CERTIK

Security Assessment

DOTC

Jun 15th, 2021

Table of Contents

Summary

This report has been prepared for DOTC smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing 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 practices 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.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	DOTC
Description	Decentralized OTC Market
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/DOTCPro/Contracts
Commit	1.576a168519ccb7518204979294d68f292e698e5a 2.8b0297a94cc4c7651ab7b87842fc10f61dabccb2 3.b1785dcd51678fc34456bf35f6bb62000207410e 4.35e656497dc18a48b048fd75f4c8c4f6dd54aab6

Audit Summary

Delivery Date	Jun 15, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Total Issues	50
● Critical	0
● Major	3
● Medium	0
● Minor	12
● Informational	35
● Discussion	0

Audit Scope

ID	file	SHA256 Checksum
DOT	DOTCFactoryDiamond.sol	5d2f764ce588482d2a7af83b4da5e8370459997da5866150760ed0fe580f7bdc
MCK	Migrations.sol	4cbc0c151998d768bdb8bf06bdfcf7957afb5c96548321747820f5a4e98724f4
ACK	defines/dArbit.sol	d91500a8b40712d6c349a21e7182f4a2a51e998ea6f805fea9431a1760201cb8
CCK	defines/dCommon.sol	51c85c9436eef322d1591c556d7448a1917da220c414976ac86d2d6435fa8076
CFC	defines/dCutFacet.sol	0b2873a0f0417b4ce3554ea772692b351ea845f1cddd5a23f3a0d745a3b43fc1
MPC	defines/dMiningPool.sol	7766dac1c8505a90745f1fc56eaf969779a763cf8ae25d6be60e5a88b5782adc
OCK	defines/dOrder.sol	7bde63efc43be27cacceaf4f26427acf98388b2f8174afb7c3b58dca71265388
RCK	defines/dRisk.sol	81cf60ff7770d5156ef27eb711bad72f9b3caf16f8c0bda7eff0891ca581b9a0
SCK	defines/dStaking.sol	61cb8a2064381a8b62b858d23eaaa27d7a3d49f75acf637d642ca16cc4df6a1b
TCK	defines/dTotal.sol	5158e403af1805804bc4b3f029459b0cd204fc307b8617a54717c14471e565b1
UCK	defines/dUser.sol	806cae91fc0c826725d24b873e668b11490e144546f94043569877ecbaf677b9
DOC	facetBase/DOTCFacetBase.sol	abe8fac73ea715f094c439b83c2b278362069002c29e6b8995e7f367a527a925
DCF	facetBase/DiamondCutFacet.sol	3615c0a7ce05cd8276cfa1cff5e4e70b2623b5aad41ccf8f7d50008cef4ed71f
DLF	facetBase/DiamondLoupeFacet.sol	d4669fa7e25b945f49ec723d08d41fe94107eb9e7d1c71c7440bb5976c45ec95
OFB	facetBase/OwnershipFacet.sol	8674021e7b55eb9e618d34b0308e48782dc60b00651df38627f1c396ef9a029c
DOA	facetLogic/DOTCAdOrderFacet.sol	9d7eec0afc73a64bd92d4715d99674caaa9972430b7a8c09a79672b13cfc776e
DOB	facetLogic/DOTCArbitBase.sol	03f89b9f7931173a9f5ff18ba62ab205e6ee9a94fb95d4df0b26a84d243e9a08
DOF	facetLogic/DOTCArbitFacet.sol	3dc8ccfe37a8e1fe6eec6c8960747117e8c2aa4fd780767b5cd11d2c1e5b27dd
DOS	facetLogic/DOTCArbitSettleFacet.sol	3b416e8bcd1b7f9b06a9316de90e7d9fbaffa29885a406855d76a2e03d16b01b

ID	file	SHA256 Checksum
DOL	facetLogic/DOTCCardAribtFacet.sol	83e59fd5a0090b5a15a352ce98648b6cd4e55f85ea533871993182dce48cc6d3
DOE	facetLogic/DOTCExOrderBase.sol	6f20372ec8bcc9a2ec05d2c022c5ed7912897f3d0b06ec143e62b0bc28140c67
DOO	facetLogic/DOTCExOrderFacet.sol	9c70e314bf9cd9fdc2d65df6689e6e967117a03b83f99440957e34bf48731976
DOK	facetLogic/DOTCFeeFacet.sol	1c078301a988913d68b16f7529b72adb4673f5970509f2f347daa97ffdd2e90b
DOM	facetLogic/DOTCManageFacet.sol	0b22b520dcc86e4f5ab1d77bb45b685576a87ede98dc8f93a26e39361ed0b5d7
DOP	facetLogic/DOTCMiningFacet.sol	294b9722268477278c0bd9e5ece1499c584688a652f8787d05ba849b8fbbf07b
DTC	facetLogic/DOTCOracleFacet.sol	7f69a8e2f16bda471cbc707ab12b9a569e1428cfad4b63e3037db0e480cfff1b
DOR	facetLogic/DOTCRiskFacet.sol	f55012324dfb2bd186b28b92c6dfff10084dd51b9c966004562186a775106f00
DTS	facetLogic/DOTCStakingFacet.sol	56b80b3fcf2a171dec1ac2c42d170709cac461a4ba336e894d5b289d8d6d6b79
DOU	facetLogic/DOTCUserFacet.sol	1179240207109a53297892fff1cc05fe975e313c41674b051b05512a164e1a34
DTT	governance/DOTCTimelock.sol	3b494700183c7a095f3179130c0f2bed87d5c62d3808cd570c44c2743be0837b
IDO	interfaces/IDOTCFacetBase.sol	a2d018cfb500bd86e6a6915bb8570e5ea8e69f72d3042d24d77f11e3071e10fe
IDT	interfaces/IDOTCFactoryDiamond.sol	b40c0be7f78bc4e4a98073d35140a8e0ebe81a52eb498d03b6f441895ac2962c
IDC	interfaces/IDOTCManageFacet.sol	87c6b8dd2a24879c8ae6d14518ec54628e77d041d47317469decdbca946fed11
IDK	interfaces/IDiamondCut.sol	55ad7ff6cfe097b540db0206d679f60ca27f6ef040927c98c5d63955ddf6e814
IDL	interfaces/IDiamondLoupe.sol	dc13ba04225a87981e34b5490f5a5d461b89d44b791abb556916c21ea16919f8

ID	file	SHA256 Checksum
IER	interfaces/IERC165.sol	bcd90d99170e7cdd8a20b59362ca26045b307dbc3a7902cb51363d4b9a0cc0f0
IEC	interfaces/IERC173.sol	6fc79ae8305e6e051815f56db4a4ef0c7df4adef637495d9907ca06a1b02e4a4
IEK	interfaces/IERC20.sol	72e6781196c6c4124cb2a1280e5f20c8224c5a8b4f69f36a0e03167c69650ecf
IFF	interfaces/IFeeFacet.sol	eada45db7ce4a230d9a02364b4d5aaf7722c15f3f2045f62d7948823c1bfe8c3
ASC	libraries/AppStorage.sol	739537526f55170cb3d5de7036c5f40ca71e4d63605b587ced49e760e6387382
DTL	libraries/DOTCLib.sol	c445612680f17204ff759700e5ec7141f2042dc82170e37e551eb9a591b5ca22
LDC	libraries/LibDiamond.sol	f0c223a7e01de70f916392210ac962ddf72a43cb6153556ed2552a59960e5617
LER	libraries/LibERC20.sol	ee70706bd903e16d31d6a9819023a6da072625bc7ae8692588923b03394674fc
LSC	libraries/LibStrings.sol	454caf2bc3f9a1f1fd6736f3ea67331fa5f2d4c27826101a6deaf2a3f01811ac
DTO	oracle/DOTCOracleRobot.sol	d98e63c749631e8122c0b795dedfa176acaec1a6857f4dca8ff4ffb521dfa470
IDR	oracle/IDOTCOracleRobot.sol	c4e29ecfdb564f73b3bd57614bad943468af4437af468985d9a2716301095d70
BCK	oracle/libraries/Babylonian.sol	4045794dabf740ac9054a743de1d854017acf967492747171f9ccc98454ce0b2
FPC	oracle/libraries/FixedPoint.sol	a6388aa687dff62d74a6ee182dbe6060c301518a05b82633a079a42fd12265b9
IUV	oracle/libraries/IUniswapV2Factory.sol	9295cd590e354c8fec640ff3d7c3b0536eb2c7f543c0f204f69935d9fc461052
IUP	oracle/libraries/IUniswapV2Pair.sol	71ffcf80ed7b6cd38f82c53e0f3f2f3f632ebc85d3dd1428dd3a4836c77c1ad4
UVL	oracle/libraries/UniswapV2Library.sol	48e85b26dfb76bda3c9602896dcba8ba8bf937e45d0c0f6e78c50022d0cbe9ca
UVO	oracle/libraries/UniswapV2OracleLibrary.sol	e23f2072a4ae72917a1c7efe1eefbcedc67762cb21dc5de56a42f3cdbcbe9385
DTK	token/DOTCToken.sol	b5e1718095d20e4832ea660992461bc90f5535db407b129165beb09ceaeeb561
CCP	utils/Closeable.sol	3a9bb149cc8d027be747ea6316c6acd49655c93ba34847598f5fc43675ae8a46
OCP	utils/Ownable.sol	3d3fdd258a8d2dd703b55af22b70add1af8aa989e53269b2bc84c13e77d799ea
RHC	utils/RandomHelper.sol	7d437e92160b4f4443f135c61d88f4564bbceae3a82e12e98bd5a3f8fb9bee7f

ID	file	SHA256 Checksum
SAC	utils/SafeArray.sol	ef2bca8bdea520d1107e84326e4c45c0a965266a911c2e983d8fd9190a252900
SMC	utils/SafeMath.sol	4a990fffa55378be5ddedc575488ef1d1bf35da4df41fc68b523981a54a43fe8
SHC	utils/SignHelper.sol	0ecde818eb77e19ce126798b3da46cd51b381c94dbd035be8bb7d1b6ff588a31

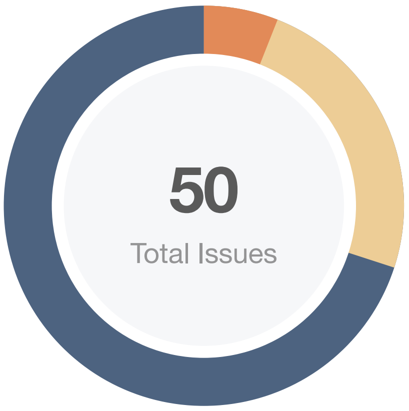
It should be noted that the system design includes a number of economic arguments and assumptions. These were explored to the extent that they clarified the intention of the code base, but we did not audit the mechanism design itself. Note that financial models of blockchain protocols need to be resilient to attacks. It needs to pass simulations and verifications to guarantee the security of the overall protocol. The correctness of the financial model is not in the scope of the audit.

Additionally, as of the date of publishing, the contents of this document reflect the current understanding of known quality and security patterns regarding smart contracts and compilers. Given the size and complexity of the project, the findings detailed here are not to be considered exhaustive, and further testing and auditing are recommended after the issues covered are fixed.

All the contracts use the Diamonds pattern (EIP-2535) and can be upgraded through administrator actions.

Note that the scope of audit only includes the contracts in commit 576a168519ccb7518204979294d68f292e698e5a. We have explored subsequent commits only to help address the issues found in that scope and check if they are fixed while other changes are ignored.

Findings



🟥	**Critical**	**0** (0.00%)
🟧	**Major**	**3** (6.00%)
🟨	**Medium**	**0** (0.00%)
🟨	**Minor**	**12** (24.00%)
🟦	**Informational**	**35** (70.00%)
🟩	**Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
DCF-01	`ds.selectorSlots` Not Updated	Logical Issue	● Major	✓ Resolved
DOA-01	Unused Constant	Logical Issue	● Informational	✓ Resolved
DOA-02	Unnecessary Gas Cost to Call Variable from Structs	Gas Optimization	● Informational	✓ Resolved
DOA-03	Function Name Misspellings	Coding Style	● Informational	✓ Resolved
DOA-04	Check Condition Inconsistent with Message	Logical Issue	● Informational	Partially Resolved
DOB-01	Repetitive Function Implementations	Logical Issue	● Minor	ⓘ Acknowledged
DOC-01	Repetitive Function Implementations	Logical Issue	● Minor	ⓘ Acknowledged
DOC-02	Unnecessary Calculation Out of Bound Risk	Mathematical Operations	● Informational	Partially Resolved
DOC-03	Too Many Digits	Coding Style	● Informational	✓ Resolved
DOC-04	Incorrect Naming Convention Utilization	Coding Style	● Informational	✓ Resolved
DOC-05	Unused Function	Logical Issue	● Minor	✓ Resolved
DOC-06	Redundant Length Getter	Gas Optimization	● Informational	⊗ Declined
DOC-07	Redundant Remove Operation	Gas Optimization	● Informational	ⓘ Acknowledged
DOC-08	Logic Issue When Removing Arbiter	Logical Issue	● Informational	ⓘ Acknowledged

ID	Title	Category	Severity	Status
DOE-01	Check Condition Inconsistent with Message	Logical Issue	● Informational	Partially Resolved
DOF-01	Repetitive Function Implementations	Logical Issue	● Minor	Acknowledged
DOF-02	Redundant Conditional	Logical Issue	● Informational	Resolved
DOF-03	Logical issues in `_clearInvitorSponsor`	Logical Issue	● Informational	Acknowledged
DOK-01	Check Condition Inconsistent with Message	Logical Issue	● Informational	Partially Resolved
DOL-01	Volatile Type Conversion	Volatile Code	● Minor	Resolved
DOL-02	Magic Reward and Margin Rates	Magic Numbers	● Informational	Declined
DOM-01	Inconsistent Getter Function	Logical Issue	● Minor	Resolved
DOM-02	Owner Privileges	**Centralization / Privilege**	● **Informational**	**Acknowledged**
DOO-01	Emit Events Missing `indexed`	Volatile Code	● Informational	Resolved
DOO-02	Magic Reward and Margin Rates	Magic Numbers	● Informational	Declined
DOO-03	Function Name Misspellings	Coding Style	● Informational	Resolved
DOO-04	Check Condition Inconsistent with Message	Logical Issue	● Informational	Partially Resolved
DOP-01	Check-Effect-Interaction Pattern Violation	Control Flow	● Minor	Resolved
DOP-02	DAO Pools Only Use DOTC	Logical Issue	● Informational	Acknowledged
DOR-01	Check-Effect-Interaction Pattern Violation	Control Flow	● Minor	Resolved
DOR-02	DAO Pools Only Use DOTC	Logical Issue	● Informational	Acknowledged
DOS-01	Repetitive Function Implementations	Logical Issue	● Minor	Acknowledged
DOS-02	Magic Reward and Margin Rates	Magic Numbers	● Informational	Declined
DOU-01	Locked Assets Can Not Be Unlocked	Logical Issue	● Informational	Declined
DOU-02	Token Permissibility	Logical Issue	● Minor	Acknowledged

ID	Title	Category	Severity	Status
DTK-01	Unlocked Compiler Version	Language Specific	● Informational	✓ Resolved
DTK-02	Set `constant` to Variables	Logical Issue	● Informational	✓ Resolved
DTK-03	Proper Usage of `require` and `assert` Functions	Coding Style	● Informational	ⓘ Acknowledged
DTK-04	Unused Return Value	Coding Style	● Informational	✓ Resolved
DTK-05	Incorrect ERC20 Interface	Logical Issue	● Major	✓ Resolved
DTS-01	Inaccurate Revert Message	Inconsistency	● Informational	✓ Resolved
DTS-02	Unused Constants	Logical Issue	● Informational	✓ Resolved
DTS-03	Miscalculation of `WeightTime`	Mathematical Operations	● Major	✓ Resolved
DTS-04	Incorrect Naming Convention Utilization	Coding Style	● Informational	✓ Resolved
DTS-05	Unlock From Pool A	Logical Issue	● Informational	⊗ Declined
DTS-06	Unusual Bonus Distribution Algorithm	Logical Issue	● Informational	ⓘ Acknowledged
DTS-07	Unused Function	Logical Issue	● Minor	✓ Resolved
DTS-08	Check Condition Inconsistent with Message	Logical Issue	● Informational	Partially Resolved
DTT-01	Lack of Input Validation	Volatile Code	● Minor	✓ Resolved
DTT-02	Incorrect Naming Convention Utilization	Coding Style	● Informational	✓ Resolved

DCF-01 | `ds.selectorSlots` Not Updated

Category	Severity	Location	Status
Logical Issue	● Major	facetBase/DiamondCutFacet.sol: 130~146	✓ Resolved

Description

`ds.selectorSlots` is not updated in `_ReplaceFacetSelectors()`, which may lead to the functions errors in contract `DiamondLoupeFacet`.

Recommendation

We advise client to update `ds.selectorSlots` in `_ReplaceFacetSelectors()`.

Alleviation

The issue is patched in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOA-01 | Unused Constant

Category	Severity	Location	Status
Logical Issue	● Informational	facetLogic/DOTCAdOrderFacet.sol: 18, 81	✓ Resolved

Description

The constant `nPriceDecimals` is not used in the contract.

Recommendation

We advise the client to review its functionality and remove it is of no use.

Alleviation

The client removed the unused constant and the issue is resolved in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOA-02 | Unnecessary Gas Cost to Call Variable from Structs

Category	Severity	Location	Status
Gas Optimization	● Informational	facetLogic/DOTCAdOrderFacet.sol: 145~169	✓ Resolved

Description

Due to the complexity of the project, calling variables in nested structs can incur a significant gas cost. Yet some of these consumptions are avoidable.

The `removeAdOrder()` function requires the message sender to be `db.orderTable.otcAdOrders[orderId].makerAddress` in order to proceed. Thus it is logically correct to substitute `makerAddress` with `msg.sender` to save gas. Note that issues of this type are not limited to the aforementioned location.

Recommendation

We advise the client to substitute `db.orderTable.otcAdOrders[orderId].makerAddress` for `msg.sender` to save gas. Besides, one could store the target value from a nested struct in a temporary variable first to avoid repeatedly accessing that data structure storage.

Alleviation

The client replaced `db.orderTable.otcAdOrders[orderId].makerAddress` with `msg.sender` as we had suggested as an effort to avoid unnecessary gas cost in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOA-03 | Function Name Misspellings

Category	Severity	Location	Status
Coding Style	● Informational	facetLogic/DOTCAdOrderFacet.sol: 193	✓ Resolved

Description

There are misspellings in `ConfermMoneyPayed()`, `ConfermMoneyReceived()` and `queruMultiAdOrdersStatus()`.

Recommendation

We advise the client change function names to `ConfirmMoneyPayed()`, `ConfirmMoneyReceived()` and `queryMultiAdOrdersStatus()` respectively.

Alleviation

The client fixed the spelling of function names in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOA-04 | Check Condition Inconsistent with Message

Category	Severity	Location	Status
Logical Issue	● Informational	facetLogic/DOTCAdOrderFacet.sol: 37, 67, 69	Partially Resolved

Description

Argument values that equal thresholds are permitted in these checks while the revert messages state otherwise. Particularly, issues of this type are found in `setMakerFee()`, `setTakerFee()`, `addStakingA()`, `createAdOrder()`, `_checkAdOrder()`, `_checkExOrder()`, `createExOrder()` as listed below.

```
require(_fee>=0,'fee must be greater than 0');
```

```
require(db.stakingTable.poolA[db.config.dotcContract].totalAccount<=nPoolMax,"Pool accounts have been the maximum");
```

```
require(amount>=consts.stakingParam.poolAMin,'amount must be greater than 100 DOTC');
```

```
require(db.stakingTable.poolB[db.config.dotcContract].totalAccount<=nPoolMax,"Pool accounts have been the maximum");
```

```
require(amount>=consts.stakingParam.poolBMin,'amount must be greater than 10 DOTC');
```

```
require(nOrderValue >= 20*nUsdtDecimals,'AdOrder value must be greater than 20 USDT.');
```

```
require(adInput.minAmount>= 20*nUsdtDecimals,'AdOrder value must be greater than 20 USDT.');
```

```
require(adInput.totalAmount>=adInput.maxAmount,"totalAmount must be greater than maxAmount");
```

```
require(adInput.minAmount<=adInput.maxAmount,"maxAmount must be greater than minAmount");
```

Recommendation

We advise client to fix either the conditionals or messages to make them consistent.

Alleviation

The client fixed some of the error messages to provide more lucid feedbacks in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOB-01 | Repetitive Function Implementations

Category	Severity	Location	Status
Logical Issue	● Minor	facetLogic/DOTCArbitBase.sol: 17~26, 173~208, 27~39, 51~104	ⓘ Acknowledged

Description

`_calculateArbitPeriod()`, `_checkExArbitApply()`, `_checkExArbitAccess()`, `_checkCancelOrderArbit()` and `_updateArbitResult()` have been implemented in the same way twice in different locations. Particularly, `_calculateArbitPeriod()` in DOTCArbitBase is missing "override" specifier and may not compile.

Recommendation

We advise client to keep only one implementation for each function.

Alleviation

The client agrees to implement these functions only once in their feedback but the issue is not yet fixed at this moment as it hasn't affect the contract functionality.

DOC-01 | Repetitive Function Implementations

Category	Severity	Location	Status
Logical Issue	● Minor	facetBase/DOTCFacetBase.sol: 296~327	ⓘ Acknowledged

Description

`_calculateArbitPeriod()`, `_checkExArbitApply()`, `_checkExArbitAccess()`, `_checkCancelOrderArbit()` and `_updateArbitResult()` have been implemented in the same way twice in different locations. Particularly, `_calculateArbitPeriod()` in DOTCArbitBase is missing "override" specifier and may not compile.

Recommendation

We advise client to keep only one implementation for each function.

Alleviation

The client agrees to implement these functions only once in their feedback but the issue is not yet fixed at this moment as it hasn't affect the contract functionality.

DOC-02 | Unnecessary Calculation Out of Bound Risk

Category	Severity	Location	Status
Mathematical Operations	● Informational	facetBase/DOTCFacetBase.sol: 129	Partially Resolved

Description

The calculation in `_getBackRate()` could easily lead to overflow when `nPeriodCount` accumulates, considering 1000^26 > 2^256 and 1000*(0.7^20) < 1. This can be easily avoided by replacing exponentials with a loop where the back rate is multiplied by 700 then divided by 1000 each time. Also, relevant financial model should be reviewed to ensure the calculation result stays inbound.

Recommendation

We advise the client to calculate the compounded `_backrate` with a loop.

Alleviation

The client added a max limit to `backRate` which partially resolved the issue in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOC-03 | Too Many Digits

Category	Severity	Location	Status
Coding Style	● Informational	facetBase/DOTCFacetBase.sol: 36	✓ Resolved

Description

Literals with many digits are difficult to read and review, such the several variables in `DOTCFacetBase.sol`.

Recommendation

Use:

- Ether suffix
- Time suffix, or
- The scientific notation

Alleviation

The client applies scientific notiations to improve readability and the issue is resolved in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOC-04 | Incorrect Naming Convention Utilization

Category	Severity	Location	Status
Coding Style	● Informational	facetBase/DOTCFacetBase.sol: 36, 37, 39, 40, 42, 44, 46	✓ Resolved

Description

Solidity defines a naming convention that should be followed. In general, the following naming conventions should be utilized in a Solidity file:

- Constants should be in UPPER_CASE_WITH_UNDERSCORES

In case the naming conventions are not followed, there should be proper documentation to explain the naming and the purpose of the variable. Issues of this type are found in `DOTCFacetBase` and `DOTCStakingFacet`.

Recommendation

The recommendations outlined here are intended to improve the readability, and thus they are not rules, but rather guidelines to try and help convey the most information through the names of things.

Alleviation

The client renamed some of the variables to improve readabllity and the issue is resolved in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOC-05 | Unused Function

Category	Severity	Location	Status
Logical Issue	● Minor	facetBase/DOTCFacetBase.sol: 58~63	✓ Resolved

Description

The internal function `_burnToken()` is never called in the project.

Recommendation

We advise the client to review its functionality and remove it if there is no plan for further use.

Alleviation

The client removed the function and the issue is resolved in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOC-06 | Redundant Length Getter

Category	Severity	Location	Status
Gas Optimization	● Informational	facetBase/DOTCFacetBase.sol: 328~334	⊗ Declined

Description

It appears that under no circumstances can `db.arbitTable.arbiterList[i]==0` therefore the loop and count operations are redundant.

Recommendation

We advise client to simply return `db.arbitTable.arbiterList.length` in `_getArbiterLength()`.

Alleviation

The client believes this is part of a fault-tolerant design and leaves it be.

DOC-07 | Redundant Remove Operation

Category	Severity	Location	Status
Gas Optimization	● Informational	facetBase/DOTCFacetBase.sol: 225~228	ⓘ Acknowledged

Description

The gas consumption of the current algorithm in `_removeArbiterFromDB()` scales with list length which is easily avoidable on the premise that the sequence of arbiters in `arbiterList` does not matter.

Recommendation

We advise the client to swap `db.arbitTable.arbiterList[i]` and `db.arbitTable.arbiterList[length-1]` then delete `db.arbitTable.arbiterList[length-1]` to cut gas consumption down to a constant.

Alleviation

The client agrees to revise the code as we suggested in a later version.

DOC-08 | Logic Issue When Removing Arbiter

Category	Severity	Location	Status
Logical Issue	● Informational	facetBase/DOTCFacetBase.sol: 233~236	ⓘ Acknowledged

Description

The locked issue would always be refunded full `consts.arbiterDOTC` in `_removeArbiterFromDB()` which makes the arbiter penalty less effective. We would like to know more about arbiter penalty rules that are missing in the whitepaper.

Alleviation

The client responded that honest arbitrators are not to be punished and finds no issue in the coded logic.

DOE-01 | Check Condition Inconsistent with Message

Category	Severity	Location	Status
Logical Issue	● Informational	facetLogic/DOTCExOrderBase.sol: 23, 22	Partially Resolved

Description

Argument values that equal thresholds are permitted in these checks while the revert messages state otherwise. Particularly, issues of this type are found in `setMakerFee()`, `setTakerFee()`, `addStakingA()`, `createAdOrder()`, `_checkAdOrder()`, `_checkExOrder()`, `createExOrder()` as listed below.

```
require(_fee>=0,'fee must be greater than 0');
```

```
require(db.stakingTable.poolA[db.config.dotcContract].totalAccount<=nPoolMax,"Pool accounts have been the maximum");
```

```
require(amount>=consts.stakingParam.poolAMin,'amount must be greater than 100 DOTC');
```

```
require(db.stakingTable.poolB[db.config.dotcContract].totalAccount<=nPoolMax,"Pool accounts have been the maximum");
```

```
require(amount>=consts.stakingParam.poolBMin,'amount must be greater than 10 DOTC');
```

```
require(nOrderValue >= 20*nUsdtDecimals,'AdOrder value must be greater than 20 USDT.');
```

```
require(adInput.minAmount>= 20*nUsdtDecimals,'AdOrder value must be greater than 20 USDT.');
```

```
require(adInput.totalAmount>=adInput.maxAmount,"totalAmount must be greater than maxAmount");
```

```
require(adInput.minAmount<=adInput.maxAmount,"maxAmount must be greater than minAmount");
```

Recommendation

We advise client to fix either the conditionals or messages to make them consistent.

Alleviation

The client fixed some of the error messages to provide more lucid feedbacks in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOF-01 | Repetitive Function Implementations

Category	Severity	Location	Status
Logical Issue	● Minor	facetLogic/DOTCArbitFacet.sol: 140~171, 151~160, 175~210, 161~173	ⓘ Acknowledged

Description

`_calculateArbitPeriod()`, `_checkExArbitApply()`, `_checkExArbitAccess()`, `_checkCancelOrderArbit()` and `_updateArbitResult()` have been implemented in the same way twice in different locations. Particularly, `_calculateArbitPeriod()` in DOTCArbitBase is missing "override" specifier and may not compile.

Recommendation

We advise client to keep only one implementation for each function.

Alleviation

The client agrees to implement these functions only once in their feedback but the issue is not yet fixed at this moment as it hasn't affect the contract functionality.

DOF-02 | Redundant Conditional

Category	Severity	Location	Status
Logical Issue	● Informational	facetLogic/DOTCArbitFacet.sol: 25, 28	✓ Resolved

Description

The conditionals in `createOrderArbit()` are unnecessarily repetitive.

Recommendation

We advise client to combine the codes under a single conditional.

Alleviation

The client combined the conditionals as we suggested and the issue is resolved in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOF-03 | Logical issues in `_clearInvitorSponsor`

Category	Severity	Location	Status
Logical Issue	● Informational	facetLogic/DOTCArbitFacet.sol: 325~341	ⓘ Acknowledged

Description

Judging from the context, the `invitor` should be the sponsor of `loser` rather than `msg.sender` and the inviter's assets rather than the loser's assets should be cleared in this function. Also, when `locked < nClearAmount`, the inviter does not need to pay anything rather than all that is locked. We advise the client to review the code logic and would like to know more about this part of the inviter rule that is missing in the whitepaper.

Alleviation

The client responded that invitor sposor may be zero in `_clearInvitorSponsor()` and the relevant functionality is just a credit show which doesn't work.

DOK-01 | Check Condition Inconsistent with Message

Category	Severity	Location	Status
Logical Issue	● Informational	facetLogic/DOTCFeeFacet.sol: 25, 38	Partially Resolved

Description

Argument values that equal thresholds are permitted in these checks while the revert messages state otherwise. Particularly, issues of this type are found in `setMakerFee()`, `setTakerFee()`, `addStakingA()`, `createAdOrder()`, `_checkAdOrder()`, `_checkExOrder()`, `createExOrder()` as listed below.

`require(_fee>=0,'fee must be greater than 0');`

`require(db.stakingTable.poolA[db.config.dotcContract].totalAccount<=nPoolMax,"Pool accounts have been the maximum");`

`require(amount>=consts.stakingParam.poolAMin,'amount must be greater than 100 DOTC');`

`require(db.stakingTable.poolB[db.config.dotcContract].totalAccount<=nPoolMax,"Pool accounts have been the maximum");`

`require(amount>=consts.stakingParam.poolBMin,'amount must be greater than 10 DOTC');`

`require(nOrderValue >= 20*nUsdtDecimals,'AdOrder value must be greater than 20 USDT.');`

`require(adInput.minAmount>= 20*nUsdtDecimals,'AdOrder value must be greater than 20 USDT.');`

`require(adInput.totalAmount>=adInput.maxAmount,"totalAmount must be greater than maxAmount");`

`require(adInput.minAmount<=adInput.maxAmount,"maxAmount must be greater than minAmount");`

Recommendation

We advise client to fix either the conditionals or messages to make them consistent.

Alleviation

The client fixed some of the error messages to provide more lucid feedbacks in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOL-01 | Volatile Type Conversion

Category	Severity	Location	Status
Volatile Code	● Minor	facetLogic/DOTCCardAribtFacet.sol: 60	✓ Resolved

Description

Explicit type conversion not allowed from `uint256` to `address`, such as in `createCardArbit()`.

Recommendation

Convert `uint256` to `uint160` first as in `DOTCArbitFacet.sol` L65.

Alleviation

The client revised the code so that `uint256` is converted to `uint160` first and the issue is fixed in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOL-02 | Magic Reward and Margin Rates

Category	Severity	Location	Status
Magic Numbers	● Informational	facetLogic/DOTCCardAribtFacet.sol: 1	⊗ Declined

Description

Margin rates and reward rates exists as magic numbers and are not settable unlike fee rates. Particularly in `_backWinnerDeposit()`, `_clearLoserDeposit()`, `_rewardArbiter()`, `CreateExOrder()`, `createCardArbit()` and more. We would advise the client to review the functionalities and use variables and setters to improve flexibility if appropriate.

Recommendation

We advise the client to review the functionalities and use variables and setters to improve readablity and flexibility.

Alleviation

The client introduced a few parameters in the later version but still uses magic numbers and responds that they want to solidify some parameters before evaluating the impact of variable parameters.

DOM-01 | Inconsistent Getter Function

Category	Severity	Location	Status
Logical Issue	● Minor	facetLogic/DOTCManageFacet.sol: 108	✓ Resolved

Description

`consts.stakingParam.bonusUnlockTime` is returned from `getStakingMin()` while the context suggests otherwise.

Recommendation

We advise client to change RHS to `consts.stakingParam.poolBMin`.

Alleviation

The client revised the code as we suggested and the issue is fixed in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOM-02 | Owner Privileges

Category	Severity	Location	Status
Centralization / Privilege	● **Informational**	facetLogic/DOTCManageFacet.sol	ⓘ **Acknowledged**

Description

To bridge the trust gap between owner and users, the owner needs to express a sincere attitude with the consideration of the administrator team’s anonymousness. The owner has the responsibility to notify users with the following capability:

- Designated `Manager` can modify key parameters in the market including `StakingMin`, `unLockWaitTime`, `bonusUnlockTime`, `firstBonusTime`, `bonusWaitTime`, `StakingStartTime`, `VIPConditionAmount`, `ManualDOTCPrice`, `PriceMode`, `dotcContract` and `wethContract`.
- Contract owner can force remove arbiters.
- All contracts use the Diamond design pattern (EIP-2535) and can be upgraded through the administrator actions. All facet logics are subject to potential modifications.

DOO-01 | Emit Events Missing `indexed`

Category	Severity	Location	Status
Volatile Code	● Informational	facetLogic/DOTCExOrderFacet.sol: 17~20	✓ Resolved

Description

It is generally good practice to put `indexed` before addresses and arrays in events. One can add the attribute indexed to up to three parameters which adds them to a special data structure known as "topics" instead of the data part of the log. Topics allow one to search for events, for example when filtering a sequence of blocks for certain events. One can also filter events by the address of the contract that emitted the event. Currently, event parameters in `DOTCExOrderFacet` are missing `indexd`.

Recommendation

We advise client to add `indexed` in emitted events. Note that this recommendation is not limited to this facet contract.

Alleviation

The client added `indexed` for event parameters and the issue is resolved in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOO-02 | Magic Reward and Margin Rates

Category	Severity	Location	Status
Magic Numbers	● Informational	facetLogic/DOTCExOrderFacet.sol: 1	⊗ Declined

Description

Margin rates and reward rates exists as magic numbers and are not settable unlike fee rates. Particularly in `_backWinnerDeposit()`, `_clearLoserDeposit()`, `_rewardArbiter()`, `CreateExOrder()`, `createCardArbit()` and more. We would advise the client to review the functionalities and use variables and setters to improve flexibility if appropriate.

Recommendation

We advise the client to review the functionalities and use variables and setters to improve readablity and flexibility.

Alleviation

The client introduced a few parameters in the later version but still uses magic numbers and responds that they want to solidify some parameters before evaluating the impact of variable parameters.

DOO-03 | Function Name Misspellings

Category	Severity	Location	Status
Coding Style	● Informational	facetLogic/DOTCExOrderFacet.sol: 84, 104	✓ Resolved

Description

There are misspellings in `ConfermMoneyPayed()`, `ConfermMoneyReceived()` and `queruMultiAdOrdersStatus()`.

Recommendation

We advise the client change function names to `ConfirmMoneyPayed()`, `ConfirmMoneyReceived()` and `queryMultiAdOrdersStatus()` respectively.

Alleviation

The client fixed the spelling of function names in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOO-04 | Check Condition Inconsistent with Message

Category	Severity	Location	Status
Logical Issue	● Informational	facetLogic/DOTCExOrderFacet.sol: 29	Partially Resolved

Description

Argument values that equal thresholds are permitted in these checks while the revert messages state otherwise. Particularly, issues of this type are found in `setMakerFee()`, `setTakerFee()`, `addStakingA()`, `createAdOrder()`, `_checkAdOrder()`, `_checkExOrder()`, `createExOrder()` as listed below.

`require(_fee>=0,'fee must be greater than 0');`

`require(db.stakingTable.poolA[db.config.dotcContract].totalAccount<=nPoolMax,"Pool accounts have been the maximum");`

`require(amount>=consts.stakingParam.poolAMin,'amount must be greater than 100 DOTC');`

`require(db.stakingTable.poolB[db.config.dotcContract].totalAccount<=nPoolMax,"Pool accounts have been the maximum");`

`require(amount>=consts.stakingParam.poolBMin,'amount must be greater than 10 DOTC');`

`require(nOrderValue >= 20*nUsdtDecimals,'AdOrder value must be greater than 20 USDT.');`

`require(adInput.minAmount>= 20*nUsdtDecimals,'AdOrder value must be greater than 20 USDT.');`

`require(adInput.totalAmount>=adInput.maxAmount,"totalAmount must be greater than maxAmount");`

`require(adInput.minAmount<=adInput.maxAmount,"maxAmount must be greater than minAmount");`

Recommendation

We advise client to fix either the conditionals or messages to make them consistent.

Alleviation

The client fixed some of the error messages to provide more lucid feedbacks in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOP-01 | Check-Effect-Interaction Pattern Violation

Category	Severity	Location	Status
Control Flow	● Minor	facetLogic/DOTCMiningFacet.sol: 43~45	✓ Resolved

Description

During `RemoveTokenFromRiskPool()`, `RemoveTokenFromRiskPool()` and function calls, state variables are changed after transfers. This violates the checks-effects-interactions pattern.

Recommendation

It is recommended to follow checks-effects-interactions pattern and execute the transfer after changing state variables for cases like this. It shields public and external functions from re-entrancy abuses. `checks-effects-interactions` pattern also applies to ERC20 tokens as they can inform the recipient of a transfer in certain implementations.

Alleviation

The client revised the code to follow the suggested pattern and the issue is resolved in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOP-02 | DAO Pools Only Use DOTC

Category	Severity	Location	Status
Logical Issue	● Informational	facetLogic/DOTCMiningFacet.sol	ⓘ Acknowledged

Description

It seems that currently DAO pool operations only use a single dotc address at any time. We would like an explanation for the design that the `MiningPool` and `RiskPool` data structures include multiple pool addresses. We would advise simplifying the add/remove functions by removing the token arguments and related checks if multiple tokens are to be supported. Otherwise, we advise adding a check to validate token is dotc in `remove()`.

Alleviation

The client responded that the coded logic is needed for future consideration.

DOR-01 | Check-Effect-Interaction Pattern Violation

Category	Severity	Location	Status
Control Flow	● Minor	facetLogic/DOTCRiskFacet.sol: 39~40	✓ Resolved

Description

During `RemoveTokenFromRiskPool()`, `RemoveTokenFromRiskPool()` and function calls, state variables are changed after transfers. This violates the checks-effects-interactions pattern.

Recommendation

It is recommended to follow checks-effects-interactions pattern and execute the transfer after changing state variables for cases like this. It shields public and external functions from re-entrancy abuses. `checks-effects-interactions` pattern also applies to ERC20 tokens as they can inform the recipient of a transfer in certain implementations.

Alleviation

The client revised the code to follow the suggested pattern and the issue is resolved in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DOR-02 | DAO Pools Only Use DOTC

Category	Severity	Location	Status
Logical Issue	● Informational	facetLogic/DOTCRiskFacet.sol	ⓘ Acknowledged

Description

It seems that currently DAO pool operations only use a single dotc address at any time. We would like an explanation for the design that the `MiningPool` and `RiskPool` data structures include multiple pool addresses. We would advise simplifying the add/remove functions by removing the token arguments and related checks if multiple tokens are to be supported. Otherwise, we advise adding a check to validate token is dotc in `remove()`.

Alleviation

The client responded that the coded logic is needed for future consideration.

DOS-01 | Repetitive Function Implementations

Category	Severity	Location	Status
Logical Issue	● Minor	facetLogic/DOTCArbitSettleFacet.sol: 61~114	ⓘ Acknowledged

Description

`_calculateArbitPeriod()`, `_checkExArbitApply()`, `_checkExArbitAccess()`, `_checkCancelOrderArbit()` and `_updateArbitResult()` have been implemented in the same way twice in different locations. Particularly, `_calculateArbitPeriod()` in DOTCArbitBase is missing "override" specifier and may not compile.

Recommendation

We advise client to keep only one implementation for each function.

Alleviation

The client agrees to implement these functions only once in their feedback but the issue is not yet fixed at this moment as it hasn't affect the contract functionality.

DOS-02 | Magic Reward and Margin Rates

Category	Severity	Location	Status
Magic Numbers	● Informational	facetLogic/DOTCArbitSettleFacet.sol: 1	⊗ Declined

Description

Margin rates and reward rates exists as magic numbers and are not settable unlike fee rates. Particularly in `_backWinnerDeposit()`, `_clearLoserDeposit()`, `_rewardArbiter()`, `CreateExOrder()`, `createCardArbit()` and more. We would advise the client to review the functionalities and use variables and setters to improve flexibility if appropriate.

Recommendation

We advise the client to review the functionalities and use variables and setters to improve readablity and flexibility.

Alleviation

The client introduced a few parameters in the later version but still uses magic numbers and responds that they want to solidify some parameters before evaluating the impact of variable parameters.

DOU-01 | Locked Assets Can Not Be Unlocked

Category	Severity	Location	Status
Logical Issue	● Informational	facetLogic/DOTCUserFacet.sol: 152~154	⊗ Declined

Description

Assets locked by `lockToken()` cannot be unlocked. We advise the client to review and explain the functionality of this method.

Recommendation

We advise the client to review the functionality of this method.

Alleviation

The client claims that the this is the asset that users can't use during the transaction.

DOU-02 | Token Permissibility

Category	Severity	Location	Status
Logical Issue	● Minor	facetLogic/DOTCUserFacet.sol	ⓘ Acknowledged

Description

At the current state, any ERC20 token is allowed to enter the market. The market credibility relies on the assumption that the internal asset balances are ever consistent with the external actual token balances. Yet, such consistency may be broken by either deflationary/elastic token design or a malicious attempt to scam other users (or a bit of both).

Recommendation

We advise the client to implement a whitelist to control token permissibility and avoid deflationary/rebasing/elastic tokens and add consistency checks before and after `tokenDeposit()` and `tokenWithdraw()`.

Alleviation

The client responded that the contracts will not restrict any token from entering the market just like uniswap while mainstream tokens would be highlighted at front-end and it is up to the users to assess the potential risks.

DTK-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	● Informational	token/DOTCToken.sol: 2	✓ Resolved

Description

The contract has unlocked compiler versions. 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 differing compiler version numbers. This can lead to 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

It is a general practice to instead lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

The client locked the compiler version in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DTK-02 | Set `constant` to Variables

Category	Severity	Location	Status
Logical Issue	● Informational	token/DOTCToken.sol: 11	✓ Resolved

Description

The variable `decimals` is not changed throughout the smart contract.

Recommendation

We advise the client to set `decimals` as a constant variable.

Alleviation

The `decimals` variable is now declared as a `constant` and the issue is fixed in commit `8b0297a94cc4c7651ab7b87842fc10f61dabccb2`.

DTK-03 | Proper Usage of `require` and `assert` Functions

Category	Severity	Location	Status
Coding Style	● Informational	token/DOTCToken.sol: 47	ⓘ Acknowledged

Description

The `assert` function in `_transfer()` should only be used to test for internal errors, and to check invariants. The `require` function should be used to ensure valid conditions, such as inputs, or contract state variables are met, or to validate return values from calls to external contracts.

Recommendation

Consider using the `require` function, along with a custom error message when the condition fails, instead of the `assert` function on the lines showcased above.

Alleviation

The client agrees to replace `assert` for `require` in a later version.

DTK-04 | Unused Return Value

Category	Severity	Location	Status
Coding Style	● Informational	token/DOTCToken.sol: 85	✓ Resolved

Description

The return value `success` in `approve()` is declared but never used in the function body.

Recommendation

Remove or comment out the return value.

Alleviation

The client removed `success` and fixed the issue in commit `8b0297a94cc4c7651ab7b87842fc10f61dabccb2`.

DTK-05 | Incorrect ERC20 Interface

Category	Severity	Location	Status
Logical Issue	● Major	token/DOTCToken.sol: 58	✓ Resolved

Description

Incorrect return values for ERC20 function `transfer()`. A contract compiled with Solidity > 0.4.22 interacting with these functions will fail to execute them, as the return value is missing.

Recommendation

Set the appropriate return values and types for the defined ERC20 function `transfer()`.

Alleviation

The `transfer()` function now returns a boolean in accordance with the ERC-20 and the issue is fixed in commit `8b0297a94cc4c7651ab7b87842fc10f61dabccb2`.

DTS-01 | Inaccurate Revert Message

Category	Severity	Location	Status
Inconsistency	● Informational	facetLogic/DOTCStakingFacet.sol: 48	✓ Resolved

Description

The fourth revert messages in `addStakingA()` and `addStakingB()` are potentially misleading if the manager adjusts the corresponding parameter. Loose feedbacks of this kind are not limited to this location.

Recommendation

We advise the client to provide more rigorous feedback in the revert messages.

Alleviation

The client corrected the error message as we suggested and the issue is fixed in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DTS-02 | Unused Constants

Category	Severity	Location	Status
Logical Issue	● Informational	facetLogic/DOTCStakingFacet.sol: 19	✓ Resolved

Description

`nPoolAMaxDays` is not used in the `DOTCStakingFacet` contract.

Recommendation

We advise the client to remove it if there is no plan for further usage.

Alleviation

The client removed `nPoolAMaxDays` and the issue is fixed.

DTS-03 | Miscalculation of `WeightTime`

Category	Severity	Location	Status
Mathematical Operations	● Major	facetLogic/DOTCStakingFacet.sol: 233	✓ Resolved

Description

`NewLockTime` in `_RecalculateWeightTime()` needs to be multiplied by 86400 before it is subtracted to get ‘newWeightTime’ so that their dimensions are both "seconds" according to the whitepaper.

Recommendation

We advise client to multiply `NewLockTime` by 86400 before it is subtracted to get ‘newWeightTime’.

Alleviation

The client corrected the calculation as we suggested and the issue is fixed in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DTS-04 | Incorrect Naming Convention Utilization

Category	Severity	Location	Status
Coding Style	● Informational	facetLogic/DOTCStakingFacet.sol: 46, 15~46, 19	✓ Resolved

Description

Solidity defines a naming convention that should be followed. In general, the following naming conventions should be utilized in a Solidity file:

- Constants should be in UPPER_CASE_WITH_UNDERSCORES

In case the naming conventions are not followed, there should be proper documentation to explain the naming and the purpose of the variable. Issues of this type are found in `DOTCFacetBase` and `DOTCStakingFacet`.

Recommendation

The recommendations outlined here are intended to improve the readability, and thus they are not rules, but rather guidelines to try and help convey the most information through the names of things.

Alleviation

The client renamed some of the variables to improve readabllity and the issue is resolved in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DTS-05 | Unlock From Pool A

Category	Severity	Location	Status
Logical Issue	● Informational	facetLogic/DOTCStakingFacet.sol	⊗ Declined

Description

It seems that at the moment `nPoolAMaxDays` is not yet used, the functionality described in the white paper is yet to be fulfilled and one cannot unlock stakes from pool A. We would like to first make sure there is no misunderstanding and inquire about further development plans.

Alleviation

The client modified the staking facet contract but the issue remains that the stakes in pool A cannot be unlocked.

DTS-06 | Unusual Bonus Distribution Algorithm

Category	Severity	Location	Status
Logical Issue	● Informational	facetLogic/DOTCStakingFacet.sol: 265, 223~235	ⓘ Acknowledged

Description

The staking bonus distribution employs an algorithm that gives some unusual results in `_RecalculateWeightTime()` and `_calculateAvailBonus()`. We found that sometimes a recent larger staking would decrease one's share for the bonus. We advise the client to review the financial models and give an explanation for such counter-intuitive cases.

Alleviation

The client claims that they will improve the bonus algorithm in a later version.

DTS-07 | Unused Function

Category	Severity	Location	Status
Logical Issue	● Minor	facetLogic/DOTCStakingFacet.sol: 220~222	✓ Resolved

Description

`WeightTimeTest()` has not been used in the contract.

Recommendation

We advise client to review its functionality and either declare it as `external` or remove it.

Alleviation

The client removed the function and the issue is resolved in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DTS-08 | Check Condition Inconsistent with Message

Category	Severity	Location	Status
Logical Issue	● Informational	facetLogic/DOTCStakingFacet.sol: 48	Partially Resolved

Description

Argument values that equal thresholds are permitted in these checks while the revert messages state otherwise. Particularly, issues of this type are found in `setMakerFee()`, `setTakerFee()`, `addStakingA()`, `createAdOrder()`, `_checkAdOrder()`, `_checkExOrder()`, `createExOrder()` as listed below.

```
require(_fee>=0,'fee must be greater than 0');
```

```
require(db.stakingTable.poolA[db.config.dotcContract].totalAccount<=nPoolMax,"Pool accounts have been the maximum");
```

```
require(amount>=consts.stakingParam.poolAMin,'amount must be greater than 100 DOTC');
```

```
require(db.stakingTable.poolB[db.config.dotcContract].totalAccount<=nPoolMax,"Pool accounts have been the maximum");
```

```
require(amount>=consts.stakingParam.poolBMin,'amount must be greater than 10 DOTC');
```

```
require(nOrderValue >= 20*nUsdtDecimals,'AdOrder value must be greater than 20 USDT.');
```

```
require(adInput.minAmount>= 20*nUsdtDecimals,'AdOrder value must be greater than 20 USDT.');
```

```
require(adInput.totalAmount>=adInput.maxAmount,"totalAmount must be greater than maxAmount");
```

```
require(adInput.minAmount<=adInput.maxAmount,"maxAmount must be greater than minAmount");
```

Recommendation

We advise client to fix either the conditionals or messages to make them consistent.

Alleviation

The client fixed some of the error messages to provide more lucid feedbacks in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DTT-01 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	● Minor	governance/DOTCTimelock.sol: 29	✓ Resolved

Description

The assigned values to `admin_` should be verified as non-zero values to prevent being mistakenly assigned as address(0) in the `constructor()` function.

Recommendation

Check that the addresses are not zero by adding the following checks in the `constructor()` function.

require(admin_ != address(0),"Zero address");

Alleviation

The client added a check as we had suggested and the issue is fixed in commit b1785dcd51678fc34456bf35f6bb62000207410e.

DTT-02 | Incorrect Naming Convention Utilization

Category	Severity	Location	Status
Coding Style	● Informational	governance/DOTCTimelock.sol: 24	✓ Resolved

Description

Solidity defines a naming convention that should be followed. In general, the following naming conventions should be utilized in a Solidity file:

- Functions and parameters should be in mixedCase

In case the naming conventions are not followed, there should be proper documentation to explain the naming and the purpose of the variable. `admin_initializd` does not conform to this convention unlike other variables in the context.

Recommendation

The recommendations outlined here are intended to improve the readability, and thus they are not rules, but rather guidelines to try and help convey the most information through the names of things.

Alleviation

The client renamed some of the variables to improve readability and the issue is resolved in commit b1785dcd51678fc34456bf35f6bb62000207410e.

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.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

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.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

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

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

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

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

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.

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 Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with 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.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK