支付系统设计

——基于OAuth理念

小组成员

葛剑航 关宏朗 韩龙粤 马博轩 张楠 杨振杰

分工	
葛剑航	根据对 Oauth 的理解，使用通讯图描述用户、用户代理、第三方网站、授权服务器、支付服务器（资源服务器）之间的协作过程，并分析潜在安全威胁
杨振杰	
马博轩	
张楠	描写业务步骤，以及关键步骤的 HTTP 协议内容；根据设计，整理并编写 API 接口列表，并选择 3 个服务 API 编写详细说明。 撰写实验报告。
关宏朗	
韩龙粤	

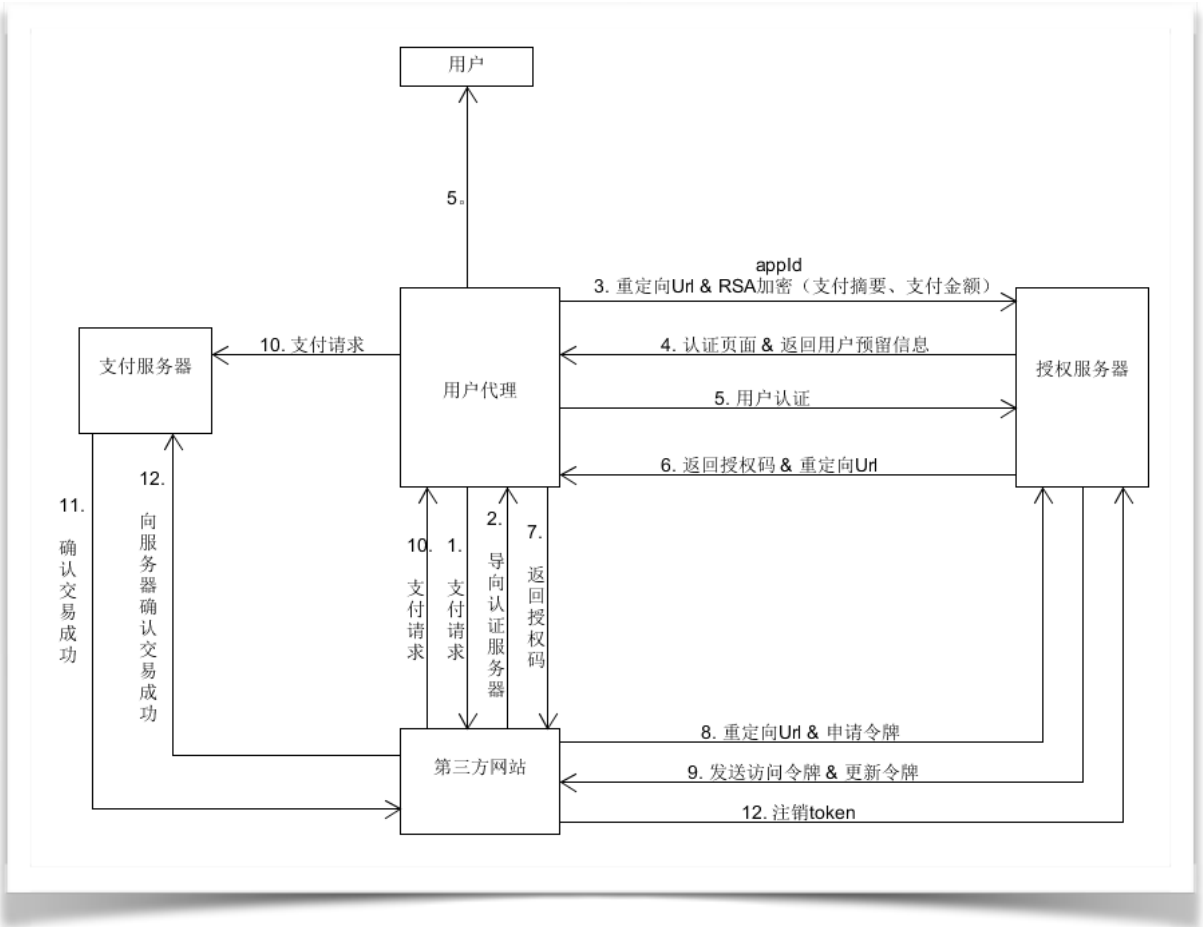
1. 需求描述

中山大学为了鼓励学生创业，计划由校园卡电子钱包服务部门建立校内支付平台，为校内授信的第三方网站和手机应用提供校内在线小额支付服务。为此，学校开放了 LDAP Server 查询接口，支持平台实施教师、学生、校友的统一认证服务，并实现了 OAuth 授权。你的任务是在 OAuth 基础上做扩展业务流程，支持每笔交易用户独立授权。

已完成的工作包括，第三方网站都有自己的 appId 和 appKey ，并注册了你认为必要的信息 （如最大交易额等）。平台需求工程师建议的支付过程是：

1. 用户在第三方网站提交中大支付请求；
2. 网站通过用户代理（浏览器）向授权网站发送支付摘要，支付金额等；
3. 授权网站向用户提供认证页面， 页面需要展示用户在授权中心的预留信息，如学号的后四位防范李鬼网站；
4. 认证完成后，通过用户代理向支付网站提交支付请求；
5. 支付网站完成支付记账后，向第三方网站提交是否成功。

2. UML通讯图



3. 业务步骤

用户通过用户代理，分别与支付服务器、第三方网站、授权服务器进行业务交流（如UML图所示）

步骤如下：

1. 用户通过用户代理向第三方网站发出支付请求
2. 重定向到授权服务器，提供支付摘要，金额等信息
3. 授权服务器返回认证页面，用户通过认证页面进行身份认证
4. 成功认证则返回授权码给用户代理，获得支付权限，重定向回到第三方网站
5. 第三方网站再次向授权服务器申请令牌，直到获得授权服务器的令牌
6. 此时第三方网站则告知用户代理可以支付，用户代理跳转至支付服务器，进行交易
7. 支付服务器向第三方网站确认交易成功，同时获得第三方网站反馈的成功交易信息
8. 交易结束，第三方网站向授权服务器取消令牌
9. 一次支付流程结束

4. 关键步骤HTTP协议

- 客户端申请认证的URI，包含授权类型，客户端id，重定向uri，以及状态

```
GET /authorize?response_type=code&client_id=sysuerid233&state=xyz
        &redirect_uri=https%3A%2F%2Fclient%2Esysu%2Eedu%2Ecn%2Fcb HTTP/1.1
Host: auth_server.sysu.edu.cn
```

- 服务器回应客户端的URI, 包含授权码与状态

```
HTTP/1.1 302 Found
Location: https://client.sysu.edu.cn/cb?code=SplxlOBeZQQYbYS6WxSbIA
        &state=xyz
```

- 客户端向认证服务器申请令牌的HTTP请求，包含授权模式，授权码，uri和客户端id。

```
POST /token HTTP/1.1
Host: server.example.edu.cn
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
        &redirect_uri=https%3A%2F%2Fclient%2sysu%2Eedu%2Ecn%2Fcb
```

- 认证服务器发送的HTTP回复，包含访问令牌，令牌类型，过期时间，更新令牌。

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "token_type":"example",
  "expires_in":3600,
  "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
  "example_parameter":"example_value"
}
```

5. API列表

API作用	API接口	功能补充说明
访问用户基本资料	get_basic_info	获取用户姓名、性别、学号等
查询校园卡余额	get_balance	从服务器端获得当前校园卡余额
查询热水钱包余额	get_water_balance	从服务器端获得当前校园卡的热水钱包余额
支付行为	pay	将消费金额POST到服务器进行后续操作
查询消费历史	get_consume_list	获得消费历史
查询网费信息	get_net_status	查询网费情况，方便缴交网费、续费
获得校园卡头像	get_user_icon	获得图片，可用于设置用户头像
查询图书馆借阅历史	get_libstatus_list	查询图书借阅情况，方便缴交违约金

6. API详解（部分）

① `get_basic_info`

1. 接口说明

获取登录用户在校园卡存储的信息，具体为校园卡持有者姓名、性别、学号。

2. 使用场景

此接口主要用于网站使用校园卡信息登录时，直接拉取用户在校园服务中心登记的校园卡信息（姓名、学号、性别等信息，降低 用户的注册成本。

3. 接口调用说明

3.1 请求说明

url	https://pay.sysu.edu.cn/user/get_basic_info
支持验证方式	oauth2.0
格式	JSON
http请求方式	GET
是否需要鉴权	需要

3.2 输入参数说明

各个参数请进行URL 编码，编码时请遵守 RFC 1738。

（1）通用参数

–OAuth2.0协议必须传入的通用参数

3.3 请求示例

以OAuth2.0协议为例（敏感信息都用*号进行了处理，实际请求中需要替换成真实的值）：

```
https://pay.sysu.edu.cn/user/get_basic_info?
access_token=**************&
oauth_consumer_key=12345&
openid=****************&
format=json
```

3.4 返回参数说明

参数说明	描述
ret	返回码
msg	如果ret<0，会有相应的错误信息提示，返回数据全部用UTF-8编码。
studentName	学生的名字
studentGender	学生的性别
studentId	学生的学号

3.5 返回码说明

0:　　正确返回

其它:　失败。错误码说明详见：公共返回码说明。

3.6 正确返回示例

JSON示例:

```
Content-type: text/html; charset=utf-8
{
"ret":0,
"msg":"",
"studentName":"张山",
"studentGender":"男",
"studentId":"12345678"
}
```

3.7 错误返回示例

```
Content-type: text/html; charset=utf-8
{
"ret":1002,
"msg":"请先登录"
}
```

② get_consume_list

1. 接口说明

获取登录用户的历史交易列表。

1.1 URL

OAuth2.0协议： https://pay.sysu.edu.cn/relation/get_consume_list

1.2 格式

JSON，XML

1.3 HTTP请求方式

GET

1.4 输入参数说明

各个参数请进行URL 编码，编码时请遵守 RFC 1738

（1）通用参数

–OAuth2.0协议必须传入的通用参数。

（2）私有参数

参数名称	是否必须	类型	描述
format		string	定义API返回的数据格式。 取值说明：为xml时表示返回的格式是xml；为json时表示返回的格式是json。 注意：json、xml为小写，否则将不识别。format不传或非xml，则返回json格式数据。
reqnum	必须	string	请求获取的纪录个数。取值范围为1-30。
startdate	必须	string	请求获取交易列表的起始时间。时间格式：xxxx/xx/xx
enddate	必须	string	请求获取交易列表的终止时间。时间格式：xxxx/xx/xx

1.5 请求示例

以OAuth2.0协议为例（敏感信息都用*号进行了处理，实际请求中需要替换成真实的值）：

```
https://pay.sysu.edu.cn/relation/get_consume_list?
access_token=************&
oauth_consumer_key=123456&
openid=***************&
format=xml&
reqnum=30&
startdate=2015/05/15&
enddate=2015/11/11
```

1.6 返回参数说明

参数名称	描述
ret	返回码。
errcode	二级错误码。
msg	如果ret不为0，会有相应的错误信息提示，返回数据全部用UTF-8编码。
data	交易记录的列表信息。
hasnext	表示是否还有交易信息可以拉取。0：还有交易信息可以拉取。1：已拉取完。
info	交易的详细信息列表。
payname	付款方的账户名。
receivename	收款方的账户名。
openid	单条交易记录的唯一ID。
location	交易发生的所在地。
date	交易发生的日期。
timestamp	发生交易的时间。
tag	交易类型标签。
id	标签ID。
name	标签名。

1.7 正确返回示例

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
  <ret>0</ret>
  <errcode>0</errcode>
  <msg>ok</msg>
  <data>
     <timestamp>128679200</timestamp>
     <hasnext>0</hasnext>
     <info>
          <payname>13332333</payname>
  <receivename>canteen_4</receivename>
          <openid>B624064BA065E01CB73F835017FE96FA</openid>

          <location>广东 广州 中山大学 第四饭堂</location>
          <date>2015/11/11</date>
          <timestamp>1285813236</timestamp>
          <tag>
               <id>1</id>
               <name></name>
          </tag>
     </info>
     ............
  </data>
</data>
```

1.8 错误返回示例

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<root>
    <data>
        <ret>1002</ret>
        <msg><![CDATA[请先登录]]></msg>
    </data>
</root>
```

③ get_balance

1. 功能说明

调用本接口，可跳出余额查询弹框，显示用户余额，还可以通过该页面去查询用户的交易历史等。适用于余额查询。

2. 使用场景

此接口主要用于第三方网站获取用户的余额。

3. 接口调用说明

3.1 请求说明

url	https://graph.qq.com/user/get_balance
支持验证方式	oauth2.0
格式	JSON
http请求方式	GET
是否需要鉴权	需要

3.2 输入参数说明

各个参数请进行URL 编码，编码时请遵守 RFC 1738。

（1）通用参数

–OAuth2.0协议必须传入的通用参数

3.3 请求示例

以OAuth2.0协议为例（敏感信息都用*号进行了处理，实际请求中需要替换成真实的值）：

```
https://pay.sysu.edu.cn/user/get_balance?
access_token=************&
oauth_consumer_key=12345&
openid=***************&
format=json
```

3.4 返回参数说明

参数说明	描述
ret	返回码
msg	如果ret<0，会有相应的错误信息提示，返回数据全部用UTF-8编码。
is_lost	判断是否有数据丢失。如果应用不使用cache，不需要关心此参数。 0或者不返回：没有数据丢失，可以缓存。 1：有部分数据丢失或错误，不要缓存。
balance	余额

3.5 返回码说明

0:　　正确返回

其它:　失败。错误码说明详见：公共返回码说明。

3.6 正确返回示例

JSON示例:

```
Content-type: text/html; charset=utf-8
{
"ret":0,
"is_lost":0,
"balance":"1234"
}
```

3.7 错误返回示例

```
Content-type: text/html; charset=utf-8
{
"ret":1002,
"msg":"请先登录"
}
```

7. 安全隐患分析

1. 授权服务器有可能受到钓鱼网站攻击取代
2. 木马可能会篡改重定向Url
3. 没有设置相应的出错回滚机制，当某一服务器崩溃时信息因为不能及时消除而被黑客拦截

8. 心得体会

通过对 OAuth 的学习和运用，我们可以得知，OAuth 可以使中小第三方网站和应用能够得到用户基本信息外的其他信息资料和账户部分使用权限；对大网站平台来说，OAuth 可以完美的解决用户的账户安全和开发者授权的平衡问题。因此 OAuth 协议确定后就获得了包括国外 Twitter、Facebook 和 Google 等认可，之后在国内也得到了有效跟进。

OAuth 不会使第三方网站或应用接触到用户的帐号信息（如用户名与密码），授权后的 http 通信中也不再传输用户信息而是以数字签名和访问令牌（Access Token）取代，即使截到数据包，也无法还原出用户的登录信息。这是 OAuth 最大的优点，也是它得以逐渐成为现在通用的授权标准的原因。

然而对于个人账号来说，安全性是一个最应该被关注的方面。在国内外都有许多滥用 OAuth 的第三方服务的案例。授权的实质相当于系统为第三方网站/应用开了一个后门，用户给予的授权就是允许它们可以走后门进来获取用户的隐私资料和使用权限；所以滥用 OAuth 授权自然会降低账号的安全性。

从开发者的角度来说，在部署 OAuth 的时候就应该考虑好一些相关细节，例如强制使用 https 加密、强制部署 OAuth 2.0、注意在信息回传过程中进行信息防护等。同时，OAuth 的安全性相当一部分需要依靠开发者的高度自律；作为开发者，不去申请不该有的权限是最基本应该做到的事情（例如在大多数情况下只需要只读权限即可）。