## 登录安全

要保证密码在传输过程中不泄露，必须使用https。计划使用OpenSSL作CA实现Https。

成功登录后，登录之外的请求无需重新登录，也不用Https，用JWT实现用户认证和鉴权。

## 用户敏感数据存储

使用加盐哈希。每个用户注册后生成唯一的随机盐值，先MD5哈希用户密码，将盐值连接在哈希值后，再进行一次MD5，存储最后哈希值。

盐值长度为64byte。

## JWT实现身份认证与鉴权

用户登录后在前端存储JWT（JSON Web Token），其中记录用户身份、权限信息、token过期时间，用户在注册后服务器保存唯一的随机密钥，使用此密钥签名JWT保证不被修改。

十分建议将用户登录盐值与JWT密钥**保存在Redis**以减少每次请求服务器处理的时间。

Token分为**access token**和**refresh token**。前者过期时间短（通常5分钟），普通请求前端带上此token；后者过期时间长（通常2周），在access token过期后前端使用此token请求新的access token。如果refresh token也过期，需要用户重新登录。

这样设计的好处是：

1. Access token如果被嗅探，能够被黑客使用的时间较短。
2. 对黑客来说Refresh token可以替代access token，且可以长时间使用；但前端很久才会请求一次refresh token，所以更难以被嗅探。

虽然这样不保证绝对安全，但仍然提高安全性，减少长过期时间的token被嗅探的可能性。

如果要彻底防止token被嗅探，需要所有请求**全程使用HTTPS**。

浏览器cookie启用HttpOnly和Secure。HttpOnly防止js操作token进行XSS攻击；Secure保证cookie只会通过HTTPS传输。

为了彻底防止重放攻击，还可以考虑为access token 加上访问次数标识，并在服务器保存每个用户当前的访问次数标识。每次请求先验证token中访问次数标识是否与服务器保存的一致，请求完成后更新服务器保存的访问次数标识，并给前端签发含有新访问次数标识的token。这样每个token只能使用一次，也就不存在重放的可能性。虽然服务器同样可以在redis保存访问次数标识，但**每次请求都需生成新签名**带来的时间代价也是值得关注的。