Bondly

SMART CONTRACT AUDIT

ZOKYO.

Dec 4, 2020 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

SCORE

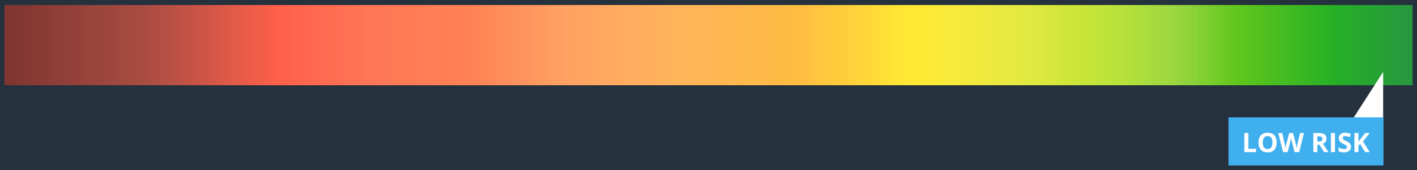
98

TECHNICAL SUMMARY

This document outlines the overall security of the Bondly smart contracts, evaluated by Zokyo's Blockchain Security team.

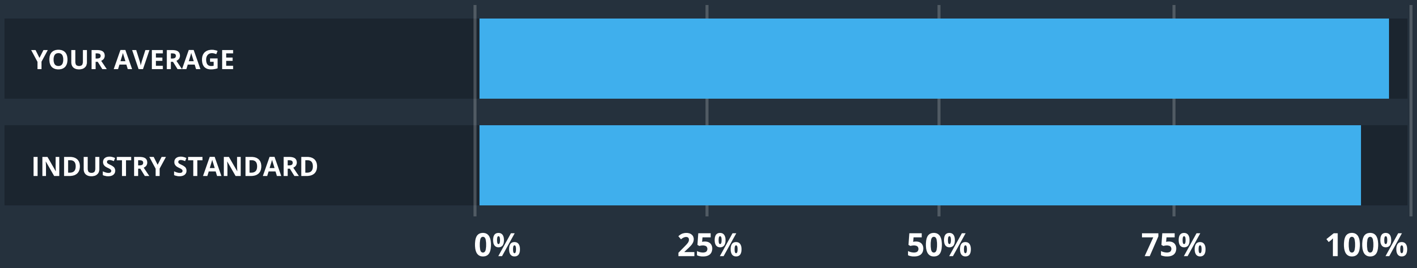
The scope of this audit was to analyze and document the Bondly smart contract codebase for quality, security, and correctness.

Contract Status



There were no critical issues found during the audit.

Testable Code



Testable code is 98%, which is above the industry standard of 95%.

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

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract’s source code was taken from the Bondly repo main branch (last commit - We agreed that tokens should be available at the beginning of the mon).

Bondly repositore – https://github.com/Bondreposit/bondly-token

Main Branch – https://github.com/Bondreposit/bondly-token/tree/main

Last commit – https://github.com/Bondreposit/bondly-token/commit/483e43859f27e8ffb4159458f684eb409438daf9.

Requirements:

White paper – https://drive.google.com/drive/u/2/folders/10I3CvaCx8o-YKGgQUYZN0jXOfUD_GKY0

Changes to the allocation – https://docs.google.com/document/d/1owmdcgLwiYx2CaVyQgu3jPkYhfCSbVIT8YIlyGzMY0Y/edit

Bondly Token Metrics Final – https://docs.google.com/spreadsheets/d/1ltPLrNuLduUU9Y0q4jvezDCEqSKKdpptaGD9kEM8bMw/edit#gid=1511546506

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of Bondly smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1. Due diligence in assessing the overall code quality of the codebase.
2. Cross-comparison with other, similar smart contracts by industry leaders.
3. Testing contract logic against common and uncommon attack vectors.
4. Thorough, manual review of the codebase, line-by-line.

SUMMARY

There were no critical issues found during the audit. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner.

Contracts are well written and structured. The findings during the audit have no impact on contract performance or security, so it is fully production-ready.

Despite the fact, the expected logic is managing all vestings by the owner, it should be careful with parameters to avoid mistakes during the vesting process.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. 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 ability of the contract to compile or operate in a significant way.

● 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

The contract owner has too many permissions

LOW | RESOLVED

The vesting functionality is built based on token allocation performed by the contract owner. It means only token owners can send vested tokens to token holders. The potential token holders don't have any guarantees they receive these tokens. The function send can accept each time another address. As the function is executed manually by the owner, there is a risk to put the wrong holder address.

Recommendation:
Make vesting in a way to claim tokens by future holders, or implement lock up functionality for automated unlock.

Re-audit:
The client marked it as expected logic. It was agreed that this was the approach we would take and is not uncommon. Token holders have guarantees as per their SAFT agreements.

Unreachable code is present in contract

INFORMATIONAL | UNRESOLVED

The code in contract BondlyTokenSale (line 61) and BondlyTokenHolder (line 43) is unreachable.

Recommendation:
The unreachable code can be removed.

Contracts code style can be upgraded

INFORMATIONAL | UNRESOLVED

The code in the contract does not fully correspond to recommendations created by the solidity comunity https://docs.soliditylang.org/en/v0.6.12/style-guide.html.

Recommendation:
To increase contract readability update the contract according to style guide.

	BondlyToken	BondlyTokenSale BondlyTokenSeedSale BondlyTokenP1Sale BondlyTokenP2Sale BondlyTokenBondlyCardGameSale BondlyTokenPreOfferingSale
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

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

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Bondly team

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts/	98.15	88.89	100.00	98.11	
BondlyToken.sol	100.00	100.00	100.00	100.00	
BondlyTokenAdvisorsPool.sol	100.00	100.00	100.00	100.00	
BondlyTokenBondlyCardGameSale.sol	100.00	100.00	100.00	100.00	
BondlyTokenEcosystemPool.sol	100.00	100.00	100.00	100.00	
BondlyTokenFoundersPool.sol	100.00	100.00	100.00	100.00	
BondlyTokenHolder.sol	93.75	90.00	100.00	93.33	43
BondlyTokenInitialDexSale.sol	100.00	100.00	100.00	100.00	
BondlyTokenP1Sale.sol	100.00	100.00	100.00	100.00	
BondlyTokenP2Sale.sol	100.00	100.00	100.00	100.00	
BondlyTokenPreOfferingSale.sol	100.00	100.00	100.00	100.00	
BondlyTokenSale.sol	96.97	86.36	100.00	96.88	61
BondlyTokenSeedSale.sol	100.00	100.00	100.00	100.00	
BondlyTokenStakingRewards.sol	100.00	100.00	100.00	100.00	
BondlyTokenTeamPool.sol	100.00	100.00	100.00	100.00	
All files	**98.15**	**88.89**	**100.00**	**98.11**	

Test Results

```
Contract: btFoundersPool
  Initialization
    ✓ should deploy BondlyToken (1079ms)
    ✓ should deploy Founders pool (395ms)
    ✓ should transfer maxCap funds to the pool (778ms)
  Send the tokens [month #1]
    ✓ should init tranches
    ✓ should not send tokens if fullLockMonths > 0 (368ms)
    ✓ should increaseTime to skip full lock and get successful mint (507ms)
    ✓ should correctly transfer leftover (865ms)
  Do not send portion 2 yet
    ✓ should increase time but not enough [29 days] (308ms)
  Send two portions [for 2 months]
    ✓ should increaseTime to +2 month (603ms)
    ✓ should not send anymore (313ms)
  check final portion
    ✓ should increaseTime exactly to get all tokens
    ✓ should match available balance (101ms)
  Send final portion
    ✓ should increaseTime to random amount to test overflow
    ✓ should not send [overflow] (280ms)
    ✓ should not show availability overflow (66ms)
    ✓ should send (233ms)
    ✓ should not send anymore, even 1 token (245ms)
    ✓ should be 0 now [getAvailableTokens] (80ms)

Contract: btAdvisorsPool
  Initialization
    ✓ should deploy BondlyToken (276ms)
    ✓ should deploy Advisors pool (228ms)
    ✓ should transfer maxCap funds to the pool (516ms)
  Send the tokens [month #1]
    ✓ should init tranches
    ✓ should not send tokens if fullLockMonths > 0 (189ms)
    ✓ should increaseTime to skip full lock and get successful mint (300ms)
    ✓ should correctly transfer leftover (748ms)
  Do not send portion 2 yet
    ✓ should increase time but not enough [29 days] (280ms)
  Send two portions [for 2 months]
    ✓ should increaseTime to +2 month (540ms)
    ✓ should not send anymore (253ms)
  check final portion
    ✓ should increaseTime exactly to get all tokens
    ✓ should match available balance (70ms)
  Send final portion
    ✓ should increaseTime to random amount to test overflow
    ✓ should not send [overflow] (142ms)
    ✓ should not show availability overflow (50ms)
    ✓ should send (542ms)
    ✓ should not send anymore, even 1 token (170ms)
    ✓ should be 0 now [getAvailableTokens] (47ms)

Contract: btTeamPool
  Initialization
    ✓ should deploy BondlyToken (314ms)
    ✓ should deploy Team pool (238ms)
    ✓ should transfer maxCap funds to the pool (297ms)
  Send the tokens [month #1]
    ✓ should init tranches
    ✓ should not send tokens if fullLockMonths > 0 (281ms)
    ✓ should increaseTime to skip full lock and get successful mint (831ms)
    ✓ should correctly transfer leftover (1369ms)
  Do not send portion 2 yet
    ✓ should increase time but not enough [29 days] (749ms)
  Send two portions [for 2 months]
    ✓ should increaseTime to +2 month (379ms)
    ✓ should not send anymore (778ms)
  check final portion
    ✓ should increaseTime exactly to get all tokens
    ✓ should match available balance (146ms)
  Send final portion
    ✓ should increaseTime to random amount to test overflow (100ms)
    ✓ should not send [overflow] (312ms)
    ✓ should not show availability overflow (53ms)
    ✓ should send (243ms)
    ✓ should not send anymore, even 1 token (535ms)
    ✓ should be 0 now [getAvailableTokens] (57ms)

Contract: btEcosystemPool
 Initialization
    ✓ should deploy BondlyToken (307ms)
    ✓ should deploy Ecosystem pool (265ms)
    ✓ should transfer maxCap funds to the pool (567ms)
 Send the tokens [month #1]
    ✓ should init tranches
    ✓ should not send tokens if fullLockMonths > 0
    ✓ should increaseTime to skip full lock and get successful mint (297ms)
    ✓ should correctly transfer leftover (633ms)
 Do not send portion 2 yet
    ✓ should increase time but not enough [29 days] (240ms)
 Send two portions [for 2 months]
    ✓ should increaseTime to +2 month (300ms)
    ✓ should not send anymore (499ms)
 check final portion
    ✓ should increaseTime exactly to get all tokens
    ✓ should match available balance (75ms)
 Send final portion
    ✓ should increaseTime to random amount to test overflow
    ✓ should not send [overflow] (133ms)
    ✓ should not show availability overflow (45ms)
    ✓ should send (234ms)
    ✓ should not send anymore, even 1 token (447ms)
    ✓ should be 0 now [getAvailableTokens] (52ms)

Contract: BondlyToken
 Initialization
    ✓ should deploy BondlyToken (283ms)
    ✓ should transfer some tokens (116ms)
    ✓ should NOT set burner [wrong owner] (107ms)
    ✓ should NOT burn tokens [wrong burner address] (97ms)
    ✓ should set burner
    ✓ should burn some tockens pool (196ms)
```

```
        ✓ should be correct balance (113ms)

Contract: btStakingRewards
    Initialization
        ✓ should deploy BondlyToken (291ms)
        ✓ should deploy StakingRewards pool (181ms)
        ✓ should transfer maxCap funds to the pool (173ms)
        ✓ should not send [overflow] (109ms)
    Send the tranches
        ✓ should init tranches
        ✓ should send without time increase #1 (473ms)
        ✓ should send without time increase #2 (202ms)
    Send final portion
        ✓ should not show availability overflow
        ✓ should send (133ms)
        ✓ should not send anymore, even 1 token (142ms)
        ✓ should be 0 now [getAvailableTokens] (62ms)

Contract: btBondlyCardGameSale
    Initialization
        ✓ should deploy BondlyToken (307ms)
        ✓ should deploy BondlyCardGame pool (325ms)
        ✓ should transfer maxCap funds to the pool (236ms)
        ✓ should init tranches (601ms)
    Adding Users
        ✓ should not add user [empty amount] (121ms)
        ✓ should add 5 users (1071ms)
        ✓ should get contributor (594ms)
        ✓ should not add user [duplicate] (185ms)
        ✓ should not add user [supply overflow] (456ms)
    Contributors - prepare
        ✓ should accept only contributors (256ms)
        ✓ should fill contributorsList (346ms)
    Get Available amount
        ✓ should show 0 for unexisting contributor (223ms)
        ✓ should show all available balance (104ms)
        ✓ should not send tokens if fullLockMonths > 0
    Send the tokens [month #1]
```

```
      ✓ should increaseTime to skip full lock and get successful mint (4177ms)
      ✓ should correctly mint2 of first month (2607ms)
      ✓ should correctly mint3 of first month (5992ms)
    Send the tokens [month #2]
      ✓ should NOT mint right away (210ms)
      ✓ should increase time but not enough [29 days] (280ms)
      ✓ should increaseTime to +1 day (30 total) and get successful mint (2265ms)
      ✓ should not mint, even 1 item (1427ms)
    Send the tokens [month #3-xx]
      ✓ should correctly mint leftover of other months (1785ms)
      ✓ check if any change left (851ms)
    Final Check
      ✓ should match maxCap and totalTransferred
      ✓ should match maxCap and totalReserved
      ✓ bondlyTokenSale balance should be 0
      ✓ should show all available balance (200ms)

  Contract: btBondlyTokenInitialDexSale
    Initialization
      ✓ should deploy BondlyToken (352ms)
      ✓ should deploy InitialDEX pool (241ms)
      ✓ should transfer maxCap funds to the pool (523ms)
      ✓ should init tranches
    Adding Users
      ✓ should not add user [empty amount] (118ms)
      ✓ should add 5 users (898ms)
      ✓ should get contributor (93ms)
      ✓ should not add user [duplicate] (122ms)
      ✓ should not add user [supply overflow] (139ms)
    Contributors - prepare
      ✓ should accept only contributors (136ms)
      ✓ should fill contributorsList (352ms)
    Get Available amount
      ✓ should show 0 for unexisting contributor (71ms)
      ✓ should show all available balance (47ms)
      ✓ should not send tokens if fullLockMonths > 0
    Send the tokens [month #1]
      ✓ should increaseTime to skip full lock and get successful mint (1910ms)
      ✓ should correctly mint2 of first month (1376ms)
      ✓ should correctly mint3 of first month (2892ms)
    Send the tokens [month #2]
      ✓ should NOT mint right away
      ✓ should increase time but not enough [29 days]
      ✓ should increaseTime to +1 day (30 total) and get successful mint
      ✓ should not mint, even 1 item (1427ms)
    Send the tokens [month #3-xx]
      ✓ should correctly mint leftover of other months
      ✓ check if any change left (717ms)
    Final Check
      ✓ should match maxCap and totalTransferred
      ✓ should match maxCap and totalReserved
      ✓ bondlyTokenSale balance should be 0
      ✓ should show all available balance (178ms)

  Contract: BondlyTokenP1Sale
    Initialization
      ✓ should deploy BondlyToken (382ms)
      ✓ should deploy P1 pool (242ms)
      ✓ should transfer maxCap funds to the pool (694ms)
      ✓ should init tranches
    Adding Users
      ✓ should not add user [empty amount] (134ms)
      ✓ should add 5 users (1053ms)
      ✓ should get contributor (398ms)
      ✓ should not add user [duplicate] (101ms)
      ✓ should not add user [supply overflow] (105ms)
    Contributors - prepare
      ✓ should accept only contributors (118ms)
      ✓ should fill contributorsList (178ms)
    Get Available amount
      ✓ should show 0 for unexisting contributor (64ms)
      ✓ should show all available balance (166ms)
      ✓ should not send tokens if fullLockMonths > 0
    Send the tokens [month #1]
      ✓ should increaseTime to skip full lock and get successful mint (2199ms)
      ✓ should correctly mint2 of first month (2021ms)
      ✓ should correctly mint3 of first month (5309ms)
    Send the tokens [month #2]
      ✓ should NOT mint right away (549ms)
      ✓ should increase time but not enough [29 days] (316ms)
      ✓ should increaseTime to +1 day (30 total) and get successful mint (3544ms)
      ✓ should not mint, even 1 item (1790ms)
    Send the tokens [month #3-xx]
      ✓ should correctly mint leftover of other months (1989ms)
      ✓ check if any change left (4060ms)
    Final Check
      ✓ should match maxCap and totalTransferred
      ✓ should match maxCap and totalReserved
      ✓ bondlyTokenSale balance should be 0
      ✓ should show all available balance (226ms)

  Contract: BondlyTokenP2Sale
    Initialization
      ✓ should deploy BondlyToken (227ms)
      ✓ should deploy P2 pool (203ms)
      ✓ should transfer maxCap funds to the pool (680ms)
      ✓ should init tranches
    Adding Users
      ✓ should not add user [empty amount] (113ms)
      ✓ should add 5 users (916ms)
      ✓ should get contributor (70ms)
      ✓ should not add user [duplicate] (95ms)
      ✓ should not add user [supply overflow] (294ms)
    Contributors - prepare
      ✓ should accept only contributors (122ms)
      ✓ should fill contributorsList (156ms)
    Get Available amount
      ✓ should show 0 for unexisting contributor (41ms)
      ✓ should show all available balance (44ms)
      ✓ should not send tokens if fullLockMonths > 0
    Send the tokens [month #1]
      ✓ should increaseTime to skip full lock and get successful mint (2218ms)
      ✓ should correctly mint2 of first month (1875ms)
      ✓ should correctly mint3 of first month (5145ms)
    Send the tokens [month #2]
      ✓ should NOT mint right away (455ms)
      ✓ should increase time but not enough [29 days] (192ms)
      ✓ should increaseTime to +1 day (30 total) and get successful mint (2020ms)
      ✓ should not mint, even 1 item (1177ms)
    Send the tokens [month #3-xx]
      ✓ should correctly mint leftover of other months (1978ms)
      ✓ check if any change left (775ms)
    Final Check
      ✓ should match maxCap and totalTransferred
      ✓ should match maxCap and totalReserved
      ✓ bondlyTokenSale balance should be 0
      ✓ should show all available balance (175ms)

  Contract: btPreOfferingSale
    Initialization
      ✓ should deploy BondlyToken (374ms)
      ✓ should deploy PreOffering pool (499ms)
      ✓ should transfer maxCap funds to the pool (228ms)
      ✓ should init tranches
    Adding Users
      ✓ should not add user [empty amount] (87ms)
      ✓ should add 5 users (800ms)
      ✓ should get contributor (90ms)
      ✓ should not add user [duplicate] (87ms)
      ✓ should not add user [supply overflow] (340ms)
    Contributors - prepare
      ✓ should accept only contributors (123ms)
      ✓ should fill contributorsList (156ms)
    Get Available amount
      ✓ should show 0 for unexisting contributor
      ✓ should show all available balance (49ms)
      ✓ should not send tokens if fullLockMonths > 0
    Send the tokens [month #1]
      ✓ should increaseTime to skip full lock and get successful mint (2293ms)
      ✓ should correctly mint2 of first month (2257ms)
      ✓ should correctly mint3 of first month (6151ms)
    Send the tokens [month #2]
      ✓ should NOT mint right away (402ms)
      ✓ should increase time but not enough [29 days] (234ms)
      ✓ should increaseTime to +1 day (30 total) and get successful mint (2838ms)
      ✓ should not mint, even 1 item (2784ms)
    Send the tokens [month #3-xx]
      ✓ should correctly mint leftover of other months (3832ms)
      ✓ check if any change left (1180ms)
    Final Check
      ✓ should match maxCap and totalTransferred (51ms)
      ✓ should match maxCap and totalReserved
      ✓ bondlyTokenSale balance should be 0
      ✓ should show all available balance (247ms)

  Contract: btSeedSale
    Initialization
      ✓ should deploy BondlyToken (370ms)
      ✓ should deploy Seed pool (1010ms)
      ✓ should transfer maxCap funds to the pool (893ms)
      ✓ should init tranches
    Adding Users
      ✓ should not add user [empty amount] (166ms)
      ✓ should add 5 users (1480ms)
      ✓ should get contributor (304ms)
      ✓ should not add user [duplicate] (351ms)
      ✓ should not add user [supply overflow] (150ms)
    Contributors - prepare
      ✓ should accept only contributors (216ms)
      ✓ should fill contributorsList (443ms)
    Get Available amount
      ✓ should show 0 for unexisting contributor (39ms)
      ✓ should show all available balance (49ms)
      ✓ should not send tokens if fullLockMonths > 0 (781ms)
    Send the tokens [month #1]
      ✓ should increaseTime to skip full lock and get successful mint (4019ms)
      ✓ should correctly mint2 of first month (3078ms)
      ✓ should correctly mint3 of first month (6722ms)
    Send the tokens [month #2]
      ✓ should NOT mint right away (222ms)
```

```
        ✓ should increase time but not enough [29 days] (505ms)
        ✓ should increaseTime to +1 day (30 total) and get successful mint (2395ms)
        ✓ should not mint, even 1 item (1915ms)
      Send the tokens [month #3-xx]
        ✓ should correctly mint leftover of other months (2087ms)
        ✓ check if any change left (4123ms)
      Final Check
        ✓ should match maxCap and totalTransferred (224ms)
        ✓ should match maxCap and totalReserved
        ✓ bondlyTokenSale balance should be 0
        ✓ should show all available balance (158ms)


  252 passing (3m)
```

Tests written by Zokyo Secured team

As part of our work assisting Bondly in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the Bondly contract requirements (for Audit) for details about issuance amounts and how the system handles these.

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts/	98.15	86.11	100.00	98.11	
BondlyToken.sol	100.00	100.00	100.00	100.00	
BondlyTokenAdvisorsPool.sol	100.00	100.00	100.00	100.00	
BondlyTokenBondlyCardGameSale.sol	100.00	100.00	100.00	100.00	
BondlyTokenEcosystemPool.sol	100.00	100.00	100.00	100.00	
BondlyTokenFoundersPool.sol	100.00	100.00	100.00	100.00	
BondlyTokenHolder.sol	93.75	90.00	100.00	93.33	43
BondlyTokenInitialDexSale.sol	100.00	100.00	100.00	100.00	
BondlyTokenP1Sale.sol	100.00	100.00	100.00	100.00	
BondlyTokenP2Sale.sol	100.00	100.00	100.00	100.00	
BondlyTokenPreOfferingSale.sol	100.00	100.00	100.00	100.00	
BondlyTokenSale.sol	96.97	81.82	100.00	96.88	61
BondlyTokenSeedSale.sol	100.00	100.00	100.00	100.00	
BondlyTokenStakingRewards.sol	100.00	100.00	100.00	100.00	
BondlyTokenTeamPool.sol	100.00	100.00	100.00	100.00	
All files	**98.15**	**86.11**	**100.00**	**98.11**	

Test Results

```
Contract: BondlyToken
  burn
    ✓ check burn (979ms)

Contract: TokenSale
  ✓ deploy contract & check (728ms)
  ✓ non contributor cannot claim (606ms)
  ✓ check contribution data (914ms)
  ✓ contributor can claim (1127ms)
  ✓ owner can send tokens to contributor (2644ms)
  contributor can claim amount for a several months
    ✓ check claim after two months (718ms)
  contributor can claim amount for a several months
    ✓ check claim after 5 months (610ms)
  contributor can claim seed tokens
    ✓ check claim after 1 months (625ms)
    ✓ check claim after 3 months (1182ms)
    ✓ check claim 11 months (891ms)
    ✓ check claim after 12 months (880ms)
    ✓ check claim after 14 months (681ms)
  contributor can claim p1 tokens
    ✓ check claim in 1st month (664ms)
    ✓ check claim after 2 months (1116ms)
    ✓ check claim after 12 months (716ms)
    ✓ check claim of full amout (468ms)
  check bondlyTokenBondlyCardGameSale
    ✓ check claim right after adding contributor (663ms)
    ✓ check claim after 1 months (975ms)
    ✓ check claim after 12 months (2750ms)

Contract: TokenSale
  owner can allocate bondlyTokenAdvisorsPool tokens
    ✓ check allocate after 12 months (1133ms)
    ✓ check before unlock period (274ms)
    ✓ check claim after 25 months (839ms)
```

```
Contract: BondlyTokenStakingRewards
    owner can allocate bondlyTokenStakingRewards tokens
        ✓ check getAvailableTokens (602ms)
        ✓ check send (949ms)


25 passing (2m)
```

We are grateful to have been given the opportunity to work with the Bondly team

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

Zokyo's Security Team recommends that the Bondly team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.