web开发安全问题

**主要需要注意以下的一些问题：**

1. sql注入：这个很常规了，不要拼字符串以及过滤关键字都可以防住，需要注意的是，Cookie提交的参数也是可以导致注入[漏洞](https://link.zhihu.com/?target=http%3A//www.2cto.com/" \t "_blank)的。
2. 旁注：就是说在保证自己的程序没问题的同时，也要保证同台服务器的其他站点没问题。至少要设置好系统权限，即使别人的站点出问题也不能影响自己的站点。
3. 上传：尽量不要有上传功能，如果必须有上传功能。也要做到以下方面：不能让用户定义路径、文件名，限制好可上传的文件类型（也可以根据不同文件类型的魔术数字来识别不同类型的文件）。同时要限制好权限，基本规则：执行和可写是互斥权限，不应同时存在。
4. 口令强度：在设置密码之类的功能上应加入密码强度要求。服务器上线部署的时候，应该立即把默认密码修改掉。
5. 防穷举机制：适当的加入验证码，防止别人用程序穷举账户密码。
6. 第三方控件：使用第三方控件，应经过严格的审核（很多第三方控件上作者故意留有缺陷），并且剔除不必要的功能再使用。
7. 权限最小化：能给只读就给只读，尽量具体到每一个子目录。
8. 目录非常规化：得到管理员账户密码，但是找不到后台登录地址也是很难入侵的。后台路径不要动不动就是[http://www.2cto.com](https://link.zhihu.com/?target=http%3A//www.2cto.com" \t "_blank) /admin、manager、gl之类的，很容易猜解。
9. XSS：俗称跨站脚本攻击。用户把HTML、JS之类的标签输入到编辑框，入库之后，再显示的时候可以导致版面错误、JS能解析执行之类的都属于XSS的范畴。如果攻击者插个Iframe连的是个木马网页，那查看这个内容的人就悲剧了。解决方法：过滤大于小于号即可。
10. CSRF URL跳转未验证漏洞：类似于XSS，只是把代码写在URL里，如 http:///[http://www.2cto.com](https://link.zhihu.com/?target=http%3A//www.2cto.com" \t "_blank)/logout.aspx?preURL=aaaa.html 即经常出现在登录退出的页面，通过参数的preURL决定完成动作的时候跳向哪个页面，如果照样 [http://www.2cto.com](https://link.zhihu.com/?target=http%3A//www.2cto.com" \t "_blank) /logout.aspx?preURL=javascript:alert('test') 就可以弹框，说明我们的js代码已经被执行起来了。

**总之一句话，开发过程中，不要相信用户提交的任何数据，规划好目录，做到权限最小化，关闭、删除不必要的东西，就相对会安全很多了。**

**附上cert 安全编码建议：**

1. 验证输入：从不可信任的数据源中进行的输入需要验证。合适的输入验证能减少大量软件的弱点。必须对大部分的数据源持怀疑的态度，包括命令行参数，网络接口，环境变量及用户文件。
2. 留言编译器警告：编译代码时使用编译器的最高警告级别，通过修改代码来减少警告。
3. 针对安全策略的架构和设计：构建软件架构和设计软件时采用安全策略。例如：如果系统在不同的时间需要不同的权限，则考虑将系统分成不同的互相通信的子系统，每个系统拥有合适的权限。
4. 保持简单性：设计越简单越好，复杂的设计提高了实现时错误的可能性。
5. 默认拒绝：默认的访问策略建立在允许的基础上。也就是说，默认的访问是拒绝的，除非标明是允许的。
6. 最小权限原则：每个进程拥有完成工作所需的最小权限。任何权限的拥有时间要尽可能的短。这一方法能阻止攻击者利用权限提升执行任意代码的机会。
7. 清洁发送给其他系统的数据：清洁所有发送给复杂子系统的数据，例如：命令外壳（shells），关系[数据库](https://link.zhihu.com/?target=http%3A//www.2cto.com/database/" \t "_blank)，商用组件。攻击者可能通过SQL命令或者注入进行攻击。这不是靠子系统通过输入验证来避免的问题，因为子系统不清楚调用的上下文，而调用过程指导上下文，所以有责任在调用子系统时清洁数据。
8. 纵深防御：这是一个通用的安全原则，从多个防御策略中规避风险，如果一层防御失效，则另一层防御还在发挥作用。
9. 使用有效的安全质量保证技术：好的质量保证技术能有效的发现和消除弱点。[渗透](https://link.zhihu.com/?target=http%3A//vip.2cto.com/Article/61.html" \t "_blank)测试、Fuzz测试，以及源代码审计都能作为一种有效的质量保证措施。独立的安全审查能够建立更安全的[系统](https://link.zhihu.com/?target=http%3A//www.2cto.com/os/" \t "_blank)。