

Cross-Site-Scripting Attack and Protection Mechanisms

Author: EUROSEC GmbH Chiffriertechnik & Sicherheit
Tel: 06173 / 60850, www.eurosec.com

© EUROSEC GmbH Chiffriertechnik & Sicherheit, 2005

Presentation Overview

- What is Cross-Site Scripting?
- What is the impact of Cross-Site Scripting?
- What are Cross-Site Scripting Techniques?
 - Basic anatomy of Attacks
- How can we protect applications against it?
 - Basic protection mechanisms

What is Cross-Site Scripting?

The three conditions for Cross-Site Scripting:

1. A Web application accepts user input
 - Well, which Web application doesn't?
2. The input is used to create dynamic content
 - Again, which Web application doesn't?
3. The input is insufficiently validated
 - Most Web applications don't validate sufficiently!

What is Cross-Site Scripting?

- Cross-Site Scripting aka „XSS“ or „CSS“
- The players:
 - An Attacker
 - Anonymous Internet User
 - Malicious Internal User
 - A company's Web server (i.e. Web application)
 - External (e.g.: Shop, Information, CRM, Supplier)
 - Internal (e.g.: Employees Self Service Portal)
 - A Client
 - Any type of customer
 - Anonymous user accessing the Web-Server

What is Cross-Site Scripting?

- Scripting: Web Browsers can execute commands
 - Embedded in HTML page
 - Supports different languages (JavaScript, VBScript, ActiveX, etc.)
 - Most prominent: JavaScript
- “Cross-Site” means: Foreign script sent via server to client
 - Attacker „makes“ Web-Server deliver malicious script code
 - Malicious script is executed in Client’s Web Browser
- Attack:
 - Steal Access Credentials, Denial-of-Service, Modify Web pages
 - Execute any command at the client machine

XSS-Attack: General Overview

Attacker



Post Forum Message:
Subject: GET Money for FREE !!!
Body:
<script> attack code </script>

Web Server



Did you know this?
GET Money for FREE !!!
<script> attack code </script>
Re: Error message on startup
I found a solution!
Can anybody help?
Error message on startup
.....

Get /forum.jsp?fid=122&mid=2241

This is only **one**
example out of many
attack scenarios!

GET Money for FREE !!!
<script> attack code </script>

Client



!!! attack code !!!

XSS – A New Threat?



CERT® Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests

Original release date: February 2, 2000
Last revised: February 3, 2000

A web site may inadvertently include malicious HTML tags or script in a dynamically generated page based on unvalidated input from untrustworthy sources. This can be a problem when a web server does not adequately ensure that generated pages are properly encoded to prevent unintended execution of scripts, and when input is not validated to prevent malicious HTML from being presented to the user.

- XSS is an old problem
 - First public attention 5 years ago
 - Now regularly listed on BUGTRAQ
- Nevertheless:
 - Many Web applications are affected

What's the source of the problem?

- Insufficient input/output checking!
- Problem as old as programming languages

Who is affected by XSS?

- XSS attack's first target is the Client
 - Client trusts server (Does not expect attack)
 - Browser executes malicious script
- But second target = Company running the Server
 - Loss of public image (Blame)
 - Loss of customer trust
 - Loss of money

Impact of XSS-Attacks

Access to authentication credentials for Web application

- Cookies, Username and Password
 - XSS is not a harmless flaw !
- Normal users
 - Access to personal data (Credit card, Bank Account)
 - Access to business data (Bid details, construction details)
 - Misuse account (order expensive goods)
- High privileged users
 - Control over Web application
 - Control/Access: Web server machine
 - Control/Access: Backend / Database systems

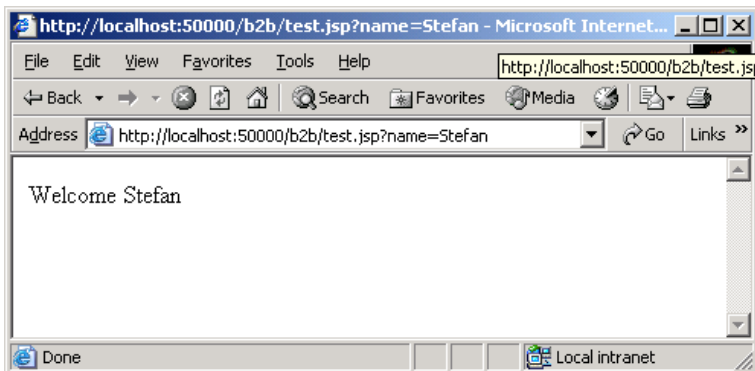
Impact of XSS-Attacks

- Denial-of-Service
 - Crash Users` Browser, Pop-Up-Flodding, Redirection
- Access to Users` machine
 - Use ActiveX objects to control machine
 - Upload local data to attacker`s machine
- Spoil public image of company
 - Load main frame content from „other“ locations
 - Redirect to dialer download

Simple XSS Attack

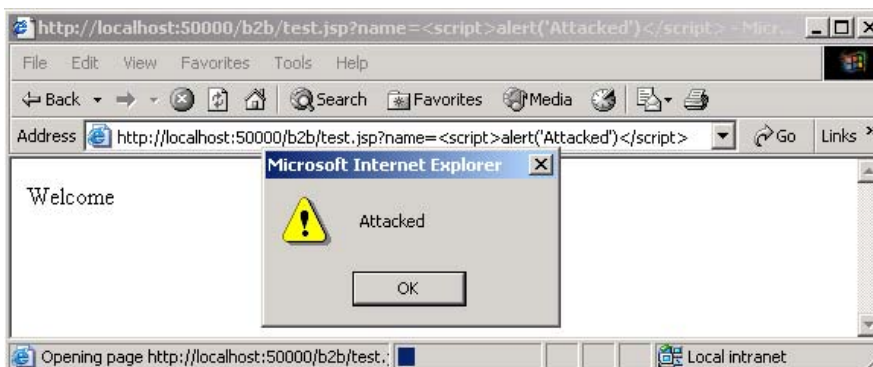
```
test.jsp - Notepad
File Edit Format Help
<% out.println("welcome " + request.getParameter("name")); %>
```

<http://myserver.com/test.jsp?name=Stefan>



```
<HTML>
<Body>
Welcome Stefan
</Body>
</HTML>
```

[http://myserver.com/welcome.jsp?name=<script>alert\("Attacked"\)</script>](http://myserver.com/welcome.jsp?name=<script>alert('Attacked')</script>)



```
<HTML>
<Body>
Welcome <script>alert("Attacked")</script>
</Body>
</HTML>
```

Where script is executed ...

```

<a href="javas&#99;ript&#35;[code]">
  <div onmouseover="[code]">
    
  [IE] 
  [IE] <input type="image" dynsrc="javascript:[code]">
  [IE] <bgound src="javascript:[code]">
    &<script>[code]</script>
  [N4] &{[code]};
  [N4] <img src=&{[code]};>
  <link rel="stylesheet" href="javascript:[code]">
  [IE] <iframe src="vbscript:[code]">
  [N4] 
  [N4]
  <a href="about:<s&#99;ript>[code]</script>">
  <meta http-equiv="refresh"
    content="0;url=javascript:[code]">
  <body onload="[code]">
  <div style="background-image:
    url(javascript:[code]);">

```

```

[IE] <div style="behaviour: url([link to code]);">
[Mozilla] <div style="binding: url([link to code]);">
[IE] <div style="width: expression([code]);">
[N4] <style type="text/javascript">[code]</style>
[IE] <object classid="clsid:..."
  codebase="javascript:[code]">
  <style><!--</style><script>[code]!--</script>
  <![CDATA[<!--]]><script>[code]!--</script>
  <!-- --- --><script>[code]</script><!-- --- -->
  <<script>[code]</script>
  
   onmouseover="[code]">
  <xml src="javascript:[code]">
  <xml d="X"><a><b>&lt;script>[code]&lt;/script>;
    </b></a> </xml>
  <div datafld="b" dataformatas="html"
    datasrc="#X"></div>
[UTF-8; IE, Opera]
[\xC0][\xBC]script>[code][\xC0][\xBC]/script>

```

Source: <http://www.securityfocus.com/archive/1/272037/2002-05-09/2002-05-15/0>

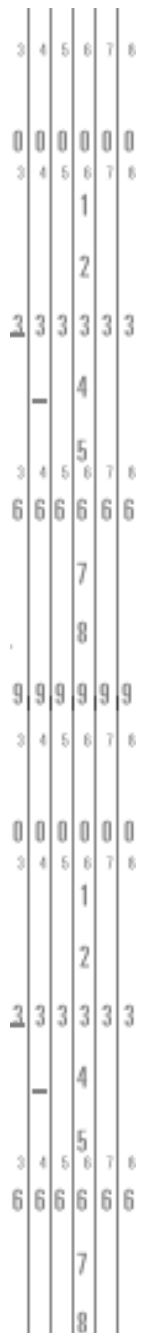
Preventing XSS means Preventing...

- Subversion of separation of clients
 - Attacker can access affected clients' data
 - Industrial espionage
- Identity theft
 - Attacker can impersonate affected client
- Illegal access
 - Attacker can act as administrator
 - Attacker can modify security settings

XSS Solution

Input Validation

But what is to consider "Input"?



Typical HTTP Request

POST /thepage.jsp?var1=page1.html HTTP/1.1

Accept: */*

Referer: http://www.myweb.com/index.html

Accept-Language: en-us,de;q=0.5

Accept-Encoding: gzip, deflate

Content-Type: application/x-www-url-encoded

Content-Lenght: 59

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)

Host: www.myweb.com

Connection: Keep-Alive

uid=fred&password=secret&pagestyle=default.css&action=login

This all is input:

Requested Resource

GET and POST Parameters

Referer and User Agent

HTTP Method

What to Consider Input?

- Not only field values with user supplied input
- Should be treated as Input:
 - All field values: Even hidden fields
 - All HTTP header fields: Referer
 - And even the HTTP method descriptor

What if you request the following from your Web Server?

```
<script>alert("Hello")</script> / HTTP/1.0
```

- Input is any piece of data sent from the client!
 - That is the whole client request

How to perform Input Validation

- Check if the input is what you expect
 - Do not try to check for "bad input"
- Black list testing is no solution
 - Black lists are never complete!
- White list testing is better
 - Only what you expect will pass
 - (correct) Regular expressions

HTML Encoding may help ...

- HTML encoding of all input when put into output pages
- There are fields where this is not possible
 - When constructing URLs from input (e.g. redirections)
 - Meta refresh, HREF, SRC,
- There are fields where this is not sufficient
 - When generating Javascript from input
 - Or when used in script enabled HTML Tag attributes

```
Htmleencode("javascript:alert(`Hello`)") = javascript:alert(`Hello`)
```

Cookie Options mitigate the impact

Complicate attacks on Cookies

- "httpOnly" Cookies
 - Prevent disclosure of cookie via DOM access
 - IE only currently
 - use with care, compatibility problems may occur
 - But: cookies are sent in each HTTP requests
 - E.G. Trace-Method can be used to disclose cookie
 - Passwords still may be stolen via XSS
- "secure" Cookies
 - Cookies are only sent over SSL

Web Application Firewalls

- Web Application Firewalls
 - Check for malicious input values
 - Check for modification of read-only parameters
 - Block requests or filter out parameters
- Can help to protect „old“ applications
 - No source code available
 - No know-how available
 - No time available
- No general solution
 - Usefulness depends on application
 - Not all applications can be protected

This is NO Solution!

- SSL:
 - Attack is not based on communication security flaws
 - Attack is based on application security problems
- Client side input checking:
 - Can be subverted easily
 - Direct URL access

```
<form method="GET" action="/file.jsp">
```

```
<input type="text" name="fname" maxlength="10">
```

```
GET /file.jsp?fname=123456789012345
```

By the way

- Web Services are affected by XSS too
 - Become more and more standard
 - Access protocol is often HTTP
 - Data transfer using XML
- Attack: Submitting SOAP-Response-Values as Request-Values
- Often HTML rendering engines are used for display
 - Force “traditional” XSS attack code in output

Summary

- Cross-Site Scripting is extremely dangerous
 - Identity theft, Impersonation
- Cause: Missing or in-sufficient input validation
- XSS-Prevention Best Practices
 - Implement XSS-Prevention in application
 - Do not assume input values are benign
 - Do not trust client side validation
 - Check and validate all input before processing
 - Do not echo any input value without validation
 - Use one conceptual solution in all applications

Appendix

3	4	5	6	7	8
0	0	0	0	0	0
3	4	5	6	7	8
		1			
		2			
3	3	3	3	3	3
-		4			
3	4	5	6	7	8
6	6	6	6	6	6
		7			
		8			
9	9	9	9	9	9
3	4	5	6	7	8
0	0	0	0	0	0
3	4	5	6	7	8
		1			
		2			
3	3	3	3	3	3
-		4			
3	4	5	6	7	8
6	6	6	6	6	6
		7			
		8			

Why this presentation from us?

- Our professional background:

- Several project years' in research projects on secure software development; with partners like SAP, Deutsche Bank, Commerzbank, Universities, etc.
- Large amount of vulnerability checks for software companies, including reviews and coaching for development staff
- Security requirements- and design specifications for large development projects
- Writing of company guidelines and checklists for secure development, mostly in banking and finance sector
- Reverse engineering and reviews of security mechanisms and crypto algorithms
- Implementation of security functions for customers

Final Remark

- The present work has been conducted within the so-called **secologic** research project (term: 2005+2006)
- For more details see www.secologic.org
- We are grateful to the German Ministry Bundesministerium für Wirtschaft for supporting this project
- We appreciate all suggestions and feedback with respect to our presentation slides and white papers

Copyright Remark

- This document has been prepared by EUROSEC and serves the purpose of conducting courses and seminars about secure software development (focus on web applications)
- We published these slides to support further activities in the development of better software applications
- These slides can be used for your own purposes/employees within your company, as long as you include an information about authorship by EUROSEC
- Commercial use by companies specialized in seminars and/or consulting, is not allowed without a separate agreement; please contact us: kontakt@eurosec.com