攻击形式：

1、密码破解：

猜测（社会工程学、密码心理学）、暴力破解

加强密码复杂度，限制失败登陆次数

木马截取（击键记录、屏幕记录）、

客户端和服务器不乱装软件

网络钓鱼

收藏常用网站，注意域名是否正确

网络嗅探器

不再公共场所使用重要应用

2、利用漏洞：

* SQL注入：字符串组合，uncode字符串等

解决办法：格式化变量

Intval(), floatval() //安全

Addslashe() //不安全！！已经不建议使用

mysql\_real\_escape\_string() //需要同时指定客户端和服务器端编码才安全！！

mysql\_real\_escape\_string能够防注入是因为mysql\_escape\_string本身并没办法判断当前的编码，必须同时指定服务端的编码和客户端的编码，加上就能防编码问题的注入。虽然是可以一定程度上防止SQL注入，但还是建议使用拥有Prepared Statement机制的PDO和MYSQLi来代替mysql\_query (注：mysql\_query自 PHP 5.5.0 起已废弃，并在将来会被移除)

* XSS跨站攻击：篡改页面，截取登陆信息，cookie窃取，session挟持欺骗

strip\_tags() //可能损坏丢失数据！！

htmlspecialchars() //设置quotestyle选项为ENT\_QUOTES过滤单引号

htmlspecialchars默认不过滤单引号的，只有设置quotestyle选项为ENT\_QUOTES才会过滤单引号

自建通用函数过滤关键字

* 获取系统敏感文件内容： 根据变量组合文件名 file\_get\_contents() 、fread()

preg\_match(‘/[a-z0-9,%]/’，$input)

* 上传可执行文件

加强上传文件类型校验

限制上传目录权限，如禁止执行或禁止浏览器直接访问

* 绕过破解：逻辑错误，若变量比较，SQL注入绕过

总结：

不可信变量范围：

所有来自客户端的数据都不可信：

$\_SERVER 、 $\_GET 、 $\_POST 、 $\_REQUEST 、 $\_FILES 和 $\_COOKIE

处理：

1、格式化变量：intval() 、floatval() 、strip\_tags() 、trim()

2、校验数据：preg\_match()、strstr()

3、写入数据库前格式化：mysql\_real\_escape\_string()

可信任变量范围：

$\_SESSION： 不完全可信（可能存在session欺骗）

来自数据库的数据：不完全可信,写入数据库前是否进行了有效性和防XSS处理

今天文档在： \vhost\test.cn\php安全