# 前言

      1.在app开放接口api的设计中，避免不了的就是安全性问题，因为大多数接口涉及到用户的个人信息以及一些敏感的数据，所以对这些接口需要进行身份的认证，那么这就需要用户提供一些信息，比如用户名密码等，但是为了安全起见让用户暴露的明文密码次数越少越好，我们一般在web项目中，大多数采用保存的session中，然后在存一份到cookie中，来保持用户的回话有效性。但是在app提供的开放接口中，后端服务器在用户登录后如何去验证和维护用户的登陆有效性呢，以下是参考项目中设计的解决方案，其原理和大多数开放接口安全验证一样，如淘宝的开放接口token验证，微信开发平台token验证都是同理。

      2.接口安全等级

       完全开放的接口,没有任何验证

       接口参数加密(签名)

       接口参数加密+时效性验证

       接口参数加密+时效性验证+私钥

       接口参数加密+时效性验证+私钥+Https



**签名设计**  
     **对于敏感的api接口，需使用https协议**

           https是在http超文本传输协议加入SSL层，它在网络间通信是加密的，所以需要加密证书。

           https协议需要ca证书，一般需要交费。  
     **签名的设计**

           原理：用户登录后向服务器提供用户认证信息（如账户和密码），服务器认证完后给客户端返回一个Token令牌，用户再次获取信息时，带上此令牌，如果令牌正取，则返回数据。对于获取Token信息后，访问用户相关接口，客户端请求的url需要带上如下参数：

         时间戳：timestamp

         Token令牌：token

         然后将所有用户请求的参数按照字母排序（包括timestamp，token），然后更具MD5加密（可以加点盐），全部大写，生成sign签名，这就是所说的url签名算法。然后登陆后每次调用用户信息时，带上sign，timestamp，token参数。

例如：原请求https://www.andy.cn/api/user/update/info.shtml?city=北京 （post和get都一样，对所有参数排序加密）

 加上时间戳和token

       https://www.andy.cn/api/user/update/info.shtml?city=北京&timestamp=12445323134&token=wefkfjdskfjewfjkjfdfnc

      然后更具url参数生成sign

      最终的请求如

         https://www.andy.cn/api/user/update/info.shtml?city=北京&timestamp=12445323134&token=wefkfjdskfjewfjkjfdfnc&sign=FDK2434JKJFD334FDF2  
其最终的原理是减小明文的暴露次数；保证数据安全的访问。

具体实现如下：

           1. api请求客户端想服务器端一次发送用用户认证信息（用户名和密码），服务器端请求到改请求后，验证用户信息是否正确。

        如果正确：则返回一个唯一不重复的字符串（一般为UUID），然后在Redis（任意缓存服务器）中维护Token----Uid的用户信息关系，以便其他api对token的校验。

        如果错误：则返回错误码。

            2.服务器设计一个url请求拦截规则

               （1）判断是否包含timestamp，token，sign参数，如果不含有返回错误码。

               （2）判断服务器接到请求的时间和参数中的时间戳是否相差很长一段时间（时间自定义如半个小时），如果超过则说明该url已经过期（如果url被盗，他改变了时间戳，但是会导致sign签名不相等）。

               （3）判断token是否有效，根据请求过来的token，查询redis缓存中的uid，如果获取不到这说明该token已过期。

               （4）根据用户请求的url参数，服务器端按照同样的规则生成sign签名，对比签名看是否相等，相等则放行。（自然url签名也无法100%保证其安全，也可以通过公钥AES对数据和url加密，但这样如果无法确保公钥丢失，所以签名只是很大程度上保证安全）。

                （5）此url拦截只需对获取身份认证的url放行（如登陆url），剩余所有的url都需拦截。  
            3.Token和Uid关系维护

               对于用户登录我们需要创建token--uid的关系，用户退出时需要需删除token--uid的关系。

**接口特点汇总：**  
1、因为是非开放性的，所以所有的接口都是封闭的，只对公司内部的产品有效；  
2、因为是非开放性的，所以OAuth那套协议是行不通的，因为没有中间用户的授权过程；  
3、有点接口需要用户登录才能访问；  
4、有点接口不需要用户登录就可访问；

**PHP Token(令牌)**

针对以上特点，移动端与服务端的通信就需要2把钥匙，即2个token。  
第一个token是针对接口的（api\_token）；  
第二个token是针对用户的（user\_token）；

**先说第一个token（api\_token）**

它的职责是保持接口访问的隐蔽性和有效性，保证接口只能给自家人用，怎么做到？参考思路如下：  
按服务器端和客户端都拥有的共同属性生成一个随机串，客户端生成这个串，服务器也按同样[**算法**](http://lib.csdn.net/base/datastructure)生成一个串，用来校验客户端的串。  
现在的接口基本是mvc模式，URL基本是restful风格，URL大体格式如下：  
http://blog.snsgou.com/模块名/控制器名/方法名?参数名1=参数值1&参数名2=参数值2&参数名3=参数值3  
接口token生成规则参考如下：  
api\_token = md5 ('模块名' + '控制器名' + '方法名' + '2013-12-18' + '加密密钥') = 770fed4ca2aabd20ae9a5dd774711de2  
其中的   
1、 '2013-12-18' 为当天时间，  
2、'加密密钥' 为私有的加密密钥，手机端需要在服务端注册一个“接口使用者”账号后，系统会分配一个账号及密码，数据表设计参考如下：  
字段名 字段类型 注释   
client\_id varchar(20) 客户端ID   
client\_secret varchar(20) 客户端(加密)密钥

### 再说第二个token（user\_token）

它的职责是保护用户的用户名及密码多次提交，以防密码泄露。  
如果接口需要用户登录，其访问流程如下：  
1、用户提交“用户名”和“密码”，实现登录（条件允许，这一步最好走https）；  
2、登录成功后，服务端返回一个 user\_token，生成规则参考如下：  
user\_token = md5('用户的uid' + 'Unix时间戳') = etye0fgkgk4ca2aabd20ae9a5dd77471fgf  
服务端用数据表维护user\_token的状态，表设计如下：  
字段名 字段类型 注释   
user\_id int 用户ID   
user\_token varchar(36) 用户token   
expire\_time int 过期时间（Unix时间戳）   
（注：只列出了核心字段，其它的再扩展吧！！！）  
服务端生成 user\_token 后，返回给客户端（自己存储），客户端每次接口请求时，如果接口需要用户登录才能访问，则需要把 user\_id 与 user\_token 传回给服务端，服务端接受到这2个参数后，需要做以下几步：  
1、检测 api\_token的有效性；  
2、删除过期的 user\_token 表记录；  
3、根据 user\_id，user\_token 获取表记录，如果表记录不存在，直接返回错误，如果记录存在，则进行下一步；  
4、更新 user\_token 的过期时间（延期，保证其有效期内连续操作不掉线）；  
5、返回接口数据；