网络安全协议及分析
代理安全Socks

密码与网络安全系　刘虹

2025年春季学期

课程体系

第一章 概述

第二章 链路层扩展L2TP

第三章 IP层安全IPSec

第四章 传输层安全SSL和TLS

第五章 会话安全SSH

第六章 代理安全Socks

第七章 网管安全SNMPv3

第八章 认证协议Kerberos

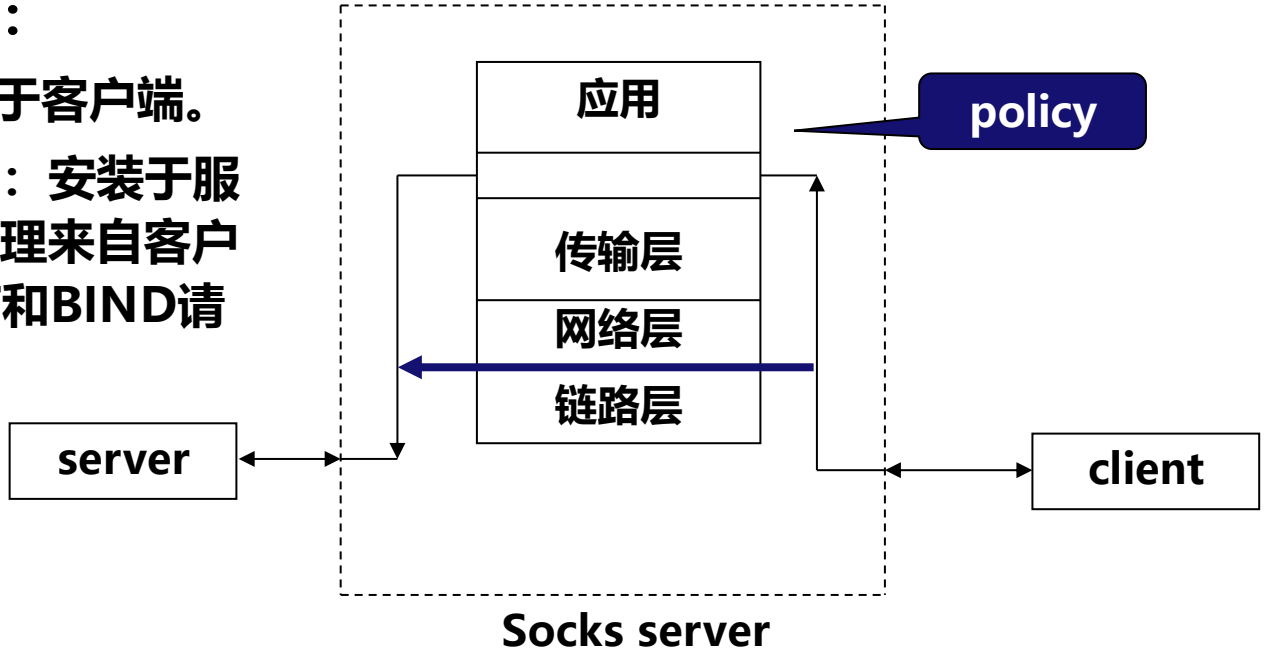
第九章 应用安全

本章学习目标

- 通用安全服务应用程序接口GSSAPI
- Socks应用

Socks框架

- Socks使用C/S模型
- Socks软件包组成：
  - Socks库：安装于客户端。
  - Sockd守护程序：安装于服务器，接收并处理来自客户端的CONNECT和BIND请求。



Socks框架

- **Socks客户端命令：**
  - **CONNECT**：通告代理服务器与远程主机建立连接，调用函数：Rconnect
  - **BIND**：通告服务器接收来自某个远程主机的连接请求，调用函数：Rbind，Rlisten和Raccept.

提纲

通用安全服务应用程序接口

- 通用安全服务应用程序接口（Generic Security Services Application Program Interface，GSSAPI）
  - 安全应用开发者经常使用的编程接口
  - GSSAPI本身不提供任何安全保护，它的目的是为有安全需求的应用开发者屏蔽不同底层安全机制的差异，由此提供一套统一的编程接口

通用安全服务应用程序接口

- GSSAPI涉及三个关键概念
  - **信任状**：客户端和服务器之间建立安全上下文的先决条件；
  - **安全上下文**：客户端和服务器通过协商所建立的安全通道，建立了安全上下文就意味着双方已经认证了对方的身份，并且生成了用户保护通信所需的安全参数；
  - **令牌**：包含安全参数
    - 安全上下文令牌：用于安全上下文的建立和维护
    - 单条消息令牌：用于保护每条应用消息

通用安全服务应用程序接口

- 使用基于X.509公钥证书的认证方法时，客户端应首先调用GSSAPI的GSS_Acquire-cred()函数以建立信任状，其目的是使得客户私钥能够作为客户端身份认证的依据。
- 客户端调用GSS_Init_sec_context()函数，后台完成以下工作：
  - 1）查询CA的证书目录以获取服务器公钥证书；
  - 2）验证证书签名的有效性及证书的时效性；
  - 3）生成共享的会话密钥，并用服务器公钥加密；
  - 4）整个消息用客户端的私钥签名。

通用安全服务应用程序接口

- 加密的会话密钥、签名以及包含客户端证书信息的目录就是使用公钥认证方法时的客户端输出令牌，它也是GSS_Init_sec_context()函数的输出。这个令牌被发送给服务器。
- 服务器收到这个令牌后，调用GSS_Accept_sec_context()函数，并把其作为该函数所需的输入令牌参数。后台则完成以下工作：
  - 1）查询目录以获取客户端证书；
  - 2）验证客户端签名；
  - 3）还原会话密钥。

通用安全服务应用程序接口

- 经过上述步骤后，客户端和服务器就建立了安全上下文。
- 在发送数据时，可以调用GSS_Sign()或GSS_Seal()函数对数据进行安全处理。
  - 前者仅提供完整性保护和数据源发认证功能
  - 后者还可以提供机密性保护。
- 对于收到的数据，则可以调用GSS_Verify()和GSS_Unseal()处理。

通用安全服务应用程序接口

- Socks5 GSSAPI
  - 当Socks5客户端和服务器协定使用GSSAPI认证方法后，即可启动GSSAPI这个认证过程的第一步是建立默认信任状；
  - 之后建立安全上下文并互相交换令牌；
  - 最后在安全上下文的保护下开展子协商以确定对每条消息的保护方式。
- 上述过程完成后，通信双方就可以利用协商好的安全参数保护通信数据。

通用安全服务应用程序接口

（1）建立安全上下文

- 客户端调用GSS_import_name以构造GSSAPI所需的服务器内部名
  - user@machine表示登录到另一台计算机的用户，
  - nfs@machine表示网络文件系统（Network File System，NFS)这个网络服务应用
  - host@example.com表示一台计算机

GSS_import_name将实体名转化为不同机制所需的“内部名”。

- 输入参数包括字符串形式的input_name_string，以及对象标识符（Object Identifier)形式的input_name_type
- 输出包括两个状态码major_status，minor_status，及内部名形式的output_name

通用安全服务应用程序接口

（1）建立安全上下文

- 客户端调用GSS_Init_sec_context以建立安全上下文
  - 机制类型：标明使用的低层安全机制，其类型为OID。
  - 通道绑定：增强身份认证功能，常用于标识建立安全上下文的双方，即安全通道的起点和终点。
- GSSAPI定义的通道绑定数据结构中包含发起方地址类型、发起方地址、回应方地址类型、回应方地址、应用数据

通用安全服务应用程序接口

GSS_Init_sec_context函数输入项

claimant_cred_handle	信任状句柄
Input_context_handle	上下文句柄
targ_name	目标名称
mech_type	机制类型
life_time	生存期需求
deleg_req_nag	TRUE表示请求访问权限的授权
mutual_req_flag	TURE表示需要双向认证
reply_det_req_flag	TURE表示需要抗重播服务
sequence_req_nag	TURE表示要保证数据顺序
chan_bindings	可以为空
input_token	空

通用安全服务应用程序接口

GSS_Init_sec_context函数输出项

major_status	主状态码，描述函数执行结果
minor_status	次状态，与主状态配合描述函数执行结果
output_context_handle	上下文句柄
mech_type	最终可以使用的机制类型
output_token	发送给服务器的输出令牌
deleg_state	是否同意授权
mutual_state	是否可以执行双向认证
replay_det_state	是否可以提供抗重播服务
sequence_state	是否可以防止数据乱序
conf_avail	是否能够为每条消息提供机密性保护
integ_avail	是否能够为每条消息提供完整性保护
lifetime_rec	最终的生存期，INDEFINITE表示无限长

通用安全服务应用程序接口

GSS_Init_sec_context函数“major_status”返回取值

GSS_COMPLETE	建立安全上下文的操作已经完成
GSS_CONTINUE_NEEDED	期望接收对方的输出令牌
GSS_DEFECTIVE_TOKEN	输入令牌验证不通过
GSS_DEFECTIVE_CREDENTIAL	信任状验证不通过
GSS_BAD_SIG	输入令牌中包含的签名不正确
GSS_NO_CRED	由于输入的信任状句柄不正确或调用者无使用信任状的权限而无法建立上下文
GSS_CREDENTIALS_EXPIRED	利用输入参数所传递的信任状已过期
GSS_BAD_BINDINGS	提供的通道绑定与本地提取的通道绑定信息不一致
GSS_NO_CONTEXT	与输入的上下文句柄对应的上下文中无可识别者
GSS_BAD_NAMETYPE	输入的目标名称类型无法识别
GSS_BAD_NAME	输入的目标名称不正确
GSS_FAILURE	发生了GSSAPI未定义的错误

通用安全服务应用程序接口

（1）建立安全上下文

- 服务器将令牌作为输入参数传递给GSS_Accept_sec_context函数

major_status	主状态码，描述函数执行结果
manor_status	次状态码，与主状态码配合描述函数执行结果
src_name	与所用机制相关的客户端名称
mech_type	最终使用的机制类型
output_context_handle	上下文句柄
deleg_state	是否同意授权
mutual_state	是否可以执行双向认证
reply_det_name	是否可以提供抗重播服务
sequence_state	是否可以防止数据乱序
integ_avail	是否能够为每条消息提供机密性保护
conf_avail	是否能够为每条消息提供完整性保护
lifetime_rec	最终的生存期，INDEFINITE表示无限长
delegated_cred_handle	当"deleg_state"为TRUE时，提供了代理信任状的句柄
output_token	返回给客户端的输出令牌

通用安全服务应用程序接口

（1）建立安全上下文

- GSS_Accept_sec_context函数“major_status”返回取值

GSS_COMPLETE	建立安全上下文的操作已经完成
GSS_CONTINUE_NEEDED	期望接收对方的输出令牌
GSS_DEFECTIVE_TOKEN	输入令牌验证不通过
GSS_DEFECTIVE_CREDENTIAL	信任状验证不通过
GSS_BAD_SIG	输入令牌中包含的签名不正确
GSS_DUPLICATE_TOKEN	令牌中包含的签名正确，但该令牌与之前收到的令牌相同，无法建立上下文
GSS_OLD_TOKEN	令牌中包含的签名正确，但该令牌存在时间过长，无法建立上下文
GSS_NO_CRED	由于输入的信任状句柄不正确或调用者无使用信任状的权限，无法建立上下文
GSS_CREDENTIAIS_EXPIRED	利用输入参数所传递的信任状已过期
CSS_BAD_BINDINGS	提供的通道绑定与本地提取的通道绑定信息不一致
GSS_NO_CONTEXT	与输入的上下文句柄对应的上下文中无可识别者
GSS_FAILURE	发生了GSSAPI未定义的错误

通用安全服务应用程序接口

（2）协商单条消息保护方式

- Sock5能够同时为基于TCP和UDP的应用提供安全保护，考虑到不同应用可能有不同的安全需求，专门规定了“子协商"以进一步协商对单条消息的保护方式。
  - 必需的完整性
  - 必需的完整性和机密性
  - 根据客户端和服务器的本地配置，完整性或机密性可选

通用安全服务应用程序接口

（2）单条消息保护

- 对单条消息的封装和验证通过GSS_Seal/GSS_Wrap和GSS_Unseal/GSS_Unwrap函数实现。
- 所有被Socks5 GSSAPI封装的消息都具备1B的版本字段，1B的类型字段，2B的长度字段以及变长的数据部分。
  - 采用第一种保护方法，数据部分包括明文的数据及消息验证码；
  - 采用第二种保护方法，包括加密的数据及消息验证码；
  - 采用第三种保护方式，数据部分的内容由客户端和服务器的本地配置决定。

提纲

一、GSSAPI

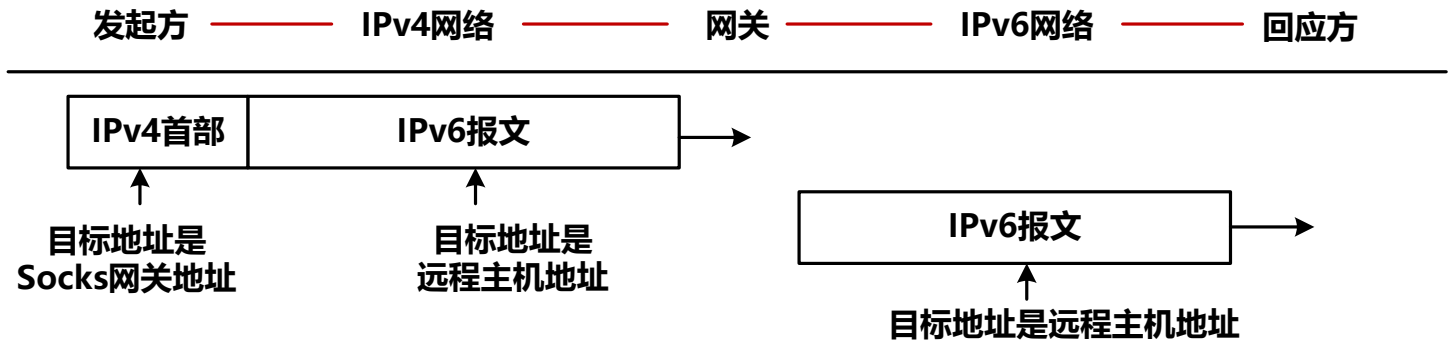
二、Socks应用

三、Socks代理案例

二 Socks应用

基于Socks的IPv4/IPv6网关

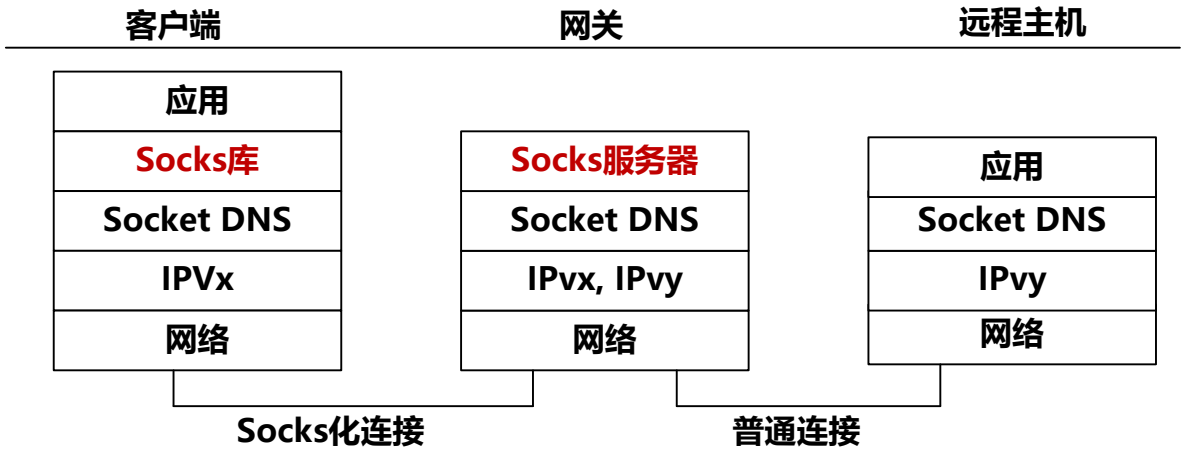
- 如果通信双方都具备IPv4和IPv6协议栈的网络，假设发起方连接IPv4网络，回应方连接IPv6，中间是Socks网关：
- 发起方构造IPv6报文，并在报文前添加IPv4首部，该报文首部的目标地址设置为Socks网关地址。
- Socks网关剥掉IPv4首部，把IPv6报文发送给回应方。



二 Socks应用

基于Socks的IPv4/IPv6网关

- 如果通信双方只有一个协议栈，且分别是IPv4和IPv6，可使用基于Socks 5的IPv4/IPv6网关实现互通。因为Sock5支持基于域名的寻址方式，可以屏蔽两类地址的差异。



二 Socks应用

- 假设客户端使用IPv4协议栈，远程主机使用IPv6协议栈：
  1. 通信前，网关知道通信双方的域名，并调用域名解析函数获取双方的IP地址。
  2. Socks库收到域名解析请求后，向应用返回某个IP地址，形式通常为0.0.0.*，并维护域名与该IP的映射关系。
  3. 应用调用Socks库函数建立与远程主机的连接，并把目标地址设置为该IP地址。
  4. Socks库收到请求后，根据本地维护的域名/IP地址映射关系，找到对应的域名。
  5. Socks客户端与Socks服务器建立连接，并发送CONNECT命令，其中包含目标域名。
  6. 服务器收到请求后，验证客户端身份，然后查询域名找到对应的IP地址。
  7. 服务器与查询到的IP建立连接，并把连接状态通告给客户端。
  8. 随后的通信中，网关将利用Socks服务器转发客户端和远程主机之间的通信量。

提纲

一、GSSAPI

二、Socks应用

三、Socks代理案例

三 Socks代理案例

伪装来源

- 当“客户”组件与“服务器”组件建立连接的时候，服务器是可以取回客户的连接地址的，有时候为了某种原因大家可能会不想让服务器知道客户的地址，那么可以通过Socks5代理连接“服务器”组件，这样在“服务器”组建那里得到的客户地址就是Socks5代理服务器的地址，从而达到隐藏真实地址的目的。

局域网通信

- 在网络飞速发展的今天，局域网内多机公用一个IP地址的实际组网结构已经是司空见惯了，但是这给网络通信带来了巨大的困难，局域网内部与外部的连接、不同局域网之间的连接，已经成了很扰人的一件事，而通过Socks5代理就是一种理想的解决方法。

三 Socks代理案例

- Socks5是一个代理协议，它在使用 TCP/IP协议通讯的前端机器和服务器机器之间扮演一个中介角色，使得内部网中的前端机器变得能够访问Internet网中的服务器，或者使通讯更加安全。
  - 对于各种基于 TCP/IP的应用层协议都能够适应，几乎是万能的。
  - 它虽然不能理解自己转发的数据的内部结构，但是它能够忠实地转发通讯包，完成协议本来要完成的功能。

小结

- Socks是目前较为常用的一种代理技术，使用C/S模型。
  - 客户端的Socks库代替了Socket库
  - 服务器的Sockd实现代理转发功能
- Socks4仅能转发TCP连接，包括CONNECT和BIND两个命令
  - CONNECT用于客户端发起到远程主机的连接；
  - BIND是在已经利用CONNECT建立主连接的基础上，客户端接收来自远程主机的连接请求。
- Socks5对Socks4进行了三种扩展，包括：
  - 用户身份认证
  - 基于IPv6及域名的寻址机制
  - 转发UDP会话

小结

- Socks5支持用户名/口令及GSSAPI认证方法。
- GSSAPI是一套安全编程接口，它提供双向身份认证功能以及数据机密性和完整性保护。
  - Socks5 GSSAPI包括建立安全上下文、子协商以及数据保护等步骤
  - 子协商是对GSSAPI的扩充，用于进一步协商对单条消息的保护方式。
- 常用Socks代理软件包括Sockscap、Hummingbirdsocks等。
- 除代理功能外，Socks还被应用于解决IPv4/IPv6网络的互通问题。

办公地点：理科大楼B1209

联系方式：17621203829

邮箱：liuhongler@foxmail.com