



网络安全系统设计

中科院计算所教育中心



内容

- **1. Firewall**
- **2. SSL/TLS**
- **3. IPSec**
- **4. IDS**



防火墙(Firewall)

- 防火墙的基本设计目标
 - 对于一个网络来说，所有通过“内部”和“外部”的网络流量都要经过防火墙
 - 通过一些安全策略，来保证只有经过授权的流量才可以通过防火墙
 - 防火墙本身必须建立在安全操作系统上
- 防火墙的控制能力
 - 服务控制，确定哪些服务可以被访问
 - 方向控制，对于特定的服务，可以确定允许哪个方向能够通过防火墙
 - 用户控制，根据用户来控制对服务的访问
 - 行为控制，控制一个特定的服务的行为



防火墙能为我们做什么

- 定义一个**必经之点**
 - 挡住未经授权的访问流量
 - 禁止具有脆弱性的服务带来危害
 - 实施保护，以避免各种IP欺骗和路由攻击
- 防火墙提供了一个监视各种安全事件的位置，所以，**可以在防火墙上实现审计和报警**
- 对于有些**Internet**功能来说，防火墙可以是一个理想的平台，比如地址转换，**IPSec/VPN**，**Internet**日志、审计，甚至计费功能



防火墙本身的一些局限性

- 对于绕过防火墙的攻击，它无能为力。如内部网用户通过**SLIP**或**PPP**直接进入**Internet**。
- 防火墙不能防止内部的攻击，以及内部人员与外部人员的联合攻击(比如，通过**tunnel**进入)
- 防火墙对用户不完全透明，可能带来传输延迟、瓶颈及单点失效。
- 防火墙不能防止被病毒感染的程序或者文件、邮件等



防火墙的类型

- 包过滤路由器
- 应用层网关
- 电路层网关

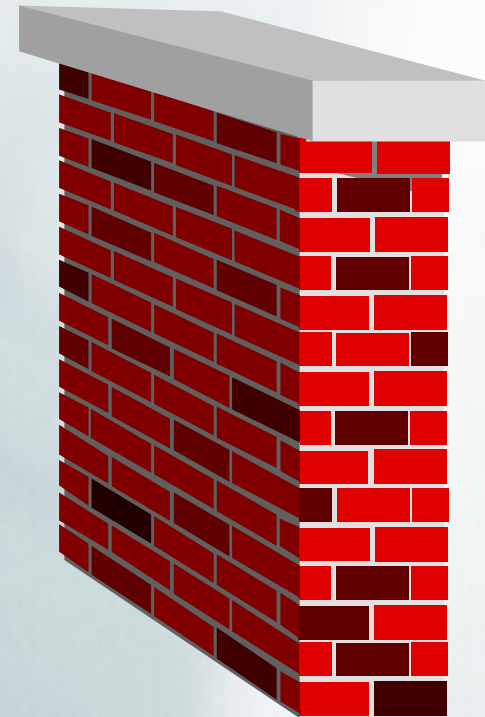


静态包过滤路由器

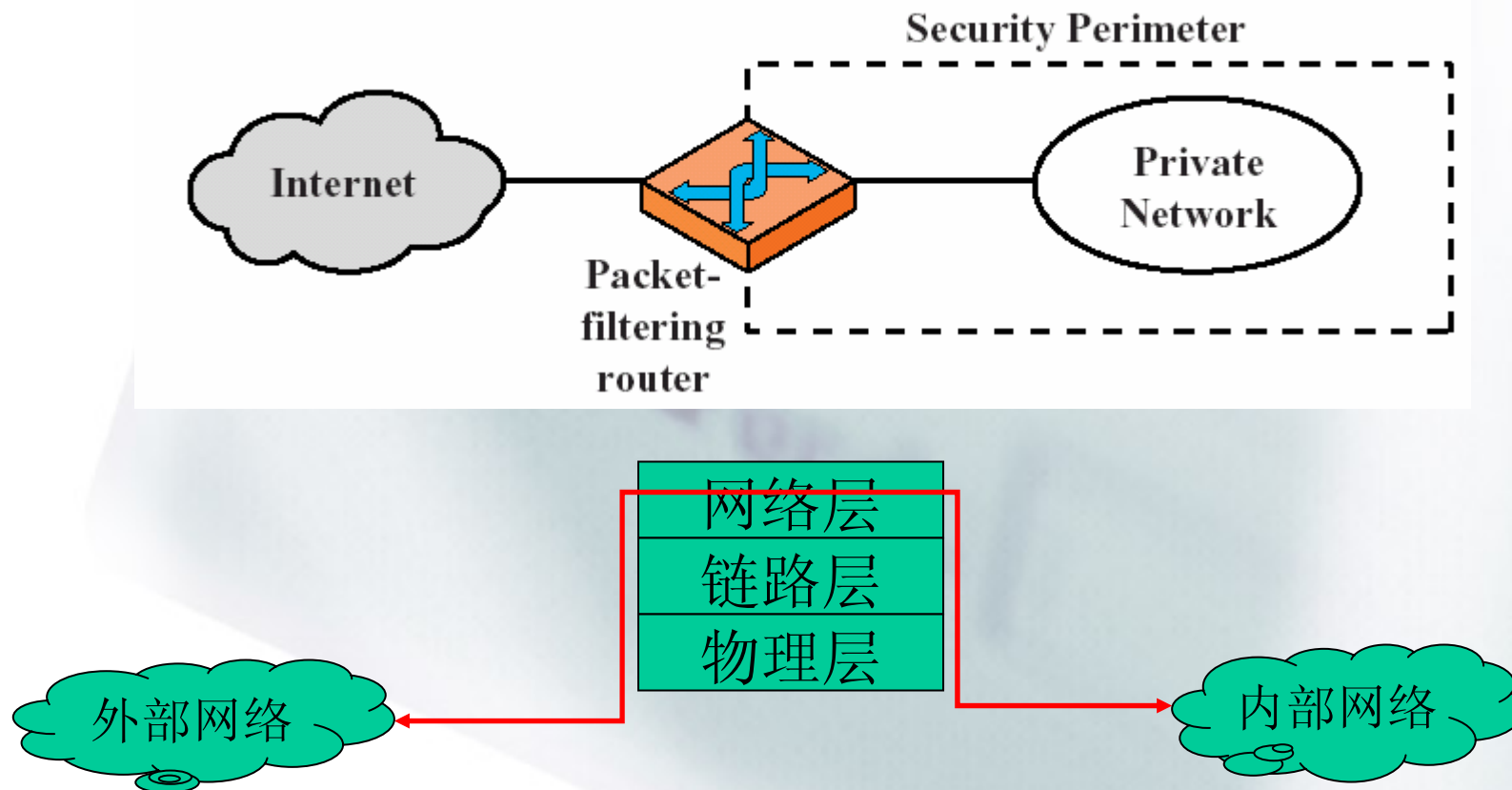
- 基本的思想很简单
 - 对于每个进来的包，适用一组规则，然后决定转发或者丢弃该包
 - 往往配置成双向的
- 如何过滤
 - 过滤的规则以IP和传输层的头中的域(字段)为基础，包括源和目标IP地址、IP协议域、源和目标端口号
 - 过滤器往往建立一组规则，根据IP包是否匹配规则中指定的条件来作出决定。
 - 如果匹配到一条规则，则根据此规则决定转发或者丢弃
 - 如果所有规则都不匹配，则根据缺省策略

安全缺省策略

- 两种基本策略，或缺省策略
 - 没有被拒绝的流量都可以通过
 - 管理员必须针对每一种新出现的攻击，制定新的规则
 - 没有被允许的流量都要拒绝
 - 比较保守
 - 根据需要，逐渐开放



静态包过滤路由器示意图





静态包过滤防火墙特点

- 在网络层上进行监测
 - 并没有考虑连接状态信息
- 通常在路由器上实现
 - 实际上是一种网络的访问控制机制
- 优点：
 - 实现简单
 - 对用户透明
 - 效率高



静态包过滤防火墙缺点

- 配置基于包过滤方式的防火墙，需要对IP、TCP、UDP、ICMP等各种协议有深入的了解，否则容易出现因配置不当带来的问题；
- 据以过滤判别的只有网络层和传输层的有限信息，因而各种安全要求不能得到充分满足；
- 由于数据包的地址及端口号都在数据包的头部，不能彻底防止地址欺骗；
- 允许外部客户和内部主机的直接连接；不提供用户的鉴别机制。

包过滤防火墙的设置(1)

- 从内往外的telnet服务



往外包的特性(用户操作信息)

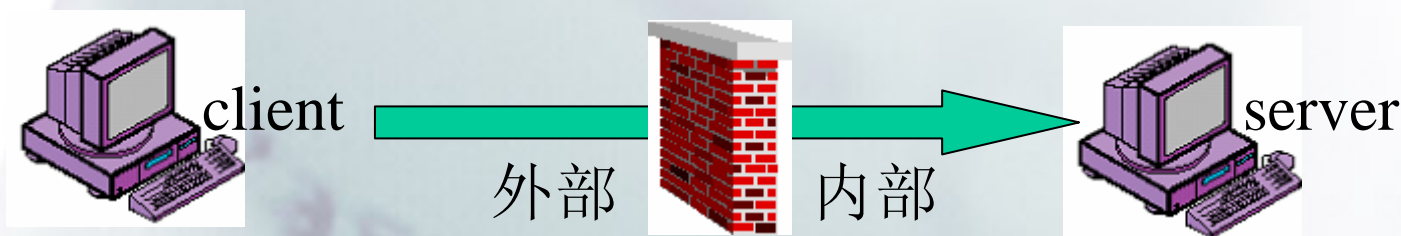
- IP源是内部地址
- 目标地址为server
- TCP协议，目标端口23
- 源端口>1023
- 连接的第一个包ACK=0，其他包ACK=1

往内包的特性(显示信息)

- IP源是server
- 目标地址为内部地址
- TCP协议，源端口23
- 目标端口>1023
- 所有往内的包都是ACK=1

包过滤防火墙的设置(2)

- 从外往内的telnet服务



往内包的特性(用户操作信息)

- IP源是外部地址
- 目标地址为本地server
- TCP协议，目标端口23
- 源端口>1023
- 连接的第一个包ACK=0，其他包ACK=1

往外包的特性(显示信息)

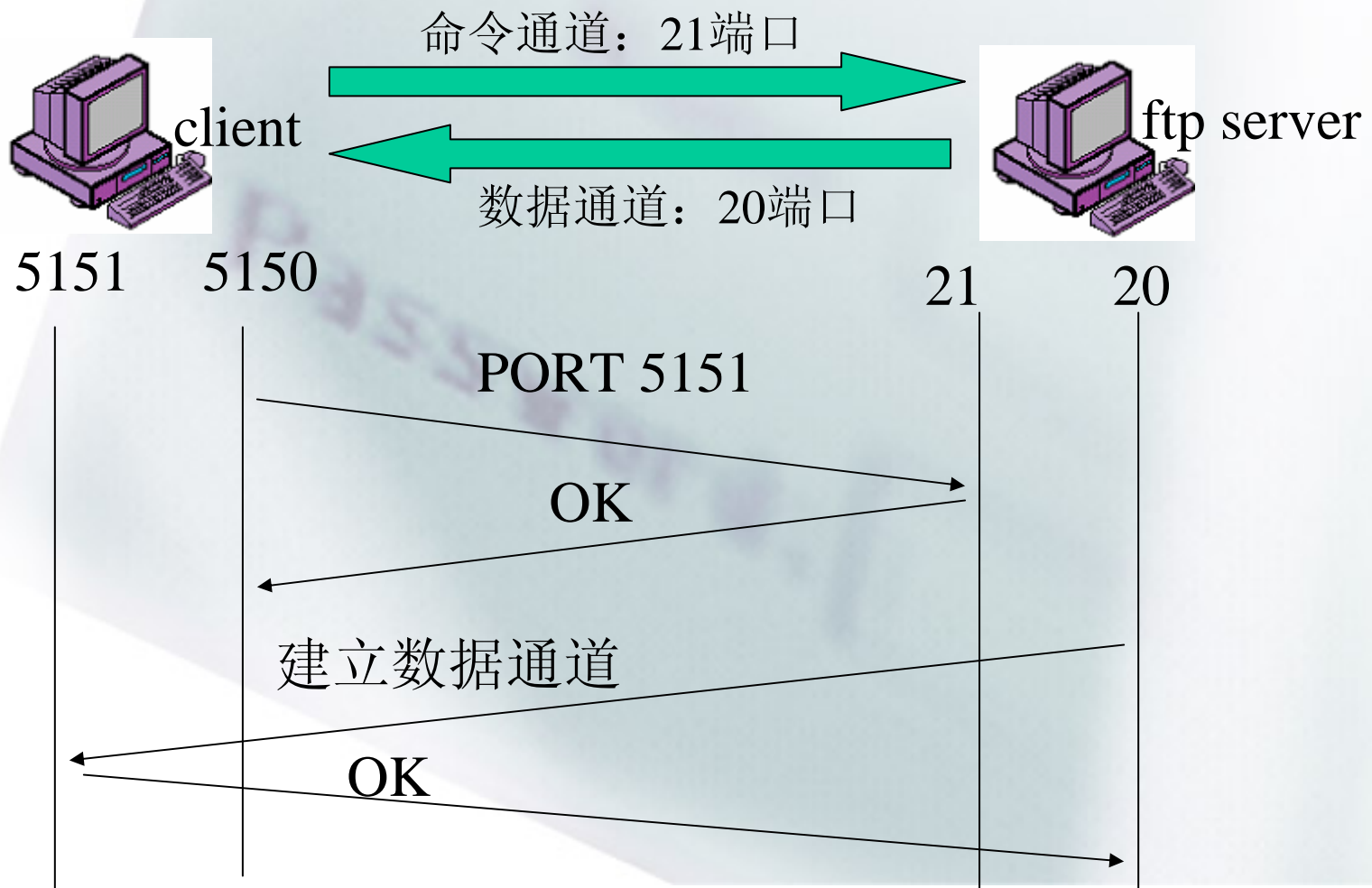
- IP源是本地server
- 目标地址为外部地址
- TCP协议，源端口23
- 目标端口>1023
- 所有往内的包都是ACK=1

针对telnet服务的防火墙规则

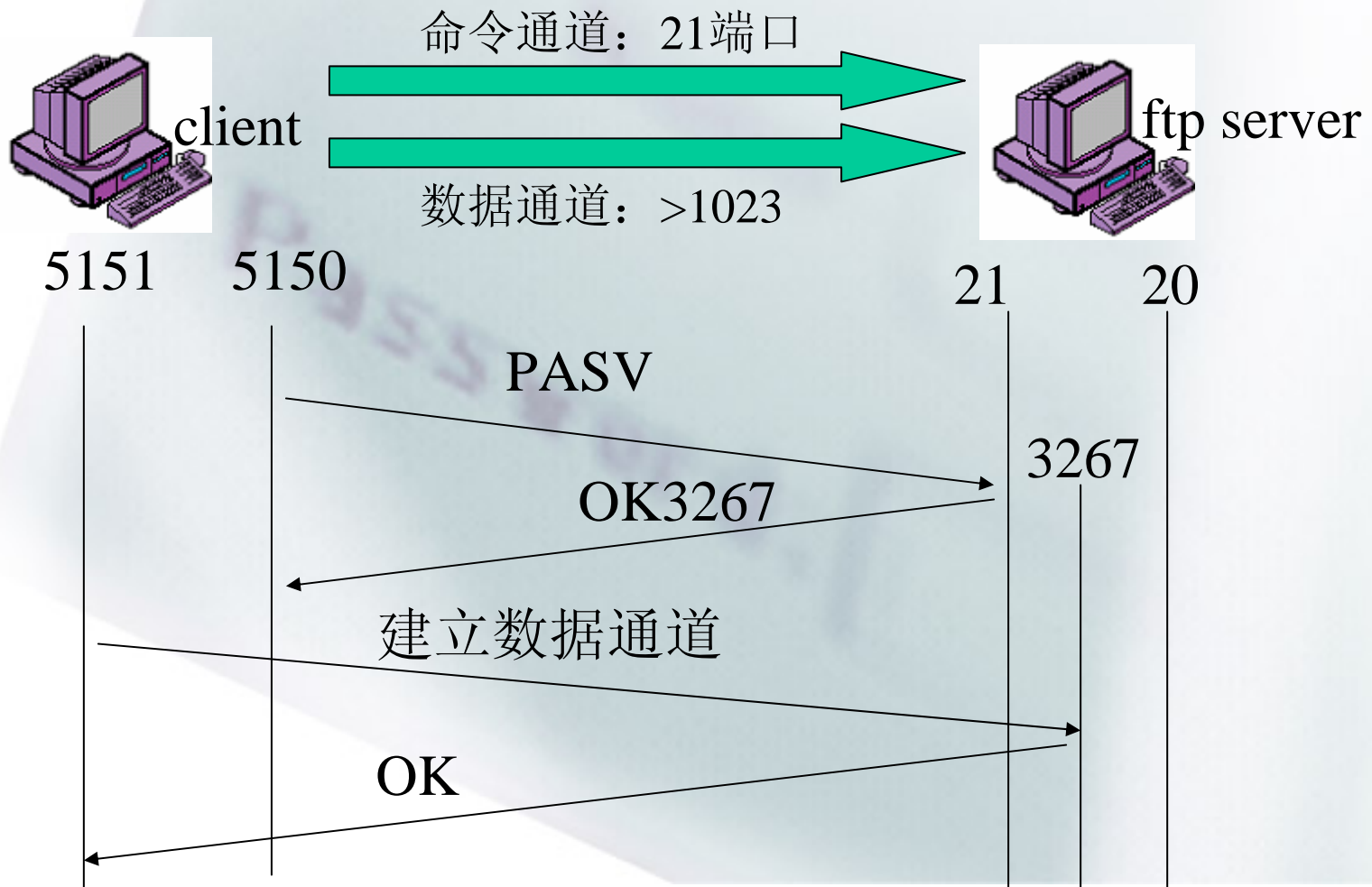
站在内部的角度看:

服务方向	包方向	源地址	目标地址	包类型	源端口	目标端口	ACK
连接外部的telnet服务器	OUT	内部	外部	TCP	>1023	23	*
	IN	外部	内部	TCP	23	>1023	1
向外提供Telnet服务	IN	外部	内部	TCP	>1023	23	*
	OUT	内部	外部	TCP	23	>1023	1

ftp文件传输协议



ftp文件传输协议





针对ftp的包过滤规则注意事项

- 建立一组复杂的规则集
 - 是否允许正常模式的ftp数据通道？
 - 有些ftp client不支持pasv模式
- 动态监视ftp通道发出的port命令
 - 有一些动态包过滤防火墙可以做到
- 启示
 - 包过滤防火墙比较适合于单连接的服务(比如smtp, pop3)，不适合于多连接的服务(比如ftp)



针对静态包过滤防火墙的攻击

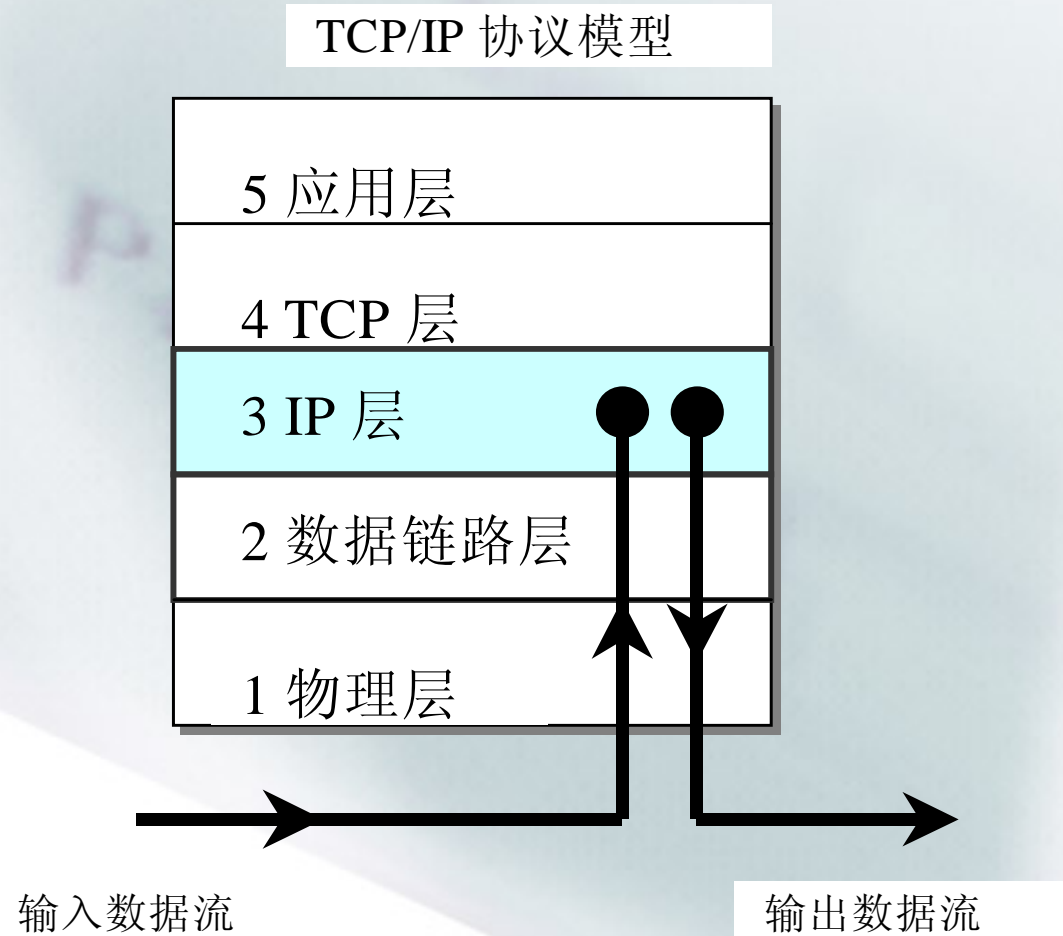
- **IP地址欺骗**，例如，假冒内部的**IP**地址
 - 对策：在外部接口上禁止内部地址
- **小碎片攻击**，利用**IP**分片功能把**TCP**头部切分到不同的分片中
 - 对策：丢弃分片太小的分片
- **利用复杂协议和管理员的配置失误进入防火墙**
 - 例如，利用ftp协议对内部进行探查, ftp proxy scan



静态包过滤特点

- 根据流经该设备的数据包地址信息，决定是否允许该数据包通过
- 判断依据有(只考虑IP包):
 - 数据包协议类型：TCP、UDP、ICMP、IGMP等
 - 源、目的IP地址
 - 源、目的端口：FTP、HTTP、DNS等
 - IP选项，TCP选项及flag
 - 其它协议选项：ICMP ECHO、ICMP ECHO REPLY等
 - 数据包流向：in或out
 - 数据包流经网络接口：eth0、eth1
- 实例：[个人防火墙](#)

静态包过滤防火墙工作层次



静态包过滤设置实例

- 按地址

规则	方向	源地址	目标地址	动作
A	出	内部网络	202.110.8.0	拒绝
B	入	202.110.8.0	内部网络	拒绝

- 按服务

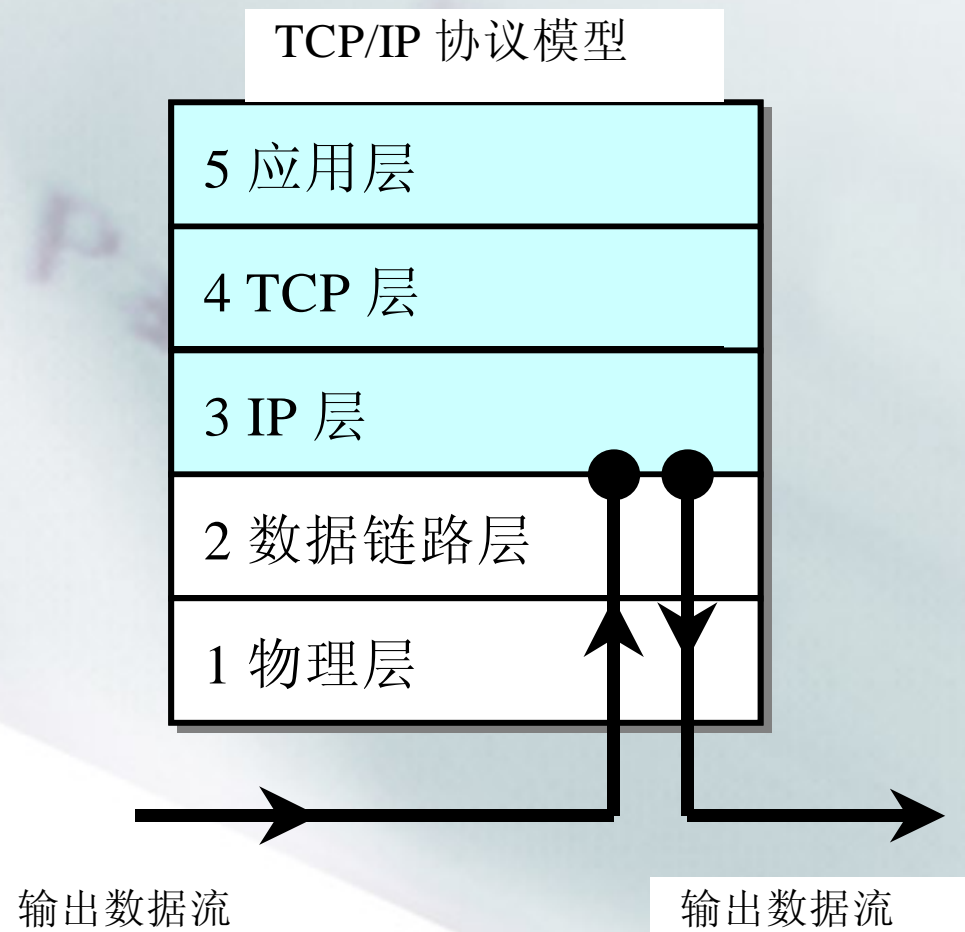
规则	方向	动作	源地址	源端口	目的地址	目的端口	注释
A	进	拒绝	m	*	E-mail	25	不信任
B	出	允许	*	*	*	*	允许
C	双向	拒绝	*	*	*	*	缺省状态



动态包过滤

- Check point 首创称为“**Stateful Inspection**”的技术
- 它根据数据包的头信息打开或关闭端口。当一组数据包通过打开的端口到达目的地，防火墙就关闭这些端口。
- 可动态生成/删除规则
- 分析高层协议

动态包过滤防火墙工作示意图



动态包过滤特点

- 对网络通信的各层实施监测分析，提取相关的通信和状态信息，并在动态连接表中进行状态及上下文信息的存储和更新，这些表被持续更新，为下一个通信检查提供累积的数据。
- 状态监视器的另一个优点是：能够提供基于端口动态分配协议（RPC）的应用（如NFS、NIS）的安全支持，减少了端口的开放时间，提供了对几乎所有服务的支持，缺点是它也允许外部客户和内部主机的直接连接；不提供用户的鉴别机制。



Linux中的包过滤防火墙

Iptables

Linux 在2.4内核版本中向用户提供的包过滤防火墙工具，用于替代**Ipchains**。内核部分需要有**netfilter**支持。

相对于**ipchains**的进步：

1. 具备连接跟踪（**Connection tracking**）能力，可进行包的状态监测。
2. **forward**类型的包只过一个链，不像**ipchains**每个**forward**包都要过3个链
3. 包过滤和**NAT**（**network address translation**）有了清晰的分离。
4. 有了连接速率限制功能(**Rate-limited**)。
5. 可以对**TCP**的**flag**和**option**以及**MAC**地址进行过滤。



iptables 实现的功能

包过滤（**packet filter**）

对数据报进行过滤

连接跟踪（新增功能）

网络地址转换（**NAT**）

源NAT（SNAT）、目的NAT（DNAT）

伪装IP 透明代理 IP映射

数据报更改处理 (**Mangle**)

可以实现对数据报的修改



NAT

- SNAT 变换数据包的源地址。

例:更改所有来自**192.168.1.0/24**的数据包的源ip地址为**1.2.3.4**:

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -  
o eth0 -j SNAT --to 1.2.3.4
```

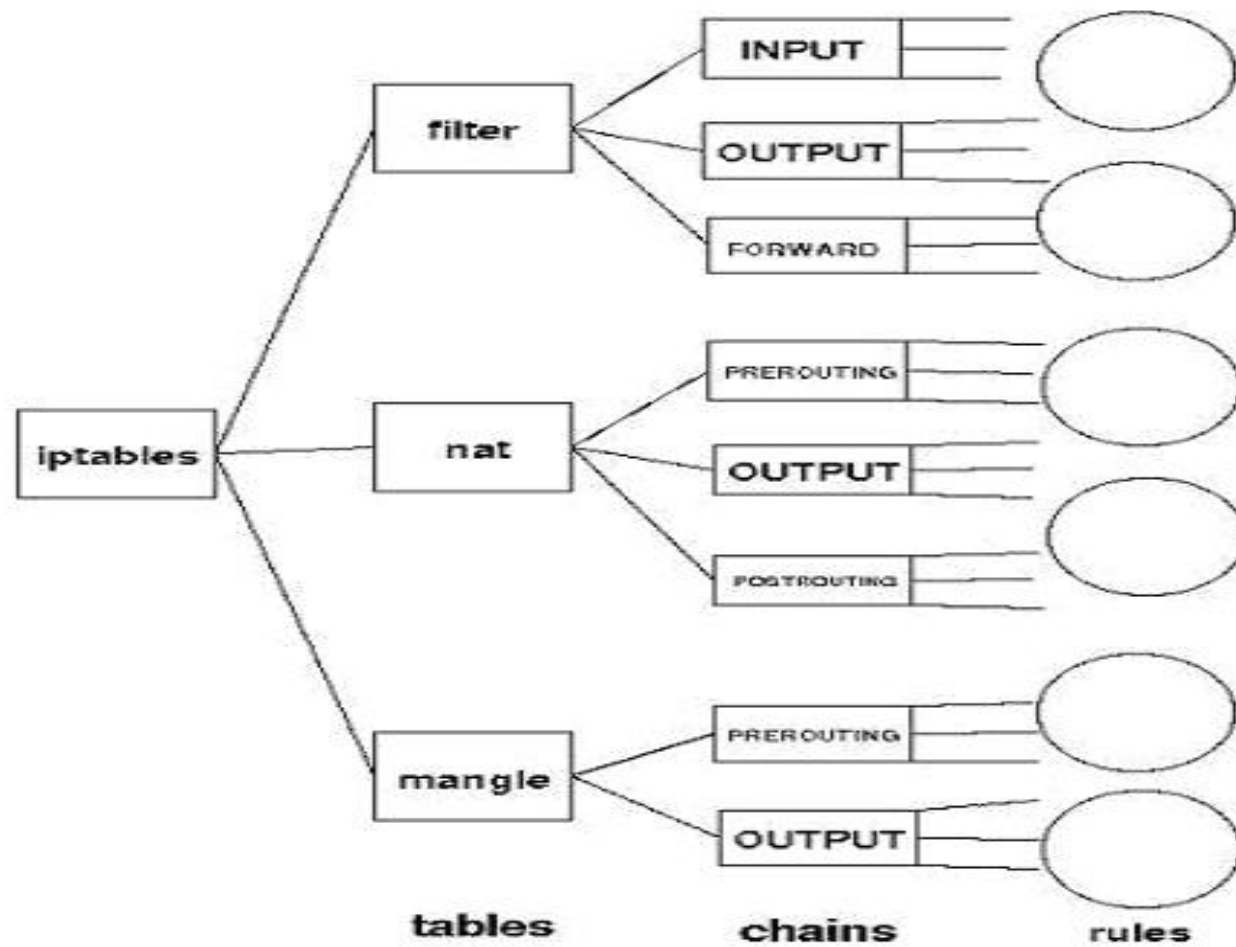
- DNAT, 用于对外部提供www,ftp等服务

例:对接收到的ip为**1.2.3.4**的数据包转到目的**192.168.1.3**:

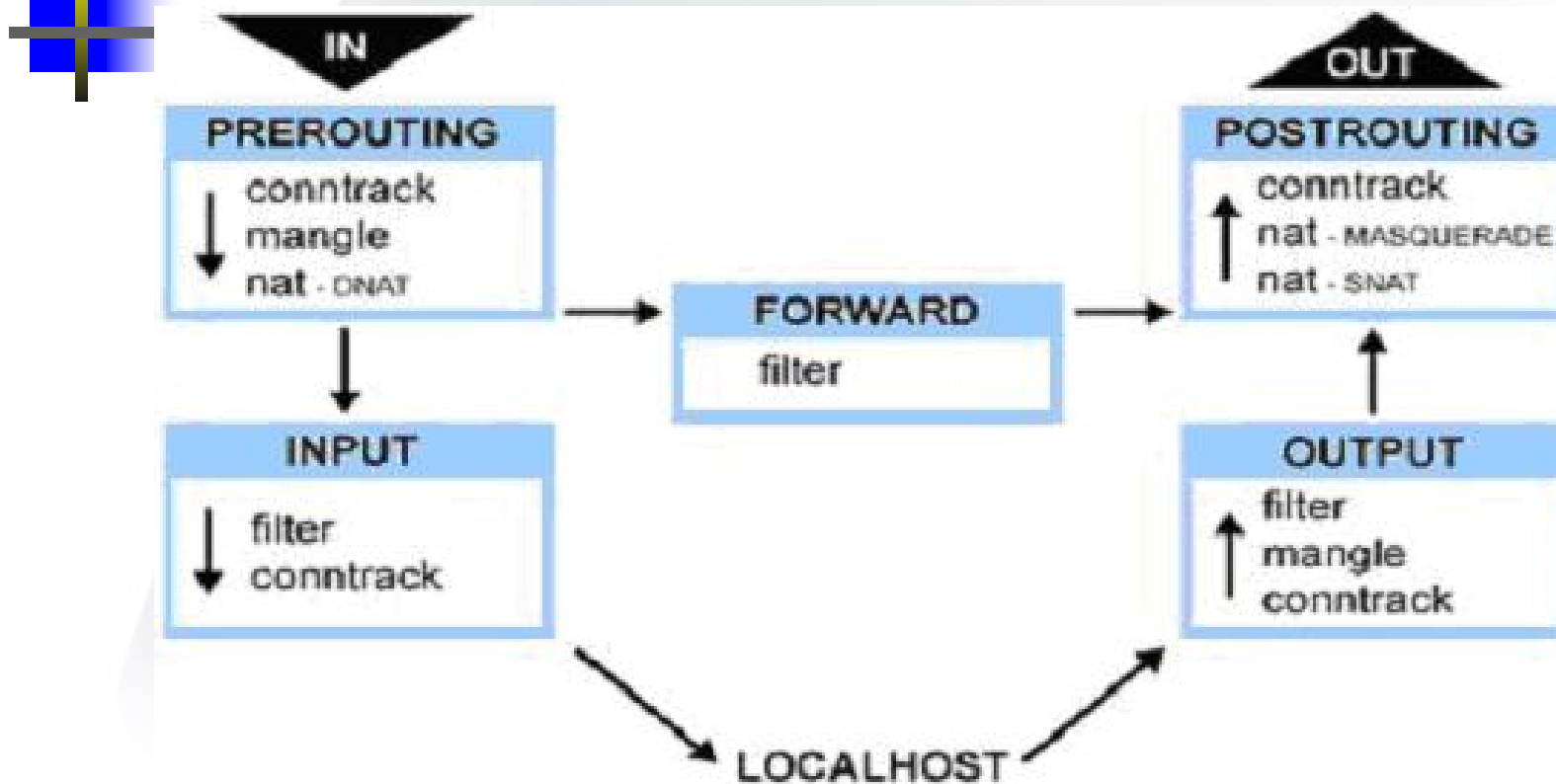
```
iptables -t nat -A PREROUTING -i eth0 -d 1.2.3.4 -  
j DNAT --to 192.168.1.3
```

这里需要注意的是, 系统是先进行DNAT, 然后才进行路由及过滤等操作

iptables 结构图



包进入规则表及规则链图



注意: Iptables 命令通过不同的表, 在不同的位置插入了检查和动作,
注意: nat filter 和 mangle的位置是不一样的, DNAT,SNAT,和MASQUERADE只是
目标而已(-j), 需要在iptables中nat表的PRERUTING链或POSTROUTING链中定义
SNAT,DNAT目标而达到目的。

iptables 的命令格式

一个iptables命令基本上包含如下五部分：希望工作在哪个表上、希望使用该表的哪个链、进行的操作(插入，添加，删除，修改)、对特定规则的目标动作、匹配数据报条件。

基本的语法为：

iptables -t table -Operation chain -j target_ match

例如：希望添加一个规则，允许所有从任何地方到本地smtp端口的连接：

iptables -t filter -A INPUT -j ACCEPT -p tcp --dport smtp



基本的 Operation

- A** 在链尾添加一条规则;
- I** 插入规则;
- D** 删除规则
- R** 替代一条规则;
- L** 列出规则。



基本的 target

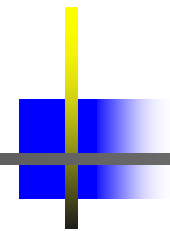
适用于所有的链：

ACCEPT 接收该数据报；

DROP 丢弃该数据报；

QUEUE 排队该数据报到用户空间；

RETURN 返回到前面调用的链；



基本的match

适用于所有的链:

- p** 指定协议(tcp/icmp/udp/...);
- s** 源地址(ip address/masklen);
- d** 目的地址(ip address/masklen);
- i** 数据报输入接口;
- o** 数据报输出接口;



命令和应用举例

```
iptables -A INPUT -s 205.168.0.1 -j ACCEPT
```

该示例命令将一条规则附加到 INPUT 链的末尾，确定来自源地址 205.168.0.1 的信息包可以 ACCEPT。

```
iptables -D INPUT --dport 80 -j DROP
```

```
iptables -D OUTPUT 3
```

第一条命令从 INPUT 链删除规则，它指定 DROP 前往端口 80 的信息包。

第二条命令只是从 OUTPUT 链删除编号为 3 的规则。



命令和应用举例

ISP分配给A单位www服务器的ip为:

伪ip: 192.168.1.100

真实ip: 202.110.123.100

该ISP分配给B单位www服务器的ip为:

伪ip: 192.168.1.200

真实ip: 202.110.123.200

命令和应用举例

- **linux**防火墙的**ip**地址分别为：

内网接口**eth1**: **192.168.1.1**

外网接口**eth0**: **202.110.123.1**

然后我们将分配给A、B单位的真实**ip**绑定到防火墙的**外网接口**，以**root**权限执行以下命令：

```
ifconfig eth0 add 202.110.123.100 netmask 255.255.255.0
```

```
ifconfig eth0 add 202.110.123.200 netmask 255.255.255.0
```



命令和应用举例

首先，对防火墙接收到的目的ip为
202.110.123.100和**202.110.123.200**的所有数据包
进行目的NAT(DNAT):

```
iptables -A PREROUTING -i eth0 -  
d 202.110.123.100 -j DNAT --to 192.168.1.100  
iptables -A PREROUTING -i eth0 -  
d 202.110.123.200 -j DNAT --to 192.168.1.200
```



命令和应用举例

其次，对防火墙接收到的源ip地址为192.168.1.100和192.168.1.200的数据包进行源NAT(SNAT):

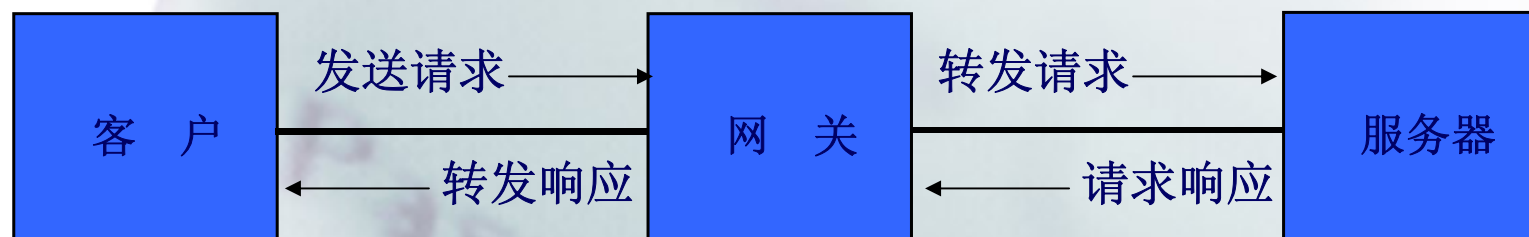
```
iptables -A POSTROUTING -o eth0 -  
s 192.168.1.100 -j SNAT --to 202.110.123.100  
iptables -A POSTROUTING -o eth0 -  
s 192.168.1.200 -j SNAT --to 202.110.123.200
```



应用层网关

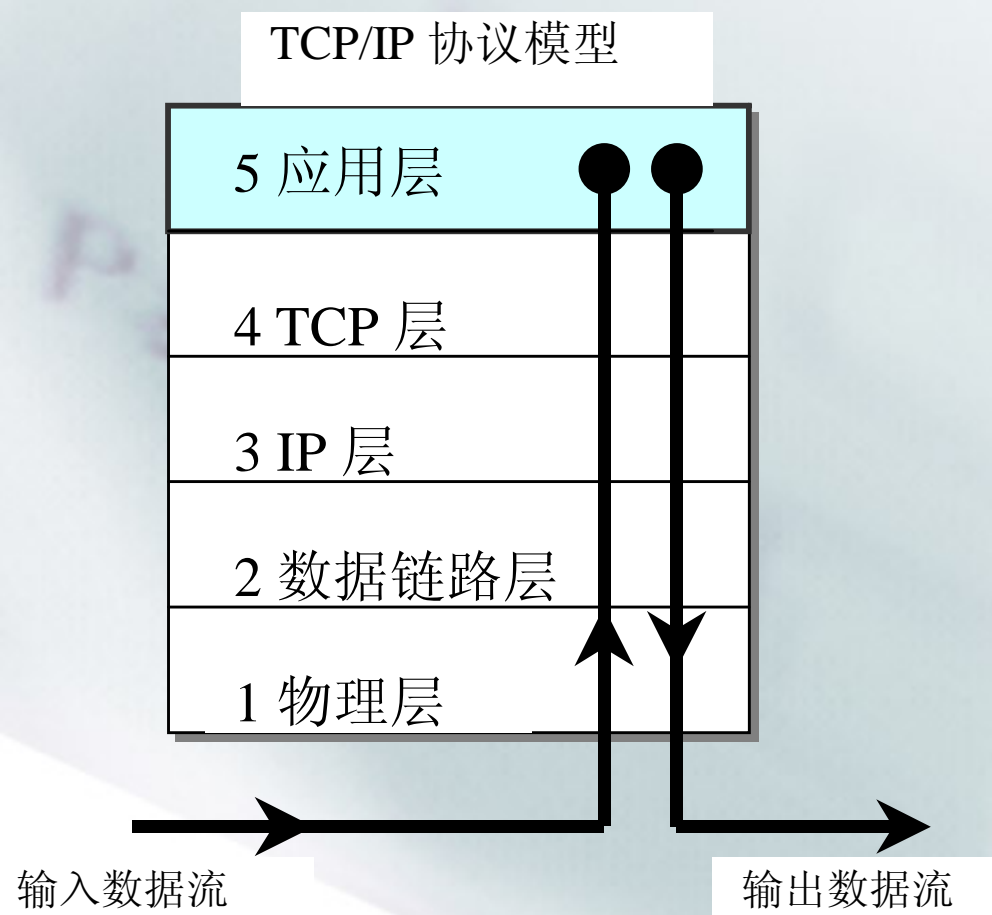
- 也称为代理服务器
- 特点
 - 所有的连接都通过防火墙，防火墙作为网关
 - 在应用层上实现
 - 可以监视包的内容
 - 可以实现基于用户的认证
 - 所有的应用需要单独实现
 - 可以提供理想的日志功能
 - 非常安全，但是开销比较大

应用级网关

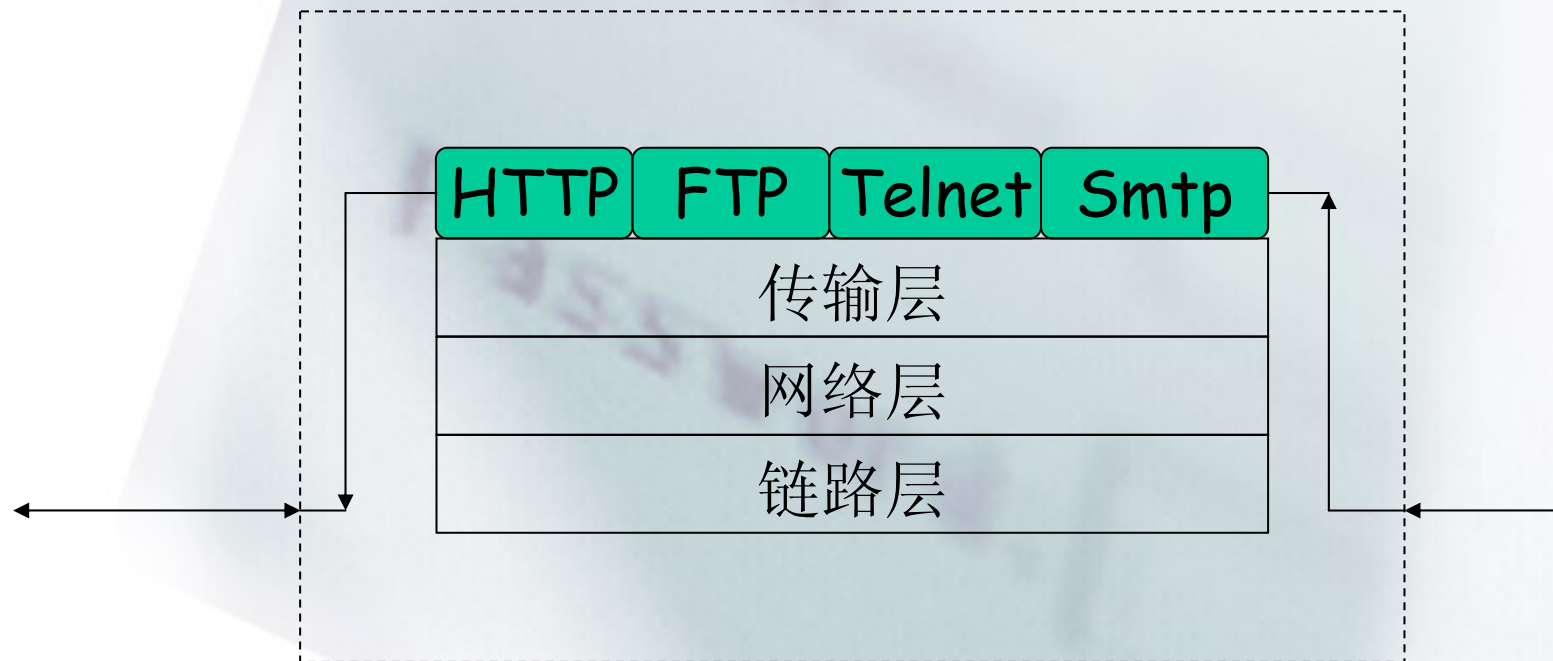


1. 网关理解应用协议，可以实施更细粒度的访问控制
2. 对每一类应用，都需要一个专门的代理

应用级网关防火墙工作层次



应用层网关的协议栈结构





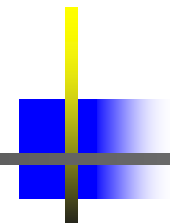
应用层网关的优缺点

- 优点
 - 允许用户“直接”访问Internet
 - 易于记录日志
- 缺点
 - 新的服务不能及时地被代理
 - 每个被代理的服务都要求专门的代理软件
 - 客户软件需要修改，重新编译或者配置
 - 有些服务要求建立直接连接，无法使用代理
 - 比如聊天服务、或者即时消息服务
 - 代理服务不能避免协议本身的缺陷或者限制



应用层网关实现

- 编写代理软件
 - 代理软件一方面是服务器软件
 - 但是它所提供的服务可以是简单的转发功能
 - 另一方面也是客户软件
 - 对于外面真正的服务器来说，是客户软件
 - 针对每一个服务都需要编写模块或者单独的程序
 - 实现一个标准的框架，以容纳各种不同类型的服务
 - 软件实现的可扩展性和可重用性
- 客户软件
 - 软件需要定制或者改写
 - 对于最终用户的透明性差
- 协议对于应用层网关的处理
 - 协议设计时考虑到中间代理的存在，特别是在考虑安全性，比如数据完整性的时候



应用层网关——代理服务器

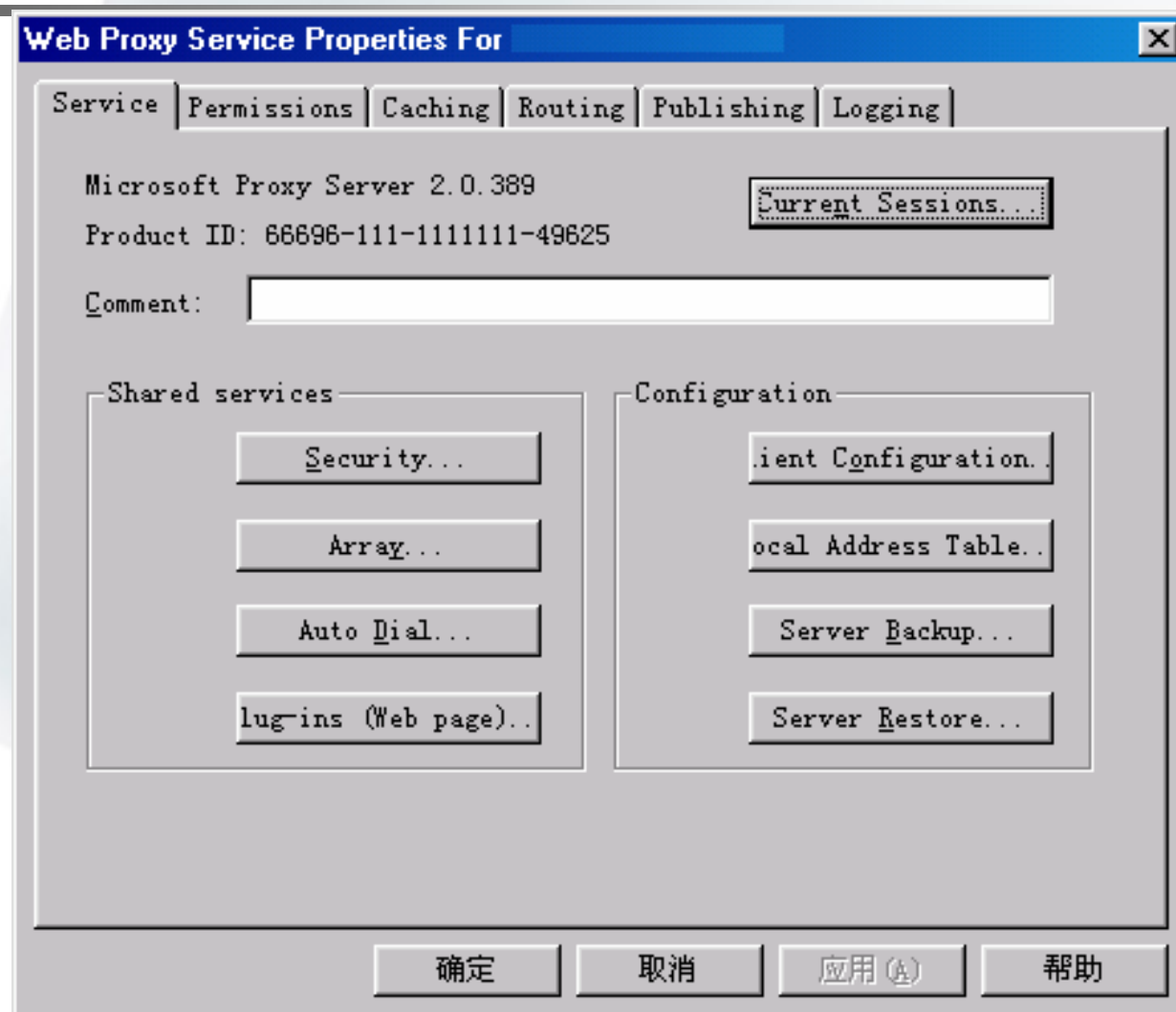
- 发展方向——智能代理
 - 不仅仅完成基本的代理访问功能
 - 还可以实现其他的附加功能，比如
 - 对于内容的自适应剪裁
 - 增加计费功能
 - 提供数据缓冲服务
- 两个代理服务器的实现例子
 - MSP – Microsoft Proxy Server
 - squid



Microsoft Proxy Server

- 一个功能强大的代理服务器
 - 提供常用的Internet服务
 - 除了基本的Web Proxy，还有Socks Proxy和Winsock Proxy
 - 强大的cache和log功能
 - 安装管理简单
 - 与NT/2000集成的认证机制
- 扩展的一些功能
 - 反向proxy: proxy作为Internet上的一个Web server, 以虚拟Web Server的方式为后面的Web Server独立地发布到Internet上
 - 安全报警

Microsoft Proxy Server管理示意图

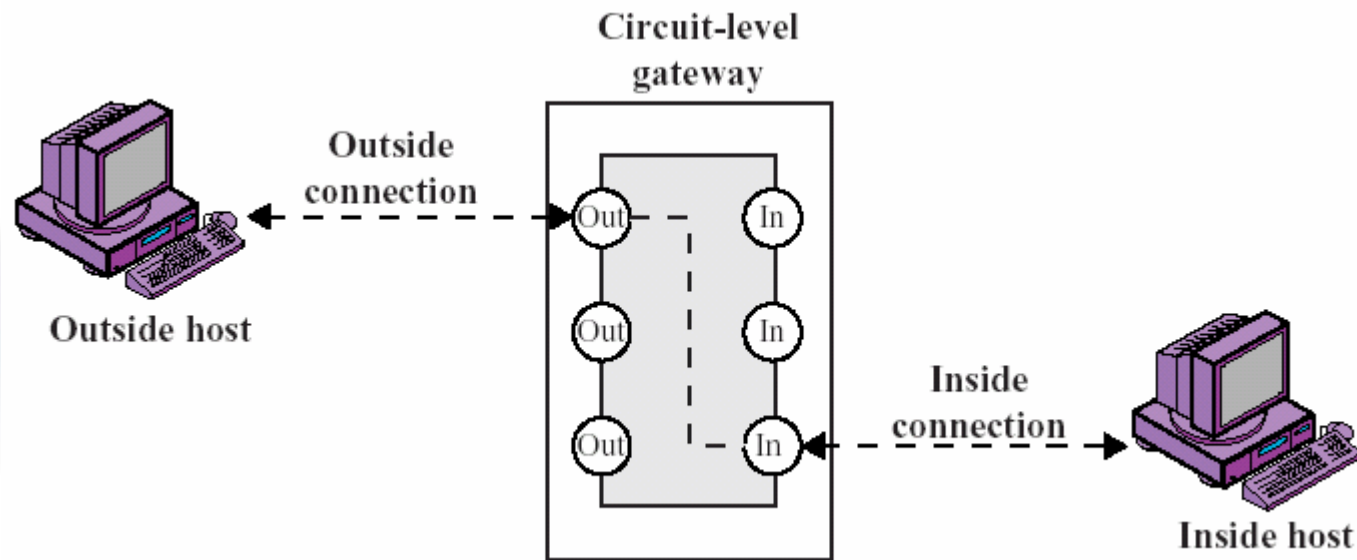




squid

- 关于squid
 - 是一个功能全面的Web Proxy Cache
 - 可以运行在各种UNIX版本上
 - 是一个自由软件，而且源码开放
 - 功能强大，除了http协议之外，还支持许多其它的协议，如ftp、ssl、DNS等代理服务
- 管理和配置
 - /usr/local/squid /etc/squid.conf
 - 默认端口3128
- 一种使用方案
 - Linux+Squid

电路层网关





电路层网关

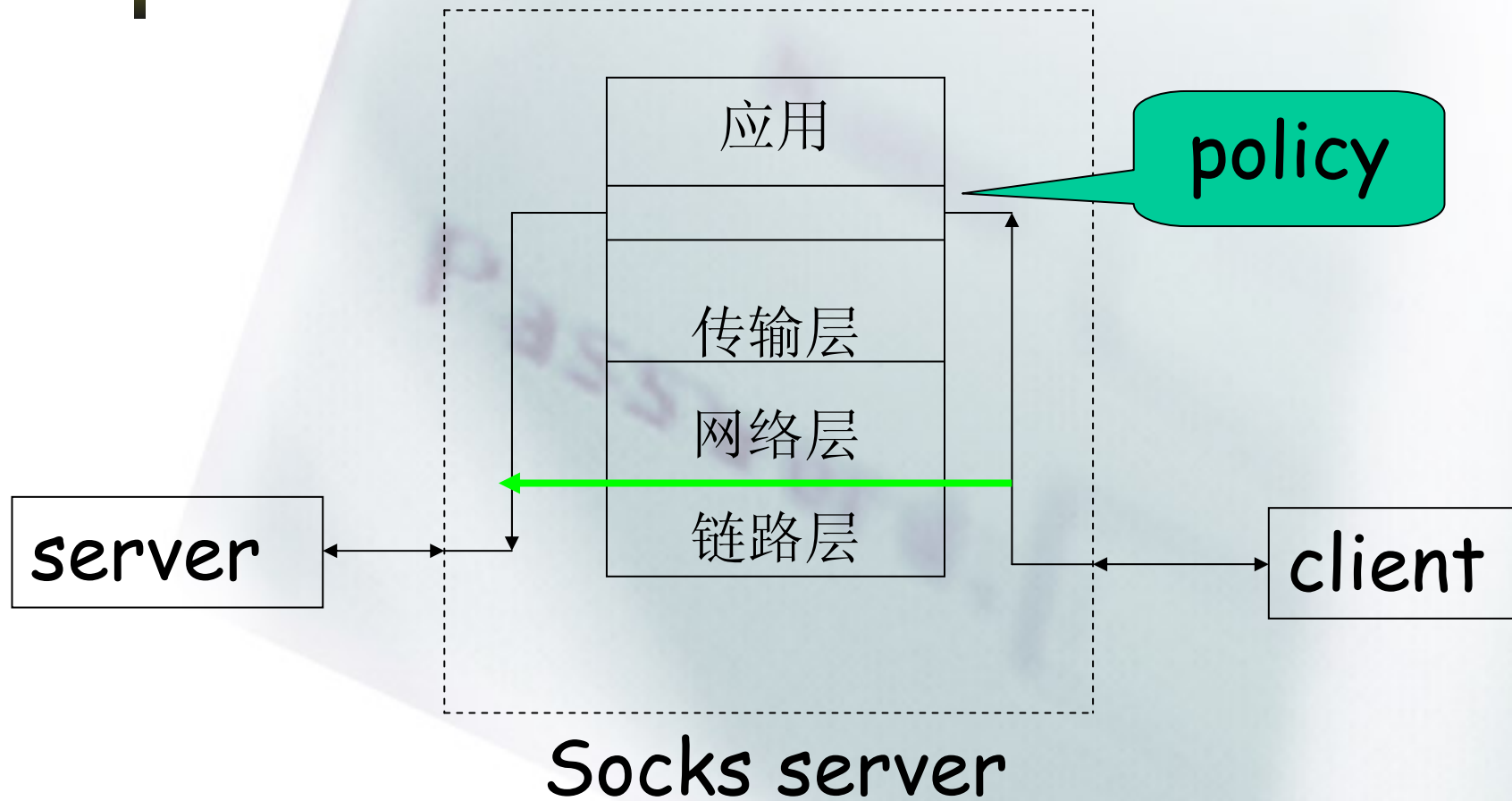
- 本质上，也是一种代理服务器
 - 有状态的包过滤器
 - 动态包过滤器
- 状态
 - 上下文环境
 - 流状态
- 认证和授权方案
- 例子： **socks**



socks

- 专门设计用于防火墙
- 在应用层和传输层之间垫了一层
 - Socks lib和socks server
 - 不支持ICMP
- **Socks v4和socks v5**
 - 基于用户的认证方案
 - 对UDP的支持
- 正在普及和流行
 - 可以参考有关的RFC

socks





Socks v5

- 在socks v4的基础上发展起来
 - Socks v4支持基于TCP的应用穿越防火墙
 - Socks v5增加了对于UDP协议的支持
 - Socks v5增加了对于认证方案的支持
 - Socks v5支持IPv6地址
- Socks要求
 - 修改客户程序，链接到socks library，以便socksified
 - 首先连接到socks server，然后再同目标服务器连接
 - 对于UDP协议，首先同socks server建立一个TCP连接，然后再传送UDP数据，这个连接上传送加了请求头的UDP数据。然后socks服务器建立一个UDP转发进程来发送UDP到远端。



TCP客户的处理过程

- 客户首先与socks server建立一个TCP连接，通常SOCKS端口为1080
- 然后客户与服务器协商认证方案
- 然后进行认证过程
- 认证完成之后
 - 客户发出请求
 - 服务器送回应答
- 一旦服务器应答指示成功，客户就可以传送数据了

认证方案的协商

Client->Server:

-----+	-----+	-----+
VER	NMETHODS	METHODS
-----+	-----+	-----+
1	1	1 to 255
-----+	-----+	-----+

Server->Client

-----+	-----+
VER	METHOD
-----+	-----+
1	1
-----+	-----+

Method:

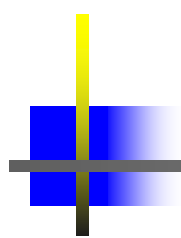
X'00: No Authentication require

X'01: GSSAPI

X'02: Username/password

X'FF: No Acceptable Methods

客户发出的请求



+	-	-	-	+	-	-	-	+	-	-	-	-	+
	VER		CMD		RSV		ATYP		DST.ADDR		DST.PORT		
+	-	-	-	+	-	-	-	+	-	-	-	-	+
	1		1		X'00'		1		Variable		2		
+	-	-	-	+	-	-	-	+	-	-	-	-	+

CMD:

X'01: connection

X'02: bind

X'03: UDP associate

ATYP:

X'01: IP v4

X'03: Domain name

X'04: IP v6

服务器的应答

+	-----+	-----+	-----+	-----+	-----+	-----+	-----+
	VER	REP	RSV	ATYP	BND.ADDR	BND.PORT	
+	-----+	-----+	-----+	-----+	-----+	-----+	-----+
	1	1	X'00'	1	Variable	2	
+	-----+	-----+	-----+	-----+	-----+	-----+	-----+

REP

- 00: succeed
- 01: general SOCKS server failure
- 02: connection not allowed by ruleset
- 03: network unreachable
- 04: host unreachable
- 05: connection refused
- 06: TTL expired
- 07: Command not supported
- 08: address type not supported
- 09-FF: not assigned

BND.ADDR & BND.PORT:

Specify the addr and port of the socks server which will accept an inbound connection

UDP客户的处理过程

- 请求命令为“**UDP associate**”
- 客户 → **UDP relay server** → 目标机器
- 每一个**UDP**包包含一个**UDP**请求头

UDP packet

RSV	FRAG	ATYP	DST.ADDR	DST.PORT	DATA
2	1	1	Variable	2	Variable

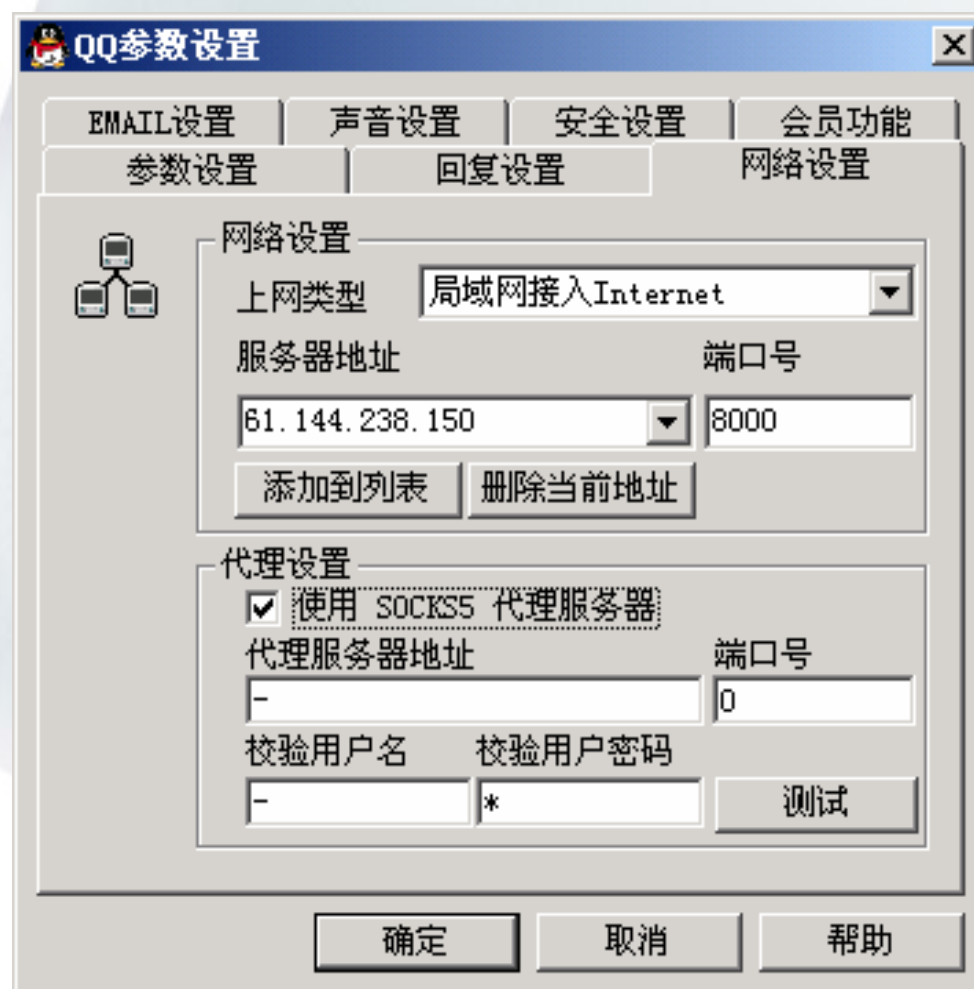
FRAG: current fragment number



电路层网关的优缺点

- 优点
 - 效率高
 - 精细控制，可以在应用层上授权
 - 为一般的应用提供了一个框架
- 缺点
 - 客户程序需要修改

电路层网关需要客户端支持—例子





防火墙的配置

- 几个概念

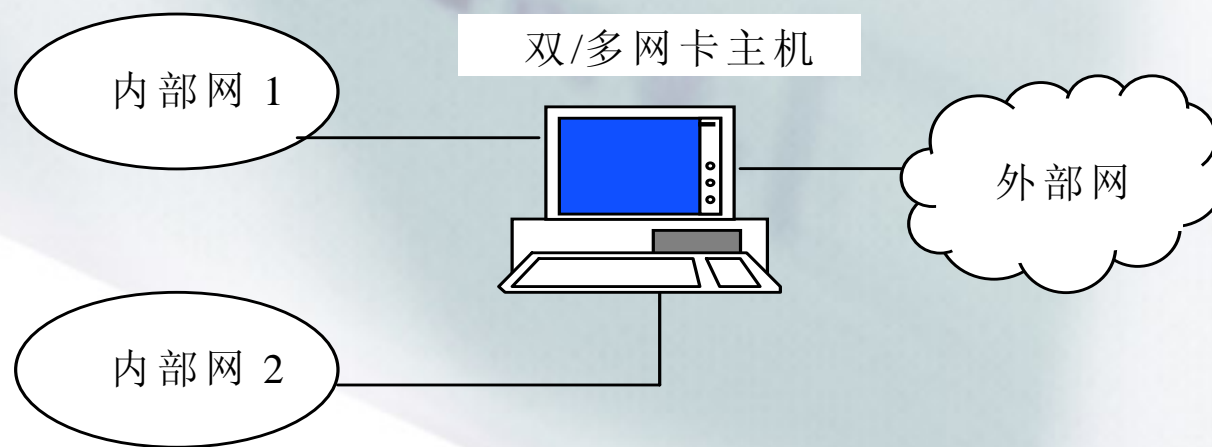
- 堡垒主机(Bastion Host): 堡垒主机是一种配置了安全防范措施的网络上的计算机, 堡垒主机为网络之间的通信提供了一个阻塞点, 也就是说如果没有堡垒主机, 网络之间将不能相互访问
- 双宿主主机(dual-homed host): 至少有两个网络接口的通用计算机系统
- DMZ(Demilitarized Zone, 非军事区或者停火区): 在内部网络和外部网络之间增加的一个子网, 堡垒主机通常位于DMZ中。

防火墙的典型配置模式

- 双宿/多宿主机模式 (**Dual-Homed /Multi-Homed Host Firewall**)
- 屏蔽主机模式(**Screened Host Firewall**)
- 屏蔽子网模式(**Screened Subnet mode**)

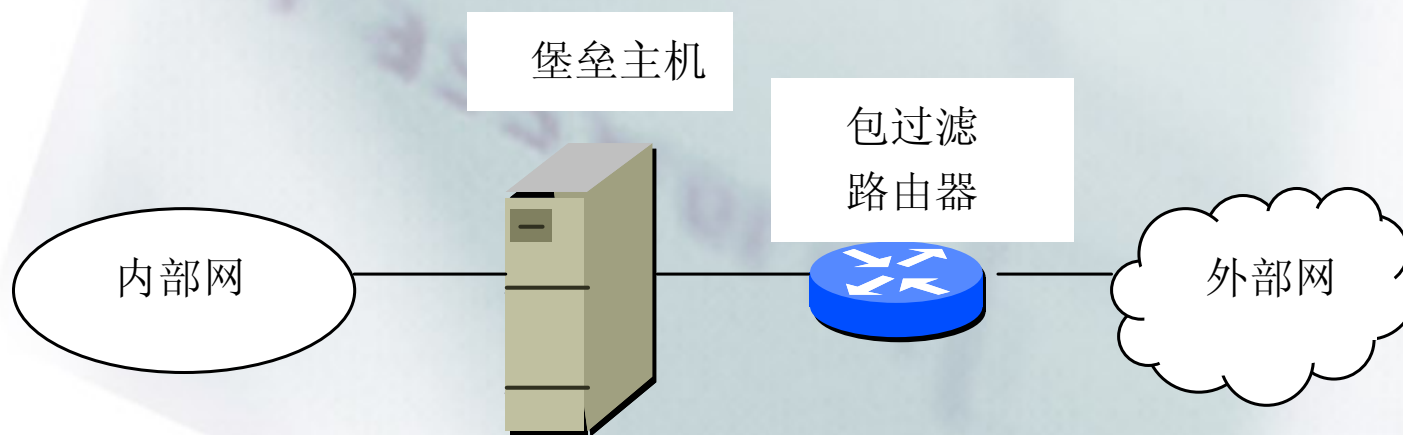
双宿/多宿主机模式

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做防火墙，两块或多块网卡各自与受保护网和外部网相连



屏蔽主机模式

- 屏蔽主机防火墙由包过滤路由器和堡垒主机组成



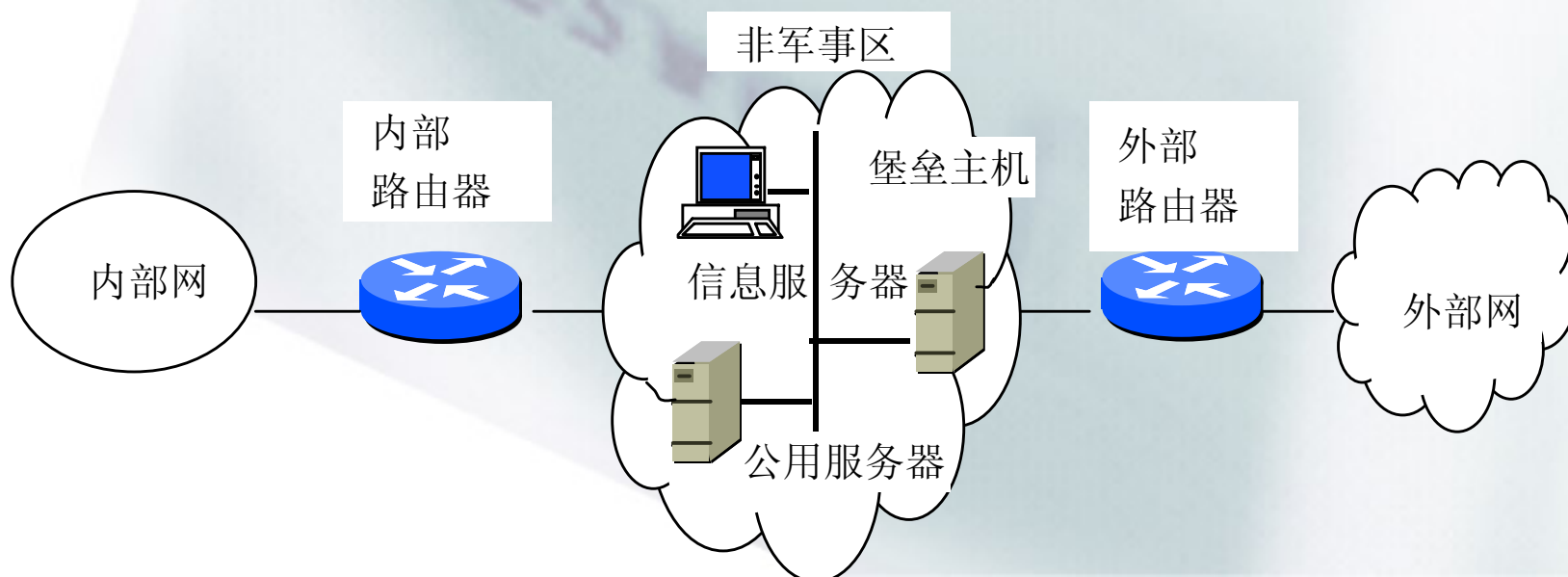


屏蔽主机模式特点

- 在这种方式的防火墙中，堡垒主机安装在内部网络上，通常在路由器上设立过滤规则，并使这个堡垒主机成为从外部网络唯一可直接到达的主机，这确保了内部网络不受未被授权的外部用户的攻击。
- 屏蔽主机防火墙实现了网络层和应用层的安全，因而比单独的包过滤或应用网关代理更安全。
- 在这一方式下，过滤路由器是否配置正确是这种防火墙安全与否的关键，如果路由表遭到破坏，堡垒主机就可能被越过，使内部网完全暴露。

屏蔽子网模式

- 采用了两个包过滤路由器和一个堡垒主机，在内外网络之间建立了一个被隔离的子网，定义为“非军事区（**de-militarized zone**）”网络，有时也称作周边网（**perimeter network**）



屏蔽子网模式特点

- 网络管理员将堡垒主机，**WEB**服务器、**Mail**服务器等公用服务器放在非军事区网络中。
- 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，即使堡垒主机被入侵者控制，内部网仍受到内部包过滤路由器的保护。
- 堡垒主机运行各种代理服务

防火墙的发展

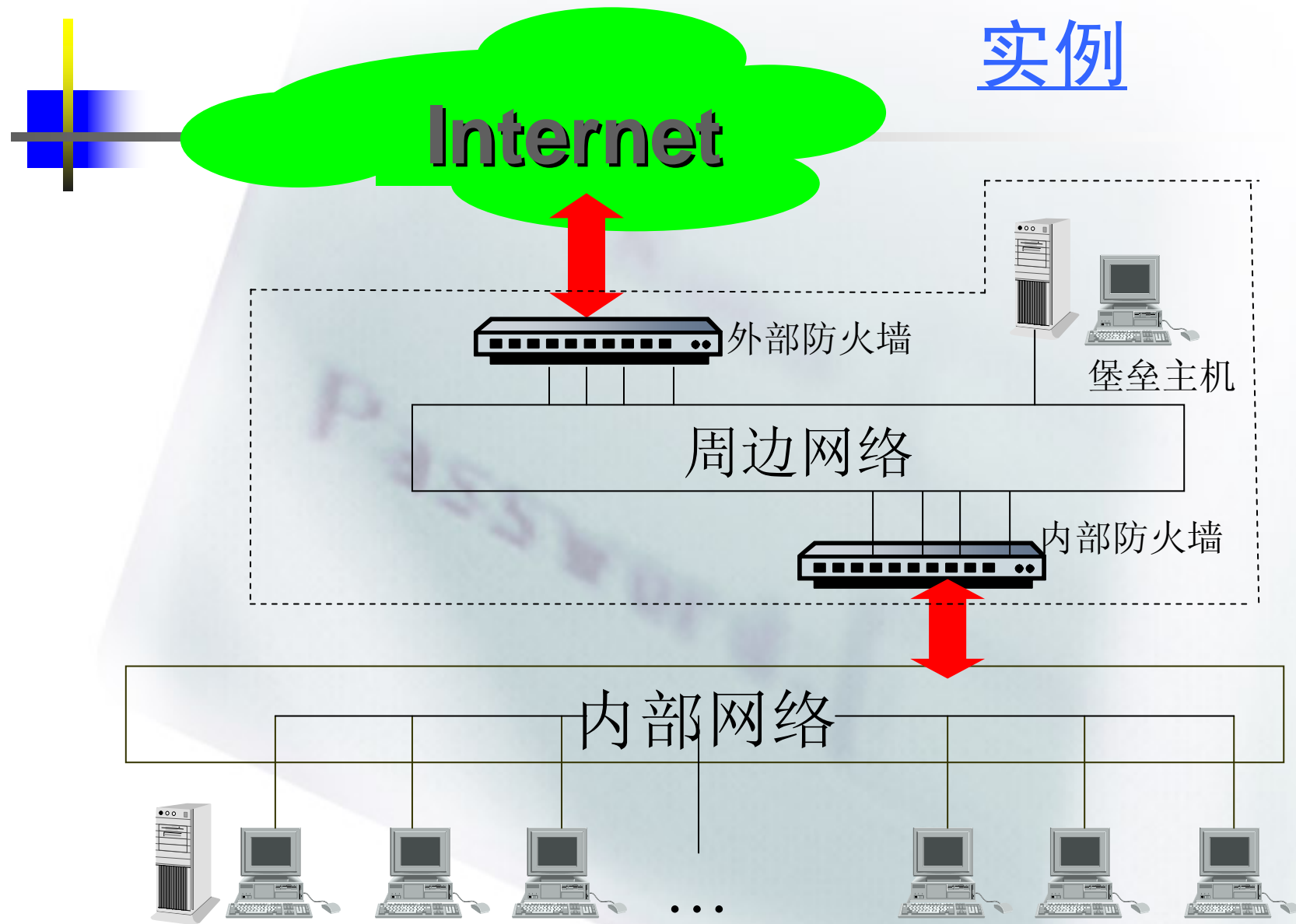
- 分布式防火墙
- 应用层网关的进一步发展
 - 认证机制
 - 智能代理
- 与其他技术的集成
 - 比如NAT、VPN(IPSec)、IDS，以及一些认证和访问控制技术
 - 防火墙自身的安全性和稳定性



分布式防火墙

- 传统防火墙技术的几个问题
 - 依赖于防火墙一端可信，另一端是潜在的敌人
 - Internet的发展使从外部穿过防火墙访问内部网的需求增加了
 - 一些内部主机需要更多的权限
 - 过于依赖物理拓扑结构
- 方法
 - 主要工作防护工作在主机端
 - 由安全策略来划分内外网，不单纯依靠物理位置来划分内外
 - 策略描述语言：说明什么连接允许，什么连接不允许
 - 一系列系统管理工具：用于将策略发布到每一主机处，以保证机构的安全

实例





内容

- **1. Friewall**
- **2. SSL/TLS**
- **3. IPSec/VPN**
- **4. IDS**

SSL/TLS协议

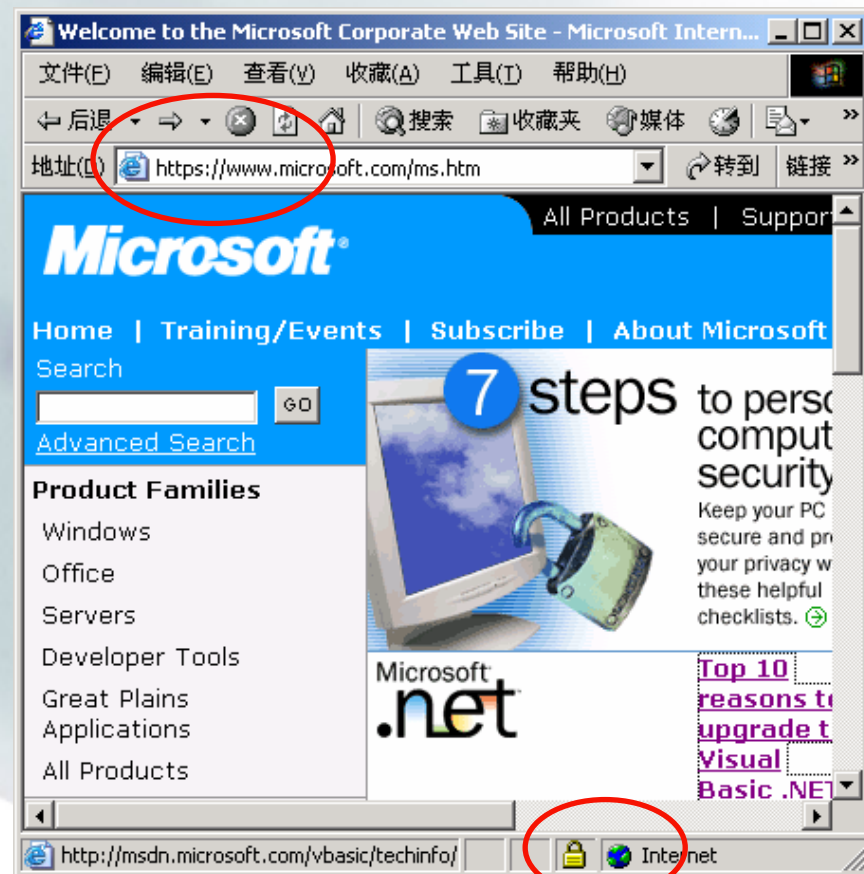
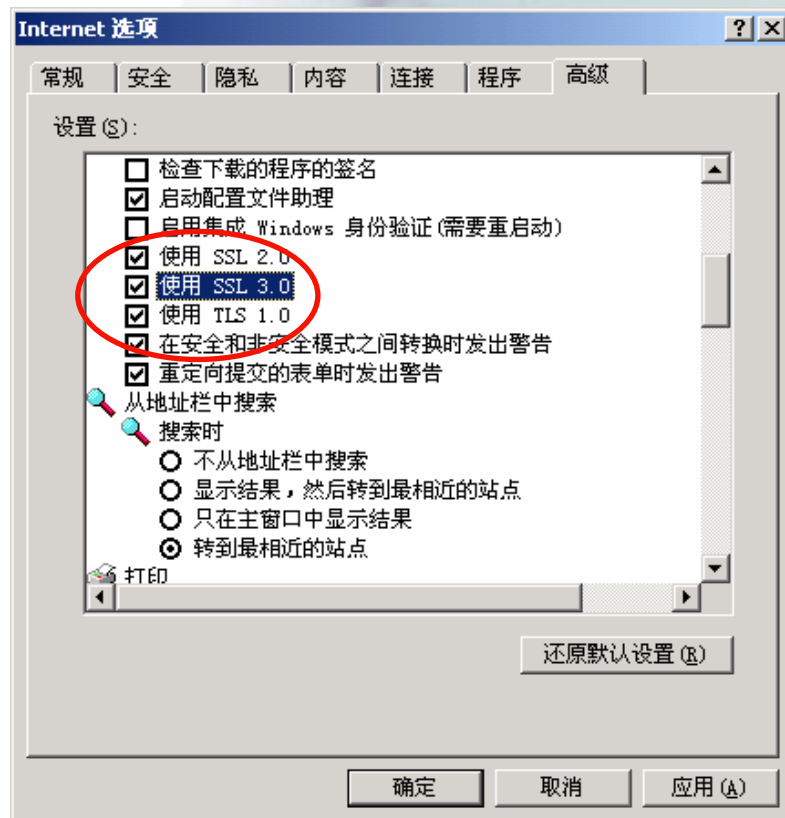
- **1994年Netscape开发了SSL(Secure Socket Layer)协议，专门用于保护Web通讯**
- **版本和历史**
 - 1.0，不成熟
 - 2.0，基本上解决了Web通讯的安全问题
 - Microsoft公司发布了PCT(Private Communication Technology)，并在IE中支持
 - 3.0，1996年发布，增加了一些算法，修改了一些缺陷
 - TLS 1.0(Transport Layer Security, 也被称为SSL 3.1)，1997年IETF发布了Draft，同时，Microsoft宣布放弃PCT，与Netscape一起支持TLS 1.0
 - 1999年，发布RFC 2246(The TLS Protocol v1.0)

SSL/TLS协议

- 协议的设计目标

- 为两个通讯个体之间提供保密性和完整性(身份认证)
- 互操作性、可扩展性、相对效率

- 协议的使用



SSL/TLS概况

- 协议分为两层
 - 底层：TLS记录协议
 - 上层：TLS握手协议、TLS密码变化协议、TLS警告协议
- TLS记录协议
 - 建立在可靠的传输协议(如TCP)之上
 - 它提供连接安全性，有两个特点
 - 保密性，使用了对称加密算法
 - 完整性，使用HMAC算法
 - 用来封装高层的协议

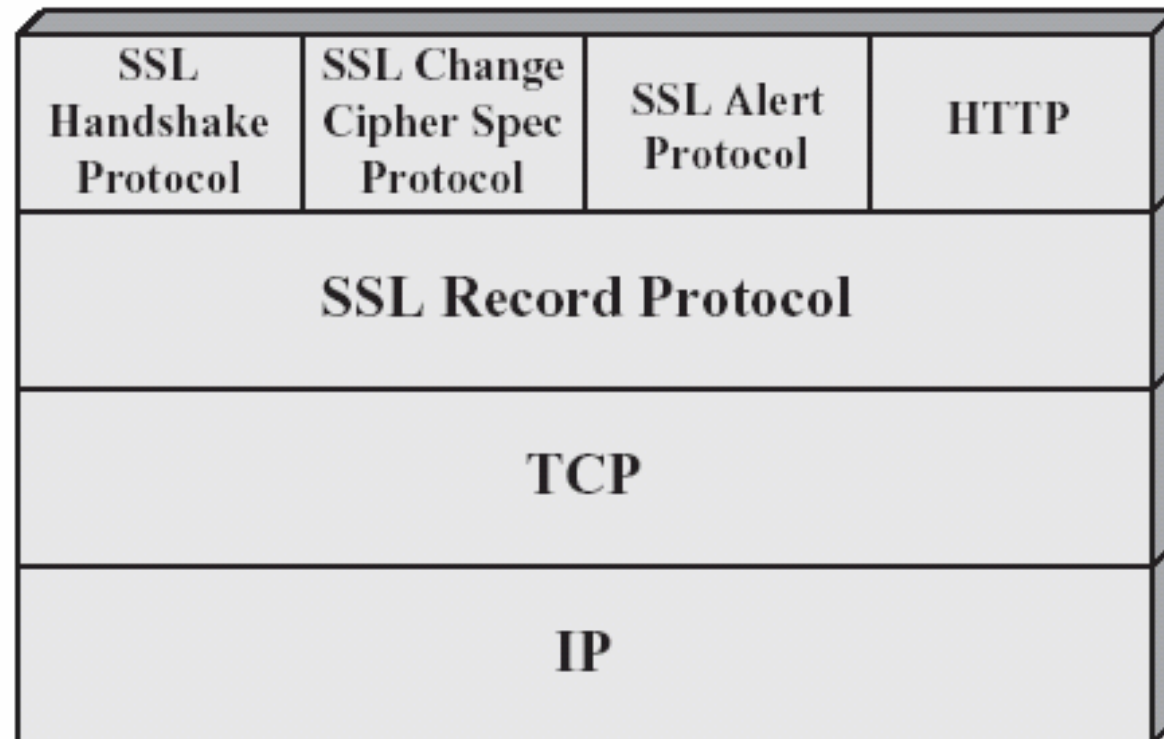


SSL/TLS概况

- **TLS握手协议**
 - 客户和服务端之间相互认证
 - 协商加密算法和密钥
 - 它提供连接安全性，有三个特点
 - 身份认证，至少对一方实现认证，也可以是双向认证
 - 协商得到的共享密钥是安全的，中间人不能够知道
 - 协商过程是可靠的

SSL/TLS协议栈

- 为上层协议提供安全性
 - 保密性
 - 身份认证和数据完整性





TLS会话与连接

- **TLS Session:**

指客户和服务端之间的一个关联关系。通过TLS握手协议创建session，它确定了一组密码算法的参数。Session可以被多个连接共享，从而可以避免为每个连接协商新的安全参数而带来 昂贵的开销。

- 每个TLS Session都有一个当前状态

- **TLS connection**

- 与底层协议的点对点连接相关联
- 每个connection都与一个session相关联
- 连接是短暂的

TLS会话的状态

- 实际上是一组参数，包括
 - Session identifier, 字节序列，由服务器产生，用来标识一个会话状态
 - Peer certificate(可以为NULL), 对方的X.509 v3证书
 - Compression method, 加密之前用来压缩数据的算法
 - Cipher spec, 密文规约，指定数据加密算法和用于HMAC的散列算法，以及算法的有关参数
 - Master secret, 主密码，客户和服务端之间共享的48字节的密码（共享秘密）
 - Is reusable, 标记是否这个会话可以被用来初始化一个新的连接



TLS连接的状态

连接状态包含的参数：

Server and client random，客户和服务端为每个连接选择的字节序列

Server write MAC secret，服务器在发送数据时，用于MAC运算的密钥

Client write MAC secret，客户在发送数据时，用于MAC运算的密钥

Server write key，服务器加密数据的密钥，以及客户解密数据的密钥

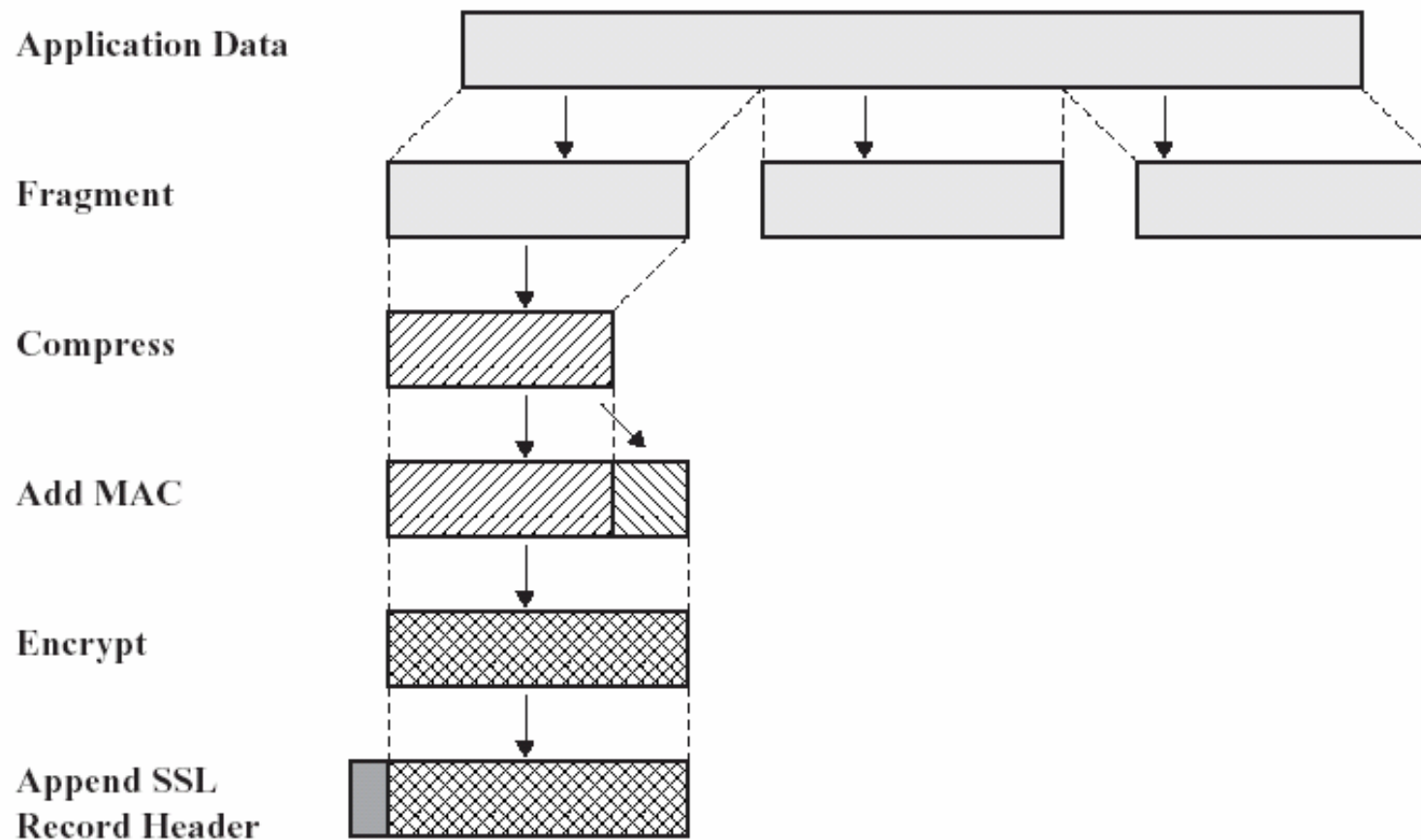
Client write key，客户加密数据的密钥，以及服务器解密数据的密钥

Initialization vectors，在CBC模式中用到的IV，最初由握手协议初始化，以后，每一个记录的最后密文块被用作下一个记录的IV

Sequence numbers，每一方为每一个连接维持一个单独的序列号，当发送或接受密码变化协议报文时，重置为0

TLS Record Protocol

- 操作过程示意图



TLS记录协议中的操作

- **第一步, fragmentation**
 - 上层消息的数据被分片成 2^{14} (16384) 字节大小的块, 或者更小
- **第二步, compression(可选)**
 - 必须是无损压缩, 如果数据增加的话, 则增加部分的长度不超过1024字节。对于非常短的数据, 由于压缩格式的约定, 压缩算法的输出可能长于输入。
- **第三步, 计算消息认证码(MAC)**
 - 计算公式:

HMAC_hash(MAC_write_secret, seq_num	
更高层的协议	TLSCompressed.type
版本	TLSCompressed.version
这个压缩分片 (明文) 的长度	TLSCompressed.length
压缩分片或明文	TLSCompressed.fragment)

TLS记录协议中的操作

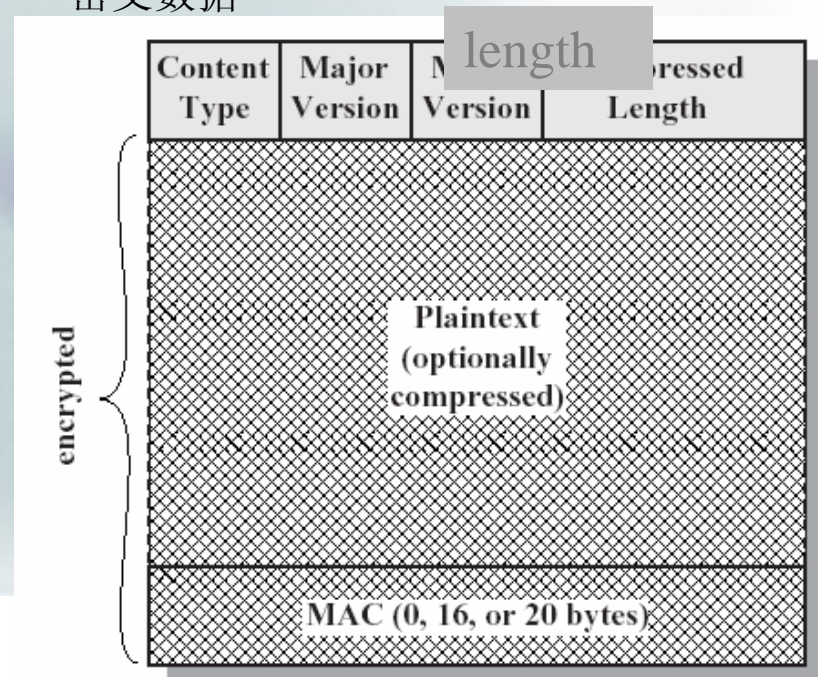
- 第四步，**encryption**

- 采用CBC，算法由cipher spec指定
- 加密后长度增长不能超过1024，所以最后数据长度不会超过 $2^{14}+2048$ 字节，包括
 - 加密之后的数据内容
 - HMAC
 - padding_length 分组密码算法加密之前要进行填充，添为分组长度的整数倍
 - padding, 共padding_length个字节，每个字节的值也是padding_length
- IV，初始协商指定，以后，前后记录连接起来
- 说明：如果是流密码算法，则不需要padding

TLS记录协议的处理结果

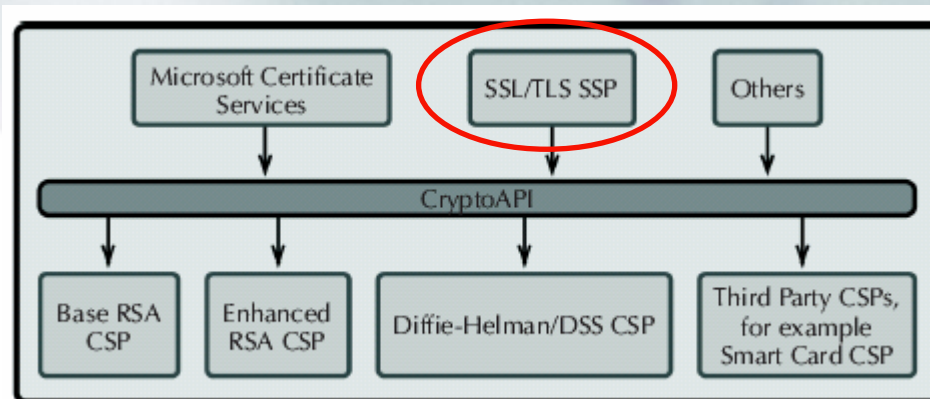
- 第五步，添加TLS纪录首部, 结果如下：

```
struct {  
    ContentType      type;    —— 8位，上层协议类型  
    ProtocolVersion  version; —— 16位，主次版本  
    uint16           length;  —— 加密后数据的长度，  
                                不超过 $2^{14}+2048$ 字节  
    EncryptedData    fragment; —— 密文数据  
} TLSCiphertext;
```



SSL实现

- **OpenSSL, 实现了SSL(2,3), TLS(1.0)**
 - openssl —— a command line tool.
 - ssl(3) —— the OpenSSL SSL/TLS library.
 - crypto(3) —— the OpenSSL Crypto library.
 - **URL: <http://www.openssl.org>**
- **SSLeay**
 - <http://www2.psy.uq.edu.au/~ftp/Crypto/>
- **Microsoft Win2k SSL implementation**





内容

- **1. Friewall**
- **2. SSL/TLS**
- **3. IPSec/VPN**
- **4. IDS**



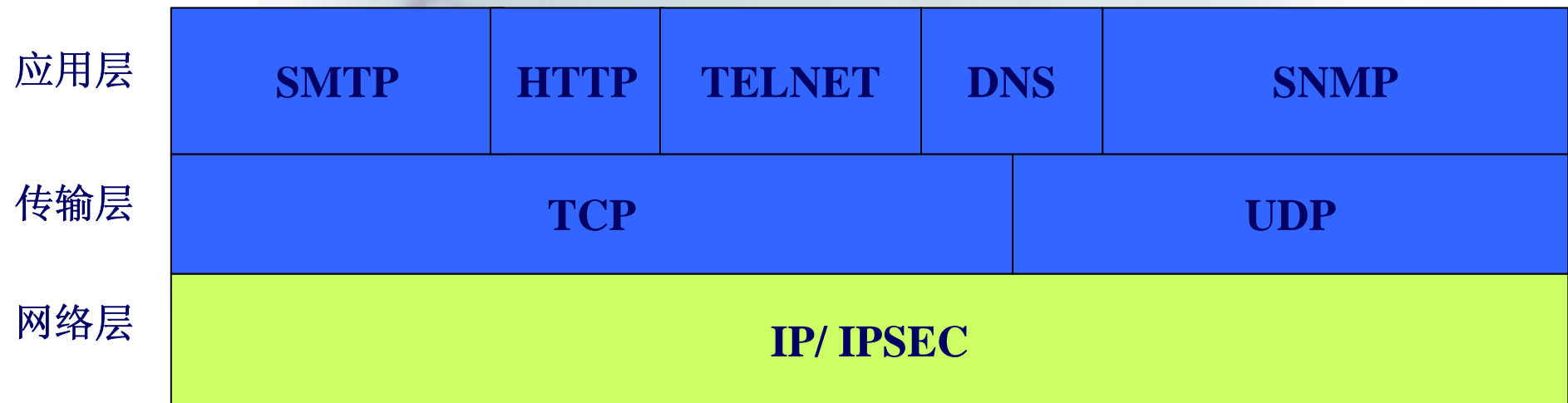
IPSec

- **网络层安全性**
 - 需求：真实性(认证)、完整性和保密性
 - 好处：对于应用层透明
 - 可以针对链路，也可以针对最终用户
 - 可以实现在防火墙或者路由器上
 - 弥补IPv4在协议设计时缺乏安全性考虑的不足
- **IPSec是IETF定义的一组安全IP协议集**
 - 它在IP包这一级为IP通讯业务提供保护
 - IPv4中是一项可选支持的服务，在IPv6中是一项必须支持的服务

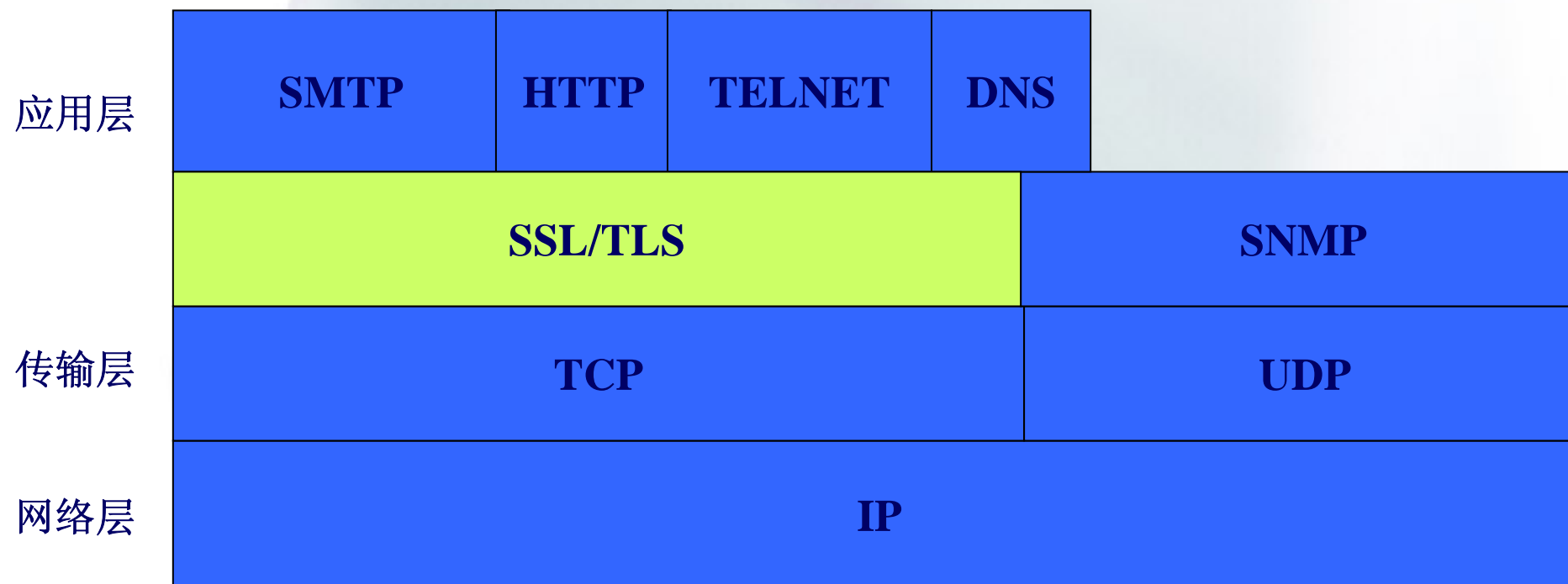
Internet安全性途径

- 网络层——IP 安全性(IPSec)
- 传输层—— SSL / TLS
- 应用层——S/MIME, PGP, PEM, SET, Kerberos, SHTTP, SSH

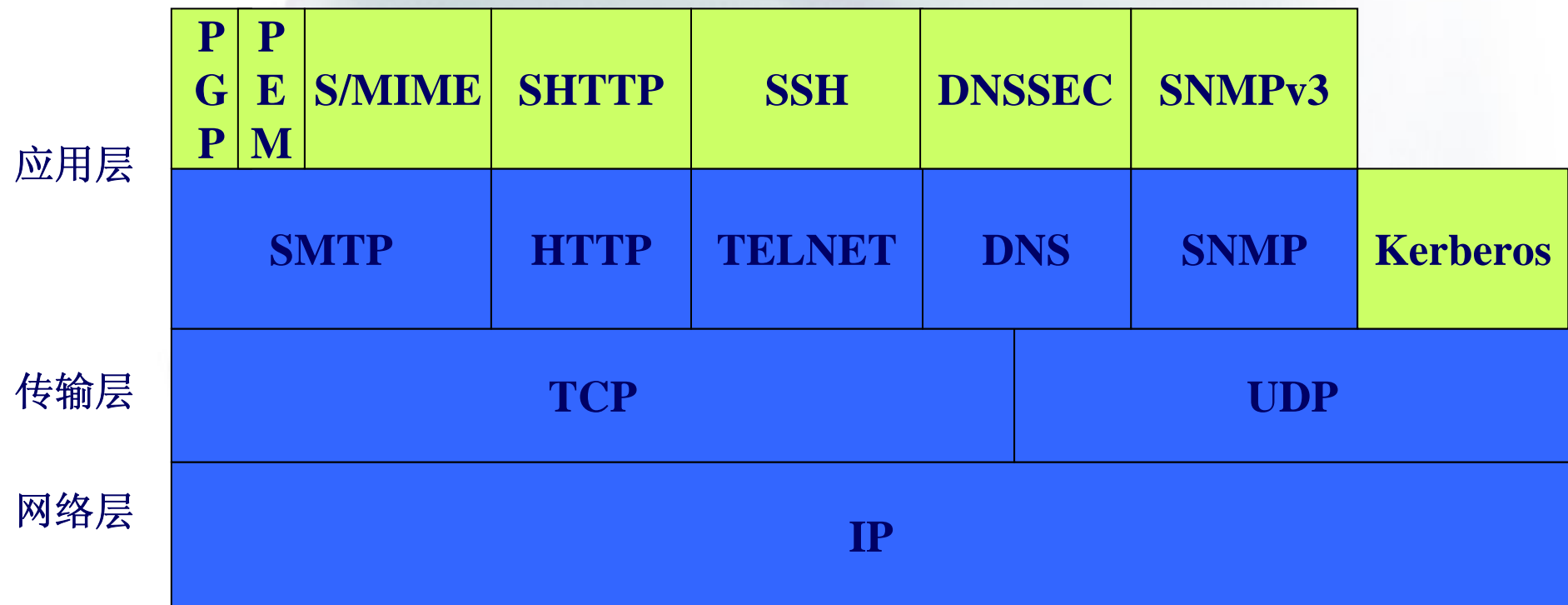
网络层安全



传输层安全



应用层安全



IPv4头

4位版本号	4位报头长度	8位服务类型（TOS）	16位总长度（以8位组为单位）	
16位标识			3位标志	13位分段移位
8位生存期	8位协议	16位报头校验和		
32位源IP地址				
32位目的IP地址				
选项			填充	

20个八位组

TOS

IPv4的缺陷

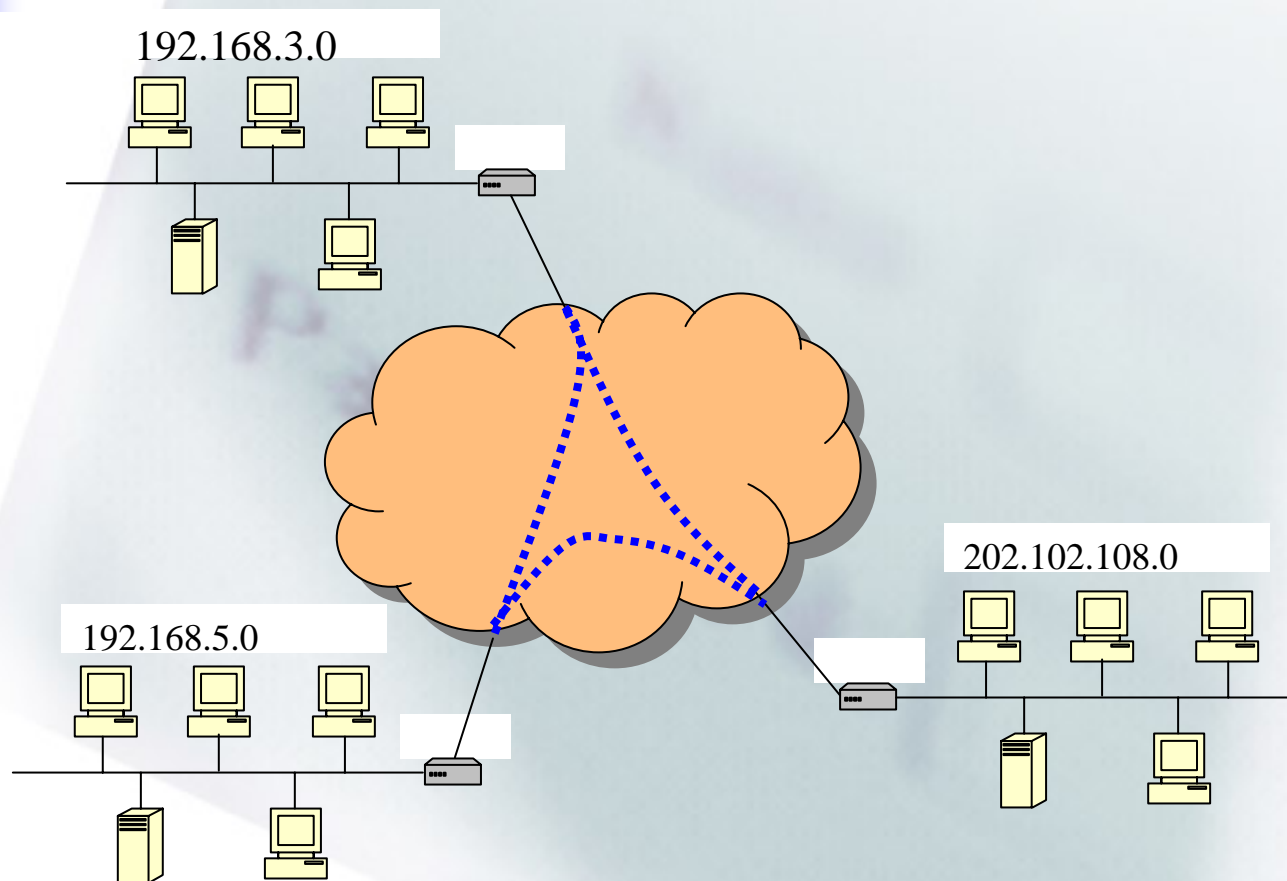
- 缺乏对通信双方身份真实性的鉴别能力
- 缺乏对传输数据的完整性和机密性保护的机制
- 由于**IP**地址可软件配置以及基于源**IP**地址的鉴别机制，**IP**层存在被监听、**IP**地址欺骗、信息泄露和数据项篡改等攻击



IPSec的应用

- **IPSec为在LAN、WAN和Internet上的通讯提供安全性**
 - 分支办公机构通过Internet互连。(Secure VPN)
 - 通过Internet的远程访问。
 - 与合作伙伴建立extranet与intranet的互连。
 - 增强电子商务安全性
- **IPSec的主要特征是可以支持IP层所有流量的加密和/或鉴别。因此可以增强所有分布式应用的安全性。**

IPSec的应用—VPN



IPSec提供的服务

- **IPSec在IP层提供安全服务**，使得系统可以选择所需要的安全协议，确定该服务所用的算法，并提供安全服务所需任何加密密钥。

	AH	ESP(仅加密)	ESP(加密+鉴别)
访问控制	√	√	√
无连接完整性	√		√
数据源鉴别	√		√
拒绝重放分组	√	√	√
机密性		√	√
有限通信量的机密性		√	√



IPSec的内容

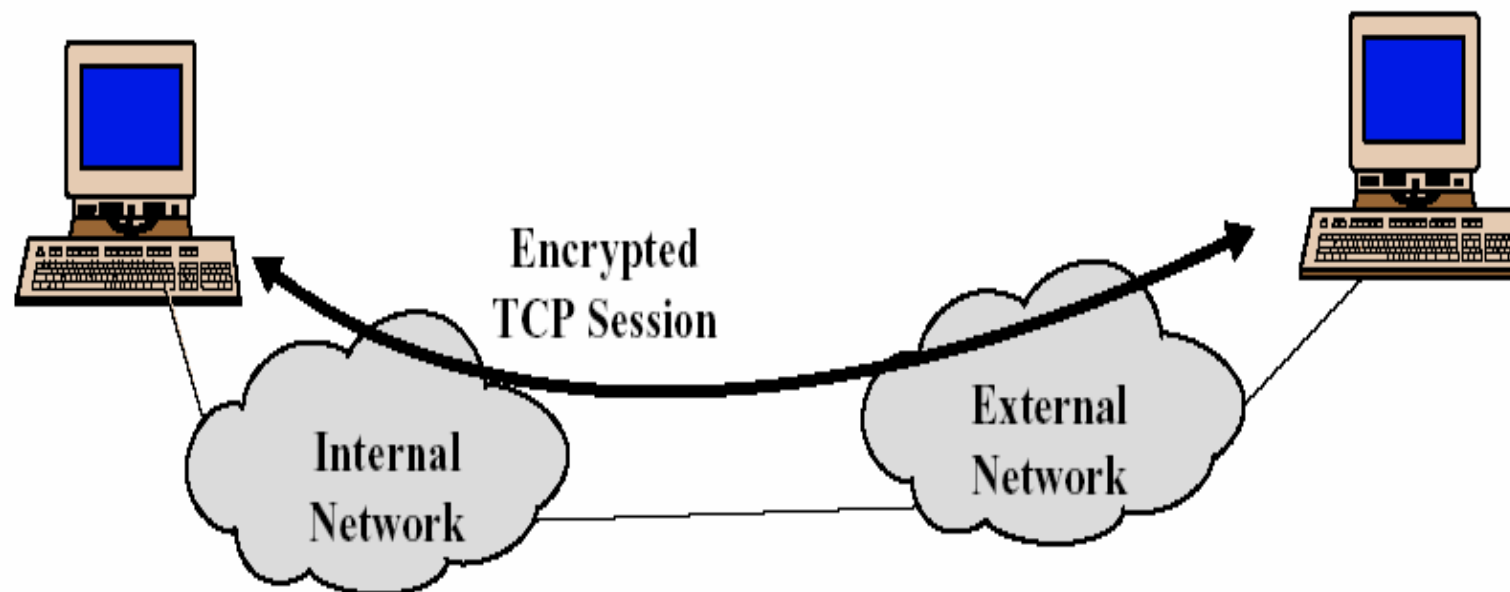
- 协议部分，分为
 - AH: Authentication Header
 - ESP: Encapsulating Security Payload
- 密钥管理(**Key Management**)
 - SA(Security Association)
 - ISAKMP定义了密钥管理框架
 - IKE是目前正式确定用于IPSec的密钥交换协议



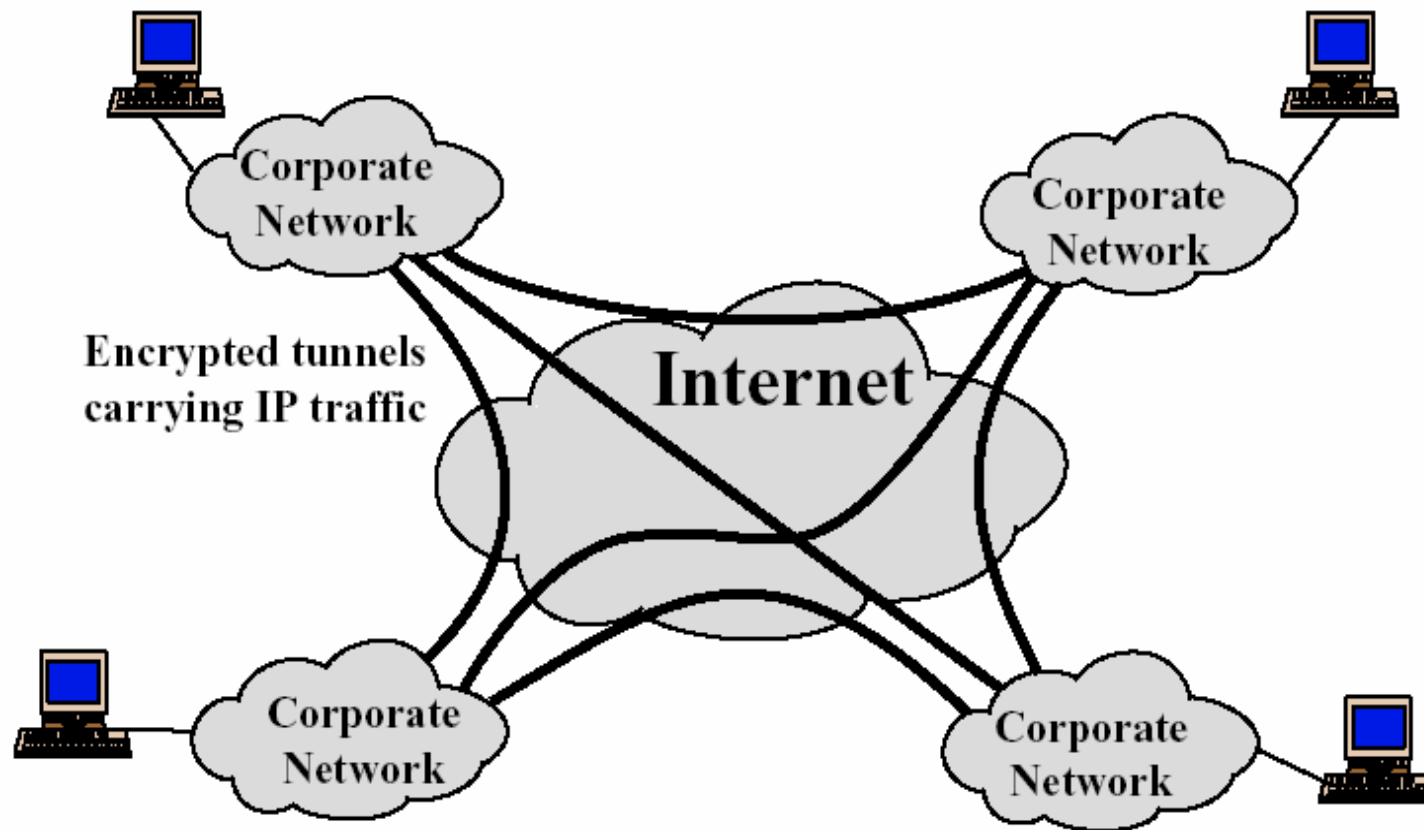
SA(Security Association)

- 基本概念
 - SA是发送者和接收者两个IPSec系统之间的一个简单的单向逻辑连接，是与给定的一个网络连接或一组网络连接相关联的安全信息参数集合
 - 因为SA是单个方向的，所以，对于一个双向通信，则需要两个SA
- 每个SA通过三个参数来标识
 - SPI(Security Parameters Index)
 - 目标地址IP
 - 安全协议标识

传输方式的加密



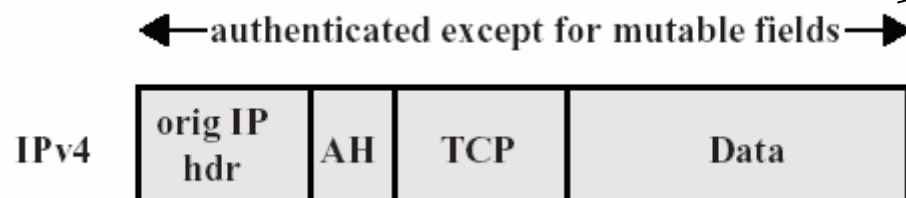
隧道方式的加密



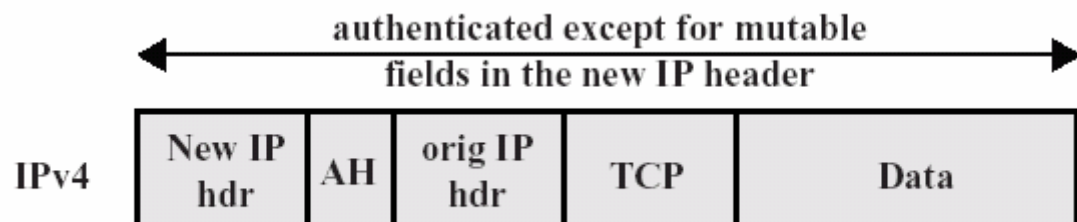
AH两种模式示意图



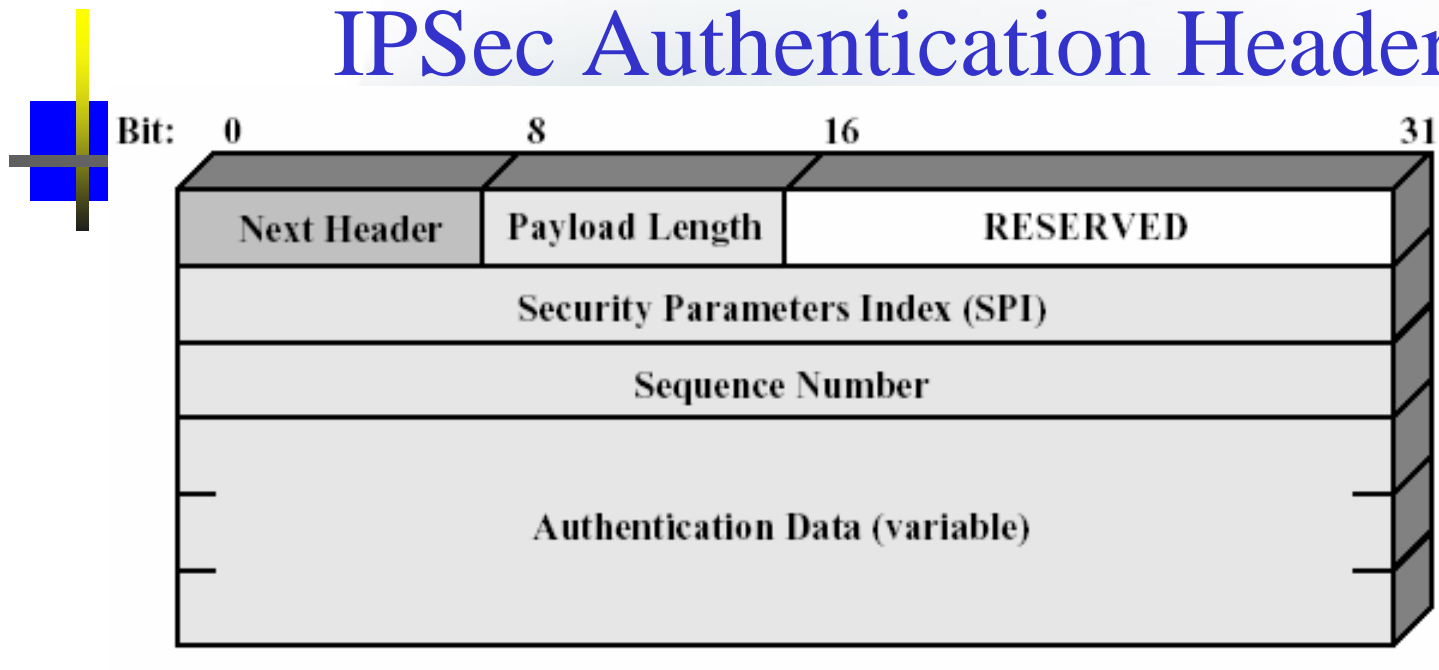
传输模式



隧道模式



IPSec Authentication Header



- **Next Header:** 下一个头的类型
- **Payload Length:** AH的长度(32位字为单位)
- **SPI:** 用来标识SA
- **Sequence Number:** 用来避免重放攻击
- **Authentication Data:** 可变长度的域，包含针对这个包的ICV或者MAC

AH处理过程

- **AH定位**
 - 在IP头之后，在上层协议数据之前
- **认证算法**
 - 计算ICV或者MAC
- **对于发出去的包(Outbound Packet)的处理，构造AH**
 - 查找SA
 - 产生序列号
 - 计算ICV(Integrity Check Value)
 - 内容包括：IP头中部分域、AH自身、上层协议数据
 - 分片

AH处理过程(续)

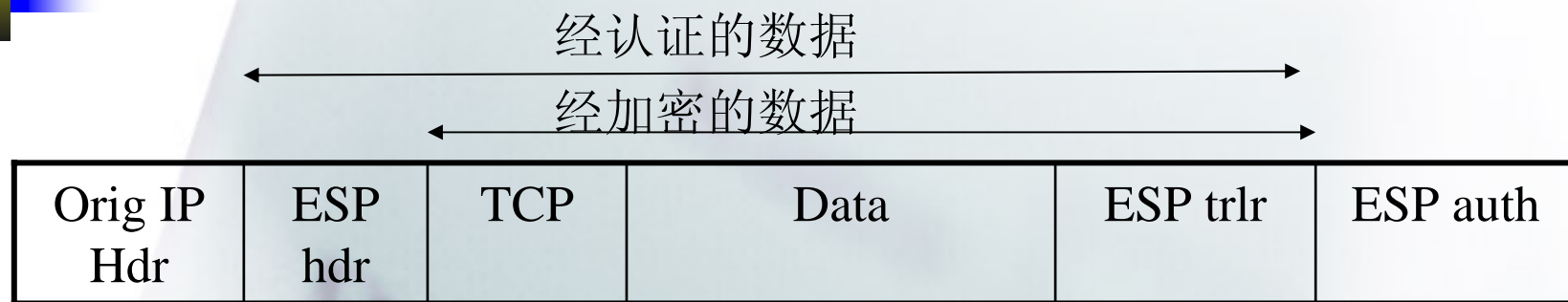
- 对于接收到的包(Inbound Packet)的处理
 - 分片装配
 - 查找SA
 - 依据：目标IP地址、AH协议、SPI
 - 检查序列号(可选，针对重放攻击)
 - 使用一个滑动窗口来检查序列号的重放
 - ICV检查



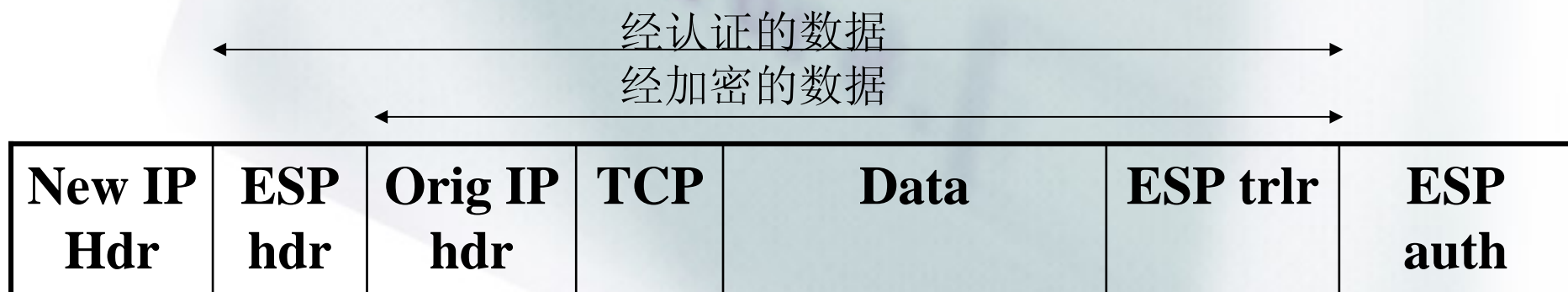
ESP(Encapsulating Security Payload)

- 提供保密功能，也可以提供认证服务
- 将需要保密的用户数据进行加密后再封装到一个新的IP包中，ESP只认证ESP头之后的信息
- 加密算法和认证算法由SA指定
- 也有两种模式：传输模式和隧道模式

ESP两种模式示意图

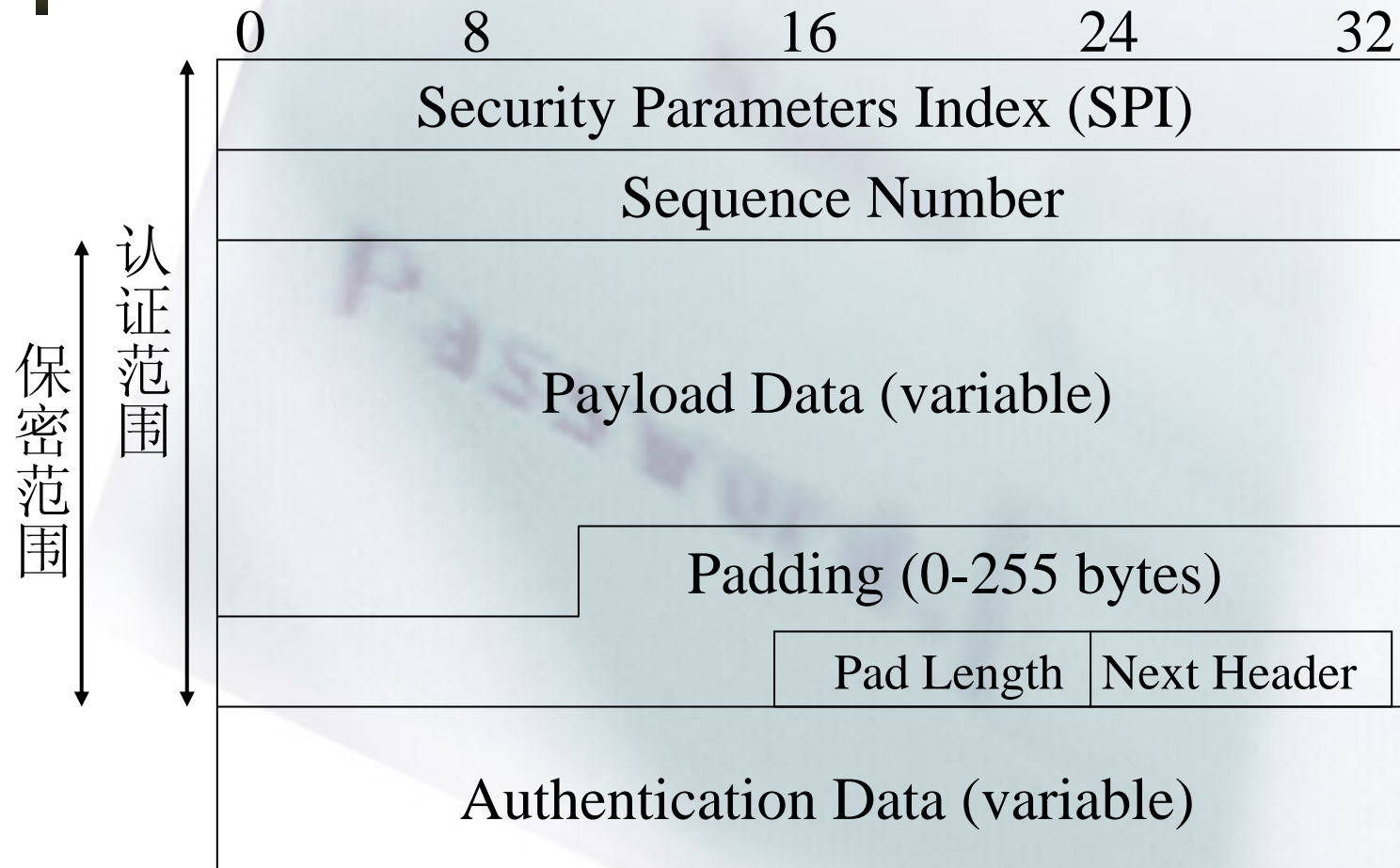


(1) 传输模式



(2) 隧道模式

IPSec ESP格式





ESP处理过程

- **ESP头定位**
- 加密算法和认证算法由SA确定
- 对于发出去的包(Outbound Packet)的处理
 - 查找SA
 - 加密
 - 封装必要的的数据，放到payload data域中
 - 不同的模式，封装数据的范围不同
 - 增加必要的padding数据
 - 加密操作
 - 产生序列号
 - 计算ICV，注意，针对加密后的数据进行计算
 - 分片



ESP处理过程

- 对于接收到的包(Inbound Packet)的处理
 - 分片装配
 - 查找SA
 - 依据：目标IP地址、ESP协议、SPI
 - 检查序列号(可选，针对重放攻击)
 - 使用一个滑动窗口来检查序列号的重放
 - ICV检查
 - 解密
 - 根据SA中指定的算法和密钥、参数，对于被加密部分的数据进行解密
 - 去掉padding
 - 重构原始的IP包

AH和ESP的典型组合

Transport

Tunnel

- 1. [IP1][AH][upper]
- 2. [IP1][ESP][upper]
- 3. [IP1][AH][ESP][upper]

- 4. [IP2][AH][IP1][upper]
- 5. [IP2][ESP][IP1][upper]
- 6. [IP2][ESP][IP1][AH][upper]

- 这里upper指上层协议数据
- IP1指原来的IP头
- IP2指封装之后的IP头

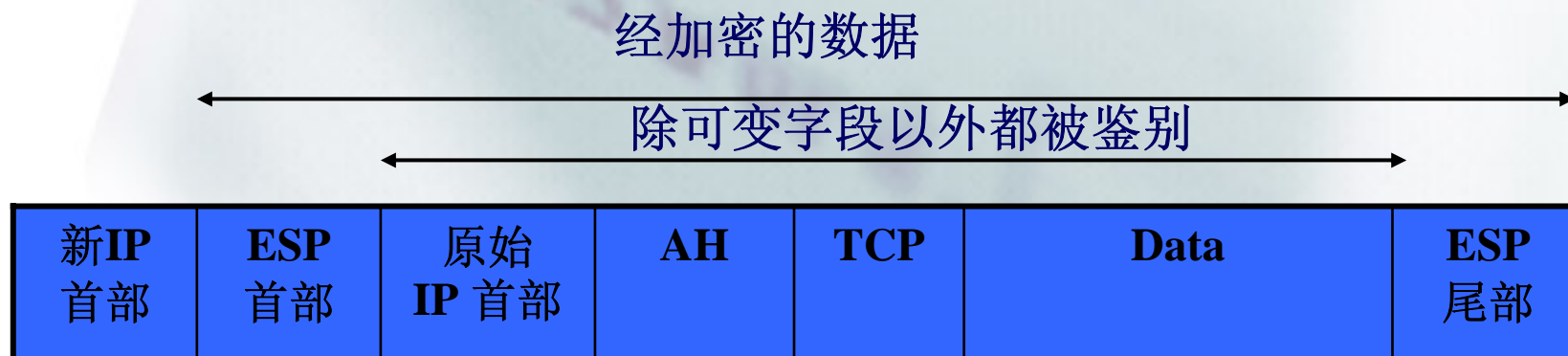
AH与ESP的组合

- 传输邻接
使用两个捆绑的传输SA，内部是ESP SA（没有鉴别选项），外部是AH SA



AH与ESP的组合

- 传输隧道束
加密之前进行鉴别
内部的AH传输SA +外部的ESP隧道SA



IPSec密钥管理

- **ISAKMP: Internet Security Association and Key Management Protocol**
 - RFC 2408
 - 是一个针对认证和密钥交换的框架
- **IKE: The Internet Key Exchange**
 - 基于ISAKMP框架
 - 结合了Oakley和SKEME的部分密钥交换技术

ISAKMP

- 基本的需求
 - 商定SA
 - 确定身份，密钥交换
 - 与协议、算法、厂商都无关
- 框架结构
 - 针对身份认证和密钥交换的协商过程
 - 定义了基本的消息结构
 - 定义了各种消息类型和payload结构
 - 提供了几种缺省的密钥交换模型



两阶段协商

- **1. 建立起ISAKMP SA**
 - 双方(例如ISAKMP Servers)商定如何保护以后的通讯
 - 此SA将用于保护后面的protocol SA的协商过程
- **2. 建立起针对其他安全协议的SA(如IPSec SA)**
 - 这个阶段可以建立多个SA
 - 此SA将被相应的安全协议用于保护数据或者消息的交换



两阶段协商的好处

- 对于简单的情形，两阶段协商带来了更多的通讯开销
- 但是，它有以下一些好处
 - 第一阶段的开销可以分摊到多个第二阶段中
所以，这允许多个SA建立在同样的ISAKMP SA基础上
 - 第一阶段商定的安全服务可以为第二阶段提供安全特性
 - 例如，第一阶段的ISAKMP SA提供的加密功能可以为第二阶段提供身份认证特性，从而使得第二阶段的交换更为简单
 - 两阶段分开，提供管理上的便利

IKE中几个概念

- **PFS: Perfect Forward Secrecy**
 - 一旦一个密钥被泄漏，只会影响到被一个密钥保护的数据
- **Phase**
 - 同ISAKMP中的phase
- **Group**
 - Diffie-Hellman密钥交换参数
- **Mode**
 - 来自Oakley中的定义
 - 指一个密钥交换过程
 - 四个mode
 - Main Mode, Aggressive Mode用于phase 1
 - Quick Mode用于phase 2
 - New Group Mode用在phase 1之后，为将来的协商商定一个新的组

IPSec和IKE小结

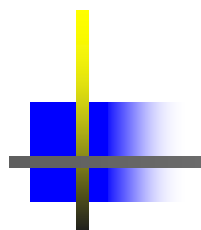
- **IPSec**在网络层上提供安全服务
 - 定义了两个协议AH和ESP
- **IKE**提供了密钥交换功能
 - 利用ISAKMP提供的框架、以及Oakley和SKEME的密钥交换协议的优点
- 通过SA把两部分连接起来
- 已经发展成为Internet标准
- 一些困难
 - IPSec太复杂
 - 协议复杂性，导致很难达到真正的安全性
 - 管理和配置复杂，使得安全性下降
 - 实现上的兼容性
 - 攻击：DOS，网络层之下的协议、之上的协议的弱点

IPSec实现

- 一些网络设备厂商
 - 比如CISCO等
- 各种操作系统
 - 例如Linux、各种UNIX版本
- **IPv6实现**
- 一些**VPN**软件包
-

VPN实例—LinkSys





VPN实例—LinkSys

Advanced Settings for Selected IPsec Tunnel

Tunnel 1

Phase 1:

Operation mode : ☒ Main mode

☐ Aggressive mode ☐ Username:

Proposal 1:

Encryption :

Authentication :

Group :

Key Lifetime : seconds

(Note: Following three additional proposals are also proposed in Main mode:

DES/MD5/768, 3DES/SHA/1024 and 3DES/MD5/1024.)

Phase 2:

Proposal :

Encryption : 3DES

Authentication : SHA

PFS : ON

Group :

Key Lifetime : seconds



内容

- **1. Friewall**
- **2. SSL/TLS**
- **3. IPSec/VPN**
- **4. IDS**



防火墙的弱点

- ❑ 防火墙只能抵挡外部来的入侵行为
- ❑ 防火墙本身可能也存在弱点，以及其他安全性的设定错误
- ❑ 透过防火墙的保护，合法的使用者仍会非法地使用系统，甚至提升自己的权限



IDS和扫描器的关系

- IDS和扫描器都是简化管理员的工作，发现网络中的问题

扫描器是完全主动式安全工具，能够了解网络现有的安全水平

IDS是相对被动式安全工具，能够了解网络中即时发生的攻击

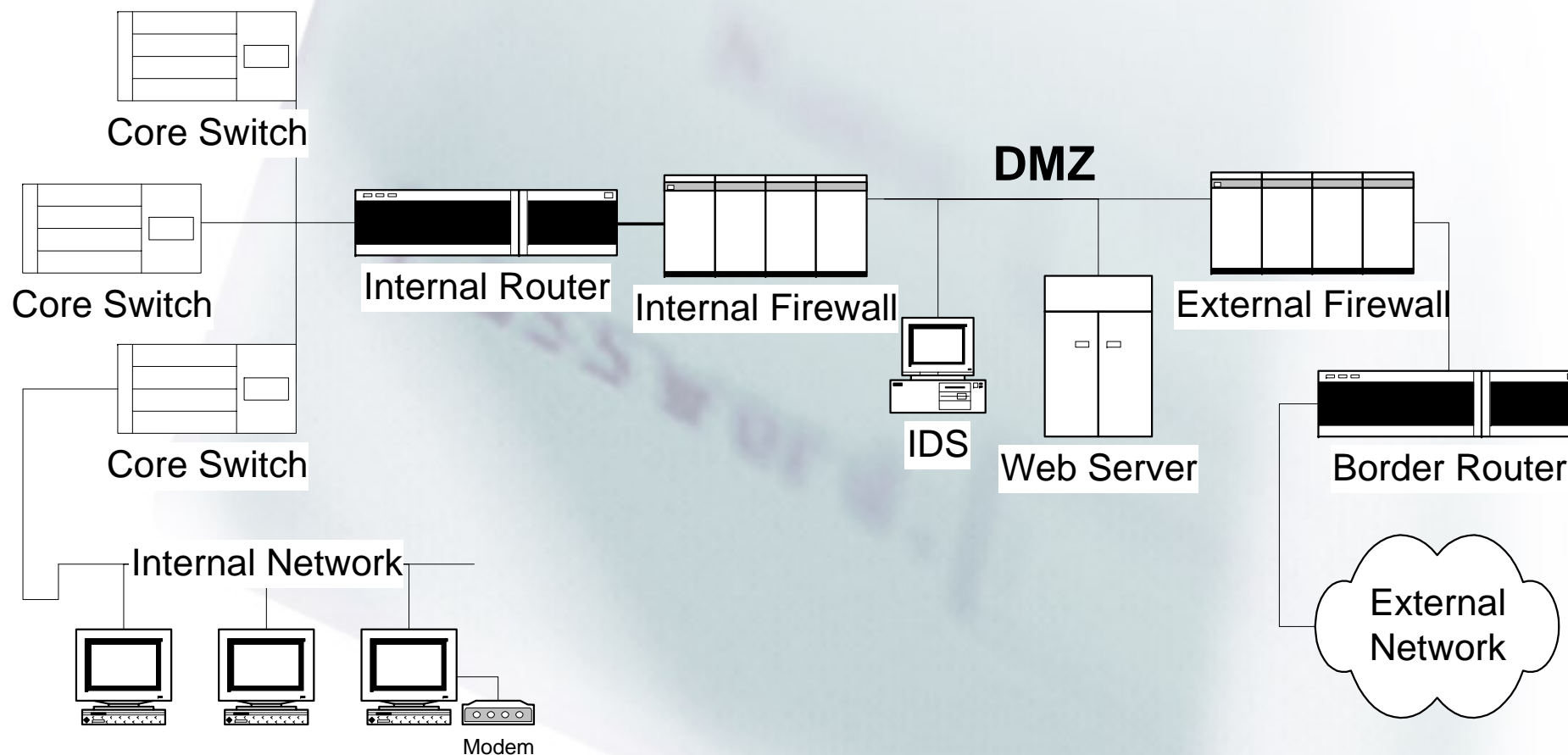


定义

入侵检测系统（Intrusion Detection System）工作在计算机网络系统中的关键节点上，通过实时地收集和分析计算机网络或系统中的信息，来检查是否出现违反安全策略的行为和遭到袭击的迹象，进而达到防止攻击、预防攻击的目的。

入侵检测系统作为主动保护自己免受攻击的网络安全技术，处于防火墙之后，成为防火墙之后的第二道安全闸门。

IDS与防火墙安全结构示意



入侵检测的起源

- 审计技术：产生、记录并检查按时间顺序排列的系统事件记录的过程。
- 从1984年到1986年，乔治敦大学的**Dorothy Denning**和**SRI/CSL**的**Peter Neumann**研究出了一个实时入侵检测系统模型，取名为**IDES**（入侵检测专家系统）。
- 1990年是入侵检测系统发展史上的一个分水岭。这一年，加州大学戴维斯分校的**L. T. Heberlein**等人开发出了**NSM**（**Network Security Monitor**）。该系统第一次直接将网络流作为审计数据来源。
- 从此，入侵检测系统发展史翻开了新的一页，两大阵营正式形成：基于网络的**IDS**和基于主机的**IDS**。

入侵检测的分类

- 按照分析方法（检测方法）
 - 异常检测模型（Anomaly Detection）: 首先总结正常操作应该具有的特征（用户轮廓），当用户活动与正常行为有重大偏离时即被认为是入侵。
 - 误用检测模型（Misuse Detection）: 收集非正常操作的行为特征，建立相关的特征库，当监测的用户或系统行为与库中的记录相匹配时，系统就认为这种行为是入侵。
- 如果系统错误地将正常活动定义为入侵，称为**误报(false positive)**；
- 如果系统未能检测出真正的入侵行为，称为**漏报(false negative)**。

异常检测模型

- 方法：构造一个当时活动的主机或网络的大致轮廓，当有一个轮廓外的事件发生时，IDS就会告警，例如有人做了以前他没有做过的事情的时候，例如，一个用户突然获取了管理员或根目录的权限
- 特点：不需要对每种入侵行为进行定义，能有效检测未知的入侵。同时系统能针对用户行为的改变进行自我调整和优化，但随着检测模型的逐步精确，异常检测会消耗更多的系统资源。



误用检测模型

- 特点：采用特征匹配，误用检测能明显降低错报率，但漏报率随之增加。攻击特征的细微变化，会使得误用检测无能为力。

入侵检测的分类

- 按照数据来源：
 - 基于主机：系统获取数据的依据是系统运行所在的主机，保护的目标也是系统运行所在的主机。 **HDIS**
 - 基于网络：系统获取的数据是网络传输的数据包，保护的是网络的运行。 **NIDS**
 - 混合型



两类IDS监测软件

- **HIDS**

- 视野集中
- 易于用户自定义
- 保护更加周密
- 对网络流量不敏感

- **NIDS**

- 侦测速度快
- 隐蔽性好
- 视野更宽
- 较少的监测器
- 占资源少



信息收集的来源

- 来自主机的
 - 基于主机的监测收集通常在操作系统层的来自计算机内部的数据，包括操作系统审计跟踪信息和系统日志
- 来自网络的
 - 检测收集网络的数据
- 来自应用程序的
 - 监测收集来自运行着的应用程序的数据，包括应用程序事件日志和其它存储在应用程序内部的数据

信息收集

- 日志文件中记录了各种行为类型，每种类型又包含不同的信息，例如记录“用户活动”类型的日志，就包含登录、用户**ID**改变、用户对文件的访问、授权和认证信息等内容。
- 重复登录失败、登录到不期望的位置以及非授权的企图访问重要文件等等。
- 入侵者经常替换、修改和破坏他们获得访问权的系统上的文件，同时为了隐藏系统中他们的表现及活动痕迹，都会尽力去替换系统程序或修改系统日志文件。
- 一个进程的执行行为由它运行时执行的操作来表现，操作执行的方式不同，它利用的系统资源也就不同。操作包括计算、文件传输、设备和其它进程，以及和网络间其它进程的通讯。



统计分析

- 统计分析方法首先给系统对象（如用户、文件、目录和设备等）创建一个统计描述，统计正常使用时的一些测量属性（如访问次数、操作失败次数和延时等）。
- 测量属性的平均值将被用来与网络、系统的行为进行比较，任何观察值在正常值范围之外时，就认为有入侵发生

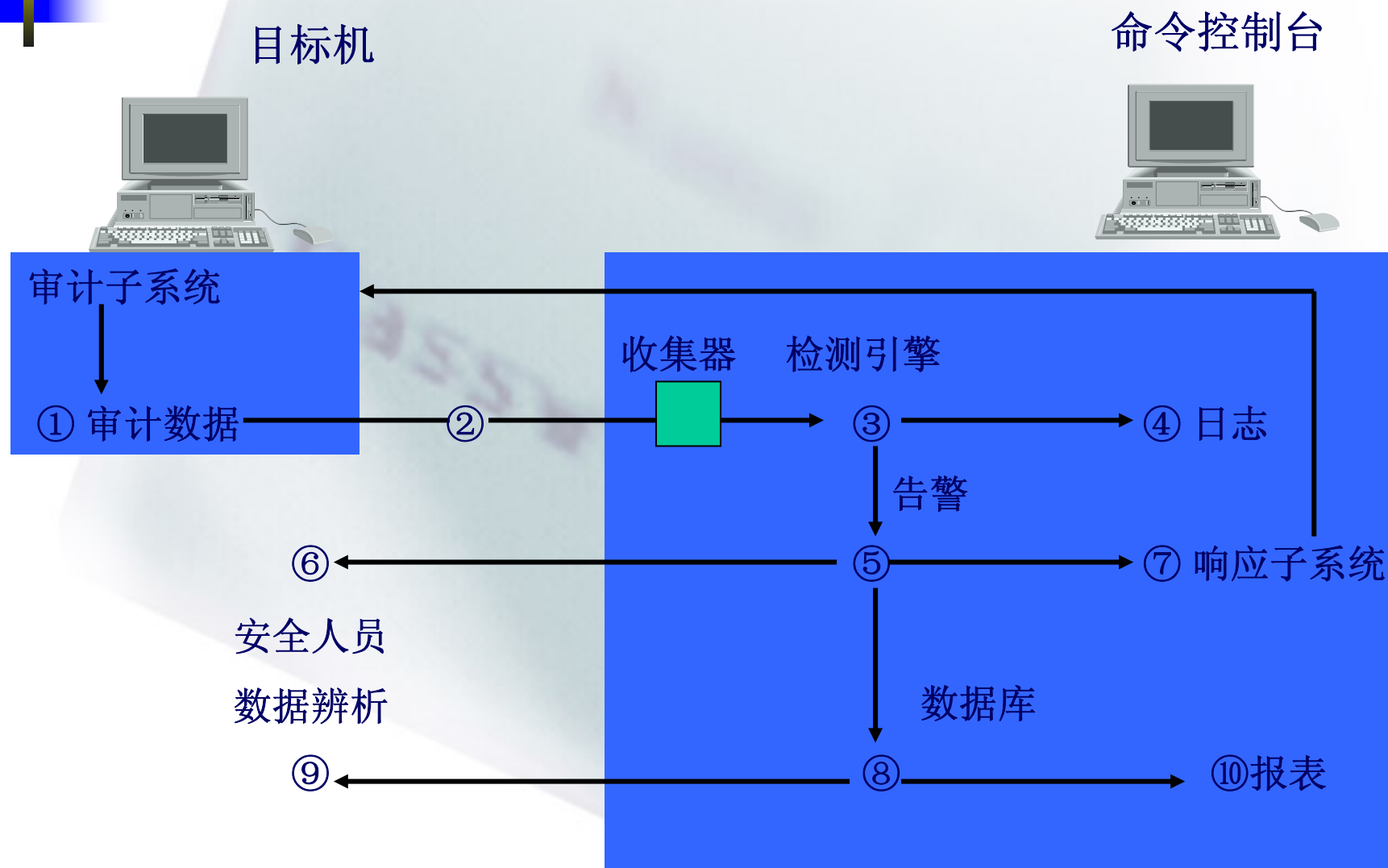
模式匹配

- 模式匹配就是将收集到的信息与已知的网络入侵和系统误用模式数据库进行比较，从而发现违背安全策略的行为。
- 一般来讲，一种进攻模式可以用一个过程（如执行一条指令）或一个输出（如获得权限）来表示。该过程可以很简单（如通过字符串匹配以寻找一个简单的条目或指令），也可以很复杂（如利用正规的数学表达式来表示安全状态的变化）。

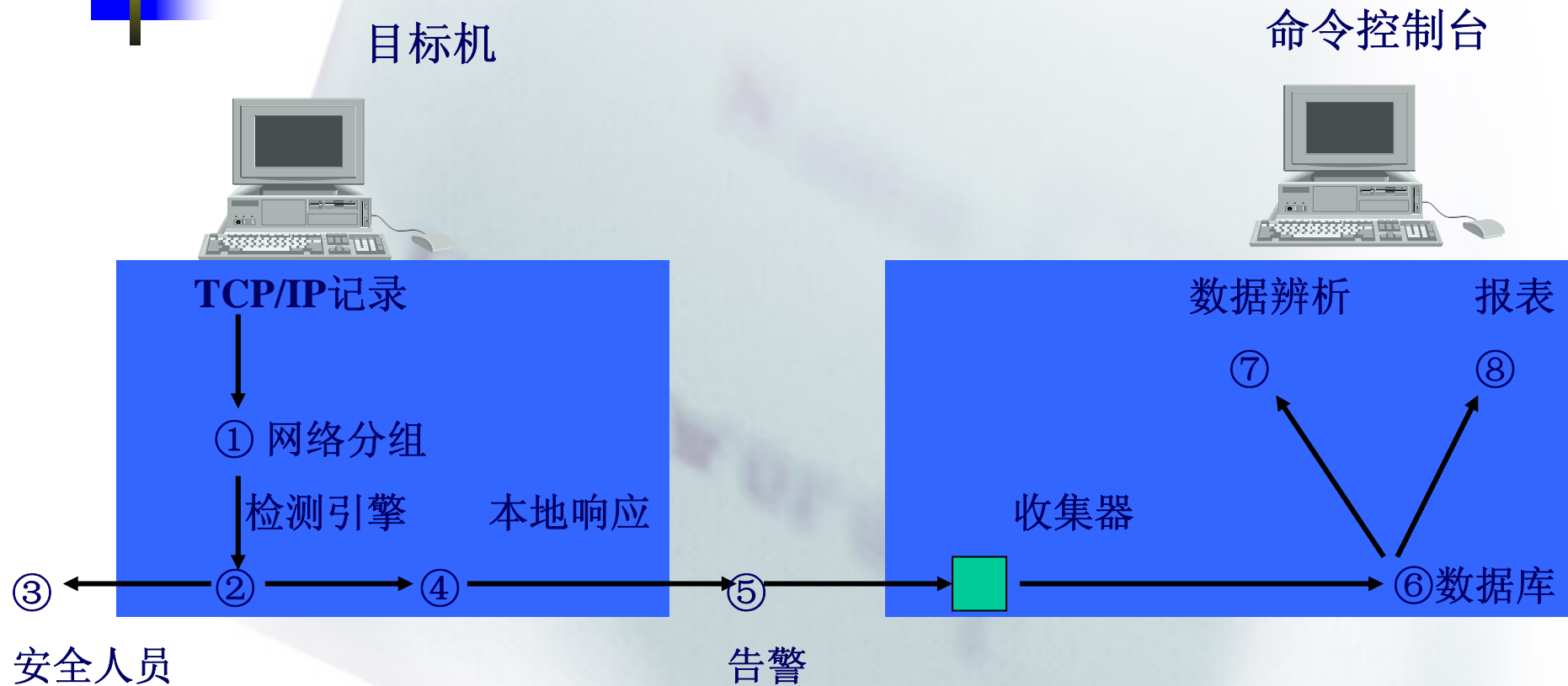
完整性分析

- 完整性分析主要关注某个文件或对象是否被更改，这经常包括文件和目录的内容及属性，它在发现被更改的、被安装木马的应用程序方面特别有效。

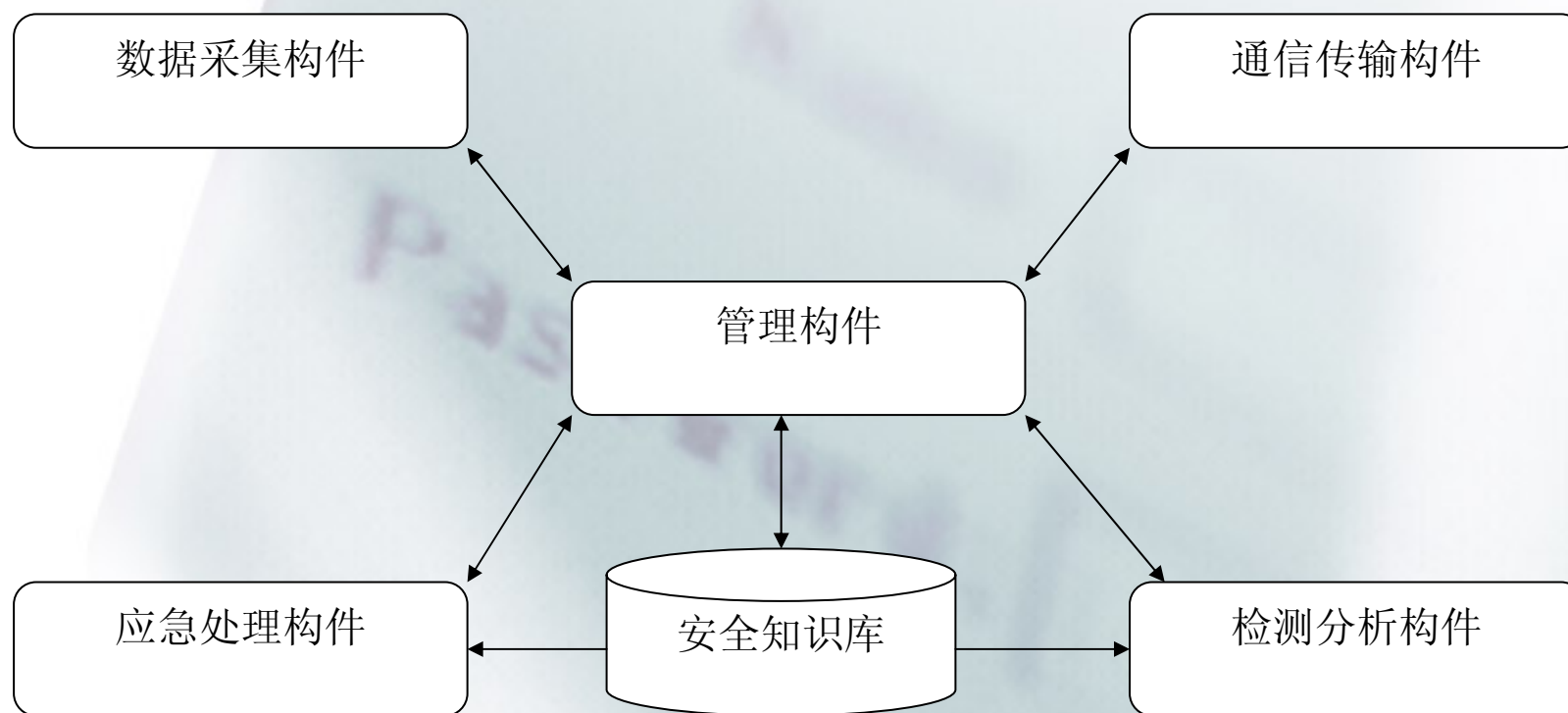
集中式基于主机的入侵检测结构



分布式基于网络的入侵检测结构



入侵检测系统结构示意图





一个攻击的检测实例

- 老版本的**Sendmail**漏洞利用

\$ telnet mail.victim.com 25

WIZ

shell

或者

DEBUG

#

直接获得rootshell！

简单的匹配

- 检查每个packet是否包含：

“WIZ”

| “DEBUG”

检查端口号

- 缩小匹配范围

**Port 25:{
 “WIZ”
 | “DEBUG”
}**

深入决策树

- 只判断客户端发送部分

```
Port 25:{  
    Client-sends: “WIZ” |  
    Client-sends: “DEBUG”  
}
```


功能-响应策略

- 弹出窗口报警
- **E-mail**通知, **SMS**
- 切断**TCP**连接
- 执行自定义程序
- 与其他安全产品交互
 - Firewall
 - SNMP



功能-检测非法外联、地址欺骗

★ 非法外联检测

检测网络中的非法拨号

和入侵检测无缝集成，无需客户端代理

★ 防IP地址欺骗

受护网络中主机的IP—MAC的纪录跟踪



功能-流量统计

★ 网络流量统计

实时刷新的图形化界面:

比特统计

报文统计

关键端口流量统计

TCP连接统计

报警事件统计



功能-内容检测

★ 网络敏感内容检测

监测内部的网络滥用行为:

URL地址

FTP、TELNET的用户名、密码、命令

邮件主题



功能-会话回放

★ 网络事件回放

重现网络行为：

HTTP

FTP

SMTP

POP3

TELNET



功能-远程管理

★ 远程管理

安全性:

SSL加密信道

分级部署:

CONSOLE—SENSOR 一对多控制

CONSOLE支持级连



功能-远程升级

★ 远程升级

规则库升级

探测器升级

“在线”、“手动”两种方式

测试

- 攻击检测
 - 误报 (false positives)
 - 漏报 (false negatives)
- 性能测试
- **IDS**躲避测试
- 状态性测试



攻击检测

- 端口扫描和信息收集
 - nmap、finger、rpcinfo、DNS query
- 漏洞扫描
 - nessus
- 常见攻击手法
 - buffer overflow (IIS、Sun RPC、Linux)
- 后门木马
 - NetBus、BO2K
- 拒绝服务
 - SYN Flood、UDP Bomb、IPFrag、DDoS

性能测试

- 实际生产环境
- 模拟流量
 - 硬件: SmartBits
 - 人为构造一定大小的数据报, 从64bytes到1500bytes, 衡量不同pps(packets per second)下IDS对攻击的检测情况。
 - 软件: tcpdump & tcpreplay
 - 对流量的回放



IDS躲避测试

- **URL编码**
 - “cgi-bin” → “%63%67%69%2d%62%69%6e”
- **Inserion**, 插入其他字符
 - “GET /cgi-bin/phf” → “GET //cgi-bin//phf”
以绕过攻击特征库。
- 将攻击包以碎片方式发出
 - fragrouter

状态性测试

- **Stateful ?**
- **测试工具: Stick/Snot**
 - 不建立连接, 直接发送攻击数据包
 - 只分析单个数据包的IDS会大量误报
- **要减少误报, IDS必须维护连接状态**



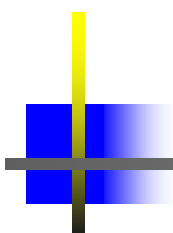
面临的问题

1. 随着能力的提高，入侵者会研制更多的攻击工具，以及使用更为复杂精致的攻击手段，对更大范围的目标类型实施攻击；
2. 入侵者采用加密手段传输攻击信息；
3. 日益增长的网络流量导致检测分析难度加大；
4. 不适当的自动响应机制存在着巨大的安全风险；
5. 存在对入侵检测系统自身的攻击；
6. 过高的错报率和误报率，导致很难确定真正的入侵行为；
7. 高速网络环境导致很难对所有数据进行高效实时分析



发展方向

1. 更有效的集成各种入侵检测数据源，包括从不同的系统和不同的传感器上采集的数据，提高报警准确率；
2. 在事件诊断中结合人工分析；
3. 提高对恶意代码的检测能力，包括 email 攻击，Java，ActiveX等；
4. 采用一定的方法和策略来增强异种系统的互操作性和数据一致性；
5. 研制可靠的测试和评估标准；
6. 提供科学的漏洞分类方法
7. 提供对更高级的攻击行为如分布式攻击、拒绝服务攻击等的检测手段；

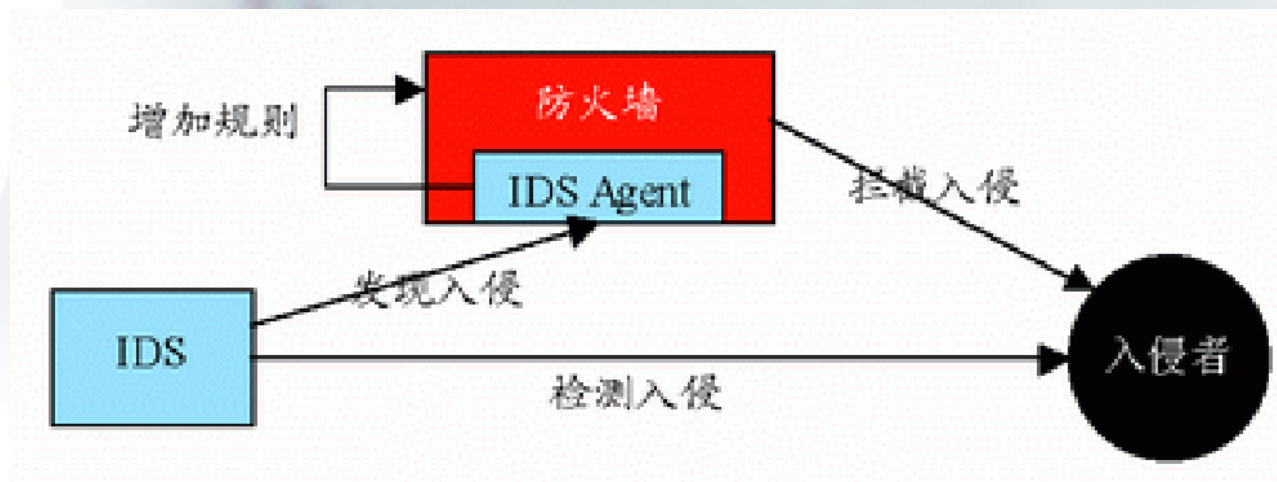


蜜罐

- 把攻击者引导到一个经过特殊装备的系统上，这种系统被成为蜜罐。
- 蜜罐是一种欺骗手段，它可以用于错误地诱导攻击者，也可以用于收集攻击信息，以改进防御能力。
- 蜜罐能采集的信息量由自身能提供的手段以及攻击行为数量决定。

IDS与Firewall联动

通过在防火墙中驻留的一个IDS Agent对象，以接收来自IDS的控制消息，然后再增加防火墙的过滤规则，最终实现联动



Cisco

CIDF(CISL)

ISS

Checkpoint



协议分析 + 命令解析

- 协议分析加命令解析技术是一种新的入侵检测技术，它结合高速数据包捕捉、协议分析和命令解析来进行入侵检测，给入侵检测战场带来了许多决定性的优势
- 由于有了协议分析加命令解析的高效技术，基于运行在单个Intel架构计算机上的入侵检测系统的千兆网络警戒系统，就能分析一个高负载的千兆以太网上同时存在的超过300万个连接，而不错漏一个包



协议分析

协议分析充分利用了网络协议的高度有序性，使用这些知识快速检测某个攻击特征的存在

- Ether、IP、ARP
- TCP、UDP、ICMP
- HTTP、Telnet、DNS、FTP、IRC、NetBIOS、SMB、SMTP、SNMP、TFTP、RPC、POP3、Finger、rlogin、MIME、IMAP4、VNC、RealAudio、NetGames、MS SQL



协议分析的优势

- 效率高
- 检测漏洞脚本
 - 例如大量的90字符可能是ShellCode中的NOOP操作。
- 依据**RFC**，执行协议异常分析



命令解析

- 解析器是一个命令解释程序，入侵检测引擎包括了多种不同的命令语法解析器，因此，它能对不同的高层协议——如Telnet、FTP、HTTP、SMTP、SNMP、DNS等的用户命令进行详细的分析。
- 命令解析器具有读取攻击串及其所有可能的变形，并发掘其本质含义的能力。这样，在攻击特征库中只需要一个特征，就能检测这一攻击所有可能的变形。
- 解析器在发掘出命令的真实含义后将给恶意命令做好标记，主机将会在这些包到达操作系统、应用程序之前丢弃它们。



典型例子

第一步——直接跳到第13个字节，并读取2个字节的协议标识。如果值是0800，则说明这个以太网帧的数据域携带的是IP包，基于协议解码的入侵检测利用这一信息指示第二步的检测工作。

第二步——跳到第24个字节处读取1字节的第四层协议标识。如果读取到的值是06，则说明这个IP帧的数据域携带的是TCP包，入侵检测利用这一信息指示第三步的检测工作。

第三步——跳到第35个字节处读取一对端口号。如果有一个端口号是0080，则说明这个TCP帧的数据域携带的是HTTP包，基于协议解码的入侵检测利用这一信息指示第四步的检测工作。

第四步——让解析器从第55个字节开始读取URL。

URL串将被提交给HTTP解析器，在它被允许提交给Web服务器前，由HTTP解析器来分析它是否可能会做攻击行为。



优点

- **提高了性能：**协议分析利用已知结构的通信协议，与模式匹配系统中传统的穷举分析方法相比，在处理数据帧和连接时更迅速、有效。
- **提高了准确性：**与非智能化的模式匹配相比，协议分析减少了虚警和误判的可能性，命令解析（语法分析）和协议解码技术的结合，在命令字符串到达操作系统或应用程序之前，模拟它的执行，以确定它是否具有恶意。
- **基于状态的分析：**当协议分析入侵检测系统引擎评估某个包时，它考虑了在这之前相关的数据包内容，以及接下来可能出现的数据包。与此相反，模式匹配入侵检测系统孤立地考察每个数据包。
- **反规避能力：**因为协议分析入侵检测系统具有判别通信行为真实意图的能力，它较少地受到黑客所用的像URL编码、干扰信息、TCP/IP分片等入侵检测系统规避技术的影响。

产品

- 免费
 - Snort
 - <http://www.snort.org>
 - SHADOW
 - <http://www.nswc.navy.mil/ISSEC/CID/>

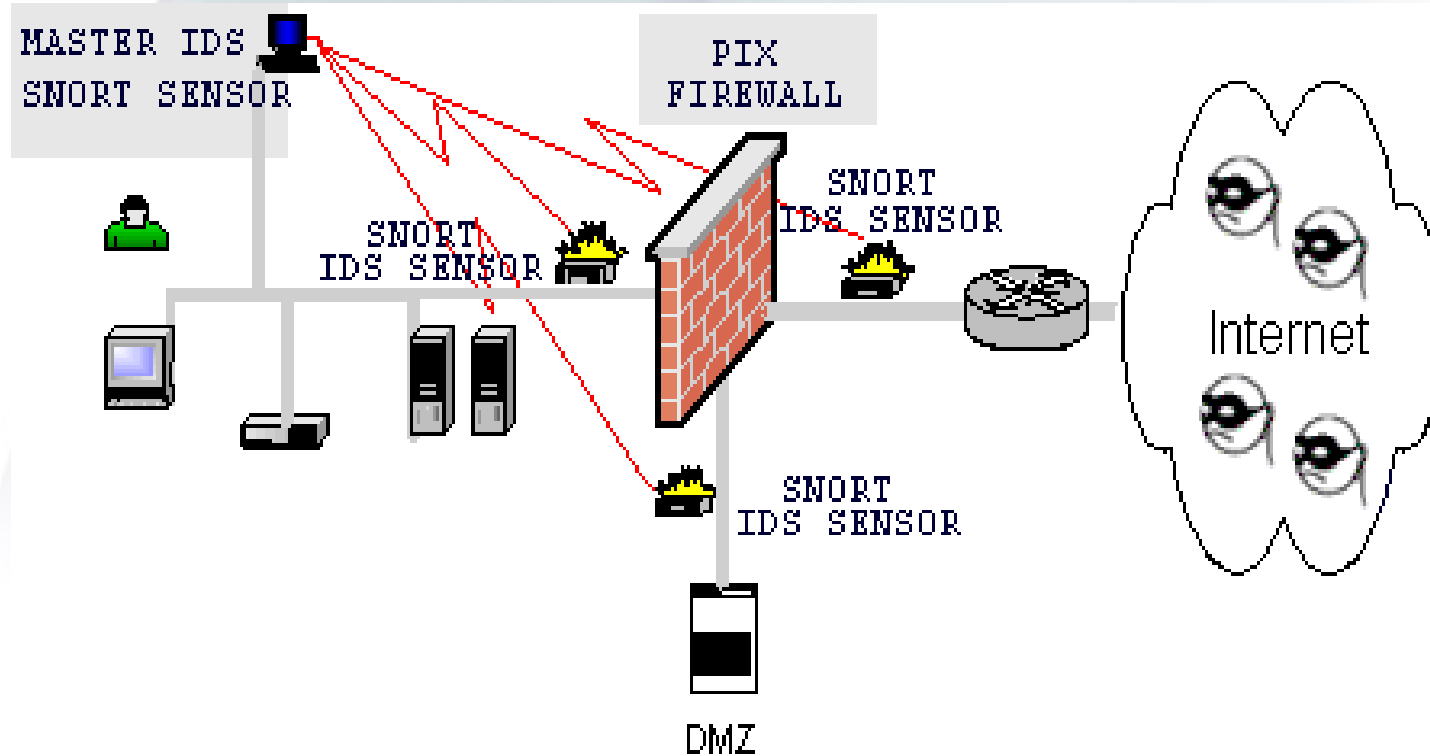
IDS- snort

- **snort概述**

- Snort是一种开放资源网络侵扰探测系统（IDS）,这一系统得到了来自忠实的志愿程序设计人员的技术支持。最初，Snort是为了Linux/UNIX操作系统而编写的，但最终它也应用到了Windows操作系统。

IDS- snort

- **Snort on Windows (windows NT4、2000、 XP)**



Snort 安装&使用

- **Snort安装**
 - 运行下载的snort-1_9_1.exe即可
- **Snort使用**
 - 在命令行下输入 `c:\snort -v`，就可在屏幕上显示本机数据包IP和TCP、UDP、 ICMP头信息 如下图



```
选定 C:\WINNT\System32\cmd.exe
***S*** Seq: 0x502606E  Ack: 0x30BC435A  Win: 0x8000  TcpLen: 24
TCP Options <1> => MSS: 1460
+++++
04/03-23:22:24.792992 192.168.102.111:2215 -> 162.105.202.100:8080
TCP TTL:128 TOS:0x0 ID:17004 IpLen:20 DgmLen:40
***A*** Seq: 0x30BC435A  Ack: 0x502606F  Win: 0xFAF0  TcpLen: 20
+++++
04/03-23:22:24.794068 192.168.102.111:2215 -> 162.105.202.100:8080
TCP TTL:128 TOS:0x0 ID:17005 IpLen:20 DgmLen:287
***AP*** Seq: 0x30BC435A  Ack: 0x502606F  Win: 0xFAF0  TcpLen: 20
+++++
04/03-23:22:24.796132 162.105.202.100:8080 -> 192.168.102.111:2215
TCP TTL:61 TOS:0x0 ID:60492 IpLen:20 DgmLen:893
***AP*** Seq: 0x5026623  Ack: 0x30BC4451  Win: 0x832C  TcpLen: 20
```

Snort 安装&使用

- 可以用如下命令将本子网内所有的包头记入日志文件
 - `c:\snort -v -h xxx.xxx.102.1/24 -l c:\snort\log`
 - 也可以应用安装目录子目录rules下的规则。
 - Snort具体参数可以输入以下命令来看
 - `C:\ Snort /?`
 - Snort的具体使用请参考安装目录子目录doc下的SnortUsersManual.pdf文档。



ICMP.rules

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP mytest  
Pinger"; content:"|12 34 56 78|"; itype:8; depth:32; reference:arachnids,158;  
classtype:attempted-recon; sid:465; rev:1;)
```

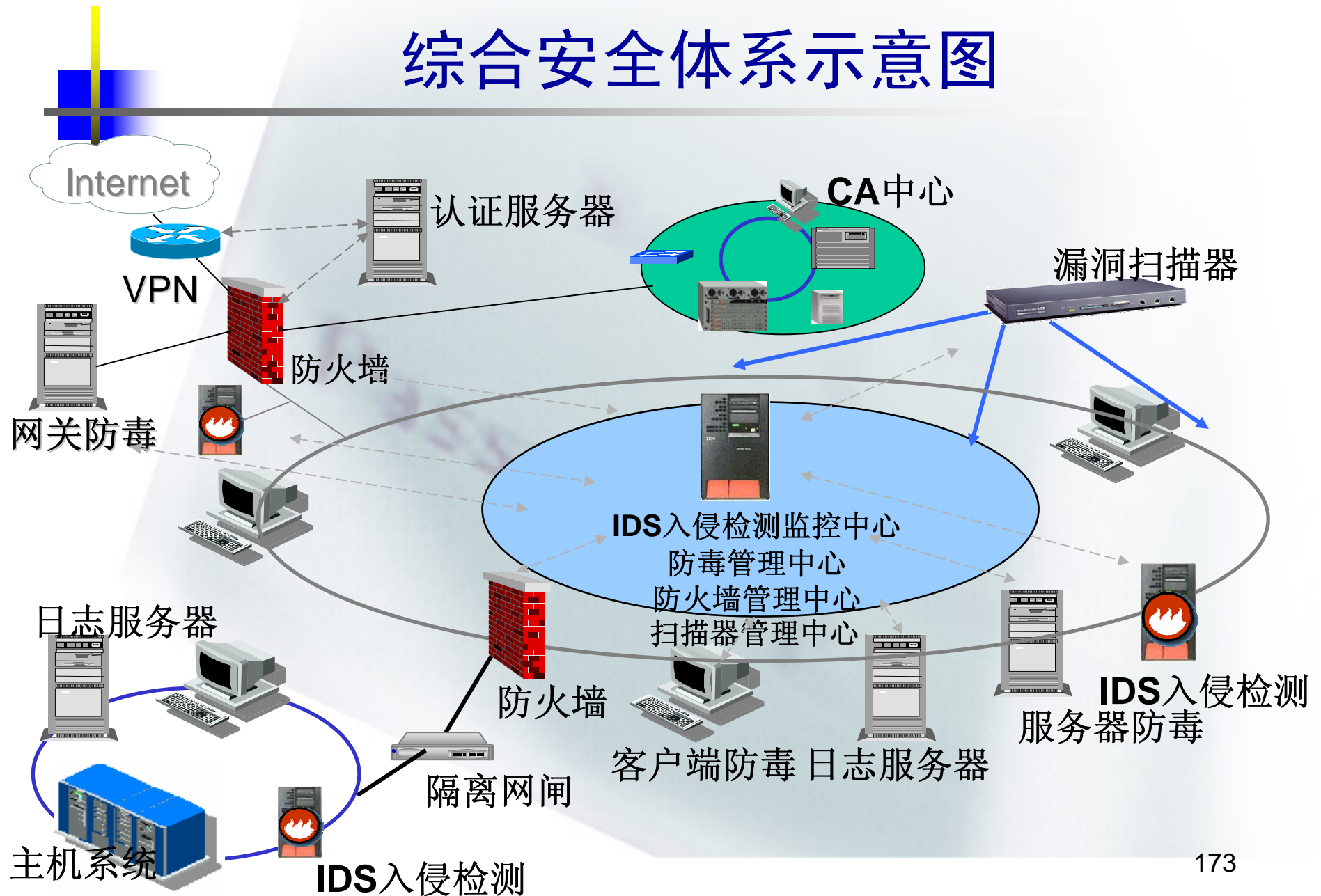
```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP ISS  
Pinger"; content:"|495353504e475251|"; itype:8; depth:32; reference:arachnids,158;  
classtype:attempted-recon; sid:465; rev:1;)
```

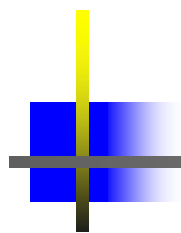
```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP L3retriever  
Ping"; content: "ABCDEFGHJKLMNOPQRSTUVWXYZABCDEFGHI"; itype: 8;  
icode: 0; depth: 32; reference:arachnids,311; classtype:attempted-recon; sid:466;  
rev:1;)
```

.....

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Large  
ICMP Packet"; dsize: >800; reference:arachnids,246; classtype:bad-unknown;  
sid:499; rev:3;)
```

综合安全体系示意图





Q&A

谢谢！