

## Insecure Authentication control in J2EE implemented using `sendRedirect()`

### INTRODUCTION

There are a few myths or misconceptions about certain programming entities in J2EE, which if left unexplored, can inadvertently lead to major programming flaws in the application.

Let's take the `response.sendRedirect()` method.

Let's first understand what does this method do?

```
response.sendRedirect("home.html");
```

The server sends a redirection response to the user who then gets redirected to the desired web component, passed as an argument to the method –in this case `home.html`, which processes the request further.

A misconception here is that the execution within a Servlet/JSP stops, after a call to `response.sendRedirect(another page/servlet)`.

*What happens if the execution flow continues even after a `response.sendRedirect()`? Is there a security flaw in it?*

To understand that we would require a little demonstration; take a look at the code snippet below, it checks for authenticated session using an `"if"` condition and If the condition fails the `response.sendRedirect()` redirects the user to an error page.

```
1 <%@page import="java.sql.*" pageEncoding="UTF-8"%>
2 <%@page import="DataAccess.DataStoreAccess"%>
3 <%@page import="DataAccess.QueryStore"%>
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1"%>
7 <title>Add Branch Details</title>
8 </head>
9 <body>
10 <font face="Tahoma"> <% String username = (String)session.getAttribute("user");
11 <div style="border: 2px solid red; border-radius: 15px; padding: 5px; width: fit-content; margin: 10px 0 10px 20px;">
12 <code>If(username ==null){
13 <code>response.sendRedirect("../ErrorPage.jsp?message=User Not Authenticated");
14 </code>
15 </div>
16 }
17
18
19 String branchname = request.getParameter("branchname");
20 String address = request.getParameter("address");
21 String phone = request.getParameter("phone");
22 String fax = request.getParameter("fax");
23 String city = request.getParameter("city");
24
25 Connection con = null;
26 PreparedStatement stmt = null;
27
28 con = DataStoreAccess.getConnection(1);
29 String query = QueryStore.addBranch;
30 stmt = con.prepareStatement(query);
```

Note that there is code present after the If condition, which continues to take request parameters and adds a branch into the system. Hence, in this case if the execution flow in the above JSP page continues even after `response.sendRedirect()` then the branch would get added even for invalid requests.

### *Let's check how does this flaw manifest?*

This flaw manifests as a result of the misconception that `sendRedirect` method terminates the execution flow in the calling JSP/Servlet class. In the above scenario the JSP page uses a session based authentication check, and redirects the user to an error page, if an authenticated username is not found in the session variable. However, in reality as the execution of the JSP page does not stop after the "sendRedirect" call, the request would get processed even for unauthenticated requests.

Thus, an authentication or any other security control built in this way can be easily bypassed and results into a big security flaw in the application.

### *How do we exploit this flaw?*

In our scenario an application allows only admin users to create new branches. If we try accessing the ADD branch page without authentication we will be redirected to an error page, due to the "sendRedirect" call present in the authentication check of the JSP page.

Let's see how we can bypass this check with detailed steps

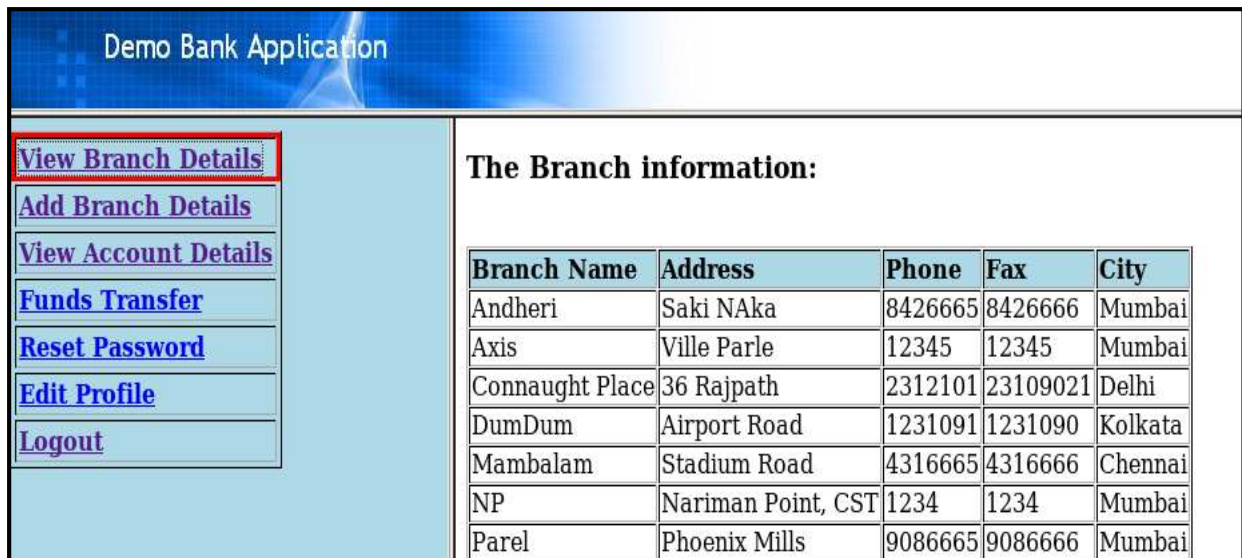
## **EXPLOIT STEPS**

Let us understand how the "add a branch" feature works.

Login to the application as an admin user



View the current set of branches available, with view branches option



The screenshot shows the 'Demo Bank Application' interface. On the left, a navigation menu contains links: View Branch Details (highlighted with a red border), Add Branch Details, View Account Details, Funds Transfer, Reset Password, Edit Profile, and Logout. The main content area is titled 'The Branch information:' and displays a table of branch data.

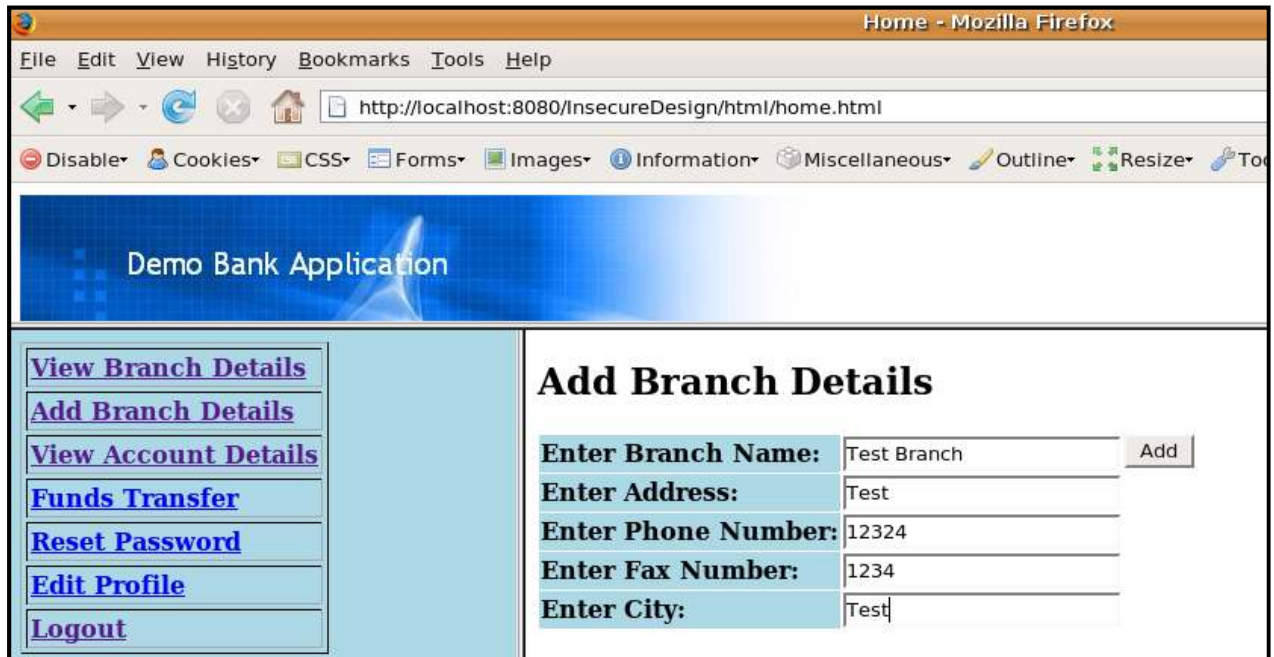
Branch Name	Address	Phone	Fax	City
Andheri	Saki Naka	8426665	8426666	Mumbai
Axis	Ville Parle	12345	12345	Mumbai
Connaught Place	36 Rajpath	2312101	23109021	Delhi
DumDum	Airport Road	1231091	1231090	Kolkata
Mambalam	Stadium Road	4316665	4316666	Chennai
NP	Nariman Point, CST	1234	1234	Mumbai
Parel	Phoenix Mills	9086665	9086666	Mumbai

Next go to the ADD branches section and proceed to add a new Test branch.

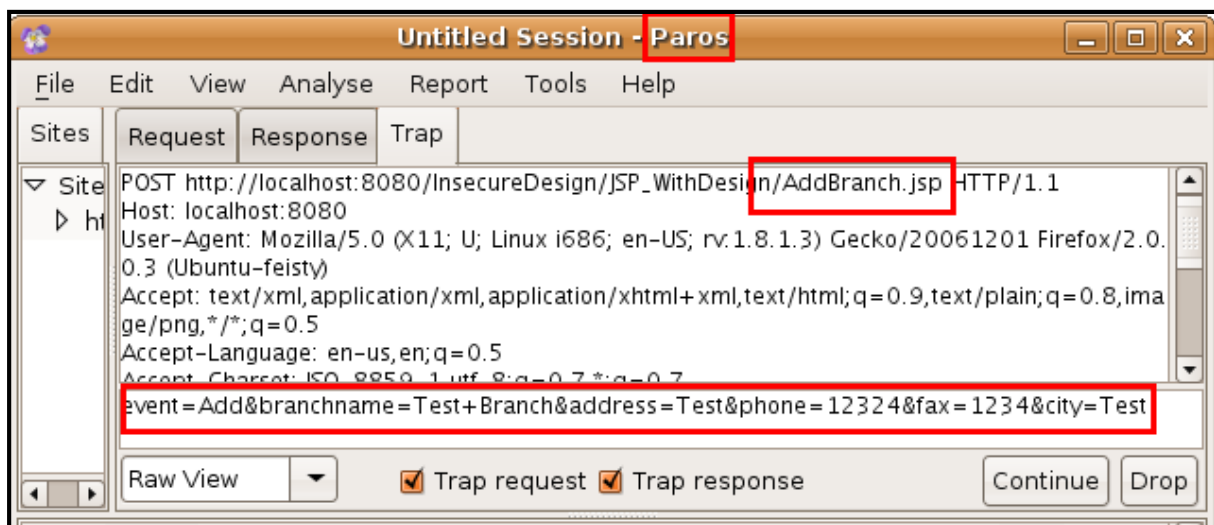


The screenshot shows the 'Demo Bank Application' interface in a Mozilla Firefox browser window. The browser's address bar shows the URL 'http://localhost:8080/InsecureDesign/html/home.html'. The main content area is titled 'Add Branch Details' and contains a form with the following fields: 'Enter Branch Name:', 'Enter Address:', 'Enter Phone Number:', 'Enter Fax Number:', and 'Enter City:'. Each field has a corresponding input box. An 'Add' button is located to the right of the 'Enter Branch Name' input box. The left navigation menu is visible, with 'Add Branch Details' highlighted with a red border.

Fill in the form



Proceed to click ADD. You will notice in a Web Proxy editing tool like Paros that the request is sent to *ADDbranch.jsp* page, along with a set of branch parameters.



Notice that the Test branch was created.

Demo Bank Application

[View Branch Details](#)  
[Add Branch Details](#)  
[View Account Details](#)  
[Funds Transfer](#)  
[Reset Password](#)  
[Edit Profile](#)  
[Logout](#)

**The Branch information:**

Branch Name	Address	Phone	Fax	City
Andheri	Saki Naka	8426665	8426666	Mumbai
Axis	Ville Parle	12345	12345	Mumbai
Connaught Place	36 Rajpath	2312101	23109021	Delhi
DumDum	Airport Road	1231091	1231090	Kolkata
Mambalam	Stadium Road	4316665	4316666	Chennai
NP	Nariman Point, CST	1234	1234	Mumbai
Parel	Phoenix Mills	9086665	9086666	Mumbai
Test Branch	Test	12324	1234	Test
TNagar	Valluvar Kotam	6316665	6316666	Chennai

Now logout of the application.

Insert title here - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:8080/InsecureDesign/Login.jsp

Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tool

## Login to the Bank

User ID:

Password:

Log In

Now, attempt to send the same request to *ADDbranch.jsp* page with all the parameters to for adding a branch as query string, directly without authentication.

File Edit View History Bookmarks Tools Help

http://localhost:8080/InsecureDesign/AddBranch.jsp?branchname=HACKED&address=HACKED&phone=1234&...

Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tool

## Login to the Bank

User ID:

Password:

Log In

Access Add Branch Page without login.



We will be immediately redirected to the error page, since the authentication check would fail, and sendRedirect() method would be invoked.

```
5<%@page import="java.sql.PreparedStatement"%>
6<%@page import="DataAccess.DataStoreAccess"%>
7<%@page import="DataAccess.QueryStore"%>
8<html>
9<head>
10<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
11<title>Add Branch Details</title>
12</head>
13<body>
14<font face="tahoma"> <% String username = (String)session.getAttribute("user");
15if(username ==null){
16response.sendRedirect("../ErrorPage.jsp?message=User Not Authenticated");
17}
```



Now login the application as an admin, to check if the malicious request actually added a branch.

You will notice in the screenshot below, the request got added.

**Demo Bank Application**

[View Branch Details](#)  
[View Account Details](#)  
[Funds Transfer](#)  
[Reset Password](#)  
[Edit Profile](#)  
[Logout](#)

**The Branch information:**

Branch Name	Address	Phone	Fax	City
Andheri	Saki Naka	26666	26666	Mumbai
Axis	Ville Parle	12345	12345	Mumbai
Connaught Place	36 Rajpath	2312101	23109021	Delhi
DumDum	Airport Road	1231091	1231090	Kolkata
HACKED	HACKED	1234	1234	HACKED
Mambalam	Stadium Road	4316665	4316666	Chennai
NP	Nariman Point, CST	1234	1234	Mumbai

Since the control flow in the JSP page is not terminated even after a redirect, the rest of JSP page code, which is meant to process the incoming request and add the branch details into the system, gets executed, as shown below. And a new branch is added to the database.

```
19 String branchname = request.getParameter("branchname");
20 String address = request.getParameter("address");
21 String phone = request.getParameter("phone");
22 String fax = request.getParameter("fax");
23 String city = request.getParameter("city");
24
25 Connection con = null;
26 PreparedStatement stmt = null;
27
28 con = DataStoreAccess.getConnection(1);
29 String query = QueryStore.addBranch;
30 stmt = con.prepareStatement(query);
```

Thus, we were able to access the internal features of the application without authentication. Due to insecure authentication check present in the JSP page built only using – “*sendRedirect*” method.

**Note:** The fact that execution of a servlet or JSP continues even after `sendRedirect()` method, also applies to *Forward* method of the *RequestDispatcher* Class.

However, `<jsp:forward>` tag is an exception, it was observed that the execution flow stops after the use of `<jsp:forward>` tag.

### **RECOMMENDED FIX**

Since this flaw resulted from the assumption made by developers that control flow execution terminates after a *sendRedirect* call, the recommendation would be to ensure that the execution flow is terminated.

The fix is to use a *return;* statement after the `sendRedirect()` call within the if condition check for authenticated session. As shown in a dummy code below:

```
if (<request is invalid>) {
response.sendRedirect("Errorpage");
return;
}
```

### **REFERENCES**

Control flow myths busted in Java - <http://palizine.plynt.com/issues/2011Dec/java-myths/>

### **PLATFORM/VERSIONS AFFECTED**

All applications developed using JSP/Servlets

## About Author

### Ashish Rao

Ashish Rao is a Security Consultant at Paladion Networks Pvt. Ltd. He has a good application development background and is an expert in performing secure code reviews for J2EE and ASP.Net applications. He has reviewed many complex multi-tiered web and standalone applications of different frameworks and programming languages. He has authored articles and blogs about secure coding and security best practices. He has also worked closely with development teams across the globe and has helped them to secure applications at the design and architecture level. He also has the working knowledge of many static code analysis tools and has contributed immensely to enhance Paladion's automated review capabilities by writing various easy-to-use code review scripts. Other than secure code reviews, he possesses extensive knowledge regarding Penetration Tests and Vulnerability Assessment projects, and has conducted various internal and external trainings for Paladion. He has recently conducted workshops in OWASP India 2012 and Clubhack 2012 conferences. **Personal Blog:** <http://artechtalks.blogspot.in>