Java Webstart Configuration

A simple configuration that supports the Spk MDA security model is described.

Table of Contents

Introduction

MDA security depends on the following sequence of actions:

1. Using a web browser, the user visits the home page of an Spk site. The URL for the home page specifies the ordinary, insecure, http protocol.
2. The user selects a link to a login page. The URL for this link specifies the secure https protocol.
3. The user's browser verifies the authenticity of the remote server. If the server does not present a certificate cryptographically signed by a trusted third party, the browser warns the user. This way, the user knows that he is logging in to the service that he intends to connect to, rather than an impostor.
4. Assuming that the server's credentials are accepted, the browser presents the login page to the user. The user fills in name and password and sends the information back to the server, using the encrypted https protocol.
5. The server checks the username and password against values in its user database. If these are correct, the user is linked to his personal, mySpk, page.

   Note: at this point, the server has proved its authenticity to the user by means of a certificate and the user has proved his authenticity by means of username and password. All communications required for this mutual authentication were encrypted.

6. On his mySpk page, the user is presented with a link to start an MDA. The URL references a `.jnlp` file on the server, and specifies the https protocol.

   A `.jnlp` file is a special configuration file for a Java Webstart application. In our case, that application is an MDA and the file specifies three critical pieces of information:

   - A URL to the location on the server of the Java files for this application. The URL specifies that the secure https protocol will be used for the downloading of the application.
   - A unique session number, which will be used to identify subsequent messages from the client as being associated with this session.

- A unique, session-specific 128-bit integer which will be used as a symmetric encryption key for subsequent messages between the MDA and the server.

7. If the server has a more recent version of the MDA than is present on the user's machine, the browser downloads it. It then starts the MDA, providing the session number and encryption key as data.

   Note: the Java code for the MDA, the session number and the encryption key were all obtained by the browser via secure https.

8. From this point onward, the MDA can communicate directly with the server without intervention of the browser. Messages are identified by session number, and are encrypted and decrypted using the 128-bit encryption key, which is known only to the MDA and the server.

This document describes a process for setting up a server to satisfy the security requirements of the MDA, for development and testing purposes.

Note to maintainers of this document: many of the details of this process are taken from the *Red Hat Linux Customization Guide*.

Prerequisites

- RedHat, version 8.0 or later, on the server.
- Apache installed on the server as rpm package httpd-2.0.40-11.5, or later.
- SSL capability added to Apache as rpm package mod_ssl-2.0.40-11.5, or later.
- Java SDK installed on Linux client: j2sdk1.4.2, or later.

Obtaining a Certificate

We will need a certificate for the Apache web server. In a production system, this must be cryptographically signed by a trusted third party known as a Certificate Authority (CA). CAs recognized by all popular browsers include Verisign, Thawte and Geotrust. In a development and test environment, however, it is more practical to use a self-signed certificate, not only because a certificate signed by a CA costs several hundred dollars per year, but because a certificate is issued for a single domain name and cannot be transferred.

Procedure for creating a self-signed procedure:

1. As root, generate a key:

```
cd /etc/httpd/conf
rm ssl.key/server.key
rm ssl.crt/server.crt

/usr/bin/openssl genrsa 1024 > ssl.key/server.key
chmod go-rwx ssl.key/server.key
```

2. Create the certificate:

```
cd /usr/share/ssl/certs
make testcert
```

The program will ask you additional questions. Here are the questions and sample answers:

```
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:Washington
Locality Name (eg, city) [Newbury]:Seattle
Organization Name (eg, company) [My Company Ltd]:University of Washington
Organizational Unit Name (eg, section) []:RFPK
Common Name (eg, your name or your server's hostname) []:jambutty.rfpk.washington.
Email Address []:afw@u.washington.edu
```

Configuring and Running a Secure Web Server

If you installed the httpd and mod_ssl packages described above in the *Prerequisites* section, the Apache server is already configured. The existence of the certificate created in the previous section is sufficient to enable its secure features.

To start the Apache server, use the following command (as root):

```
/etc/rc.d/init.d/httpd start
```

The server will run until you stop it like this (as root):

```
/etc/rc.d/init.d/httpd stop
```

To configure the system initialization process so that it automatically starts the server each time the system boots, issue the following command (as root):

```
chkconfig --level 345 httpd on
```

Browser Configuration

To use webstart, you may need to configure your browser. The following instructions are for Mozilla, and the java configuration described above. Configuration for other browsers and other operating system environments may be somewhat different.

Bring up the *Helper Applications* window by following this menu sequence:

`Edit` => **`Preferences`** => **`Navigator`** => **`Helper Applications`**

If *application/x-java-jnlp-file* does not appear in the *File types* list, press the **New Type** button and fill out the resulting form as follows:

- *Description of type:* Java Network Launching Protocol
- *File extension:* jnlp
- *MIME type:* application/x-java-jnlp-file
- *Application to use:* /usr/java/j2sdk1.4.2/jre/javaws/javaws

Directories and Access Privileges

Web site security is a complex topic, certainly beyond the scope of this document. We do need to set up some directories, however, and will employ a strategy consistent with good security practices.

The basic idea is that each web maintenance function should be performed by people who have been administratively assigned to that function, and should not be accessible to others. To do this we will make use of the user private groups[1] (UPG) scheme that is favored by RedHat.

We will need two new groups, if comparable groups have not already been defined. The groups are

- *webtext*: for those who are authorized to create or modify text on the web site.
- *webjava*: for those who are authorized to create or modify java code on the web site.

To accomplish the following configuration, you will need to be the root user. Create the required groups as follows:

```
groupadd -r webtext
groupadd -r webjava
```

Next add some users to these groups. For example:

```
gpasswd -a tom  webtext
gpasswd -a dick webtext
gpasswd -a tom  webjava
gpasswd -a jane webjava
```

You should be able to observe these changes by examining `/etc/group`

Next, use these new groups to set group ownership and mode bits in several web directories:

```
cd /var/www
mkdir java
chgrp webtext html
chgrp webjava java
chmod 2775 html
chmod 2775 java
```

Demonstration

This section describes a simple demonstration that illustrates that the webstart configuration works. Note that the domain name `jambutty.rfpk.washington.edu` occurs in the files below. Replace this domain name with your own.

1. Copy the `.jar` file for the application into `/var/www/java`
2. `/var/www/html/index.html` is the top-level file for our demonstration website:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "xhtml1-transitional.dtd">
<html>
<head>
<title>Spk Home</title>
</head>
<body>
<h3>Spk Home Page</h3>
<p>
  Go
  <a href="https://jambutty.rfpk.washington.edu/login.html">
    log in
  </a>
</p>
</body>
</html>
```

   Note that the URL to the next page utilizes https rather than http. This shifts the protocol from the insecure http to the secure https.

3. `/var/www/html/login.html` is a place keeper for the page that would get the login name and password from the user.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "xhtml1-
transitional.dtd">
<html>
  <head>
    <title>MySpk Login</title>
  </head>
  <body>
    Go to
    <a href="myspk.html">MySpk</a>
  </body>
</html>
```

   Note that the link to the next file does not require a full URL, because we are not changing protocol.

4. `/var/www/html/myspk.html` represents the user's personalized Spk page:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "xhtml1-
transitional.dtd">
<html>
  <head>
    <title>My Spk</title>
  </head>
  <body>
    <p>
      Welcome to MySpk.
```

```
    </p>
    <p>
      <a href=time.jnlp>Launch the application</a>
    </p>
  </body>
</html>
```

5. `/var/www/html/time.jnlp` is the configuration file that provides webstart with information about a java application called *time*.

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+"
  codebase="https://jambutty.rfpk.washington.edu/java"
>
<information>
  <title>Time Check</title>
  <vendor>2Java Developer Connection</vendor>
  <homepage href="http://java.sun.com/jdc" />
  <description>Demonstration of JNLP</description>
</information>
<offline-allowed/>
<security>
  <j2ee-application-client-permissions/>
</security>
<resources>
  <j2se version="1.2+" />
  <jar href="../java/JNLPTime.jar"/>
</resources>
<application-desc main-class="TheTime" />
</jnlp>
```

   Note the value assigned to the *codebase* variable. It is a URL incorporating https, which will insure that webstart uses https to download the application.

To run the demonstration, just use a browser to access the web server. In the above example, the URL would be `http://jambutty.rfpk.washington.edu`

Notes

1. http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/ref-guide/s1-users-groups-private-groups.html