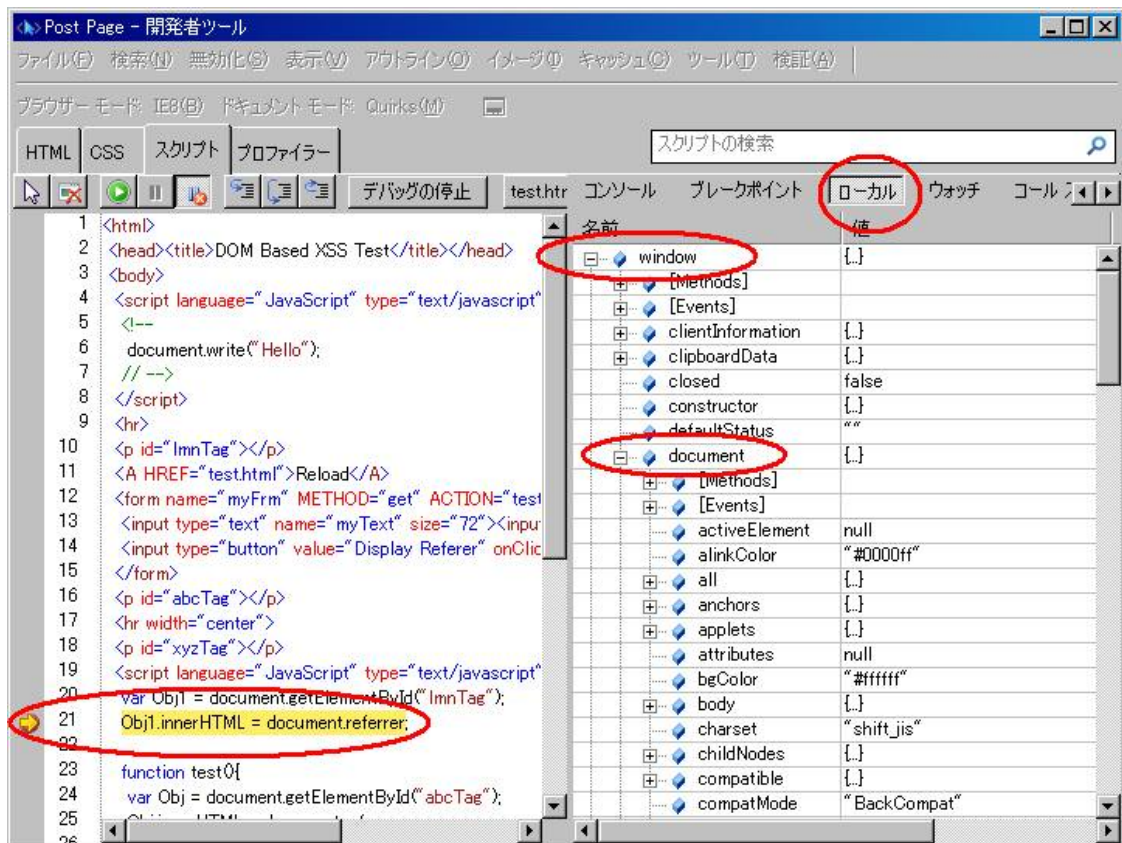


図 3.2-7 : 図 3.2-6後の画面 1。test.htmlのブレークポイントで処理が一時停止する

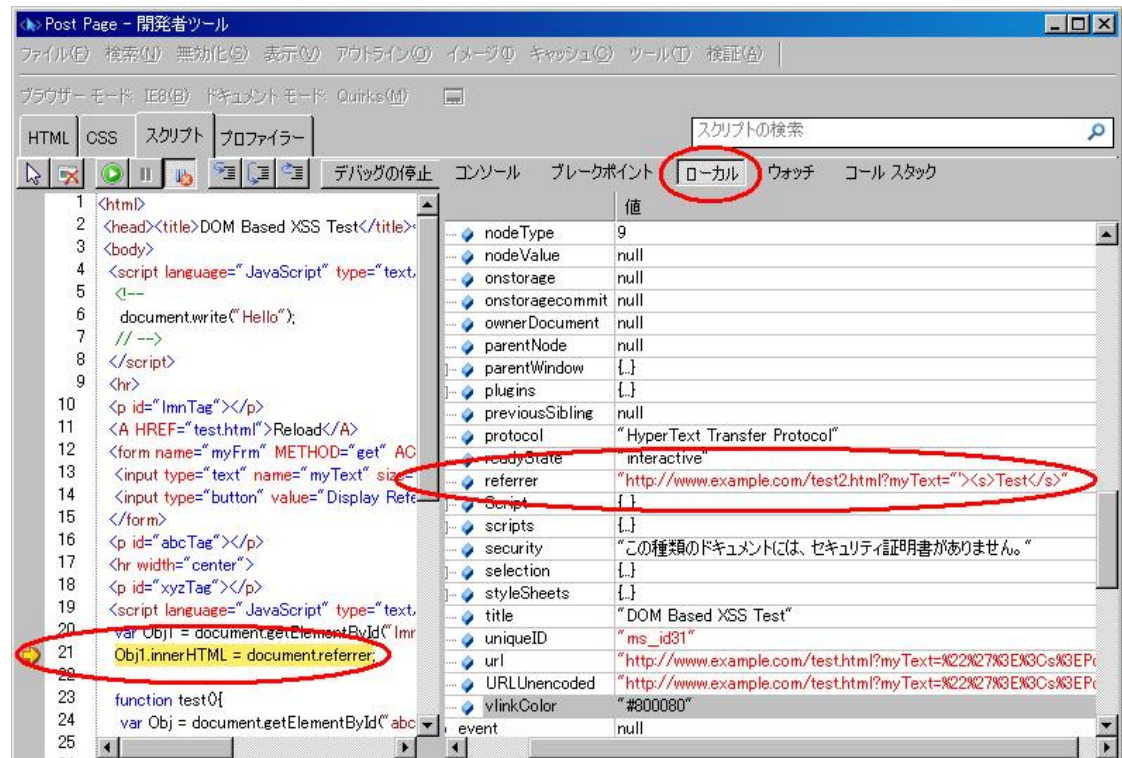


図 3.2-8 : 図 3.2-6の後の画面 2。「F12 開発者ツール」にはDOMの状態を細かく確認できる

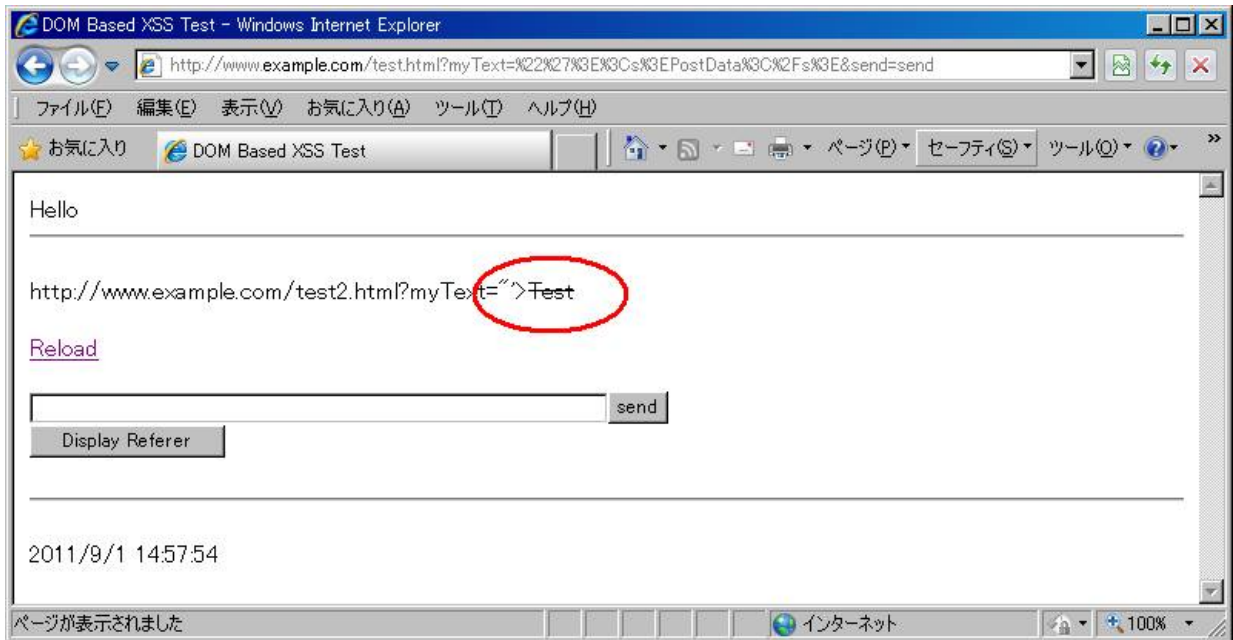


図 3.2-9 : 図 3.2-7～図 3.2-8の結果。XSSが発現した。

3.3. ユーザイベント駆動型のスクリプト

この節では、ボタンなどの押下処理時の DOM Based XSS に対して考察する。

まずは、HTMLのソースを確認し、ボタン押下など、どのJavaScript関数が呼び出されるかを確認する(図 3.3-1)。

図 3.3-1から、test()関数がボタン押下によって、呼び出されることから、test()関数にブレークポイントを設定し、「デバッグ開始」ボタンを押下する(図 3.3-2)。

そして、HTML上のボタン(「Display Referer」ボタン)を押下すると、test()関数が実行され、指定したブレークポイントで処理が一旦停止する(図 3.3-3)。

さらに、ステップ実行(「F11」キー)すると、図 3.3-4のようになり、DOM Based XSSとして注目すべき「innerHTML」プロパティが登場した。また、その値に「Referer」がセットされることが分かった。さらには、「ローカル」というプロパティで、DOM の状態を確認することも可能だ。

以上より、Referer の値に XSS を引き起こすような情報を埋め込むことができれば、上記のところで、XSS を発生させることが可能ではないかと推定できる。

よって、今度はXSSを引き起こすような情報をWebブラウザが「Referer」として認識させることが可能かという点であるが、「send」ボタンによって、任意のクエリ文字列を与えることが可能であるため、実際に(XSSを引き起こすような)文字列をセットし、XSSを誘発してみる(図 3.3-5～図 3.3-12)。

最初におこなったことは、そのままのクエリ文字列を使ってみることである(図 3.3-5～図 3.3-8)。

つまり、テキストボックスにXSS試験用文字列を埋め込み(図 3.3-5)、サーバへリクエストを送りページ書き換えを実施し、WebブラウザにRefererを記憶させる(図 3.3-6～図 3.3-7)。

その上で「Display Referer」を押下してみたが、結果はうまく行かなかった(図 3.3-8)。

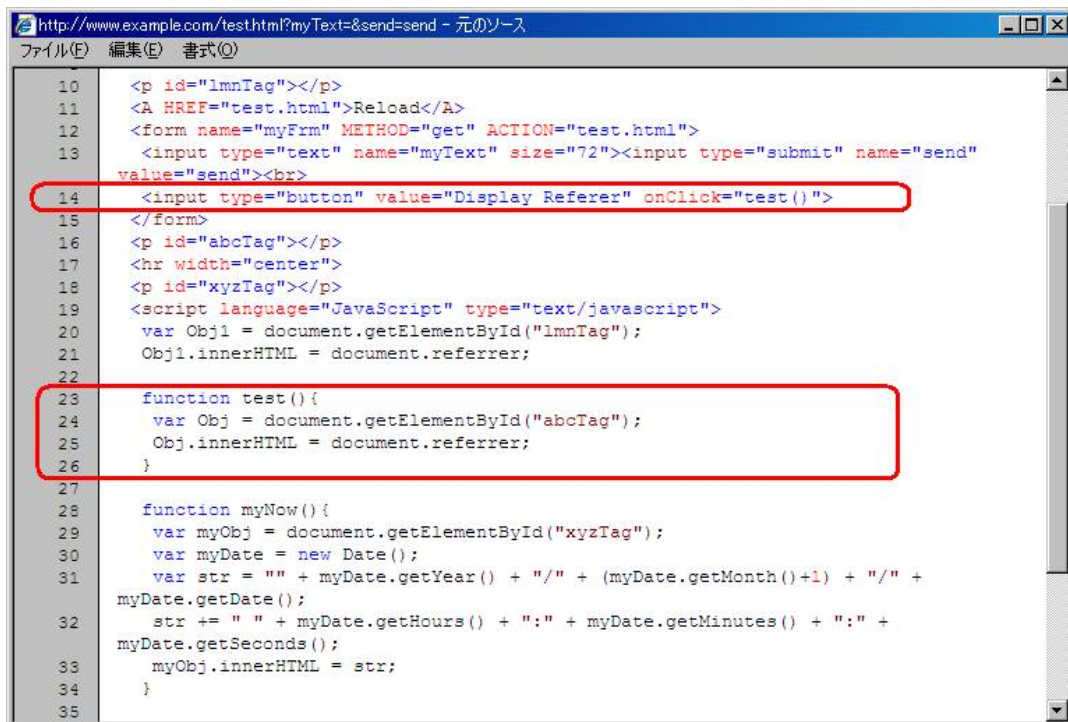
クエリ文字列が URL エンコードされてしまうことが原因である。

次は、URLエンコードされないようにアドレス欄に直接入力してみる(図 3.3-9～図 3.3-12)。

まず、アドレス欄に直接クエリ文字列を記述し、ページ書き換えを行う(図 3.3-9)。

図 3.3-9の状態ではアドレス欄のURLは、まだRefererの値とはなっていないので、「send」ボタンを押下し、もう一度ページ書き換えを行い、WebブラウザにRefererとして認識させる(図 3.3-10)。

この状態で、「Display Referer」を押下げると、「F12 開発者ツール」のブレークポイントで処理が停止する(図 3.3-11)。さらに処理を進めた結果が、図 3.3-12であり、XSSを発現させることに成功した。



```

10 <p id="lmnTag"></p>
11 <A HREF="test.html">Reload</A>
12 <form name="myFrm" METHOD="get" ACTION="test.html">
13 <input type="text" name="myText" size="72"><input type="submit" name="send"
value="send"><br>
14 <input type="button" value="Display Referer" onClick="test()">
15 </form>
16 <p id="abcTag"></p>
17 <hr width="center">
18 <p id="xyzTag"></p>
19 <script language="JavaScript" type="text/javascript">
20 var Obj1 = document.getElementById("lmnTag");
21 Obj1.innerHTML = document.referrer;
22
23 function test(){
24 var Obj = document.getElementById("abcTag");
25 Obj.innerHTML = document.referrer;
26 }
27
28 function myNow(){
29 var myObj = document.getElementById("xyzTag");
30 var myDate = new Date();
31 var str = "" + myDate.getYear() + "/" + (myDate.getMonth()+1) + "/" +
myDate.getDate();
32 str += " " + myDate.getHours() + ":" + myDate.getMinutes() + ":" +
myDate.getSeconds();
33 myObj.innerHTML = str;
34 }
35
    
```

図 3.3-1: 「HTML ソース」を確認する。

ボタンクリックによって「test()」という関数が呼び出される

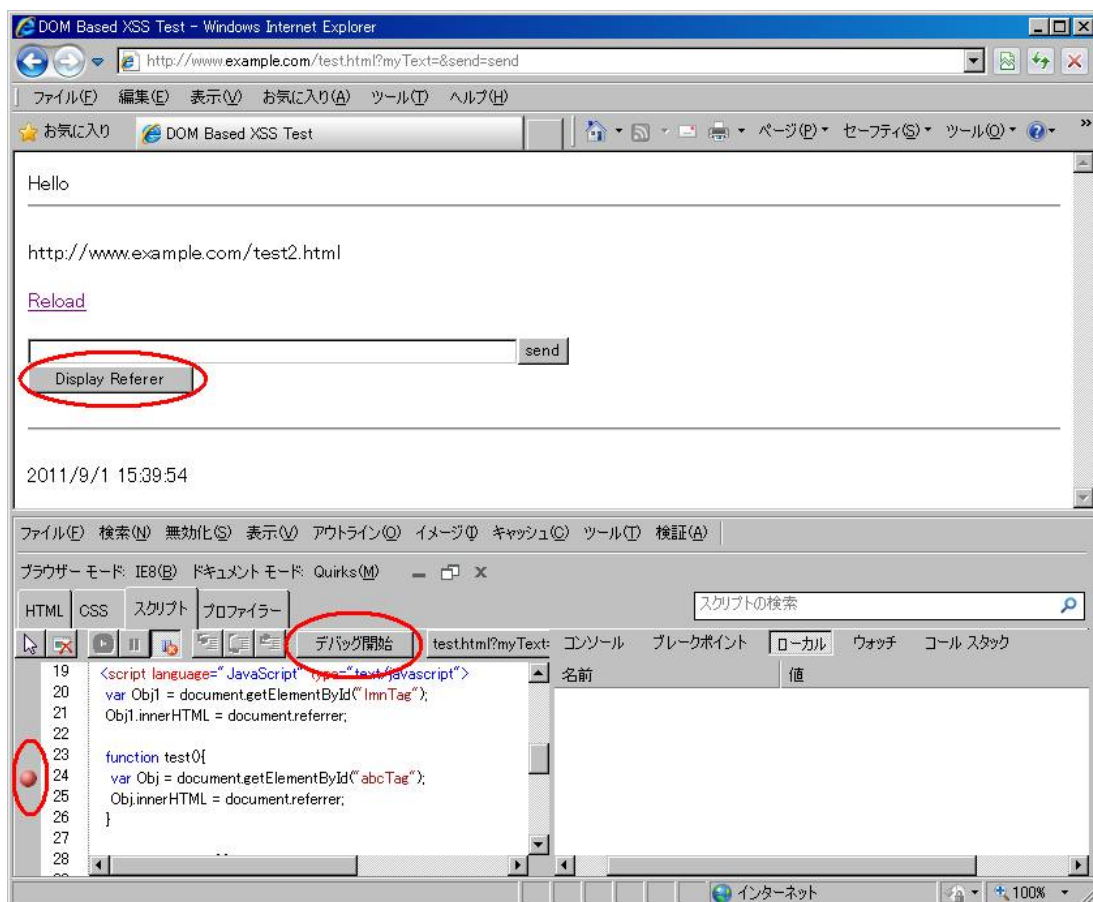


図 3.3-2: 「test()関数」にブレークポイントを設定し、「デバッグ開始」を押下する

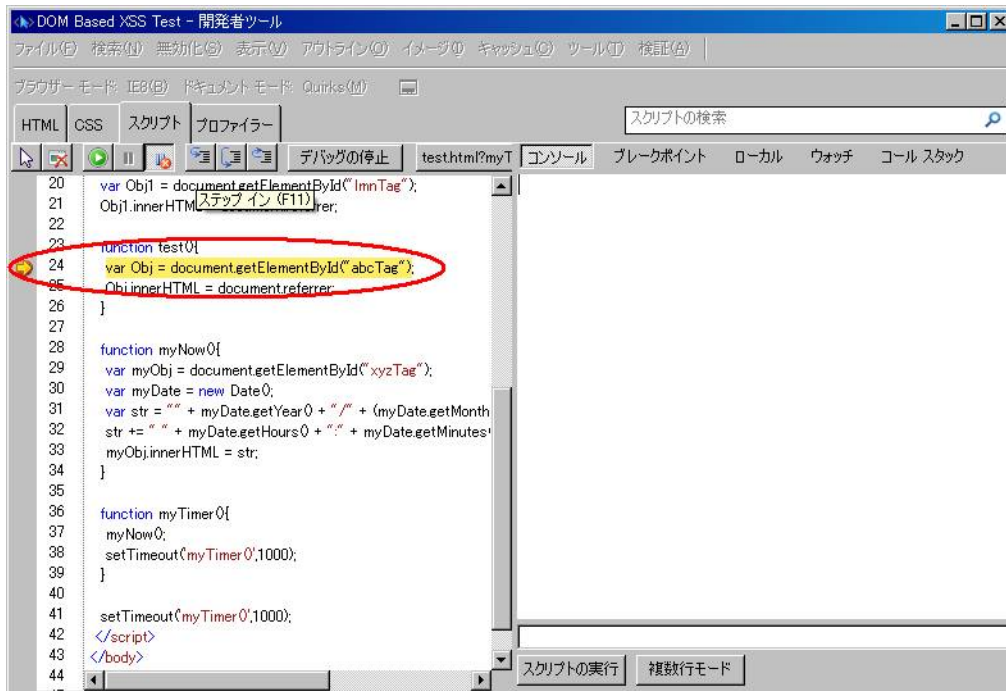


図 3.3-3 : 図 3.3-2で設定したブレークポイントで停止する

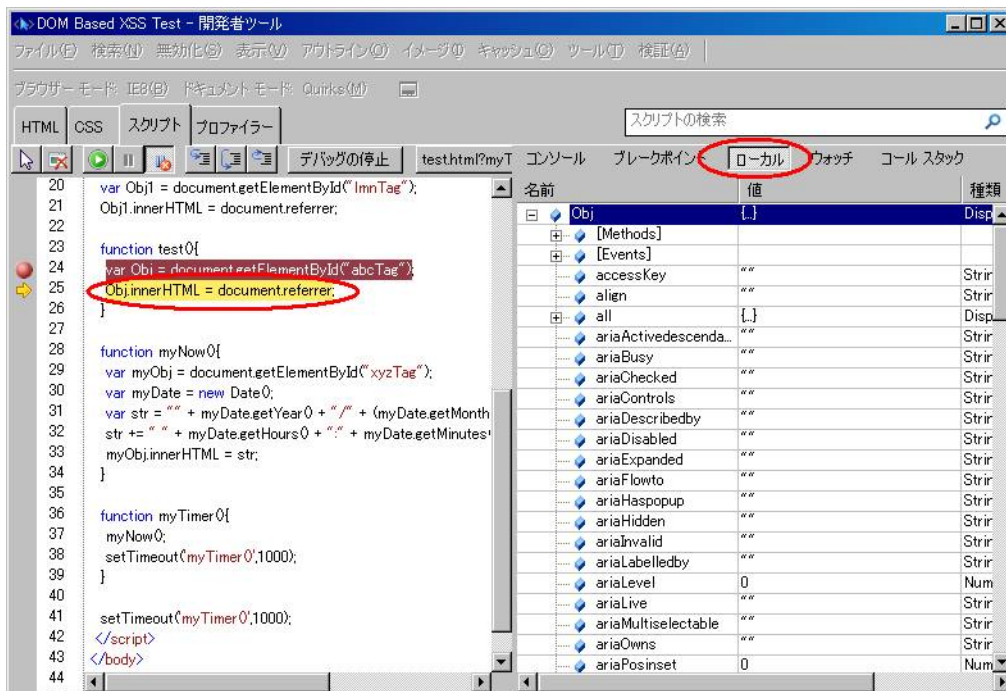


図 3.3-4 : 図 3.3-3後、さらに進める(ステップ実行は「F11」)すると、

注目の「innerHTML」と外部から書き換え可能なプロパティ「document.referrer」が出現した

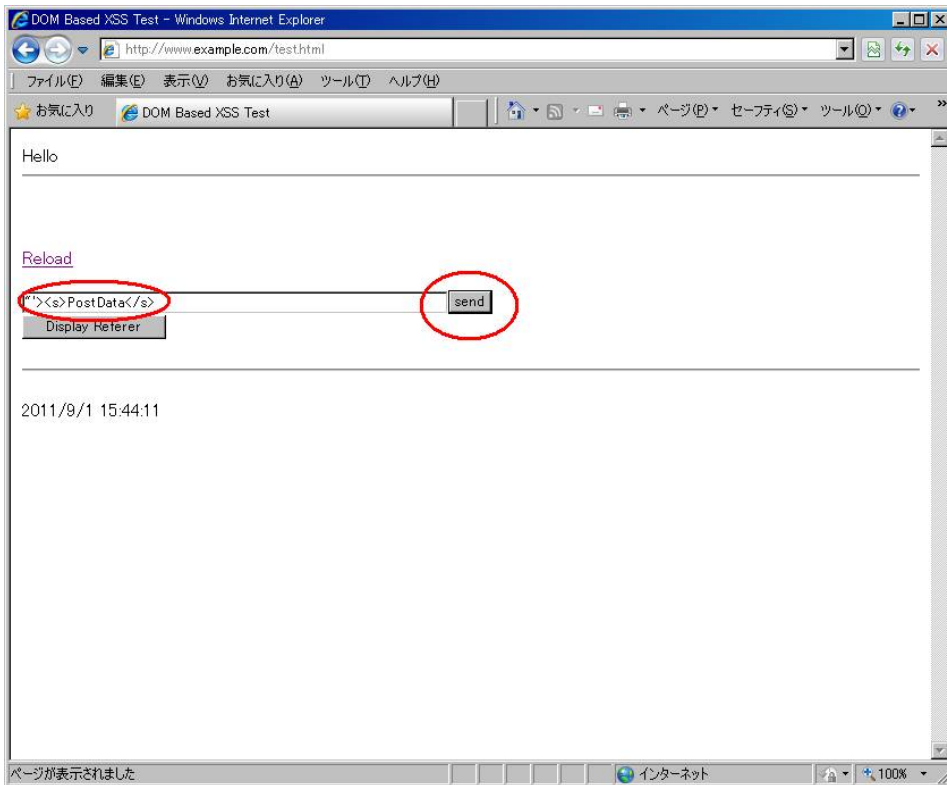


図 3.3-5 : 最初はテキストボックスに XSS 試験用文字列を与え、
「send」ボタンによって画面書き換えを実施する

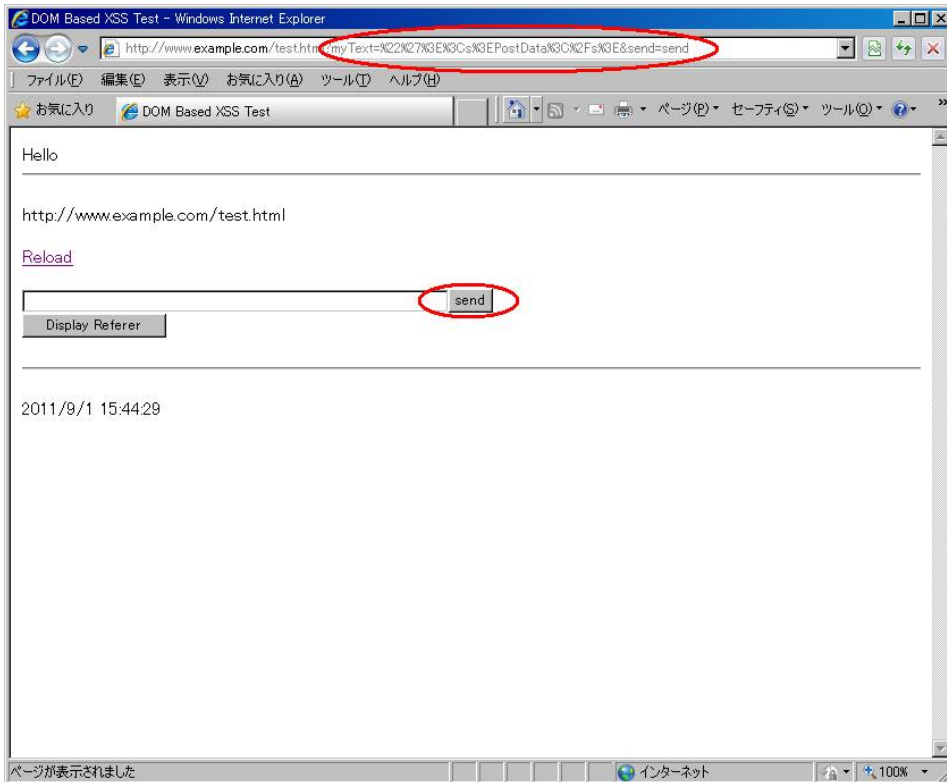


図 3.3-6 : 図 3.3-5の結果。再度の「send」ボタンによって、
図 3.3-5で入力した情報をRefererに記憶させる

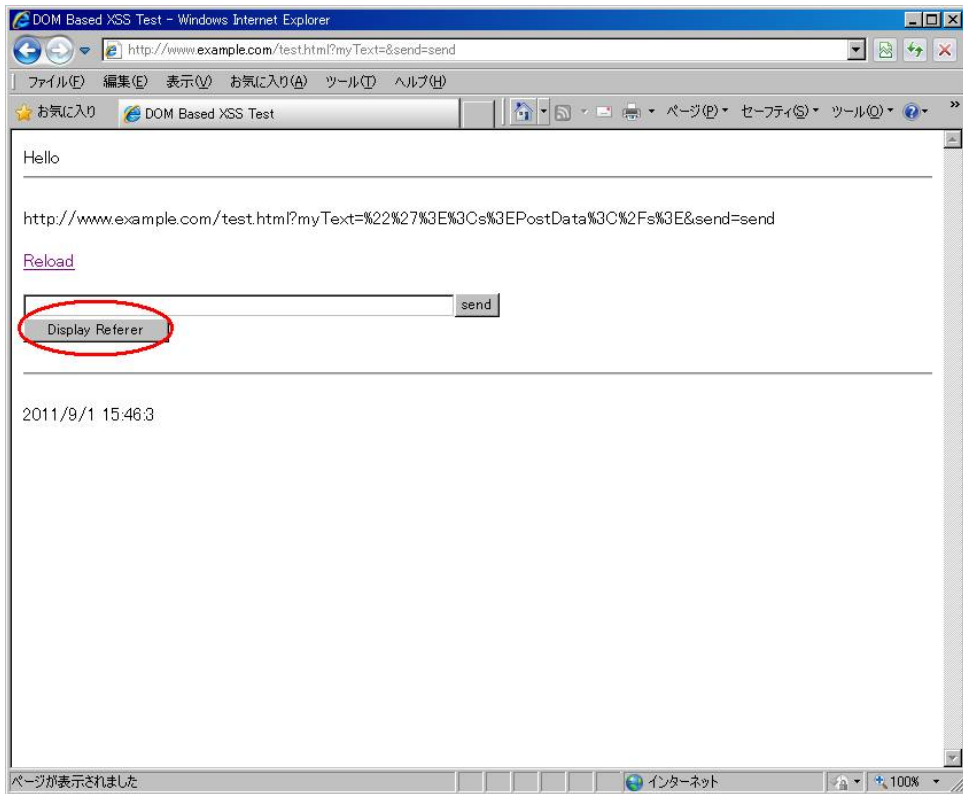


図 3.3-7 : 図 3.3-6の結果。次は「Display Referer」ボタンで、
Refererを表示してXSS発現のはずだが・・・

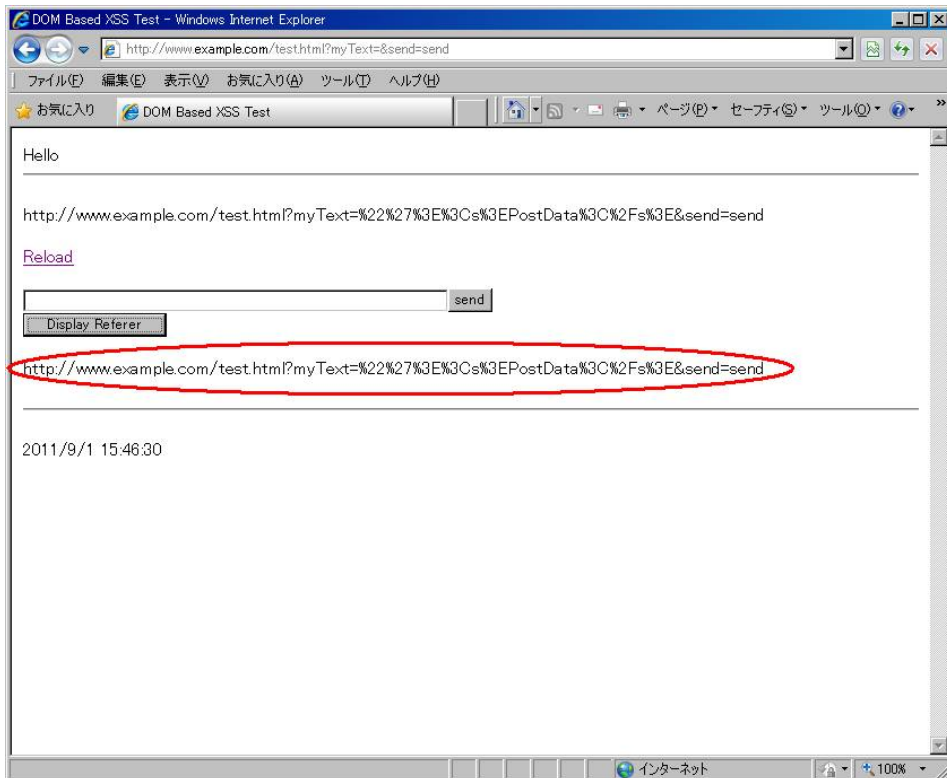


図 3.3-8 : 図 3.3-7の結果。XSSが発現していないのはクエリ文字列が
URLエンコードされていることが原因だ

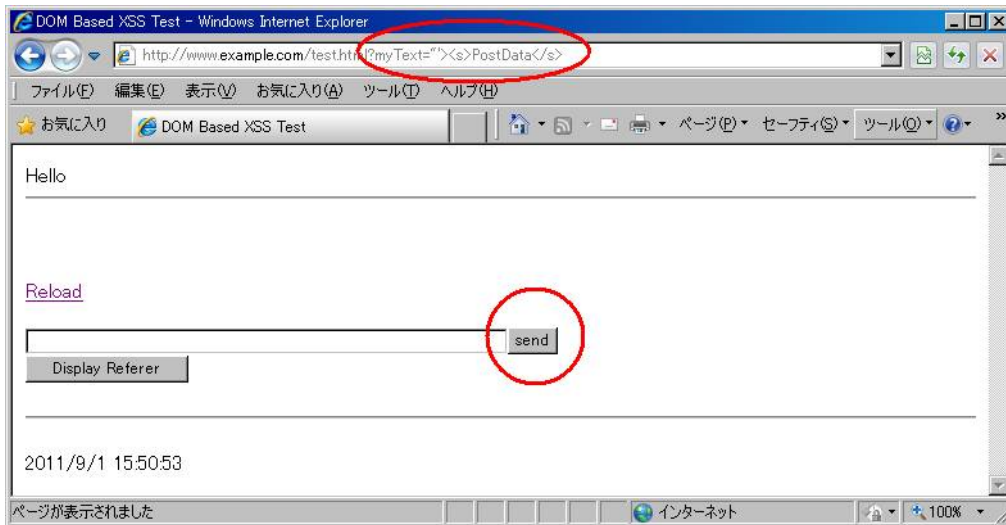


図 3.3-9 : 次は URL エンコードされないように、アドレス欄に直接記述し、画面呼び出しを行う。その上で「send」ボタンで画面書き換えを行い、現在の URL を Web ブラウザが Referer として認識させる

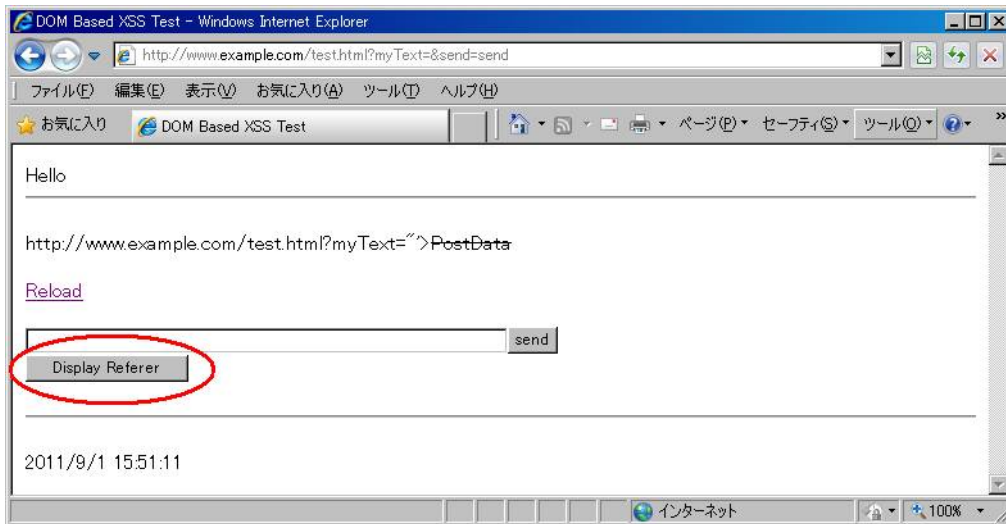


図 3.3-10 : 図 3.3-9の結果。いよいよ「Display Referer」をクリックする

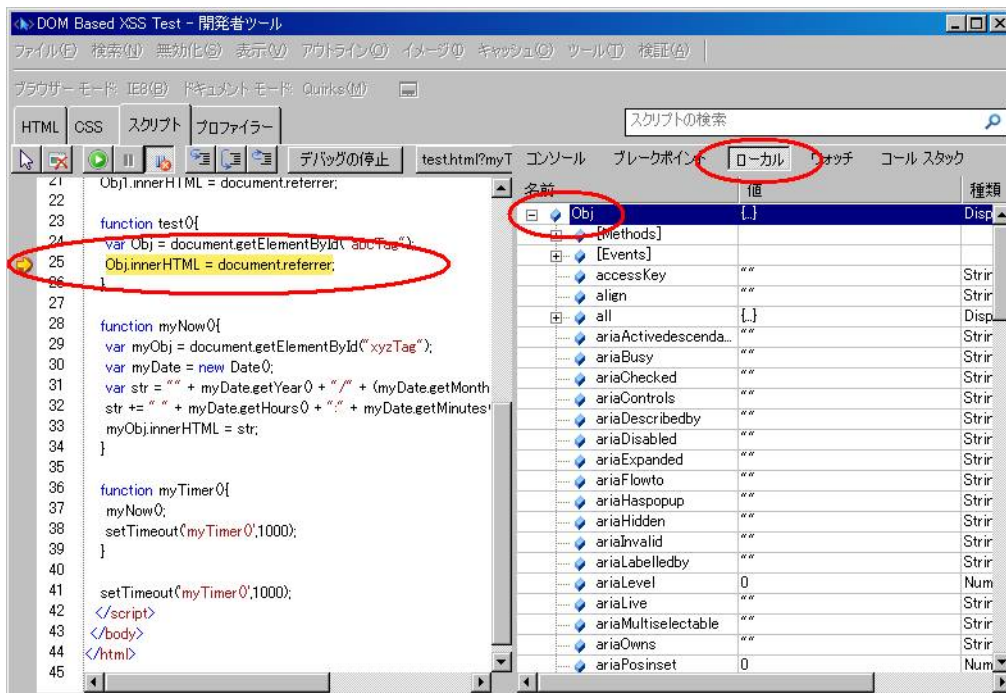


図 3.3-11 : 図 3.3-10の結果。ブレークポイントで処理が停止した

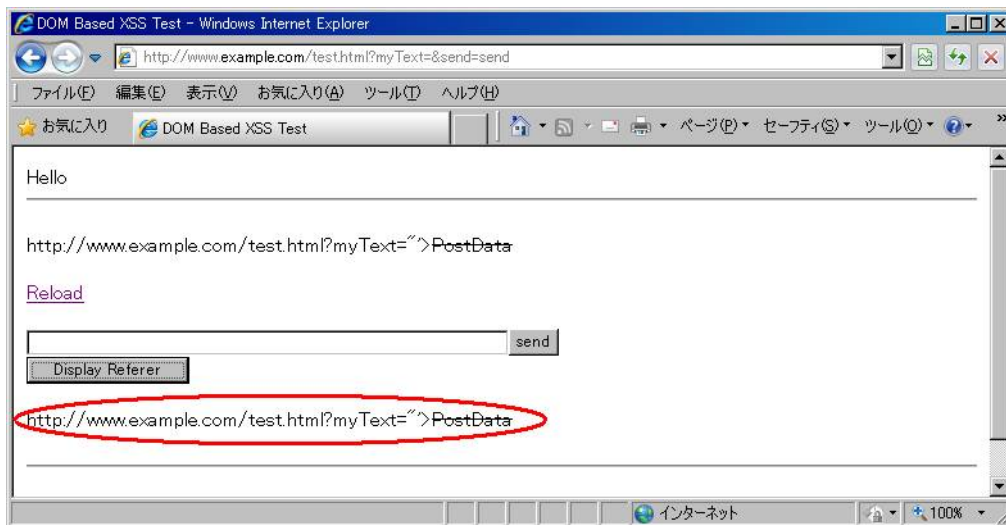


図 3.3-12 : 図 3.3-10~図 3.3-11の結果。XSSが発現した

4. まとめ

本文書の例題は、非常にシンプルな DOM Based XSS であったが、より实际的、より現実的で複雑なものであっても、同様の方法で確認することは可能であろう。

実際のところ、多くの Web アプリケーションは、未だに「WEB2.0」のレベルではなく、AJAX も使われていない Web アプリケーションが多いと思われるが、いざ AJAX をふんだんに駆使した Web アプリケーションに対しても、既存の Web アプリケーション診断の方法に加えて、このようなデバッグ機能を駆使することで、より精度の高いセキュリティ診断を行うことができる。

5. 検証作業

NTT コミュニケーションズ株式会社
ソリューションサービス部 第四エンジニアリング部門 セキュリティオペレーション担当
佐名木 智貴

6. 参考

1. Windows Internet Explorer 8 開発者ツールを使用してサイトを修正する
<http://msdn.microsoft.com/ja-jp/library/cc817576.aspx>
2. Internet Explorer 8 の開発者ツールの概要
[http://msdn.microsoft.com/ja-jp/library/cc848894\(v=vs.85\).aspx](http://msdn.microsoft.com/ja-jp/library/cc848894(v=vs.85).aspx)
3. 使ってみよう! [F12] IE9 開発者ツール - HTML と JavaScript のデバッグ編 - monoe's blog
<http://blogs.msdn.com/b/osamum/archive/2011/08/04/f12-ie9-html-javascript.aspx>
4. 使ってみよう! [F12] IE9 開発者ツール - さまざまな情報の表示 - monoe's blog
<http://blogs.msdn.com/b/osamum/archive/2011/08/09/f12-ie9.aspx>
5. F12 開発者ツールで Web ページをデバッグする方法 Web ページのデバッグ Web ページのトラブルシューティング F12 開発者ツール開発者ツール (Windows)
[http://msdn.microsoft.com/ja-jp/library/gg589507\(v=vs.85\).aspx](http://msdn.microsoft.com/ja-jp/library/gg589507(v=vs.85).aspx)

7. 履歴

- 2011 年 09 月 05 日 : ver1.0 最初の公開

8. 最新版の公開URL

http://www.ntt.com/icto/security/data/soc.html#security_report

9. 本レポートに関する問合せ先

NTT コミュニケーションズ株式会社
ソリューションサービス部 第四エンジニアリング部門 セキュリティオペレーション担当

e-mail: scan@ntt.com

以 上