ACA NETWORK SMART CONTRACT FINAL AUDIT REPORT

by QuillAudits, February 2019

Introduction :

This Audit Report highlights the overall security of **ACA** Smart Contract. With this report, we have tried to ensure the reliability of their smart contract by complete assessment of their system's architecture and the smart contract codebase.

Auditing Approach and Methodologies applied :

Quillhash team has performed thorough testing of the project starting with analysing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third party smart contracts and libraries.

Our team then performed a formal line by line inspection of the Smart Contract in order to find any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

In the Unit testing Phase we coded/conducted Custom unit tests written for each function in the contract to verify that each function works as expected. In Automated Testing, We tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was tested in collaboration of our multiple team members and this included -

1. Testing the functionality of the Smart Contract to determine proper logic has been followed throughout.
2. Analyzing the complexity of the code by thorough, manual review of the code, line-by-line.
3. Deploying the code on testnet using multiple clients to run live tests
4. Analysing failure preparations to check how the Smart Contract performs in case of bugs and vulnerabilities.
5. Checking whether all the libraries used in the code are on the latest version.
6. Analyzing the security of the on-chain data.

Audit Details

- Project Name: ACA Network
- Client Name: ACA network
- GitHub : https://github.com/acanetwork/tokensale
- Languages: Solidity,Javascript

Summary of ACA Coin Smart Contract :

QuillAudits conducted a security audit on smart contracts of aca network.There are two major smart contracts .One contract is used to create the Erc20 token which is ACAToken contract and second is used to issue these tokens in crowdsale which is ACATokenSale contract.

Some of the key features of the smart contract are:-

- Locking and unlocking of transfer function by owner.
- ERC20 emergency drain functionality that will transfer any ERC20 token to owner address.
- Bounty and referral tokens functionality in crowdsale contract.

Audit Goals

The focus of the audit was to verify that the smart contract system is secure, resilient and working according to its specifications. The audit activities can be grouped in the following three categories:

Security: Identifying security related issues within each contract and within the system of contracts.

Sound Architecture: Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.

Code Correctness and Quality: A full review of the contract source code. The primary areas of focus include:

- Correctness
- Readability
- Sections of code with high complexity
- Quantity and quality of test coverage

Security Level references :

Every issue in this report was assigned a severity level from the following:

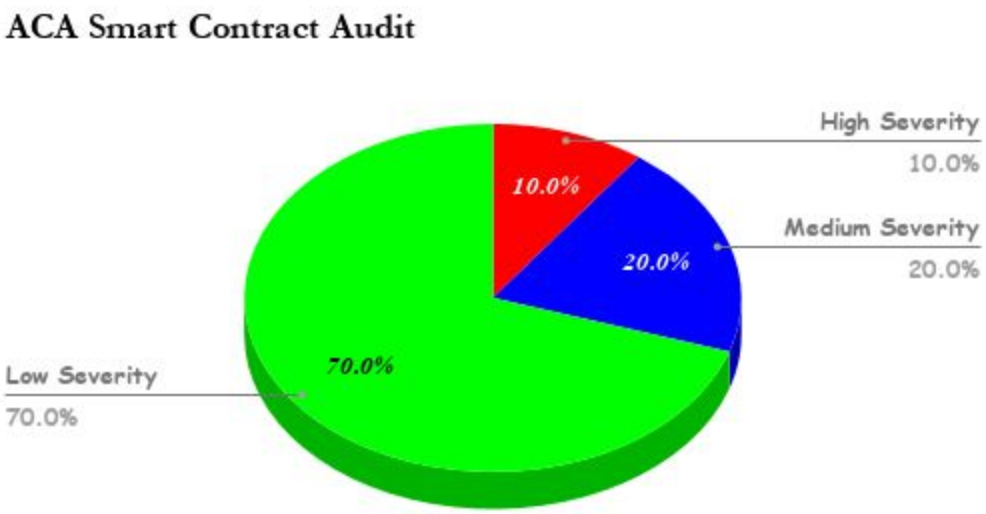
High severity issues will probably bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Number of issues per severity

	Low	Medium	High
Open	1	1	0
Closed	6	1	1



High severity issues:-

1. delBounty() function of sale contract will not work because the require statement in this function is checking that bounty amount to be deleted must be greater than the bounty which is sent. But in real use case the bounty amount which is to be deleted must be less than the bounty amount sent.

```
require(_amount >= bountySent);
```

Use

```
require(_amount <= bountySent);
```

Status : Fixed ✓

Medium Severity Issues:-

1. In _preValidatePurchase() function of sale contract check below condition Line no 455 of sale contract

   require(remains > amount);

So in case when remains is equal to amount this condition always fail.

(remains == amount) Fail.

USE :

require(remains >= amount);

Status : Fixed ✓

Low Severity Issues:-

1. This **modifier** is never used in contract .

   `onlyWhileOpen`

   Crowdsale contract line no 99.

   Status : Fixed ✓

   Either use modifier at appropriate position or remove it from smart contract code.

2. These Events never used in contract.

   `LogUint256`,
   `LogAddress`
   In both contracts.

   `StageCapReached`, line 81 sale contract.

   Status : Fixed (In TokenSale contract Only).

Either use event at appropriate position or remove it from smart contract code.

3. _updatePurchasingState() function of sale contract is called from buyTokens() but doesn't contain any logic or operations .

Status : Fixed ✓

Remove the function from smart contract or add required operations in it.

4. setCapacity() function of sale contract doesn't check any of the condition that that was checked while adding stage.

```
require(_capacity < hardCap);
require(_minimumWei.mul(_rate) < _capacity);
require(_maximumWei.mul(_rate) < _capacity);
```

Status : Fixed (update as described in Initial audit reviews).

5. setTransferable() should check that you are not setting same previous stage of a transferable.

Status : Not considered as an Issue.

As this will just waste the gas.

6. The return value of an external call should check as `transferfrom()` is returning bool value, that should be checked in `deliverdToken()` function of sale contract.

Status : Fixed ✓

Use

```
require(token.transferFrom(owner, _beneficiary,
_tokenAmount));
```

7. Solidity style guide is not followed :

The visibility modifier for a function should come before any custom modifiers.

Line 193,204,213,222,230,298,326,352,356,386,537 of Sale contract .

Status : Fixed ✓

❌ function setRate(uint _index, uint256 _rate) onlyOwner public

It should be

✔️ function setRate(uint _index, uint256 _rate) public onlyOwner

Review Initial Audit Fixes

Medium Severity Issues:-

These statements can be used only while adding stages using addStage() function of token contract.

```
require(_minimumWei.mul(_rate) < capacity);
require(_maximumWei.mul(_rate) < capacity);
```

For checking require condition using stage index :

Use

```
require(stages[_index].minimumWei.mul(stages[_index].rate) < _capacity);
require(stages[_index].maximumWei.mul(stages[_index].rate) < _capacity);
```

Low Severity Issues:-

LogUint256, LogAddress is not removed from token Contract.(Audit Review)

Initial Audit Fixes commit hash:

220f6c7a8312b6670639b8e3d927dcf480c9a562

Unit Testing

Test Suite

Contract: ACA

✓ Should Deploy ACA Token Sale Contract (594ms)

✓ Should correctly initialize constructor values of ACA Token Sale Contract (239ms)

✓ Should Not Deploy ACA Token Contract with same owner and Admin address (351ms)

✓ Should Deploy ACA Token Contract (404ms)

✓ Should correctly initialize constructor values of ACA Token Contract (167ms)

✓ Should Not WhiteList for accounts [3] by Non Admin Accounts (136ms)

✓ Should WhiteList for accounts [4],[5],[6] using Add many users function (430ms)

✓ Should Not Verify accounts [3] by Non Admin Accounts (124ms)

✓ Should Verify for accounts [3] (297ms)

✓ Should Verify for accounts [4],[5],[6] using Add many users function (575ms)

✓ Should Not Verify accounts [6] from Non Owner Accounts (129ms)

✓ Should Not add to whitelist accounts [6] from Non Owner Accounts (144ms)

✓ Should UnVerify for accounts [6] (194ms)

✓ Should remove From Whitelist for accounts [6] (173ms)

✓ Should be able to Setup Bounty from Owner Accounts (387ms)

✓ Should WhiteList for accounts [3] (293ms)

✓ Should Not be able to add Stage from non Owner Accounts (256ms)

✓ Should be able to add Private Stage from Owner Accounts (252ms)

✓ Should be able to check private sale token sold (41ms)

✓ Should be able to check capacity of a current stage (55ms)

✓ Should be able to add Private sale Tokens (158ms)

✓ Should be able to check current stage token sold (38ms)

✓ Should be able to sub Private sale Tokens (193ms)

✓ Should be able to add another Private sale Tokens (159ms)

✓ Should Not be able to enable Token Sale from Non Owner Accounts (102ms)

✓ Should be able to add Pre Stage from Owner Accounts (214ms)

✓ Should be able to check opening time through stage (43ms)

✓ Should be able to check closing time through stage (43ms)

✓ Should be able to check capacity of a stage (57ms)

✓ Should be able to check Rate of a stage (47ms)

✓ Should be able to set Sale Period for Pre Stage (150ms)

✓ Should be able to set Rate for Pre Stage (182ms)

✓ Should be able to check Token Sale is enabled or not (50ms)

✓ Should be able to enable Token Sale (239ms)

✓ Should Not be able to buy Tokens from Non KYC, Whitelist accounts (155ms)

✓ Should be able to buy Tokens from accounts[3] in Pre Sale (804ms)

✓ Should be able to check current stage token sold (52ms)

✓ Should be able to check opening time (40ms)

✓ Should be able to check closing time (48ms)

✓ Should invite accounts[4] by accounts[3] (176ms)

✓ Should be able to buy Tokens from accounts[4] Pre Sale (693ms)

✓ Should check if soft cap Reached (86ms)

✓ Should add Token Address to Crowdsale (147ms)

✓ Should Not be able to set Claimable to True from Non Owner Accounts (86ms)

✓ Should set Claimable to True (200ms)

✓ Should Not be able to claim Tokens for Non Participant Accounts[6] (157ms)

✓ Should claim Tokens for Accounts[3] (327ms)

✓ Should claim Tokens for Accounts[4] (326ms)

✓ Should be able to buy Tokens from accounts[4] Pre sale (924ms)

✓ Should be able to check Stage of Crowdsale (43ms)

✓ Should claim Tokens for Accounts[4](Fail) (326ms)

✓ Should be able to update Stage (244ms)

✓ Should be able to check Balance Of Smart contract (890ms)

✓ Should be able to check sale Status and finalize stage (338ms)

✓ Should Not be able to finalize stage Once it's finalized (102ms)

✓ Should be able to check Balance Of Smart contract vault After Finalization (934ms)

✓ Should WhiteList for accounts [6] to Send Bounty Tokens (168ms)

✓ Should Verify for accounts [6] to Send Bounty Tokens (287ms)

✓ Should Not be able to Add Bounty Tokens to accounts[6] from Non owner accounts (106ms)

✓ Should Add Bounty Tokens to accounts[6] from owner (254ms)

✓ Should check Bounty Tokens of accounts[6] (41ms)

✓ Should Delete Bounty Tokens of accounts[6] from owner (failed) (241ms)

✓ Should claim Bounty Tokens for Accounts[6] (325ms)

✓ Should be able to Transfer ownership (163ms)

✓ Should be able to Transfer Admin (170ms)

✓ Should be able to check Total Supply of token contract

✓ Should be able to check Tokens are transferable or not

✓ Should Not be able to transfer Tokens before transferable is enable (83ms)

✓ Should be able to check Balance Of owner

✓ Should be able to Transfer ownership (289ms)

✓ Should be able to Transfer Admin (162ms)

✓ Should Not be able to enable transferable by Non Admin Accounts (78ms)

✓ Should be able to enable transferable (120ms)

✓ Should check transfer allowed or Not

✓ Should be able to transfer Tokens after transferable is enable (249ms)

✓ should Approve address to spend specific token (106ms)

✓ should be able to increase Approval (151ms)

✓ should be able to decrease Approval (155ms)

✓ Should be able to transfer Tokens on the behalf of accounts[4] (246ms)

✓ Should Not be able to Burn Tokens larger than balance (99ms)

✓ Should be able to Burn Tokens from token Contract (203ms)

✓ Should be able to Lock transfer for particular Account (128ms)

✓ Should Not be able to transfer Tokens after locked by Owner (94ms)

✓ Should be able to UnLock transfer for particular Account (129ms)

✓ Should be able to Lock transfer for Own Account (139ms)

✓ Should be able to check previous and current owner balance (56ms)

✓ Should be able to check allowance of Previous owner to Token Contract

✓ Should be able to check allowance of New owner to Token Contract

✓ Should be able to Refund Balance Of Investors when softcap not reached.

✓Should Deploy SampleERC20 Token Contract (116ms)

✓ Should be able to transfer Tokens to Token contract address (138ms)

✓ Should be able to transfer Tokens from Token contract to Owner address using Emergency drain function (174ms)

✓ Should Delete Bounty Tokens of accounts[6] from owner (212ms).

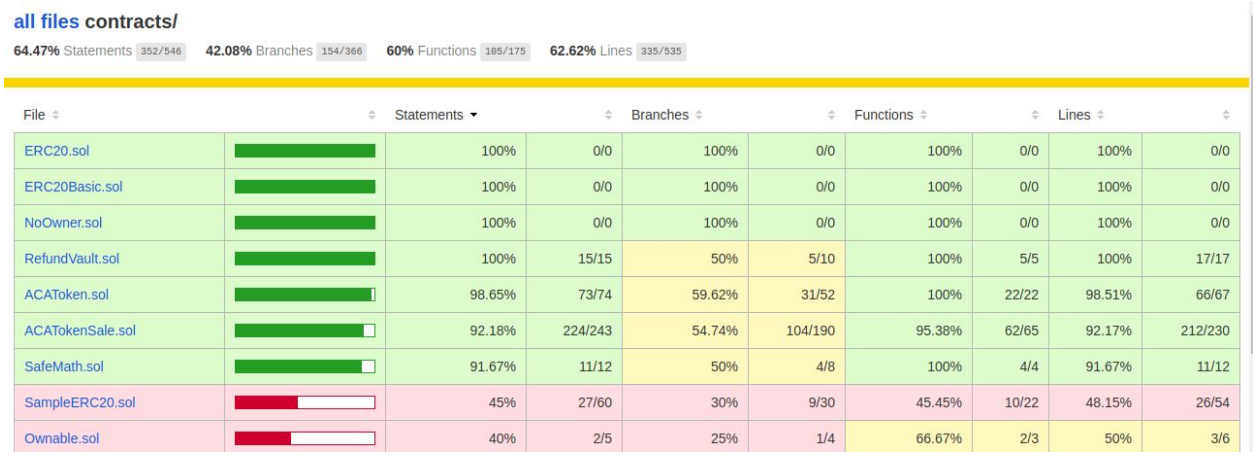
✓ Should Not be able to set capacity of a stage equal to hardcap (192ms)

Final Result of Test:

✓ 97 Passing (16s) PASSED

❌ 0 Failed

Coverage Report :



all files contracts/

64.47% Statements 352/546 42.08% Branches 154/366 60% Functions 105/175 62.62% Lines 335/535

File		Statements		Branches		Functions		Lines	
ERC20.sol		100%	0/0	100%	0/0	100%	0/0	100%	0/0
ERC20Basic.sol		100%	0/0	100%	0/0	100%	0/0	100%	0/0
NoOwner.sol		100%	0/0	100%	0/0	100%	0/0	100%	0/0
RefundVault.sol		100%	15/15	50%	5/10	100%	5/5	100%	17/17
ACAToken.sol		98.65%	73/74	59.62%	31/52	100%	22/22	98.51%	66/67
ACATokenSale.sol		92.18%	224/243	54.74%	104/190	95.38%	62/65	92.17%	212/230
SafeMath.sol		91.67%	11/12	50%	4/8	100%	4/4	91.67%	11/12
SampleERC20.sol		45%	27/60	30%	9/30	45.45%	10/22	48.15%	26/54
Ownable.sol		40%	2/5	25%	1/4	66.67%	2/3	50%	3/6

SampleERC20 token contract is used to test `emergencyERC20Drain()` Function of token contract, only transfer and balanceOf function is used to test functionality of `emergencyERC20Drain()`.

`Ownable` Contract only covers 40% because `transferOwnership()` function of Ownable contract is overwritten in both token and sale contract.

ACAToken.sol	|98.65	|59.62 |	100	|98.51 |	176 |

ACATokenSale.sol	92.18	54.74	95.38	92.17	... 620,621,623

Implementation Recommendations :

- Use case of smart contract is very well designed and Implemented.
- Smart contract is already deployed on mainnet before Audit, we recommend ACA team to carefully use all the bug containing functions on mainnet as **ACA** team has decided not to redeploy contract.

Comments:

Overall, the code is clearly written, and demonstrates effective use of abstraction, separation of concerns, and modularity. **ACA** development team demonstrated high technical capabilities, both in the design of the architecture and in the implementation.

Most of the issues are fixed by **ACA** Development team or resolved as not an issue for them.

Initial Audit Report : 8th February 2019.

Initial Audit Report feedback by ACA : 12th February 2019.

Final Audit Report (Review of Initial Audit report Fixes) : 12th February 2019