文章编号:1001-9081(2001)12-0039-03

HTTP 和 FTP 代理服务器中的透明代理与认证机制

代炼忠,赵东升,周士波
(军事医学科学院 情报研究所,北京 100850)

摘 要:对 HTTP 和 FTP 代理服务器中透明代理和身份认证的机制进行了比较,并对实际应用中可能出现的问题进行了具体的分析。

关键词:HTTP;FTP;代理服务器;透明代理;身份认证
中图分类号:TP393.07;TP393.08 **文献标识码**:A

1 HTTP 代理服务器中的透明代理机制

透明代理是指用户通过代理服务器访问 Internet 资源却感觉不到代理服务器的存在,其好处是为管理员和用户提供管理和使用的方便。

在 IE 和 Netscape 中代理设置有两种方式:手工设置和自动设置。前者需要填写代理服务器的 IP 地址和端口,后者需要填写一个 URL 地址,该地址指向一个关于代理服务器配置的 Javascript 脚本。相对来说自动设置的方式更好一些,因为一旦设置好以后,在网络中代理服务器的配置可以任意改变,网管人员只需修改相应的 Javascript 脚本,无需用户作任何改动。但无论如何,用户都必须事先对浏览器进行适当的设置。

而当代理服务器具有透明代理的功能时,用户可以通过代理服务器访问 Web 资源却无需在浏览器中进行代理配置。透明代理对网络管理的好处是显而易见的,网络用户经常因为各种原因需要重新配置浏览器,因为用户不是网络专家,他们往往不知道或者早已忘了该如何配置,唯一的办法就是打电话给网管人员,这对双方来说都是一个负担,透明代理无疑可以解决这个问题。

透明代理实现的关键是如何截获 HTTP 连接请求,具体的实现大致如下:

1) 浏览器向某个 WWW 服务器,如 www.example.com 发送 HTTP 连接请求;

2) 网关设备将 HTTP 连接请求转发给代理服务器;

3) 代理服务器向浏览器发送响应,假装它就是 www.example.com;

4) 浏览器收到响应,相信它已经和 www.example.com 连接上。

透明代理有以下三种网络配置方式。

1.1 代理服务器作为缺省网关

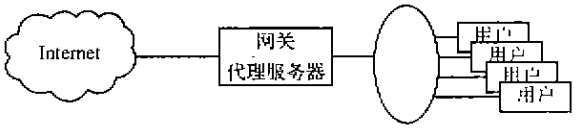

图 1

在这种方式中,代理服务器安装在网关设备上,用户在网络设置中需要把该设备的地址设为自己的缺省网关。因为用户访问 Internet 的所有 TCP/IP 通信都必须通过该设备,所以只需在该设备中进行适当的配置,将所有 80 端口的请求交给代理服务器处理即可。这种方式的缺点是明显的,一方面是适合于代理服务的配置对于一个网关设备来说运行的效率不高,另一方面,由于代理服务器位于网络的关键位置上,如果它出现故障可能带来整个网络的崩溃。其结构如图 1 所示。

1.2 基于交换的透明代理

在这种方式中,一个四层(L4)交换设备将它收到的 80 端口的 TCP/IP 包转发给代理服务器,而允许其它类型的包直接通过。其结构如图 2 所示。

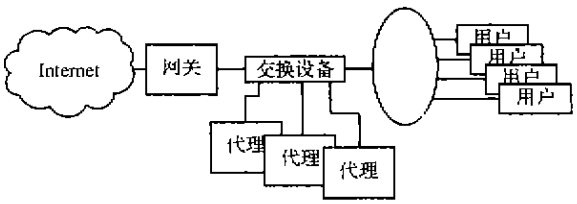

图 2

1.3 基于路由的透明代理

配置和上一种方式类似,只是交换设备换成路由设备。其结构如图 3 所示。

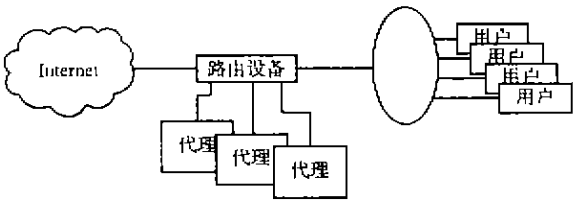

图 3

2 FTP 代理服务器的透明代理机制

在 FWTK 和 Gauntlet Internet Firewall 的 FTP 代理实现中,在非透明代理的情况下,假如我们准备以用户 abc 的身份访问 ftp.example.com,而代理服务器的名称是 proxy.mynet.com,需要经过以下步骤:

1) 键入命令 ftp proxy.mynet.com;

2) 在提示符 username 下键入 abc@ftp.example.com;

3) 在提示符 password 下键入用户 abc 在 ftp.example.com 上的口令。

如果我们使用别的 FTP 客户软件,则需要在客户软件中进行适当的代理服务器设置,很明显,非透明的 FTP 代理服务器使用起来不太方便。

收稿日期:2001-06-28

作者简介:代炼忠(1969-),男,四川荣县人,助理研究员,硕士,主要研究方向:计算机网络; 赵东升(1970-),男,河北赵县人,副研究员,博士,主要研究方向:计算机网络; 周士波(1944-),男,浙江奉化人,研究员,硕士,主要研究方向:计算机网络应用.

FTP 透明代理与 HTTP 透明代理的实现机制在本质上是一样的，只是端口号不同而已，在网络结构中的配置方式也与 HTTP 透明代理类似。

3 HTTP 代理服务器中的身份认证机制

在 HTTP/1.1 中明确规定了两种认证：用户在 WWW 服务器上的身份认证和代理服务器上的身份认证。

前者是一种端到端(end - to - end)的认证，就是说，用户的认证是在用户的主机和 WWW 服务器之间进行的，如果用户和 WWW 服务器之间有代理服务器之类的中间介质存在，中间介质必须转发用于认证的信息。而后者是一种相邻站点间(hop - to - hop)的认证，包括用户到代理服务器和代理服务器到代理服务器的认证。

我们来看看典型的认证过程：

用户在 WWW 服务器上的认证

1) 用户向 WWW 服务器发送 GET 请求；

2) 该请求到达 WWW 服务器，中间可能经过中间介质的转发；

3) WWW 服务器返回 401 Unauthorized 的响应；

4) 该响应到达用户所用计算机；

5) 用户重新发送包含认证信息的 GET 请求；

6) 该请求到达 WWW 服务器；

7) 认证通过；

8) WWW 服务器向用户发送包含用户请求的对象的响应。

用户在代理服务器上的认证

1) 用户向代理服务器发送 GET 请求；

2) 该请求到达代理服务器；

3) 代理服务器返回 407 Proxy Authentication Required 的响应；

4) 该响应到达用户所用的计算机；

5) 用户重新发送包含认证信息的 GET 请求；

6) 该请求到达代理服务器；

7) 认证通过；

8) 代理服务器从 WWW 服务器或缓存中取用户请求的对象返回给用户。

可以看出，两种认证的过程与机制十分相似，我们重点看一下 401 和 407 响应中与认证有关的信息，以及用户收到 401 和 407 响应后重新发送的请求和原先发送的请求有何不同。

401 响应中包含一个 WWW - Authenticate 报头域，该域包含一个应用于请求的资源域的 challenge，收到 401 响应的用户重发的请求中包含一个 Authorization 报头域，该域包含证明用户身份的证书，如果认证通过，则用户被授权可以访问在 WWW - Authenticate 中规定的资源域，以后用户对该资源域的访问请求都带着一个包含证书的 Authorization 报头域。

407 响应中包含一个 Proxy - Authenticate 报头域，该域包含一个应用于代理服务器认证的 challenge，收到 407 响应的用户重发的请求中包含一个 Proxy - Authorization 的报头域，该域包含证明用户身份的证书，如果认证通过，则用户被授权可以使用该代理服务器访问 Internet 资源，以后用户向该代理服务器发送的请求都带着一个包含证书的 Proxy - Authorization 报头域。

不管是在 WWW 服务器还是在代理服务器上的认证，都有两种不同的认证方案，这在 RFC2617 有详细的描述。一种是基本认证(Basic Authentication)，另一种是摘要认证(Digest Authentication)。前者要求证明用户身份的证书包含用户的用户名和口令的 base64 代码，而后者要求这个证书包含用户名、口令、一个给定的当前值(nonce value)、HTTP 方法和请求的 URI 的一个校验和(缺省情况是 MD5 校验和)。由于前者的用户名和口令仅仅使用 base64 编码传送，对一个拥有 base64 的解码器的偷听者来说，无异于明码传送，因此极易受到重放进攻。而摘要认证基本上可以避免这样的进攻，但也不是一个对 HTTP 协议中安全性问题的完整的解决方案。更安全的方案有 TLS(Transport Layer Security)，S - HTTP(Secure HyperText Transfer Protocol)等，但是对于安全性要求并非很高的应用来说，HTTP/1.1 提供了一个易于实现的解决方案。实际上，在 IE 和 Netscape 的现在的浏览器版本中，根本就没有完全遵从 HTTP/1.1 来实现基于摘要的认证机制，因此在现在的大部分 WWW 服务器和代理服务器中也仅仅只有基本认证的功能。

很遗憾的是，在实际的应用中，透明代理和用户在代理服务器上的身份认证这两个功能无法在 HTTP 代理服务器上同时配置。当需要进行用户身份认证的代理服务器收到用户请求时，它向用户返回 407 Proxy Authentication Required 的响应，而在透明代理的情况下，浏览器并不知道它正在使用透明代理，所以它并没有期待接收一个 407 响应，结果是程序发生异常错误而退出，在 IE 和 Netscape 中均是如此。出现这种现象的原因可能在于 IE 和 Netscape 的程序设计中根本就没有考虑透明代理的问题，只是简单地认为，只要在浏览器没有对代理服务器的选项进行设置，就是不使用代理服务器，因此也就把收到 407 响应作为一个异常错误处理。如果这样的猜测属实的话，我们可以期待在 IE 和 Netscape 的后续版本中可以修正这样的错误(如果这算是一个错误的话)，或者加上这样的功能，因为从 HTTP/1.1 的协议描述中，实在找不出透明代理和认证冲突的任何理由。

4 FTP 代理服务器中的身份认证机制

在 FTP 代理服务器上进行身份认证以一种很奇怪的方式实现。我们知道，在一次典型的 FTP 会话中，客户端和服务器端需要建立两条 TCP 连接，一条用于访问控制，一条用于数据传输，在控制连接上传送的是客户端向服务器端发送的命令和服务器端向客户端发送的响应。当建立控制连接的任务完成后，首先要在服务器上进行身份认证，这是通过客户端向服务器端发送 USER 和 PASS 命令以及服务器端作出相应的应答完成。在实际的应用环境中，例如我们在一台 Linux 的主机上使用 FTP 的客户程序 ftp 从某个 FTP 服务器上下载文件，当我们在 Linux 的终端上键入 ftp ftp.example.com 的命令后，如果连接成功，系统在输出一些提示信息之后弹出提示符 Username:，然后等待用户输入，之后用户的输入作为 FTP 协议的 USER 命令的参数部分传送到服务器端，同样用户在 Password:提示符下的输入作为 PASS 命令的参数部分传送到服务器。如果会话是通过一个 FTP 代理服务器进行的，则 FTP 代理服务器需要对命令和响应进行转发，如果代理服务器是以透明方式运行的，用户甚至都感觉不到代理服务器的存在。当代理服务器需要进行身份认证时，用户第一次输入的用户名和口令是用于代理服务器上的身份认证，当认证通过后，系统在 ftp> 的提示符下等待用户的输入，这时候只有用户输入 user 命令，系统才会弹出 Username:提示符，接下来用户的输入作为认证信息被代理服务器转发给 FTP 服务器进行认证。在用户登录到 FTP 服务器的过程中，用户需要两次输入用户名和口令，其间还要用户自己输入 user 命令，这可能

使一个没有经验的用户感到很困惑,但这是不得已的事情,在现有的 FTP 协议及应用实现中这可能是唯一的解决办法。

在以透明方式运行的 FTP 代理服务器上可以进行身份认证。之所以无法在 HTTP 透明代理服务器上进行身份认证,除了我们前面提到的浏览器的设计问题外,还有一点与这个事实有关,就是在我们用浏览器访问一个网页时,由于一个网页中往往包含很多的 Web 对象,因此用户端和服务器端往往要建立多个 TCP 连接分别传送不同的 Web 对象,在一次 HTTP 会话中建立多个 TCP 连接,这是 HTTP 协议的特点。而在一次 FTP 会话中,控制连接是唯一的,因此容易对该连接传送的信息进行全程跟踪,在写代理服务器的源代码时,我们很容易知道第一次到达的用户名和口令用于代理服务器上的身份认证,而第二次到达的用户名和口令才用于 FTP 服务器的身份认证　如果硬要在 HTTP 透明代理服务器上实施身份认证,还有一个办法就是在代理服务器收到未经认证的请求时,不返回 407 响应,而是返回 401 响应"欺骗"浏览器,这在 WWW 服务器不需要进行身份认证的情况下行得通,但在 WWW 服务器需要进行身份认证的情况下,由于 HTTP 这种多连接的特性,代理服务器就无法区分用户送来的认证信息是用于代理服务器认证还是 WWW 服务器认证。

FTP 透明代理 + 身份认证的模式不适合在浏览器中使用。FTP 在浏览器中使用按照这样顺序进行:建立连接,浏览器和服务器之间进行通讯,关闭连接。这些动作在用户自浏览器中点击一个链接或在地址栏中输入一个 URL 地址并按回车键后就全部由浏览器自动完成,用户根本就没有机会对浏览器的行为进行干预,因此也就没有机会两次输入用户名和口令。

5　结语

从上面的讨论我们可以看出,不管是 HTTP 还是 FTP 代理服务器,透明特性都可以为管理员和用户带来管理和使用上的方便;然而,一旦我们因安全方面的需求,或需要按用户进行计费而必须在代理服务器上进行身份认证时,由于在 HTTP 代理服务器中透明代理和身份认证的特性互不相容,因此只能放弃透明代理的特性;在 FTP 代理服务器中进行身份认证以一种很奇怪的方式实现,无法在浏览器中通过需要进行身份认证的 FTP 代理访问 FTP 服务器,这是实施身份认证的代价。

在实际的网络配置中,两种极端的情形是:

· 透明的 HTTP 和 FTP 代理服务器,不要用户认证。

· 需要进行身份认证的 HTTP 代理服务器、FTP 访问只能通过 HTTP 代理服务器进行。

前者最方便,后者最安全,在实际的应用中,究竟如何配置应视具体需求而定。

参考文献

[1] RFC2616, Hypertext Transfer Protocol – HTTP/1.1[S], June 1999.

[2] Rfc959, File Transfer Protocol(FTP)[S], October 1985.

[3] RFC2617, HTTP Authentication: Basic and Digest Access Authentication [S], June 1999.

[4] Bert Williams, Transparent Web Caching Solutions [EB/OL], http://www.cache.ja.net/events/workshop/33/cachpaper.html, April2001.

[5] Peter Danzig, Karl L. Swartz. Transparent, Scalable, Fail-Safe Web Caching [EB/OL]. http://www.netapp.com/tech_library/3033.html, April 2001.

(上接第 38 页)

1) 要求管理员在管理 MIS 用户是只用 MIS 的用户管理界面而不要用 Windows NT 的用户管理器来管理 MISUsers 组;

2) 编程实现 Windows NT 的用户和 MIS 用户同步。

表 3　业务定义

字段名称	数据类型	字段描述说明
IdentityId	Int	自动编号,业务内部标识号
Name	Varchar(40)	业务名称
Create_Date	Datatime	创建日期
Delete_Date	Udf_Delete_Date	删除日期

表 4　用户与管理的业务关系

字段名称	数据类型	字段描述说明
UserIdentityId	Int	用户标识号,以用户定义表(表 2)中的 IdentityId 字段为外部键的主键
BusIdentityId	Int	业务管理员所能管理的业务标识号,以业务定义表中的 IdentityId 字段为外部键的主键

表 5　用户与拥有的录入界面关系

字段名称	数据类型	字段描述说明
UserIdentityId	Int	用户标识号,以用户定义表中(表 2)的 IdentityId 字段为外部键的主键
InpIdentityId	Int	用户所拥有的录入界面标识号,以输入界面定义表中的 IdentityId 字段为外部键的主键

表 6　用户与拥有的查询界面关系

字段名称	数据类型	字段描述说明
UserIdentityId	Int	用户标识号,以用户定义表中(表 2)的 IdentityId 字段为外部键的主键
qurIdentityId	Int	用户所拥有的查询界面标识号,以查询界面定义表中的 IdentityId 字段为外部键的主键

6　结束语

本文主要解决广州市经济技术开发区信息管理系统的用户管理模块的设计,它阐述了如何把操作系统用户和 MIS 用户集成管理的新方案,用户认证安全可靠,同时保留了较大的开发灵活性,为 MIS 用户管理提供了新的思路。

参考文献

[1] Philip J. Pratt, Joseph J. Adamski. The Concepts of Database Managemen [M]. 北京: 机械工业出版社, 1999. 67 – 143.

[2] Leonid Braginski, Matthew well. Microsoft Internet Information Server 4.0 使用大全[M]. 北京:北京华中兴业科技发展有限责任公司,译. 北京:人民邮电出版社, 1999. 76 – 189.

[3] Don Benage, Azam Mirza, 等. 应用 Visual Studio 6.0 构建企业解决方案[M]. 潇湘工作室,译. 北京:人民邮电出版社, 1999. 88 – 143.

[4] Sue Plumley. Windows NT 网络规划与管理[M]. 杨武杰,译. 北京:机械工业出版社, 1998. 77 – 96.