C.R.E.A.M. FINANCE COMPOUND PROTOCOL SMART CONTRACT AUDIT

May 04, 2021

MixBytes()

CONTENTS

1. INTRODUCTION

1.1 DISCLAIMER

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Yearn. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 PROJECT OVERVIEW

C.R.E.A.M. Finance is a project that allow users lending and borrowing via using special ERC-20 tokens named CToken. Each Ctoken generates it's own market which goverment by Comptroller smart contract. C.R.E.A.M. Finance is fork of well known Compound project. In the second version of C.R.E.A.M. Finance project developers added oppurtuninty to administrator of project to set supply and borrow caps for specific money market and for all markets in general.

1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 "Blind" audit includes:
> Manual code study
> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document
> Cross check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level
Provide the client with an interim report

05 Bug fixing & re-check.
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.4 EXECUTIVE SUMMARY

C.R.E.A.M. bridges liquidity across underserved assets by providing algorithmic money markets to these underserved assets. Users can supply any supported assets and use these supplied assets as collateral to borrow any other supported assets. C.R.E.A.M. has launched on Ethereum and Binance Smart Chain.

1.5 PROJECT DASHBOARD

Client	Yearn
Audit name	C.R.E.A.M. Finance Compound Protocol
Initial version	23a4ae93adc70334553f5a83429a4e967c1eefaa
Final version	566deba52a13dbc33e20395946b8e0c99932ab9f
SLOC	1178
Date	2021-04-01 - 2021-05-04
Auditors engaged	3 auditors

FILES LISTING

CCollateralCapErc20.sol	CCollateralCapErc20.sol
CCollateralCapErc20Delegate.sol	CCollateralCapErc20De...
CTokenInterfaces.sol	CTokenInterfaces.sol
Comptroller.sol	Comptroller.sol
ComptrollerStorage.sol	ComptrollerStorage.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	10
Comment	19

CONCLUSION

Smart contract has been audited and several suspicious places were found. During audit no critical or major issues were identified. Several issues were marked as warnings and comments. After working on audit report all issues were fixed or acknowledged(if the issue is not critical) by client, so contracts assumed as secure to use according our security criteria.Final commit identifier with all fixes: `566deba52a13dbc33e20395946b8e0c99932ab9f`

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	Incorrect first borrow for user
File	Comptroller.sol
Severity	Warning
Status	Fixed at f634f8fc

DESCRIPTION

In current version of smart contract if user wasn't registered before, but has balance of `CToken > 0`, then borrowing will always fail, until user is registered his collateral.
Comptroller.sol#L366

RECOMMENDATION

We recommend to invoke `registerCollateral` function after comptroller added user to market:

```
Error err = addToMarketInternal(CToken(msg.sender), borrower);
if (err != Error.NO_ERROR) {
    return uint(err);
}
CCollateralCapErc20Interface(msg.sender).registerCollateral(borrower);
```

WRN-2	Possible excess of gas limit
File	Comptroller.sol
Severity	Warning
Status	No issue

DESCRIPTION

In following function it is worth to add check of how much gas left.
Comptroller.sol#L1266

RECOMMENDATION

We recommend add following check in head of `for` loop:

```
1  if (gasleft < 40000) { return;}
```

WRN-3	Possible incorrect tokens redeeming
File	CCollateralCapErc20.sol
Severity	Warning
Status	No issue

DESCRIPTION

In following function tx can fail if user try redeem incorrect amount of tokens: CCollateralCapErc20.sol#L555

RECOMMENDATION

We recommend to add a simple check:

```
require(vars.redeemTokens <= accountTokens[redeemer], "insufficient amount of tokens");
```

CLIENT'S COMMENTARY

We have a check on here: CCollateralCapErc20.sol#L625

WRN-4	Possible flashloan attack
File	CCollateralCapErc20.sol
Severity	Warning
Status	Fixed at **0d23116f**

DESCRIPTION

Current version of flashloan function gives opportunity to any user to set `exchangeRate = 0`, which can be used in another protocols to steal assets: CCollateralCapErc20.sol#L169

RECOMMENDATION

We recommend to transfer tokens to user directly without using `doTransferOut` function.

WRN-5	Incorrect tokens transfer
File	CCollateralCapErc20.sol
Severity	Warning
Status	No issue

DESCRIPTION

Tx fail if user tries to send more tokens than he has:
CCollateralCapErc20.sol#L339

RECOMMENDATION

We recommend to add simple require to save gas for user and give him some information about why tx failed:

```
1  require(accountTokens[src] >= tokens, "Insufficient balance");
```

CLIENT'S COMMENTARY

We have a check here: CCollateralCapErc20.sol#L380

WRN-6	Allowed zero `mintFresh`
File	CCollateralCapErc20.sol
Severity	Warning
Status	Fixed at **f634f8fc**

DESCRIPTION

Zero `mintFresh` doesn't change the state of supply and collateral tokens. It only consumes gas and emits an "empty" event
CCollateralCapErc20.sol#L476

RECOMMENDATION

We recommend quitting `mintFresh` after checks CCollateralCapErc20.sol#L490

```
if (mintAmount == 0) {
    return (uint(Error.NO_ERROR), 0);
}
```

WRN-7	Allowed zero `borrowFresh` and `repayBorrowFresh`
File	CToken.sol
Severity	Warning
Status	Fixed at f634f8fc

DESCRIPTION

Zero `borrowFresh` and `repayBorrowFresh` don't change the state of borrow balance. They only update borrower `borrowIndex`, consume gas and emit an "empty" event
CToken.sol#L437
CToken.sol#L541

RECOMMENDATION

We recommend quitting `borrowFresh` and `repayBorrowFresh` after checks CToken.sol#L453 CToken.sol#L552

```
if (borrowAmount == 0) {
    accountBorrows[borrower].interestIndex = borrowIndex;
    return (uint(Error.NO_ERROR), 0);
}
```

```
if (repayAmount == 0) {
    accountBorrows[borrower].interestIndex = borrowIndex;
    return (uint(Error.NO_ERROR), 0);
}
```

WRN-8	Allowed zero `redeemFresh`
File	CCollateralCapErc20.sol
Severity	Warning
Status	Fixed at **f634f8fc**

DESCRIPTION

`redeemTokensIn` and `redeemAmountIn` may be both zero: CCollateralCapErc20.sol#L559

RECOMMENDATION

Add a check:

- append existing require

```
require(redeemTokensIn == 0 && redeemAmountIn > 0 ||
redeemAmountIn == 0 && redeemTokensIn > 0, "one of redeemTokensIn or redeemAmountIn
must be zero");
```

- or quit function if both are zero

```
if (redeemTokensIn == 0 && redeemAmountIn == 0) {
    return uint(Error.NO_ERROR);
}
```

WRN-9	Allowed zero `seizeInternal`
File	CCollateralCapErc20.sol
Severity	Warning
Status	Fixed at **f634f8fc**

DESCRIPTION

Zero `seizeInternal` doesn't change the state of supply and collateral tokens. It only consumes gas and emits an "empty" event CCollateralCapErc20.sol#L654

RECOMMENDATION

We recommend quitting `seizeInternal` after checks CCollateralCapErc20.sol#L669

```
if (seizeTokens == 0) {
    return uint(Error.NO_ERROR);
}
```

WRN-10	Allowed zero `flashLoan`
File	CCollateralCapErc20.sol
Severity	Warning
Status	Fixed at f634f8fc

DESCRIPTION

Zero `flashLoan` doesn't earn fees. It only consumes gas and emits an "empty" event CCollateralCapErc20.sol#L160

RECOMMENDATION

We recommend adding a check in `flashLoan`

```
require(amount > 0, "flashLoan amount should be greater than zero");
```

2.4 COMMENTS

CMT-1	Possible gas saving
File	Comptroller.sol
Severity	Comment
Status	Fixed at f634f8fc

DESCRIPTION

In following function another variable can be used to save little gas and not to invoke `address` function.
Comptroller.sol#L183

RECOMMENDATION

Use variable `cTokenAddress`

```
1 Market storage marketToExit = markets[cTokenAddress];
```

CMT-2	Remove unused functions
File	Comptroller.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

In current version of smart contract there are some functions which are not currently used.
Comptroller.sol#L273
Comptroller.sol#L410
Comptroller.sol#L459
Comptroller.sol#L526
Comptroller.sol#L591
Comptroller.sol#L644

RECOMMENDATION

We recommend to remove these functions to save gas on deployment.

CLIENT'S COMMENTARY

We can't remove the hook from comptroller since old CToken will still need to access this function. Moreover, the CEther (native token) can't be upgraded so we can't arbitrarily remove functions from comptroller.

CMT-3	Gas saving in copying storage variable
File	Comptroller.sol
Severity	Comment
Status	Fixed at f634f8fc

DESCRIPTION

In following function simple check can be added to save gas:
Comptroller.sol#L213

RECOMMENDATION

We recommend to add following check:

```
if (assetIndex != storedList.length - 1){
    storedList[assetIndex] = storedList[storedList.length - 1];
}
```

CMT-4	Possible gas saving
File	Comptroller.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

In following function adding of return value can save some gas
Comptroller.sol#L1303

RECOMMENDATION

We recommend to change function as follows:

```
function _setCompSpeeds(address[] memory cTokens, uint[] memory speeds) public {
    ...
    uint res = 3;
    if (speeds[i] > 0) {
        res = _initCompState(cTokens[i]);
    }
    CToken cToken = CToken(cTokens[i]);
    Exp memory borrowIndex = Exp({mantissa: cToken.borrowIndex()});
    if (res == 3)
    {
        updateCompSupplyIndex(address(cToken));
        updateCompBorrowIndex(address(cToken), borrowIndex);
    }
    if (res == 1)
    {
        updateCompSupplyIndex(address(cToken));
    }
    if (res == 2)
    {
        updateCompBorrowIndex(address(cToken), borrowIndex);
    }
    ...
}
function _initCompState(address cToken) internal returns (uint) {
    uint res = 0;
    if (compSupplyState[cToken].index == 0 && compSupplyState[cToken].block == 0) {
        res = 1;
        compSupplyState[cToken] = CompMarketState({
            index: compInitialIndex,
            block: safe32(getBlockNumber(), "block number exceeds 32 bits")
        });
    }

    if (compBorrowState[cToken].index == 0 && compBorrowState[cToken].block == 0) {
        res = res + 2;
        compBorrowState[cToken] = CompMarketState({
            index: compInitialIndex,
            block: safe32(getBlockNumber(), "block number exceeds 32 bits")
        });
    }
    return res;
}
```

CLIENT'S COMMENTARY

We don't use COMP (CREAM) rewards anymore and these functions are admin functions.

CMT-5	Pure function definition
File	Comptroller.sol
Severity	Comment
Status	No issue

DESCRIPTION

Following function can be marked `pure`:
Comptroller.sol#L1350

RECOMMENDATION

We recommend to mark this function `pure` to save gas.

CMT-6	Unnecessary usage of safe calculations
File	CCollateralCapErc20.sol
Severity	Comment
Status	Fixed at f634f8fc

DESCRIPTION

In the following function simple substraction can be used to save gas:
CCollateralCapErc20.sol#L353
CCollateralCapErc20.sol#L594

RECOMMENDATION

We recommend using simple substraction:

```
collateralTokens = tokens - bufferTokens;
```

CMT-7	Unnecessary calculation of variable
File	CCollateralCapErc20.sol
Severity	Comment
Status	Fixed at f634f8fc

DESCRIPTION

In the following function calculation of `allowanceNew` can be moved inside `if` block to save gas:
CCollateralCapErc20.sol#L379

RECOMMENDATION

We recommend to move variable into `if` block:

```
if (startingAllowance != uint(-1)) {
    uint allowanceNew = sub_(startingAllowance, tokens);
    transferAllowances[src][spender] = allowanceNew;
}
```

CMT-8	Move redeem hook
File	CCollateralCapErc20.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

In the following function security hook can be moved to save some gas for user: CCollateralCapErc20.sol#L639

RECOMMENDATION

We recommend moved function like this:

```
function redeemFresh(address payable redeemer, uint redeemTokensIn, uint
redeemAmountIn) internal returns (uint) {
    ...
    comptroller.redeemVerify(address(this), redeemer, vars.redeemAmount,
vars.redeemTokens);

    uint bufferTokens = sub_(accountTokens[redeemer],
accountCollateralTokens[redeemer]);
    ...
}
```

CLIENT'S COMMENTARY

We don't want to change the flow of `CToken` (at least not in this PR). Normally, every action (mint, borrow, transfer, redeem, repay) will have two comptroller hooks. One is allowance hook and the other one is verification hook. Verification is put at the rear of the function to act as a defense hook while most of the verification hooks are never used.

CMT-9	Possible liquidity lost
File	CCollateralCapErc20.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

Random user can lose tokens if he invokes following function:
CCollateralCapErc20.sol#L127

RECOMMENDATION

We recommend to add following check:

```
require(msg.sender == admin, "only admin can add reserves");
```

CLIENT'S COMMENTARY

This function is designed for everyone to add reserves.

CMT-10	Necessary initialization
File	Comptroller.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

If `closeFactorMantissa` not initialized then all borrow liquidation always fail: Comptroller.sol#L869

RECOMMENDATION

We recommend to initialize `closeFactorMantissa` in constructor.

CLIENT'S COMMENTARY

The close factor has been set and only admin could adjust the value.

CMT-11	Cap initialization
File	CCollateralCapErc20.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

If `collateralCap` is initialized after smart contract has accumulated assets > `collateralCap`, then users will not be able to increase their collateral for particular market:
CCollateralCapErc20.sol#L135

RECOMMENDATION

We recommend to initialize `collateralCap` in `initialize` function.

CLIENT'S COMMENTARY

We don't expect to put collateral cap on all markets. By default, collateral cap equals to 0 which means no cap. And yes, if the collateral cap is reached, it's by design that no user could increase collateral anymore.

CMT-12	Unreachable code in `getCTokenBalanceInternal`
File	CCollateralCapErc20.sol
Severity	Comment
Status	No issue

DESCRIPTION

Affecting variable `accountTokens[account]` always makes `isCollateralTokenInit[account]` true, so `accountCollateralTokens[account]` will be returned. Otherwise balance is 0
CCollateralCapErc20.sol#L408

RECOMMENDATION

We recommend to remove unreachable part

CLIENT'S COMMENTARY

This one is a little bit tricky. Basically you are right about affecting variable `accountTokens[account]` always makes `isCollateralTokenInit[account]` `true`. However, every market has a function called `getAccountSnapshot` to show a specific user's account snapashot. It contains the `CToken` balance of the given user. When a market is upgraded to the `CCollateralCap` version, a user might have `CToken` balance but not initialize its collateral token.
There is one function called `getHypotheticalAccountLiquidityInternal` in comptroller that will call function `getAccountSnapshot` When calculating the account liquidity, the comptroller will iterate every asset in `AccountAssets` to check `CToken` balance. Every market in `ccountAssets` is considered 'entered' by the user, so if the collateral token is not initialized, it should return `AccountTokens[account]`.

CMT-13	Unlimited `liquidationIncentiveMantissa` and `closeFactorMantissa`
File	ComptrollerG1.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

There are max and min limits for `liquidationIncentiveMantissa` and `closeFactorMantissa` in `ComptrollerG1.sol` but not in `Comptroller.sol`
ComptrollerG1.sol#L83-L96

RECOMMENDATION

We recommend adding limits and checks
Comptroller.sol#L78

```
// closeFactorMantissa must be strictly greater than this value
uint constant closeFactorMinMantissa = 5e16; // 0.05

// closeFactorMantissa must not exceed this value
uint constant closeFactorMaxMantissa = 9e17; // 0.9

// liquidationIncentiveMantissa must be no less than this value
uint constant liquidationIncentiveMinMantissa = mantissaOne;

// liquidationIncentiveMantissa must be no greater than this value
uint constant liquidationIncentiveMaxMantissa = 15e17; // 1.5
```

Comptroller.sol#L930

```
Exp memory newLiquidationIncentive = Exp({mantissa:
newLiquidationIncentiveMantissa});
Exp memory minLiquidationIncentive = Exp({mantissa:
liquidationIncentiveMinMantissa});
if (lessThanExp(newLiquidationIncentive, minLiquidationIncentive)) {
    return fail(Error.INVALID_LIQUIDATION_INCENTIVE,
FailureInfo.SET_LIQUIDATION_INCENTIVE_VALIDATION);
}

Exp memory maxLiquidationIncentive = Exp({mantissa:
liquidationIncentiveMaxMantissa});
if (lessThanExp(maxLiquidationIncentive, newLiquidationIncentive)) {
    return fail(Error.INVALID_LIQUIDATION_INCENTIVE,
FailureInfo.SET_LIQUIDATION_INCENTIVE_VALIDATION);
}
```

Comptroller.sol#L869

```
Exp memory newCloseFactorExp = Exp({mantissa: newCloseFactorMantissa});
Exp memory lowLimit = Exp({mantissa: closeFactorMinMantissa});
if (lessThanOrEqualExp(newCloseFactorExp, lowLimit)) {
    return fail(Error.INVALID_CLOSE_FACTOR, FailureInfo.SET_CLOSE_FACTOR_VALIDATION);
}

Exp memory highLimit = Exp({mantissa: closeFactorMaxMantissa});
if (lessThanExp(highLimit, newCloseFactorExp)) {
    return fail(Error.INVALID_CLOSE_FACTOR, FailureInfo.SET_CLOSE_FACTOR_VALIDATION);
}
```

CLIENT'S COMMENTARY

These functions are admin functions and we barely call them. We removed the check to decrease the contract size as it was too large to deploy.

CMT-14	New market borrow or supply cap may block borrowing or minting
File	Comptroller.sol
Severity	Comment
Status	Fixed at f634f8fc

DESCRIPTION

If market borrow balance or supply already exceeds new cap when setting it may block borrowing or minting for market
Comptroller.sol#L1005
Comptroller.sol#L1025

RECOMMENDATION

We recommend adding comments about it in the code

CMT-15	Unused values/statements in functions
File	Comptroller.sol Comptroller.sol CCollateralCapErc20Delegate.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

In `Comptroller.sol` at lines
Comptroller.sol#L239

Comptroller.sol#L323
`minter`, `cToken` and `redeemer` values are unused
In `CCollateralCapErc20Delegate.sol` at line CCollateralCapErc20Delegate.sol#L22
the `data` value is initialized but never used
at line
CCollateralCapErc20Delegate.sol#L25
the `if` statement will never be executed

RECOMMENDATION

It is recommended to remove redundant code to avoid confusion and increase clarity and readability of the code

CLIENT'S COMMENTARY

These unused values and statements are used to silence the compiler.

CMT-16	Unused functions wrapped in commentaries
File	Comptroller.sol CCollateralCapErc20.sol
Severity	Comment
Status	No issue

DESCRIPTION

In `Comptroller.sol` at line Comptroller.sol#L323
`CCollateralCapErc20.sol` at lines CCollateralCapErc20.sol#L398
CCollateralCapErc20.sol#L689
CCollateralCapErc20.sol#L536

functions are stated in commentaries, but unused

RECOMMENDATION

We recommend to remove these unnecessary commentaries to improve clarity and readability of the code

CLIENT'S COMMENTARY

`xxxVerify` functions in comptroller are still used by some `CTokens`.

CMT-17	Explicit statement of `uint256` values
File	CCollateralCapErc20.sol Comptroller.sol CTokenInterfaces.sol ComptrollerStorage.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

In `CCollateralCapErc20.sol`, `CTokenInterfaces.sol`, `Comptroller.sol`, `ComptrollerStorage.sol` all instances of `uint` values are not stated as `uint256` explicitly

RECOMMENDATION

We recommend to expicitly state `uint` values as `uint256` to increase clarity and readability of the code

CMT-18	Unclear commentary
File	CCollateralCapErc20.sol
Severity	Comment
Status	Fixed at f634f8fc

DESCRIPTION

At line CCollateralCapErc20.sol#L552 the `@param` comment states that only one of `redeemTokensIn` or `redeemAmountIn` values may be non-zero, however, it may be confusing to acknowledge

RECOMMENDATION

It is recommended to rewrite the comment to explicitly state that `redeemTokensIn` and `redeemAmountIn` can both be zero values at the same time

CMT-19	Possible value truncation issues
File	CCollateralCapErc20.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

In `CCollateralCapErc20.sol` at line CCollateralCapErc20.sol#L513 the `div_ScalarByExpTruncate()` may round the result down to the next nearest integer if it is calculated to be a non-integer number of cToken units, sufficiently small loans may be affected, however, the loss should never be more than one indivisible unit of a token used

RECOMMENDATION

This is a relatively unavoidable error which appears due to to EVM operation result. It should be acknowledged by suppliers with extremely small amount of tokens

CLIENT'S COMMENTARY

It's a known issue that only effects extremely small amount of tokens.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build open-source solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

BLOCKCHAINS

 Ethereum

 Cosmos

 EOS

 Substrate

TECH STACK

 Python

 Solidity

 Rust

 C++

CONTACTS

 https://github.com/mixbytes/audits_public

 https://mixbytes.io/

 hello@mixbytes.io

 https://t.me/MixBytes

 https://twitter.com/mixbytes