目录

开放平台安全支付 1.0

李春进

第一章　安全支付概述

本片文章主要介绍开放平台安全支付的技术架构。

旧有的支付流程缺乏对 app 以及 user 的身份识别，安全性不够；而且整个过程没有使用 https，因此支付数据无法安全传输；再加上回调 app 时的一些参数也不是很合理，束缚了支付版本的升级。

新的安全支付，针对已有的流程做了优化，以后旧有的流程会逐步停止支持，过渡到新流程上。

1.1　前置条件

- 开发者需要申请一个应用，获得人人网发放的 app_id 以及 app_secret 信息，如果已有无需申请。

  应用也可以称之为 app。

  申请地址：http://app.renren.com/developers/createAppNew，需先登录

- 需要开通支付。

  开发者申请应用之后，需要申请开通支付功能。

  申请地址：http://app.renren.com/developers/app/{app_id}/pay

  app 首先提交申请，待平台审核通过之后，即可使用。在申请时，请填写好订单号的获取地址，订单成功后的通知地址，并且设置好密码，以用作双方签名比对。

  测试流程不需要审核通过，申请之后即可使用。

- 用户登录处理，安全支付支持不同的用户登陆方式。

  如果是 oauth2.0 那么统一传递 access_token 参数。

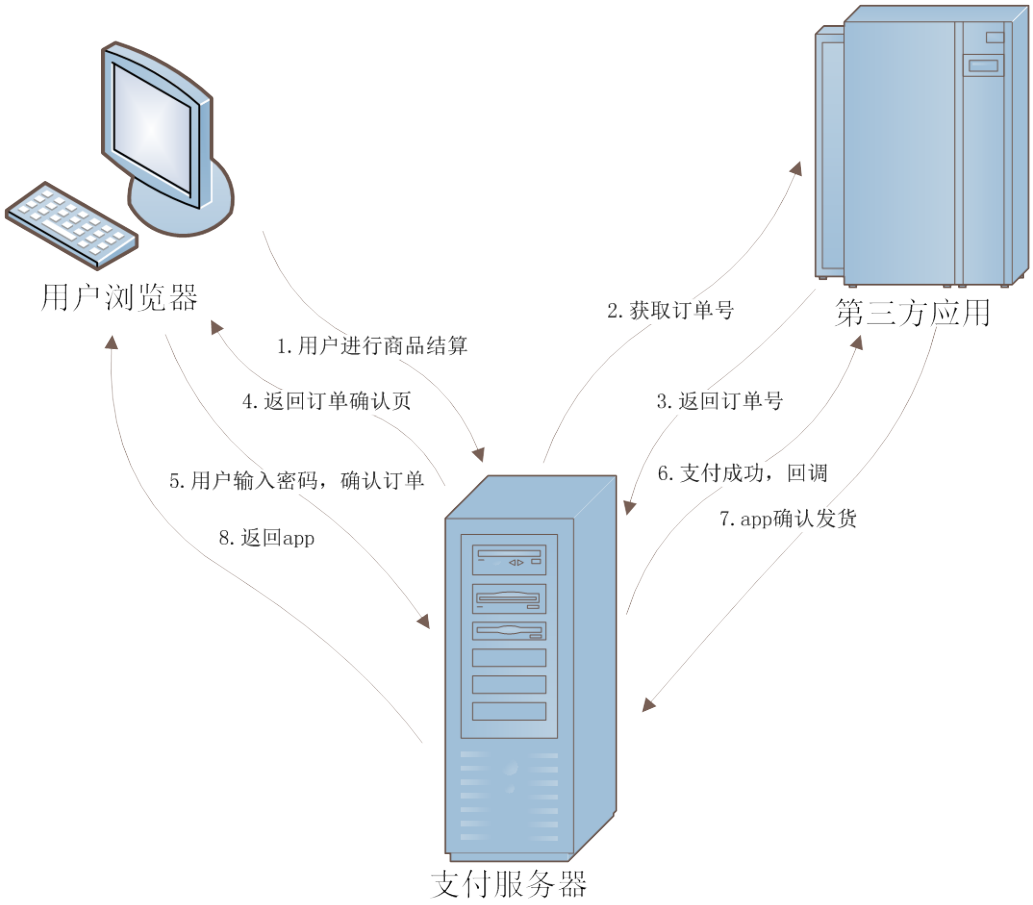
  如果是人人主站登陆，无须传递任何参数，存在登陆 cookie 即可。

- 需要浏览器或客户端支持 https。

1.2 基本流程

1.1.1 流程图

这一部分介绍安全支付的基本流程，我们先来看示意图：



图表 1 安全支付流程图

1.1.2 详细说明

1) 用户进行商品结算

注意，用户必须先登陆。APP 维护用户购物车，用户在 APP 中选择购买商品，然后发起支付请求，app 准备好数据，提交到平台。

订单提交参数示例：	
Action：https://graph.renren.com/spay/android/submitOrder	
Method: POST	
参数：	
amount	2
app_encode	2ffffb9703f29c707845886f98b91159
app_id	105381
descry	人人豆测试 order_number1322115450572
payment	{'game': 'game1' }

submitTime	1322115450572
user_id	391871337

不同的支付，提交的 url 不同，比如 iphone，提交到

https://graph.renren.com/spay/iphone/submitOrder，

用户登陆方式不同，传递的参数也不同，oauth 登陆传递 access_token 参数。

2) 获取订单号：提交的请求会进行校验。

主要检查：

用户是否存在
用户账户状态是否能够消费人人豆，排除用户账户锁定等情况
用户是否授权 app
用户账户余额是否充足
App 是否开通支付
App 是否处于维护期
订单金额是否超过上限，提供 app server 的，最大值 5000；不提供 app server 的最大值 15
App 是否被用户列为黑名单
App 提交参数是否正确，是否包含一些特殊字符串，比如<script>
订单号校验，订单号是 1~2^{64}-1 之间的整数，如果app未提交订单号，则去app server抓取一个

注意：会随着需要调整检查策略。

订单号回调地址是由 app 在申请开通支付功能时注册的一个 url，请保证 get 与 post 都能够回调成功

订单号提交参数示例：
Action：http://www.demo.com/pay/web/generateOrderNo
参数：

xn_sig_user	391871337
xn_sig_app	105381
xn_sig_sandbox	false
xn_sig_model	iphone
xn_sig_version	1.0
xn_sig_skey	34312341324132423141321433244342 3ad31
xn_sig_payment	{'game': 'game1' }

3) app 返回订单号

app 提供了订单号，并不代表订单一定成功，有可能用户会取消订单。

4) 返回订单确认页

5) 用户输入密码，确认订单：这里使用的是用户的支付密码。用户共有三次输入密码的机会，三次失败会锁定账户 10 分钟。

支付密码设置地址：http://safe.renren.com/password/sec/openpf

订单有效时间为 6 分钟，6 分钟之后失效。

6) 支付成功，回调：如果 app 提供了回调地址，则进行回调。如果不提供回调地址，则限定单笔订单额度不超过 10 个豆。

订单回调提交参数示例：
Action：http://www.demo.com/pay/web/callback
参数说明：

xn_sig_result	true
xn_sig_bid	20111024174859000000001
xn_sig_order_id	123456622
xn_sig_user	391871337
xn_sig_app	105381
xn_sig_sandbox	false
xn_sig_model	iphone
xn_sig_version	1.0
xn_sig_skey	34311234123412341234123412334321431

7) App 确认发货

如果回调失败，比如 app 填写的回调地址错误，每个 10 分钟会再次回调，共计回调 160 次，回调策略以后会调整。

8) 返回 app：返回支付结果页，用户返回 app

1.3 直冲流程

在用户账户余额不足时会引导用户去充值，在充值完毕后会直接扣除掉当前订单的金额，app 并不会感知到用户是在直冲扣费，因此无需关注。

1.4 订单修复

App 必须提供订单修复入口，用户登录后可以修复已有的订单。订单修复有两个入口，一个返回 json，app 可以自己定制提示界面，一个返回一个 html 页面。

注意：对于已经成功的订单，平台返回修复成功，因此，app 最好在发起修复流程时就过滤掉这一部分请求。

json:

订单修复提交参数示例：
Action：https://graph.renren.com/spay/android/fixOrderJson
Method: POST
参数：

app_id	105381
order_number	123456

amount	2
fix_time	1322115450572
fix_encode	123412341234122341324312412341324 1

Html:

订单修复提交参数示例：
Action：https://graph.renren.com/spay/android/fixOrder
Method: POST
参数：

app_id	105381
order_number	123456
amount	2
fix_time	1322115450572
fix_encode	123412341234122341324312412341324 1

1.5 测试流程

测试流程与正常流程一致，除了：

不会检查除了黑名单之外的 app 状态
订单成功后只回调 app 一次
订单成功后的回调参数 xn_sig_sandbox 为 true

测试流程的提交 url 比正常流程多了一个 test，比如正常流程的提交 url 为：

https://graph.renren.com/spay/android/submitOrder

那么测试流程的提交 url 为

https://graph.renren.com/spay/android/test/submitOrder

1.6 参数说明

安全支付支持多种版本，每一种版本都支持基本的通用参数。

1.6.1 提交订单

假设是 android 平台，则提交到 https://graph.renren.com/spay/android/submitOrder

参数格式：

参数类型	参数名	参数含义	备注
必须	app_id	接入方	开发者创建 app 时由人人网分配
必须	amount	金额	用户消费金额

必须	submitTime	订单提交时间	自 1970 年 1 月 1 日到现在的毫秒数
必须	app_encode	参数的 md5 签名	由 app_id+order_number+submitTime+secret 进行 MD5 而得，整形的值要转换成字符串进行拼接
可选	order_number	订单号	值的范围[1~2^{64}-1]，app可以提供此参数，如果不提供，则会通过回调app获取一个
可选	redirect_url	返回地址	返回应用时的跳转地址
可选	descr	订单描述	确认订单时用作用户提示
可选	payment	第三方自定义参数	在从第三方抓取订单号时作为参数传递回去，此参数不能超过 500 个字符，不能包含<script>等特殊字符
可选	access_token	oauth 的访问 token	使用 oauth 登陆的必须传递此参数

1.6.2 获取订单号

第三方可以在开通支付时设置订单号抓取地址。

参数格式：

参数类型	参数名	参数含义	备注
必须	xn_sig_user	用户	用户在人人网的唯一标识，用户 id
必须	xn_sig_app	第三方	第三方在创建应用时分配的 app_id

必须	xn_sig_sandbox	沙盒模式	true 表示为沙盒模式，发起消费时并不会扣除用户的人人豆，false 表示正常模式
必须	xn_sig_model	支付模式	字符串，平台为每一种支付方式分配一个唯一标识，比如 android 或者 iphone，以及 web、wap
必须	xn_sig_version	支付版本号	字符串，比如 1.0.0，支付版本，配合 xn_sig_model 唯一标识了支付的入口
必须	xn_sig_time	订单提交时间	自 1970 年 1 月 1 日到现在的毫秒数，平台生成，与提交订单时 app 传递的 submitTime 不一样
必须	xn_sig_skey	参数 md5 签名	第三方申请开通支付时会设置一个用于支付的密码，签名值由预设密码+ xn_sig_sandbox + xn_sig_user+ xn_sig_app+ xn_sig_time 进行 MD5 而成，数值型和 boolean 型先转换成字符串在进行拼接
必须	xn_sig_payment	第三方预设参数	提交订单时传递给平台的 payment 参数

返回一段 json 串：

```
{‘app_res_code’:’ok’, ‘app_res_order_id’:’100000’,’app_res_skey’:’1234123412313324’}
```

app_res_code 标志了订单号获取状态，只有返回 ok 才会认为获取成功，app_res_order_id 是订单号，app_res_skey 是 app 对返回参数的 md5 签名，由 user_id+ app_res_order_id+app_id+支付预设密码拼接而成，数值先转换为字符串。

1.6.3 支付成功回调

回调地址是由第三方在申请开通支付时设置。

参数格式：

参数类型	参数名	参数含义	备注
必须	xn_sig_result	支付结果	true：成功 false：失败
必须	xn_sig_bid	平台生成的业务流水号	字符串
必须	xn_sig_order_id	订单号	由第三方生成
必须	xn_sig_user	用户	用户在人人网的唯一标识，用户 id
必须	xn_sig_app	第三方	第三方在创建应用时分配的 app_id
必须	xn_sig_sandbox	沙盒模式	true 表示为沙盒模式，发起消费时并不会扣除用户的人人豆，false 表示正常模式
必须	xn_sig_model	支付模式	字符串，平台为每一种支付方式分配一个唯一标识，比如 android 或者 iphone，以及 web、wap
必须	xn_sig_version	支付版本号	字符串，比如 1.0.0，支付版本，配合 xn_sig_model 唯一标识了支付的入口
必须	xn_sig_skey	参数 md5 签名	签名值由 xn_sig_sandbox+

			xn_sig_result+ xn_sig_user+ xn_sig_app+ xn_sig_order_id+订单金额+支付预设密码进行 MD5 而成，数值型和 boolean 型先转换成字符串在进行拼接

返回一段 json 串：

{’ app_res_skey’:’12341234asdfsadfsq12341234’ }

app_res_skey 由 user_id+app_id+订单金额+订单号+支付预设密码拼接进行 md5 而成，数值先转换为字符串。

1.6.4 返回应用

返回应用分为两种情况，支付成功和支付失败。

如果提交订单时传递了 redirect_url，则直接向此 url 进行 post 方式的提交。如果没有提供此 url，则由各个支付方式默认处理。

支付成功：

在支付成功后，平台不仅会回调第三方的 server 端，返回到前端的 html 中也包含了支付结果 在 html 中包含一个 payResultEncode 参数，用作签名比对。在用户返回应用，跳转到 redirect_url 时，平台会进行 post 提交。

payResultEncode 由是否沙盒模式+支付是否成功+user_id+app_id+order_no+订单金额+app 支付密码进行 md5 签名而得，boolean 值和数值需要先转换为字符串。

android 和 iphone 支付成功返回地址为：rrpay://success。

支付失败：

android 和 iphone 支付失败返回地址为：rrpay://error

传递参数 code 和 description

1.6.5 修复订单

修复失败返回 json 格式：

{'ret':false,'code':201,'msg':'app 不存在','app_id':105381,'order_number':'1234565', 'sandbox':'false'}

修复成功返回 json 格式：

{'ret':true,'code':417,'msg':'app 不存在', ' payResultEncode': '1234123e41234bc1324a1234','app_id':105381,'order_number':'1234565', 'amount': '2', 'sandbox':'true'}

payResultEncode 与支付成功返回应用一样。

1.7 订单状态码

```
int PAY_PASS = 100;//可以消费,还未启动扣费流程
int PAY_PAYCENTER_SUC= 101;//支付扣费成功,订单还未处理成功
int PAY_SUC = 102;//支付扣费成功,订单成功
int APP_NOT_EXIST = 201;//app不存在
int APP_AUDIT_FAIL = 202;//app未通过审核
int APP_NOT_SUPPORT_PAY = 203;//app未开通支付
int APP_MAINTAIN = 204;//app维护
int APP_PAY_CHECK_FAIL = 205;//app身份认证失败
int APP_SERVER_CONNECT_FAIL = 206;//app服务端连接失败
int APP_SERVER_ORDERNO_FETCH_FAIL = 207;//app服务端回调获取订单号返回格式错误
int APP_SERVER_CALL_SUC = 208;//app服务端回调成功,返回数据正确
int APP_CALL_SERVER_DATA_FAIL = 209;//回调app服务端返回数据格式不正确
int APP_ORDER_MONEY_EXCEED= 210;//超过app单笔订单金额上限
int APP_ORDER_MONEY_ERROR = 211;//订单金额与实际不符合
int APP_CALL_DATA_FORMA_ERROR = 212;//app发送数据格式不正确
int APP_CALL_DATA_FORMA_TOO_LARGE = 213;//app发送数据太多
int APP_ORDERNO_FORMAT_ERROR = 214;//不正确的订单号
int APP_ORDERNO_REPEAT = 215;//重复的订单号
int APP_ORDERNO_NOT_EXIST = 216;//订单号不存在
int APP_ORDER_AMOUNT_ZERO = 217;//订单金额不大于0
int APP_ORDER_DIRECTPAY_CHECK_FAIL = 218;//订单直冲认证失败
int APP_ORDER_BID_NO_COMPAREWITH_ORDER = 219;//传入的订单业务流水号与实际不一致
int USER_NOT_EXIST = 301;//用户不存在
int USER_LOGIN_FAIL = 302;//用户登陆失败
int USER_LOCK = 303;//账户冻结
int USER_SELF_KILLED = 304;//用户自杀
int USER_REQUIRED_ACTIVE = 305;//用户账户未激活
int USER_STATUS_ERROR = 306;//用户账户异常
int USER_UNSAFE_PASSWORD = 307;//账户密码不够安全
int USER_PASSWORD_ERROR = 308;//用户密码错误
int USER_AUTHEN_NUMBER_EXCEED = 309;//用户认证次数过多,暂时锁定账户
int USER_ATTACKED = 310;//用户账户被攻击,小心钓鱼
```

```
int USER_MONEY_NOT_ENOUGH = 311;//用户账户余额不足
int USER_NOT_ADD_APP = 312;//用户未授权app
int USER_NO_AUTHORITY = 313;//用户没有权限
int USER_ORDER_EXPIRED = 314;//订单超时了,过时不候
int USER_ORDER_MONEY_EXCEED= 315;//超过用户设定的订单额上限
int USER_CANCE_PAY = 316;//用户取消订单
int USER_NO_COMPAREWITH_ORDER = 317;//订单所属用户与当前用户不一致
int USER_CHECKCODE_ERROR = 318;//验证码错误
int RENREN_MAINTAIN = 401;//系统维护
int RENREN_API_STOP = 402;//该接口已经停用
int RENREN_DB_ERROR = 403;//存储出现故障
int RENREN_PAYCENTER_ERROR = 404;//支付中心故障
int RENREN_ORDER_DELAY= 405;//系统成功将appbeanorder设置为delay状态
int RENREN_ORDER2DELAY_FAIL= 406;//系统更新订单为delay失败
int RENREN_ORDER2DEAL_FAIL= 407;//系统更新订单为deal失败
int RENREN_ORDERDELAY2DEAL_FAIL= 408;//系统更新订单从delay至deal失败
int RENREN_APPINCOME_UPDATE_FAIL= 409;//系统更新app的收入失败
int RENREN_APPINCOME_UPDATE_SUC= 410;//系统更新app的收入成功
int RENREN_PAY_CENTER_NOTIFYED= 411;//支付中心已经通知,还未付费
int RENREN_PAY_EXCEPTION= 412;//不正确的订单状态
int RENREN_ORDER_NO_PAY= 413;//未扣费的订单状态
int RENREN_PAY_ERROR_ORDER_DATA= 414;//错误的订单数据
int RENREN_PAY_ORDER_ALREAD_SUC= 415;//订单已经成功
int RENREN_PAY_ORDER_FIXING = 416;//订单正在修复
int RENREN_PAY_ORDER_FIXED = 417;//订单修复成功
int SYS_NET_ERROR = 501;//网络异常
int SYS_RESOURCE_LOST = 502;//所需资源缺失
int SYS_ERROR_ACCESS_SOURCE = 503;//错误的访问源
int SYS_PAYCENTER_ERROR = 504;//支付中心异常
int SYS_HTTPS_ERROR = 505;//仅支持https协议
int SYS_SECURITY_ERROR = 506;//因安全原因锁定拒绝用户的消费
```