第 5 章 网 络 安 全

本章学习要点：

- 了解网络所面临的安全威胁；
- 掌握防止网络攻击的控制措施；
- 了解防火墙的体系结构、类型、能力和限制，掌握防火墙的基本工作原理；
- 了解入侵检测系统的功能及类型；
- 了解虚拟专用网的类型和协议；
- 了解移动通信网络安全和无线局域网安全。

网络安全从其本质上来讲就是网络上的信息安全，涉及的领域相当广泛，这是因为在目前的公用通信网络中存在着各种各样的安全漏洞和威胁。凡是涉及网络信息的保密性、完整性、可用性、真实性和可控性的相关技术和理论，都是网络安全所要研究的领域。严格地说，网络安全指网络系统的硬件、软件及其系统中的数据受到保护，不受偶然的或者恶意的原因而遭到破坏、更改、泄露，系统连续可靠正常地运行，网络服务不中断。

5.1 网络安全威胁与控制

5.1.1 网络安全威胁

5.1.1.1 威胁分类

网络所面临的安全威胁大体可分为两种：一种是对网络本身的威胁，另一种是对网络中信息的威胁。对网络本身的威胁包括对网络设备和网络软件系统平台的威胁；对网络中信息的威胁除了包括对网络中数据的威胁外，还包括对处理这些数据的信息系统、应用软件的威胁。

这些威胁主要来自人为的无意失误、人为的恶意攻击、网络软件系统的漏洞和“后门”三方面的因素。

（1）人为的无意失误是造成网络不安全的重要原因。网络管理员在这方面不但肩负重任，还面临越来越大的压力。稍有考虑不周，安全配置不当，就会造成安全漏洞。另外，用户安全意识不强，不按照安全规定操作（如口令选择不慎，将自己的账户随意转借他人或与别人共享），都会对网络安全带来威胁。

（2）人为的恶意攻击是目前计算机网络所面临的最大威胁。人为的恶意攻击又可以分为两类：一类是主动攻击，它以各种方式有选择地破坏系统和数据的有效性和完整性；另一类是被动攻击，它是在不影响网络和应用系统正常运行的情况下，进行截获、窃取、破译，以获得重要机密信息。这两种攻击均可对计算机网络造成极大的危害，导致网络瘫痪或机密泄露。

(3) 网络软件系统不可能百分之百无缺陷和无漏洞。许多软件都存在编程人员为了方便而设置的“后门”。这些漏洞和“后门”恰恰是黑客进行攻击的首选目标。

多数安全威胁都具有相同的特征,即威胁的目标都是破坏机密性、完整性或可用性;威胁的对象包括数据、软件和硬件;实施者包括自然现象、偶然事件、无恶意的用户和恶意攻击者。

5.1.1.2 对网络本身的威胁

1. 协议的缺陷

网络协议是网络的基础,协议的缺陷是网络安全威胁的根源之一。互联网联盟为了详细检查所有因特网协议,而将它们公开发布出来。每一种被接受的协议都被分配了一个互联网征求意见稿(Request For Comments,RFC)标准(草案)编号。在协议被接受成为一个标准之前,许多协议中存在的问题就已经被那些敏锐的检查者发现并得到了校正。

但是,协议的定义是由人制定和审核的,协议本身可能是不完整的,也难免存在某些缺陷。某些网络协议的实现是很多安全缺陷的源头,攻击者可以利用这些错误,特别是下述软件的故障:网络管理(SNMP),寻址服务(DNS)和电子邮件(E-mail)服务(如 SMTP 和 S/MIME)。虽然不同的厂商会编写实现他们自己服务的代码,但他们常常基于通用(有缺陷)的原型。这样,在 Windows 上成功的交互,有可能在 UNIX 上失效。例如,针对 SNMP 缺陷(漏洞代码:107186),CERT 报告列出了建议使用的近 200 套不同的实现方案。

2. 网站漏洞

因为网络几乎完全暴露在用户面前,所以非常脆弱。如果用户使用应用程序,不会获取并查看程序代码。对于网站来说,攻击者能下载网站代码,再离线长时间研究。对于程序而言,几乎不能控制使用哪种顺序访问程序的不同部分,但是,网站攻击者可以控制以哪种顺序访问网页,甚至直接访问网页 5,而不按 1～4 的顺序访问。攻击者也能选择提供哪种数据,以及用不同的数据进行实验,以测试网站的反应。简而言之,攻击者在挑战控制权方面具有优势。

(1) 网站被“黑”。一种最广为人知的攻击方式是网站被“黑”式攻击。这不仅因为其结果是可见的,而且实施起来也比较容易。由于网站的设计使得代码可以下载,这就允许攻击者能够获取全部超文本文档和在加载进程中与客户相关的所有程序。攻击者甚至可以看到编程者在创建或维护代码时遗留下来的注解。下载进程实质上为攻击者提供了一份该网站的规划图。

(2) 缓冲区溢出。网页也存在缓冲区溢出问题。攻击者向一个程序中输入大量数据,比预期所要接收的数据多得多。由于缓冲区的大小是有限的,所以过剩的数据就会溢出到相邻的代码和数据区域中去。

知名的网页服务器缓冲区溢出之一是被称为 iishack 的文件名问题。这种攻击方式被写进了一个程序中(参见 http://www.technotronic.com),只需提供要攻击的站点和攻击者想要服务器执行的程序的 URL 作为参数,攻击者就可以执行该程序实施攻击。

其他网页服务器对于极长的参数字段也很容易发生缓冲区溢出错误。比如,长度为 10 000 的口令或者填充大量空格或空字符的长 URL。

(3) “../”问题。网页服务器代码应该一直在一个受到限制的环境中运行。在理想情况下,网页服务器上应该没有编辑器、xterm 和 Telnet 程序,甚至连绝大多数系统应用程序都不应该安装。通过这种方式限制了网页服务器的运行环境以后,即使攻击者从网页服务器的应用程序区跳到了别处,也没有其他可执行程序可以帮助攻击者使用网页服务器所在的计算机

和操作系统来扩大攻击的范围。用于网页应用程序的代码和数据可以采用手工方式传送到网页服务器。但是，相当多的应用软件程序员却喜欢在存放网页应用程序的地方编辑它，因此，他们认为有必要保留编辑器和系统应用程序，为他们提供一个完整的开发环境。

第二种阻止攻击的方法是创建一个界地址来限制网页服务器应用程序的执行区域。有了这样一个界地址，服务器应用程序就不能从它的工作区域中跳出来访问其他具有潜在危险的系统区域（如编辑器和系统应用程序）了。服务器把一个特定的子目录作为根目录，服务器需要的所有东西都放在以此根目录开始的同一个子目录中。

无论是在 UNIX 还是在 Windows 操作系统中，“..”都代表某一个目录的父目录。以此类推，“../..”就是当前位置的祖父目录。因此，每输入一次“..”就可以进入目录树的上一层目录。Cerberus Information Security 的分析家们发现微软索引服务器的扩展文件 webhits.dll 中就存在这个漏洞。例如，传递一个如下的 URL 会导致服务器返回请求的 autoexec.nt 文件，从而允许攻击者修改或者删除它：

```
http://yoursite.com/webhits.htw?ciwebhits&file=../../../../../winnt/
system32/autoexec.nt
```

（4）应用代码错误。用户的浏览器与网页服务器之间传递着一种复杂且无状态的协议交换。网页服务器为了使自己的工作更轻松一些，会向用户传递一些上下文字符串，而要求用户浏览器用全部上下文进行应答。一旦用户可以修改这种上下文内容，就会出现问题。

下面用一个假想的 CD 销售站点 CDs-R-Us 来说明这个问题。在某一个特定时刻，该站点的服务器可能有一千甚至更多个交易正处于不同的状态。该站点显示了供订购的货物清单网页，用户选择其中的一种货物，站点会显示出更多的货物，用户又选择其中的几种，如此进行下去，直到用户结束选择为止。然后，很多人会通过指定付账和填入邮购信息继续完成这份订单，但也有一些人将该网站作为在线目录或者指南，而没有实际订购货物的意图。比如利用该站点来查询 CD 的价格，即使用户确实有购物的诚意，有时也会由于网页连接失败而留下一个不完整的交易。正是考虑到这些因素，网页服务器常常通过一些紧跟在 URL 之后的参数字段来跟踪一个还没有完成的订单的当前状态。随着每个用户的选择或者页面请求操作，这些字段从服务器传递到浏览器，然后又返回给服务器。

假设你已经选择了一张 CD，正在查看第二个网页。网页服务器已经传递给你一个与此类似的 URL：

```
http://www.CDs-r-us.com/buy.asp?i1=459012&p1=1599
```

该 URL 意味着你已经选择了一张编号为 459012 的 CD，单价是 15.99 美元。现在，你选择了第二张 CD，而 URL 变成了：

```
http://www.CDs-r-us.com/buy. asp? i1=459012&p1=1599&i2=365217&p2=1499
```

如果你是一位高明的攻击者，就会知道在用户浏览器的地址窗口中的 URL 是可以编辑的。结果，将其中的 1599 和 1499 都改成了 199。这样，当服务器汇总你的订单时，两张 CD 的单价都只有 1.99 美元了。

在第一次需要显示价格的时候，服务器会设置（检查）每一项物品的价格。但后来，被检查过的数据项失去了控制，而没有对它们进行复核。这种情况经常出现在服务器应用程序代码

中，因为应用程序编程人员常常没有意识到其中存在的安全问题，以至于常常对一些恶意的举动没有预见性。

(5) 服务器端包含。一种具有代表性的更严重问题是服务器端包含(Server-Side Include，SSI)问题。该问题利用了一个事实：网页中可以自动调用一个特定的函数。例如，很多页面的最后都显示了一个“请与我联系”链接，并使用一些 Web 命令来发送电子邮件消息。这些命令(如 E-mail、if、goto 和 include 等)都被置于某一个区域，以便转换成 HTML 语言。

其中一种服务器端包含命令称为 exec，用于执行任意一个存放于服务器上的文件。例如，以下服务器端包含命令：

```
<!-#exec cmd="/usr/bin/telnet &"->
```

会以服务器的名义(也就是说，具有服务器的特权)打开一个在服务器上运行的 Telnet 会话。攻击者会对执行像 chmod(改变一个对象的访问权限)、sh(建立一个命令行解释器)或 cat(复制到一个文件)这样的命令很感兴趣。

3. 拒绝服务

可用性攻击，有时被称为拒绝服务攻击或 DoS 攻击，在网络中比在其他的环境中更加值得重视。可用性或持续服务面临着很多意外或恶意的威胁。

(1) 传输故障。有很多原因会导致通信故障。比如：电话线被切断；网络噪声使得一个数据包不能被识别或不能被投递；传输路径上的一台设备出现软件或硬件故障；一台设备因维修或测试而停止服务；某台设备被太多任务所淹没，从而拒绝接收其他输入数据，直到所有过载数据被清除为止。在一个主干网络(包括因特网)中，其中的许多问题都是临时出现或能够自动恢复(通过绕道的方式)的。

然而，一些故障却很难修复。比如，连接到你使用的计算机的唯一一根通信线路(例如，从网络到你的网卡或者连到你的 Modem 上的电话线)被折断了，就只能通过另外接一根线或修理那根被损坏的线来进行恢复。网络管理员会说“这对网络的其他部分不会造成影响”，但对你而言，这句话起不到任何作用。

站在一个恶意的立场来看，所有可以切断线路、干扰网络或能使网络过载的人都可以造成你得不到服务。来自物理上的威胁是相当明显的。

(2) 连接洪泛。最早出现的拒绝服务攻击方式是使连接出现泛滥。如果一名攻击者给你发送了太多数据，以至于你的通信系统疲于应付，这样，就没空接收任何其他数据了。即使偶尔有一两个来自其他人的数据包被你收到，你们之间的通信质量也会出现严重降级。

一些更为狡猾的攻击方式使用了互联网协议中的元素。除了 TCP 和 UDP 协议以外，互联网协议中还有一类协议，被称为网际控制报文协议(Internet Control Message Protocol，ICMP)，通常用于系统诊断。这些协议与用户应用软件没有联系。ICMP 协议内容如下。

- Ping：用于要求某个目标返回一个应答，目的是看目标系统是否可以到达，以及是否运转正常。
- Echo：用于请求一个目标将发送给它的数据发送回来，目的是看连接链路是否可靠(Ping 实际上是 Echo 的一个特殊应用)。
- Destination Unreachable：用于指出一个目标地址不能被访问。
- Source Quench：意味着目标即将达到处理极限，数据包的发送端应该在一段时间内暂

停发送数据包。

这些协议对于网络管理有重要的作用。但是，它们也可能用于对系统的攻击。由于这些协议都是在网络堆栈中进行处理的，因而在接收主机端检测或阻塞这种攻击是很困难的。下面介绍怎样发动该类型的攻击。

① 响应索取。这种攻击发生在两台主机之间。chargen 是一个用于产生一串数据包的协议，常常用于测试网络的容量。攻击者在主机 A 上建立起一个 chargen 进程，以产生一串包，作为对目标主机 B 的响应包。然后，主机 A 生成一串包发送给主机 B，主机 B 通过响应它们，返回这些包给主机 A。这一系列活动使得网络中包含主机 A 和主机 B 部分的基础设施进入一种无限循环状态。更有甚者，攻击者在发送第一个包的时候，将它的目标地址和源地址都设置成主机 B 的地址，这样，主机 B 就会陷入一个循环之中，不断地对它自己发出的消息做出应答。

② 死亡之 Ping。死亡之 Ping(Ping of death)是一种简单的攻击方式。因为 Ping 要求接收者对 Ping 请求做出响应，故攻击者所要做的事情就是不断地向攻击目标发送大量的 Ping，以淹没攻击目标。然而，这种攻击要受攻击路径上最小带宽的限制。如果攻击者使用的是 10 Mb/s 带宽的连接，而被攻击目标的路径带宽为 100 Mb/s 甚至更高，那么，单凭攻击者自身是不足以淹没攻击目标的。但是，如果将这两个数字对换一下，即攻击者使用 100 Mb/s 的连接，而被攻击目标的路径带宽为 10 Mb/s，则攻击者可以轻易地淹没攻击目标。这些 Ping 包将会把攻击目标的带宽堵塞得满满当当。

③ Smurf 攻击。Smurf 攻击是 Ping 攻击的一个变体。它采用与 Ping 攻击方式相同的载体——Ping 包，但使用了另外两种手法。首先，攻击者需要选择不知情的受害者所在的网络。攻击者假造受害者的主机地址作为 Ping 包中的源地址，以使 Ping 包看起来像是从受害者主机发出来的一样。然后，攻击者以广播模式(通过将目标地址的最后一字节全部设置为 1)向网络发送该请求，这些广播包就会发布给网络上的所有主机，如图 5.1 所示。

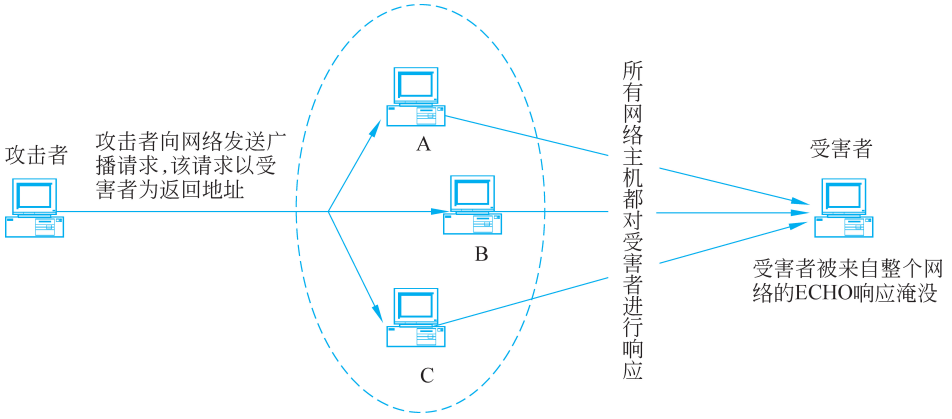

图 5.1 Smurf 攻击

④ 同步洪泛。同步洪泛(SYN flood)是另一种流行的拒绝服务攻击。这种攻击利用了 TCP 协议组，使用这些面向会话的协议来实施攻击。

对于一个协议(如 Telnet)，在协议的对等层次之间将建立一个虚拟连接，称为一个会话(session)，以便对 Telnet 终端模仿自然语言中来来回回、有问有答的交互过程进行同步。三次 TCP 握手建立一个会话。每个 TCP 包都有一些标记位，其中有两个标记位表示同步

(SYN)和应答(ACK)。在开始一次 TCP 连接时,连接发起者会发送一个设置了 SYN 标记的包。如果接收方准备建立一个连接,就会用一个设置了 SYN 和 ACK 标记的包进行应答。然后,发起方发送一个设置了 ACK 标记的包给接收方,这样就完成了建立一个清晰完整的通信通道的交换过程,如图 5.2 所示。

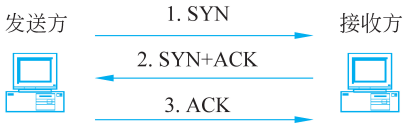

图 5.2　三次连接握手

包在传输过程中偶尔会出现丢失或者损坏的情况。因此,在接收端维持着一个称为 SYN_RECV 连接的队列,用于跟踪已经发送了 SYN-ACK 信号但还没有收到 ACK 信号的项。在正常情况下,这些工作在一段很短的时间内就会完成。但如果 SYN+ACK 或 ACK 包丢失,最终目标主机会由于这个不完整的连接超时而将它从等待队列中丢掉。

攻击者可以通过发送很多 SYN 请求而不以 ACK 响应,从而填满对方的 SYN_RECV 队列来对目标进行拒绝服务攻击。通常 SYN_RECV 队列相当小,只能容纳 10 个或者 20 个表项。由于在互联网中存在潜在的传输延迟,通常在 SYN_RECV 队列中保留数据的时间最多可达几分钟。因此,攻击者只需要每隔几秒钟发送一个新的 SYN 请求,就可以填满该队列。

攻击者在使用这种方法的时候,通常还要做一件事情:在初始化 SYN 包中使用一个不存在的返回地址来欺骗对方。这样做有两个原因:第一,攻击者不希望泄露真实的源地址,以免被通过检查 SYN_RECV 队列中的包而试图识别攻击者的人认出来;第二,攻击者想要使这些伪造的 SYN 包与用于建立真实连接的合法 SYN 包没有区别。为每个包选择一个不同的(骗人的)源地址,以使它们是唯一的。一个 SYN+ACK 包发往一个不存在的地址会导致网络发出一个“目标不可达”的 ICMP 报文,但这不是 TCP 所期待的 ACK 信号。TCP 和 ICMP 是不同的协议组,因此,一个 ICMP 应答不需要返回到发送者的 TCP 处理部分。

⑤ Teardrop 攻击。Teardrop 攻击滥用了被设计用来改善网络通信的特性。一个网络 IP 数据报是一个变长的对象。为了支持不同的应用和不同的情况,数据报协议允许将单个数据单元分片,即分成小段数据,分别发送。每个分片可表明其长度和在数据单元中的相对位置。接收端负责重新将分片组装成单个数据单元。

在 Teardrop 攻击中,攻击者发送一系列数据报,这些数据报不能被正确组装在一起。一个数据报表明它的位置在长度为 60 字节的数据单元的位置 0 处。另一个表明它在 90 字节的数据单元的位置 30 处,还有一个表明它在 173 字节的数据单元位置 41 处。这三个分片是重叠的,所以,其不能正确重组。在极端情况下,操作系统将把不能重组的数据单元部分锁住,而导致拒绝服务。

(3) 流量重定向。路由器工作在网络层,是一种在源主机所在网络与目标主机所在网络之间,通过一些中间网络来向前传递消息的设备。因此,如果攻击者可以破坏寻址,路由器就不能正确传递消息。

路由器使用复杂的算法来决定如何进行路径选择。不管采用何种算法,从本质上说都是为了寻找一条最好的路径(在这里,“最好”是通过一些综合指标来进行衡量的,如距离、时间、费用和质量等)。每个路由器只知道与它共享相同网络连接的路由器,路由器之间使用网关协议来共享一些信息,这些信息是关于彼此之间的通信能力的。每个路由器都要向它的相邻路由器通告它自己到达其他网络的路径情况。这个特点可以被攻击者用来破坏网络。

说到底,路由器都只是一台带有两块或者更多网卡的计算机。假设一台路由器向它的所有相邻路由器报告:它到整个网络的每一个其他地址都有最好的路径。很快,所有相邻路由

器都会将所有通信传递到该路由器。这样，这台路由器就会被大量通信所淹没，或者只能将大多数通信一丢了之。无论出现哪一种情况，都会造成大量通信永远不能到达预期的目标。

（4）DNS攻击。最后一种拒绝服务攻击是一类基于域名服务器（Domain Name Server，DNS）的攻击。DNS是一张表，用于将域名（如ATT.COM）转换成对应的网络地址（如211.217.74.130），这个过程称为域名解析。域名服务器在遇到它不知道的域名时，通过向其他域名服务器提出询问来进行解析。出于效率的考虑，它会将收到的域名解析应答进行缓存，以便将来在解析该域名的时候能够更快一些。

在绝大多数采用UNIX实现域名服务的系统中，域名服务器运行的软件称为BIND（Berkeley Internet Name Domain）或Named（name daemon的简写）。在BIND中存在着大量缺陷，包括现在大家熟悉的缓冲区溢出缺陷。

通过接管一个域名服务器或者使其存储一些伪造的表项（称为DNS缓存中毒），攻击者可以对任何通信进行重定向，从而造成拒绝服务。

在2002年10月，大量洪泛流量淹没了顶级域名DNS服务器，这些服务器构成了互联网寻址的基石。大约一半的流量仅来自200个地址。虽然人们认为这些问题是防火墙的误配置，但没有人确知是什么引起了攻击。

在2005年3月，一次攻击利用了Symantec防火墙的漏洞，该漏洞允许修改Windows系统中的DNS记录。但这次攻击的目的不是拒绝服务。在这次攻击中，“中招”的DNS缓存将用户重定向到了广告网站，这些广告网站在每次用户访问网站时进行收费。同时，这次攻击也阻止用户访问合法网站。

4. 分布式拒绝服务（DDoS）

上面所列举的拒绝服务攻击本身就已经非常具有威力了，但是，攻击者还可以采取一种两阶段的攻击方式，攻击效果可以扩大很多倍。这种乘数效应为分布式拒绝服务攻击提供了巨大威力。攻击者发起DDoS攻击的第一步是在Internet上寻找有漏洞的主机并试图侵入，入侵成功后在其中安装后门或木马程序；第二步是在入侵各主机上安装攻击程序，由程序功能确定其扮演的不同角色；最后由各部分主机各司其职，在攻击者的调遣下对目标主机发起攻击，制造数以百万计的数据分组流入欲攻击的目标，致使目标主机或网络极度拥塞，从而造成目标系统的瘫痪。

与DoS一次只能运行一种攻击方式攻击一个目标不同，DDoS可以同时运用多种DoS攻击方式，也可以同时攻击多个目标。攻击者利用成百上千个被“控制”节点向受害节点发动大规模的协同攻击。通过消耗带宽、CPU和内存等资源，造成被攻击者性能下降，甚至瘫痪和死机，从而造成合法用户无法正常访问。与DoS相比，其破坏性和危害程度更大，涉及范围更广，更难发现攻击者。DDoS的攻击原理如图5.3所示。

（1）攻击者。攻击者可以是网络上的任何一台主机。在整个攻击过程中，它是攻击主控台，向主控机发送攻击命令，包括被攻击者主机地址，控制整个攻击过程。攻击者与主控机的通信一般不包括在DDoS工具中，可以通过多种连接方法完成，最常用的是Telnet TCP终端会话，还可以是绑定到TCP端口的远程Shell、基于UDP的客户/服务器远程Shell等。

（2）主控机。主控机和代理主机都是攻击者非法侵入并控制的一些主机，它们分成了两个层次，分别运行非法植入的不同的攻击程序。每个主控机控制一部分代理主机，主控机有其控制的代理主机的地址列表，它监听端口接收攻击者发来的命令后，将命令转发给代理主机。主控机与代理主机的通信根据DDoS工具的不同而有所不同。例如，Trinoo使用UDP协议，

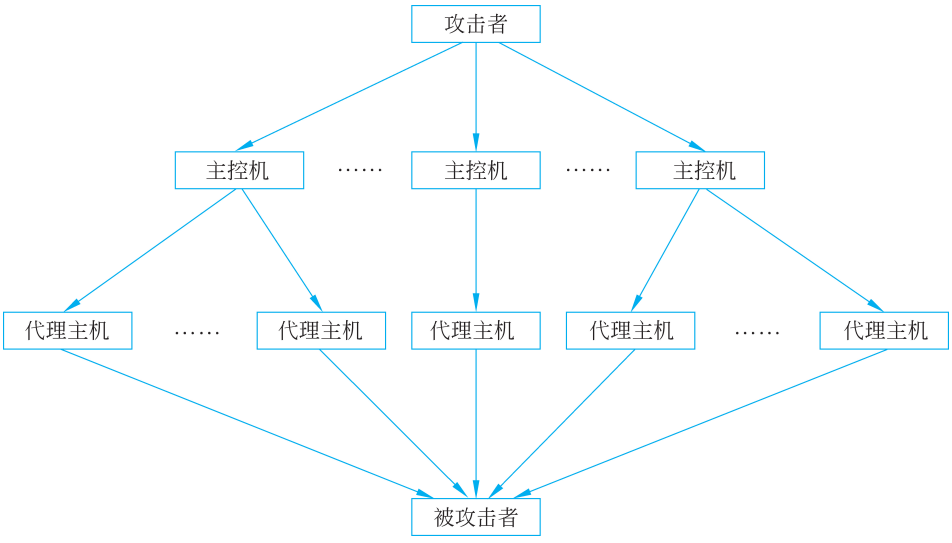

图 5.3 分布式拒绝服务攻击原理图

TFN 使用 ICMP 协议，Stacheldraht 使用 TCP 和 ICMP 协议。

(3) 代理主机。代理主机运行攻击程序，监听端口接收和运行主控机发来的命令，是真正进行攻击的机器。

(4) 被攻击者。被攻击者可以是路由器、交换机、主机等。遭受攻击时，它们的资源或带宽被耗尽。防火墙、路由器的阻塞还可能导致恶性循环，加重网络拥塞情况。

除了巨大的乘数效应以外，也很容易通过脚本来实施分布式拒绝服务攻击，这也是一个严重的问题。只要给出了一套拒绝服务攻击方式和一种特洛伊木马繁殖方式，人们就可以很容易地写出一个程序来植入特洛伊木马，该特洛伊木马就可以用任何一种或所有的拒绝服务攻击方法实施攻击。DDoS 攻击工具最早出现于 1999 年，包括 TFN(Tribal Flood Network)、Trinoo 及 TFN2K(Tribal Flood Network，Year 2000 Edition)。随着一些新弱点的发现，特洛伊木马的植入方式也随之发生了一些改变，而且，随着一些新的拒绝服务攻击方式被发现，也相应出现了一些新的组合工具。

5. 来自活动或移动代码的威胁

活动代码(active code)或移动代码(mobile code)是对被“推入”客户端执行的代码的统称。网页服务器为什么要浪费宝贵的资源和带宽去做那些客户工作站能做的简单工作呢？假设想让你的网站上出现一些熊跳着舞跨过页面顶部的画面。为了下载这些正在跳舞的熊，你可能会在这些熊每一次运动的时候下载一幅新图像：向前移动一点，再向前移动一点，如此继续下去。然而，这种方法占用了服务器太多的时间和带宽，因为需要服务器来计算这些熊的位置并下载很多新的图像。一种更有效利用(服务器)资源的方式是直接下载一个实现熊运动的程序，让它在客户计算机上运行即可。

下面将介绍不同种类活动代码的相关潜在弱点。

(1) Cookie。严格来说，Cookie 不是活动代码，而是一些数据文件，远程服务器能够存入或获取 Cookie。然而，由于 Cookie 的使用可能造成从一个客户到服务器的不期望的数据传送，所以它的一个缺点就是失去了机密性。

Cookie 是一个数据对象，可以存放在内存中(一次会话 Cookie)，也可以为将来使用而存

储在磁盘上(持久 Cookie)。Cookie 可以存储浏览器允许的与客户相关的任何内容,如用户按键、机器名称、连接详细内容(比如 IP 地址)、日期和类型等。在服务器命令控制下,浏览器将 Cookie 的内容发送给服务器。一次会话 Cookie 在关闭浏览器的时候被删除,而持久 Cookie 却可以保留一段预先设定的日期,可能几天甚至是未来的几年时间。

Cookie 为服务器提供了一个上下文。通过使用 Cookie,某些主页可以使用“欢迎回来,James Bond”这样的欢迎词来对用户表示欢迎,或者反映出用户的一些选择,比如“我们将把该订单上的货物邮寄到××大街××号,对吗?”但是,任何人只要拥有了某人的 Cookie,他在某些情形中就代表着这个人。这样,任何人只要窃听或获得了一个 Cookie,就可以冒充该 Cookie 的所有者。

Cookie 中究竟包含着关于用户的哪些信息呢? 尽管这些都是用户的信息,但通常用户不会知道 Cookie 里边到底是些什么内容,因为 Cookie 的内容使用一个来自服务器的密钥进行了加密。

因此,Cookie 会占用用户的磁盘空间,保存着一些用户不能看到但与用户相关的信息,能传递给服务器但用户不知道服务器什么时候想要它,服务器也不会通知用户。

(2) 脚本。客户可以通过执行服务器上的脚本来请求服务。通常情况是,网页浏览器显示一个页面,当用户通过浏览器与网站进行交互时,浏览器把用户输入的内容转化成一个预先定义好的脚本中需要的参数;然后,它发送这个脚本和参数给服务器执行。但是,所有通信都是通过 HTML 来进行的,服务器不能区分这些命令到底是来自一个浏览器上的用户完成一个主页后提交的,还是一个用户用手工写出来的。一些怀有恶意的用户可能会监视一个浏览器与服务器之间的通信,观察怎样改变一个网页条目可以影响浏览器发送的内容,以及其后服务器会做出何种反应。具备了这些知识,怀有恶意的用户就可以操纵服务器的活动。

这种操纵活动是十分容易的。程序员们通常不能预见到恶意的举动;通常程序员们认为用户都是合法的,会按照程序预先设定的操作规程来使用一个程序。正是由于这个原因,程序员们常常忽略过滤脚本参数,以保证用户的操作是合理的,而且执行起来也是安全的。一些脚本允许包含到任何文件中,或者允许执行任何命令。攻击者可以在一个字符串中看到这些文件或命令,并通过改变它们来做一些实验。

一种大家都很熟悉的针对网页服务器的攻击方式是 Escape 字符(Escape-Character)攻击。一种常用于网页服务器的脚本语言——公共网关接口(Common Gateway Interface,CGI),定义了一种不依赖于具体机器的方法来对通信数据编码。按照编码惯例,使用“%nn”来代表特殊的 ASCII 字符。例如,“%0A”(行结束)指示解释器将紧接着的一些字符当作一个新的命令。下面的命令是请求复制服务器的口令文件:

```
http://www.test.com/cgi-bin/query?%0a/bin/cat%20/etc/passwd
```

CGI 脚本也可以直接在服务器上启动一个动作。例如,如果攻击者观察到一个 CGI 脚本中包含着如下格式的一个字符串:

```
<!--#action arg1=value arg2=value -->
```

攻击者用以下字符串替代上述字符串后,就提交一个命令:

```
<!--#exec cmd="rm *"-->
```

这就会引起命令行解释器执行一个命令删除当前目录下的所有文件。

微软的动态服务器页面(Active Server Page,ASP)也具有像脚本一样的能力。这些页面指导浏览器怎样显示文件、维护上下文及与服务器交互。它们在浏览器端也可以被看到,所以任何存在于 ASP 代码中的编程漏洞都可用于侦察和攻击。

服务器永远不应相信来自客户端的任何东西,因为远程用户可以向服务器发送手工写出来的字符串,替换由服务器发送给客户端的善意的程序。正是由于有如此多的远程访问方式,所有这些例子证明了这样一点:如果用户允许其他人在其机器上运行程序,那这台机器就不会有绝对的安全保障。

(3) 活动代码。通过以下几个步骤就可以开始显示主页:产生文本,插入图像,并通过鼠标单击来获取新页。很快,人们就在他们的站点上使用了一些精心设计的内容:蹒跚学步的孩子在页面上跳舞、三维旋转的立方、图像时隐时现、颜色不断改变,以及显示总数等。其中,特别是涉及运动的小技巧显然会占用重要的计算能力,还需要花大量时间和通信从服务器上把它们下载到客户端。然而,通常情况下,客户自身有一个有能力却没有被充分利用的处理器,因此,无须担心活动代码占用客户端计算时间的问题。

为了充分利用处理器的能力,服务器可以下载一些代码到客户端去执行。这些可执行代码被称为活动代码。两种主要的活动代码是 Java 代码(Java code)和 Activex 控件(Activex control)。

① **Java 代码**。恶意的 Applet(Hostile Applet)是一种可以下载的 Java 代码,会对客户系统造成损害。由于 Applet 在下载以后失去了安全保护,而且通常以调用它的用户的权限运行,因此恶意的 Applet 会造成严重破坏。安全执行 Applet 的必要条件包括如下几个方面。

- 系统必须控制 Applet 对重要系统资源的访问,如文件系统、处理器、网络、用户显示和内部状态变量等。
- 编程语言必须通过阻止伪造内存指针和数组(缓冲区)溢出来保护内存。
- 在创建新对象的时候,系统必须通过清除内存内容来阻止对象的重用;在不再使用某些变量的时候,系统应该使用垃圾回收机制来收回所占用的内存。
- 系统必须控制 Applet 之间的通信,以及控制 Applet 通过系统调用对 Java 系统外的环境产生的影响。

② **Activex 控件**。微软公司针对 Java 技术的应对措施是 ActiveX 控制。使用 ActiveX 控件以后,任何类型的对象都可以下载到客户端。如果该客户有一个针对这种对象类型的阅读器或处理程序,就可以调用该阅读器来显示这个对象。例如,下载一个微软 Word 的.docx 文件就会调用系统上安装的 Word 程序来显示该文件。对于那些客户端没有相应处理程序的文件将会导致下载更多的其他代码。正是由于这个特点,从理论上来说,攻击者可以发明一种新的文件类型,如.bomb 的类型,就会导致那些毫无戒心的用户在下载一个包含.bomb 文件的主页时,也随同下载了可以执行.bomb 类型文件的代码。

为了阻止任意下载文件,微软公司使用了一种鉴别方案,在这种鉴别方案下,下载的代码是有密码标记的,而且在执行之前需要验证签名。但是,鉴别验证的仅仅是源代码,而不是它们的正确性或安全性。来自微软公司(Netscape 或任何其他生产商)的代码并不是绝对安全的,具有未知来源的代码可能会更安全,但也可能更不安全。事实证明,不论代码来自何处,用户都不能假设它到底有多好或者有多安全。况且,有些弱点还可以允许 ActiveX 绕过这种鉴别。