Web start and code signing

Richard Dallaway <richard@dallaway.com>
May 2002

I had what I thought were two separate problems: how to provide an easy way to install a Java application I'd written; and how to start a web browser from the application regardless of what operating system the application was running on. Both problems were solved by using Java Web Start, but to get that working I also had to dip into the world of code signing with digital certificates. These are my notes on the experience.

Sloppy

Sloppy is an application that can be used to slow down your internet connection. Often a web site is developed on a fast local network. To see how it would look to a 28.8k or 56k modem user, you could install the appropriate modem, get a dial-up account, configure your machine to use the dial-up... all this generally involves much cursing. And if you want to see what conditions are like under a 28.8k modem *and* and 56k modem, repeat the above.

Or you could use Sloppy. Sloppy is a "slow proxy". It pretends to be the web site you're testing, and artificially slows down the flow of web pages and graphics to your browser. It's not perfect, but it's easier than fiddling around with modem cables.

When I first wrote Sloppy, it ran as a server application. You had to have Java installed, get yourself to a shell or command prompt, run the right magic command, edit the right configuration file, and also launch your web browser on just the right address to see the results. From the feedback I received, it seemed that Sloppy needed an easier way to be installed and a graphical user interface.

Web start

Putting a graphical front end on Sloppy wasn't so hard, but I need a way to get Sloppy installed. I also decided that because of the way Sloppy worked I wanted to be able to start the user's web browser. At the same time, I wanted to keep all of this platform-neutral: it should run on Macs, Linux, Sunkit, Windows machines... all without modification.

Web Start has all the key ingredients. You can think of web start as a browser plug-in. When you click on a link to a `.jnlp` file, the plug-in takes over, downloads the application, makes sure you have the right version of the Java runtime (installs it if necessary), performs a bunch of security checks, and then launches it for you.

Writing a Web Start application just means writing a regular Java application and providing a `.jnlp` file. For Sloppy the file looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+" codebase="http://www.dallaway.com/sloppy" href="sloppy.jnlp">
  <information>
    <title>Sloppy</title>
    <vendor>Richard Dallaway</vendor>
    <homepage href="index.html" />
    <description>Sloppy allows you to simulate different dial-up speeds.</description>
    <offline-allowed />
  </information>
  <resources>
    <j2se version="1.3+"/>
    <jar href="sloppy.jar" main="true" />
  </resources>
  <application-desc main-class="com.dallaway.sloppy.Sloppy" />
  <security><all-permissions/> </security>
</jnlp>
```

The file states where to find Sloppy (the codebase), the names of the Jar files to download (`sloppy.jar`), the version of the Java runtime required (1.3 or better), gave some description about the application, and said which `main()` method to run to start Sloppy.

So, with Web Start I had the way to install the application on a user's machine, a way to keep it up to date, and also a way to launch the user's web browser (that's one of the "services" web start gives a developer). The end user gets a relatively hassle-free way of running Java applications and some faith that the security is going to be sound.

Certificates

The only tricky part to all of this was the security. By default, Web Start has a very strict security policy. This is a good thing, because you know a Web Start application won't go and trash your hard drive, start looking through your email, or send messages over the net to some ad company.

The nature of Sloppy is such that it needs to be able to talk to other web sites. That's what it does: it takes your requests, sends it to a web site, and then slows down the response. For this to work under Web Start, I needed to give Sloppy some extra security permissions. In fact, the current version of Web Start means that I have to give Sloppy *unrestricted* access to your computer. This is a bit over-the-top, but it's all or nothing right now. In the future the JNLP specification might allow more fine grained permissions.

To get this unrestricted permission I have to "sign" the application. This proves that it's me that sent the code, and that no-one has tampered with it. If you trust me, you can run the code, safe in the knowledge that I'm not interested in the contents of your hard drive or your email.

To be able to digitally sign the code (the JAR files), I needed a certificate. To get a certificate, I needed to hand over money to a certificate authority so they could run some background checks on me to prove that I am who I say I am. Web Start comes with a list of certificate authorities it knows about. One of them, Verisign, would like US$200 to US$400 from me each year for a certificate. I'm not sure I'm willing to pay that for software I'm giving away.

To see why this is important, take a look at the screen-shot of how Sloppy starts if I use a fake certificate (figure 1).

Figure 1: Sloppy starting with a developer certificate.

Notice the worrying phrases such as "it is highly recommended not to install and run the code"? The point of this exercise was to make Sloppy more available, not to scare everyone off.

I took a good look at the list of certificate authorities known to Web Start and one stuck out: Thawte Freemail. In particular, the word "free". I chased this down and it is indeed a free (no money) way to get a certificate for code signing.

The steps are simple enough:

1. Sign up and follow the steps through up until the point where you're asked to select the X.509 email certificate you want. At that point scroll down and select "paste-in CSR Certificate Enrollment".

2. Run the Java `keytool` utility.

   First, generate a RSA key. You'll be asked all sorts of information. The important thing to remember is that your name must be set according to the common name you're told to use on the Thawte web site. It'll be something like XVV6PaMGHEPJN22. The other important thing to remember is the password you use when you create a key.

   Obviously change your "alias" name and the location of the "keystore" file if you like, but run something like this:

   ```
   keytool -genkey -keyalg RSA -keystore keystore -alias dallaway
   ```

3. Next, export the key to a text file, which in this example is "csr.txt":

   ```
   keytool -certreq -keystore keystore -file csr.txt -alias dallaway
   ```

4. Take the text of "csr.txt" and paste it into the Thawte form and wait. That's your certificate request. Mine took about 20 minutes to process, and I was notified that it was ready by email.

5. When you download your certificate, it's in two formats: Netscape and PKCS7. You want the save the PKCS7 version in a file called "my.cert" (or similar) and then run:

   ```
   keytool -import -file my.cert -alias dallaway -trustcacerts -keystore keystore
   ```

That's it. You have a certificate that's good for a year, after which it can be renewed.

To sign the code with the certificate, you can run an Ant task such as:

```
<signjar keystore="keystore"
         jar="sloppy.jar"
         alias="dallaway"
         storepass="passwordHere"  />
```

When I run the Web Start with my newly signed application I get a much more respectable prompt, as shown in figure 2:

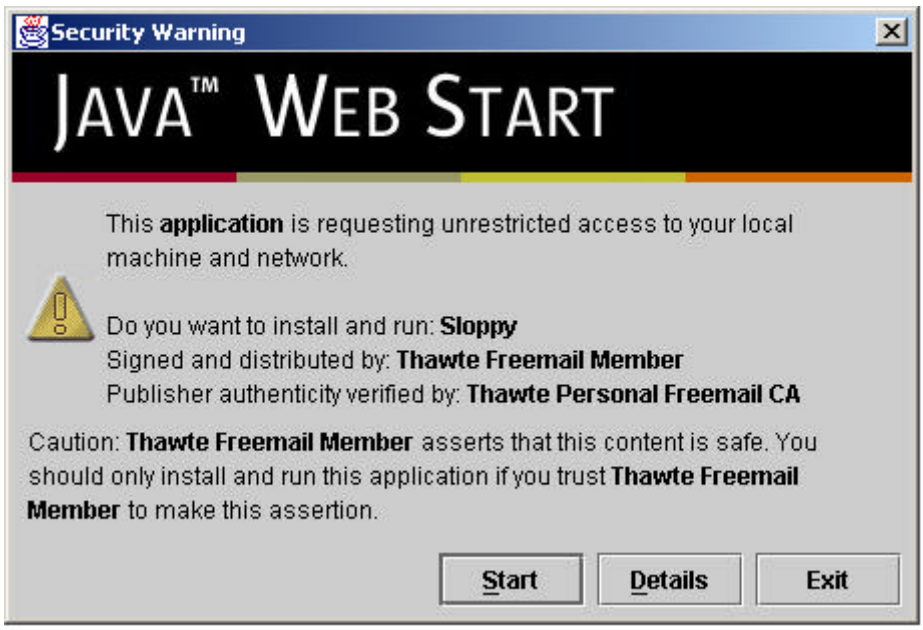

Figure 2: Sloppy starting with a Thawte Freemail certificate.

Web of Trust

It may be possible to replace the phrase "Thawte Freemail Member" in the certificate by joining the Thawte Web of Trust. This involves meeting individuals already OK-ed by Thawte and demonstrating that you are who you say you are by producing various forms of identification. I've not tried this yet.

Summary

Java Web Start is a useful way to deploy Java applications. To gain the trust of end users it's important to sign the code being downloaded, but it is possible to do this at no cost using the Thawte Freemail service.

Resources

1. *Java Web Start*
   http://java.sun.com/products/javawebstart/

2. *Sloppy Homepage.*
   http://www.dallaway.com/sloppy/

3. *Thawte's Digital Certificate Center*
   http://www.thawte.com/getinfo/products/personal/join.html

4. *Thawte's Java Developer Guide*
   http://www.thawte.com/getinfo/products/devel/sunjava.html

5. *Thawte's Web of Trust*
   http://www.thawte.com/getinfo/programs/wot/contents.html