**HTTPS加密**

HTTPS 并不是新协议，而是让 HTTP 先和 SSL（Secure Sockets Layer）通信，再由 SSL 和 TCP 通信，也就是说 HTTPS 使用了隧道进行通信。

通过使用 SSL，HTTPS 具有了加密（防窃听）、认证（防伪装）和完整性保护（防篡改）。

对称密钥加密

**对称密钥加密**（Symmetric-Key Encryption），加密和解密使用同一密钥。优点：运算速度快；

缺点：无法安全地将密钥传输给通信方。

**非对称密钥加密**

非对称密钥加密，又称公开密钥加密（Public-Key Encryption），加密和解密使用不同的密钥。

公开密钥所有人都可以获得，通信发送方获得接收方的公开密钥之后，就可以使用公开密钥进行加密，接收方收到通信内容后使用私有密钥解密。

优点：可以更安全地将公开密钥传输给通信发送方；

缺点：运算速度慢。

**HTTPS 采用的加密方式**

上面提到对称密钥加密方式的传输效率更高，但是无法安全地将密钥 Secret Key 传输给通信方。而非对称密钥加密方式可以保证传输的安全性，因此我们可以利用非对称密钥加密方式将 Secret Key 传输给通信方。HTTPS 采用混合的加密机制，正是利用了上面提到的方案：

1. 使用非对称密钥加密方式，传输对称密钥加密方式所需要的 Secret Key，从而保证安全性;
2. 获取到 Secret Key 后，再使用对称密钥加密方式进行通信，从而保证效率。（下图中的 Session Key 就是 Secret Key）

**HTTP 头部包含哪些信息**

HTTP 头部本质上是一个传递额外重要信息的键值对。主要分为：通用头部，请求头部，响应头部和实体头部。

协议头

Cache-Control : 用来指定当前的请求/回复中是否使用缓存机制

Accept : 告诉服务器自己允许哪些媒体类型

Age : 创建响应的时间

ETag : 唯一标识分配的资源

Allow 对某网络资源的有效的请求行为，不允许则返回405 Allow: GET, HEAD

Content-encoding 返回内容的编码方式 Content-Encoding: gzip

Content-Length 返回内容的字节长度 Content-Length: 348

Expires 响应过期的日期和时间

**Keep-Alive 和非 Keep-Alive 区别，对服务器性能有影响吗**

因为早期的HTTP/1.0 每次发起HTTP请求都和服务器建立全新的TCP连接 服务器完成处理后立即断开TCP连接

服务器不跟踪每个客户也不记录过去的请求。

在HTTP/1.1版本 默认使用持久连接Keep alive(connetion 头部) keep-alive告诉请求响应完成后不关闭 下次

还用这个请求继续交流 减少资源消耗 缩短响应时间。

**HTTP长连接和短连接的使用场景：**

长连接：多用于操作频繁 点对点通信 而且客户端连接数目较少时的情况

短连接：用于用户数目较多的网站。客户端数量大的情况采用长连接会使得服务端大量资源无效占用。

**HTTP方法知道哪些？**

HTTP/1.0 ： GET, POST, HEAD

HTTP/1.1 : OPTIONS, PUT, PATCH, DELETE, TRACE, CONNECT

GET 请求指定的页面信息，并返回具体内容，通常只用于读取数据。

HEAD 类似于 GET 请求，只不过返回的响应中没有具体的内容，用于获取报头。

POST 向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST 请求可能会导致新的资源的建立或已有资源的更改。

**GET和POST区别**

- GET提交数据放在URL之后 参数直接暴露在URL中 存在安全问题 | 而POST参数放在请求主体中 安全

- GET请求只支持URL编码 post支持多种编码格式

- GET只支持ASCII字符格式参数 POST没有限制

- GET提交数据有限制 而POST没有限制

**GET的长度限制**

HTTP 中的 GET 方法是通过 URL 传递数据的，而 URL 本身并没有对数据的长度进行限制，真正限制 GET 长度的是浏览器

**HTTP 与 HTTPs 的工作方式【建立连接的过程】**

**HTTP**

HTTP（Hyper Text Transfer Protocol: 超文本传输协议） 是一种简单的请求 - 响应协议，被用于在 Web 浏览器和网站服务器之间传递消息。HTTP 使用 TCP（而不是 UDP）作为它的支撑运输层协议。其默认工作在 TCP 协议 80 端口，HTTP 客户机发起一个与服务器的 TCP 连接，一旦连接建立，浏览器和服务器进程就可以通过套接字接口访问 TCP。

**HTTPS**

HTTPS（Hyper Text Transfer Protocol over Secure Socket Layer）是以安全为目标的 HTTP 协议，在 HTTP 的基础上通过传输加密和身份认证的方式保证了传输过程的安全性。其工作流程如下：

① 客户端发起一个 HTTPS 请求，并连接到服务器的 443 端口，发送的信息主要包括自身所支持的算法列表和密钥长度等；

② 服务端将自身所支持的所有加密算法与客户端的算法列表进行对比并选择一种支持的加密算法，然后将它和其它密钥组件一同发送给客户端。

③ 服务器向客户端发送一个包含数字证书的报文，该数字证书中包含证书的颁发机构、过期时间、服务端的公钥等信息。

④ 最后服务端发送一个完成报文通知客户端 SSL 的第一阶段已经协商完成。

⑤ SSL 第一次协商完成后，客户端发送一个回应报文，报文中包含一个客户端生成的随机密码串，称为 pre\_master\_secre，并且该报文是经过证书中的公钥加密过的。

⑥ 紧接着客户端会发送一个报文提示服务端在此之后的报文是采用pre\_master\_secre 加密的。

⑦ 客户端向服务端发送一个 finish 报文，这次握手中包含第一次握手至今所有报文的整体校验值，最终协商是否完成取决于服务端能否成功解密。

⑧ 服务端同样发送与第 ⑥ 步中相同作用的报文，已让客户端进行确认，最后发送 finish 报文告诉客户端自己能够正确解密报文。

当服务端和客户端的 finish 报文交换完成之后，SSL 连接就算建立完成了，之后就进行和 HTTP 相同的通信过程，唯一不同的是在 HTTP 通信过程中并不是采用明文传输，而是采用对称加密的方式，其中对称密钥已经在 SSL 的建立过程中协商好了。

**HTTPS和HTTP区别**

- HTTP协议以明文方式发送内容 数据都是未加密的 安全性较差。 HTTPS数据传输过程是加密的 安全性较好。

- HTTP和HTTPS使用完全不同的连接方式 端口不一样 前者80后者443

- HTTPS协议需要数字认证机构申请证书 需要一定的费用

- HTTP页面响应比HTTPS快 因为HTTP使用3次握手 而HTTPS除了TCP的三次握手 还需要SSL的协商过程。

**HTTP 是不保存状态的协议,如何保存用户状态**

因为一个 HTTP 服务器并不保存关于客户机的任何信息，所以我们说 HTTP 是一个无状态协议。

① 基于 Session 实现的会话保持

在客户端第一次向服务器发送 HTTP 请求后，服务器会创建一个 Session 对象并将客户端的身份信息以键值对的形式存储下来，然后分配一个会话标识（SessionId）给客户端，这个会话标识一般保存在客户端 Cookie 中，之后每次该浏览器发送 HTTP 请求都会带上 Cookie 中的 SessionId 到服务器，服务器根据会话标识就可以将之前的状态信息与会话联系起来，从而实现会话保持。

② 基于 Cookie 实现的会话保持

当服务器发送响应消息时，在 HTTP 响应头中设置 Set-Cookie 字段，用来存储客户端的状态信息。客户端解析出 HTTP 响应头中的字段信息，并根据其生命周期创建不同的 Cookie，这样一来每次浏览器发送 HTTP 请求的时候都会带上 Cookie 字段，从而实现状态保持。基于 Cookie 的会话保持与基于 Session 实现的会话保持最主要的区别是前者完全将会话状态信息存储在浏览器 Cookie 中。

**状态码**

1XX 指示信息--表示请求正在处理

2XX 成功--表示请求已被成功处理完毕

3XX 重定向--要完成的请求需要进行附加操作

4XX 客户端错误--请求有语法错误或者请求无法实现，服务器无法处理请求

5XX 服务器端错误--服务器处理请求出现错误

**HTTP/1.1 和 HTTP/1.0 的区别**

缓存处理：在 HTTP/1.0 中主要使用 header 里的 if-modified-Since, Expries 来做缓存判断的标准

Host 请求头：早期 HTTP/1.0 中认为每台服务器都绑定一个唯一的 IP 地址并提供单一的服务，请求消息中的 URL 并没有传递主机名。

长连接：HTTP/1.0 默认浏览器和服务器之间保持短暂连接，浏览器的每次请求都需要与服务器建立一个 TCP 连接，服务器完成后立即断开 TCP 连接。

**HTTP/1.X 和 HTTP/2.0 的区别**

相比于 HTTP/1.X 的文本（字符串）传送， HTTP/2.0 采用二进制传送。客户端和服务器传输数据时把数据分成帧，帧组成了数据流，流具有流 ID 标识和优先级，通过优先级以及流依赖能够一定程度上解决关键请求被阻塞的问题。

HTTP/2.0 支持多路复用。

HTTP/2.0 头部压缩。

HTTP/2.0 支持服务器推送。

**HTTP/3 了解吗**

QUIC（Quick UDP Internet Connections），直译为快速 UDP 网络连接，是谷歌制定的一种基于 UDP 的低延迟传输协议。其主要目的是解决采用传输层 TCP 协议存在的问题，同时满足传输层和应用层对多连接、低延迟等的需求。该协议融合了 TCP, TLS, HTTP/2 等协议的特性，并基于 UDP传输。

HTTP/3 是在 QUIC 基础上发展起来的，其底层使用 UDP 进行数据传输，上层仍然使用 HTTP/2。在 UDP 与 HTTP/2 之间存在一个 QUIC 层，其中 TLS 加密过程在该层进行处理

**DNS 的作用和原理**

DNS（Domain Name System）是域名系统的英文缩写，是一种组织成域层次结构的计算机和网络服务命名系统，用于 TCP/IP 网络。

通常我们有两种方式识别主机：通过主机名或者 IP 地址。人们喜欢便于记忆的主机名表示，而路由器则喜欢定长的、有着层次结构的 IP 地址。为了满足这些不同的偏好，我们就需要一种能够进行主机名到 IP 地址转换的目录服务，域名系统作为将域名和 IP 地址相互映射的一个分布式数据库，能够使人更方便地访问互联网。

树形结构 - 根域名服务器、顶级域名服务器和权威域名服务器

**怎么实现DNS劫持**

DNS 劫持即域名劫持，是通过将原域名对应的 IP 地址进行替换从而使得用户访问到错误的网站或者使得用户无法正常访问网站的一种攻击方式。

**socket套接字有哪些**

套接字（Socket）是对网络中不同主机上的应用进程之间进行双向通信的端点的抽象，网络进程通信的一端就是一个套接字，不同主机上的进程便是通过套接字发送报文来进行通信。

1. 流套接字 SOCK\_STREAM TCP

2. 数据报套接字 SOCK\_DGRAM UDP

3. 原始套接字 SOCK\_RAW

**URI（统一资源标识符）和 URL（统一资源定位符）之间的区别**

URL，即统一资源定位符 (Uniform Resource Locator )，URL 其实就是我们平时上网时输入的网址，它标识一个互联网资源，并指定对其进行操作或获取该资源的方法。

URL 是 URI 的一个子集，两者都定义了资源是什么，而 URL 还定义了如何能访问到该资源。URI 是一种语义上的抽象概念

**如果你访问一个网站很慢，怎么排查和解决**

查看本地网络是否正常

通过 ping 命令查看链接到服务器的时间和丢包等情况

**网页解析全过程【用户输入网址到显示对应页面的全过程】**

① DNS 解析：当用户输入一个网址并按下回车键的时候，浏览器获得一个域名，而在实际通信过程中，我们需要的是一个 IP 地址，因此我们需要先把域名转换成相应 IP 地址

② TCP 连接：浏览器通过 DNS 获取到 Web 服务器真正的 IP 地址后，便向 Web 服务器发起 TCP 连接请求，通过 TCP 三次握手建立好连接后，浏览器便可以将 HTTP 请求数据发送给服务器了。

③ 发送 HTTP 请求：浏览器向 Web 服务器发起一个 HTTP 请求，HTTP 协议是建立在 TCP 协议之上的应用层协议，其本质是在建立起的TCP连接中，按照HTTP协议标准发送一个索要网页的请求。在这一过程中，会涉及到负载均衡等操作。

拓展：什么是负载均衡？

负载均衡，英文名为 Load Balance，其含义是指将负载（工作任务）进行平衡、分摊到多个操作单元上进行运行，例如 FTP 服务器、Web 服务器、企业核心服务器和其他主要任务服务器等，从而协同完成工作任务。负载均衡建立在现有的网络之上，它提供了一种透明且廉价有效的方法扩展服务器和网络设备的带宽、增加吞吐量、加强网络处理能力并提高网络的灵活性和可用性。

④ 处理请求并返回：服务器获取到客户端的 HTTP 请求后，会根据 HTTP 请求中的内容来决定如何获取相应的文件，并将文件发送给浏览器。

⑤ 浏览器渲染：浏览器根据响应开始显示页面，首先解析 HTML 文件构建 DOM 树，然后解析 CSS 文件构建渲染树，等到渲染树构建完成后，浏览器开始布局渲染树并将其绘制到屏幕上。

⑥ 断开连接：客户端和服务器通过四次挥手终止 TCP 连接。