

內容介紹：

- * Renegade 程式
- * HotJava
- * Navigator
- * Internet Explorer
- * 總結

第八章

經過簽署的 APPLETS

經過簽署的 Applets (Signed Applets) 是 Java 1.1 的新發明。一個經過簽署的 Applet，乃是由一群經過簽署的 .class 檔及其它附加的圖檔或音效檔所組成。經過簽署的 Applets 令人既興奮又期待，它可以跨出傳統 Java 1.0 applet sandbox 的限制，讓這些 Applets 從事更多有趣且有用的工作，比如讀取磁碟檔案，及開啟與指定主機間的網路連線。

在理論上，一個經過簽署的 Applet 運作方式如下：

- 1) 一個軟體開發者（就叫它約瑟芬好了）從一個可信任的證明權威 CA 那兒（比如 VeriSign）取得了一份證明，在開立給約瑟芬證明之前，CA 必須查證約瑟芬的身份。
- 2) 當你在 Web 上遊蕩時，偶然間瀏覽到一頁包含約瑟芬所寫的 Applet 的網頁。她已經用她的 Private Key 將此 Applet 簽署。你的瀏覽器這時會問你，是否接受這個由約瑟芬所簽署的 Applet 跨出 Sandbox 的藩籬。

是什麼原因讓你對執行這些簽證過的 Applets 感到放心呢？

- 因為這個 Applet 已經被簽署過了，所以你可以確定這個 Applet 未遭邪惡第三者的變造。
- 因為約瑟芬的身份已經經過一個 CA 背書。這個 CA 簽署了約瑟芬的證明，某種程度上，你可以確定約瑟芬的確是她所自稱的那個人。

以目前來說，經過簽證的 Applets 是缺乏彈性且結構複雜的怪獸。三種最常用的瀏覽器（Sun 的 HotJava, Netscape 的 Navigator 及 Microsoft Internet Explorer）都支援這種 Applets，但每種瀏覽器都有自己的 Applet 儲存格式。而且在處理方式上，這三種瀏覽器也各自有其不同的方式。Netscape Navigator 及 Internet Explorer 更由於與 CA 之間的互動不佳，使得情況目前仍是渾沌不明。

每一個瀏覽器都有不同的簽證方法，因為目前尚未有標準的為客戶端設計的證明資料庫。當一個經過簽證的 Applet 被下載的時候，它的簽章會經由簽署者的 Public Key 查驗。簽署者的 Public Key 是包含在一份和 Applet 一同下載的證明當中。每個瀏覽器都有它自己內部的證明資料庫。

先不管程序或格式不相容的問題，經過簽署的 Applets 很難使用的原因，其實是整個技術尚未成熟。簡而言之，就是目前經過簽署的 Applet 開發工具的 Bug 太多。

在本章中，我將描述如何建立並使用一個經過簽署的 Applet 於 HotJava 1.0, Netscape Navigator 4.01 及 Internet Explorer 這些瀏覽器上。我會簡單的介紹一下各種瀏覽器的設定，不過焦點還是集中在「打包」(package) 與簽署 Applets 上。我不會花很多時間來介紹這些公司所開發的工具及瀏覽器的細節；相反的，本章是作為在這三個瀏覽器平台上，開發經過簽證的 Applets 的快速指南。

我從介紹一個最簡單的 Renegade Applet 開始。這個 Applet 將會大搖大擺的走出 Sandbox。之後，我會向大家示範如何在三種瀏覽器上簽署這個 Applet。

Renegade 程式

等一下我們在三種瀏覽器上工作的時候，都會用到同一個叫 Renegade 的 Applet。Renegade 並不是一個危險的程式，但它會用 `System.getProperty("user.name")` 這個函式呼叫來獲得你的名稱。這個動作在 Applet sandbox 中是不被允許的。在 Renegade 程式中，我們將這個函式呼叫放在一個 Try 區塊中，以便在 SecurityException 例外產生時能適當的反應。請將接下來的程式碼放在一個叫 Renegade.java 的檔案中。

```
import java.applet.*;
import java.awt.*;

public class Renegade extends Applet {
    private String mMessage;

    public void init() {
        try {
            mMessage = "Your name is " + System.getProperty("user.name") + ".";
        }
        catch (SecurityException e) {
            mMessage = "Can't get your name, due to a SecurityException.";
        }
    }

    public void paint(Graphics g) {
        g.drawString("Renegade", 25, 25);
        g.drawString(mMessage, 25, 50);
    }
}
```

包含這個 Applet 的網頁名稱叫 Renegade.html，其內容如下：

```
<html>
<head>
</head>
<body>
    <applet code = Renegade width = 300 height = 200></applet>
</body>
</html>
```

當你瀏覽 `Renegade.html` 這個網頁，瀏覽器會試圖執行 `Renegade` 這個 Applet。如我們所預期的，`System.getProperty()` 這個函式呼叫一定會失敗。圖 8-1 顯示了這個 Applet 在 Navigator 4.1 執行的情況。

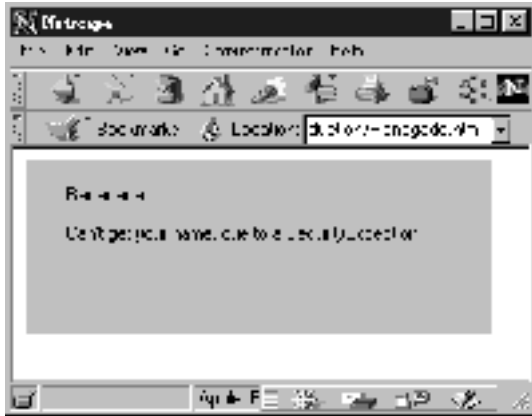


圖 8-1：尚未簽證的 `Renegade` applet 不能跨出 applet sandbox。

Hot Java

要順利執行上面的 Applet，在 HotJava 裡的程序可說是最簡單的。因為發展這個瀏覽器的人和寫 Java Security API 的是同一群人嘛！HotJava 可以辨識壓在 JAR 檔案中的 Applet。（請見附錄 C，裡頭有 `jar` 的使用方式。）如同在附錄 D 中所討論的，你可以用 `javakey` 這個工具來簽署一個 JAR 格式的檔案。簽署時所需的 Key 是從 `javakey` 資料庫中的 Identity 物件來取得。HotJava 可以辨識經過簽署的 JAR 檔案，並讓你自行定義針對特定簽署者的安全策略。

所謂安全策略，是針對某特定簽署者所訂定的一套安全規則。舉個例子來說，我也許會針對約瑟芬所簽署的 Applet 定義下列的安全策略：

- Applets 可以存取 `user.name` 這個系統資訊。
- Applets 可以將檔案寫到本地磁碟機的 `c:\temp` 目錄下。
- Applets 可以連線到 `www.josephine.com` 這個網站。

HotJava 讓你可以針對不同的簽署者，定義安全策略到極為細密的程度。我們待會兒還會針對這個功能稍作討論。

對 HotJava 來說，建立一個簽署過的 Applet 需要三個步驟：

- 1) 準備簽署工具。
- 2) 將 Applet 的各個元件包紮好壓到一個 JAR 檔案中。
- 3) 簽署 Applet

準備簽署工具

你可以用 `javakey` 來建立一個簽署工具，並產生其所需要的 Key。如果你不確定該怎麼做，請參考附錄 D。瑪麗安將成為 Renegade 這個 Applet 的簽署者。我們會用一份自我簽署的證明來簽署此 Applet。

如果你還沒建立瑪麗安專用的簽署工具，請現在就開始建立。首先，請先在 `javakey` 資料庫中建立瑪麗安的紀錄：

```
C:\ javakey -cs Marian true
Create identity [Signer] Marian [identitydb.obj][trusted]
```

產生一套瑪麗安專用的 Keys 如下：

```
C:\ javakey -gk Marian DSA 1024
Generated DSA keys for Marian (strength: 1024).
```

現在我們必須產生瑪麗安自我簽署的一份證明。首先，請建立一個證明指示檔，`MarianCertificate directive`，檔案的內容如下：

```
issuer.name=Marian

subject.name=Marian
subject.real.name=Maid Marian
subject.org.unit=Overprotected Daughters
subject.org=Royal Castle
subject.country=England
```

```
start.date=06 February 1998
end.date=31 December 1998
serial.number=1001

signature.algorithm=DSA

out.file=Marian.certificate
```

接下來，請用 `-gc` 這個選項產生證明如下：

```
C:\ javakey -gc MarianCertificate.directive
Generated certificate from directive file MarianCertificate.directive.
```

到此為止，瑪麗安這位簽署者已經有了一對 Keys 及一份自我簽署證明。比較符合現實的情況是，你擁有的是由 CA 所開立的證明，以為瑪麗安的身份作保。自我簽署證明比較難以讓人信賴。我會在這裡用它，只是為了舉例方便。

很不幸的，JDK 1.1 並未提供與 CA 互動以產生「真正」的證明工具。在 JDK 1.2 中，`javakey` 已被 `Keytool` 及 `jarsigner` 所取代。`Keytool` 工具程式有一個用來向 CA 索取證明的功能，詳情請見第五章。

註解

在 JDK 1.2 中，`javakey` 已經被淘汰。一個新的工具程式 `jarsigner` 會從一個 `KeyStore` 中獲得 key 資訊，來簽署 JAR 格式的檔案。`KeyStore` 物件可以用 `keytool` 來管理。但是和 JDK 1.2 一樣，是由 `jarsigner` 所產生，經過簽署的 JAR 檔案，是不能在 HotJava 1.0 或 HotJava 1.1 上運作的，瀏覽器只會把這些 JAR 檔案當成一般的未經簽署的檔案來看待。

包裹 APPLET 的各個元件

HotJava 可以辨識 jar tool 所建立的 JAR 格式的檔案。我們的 Applet 只包含一個檔案，所以我們可以利用下列命令來建立一個 JAR 檔案：

```
jar -cf Renegade.jar Renegade.class
```

上面這個命令會將 Renegade.class 這個檔案壓到一個叫 Renegade.jar 的 JAR 檔案中。

簽署 APPLET

我們可以如附錄 D 中所描述的，用 javakey 工具程式來簽署 JAR 檔案。指示檔案的內容如下：

```
signer=Marian
cert=1
chain=0
signature.file=MARIANSG
out.file=hjRenegade.jar
```

請將此檔案存成 MarianSign.directive，利用下列命令來簽署此 JAR 檔案：

```
javakey -gs MarianSign.directive Renegade.jar
```

此 JAR 檔案是由瑪麗安的 Private Key 所簽署，簽署結果產生的 JAR 檔案為 hjRenegade.jar。

測試 APPLET

我們需要一個參考到上列那個經過簽署的 JAR 檔案的 HTML 網頁，來執行我們的 Applet。網頁 `hjRenegade.html` 的內容如下所示：

```
<html>
<head>
</head>
<body>
  <applet code = Renegade.class archive = hjRenegade.jar
        width = 400 height = 200></applet>
</body>
</html>
```

如果 `Renegade.class` 及 `hjRenegade.jar` 這兩個檔案同時存在於 HTML 檔案所在的目錄下，HotJava 將會選用 `Renegade.class` 這個未經簽署的 Applet 類別。為了避免困擾，請將 `Renegade.class` 這個檔案從包含該 HTML 及 JAR 檔案的目錄下移除。

現在請執行 HotJava 並開啟 `hjRenegade.html` 這個檔案。HotJava 會代表 Applet 來向你請求存取 `user.name` 這個屬性的權限，如圖 8-2 所示。



圖 8-2：HotJava 請求權限的狀況

如果你允許了請求，這個 Applet 便能夠順利的展示與執行。在你接下來瀏覽網頁的過程中，HotJava 都會記得你已經同意了此項系統屬性的存取。

設定瀏覽器

在 HotJava 中，你可以讓一些存取權限永久有效。特別是我們可以針對某特定簽署者給予一套特許的存取權限。任何 HotJava 所遭遇到的經過簽署的 Applets 都會自動被給予簽署者的權限。

要開啟這項功能，請到 HotJava 選單的 Edit->Preference->Applet Security 項目下選擇 Advanced Security Settings。瑪麗安的證明此時應該出現在表單盒當中。要將指定一套特殊的存取權限給瑪麗安，首先你必須查驗她的證明。請按下 Verify 按鈕來達成這項工作，然後按 OK。現在你可以在表單盒下面的選項當中選擇適用於瑪麗安的存取權限。圖 8-3 顯示了在查驗瑪麗安證明的的工作完成後，進階安全性設定的選項。

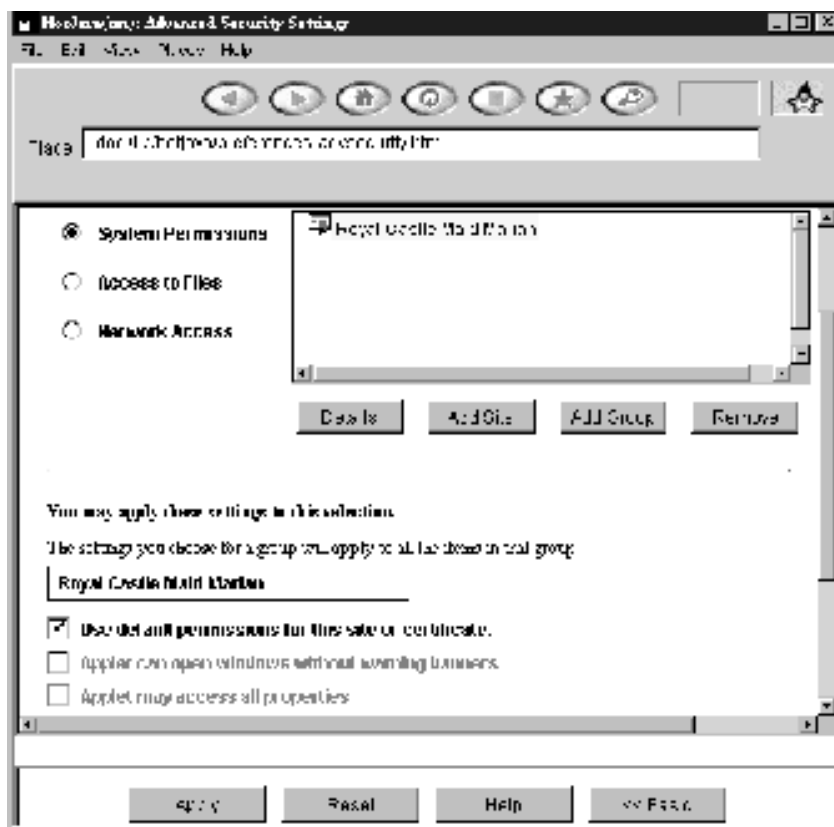


圖 8-3：HotJava 的進階安全設定

由瑪麗安所簽署的 Applet 獲得的是預設的存取權限。要指派給他其他特定的存取權限，請首先取消 User default permissions for this site or certificate 選項。接下來你可以改選 Applet may access all properties 選項。下次你在 HotJava 上執行 Renegade 這個 Applet 時，就不會再被問相同的存取權限問題了。

我們暫且略過關於查驗證明簽章的討論，就當作證明驗證成功，把這件好消息記起來，繼續下面未完的工作。HotJava 很抗拒去接受一個自我簽署的證明，所以它會問你要不要執行額外的查驗工作。原始的想法，是你應該和證明的擁有者進行面對面的確認，仔細核對證明簽章的真實性如何。請注意，這個提議只適用於自我簽署的證明。如果你已經查證瑪麗安的證明，並確認無誤，其它由瑪麗安所簽署的證明便可以自動依據瑪麗安的證明查驗。

比較完美的狀況是，扮隨瑪麗安所簽署的 Applet 一起傳送到瀏覽器的是一個由證明權威 CA 所開立貨真價實如假包換的證明，而非瑪麗安的自我簽署證明。JDK 1.1 並沒有包含任何能產生真正證明的工具。JDK 1.2 中的 Keytool 加入了向 CA 請求一份證明的功能。不過很可惜的，由 jarsigner 所產生經過簽署的 JAR 檔案無法在 HotJava 上運作。

Netscape Navigator

要建立一個可以在其它瀏覽器中執行的經過簽署的 Applet，其步驟和上面大致雷同。Netscape Navigator 能夠辨識 JAR 格式的檔案，但不能辨識由 javakey 所產生的簽章。由此可推，HotJava 無法辨識供 Navigator 使用的簽章。

準備簽署工具

要簽署一個 Applet 以使它能在 Navigator 上順利執行，Navigator 的資料庫必須有一份與簽署該 Applet 的 Key 相應的證明，如此裡應外合，Applet 才能正確執行。你可以向證明權威 CA 購買這個證明。像我就花了 19.95 塊美金從 VeriSign (<http://www.verisign.com/>) 那兒買了一份“Class 2 Digital ID”的證明。我可以這份證明來簽署我的程式碼，使用期限是一年。請注意，VeriSign 並不是唯一提供這種服務的機構，世界上有很多的 CA。我會用 VeriSign 的原因，是因為透過網

路申購 VeriSign 的證明非常方便。VeriSign 把申請證明的整個流程弄得極為流暢，你可以簡單付錢（20 美金），輕鬆拿到證明，一點兒也不費力。在這個簡單的流程背後，涉及許多繁瑣的法律條文與簽約細節，如果你對使用證明抱持嚴肅的態度，應該把這些文件仔細讀過。

VeriSign 提供兩個用來簽署程式碼的證明類別：

- Class 2 證明：適用於從事軟體開發的個人用戶，一年 20 塊美金。VeriSign 會用自動化的程序查驗你所提供給他們的個人資訊，並且會在收到你個人資料的五分鐘內開立一份證明。
- Class 3 證明：適用於從事軟體開發的公司行號，一年 400 美金。VeriSign 對於這類證明的開立比較謹慎。公司需要提供更多的資料，經過詳細的查驗後，才能順利獲得一份證明。

為了要簽署一份程式碼以使其可在 Navigator 上執行，我買了一份 Class 2 的證明。如果你也想買的話，請先連到 VeriSign 的網頁 <http://www.verisign.com/>，順著網頁上的指示註冊以獲得一份證明。當我順著流程進行時，Navigator 為我產生了一對 Key，並將其中的 Public Key 傳給 VeriSign。幾分鐘之後，VeriSign 寄了一封電子郵件給我，要我去指定的 URL 下領取我的證明。這份證明由 Navigator 下載之後，Navigator 會將它連同之前產生的 Private Key 儲存到一個私有證明資料庫中。Netscape 的簽署工具 zigbert 能夠存取這個資料庫以簽署程式。我很想好好的討論一下細節，但因為 VeriSign 的證明申請流程實在太善變，我告訴你的資訊可能在本書付梓時就已經不適用了，所以這裡只說個大概。

有另一個免付費的方法可以在 Navigator 中測試經過簽署的 Applet。這種方法 Netscape 稱為 codebase trust。請參考 <http://developer.netscape.com/library/technote/security/sectn2.html> 以獲得更詳盡的資訊。

請求存取權限

我們可以在 HotJava 上執行 Renegade 這個 Applet，而不必修改任何程式原始碼；但在 Netscape Navigator 情況就沒這麼單純了。就算你的 Applet 已經經過簽署，Navigator 還是不會在請求存取權限之前，讓你執行一些可能危及安全性的敏感動作。在你跨出 sandbox 前，你的 Applet 會詢問你是否允許這樣的動作。Netscape 把這整個防護系統叫作 Capabilities API，其運作方式如下：

- 1) 你的經過簽署的 Applet 會用 Capabilities API 來詢問 Navigator，是否允許 Applet 執行某個具有潛在危險的動作。舉例來說，你的 Applet 或許會請求寫檔到本地磁碟機的權限。
- 2) 接下來 Navigator 會檢查它記載權限的資料庫，看看該 Applet 的簽署者是否被允許執行被請求的動作。如果資料庫中沒有相關的記錄，會有個對話框跳出詢問使用者的意見。使用者可以允許或拒絕該動作的執行，對話框詢問的結果將儲存在記載權限的資料庫中，供日後參考。

上面的流程看起來相當不錯，但是它意味著你必須修改原本 Renegade 這個 Applet 的原始碼。此外，由於加進了對 Capabilities API 的支援，我們所製作的這個 Applet 只能在 Netscape 的瀏覽器上執行。這與 Java 原先大力鼓吹的「只寫一次，各處通用」是相違背的。沒錯，每一種瀏覽器能夠識別的格式各有不同，使你建立一個經過簽署且各處通用的 Applet 的美夢必將幻滅。但是，加入對 Capabilities API 的支援並不僅只是針對同一個原始的 Applet 使用不同的簽署方式而已，你還必須去修改 Applet 的程式碼，才能讓 Applet 順利執行。

如果你願意改變 Applet 程式碼，這裡有個方法給你參考。在 Renegade 這個 Applet 中，方法很簡單，只要在程式中加入呼叫 `netscape.security.PrivilegeManager` 類別的成員函式，要求讀取系統屬性的權限即可：

```
import java.applet.*;
import java.awt.*;

import netscape.security.PrivilegeManager;

public class PrivilegedRenegade extends Applet {
    private String mMessage;
```

```
public void init() {
    try {
        PrivilegeManager.enablePrivilege("UniversalPropertyRead");
        mMessage = "Your name is " + System.getProperty("user.name") + ".";
    }
    catch (netscape.security.ForbiddenTargetException e) {
        mMessage = "Can't get your name, due to a ForbiddenTargetException.";
    }
    catch (SecurityException e) {
        mMessage = "Can't get your name, due to a SecurityException.";
    }
}

public void paint(Graphics g) {
    g.drawString("PrivilegedRenegade", 25, 25);
    g.drawString(mMessage, 25, 50);
}
}
```

在你的 CLASSPATH 中，你必須擁有 Netscape Java 類別才能成功編譯上面的程式。在我的機器上，這表示我必須加入以下的路徑：

```
c:\Program Files\Netscape\Communicator\Program\Java\Classes\java40.jar
```

<http://developer.netscape.com/library/documentation/signedobj/capsapi.html> 有許多關於 Capabilities API 的資料，請自行參考。

簽署 APPLET

Netscape 的簽署工具叫 zigbert。你可以在 <http://developer.netscape.com/library/documentation/signedobj/zigbert/> 下找到關於 zigbert 的資訊。Zigbert 簽署的是一整個目錄，而不是由一堆檔案壓縮而成的 JAR 檔案，所以你必須在建立 JAR 檔案之前先執行簽署的動作。

在這個範例當中，我們只需要簽署一個檔案 `PrivilegedRenegade.class`。請將這個檔案移到它專屬的目錄 `signdir` 當中。在簽署目錄的時候，你必須告訴 zigbert 哪裡可以找到你的 Keys 和證明，還有你要用哪一份證明簽署資料。在我的 Win95 機器上，Netscape 的 Key 和證明是放在 `c:\Program Files\Netscape\users\jonathan` 這個目錄下。我用的是從 VeriSign 購買的證明：

```
C:\> zigbert -d"c:\Program Files\Netscape\users\jonathan" -k"Jonathan B Knudsen VeriSign Trust Network ID" signdir
using key "Jonathan B Knudsen VeriSign Trust Network ID"
using certificate directory: c:\Program Files\Netscape\users\jonathan
Generating signdir/META-INF/manifest.mf file..
--> PrivilegedRenegade.class
Generating zigbert.sf file..
using key database: c:\Program Files\Netscape\users\jonathan/key3.db
tree "signdir" signed successfully

C:\>
```

zigbert 會在 META-INF 目錄下建立簽章檔及清單檔。

將 APPLET 的各個元件包裹好

好啦，現在你已經用 zigbert 建立好簽署資訊了。接下來的工作就是把 `PrivilegedRenegade.class` 這個檔案及簽署資訊一起包到一個 JAR 檔案中。當你下載 zigbert 的時候，應該同時也會下載一個叫 zip 的工具程式，它是拿來建立 JAR 檔案用的。你只要告訴它 JAR 檔案的名稱，以及你要壓縮的各檔案的名稱。要為我們的 `PrivilegedRenegade.class` 檔案建立一個「收藏檔」(Archive File)，請在 `signdir` 目錄下執行下列命令：

```
C:\ zip -r ..\nsRenegade.jar *
  adding: PrivilegedRenegade.class (deflated 44%)
  adding: META-INF/manifest.mf (deflated 14%)
  adding: META-INF/zigbert.sf (deflated 27%)
  adding: META-INF/zigbert.rsa (deflated 40%)
```

```
C:\
```

一切到此大功告成。你可以用 zigbert 來查驗一下收藏檔是否已被正確無誤的簽署：

```
C:\ zigbert -d"c:\Program Files\Netscape\users\jonathan" -v nsRenegade.jar
using certificate directory: c:\Program Files\Netscape\users\jonathan
archive "nsRenegade.jar" has passed crypto verification.
```

```
      status  path
-----  -
      verified PrivilegedRenegade.class
```

```
C:\
```

使用 0.6 版本而非 0.6a

當我在寫這本書時，zigbert 有兩個版本。0.6 版本需要一個系統檔 msvcrt.dll 才能恰當的在你系統上呈現。如果沒有這個檔案，你會被強迫使用 0.6a 版本的 zigbert。千萬別這麼做。0.6a 版本的 zigbert 完全沒用。你用 0.6a 版本 zigbert 程式簽署產生的任何檔案都不能正確的運作；事實上，連 0.6a zigbert 程式自己都不能查驗它自己簽署檔案的真實性。

要使用能正確運作的程式版本 0.6 zigbert，你需要 msvcrt.dll 這個檔案。別擔心，你的系統裡可能早就有這個檔了。（最有可能的地方是 c:\windows\system）。要讓 zigbert 0.6 順利執行，請直接複製一份 msvcrt.dll 這個檔案，並把複製好的檔案也命名為 msvcrt.dll。

測試 Applet

我們需要一個和前面稍微不同的 HTML 檔案，來和 Netscape 專用的 Applet 搭配。類別名稱現在已改稱為 `PrivilegedRenegade.class`，而包含它的收藏檔則稱為 `nsRenegade.jar`。下面是修改過的 HTML 檔案 `nsRenegade.html`：

```
<html>
<head>
</head>
<body>
  <applet code = PrivilegedRenegade.class archive = nsRenegade.jar
    width = 400 height = 200</applet>
</body>
</html>
```

請注意，如果 `PrivilegedRenegade.class` 和 `nsRenegade.jar` 這兩個檔案和 `nsRenegade.html` 這個檔案出現在同一個目錄下，瀏覽器將使用 `.class` 檔而忽略收藏檔 `.jar` 的存在，這會讓情況變的十分混亂，因為 Applet 會照常執行，但執行的不是被簽署的 Applet。要避免這種混淆，請將 `PrivilegedRenegade.class` 檔案從目錄中移除，以強迫瀏覽器執行收藏檔中所包含的 Applet。

現在請執行 Navigator 並打開 `nsRenegade.html` 這個檔案。如同在 HotJava 的情況，Navigator 會決定這個 Applet 能否跨出 sandbox 運作。由於沒有前例，Navigator 會詢問使用者是否允許這個 Applet 的動作，如圖 8-4 所示。

你可以同意或拒絕存取權限的請求，還可以選擇性的選取 `Remember this decision` 這個選項，讓 Navigator 下次繼續使用你這次所下的決定。

設定瀏覽器

現在你可以仔細檢閱（而非設定）針對這個 Applet 簽署者所訂定的存取權限。請選取 Communicator 選單下的 Security Info 項目，然後選取 Java/JavaScript 選項。你應該會在一個清單中看到 Applet 簽署者的名字，如圖 8-5 所示。

雖然這個視窗中包含了一個 `Edit Privileges` 的按鈕，但按下這個按鈕，只能看到一個你曾經針對此簽署者所頒定的存取權限大綱，你並不能在此修改。之後的版本也許會讓使用者在這裡做些存取權限的細部修改。

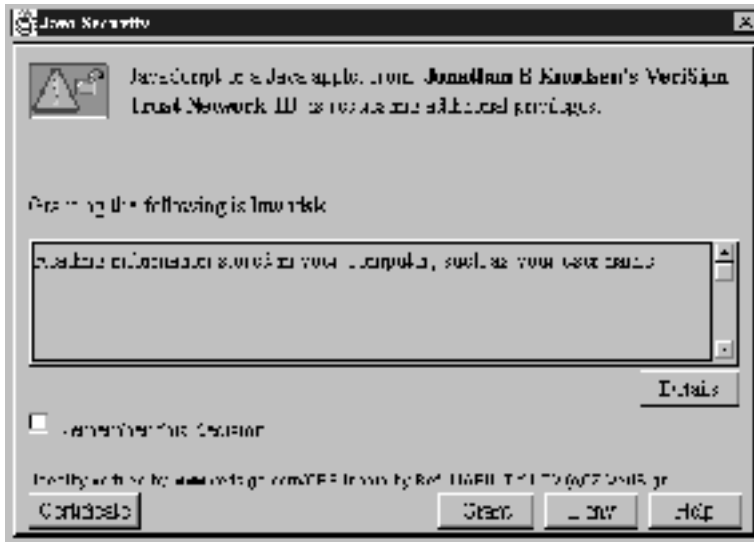


圖 8-4 Navigator 請求存取權限的情況



圖 8-5 Netscape Navigator 的程式碼簽署工具

Internet Explorer

和往常一樣，Microsoft 自己發明一套方法來解決程式碼簽署的問題。它使用自己的收藏檔格式，並且開發了一套以 Microsoft CryptoAPI 為基礎的程式碼簽署工具，來完成簽署 Applet 的工作。

選購材料

你必需去下載兩個軟體才能製作並測試在 IE 上執行經過簽署的 Applet。首先，請到 www.microsoft.com/java/ 下載簽署程式碼與壓縮收藏檔的工具程式 SDK for Java 2.0。

準備簽署工具

Microsoft 的工具允許你建立一份試用版的證明以達成簽署的工作，所以你不需花 20 塊美金買一份真正的證明，就可以體驗一下製作經過簽署 Applet 的滋味。

如果你想要用真正的證明簽署你的程式碼，可以向 VeriSign (<http://www.verisign.com/>) 購買。

你可以使用安裝在 SDK for Java 中的工具程式來建立一份試用版的證明，在 SDK-Java2.0\Bin\PackSign 目錄下鍵入下列命令：

```
MakeCert -sk JonathanKey -n CN=JonathanCompany Jonathan.cert
```

以上的動作會建立一個叫 Jonathan.cert 的證明檔。它使用一個叫 JonathanKey 的 secret Key。如果這個 Key 原先不存在，MakeCert 會幫你建立一個。這把 Key 會被存放在一個 Private Key 管理資料庫中，以待你將來存取。-n 選項是用來指定這個新建立的證明是歸屬誰的名下。你需要一個軟體出版者證明 SPC (Software Publisher Certificate) 才能簽署程式碼。SDK for Java 有一個相當實用的工具程式，可以將一份證明轉成一份 SPC：

```
Cert2SPC Jonathan.cert Jonathan.spc
```

建立測試版本的簽署工具的工作到此完成。你已經擁有一個 SPC 檔及一把 Private Key，這對簽署程式碼來說已經足夠。

如果你是從 VeriSign 那兒買一份證明來使用，你可以自行決定 Private Key 及 SPC 檔的格式。請將這兩個檔都存放在安全的地方，特別是 Private Key 更得特別小心。在 Windows 95 的系統上，最好把這兩個檔放在磁片或其它可抽取的資料儲存裝置；在 NT 系統上，檔案所有權與存取權的做法可以給你的硬碟資料某種程度上的保全，或許可以冒一點風險，將這些檔案存在你的硬碟中。

將 APPLET 的各個元件包好

Microsoft Applet 會被包到一種名叫 cabinet 的檔案中，這種檔案的副檔名是 .cab。你可以使用 cabarc 這個工具程式來處理 cabinet 檔案。在這裡的範例當中，我們只需要新建立一個包含 Renegade.class 檔的 cabinet 檔。我們使用 n 選項代表建立一個新的 cabinet 檔：

```
C:\ cabarc n ieRenegade.cab Renegade.class

Microsoft (R) Cabinet Tool - Version 1.00.0601 (03/18/97)
Copyright (c) Microsoft Corp 1996-1997. All rights reserved.

Creating new cabinet 'ieRenegade.cab' with compression 'MSZIP':
  -- adding Renegade.class

Completed successfully

C:\
```

簽署 APPLET

這裡用來簽署的工具名叫 SignCode。如果你已經建立好一份測試版的簽署工具，接著可用下列命令來簽署你的 cabinet 檔：

```
SignCode -spc Jonathan.spc -k JonathanKey ieRenegade.cab
```

上面的命令會使用 JonathanKey 這把 Private Key，來簽署給定的 cabinet 檔，並將 Jonathan.spc 這個證明檔附到 ieRenegade.cab 檔案中。

如果你擁有來自 CA 如假包換的證明，你應該收到一個 SPC 檔及一個 Private Key 檔。當我向 VeriSign 購買證明時，我將收到的檔案儲存成 Jonathan.spc 及 Jonathan.pvk。我用來簽署 Renegade Applet 鍵入的命令如下：

```
SignCode -spc Jonathan.spc -v Jonathan.pvk ieRenegade.cab
```

你可以用 ChkTrust 命令來檢查一個 cabinet 檔是否已經被簽署：

```
ChkTrust ieRenegade.cab
```

上面這個命令會帶出一個視窗，視窗中會描述 cabinet 檔中的簽章長相為何。如果你有興趣的話，也可以檢視與該 cabinet 檔相關聯的證明。

測試 Applet

又來了，我們需要一個新的 HTML 檔案以執行此 Applet。這兒是一個 IE 版本的 HTML 檔案 ieRenegade.html：

```
<html>
<head>
</head>
<body>
  <applet code = Renegade width = 300 height = 200>
  <param name = cabbase value = ieRenegade.cab>
  </applet>
</body>
</html>
```

如同其它的瀏覽器，你應該將 `renegade.class` 這個檔案從包含此 HTML 檔案的目錄中移除，以迫使 IE 去收藏檔中挖出 `Renegade` 類別來執行。

請執行 IE 並打開 `ieRenegade.html` 檔案，和之前一樣，瀏覽器會辨識出此經過簽署的收藏檔並顯示一個訊息，詢問你是否允許這個 Applet 執行。如圖 8-6 所示：

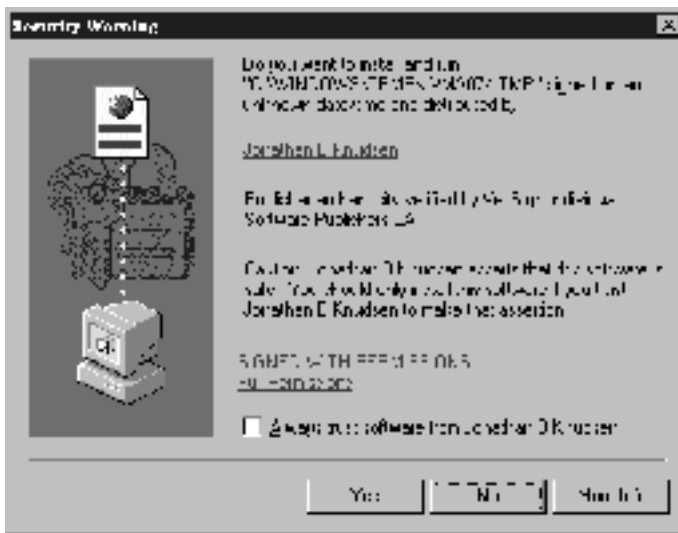


圖 8-6：IE 詢問使用者是否要執行一個經過簽署的 applet

請注意，IE 要你做出一個全有或全無的決定。你要嘛就允許這個 Applet 的執行，讓它能夠跨出 sandbox 完整的存取本地系統的資源；要嘛就不讓它執行，休想越雷池半步。回憶一下，HotJava 會在每次 Applet 要跨出 sandbox 時，就請求存取權限一次。Microsoft 所用的方法或許反應出了它的實質利益：ActiveX。ActiveX 並不受規範詳細的存取權限政策所控制，這和 Java Applet 明顯不同。

Microsoft 擁有一個類似 Netscape Capabilities API 的系統，用來詢問指定的存取權限。這個系統在 <http://www.microsoft.com/java/security/> 目錄下有描述，請自行參考。

總結

目前經過簽署的 Applet 在執行上有一些困難，主要是因為下列兩個原因：

- 各個主流瀏覽器各自為政，使用不同的格式簽署 Applet。
- 程式碼簽署工具目前還是充斥著 Bug。

你可以在伺服器端偵測使用者所使用的瀏覽器型態，並傳回與瀏覽器相關的 Applet 以解決第一個問題；你也可以使盡吃奶的力氣來解決第二個問題。如果你的應用程式非常依賴經過簽署的 Applet，你應該在放手實作前好好沉思，慢慢琢磨你的設計。在第十二章中，我會探討一些當紅的應用程式架構，以及把 Applet 當成客戶端使用的優缺點。如果你正在開發的是一個就某方面來說封閉環境的應用程式，即你的 Applet 只在某特定廠牌的瀏覽器上執行，那麼經過簽署的 Applet 或許非常實用。

當本書正當付梓之際，一個相當有趣而特殊的技術正逐漸出頭：那就是 Sun 的 Java Activator。這是一個可以在 Netscape 或 IE 下以 plug-in 型態執行的 Java 執行期作業環境 JRE (Java Runtime Environment)。經過簽署的 Applet 可以被這個 Java Activator 所支援，不過更高階的技術也在不斷發展當中。

再者，Navigator 及 Java 之間的關係目前尚未明朗。現在 Netscape 是完全免費的，連 Communicator 的原始程式碼都公開了。Netscape 已經聲明，它將不再生產 JVM，而是讓瀏覽器能接受任何廠商發展的 JVM。這會帶來何種影響？請大家靜觀其變。