PARK ONE

SMART CONTRACTS REVIEW

zokyo

February 9th 2024 | v. 1.0

Security Audit Score

PASS

Zokyo Security has concluded that these smart contracts passed a security audit.

ZOKYO AUDIT SCORING DARKPOOL ONE

1. Severity of Issues:
    - Critical: Direct, immediate risks to funds or the integrity of the contract. Typically, these would have a very high weight.
    - High: Important issues that can compromise the contract in certain scenarios.
    - Medium: Issues that might not pose immediate threats but represent significant deviations from best practices.
    - Low: Smaller issues that might not pose security risks but are still noteworthy.
    - Informational: Generally, observations or suggestions that don't point to vulnerabilities but can be improvements or best practices.
2. Test Coverage: The percentage of the codebase that's covered by tests. High test coverage often suggests thorough testing practices and can increase the score.
3. Code Quality: This is more subjective, but contracts that follow best practices, are well-commented, and show good organization might receive higher scores.
4. Documentation: Comprehensive and clear documentation might improve the score, as it shows thoroughness.
5. Consistency: Consistency in coding patterns, naming, etc., can also factor into the score.
6. Response to Identified Issues: Some audits might consider how quickly and effectively the team responds to identified issues.

HYPOTHETICAL SCORING CALCULATION:

Let's assume each issue has a weight:
- Critical: -30 points
- High: -20 points
- Medium: -10 points
- Low: -5 points
- Informational: -1 point

Starting with a perfect score of 100:
- 0 Critical issues: 0 points deducted
- 1 High issue: 1 resolved = 0 points deducted
- 0 Medium issues: = 0 points deducted
- 1 Low issue: 1 resolved = 0 points deducted
- 1 Informational issue: 1 resolved = 0 points deducted

Hence, the score is 100.

TECHNICAL SUMMARY

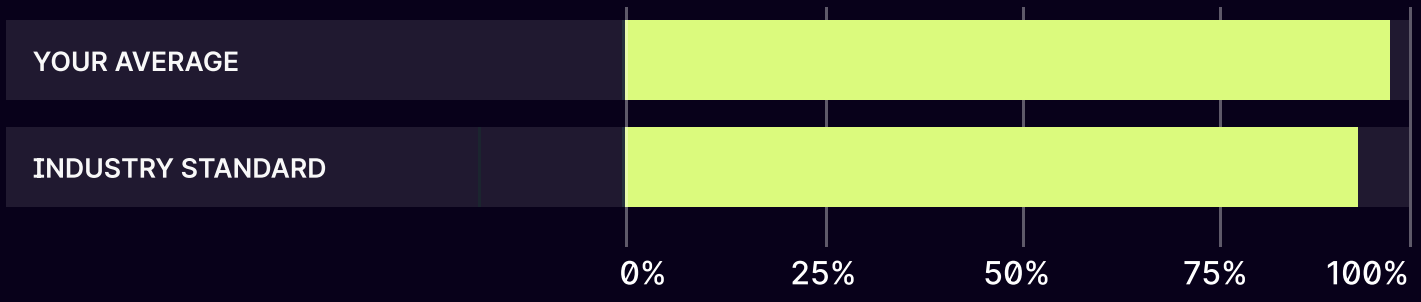
This document outlines the overall security of the DarkPool One smart contracts evaluated by the Zokyo Security team.

The scope of this audit was to analyze and document the DarkPool One smart contracts codebase for quality, security, and correctness.

Contract Status

There were 0 critical issues found during the review. (See Complete Analysis)

Testable Code



The testable code is 92,3%. (See Complete Analysis)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contracts but rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that can withstand the Ethereum network's fast-paced and rapidly changing environment, we recommend that the DarkPool One team put in place a bug bounty program to encourage further active analysis of the smart contracts.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the DarkPool One repository:
Repo: https://github.com/brightpoolone/web3-contracts

Last commit - 3c9f5d80d52b7bbddd095d9c402368d2277a1728

Within the scope of this audit, the team of auditors reviewed the following contract(s):

- ./IBrightPoolConsumer.sol
- ./IBrightPoolExchange.sol
- ./BrightPoolTreasury.sol
- ./Errors.sol
- ./BrightPoolLedger.sol
- ./Ownable.sol
- ./UniswapExchange.sol
- ./BRI.sol
- ./LedgerOwner.sol
- ./BRIX.sol
- ./BrightPoolManager.sol
- ./IBrightPoolLedger.sol
- ./BrightPoolWarden.sol
- ./IBrightPoolExchangeable.sol
- ./Vesting.sol
- ./VestingToken.sol
- ./interfaces/IUniswapV2Router01.sol
- ./interfaces/IUniswapV2Router02.sol
- ./interfaces/IUniswapV2Pair.sol
- ./IBrightPoolTreasury.sol

During the audit, Zokyo Security ensured that the contract:

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices, efficiently using resources without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the most resent vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of DarkPool One smart contracts. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. Part of this work includes writing a test suite using the Foundry testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01	Due diligence in assessing the overall code quality of the codebase.	03	Testing contracts logic against common and uncommon attack vectors.
02	Cross-comparison with other, similar smart contracts by industry leaders.	04	Thorough manual review of the codebase line by line.

Executive Summary

The review revealed one high-severity finding, one instance of low-severity and one informational issue. Detailed explanations of these findings can be found in the "Complete Analysis" section.

STRUCTURE AND ORGANIZATION OF THE DOCUMENT

For the ease of navigation, the following sections are arranged from the most to the least critical ones. Issues are tagged as “Resolved” or “Unresolved” or “Acknowledged” depending on whether they have been fixed or addressed. Acknowledged means that the issue was sent to the DarkPool One team and the DarkPool One team is aware of it, but they have chosen to not solve it. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Client or remains disregarded by the Client. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

FINDINGS SUMMARY

#	Title	Risk	Status
1	Certain ERC20 tokens do not return bool from approve which breaks the core logic	High	Resolved
2	Validate Array Lengths	Low	Resolved
3	Unreachable Branch	Informational	Resolved

HIGH-1 | RESOLVED

Certain ERC20 tokens do not return bool from approve which breaks the core logic

The BrightPoolLenger and UniswapExchange contracts attempt to approve the ask or bid token for other contracts and require the approve function to return true. However, some ERC20 implementations (e.g., mainnet USDT) do not return a value. This will cause a revert when the implementation tries to decode the boolean return value of the call to approve.

```
if (!order.bid.token.approve(address(exchangeable), order.bid.amount)) revert
InsufficientFunds();
```

However, some ERC20 implementations (e.g., mainnet USDT) do not return a value. This will cause a revert when the implementation tries to decode the boolean return value of the call to approve.

Recommendation:
Consider using SafeApprove instead of approve.

LOW-1 | RESOLVED

Validate array lengths

In the contract Vesting.sol's constructor array of vestedPlans , receivers , amounts and timestamps is passed but there are not sufficient checks to verify the lengths of these arrays.
It should be made sure that these arrays are of equal lengths otherwise it might result in erroneous assignment.

Recommendation:
Ensure all these arrays are of equal lengths.

INFORMATIONAL-1 | RESOLVED

Unreachable Branch

In the function _internalDeposit in the contract Vesting.sol the condition if (receiver_ == address(0)) revert ZeroAddress() is unnecessary since at L114 ERC20's mint is called which reverts on `0` address mint , therefore checking the condition again is not needed.

Recommendation:
The check can be removed.

	./IBrightPoolConsumer.sol ./IBrightPoolExchange.sol ./BrightPoolTreasury.sol ./Errors.sol ./BrightPoolLedger.sol ./Ownable.sol ./UniswapExchange.sol
Re-entrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass

	./BRI.sol ./LedgerOwner.sol ./BRIX.sol ./BrightPoolManager.sol ./IBrightPoolLedger.sol ./BrightPoolWarden.sol ./IBrightPoolExchangeable.sol
Re-entrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass

	./Vesting.sol **./VestingToken.sol** **./interfaces/IUniswapV2Router01.sol** **./interfaces/IUniswapV2Router02.sol** **./interfaces/IUniswapV2Pair.sol** **./IBrightPoolTreasury.sol**
Re-entrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Security

As a part of our work assisting DarkPool One in verifying the correctness of their contracts code, our team was responsible for writing integration tests using the Foundry testing framework.

The tests were based on the functionality of the code, as well as a review of the DarkPool One contracts requirements for details about issuance amounts and how the system handles these.

Running 6 tests for test/SakBrightPoolWardenTest.t.sol:SakBrightPoolWardenTest
[PASS] testChangeValue() (gas: 84500)
[PASS] testChangeValueCaseFive() (gas: 87881)
[PASS] testChangeValueCaseFour() (gas: 65619)
[PASS] testChangeValueCaseSix() (gas: 69004)
[PASS] testChangeValueCaseThree() (gas: 87103)
[PASS] testChangeValueCaseTwo() (gas: 22479)
Test result: ok. 6 passed; 0 failed; 0 skipped; finished in 13.09s

Running 6 tests for test/SakVestingTokenTest.t.sol:SakVestingTokenTest
[PASS] testBurn() (gas: 64834)
[PASS] testBurnRevertsNotOwner() (gas: 63661)
[PASS] testMint() (gas: 61115)
[PASS] testMintRevertsNotOwner() (gas: 13245)
[PASS] testNameSymbolDecimal() (gas: 19904)
[PASS] testTransfer() (gas: 10992)
Test result: ok. 6 passed; 0 failed; 0 skipped; finished in 24.63s

Running 3 tests for test/BRITest.t.sol:BRITokenTest
[PASS] testBurn() (gas: 23742)
[PASS] testDecimals() (gas: 5549)
[PASS] testGetOwner() (gas: 11316)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 29.09s

Running 14 tests for test/BrightPoolTreasuryTest.t.sol:BrightPoolTreasuryTest
[PASS] testAllAffiliates() (gas: 129817)
[PASS] testBalanceOf() (gas: 13230)
[PASS] testConsume() (gas: 93508)
[PASS] testExchange() (gas: 109477)

[PASS] testMoveAsset() (gas: 50344)
[PASS] testRemoveAffiliate() (gas: 101570)
[PASS] testRemoveAffiliateZeroValue() (gas: 131696)
[PASS] testRewardForAffiliate() (gas: 128479)
[PASS] testSetAffilate() (gas: 127373)
[PASS] testSetAffilateTooBig() (gas: 14663)
[PASS] testSetAffilateZeroBps() (gas: 14192)
[PASS] testSetAffilateZeroValue() (gas: 14405)
[PASS] testSetShares() (gas: 28013)
[PASS] testSetSharesTooBig() (gas: 27752)
Test result: ok. 14 passed; 0 failed; 0 skipped; finished in 29.09s

Running 10 tests for test/SakBrixTest.t.sol:SakBrixTest
[PASS] testBurnReward() (gas: 70101)
[PASS] testDecimals() (gas: 5703)
[PASS] testGetOwner() (gas: 10199)
[PASS] testKillTransfersRevertCaseOne() (gas: 21080)
[PASS] testKillTransfersRevertCaseThree() (gas: 37993)
[PASS] testKillTransfersRevertCaseTwo() (gas: 37735)
[PASS] testNameSymbol() (gas: 19538)
[PASS] testReward() (gas: 64502)
[PASS] testRewardRevert() (gas: 17874)
[PASS] testRewardRevertNotLedger() (gas: 15932)
Test result: ok. 10 passed; 0 failed; 0 skipped; finished in 29.09s

Running 8 tests for test/SakVestingTest.sol:SakVestingTest
[PASS] testAvailableToWithdraw() (gas: 38813)
[PASS] testAvailableToWithdrawInPlan() (gas: 29516)
[PASS] testFinal() (gas: 10890)
[PASS] testGetVestingPlans() (gas: 22255)
[PASS] testInternalDeposit() (gas: 944552)
[PASS] testWithdrawCaseOne() (gas: 28710)
[PASS] testWithdrawCaseThree() (gas: 103370)
[PASS] testWithdrawCaseTwo() (gas: 104937)
Test result: ok. 8 passed; 0 failed; 0 skipped; finished in 49.00s

Running 45 tests for test/SakBrightPoolManagerTest.t.sol:SakBrightPoolManagerTest
[PASS] testCancelOrder() (gas: 87696)
[PASS] testCancelOrderLedgerReturnsFalse() (gas: 87896)
[PASS] testCancelOrderRevertCaseOne() (gas: 82886)
[PASS] testCancelOrderRevertCaseThree() (gas: 84628)

```
[PASS] testCancelOrderRevertCaseTwo() (gas: 80295)
[PASS] testCancelOrderWithPermit() (gas: 100450)
[PASS] testCancelOrderWithPermitInsufficientFunds() (gas: 87585)
[PASS] testCancelOrderWithPermitLedgerReturnsFalse() (gas: 88375)
[PASS] testCancelOrderWithPermitRevertCaseOne() (gas: 90551)
[PASS] testCancelOrderWithPermitRevertCaseThree() (gas: 23662)
[PASS] testCancelOrderWithPermitRevertCaseTwo() (gas: 16198)
[PASS] testCancelOrderWithPermitSufficientAllowance() (gas: 132960)
[PASS] testConstructParamOne() (gas: 167244)
[PASS] testExecute() (gas: 88871)
[PASS] testExecuteLedgerReturnsFalse() (gas: 83507)
[PASS] testExecuteMultipleOrders() (gas: 125258)
[PASS] testExecuteMultipleOrdersRevertCaseOne() (gas: 119498)
[PASS] testExecuteRevertCaseOne() (gas: 83562)
[PASS] testGetTreasury() (gas: 10501)
[PASS] testMakeOrder() (gas: 77458)
[PASS] testMakeOrderRevertCaseFour() (gas: 25276)
[PASS] testMakeOrderRevertCaseOne() (gas: 21025)
[PASS] testMakeOrderRevertCaseThree() (gas: 19045)
[PASS] testMakeOrderRevertCaseTwo() (gas: 18636)
[PASS] testMakeOrderWithPermit() (gas: 141556)
[PASS] testMakeOrderWithPermitBidTokenZero() (gas: 138621)
[PASS] testMakeOrderWithPermitRevertCaseFour() (gas: 37863)
[PASS] testMakeOrderWithPermitRevertCaseOne() (gas: 33091)
[PASS] testMakeOrderWithPermitRevertCaseThree() (gas: 31602)
[PASS] testMakeOrderWithPermitRevertCaseTwo() (gas: 31380)
[PASS] testMakeOrderWithPermitZeroDeadline() (gas: 43735)
[PASS] testSetBackend() (gas: 22359)
[PASS] testSetBackendRevertNonAdmin() (gas: 16696)
[PASS] testSetBackendRevertSameCron() (gas: 18550)
[PASS] testSetBackendRevertZeroAddress() (gas: 18767)
[PASS] testSetCron() (gas: 21416)
[PASS] testSetCronRevertNonAdmin() (gas: 15728)
[PASS] testSetCronRevertSameCron() (gas: 17846)
[PASS] testSetCronRevertZeroAddress() (gas: 18437)
[PASS] testSetLedger() (gas: 11729)
[PASS] testSetLedgerNonAdmin() (gas: 40369)
[PASS] testSetTreasury() (gas: 92922)
[PASS] testSetTreasuryAlreadySet() (gas: 95833)
[PASS] testSetTreasuryNonOwnerOrAdmin() (gas: 15745)
[PASS] testSetTreasuryToZeroAddress() (gas: 18992)
Test result: ok. 45 passed; 0 failed; 0 skipped; finished in 70.45s
```

Running 17 tests for test/UniswapTests.t.sol:UniswapTests
[PASS] testAmountOut() (gas: 21132)
[PASS] testAmountOutCaseThree() (gas: 17200)
[PASS] testAmountOutCaseTwo() (gas: 14398)
[PASS] testUniswapExchangeApproveFails() (gas: 32510)
[PASS] testUniswapExchangeCaseEight() (gas: 116785)
[PASS] testUniswapExchangeCaseFive() (gas: 115453)
[PASS] testUniswapExchangeCaseFour() (gas: 117054)
[PASS] testUniswapExchangeCaseNine() (gas: 106861)
[PASS] testUniswapExchangeCaseOne() (gas: 115739)
[PASS] testUniswapExchangeCaseOneAskTokenApproveFails() (gas: 112797)
[PASS] testUniswapExchangeCaseTen() (gas: 106983)
[PASS] testUniswapExchangeCaseThree() (gas: 74155)
[PASS] testUniswapExchangeCaseThreeApproveFails() (gas: 71066)
[PASS] testUniswapExchangeCaseTwo() (gas: 117005)
[PASS] testUniswapExchangeCaseTwoApproveFails() (gas: 30750)
[PASS] testUniswapExchangeCaseseven() (gas: 38818)
[PASS] testUniswapExchangeCasesix() (gas: 73745)
Test result: ok. 17 passed; 0 failed; 0 skipped; finished in 71.67s

Running 25 tests for test/BrightPoolLeadger.t.sol:BrightPoolLedgerTest
[PASS] testAddExchangeZeroAddress() (gas: 14010)
[PASS] testAddRemoveExchange() (gas: 93516)
[PASS] testExecuteOrder() (gas: 292856)
[PASS] testExecuteOrderAffRcpt() (gas: 348956)
[PASS] testExecuteOrderBlocked() (gas: 288873)
[PASS] testExecuteOrderETH() (gas: 200896)
[PASS] testExecuteOrderRevoked() (gas: 291018)
[PASS] testExecuteOrderRewardConsumed() (gas: 319076)
[PASS] testExecuteOrderZeroAddress() (gas: 353405)
[PASS] testExecuteOrderZeroOrderId() (gas: 290930)
[PASS] testExecuteOrderZeroValue() (gas: 355688)
[PASS] testMakeAffilationId() (gas: 382728)
[PASS] testMakeOrder() (gas: 350647)
[PASS] testMakeOrderAlreadyExists() (gas: 263281)
[PASS] testMakeOrderETH() (gas: 233114)
[PASS] testMakeOrderETHInsufficientFunds() (gas: 233028)
[PASS] testMakeOrderTimeout() (gas: 18157)
[PASS] testMakeOrderWrongAffiliatePlan() (gas: 23072)
[PASS] testMakeOrderWrongCurrency() (gas: 255402)
[PASS] testMakeOrderZeroValue() (gas: 18209)
[PASS] testNewBaseURI() (gas: 19060)
[PASS] testRemoveExchangeNotExists() (gas: 16021)

[PASS] testRewardToken() (gas: 6398)
[PASS] testRoyaltyInfo() (gas: 9310)
[PASS] testSupportInterface() (gas: 5478)
Test result: ok. 25 passed; 0 failed; 0 skipped; finished in 19.59s

Running 22 tests for test/BrightPoolTreasuryTest.t.sol:BrightPoolTreasuryTest
[PASS] testAllAffiliates() (gas: 129817)
[PASS] testBalanceOf() (gas: 17268)
[PASS] testChangeAffilate() (gas: 133951)
[PASS] testConsume() (gas: 93596)
[PASS] testExchange() (gas: 109236)
[PASS] testExchangeEthEth() (gas: 51299)
[PASS] testExchangeEthInsufficientFunds() (gas: 39478)
[PASS] testExchangeEthToken() (gas: 119204)
[PASS] testExchangeEthTokenOutOfFunds() (gas: 72384)
[PASS] testExchangeTokenEth() (gas: 78637)
[PASS] testMoveAsset() (gas: 50454)
[PASS] testRemoveAffiliate() (gas: 178376)
[PASS] testRemoveAffiliateZeroValue() (gas: 131850)
[PASS] testRewardForAffiliate() (gas: 128567)
[PASS] testSetAffilate() (gas: 127461)
[PASS] testSetAffilateTooBig() (gas: 14828)
[PASS] testSetAffilateZeroBps() (gas: 14225)
[PASS] testSetAffilateZeroValue() (gas: 14526)
[PASS] testSetShares() (gas: 28101)
[PASS] testSetSharesTooBig() (gas: 27906)
[PASS] testSetSharesZeroAddress() (gas: 19564)
[PASS] testSetSharesZeroValue() (gas: 21584)
Test result: ok. 22 passed; 0 failed; 0 skipped; finished in 11.96s

Running 3 tests for test/BRITest.t.sol:BRITokenTest
[PASS] testBurn() (gas: 23742)
[PASS] testDecimals() (gas: 5549)
[PASS] testGetOwner() (gas: 11316)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 10.95s

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	% UNCOVE-RED LINES
UniswapExchange	75.00	81.82	100	78.72	
Vesting	94.12	92.85	100	94.29	
VestingToken	100	100	100	100	
LedgerOwner	95.65	100	100	100	
BRIX	96	90	100	100	
BrightPoolManager	90.48	75.81	100	87.72	
BRI	100	100	100	100	
BrightPoolTreasury	96.55	78.12	100	96.43	
BrightPoolLender	75.41	60.71	100	78.21	
Ownable	100	100	100	100	
All files	**92.3**	**87.93**	**100**	**83.53**	

We are grateful for the opportunity to work with the DarkPool One team.

The statements made in this document should not be interpreted as an investment or legal advice, nor should its authors be held accountable for the decisions made based on them.

Zokyo Security recommends the DarkPool One team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo