INTRODUCTION
Applets are small programs that reside on a host computer and are downloaded to a client computer to be executed. This model makes it very easy to distribute and update software. Because the new version of an application only needs to be placed on the server, clients automatically receive and run the updated version the next time they access the application.

The use of applets is possible because of the increasing bandwidth available to Internet and intranet users. The time required to download the programs has been decreasing even as program complexity has been increasing. The development of cross-platform languages such as Sun Microsystems, Inc.’s Java, Microsoft Corp.’s ActiveX, and Netscape Communications Corp.’s JavaScript has made writing applets for many different computers simple — the same exact Java or JavaScript code can be run on a Windows-based PC, a Macintosh, or a UNIX-based system without any porting or recompiling of code. Microsoft is working to port ActiveX to UNIX and Macintosh platforms.

APPLETS AND THE WEB
The World Wide Web is the place that users are most likely to encounter applets today. Java (and to a lesser degree, JavaScript) has be-
come the Webmaster's tool of choice to add interesting effects to Web sites or to deliver applications to end users. Most of the scrolling banners, animated icons, and other special effects found on today's Web pages depend on applets to work. Some Web pages use applets for more substantial applications. For example, MapQuest (http://www.mapquest.com) uses Java and ActiveX to deliver an interactive street atlas of the entire U.S. Wired magazine offers a Java-based chat site that, when accessed over the Web, allows users to download an applet that lets them participate in real-time conferencing.

THE SECURITY ISSUE
Every silver lining has a cloud, and applets are no exception. Applets can present a real security hazard for users and network managers. When Web pages use applets, the commands that tell the client's browser to download and execute the applets are embedded in the pages themselves. Users have no way of knowing whether or not the next page that they download will contain an applet, and most of the time, they do not care. The Internet offers an almost limitless source of applets for users to run; however, no one knows who wrote them, whether they were written with malicious intent, or whether they contain bugs that might cause them to crash a user's computer.

Applets and computer viruses have a lot in common. Both applets and viruses are self-replicating code that executes on the user's computer without the user's consent. Some security experts have gone as far as to say that the corporate network manager should prohibit users from running applets at all. However, applets are becoming an increasingly common part of how users interact with the Internet and corporate intranets, so learning to live safely with applets is important for network managers.

WHAT ARE THE RISKS?
According to Princeton University's Safe Internet Programming (SIP) research team, there have been no publicly reported, confirmed cases of security breaches involving Java, though there have been some suspicious events that may have involved Java security problems. The lack of reported cases is no guarantee that there have not been breaches that either were not discovered or were not reported. But it does indicate that breaches are rare.

As Web surfing increasingly becomes a way to spend money, and applets become the vehicle for shopping, attacks on applets will become more and more profitable, increasing the risk. Sun, Netscape, and Microsoft all designed their applet languages with security in mind.
Java programs are developed in a language similar to C++ and stored as source code on a server. When a client, such as a Web browser, requests a page that references a Java program, the source code is retrieved from the server and sent to the browser, where an integrated interpreter translates the source code statements into machine-independent bytecodes, which are executed by a virtual machine implemented in software on the client. This virtual machine is designed to be incapable of operations that might be detrimental to security, thus providing a secure sandbox in which programs can execute without fear of crashing the client system.

Java applets loaded over a network are not allowed to:

- Read from files on the client system.
- Write to files on the client system.
- Make any network connections, except to the server from which they were downloaded.
- Start any client-based programs.
- Define native method calls, which would allow an applet to directly access the underlying computer.

Java was designed to make applets inherently secure. Following are some of the underlying language security features offered by Java:

- All of an applet’s array references are checked to make sure that programs will not crash because of a reference to an element that does not exist.
- Complex and troublesome pointer variables (found in some vendors’ products) that provide direct access to memory locations in the computer do not exist in Java, removing another cause of crashes and potentially malicious code.
- Variables can be declared as unchangeable at runtime to prevent important program parameters from being modified accidentally or intentionally.

The fact that Sun has licensed Java to various browser vendors adds another level of complexity to the security picture. Not only can security be compromised by a flaw in the Java specification, but the vendor’s implementation of the specification may contain its own flaws and bugs.
DENIAL-OF-SERVICE THREATS

Denial-of-service attacks involve causing the client’s Web browser to run with degraded performance or crash. Java does not protect the client system from these types of attacks, which can be accomplished simply by putting the client system into a loop to consume processor cycles, creating new process threads until system memory is consumed, or placing locks on critical processes needed by the browser.

Because denial-of-service attacks can be programmed to occur after a time delay, it may be difficult for a user to determine which page the offending applet was downloaded from. If an attacker is subtle and sends an applet that degrades system performance, the user may not know that their computer is under attack, leading to time-consuming and expensive troubleshooting of a nonexistent hardware or software problem.

Java applets are not supposed to be able to establish network connections to machines other than the server they were loaded from. However, there are applets that exploit bugs and design flaws that allow it to establish a back-door communications link to a third machine (other than the client or server). This link could be used to send information that may be of interest to a hacker. Because many ready-to-use Java applets are available for download from the Internet, it would be possible for an attacker to write a useful applet, upload it to a site where Webmasters would download it, and then sit back and wait for information sent by the applet to reach their systems.

WHAT KIND OF INFORMATION CAN THE APPLET SEND BACK?

Due to another implementation problem found in August 1996 by the Safe Internet Programming research team at Princeton University, the possibilities are literally endless. A flaw found in Netscape Navigator versions 3.0 beta 5 and earlier versions, and Microsoft Internet Explorer 3.0 beta 2 and earlier versions, allows applets to gain full read and write access to the files on a Web surfer’s machine. This bug means that the attacker can get copies of any files on the machine or replace existing data or program files with hacked versions.

Giving Java applets the ability to connect to an arbitrary host on the network or Internet opens the door to another type of attack. A malicious applet, downloaded to and running on a client inside of a firewalled system, could establish a connection to another host behind the firewall and access files and programs. Because the attacking host is actually inside the secured system, the firewall will not know that the access is actually originating from outside the network.

Another bug found in August 1996 by the Princeton team affects only Microsoft Internet Explorer version 3.0 and allows applets (which are not supposed to be allowed to start processes on the client machine) to execute any DOS command on the client. This allows the applet to delete
or change files or programs or insert new or hacked program code such as viruses or backdoors. Microsoft has issued a patch (available on its Web site at http://www.microsoft.com/ie) to Internet Explorer that corrects the problem.

Princeton’s SIP team also found a hole that would allow a malicious application to execute arbitrary strings of machine code, even though the Java virtual machine is only supposed to be able to execute the limited set of Java bytecodes. The problem was fixed in Netscape Navigator 3.0 beta 6 and Microsoft Internet Explorer 3.0 beta 2.

JAVASCRIPT: A DIFFERENT GRIND

Netscape’s JavaScript scripting language may be named Java, but it is distinct from Sun’s applet platform. JavaScript is Netscape Navigator’s built-in scripting language that allows Webmasters to do cross-platform development of applets that control browser events, objects such as tables and forms, and various activities that happen when users click on an object with their mouse.

Like Java, JavaScript runs applications in a virtual machine to prevent them from performing functions that would be detrimental to the operation of the client workstations. Also like Java, there are several flaws in the implementation of the security features of JavaScript. Some of the flaws found in JavaScript include the ability for malicious applets to:

- Obtain users’ E-mail addresses from their browser configuration.
- Track the pages that a user visits and mail the results back to the script author.
- Access the client’s file system, reading and writing files.

A list of JavaScript bugs and fixes can be found on John LoVerso’s Web page at the Open Software Foundation (http://www.osf.org/~loverso/javascript/).

ActiveX: Microsoft’s Vision for Distributed Component Computing. Microsoft’s entry in the applet development tool wars, ActiveX, is very different from Java and presents its own set of security challenges. ActiveX is made up of server and client components, including:

- Controls, which are applets that can be embedded in Web pages and executed at the client. Controls can be written in a number of languages, including Visual Basic and Visual C++.
- Documents that provide access to non-HTML content, such as word processing documents or spreadsheets, from a Web browser.
- The Java virtual machine, which allows standard Java applets to run at the client.
• Scripting, which allows the Web developer to control the integration of controls and Java applets on a Web page.
• The server framework, which provides a number of server-side functions such as database access and data security.

Java applets running in an ActiveX environment (e.g., Microsoft’s Internet Explorer Web browser) use the same security features and have the same security issues associated with JavaScript. Microsoft offers a Java development environment (i.e., Visual J++) as well as other sandbox languages (i.e., VBScript, based on Visual Basic and JScript, Microsoft’s implementation of Netscape’s JavaScript) for the development of applications that are limited as to the functions they can perform.

When developers take advantage of ActiveX’s ability to integrate programs written in Visual Basic or C++, the virtual machine model of Java no longer applies. In these cases, compiled binaries are transferred from the server to the Web client for execution. These compiled binaries have full access to the underlying computing platform, so there is no reason that the application could not read and write files on the client system, send information from the client to the server (or another machine), or perform a destructive act such as erasing a disk or leaving a virus behind.

USING AUTHENTICODE FOR ACCOUNTABILITY
Microsoft’s approach to security for non-Java ActiveX applications is based on the concept of accountability — knowing with certainty the identity of the person or company that wrote a piece of software and that the software was not tampered with by a third party. Microsoft sees the issues related to downloading applets from the Web as similar to those involved in purchasing software; users need to know where the software is coming from and that it is intact. Accountability also means that writers of malicious code could be tracked down and would have to face consequences for their actions.

The mechanism that Microsoft offers to implement this accountability is called Authenticode. Authenticode uses a digital signature attached to each piece of software downloaded from the Internet. The signature is a cryptographic code attached by the software developer to an applet. Developers must enter a private key (known only to them) to sign their application, assuring their identity. The signature also includes an encrypted checksum of the application itself, which allows the client to determine if the applet has changed since the developer released it.

ACTIVEX: THE DOWNSIDE
This approach provides developers and users with access to feature-rich applications, but at a price. If an application destroys information on a user’s computer, accountability will not help recover their data or repair
damage done to their business. Once the culprit has been found, bringing them to justice may be difficult because new computer crimes are developing faster than methods for prosecuting them.

Microsoft acknowledges that Authenticode does not guarantee that end users will never download malicious code to their PCs and that it is a first step in the protection of information assets.

Further information on ActiveX can be found on Microsoft's Web site (http://www.microsoft.com/activex) and at the ActiveX Web site run by CNet Technology Corp. (http://www.activex.com).

AN OUNCE OF PREVENTION
So far, this article has discussed problems posed by applets. Following are some steps that can be taken to lessen the exposure faced by users.

Make Sure the Basics Are Covered
Users need to back up their data and programs consistently, and sensitive data should be stored on secure machines. The surest way to avoid applet security problems is to disable support for applet execution at the browser. If the code cannot execute, it cannot do damage.

Of course, the main downside of this approach is that the users will lose the benefits of being able to run applets. Because the ability to run applets is part of the client browser, turning off applets is usually accomplished at the desktop and a knowledgeable user could simply turn applet support back on. Firewall vendors are starting to provide support for filtering out applets, completely or selectively, before they enter the local network.

Users Should Run the Latest Available Versions of Their Web Browsers
Each new version corrects not only functional and feature issues, but security flaws. If an organization is planning to use applets on its Web pages, it is preferable to either write them internally or obtain them from trusted sources. If applets will be downloaded from unknown sources, a technical person with a good understanding of the applet language should review the code to be sure that it does only what it claims to.

Mark LaDue, a researcher at Georgia Tech has a Web page (available at http://www.math.gatech.edu/~mladue/HostileApplets.html) containing a number of hostile applets available for download and testing. Seeing some real applications may help users recognize new problem applets that may be encountered.

CONCLUSION
IS personnel should monitor the Princeton University Safe Internet Programming group's home page (located at http://www.cs.prince-
ton.edu/sip) for the latest information on security flaws and fixes (under News). It is also a good idea to keep an eye on browser vendors’ home pages for news of new versions.

Applets offer users and network managers a whole new paradigm for delivering applications to the desktop. Although, like any new technology, applets present a new set of challenges and concerns, their benefits can be enjoyed while their risks can be managed.

Al Berg is the director of Strategic Technologies at NETLAN Inc. in New York, NY. He can be reached via E-mail at: al_berg@netlan.com.