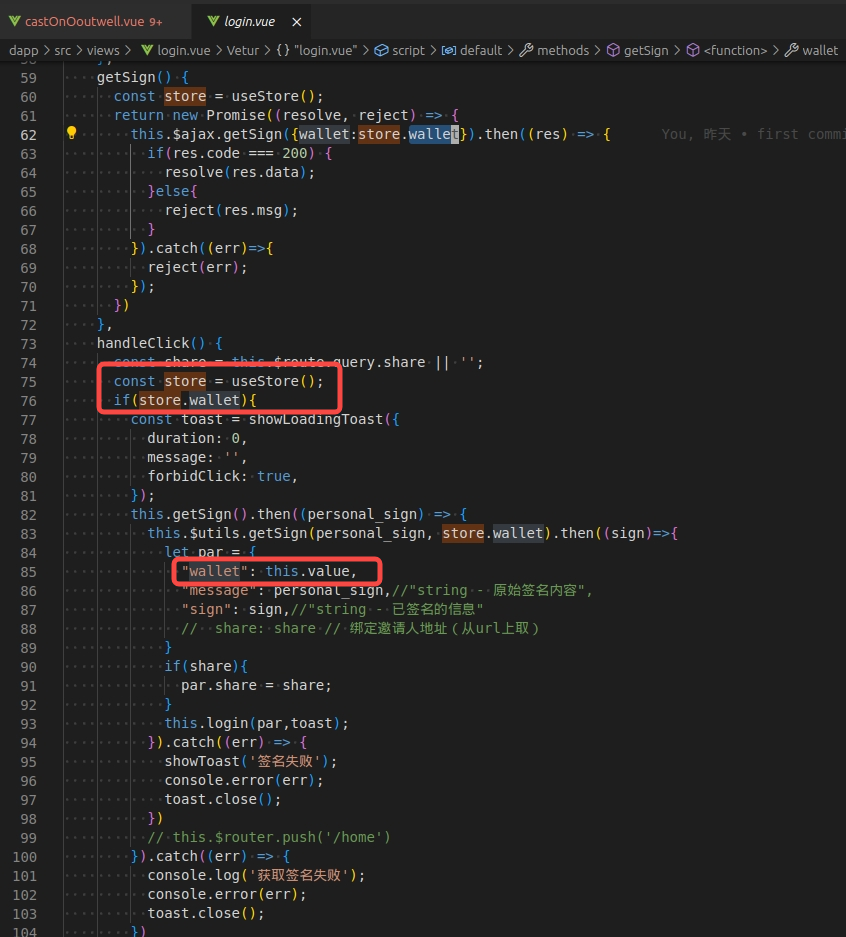
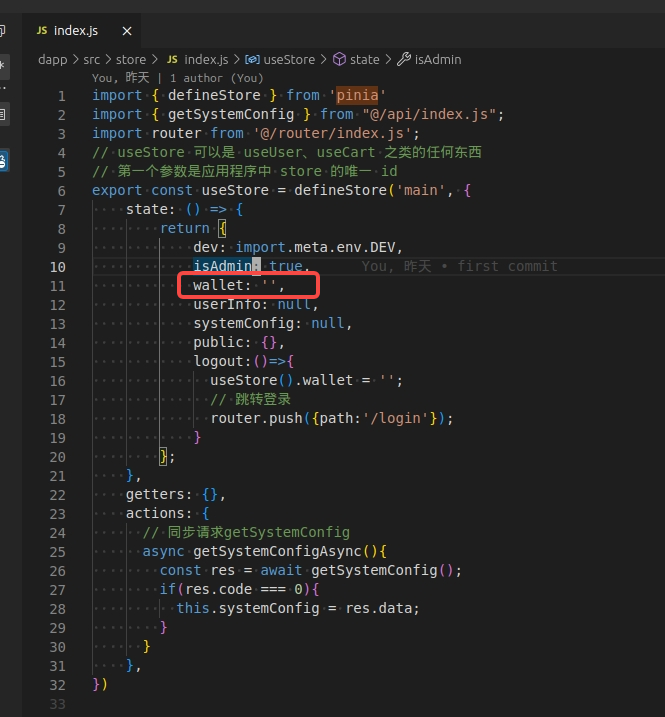
## 问题1： 登录

* 提币的目的地址（收币地址）是从本地浏览器缓存中读取， 存在资金风险

代码路径： dapp/src/views/login.vue



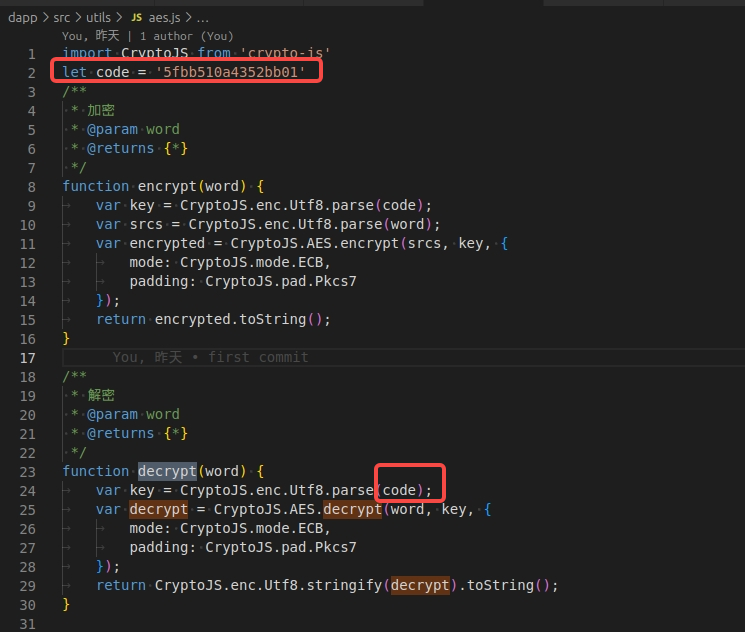


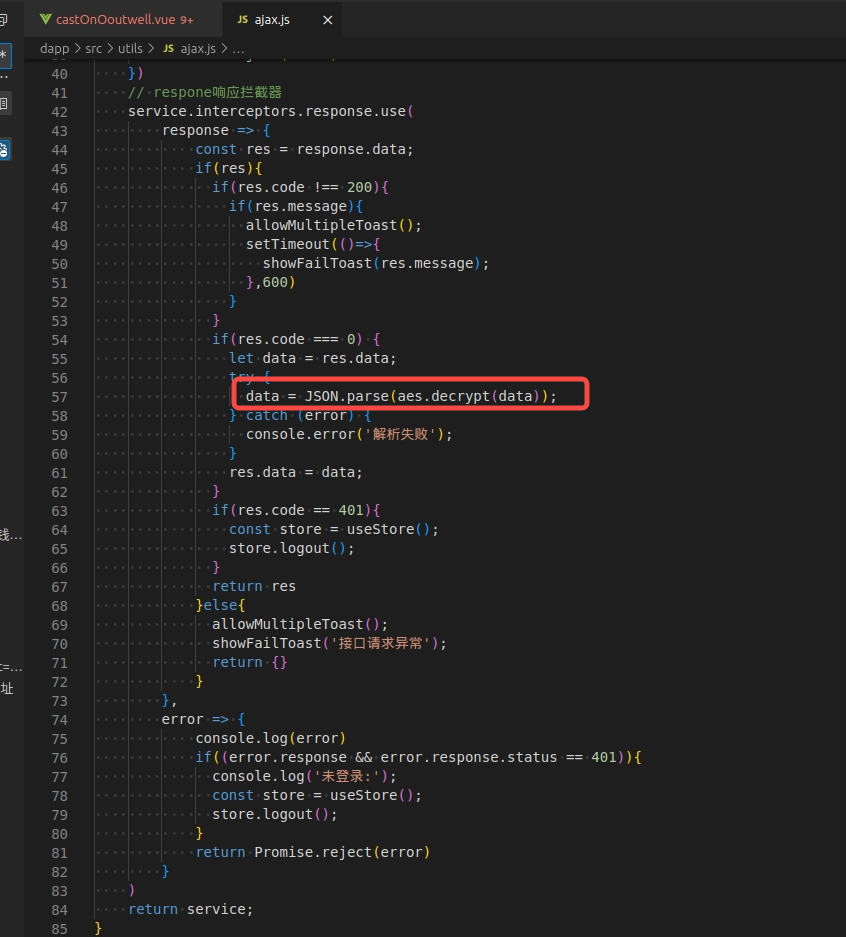
修改建议：

* 实时获取metamask的地址，不能用缓存中的地址
* 需要在前端页面显示用户的登录地址，以便用户进行确认操作
* 后端对签名进行验签时，需要核对进行签名的地址目的地址是否一致

## 问题2： 接口响应的AES解密密钥以明文格式写在代码里面

代码路径：dapp/src/utils/aes.js





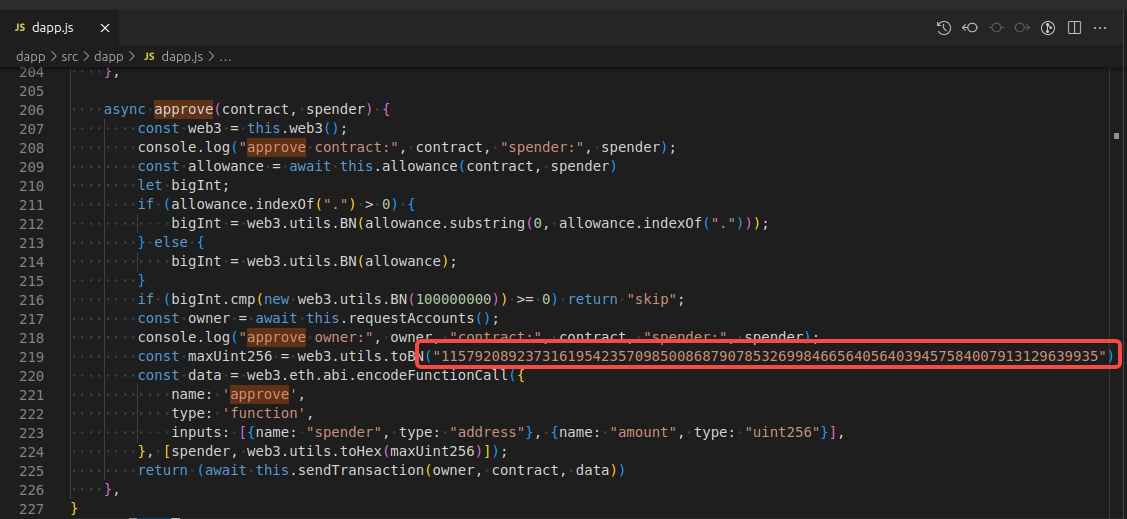
修改建议：

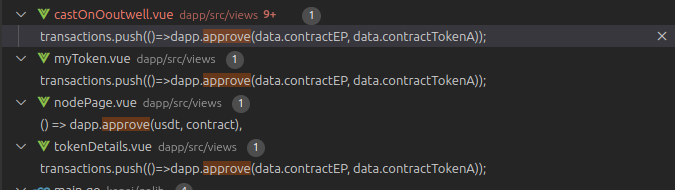
* 前后端应该进行**密钥协商**：使用非对称加解密技术，例如RSA， 前端的密钥（私钥和公钥）应该临时随机生成，并将**公钥**传给后端，后端应该将前端的token和公钥进行一一对应，并设置**一致的**过期时间
* 使用非对称加密解密， 后端使用前端的**公钥**对数据进行加密，前端使用自己的**私钥**进行解密

## 问题3： approve授权函数存在资金风险

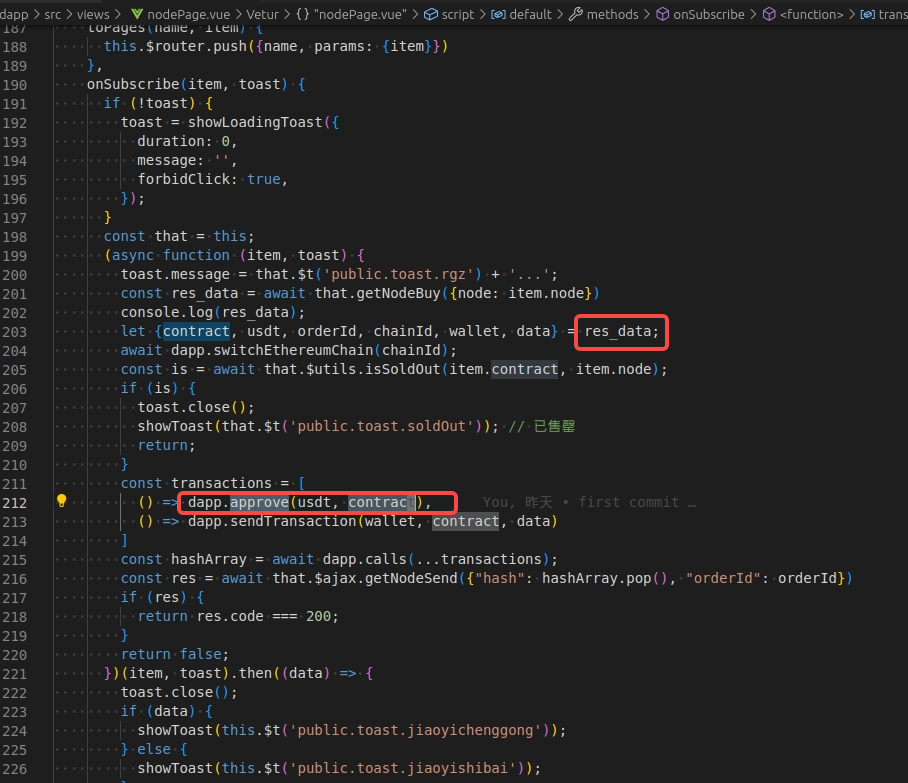
代码路径：dapp/src/dapp/dapp.js

目前是将地址中所有的代币，一次性全部授权spender

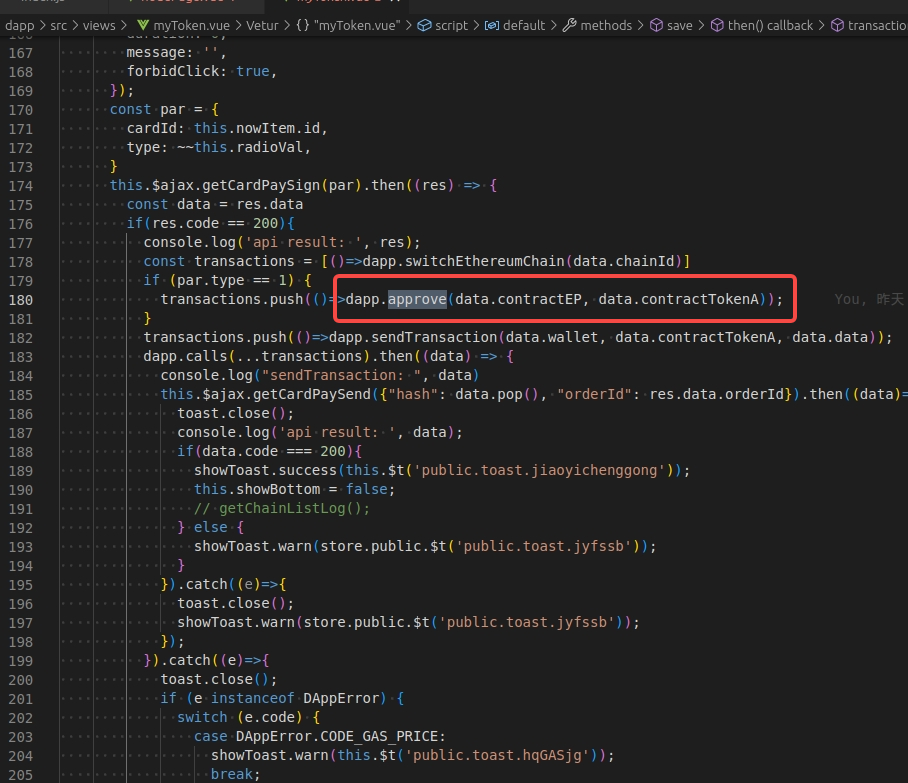




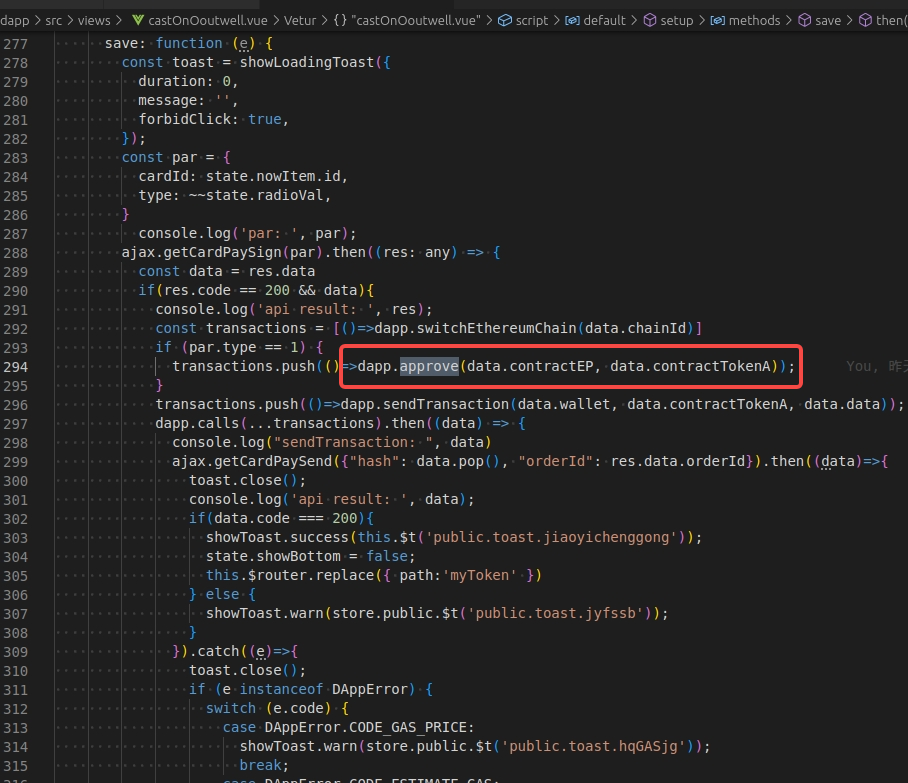
代码路径：dapp/src/views/nodePage.vue



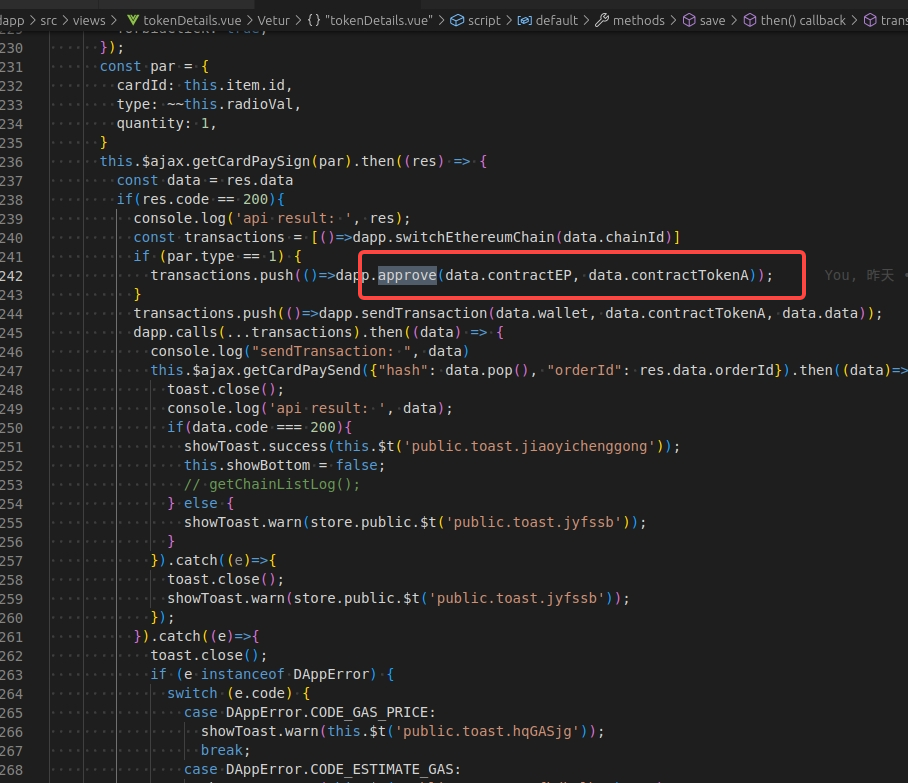
代码路径：dapp/src/views/myToken.vue



代码路径：dapp/src/views/castOnOoutwell.vue



代码路径：dapp/src/views/tokenDetails.vue



修改建议：

根据dapp的实际需要的金额，进行**有限数量**的授权，而不是**无限制数量**的授权，降低用户的**资金风险**