

基于 Applet 的 Telnet 功能与实现

姚 剑¹, 徐汀荣¹, 梅 娟²

(1. 苏州大学 计算机科学与技术学院, 江苏 苏州 215006;

2. 东南大学 集团经济与产业组织研究中心, 江苏 南京 210096)

摘 要:介绍了在网络应用中日益重要的 Java Applet 所存在的优缺点, 并由此讨论了产生这些优缺点的 Java Applet 安全模型的发展历程, 进而分析了这种模型在安全性方面的优点, 同时也指出了 Java Applet 给用户带来的不便。在分析了 Java Applet 安全模型结构后, 提出了绕过这种安全模型实现远程登陆功能的系统架构, 并列出了用 Java Applet 实现的基于 Web 的 Telnet 功能的部分代码。

关键词:Applet 程序; 安全; 通信

中图分类号:TP393.08

文献标识码:A

文章编号:1005-3751(2006)01-0210-03

Functions and Implementation of Telnet Based on Applet

YAO Jian¹, XU Ting-rong¹, MEI Juan²

(1. School of Computer Science & Technology, Soochow University, Suzhou 215006, China;

2. Research Center of Group Economy and Industry Organization, Southeast University, Nanjing 210096, China)

Abstract: Analyzes the advantages and disadvantages of Java Applet, which nowadays has been greatly used on network. After that, it discusses the evolutive process of the Java Applet model that engenders these advantages and disadvantages. And it analyzes the inconvenience brought about to users by Java Applet, and the advantages that exist in the safe aspect of this model. Also puts forward a system framework that averts the safe model and achieves Telnet. Eventually, based on Web, presents some programs of Java Applet which are used to implement the application of Telnet.

Key words: Applet; security; communication

0 引言

Internet 显著地改变了人们的生活与工作方式。Java 作为一种为 Web 而产生的语言, 正站在计算机浪潮的最前沿。自从 Java 技术进入应用以来, 人们给予了 Java 平台的安全性以及由于部署 Java 技术所引发的安全问题以极大的关注。Applet 是 Java 最具有意义的特征之一, 因为它能够创建可移植的、可动态下载的、能够在浏览器执行的程序。通过使用 Applet, Web 程序员可以方便地在静态的 HTML 页面中添加动态的内容, Java Applet 使网页变得生动起来。虽然这种新结构会带来很多功能, 但安全问题也随之而来。因为下载了恶意的代码会造成极其严重的后果, 例如随意访问本地文件系统、读取客户机中的重要参数及通过客户机去攻击其他计算机等。

1 Java Applet 安全模型

Java 语言的优势就是在于其网络功能的强大和操作的简易性, 而 Java Applet 作为 Java 技术的一部分, 更是锦

上添花。Java Applet 是可用于嵌入在网页中的可移动代码, 它源自下载其嵌入网页的远程主机, 是在本地主机执行的小程序。

由于受到许多恶意应用的威胁或 Applet 代码本身不够健壮, 因此开始 Java1.0 认为从互联网上下载的一般 Applet 是不可信任的。这些 Applet 小程序在 Java Applet 安全模型(即沙箱模型)中运行, 沙箱模型的实质在于信任本地代码, 对下载的 Applet 小程序运行于一个受限的执行环境中, 并由沙箱模型给 Applet 小程序分配资源和特权, 因而 Applet 小程序只能访问沙箱提供的有限的资源(如图 1 所示)^[1]。实现沙箱模型这种安全机制的防范区域主要有: 字节码验证器、类装载器和安全管理器。

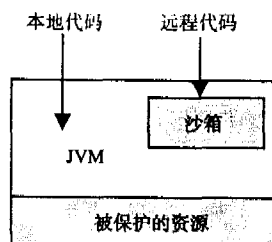


图 1 Java1.0 沙箱模型

1) 在 Applet 小程序被下载到本地主机后, 字节码验

收稿日期: 2005-04-30

作者简介: 姚 剑(1980—), 男, 江苏人, 硕士研究生, 研究方向为网络与数据库; 徐汀荣, 教授, 研究方向为网络与数据库。

证器将对该小程序进行验证,进而保证不会发生溢出,保证不会发生非法类型转换及保证在一个对象的内部只有公共数据和函数才可被访问^[2]。

2)类装载器重要的功能是确保在运行环境中的正在执行的 Applet 小程序不会破坏和冒充本地的安全类。

3)安全管理器是沙箱模型中最重要的。基本 Java 类库中的所有方法必须在执行任何“危险操作”(如本地文件的输入/输出、网络资源的访问等)之前都要得到安全管理器的许可。它对这些“危险方法”进行 runtime 验证。安全管理器对任何请求都有否决权。

因为下载的 Applet 都在沙箱模型中运行,而此安全模型对在其中运行的代码能否访问本地机的资源都作出了明确的规定,所以即使下载的 Applet 小程序是不安全的也不会破坏本地主机系统。

沙箱模型给 Applet 小程序构建了安全的运行环境,但是其弊端也很明显,相对于一般的客户端应用程序,Applet 小程序受到了很多限制,包括 Applet 不允许读写本地的文件系统,只能读其宿主主机上的文件;Applet 不能与非宿主机的任何机器建立连接,它只能对其宿主主机侦听或发送数据包;不能访问系统的属性;不能定义自己的类装载器;不能直接调用本地函数等^[3]。

无疑在 Java1.0 中沙箱安全模型限制了功能更强大的应用程序的开发。

在 Java1.1 中引入了 Applet 签名的概念,使得经过数字签名的 Applet 代码和本地应用程序一样,可以无限制地访问系统资源(如图 2 所示)^[1]。

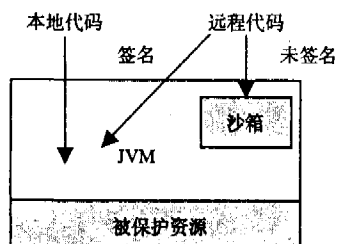


图 2 加入签名的沙箱模型

Java2 后引入了一个涵盖应用程序和 Applet 的更加细致的新的安全策略系统(如图 3 所示),应用策略文件可以给特定的 Applet 制定权限,同时根据签名的 Applet 和策略文件,给带签名的 Applet 分配适当的权限。这样让运行 Applet 客户端更安全,同时可以灵活地分配权限。但

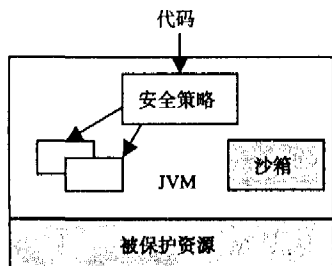


图 3 加入安全策略的沙箱模型

是这里要求每个普通用户都要自己去编辑自己的策略文

件,这又带来了不便^[1]。在实际应用中可以利用 Applet 能和宿主主机通信这一特性来实现某些特殊的功能^[4]。下面介绍的 Telnet 功能的实现就是一例,这种实现方式可以在公司内部局域网范围内使用。

2 Telnet 功能的实现

突破 Java 的安全模型,可以使 Java 小程序实现某些特殊的应用。文中叙述了用 Applet 及在宿主主机上运行的简单自制代理实现了基于 Web 的 Telnet 功能。通过 Applet 程序中的客户端 Socket 向由宿主主机上运行的自制代理 Transocket 对象创建的 ServerSocket 请求连接。然后再由 Transocket 对象创建与第三方 Telnet 服务器连接的 Socket 来建立和第三方主机的连接(如图 4 所示)^[5]。



图 4 结构图

2.1 Applet 和宿主主机通信实现

客户端运行的 Applet:

创建 Socket 对象 client

```
client = new Socket(host, port)
```

其中,host 是宿主机的 IP 地址,port 是自制代理的端口号。

由该对象得到和宿主主机通信的 I/O 流:

```
in = s.getInputStream();
out = s.getOutputStream();
```

在客户端发送连接请求时,客户端首先将客户要连接的作为第三方的 Telnet 服务器的 IP 地址和端口号发送到服务器端,然后由代理服务器根据此 IP 地址和端口号实现和 Telnet 服务器建立连接。

2.2 宿主主机创建自制代理 Transocket 对象

(1)由 Transocket 创建用于侦听客户端请求的 ServerSocket 对象。

```
server = new ServerSocket(port)
```

其中,port 定义为自制代理的通信端口号。

然后用一个循环不断侦听来自客户端的请求信息并返回和客户端通信的 Socket 对象 clientin:

```
while(true){
    clientin = server.accept();
    handle(clientin);
}
```

通过返回的 Socket 对象进一步得到和客户端通信的 I/O 流对象:

```
InputStream cinputstream = clientin.getInputStream();
OutputStream coutputstream = clientin.getOutputStream();
```

这样,服务器端通过输入流对象 cinputstream 接收客户端发送的命令,再由输出流对象 coutputstream 发送返

回的响应信息。

(2) 创建 Socket 对象。

在 ServerSocket 侦听到 Applet 的连接请求信息后由 ServerSocket 对象的 Accept() 方法返回了与 Applet 客户端通信的 Socket 对象, 同时创建和第三方 Telnet 服务器通信的 Socket 对象, 这和 Applet 客户端创建 Socket 对象的过程一样也同样得到了 I/O 流对象:

```
Socket serv = new Socket(host, port);
```

```
InputStream sinputstream = serv.getInputStream();
```

```
OutputStream soutputstream = serv.getOutputStream();
```

其中 host, port 是第三方 Telnet 服务器的 IP 地址和端口号。

到此已经分别得到了与客户端 Applet 和与 Telnet 服务器端的输入/输出流。接着将这些输入/输出流对象传递给分别实现指令代理发送和响应信息代理接收的两个线程 ClientToTelnet 和 TelnetToClient:

```
ClientToTelnet cs = new cts(sinputstream, soutputstream);
```

```
TelnetToClient sc = new stc(sinputstream, coutputstream);
```

```
Thread threadcs = new Thread(cs);
```

```
Thread threadsc = new Thread(sc);
```

```
threadcs.start();
```

```
threadsc.start();
```

其中, 代理发送线程 ClientToTelnet 由 clientin 的输入流对象和自制代理创建的 Socket 的输出流对象组成。当 clientin 的输入流对象接收到来自客户端的指令信息时,

Socket 的输出流对象就将这些指令发送到作为第三方的 Telnet 服务器; 相反代理接收线程 TelnetToClient 是由 clientin 的输出流对象和由自制代理服务器创建的 Socket 的输入流对象组成, 当 Socket 的输入流对象接收到来自 Telnet 服务器的响应信息后, 再由 clientin 的输出流对象将这些响应信息返回给客户端显示。

3 结束语

Java 的安全模型很大程度上解决了网络安全问题, 但也伴随着 Java 在开发网络应用程序时的诸多不便。在某些应用领域确实可以通过一些手段来绕过 Java 的安全模型的限制, 但是其性能优越度可能有一定程度的降低。随着技术的不断发展, 这个问题一定会得到更好的解决。

参考文献:

- [1] Garms J, Somerfield D. Java 安全性编程指南[M]. 庞南等译. 北京: 电子工业出版社, 2001.
- [2] Oaks S. Java 安全[M]. 林琪译. 北京: 中国电力出版社, 2002.
- [3] Jaworski J. Java Security Handbook[M]. 邱仲潘等译. 北京: 电子工业出版社, 2001.
- [4] Horstmann C S, Cornell G. Java2 核心技术[M]. 李如豹等译. 北京: 机械工业出版社, 2001.
- [5] 徐迎晓. Java 安全性编程实例[M]. 北京: 清华大学出版社, 2003.

(上接第 139 页)

通过上述的各种扩展, workflow 系统使用 Wf-XML 协议在 Web Services 方式下的通信过程更加清晰, 语义更加明确, 如图 5 所示^[5]。

Wf-XML 协议针对 workflow 系统丰富了资源类型, 扩展了表示动作类型的消息体元素, 利用与 ASAP 协议类似的通信过程, 基本满足 workflow 系统在 Web Services 方式下的通信要求。

4 总结

当然, 由于 Wf-XML 协议目前还处于草案阶段, 仍有可改进之处。例如, workflow 系统中业务过程的运行必然会涉及大量的数据交换, Wf-XML 协议强调通信双方的逻辑交互过程, 对于涉及的数据没有针对 workflow 应用进行特别的定义, 只是利用 XLink, Web 服务寻址 (WS-Addressing) 技术指定相关的数据引用。是否可以对流程的一般性数据直接在 SOAP 消息中传递, 这个问题值得商榷。

不过, Wf-XML 协议已有简单的应用: 现有几种业务过程定义语言, 例如 WfMC 的 XPDL (XML Process Definition Language), OASIS 的 WS-BPEL (Web Services Business Process Execution Language), 只是用于定义流程,

没有说明如何将流程定义载入到 workflow 引擎中。把过程定义工具视为 workflow 系统中的第三方应用程序, 利用 Wf-XML 协议与 workflow 引擎通讯, 就可以使上述流程定义通过 Web Services 方式载入到 workflow 引擎中, 还可以修改流程定义。目前 WfMC 组织已经完成了几次全球范围的 Wf-XML 协议测试演示, 相信 WfMC 组织不久就会将目前的草案标准化, 届时 Wf-XML 协议将会得到广泛的应用。

参考文献:

- [1] WfMC. The Workflow Reference Model[S]. Workflow Management Coalition Specification WFMC - TC00 - 1003, 1995.
- [2] Graham S. Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI[M]. 刘晓晖等译. 北京: 机械工业出版社, 2003.
- [3] Snell J, Tidwell D. Programming Web Services with SOAP[M]. 胡军译. 北京: 中国电力出版社, 2002.
- [4] OASIS. Asynchronous Service Access Protocol Version 1.0[S]. wd-asap-spec-01, 2004.
- [5] Swenson K D, Pradhan S, Gilger M D. Wf-XML 2.0 XML Based Protocol for Run-Time Integration of Process Engines[Z]. Workflow Management Coalition Draft, 2004.