CERTIK

Audit Report

Produced by CertiK

for QURAS

Nov 14, 2019

CertiK Audit Report For Quras

Request Date: 2019-10-14
Revision Date: 2019-11-14

Contents

Disclaimer **1**

About CertiK **2**

Executive Summary **3**

Vulnerability Classification **3**

Summary **4**
Vulnerability Details 4

Manual Review Notes **5**
Quras Web 5
Quras API Service 21

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and Quras(the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the "Agreement"). This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, has developed a proprietary Formal Verification technology to apply rigorous and complete mathematical reasoning against code. This process ensures algorithms, protocols, and business functionalities are secured and working as intended across all platforms.

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 6.2B in assets.

For more information: https://certik.org/

Executive Summary

This report has been prepared for Quras to discover issues and vulnerabilities in the source code of their javascript. A comprehensive examination has been performed, utilizing CertiK's Formal Verification Platform, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practice and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line by line manual review of the entire codebase by industry experts.

Vulnerability Classification

CertiK categorizes issues into 3 buckets based on overall risk levels:

Critical

The code implementation does not match the specification, or it could result in the loss of funds for contract owner or users.

Medium

The code implementation does not match the specification under certain conditions, or it could affect the security standard by lost of access control.

Low

The code implementation does not follow best practices, or use suboptimal design patterns, which may lead to security vulnerabilies further down the line.

Summary

Vulnerability Details

Categories Breakdown	**Issues**
Key Management	No issue found
Cryptography	No critical issues found that impacting the current stage, highly recommend for taking consideration improvement for the long-term
Session Management	No issue found
Data Validation	No critical issues found that impacting the current stage, highly recommend for taking consideration improvement for the long-term
Error Handling	No critical issues found that impacting the current stage, highly recommend for taking consideration improvement for the long-term

Manual Review Notes

Quras Web

Quras's mission is to provide a privacy-first blockchain for both users and enterprises. Within Quras's TSdBFT consensus mechanism based blockchain ecosystem, Quras wallet utilizes zk-SNARKs and ring signature for two types of coins - Quras Coin (XQC) and Quras Gas (XQG), secp256r1 standard for public key generation, and off-chain encrypted solution - Offerors for data storage and security.

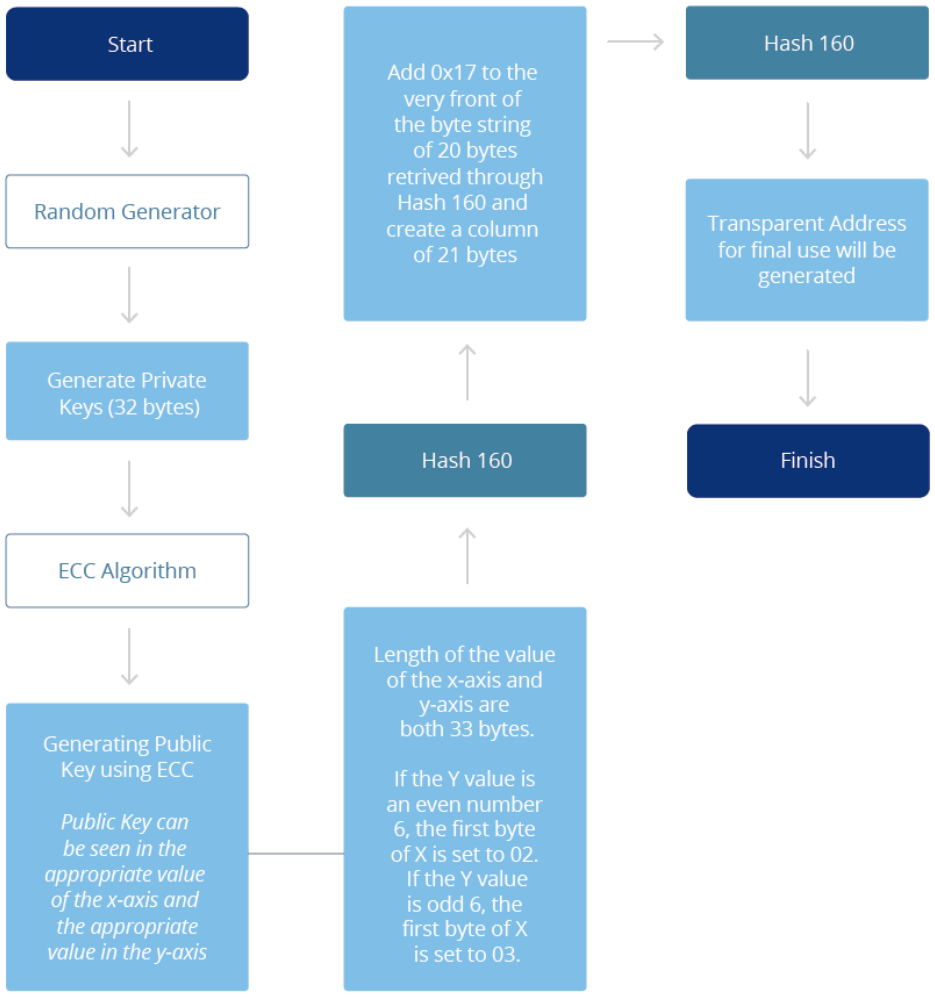
Scope of Audit:

CertiK was chosen by Quras to audit the design and implementation of its wallet application based on Quras chain. To ensure comprehensive protection, the source code has been analyzed by the proprietary CertiK manually reviewed and penetration testing by our engineers experts. That end-to-end process ensures proof of stability as well as a hands-on, engineering-focused process to close potential loopholes and recommend design changes in accordance with the best practices in the space.
The following features are completed during the stage of audit:

- Create New Wallet: keystore generation, private key generation, public key generation
- Unlock A Wallet: keystore file, private key
- Account Balance: view balance, download qrcode key, print account

Source of Truth

- Website
- White-paper - Structure of QURAS Wallet


Start
Random Generator
Generate Private Keys (32 bytes)
ECC Algorithm
Generating Public Key using ECC
Public Key can be seen in the appropriate value of the x-axis and the appropriate value in the y-axis
Length of the value of the x-axis and y-axis are both 33 bytes.
If the Y value is an even number 6, the first byte of X is set to 02. If the Y value is odd 6, the first byte of X is set to 03.
Hash 160
Add 0x17 to the very front of the byte string of 20 bytes retrived through Hash 160 and create a column of 21 bytes
Hash 160
Transparent Address for final use will be generated
Finish

Source Code SHA-256 Checksum

server-php/

- **Constants.php**
  `2467d644784e17f6a803ab4c20b51951d0e61406b8693f3c379da877f009aba1`
- **Localization.php**
  `a814255d01cfcb2c8ac86527c5979560120837753a476a7c9ca3af5960812a4d`
- **OtherRequest.php**
  `0b893a8836749658651c52c9e88c05e5c91d5310c672df75902a6ba02e8e5e21`
- **QurasUtil.php**
  `5ce3e3716b6f6920bbae65b47dbfda3e1251ac6e4b61d5eaacb36c9944d678fc`
- **Request.php**
  `53cfca24cf118a3a79d06f66d4088b836d4c5eb7bfe23e96c758e009afff72ac`
- **SQLite.php**
  `33673e2a24a034893b2f41fcad472fdde89dd8c0b78c1e84f8c3b3f22977cbcb`
- **SafetyModule.php**
  `079a66def33fb158b9b12b6644140d625de84618a3be6eb67463defb7db35980`

- **Utils.php**
  `68266e53a412b3b27ec07598c2818f46f1533f5b1724a96d11d30d7940c74ded`
- **backup.php**
  `6e32bac02c6c0d0a6d4354c83b5f1eb03769742f2209dc33dbc1e95ad4a08b46`
- **index.php**
  `fc106738db9cbe4d06b1bbc7aa221e7563750b03c83cd748a209866cf6878e09`

server-nodejs/

- **localization.js**
  `cdfca0dd0519eb24161cbff6a164ff059f5532cd67ca095ff59ee685fbcd48a6`
- **quras-asset.js**
  `73a95b480e0d2b3365bb6e6b2e9a6ee0719bda753fb361226cfafa243c650e80`
- **quras-module.js**
  `0e0799c4bc6ea5f8c8ce90ee3206290e3639d38a2d2afa379841c269b0acd308`
- **request.js**
  `54fc47ecec3a6d86f39b6827d94192c73261b58508f4fb355705c13d39b42a34`
- **router.js**
  `61a8a98485bfcd5fd17f7917648adc54ffde870953360d31b797c3c51fcb6416`
- **server-constant.js**
  `d3dfe27d8ee3a442574c859e8510abba5cc191c5dd435d8eb1f01310d90fafe0`
- **server-crypto.js**
  `96dba6433fdd2223e945b6bf7b0257215c9800451290b2892aea105f78457085`
- **server-main-http.js**
  `f84dbb9eafc068d2f716d17f99fbd15baa16f7709c8d90028216ffccf461037a`
- **server-main-https.js**
  `0c4f1265da57d38a04d4dc2a03419b4d101613a63a04ee01a61ecc65d24a3c09`
- **start.js**
  `273c5250d5d910ba46893d113b9443d2a7796ea167e241b4d3205f864a43c5e1`
- **utils.js**
  `315130bbb9512deb739f0f797011c100593888c712f71f6e7a5c89082b29b558`
- **websocket.js**
  `3c88a93e9ea7240ac92e3088f2de8dc48c2cd1db5f6803f6c40d6383c00df982`
- **wsRequests/ws-claim-request.js**
  `49eeaf636c66ee9604c76ae8f90a72442662ef542481a31ec1c15c554f036282`
- **wsRequests/ws-getInfo-request.js**
  `649b7110957012d8f7ed6e6a7c29135df8a197ee5652b6bd1527aef71e31e3b8`

- **wsRequests/ws-send-request.js**
  `27cfef9e48087f167c01721a8257e1b6e5812bb02dbbb0295b49e86813627576`
- **wsRequests/ws-switch.js**
  `79cde231c5b36d4f3996fef2d9486857eeb7c3db76b649ae93f348c3fe4ad9e7`
- **requests/asset-request.js**
  `5423a476d75c2d1c6b329d5b2735125f2882e8fec401085d26d7d08b730331ae`
- **requests/check-request.js**
  `c0528352bc1faabc1d5e70a9106896db9f8871f2d4e7c7d909d1dd1e4a457af6`
- **requests/claim-request.js**
  `6f239a602076f51d59165392b33f1244668f2dee93c546d576778245b0a2663e`
- **requests/crypto-request.js**
  `238de32d87abb9367fa742ecf077238ef21e623496b96fa782435150cfbfb21b`
- **requests/getInfo-request.js**
  `09e1193d4e7a0297b4833f1964b8a9f561dd34a7649e71ea1bf46ea489291ba2`
- **requests/send-request.js**
  `2a287d3f30acf10512b4938722e656ca44adb1eb998567c71410f8d4e47aa49e`
- **requests/version-download-request.js**
  `b89256b217913e4ffcf774cd8d4980c322e629cdaeb8048be6eef3613f50ce06`

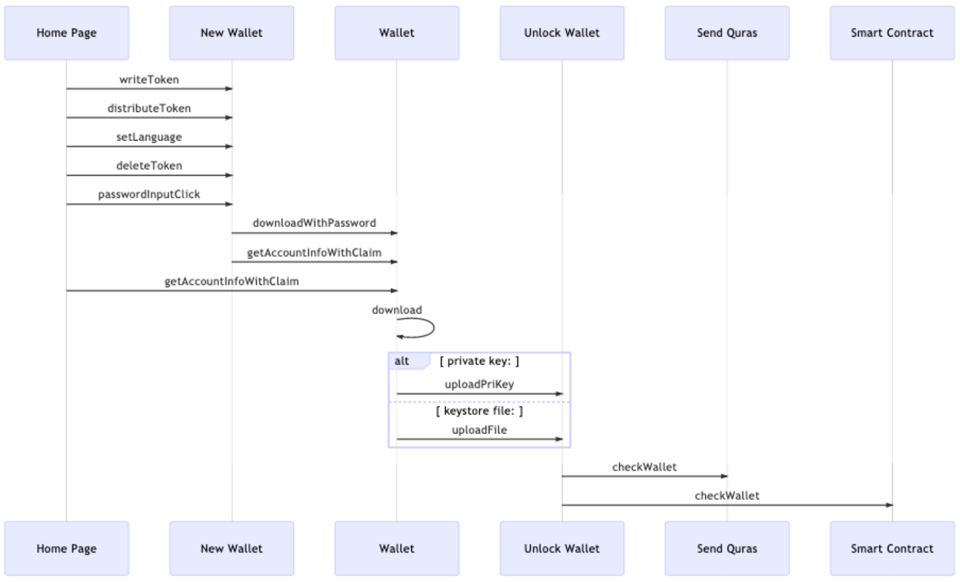
src/js/

- **index.js**
  `1da40334f2051fdb51dd88ec76750d6335ec00eefdcd0aac02654e9d44f1c909`
- **language-switch.js**
  `d55438b7495c8dde38b44f94c9065eff10a18f3f74f54d0e7f59136d5752a478`
- **new-wallet.js**
  `2a55a2a534a9000754937f32ea4c48258931b03da99d91de35f7355a67cb29d3`
- **paper-wallet.js**
  `650967c6af6d596f03f3e3a4babd783d3d2112d014de0b1e4768d1869239316e`
- **send-quras-offline.js**
  `34bf1d25d9f1c15ff82418e7c056024ccc322de1838fd393a35f3e1fc44fa343`
- **send-quras.js**
  `864bd47dcd0f062fb6c63eb2318684849276f78db2a2be942f12c27a34bfcf8a`
- **smart-contract.js**
  `b48adab55547fdb2cb1f8a0b5eb813dded2e5750a0fbc7c19c8c54bb843139d6`
- **transaction-history.js**
  `283a3bd5cd7a19af56b743e8a7767d3bd6f183735e0f3b01587e30f175507fe3`

- **unlock-utils.js**
  `79e767a2e17e517d02b889e3048ac87fc2152ed007a80067cce4069b88ad79a5`
- **unlock-wallet.js**
  `fd7655fb68368e7ce54438106c1369c64325d92a01c473d32d764851e7f0d007`
- **utf.js**
  `a42f02ece25001a27a3cb5b399ceb2cc75f4a6e5fa97d54c6d3f7bf7ae3162dd`
- **utils.js**
  `f63b989e29be328da8a8892cc07126a35b35a92b501effa57fa7a1dbbe0588c4`

Architect & Workflow Overview

Web Wallet Workflow



Blockchain wallet audit checklist

The blockchain wallet source code audit will conduct and focus on answering the following listed areas, checkpoints and harm level. ✓ indicates satisfaction; × indicates unsatisfaction; – indicates inapplicable
Key Management

- ✓ Private Key & Mnemonic Generation Correctness
- ✓ Private Key & Mnemonic Storage Safety Management
- ✓ Private Key & Mnemonic Confidentiality

Cryptography

- ✓ Random hash algorithm correctness in terms of hash function, and signature
- ✓ Hash generation within normal distribution probability

Verification

– Upload documents

– Pass verification

Session Management

✓ Bypassing Session Management Schema

✓ Cross-Site Request Forgery

✓ Session Fixation and Rotation

Data Validation

✓ SQL Injection

✓ Code Injection

✓ Command Injection

✓ Sensitive Data Exposed in Query Parameters

✓ HTTP Splitting/Smuggling

Client-Side

✓ DOM-based Cross-Site Scripting

✓ Cross-Origin Resource Sharing

✓ Cross-Site Flashing

✓ Local Storage

✓ Clickjacking

Error Handling

✓ Sensitive information & interface accessibility

× Using appropriate error codes & message

Business Logic

× Business Logic Data Validation

× Integrity Checks

– Circumvention of Work Flows

General

× Using the latest version of the third party libraries with its new features and bug fix updates. Avoid using deprecated libraries or functions.

× High Test Coverage.
Implementing the unit test as many as possible for ensuring function behavior is meeting its specification. A lot of time, unit-test can discover many unexpected vulnerabilities at the early stage of product release before the lost.

× Provide System testing or End-to-End testing scripts.
Strongly recommend to development some test workflows and scenarios covering the critical business process for capturing the errors at the beginning. This also would benefit for software update and release process later.

× Project github documentation with latest update.
Strongly recommend update the project Readme file for better guideline to other audiences.

× White-paper and technical design documentation with latest update.
Strongly recommend update the white-paper and technical design documents with the latest changes make due to the business model and requirement changes.

Review Comments

Server-side

server-php/ quraswallet-web/server-php/backup.php

- DISCUSSION Functions `download()` and `downloadWithPassword()` are not in classes and have the same name of functions in `Request.php`.

quraswallet-web/server-php/OtherRequest.php

- INFO Function `getLanguage()` is linked with localhost for testing and never used in product.

quraswallet-web/server-php/QurasUtil.php

- DISCUSSION In function `issueAsset()`, we found that `ownerAddress`,`adminAddress` and `issuerAddress` all have value of `_SESSION[_SESSION_Address]`. It seems that we dont have some functions like `changeOwner` or `changeAdmin`, where would `ownerAddress` and `adminAddress` be changed?

- DISCUSSION In function `issueAsset()`, `priKey` and `ownerPriKey` both have value of `_SESSION[_SESSION_PrivateKey]`. It seems that we don't have function of `changeOwner`, where would `ownerPriKey` be changed?

quraswallet-web/server-php/Request.php

- INFO `newWallet()` encrypt `\(password` and `priKey`, where password is derived from **quraswallet-web/src/js/new-wallet.js** directly without hashing. As per security consideration, highly recommend do not directly use any raw password, but instead a password hash value for interacting thru the whole application. Php provide native password hashing API for Cryptography Extension. However, there should not be security concerns using `https` connections.

- INFO `password-create` is only available via `onclick`, ENTER would not work.
- DISCUSSION `downloadWithPassword()` encrypt `(_POST['password'])` and `_SESSION[_SESSION_PrivateKey]`, where password is derived from **quraswallet-web/src/js/new-wallet.js** directly without hashing.
- INFO Function `deleteFile()` is not implemented.
- INFO `checkWallet()` only checking if `_SESSION[_SESSION_PrivateKey]` is null, in which may be not a strong checking condition.
- INFO Function `uploadDB3()` is implemented but never used.
- INFO Function `downloadB3()` is implemented but never used.
- INFO Function `distributeToken()` calls `setIdentity()` from `SafetyModule.php`, which may have some security concerns from the pseudo random number generator.
- INFO `issueAsset()` function and `getAsset()` function has `result = checkSecurity()` commented out.
- DISCUSSION An instance of `SafetyModule` should be initialized in the constructor in order to access to functions of `SafetyModule`.

quraswallet-web/server-php/SafetyModule.php

- DISCUSSION Function `setIdenty()` use `mt_rand()` to generate random numbers as value of `_SESSION[_CONSTANT_Token]`, which is not safe. From mt_rand:
  This function does not generate cryptographically secure values, and should not be used for cryptographic purposes. If you need a cryptographically secure value, consider using random_int(), random_bytes(), or openssl_random_pseudo_bytes() instead.

quraswallet-web/server-php/Utils.php

- INFO `getQRSFromObj()` and `getQRGFromObj()` return value respectively from json. Confuse on the following `+ 0`.

server-nodejs qurawallet-web/server-nodejs/quras-asset.js

- INFO Recommend to add sanity check for `assetHash` of functions `getAssetInfo` and `getAsset`.
- INFO Recommend to add sanity check for `params` of functions `issueAsset` and `sendAsset`.
- INFO Function `getAsset` returns constant values directly, which does not have the getter functionality.
- INFO Function `sendAsset` returns `data: succeed` directly, which does not have the setter functionality.

qurawallet-web/server-nodejs/quras-module.js

- INFO Recommend to add sanity check for `privateKey` in functions `getPublicKey` and `getAddressFromPriKey`, by calling `isPrivateKey`
- INFO Recommend to add sanity check for `publicKey` in function `getAddressFromPubKey` , by calling `isPublicKey`
- INFO Recommend to add sanity check for `address` in functions `balance`, `transactions` and `getClaimInfo`, by calling `isAddress`
- INFO Recommend to add sanity check for `destination`, `amount`, `privateKey`, `balanceData` and `isQRS` in function `buildTransaction`
- INFO Recommend to add sanity check for `destination`, `amount`, `privateKey` and `balanceData` in functions `sendQRS` and `sendQRG`
- INFO Recommend to add sanity check for `rawTx` in function `broadcastTx`
- INFO Recommend to add sanity check for `address` and `data` in function `claimQRG`

qurawallet-web/server-nodejs/request.js

- **DISCUSSION** What is the use case and intention for `app.get('/.well-known/pki-validation/BA3098037F664DDECE739D889C12ED1C.txt', function())`?
  - ✓ Quras Confirmed with Quras team, the purpose of this design used for the initial setup.

qurawallet-web/server-nodejs/router.js

- INFO Typo, recommend renaming the `versionAndDowload` to `versionAndDownload`.

qurawallet-web/server-nodejs/server-cypto.js

- **MINOR** Function `symmetricEncrypt()` uses `crypto` module's`createCipher()` function, which is deprecated according to Node.js Documentation, instead use `crypto.createCipheriv()`.
- **MINOR** Function `symmetricDecrypt()` uses `crypto` module's`createDecipher()` function, which is deprecated according to Node.js Documentation, instead use `crypto.createDecipheriv()`.
- INFO `module.export.encrypt`: Recommend to add sanity check for input parameters `priKey` and `msg`.
- INFO `module.export.decrypt`: Recommend to add sanity check for input parameters `priKey` and `msg`.
- INFO Function `setECDH`: Recommend to add sanity check for input parameter `priKey`.

- $\boxed{\text{INFO}}$ For functions `symmetricEncrypt()` and `symmetricDecrypt()`: Recommend to add sanity check for input parameters `cypherName`, `iv`, `key` and `plaintext`, and related logging message for error tracking.
- $\boxed{\text{INFO}}$ For functions `hashMessage()` and `macMessage`: Recommend to add sanity check for input parameters `cypherName`, `key` and `message`.
- $\boxed{\text{INFO}}$ Function `encrypt()`: Recommend to add sanity checks for input parameters `publicKey` and `msgStr`. Another approach would be tracking output of each function calls.
- $\boxed{\text{INFO}}$ Function `decrypt()`: Recommend to add sanity checks for input parameters `ecdh` and `strMsg`. Another approach would be tracking output of each function calls.

quraswaller-web/server-nodejs/utils.js

- $\boxed{\text{INFO}}$ Recommend to add sanity check for `value` in function `getBalanceJSONData`.

quraswaller-web/server-nodejs/websocket.js

- $\boxed{\text{INFO}}$ Recommend to add sanity check for `wssServer` in function `server`.
- $\boxed{\text{INFO}}$ Function `server`: Recommend to add error message logging in the `catch` block, as well as in `open()` and `close()`.

qurawallet-web/server-nodejs/wsRequests/ws-claim-request.js

- $\boxed{\text{INFO}}$ Recommend to add sanity check for `params` and `params.value` of functions in both `getClaimInfo` and `claimQRG`, before they are assigned to `const address`.
- $\boxed{\text{INFO}}$ Recommend to use `console.error(value)` for error/failure tracking.

qurawallet-web/server-nodejs/wsRequests/ws-getInfo-request.js

- $\boxed{\text{INFO}}$ Recommend to add sanity check for `params` of functions in `txList`, `balance`, `getAccount`, such that `params.value` can be assigned to `const address`.
- $\boxed{\text{INFO}}$ Recommend add sanity check for `params` of function in `getAccountByEncryptInfo` , such that `params.encryptInfo` can be assigned to `encryptInfo`, `params.password` can be assigned to `password` and `params.language` can be assigned to `language`.
- $\boxed{\text{INFO}}$ `const language` in function in `getAccountByEncryptInfo` is assigned but not used.
- $\boxed{\text{INFO}}$ `localization` is imported but not used.

qurawallet-web/server-nodejs/wsRequests/ws-send-request.js

- $\boxed{\text{INFO}}$ Recommend to add sanity check for `params` of functions in `sendQRS` and `sendQRG`, such that `params` to be passed to `sendCoin()` can successfully assign value of `params.address`, `params.language`, `params.amount` and `params.priKey` as local variables in `sendCoin()`.

- INFO Function `sendCoin()`: Recommend to have error logging message for `fail` case, using `console.error()`.

qurawallet-web/server-nodejs/wsRequests/ws-switch.js

- INFO Recommend to add sanity check for `params` of function in `dealWithMsg`, such that `params.order` can be used to switch case, and `params` can be valid input paramters for function calls.

qurawallet-web/server-nodejs/requests/asset-request.js

- INFO Recommend to add sanity check for input parameters of functions in routes of `/issueAsset`, `/getAsset`, `/sendAsset` and `/assetTest`.
- DISCUSSION Route of `assetTest/` sends `'data':'adasd'`, Is the response message as expected?
- INFO Function in `/assetTest` route, recommend to use `console.error(err)` for error message instead of `console.log()`.
- INFO Recommend to remove dead codes.

qurawallet-web/server-nodejs/requests/check-request.js

- INFO Recommend to add sanity check for input parameters of functions in routes of `/isPriKey`, `/isPubKey`, `/isAddress` and `/test`.

qurawallet-web/server-nodejs/requests/claim-request.js

- INFO Recommend to add sanity check for input parameters of functions in routes of `/getClaimInfo` and `/claimQRG`.

qurawallet-web/server-nodejs/requests/crypto-request.js

- INFO Recommend to add sanity check for input parameters of functions in routes of `/encrypt` and `/decrypt`.

qurawallet-web/server-nodejs/requests/getInfo-request.js

- INFO Recommend to add sanity check for input parameters of functions in routes of `/priKey`, `/pubKey`, `/address`, `/balance`, `/balanceData`, `/txList`, `/getAccount`, `/getAccountWithoutPassword`, `/getAccountByPrivateKey` and `/getAccountByEncryptInfo`.
- INFO For functions `returnAccountInfoWithPriKey()` and `returnAccountInfo()`, since most of the codes are duplicated, recommend to merge these two functions into one and pass `password` as an optional input parameter.

qurawallet-web/server-nodejs/requests/send-request.js

- INFO Recommend to add sanity check for input parameters of functions in routes of `/sendQRS`, `/sendQRG`, `/broadcastTx`, `/wsTest` and `wsPostTest`.
- INFO Recommend to remove dead codes.

qurawallet-web/server-nodejs/requests/version-download-request.js

- INFO Recommend to add sanity check for input parameters of functions in routes of `/downloadMacWallet` and `/macWalletVersion`.

Client-side

qurawallet-web/src/js/index.js

- INFO `checkProtocol()` Please do not leave lstinline[language=Solidity]console.log for any production grade. If the logger is valuable context, recommend to use logger library as `winton`

qurawallet-web/src/js/language-switch.js

- INFO Consider grouping the same type of the variables together for better readability. i.e:

```
var language = 'en'
var callbackFunction;

const languageCookie = 'language';
```

  - INFO `getBrowserLang()` can be simplify as:

```
function getBrowserLang() {
    const cookieLang = getCookie(languageCookie);
    language = cookieLang;
    if (cookieLang === ""){
        language = (navigator.language).toLocaleLowerCase();
    }
    return language;
}
```

  - INFO Consider using `let` for having variable as block scope/ local scope.

    var: function scoped let: block scoped Reference

  - INFO `getLangArr(lang)` Consider some of following items:
    - Consider declaring `local` var using `let`.
    - When using try & catch, there is no error handling implementation. This is not a good practice of keeping the error silent.
    - What kind of errors will be possible occurred in `getLangArr`?

quraswallet-web/src/new-wallet.js

- **MINOR** `passwordInputClick()` The password input check is kind of weak(only empty check, length, and complexity expecting to handle by backend)

- INFO `filterRandom()` condition check need to be strong `res = Number(d)%2 === 0 ? r + d : d + r;`

  - What is the intention of `filterRandom()`, is it normal distributed?

- INFO Consider grouping the same type of the variables together for better readability. i.e:

```
...
const localZh_CN={...}
const localKo_Kr={...}
const jsEn={...}
const jsJa={...}
...
```

quraswallet-web/src/send-quras.js

`send-quras` includes following functions & its intention:

- `getAccountInfo()`: an ajax post call to server for getAccountInfoWithClaim. If success, display account info and else set highlighted the `#prompt-info`. When error, display alert box.
- `displayAccountInfo()`:
  - if address not empty, set `#address` text.
  - set `#qrs-balance` and `QRSBalance` data
  - set `#qrg-balance` and `QRGBalance` data
- `switchCoinType()`: set the coinType to type variable
- `sendCoin()`: trigger the sendQRS() or sendQRG() by active coin type.
- `enableSendBtnAndPrintMsg()`: as named enabled the `#send` button and set the msg on `\#errorInfo`.
- `updateAccountInfo()`: an ajax post call to server for updating the balance. If success, display account info and else set highlighted the `#prompt-info`. When error, display alert box.
- $\boxed{\text{INFO}}$ The below code are duplicating in functions `sendQRS()[79:97]` & `sendQRG()[131:149]`, please consider extracting those codes for re-usability.

```
const send = ('\#send');
enableDisableBtn(send,false,getGeneralString('button-sending'));
const errorInfo = ('\#errorInfo');
var address = ('\#InputAddress').val();
errorInfo.text('');
const amount = Number(('\#InputAmount').val());
if (!Number.isInteger(amount)){
    enableSendBtnAndPrintMsg(getString('invalidAmount'));
    return;
}
var amountStr = amount.toString();
if (amount <= 0){
    enableSendBtnAndPrintMsg(getString('lessThanZero'));
    return;
}
if (address === ''){
    enableSendBtnAndPrintMsg(getString('emptyAddr'));
    return;
}
```

- INFO `switchCoinType()` only assigns `activeCoin` to `type`. All other lines of code are not used.

quraswallet-web/src/smart-contract.js

- **MINOR** Consider removing the `Array.prototype.contain`. Javascript provide includes function as default by its nature, in which is exactly same as contain here.

- INFO `getAccountInfo()` can be simply to:

```
...
success: function(data){
    if (data.success) {
        if (data.address){
            displayAccountInfo(data);
        }else {
            jumpUrl("unlock-your-wallet.html");
        }
    }
},
...
```

- INFO The function name is not meet its intention for `deploy()`. Recommend move the checking logic from function `deploy()` to `addAsset()` with minimum change as:

```
  function addAsset() {
    if ("#asset-type").val() !== 'Asset') return;

    enableDisableDeployBtn(false);
    const assetName = $('#asset-name').val();
    ...
  }
```

- MINOR Function `addAsset()` has ajax call commented out, which means the functionality would not be available.

- INFO Validness checking for `assetAmount` and `assetPrecision` are not strong enough, recommend add upper bound for the checking.

- MINOR After `Deploy` button is clicked and disabled, recommend to have some more messages as feedbacks. Instead of displaying `alert('Add asset failed!')`, some message like `Failed: Not enough balance`, `Failed: Precision invalid`, etc. could help the user getting more context about the on-going error.

- INFO Function `enableDisableAmount()`: Recommend renaming the function as `disableAmount()` for better readability. Refer to Airbnb JavaScript Style Guide.

- INFO Function `enableDisableDeployBtn()`: Recommend renaming the function as `disableDepolyButton()` for better readability.

- INFO Function `send()` uses `switch case` statement to handle different cases of smart contract logics. However only `transfer` logic is implemented, recommend remove the `switch case` statement and use `if` statement to check for `case transfer`.

- INFO Recommend move the language constants to another file to ensure contents in smart-contract.js are to support the smart contract logics
- INFO Recommend to remove `getStyle()` and put '`none`' or '`block`' directly for `display()` function.

quraswallet-web/src/unlock-utils.js

- INFO Recommend renaming the function `doChange()` to be `uploadKeyStoreFile()` for better readability.
- **MINOR** `doChange()`: the variable `filepath` does not have sanity check for case empty string. This may lead to failure of the `filename` and extension may happen index out of range if the `fileName` do not contain '.'
- INFO For `ajax` calls in functions `uploadFile()`, `uploadPriKey()`, `uploadMnemonicPhrase()`, `requestWriteToken()` and `claimQRG()`, event `complete` is not implemented.
- INFO Recommend to remove `getStyle()` and put '`none`' or '`block`' directly for `display()` function.
- MINOR `uploadFile()`: Consider addressing the following issues:
  - The `password` do not have basic sanity check.
- INFO `uploadFile()`: The part password wrong can be simplify as:

```
...
//password wrong
if (keyStore && callback) {
    callback();
}
...
```

- DISCUSSION `uploadFile()`: Not sure in what use-case that possible fall to here[76:80]

```
...
if (!keyStore) {
    if (callback != null) {
        callback();
    }
}
...
```

quraswallet-web/src/unlock-wallet.js

- **MINOR** If a user reaches the `unlock-your-wallet` page after creating an account and clicks "View History on the left (which will invoke the action `viewHistory()` in **quraswallet-web/src/unlock-wallet.js**)", the user will be redirected to the block explorer. However, the block explorer will run into a rendering error.
- INFO Function `checkMnemonicPhrase()` would always return true.

- Function `importWallet()`:
  - **MINOR** For `case 'Mnemonic phrase'`, the else statement would never be used since `checkMnemonicPhrase(mne)` always return true. Recommend to add logging message for `mne === ''''` check and implement the `checkMnemonicPhrase(mne)` case.
  - INFO For `case 'Private key'`, recommend to add logging message for `pri === ''` check.
- INFO Function `displayAccountInfo()` set `const a = 0` when `privateKey` is null, where `const a` is never used.
- INFO Consider removing the function `initCtl()`, which is implemented but never used.
- INFO `getBalanceStr()` Consider checking if the type is an empty string, and the function can be simply as:

```
function getBalanceStr(balance, type) {
    var balanceStr = type + ': ';
    return balanceStr + (balance || '0');
}
```

quraswallet-web/src/utils.js

- INFO Function `generateQrCode()`: Recommend to give `length` variable as a default value of 180 and remove the conditional `length` assignment.

```
function generateQrCode(code,text,length=180) {
    code.empty()
...
```

- INFO Function `errorHandler()` used `alert()` to handle error message. Recommend using `logger` module for better logging practice.
- INFO Function `windowLoadHeader()` is not implemented.
- INFO Function `enableDisableBtn()`: Recommend to change function name to `disableButton()` for better naming variable practice according to Airbnb JavaScript Style Guide.
- INFO For `ajax` calls in functions `getToken()`, `updateToken()`, `requestDeleteToken()`, `requestWriteToken()` and `claimQRG()`, event `complete` is not implemented.
- INFO `jumpUrl()` and `requestWriteToken()` are duplicate.

Quras API Service

Source Code SHA-256 Checksum

- **address.js**
  `4121b6fece05dac99094369c458e81b1f849fc1899606584cce5f9b1add6a993`
- **addresses.js**
  `963b91501d6237b64853eaad3ccb7e076cfa6264451a21b77f0808ca44756315`
- **assets.js**
  `23e3790e09590fe71a97f28f03871a41b15aa467e0a43a84c1101cb77025db97`
- **block.js**
  `62328f0a20dffaccfa4f75e3a0b5243a25e9026a7786ec040617069782a73e85`
- **blocks.js**
  `edec35d708f953b57cf6e0de5cf1ba0414cf80f45f7b85b9d8522215269c1b7d`
- **nodes.js**
  `d94806b99a4a565090853db038133dc608991554d3c62d4e9e37fbb0116d4366`
- **status.js**
  `245166b170bc0d2049652bee44ce4b603ec4ff899b6ba651e12be3714c076449`
- **tx.js**
  `2301421f3b67994427702e1128a1c246150f54b98407b2a5bfe79cb9256a88f5`
- **txs.js**
  `aa6c77fa6e453521964e82da243a012640631821abb8300d2308d905ed21bc45`

quras-api-service

The quras-api-service is an api service that interact with Quras Blockchain.
quras/quras-api-service/common/commonf.js
commonf.js contains the data layer logic for block, transaction, status, account.

- INFO `getTransactionHistory()`: there no state change for `conn`, `var connection` can be removed.

```
...
function getConn(callback) {
    pool.getConnection(function (err, conn) {
        // ensure connection
        if (!conn) return;
        if (err) {
            callback(err, conn);
        } else {
            callback(null, conn, addr);
        }


});
}
...
```

- INFO `getUnspent()`: `getUnspentList` (LOC:152) and (LOC: 154) recommend to have strong type checking.
- INFO `getFormatedBlock()`: Inconsistent coding style with other functions for variable assignment, and `JSON.parse(block.script)` may thru error when `block` or `block.script` is null.

```
...
getFormatedBlock: function (block) {
    let formatedBlock = {};
    ...
    formatedBlock.script =
    JSON.parse(block \&\& block.script ? block.script || ‘’);
    formatedBlock.time = block.time;
    ...
    return formatedBlock;
},
```

- `getCurrentBlockHeight()`
  - getBlockHeight() using sql statement as `SELECT * FROM status WHERE id = 0` where the result is not sorted by any order, how is the id works in the status table.

quras/quras-api-service/routes/v1/address.js

- INFO Consider removing unused variable(s), libraries: `cryptof`, `generator`, `crypto`, `rpcServer`.
- INFO Consider using `const` for `constants`
- Get balance endpoint: `/balance/:addr`
  - INFO There is no sanity check for `addr`
- Get history endpoint: `/history/:addr`
  - INFO Consider removing unused variable `asset`.
  - INFO Consider change logging message from `GetBalance` to `GetHistory`.

quras/quras-api-service/routes/v1/addresses.js

- INFO Consider removing unused variable(s), libraries: `cryptof`, `controller`, `promisify`, `pool`, `generator`, `crypto`.
- INFO `getAddress()` (LOC 69) Consider using the `IN` operator, it allows you to easily test if the expression matches any value in the list of values. The code can be simplified as:

```
...
SELECT name, txid FROM register_transaction WHERE txid in(?);

var sqlAssetName =[];
retTx.balances.forEach(balance => {
    sqlAssetName.push(balance.asset_hash);
});
```

- $\boxed{\text{INFO}}$ Function `getAddress()`: Recommend to log the caught errors by variable `err`, instead of only passing `Connection Error` to the callback function.
- $\boxed{\text{INFO}}$ Recommend to add sanity check for `address` as an input parameter either at the API endpoint `/:address`, or at the function `getAddress()`.

quras/quras-api-service/routes/v1/assets.js

- $\boxed{\text{INFO}}$ Consider removing unused variable(s), libraries: `cryptof`, `controller`, `promisify`, `rpcServer`, `pool`, `generator`, `crypto`, `hash`.
- $\boxed{\text{INFO}}$ Get asset endpoint `/:offset/:limit`: Recommend not to call function `getAssets()`, when `offset` or `limit` is not number. Since function `getAssets()` does not handle the `-1` case separately.
- $\boxed{\text{INFO}}$ Recommend to rename functions `getAssets()` and `getAsset()` to be `getAssetsFromParams()` and `getAssetsFromHash()` for better readability.
- $\boxed{\text{INFO}}$ Consider change the logging message for get asset endpoint `/`, which is the same as the logging message of `/:offset/:limit`. Recommend to make some differences to be easy tracking.
- $\boxed{\text{INFO}}$ Recommend to add sanity check for `hash` in get asset endpoint `/:hash`.

quras/quras-api-service/routes/v1/block.js

- $\boxed{\text{INFO}}$ Consider removing unused variable(s), libraries: `cryptof`, `controller`, `rpcServer`, `generator`, `crypto`.

quras/quras-api-service/routes/v1/blocks.js

- $\boxed{\text{INFO}}$ Consider removing unused variable(s), libraries: `cryptof`, `controller`, `promisify`, `rpcServer`, `pool`, `generator`, `crypto`.
- $\boxed{\text{INFO}}$ Recommend to rename functions `getBlock()` and `getBlocks()` to be `getBlockFromHeight()` and `gerBlockFromParams()` for better readability.
- $\boxed{\text{INFO}}$ Function `getBlock()`: Recommend to remove the dead codes.
- $\boxed{\text{INFO}}$ For number `-1` and `-2` to handle the corner cases, recommend to save as constants.
- $\boxed{\text{INFO}}$ Functions `getBlock()` and `getBlocks()`: Recommend to log the caught errors by variable `err`, instead of only passing `Connection Error` to the callback function.

quras/quras-api-service/routes/v1/blocks.js

- $\boxed{\text{INFO}}$ Consider removing unused variable(s), libraries: `cryptof`, `controller`, `promisify`, `rpcServer`, `pool`, `generator`, `crypto`.
- $\boxed{\text{INFO}}$ Recommend to rename functions `getBlock()` and `getBlocks()` to be `getBlockFromHeight()` and `gerBlockFromParams()` for better readability.

- DISCUSSION Confused on the error message `''Page is not a valid integer''`.
- INFO For number `-1` and `-2` to handle the corner cases, recommend to save as constants.
- INFO Functions `getBlock()` and `getBlocks()`: Recommend to log the caught errors by variable `err`, instead of only passing `Connection Error` to the callback function.

quras/quras-api-service/routes/v1/nodes.js

- INFO Consider removing unused variable(s), libraries: `cryptof`, `controller`, `rpcServer`, `generator`, `crypto`.
- INFO `getNodes()`: The function intention will return all nodes record from database. The only concern here is when the data getting larger and larger, it might be encountered in performance issue when returning at once.
- INFO Functions `getNodes()` and `getNodeFromHash()`: Recommend to log the caught errors by variable `err`, instead of only passing `unexpected request` to the callback function.

quras/quras-api-service/routes/v1/status.js

- INFO Consider removing unused variable(s), libraries: `cryptof`, `controller`, `promisify`, `rpcServer`, `pool`, `generator`, `crypto`.
- **DISCUSSION** `getStatus()`: The function name is `getTxFromTxid`, but the function behavior is not same as named. The function intention is to return the first record of the statuses, which can be complete in the query w/o wasting any additional storage resource.
  - The status query is not storing by any order, is this meeting the design intention?

    ```
    var sqlStatus = ''SELECT * FROM status LIMIT 1";
    ```

- INFO Function `getStatus()`: Recommend to log the caught errors by variable `err`, instead of only passing `unexpected request` to the callback function.

quras/quras-api-service/routes/v1/tx.js

- INFO Consider removing unused variable(s), libraries: `commonf`, `cryptof`, `controller`, `async`, `logger`, `rpcServer`, `pool`, `generator`, `crypto`, `RESPONSE_ERR`.
- INFO Get send endpoint: Variables `privKey`, `asset` and `response` are initialized but not used.
- INFO Get send endpoint: This function only get values of `addr` and `amount`, log the message to console.

quras/quras-api-service/routes/v1/txs.js

- INFO Consider removing unused variable(s), libraries: `cryptof`, `controller`, `promisify`, `rpcServer`, `pool`, `generator`, `crypto`.

- INFO Function `getTx()`: Recommend to change the `if-else if-else` block to `switch-case` block to handle complex conditions and increase the readability.

```
let sqlExclusiveTx;
let exclusiveTx;
switch (txsResult[0].type) {
    case ``MinerTransaction":
        sqlExclusiveTx = ``SELECT * FROM miner_transaction WHERE txid=?";
    break;
    case ``IssueTransaction":
        sqlExclusiveTx = ``SELECT * FROM issue_transaction WHERE txid=?";
    break;
    case ``ClaimTransaction":
        sqlExclusiveTx = ``SELECT * FROM claim_transaction WHERE txid=?";
    break;
    case ``EnrollmentTransaction":
        sqlExclusiveTx = ``SELECT * FROM enrollment_transaction WHERE txid=?";
    break;
    ...
}
....
exclusiveTx = connection.query(sqlExclusiveTx, [txid]);
exclusive = exclusiveTx[0];
...
```

- **MINOR** The `getTx()` What is the intention of the function in [Line 90:105]?

- The function behavior, when `vins` is greater 0 then iterate the array and append the `sqlwhere` to `sqlFindUtxos`. However, by line 105 `vinUtxos` always query and return 1 from db?

```
...
sqlFindUtxos = ``SELECT * FROM utxos WHERE txid in (txid) AND \
tx_out_index in (?)";

let txOutIndexes = [];

vouts.forEach(vout => {
    txOutIndexes.push(vout.n);
});


voutUtxos = connection.query(sqlFindUtxos, txOutIndexes);
...
```

- INFO Functions `getTransactions()`and `getTx()`: Recommend to log the caught errors by variable `err`, instead of only passing `"Connection Error"` to the callback function.

CERTIK

Building Fully Trustworthy Smart Contracts and Blockchain Ecosystems