SMART CONTRACT AUDIT (HTTPS://BLOG.COINFABRIK.COM/CATEGORY/SMART-CONTRACTS/SMART-CONTRACT-AUDIT-SMART-CONTRACTS/)

Casper CST Token Sale Security Audit

Ariel Yabo (https://blog.coinfabrik.com/author/ariel-yabo/)

June 21, 2018 (https://blog.coinfabrik.com/smart-contracts/casper-cst-token-sale-security-audit/)

Coinfabrik has been hired to audit the contracts for the Casper Token sale. Firstly, we will provide a summary of our discoveries and secondly, we will show the details of our findings.

Contents

Summary

The contracts audited are from the *presale* repository at https://github.com/Casper-dev/presale (https://github.com/Casper-dev/presale). The audit is based on the commit *3c66514423277c39bea26e62a7de47d51d712108* from branch *feat/presale*. This audit was updated to reflect the changes done on commit *8a702734c70e335661d21d04b9c995748d7b27b8* on the same branch.

The audited contracts are:

- contracts/AdviserCasperToken.sol: An adviser token which can be converted directly to CST
- contracts/CasperToken.sol: The token itself

The Casper token sale will have two phases: a presale one where each token will be sold at a fixed rate of $0.12 USD per token, and a sale one where they will be sold at $0.16 USD. There also is an adviser token sale contract which will be able to be exchanged 1:1 with the main token.

The following analyses were performed:

- Misuse of the different call methods: call.value(), send() and transfer().
- Integer rounding errors, overflow, underflow and related usage of SafeMath functions.
- Old compiler version pragmas.
- Race conditions such as reentrancy attacks or front-running.
- Misuse of block timestamps, assuming anything other than them being strictly increasing.
- Contract soft locking attacks (DoS).
- Potential gas cost of functions being over the gas limit.
- Missing function qualifiers and their misuse.
- Fallback functions with a higher gas cost than the one that a transfer or send call allow.
- Fraudulent or erroneous code.
- Code and contract interaction complexity.
- Wrong or missing error handling.
- Overuse of transfers in a single transaction instead of using withdrawal patterns.
- Insufficient analysis of the function input requirements.

Detailed findings

Minor severity

Possible overflow in purchase functions

In CasperToken.sol, there are lines in the purchase functions where a value is set with an addition, but these are unprotected against overflow. One of those is only callable by the contract owner, while the other one is public, in the functions *purchaseWithBTC* and *purchaseWithPromoter*. The lines which could be rewritten are the following:

```
function purchaseWithPromoter(address _to, address _ref) payable public {

require(now >= presaleStartTime && now <= crowdsaleEndTime);

uint _wei = msg.value;

 uint cst;

ethSent[msg.sender] += _wei;

 ethSold += _wei;

 //...

}
```

And:

```
function purchaseWithBTC(address _to, uint _satoshi, uint _wei) public {

require(msg.sender == admin || msg.sender == director || msg.sender == owner);

require(now >= presaleStartTime && now <= crowdsaleEndTime);

ethSold += _wei;

 uint cst;

 // ...

}
```

While the above was fixed in the latest commits, we've found the same possibility in the withdrawTeam function, in the latest commit., in the following:

```
function withdrawTeam() public {

 require(now >= teamETHUnlock1);

 uint amount = 0;

 if (now < teamETHUnlock2) {

 amount = teamETH1;

 teamETH1 = 0;

 } else if (now < teamETHUnlock3) {

 amount = teamETH1 + teamETH2;

 teamETH1 = 0;

 teamETH2 = 0;

} else {

amount = teamETH1 + teamETH2 + teamETH3;

 teamETH1 = 0;

 teamETH2 = 0;

 teamETH3 = 0;
 }

//...
```

Enhancements

Consistent use of SafeMath

In some parts of the code we've found lack of usage of SafeMath. We advise the use of the aforementioned code library in lines like:

```
 cst = _satoshi.mul(btcRate.mul(10000)) / 12;
```

By forcing the use of such a code library the code is guaranteed to avoid situations like overflow.

Use of modern solidity compiler pragmas

The contracts provided use the following compiler pragmas:

```
pragma solidity ^0.4.19;
```

(https://blog.coinfabrik.com/)

We strongly suggest updating the compiler version, so as to avoid any possible legibility issues, since newer compiler versions include features such as distinguishing the constructor and event emission with keywords, which means the legibility of the contract is improved against possible future developments.

(https://blog.coinfabrik.com/)

This observation has been fixed in the last commit sent to us.

Use of modifiers for commonly require clauses

The contract uses require clauses mixed in the function code. In general, the contract's legibility could be improved by putting frequently used clauses in modifiers, by typing:

```
modifier onlyOwnerAndAdmin() {

    require(msg.sender == owner || msg.sender == admin);

    _;

}
```

```
modifier onlyOwnerAndDirector(){
    require(msg.sender == owner || msg.sender == director);

    _;
}
```

This issue has been fixed in the last commit, save for the redundant use in setMaxRate.

Conclusion

We found the contracts to be simple and straightforward and have a fair amount of documentation. Some minor issues have been found, but it is unlikely they could cause critical security problems in the future. After contacting Casper development team, these minor issues were fixed in the subsequent commits.

Do you want to know what is Coinfabrik Auditing Process?

Check our A-Z Smart Contract Audit Guide (https://blog.coinfabrik.com/smart-contract-audits-ultimate-security-guide/) or you could request a quote (https://www.coinfabrik.com#contact_us) for your project.

Related Posts

(https://blog.coinfabrik.com/smart-

Theta Token Sale Security Audit

contracts/tetha-token-sale-security-audit/)

(https://blog.coinfabrik.com/smart-contracts/tetha-token-sale-security-audit/)

(https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/)

Coinfabrik was asked to audit the contracts for the Theta Token sale. In the first...

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/dmtoken-token-sale-smart-contract-audit/)

DMToken Token Sale Smart Contract Audit (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/dmtoken-token-sale-smart-contract-audit/)

Coinfabrik was hired to audit the contract in terms of its security. First of all,...

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/flixxo-token-sale-audit/)

Flixxo Token Sale Security Audit (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/flixxo-token-sale-audit/)

Coinfabrik's smart contracts auditing team was asked to audit the contracts for the Flixxo Token...

(https://blog.coinfabrik.com/smart-contracts/tetha-token-

Theta Token Sale Security Audit (https://blog.coinfabrik.com/smart-contracts/tetha-token-sale-security-audit/)

sale-security-audit/)

Coinfabrik was asked to audit the contracts for the Theta Token sale. In the first…

(https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/)

Tags:

casper api security audits (https://blog.coinfabrik.com/tag/casper-api-security-audits/)

casper cst api descentralized cloud storage security audit (https://blog.coinfabrik.com/tag/casper-cst-api-descentralized-cloud-storage-security-audit/)

SHARE ON

(https://www.facebook.com/sharer/sharer.php?u=https://blog.coinfabrik.com/smart-contracts/casper-cst-token-sale-security-audit/)
(https://twitter.com/intent/tweet?text=Casper%20CST%20Token%20Sale%20Security%20Audit&url=https://blog.coinfabrik.com/smart-contracts/casper-cst-token-sale-security-audit/)
(https://pinterest.com/pin/create/button/?url=&media=https://blog.coinfabrik.com/wp-content/uploads/2018/06/casper-logo.png&description=Casper+CST+Token+Sale+Security+Audit)
(https://www.linkedin.com/shareArticle?mini=true&url=https://blog.coinfabrik.com/smart-contracts/casper-cst-token-sale-security-audit/&title=Casper%20CST%20Token%20Sale%20Security%20Audit&source=CoinFabrik%20Blog)

← PREVIOUS ARTICLE

DreamTeam Token Audit (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/dreamteam-token-audit/)

NEXT ARTICLE →

Stasis Token Smart Contract Audit (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/stasis-token-smart-contract-audit/)

You may also like

(https://blog.coinfabrik.com/smart-contracts/magic-bridge-audit/)

Magic Bridge Audit (https://blog.coinfabrik.com/smart-contracts/magic-bridge-audit/)

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/mintingfactoryv2-baseupgradablemarketplace-kodav3upgradablegatedmarketplace/)

2 months ago

Smart Contract Audit (https://blog.coinfabrik.com/category/smart-contracts/smart-contract-audit-smart-contracts/)

MintingFactoryV2, BaseUpgradableMarketplace & KODAV3UpgradableGatedMarketplace (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/mintingfactoryv2-

baseupgradablemarketplace-kodav3upgradablegatedmarketplace/)
(https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/feed/)

(https://ar.linkedin.com/company/coinfabrik)

(https://twitter.com/coinfabrik)

(https://www.youtube.com/channel/UC2GmjCr7aEz-il31kqOy9aw)

(https://www.facebook.com/CoinFabrik/)

(https://www.reddit.com/r/CoinFabrik/)

(https://github.com/coinfabrik)