常见协议和服务下

HTTP协议

HTTP（Hypertext Transfer Protocol，超文本传输协议）是用于万维网上通信的基础协议。它定义了客户端（如浏览器）与服务器之间如何请求和传输数据，主要用于传输超文本文件（如HTML），以及图像、视频等多种形式的资源。通常运行于TCP端口80。在HTTPS中，通信通过SSL或TLS进行加密，更加安全，通常使用端口443。

工作模式：HTTP协议采用请求/响应模式。客户端向服务器发送一个HTTP请求，服务器以一个HTTP响应作为回复。

无状态性：HTTP协议是无状态的，这意味着同一个客户的连续请求之间没有直接关系，服务器不会保存之前的请求状态。

请求格式

一个HTTP请求包括以下几个部分：

请求行：包括请求方法、URL和HTTP版本。

例如：`GET /index.html HTTP/1.1`

请求头：包含了关于客户端环境和请求本身的信息，例如：

- `Host: www.example.com`
- `User-Agent: Mozilla/5.0`
- `Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8`

空行：请求头和请求体之间必须有一个空行。

请求体（不是所有请求都有）：当使用POST或PUT等方法时，发送到服务器的数据包含在请求体中。

响应格式

一个HTTP响应包括以下几个部分：

状态行：包括HTTP版本、状态码和状态描述。

- 例如：`HTTP/1.1 200 OK`

响应头：例如内容类型、服务器信息、日期等。

- `Content-Type: text/html`
- `Server: Apache/2.4.1`

空行：响应头和响应体之间必须有一个空行。

响应体：包含请求的资源，如网页文本、图片等。

连接http

可以用浏览器或命令行工具发起HTTP网络请求。

```
┌──(root㉿kali)-[~]
└─# nc httpbin.org 80
GET /ip HTTP/1.0
Host: httpbin.org

HTTP/1.1 200 OK
Date: Thu, 14 Dec 2023 03:22:41 GMT
Content-Type: application/json
Content-Length: 31
Connection: close
Server: gunicorn/19.9.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true

{
  "origin": "112.48.18.94"
}

```

TLS协议

TLS (Transport Layer Security) 介绍

TLS（传输层安全协议，Transport Layer Security）是一种用于确保网络通信安全的加密协议。它是SSL（Secure Sockets Layer）的升级版，提供了更强大的安全性，并广泛应用于各种网络服务，如HTTPS、电子邮件传输、VPN等。TLS 的主要作用是保护数据的完整性、保密性和真实性。常见安全协议及端口如下：

协议	默认端口	安全协议	TLS默认端口
HTTP	80	HTTPS	443
FTP	21	FTPS	990
SMTP	25	SMTPS	465
POP3	110	POP3S	995
IMAP	143	IMAPS	993

TLS的关键功能

数据加密：TLS通过对数据进行加密，确保通信内容无法被第三方窃听。即使数据被拦截，也无法解读其内容。

数据完整性：TLS可以防止数据在传输过程中被篡改。通过消息完整性检查机制（如消息验证码HMAC），接收端能够验证数据在传输过程中没有被修改。

身份认证：TLS支持服务器和客户端的双向认证，通常使用数字证书来验证服务器的身份，确保客户端连接的是合法的服务器。

密钥交换：TLS使用安全的密钥交换机制（如 Diffie–Hellman、RSA），在通信双方安全地协商出一个对称加密密钥，用于保护后续的通信。

TLS的工作流程

TLS 的通信通常可以分为两个阶段：**握手阶段**和**数据传输阶段**。

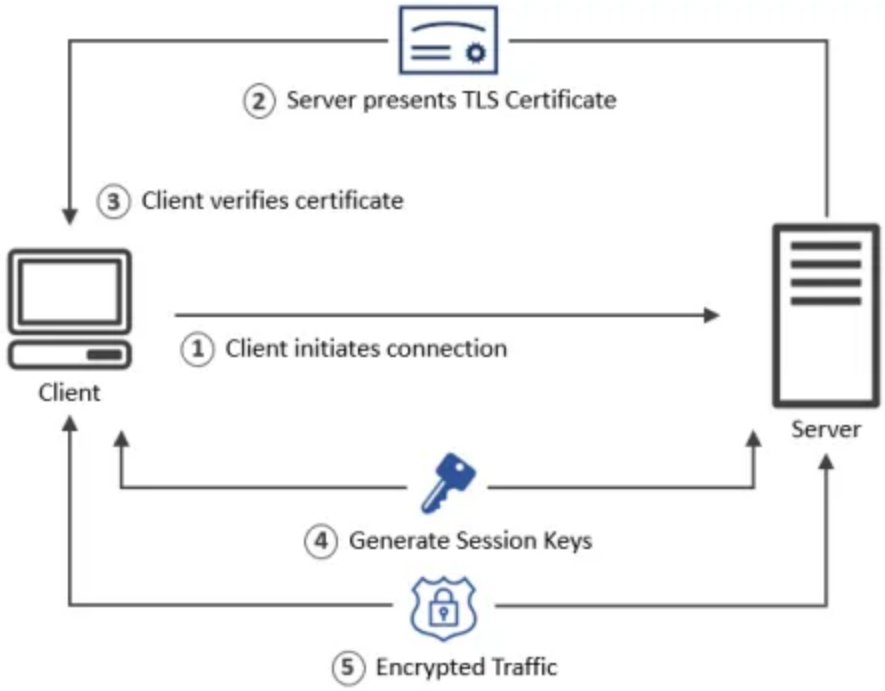
握手阶段：

握手过程是 TLS 连接的起点，用于协商加密算法和会话密钥。具体步骤如下：

- **客户端Hello**：客户端发起连接请求，并发送其支持的加密算法、TLS版本、随机数等信息。
- **服务器Hello**：服务器选择客户端支持的加密算法和TLS版本，并发送数字证书（用于身份验证）、随机数。
- **密钥交换**：客户端验证服务器的身份（通过数字证书），然后生成一个预主密钥，通过加密算法与服务器协商一个对称密钥。
- **握手完成**：双方确认使用对称密钥进行数据加密，握手结束，进入数据传输阶段。

数据传输阶段：

在握手完成后，双方开始使用对称密钥对数据进行加密传输。整个过程中的通信内容都被加密，确保数据的机密性与完整性。



TLS版本

TLS 已经历了多个版本的迭代，不同的版本在安全性和功能上有所提升：

TLS 1.0：早期的版本，作为SSL 3.0的升级版本，已经被弃用。

TLS 1.1：修复了一些TLS 1.0的漏洞，但仍然存在部分安全性问题。

TLS 1.2：这是目前广泛使用的版本，提供了更强的加密算法和更安全的密钥交换机制。

TLS 1.3：是最新的版本，大幅简化了握手过程，提高了安全性和性能，删除了不安全的加密算法。

TLS与SSL的区别

SSL（Secure Sockets Layer）是TLS的前身，最早由Netscape开发。SSL 2.0 和 SSL 3.0 已经不再安全，因此被废弃。**TLS** 是 SSL 的继任者，具有更强大的加密算法和更高的安全性，目前已经成为主流的加密协议。

TLS的应用场景

HTTPS：TLS 最常见的应用场景是在 HTTPS（HTTP over TLS）中，用于加密浏览器和服务器之间的通信，保护用户的隐私信息。

电子邮件传输：通过 SMTPS、POP3S、IMAPS 等协议，TLS 用于加密电子邮件的发送和接收，防止邮件被第三方窃听。

VPN：TLS 可用于虚拟专用网络（VPN）中，通过加密隧道传输数据，确保远程通信的安全性。

即时消息：许多即时通讯应用程序也依赖TLS来确保消息在传输过程中保持加密状态。

SSH协议

SSH（Secure Shell，安全外壳协议）是为计算机之间的安全通信设计的加密网络协议，通常用于远程登录、远程执行命令和数据传输。它最常见的用途是通过不安全的网络（如互联网）为用户提供安全的远程登录和其他服务。

SSH的特点

安全性：SSH 使用**对称加密**、**非对称加密**和**哈希函数**来确保数据在传输过程中不会被窃听或篡改，能够有效防止中间人攻击。

远程登录：SSH 是一种常用的远程登录协议，能够安全地访问远程服务器，替代了早期不安全的远程访问协议（如Telnet）。

文件传输：通过 SSH 可以实现加密的文件传输，常用的文件传输工具如 **SCP**（Secure Copy Protocol）和 **SFTP**（SSH File Transfer Protocol）都是基于 SSH 实现的。

端口转发：SSH 支持**端口转发**（Port Forwarding），可以通过加密隧道转发网络服务，保护敏感的网络通信。

身份验证：SSH 提供多种身份验证方式，包括**基于用户名和密码**的验证，以及更安全的**基于密钥对**的验证。后者使用**公钥**和**私钥**来保证只有拥有私钥的用户才能访问服务器。

SSH的工作原理

SSH 协议通过**客户端–服务器模型**工作。通常由 SSH 客户端（如 `ssh` 命令）发起连接请求，服务器端（如 `sshd` 服务）监听客户端请求。其工作流程如下：

1. **建立连接**：客户端发起连接请求，服务器接受连接。
2. **协商协议版本**：客户端和服务器协商使用的 SSH 协议版本及加密算法。SSH 协议当前有两个主要版本：**SSH-1**（已废弃）和 **SSH-2**（目前广泛使用）。
3. **密钥交换**：通过 **Diffie-Hellman** 或其他算法交换密钥，以确保接下来的通信内容是加密的。

4. **认证过程**：服务器会要求客户端进行身份验证。可以通过用户名和密码验证，也可以通过**公钥/私钥对**进行验证。公钥存储在服务器上，私钥保留在客户端，使用私钥签名来证明用户身份。
5. **开始加密通信**：通过前面建立的加密会话，客户端和服务器之间的所有数据都会被加密传输。

SSH的常见用途

远程登录：通过 SSH，用户可以安全地登录远程服务器并执行命令，这种方式取代了早期不加密的 Telnet 协议。

安全文件传输：通过 **SCP** 或 **SFTP**，可以安全地在本地和远程服务器之间传输文件。

端口转发：

- **本地端口转发**：把本地端口流量通过 SSH 隧道转发到远程服务器的某个端口，保护本地应用的数据安全。
- **远程端口转发**：把远程服务器的某个端口流量转发到本地指定端口。
- **动态端口转发**：类似于 SOCKS 代理，允许客户端通过SSH访问其他网络服务。

隧道加密：通过 SSH 隧道加密，可以在公共网络中安全地传输敏感数据。

SSH协议的组成

SSH客户端：用于发起连接和与服务器通信的应用程序。常见的 SSH 客户端工具包括 Linux 和 macOS 系统自带的 `ssh` 命令、Windows 上的 PuTTY 等。

SSH服务端：运行在服务器上的守护进程，用于监听 SSH 连接请求，验证用户身份，并进行加密通信。最常见的 SSH 服务端是 **OpenSSH**。

加密算法：

- **对称加密**：SSH 使用对称加密（如 AES、Blowfish）来加密数据，通信双方使用同一个会话密钥加密和解密数据。
- **非对称加密**：用于密钥交换和身份验证，常见的非对称加密算法有 RSA、DSA 和 ECDSA。
- **哈希函数**：用于数据完整性校验，确保数据在传输过程中没有被篡改。常用的哈希算法有 SHA-1、SHA-256 等。

常用的SSH命令

远程登录：`ssh username@hostname`。例如，`ssh root@192.168.1.1` 用于以 `root` 用户身份登录到 IP 为 192.168.1.1 的服务器。

SCP 文件传输：`scp local_file username@hostname:/remote_directory`。例如，`scp file.txt user@192.168.1.1:/tmp/`。

SFTP 文件传输：`sftp username@hostname`，登录后可以使用类似 FTP 的命令操作远程文件。

SFTP 常用命令

连接远程服务器	sftp username@hostname
退出会话	bye 或 exit
列出远程目录	ls
列出本地目录	lls
切换远程目录	cd <remote-directory>
切换本地目录	lcd <local-directory>
显示当前远程目录	pwd
显示当前本地目录	lpwd
上传文件	put <local-file> [<remote-path>]
下载文件	get <remote-file> [<local-path>]
上传多个文件	mput <local-file-pattern>
下载多个文件	mget <remote-file-pattern>
创建远程目录	mkdir <remote-directory>
删除远程文件	rm <remote-file>
删除远程目录	rmdir <remote-directory>
重命名远程文件	rename <oldname> <newname>
更改远程文件权限	chmod <permissions> <remote-file>
显示帮助	help 或 ?

SCP常用命令

操作	命令
从本地复制文件到远程服务器	scp <local-file> username@hostname:<remote-path>
从远程服务器复制文件到本地	scp username@hostname:<remote-file> <local-path>
从本地复制目录到远程服务器	scp -r <local-directory> username@hostname:<remote-path>
从远程服务器复制目录到本地	scp -r username@hostname:<remote-directory> <local-path>
指定端口进行文件传输	scp -P <port> <local-file> username@hostname:<remote-path>
限制带宽进行文件传输	scp -l <limit> <local-file> username@hostname:<remote-path>
使用指定的 SSH 密钥文件	scp -i <key-file> <local-file> username@hostname:<remote-path>
显示文件传输过程中的详细信息	scp -v <local-file> username@hostname:<remote-path>
显示文件传输进度	scp -p <local-file> username@hostname:<remote-path>

SSH协议的安全性

防止中间人攻击：通过加密和密钥交换，SSH 能够防止中间人攻击，即使攻击者能够窃取通信数据，也无法解密内容。

密钥验证：SSH 的公钥/私钥验证机制可以有效防止密码被暴力破解或窃取。

防止重放攻击：SSH 在每次会话中生成一个唯一的会话密钥，确保即便是同一用户在不同时间发起连接，攻击者也无法重用旧的通信数据。

暴力破解

Hydra是一款强大的网络登录破解工具，专门用于暴力破解远程认证服务的密码。它被广泛应用于渗透测试和安全评估中，帮助安全专家检查系统的密码强度和认证机制的安全性。

常用命令：

`hydra -l mark -P /usr/share/wordlists/rockyou.txt 10.10.96.210 ftp`

`hydra -l mark -P /usr/share/wordlists/rockyou.txt ftp://10.10.96.210`

`hydra -l frank -P /usr/share/wordlists/rockyou.txt 10.10.96.210 ssh`

`hydra -L /usr/share/wordlists/seclists/Usernames/top-usernames-shortlist.txt -p 'Goktech' 192.168.134.179 rdp`

一些常用参数：

参数	解释
`-l username`	提供登录名
`-P WordList.txt`	指定要使用的密码列表
`server service`	设置服务器地址和服务进行攻击
`-s PORT`	在非默认服务端口号的情况下使用
`-V` 或 `-vV`	显示正在尝试的用户名和密码组合
`-d`	如果详细输出没有帮助，则显示调试输出

SNMP

SNMP基于UDP，是一种简单的无状态协议，因此容易受到IP欺骗和重放攻击的影响。此外，常用的SNMP协议1、2和2c没有流量加密功能，这意味着SNMP信息和凭据可以在局域网中轻松被拦截。传统的SNMP协议还具有弱的身份验证方案，并且通常以默认的公共和私有社区字符串配置。

默认端口：161