# 模型

TCP/IP四层模型

应用层；传输层；网络层；数据链路层

OSI七层模型

应用层；表示层；会话层；传输层；网络层；数据链路层；物理层

应用层，传输层，网络层常见协议

应用层：HTTP，FTP，DNS，DHCP

传输层：TCP，UDP

网络层：IP，ARP（在OSI模型中属于数据链路层）

常见协议举例

# http和https

## http

### 【2】http request和response报文结构

Request

请求行

请求方法+请求URL+HTTP版本

GET [www.myweb.com/home/img/](http://www.myweb.com/home/img/) HTTP 1.1

请求方法包括：GET，POST，PUT，HEAD，DELETE等

url用于标识请求资源的位置

HTTP版本基本都为1.1 部分服务器使用1.0的版本

请求首部字段

Accept：可处理的媒体类型

Host：请求资源所在服务器

Range：实体的字节请求范围

通用首部字段

Cache-Control：控制缓存的行为

Connection：逐跳首部、连接的管理

Via：代理服务器的相关信息

Warning：错误通知

实体首部字段

Allow：资源可支持的HTTP方法

Content-Encoding：实体主体适用的编码方式

Content-Length：实体主体的长度

其他

Response

状态行

HTTP版本+状态码+状态码简单描述

HTTP 1.1 200 OK

响应首部字段

Accept-Ranges：是否接受字节范围请求（如果不接受，返回完整的请求对象）

Location：重定向至指定URL

Server：HTTP服务器的安装信息

通用首部字段

和Request一致，因此称为通用

实体首部字段

和Request一致，和实体内容相关

### 状态码

1XX：信息性状态码，表示接受的请求正在处理

2XX：成功状态码，请求正常处理完毕

200：ok，请求正常处理，如果请求了资源则资源一起返回。

204：No Content，请求正常处理，但是返回的响应报文不含实体的主体部分，收到此类响应浏览器显示的页面不发生更新（一般用于只需要客户端往服务端发送请求，服务端无需给客户端发送新内容时使用）

206：Partial Content，表示客户端进行了范围请求，服务器成功执行，返回了客户端所需要的部分数据（多个这样的数据进行拼接）

3XX：重定向状态码，需要进行附加操作以完成请求

301：永久重定向，表示资源已经分配了新的URI，新的URI会在响应报文中的Location首部字段中给出

302：Found，临时重定向，禁止POST变GET

303：See Other，请求的资源存在另一个URI，应该使用GET方法想请求获取资源。与302的区别在于303表示希望用户使用GET方法（比如用户使用POST方法访问一个临时重定向的资源）（HTTP1.1之前的版本不支持303，只有302，很多浏览器会默认临时重定向的时候改成GET方法，并不遵守302的要求）

304：Not Modified，和重定向没有关系，表示资源已找到，但是服务端不能满足客户端发送的附带条件。（例如以某种语言版本请求一个网页）

307，临时重定向，允许POST变GET

4XX：客户端错误状态码，服务器无法处理请求

400：Bad Request，请求报文存在语法错误，无法解析

401：Unauthorized，需要进行身份认真

403：Forbidden，对资源的访问被拒绝。和401的区别在于，403可以用来表示可能已经成功登录，但是请求了没有权限的资源

404：Not Found，请求的资源没有找到

5XX：服务端错误状态码，服务器处理请求出错

500：Internal Server Error，服务器在执行请求时出错

503：Service Unavailable，服务器处于不可用的状态，例如超载或者停机维护

### http1.0和1.1和2.x的区别

HTTP 1.1

默认使用长连接，可以减少TCP的握手挥手开销

支持只发送header，不携带body，如果服务器端返回有权限，则发送body

支持host域（一个IP地址对应多个域名的情况，虚拟主机）

支持分批传输，一次只传输一个实体的部分（断点续传的基础）

支持了服务协商（客户端按照不同偏好等级需要一个资源的不同版本，服务端可以识别并提供）

状态响应码有新增，例如303重定向

增加了缓存的一些新特性

增加了OPTIONS方法，用于获取服务端所支持的方法

HTTP 1.0

不支持上述所说的1.1中的内容

HTTP 2.0

使用多路复用技术，允许同时通过单一的HTTP连接发起多重请求/响应消息

新增服务端推送

## https

### 【2】https的过程

准备工作：

服务端生成公钥，在CA（第三方数字证书认证机构）进行注册，机构会对申请的公钥生成数字签名并发布，用于防止服务端时伪造的。

多数浏览器开发商在发布版本时，会预置常用的认证机关的公开密钥。

HTTPS通信步骤

1. 客户端发起握手请求：ClientHello，同时发送自己的SSL版本，加密组件列表等
2. 服务端响应握手请求：ServerHello，包含了服务端筛选后可用的加密组件信息
3. 服务端发送Certificate报文，里面含有公钥和证书
4. 服务端发送：Server Hello Done报文，最初截断握手协商结束
5. 第一次握手后，客户端生成用于对称加密的密钥Pre-master secret，并使用服务端的公钥加密，通过Client Key Exchange报文发送给服务端
6. 接着客户端发送Change Cipher Spec报文，提示服务器之后的报文会采用Pre-master secret进行对称加密
7. 客户端发送Finished报文，其中包括至今全部报文的整体校验值（验证完整性，加密的工作已经在前面完成了，但是方式服务端没有收到或者收到的报文有损）。如果服务器可以正确解密此次报文，则认为此次握手协商成功。
8. 服务器同样发送 Change Cipher Spec报文
9. 服务器发送Finished报文
10. Finished报文交换完成后，SSL连接建立完成
11. 应用层协议通信，发送HTTP报文
12. 客户端断开连接

在上述流程中，应用层发送数据的时候会附加MAC（Message Authentication Code）的报文摘要，用于验证完整性。

### https证书签发过程

1. 服务器把自身公钥登录至数字证书认证机构
2. 数字证书认证机构用自己的私钥对服务器的公钥生成数字签名并颁发公钥证书
3. 客户端从服务端获取公钥证书，向数字证书认证机构确认真实性
4. 公钥确认无误后用公钥加密后续的对称密钥
5. 服务器用私钥进行解密

### 非对称加密，对称加密

非对称加密：

加密密钥和解密密钥不同，加密密钥（公钥）对外发布，解密密钥（私钥）私人持有。如果A和B采用非对称加密进行通讯，那么AB各自都要拥有公钥和私钥。

RSA算法：大数质因数分解，大数作为公钥，因子作为私钥

ECC算法（椭圆曲线算法）：比RSA算法密钥长度更短，但是强度更高

对称加密

使用相同的密钥进行加解密，密钥只能通讯双方持有，不能为第三方所知。但是有着计算量小，加密速度快的特点

DES算法：明文置换算法

### 【2】https和http的区别

http：通常使用80端口，通过TCP完成连接。信息明文传输

https：通常使用443端口。在http之上进行了加密，通过SSL对其提供了安全性的保障，是https的安全版。但是由于握手动作长，内容需要加解密，导致性能要比http低。

SSL使用了序列号来保护报文免受重放攻击

https的局限性：在DDoS和服务器劫持方面难以有效；降低性能；增加成本

### https 和ssl

http + ssl = https

ssl不仅可以用于http，也可以用户ftp等其他应用层协议之上。

## cookie和session

### 【2】cookie和session的区别

两者的目的都是为了解决http无连接的问题

客户端和服务端首次建立连接时，服务端生成Session用于记录用户信息，同时生成cookie，把Session Id加入Cookied返回给客户端。此后客户端为了保持“有状态”的连接效果，访问时要在http报文中加入自己的cookie，服务器读取其中的Sesion Id，找到客户端的对应信息。

cookie在客户端和服务端之间来回传输，而session只有服务端可见

cookie存储在本地浏览器，session存储在服务器

（也存在将session加密存储在cookie中的解决方案）

### cookie的参数有哪些

name：必需。指定Cookie名称

value：必须。规定cookie的值

expire：可选。规定cookie的有效期

path：可选。cookie的服务器路径

domain：可选。规定cookie的域名（“.baidu.com”，必须要以.开头）

secure：可选。是否通过https来传输cookie

### 服务器的域名是google，我可以修改cookie的domain为baidu吗？

可以修改

domain字段决定了该cookie可以被哪个服务器读取而不能被其他服务器读取，如果修改domain为baidu，再去访问Google会发现原来的登录状态等信息失效

如果domain被人篡改为恶意网站，并且进行了转发，则恶意网站可以读取cookie中的内容

### cookie被仿造怎么办（cookie欺骗）

设置响应首部中的Set-Cookie字段HttpOnly，防止XSS跨站脚本攻击

用户不要在cookie中存储明文密码，使用加盐的方式防止敏感信息被窃取

增加完整性校验字段（MAC）用于防篡改

## 【2】GET和POST

### 区别

GET在浏览器退回时无害，POST会再次发送

GET参数通过URL传递，POST参数放在request body中，因此GTE更加不安全

GET只支持ASCII码字符，POST没有限制

GET请求参数会在浏览器的历史记录中保存，POST不会保存

GET的参数长度有限制，POST没有，POST可以用来传输文件

GET和POST都是通过TCP来发送数据，但是GET发送一个数据包，POST至少发送两个（第一次不携带body，防止要进行身份验证等权限问题）（也不是都发两次，这个是由浏览器决定，Firefox只发一次）

### 详解

### 什么协议规定的

HTTP协议

# tcp

## 【3】三次握手

### 是什么

### 为什么需要三次

## 【4】四次挥手

### 【3】time\_wait和close\_wait

### 大量连接处于time\_wait的原因，造成什么问题

### tcp是长连接还是短链接

## 【2】tcp的可靠性

### 拥塞控制

### UDP有拥塞控制吗

### 滑动窗口

### 拥塞窗口

# 网络安全

## DDOS攻击

### 攻击手段

拒绝服务的方式有两种

1.占用服务器的资源，例如网络带宽，磁盘空间等，使服务器无法为正常的请求服务

2.通过安全漏洞使服务停止

攻击方式

SYN Flood：发起大量tcp连接请求（SYN包），进入半连接状态，占满服务器的半连接队列，使真正的请求无法建立TCP连接

UDP洪泛：

Ping of Death

泪滴攻击

### 如何解决

这里只提供SYN Flood的防御思路

1. 缩短半连接队列的等待时间，加速淘汰
2. 第一次握手时不加入等待队列，发送验证信息（例如客户端信息的组合，合法用户返回时会+1，服务端-1再逆向验证）
3. 设置访问频率阈值
4. 验证码

### 关键点

防止被洪泛攻击占用系统资源

提高连接/重连门槛

发现恶意IP并处理

## MD5

### 可逆吗

不可逆

### 会被破解吗？如何破解？

已经存在了较好的碰撞算法，使用彩虹表，在小时级的时间内进行破解。

应对方案

敏感数据进行md5值计算前进行加盐

增加md5的计算轮次

设置较端的过期时间

SHA-1

安全散列算法，和MD5类似，但是长度由40位16进制（MD5只有32位）

# 综合题

【2】输入一个url（例如www.baidu.com）之后发生了什么

短url生成