DeHacker

Security Assessment

TARS Protocol

June 10th, 2022

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.
Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

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

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	TARS Protocol
Platform	BSC
website	https://tars.pro/
Type	DeFi Infrastructure
Language	Solidity

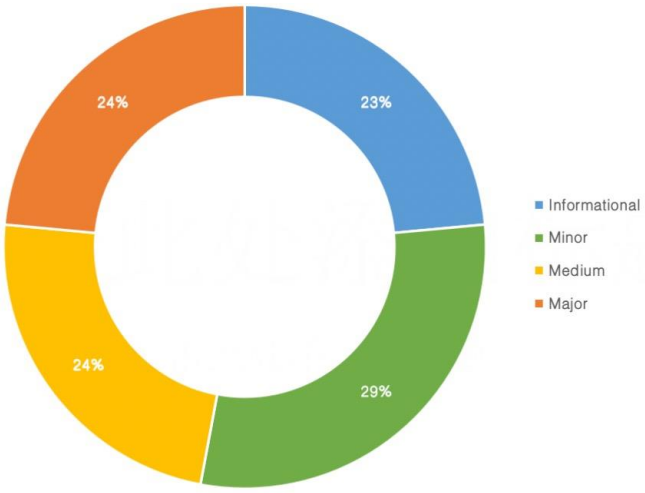
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	4	0	0	1	0	3
Medium	4	0	0	1	0	3
Minor	5	0	0	3	1	1
Informational	4	0	0	3	0	1
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
CSI	CoinSalesIDO.sol	cd8d2415a75364d4473c7774403ee697cf6b575094861dc53ebc5d64bb772eee
ALT	AssemblyLine.sol	52b0b2ab0e134dad04dfab7834b827f340fc8b03b9c5e4b5bf3b0ff06f81168e
VNT	token/VoucherNft.sol	d6f6f43ab261c2986193396dc988353001280a002b730f67998ccca341470247
FID	FactoryIDO.sol	7a5dec9eae43ca4dcf43de9ac53397e9f6ad143847507a228b9be08ba5475556
ITP	Insurance.sol	75c2b0a6ff5cd277083e605249fc4973dcb18c8da4961edfa05cb53a1c796cc0

Findings



ID	Title	Category	Severity	Status
GLOBAL-01	Centralization Related Risks	Centralization / Privilege	major	Acknowledged
CSI-01	Missing Validation	Logical Issue	Major	Resolved
CSI-02	Missing Input Validation	Volatile Code	Minor	Resolved
CSI-03	Lack Of Reasonable Boundary	Volatile Code	Minor	Partially Resolved
CSI-04	Missing Input Validation	Volatile Code	Minor	Acknowledged
CSI-05	LockWarehouse Can Be Added AfterUser Investment	Logical Issue	Minor	Acknowledged
CSI-06	Condition Can Be Simplified	Volatile Code	Informational	Resolved
FID-01	Incorrect Logic In FunctioncustomProductionIDO	Logical Issue	Medium	Acknowledged
ITP-01	Possible Loss Of Accuracy In FunctionjoinInsurance	Logical Issue	Medium	Resolved
ITP-02	Incorrect Operator Used	Logical Issue	Medium	Resolved
ITP-03	Inadequate Validation	Logical Issue	Medium	Resolved
TPC-01	Permission Issues For Super Admin	Logical Issue	Major	Resolved

TPC-02	Missing Input Validation	Volatile Code	Minor	Acknowledged
TPC-03	Unlocked Compiler Version	Language Specific	Informational	Acknowledged
TPC-04	Missing Emit Events	Coding Style	Informational	Acknowledged
TPC-05	Variables That Could Be Declared AsImmutable	Gas Optimization	Informational	Acknowledged
VNT-01	Missing Creation In FunctionqueryUserAllNft	Language Specific	Major	Resolved

Major

GLOBAL-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	Major		Acknowledged

Description

In the contract CoinSalesIDO, the role superAdmin has the authority over the following function:

1. function adminSafeTransfer()
2. function updateInsuranceAddress()
3. function updateIsPrivate()
4. function updateTime()
5. function updatePoolQuota()
6. function updateQuantitySold()
7. function updateWhiteListQuota()
8. function addWhiteList()
9. function addLockWarehouse()
10. function updateSellingToken()
11. function updateSellingTokenName()
12. function cancelProject()
13. function identification()
14. function transferCoin()

In the contract CoinSalesIDO, the role secondParty has the authority over the following function:

1. function addWhiteList()
2. function addLockWarehouse()
3. function updateSellingToken()

4. function updateSellingTokenName()
5. function cancelProject()
6. function secondPartyClaim()

In the contract CoinSalesIDO, the role firstParty has the authority over the following function:

1. function identification()
2. function transferCoin()
3. function firstPartyClaim()

In the contract FactoryIDO, the role superAdmin has the authority over the following function:

1. function adminSafeTransfer()
2. function updateAdmin()
3. function updateInsurance()
4. function updateAssemblyLine()
5. function addAssemblyLine()
6. function deleteAssemblyLine()

In the contract FactoryIDO , the role voucherBoss has the authority over the following function:

1. function updateProxyInvestment()
2. function foundCoinSalesIdo()
3. function customProductionIDO()

In the contract FactoryIDO , the role operator has the authority over the following function:

1. function updatePayee()
2. function updateBusinessToken()
3. function updatePrice()
4. function updateClosePrice()
5. function addFundRaisingToken()
6. function deleteFundRaisingToken()
7. function poolAuthentication()

In the contract Insurance, the role superAdmin has the authority over the following function:

1. function adminSafeTransfer()
2. function updateAdmin()

3. function setFactoryIdoAddress()

In the contract Insurance, the role operator has the authority over the following function:

1. function adminAllSafeTransfer()
2. function updateInsuranceRate()
3. function settlementOfClaims()

In the contract VoucherNft, the role operator has the authority over the following function:

1. function setNftUri()
2. function mintNft()

Any compromise to the accounts may allow a hacker to take advantage of this authority and causes security problems.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a differentlevel in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination mitigate by delaying the sensitive operation and avoiding asingle point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to theprivate key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information withthe public audience.

Long Term:

Timelock and DAO, the combination, mitigate by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The team response: Some permissions have been cancelled, and the remaining permissions are due to business needs.

CSI-01 | Missing Validation

Category	Severity	Location	Status
Logical Issue	Major	CoinSalesIDO.sol: 418~427, 429~431	Resolved

Description

The sellingToken can be changed after startTime. So sellingToken can be changed after the user hamade a purchase. The user may not get the sellingToken that he wanted to buy.

Recommendation

We recommend adding a check to ensure the sellingToken can only be changed before startTime.

Alleviation

The development team has resolved this issue in commit a38539d19c200c900888315ad35dcd9acf938470.

TPC-01 | Permission Issues For Super Admin

Category	Severity	Location	Status
Logical Issue	Major	FactoryIDO.sol: 150~154; Insurance.sol: 111~116	Resolved

Description

The function updateAdmin can only be called by the superAdmin . The functions removeAuth and addAuthcan only be called by the owner . But the superAdmin is set in constructor , maybe not the owner . So the superAdmin may not have the permission to call the functions removeAuth and addAuth .

Recommendation

We recommend assigning owner to superAdmin in constructor.

Alleviation

The development team has resolved this issue in commit a38539d19c200c900888315ad35dcd9acf938470.

VNT-01 | Missing Creation In Function queryUserAllNft

Category	Severity	Location	Status
Logical Issue	Major	token/VoucherNft.sol: 74~79	Resolved

Description

In function `queryUserAllNft` , `nftArray` is not created. When this method be called there will be an error.

Recommendation

Consider modifying as below:

```
    nftArray = new NftInfo[](_ids.length);
    for (uint256 i = 0; i < _ids.length; i++) {
        nftArray[i] = nftInfo[_ids[i]];
    }
    return nftArray;
```

Alleviation

The development team has resolved this issue in commit `a38539d19c200c900888315ad35dcd9acf938470`.

Medium

FID-01 | Incorrect Logic In Function customProductionIDO

Category	Severity	Location	Status
Logical Issue	Medium	CoinSalesIDO.sol: 403~435	Acknowledged

Description

function foundCoinSalesIdo :

```
1     _addressArray[3] = msg.sender;
2     _addressArray[4] = superAdmin;
3     _addressArray[5] = insurance;
4     _addressArray[6] = address(this);
5
6     require(isInsuranceToken(_addressArray[1]), "Illegal token!");
7     _intArray[9] = uint256(ERC20(_addressArray[1]).decimals());
```

function customProductionIDO :

```
1     _addressArray[0] = msg.sender;
2     _addressArray[1] = superAdmin;
3     _addressArray[2] = insurance;
4     _addressArray[3] = address(this);
5     _addressArray[4] = _assemblyLine;
6
7     require(isInsuranceToken(_addressArray[5]), "Illegal token!");
8     _intArray[0] = uint256(ERC20(_addressArray[5]).decimals());
```

The functions foundCoinSalesIdo and customProductionIDO have similar features. So the parameters passed to assemblyLine should be the same. But the real parameters shown above are different. The parameters in customProductionIDO are incorrect. Please correct the parameters.

Additionally, The newly created ido is not added to the firstPartyPoolArray in function customProductionIDO , like in foundCoinSalesIdo .

Recommendation

We recommend to correct the logic in function customProductionIDO .

Alleviation

The team response: customProductionIDO is a later reserved extension function and does not correspond to the productionIdo function in AssemblyLine , it is a reserved interface for the new AssemblyLine later.

ITP-01 | Possible Loss Of Accuracy In Function joinInsurance

Category	Severity	Location	Status
Logical Issue	Medium	Insurance.sol: 169, 172	Resolved

Description

Variables totalInsuredAmount , userInsuredTotal are without precision. The default value of insuranceRate is 1%. When the investment amount is not an integer multiple of 100, totalInsuredAmount and userInsuredTotal will have a loss of precision.

For example: _investedAmount = 99 * 10 ** decimals; _amount = 0.99 * 10 ** decimals; _amount.div(10 **decimals) = 0;

Recommendation

We recommend retaining the precision of totalInsuredAmount and userInsuredTotal.

Alleviation

The development team has resolved this issue in commit a38539d19c200c900888315ad35dcd9acf938470.

ITP-02 | Incorrect Operator Used

Category	Severity	Location	Status
Logical Issue	Medium	Insurance.sol: 169, 172	Resolved

Description

```
1      totalInsuredAmount = totalInsuredAmount.add(_amount.div(10 * decimals));
2
3      userInsuredTotal[_user] = userInsuredTotal[_user].add(_amount.div(10 *
decimals));
```

The operator in Line169 and Line172 should be **, but not *.

Recommendation

We recommend using the correct operator.

Alleviation

The development team has resolved this issue in commit `a38539d19c200c900888315ad35dcd9acf938470`.

ITP-03 | Inadequate Validation

Category	Severity	Location	Status
Logical Issue	Medium	Insurance.sol: 150~179	Resolved

Description

The validation in `Insurance.joinInsurance` is not sufficient. Users can forge their own contracts into `CoinSalesIDO` to mint themselves NFT.

Recommendation

We recommend using of more adequate calibration `FactoryIDO.isVoucherIdo(msg.sender)` to ensure the `CoinSalesIDO` is a valid one.

Alleviation

The development team has resolved this issue in commit `a38539d19c200c900888315ad35dcd9acf938470`.

Minor

CSI-02 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	CoinSalesIDO.sol: 264~265	Resolved

Description

The `userFloor` means the minimum amount of tokens one user can buy. The `userQuota` means `themaximum` amount of tokens one user can buy. So the `userFloor` should be less than `userQuota`.

Recommendation

Consider adding a check as below:

```
1     require(intArr[7] > intArr[8], "userQuota should be greater than userFloor!");
```

Alleviation

The development team has resolved this issue in commit `a38539d19c200c900888315ad35dcd9acf938470`.

CSI-03 | Lack Of Reasonable Boundary

Category	Severity	Location	Status
Volatile Code	Minor	CoinSalesIDO.sol: 277, 372~416	Partially Resolved

Description

1. The variable yield means the percentage a second party can get from a CoinSalesIDO . Its current range values is yield<=100 . We think its range of values should be more reasonable.

2. If _isLockWarehouse == 2 , the selling token may be locked for a certain period of time. This time should be within a reasonable range.

Recommendation

We recommend adding reasonable upper and lower boundaries to all the configuration variables.

Alleviation

1. The range of values changed to 0<yield<100 .
2. The final unlock time limit is 10 years after the current time.

Code was applied in commit a38539d19c200c900888315ad35dcd9acf938470

CSI-04 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	CoinSalesIDO.sol: 317~335	Acknowledged

Description

According to the logic, `startTime` should be less than `endTime` , `endTime` should be less than `claimTime` and `closeTime` . So there need a check before updating the variables `startTime` , `endTime` , `claimTime` and `closeTime` .

Recommendation

Consider adding a check to ensure the variables mentioned above have a reasonable value.

Alleviation

The team response: Because of the needs of business needs, there is no need to solve them.

CSI-05 | LockWarehouse Can Be Added After User Investment

Category	Severity	Location	Status
Logical Issue	Minor	CoinSalesIDO.sol: 372~416	Acknowledged

Description

The LockWarehouse can be added after user investment, so users may not know the lockdown program when they participate exchange.

Recommendation

It's best to add LockWarehouse before users participate exchange.

Alleviation

The team response: Because of the needs of business needs, there is no need to solve them.

TPC-02 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	AssemblyLine.sol: 20, 25; CoinSalesIDO.sol: 420; FactoryIDO.sol: 99, 102~106, 152; Insurance.sol: 61~63	Acknowledged

Description

The given input is missing the check for the non-zero address.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:For example:

```
1     require(_lineLeader != address(0), "lineLeader can not be zero address!");
```

Alleviation

The team response: No need to resolve.

Informational

CSI-06 | Condition Can Be Simplified

Category	Severity	Location	Status
Volatile Code	Informational	CoinSalesIDO.sol: 481	Resolved

Description

In Line481, `_amount <= investment.add(_amount)` . So Line481 can be simplied.

Recommendation

Consider adding a check as below:

```
1     require(investment.add(_amount) <= quota, "Err_46");
```

Alleviation

The development team has resolved this issue in commit `a38539d19c200c900888315ad35dcd9acf938470`.

TPC-03 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	AssemblyLine.sol: 2; token/VoucherNft.sol: 2; CoinSalesIDO.sol: 2; Insurance.sol: 2; FactoryIDO.sol: 2	Acknowledged

Description

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to different compiler versions. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract canbe compiled at. For example, for version v0.6.2 the contract should contain the following line:

```
pragma solidity 0.6.2;
```

Alleviation

The team response: No need to resolve.

TPC-04 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	AssemblyLine.sol: 23, 28; CoinSalesIDO.sol: 228, 300, 308, 313, 317, 337, 341, 345, 356, 372, 418, 429, 433, 438, 452, 515, 530; Insurance.sol: 88, 111, 118; FactoryIDO.sol: 142, 150, 176, 181, 215, 222, 252, 295	Acknowledged

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

The team response: No need to resolve.

TPC-05 | Variables That Could Be Declared An Immutable

Category	Severity	Location	Status
Gas Optimization	Informational	AssemblyLine.sol: 11; CoinSalesIDO.sol: 27; Insurance.sol: 20, 22; FactoryIDO.sol: 19	Acknowledged

Description

The linked variables assigned in the constructor can be declared as `immutable` . Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

Recommendation

We recommend declaring these variables as `immutable`. Please note that the immutable keyword only works in Solidity version `v0.6.5` and up.

Alleviation

The team response: No need to resolve.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style

Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

Ethereum
Cosmos
Eos
Substrate

TECH STACK

Python
Solidity
Rust
c++

CONTACTS

https://dehacker.io
https://twitter.com/dehackerio
https://github.com/dehacker/audits_public
https://t.me/dehackerio
https://blog.dehacker.io/

June 2022