ABDK CONSULTING

SMART CONTRACT AUDIT

TrueFi

TrueFi.V2

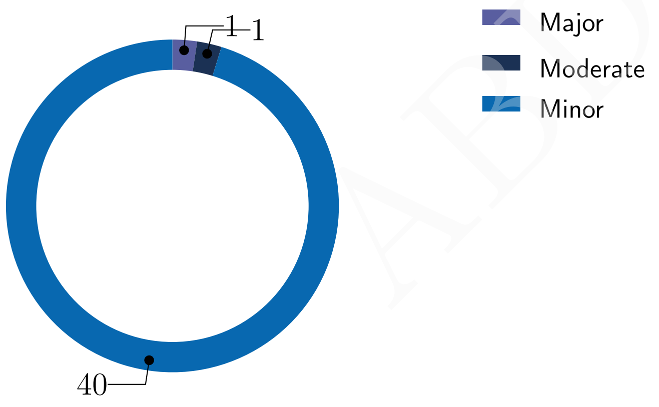
Solidity

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
7th June 2022

We've been asked to review 47 files in a Github repository. These are changes to the V1 version and were reviewed as a git diff. We found 1 major, and a few less important issues.



Findings

ID	Severity	Category	Status
CVF-1	Minor	Overflow/Underflow	Info
CVF-2	Minor	Suboptimal	Info
CVF-3	Major	Flaw	Info
CVF-4	Minor	Suboptimal	Info
CVF-5	Minor	Suboptimal	Info
CVF-6	Minor	Suboptimal	Info
CVF-7	Minor	Documentation	Info
CVF-8	Minor	Bad naming	Info
CVF-9	Minor	Procedural	Info
CVF-10	Minor	Suboptimal	Info
CVF-11	Minor	Unclear behavior	Fixed
CVF-12	Minor	Unclear behavior	Fixed
CVF-13	Minor	Suboptimal	Fixed
CVF-14	Minor	Unclear behavior	Fixed
CVF-15	Minor	Bad naming	Info
CVF-16	Minor	Suboptimal	Info
CVF-17	Minor	Unclear behavior	Fixed
CVF-18	Minor	Bad naming	Info
CVF-19	Minor	Suboptimal	Info
CVF-20	Minor	Suboptimal	Info
CVF-21	Minor	Suboptimal	Info
CVF-22	Minor	Suboptimal	Info
CVF-23	Minor	Suboptimal	Info
CVF-24	Minor	Suboptimal	Info
CVF-25	Minor	Suboptimal	Info
CVF-26	Minor	Procedural	Info
CVF-27	Minor	Bad naming	Info

ID	Severity	Category	Status
CVF-28	Minor	Bad naming	Info
CVF-29	Minor	Bad datatype	Info
CVF-30	Minor	Suboptimal	Info
CVF-31	Minor	Bad naming	Info
CVF-32	Minor	Bad datatype	Info
CVF-33	Minor	Unclear behavior	Fixed
CVF-34	Minor	Suboptimal	Info
CVF-35	Minor	Documentation	Info
CVF-36	Minor	Bad datatype	Info
CVF-37	Minor	Bad naming	Info
CVF-38	Minor	Suboptimal	Info
CVF-39	Minor	Bad naming	Info
CVF-40	Minor	Bad naming	Info
CVF-41	Minor	Procedural	Info
CVF-42	Moderate	Suboptimal	Info

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	June 6, 2022	D. Khovratovich	Initial Draft
0.2	June 7, 2022	D. Khovratovich	Minor revision
1.0	June 7, 2022	D. Khovratovich	Release

Contact

D. Khovratovich

khovratovich@gmail.com

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting at the customer request. The audit goal is a general review TrueFi protocol update.
We have reviewed the contracts at repository:

- access/interfaces/IManageable.sol
- access/InitializableManageable.sol
- access/Manageable.sol
- access/Upgradeable.sol
- governance/common/OwnedUpgradeabilityProxy.sol
- governance/common/StkClaimableContract.sol
- governance/AllowedDelegatesList.sol
- governance/DaoGovernor.sol
- governance/DaoToken.sol
- governance/StkTruToken.sol
- governance/VoteToken.sol
- interfaces/DSRegistryServiceInterface.sol
- interfaces/IAutomatedLineOfCredit.sol
- interfaces/IBasePortfolio.sol
- interfaces/IDebtInstrument.sol
- interfaces/IERC20WithDecimals.sol
- interfaces/IFinancialInstrument.sol
- interfaces/IFixedInterestOnlyLoans.sol
- interfaces/IFlexiblePortfolio.sol
- interfaces/IProtocolConfig.sol
- interfaces/ITransferStrategy.sol
- interfaces/IValuationStrategy.sol
- proxy/ProxyWrapper.sol
- strategies/DepositStrategy.sol
- strategies/FixedInterestOnlyLoansValuationStrategy.sol

- strategies/MultiInstrumentValuationStrategy.sol
- strategies/NonUsOnlyDeposit.sol
- strategies/PortfolioClosedWithdrawStrategy.sol
- strategies/TransferAgentWhitelistDepositStrategy.sol
- strategies/TransferAgentWhitelistTransferStrategy.sol
- strategies/TransferStrategy.sol
- strategies/WhitelistDeposit.sol
- strategies/WithdrawStrategy.sol
- utils/Multicall2.sol
- AutomatedLineOfCredit.sol
- AutomatedLineOfCreditFactory.sol
- BasePortfolio.sol
- BasePortfolioFactory.sol
- FixedInterestOnlyLoans.sol
- FlexiblePortfolio.sol
- FlexiblePortfolioFactory.sol
- ProtocolConfig.sol

The fixes were provided in a new commit.

2.1 About ABDK

ABDK Consulting, established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function. The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

2.2 Disclaimer

Note that the performed audit represents current best practices and smart contract standards which are relevant at the date of publication. After fixing the indicated issues the smart contracts should be re-audited.

2.3 Methodology

The methodology is not a strict formal procedure, but rather a collection of methods and tactics that combined differently and tuned for every particular project, depending on the project structure and and used technologies, as well as on what the client is expecting from the audit. In current audit we use:

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows code best practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places and that their visibility scopes and access levels are relevant. At this phase we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and is done properly. At this phase we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. We also check that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

3 Detailed Results

3.1 CVF-1

- **Severity** Minor
- **Category** Overflow/Underflow
- **Status** Info
- **Source** FixedInterestOnlyLoans.sol

Description Overflow is possible when converting to "uint40".
Recommendation Consider using safe conversion.
Client Comment Irrelevant, uint40 is 35k years so shouldn't be reached any time soon

Listing 1:

```
+loan.currentPeriodEndDate = uint40(block.timestamp + _periodDuration);
```

3.2 CVF-2

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** StkTruToken.sol

Description These checks are redundant, as it is anyway possible to pass a dead token address.

Listing 2:

```
+require(address(_tru) != address(0), "StkTruToken: TRU token address must not be 0");
+require(address(_tfusd) != address(0), "StkTruToken: tfUSD token address must not be 0");
+require(address(_feeToken) != address(0), "StkTruToken: fee token address must not be 0");

+require(address(_feeToken) != address(0), "StkTruToken: fee token address must not be 0");
```

3.3 CVF-3

- **Severity** Major
- **Category** Flaw
- **Status** Info
- **Source** StkTruToken.sol

Description This function shouldn't be public, as it should never be called outside of the "initialize" function. If this function would be called before the "initialize" function, it would make impossible to initialize the smart contract. If it would be called after the "initialize" function, the call would be revered.
Recommendation Consider declaring this function as internal.
Client Comment It is intended, because stkTruToken is already deployed and we need to call initializeTotalSupplyCheckpoints manually to initialize it.

Listing 3:

```
+function initTotalSupplyCheckpoints() public onlyOwner {
```

3.4 CVF-4

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** StkTruToken.sol

Description The expression "cooldownStart + cooldownTime" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 4:

```
+if (cooldownStart == 0 || cooldownStart + cooldownTime +
  ↪ unstakePeriodDuration < block.timestamp) {

+return cooldownStart + cooldownTime;
```

3.5 CVF-5

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** StkTruToken.sol

Recommendation Right shift would be more efficient.

Listing 5:

```
+uint256 halfAmount = amount / 2;
```

3.6 CVF-6

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** StkTruToken.sol

Description This check doesn't save gas, as storing the already stored value in a storage slot is actually as cheap. As reading a value from the storage

Listing 6:

```
if (nextDistributionIndex != index) {
```

3.7 CVF-7

- **Severity** Minor
- **Category** Documentation
- **Status** Info
- **Source** StkTruToken.sol

Description This function is still flawed.
Recommendation Consider documenting.

Listing 7:

```
+function _checkpointsLookup(Checkpoint[] storage checkpoints,
  ↪ uint256 blockNumber) internal view returns (uint256) {
```

3.8 CVF-8

- **Severity** Minor
- **Category** Bad naming
- **Status** Info
- **Source** AutomatedLineOfCredit.sol

Recommendation Should be "_setupRole" rather than "_grantRole".
Client Comment _setupRole is deprecated in favor of _grantRole, here is the link https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/AccessControl.sol#L203

Listing 8:

```
+_grantRole(DEPOSIT_ROLE, _depositStrategy);
+_grantRole(WITHDRAW_ROLE, _withdrawStrategy);
```

3.9 CVF-9

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** AutomatedLineOfCredit.sol

Recommendation This check should be made in the "initialize" function before assigning the "borrower" variable.
Client Comment This are security measures taken to prevent unexpected vectors of attacks. Even though we cannot reach them for now, we would like to keep them just in case. They are not stated in the initialize function, because borrower might be updated thorugh ‘upgradeTo‘ call.

Listing 9:

```
+require(address(this) != borrower, "AutomatedLineOfCredit: Pool cannot borrow from itself");

+require(borrower != address(this), "AutomatedLineOfCredit: Pool cannot repay itself");

+require(borrower != address(this), "AutomatedLineOfCredit: Pool cannot repay itself");
```

3.10 CVF-10

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** AutomatedLineOfCredit.sol

Description This check is redundant, as (msg.sender == borrower) and (borrower != address(this)) imply (msg.sender != address(this)).
Recommendation Consider removing this check.
Client Comment As above.

Listing 10:

```
+require(msg.sender != address(this), "AutomatedLineOfCredit: Pool cannot repay itself");

+require(msg.sender != address(this), "AutomatedLineOfCredit: Pool cannot repay itself");
```

3.11 CVF-11

- **Severity** Minor
- **Category** Unclear behavior
- **Status** Fixed
- **Source** AutomatedLineOfCredit.sol

Description This should be called only when the amount is non-zero.

Listing 11:

```
+_repay(amount);
```

3.12 CVF-12

- **Severity** Minor
- **Category** Unclear behavior
- **Status** Fixed
- **Source** AutomatedLineOfCredit.sol

Description This event is emitted even if nothing actually changed.

Listing 12:

```
+emit MaxSizeChanged(_maxSize);
```

3.13 CVF-13

- **Severity** Minor
- **Category** Suboptimal
- **Status** Fixed
- **Source** ProtocolConfig.sol

Recommendation The parameter should be indexed.

Listing 13:

```
+event PauserAddressChanged(address newPauserAddress);
```

3.14 CVF-14

- **Severity** Minor
- **Category** Unclear behavior
- **Status** Fixed
- **Source** ProtocolConfig.sol

Description This even is emitted even if nothing actually changed.

Listing 14:

```
+emit PauserAddressChanged(newPauserAddress);
```

3.15 CVF-15

- **Severity** Minor
- **Category** Bad naming
- **Status** Info
- **Source** FlexiblePortfolio.sol

Recommendation Should be "_setupRole" rather than "_grantRole".
Client Comment As above.

Listing 15:

```
+_grantRole(DEPOSIT_ROLE, _strategies.depositStrategy);
+_grantRole(WITHDRAW_ROLE, _strategies.withdrawStrategy);
```

3.16 CVF-16

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** FlexiblePortfolio.sol

Description This conditions is quite limiting.
Recommendation Consider separating the manager fee recipient from managers with elevated privileges.
Client Comment We keep it that way for this version and consider changes for the next one.

Listing 16:

```
+require(getRoleMemberCount(MANAGER_ROLE) == 1, "
   ↪ FlexiblePortfolio: Portfolio has multiple managers");
```

3.17 CVF-17

- **Severity** Minor
- **Category** Unclear behavior
- **Status** Fixed
- **Source** FlexiblePortfolio.sol

Description These events are emitted even if nothing actually changed.

Listing 17:

```
+emit ManagerFeeChanged(newManagerFee);

+emit MaxValueChanged(_maxValue);
```

3.18 CVF-18

- **Severity** Minor
- **Category** Bad naming
- **Status** Info
- **Source** OwnedUpgradeabilityProxy.sol

Recommendation Events are usually named via nouns, such as "ProxyOwnershipTransfer".

Listing 18:

```
+event ProxyOwnershipTransferred(address indexed previousOwner,
    ↪ address indexed newOwner);

+event Upgraded(address indexed implementation);
```

3.19 CVF-19

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** OwnedUpgradeabilityProxy.sol

Description The "previousOwner" parameter is redundant as its value could be derived from the previous event.

Listing 19:

```
+event ProxyOwnershipTransferred(address indexed previousOwner,
    ↪ address indexed newOwner);
```

3.20 CVF-20

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** OwnedUpgradeabilityProxy.sol

Description The "current Owner" parameter is redundant as its value could be derived from the previous event.

Listing 20:

```
+event NewPendingOwner(address currentOwner, address
    ↪ pendingOwner);
```

3.21 CVF-21

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** OwnedUpgradeabilityProxy.sol

Description Solidity compiler is smart enough to calcualte hashes of constant strings at compile time.
Recommendation Consider using expressions instead of precomputed hashes.

Listing 21:

```
+bytes32 private constant proxyOwnerPosition = 0x6279e8199720cf3557ecd8b58d667c8edc486bd1cf3ad59ea9ebdfcae0d0dfac; //keccak256("trueUSD.proxy.owner");
+bytes32 private constant pendingProxyOwnerPosition = 0x8ddbac328deee8d986ec3a7b933a196f96986cb4ee030d86cc56431c728b83f4; //keccak256("trueUSD.pending.proxy.owner");
```

3.22 CVF-22

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** OwnedUpgradeabilityProxy.sol

Description This check makes it impossible to cancel a pending ownership transfer.
Recommendation Consider removing this check.
Client Comment We could cancel by setting it to address 0x00...01. This is a proxy code, so we can't redeploy it.

Listing 22:

```
+require(newOwner != address(0));
```

3.23 CVF-23

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** OwnedUpgradeabilityProxy.sol

Description This function is redundant, as a payable fallback function is already defined.

Listing 23:

```
+receive() external payable {
```

3.24 CVF-24

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** OwnedUpgradeabilityProxy.sol

Description Using the free memory pointer value here is redundant.
Recommendation Just use zero instead, as this function doesn't need to care about memory.

Listing 24:

```
193 +let ptr := mload(0x40)
```

3.25 CVF-25

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** OwnedUpgradeabilityProxy.sol

Description Using "returndatasize()" here is very confusing.
Recommendation Consider either using zero instead or adding a comment explaining that "returndatasize()" here is used as "cheaper zero".

Listing 25:

```
194 +calldatacopy(ptr, returndatasize(), calldatasize())
```

3.26 CVF-26

- **Severity** Minor
- **Category** Procedural
- **Status** Info
- **Source** Upgradeable.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

Listing 26:

```
11 +constructor() initializer {}

20 +function _authorizeUpgrade(address) internal override onlyRole(
    ↪ DEFAULT_ADMIN_ROLE) {}
```

3.27 CVF-27

- **Severity** Minor
- **Category** Bad naming
- **Status** Info
- **Source** Upgradeable.sol

Recommendation Should be "__Pausable_init_unchained".

Listing 27:

```
+__Pausable_init();
```

3.28 CVF-28

- **Severity** Minor
- **Category** Bad naming
- **Status** Info
- **Source** Upgradeable.sol

Recommendation Here "_setupRole" should be used instead of "_grantRole".
Client Comment As above.

Listing 28:

```
+_grantRole(DEFAULT_ADMIN_ROLE, admin);
+_grantRole(PAUSER_ROLE, pauser);
```

3.29 CVF-29

- **Severity** Minor
- **Category** Bad datatype
- **Status** Info
- **Source** BasePortfolio.sol

Recommendation The arguments type should be "ITransferStrategy".

Listing 29:

```
+event TransferStrategyChanged(address indexed oldStrategy,
   ↪ address indexed newStrategy);
```

3.30 CVF-30

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** BasePortfolio.sol

Description The "oldStrategy" argument is redundant as its value could be derived from the previous event.

Listing 30:

```
+event TransferStrategyChanged(address indexed oldStrategy,
    ↪ address indexed newStrategy);
```

3.31 CVF-31

- **Severity** Minor
- **Category** Bad naming
- **Status** Info
- **Source** BasePortfolio.sol

Recommendation Should be "_setupRole" rather than "_grantRole".
Client Comment As above

Listing 31:

```
+_grantRole(MANAGER_ROLE, _manager);
```

3.32 CVF-32

- **Severity** Minor
- **Category** Bad datatype
- **Status** Info
- **Source** BasePortfolio.sol

Recommendation The argument type could be "ITransferStrategy".

Listing 32:

```
+function setTransferStrategy(address _transferStrategy) public
    ↪ onlyRole(MANAGER_ROLE) {

+function _setTransferStrategy(address _transferStrategy)
    ↪ internal {
```

3.33 CVF-33

- **Severity** Minor
- **Category** Unclear behavior
- **Status** Fixed
- **Source** BasePortfolio.sol

Description This event is emitted even if nothing actually changed.

Listing 33:

```
+emit TransferStrategyChanged(oldTransferStrategy ,
↪ _transferStrategy);
```

3.34 CVF-34

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** PortfolioClosedWithdrawStrategy.sol

Recommendation Should probably be ">=".

Listing 34:

```
+block.timestamp > portfolio.endDate(),
```

3.35 CVF-35

- **Severity** Minor
- **Category** Documentation
- **Status** Info
- **Source** FixedInterestOnlyLoansValuationStrategy.sol

Description The semantics of the keys in this mapping is unclear.
Recommendation Consider documenting.

Listing 35:

```
+mapping(uint256 => bool) isLoanActive;
```

3.36 CVF-36

- **Severity** Minor
- **Category** Bad datatype
- **Status** Info
- **Source** IFlexiblePortfolio.sol

Recommendation The types of these fields could be more specific.

Listing 36:

```
+address depositStrategy;
+address withdrawStrategy;
+address transferStrategy;
```

3.37 CVF-37

- **Severity** Minor
- **Category** Bad naming
- **Status** Info
- **Source** StkClaimableContract.sol

Recommendation Events are usually named via nouns, such as "OwnershipTransfer" and "OwnershipClaim".

Listing 37:

```
+event OwnershipTransferred(address indexed currentOwner,
    ↪ address indexed newPendingOwner);
+event OwnershipClaimed(address indexed currentOwner, address
    ↪ indexed newOwner);
```

3.38 CVF-38

- **Severity** Minor
- **Category** Suboptimal
- **Status** Info
- **Source** StkClaimableContract.sol

Recommendation The "currentOwner" parameters are redundant as the current owner could be derived from the previous events.

Listing 38:

```
+event OwnershipTransferred(address indexed currentOwner,
    ↪ address indexed newPendingOwner);
+event OwnershipClaimed(address indexed currentOwner, address
    ↪ indexed newOwner);
```

3.39 CVF-39

- **Severity** Minor
- **Category** Bad naming
- **Status** Info
- **Source** StkClaimableContract.sol

Description The name of this event is misleading, as the event doesn't indicate an ownership transfer, but only initiation of such transfer.
Recommendation Consider renaming.

Listing 39:

```
22 +event OwnershipTransferred(address indexed currentOwner,
    ↪ address indexed newPendingOwner);
```

3.40 CVF-40

- **Severity** Minor
- **Category** Bad naming
- **Status** Info
- **Source** AllowedDelegatesList.sol

Recommendation Events are usually named via nouns, such as "DelegateStatus".

Listing 40:

```
11 +event AllowedListChanged(address account, bool isAllowed);
```

3.41 CVF-41

- **Severity** Minor
- **Category** Procedural
- **Status** Info
- **Source** AllowedDelegatesList.sol

Recommendation The value "10**tru.decimals()" could be precomputed in the constructor and stored in am immutable variable. Alternatively, consider passing this value as a constructor argument or making it a named constant.

Listing 41:

```
21 +tru.transferFrom(msg.sender, address(this), 10**tru.decimals())
    ↪ ;

28 +tru.transfer(msg.sender, 10**tru.decimals());
```

3.42 CVF-42

- **Severity** Moderate
- **Category** Suboptimal
- **Status** Info
- **Source** AllowedDelegatesList.sol

Description The returned value is ignored.
Recommendation Consider explicitly requiring the returned value to be true.

Listing 42:

```
21 +tru.transferFrom(msg.sender, address(this), 10**tru.decimals())
     ↪ ;

28 +tru.transfer(msg.sender, 10**tru.decimals());
```