Audit Smart Contract项目文档

AUTHOR: Liu Changshuo，Zhang Yuting，Jin Yibo

Date : 2019-08-07

Contents

Chapter 1 : 创意简介

1.1 项目简介

1.1.1 项目背景

- **项目名称：**智能合约审计平台
- **开发者：**浙江大学INCAS实验室SRTP小组
- **用户：**智能合约撰写用户，审核者
- **相关背景介绍**

  智能合约作为在没有第三方参与的情况下进行可信交易的保障，其安全性、隐私性和逻辑性需要非常缜密的判断证明。现有的形式化验证等自动化证明工具还不是非常成熟，不足以找出智能合约的所有潜在漏洞，无法完全确保智能合约的安全性，所以智能合约的审计工作仍需要人工参与。

  为了确保人工审计的结果安全性、透明性，区块链作为一个去中心化的应用，能够保证人工审计报告的及时公示和不被篡改，通过将审计报告上链和身份确认的形式，确保审计报告的来源可靠性。最终实现一个公开公正透明的智能合约审计平台。

1.1.2 项目概述

智能合约的审计平台是一个基于VNT chain的智能合约提交、审计、报告生成系统。该系统模拟了一个智能合约审计公司的实际场景：

智能合约审计平台用户分为两类：智能合约上传用户，智能合约审计人员。当平台用户在DAPP上提交了智能合约后，抵押一定数量的VNT，系统会自动将智能合约分发给一定数量的在线的智能合约审计人员。公司工作人员（智能合约审计人员）上班后，将自己的状态调整为在线，表示可以接收审计报告并审核，当完成审计后，审计人员获得一定积分（积分和其审核的效果相关），将审计人员的审计报告汇总起来之后，用户将支付抵押的VNT获得审计报告。用户和审计人员在提交智能合约和审计合约之间进行利益交换，实现各自的目标。可以通过审计人员的积分多少对其进行相应的奖惩。

1.1.3 项目创新点

- **提高报告可信度和不可篡改性**

  项目主要的创新点是让审计人员的报告可信度得到提高，现今市场上的各项智能合约审计平台大多是通过专业安全团队咨询审计，报告提交者无法实时具体了解审计过程以及审计提交报告的可信度。而采用审计报告上链的形式，能够用户实时监测审计人员的报告审计进度。

- **去中心化防止审计人员对结果的操作**

  审计人员无法观测到其他审计人员的审计结果，审计人员也无法了解其余审计人员的真实身份，防止审计人员中出现对审计报告结果的操纵情况。

- **VNT激励措施机制**

  用户提交报告进行审计需要支付一定数量的VNT，作为一种激励措施，使审计人员能够积极参与到报告审计过程中。

- **消极审计惩罚机制**

其次，在该项目的智能合约审计平台上，审计人员的审计工作完成度关系到其收益，通过将所有审计人员提交的审计报告进行综合，得到最终的审计报告，与最终审计报告结果相差较大的审计人员将会被认定为消极审计，扣除其收益。这一举措可以提高审计人员的工作积极性和审计质量。

1.1.4 项目应用范围

智能合约审计平台的用户是社会中具有智能合约编写能力的社会成员和具有智能合约开发经验的专业平台审计人员。智能合约提交者具有智能合约安全性审计的需求，审计人员具有专业经验和开发经历，能够满足提交者的需求。

该项目面向广大需要检测并提高其智能合约安全性的用户，能够适用于各种专业研究方向和各种语言的智能合约，同时支持审计人员的审计工作并，运用VNT支付机制作为审计人员的工作量回报。

1.2 项目意义

1.2.1 社会经济意义

- **社会效益**

  满足了社会上部分智能合约书写者对其智能合约安全性的需求，提供了智能合约审计平台来满足实时的审计工作。减少了智能合约安全漏洞对社会经济造成的重大损失，众所周知，智能合约安全漏洞的存在将对企业项目造成毁灭性打击，通过智能合约审计，提前避免合约中可能出现的漏洞并进行修改，规避因合约安全问题导致的财产损失。

- **经济效益**

  智能合约审计平台占用资金少，成本支出少，主要通过虚拟货币VNT进行项目交互，能够解决大量智能合约的安全性问题，能够产生的大量的有用成果。

1.2.2 现有工作基础

- 硬件条件

  具有VNT开发文档和能够进行VNT Dapp开发的计算机设备

- 软件条件

  丰富的WEB开发知识积累和实践经验，对区块链的运行模式和智能合约有充分的了解和实践。

1.3 计划目标

实现一个基于VNT主链的智能合约审计平台。在用户方面，审计平台做到实时上传，并在固定的时间内获得优质的智能合约审计报告，进而降低智能合约的风险。在审计人员方面，审计平台做到实时显示用户上传的智能合约，提醒审计人员在固定的时间完成相应审计并提交。在VNT主链方面，审计平台实现审计人员新的积分值和审计报告hash值的上链。

1.4 推广及应用前景

1.4.1 市场前景

在智能合约漏洞频繁出现的现今，因其造成的损失已经难以估量。也正是如此，审计智能合约的需求缺口越来越大。而我们实现的审计平台恰好满足了这一需求。

此外，除了满足智能合约进行多重审计以保证审计结果质量的需求，该平台还可以满足具有丰富智能合约开发经验的工程师的工作需求，在一定程度上提高了社会的就业率，有助于社会稳定。

更具特色的是，VNT链的不可篡改性使得审计结果无法修改，进而保证了审计结果的质量和智能合约的安全。因此，同时满足了技术需求、社会需求和安全需求的智能合约审计平台会有很大可能在市场上占得一席之地并获得一定的利润。

1.4.2 社会效益分析

减少了智能合约的安全漏洞，避免了智能合约的安全漏洞对社会经济效益造成的影响。同时，召集具有丰富智能合约开发经验的工程师作为审计人员，充分利用了社会剩余劳动力，满足用户对智能合约安全审计的需求。

1.4.3 风险分析

该智能合约审计平台存在着一定风险。如果审计人员在审计报告上做了手脚，将结果全部改变为“通过”，则审计人员仍旧可以获得自己的积分而报告的质量却无法控制。但是，即便设置了相应的机制使得审计人员获得的积分与判断结果相关，其判断结果所占达到一定的比例才会判断有效，审计人员仍旧可以使得所有审计结果为“通过”进而获得积分。这种来自于平台内部的风险是真实存在并且时刻威胁着平台的运营，因此需要设立相关的规定来筛选审计人员或者采用自动的形式化验证来降低风险。

Chapter 2 : 代码部署

2.1 前端

2.1.1 环境配置

- npm和nodejs：安装完成最新的npm和nodejs。 其安装完成后运行结果如图。

```
parallels@parallels-vm:~$ npm -v
6.10.2
parallels@parallels-vm:~$ nodejs -v
v6.17.1
```

- Bootstrap：在terminal内输入bower install bootstrap完成安装bootstrap。

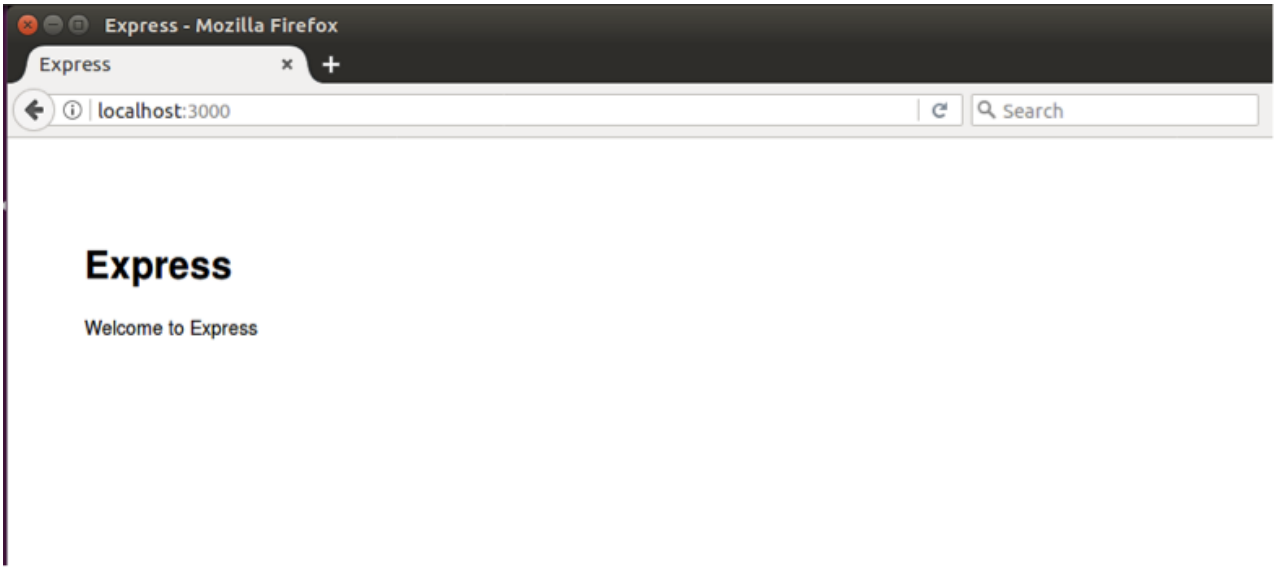
```
parallels@parallels-vm:~$ bower install bootstrap
bower bootstrap#*           not-cached https://github.com/twbs/bootstrap.git#*
bower bootstrap#*              resolve https://github.com/twbs/bootstrap.git#*
bower bootstrap#*             download https://github.com/twbs/bootstrap/archive/v4.3.1.tar.gz
bower bootstrap#*              extract archive.tar.gz
bower bootstrap#*             resolved https://github.com/twbs/bootstrap.git#4.3.1
bower bootstrap#^4.3.1         install bootstrap#4.3.1

bootstrap#4.3.1 bower_components/bootstrap
```

- Go语言：在terminal内输入sudo apt-get install golang-go即可安装Go。

```
parallels@parallels-vm:~$ sudo apt-get install golang-go
[sudo] password for parallels:
Reading package lists... Done
Building dependency tree
Reading state information... Done
golang-go is already the newest version (2:1.6-1ubuntu4).
golang-go set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 564 not upgraded.
```

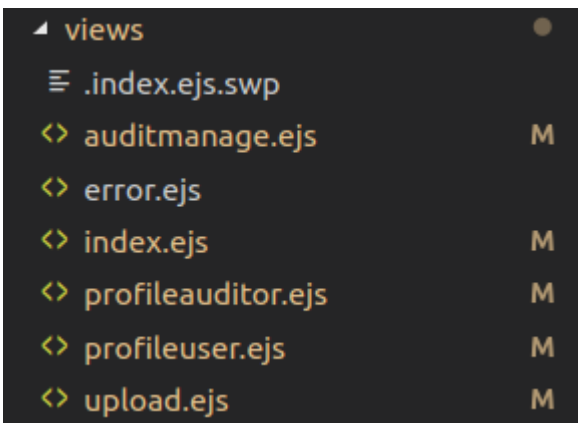
- Express框架：在terminal中输入npm install express -g 安装Express框架以及npm install express-generator -g 安装Express应用生成框架。安装完成后，在根目录下打开terminal，输入express express-demo创建应用。应用创建结束后，切换到express-demo目录下，输入npm install安装依赖。输入npm start启动应用后，在浏览器中访问http://localhost:3000即可。

```
parallels@parallels-vm:~$ cd express-demo/
parallels@parallels-vm:~/express-demo$ npm install
audited 194 packages in 8.661s
found 4 vulnerabilities (3 low, 1 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
parallels@parallels-vm:~/express-demo$ npm start

> express-demo@0.0.0 start /home/parallels/express-demo
> node ./bin/www

GET / 200 1112.415 ms - 170
GET /stylesheets/style.css 304 7.401 ms - -
GET /favicon.ico 404 60.414 ms - 1102
```



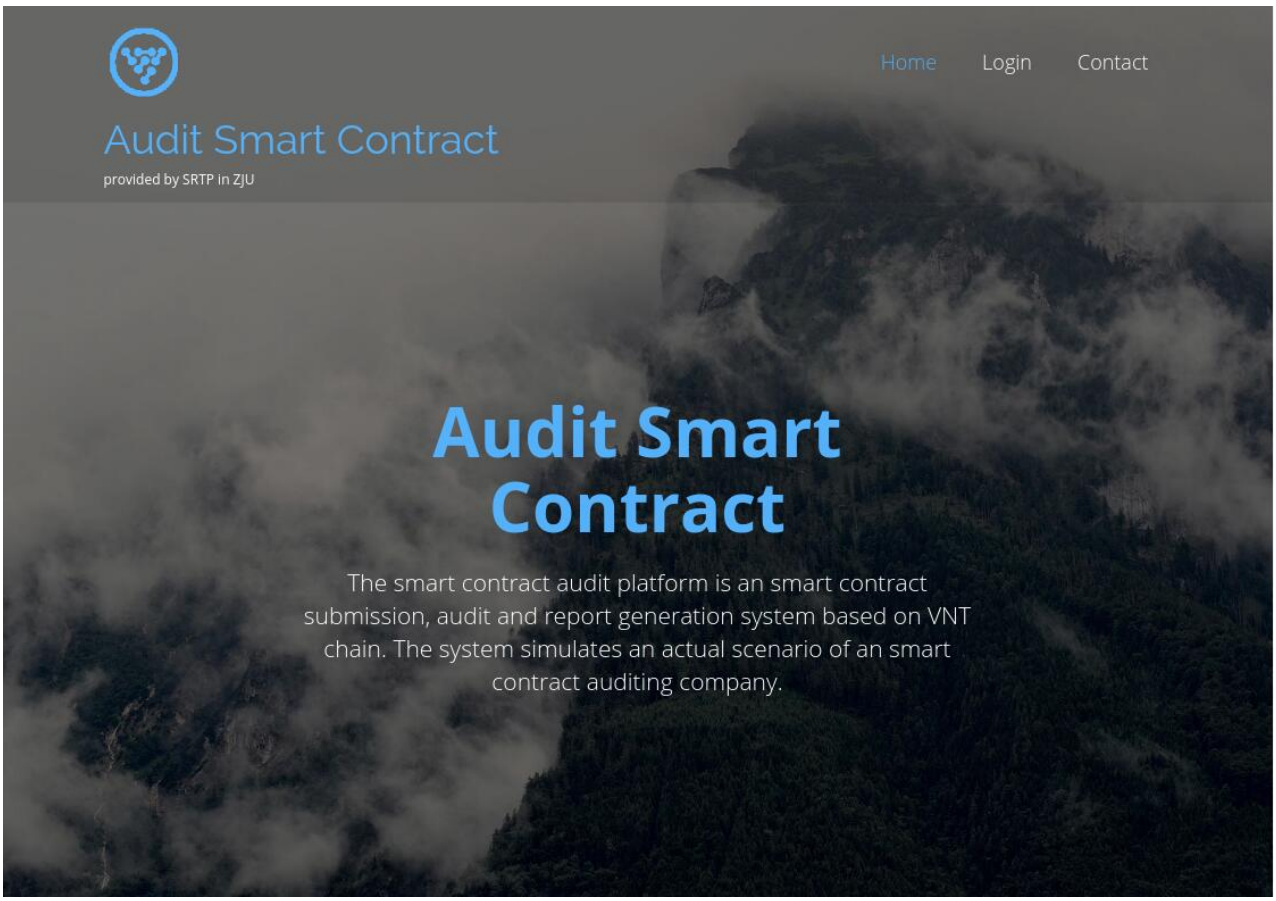
2.1.2 代码部署

前端分为主页面、用户页面和审计人员页面三部分。

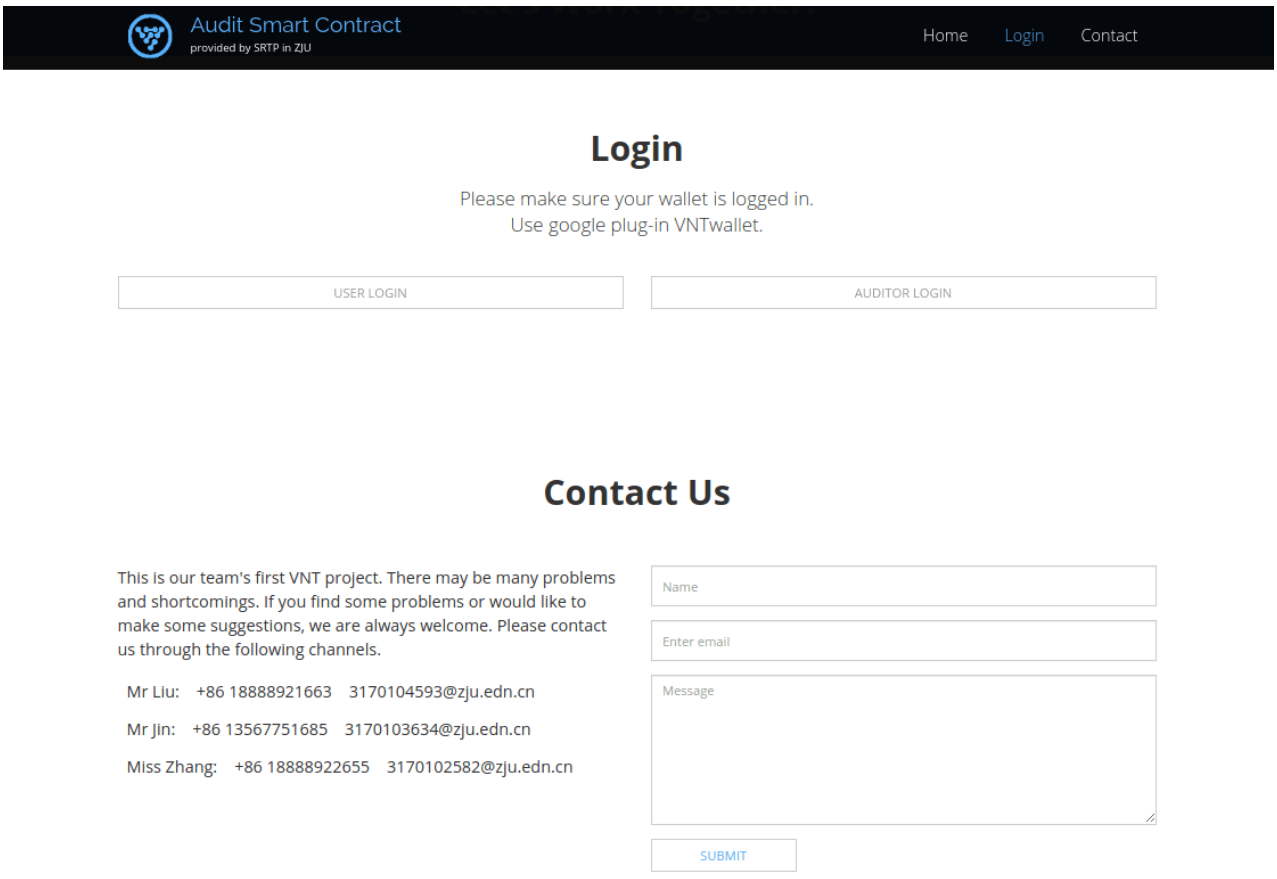


- 主页面(index.ejs)

  （1）主页显示


Audit Smart Contract
provided by SRTP in ZJU
Home
Login
Contact
Audit Smart Contract
The smart contract audit platform is an smart contract submission, audit and report generation system based on VNT chain. The system simulates an actual scenario of an smart contract auditing company.

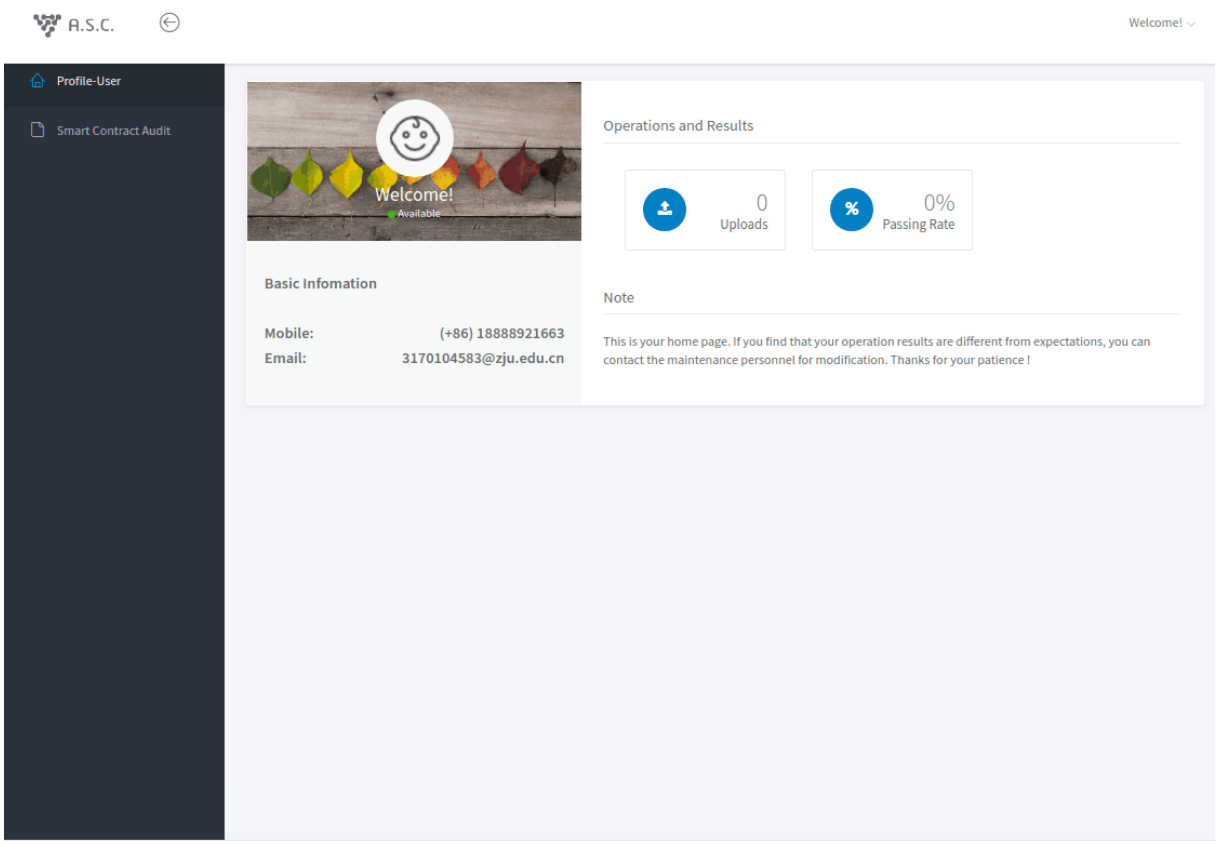
（2） 登录界面


Audit Smart Contract
provided by SRTP in ZJU
Home
Login
Contact
Login
Please make sure your wallet is logged in.
Use google plug-in VNTwallet.
USER LOGIN
AUDITOR LOGIN
Contact Us
This is our team's first VNT project. There may be many problems and shortcomings. If you find some problems or would like to make some suggestions, we are always welcome. Please contact us through the following channels.
Mr Liu: +86 18888921663 3170104593@zju.edn.cn
Mr Jin: +86 13567751685 3170103634@zju.edn.cn
Miss Zhang: +86 18888922655 3170102582@zju.edn.cn
Name
Enter email
Message
SUBMIT

- 用户页面

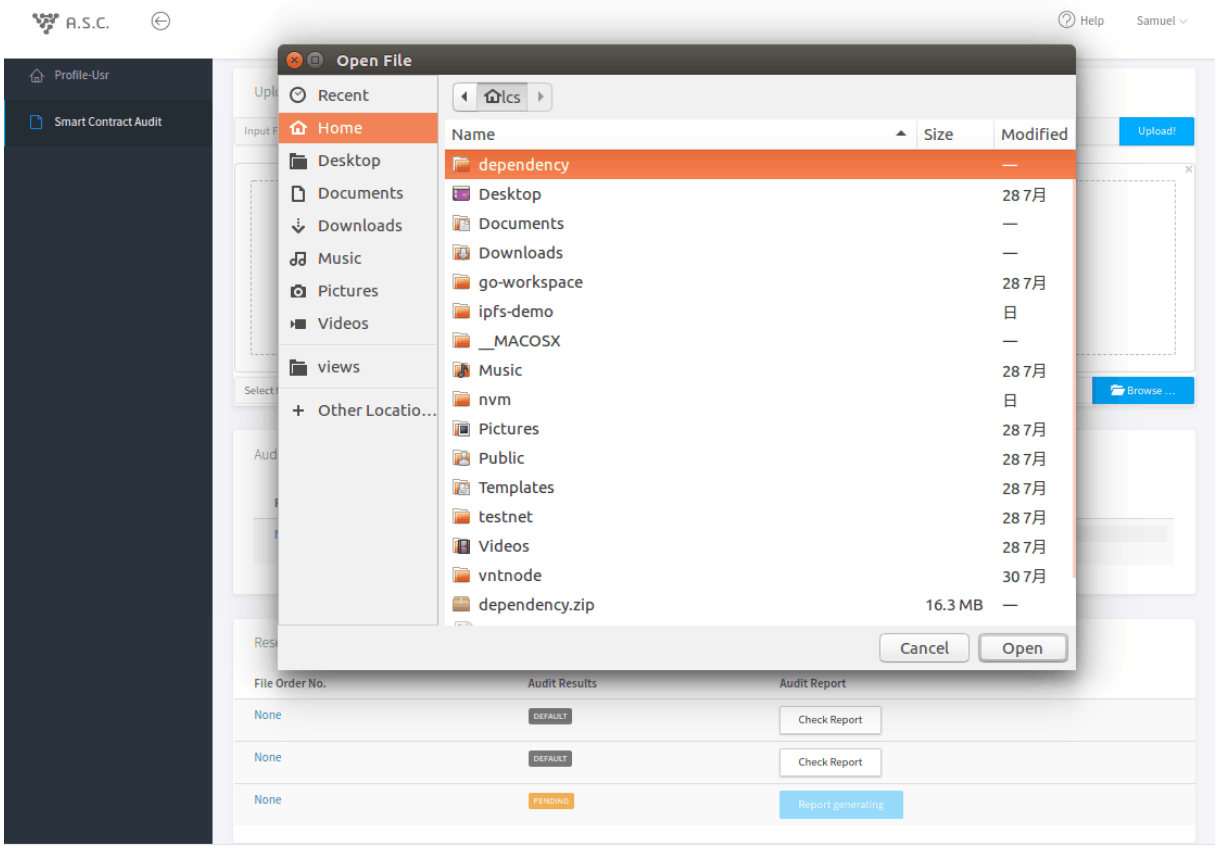
- 用户主页面(profileuser.ejs)

  （1） 个人信息简介

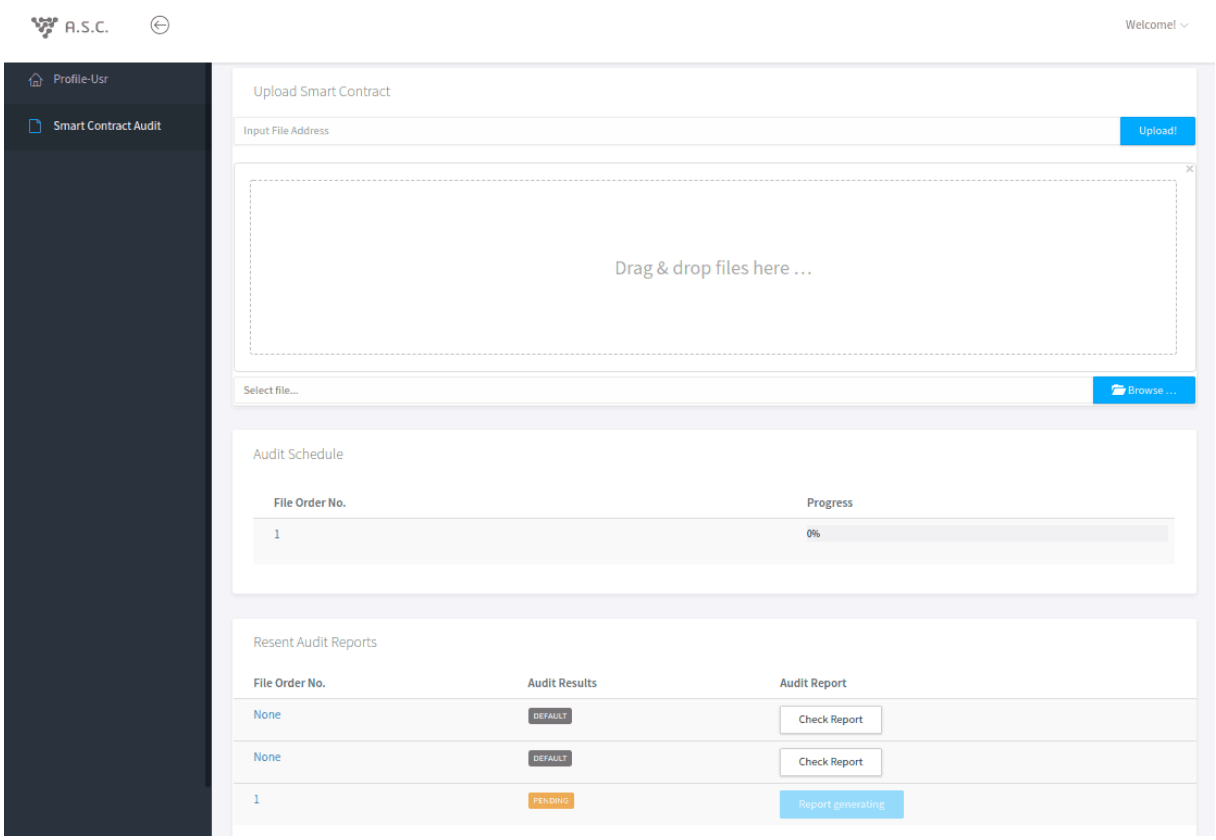


- 用户功能页面(upload.ejs)

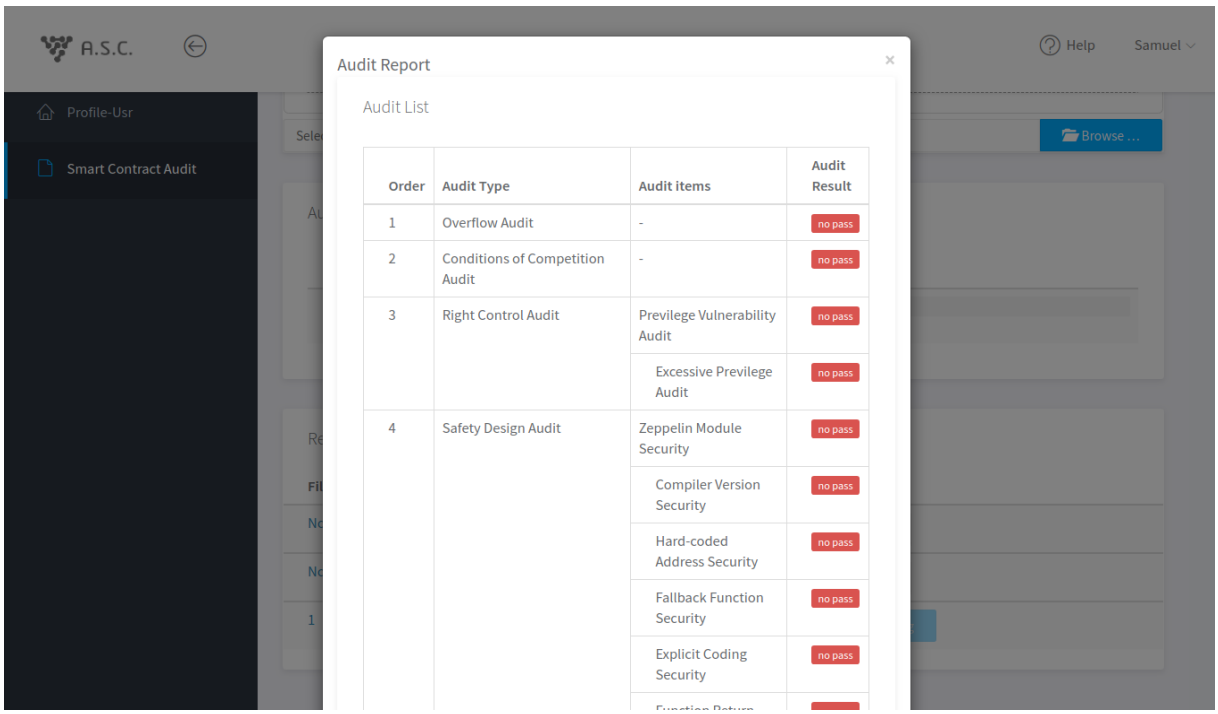
  （1） 智能合约提交



（2） 查看审计进展

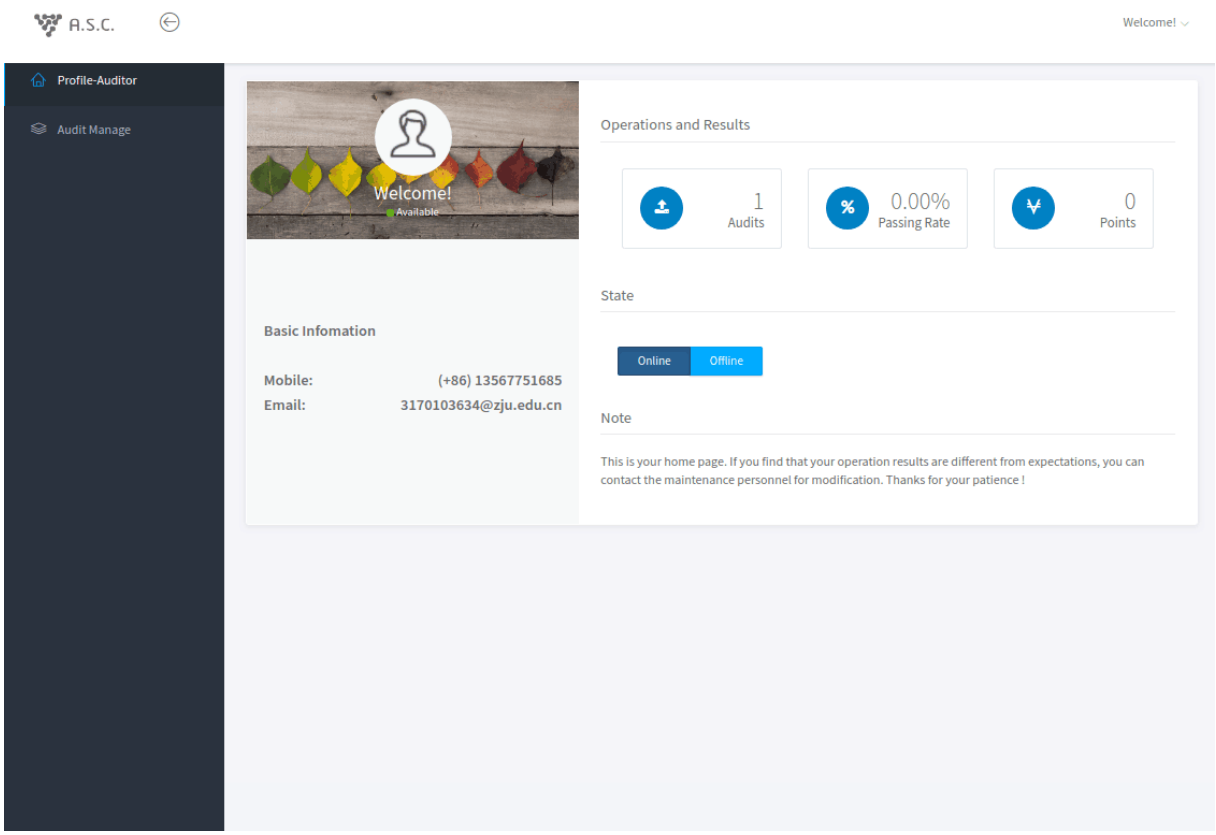


（3） 查看审计结果



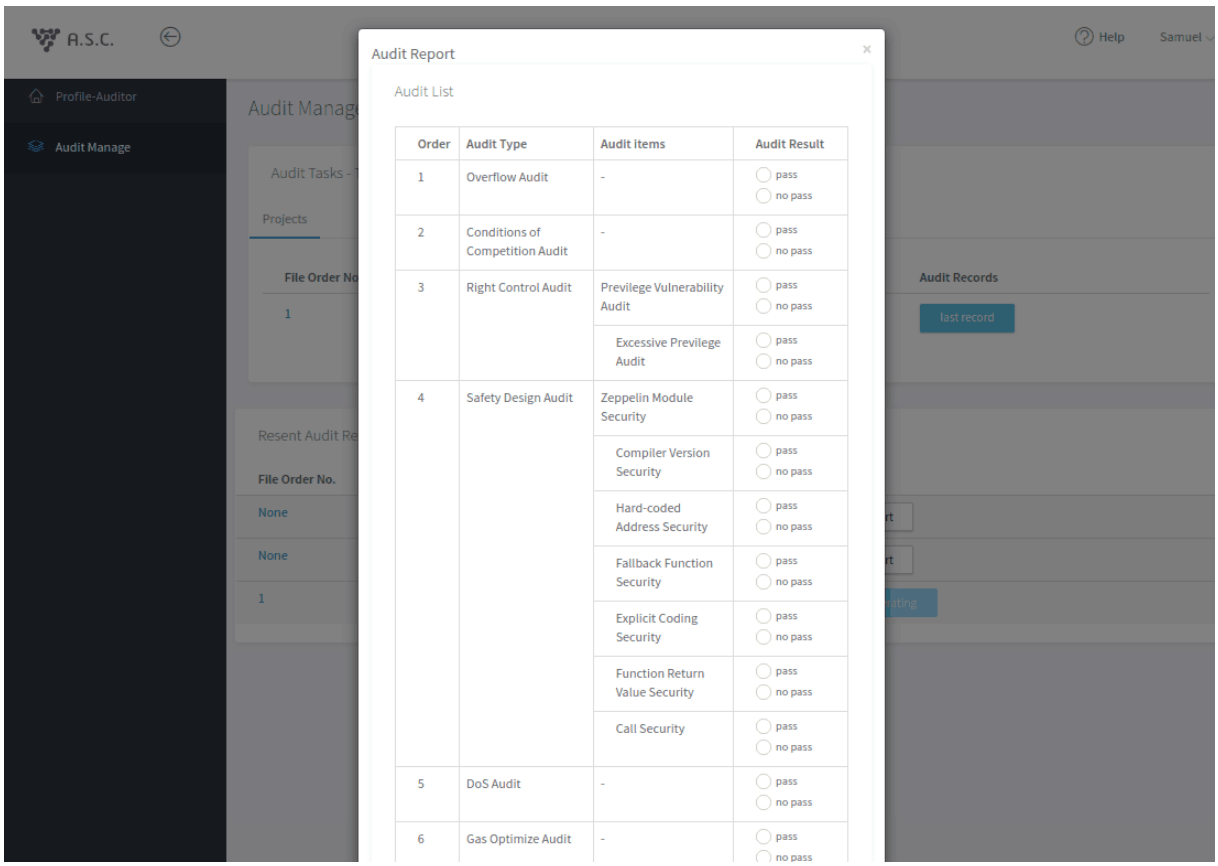
- 审计人员页面
  - 设计人员主页面(profileauditor.ejs)

（1） 个人信息简介

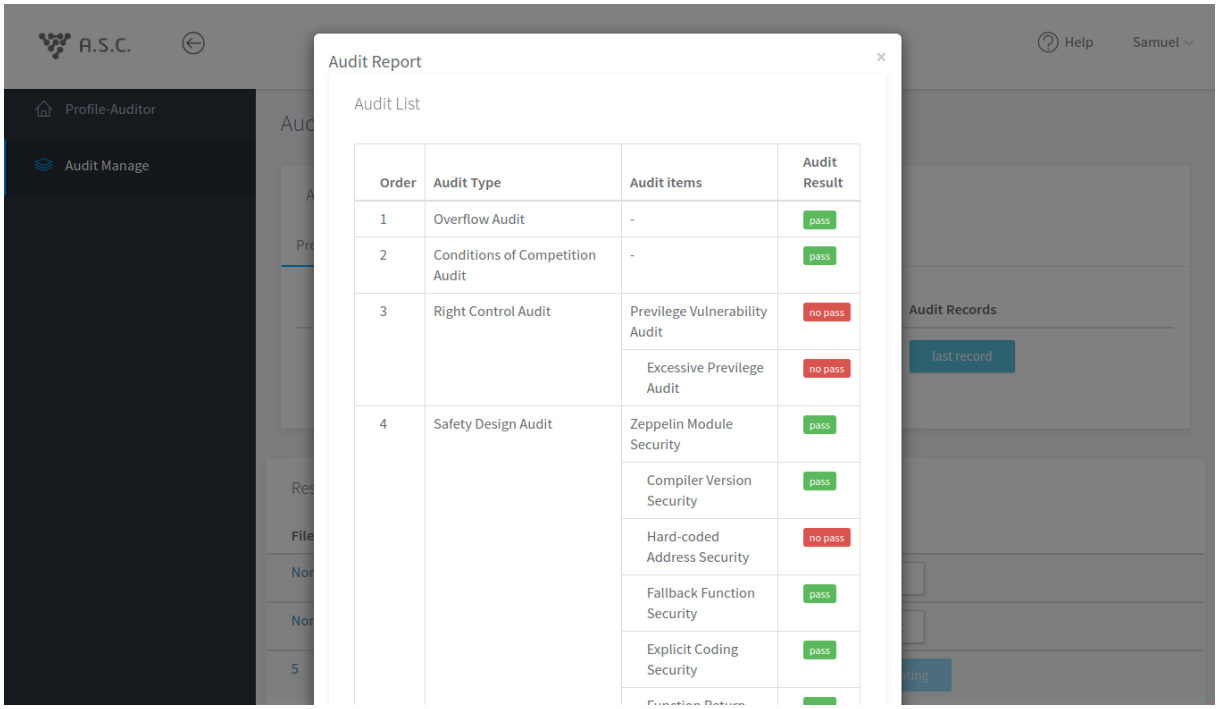


- 审计人员功能页面(auditmanage.ejs)

（1） 智能合约审计



（2） 审计结果查询



2.2 后端

2.2.1 环境配置

- 先安装好nodejs-ipfs文件系统（见http://blog.hubwiz.com/2018/02/03/ipfs-api-dev/）和sql文件系统
- 启动文件：

  根目录文件夹下的/bin/www文件
- bootstrap-fileinput: 在终端输入 sudo npm install -g bootstrap-fileinput安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g bootstrap-fileinput
[sudo] overwhelmed 的密码：
npm WARN bootstrap-fileinput@5.0.4 requires a peer of jquery@>= 1.9.0 but none is installed. You
must install peer dependencies yourself.
npm WARN bootstrap-fileinput@5.0.4 requires a peer of bootstrap@>= 3.0.0 but none is installed. Y
ou must install peer dependencies yourself.
npm WARN bootstrap@4.3.1 requires a peer of popper.js@^1.14.7 but none is installed. You must ins
tall peer dependencies yourself.

+ bootstrap-fileinput@5.0.4
```

- chinese-random-name: 在终端输入 sudo npm install -g chinese-random-name安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g chinese-random-name
[sudo] overwhelmed 的密码：
npm WARN deprecated flatten@1.0.2: I wrote this module a very long time ago; you should use somet
hing else.
+ chinese-random-name@1.0.0
added 3 packages from 3 contributors in 3.465s
```

- cookie-parser: 在终端输入 sudo npm install -g cookie-parser安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g cookie-parser
+ cookie-parser@1.4.4
added 3 packages from 4 contributors in 2.032s
```

- debug: 在终端输入 sudo npm install -g debug安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g debug
[sudo] overwhelmed 的密码：
+ debug@4.1.1
added 2 packages from 3 contributors in 1.66s
```

- ejs: 在终端输入 sudo npm install -g ejs安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g ejs
+ ejs@2.6.2
added 1 package from 2 contributors in 1.669s
```

- ethereumjs-account: 在终端输入 sudo npm install -g ethereumjs-account安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g ethereumjs-account
+ ethereumjs-account@3.0.0
updated 1 package in 4.59s
```

- ethereumjs-tx: 在终端输入 sudo npm install -g ethereumjs-tx安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g ethereumjs-tx
+ ethereumjs-tx@2.1.0
updated 1 package in 4.793s
```

- express: 在终端输入 sudo npm install -g express安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g express
+ express@4.17.1
updated 1 package in 6.319s
```

- http-errors: 在终端输入 sudo npm install -g http-errors安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g http-errors
+ http-errors@1.7.3
added 6 packages from 5 contributors in 2.342s
```

- ipfs-http-client: 在终端输入 sudo npm install -g ipfs-http-client安装, 在安装之前，一定要把nodejs升级到稳定版本的最新版本

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g ipfs-http-client

> secp256k1@3.7.1 install /usr/local/lib/node_modules/ipfs-http-client/node_modules/secp256k1
> npm run rebuild || echo "Secp256k1 bindings compilation fail. Pure JS implementation will be
used."


> secp256k1@3.7.1 rebuild /usr/local/lib/node_modules/ipfs-http-client/node_modules/secp256k1
> node-gyp rebuild
```

- morgan: 在终端输入 sudo npm install -g morgan安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g morgan
[sudo] overwhelmed 的密码：

+ morgan@1.9.1
added 9 packages from 6 contributors in 5.222s
```

- mysql2: 在终端输入 sudo npm install -g mysql2安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g mysql2
+ mysql2@1.6.5
added 13 packages from 19 contributors in 5.405s
```

- vnt: 在终端输入 sudo npm install -g vnt安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g --save https://github.com/vntchain/vnt.js.gi
t
+ vnt@0.20.7
added 6 packages from 9 contributors in 190.601s
```

- vnt-kit: 在终端输入 sudo npm install -g vnt-kit安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g --save https://github.com/vntchain/vnt-kit.j
s.git

> scrypt@6.0.3 preinstall /usr/local/lib/node_modules/vnt-kit/node_modules/scrypt
> node node-scrypt-preinstall.js
```

- bootstrap: 在终端输入 sudo npm install -g bootstrap安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g bootstrap
npm WARN bootstrap@4.3.1 requires a peer of jquery@1.9.1 - 3 but none is installed. You must install peer d
ependencies yourself.
npm WARN bootstrap@4.3.1 requires a peer of popper.js@^1.14.7 but none is installed. You must install peer
dependencies yourself.

+ bootstrap@4.3.1
added 1 package from 2 contributors in 1.442s
```

- popper.js: 在终端输入 sudo npm install -g popper.js安装

```
overwhelmed@overwhelmed-virtual-machine:~$ sudo npm install -g popper.js
+ popper.js@1.15.0
added 1 package from 2 contributors in 3.406s
```

2.2.2 代码部署

- app.js

```
var createError = require('http-errors');
var express = require('express');
var path = require('path');
//var favicon = require('serve-favicon');
var cookieParser = require('cookie-parser');
var logger = require('morgan');
//var bodyParser = require('body-parser');
var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');
//let session = require('express-session');
//let accountRouter = require('./routes/account'); //样例理由
//let encrypt = require('./models/encrypt.js');
var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
app.use(express.static('models'));

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));
```

```
app.use('/', indexRouter);
app.use('/users', usersRouter);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404));
});
app.use("*", function(req, res, next) {
  response.writeHead(200, { "Content-Type": "application/json;charset=utf-8" });
  next();
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
```

- package-json

```
{
  "name": "express-demo",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "bootstrap-fileinput": "^5.0.4",
    "chinese-random-name": "^1.0.0",
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "ejs": "^2.6.1",
    "ethereumjs-account": "^3.0.0",
    "ethereumjs-tx": "^2.1.0",
    "express": "~4.16.1",
    "http-errors": "~1.6.3",
    "ipfs-http-client": "^33.1.1",
    "jade": "~1.11.0",
    "morgan": "~1.9.1",
    "mysql2": "^1.6.5",
    "vnt": "git+https://github.com/vntchain/vnt.js.git",
    "vnt-kit": "git+https://github.com/vntchain/vnt-kit.js.git"
  },
  "devDependencies": {
```

```
    "bootstrap": "^4.0.0-beta",
    "popper.js": "^1.15.0"
  }
}
```

2.3 智能合约

2.3.1 环境配置

- 合约部署的依赖：在terminal输入以下四句命令安装依赖。

  npm install --save https:// github.com/vntchain/vnt.js.git

  npm install --save https:// github.com/vntchain/vnt-kit.js.git

  npm install --save ethereumjs-tx

  npm install --save ethereumjs-account

```
parallels@parallels-vm:~$ npm install --save https://github.com/vntchain/vnt.js.git
+ vnt@0.20.7
added 5 packages from 8 contributors and audited 75 packages in 20.772s
found 0 vulnerabilities

parallels@parallels-vm:~$ npm install --save https://github.com/vntchain/vnt-kit.js.git
+ vnt-kit@1.0.0
updated 1 package and audited 75 packages in 20.096s
found 0 vulnerabilities
```

```
parallels@parallels-vm:~$ npm install --save ethereumjs-tx

> keccak@1.4.0 install /home/parallels/node_modules/keccak
> npm run rebuild || echo "Keccak bindings compilation fail. Pure JS implementation will be used."


> keccak@1.4.0 rebuild /home/parallels/node_modules/keccak
> node-gyp rebuild

make: Entering directory '/home/parallels/node_modules/keccak/build'
  CXX(target) Release/obj.target/keccak/src/addon.o
  CC(target) Release/obj.target/keccak/src/libkeccak/KeccakSponge.o
  CC(target) Release/obj.target/keccak/src/libkeccak/KeccakP-1600-reference.o
  SOLINK_MODULE(target) Release/obj.target/keccak.node
  COPY Release/keccak.node
make: Leaving directory '/home/parallels/node_modules/keccak/build'

> secp256k1@3.7.1 install /home/parallels/node_modules/secp256k1
> npm run rebuild || echo "Secp256k1 bindings compilation fail. Pure JS implementation will be used."


> secp256k1@3.7.1 rebuild /home/parallels/node_modules/secp256k1
> node-gyp rebuild
```

```
parallels@parallels-vm:~$ npm install --save ethereumjs-account
+ ethereumjs-account@3.0.0
added 1 package from 1 contributor and audited 410 packages in 5.948s
found 0 vulnerabilities
```

- Bottle：首先，在terminal中输入git clone https:// github.com/vntchain/bottle将bottle代码clone到相应目录下；然后，输入cd bottle进入到bottle目录下并进行make bottle操作；最后输入~/bottle/build/bin/bottle命令运行bottle。

```
Kings-Mac-2:Desktop Deadpool$ ~/bottle/build/bin/bottle
NAME:
   bottle - the bottle command line interface

   Copyright 2018-2019 The bottle Authors

USAGE:
   bottle [global options] command [command options] [arguments11...]

VERSION:
   0.6.0-beta-9199b7fd
```

2.3.2 代码部署

- 合约编译

  在terminal中输入 bottle compile -code <your contract path> -output <the path of your choosing to save the compiled contract file>对VNT智能合约进行编译。编译结束后会生成相应文件夹，其中包含了.abi文件和.compress文件。abi文件用于和合约的交互，compress文件是压缩后的合约字节码，用于合约的部署。

```
Kings-Mac-2:Desktop Deadpool$ ~/bottle/build/bin/bottle compile -code Update.c -output result
Compile contract: Update
========================
Input file
==========
   > Contract path:          Update.c
Output file
===========
   > Abi path:               result/Update.abi
   > Precompile code path:   result/Update_precompile.c
   > Wasm path:              result/Update.wasm
   > Compress data path:     result/Update.compress
   > Compress hex Data path: result/Update.hex
   > Deploy js path:         result/Update.js
   > Contract json path:     result/Update.json
   > Please use Update.compress when you want to create a contract
Compile finished. 1.082678307s
============================
```

- 合约部署和调用

  (1) 通过require()函数导入vnt.js、vnt-kit.js和ethereumjs-tx库。

```
var Vnt = require("vnt")
var vntKit = require("vnt-kit")
var Cm = require("ethereumjs-common").default
var Tx = require("ethereumjs-tx").Transaction
```

  (2) 使用setProvider()函数链接到VNT链的全节点以创建vnt provider连接。

```
var vnt = new Vnt();
vnt.setProvider(new vnt.providers.HttpProvider("https://hubscan.vnt.link/rpc"));    //
链接到rpc地址
```

  (3) 使用vnt库内的unlockAccount()函数对账户进行解密，并获得其公私钥。

```
var keystore = '{"version":3,"id":"8b3da4b6-1276-4d79-9b4a-
9394996aea44","address":"f0f5acaab770a9443b6c239fc1b3a19061db4098","crypto":
{"ciphertext":"bc76fe4f794e47404e3993aa40e68eaf68cced214b51146042ae8c73e0a85740","ciph
erparams":{"iv":"202fe3ad6b8766141f4d33d5e82fc16d"},"cipher":"aes-128-
ctr","kdf":"scrypt","kdfparams":
{"dklen":32,"salt":"d59507513dbf6999489ee1010e191e78512fe09a1efc7d17f0ceead99df63a13",
"n":262144,"r":8,"p":1},"mac":"d0dc49d77c7ab4e5829521a1d2701df5464178cede24bd374cd6e1e
c33062fbb"}}'
var pass = "lcs170736"
var account = vntKit.account.decrypt(keystore, pass, false);
```

  (4) 从系统文件中读取compress文件和json文件，并对其进行解析获取到abi。

```
var codeFile = '/home/lcs/dependency/Transactions/Transactions.compress'        //定义
代码路径
var abiFile = '/home/lcs/dependency/Transactions/Transactions.abi'          //定义abi路
径
var wasmabi = '[{"name":"Transactions","constant":false,"inputs":[],"outputs":
[],"type":"constructor"},{"name":"$Transfer","constant":false,"inputs":[],"outputs":
[],"type":"function"},{"name":"addGrade","constant":true,"inputs":
[{"name":"data","type":"string","indexed":false}],"outputs":[],"type":"function"},
{"name":"subGrade","constant":true,"inputs":
[{"name":"data","type":"string","indexed":false}],"outputs":[],"type":"function"},
{"name":"setFileHash","constant":true,"inputs":
[{"name":"hash","type":"string","indexed":false}],"outputs":[],"type":"function"},
{"name":"GetTransState","constant":true,"inputs":[],"outputs":
[{"name":"output","type":"bool","indexed":false}],"type":"function"},
{"name":"SetTrans","constant":false,"inputs":[],"outputs":[],"type":"function"},
{"name":"WithdrawToSys","constant":false,"inputs":
[{"name":"amount","type":"uint256","indexed":false}],"outputs":[],"type":"function"},
{"name":"FinishTrans","constant":false,"inputs":[],"outputs":[],"type":"function"},
{"name":"EVENT_SETTRANS","anonymous":false,"inputs":
[{"name":"from","type":"address","indexed":true},
{"name":"price","type":"uint256","indexed":false},
{"name":"time","type":"uint64","indexed":false}],"type":"event"},
{"name":"EVENT_BUY","anonymous":false,"inputs":
[{"name":"sys","type":"address","indexed":true},
{"name":"user","type":"address","indexed":false},
{"name":"price","type":"uint256","indexed":false},
{"name":"time","type":"uint64","indexed":false}],"type":"event"},
{"name":"EVENT_SUCCESS","anonymous":false,"inputs":
[{"name":"sys","type":"address","indexed":true},
{"name":"user","type":"address","indexed":false},
{"name":"price","type":"uint256","indexed":false},
{"name":"time","type":"uint64","indexed":false}],"type":"event"}]'
var abi=JSON.parse(wasmabi.toString("utf-8"));
```

(5) 合约创建

```
a    这是合约创建主函数
function deployWasmContract()
{
// 通过abi与代码路径初始化合约
var contract = vnt.core.contract(abi).codeFile(codeFile)
// 生成合约创建的数据
var data = contract.packContructorData()
// 预估一个gas值
var gas = vnt.core.estimateGas({data: data});
// 获取账户的下一个nonce值
var nonce = vnt.core.getTransactionCount(account.address);
// 生成交易的结构体，指定nonce，gasPirce，gasLimit，value，data等字段
var options = {
nonce: vnt.toHex(nonce),
gasPrice: vnt.toHex(30000000000000),
```

```
gasLimit: vnt.toHex(gas),
value: '0x00',
data: data,
//chainId: 1  //这里必须指定chainId，即你所连接的node的chainId，否则交易签名将出错
}
// 生成交易
var chain=Cm.forCustomChain(1,
{name:'testnet',networkId:2,chainId:2,url:'https://hubscan.vnt.link/rpc'},'petersburg'
);
var tx = new Tx(options,{common: chain});
// 使用之前准备好的私钥，对交易签名
tx.sign(Buffer.from(account.privateKey.substring(2,), "hex"));
// 将交易数据进行序列化
var serializedTx = tx.serialize();
// 发送交易
vnt.core.sendRawTransaction('0x' + serializedTx.toString('hex'),(err,txHash)=>{
if(err)
{
console.log("err happened:",err);
}
})
}
```

(6) 合约调用

- 首先我们必须写出用来调用智能合约函数的固定函数SendData(funct,list,value)，其中参数funct表示智能合约函数，list表示智能合约参数的列表，value表示转账的金额，如果value为0，则是不产生实际交易的情况；如果value不为0，则是产生实际交易的情况。

```
async function SendData(funct,list,value)
{
    let contract = vnt.core.contract(JSON.parse(CommonData.abi));
    let data = contract.packFunctionData(funct,list);
    if(window.vnt!==undefined){
        window.vnt.core.sendTransaction({
            from: window.vnt.core.coinbase,
            to: CommonData.contractAddress,
            gasPrice: 30000000000000,
            gasLimit: 4000000,
            data: data,
            value: vnt.toWei(value)
          },function(err,txHash){
              if (err) {
                console.log("err happednedd: ",err);
                alert("发送失败！请在插件上通过交易。");
              } else {
                var txHash = ${txHash};
                alert("发送成功！交易号: " + txHash + "。");
              }

          })
        return;
    }
```

```
    let userState=JSON.parse(localStorage.getItem("userState"))
    let account
    try{
        account=vntKit.account.decrypt(userState.userName,userState.password, false);
    }
    catch(e){
        console.log("send::SendData:unlock fail")
        alert("发送失败！账户尚未解锁，请解锁后再试。");
        return
    }
    let nonce = vnt.core.getTransactionCount(account.address);
    let options = {
        to: CommonData.contractAddress,
        nonce: vnt.toHex(nonce),
        gasPrice: vnt.toHex(30000000000000),
        gasLimit: vnt.toHex(4000000),
        value: vnt.toWei(value),
        data: data,
        chainId:2
     }
    let chain=Cm.forCustomChain(1,
{name:'testnet',networkId:2,chainId:2,url:CommonData.url},'petersburg');
    let tx = new Tx(options,{common: chain});
    tx.sign(Buffer.from(account.privateKey.substring(2,),"hex"));
    let serializedTx = tx.serialize();
    vnt.core.sendRawTransaction('0x' + serializedTx.toString('hex'),(err,txHash)=>{
        if(err) {
            console.log("err happednedd: ",err);
            alert("发送失败！不定错误，请联系客服处理。");
        }
        else {
            console.info("err:",err,"txHash:",txHash);
            var txHash = ${txHash};
            alert("发送成功！交易号：" + txHash + "。");
        }
    })
}
```

- 在确定调用函数后，需要在js文件中实现该函数的调用。最后需要将这个调用导出文件用来提供接口。

```
const send=(funct,list,value)=>{
    (()=>{
    SendData(funct,list,value)
    }).call()
}
modules.export = send;
```

Chapter 3 : 使用说明

3.1 安装和初始化

根据第二部分代码部署进行环境配置并运行。

3.2 用例

3.2.1 用户用例

表3.2.1.1 用户登录

用例	用户登录
主要参与者	基本用户
目标	登录客户端
前提条件	计算机成功连接网络、安装好vnt钱包插件
触发器	用户点击登录按钮
场景	1.用户点击登录 2.用户得到钱包登录页面 3.用户注册或者使用助记词登录成功 4.得到用户地址 5.判断此用户是普通用户还是审计员 6.登录成功
异常	1.未登录钱包 2.身份认证失败（用户使用审计人登录窗口） 3.登录后页面无反应

表3.2.1.2 用户提交智能合约

用例	用户提交智能合约
主要参与者	基本用户
目标	提交智能合约
前提条件	用户已登录
触发器	用户点击文件上传或拖动文件进入窗口并上传
场景	1.用户已登录 2.用户进入"Smart Contract Audit"面板 3.用户点击"Browser"或拖动文件进入窗口 4.用户点击"Upload"（上传）或"Remove"（删除）并重新选择文件 5.上传成功
异常	1.文件过大 2.超出一个文件 3.正在审计的报告数量超出限制，无法上传

表3.2.1.3 用户查询审计结果

用例	用户查询审计结果
主要参与者	基本用户
目标	查询审计结果
前提条件	用户已登录、审计结果已产生
触发器	用户点击查询按钮
场景	1.用户已登录 2.用户进入"Smart Contract Audit"面板 3.用户点击查询选项"Check Report" 4.系统显示智能合约审计报告结果
异常	1.审计结果未产生

3.2.2 审计人员用例

表3.2.2.1 审计人员登录

用例	审计人员登录
主要参与者	审计人员
目标	登录客户端
前提条件	计算机成功连接网络、安装好vnt钱包插件
触发器	审计人员点击登录按钮
场景	1.审计人员点击登录 2.审计人员得到钱包登录页面 3.审计人员注册或者使用助记词登录成功 4.得到审计人员地址 5.判断此审计人员是普通用户还是审计员 6.登录成功
异常	1.未登录钱包 2.身份认证失败（审计人使用用户登录窗口） 3.登录后页面无反应

表3.2.2.2 审计合约

用例	审计合约
主要参与者	审计人员
目标	审计合约
前提条件	审计人员已登录、已有审计任务派发
触发器	审计人员点击“Audit”按钮
场景	1.审计人员已登录 2.进入"Audit Manage"面板 3.审计人员填入审计结果 4.提交审计结果
异常	1.审计结果未填写完整 2.超出审计时间

表3.2.2.3 查看已提交的审计报告

用例	查看已提交的审计报告
主要参与者	审计人员
目标	显示审计人员自己已经完成的审计报告
前提条件	审计人员收到用户提交的合约并已审计提交
触发器	审计人员点击查看按钮
场景	1.审计人员已登录 2.进入"Audit Manage"面板 3.审计人员点击查看 4.在当前页面上显示已提交的审计报告 5.查看完毕后关闭
异常	1.点击查看无反应 2.在当前页面可以显示弹框，但是没有出现审计结果 3.在当前页面可以显示出审计结果，但是存在错误

表3.2.2.4 设置在线状态

用例	设置在线状态
主要参与者	审计人员
目标	改变审计人员的状态：在线或者离线
前提条件	审计人员登录成功
触发器	审计人员点击状态改变按钮
场景	1.审计人员已登录 2.进入"Audit Manage"面板 3.审计人员点击状态改变按钮 4.online和offline两个按钮的样式发生改变
异常	1.点击查看无反应 2.点击按钮没有效果（按钮样式不发生改变） 3.点击按钮无法改变状态（按钮样式改变但是状态不改变）

表3.2.2.5 查看历史审计记录

用例	查看历史审计记录
主要参与者	审计人员
目标	显示当前审计人员的历史审计报告
前提条件	审计人员登录成功
触发器	审计人员点击Check Report按钮
场景	1.审计人员已登录 2.进入"Audit Manage"面板 3.审计人员点击Check Report按钮 4.当前页面显示出某一历史审计报告
异常	1.点击按钮无反应 2.在当前页面可以显示弹框，但是没有出现审计结果 3.在当前页面可以显示出审计结果，但是存在错误

3.3 求助查询

表3.3.1 求助查询

用例	求助查询
主要参与者	用户/审计人员
目标	求助查询
前提条件	用户/审计人员已登录
触发器	点击右上角"Help"按钮
场景	1.点击右上角"Help"按钮 2.获得帮助文档
异常	1.未登录