2022-2023年网络安全试卷自制答案

一、简答题（每题 10 分，共 40 分）

**1.简述操作系统识别中协议栈指纹技术的基本原理**

【崔竞松PPT网络侦查31 59页】

寻找不同操作系统之间在处理网络数据包上的差异，并且把足够多的差异组合起来，以便精确地识别出一个系统的OS版本

【网络资料】

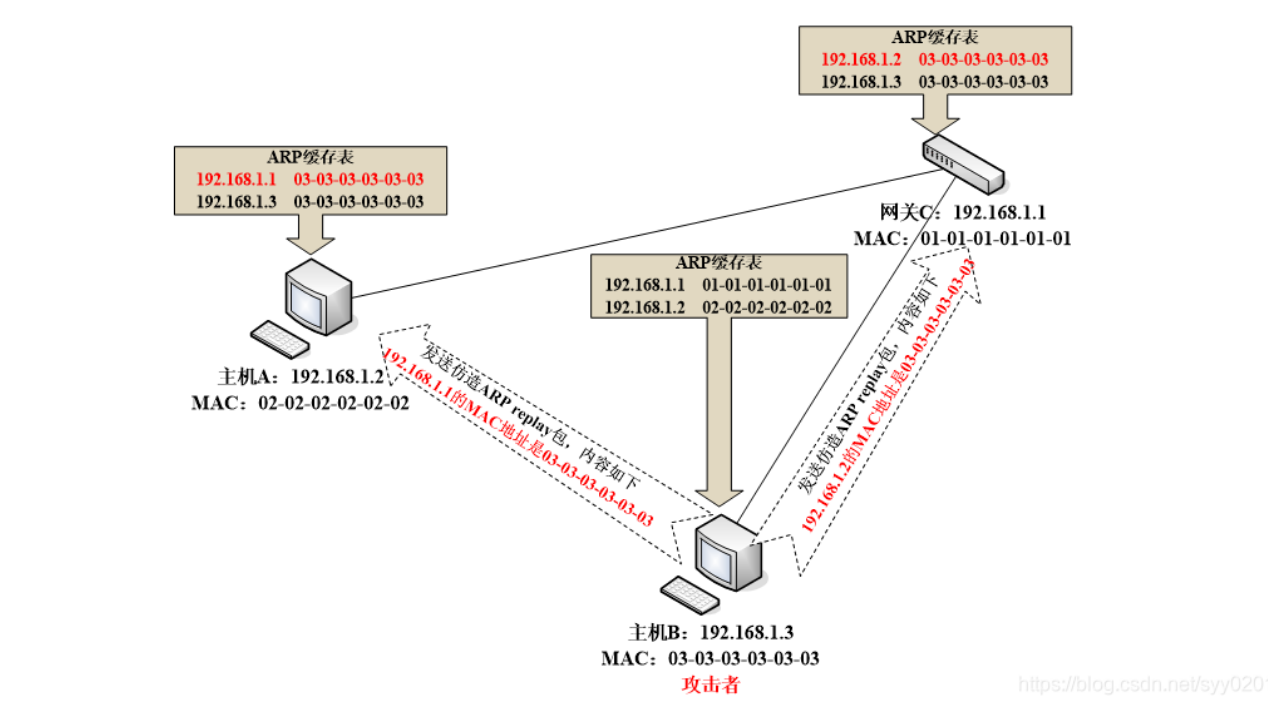
常用的网络协议是标准的，因而从理论上讲各个操作系统的协议栈应该是相同的。但是，在实践中，各种操作系统的协议栈的实现存在细微的差异。这些差异称作网络协议栈的“指纹”。

对TCP协议族来说，这些差异通常表现在数据包头的标志字段中。如window size、ACK序号、TTL等的不同取值。通过对这些差别进行归纳和总结，可以比较准确地识别出远程系统的OS类型。

**2.简述 ARP 欺骗的基本原理及防范方法**

【网络资料】

原理：主机接收到一个应答包之后，并不会验证自己是否发送过对应的arp请求包，也不会验证这个arp请求包是否可信，而是直接用应答包里的IP地址和mac地址的对应关系替换掉arp缓存表里原有的对应关系。



攻击主机B向网关C发送一个应答包，包括主机A的IP地址、主机B的mac地址。同时，向主机A发送一个应答包，包括网关C的IP地址，主机B的mac地址。

这个时候，网关C就会将缓存表里主机A的mac地址换成主机B的mac地址，而主机a就会将缓存表里网关C的mac地址换成主机B的mac地址。

所以，网关C发送给主机A的消息全被主机B接收，主机A发送给网关C的消息也全被主机B接收，主机B就成为主机A和网关C通信的“中间人”。

检测：

1、网络频繁掉线

2、网速突然变慢

3、使用arp a命令查看的网关mac地址与真实的网关mac地址不同

4、使用嗅探软件发现局域网内存在大量arp应答包

防御：

1、绑定mac地址

2、使用静态arp缓存表

3、使用arp服务器，通过服务器来查找arp转换表来响应其他机器的广播

4、使用arp欺骗防护软件

ARP欺骗的危害：

1、使同一网段内其他用户无法上网

2、可以嗅探到交换式局域网中的所有数据包

3、对信息进行篡改

4、可以控制局域网内任何主机

**3．假设使用 ping 工具执行以下命令“ping www.baidu.com”，发现结果为 Time Exceeded，简述该结果可能的原因**

【cjs ppt网络侦查31的22、26页】有关于Time Exceeded介绍

【GPT】

TTL（生存时间）值耗尽：

每个 IP 数据包都有一个 TTL 字段，它指定数据包在被丢弃之前可以经过的最大路由器数量。每经过一个路由器，TTL 值减1。如果 TTL 值降到0，路由器将丢弃该数据包并返回 "Time Exceeded" 错误消息。这个错误表明数据包在到达目标服务器前经过了太多的路由器。

网络拥塞：

网络路径中的某些节点可能存在严重的网络拥塞，导致数据包在规定的时间内无法转发。网络拥塞会导致延迟增加，使得数据包在到达目标前被丢弃。

路由循环：

数据包可能在网络路径中进入了路由循环，即数据包在同一组路由器之间循环传递，直到 TTL 耗尽为止。路由循环通常是由于网络配置错误或路由器故障引起的。

目标网络或服务器过载：

目标网络或服务器可能过载或存在故障，导致数据包在传输过程中被延迟或丢弃。这种情况下，数据包可能在到达目标之前耗尽 TTL。

防火墙或安全设备：

某些防火墙或网络安全设备可能会配置为丢弃 ping 数据包或对其进行限速。如果这些设备在路径中，可能会导致数据包被丢弃并返回 "Time Exceeded" 错误。

网络路径变化：

网络路径中的动态路由变化可能导致数据包在不同路径上传输，延迟增加或进入路由环路，导致 TTL 耗尽。

**4.简述 IPSec 协议中数字证书的作用**

【网络资料】找不到很准确的回答，所以找了比较接近的资料作为补充

IPSec应用方案之所以能在公网（如Internet）上安全地进行网络通信，其重要原因是可在对等体间的整个隧道建立和数据传输过程中均有各种安全机制来做保障，这方面如果采用的是IKE来进行自动的密钥交换和协商同样可以做到，因为IKE本身就具有一整套自我保护机制，可以在不安全的网络上安全地认证身份、分发密钥。具体体现在以下几种安全保护方面。

1、身份认证机制

当使用IKE在对等体间进行信息交换时，首先要识别对方的合法性，也就是身份认证问题。在IKE中可用于确定对等体身份（对等体的IP地址或名称）的机制比较全面，包括预共享密钥PSK（preshared key）认证、**RSA数字证书（rsasignature，或称RSA数字签名）认证**和RSA数字信封认证。

**（2）数字证书认证**

在数字证书认证中，通信双方使用CA证书进行数字证书合法性验证。在CA证书中，双方有各自的公钥（网络上传输）和私钥（自己持有）。发送方对原始报文进行哈希运算，并用自己的私钥对报文计算结果进行加密，生成数字签名。接收方使用发送方的公钥对数字签名进行解密，然后采用相同的哈希算法对解密后的报文进行哈希算，看运算的结果与解密发送方发来的哈希值是否相同。如果相同，则认证通过；否则认证失败。

对于预共享密钥认证方法，当有一个对等体对应多个对等体时，需要为每个对等体配置预共享的密钥，工作量大，所以该方法在小型网络中容易建立，但安全性较低。**使用数字证书安全性高，但需要CA来颁发数字证书，适合在大型网络中使用。**而数字信封认证用于设备需要符合国家密码管理局要求时使用（需要使用国家密码管理局要求的哈希算法SM3），且此认证方法只能在IKEv1的主模式协商过程中支持。

以上所涉及的身份认证密钥（包括预共享密钥、公/私钥）、**证书**都是作为发送方的“验证数据”要通过对应方式发给对方予以**验证**的。

【GPT】

在 IPSec（Internet Protocol Security）协议中，数字证书的作用主要包括以下几个方面：

1. 身份验证：

数字证书用于验证通信双方的身份。通过使用受信任的证书颁发机构（CA）签发的数字证书，IPSec 确保通信双方确实是其声称的实体。这种身份验证方式比基于共享密钥的方法更安全，因为共享密钥可能被泄露或被第三方攻击者猜测到。

2. 公钥分发：

数字证书包含通信双方的公钥。通过数字证书，IPSec 可以安全地分发公钥，从而避免了公钥分发中的中间人攻击问题。通信双方可以使用对方的公钥来加密会话密钥或进行数字签名验证。

3. 增强安全性：

使用数字证书可以提供更高的安全性，因为每个参与 IPSec 连接的实体都需要一个唯一的、受信任的证书。这有效地防止了未经授权的实体参与通信，并提高了整体网络的安全性。

4. 可扩展性：

数字证书使得大型网络中的身份验证管理更加可扩展和可管理。在一个大规模的网络环境中，管理共享密钥的数量和分发变得非常复杂和不安全，而数字证书可以由 CA 进行集中管理和分发，简化了密钥管理。

5. 互操作性：

数字证书提供了一种标准化的身份验证方式，使得不同厂商的 IPSec 实现能够互操作。这是因为数字证书和相关的公钥基础设施（PKI）是基于开放标准的，可以被多种硬件和软件实现所支持。

数字证书的工作流程

1. 证书获取：

通信实体向 CA 申请并获取数字证书。CA 验证实体的身份并签发证书，其中包含实体的公钥和其他信息（如实体名称、证书有效期等）。

2. 证书交换：

在建立 IPSec 连接时，通信双方交换各自的数字证书。这个过程通常发生在 IKE（Internet Key Exchange）协议的初始阶段。

3. 证书验证：

通信双方使用 CA 的公钥验证对方的数字证书，确保证书的真实性和完整性。这一步确保证书确实由受信任的 CA 签发，并未被篡改。

4. 会话密钥生成：

一旦身份验证成功，通信双方使用对方的公钥加密和交换会话密钥。会话密钥用于后续数据包的加密和解密，确保数据传输的保密性和完整性。

5. 安全通信：

通过使用数字证书和会话密钥，IPSec 建立起安全的通信通道，所有传输的数据都经过加密和认证，防止未经授权的访问和篡改。

通过这些功能，数字证书在 IPSec 协议中发挥了至关重要的作用，确保网络通信的安全性、可靠性和可管理性。

二、论述题（每题 15 分，共 60 分）

**1.叙述针对 https 协议的中间人攻击的工作原理**

【cjsPPT欺骗攻击55页】

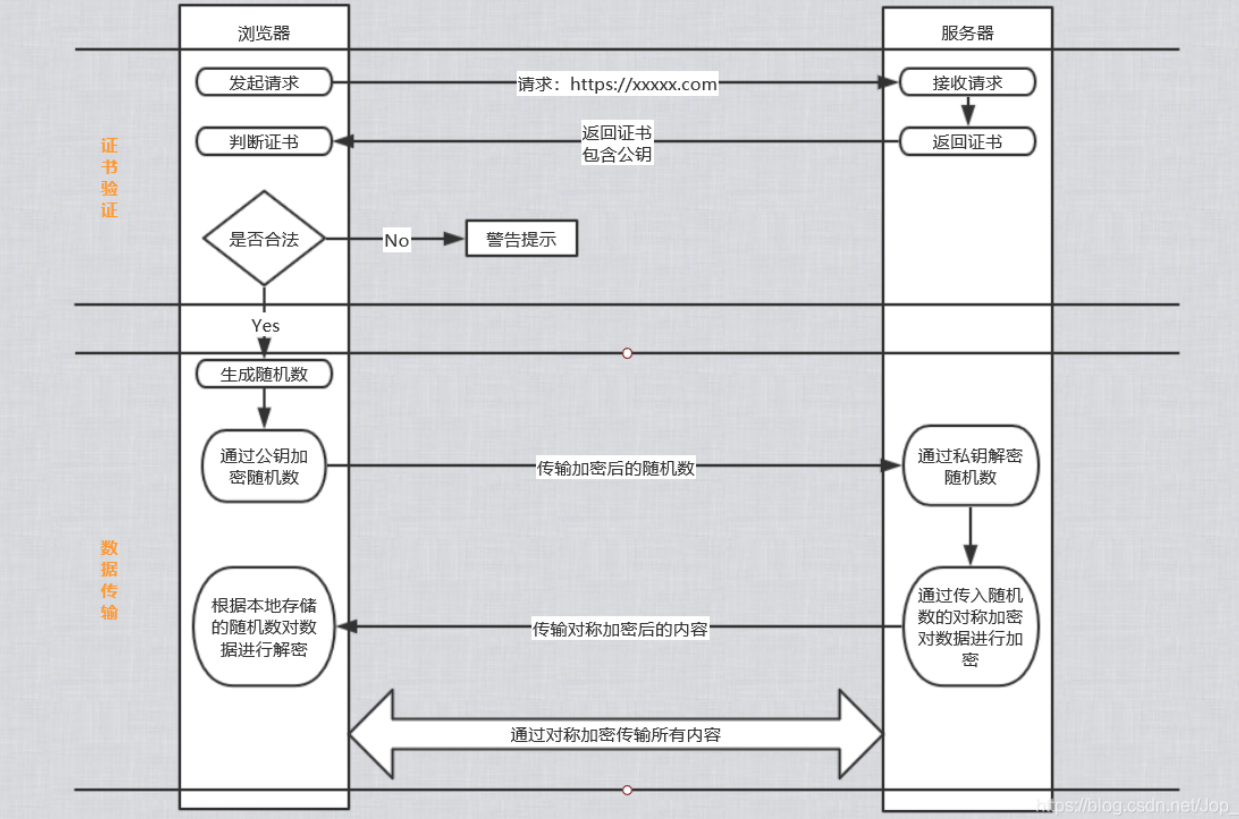
https会话劫持之SSL Strip

【网络资料】

（1）HTTPS实现原理：

Https协议在内容传输上使用的加密是“对称加密”，而“非对称加密”只作用于证书验证阶段。

Https的整体实现过程分为“证书验证”和“数据传输”两个阶段，过程如下所示：



证书验证阶段

1.浏览器发起Https请求；

2.服务器端返回Https证书；

3.浏览器客户端验证证书是否合法，若不合法则提示警告

数据传输阶段

4.当证书验证合法后，在客户端本地生成随机数；

5.通过公钥加密随机数，并将加密后的随机数传输到服务端；

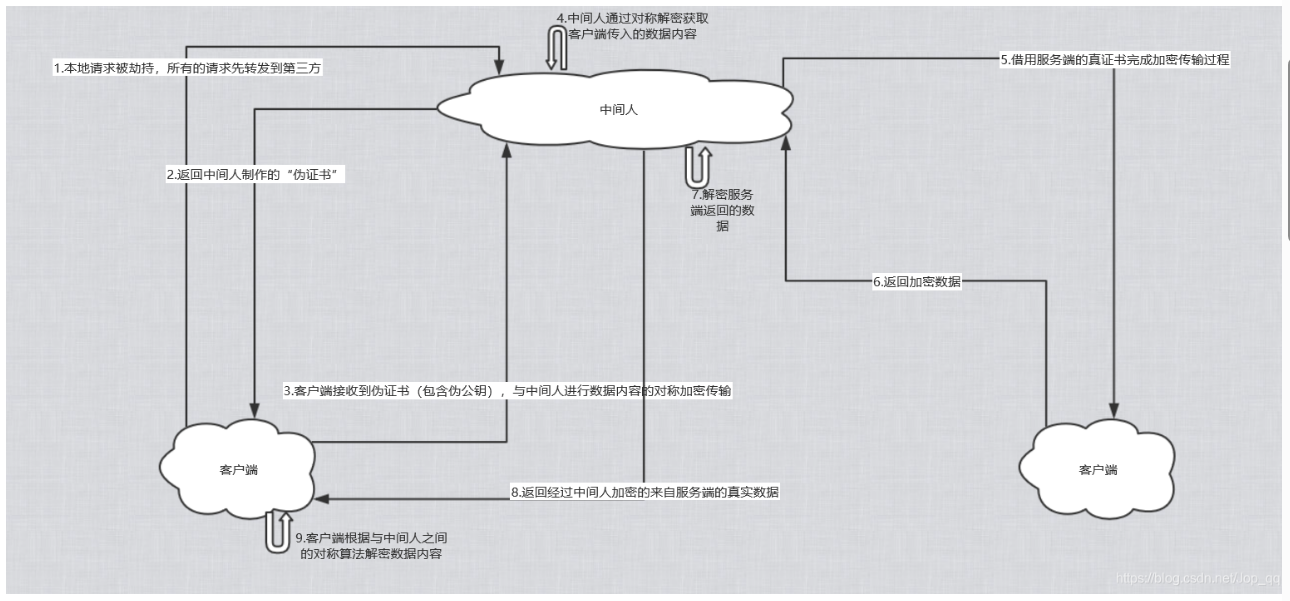
6.服务端通过私钥对接收到的加密随机数进行解密操作；

7.服务端通过客户端传入的随机数构造对称加密算法，对返回结果内容进行加密操作后再进行内容传输。

（2）为什么需要CA认证机构颁发证书？

Http协议被认为不安全是因为传输过程容易被监听者勾线监听、伪造服务器，而Https协议主要就是解决网络传输的安全性问题。

首先，我们假设不存在认证机构，任何人都可以制作证书，这存在的风险便是经典的“中间人攻击”问题。具体过程如下：



（3）“中间人攻击”原理

1.客户端请求被劫持（如DNS劫持等），所有的客户端请求均被转发至中间人的服务器；

2.中间人服务器返回中间人伪造的“伪证书”（包含伪公钥）；

3.客户端创建随机数，通过中间人证书的伪公钥对随机数进行加密后传输给中间人，然后凭随机数构造对称加密算法对要进行传输的数据内容进行对称加密后传输；

4.中间人因为拥有客户端生成的随机数，从而能够通过对称加密算法进行数据内容解密；

5.中间人再以“伪客户端”的身份向正规的服务端发起请求；

6.因为中间人与服务器之间的通信过程是合法的，正规服务端通过建立的安全通道返回加密后的数据内容；

7.中间人凭借与正规服务器建立的对称加密算法进行数据内容解密；

8.中间人再通过与客户端建立的对称加密算法对正规服务器返回的数据内容进行加密传输；

9.客户端通过中间人建立的对称加密算法对返回的数据内容进行解密；

由于缺少对证书的真伪性验证，所有客户端即使发起了Https请求，但客户端完全不知道自己发送的请求已经被第三方拦截，导致其中传输的数据内容被中间人窃取。

【GPT】

针对 HTTPS 协议的中间人攻击（ManintheMiddle Attack, MITM）的工作原理可以概括为以下几个步骤：

1. 截获通信

中间人攻击的第一步是攻击者在客户端（用户）和服务器之间插入自己，截获双方的通信。攻击者可以通过多种方式实现这一点，比如：

ARP 欺骗：在局域网内，通过发送伪造的 ARP 响应，将网关的 MAC 地址替换为攻击者的 MAC 地址，从而截获客户端的流量。

DNS 欺骗：通过劫持 DNS 请求，将用户请求的域名解析为攻击者控制的 IP 地址。

恶意 WiFi 热点：创建一个开放或看似合法的 WiFi 热点，当用户连接后，所有流量都经过攻击者的设备。

2. 建立伪造的 HTTPS 连接

一旦成功截获通信，攻击者将分别与客户端和服务器建立单独的 HTTPS 连接：

与客户端的连接：攻击者向客户端发送一个伪造的数字证书，使客户端认为它正在与目标服务器通信。这个证书看起来像目标服务器的证书，但实际上是由攻击者生成的。

与服务器的连接：攻击者与目标服务器建立真实的 HTTPS 连接，获取目标服务器的真实数字证书和公钥。

3. 转发和篡改数据

攻击者在两个 HTTPS 连接之间充当中间人：

解密和篡改数据：攻击者可以解密客户端发送的数据包，读取其中的内容，甚至进行篡改，然后再用服务器的公钥重新加密并发送给服务器。

获取服务器的响应：攻击者接收到服务器的响应后，可以进行解密、读取和篡改，再用伪造的证书加密并发送给客户端。

4. 证书欺骗

中间人攻击的关键在于如何让客户端接受伪造的证书。攻击者可能会利用以下方法：

自签名证书：攻击者使用自签名证书。如果客户端未对证书进行严格验证（例如用户忽略了浏览器的安全警告），就可能会被欺骗。

伪造受信任证书：攻击者可能会利用已被泄露的私钥或通过社会工程学手段获取受信任的证书颁发机构（CA）的签名，从而生成一个看似合法的证书。

5. 客户端防护机制

为了防止中间人攻击，HTTPS 协议和浏览器实现了一些防护机制：

证书验证：浏览器会验证服务器提供的数字证书是否由受信任的 CA 签发，以及证书的有效期和域名匹配情况。如果验证失败，浏览器会显示警告信息。

证书透明度（Certificate Transparency）：通过维护公开的证书日志，帮助检测和防止非法签发的证书。

HTTP 严格传输安全（HSTS）：服务器可以通过 HSTS 头告知浏览器，要求在一定时间内只使用 HTTPS 连接，防止降级攻击。

6. 攻击效果和后果

成功的中间人攻击允许攻击者：

窃取敏感信息：如登录凭据、信用卡信息、个人数据等。

篡改数据：如修改交易信息、注入恶意代码等。

伪装身份：攻击者可以冒充用户或服务器进行恶意操作。

**2.叙述交换式以太网中的两种常见监听方式**

ARP欺骗和+sniffer

【教材44页】

【zwh题目整合第三章第6题】



**3.叙述 sql 注入的原理及防御方法**

（完全实验课知识点）

【网络资料】

1. 什么是SQL注入

sqL注入即是指web应用程序对用户输入数据的合法性没有判断或过滤不严，攻击者可以在web应用程序中事先定义好的查询语句的结尾上添加额外的SQL语句，在管理员不知情的情况下实现非法操作，以此来实现欺骗数据库服务器执行非授权的任意查询，从而进一步得到相应的数据信息。

1. SQL注入原理

**SQL注入的本质：把用户输入的数据当作代码来执行，违背了“数据与代码分离”的原则**

SQL注射能使攻击者绕过认证机制，完全控制远程服务器上的数据库。 SQL是结构化查询语言的简称，它是访问数据库的事实标准。目前，大多数Web应用都使用SQL数据库来存放应用程序的数据。几乎所有的Web应用在后台 都使用某种SQL数据库。跟大多数语言一样，SQL语法允许数据库命令和用户数据混杂在一起的。如果开发人员不细心的话，用户数据就有可能被解释成命令， 这样的话，远程用户就不仅能向Web应用输入数据，而且还可以在数据库上执行任意命令了。

SQL注入式攻击的主要形式有两种：

♦一是直接将代码插入到与SQL命令串联在一起并使得其以执行的用户输入变量。上面笔者举的例子就是采用了这种方法。由于其直接与SQL语句捆绑，故也被称为直接注入式攻击法。

♦二是一种间接的攻击方法，它将恶意代码注入要在表中存储或者作为原书据存储的字符串。在存储的字符串中会连接到一个动态的SQL命令中，以执行一些恶意的SQL代码。注入过程的工作方式是提前终止文本字符串，然后追加一个新的命令。如以直接注入式攻击为例。就是在用户输入变量的时候，先用一个分号结束当前的语句。然后再插入一个恶意SQL语句即可。由于插入的命令可能在执行前追加其他字符串，因此攻击者常常用注释标记“—”来终止注入的字符串。执行时，系统会认为此后语句位注释，故后续的文本将被忽略，不背编译与执行。

1. 防御

　1、过滤掉一些常见的数据库操作关键字：select,insert,update,delete,and,\*等

或者通过系统函数：addslashes(需要被过滤的内容)来进行过滤

　　2、在PHP配置文件中Register\_globals=off;设置为关闭状态 //作用将注册全局变量关闭。

比如：接收POST表单的值使用$\_POST['user'],如果将register\_globals=on;直接使用$user可以接收表单的值。

3、SQL语句书写的时候尽量不要省略小引号(tab键上面那个)和单引号

4、提高数据库命名技巧，对于一些重要的字段根据程序的特点命名，取不易被猜到的

5、对于常用的方法加以封装，避免直接暴漏SQL语句

6、开启PHP安全模式Safe\_mode=on;

7、打开magic\_quotes\_gpc来防止SQL注入Magic\_quotes\_gpc=off;默认是关闭的，它打开后将自动把用户提交的sql语句的查询进行转换，把'转为\'，这对防止sql注入有重大作用。

因此开启：magic\_quotes\_gpc=on;

8、控制错误信息

关闭错误提示信息，将错误信息写到系统日志。

9、使用mysqli或pdo预处理。

【GPT】

**原理：**SQL 注入（SQL Injection）是一种常见的网络攻击技术，攻击者通过在输入字段中插入恶意的 SQL 代码，从而操控应用程序的数据库查询，进而访问、修改、删除数据库中的数据。其原理可以概括为以下几个步骤：

输入构造：攻击者在应用程序的输入字段（如登录表单、搜索框等）中输入恶意的 SQL 语句。这个输入被直接嵌入到应用程序的 SQL 查询中。

SQL 拼接：应用程序将用户输入直接拼接到 SQL 查询字符串中，没有进行适当的过滤或转义。

查询执行：拼接后的 SQL 查询被数据库执行。由于恶意输入被当作合法的 SQL 代码执行，攻击者可以执行任意 SQL 语句。

**防御：**

1. 使用预处理语句和参数化查询：预处理语句和参数化查询将 SQL 代码和数据分开处理，从而避免将用户输入当作 SQL 代码执行。

2．输入验证和清理：对用户输入进行严格的验证和清理，确保输入符合预期格式。

检查输入长度、类型和格式。

对特殊字符进行转义或删除。

1. 使用 ORM 框架：对象关系映射（ORM）框架将数据库操作抽象为对象操作，自动处理参数化查询和预处理语句。
2. 最小权限原则：数据库用户仅授予必要的最低权限，避免使用具有高级权限的数据库用户。
3. 数据库层的防御：在数据库层面启用防御机制，如限制查询的复杂性、启用防火墙规则等。
4. 错误信息隐藏：避免将详细的数据库错误信息暴露给用户，防止攻击者获取系统内部信息。

**4.针对一个 B/S 架构 web 应用，请构建一个安全防护方案，要求能够保障前后端应用的安全性并论述其可能遭受的攻击及对应的防护原理。**

【GPT】

一、前端安全

1. 输入验证和净化

可能遭受的攻击：

跨站脚本攻击（XSS）：攻击者通过在输入字段中注入恶意脚本，诱使用户浏览器执行这些脚本。

防护措施：

使用客户端脚本（如 JavaScript）对输入进行初步验证。

在发送到服务器前，对输入进行编码或净化，移除或转义特殊字符。

防护原理：

确保用户输入仅包含预期的格式和内容，防止恶意代码被执行。

2. 安全的内容展示

可能遭受的攻击：

XSS：通过在页面展示的内容中嵌入恶意脚本。

防护措施：

使用安全的模板引擎，自动对输出进行转义。

对动态生成的 HTML 内容进行编码，确保输出到页面的内容不包含可执行代码。

防护原理：

防止浏览器将用户输入解释为代码执行。

二、后端安全

1. SQL 注入防护

可能遭受的攻击：

SQL 注入：攻击者通过输入恶意 SQL 代码操纵数据库查询。

防护措施：

使用预处理语句和参数化查询。

对输入进行严格验证和清理。

防护原理：

将 SQL 代码与数据分开处理，防止用户输入被当作 SQL 代码执行。

2. 认证和授权

可能遭受的攻击：

暴力破解：攻击者尝试大量用户名和密码组合，获取访问权限。

权限提升：未授权用户通过漏洞获得更高权限。

防护措施：

使用强密码策略和密码哈希（如 bcrypt）。

实现账户锁定和速率限制机制，防止暴力破解。

采用基于角色的访问控制（RBAC）和最小权限原则。

防护原理：

保护用户账户安全，防止未授权访问和权限滥用。

三、数据传输安全

1. HTTPS

可能遭受的攻击：

中间人攻击（MITM）：攻击者拦截和篡改数据传输。

防护措施：

强制使用 HTTPS，确保所有数据传输加密。

使用强加密算法和有效的证书。

防护原理：

加密数据传输，防止数据在传输过程中被窃取和篡改。

四、服务器和网络安全

1. 防火墙和入侵检测系统（IDS）

可能遭受的攻击：

DDoS 攻击：攻击者通过大量请求使服务器过载。

端口扫描和服务漏洞利用。

防护措施：

配置网络防火墙和 Web 应用防火墙（WAF），过滤恶意流量。

部署入侵检测系统（IDS），监控和分析可疑活动。

防护原理：

保护服务器免受恶意流量和攻击，及时发现和响应安全威胁。

2. 安全的服务器配置

可能遭受的攻击：

服务配置错误：攻击者利用不安全的服务器配置。

防护措施：

定期更新和修补操作系统和应用程序。

关闭不必要的服务和端口。

使用安全配置指南，确保服务器配置符合最佳实践。

防护原理：

降低攻击面，防止攻击者利用已知漏洞和不安全配置。

五、日志和监控

可能遭受的攻击：

持续攻击：攻击者不断尝试不同方式攻击系统。

防护措施：

实施全面的日志记录机制，记录所有访问和操作。

部署实时监控系统，分析日志并触发安全警报。

防护原理：

提供攻击事件的可追溯性，及时发现和响应安全事件。

六、定期安全审计和测试

可能遭受的攻击：

未知漏洞：攻击者利用系统中未发现的漏洞。

防护措施：

进行定期的安全审计和渗透测试。

使用自动化工具扫描代码和依赖库的安全漏洞。

防护原理：

及时发现和修复安全漏洞，确保系统安全性不断提升。

**结论**

通过实施上述安全防护措施，可以有效保障 B/S 架构 Web 应用的安全性，防止常见的攻击方式。每一层防护措施都有其独特的作用，共同构建了一个全方位的安全防护体系。定期审计和持续监控确保了系统能够及时应对新的安全威胁。