Dos攻击（拒绝服务攻击）：（简单有效并且具有很大危害性的攻击方法。它通过各种手段消耗网络带宽和系统资源，或者攻击系统缺陷，使正常系统的正常服务陷于瘫痪状态，不能对正常用户进行服务，从而实现拒绝正常的用户访问服务。）

1. **缩短SYN Timeout时间**，及时将超时请求丢弃，释放被占用CPU和内存资源。
2. **限制同时打开的SYN半连接数目，关闭不必要的服务**。

（3）**设置SYN Cookie，给每一个请求连接的IP地址分配一个Cookie（**Cookie 是在 HTTP 协议下，服务器或脚本可以维护客户工作站上信息的一种方式。Cookie 是由 Web 服务器保存在用户浏览器（客户端）上的小文本文件，它可以包含有关用户的信息。无论何时用户链接到服务器，Web 站点都可以访问 Cookie 信息。）**。**如果短时间内连续受到某个IP的重复SYN报文，就认定是受到了攻击，以后从这个IP地址来的包会被一概丢弃。 一般来说，第三种方法在防范该类问题上表现更佳。同时可以在Web服务器端采用分布式组网、负载均衡、提升系统容量等可靠性措施，增强总体服务能力。

DDos攻击（分布式拒绝服务攻击）：（**借助于客户/服务器技术，将多个计算机联合起来作为攻击平台，对一个或多个目标发动DoS攻击。）**(1) **最直接的方法增加带宽。**（流量攻击、针对网络带宽进行攻击，因为每个网站的服务带宽都是有限的，利用大量的攻击包将带宽占满，使其合法网络包无法到达主机完成正常的访问。）

（2）**负载均衡。**负载均衡建立在现有网络结构之上，它提供了一种廉价有效透明的方法扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性，对DDoS流量攻击和CC攻击都很见效。

（3）**CDN流量清洗。**目前大部分的CDN节点都有200G 的流量防护功能，再加上硬防的防护，可以应付大部分DDoS攻击。

CSRF攻击（跨站请求伪造）：(1) **验证码**。应用程序和用户进行交互过程中，特别是账户交易这种核心步骤，强制用户输入验证码，才能完成最终请求。在通常情况下，验证码够很好地遏制CSRF攻击。（但增加验证码降低了用户的体验，网站不能给所有的操作都加上验证码。所以只能将验证码作为一种辅助手段，在关键业务点设置验证码。）

1. **Referer Check。**HTTP Referer是header的一部分，当浏览器向web服务器发送请求时，一般会带上Referer信息告诉服务器是从哪个页面链接过来的，服务器籍此可以获得一些信息用于处理。可以通过检查请求的来源来防御CSRF攻击。（正常请求的referer具有一定规律，如在提交表单的referer必定是在该页面发起的请求。所以通过检查http包头referer的值是不是这个页面，来判断是不是CSRF攻击。但在某些情况下如从https跳转到http，浏览器处于安全考虑，不会发送referer，服务器就无法进行check了。所以无法完全依赖Referer Check作为防御CSRF的主要手段。但是可以通过Referer Check来监控CSRF攻击的发生。）

(3) **Anti CSRF Token**。目前比较完善的解决方案是加入Anti-CSRF-Token，即发送请求时在HTTP 请求中以参数的形式加入一个**随机产生**的token，并在服务器建立一个拦截器来验证这个token。服务器读取浏览器当前域cookie中这个token值，会进行校验该请求当中的token和cookie当中的token值是否都存在且相等，才认为这是合法的请求。否则认为这次请求是违法的，拒绝该次服务。这种方法相比Referer检查要安全很多，token可以在用户登陆后产生并放于session或cookie中，然后在每次请求时服务器把token从session或cookie中拿出，与本次请求中的token 进行比对。由于token的存在，攻击者无法再构造出一个完整的URL实施CSRF攻击。（但在处理多个页面共存问题时，当某个页面消耗掉token后，其他页面的表单保存的还是被消耗掉的那个token，其他页面的表单提交时会出现token错误。）（使用cookie值加密（MD5），因为攻击伪造网站无法获取被攻击网站的cookie，因此也就无法构造出完整的URL实施CSRF攻击了。）

XSS攻击（跨站脚本攻击）：(1) **输入过滤。**永远不要相信用户的输入，对用户输入的数据做一定的过滤。如输入的数据是否符合预期的格式，比如日期格式，Email格式，电话号码格式等等。这样可以初步对XSS漏洞进行防御。（上面的措施只在web端做了限制，攻击者通过抓包工具如Fiddler还是可以绕过前端输入的限制，修改请求注入攻击脚本。因此，后台服务器需要在接收到用户输入的数据后，对特殊危险字符进行**过滤**或者**转义处理**，然后再存储到数据库中。）

1. **输出编码或转义。**服务器端输出到浏览器的数据，可以使用系统的安全函数来进行编码或转义来防范XSS攻击。在PHP中，有htmlentities()和htmlspecialchars()两个函数可以满足安全要求。相应的JavaScript的编码方式可以使用JavascriptEncode。
2. **安全编码。**开发需尽量避免Web客户端文档重写、重定向或其他敏感操作，同时要避免使用客户端数据，这些操作需尽量在服务器端使用动态页面来实现。(4) **HttpOnly Cookie。**预防XSS攻击窃取用户cookie最有效的防御手段。Web应用程序在设置cookie时，将其属性设为HttpOnly，避免客户端通过JS恶意窃取cookie。

**(5)WAF(Web Application Firewall)。**Web应用防火墙，主要的功能是防范诸如网页木马、XSS以及CSRF等常见的Web漏洞攻击。

## SQL注入攻击：(1) **服务器配置层面的防范**。保证生产环境的Webserver是关闭错误信息的，比如PHP在生产环境的配置文件php.ini中的display\_errors应该设置为Off，这样就关闭了错误提示。（2）**检查变量数据类型和格式。（**可以用黑名单或者白名单验证。白名单验证一般指，检查用户输入是否是符合预期的类型、长度、数值范围或者其他格式标准。黑名单验证是指，若在用户输入中，包含明显的恶意内容则拒绝该条用户请求。在使用白名单验证时，一般会配合黑名单验证）。**只要是有固定格式的变量，在SQL语句执行前，应该严格按照固定格式去检查，确保变量是我们预想的格式**，这样很大程度上可以避免SQL注入攻击。（比如，我们前面接受username参数例子中，我们的产品设计应该是在用户注册的一开始，就有一个用户名的规则，比如5-20个字符，只能由大小写字母、数字以及一些安全的符号组成，不包含特殊字符。此时我们应该有一个check\_username的函数来进行统一的检查。例外情况通过其它来检查）。

（3）**使用预编译语句。**使用预编译的SQL语句语义不会发生改变，并且即使我们使用sql语句伪造成参数，到了服务端的时候，这个伪造sql语句的参数也只是简单的字符，并不能起到攻击的作用。绑定变量使用预编译语句是预防SQL注入的最佳方式。(4) **数据转义**。设置php.ini选项magic\_quotes\_gpc=on，它会将提交的变量中所有的’(单引号)，”(双引号)，\(反斜杠)，空白字符等都在前面自动加上\。或者采用mysql\_real\_escape()函数或addslashes()函数进行输入参数的转义。（4）数据库中密码不应明文存储，以防被’拖库‘(数据库泄露')。在Web开发中，传统的加解密大致可以分为三种:

1、对称加密：即加密方和解密方都使用相同的加密算法和密钥，这种方案的密钥的保存非常关键，因为算法是公开的，而密钥是保密的，一旦密钥被泄露，黑客仍然可以轻易解密。常见的对称加密算法有：AES、DES等。

2、非对称加密：即使用不同的密钥来进行加解密，密钥被分为公钥和私钥，用私钥加密的数据必须使用公钥来解密，同样用公钥加密的数据必须用对应的私钥来解密，常见的非对称加密算法有：RSA等。

3、不可逆加密：利用哈希算法使数据加密之后无法解密回原数据，这样的哈希算法常用的有：md5、SHA-1等。

文件上传漏洞：1**、文件上传的目录设置为不可执行。**只要web容器无法解析该目录下面的文件，即使攻击者上传了脚本文件，服务器本身也不会受到影响。

2、**判断文件类型。**在判断文件类型时，可以结合使用MIME Type（指示文件类型的字符串，与文件一起发送，例如：一个图像文件可能被标记为image/png。与文件扩展名有相同目的）、后缀检查等方式。**白名单验证方式，**即检查文件是否符合预期的类型或者标准格式。此外，对于图片的处理，可以**使用压缩函数或者resize函数**，在处理图片的同时破坏图片中可能包含的HTML代码。

3、**使用随机数改写文件名和文件路径。**文件上传如果要执行代码，则需要用户能够访问到这个文件。在某些环境中，用户能上传，但不能访问。如果应用了随机数改写了文件名和路径，将极大地增加攻击的成本。

4、**单独设置文件服务器的域名**。由于浏览器同源策略的关系，一系列客户端攻击将失效，比如上传crossdomain.xml、上传包含Javascript的XSS利用等问题将得到解决。

5、**使用安全设备防御。**文件上传攻击的本质就是将恶意文件或者脚本上传到服务器，专业的安全设备防御此类漏洞主要是**通过对漏洞的上传利用行为和恶意文件的上传过程进行检测。**恶意文件千变万化，隐藏手法也不断推陈出新，对普通的系统管理员来说可以**通过部署安全设备来帮助防御。**