SQL注入

通过把SQL命令插入到web表单提交或者页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令

防范SQL注入

1.校验用户输入的参数，限制其长度或转义单引号或双连字符
2.使用参数化SQL或直接使用存储过程进行数据查询存取
3.每个应用使用单独的权限有限的数据库连接，不使用管理器权限连接数据库
4.机密信息不明文存放，加密或hash掉密码和敏感信息

XSS

跨站脚本攻击(Cross Site Scripting)，缩写为XSS。恶意攻击者往Web页面里插入恶意Script代码，当用户浏览该页之时，嵌入其中Web里面的Script代码会被执行，从而达到恶意攻击用户的目的。

原理

攻击者对含有漏洞的服务器发起XSS攻击（注入JS代码）。
诱使受害者打开受到攻击的服务器URL。
受害者在Web浏览器中打开URL，恶意脚本执行。

攻击方式

反射型：发出请求时，XSS代码出现在URL中，作为输入提交到服务器端，服务器端解析后响应，XSS随响应内容一起返回给浏览器，最后浏览器解析执行XSS代码，这个过程就像一次发射，所以叫反射型XSS。

存储型：存储型XSS和反射型的XSS差别就在于，存储型的XSS提交的代码会存储在服务器端（数据库，内存，文件系统等），下次请求目标页面时不用再提交XSS代码。

防御措施

编码：对用户输入的HTML实体进行编码

过滤：移除用户上的DOM属性，如onerror等，移除用户上传的style、script、iframe节点

校正：避免直接对HTML实体编码，使用DOM Prase转换，校对不配对的DOM标签

cookie：避免直接在cookie 中泄露用户隐私，例如email、密码等等。

通过使cookie 和系统ip 绑定来降低cookie 泄露后的危险。这样攻击者得到的cookie 没有实际价值，不可能拿来重放。

如果网站不需要再浏览器端对cookie 进行操作，可以在Set-Cookie 末尾加上HttpOnly 来防止javascript 代码直接获取cookie 。

CSRF

CSRF跨站点请求伪造(Cross—Site Request Forgery)

跨站请求攻击，简单地说，是攻击者通过一些技术手段欺骗用户的浏览器去访问一个自己曾经认证过的网站并执行一些操作（如发邮件，发消息，甚至财产操作如转账和购买商品）。由于浏览器曾经认证过，所以被访问的网站会认为是真正的用户操作而去执行。
这利用了web中用户身份验证的一个漏洞：简单的身份验证只能保证请求发自某个用户的浏览器，却不能保证请求本身是用户自愿发出的。

完成一次CSRF攻击，受害者必须依次完成以下两个步骤：
* 登录受信任网站A，并在本地生成Cookie。
* 在不登出A的情况下，访问危险网站B。
看到这里，你也许会问：“如果我不满足以上两个条件中的一个，我就不会受到CSRF攻击”。是滴，确实如此，但是你不能保证以下情况不会发生：
* 你不能保证你登录了一个网站之后，不再打开一个tab页面并访问其它的网站（黄网）。
* 你不能保证你关闭浏览器之后，你本地的Cookie立刻过期，你上次的会话已经结束。
* 上述中所谓的攻击网站，可能就是一个钓鱼网站或者黄色网站。

防御CSRF攻击：
* 通过 referer、token 或者 验证码 来检测用户提交。
* 尽量不要在页面的链接中暴露用户隐私信息。
* 对于用户修改删除等操作最好都使用post 操作 。
* 避免全站通用的cookie，严格设置cookie的域。