Supplement: Signed Java Applets
For Introduction to Java Programming
By Y. Daniel Liang

Java uses the so-called "sandbox security model" for executing applets to prevent destructive programs from damaging the system on which the browser is running. Applets are not allowed to use resources outside the "sandbox." Specifically, the sandbox restricts the following activities:

- Applets are not allowed to read from, or write to, the file system of the computer. Otherwise, they could damage the files and spread viruses.

- Applets are not allowed to run programs on the browser's computer. Otherwise, they might call destructive local programs and damage the local system on the user's computer.

- Applets are not allowed to establish connections between the user's computer and any other computer, except for the server where the applets are stored. This restriction prevents the applet from connecting the user's computer to another computer without the user's knowledge.

To circumvent these restrictions, you can create signed applets and let the user decide whether to accept or reject the applets.

Let us first create an applet, as shown in Listing 1. The applet displays the current directory where the applet is running.

Listing 1 SignedAppletDemo.java

```
import java.io.*;
import javax.swing.*;

public class SignedAppletDemo extends JApplet {
  public SignedAppletDemo() {
    JLabel label = new JLabel();
    add(label);
    try {
      File file = new File(".");
      label.setText("The current directory is " + file.getAbsolutePath());
    } catch (Exception ex) {
      ex.printStackTrace();
      label.setText("Security exception ");
    }
  }
}
```

This applet cannot run from a Web browser, because it accesses the client's local file system. To enable it run, make it a signed applet.

Here are the steps to create a signed applet.

1. Create a Jar file for the applet:

```
jar cvf SignedApplet.jar SignedAppletDemo.class
```

2. Use the keytool to generate a key pair:

```
keytool -genkey -alias signFiles -keystore mykeystore
     -keypass kpi135 -dname "cn=panda" -storepass ab987c
```

The command generates a key pair that is identified by the alias signFile. The generated key pair is stored in a keystore database called mykeystore, and accessed with the mykeystore password (-storepass ab987c).

3. Sign the JAR file:

```
jarsigner -keystore mykeystore -storepass ab987c -keypass
kpi135 -signedjar SSignedApplet.jar SignedApplet.jar
signFiles
```

4. Create the SignedAppletDemo.html file as follows:

```
<applet code="SignedAppletDemo.class"
     archive="SSignedApplet.jar"
     width=400 height=100>
</applet>
```

5. Now open SignedAppletDemo.html from your Web browser, you will see the Security Warning dialog box displayed as shown in Figure 1. Click Run to run the applet. You will see the applet display the current directory where the applet is executed.

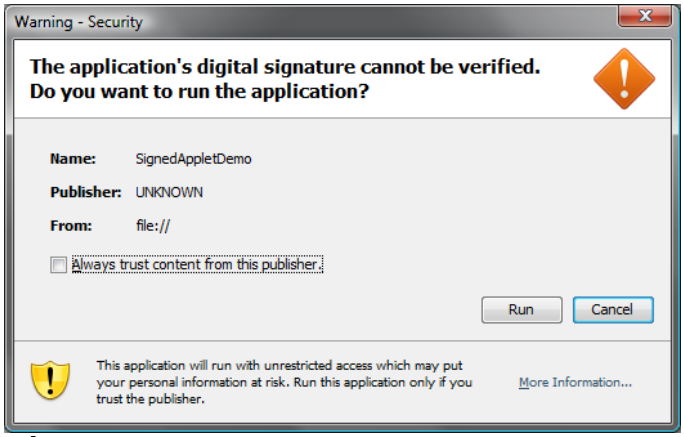

Figure 1

The security warning dialog box lets the user decide whether to accept or deny the applet.

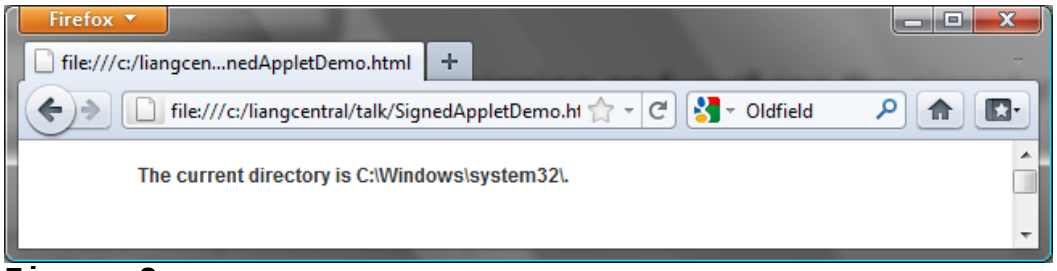

Figure 2

The applet accesses the user file system to display the current directory where the applet is running.