

第4章 防火墙面临的挑战：基础Web

本章根据超文本传输协议（HTTP，HyperText Transmission Protocol）讨论防火墙实现所面临的挑战及一些安全问题。同时讨论了 HTTP 的代理特性及其安全性。探索了安全 HTTP（S-HTTP）以及使用 SSL 以增强安全性，回顾了通用网关接口（CGI，Common Gateway Interface）隐含的安全含义。

4.1 HTTP

HTTP 作为一个为分布、协作、超媒体信息系统而开发的应用层协议，是一个通用、无状态协议，由它建立的系统与传输的数据无关。它还是一个面向对象的协议，能用于各种任务，包括但不局限于名服务器、分布式对象管理系统及它的请求方法或命令的扩展。

HTTP 的一个主要特征是数据描述的键入和协商。这个协议从 1990 年开始使用，具有 W3 全球信息主动权。

目前最新的 HTTP 版本为 1.0，市场上所有的 Web 服务器均支持该版本。该协议还有下一代，HTTP-NG，它许诺将更有效地使用可利用的带宽并将增强 HTTP 协议。

HTTP 协议一般用于用户代理和其他因特网协议代理或网关之间的通信，如 SMTP、NNTP、FTP、Gopher 和 WAIS。

尽管如此，HTTP 提供广泛的适应性是有代价的：它使得保障 Web 服务器和客户的安全非常困难。由于 Web 的开放性和无状态特性使得 Web 迅速取得成功，但这使得它难以控制和保护。

在因特网上，HTTP 通信一般发生在 TCP / IP 连接。它默认使用 80 端口，也可以使用其他端口，即 HTTP 可以在任何其他协议上实现。事实上，HTTP 能使用任何可靠的传输。

当一个浏览器接收到一个不能识别的数据类型时，它依靠一些辅助应用程序将其转变为它能识别的形式。这些应用程序通常称为浏览程序（viewer），是为维持安全特性应该首先关注的事项之一。当安装这些程序时，必须小心，因为运行在服务器上的基本 HTTP 协议不会制止浏览程序执行危险命令。

应该特别小心地使用代理和网关应用程序。当转发的请求 HTTP 不能理解时，就必须警惕了。必须考虑其所使用的 HTTP 版本，因为该协议版本标明了发送者的协议能力。一个代理或网关是不能发送比其自身版本指示符更高的报文；否则，如果收到高于其版本的请求，代理或网关必须要么降低请求的版本，发出错误响应，要么转变成隧道方式。

注意 如果你需要更多的关于 HTTP 的信息，请访问站点

<http://www.w3.org/hypertext/WWW/protocols/>

网站：<ftp://src.brunel.ac.uk/WWW/managers/> 提供了许多针对 Web 服务器管理员的实用程序。

HTTP 客户端，如 Purveyor (<http://www.process.com>) 和 Netscape Navigator 支持多种代理策略、SOCKS 和透明代理。

例如Purveyor，通过限制LAN用户的因特网行为的方式建立一个安全 LAN环境，不但提供HTTP代理，而且支持FTP和GOPHER协议代理。代理服务器通过允许内部代理缓存的方式改进其性能。Purveyor也为有多个代理服务器的公司提供代理到代理之间的支持。

如果你的Web服务器运行在 Windows NT、Windows95或NetWare的环境下，你能使用Purveyor Webserver的代理特性来增强安全性。另外，还能提高服务器性能，因为 Purveyor能在本地缓存从因特网获取的Web网页。

在你的站点上必须安装防火墙。无论如何，如果你的服务器放在保密网的内部或外部，防火墙能够阻止大多数攻击，但不是全部。 HTTP过大的开放性使你不能去冒险。除此之外，仍需担心所有的浏览程序和其他 applet。

当选择防火墙时，确保选择包含 HTTP代理服务器的防火墙。附录 A “ 防火墙销售商和产品目录”，对所有主要防火墙供应商和他们的产品作了一个完整的回顾。同样请查看一下本书附带的CD，附录A中列出的许多供应商提供他们产品的样品和评估报告，这些很值得进行测试。

防火墙将极大地帮助你保护浏览器。一些防火墙，如 TIS FWTK，为用户提供完全透明的HTTP代理。你将在第7章“究竟什么是因特网/内部网防火墙”中了解更多关于防火墙的知识。从现在起，必须意识到防火墙在处理 Web安全需求和HTTP协议时面临的挑战。

4.1.1 基础Web

你知道当一个用户连接到你的站点时会发生什么吗？如果你不知道他们是如何进入的，也就不知道如何去关上这扇大门。

如果你的站点有一个Web服务器，每次其他用户与它建立连接时，其客户将其机器的 IP地址（数值）传送到你的Web服务器。如果干得比较精明的话，Web服务器接收到的IP地址甚至不是客户的地址，而是他的请求所经过的代理服务器的地址。这样，服务器看到的将是代表客户请求文档的代理服务器的地址。但是，由于使用了 HTTP协议，进行此次请求的用户也能向Web服务器公开登录在该客户机上的用户名。

除非已经设置服务器去捕捉这样的信息，否则，服务器要得到客户的域名（例如www.MGConsulting.com），首先要做IP地址反向转换。但服务器要得到这个域名，必须先与域名服务器联系并向它提供要转换的IP地址。

很多情况下，IP地址因为设置不正确，而不能被转换。因此，服务器不能转换这个地址。接下来发生什么？服务器继续运行并且伪造这个地址！

一旦Web服务器有了客户的IP地址和可能的域名，它开始应用一系列认证规则，决定客户对所请求的文档是否有访问权。

你注意到安全漏洞了吗？这样的处理方法存在以下几个安全漏洞：

由于服务器伪造域名，使请求信息的客户永远不会得到它所请求的信息。现在这个客户可能没有被授权获取其请求的信息。

由于域名被伪造，服务器可能会将信息发往另一个客户。

更糟的是，服务器可能允许一个入侵者访问，认为它是一个合法用户！

这里的风险来自两种方式：

请关注HTTP服务器，它能给客户带来什么风险和伤害。

请关注HTTP客户，它能给服务器带来什么风险和伤害。

像前面讨论的那样，就客户对服务器的威胁而言，你应该关心服务器的安全性。应该确保客户仅仅得到它们期望获得的访问服务，并且如果有一个恶意攻击，服务器会有一些方法去保护它自己。

然而，不是没有一点办法，这里有一些可以遵循的基本步骤来增强服务器的安全性：

一定要仔细配置服务器并使用它的访问和安全特性。

也可以一个非特权用户的方式运行 Web 服务器。

如果服务器正运行在 Windows NT 环境下，确保对驱动程序及共享资源权限进行检验，并且将系统及受限区域设置为只读。或者使用 chroot 去限制对系统区域的访问。

可将服务器映射到主系统上，并将敏感文件放在主系统上，并有一个辅助系统，没有任何敏感数据暴露在因特网上。

请记住 Murphy 定律：可能出错的就一定会出错。往最坏的情况想，并且用这样的方式配置 Web 服务器，即如果一个黑客想要控制 Web 服务器，他必须穿过一堵巨大的墙（如果不是防火墙）。

更重要的是，请再看看 HTTP 服务器使用的 applet 和脚本程序，尤其是那些在因特网上同客户进行交互的 CGI 脚本。要关注外部用户触发执行内部命令的可能性。

在 Windows NT 服务器上运行 Web 服务器。这是比较安全的，尽管它没有像 UNIX 和 SUN 服务器那样多的特性。

Macintosh Web 服务器甚至更安全，但与 Windows NT 和 Windows 95 平台相比，缺乏实现特色。

为解释一个错误配置的域名能对反向 IP 地址转换的过程有何影响，请考虑在 access.conf 文件中输入的项。记住这个文件是对服务器内文件的访问控制负责的。

当建立这个文件时，必须为每个被控制的目录向 access.conf 文件输入 < directory > 标记，在 < directory > 标记内，也必须用一个 < limit > 标记，该标记带有控制对目录访问所必须的参数（allow、deny 和 order）。

下面是一个例子，这里整个网络都能访问在顶级文档目录中的文件。

```
<directory /usr/local/http/docs>
  <limit>
    order allow,deny
    allow from all
  </limit>
</directory>
```

这里关键的一行是“order”指令，告诉服务器先处理“allow”指令（来自所有客户），后处理“deny”指令。注意到我们没有任何“deny”指令吗？

现在让我们假设你必须限制服务器上的一个区域仅让内部用户访问它。与前面的例子不同，你需要一个“deny”指令。

```
<directory /usr/local/http/docscorp>
  <limit>
    order deny,allow
    deny from all
    allow from .greatplace.com
```

```
</limit>
</directory>
```

在这个例子中，“deny”指令出现在“allow”指令之前，以至于整个网络能对该公司进行有限制的访问，“allow”指令允许来自greatplace.com域的任何访问。

如果服务器不能转换客户的IP地址，就有问题了，因为这个域名对该处理过程起决定作用。简单说，用户将不能够访问Web网页。

有一个补救解决方法。可将原始IP地址数值添加到访问表。

```
<directory /usr/local/http/docscorp>
  <limit>
    order deny, allow
    deny from all
    allow from .greatplace.com 198.155.25
  </limit>
</directory>
```

在这种方式下，指令“allow”允许来自“greatplace”的任何访问，而且允许来自IP地址以198.155.25开始的机器的访问。

4.1.2 怎样监视HTTP协议

HTTP协议存在许多的安全漏洞，因此需要安装防火墙。这些安全漏洞之一就是允许远程用户向远程服务器请求连接，并且执行远程命令。这个安全漏洞可以以许多方式损坏Web服务器和客户机，包括但不限于如下这些：

- 对远程请求的武断认证。
- 对Web服务器的武断认证。
- 破坏请求和应答的保密性。
- 滥用服务器功能和资源。
- 通过搜索它的程序错误和安全漏洞，滥用服务器。
- 滥用日志信息(提取IP地址、域名和文件名等等)。

许多这样的安全漏洞是众所周知的，一些应用程序，如Netscape的SSL和NCSA的S-HTTP XE“S-HTTP”试图解决这些问题，但仅仅是其中的一部分问题。

问题是，在因特网上，Web服务器对客户的行为显得很脆弱。因此我建议，在允许HTTP访问不是为它(为其他协议)保留的端口之前，强迫Web客户机提醒该用户。否则，这会引起用户轻率地与其他有危险的协议进行事务处理。

看看GET和HEAD方法！单击一个定位来签署或应答一个服务，如此小的一个连接能在用户不知道的情况下触发一个applet运行。这能被恶意用户滥用。

HTTP的另一个安全漏洞来自于服务器日志。通常Web服务器的日志含有大量私人数据，是关于被不同用户请求的信息。显然，这些信息应该保密。HTTP允许在没有任何访问许可方案的情况下获得这些信息。

作为一个新特性，“Referer:”域增大了私有数据传输的数量。该域允许对读模式进行分析甚至允许取消反向连接。如果落在坏人手中，它会成为导致保密信息被滥用和破坏的非常有用和功能强大的工具。直到今天，人们还不知道仍有许多抑制Referer信息的情况，开发者

正从事于寻求解决方法。

即使我们解决该协议存在的以上诸方面的安全问题，许多其他的 HTTP局限性和安全漏洞仍然存在，安全HTTP技术和策略试图去定位和解决这些安全漏洞。

4.1.3 利用S-HTTP

安全超文本传输协议（S-HTTP）被开发用来填补在保护因特网上传输的敏感信息时存在的安全漏洞。随着在因特网和 Web之间对认证需要的增长，用户在相互发送加密文件之前必须进行认证。

由于S-HTTP的交易安全特性会促进自发的商务交易，因而它将促进电子商务的成长。因为S-HTTP可以保护Web客户和服务器的，所以在它们之间交换的信息将是安全的。

使用S-HTTP的安全服务器能用加密和签名的消息应答请求。通过使用相同的标记，安全客户机能验证消息的签名并对它进行认证。认证使用服务器的私有密码，该私有密码用于产生服务器的数字签名。当消息被发往客户时，服务器已同签名的消息一起递送了它的公开密码证书，以使客户能鉴定数字签名。服务器通过相同的过程向内解密来自客户的消息和向外加密发往客户的消息，能鉴定由客户发送的经过数字签名的消息的完整性。

可以使用共享的、私有的或公共的密钥加密数据。如果使用公开密码加密数据，那么就可以进行双向报文交换并且对报文解密，不需要客户的公开密码，由于服务器提供了一个单一的服务器私有密码，它被装在密钥数据库文件当中，用 Web管理员口令进行了加密。

通过CGI脚本可以控制加密和签名过程。本地安全配置文件和用 CGI脚本编写S-HTTP报文报头，将决定服务器是否签名、加密、两者都有或两者都没有。

遗憾的是，S-HTTP仅仅工作在SunOS 4.1.3,Solaris 2.4,Irix 5.2,HP-UX 9.03,DEC OSF/1和AIX 3.2.4。

4.1.4 使用SSL增强安全性

Netscape Communications设计和提议了安全套接层（SSL，Secure Socket Layer）协议，其目标是改善应用层协议（如HTTP、Telnet、NNTP、FTP）和TCP / IP之间数据安全性。

SSL以数据加密、服务器认证、消息完整性和为 TCP / IP连接进行可选的客户认证为特色。

SSL是一个开放的、非专利性协议，Netscape把它作为Web浏览器和服务器的一个标准安全方法提交给万维网协会（W3C）去考虑。它还作为因特网草案发送给因特网工程部（IETF），追求在IETF框架内使SSL标准化。

SSL的主要目标是提高两个应用程序之间通信的保密性和可靠性。它的最新版本是1996年3月发行的3.0版，替代了1995年10月的上个版本。

这个协议的基础仍然没有改变。它是一个两层协议，依赖于最低层一些可靠传输协议，如HTTP协议。较低的那层称为SSL记录协议（SSL Record Protocol），用于封装各种较高层的协议。一个例子是SSL握手协议（SSL Handshake Protocol），它允许服务器和客户互相认证，同时在进行任何传输之前协商一个加密算法和加密密钥。

连接是私有的，使用非对称密钥或公开密码对连接两端的身份进行认证，并且连接是可靠的：这些是SSL的三个基本属性。

SSL和S-HTTP之间主要的不同之处是，S-HTTP作为Web的 HTTP的超集专门针对 Web应用。然而，SSL协议通过套接字发送消息。SSL的完整概念能概括为在任何客户和使用套接字层的服务器之间进行安全传输的协议，它包括几乎所有 TCP / IP应用程序。

就进行加密而言，SSL和S-HTTP都能协商不同类型的加密算法和密钥认证策略，但Netscape和EIT（Enterprise Integration Technology）都已经获得RSA Data Security的工具包许可证，以提供消息的端到端加密及密钥生成和认证。

遗憾的是，今后的电子商务和安全 Web事务处理不能依靠一个多协议安全系统。S-HTTP和SSL既不是相同协议，工作方式也不同。幸运的是，Web Consortium正致力于开发一个包括SSL和S-HTTP的统统一的安全策略。

而且，不只建议了这些方案，还有其他的建议。EINet的Secure Server使用Kerberos和一些其他机制，它和Shen氏建议都提供比SSL或S-HTTP更全面的安全特性，如扩展地使用Privacy-Enhanced Mail XE。

4.1.5 小心Web缓存

随着越来越多的用户利用这个巨大的信息资源和越来越多的应用程序能以某种方式运行在Web上，Web通信量比社团网络上的几乎任何其他类型数据通信量都增长的快。对于前几代数据处理技术，数据通信量可预测和控制一定范围，与此不同的是，Web模式用连续的鼠标点击，经常产生完全随机和不可预测的数据模型，在访问隔壁服务器上的数据后接着访问世界另一边服务器上的数据。

通过确保经常被请求的文件尽可能存贮在本地缓存，缓存技术能显著提高Web服务器性能。然而，如果在远程服务器上的文件被修改，那么用户从缓存中取回的可能是过时的文件。

同样，这些被缓存的文件能被远程用户取走，暴露了不能被公共用户和外部用户读取的信息。

通过检查远程服务器上文件的日期并且把它与缓存中文件的日期进行比较，HTTPD服务器能解决这类问题。下面记录一个典型的缓存日志文件，它提供了域名和机器的名称。

```
xyz_pc77.leeds.ac.uk - - [21/Nov/1994:00:43:35 +0000] "GET
http://white.nosc.mil/gif_images/NM_Sunrise_s.gif HTTP/1.0"
200 18673
xyz_pc77.leeds.ac.uk - - [21/Nov/1994:00:43:38 +0000] "GET
http://white.nosc.mil/gif_images/glacier_s.gif HTTP/1.0"
200 6474
xyz_pc77.leeds.ac.uk - - [21/Nov/1994:00:43:40 +0000] "GET
http://white.nosc.mil/gif_images/rainier_s.gif HTTP/1.0"
200 18749
```

在将来，可能会把缓存链接起来。从长远来看，将开始考虑把包括公共机构的、都市的、国家的、大陆范围在内的缓存链接起来的可能性。

4.1.6 堵住漏洞：一个配置检查表

有一个配置检查表可以有助于堵住漏洞：

当配置HTTP服务器时，不要使用原始IP地址去访问你的网页，否则在访问中将最终会有一串原IP地址，这将只能使维持起来更加困难。

在任何时候，如果遇到客户错误配置域名服务器的问题，那么让其联系LAN或系统管

理员去修改它，以使你能正确地反向解析它们的域名。如果你修改这个问题，花点时间做好它！从长远的观点看，你将看到这样做是很有用的，因为不这样做，最终会在你的清单中有一个巨大的原始IP地址清单。

如果必须处理access.conf文件，那么确保每条指令仅仅放一个名字，这将易于文件的编辑，当你要注释某指令时，只需简单地在这行的开始放一个“#”字符。

在修改access.conf后，记着重启服务器，因为所做的任何改动直到重启系统后才会起作用。

总有一个顶层文档目录的访问控制表。这在以后更新这个文件时是有用的。

4.1.7 安全检查表

首先，你能有的最好的安全检查表是知道检查什么和什么时候检查。下面是因特网上的一个资源清单，它帮助你及时了解网络世界每天出现的安全问题。也能得到一些免费资源来帮助你增强站点的安全性。

订购安全邮件列表。

向CERT(Computer Emergency Response)建议的邮件列表发送电子邮件。这要求你在cert@cert.org中包含有他们的邮件列表。

试试Phrack时事通讯，这是一个不公开的黑客的时事通讯。向phrack@well.sf.ca.us发送电子邮件。

也可试试Computer Underground Digest。向tk0jut2@mvs.cso.niu.edu发送电子邮件。

4.1.8 Novell的HTTP协议：小心为妙

我们都知道Novell的HTTP协议有一个非常不安全的CGI脚本。如果正运行一个Novell Web服务器，那么应该使它本身所带的“convert.bas” CGI脚本不起作用。

遗憾的是，该脚本（是完全公开的）允许任何远程用户读取远程服务器上的任一文件。那将产生什么结果呢？以下地址有一个有害的代码：

```
http://victim.com/scripts/convert.bas?../../../../anything/you/want/to/view
```

Novell可能将提供一个修改这个脚本的方法，但当我写此章时，就我所能知道的，Novell仍然没有提供修改方法。所以确保在安装Novell HTTP时，禁止该脚本。

4.1.9 监视基于UNIX的Web服务器的安全问题

历史表明（查阅CERT的报告和Bulletin Advisories）基于UNIX的Web服务器在以下几个方面倾向于破坏安全：

口令缺陷。告诉用户不要从字典中选取口令。黑客经常使用finger或ruser来找出帐户名称，然后试图破解这个口令。挑选一个口令，一个好的启发式的方法是创建一个容易记忆的短语，如“Where Is Carmen Sandiego”，然后用这个短语中每个单词的第一个字母（“WICS”）作为口令。还有，选取的口令至少要有8个字符。

不变的口令。在第一次安装服务器时一定要改变缺省口令。要经常从口令文件中删除无用的帐户。通过在文件/etc/passwd中将口令域改为一个星号(*)使这个帐户失去效用；将其中的注册shell改变为/bin/false，保证入侵者不能通过网络上一个可信任的系

统来注册这个帐户。

重用口令。口令仅能使用一次，要意识到在因特网上的口令能被信息探测程序捕捉到。口令被盗。黑客使用次要文件传输协议（TFTP）窃取口令文件。如果不相信在你的系统上存在这样的弱点，可使用 TFTP 协议与它连接，然后试着获得 /etc/motd。如果你能访问它，那么在因特网上的每个人都能访问你的口令文件。为避免此类问题发生，要么使 TFTP 无效，要么限制它的访问。

4.2 URI/URL

统一资源标识符 (Uniform Resource Identifiers, URI) 是为 Web 网上资源（例如：网页、服务和文档）进行命名和编址的一组广泛应用的技术。这里已经有了一些编址策略，随着时间推移，可能有更多的编址策略。

图4-1给出了URI的基本结构：

URI 统一资源标识符 (Uniform Resource Identifier)。指出资源的所有名称和地址的一个普通集。

URL 统一资源定位器 (Uniform Resource Locator)。它是带有关于如何访问因特网资源的明确指令的一组 URI 方案。

URN 统一资源名 (Uniform Resource Name)。由以下内容组成：

一个在学术上认可的持续有效的 URI。

一个由 IETF 开发，以解决使用因特网名字协议的专门方案，比目前与因特网主机名或机构相关的方法更有持久性。一旦确定，URN 就可能作为 URI 的一个例子。

URC 统一资源索引 (Uniform Resource Citation)，也被称为统一资源特征 (Uniform Resource Characteristics)。它是用以描述资源的一组属性 / 值对。这些值可以是各种 URI，比如可以包括作者、出版商、数据类型、时间、版权等等。

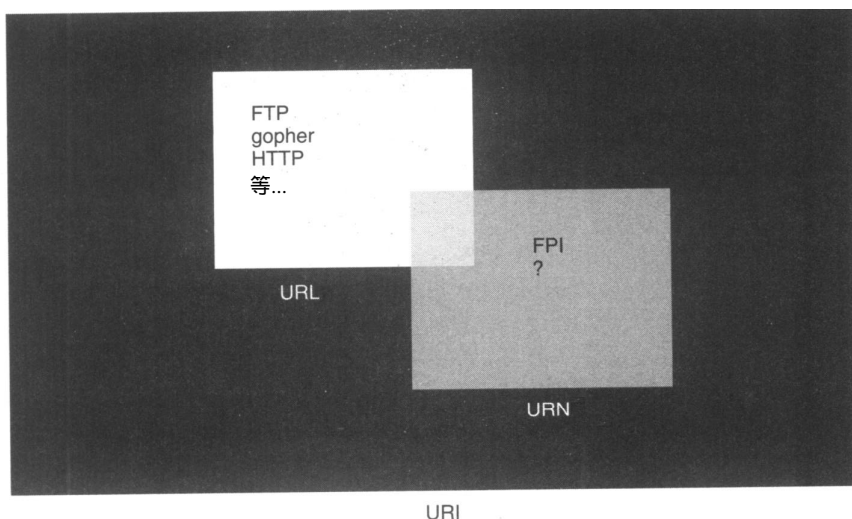


图4-1 URL 的分类

统一资源定位器 (URL) 是一类标准文件名概念的网络化扩展。一个 URL 能使你指向因

特网或内部网上任何机器中特定目录下的特定文件。同样，这个文件能通过几个不同的方式接受服务。如：HTTP、Telnet、FTP等等。

下面几节是一些像伊利诺斯大学国家计算机安全协会(<http://www.ncsa.uiuc.edu/demoweb/url-primer.html>) 所描述的那样的最普通的URL类型的概述。

4.2.1 文件URL

假定有一个叫做 foobar.txt 的文档，它位于匿名 FTP 服务器 ftp.yoyodyne.com 的目录 /pub/files 下，那么这个文件的URL为：

```
file://ftp.yoyodyne.com/pub/files/foobar.txt
```

这个FTP服务器的最高层目录为：

```
file://ftp.yoyodyne.com/
```

那么这个FTP服务器的pub目录为：

```
file://ftp.yoyodyne.com/pub
```

4.2.2 Gopher URL

因为Gopher服务器比FTP服务器需要更多的处理技巧，所以 Gopher URL 比起文件URL稍微复杂一些。访问特定 Gopher 服务器（如：在 gopher.yoyodyne.com 的Gopher服务器）使用的URL为：

```
gopher://gopher.yoyodyne.com/
```

一些Gopher服务器可以驻留在它们主机的非常用网络端口。（缺省Gopher端口号为70）。如果你知道Gopher服务器在机器（gopher.banzai.edu）上用1234端口代替了70端口，那么相应的URL为：

```
gopher://gopher.banzai.edu:1234/
```

4.2.3 新闻URL

指向Usenet新闻组（例如，rec.gardening）的URL为：

```
news:rec.gardening
```

4.2.4 部分URL

一旦正在浏览位于网络某处的一个文档（如：这个文档在 <http://www.yoyodyne.com/pub/afile.html>），就能用一个部分的、或相关的 URL，指向与该文档在同一机器上、由相同服务器软件提供服务的、同一目录下的另一个文件。例如，如果存在于同一目录下的另一个文件是anotherfile.html，那么，anotherfile.html在此就是一个有效的部分URL。

这提供了一种比较容易的方法来建立超文本文档组。如果一组超文本文档存放在同一目录下，那么这些文档仅仅使用它们的文件名就可以指向另一个文档（即：超链接）。如果阅读器到达了其中一个文档，它就可以在此将另一个文档的文件名作为部分 URL 直接跳到相同目录下的另一个文档。其他信息（访问方式、主机名、端口号、目录名等等）将被假定为基于到达第一个文档所使用的URL。

4.3 CGI

使防火墙较难保护 Web 站点的另一种威胁涉及通用网关接口 (Common Gateway Interface, CGI) 脚本程序。许多 Web 网页显示文档并将它们超链接到其他网页或站点。然而一些 Web 网页具有搜索引擎, 它允许你为特殊的信息搜索这个站点 (或几个站点), 这是通过运行 CGI 脚本的形式来实现的。

黑客能修改这些 CGI 脚本来做他们本不应该做的事情。通常, 这些 CGI 脚本只能搜索 Web 服务器区域, 但如果对它们进行修改, 就能搜索 Web 服务器以外的部分。为了阻止这类事情发生, 必须给这些脚本设置低的用户权限, 如果正在运行基于 UNIX 的服务器, 确保再次搜索那些分号 (即;)。

有许多已知的威胁方式和更多的未知方式。在下面几节, 你会了解到一些非常普遍和具有威胁性的方式。

而且, Web 服务器开放的体系结构允许在连接的服务器端运行任意的 CGI 脚本以响应远程请求。装载在站点的任何 CGI 脚本可能会包含一些小错误, 每个这样的小错误就是一个潜在的安全漏洞。

警告 当心 CGI 脚本, 因为它们是不安全的主要根源。这个协议本身是安全的, 但这些脚本一定要记住写成安全的。如果你正在站点中安装这些脚本, 注意这些问题!

对 Web 服务器软件, 情况同样如此。因为它们具有的特性越多, 产生安全漏洞的可能性就越大。提供各种特性 (如运行 CGI 脚本、实时列目录、脚本错误处理) 的服务器在防止安全漏洞方面更可能是脆弱的。甚至广泛使用的安全工具, 也不能保证总会起到作用。

注意 在站点 <http://www.webcompare.com/> 有一个 Web 服务器比较表。它包括免费软件和用于 UNIX、Novell、Windows NT、Windows 98、VMS 和许多其他操作系统的商业产品。

例如, 正当我开始写这本书的第一版之前, 我想到了两件事。第一个是众所周知的 Kerberos 系统, 为了保证分布式系统安全而被广泛采用, 它是在 20 世纪 80 年代中期由麻省理工学院 MIT 开发的。在 Purdue 大学, 来自 COAST 的人们在当时的 Kerberos 版本发现了一个易受攻击的漏洞。两个学生 Steve Lodin 和 Bryn Dole 以及 Eugene Spafford 教授找出了一个方法, 这种方法能使得对 Kerberos 4 服务器大部分执行部分没有访问特权的人, 可以攻破分配给用户的秘密会话密钥。这种方法可以允许在不知道用户口令的情况下, 对用户可用的分布式服务进行非授权访问。他们能够用一个典型的工作站在平均少于 1 分钟的记录时间演示这个过程, 有时快到 0.25 秒。

然而, 你要记住, 拒绝服务 applet 并不是病毒, 因为病毒都带有恶意的企图。事实上, 这个 Java 小错误有能力在 Web 服务器上远程执行指令, 甚至有能力从远程 Web 服务器内加载信息。自从 JDK 1.0.2 版之后以及在 NN3.0b4 中, 这个得到如此多压力的安全缺陷已被修改。

另一个你应该注意的例子是在 NCSA 和 Apache 组织提供的 HTTPD 服务器上存在的脆弱性。根据 CIAC (Computer Incident Advisory Capability), 一个用户有可能获得与 HTTPD 服务器相同的访问权限。不仅 UNIX 服务器存在这样的安全漏洞, 而且所有能运行 HTTPD 的服务器平台都存在。如果正在运行一个 NCSA HTTPD, 那么应该将它升级到最新的 1.5.1 版。

提示 你可以从下述URL下载NCSA HTTPD 1.5版

ftp://ftp.ncsa.uiuc.edu/Web/httpd/UNIX/ncsa_httpd/current/httpd_1.5.1-export_source.tar.Z。

Apache HTTPD CGI存在同样的问题：黑客能使用与正在运行 HTTPD服务器的用户相同的用户标识，在服务器主机上容易地输入任意的命令。如果 HTTPD正在以根用户运行，那么这个未授权的命令也以root运行。因为黑客正在使用与用户相同的标识，所以系统中正在运行 HTTPD服务器的用户标识可以访问的任何文件，黑客也能访问，包括但不限于毁坏服务器主机上的文件内容。

而且，如果他正在使用连接到 HTTPD服务器主机的基于 X11的终端仿真程序，那么他就能获得对服务器主机所有的交互式访问，就好像他正在本地注册一样。

如果你正在使用 Apache HTTPD，那么下面这些是必须要做的：

1) 在文件src/util.c找到escape_shell_command()函数的位置（大约第430行）。函数中，应看到下面这行：

```
if(ind( " &;` \" |*?~<^()[]{}$\\ \" ,cmd[x]) != -1){
```

2) 需要将那行改为：

```
if(ind( " &;` \" |*?~<^()[]{}$\\n \" ,cmd[x]) != -1){
```

3) 然后必须重新编译、重新安装和重启这个服务器。

运行这个升级版本是非常重要的，因为如果不对其进行升级，这个安全漏洞就能给 Web 服务器带来损害。

注意 有关更多的信息，应该在<http://ciac.llnl.gov>访问CIAC的Web网页。

该情况对Novell平台下的CGI脚本同样如此。对基于Novell平台的CGI网关实现，这个挑战性问题应归因于在产生NLM及驻留与运行在NetWare服务器上的语言编译器或解释器实现引起过多开销。为了解决这个问题，Great Lakes允许把来自Web客户的数据，存贮在NetWare服务器的一个文件中，或者作为MHS或SMTP电子邮件传送。

Netscape的通信服务器和商业服务器的NT版也受到CGI脚本处理方法的影响。下面是两个现存的问题：

Perl CGI脚本是不安全的。因为Netscape服务器不使用NT文件管理器在文件扩展名和应用之间的关联，所以当把Perl脚本放在cgi-bin目录下时，同样不能被识别。无法将扩展名.pl与Perl解释程序关联起来。如果你正在使用这些版本，Netscape技术通告推荐将Perl.exe放进cgi-bin，并且将/cgi-bin/Perl.exe?&my_script.pl作为脚本引用。

遗憾的是，这个技术在系统上打开了一个重要的安全漏洞，因为它允许一个远程用户通过执行像/cgi-bin/Perl.exe?&-e+unlink+%3C*%3E一样的脚本，在服务器上执行一组任意的Perl命令，这将使服务器当前目录下的每个文件被删除。

Netscape技术通告的另一个建议是把Perl脚本封装进批处理文件（.bat）中。然而，要注意到还存在和批处理脚本相关的问题，使得这个解决方法并不安全。

因为EMWACS，所以Purveyor和WebSite NT服务器都使用NT的文件管理扩展名关联，允许运行Perl脚本而不必将Perl.exe放进cgi-bin中。因此，这个缺陷不影响这些产品。

DOS 批处理文件是不安全的。根据Ian Redfern(redferni@logica.com)，一个相似的安全漏洞存在于作为批处理文件实现的CGI脚本中。该问题描述如下：

看看test.bat：

```
@echo off
echo Content-type: text/plain
echo
echo Hello World!
```

如果以/cgi-bin/test.bat?&dir这种形式调用这个文件，那么将得到 CGI程序的输出和一个目录清单！这就像服务器正在执行两个功能：运行批处理文件 test.bat和运行DOS列目录命令DIR。命令解释程序以/bin/sh相同的方式进行处理：运行test.bat，然后，如果完成，运行dir命令。

对这个问题的一个可能的解决方法是把批处理文件包装在经编译形成的可执行文件(.exe)当中。该可执行文件首先检查命令行参数，对DOS可能错误解释的情况进行检查，然后调用command.com子模块，并且运行批处理文件。

这需要做一些额外的工作。如果情况允许，可以在被编译的代码上做任何事情。再者如果正使用这个版本，那么一定要把它进行升级。通过在<http://www.netscape.com>访问Netscape的Web网页，就能容易地做到这一点。

同时，要记住有几个CGI脚本允许用户在线地改变他们的口令。然而，它们没有一个被充分测试而被推荐使用。如果想允许你的用户在线改变他们的口令，一些站点为这个专用目的建立了一个附加的HTTP服务器。这类附加的服务器复制口令文件。

更进一步，如果有一个FTP守护进程，通常即使在这个守护进程和Web守护进程之间共享目录，也不会破坏数据的安全性，没有远程用户能够装载文件以使Web守护进程以后能读或运行。否则，例如黑客能给你的FTP站点装载一个CGI脚本，然后使用他的浏览器从你的Web服务器请求最新装入的这个脚本文件，这样就能执行该脚本而绕过安全设施。因此，要限制FTP将文件加载到用户不能读取的目录。第8章，“Internet服务的脆弱性”有关于该问题更多的讨论。

显然，Web服务器应该支持应用网关的开发，因为这对于在信息服务器（在这种情况下是Web服务器）和另一个应用程序之间进行数据通信是必要的。

Web服务器无论在哪里需要与其他应用程序通信，都需要CGI脚本来协商服务器和外部应用程序之间的事务处理。例如，由用户用HTML格式填写的数据，可使用CGI从Web服务器传递到数据库。

但如果你想并且必须维持站点的安全，就应该对允许你的用户运行他们自己的CGI脚本保持警惕。这些脚本功能非常强大，它能给你的站点带来风险。像前面所讨论的那样，CGI脚本如果书写错误，它就能在你的系统上打开安全漏洞。因而永远不要作为root运行你的Web服务器；确保在启动时它被配置成另一个用户ID。同样考虑使用一个CGI包装来保证脚本在有作者的许可和用户ID的前提下运行。<http://www.umd.edu/~cgiwrap>下载一个包装。

提示 对于安全有关的脚本，应该查阅

<http://www.primus.com/staff/paulp/cgi-security/>。

CGI不都是有害的。控制谁正在访问Web服务器的一个好的安全工具，实际上使用CGI脚本来鉴别他们。下面5个非常重要的环境变量可帮助你这样做：

HTTP_FROM 这个变量常常被设置成用户的e-mail地址。应该使用它做为缺省的回复

e-mail地址（以e-mail格式）。

REMOTE_USER 它仅仅在安全认证被用于访问这个脚本的情况下，才被设置。可以用 *AUTH_TYPE* 变量来检查使用什么形式的认证。*REMOTE_USER* 显示被认证用户的名字。

REMOTE_IDENT 它在服务器被连接在浏览器上的一个 IDENTD 服务器上的情况下，才被设置。然而，这里没有办法保证来自浏览器的是一个诚实的应答。

REMOTE_HOST 如果服务器检索主机名，那么它提供所连接的用户站点的信息。

REMOTE_ADDR 它也提供所连接的用户站点的信息，它将提供用户的点分十进制 IP 地址。

警告 如果曾怀疑你的站点已经被非法闯入，应该与计算机应急行动小组（Computer Emergency Response Team, CERT）联系。CERT 由国防远景研究计划局（DARPA）在 1988 年组成，重点是为涉及到因特网用户的计算机安全提供服务。可以在 <http://www.cert.org> 访问他们的 Web 网页或发邮件到 cert@cert.org。

此外，CGI 能被用于在 Web 上建立 e-mail 格式。有一个由俄亥俄州大学的 Doug Stevenson 用 Perl 语言开发的 CGI e-mail 格式，它是相当安全的。称为“Web Mailto Gateway”的脚本使你能隐藏来自用户的真实的 e-mail 地址，帮助增强安全性能。下面的源代码能在 http://www.mps.ohio-state.edu/mailto_info.html 找到。

```
#!/usr/local/bin/perl
#
# Doug's WWW Mail Gateway 2.2
# 5/98
# All material here is Copyright 1998 Doug Stevenson.
#
# Use this script as a front end to mail in your HTML. Not
# every browser supports the mailto: URLs, so this is the
# next best thing. If you use this script, please leave
# credits to myself intact! :) You can modify it all you
# want, though.
#
# Documentation at:
#   http://www.bprc.mps.ohio-state.edu/mailto/
#   mailto_info.html
#
# Configurable items are just below. Also pay special
# attention to GET method arguments that this script
# accepts to specify defaults for some fields.
#
# I didn't exactly follow the RFCs on mail headers when I
# wrote this, so please send all flames my way if it
# breaks your mail client!!
# Also, you'll need cgi-lib.pl for the GET and POST
# parsing. I use version 1.7.
#
# Requires cgi-lib.pl which can be found at
#   http://www.bio.cam.ac.uk/web/form.html
#
# PLEASE: Use this script freely, but leave credits to
# myself!! It's common decency!
#
#####
```



```
#
# Changes from 1.1 to 1.2:
#
# A common modification to the script for others to make
# was to allow only a certain few mail addresses to be
# sent to. I changed the WWW Mail Gateway to allow only
# those mail addresses in the list @addrs to be mailed to
# - they are placed in a HTML <SELECT> list, with either
# the selected option being either the first one or the
# one that matches the "to" CGI variable. Thanks to Mathias
# Koerber <Mathias.Koerber@swi.com.sg> for this suggestion.
#
# Also made one minor fix.
#
#####
#
# Changes from 1.2 to 1.3:
#
# Enhancing the enhancements from 1.2. You can now specify
# a real name or some kind of identifier to go with the
# real mail address. This information gets put in the
# %addrs associative array, either explicitly defined, or
# read from a file. Read the information HTML for
# instructions on how to set this up. Also, real mail
# addresses may be hidden from the user. Undefine or set
# to zero the variable $expose_address below.
#
#####
#
# Changes from 1.3 to 1.4
#
# The next URL to be fetched after the mail is sent can be
# specified with the cgi variable 'nexturl'.
#
# Fixed some stupid HTML mistake.
#
# Force user to enter something for the username on 'Your
# Email:' tag, if identd didn't get a username.
#
# Added Cc: field, only when %addrs is not being used.
#
#####
#
# Quickie patch to 1.41
#
# Added <PRE>formatted part to header entry to make it look
# nice and fixed a typo.
#
#####
#
# Version 2.0 changes
#
# ALL cgi variables (except those reserved for mail info)
# are logged at then end of the mail received. You can put
# forms, hidden data, or whatever you want, and the info
# for each variable will get logged.
#
# Cleaned up a lot of spare code.
#
# IP addresses are now correctly logged instead of just
```

```

# hostnames.
#
# Made source retrieval optional.
#
#####
#
# Changes from 2.0 to 2.1
#
# Fixed stupid HTML error for an obscure case. Probably
# never noticed.
#
# Reported keys are no longer reported in an apparently
# random order; they are listed in the order they were
# received. That was a function of perl hashes...changed to
# a list operation instead.
#
#####
#
# Changes from 2.1 to 2.2
#
# Added all kinds of robust error checking and reporting.
# Should be easier to diagnose problems from the user end.
#
# New suggested sendmail flag -oi to keep sendmail from
# ending mail input on line containing . only.
#
# Added support for setting the "real" From address in the
# first line of the mail header using the -f sendmail
# switch. This may or may not be what you want, depending
# on the application of the script. This is useful for
# listservers that use that information for identification
# purposes or whatever. This is NOT useful if you're
# concerned about the security of your script for public
# usage. Your mileage will vary, please read the sendmail
# manual about the -f switch.
#   Thanks to Jeff Lawrence (jlaw@irus.rrri.uwo.ca) for
#   figuring this one out.
#
#####
#
# Doug Stevenson
# doug+@osu.edu
#####
# Configurable options
#####
# whether or not to actually allow mail to be sent - for
# testing purposes
$active = 1;
# Logging flag. Logs on POST method when mail is sent.
$logging = 1;
$logfile = '/usr/local/WWW/etc/mailto_log';
# Physical script location. Define ONLY if you wish to make
# your version of this source code available with GET
# method and the suffix '?source' on the url.
$script_loc = '/usr/local/WWW/cgi-bin/mailto.pl';
# physical location of your cgi-lib.pl
$cgi_lib = '/usr/local/WWW/cgi-bin/cgi-lib.pl';
# http script location
$script_http = 'http://www-bprc.mps.ohio-state.edu/
# cgi-bin/mailto.pl';

```

```

# Path to sendmail and its flags. Use the first commented
# version and define $listserver = 1 if you want the gateway
# to be used for listserver subscriptions - the -f switch
# might be necessary to get this to work correctly.
#
# sendmail options:
#   -n no aliasing
#   -t read message for "To:"
#   -oi don't terminate message on line containing '.' #
#   alone
#$sendmail = "/usr/lib/sendmail -t -n -oi -f";
$listserver = 1;
$sendmail = "/usr/lib/sendmail -t -n -oi";
# set to 1 if you want the real addresses to be exposed
# from %addrs
#$expose_address = 1;
# Uncomment one of the below chunks of code to implement
# restricted mail
# List of address to allow ONLY - gets put in a HTML
# SELECT type menu.
#
%addrs = ("Doug - main address", "doug@osu.edu",
#         "Doug at BPRC", "doug@polarmetl.mps.
#         ohio-state.edu",
#         "Doug at CIS", "stevensc@cis.ohio-state.edu",
#         "Doug at the calc lab", "dstevens@mathserver.
#         mps.ohio-state.edu",
#         "Doug at Magnus", "dmsteven@magnus.acs.
#         ohio-state.edu");
# If you don't want the actual mail addresses to be
# visible by people who view source, or you don't want to
# mess with the source, read them from $mailto_addrs:
#
$mailto_addrs = '/usr/local/WWW/etc/mailto_addrs';
open(ADDRS,$mailto_addrs);
#while(<ADDRS>) {
#   ($name,$address) = /^(.+)[\t]+([^\s]+)\n$/;
#   $name =~ s/[\t]*$//;
#   $addrs{$name} = $address;
#}
# version
$version = '2.2';
#####
# end of configurable options
#####
# source is self-contained
#####
if ($ENV{'QUERY_STRING'} eq 'source' &&
    defined($script_loc)) {
    print "Content-Type: text/plain\n\n";
    open(SOURCE, $script_loc) ||
        &InternalError('Could not open file containing
            source code');
    print <SOURCE>;
    close(SOURCE);
    exit(0);
}
require $cgi_lib;
&ReadParse();

```

```
#####
# method GET implies that we want to be given a FORM to
# fill out for mail
#####
if ($ENV{'REQUEST_METHOD'} eq 'GET') {
    # try to get as much info as possible for fields
    # To:      comes from $in{'to'}
    # Cc:      comes from $in{'cc'}
    # From:    comes from REMOTE_IDENT@REMOTE_HOST ||
    # $in{'from'} || REMOTE_USER
    # Subject: comes from $in{'sub'}
    # body comes from $in{'body'}
    $destaddr = $in{'to'};
    $cc = $in{'cc'};
    $subject = $in{'sub'};
    $body = $in{'body'};
    $nexturl = $in{'nexturl'};
    if ($in{'from'}) {
        $fromaddr = $in{'from'};
    }
    # this is for NetScape pre-1.0 beta users - probably
    # obsolete code
    elseif ($ENV{'REMOTE_USER'}) {
        $fromaddr = $ENV{'REMOTE_USER'};
    }
    # this is for Lynx users, or any HTTP/1.0 client
    # giving From header info
    elseif ($ENV{'HTTP_FROM'}) {
        $fromaddr = $ENV{'HTTP_FROM'};
    }
    # if all else fails, make a guess
    else {
        $fromaddr = "$ENV{'REMOTE_IDENT'}\
            @$ENV{'REMOTE_HOST'}";
    }
    # Convert multiple bodies (separated by \0 according
    # to CGI spec)
    # into one big body
    $body =~ s/\0//;
    # Make a list of authorized addresses if %addrs exists.
    if (%addrs) {
        $selections = '<SELECT NAME="to">';
        foreach (sort keys %addrs) {
            if ($in{'to'} eq $addrs{$_}) {
                $selections .= "<OPTION SELECTED>$_";
            }
            else {
                $selections .= "<OPTION>$_";
            }
            if ($expose_address) {
                $selections .= " <lt;$addrs{$_}>";
            }
        }
        $selections .= "</SELECT>\n";
    }
    # give them the form
    print &PrintHeader();
    print <<EOH;
<HTML><HEAD><TITLE>Doug\'s WWW Mail Gateway
$version</TITLE></HEAD>
<BODY><H1><IMG SRC="http://www-bprc.mps.ohio-state.edu/
```

```

pics/mail2.gif" ALT="">
The WWW Mail Gateway $version</H1>
<P>The <B>To</B>: field should contain the <B>full</B>
Email address that you want to mail to. The <B>Your
Email</B>: field needs to contain your mail address so
replies go to the right place. Type your message into the
text area below. If the <B>To</B>: field is invalid, or the
mail bounces for some reason, you will receive notification
if <B>Your Email</B>: is set correctly. <I>If <B>Your
Email</B>: is set incorrectly, all bounced mail will be
sent to the bit bucket.</I></P>
<FORM ACTION="$script_http" METHOD=POST>
EOH
;
print "<P><PRE>          <B>To</B>: ";
# give the selections if set, or INPUT if not
if ($selections) {
    print $selections;
}
else {
    print "<INPUT VALUE=\"$destaddr\" SIZE=40
        NAME=\"to\">\n";
    print "          <B>Cc</B>: <INPUT VALUE=\"$cc\"
        SIZE=40 NAME=\"cc\">\n";
}
print <<EOH;
<B>Your Name</B>: <INPUT VALUE="$fromname" SIZE=40
    NAME="name">
<B>Your Email</B>: <INPUT VALUE="$fromaddr" SIZE=40
    NAME="from">
<B>Subject</B>: <INPUT VALUE="$subject" SIZE=40
    NAME="sub"></PRE>
<INPUT TYPE="submit" VALUE="Send the mail">
<INPUT TYPE="reset" VALUE="Start over"><BR>
<TEXTAREA ROWS=20 COLS=60 NAME="body">$body</TEXTAREA><BR>
<INPUT TYPE="submit" VALUE="Send the mail">
<INPUT TYPE="reset" VALUE="Start over"><BR>
<INPUT TYPE="hidden" NAME="nexturl" VALUE="$nexturl"></P>
</FORM>
<HR>
<H2>Information about the WWW Mail Gateway</H2>
<H3><A HREF="http://www-bprc.mps.ohio-state.edu/mailto/
    mailto_info.html#about">
About the WWW Mail Gateway</A></H3>
<H3><A HREF="http://www-bprc.mps.ohio-state.edu/mailto/
    mailto_info.html#new">
New in version $version</A></H3>
<H3><A HREF="http://www-bprc.mps.ohio-state.edu/mailto/
    mailto_info.html#misuse">
Please report misuse!</A></H3>
<HR>
<ADDRESS><P><A HREF="/~doug/">Doug Stevenson: doug+\\
    @osu.edu</A>
</P></ADDRESS>
</BODY></HTML>
EOH
;
}
#####
# Method POST implies that they already filled out the form

```



```
# and submitted it, and now it is to be processed.
#####
elif ($ENV{'REQUEST_METHOD'} eq 'POST') {
    # get all the variables in their respective places
    $destaddr = $in{'to'};
    $cc       = $in{'cc'};
    $fromaddr = $in{'from'};
    $fromname = $in{'name'};
    $replyto  = $in{'from'};
    $sender   = $in{'from'};
    $errorsto = $in{'from'};
    $subject  = $in{'sub'};
    $body     = $in{'body'};
    $nexturl  = $in{'nexturl'};
    $realfrom = $ENV{'REMOTE_HOST'} ? $ENV{'REMOTE_HOST'} :
        $ENV{'REMOTE_ADDR'};
    # check to see if required inputs were filled - error
    # if not
    unless ($destaddr && $fromaddr && $body && ($fromaddr
        =~ /^.+@.+/)) {
        print <<EOH;
        Content-type: text/html
        Status: 400 Bad Request
        <HTML><HEAD><TITLE>Mailto error</TITLE></HEAD>
        <BODY><H1>Mailto error</H1>
        <P>One or more of the following necessary pieces of
        information was missing from your mail submission:
        <UL>
        <LI><B>To</B>:, the full mail address you wish to send mail
        to</LI>
        <LI><B>Your Email</B>: your full email address</LI>
        <LI><B>Body</B>: the text you wish to send</LI>
        </UL>
        Please go back and fill in the missing
        information.</P></BODY></HTML>
        EOH
        exit(0);
    }
    # do some quick logging - you may opt to have
    # more/different info written
    if ($logging) {
        open(MAILLOG,">>$logfile");
        print MAILLOG "$realfrom\n";
        close(MAILLOG);
    }
    # Log every CGI variable except for the ones reserved
    # for mail info.
    # Valid vars go into @data. Text output goes into
    # $data and gets.
    # appended to the end of the mail.
    # First, get an ORDERED list of all cgi vars from @in
    # to @keys
    for (0 .. $#in) {
        local($key) = split(/=/,$in[$_],2);
        $key =~ s/\+ / /g;
        $key =~ s/%(..)/pack("c",hex($1))/ge;
        push(@keys,$key);
    }
    # Now weed out the ones we want
    @reserved = ('to', 'cc', 'from', 'name', 'sub',
```

```

    'body', 'nexturl');
local(%mark);
foreach (@reserved) { $mark{$_} = 1; }
@data = grep(!$mark{$_}, @keys);
foreach (@data) {
    $data .= "$_ -> $in{$_}\n";
}
# Convert multiple bodies (separated by \0 according
# to CGI spec)
# into one big body
$body =~ s/\0//;
# now check to see if some joker changed the HTML to
# allow other
# mail addresses besides the ones in %addrs, if
# applicable
if (%addrs) {
    if (!scalar(grep($_. " <$addrs{$_}>" eq $destaddr ||
                    $destaddr eq $_,
                    keys(%addrs)))) {
        print &PrintHeader();
        print <<EOH;
<HTML><HEAD><TITLE>WWW Mail Gateway: Mail address not
    allowed</TITLE></HEAD>
<BODY>
<H1>Mail address not allowed</H1>
<P>The mail address you managed to submit,
<B>$destaddr</B>, to this script is not one of the
pre-defined set of addresses that are allowed. Go back and
try again.</P>
</BODY></HTML>
EOH
    ;
        exit(0);
    }
}
# if we just received an alias, then convert that to
# an address
$realaddr = $destaddr;
if ($addrs{$destaddr}) {
    $realaddr = "$destaddr <$addrs{$destaddr}>";
}
# fork over the mail to sendmail and be done with it
if ($active) {
    if ($listserver) {
        open(MAIL, "| $sendmail$fromaddr") ||
            &InternalError('Could not fork sendmail
with -f switch');
    }
    else {
        open(MAIL, "| $sendmail") ||
            &InternalError('Could not fork sendmail
with -f switch');
    }
    # only print Cc if we got one
    print MAIL "Cc: $cc\n" if $cc;
    print MAIL <<EOM;
From: $fromname <$fromaddr>
To: $realaddr
Reply-To: $replyto
Errors-To: $errorsto

```

```

Sender: $sender
Subject: $subject
X-Mail-Gateway: Doug\'s WWW Mail Gateway $version
X-Real-Host-From: $realfrom
$body
$data
EOM

    close(MAIL);
}
# give some short confirmation results
#
# if the cgi var 'nexturl' is given, give out the
# location, and let the browser do the work.
if ($nexturl) {
    print "Location: $nexturl\n\n";
}
# otherwise, give them the standard form.
else {
    print &PrintHeader();
    print <<EOH;
<HTML><HEAD><TITLE>Mailto results</TITLE></HEAD>
<BODY><H1>Mailto results</H1>
<P>Mail sent to <B>$destaddr</B>:<BR><BR></P>
<PRE>
<B>Subject</B>: $subject
<B>From</B>: $fromname &lt;$fromaddr>
$body</PRE>
<HR>
<A HREF="$script_http">Back to the WWW Mailto Gateway</A>
</BODY></HTML>
EOH
    }
}
# end if METHOD=POST
#####
# What the heck are we doing here????
#####
else {
    print <<EOH;
<HTML><HEAD><TITLE>Mailto Gateway error</TITLE></HEAD>
<BODY><H1>Mailto Gateway error</H1>
<P>Somehow your browser generated a non POST/GET request
method and it got here. You should get this fixed!!</P>
</BODY></HTML>
EOH
}
exit(0);
#
# Deal out error messages to the user. Gets passed a
# string containing a description of the error
#
sub InternalError {
    local($errmsg) = @_ ;
    print &PrintHeader();
    print <<EOH;
Content-type: text/html
Status: 502 Bad Gateway
<HTML><HEAD><TITLE>Mailto Gateway Internal
    Error</TITLE></HEAD>
<BODY><H1>Mailto Gateway Internal Error</H1>
<P>Your mail failed to send for the following

```

```
        reason:<BR><BR>
<B>$errmsg</B></P></BODY></HTML>
EOH
        exit(0);
}
##
## end of mailto.pl
##
```

如果服务器能运行 CGI 脚本并且配置 sendmail，那么在 HTML 上的邮件网关脚本就是好的和安全的；不过必须在服务器上能够运行 CGI 脚本。