说说API的防重放机制

我们在设计接口的时候，最怕一个接口被用户截取用于重放攻击。重放攻击是什么呢？就是把你的请求原封不动地再发送一次，两次...n次，一般正常的请求都会通过验证进入到正常逻辑中，如果这个正常逻辑是插入数据库操作，那么一旦插入数据库的语句写的不好，就有可能出现多条重复的数据。一旦是比较慢的查询操作，就可能导致数据库堵住等情况。

这里就有一种防重放的机制来做请求验证。

timestamp+nonce

我们常用的防止重放的机制是使用timestamp和nonce来做的重放机制。

timestamp用来表示请求的当前时间戳，这个时间戳当然要和服务器时间戳进行校正过的。我们预期正常请求带的timestamp参数会是不同的（预期是正常的人每秒至多只会做一个操作）。每个请求带的时间戳不能和当前时间超过一定规定的时间。比如60s。这样，这个请求即使被截取了，你也只能在60s内进行重放攻击。过期失效。

但是这样也是不够的，还有给攻击者60s的时间。所以我们就需要使用一个nonce，随机数。

nonce是由客户端根据足够随机的情况生成的，比如md5(timestamp+rand(0, 1000)); 它就有一个要求，正常情况下，在短时间内（比如60s）连续生成两个相同nonce的情况几乎为0。

服务端

服务端第一次在接收到这个nonce的时候做下面行为：
1 去redis中查找是否有key为nonce:{nonce}的string
2 如果没有，则创建这个key，把这个key失效的时间和验证timestamp失效的时间一致，比如是60s。
3 如果有，说明这个key在60s内已经被使用了，那么这个请求就可以判断为重放请求。

示例

那么比如，下面这个请求：

http://a.com?uid=123&timestamp=1480556543&nonce=43f34f33&sign=80b886d71449cb33355d017893720666

这个请求中国的uid是我们真正需要传递的有意义的参数

timestamp，nonce，sign都是为了签名和防重放使用。

timestamp是发送接口的时间，nonce是随机串，sign是对uid,timestamp,nonce(对于一些rest风格的api，我建议也把url放入sign签名)。签名的方法可以是md5({秘要}key1=val1&key2=val2&key3=val3...)

服务端接到这个请求：
1 先验证sign签名是否合理，证明请求参数没有被中途篡改
2 再验证timestamp是否过期，证明请求是在最近60s被发出的
3 最后验证nonce是否已经有了，证明这个请求不是60s内的重放请求