

第5章 防火墙技术

中国科学技术大学
曾凡平

billzeng@ustc.edu.cn

主要内容

5.1 防火墙概述

5.2 防火墙的功能和分类

- 5.2.1 防火墙的功能 5.2.2 防火墙的分类

5.3 包过滤防火墙

- 5.3.1 静态包过滤防火墙 5.3.2 动态包过滤防火墙

5.4 应用级网关防火墙

5.5 防火墙的典型部署

5.6 Linux防火墙的配置

5.7 Linux中的iptables防火墙

5.8 用iptables实现基于连接跟踪的状态防火墙

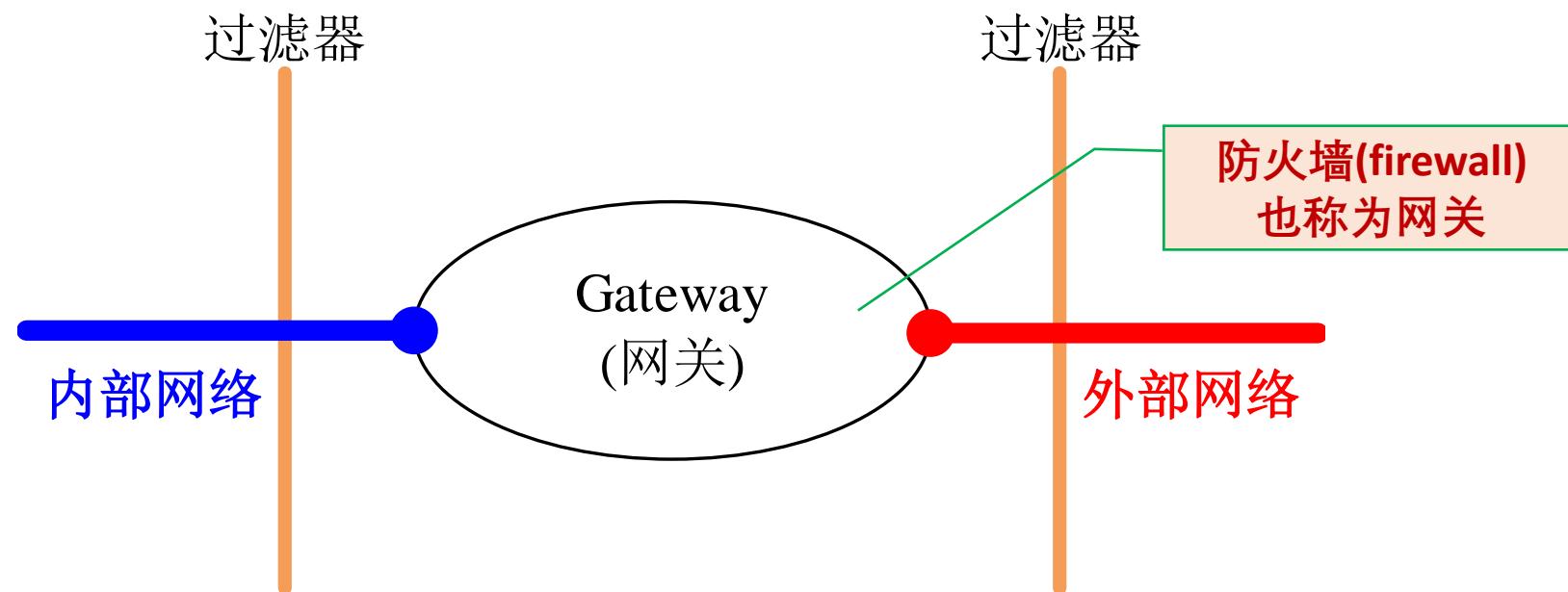
5.9 绕过防火墙的方法

5.1 防火墙概述

- **防火墙的定义：**防火墙是位于两个(或多个)网络之间执行访问控制的软件和硬件系统，它根据访问控制规则对进出网络的数据流进行过滤。
- 防火墙的概念起源于中世纪的城堡防卫系统，那时人们为了保护城堡的安全，在城堡的周围挖一条护城河，每一个进入城堡的人都要经过吊桥，并且还要接受城门守卫的检查。人们借鉴了这种防护思想，设计了一种网络安全防护系统，用于抵挡内部和外部网络的各种风险，这种系统被称为**防火墙(FireWall)**。
- **防火墙**是设置在网络之间（如：内部网络与外部网络）的一道屏障，**用于过滤和监视网络之间的数据流，实现企业的网络安全策略，实施网络访问控制规则。**

所有数据流都要经过防火墙

- 防火墙位于不同网络或网络安全域之间，从一个网络到另一个网络的所有数据流都要经过防火墙。
- 如果我们根据企业的安全策略设置合适的访问控制规则，就可以**允许、拒绝或丢弃数据流**，从而可以在一定程度上保护内部网络的安全。



对数据流的处理方式：允许、拒绝和丢弃

- 根据安全策略，防火墙对数据流的处理方式有三种：
 - ① 允许数据流通过（accepted）；
 - ② 拒绝数据流通过（rejected）：通知发送方；
 - ③ 将这些数据流丢弃（denied）：不通知发送方。
- 当数据流被拒绝时，防火墙要向发送者回复一条消息，用ICMP包告知数据源数据包被拒绝的原因，提示发送者该数据流已被拒绝。
- 当数据流被丢弃时，防火墙不会对这些数据包进行任何处理，也不会向发送者发送任何提示信息。丢弃数据包的做法加长了网络扫描所花费的时间，发送者只能等待回应直至通信超时。

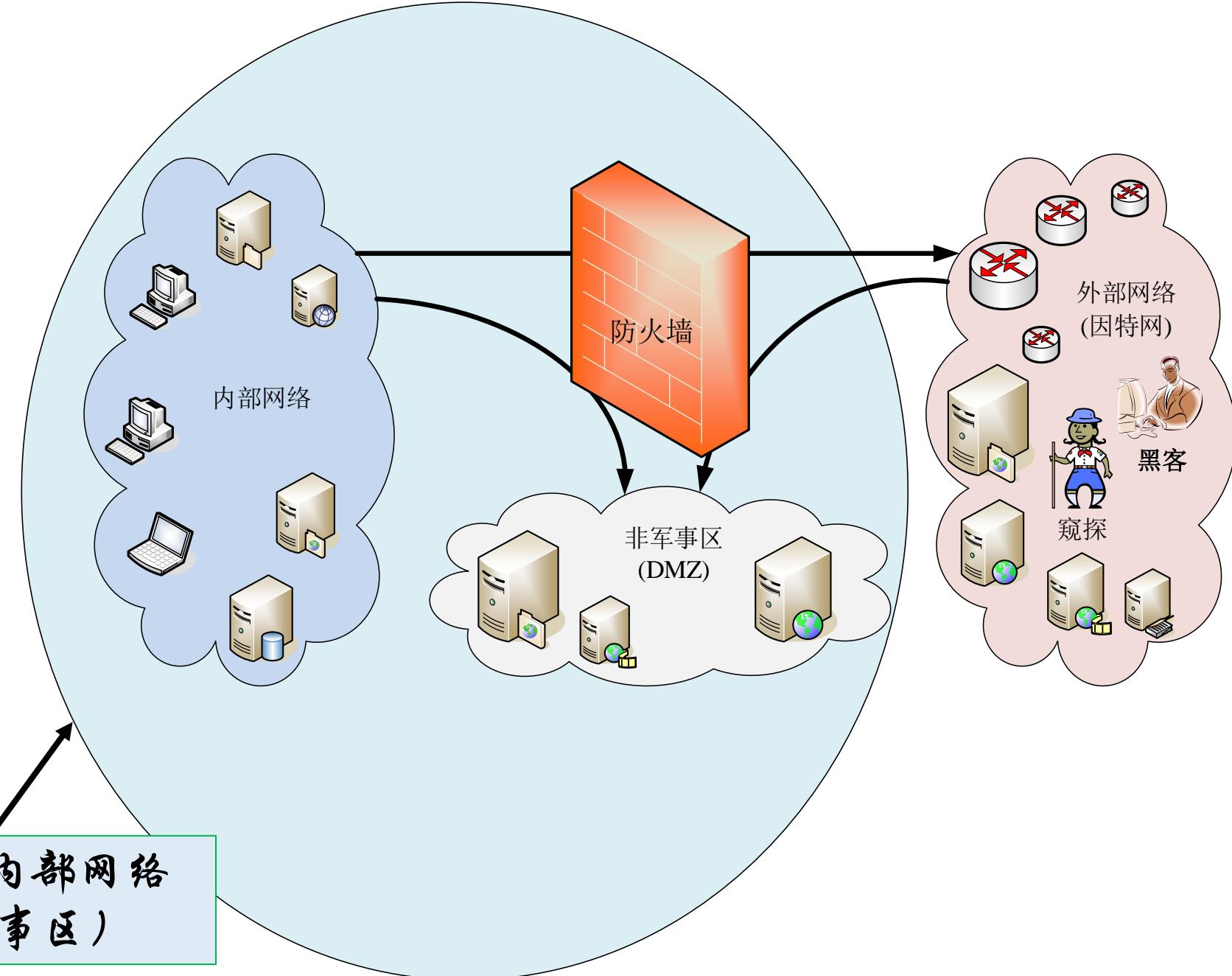


图5-1 防火墙示意图

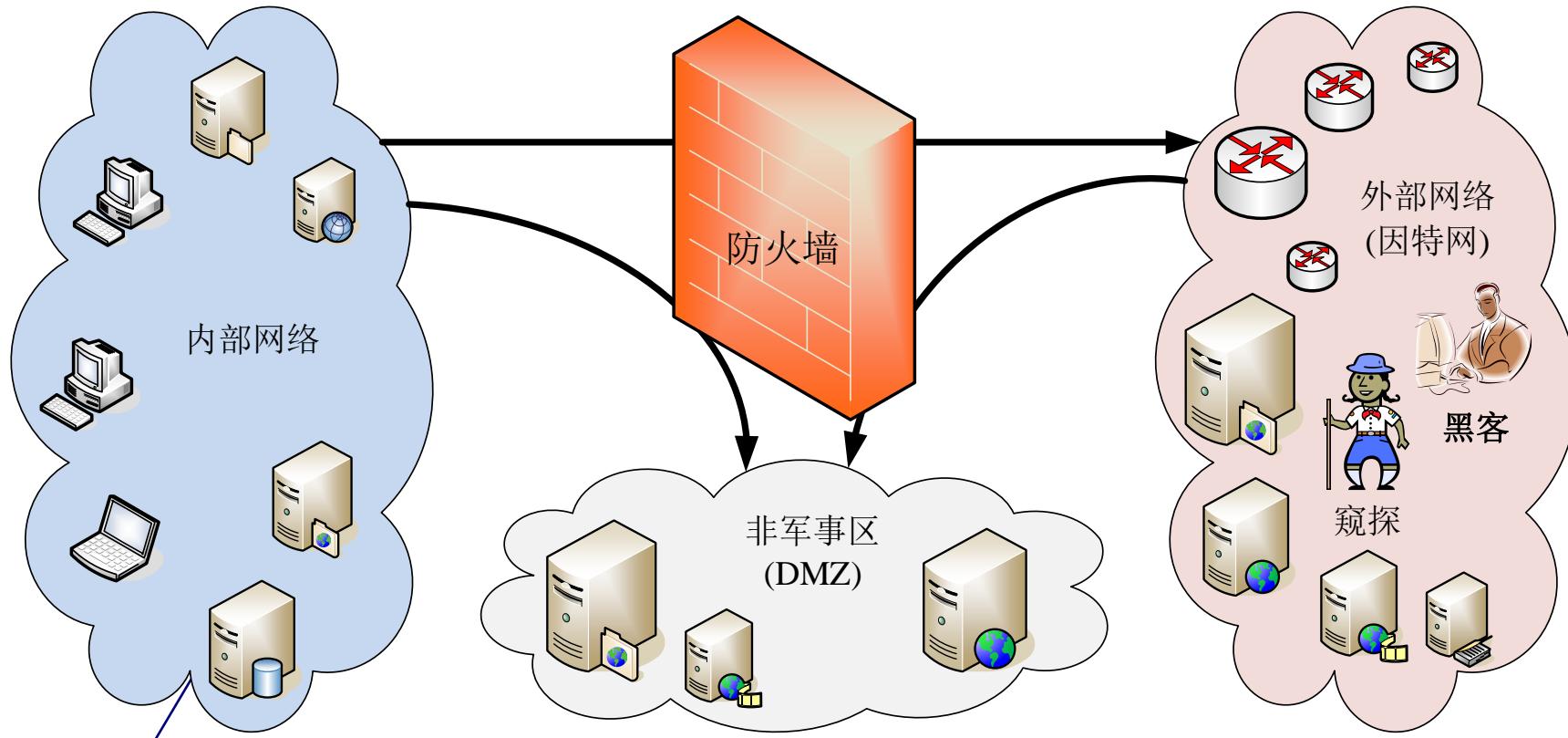
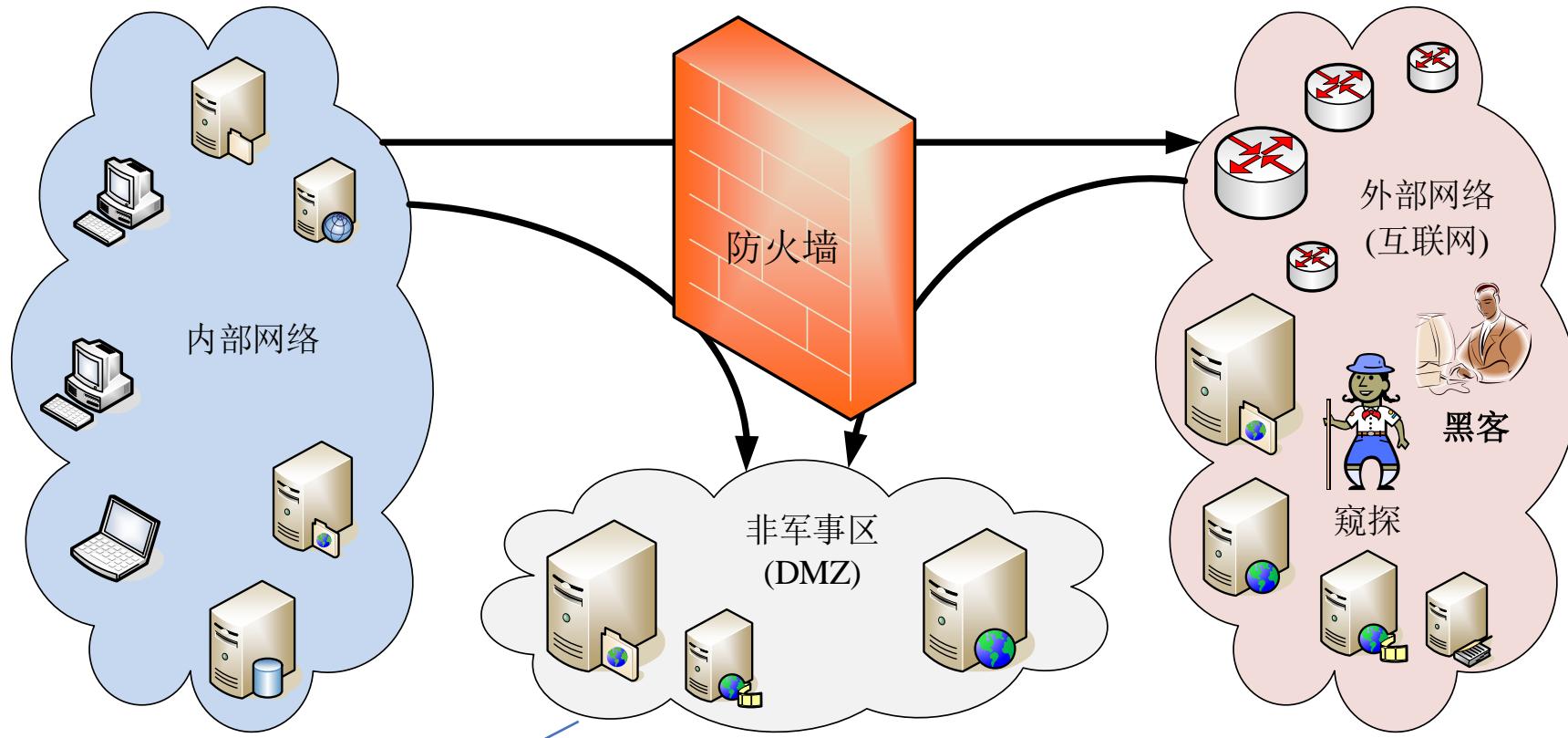


图5-1 防火墙示意图

内部网络一般是企业内部的局域网，其安全性是至关重要的，必须禁止外部网络的访问，同时只开放有限的对外部网络的访问权。



为了配置和管理方便，通常将内部网中需要向外部提供服务的服务器设置在单独的网段，这个网段被称为**非军事区(DMZ)**。

防火墙：限制网络访问的设备或软件

- 防火墙本质上是一种能够限制网络访问的设备或软件。
- 它可以是一个硬件的“盒子”，也可以是计算机和网络设备中的一个“软件”模块。许多网络设备均含有简单的防火墙功能，如路由器、无线基站、IP交换机等。现代操作系统中默认安装了软件防火墙：
 - Windows 系统和Linux系统均自带了软件防火墙，可以通过策略(或规则)定制相关功能。
- 对个人用户而言，一般用操作系统自带的防火墙或启用杀毒软件中的防火墙（腾讯电脑管家、360安全软件等均提供了个人防火墙功能）。
- 对于企业用户而言，购买专业的防火墙是比较好的选择。
 - 第一，防火墙厂商提供的接口会更多、更全；
 - 第二，过滤深度可以定制，甚至可以达到应用级的深度过滤；
 - 第三，可以获得厂商提供的技术支持服务。

5.2 防火墙的功能和分类

5.2.1 防火墙的功能

- 防火墙遵循的是一种允许或禁止业务来往的网络安全机制，也就是提供可控的过滤网络通信，**只允许授权的通信**。因此，**对数据和访问的控制、对网络活动的记录，是防火墙的基本功能。**

(1) 访问控制功能

- 这是**防火墙最基本和最重要的功能**，通过禁止或允许特定用户访问特定**资源，保护内部网络的资源和数据**。
- 防火墙配置在企业网络与Internet的连接处，是任何信息进出网络的必经之处，它保护的是整个企业网络，因此**可以集中执行强制性的信息安全策略**，可以根据安全策略的要求对网络数据进行不同深度的监测，允许或禁止数据的出入。这种**集中的强制访问控制简化了管理，提高了效率**。

内容控制功能和日志功能

(2) 内容控制功能

- 防火墙可以防止非法用户进入内部网络，也能禁止内网用户访问外网的不安全服务（比如恶意网站），这样就能有效地防止邮件炸弹、蠕虫病毒、宏病毒等攻击。
- 如果发现某个服务存在安全漏洞，则可以用防火墙关闭相应的服务端口号，从而禁用了不安全的服务。如果在应用层进行过滤，还可以过滤不良信息传入内网。比如，过滤色情、暴力信息的传播。

(3) 日志功能

- 记录通过防火墙的信息内容和活动。
- 日志是对一些可能的攻击进行分析和防范的十分重要的信息。另外，防火墙系统也能够对正常的网络使用情况做出统计，通过对统计结果的分析，可以使网络资源得到更好的使用。

告警和集中管理功能

(4) 对网络攻击的检测和告警

- 当发生可疑动作时，防火墙能进行适当的报警，并提供网络是否受到监测和攻击的详细信息。

(5) 集中管理功能

- 针对不同的网络情况和安全需要，指定不同的安全策略，在防火墙上集中实施，使用中还可能根据情况改变安全策略。
- 防火墙应该是易于集中管理的，便于管理员方便地实施安全策略。

其他功能

- 流量控制、网络地址转换(NAT)、虚拟专用网(VPN)等。

5.2.2 防火墙的分类

(1) 按防火墙的使用范围分类

- 可分为**个人防火墙**和**网络防火墙**

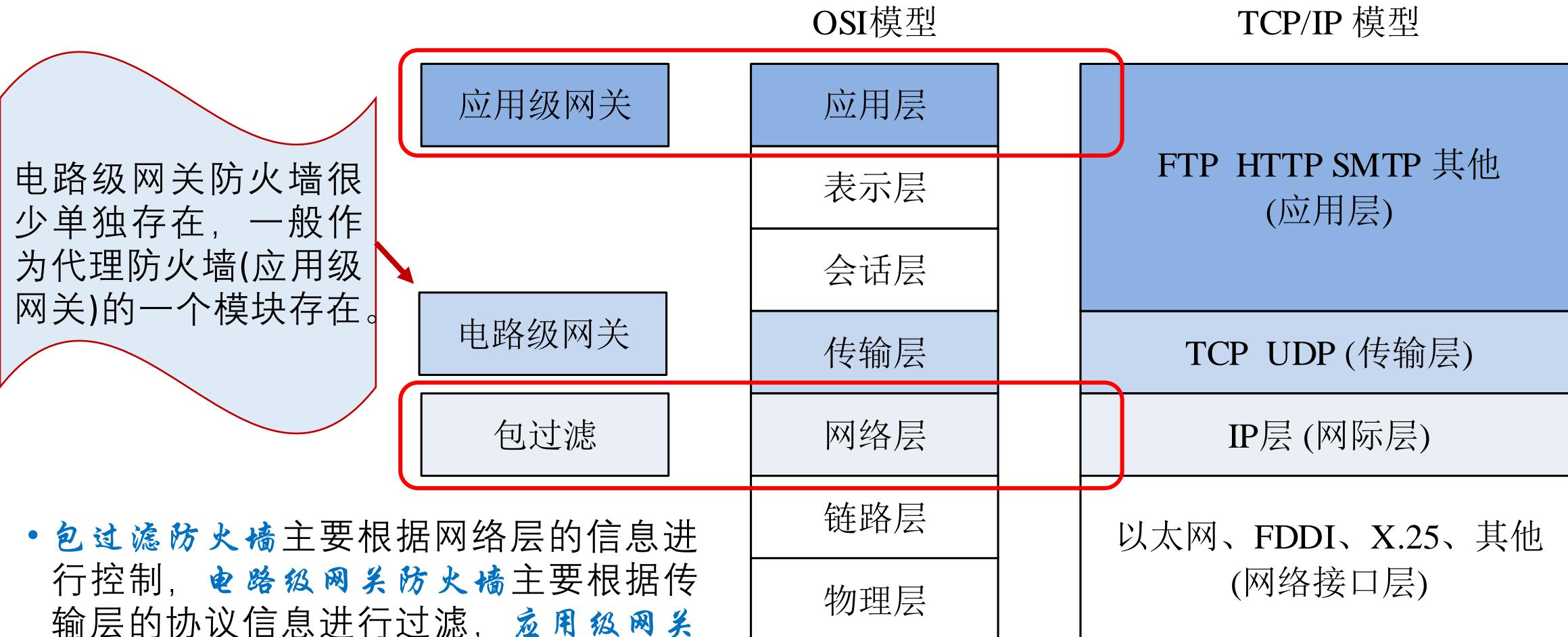
个人防火墙:

保护一台计算机，一般提供简单的**包过滤**功能，通常内置在操作系统或随杀毒软件提供。

网络防火墙:

保护一个网络中的所有主机，布置在内网与外网的连接处，通常由路由器提供或使用专业的防火墙。

(2) 根据防火墙在网络协议栈中的过滤层次分类



- 包过滤防火墙主要根据网络层的信息进行控制，电路级网关防火墙主要根据传输层的协议信息进行过滤，应用级网关防火墙主要根据应用层协议的信息进行过滤。

图5-2 防火墙示意图

(3) 按防火墙的**发展沿革(历史)**分类

按发展历史，防火墙可分为五代：

- **第一代防火墙**

- 第一代防火墙始于1985年前后，它几乎与路由器同时出现，由Cisco的IOS软件公司研制。这一代防火墙称为**包过滤防火墙**。
- 直到1988年，DEC公司的Jeff Mogul根据自己的研究，才发表了第一篇描述有关包过滤防火墙过滤过程的文章。

- **第二代防火墙**

- 在1989—1990年前后，AT&T贝尔实验室的Dave Presotto和Howard Tfickey率先提出了**基于电路中继的第二代防火墙**结构，此类防火墙被称为**电路级网关防火墙**。但是，他们既没有发表描述这一结构的任何文章，也没有发布基于这一结构的任何产品。

第三代防火墙、第四代防火墙

• 第三代防火墙

- 第三代防火墙结构是在20世纪80年代末和20世纪90年代初由Purdue University的Gene Spafford、AT&T贝尔实验室的Bill Cheswick和Marcus Ranum分别研究和开发。这一代防火墙被称为**应用级网关防火墙**。此类防火墙采用了**在堡垒主机运行代理服务**的结构。根据这一研究成果，DEC公司推出了第一个商用产品SEAL。

• 第四代防火墙

- 大约在1991年，Bill Cheswick和Steve Bellovin开始了对**动态包过滤防火墙**的研究。1992年，在USC信息科学学院工作的Bob Braden和Annette DeSchon开始研究用于“Visas”系统的动态包过滤防火墙，后来它演变为目前的**状态检测防火墙**。
- **1994年**，以色列的Check Point Software公司推出了基于**第四代结构**的第一个商用产品。

第五代防火墙

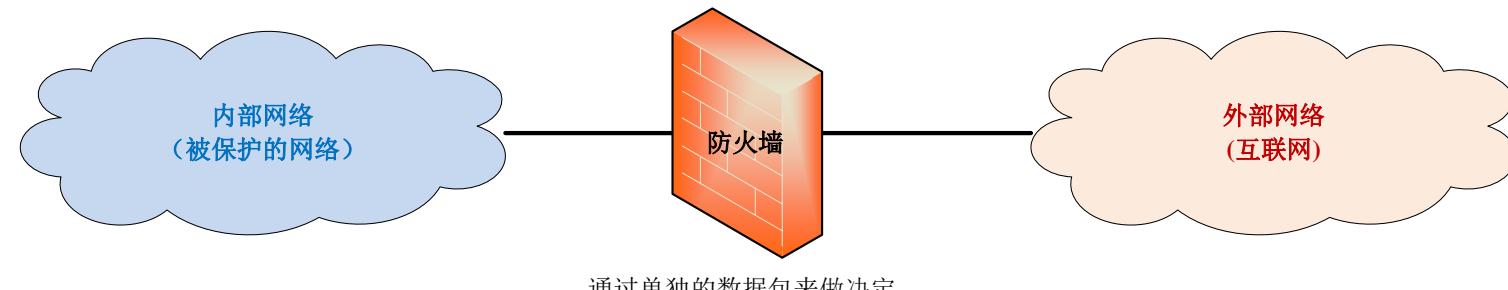
- 关于第五代防火墙，目前尚未有统一的说法，关键在于目前还没有出现获得广泛认可的新技术。
 - ① 一种观点认为，在1996年由Global Internet Software Group公司的首席科学家Scott Wiegel开始启动的**内核代理结构**(Kernel Proxy Architecture)研究计划**属于第五代防火墙**；
 - ② 还有一种观点认为，在1998年由NAI公司推出的**自适应代理**(**Adaptive Proxy**)技术给代理类型的防火墙赋予了全新的意义，可以称之为**第五代防火墙**。
-
- 现代防火墙**属于第5代**，向高速度、高吞吐量和多功能融合方向发展。

5.3 包过滤防火墙

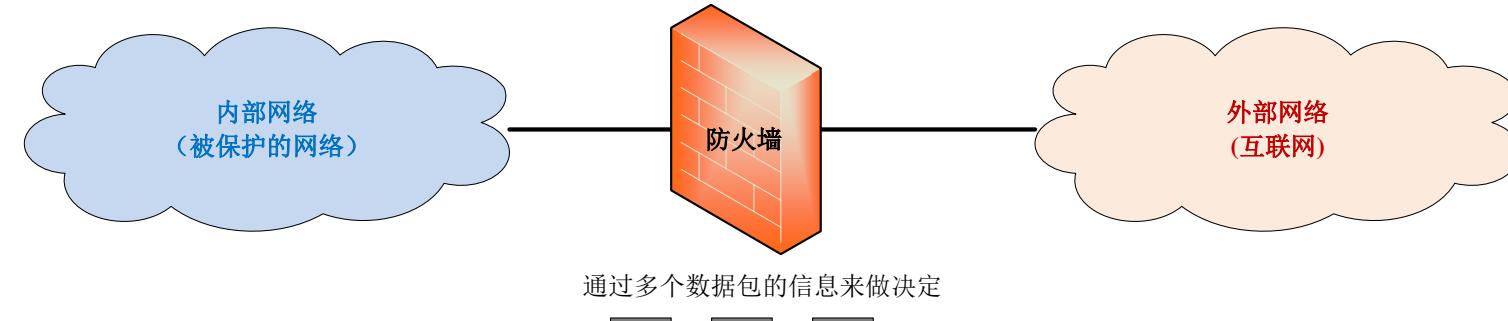
- 包过滤防火墙也称为**分组过滤**防火墙，是最早出现的防火墙，几乎与路由器同时出现，最初是作为路由器的一个过滤模块来实现的。
- 包过滤防火墙**工作在IP层（网络层）**，也用到了传输层的协议端口号等信息。根据访问控制策略的实现机制的不同，又可以分为**静态包过滤**和**动态包过滤**。
- 注：
 - 静态包过滤防火墙也称为**数据包过滤器（防火墙）**
 - 动态包过滤防火墙也称为**状态（检测）防火墙**

数据包过滤器与状态防火墙

- **数据包过滤器通过数据包的头部信息来判断是接受还是拒绝数据包**，它并不查看数据包载荷中的应用数据。
- 这种防火墙检查流经它的每个数据包，根据数据包本身所带的信息决定它的去留，而不用参考其他数据包的内容。
- **状态防火墙通过对流经的数据包的分析查找通信中的数据流，根据数据流的信息来帮助判断是否让数据包通行。**
- 数据流提供了数据包的上下文。状态防火墙有时还会检测一些常用协议的应用数据（虽然可以检测的数据量是有限的），通过这些数据来识别和跟踪相关的数据流。



(a) 数据包过滤器



(b) 状态防火墙

包过滤防火墙

- 网络管理员首先根据企业的**安全策略**定义一组**访问控制规则**，然后防火墙在内存中建立一张与访问控制规则对应的**访问控制列表**。
- 对于每个数据包，如果在访问控制列表中有对应的项，则防火墙按规则的要求允许或拒绝数据包的通过，否则应用“**默认规则**”。
- “**默认规则**”有两种，即“**默认丢弃**”或“**默认允许**”。
 - “**默认丢弃**”是指如果没有对应的规则，则丢弃数据包；
 - “**默认允许**”是指如果没有对应的规则，则允许数据包通过。
 - 显然，“**默认丢弃**”更有利于企业网的安全防护。

5.3.1 静态包过滤防火墙

- 静态包过滤防火墙的**访问控制列表**在运行过程中是**不会动态变化的**，其过滤规则只利用了IP与TCP/UDP报头中的几个字段，只适合一些对安全要求不高的场合，其访问控制规则的配置比较复杂，**对于某些需要打开动态端口的应用，很难定义合适的规则。**
- 对数据包的处理过程：
 - (1)接收每个到达的数据包。
 - (2)对数据包**按序匹配过滤规则**，对数据包的IP头和传输字段内容进行检查，执行相应的动作：
 - 如果数据包的信息与一条规则匹配，则根据该规则确定是转发还是丢弃该数据包。然后转(1)
 - 如果没有规则与数据包的信息匹配，则对数据包施加默认规则。然后转(1)

静态包过滤防火墙的优点

- 静态包过滤防火墙仅检查当前的数据包，**是否允许通过的判决仅依赖于当前数据包**的内容，检查的内容包括如下几部分：
 - ①源IP地址；②目的IP地址；③应用或协议号；
 - ④源端口号；⑤目的端口号。
- 因此，静态包过滤防火墙**对数据包的检测是孤立的、无状态的**。
- 静态包过滤防火墙的优点如下：
 - (1)对网络性能的影响较小；**
 - (2)成本较低；**
 - (3)对用户透明。**

静态包过滤防火墙的缺点

(1) 安全性较低

(2) 缺少状态感知能力

(3) 容易遭受IP欺骗攻击

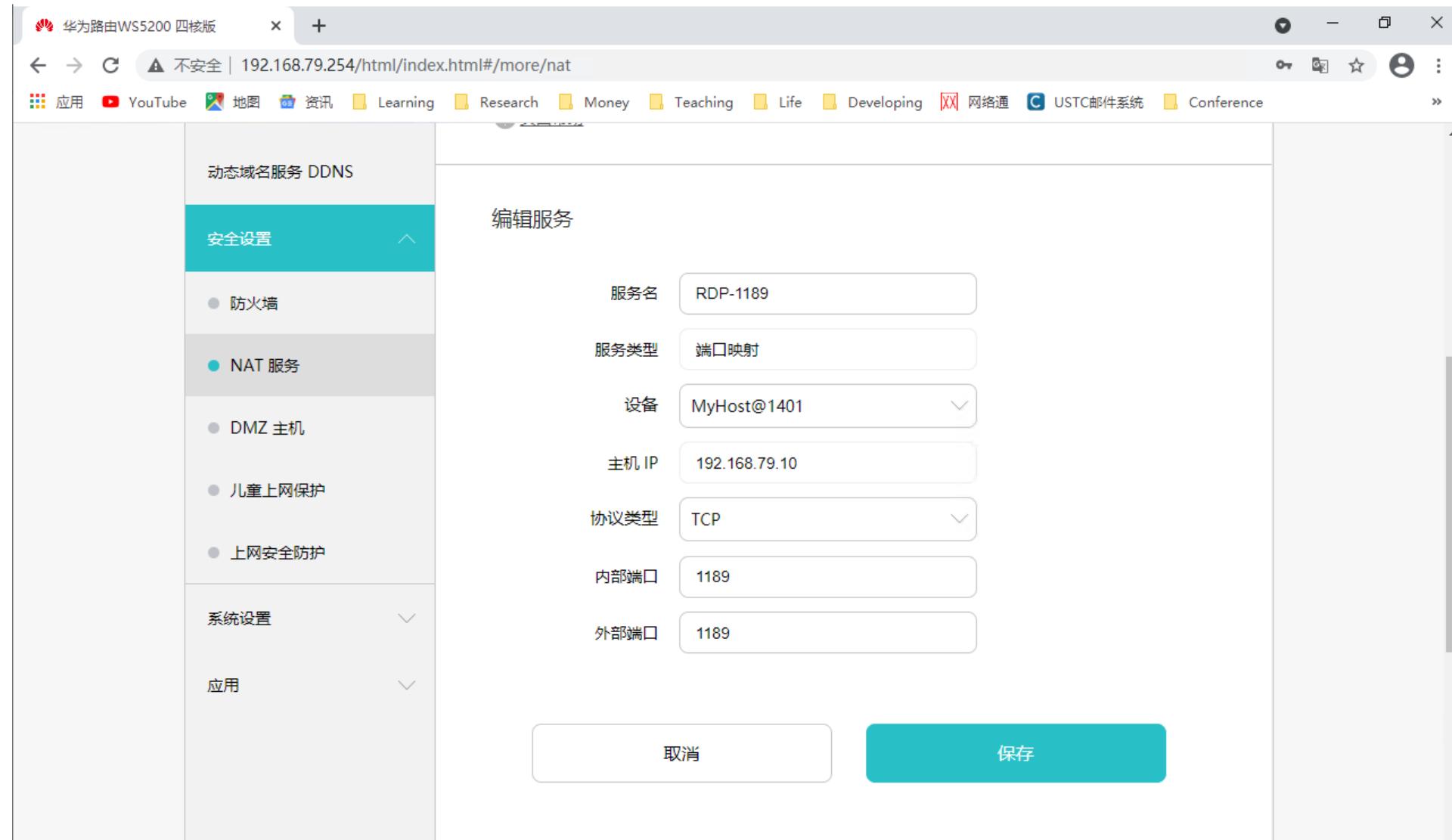
(4) 创建访问控制规则比较困难

- 需要认真地分析和研究一个组织机构的安全策略；
- 必须考虑访问控制规则的先后次序。

静态包过滤防火墙实例：Windows11基于网络协议端口的过滤



包过滤防火墙的实例：无线路由器的防火墙



华为路由WS5200 四核版 (端口开放的功能通过NAT提供)

5.3.2 动态包过滤防火墙

- 由于静态包过滤防火墙的访问控制表是固定的，这就很难应用于需要打开动态端口的一些网络服务，比如P2P协议。
- 由于事先不知道需要打开哪些端口，这种情况下如果必须采用原始的静态包过滤技术，就须允许很大范围端口号的流量通过，否则服务器将难以正常运行，这就带来了风险。
- 解决这一问题的方法是使用动态包过滤技术，它可以根据网络当前的状态检查数据包，即根据当前所交换的信息动态调整过滤规则表。
- 有些包过滤防火墙提供**基于服务（程序）的过滤功能**，通过**检测数据包与服务程序的关联**，以判断是否允许数据包通过。这种防火墙**可以看作是动态包过滤防火墙**。比如：Windows系统提供了基于程序的包过滤功能。

动态包过滤防火墙的工作原理

- 状态检测技术监视和跟踪每一个有效连接的状态，并根据这些信息决定网络数据包是否能通过防火墙。
- 它在协议底层截取数据包，然后分析这些数据包，并将当前数据包和状态信息与前一时刻的数据包和状态信息进行比较，从而得到该数据包的控制信息，来达到保护网络安全的目的。
- 状态检测防火墙有时还会检测一些常用协议的应用数据（虽然可以检测的数据量是有限的），通过这些数据来识别和跟踪相关的数据流。
- 状态检测引擎支持多种协议和应用程序，并可以很容易地实现应用和服务的扩充。

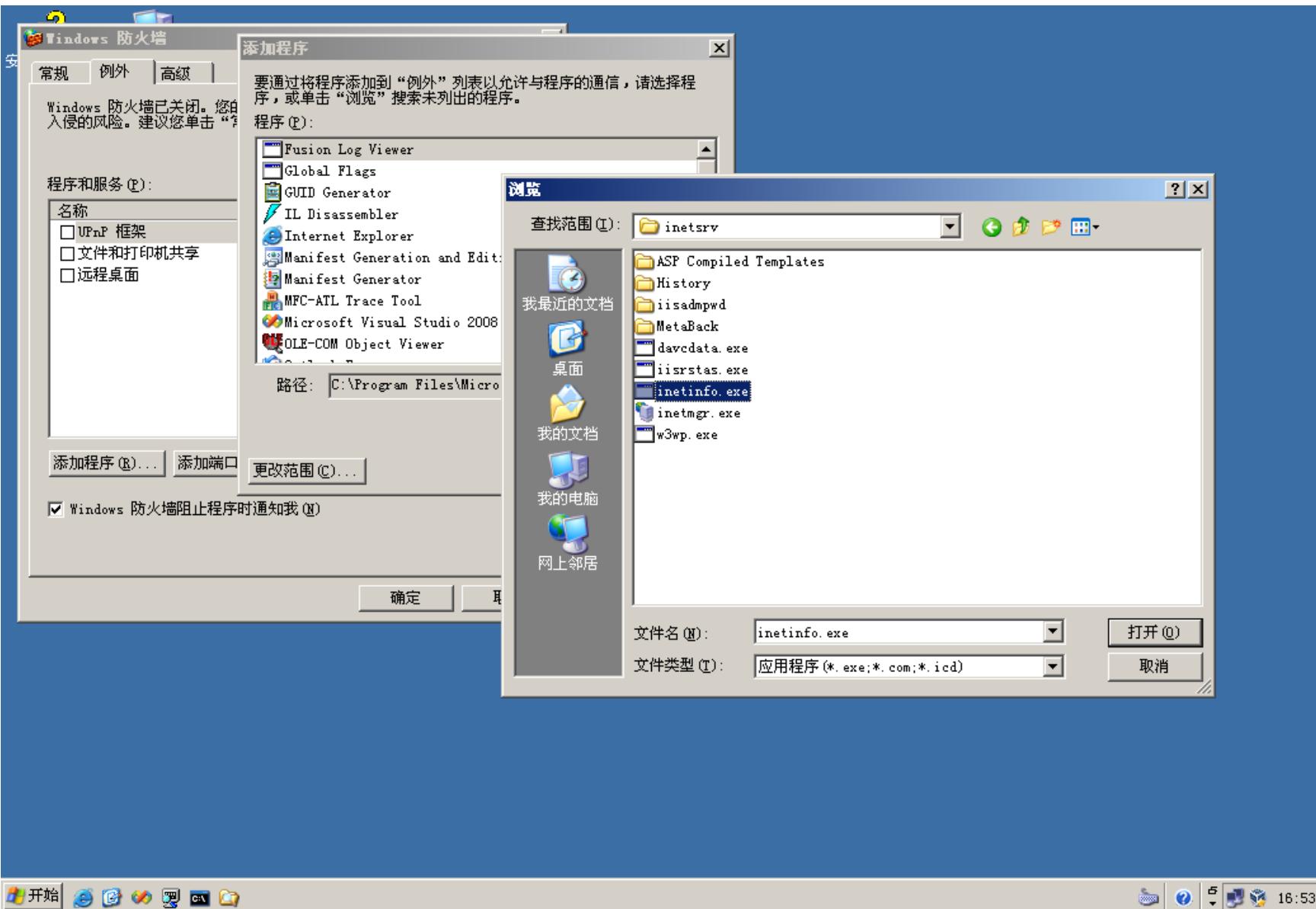
实例：状态检测防火墙的状态表

源地址	源端口	目的地址	目的端口	连接状态
192.168.1.100	1030	210.9.88.29	80	已建立
192.168.1.102	1031	216.32.42.123	80	已建立
192.168.1.101	1033	173.66.32.122	25	已建立
192.168.1.106	1035	177.231.32.12	79	已建立
223.43.21.231	1990	192.168.1.6	80	已建立
219.22.123.32	2112	192.168.1.6	80	已建立
210.99.212.18	3321	192.168.1.6	80	已建立
24.102.32.23	1025	192.168.1.6	80	已建立
223.212.21.2	1046	192.168.1.6	80	已建立

动态包过滤的例子：Windows11基于程序的过滤



动态包过滤的例子：Windows 2003基于程序的过滤



动态包过滤防火墙的优点

(1) 高安全性

- 由于具有“状态感知”能力，所以防火墙可以区分连接的发起方与接收方，也可以通过检查数据包的状态阻断一些攻击行为。

(2) 高性能

- 由于防火墙在连接建立后保存了连接状态，当后续数据包通过防火墙时，不再需要烦琐的规则匹配过程

(3) 伸缩性和易扩展性

- 基于流的过滤和基于程序的过滤，对新的应用不需要修改防火墙代码。

(4) 针对性

- 它能对特定类型的数据包中的数据进行检测。

(5) 应用范围广

- 支持多种协议（TCP、UDP、ICMP、HTTP）的状态跟踪

动态包过滤防火墙的缺点

- (1) 由于没有对数据包的净荷部分进行过滤，因此仍然只具有较低的安全性。
- (2) 容易遭受IP地址欺骗攻击。
- (3) 难于创建规则，管理员创建规则时必须要考虑规则的先后次序。
- (4) 如果动态包过滤防火墙在连接建立时没有遵循RFC建议的三步握手协议，就会引入额外的风险：如果防火墙在连接建立时仅使用两次握手，很可能导致防火墙在DoS/DDoS攻击时因耗尽所有资源而停止响应。

5.4 应用级网关防火墙

- 应用级网关防火墙**也称为代理防火墙**，是实现内容过滤的主要技术之一。
- 应用级网关防火墙针对每一种应用软件，均由对应的代理软件对其实现载荷进行分析和过滤。因此，**代理是特定于应用的**。
- 目前常用的有http代理、ftp代理、email代理等。
- 应用代理包括客户代理和服务器代理，如图5-5所示。

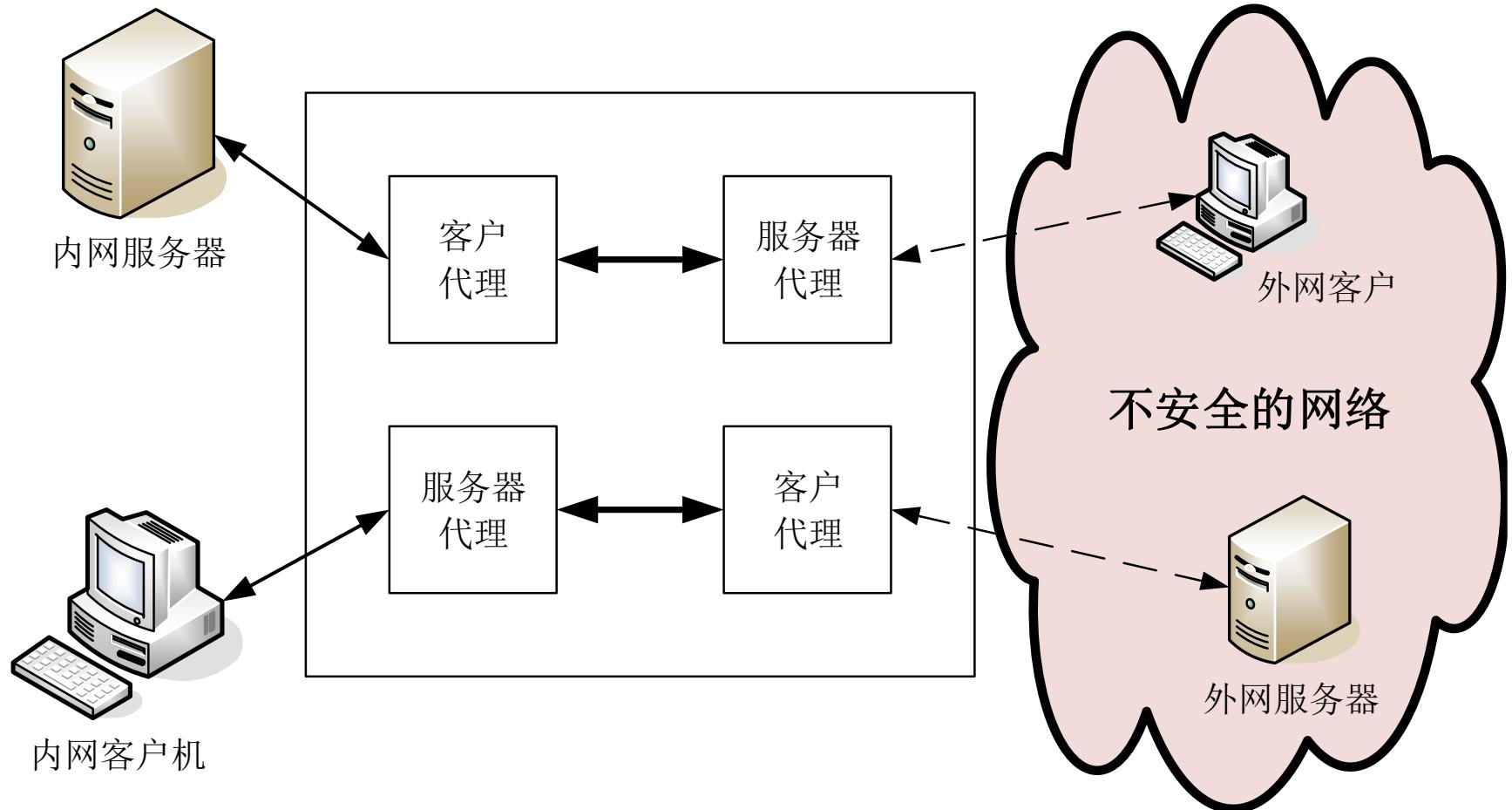


图5 - 5 代理防火墙的逻辑结构

应用级网关

- 应用级网关截获进出网络的数据包，对数据包的内容进行检查，如果符合所制定的安全规则，则允许数据通过；否则根据安全策略的要求进行处理。比如：可以直接丢弃数据包，也可以删除数据包的不良内容，将改变的数据包传递到通信的另一端。
- 由于应用代理避免了服务器和客户机之间的直接连接，其安全性是最高的。
- 虽然应用级网关防火墙具有很高的安全性，但是它有一个固有的缺点，那就是**缺乏透明性，即你所看到的未必是原来的信息**。
- 此外，缺乏对新应用、新协议的支持也成了制约应用级网关发展的主要障碍。由于各种应用软件的升级很快，应用代理要跟上应用软件的升级速度是较难的，这就制约了代理防火墙的广泛使用。

应用级网关的主要优点

(1) 在已有的安全模型中安全性较高。

- 应用级网关防火墙完全可以对服务(如HTTP、FTP等)的命令字过滤，也可以实现内容过滤，甚至可以进行病毒的过滤。

(2) 具有强大的认证功能。

- 由于应用级网关在应用层实现认证，因此它可以实现的认证方式比电路级网关要丰富得多。

(3) 具有超强的日志功能。

- 应用级网关可以记录用户通过HTTP访问了哪些网站页面、通过FTP上传或下载了什么文件、通过SMTP给谁发送了邮件，甚至邮件的主题、附件等信息，都可以作为日志的内容。

(4) 应用级网关防火墙的规则配置比较简单。

- 应用代理必须针对不同的协议实现过滤，管理员在配置应用级网关时关注的重点就是应用服务，而不必像配置包过滤防火墙一样还要考虑规则顺序的问题。

应用级网关的主要缺点

(1) 灵活性很差。

- 对每一种应用都需要设置一个代理。
- 在实际工作中，应用级网关防火墙中集成了电路级网关或包过滤防火墙，以满足人们对灵活性的需求。

(2) 配置烦琐，增加了管理员的工作量。

- 各种应用代理的设置方法不同。
- 当网络规模达到一定程度的时候，其工作量很大。

(3) 性能不高，有可能成为网络的瓶颈。

- 目前，应用级网关的性能依然远远无法满足大型网络的需求，一旦超负荷，就有可能发生停机，从而导致整个网络中断。

5.5 防火墙的典型部署

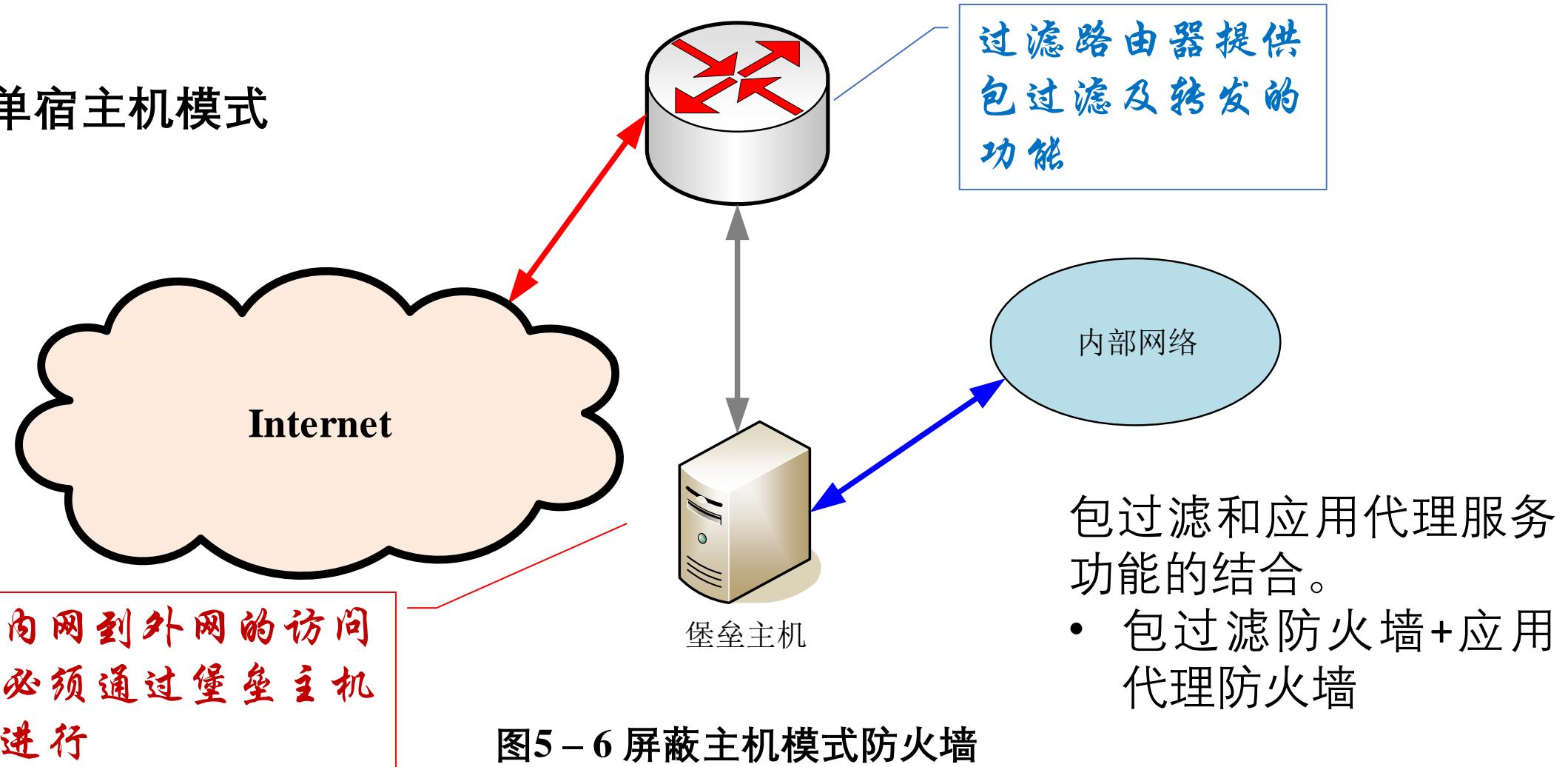
- 防火墙有三种典型的部署模式：屏蔽主机模式、双宿/多宿主机模式和屏蔽子网模式。在这些部署中，堡垒主机都承担了重要的作用。
- **堡垒主机(Bastion Host)**是一种配置了较为全面的安全防范措施的网络上的计算机，它为网络间的通信提供了一个阻塞点。
- 通常堡垒主机可以用作**应用级和电路级网关**的平台，是一个组织机构**网络安全的中心主机**。

堡垒主机(Bastion Host)的特征

- (1) 堡垒主机硬件平台运行较为安全的操作系统，成为可信任的系统。
- (2) 只有网络管理员认为必要的服务(代理和用户认证等)才会安装在堡垒主机上。
- (3) 当允许一个用户访问代理服务时，堡垒主机可能会要求进行额外认证。另外，每一个代理服务都可能需要相应的鉴别机制(Authentication)。
- (4) 每一个代理都只能支持标准应用服务命令集中的一个子集。
- (5) 每一个代理只允许访问指定主机的通信，支持对通信进行详细的审计。
- (6) 每一个代理模块都是一个为网络安全设计的一个很小的软件包。
- (7) 代理之间相互独立。
- (8) 堡垒主机是一个组织机构网络安全的中心主机，对它必须进行完善的防御。

5.5.1 屏蔽主机模式防火墙

也称为单宿主机模式



屏蔽主机模式防火墙

- **堡垒主机承担了身份鉴别和代理服务的功能。**这样的配置比单独使用包过滤防火墙或应用层防火墙更加安全。
 - ① 首先，这种配置能够实现数据包级过滤和应用级过滤，在定义安全策略时有相当的灵活性。
 - ② 其次，在入侵者威胁到内部网络的安全以前，必须能够“穿透”两个独立的系统（包过滤路由器和堡垒主机）。同时，这种配置在对Internet进行直接访问时，有更大的灵活性。例如，内部网络中有一个公共信息服务器，如Web服务器（在高级别的安全中是不需要的），这时，可以配置路由器允许网络流量在信息服务器和Internet之间传输。
- 单宿主机模式存在一个**缺陷：一旦过滤路由器遭到破坏，堡垒主机就可能被越过，使得内部网络完全暴露。**

5.5.2 双宿/多宿主机模式防火墙

- 双宿/多宿主机模式防火墙(Dual-homed/Multi-Homed Firewall)，又称为双宿 / 多宿网关防火墙。它是一种拥有两个或多个连接到不同网络上的网络接口的防火墙。通常用一台装有两块或多块网卡的**堡垒主机**作为防火墙，每块网卡各自与受保护网络或外部网连接。
- 其体系结构如图5-7所示。

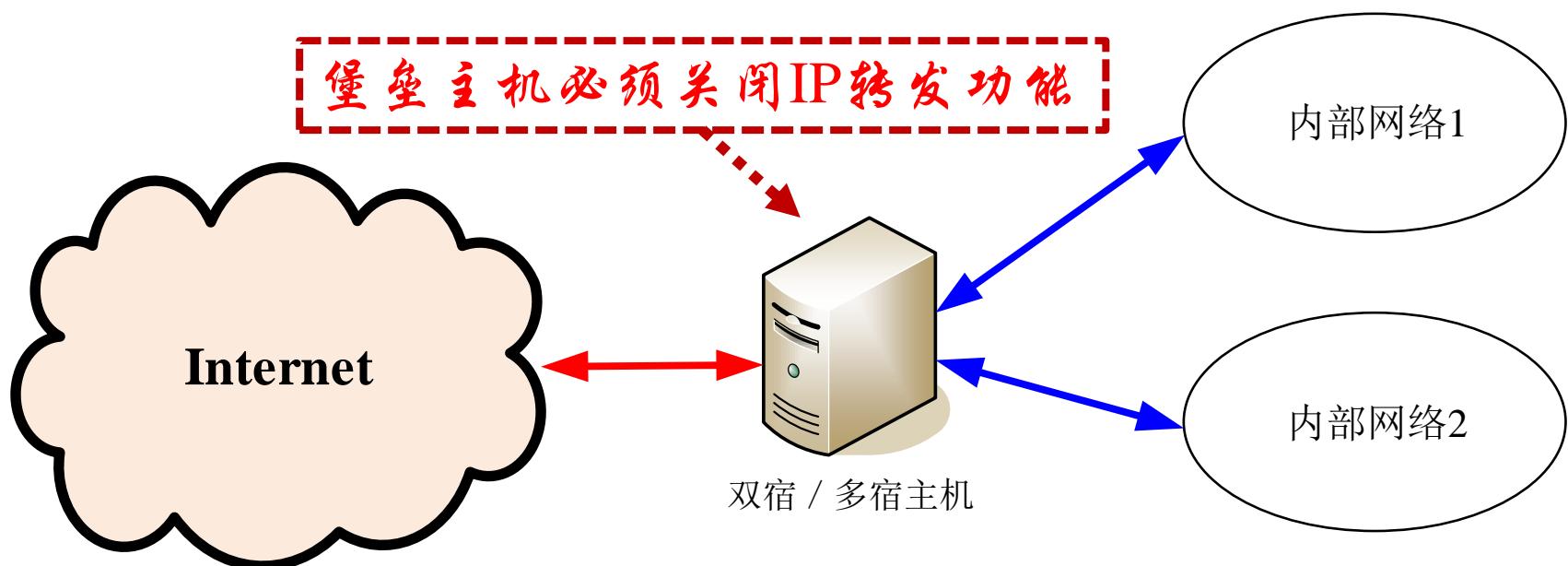


图5 - 7 双宿/多宿主机模式防火墙

双宿/多宿主机模式防火墙

- 该模式下，**堡垒主机必须关闭IP转发功能**，其网关功能是通过提供代理服务而不是通过IP转发来实现的。显然只有特定类型的协议请求才能被代理服务处理。于是，网关采用了“缺省拒绝”策略以得到很高的安全性。
- 这种体系结构的防火墙简单明了，易于实现，成本低，能够为内外网提供检测、认证、日志等功能。
- 但是这种结构也存在**弱点**，一旦黑客侵入了**堡垒主机并打开其IP转发功能，则任何网上用户均可随意访问内部网络**。因此，双宿/多宿网关防火墙对不可信任的外部主机的访问必须进行严格的身份验证。

5.5.3 屏蔽子网模式防火墙

- 与前面几种配置模式相比，屏蔽子网模式防火墙(Screened Subnet Mode Firewall)是最为安全的一种配置模式。
- 它采用了两个包过滤路由器：一个位于堡垒主机和外部网络(Internet)之间；另一个位于堡垒主机和内部网络之间。
- 该配置模式在内部网与外部网络之间建立了一个被隔离的子网，其体系结构如图5-8所示。

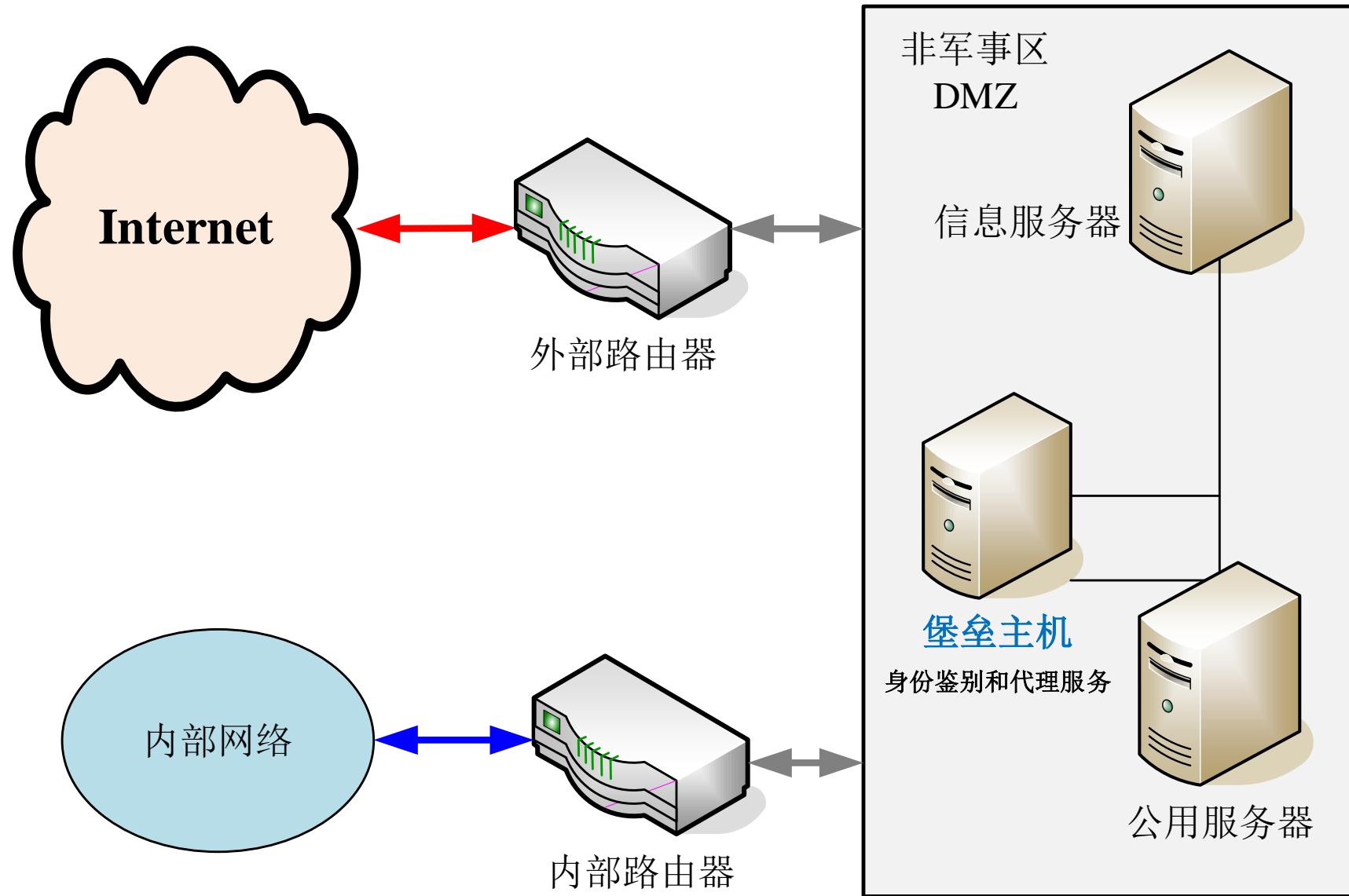


图5 - 8 屏蔽子网模式防火墙

屏蔽子网模式防火墙

- 周边防御网络是位于内部网络与外部网络之间的一个安全子网，分别和内外两个路由器相连。这个子网被定义为“**非军事区**”(**demilitarized zone, DMZ**)网络，这一网络所受到的威胁不会影响到内部网络。网络管理员可以将堡垒主机、Web服务器、E-mail服务器等公用服务器放在非军事区网络中，将重要的数据放在内部网服务器上。
- 内部网络和外部网络均可访问屏蔽子网，但禁止它们穿过屏蔽子网通信。在这一配置中，内网增加了一台内部包过滤路由器，该路由器与外部路由器的过滤规则完全不同，它只允许源于堡垒主机的数据包进入。
- 这种防火墙安全性好，但成本高。在这一配置中，即使外部路由器和堡垒主机被入侵者控制，内部网络仍受到内部包过滤路由器的保护。

5.6 Linux防火墙的配置

- Linux系统免费且源代码开源，在构建企业级的信息系统中得到了极为广泛的应用，尤其是服务器大多使用Linux系统。
- Linux系统下的防火墙最初用**iptables**进行配置，比较复杂，对管理员的要求较高。**iptables**在5.7节进行简要介绍。
- 为了提高防火墙的易用性，使之适合普通用户配置个人防火墙，近年来Linux系统的各个发行版均提供了优秀的配置工具，以简化防火墙的配置。
- 本节以Ubuntu linux为例进行简要说明。

Ubuntu linux系统的防火墙

以管理员权限(**sudo**)用**ufw**命令配置防火墙

查看帮助: `man ufw`

启动|关闭防火墙: `ufw enable | disable`

开放(关闭)某个端口: `ufw allow (deny) port`

开放(关闭)某个端口: `ufw allow (deny) 22`

开放(关闭)某服务: `ufw allow (deny) service`

开放(关闭)某个服务: `ufw allow (deny) ssh`

查看当前的防火墙状态: `ufw status | ufw status numbered`

删除第4条规则: `ufw delete 4`

实例：ubuntu Linux系统的防火墙

查看防火墙的当前状态

- i@ns64UB22:~\$ **sudo ufw status numbered**

Status: active

To	Action	From
--	-----	-----
[1] 23/tcp	ALLOW IN	Anywhere
[2] 22/tcp	ALLOW IN	Anywhere
[3] 23/tcp (v6)	ALLOW IN	Anywhere (v6)
[4] 22/tcp (v6)	ALLOW IN	Anywhere (v6)

删除第4条规则

- i@ns64UB22:~\$ **sudo ufw delete 4**

Deleting:

allow 22/tcp

Proceed with operation (y|n)? y

Rule deleted (v6)

5.7 Linux中的iptables防火墙

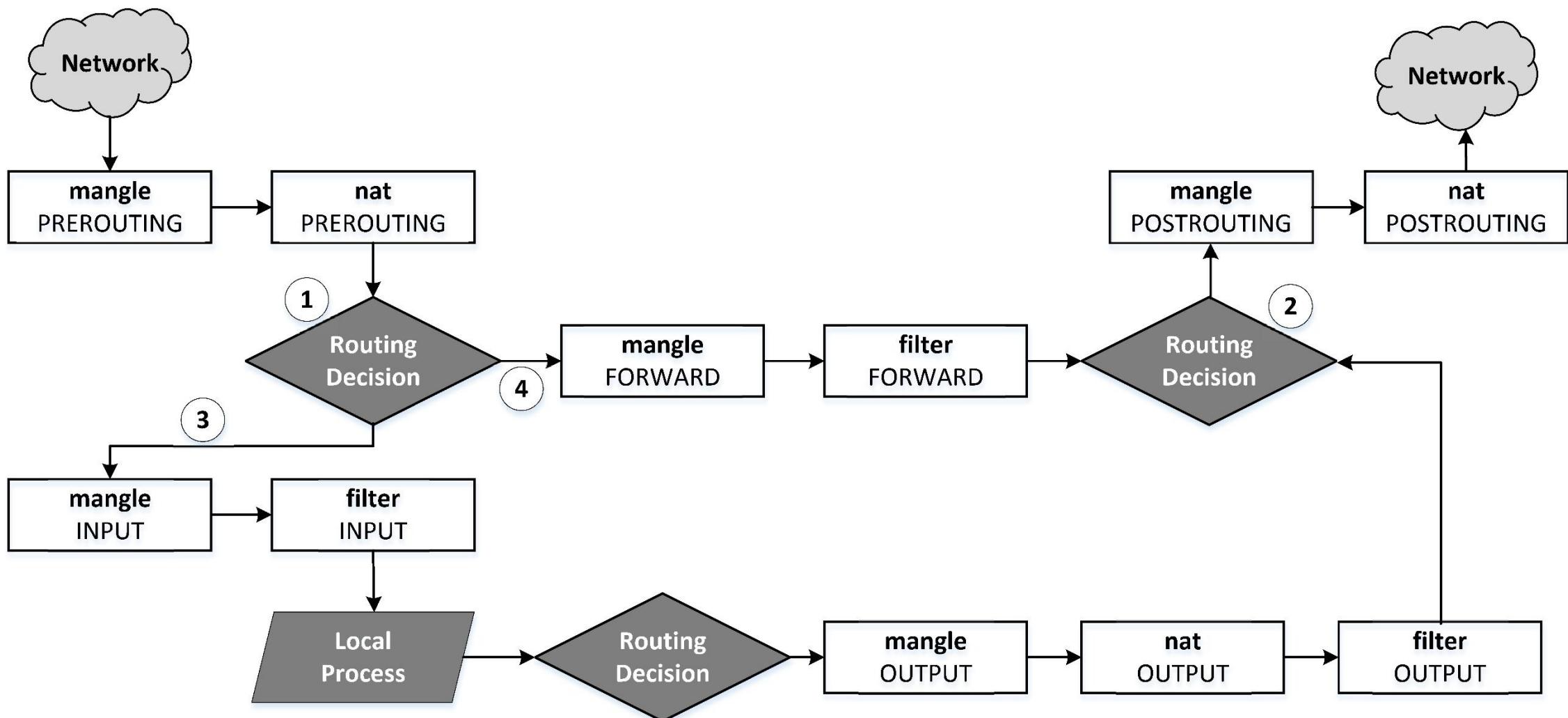
- Linux自带有一个功能很强大的防火墙，基于Netfilter，这个防火墙叫作iptables。
- 严格来说，该防火墙在内核中的部分叫作xtables，而iptables是用户空间用来设置防火墙的程序。然而，iptables经常作为内核部分和用户空间部分两者的统称。
- 注：5.7和5.8内容来自于《计算机安全导论：深度实践》，杜文亮著，2020年4月第1版，高等教育出版社。

iptables防火墙的结构

- iptables防火墙不仅可以用来过滤数据包，还可以修改数据包。
- 为了方便管理，iptables使用**表和链**来管理不同用途的防火墙规则。
- iptables主要有5个表：**filter**、**nat**、**mangle**、**raw**和**security**，每个表存放不同用途的规则。
 - 例如，过滤数据包的规则放在filter表中，而修改数据包的规则放在nat或者mangle表中；只修改源和目的地址的规则放在nat表中，其他修改规则放在mangle表中。
- 每个表中有若干个链，每个链对应一个Netfilter的钩子，每个链保存待执行的防火墙规则。

表	链	用途
filter	INPUT FORWARD OUTPUT	数据包过滤
nat	PREROUTING INPUT OUTPUT POSTROUTING	修改数据包的源和目标地址
mangle	PREROUTING INPUT FORWARD OUTPUT POSTROUTING	修改数据包的内容

遍历链和规则匹配



一个例子

- 假设要使所有数据包的生存时间(time to live, TTL)增大5，因为需要改变数据包，选择向mangle表中添加一条规则。这个表有netfilter提供的所有链，这里选择POSTROUTING链，以使所有发出去的数据包都能得到改变。完整的iptables命令如下：

// -t mangle : 加到mangle表中

// -A POSTROUTING : 附加到POSTROUTING链上

sudo iptables -t mangle -A POSTROUTING -j TTL --ttl-inc 5

- 用Wireshark观察可以看到，添加规则前后，从本机发出的数据包的TTL域的值相差5。

iptables的扩展组件

- iptables可以通过增加模块来扩展功能。很多模块并没有预先安装在标准的Linux内核中，这些iptables模块也叫作扩展组件。
- 例如，iptables本身不能设置基于连接的规则，但 conntrack模块使其成为可能。
 - 实际上，conntrack是一个很重要的模块，它可用于搭建状态防火墙，设置基于连接的防火墙规则，而不仅仅基于单个数据包。讨论状态防火墙时将介绍conntrack模块。
- 另一个例子是owner模块。标准iptables无法设置基于用户id的规则。
 - 比如，如果只想阻止用户Alice发送telnet数据包，使用标准iptables是做不到的。为了做到这一点，对于本地产生的每个数据包，都需要找到产生它的进程的用户id。
 - iptables的扩展模块owner可以提供这个功能，也就是能够找到数据包是由哪个用户或者用户组生成的。

扩展模块owner

- 这个功能只限于OUTPUT链，在其他链（如INPUT链）上很难做到，因为无法判断传入的数据包是哪个用户生成的。有时，即使在OUTPUT链中也难以做到这一点，因为有些数据包（如ICMP应答包）是由内核产生的，而不是由任何用户进程产生的，因此不存在所有者。
- 下面的规则使id为1001的用户产生的数据包全部被防火墙丢弃。

// -A OUTPUT : 附加到OUTPUT链上

// -m owner : 使用owner模块

// --uid-owner 1001 : 来自id为1000的用户的数据包（也可以使用用户名的登录名）

// -j DROP : 抛弃满足这条规则的包

sudo iptables -A OUTPUT -m owner --uid-owner 1001 -j DROP

注：有些Linux发行版本的owner模块可能不完善，使用的时候要实证。

搭建一个简单的防火墙

- 下面用iptables搭建一个简单的防火墙。这里只使用filter表。iptables命令中如果没有用“-t”选项选择一个具体的表，则默认选择的是filter表。
- 在开始设置防火墙之前，应该清除所有已经存在的防火墙设置，因为任何残存的设置都可能破坏防火墙策略。如果在一台远程计算机中实验，需要在清除规则之前确保默认策略是 ACCEPT，否则到远程计算机的访问将会因为默认策略而被阻拦。命令如下：

// 默认规则是接受 (ACCEPT) 所有数据包

```
sudo iptables -P INPUT ACCEPT
```

```
sudo iptables -P OUTPUT ACCEPT
```

```
sudo iptables -P FORWARD ACCEPT
```

// 清空filter表中的所有链

```
sudo iptables -F
```

```
sudo iptables -t mangle -F // 清空mangle表中的所有链
```

- 从INPUT链开始， 打开端口号22和80以提供SSH和Web服务。命令如下：

// 打开端口号22和80

// -A INPUT : 把规则附加到INPUT链上

// -p tcp : 该规则只用于TCP数据包

// --dport nn : 目标端口号

sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT

- 为了让SSH和HTTP服务能发送响应消息给客户端， 需要允许对外发送TCP数据包。命令如下：

// 允许对外发送TCP数据包

// -A OUTPUT : 把规则附加到OUTPUT链上

// -p tcp : 该规则只用于TCP数据包

// -j ACCEPT : 接受满足此规则的包

sudo iptables -A OUTPUT -p tcp -j ACCEPT

- 这里还需要增加一条规则，以确保系统可以正确运行。同一个系统中的不同程序间往往也使用数据包来通信。它们利用一个称为 loopback 的虚拟接口，该接口对应多个IP地址，如 127.0.0.1、127.0.1.1 和 0.0.0.0。这个接口把信息导回自己，而不是导向其他计算机。

// -I INPUT 1 : 把规则加到INPUT链的第一个位置上

// -i lo : 选择发往loopback (lo) 接口的数据包

// -j ACCEPT : 接受满足此规则的包

sudo iptables -I INPUT 1 -i lo -j ACCEPT

- 最后，需要允许DNS的查询和回复。DNS使用的是UDP端口53。

// 允许DNS请求和答复包通过

sudo iptables -A OUTPUT -p udp --dport 53 -j ACCEPT

sudo iptables -A INPUT -p udp --sport 53 -j ACCEPT

sudo iptables -A OUTPUT -p udp --sport 53 -j ACCEPT

sudo iptables -A INPUT -p udp --dport 53 -j ACCEPT

搭建一个简单的防火墙

- 上述所有设置构成一个搭建在Netfilter上的简单防火墙。使用iptables命令可以列出所有设置。

`sudo iptables -L`

- 设置完成后，应当把默认策略修改回DROP，这样一来，只有满足规则的数据包才能够进入这台计算机，其他的都会被丢弃。命令如下：

`# Setting default filter policy to DROP`

`sudo iptables -P INPUT DROP`

`sudo iptables -P OUTPUT DROP`

`sudo iptables -P FORWARD DROP`

- 设置防火墙规则的命令可以保存在一个shell脚本中，这个脚本需要用sudo命令运行。

5.8 用iptables实现基于连接跟踪的状态防火墙

- 一个状态防火墙会监视在一段时间内进出网络的数据包，通过它们之间的关系，把已经存在的网络连接找出来。有了这些连接的信息，防火墙就能设置基于这些连接的规则。这里“连接”的含义是一个广义的概念，它不同于TCP的连接，实际上指的是数据流。因此这个概念不仅仅适用于面向连接的协议(如TCP)，也适用于无连接协议(如UDP和ICMP)。一个典型的状态防火墙跟踪如下类型的连接。

- (1)TCP连接。
- (2)UDP连接。当一个UDP客户端和服务器开始交换数据包时，状态防火墙就认为连接已经建立了。
- (3)ICMP连接。ICMP同样不建立连接。然而，一些类型的ICMP消息有请求和应答模式。一对请求与应答被视为一个连接。
- (4)复杂协议的连接。一些防火墙在传输层和网络层之上跟踪连接。例如，HTTP、FTP和IRC连接就是应用层协议，但由于它们被广泛使用，很多防火墙也提供对这些连接的跟踪。

Linux的连接跟踪框架

- Linux内核提供了一个连接跟踪框架nf_conntrack (Ayuso, 2006)。该框架和iptables一样，建立在Netfilter上。这个框架保存连接状态的信息。每个进入nf_conntrack的数据包都会被标上一个连接状态，以便之后的钩子处理数据包。下面列出连接的几种状态。
 - (1) NEW: 连接刚开始，该数据包用于建立连接。
 - (2) ESTABLISHED: 连接已建立，双方开始通信。
 - (3) RELATED: 这个状态专门用来建立不同连接之间的联系，例如FTP。
 - (4) INVALID: 这个状态用于标记不符合连接行为的数据包。
- nf_conntrack仅仅跟踪连接数据，它本身并不是一个防火墙。然而，使用这个连接跟踪系统所提供的信息，可以搭建状态检测防火墙。iptables防火墙主要使用nf_conntrack来进行与状态有关的工作。

搭建一个状态防火墙

- 仔细研究5.7中搭建的防火墙，可以看到这个防火墙允许所有对外发送的TCP数据包。
- 这么做的原因是需要允许SSH和HTTP服务器对外发送数据包，但没有简便的途径来限制只允许这些服务器发送数据，因此只能允许所有发出的TCP数据包。
- 如果一个网络系统设置了这个防火墙，已经攻陷内部主机的攻击者就能通过TCP发出数据。
- 应当注意的是，攻击者不能与外部服务器建立连接，因为防火墙会阻拦进入的非通往SSH或HTTP服务器的TCP数据，因此攻击者不能通过FTP或者telnet连接到外部服务器，但在当前的防火墙设置下他们可以发送TCP数据包，尽管无法收到回复，但这已足够让他们窃取数据。

- 为了避免这种行为，需要改进防火墙。连接跟踪机制可以被用于加强防火墙策略。为此需要建立一个防火墙规则，只允许发送属于一个已有连接的TCP数据包。因为仅希望允许SSH和HTTP连接，因此要阻拦所有发出的不属于SSH或HTTP连接的TCP数据包。这里将删除5.7中允许所有发出的TCP数据包的防火墙规则（**sudo iptables -A OUTPUT -p tcp -j ACCEPT**），而用下面的一个基于连接状态的规则来取代它。

// -m conntrack：使用conntrack模块中的规则

// --ctstate ESTABLISHED,RELATED：具备ESTABLISHED或RELATED状态的包

```
sudo iptables -A OUTPUT -p tcp -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

- 有了上述配置，如果使用telnet连接防火墙保护的计算机或者外部网络，同时使用 Wireshark监视防火墙的输出，会看到并没有关于telnet的TCP数据包，它们全被防火墙阻拦了。

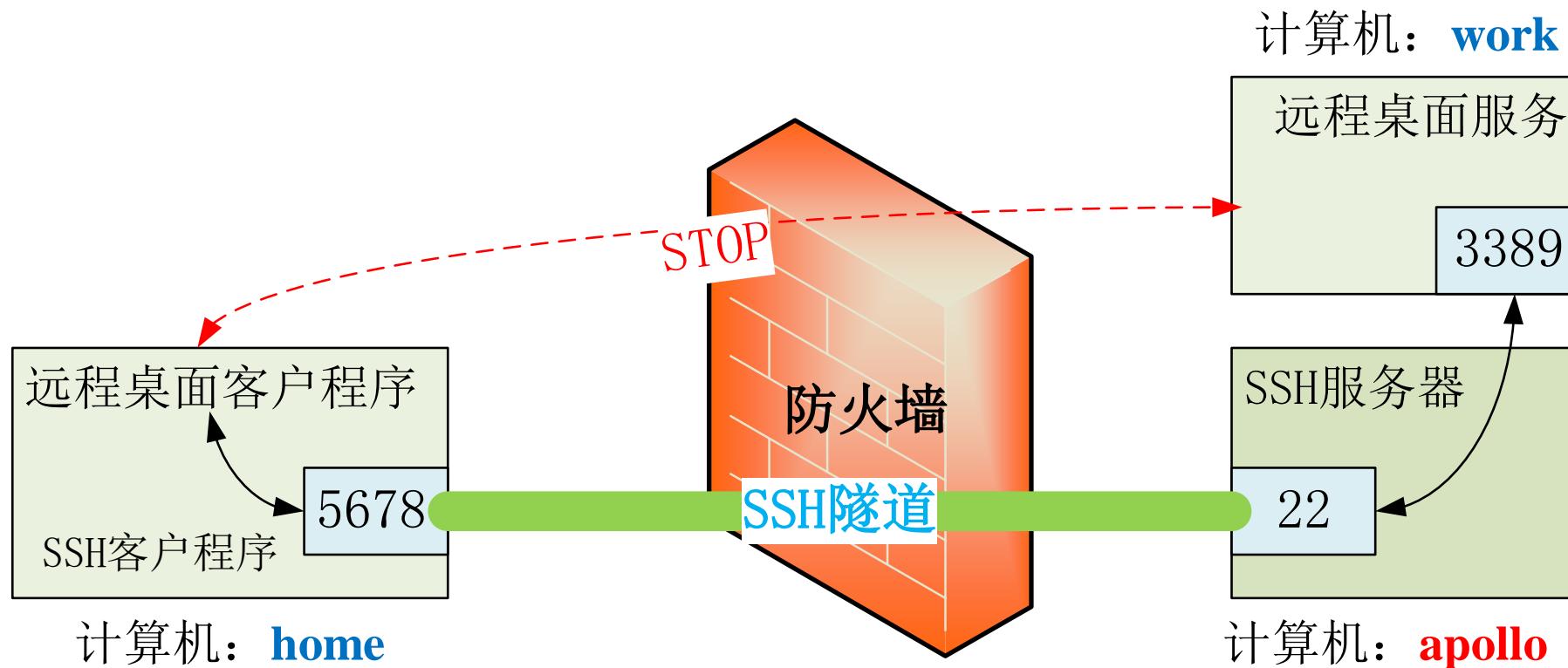
5.9 绕过防火墙的方法

- 如果防火墙的规则太过苛刻，将给用户带来许多不便。例如，企业网络只开放了少量的端口供外部访问等，要想访问企业内部网络，就必须绕过防火墙。
- 有很多绕过防火墙的方法，最典型和有效的方法是隧道技术，它能够隐藏数据的真实意图。
- 搭建隧道的方式有很多种，其中虚拟专用网络(virtual private network, VPN)在IP层搭建隧道，它被广泛用来绕过防火墙。另一种常用的方式是使用SSH隧道绕过防火墙，以访问内部主机的某个端口。
- 本节以访问校园网某实验室内部网络的主机远程桌面为例，说明通过SSH隧道绕过防火墙的方法。

使用SSH隧道绕过防火墙

- 某实验室的路由器通过一个互联网IP地址（假定为202.38.79.51）连接到校园网，实验室内部网络通过路由器可以访问外部网络。
- 目的：从外部网络叫作home的计算机通过远程桌面（TCP 3389）连接到实验室的work计算机，然而，防火墙只开放了SSH端口（TCP 22），使得无法从外网通过远程桌面连接到work计算机。
- 防火墙允许外网用户用SSH登录到内部一台叫作apollo的计算机，下面用这台计算机来绕过防火墙。
- 可以在home和apollo之间建立一条SSH隧道。SSH隧道的工作原理如图所示。

SSH隧道的工作原理



- SSH隧道在home的一端将接收从远程桌面客户程序发来的TCP数据包，SSH会把这些包中的TCP数据通过隧道传输到apollo端。
- 在apollo上运行的SSH会把数据放入另一个新创建的TCP数据包并发往work。
- 防火墙只能检测到home发往apollo的SSH数据，而检测不到apollo发往work的远程桌面数据包。并且，SSH数据是加密的，因此防火墙无法得知其中的内容。

演示SSH隧道

- 在home的命令行窗口上运行以下命令，可以在home的TCP 5678号端口到apollo的SSH端口间建立一条SSH隧道。这条隧道将把home的5678号端口上接收到的所有TCP数据转发到work的3389号端口(远程桌面端口)。

ssh -L 5678:work:3389 user@apollo

- 建立上述隧道之后，并不需要直接在home到work间建立远程桌面连接（会被防火墙屏蔽），而是通过远程桌面连接到本地主机的5678号端口，由隧道把远程桌面数据传输到apollo，并由apollo转给work。apollo会把work的回复数据通过隧道发回给本地主机的SSH，然后由SSH交给远程桌面客户程序。
- 在home的另一个命令行窗口运行以下命令，输入正确的用户名/口令，可以连接到work。

mstsc /v:localhost:5678

实例：连接到校园网的内部主机

- 环境信息：
 - SSH服务器apollo的IP地址： **114.214.215.19**, 用户名： i
 - 工作计算机work的IP地址： **10.0.5.16**
- 步骤：
 - (1) 建立SSH 隧道： 在本机(home)上运行以下命令
ssh -L 5800:10.0.5.16:3389 -p 22 i@114.214.215.19
 - (2) 用远程桌面连接到计算机work： 在本机(home)上运行以下命令
mstsc /v:localhost:5800

谢谢！