POOLS

SMART CONTRACT AUDIT

ZOKYO.

February 14th, 2022 | v. 1.0

PASS

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

TECHNICAL SUMMARY

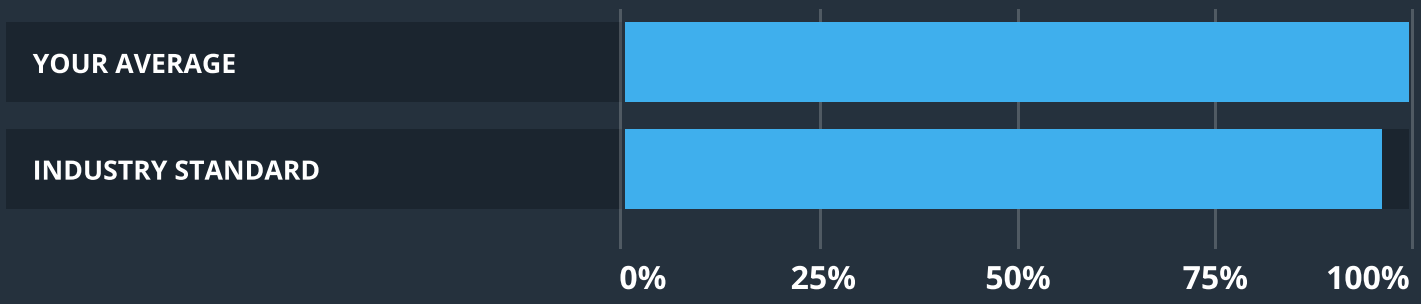
This document outlines the overall security of the Pools smart contracts, evaluated by Zokyo's Blockchain Security team.

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

Contract Status

There were critical, high and medium issues found during the audit.

Testable Code



The testable code is sufficient for 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 security of the contract we at Zokyo recommend that the Pools 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 Pools smart contract's source code was taken from the repository provided by the Pools team: https://github.com/p00ls/contracts

Commit (audited): e625f3b45016f2d173ecac4437dcd3b1768ccfaf

Commit (post-audited): e927b8e359a698262b2952a4d6d8a451383923ce

Within the scope of this audit Zokyo auditors have reviewed the following contract(s):

- P00lsDAO.sol
- P00lsTimelock.sol
- UniswapInterfaceMulticall.sol
- UniswapV2Factory.sol
- UniswapV2Pair.sol
- UniswapV2Router02.sol
- Auction.sol
- AuctionManager.sol
- Locking.sol
- Escrow.sol
- VestedAirdrops.sol
- P00lsCreatorRegistry.sol
- P00lsTokenBase.sol
- P00lsTokenCreator.sol
- P00lsTokenXCreator.sol
- interfaces.sol
- ERC1046Upgradeable.sol
- ERC1363Upgradeable.sol
- Beacon.sol
- BeaconProxy.sol
- RegistryOwnable.sol
- Math1.sol
- UniswapV2Library.sol
- UQ112x112.sol
- WETH.sol
- IWETH.sol
- ERC1363ReceiverMock.sol
- IERC1046.sol
- IERC1363.sol

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

- Implements and adheres to existing standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Follows best practices in efficient use of resources, 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 Pools 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. 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.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

EXECUTIVE SUMMARY

Contracts are well-structured and documented, have all necessary security checks and the overall code quality is very high. Nevertheless, auditors team has detected the unclear functionality which was verified with the Pools team. Though the auditors team has admitted very busy repository which (in order to keep the best practices) should be splitted into several packages.
Also, the auditors team has noticed several places which contradict the best practices for the decentralized protocol developement, though auditors have admitted that it was a necessary step for higher security insurance of contracts:

- token contracts P00lsTokenCreator and P00lsTokenXCreator are upgradeable;
- P00lsTokenXCreator has instant approve for P00lsTokenCreator contract

Though, after discussion with the Pools team the overall security of contracts was confirmed and verified, that there is no influence on contracts' security and efficiency.

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. Issues tagged “Verified” contain unclear or suspicious functionality that either needs explanation from the Customer’s side or it is an issue that the Customer disregards as an issue. 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

INFORMATIONAL | UNRESOLVED

Contract owner to transfer any amounts of tokens to anyone

contracts\tokens\P00lsTokenCreator.sol, transferFrom()
xCreatorToken can transfer any amount of tokens from anyone to anyone with no restrictions. Thus, token creator becomes overpowered.

Recommendation:
Consider changing the logic to secure token holders from overpowered admin actions.

Post-audit:
By the Pools team: the only context in which the xCreatorToken is going to call that is during `depositFor` or `withdrawTo` operation. Also, the overall logic was changed to have not the control over the transfers, but to have instant approve for the xCreatorToken.

HIGH | VERIFIED

Non clear function usage

contracts\tokens\P00lsTokenXCreator.sol
Contract implements IEscrowReceiver interface which has the only method onEscrowRelease() but its parameter is unused. Also the method is public, performs constant action (1 ether converted push) and has no restrictions, So everyone can use it and abuse the contracts storage.

Recommendation:
Review the functionality and clarify method usage and/or restrict it.

From Client:
`onEscrowRelease()` is called by the `Escrow` contract. This methods is designed with a parameter that is informative. In the case of the `P00lsTokenXCreator` we don't need the param so we just disregard it. This function is used to store the conversion rate in the history. This is not critical, and anybody could call it to "refresh the history" with no consequence at all. In particular, if there is no update to store, we are not even storing it.

MEDIUM | VERIFIED

Contracts with the same name

contracts\finance\amm\libraries\Math.sol
“Duplicate contract names found for Math”
“Duplicate contract names found for NameResolver”
Considered renaming of the contract in the repo, and consider usage of another package or split contract repos.

Recommendation:
Avoid duplicate namings

MEDIUM | VERIFIED

Unsafe conversion

contracts\utils\RegistryOwnable.sol, uint256ToAddress()
Function contains unsafe conversion from uint256 to uint160 - it lacks safe checks when we loose higher bytes of the input parameter. Though, the function is not used throughout the code, so should be either corrected or removed,

Recommendation:
Verify the functionality or remove the code.

MEDIUM | VERIFIED

Incorrect parameter

contracts\dao\P00lsDAO.sol, initializer
Initializer contains parameter for the voting period - 40320. Comment sign “1 week”, though the number does not refer for 1 week

Recommendation:
Verify the parameter and consider either parameter change or comment.

From Client:
Note that this value is a block number, it corresponds to 86400*7/15

INFORMATIONAL | VERIFIED

Compilation warning

StakingB_1.sol Line 512
Warning: Function state mutability can be restricted to pure
--> contracts/tokens/P00lsTokenXCreator.sol:150:5:
Warning: Function state mutability can be restricted to pure
--> contracts/tokens/P00lsTokenXCreator.sol:157:5:

Recommendation:
Change functions visibility to pure

From Client:
This is overriding existing code, removing a functionality that is accessed differently. The function definition is kept identical for clarity sake.

	P00lsDAO.sol	P00lsTimelock.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Delegatecall Unexpected Ether	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

	UniswapInterface Multicall.sol	UniswapV2Factory.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Delegatecall Unexpected Ether	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

	UniswapV2Pair.sol	UniswapV2Router02.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Delegatecall Unexpected Ether	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

	Auction.sol	AuctionManager.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Delegatecall Unexpected Ether	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

	Locking.sol	Escrow.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Delegatecall Unexpected Ether	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

	VestedAirdrops.sol	P00lsCreatorRegistry.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Delegatecall Unexpected Ether	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

	P00lsTokenBase.sol	P00lsTokenCreator.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Delegatecall Unexpected Ether	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

	P00lsTokenXCreator.sol	RegistryOwnable.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Delegatecall Unexpected Ether	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 the Pools team

As part of our work assisting Pools team in verifying the correctness of their contract code, our team was responsible for the verification of the existing unit tests coverage. The code has a sufficient native unit test coverage. Nevertheless, Zokyo Security team has checked every unit test for its correctness and if it has sence in the context of Pools protocol. The overall quality of tests is high and the security has been confirmed. Though Zokyo Security team has provided own exploratory testing over the Pools unit tests set.

```
Contract: ENS
  setNam
    ✓ setName for registry (189ms)
    ✓ setName for token (110ms)
    ✓ setName for xToken (108ms)
    ✓ setName for vesting (139ms)
    ✓ setName for escrow (134ms)
    ✓ setName for auction (121ms)
    ✓ setName for locking (127ms)
Contract: AMM
  with social token
  with dutch auction
    ✓ finalize too early
    ✓ finalize with funds (298ms)
Contract: Auction
    ✓ get instance
    ✓ cannot star instance without a balance
    ✓ eth payments are locked
  before auction
    ✓ check balances
  with auction
    ✓ get instance
    ✓ check balances
  before auction ends
    ✓ can commit by sending eth directly (64ms)
    ✓ can commit
    ✓ can leave (84ms)
    ✓ cannot withdraw (68ms)
    ✓ cannot finalize
```

after auction end
- ✓ cannot commit
- ✓ cannot commit by sending eth directly
- ✓ cannot leave
- ✓ can withdraw (41ms)
- ✓ can finalize (225ms)

Contract: Locking

before lock setup
- ✓ empty lock details
- ✓ empty vault details
- ✓ unauthorized cannot setup lock (51ms)
- ✓ admin can setup lock (58ms)
- ✓ cannot setup vault
- ✓ cannot deposit
- ✓ cannot withdraw

after lock setup
- ✓ lock details
- ✓ empty vault details
- ✓ cannot re setup lock
- ✓ cannot setup vault with invalid duration
- ✓ can setup vault
- ✓ cannot deposit
- ✓ cannot withdraw

after lock setup
- ✓ lock details
- ✓ vault details
- ✓ more vault details
- ✓ cannot re setup lock
- ✓ cannot re setup vault
- ✓ can deposit (159ms)
- ✓ can deposit creator token with erc1363's transferAndCall (44ms)
- ✓ can deposit extra token with erc1363's transferAndCall (38ms)
- ✓ can deposit creator token with erc1363's approveAndCall (56ms)
- ✓ can deposit extra token with erc1363's approveAndCall (49ms)
- ✓ protected against invalid erc1363's calls (228ms)
- ✓ cannot withdraw

after deposit
- ✓ check status
- ✓ can deposit more (129ms)
- ✓ can deposit more extra (142ms)
- ✓ cannot withdraw

after delay
✓ cannot deposit anymore
✓ cannot withdraw
after expiration
✓ can withdraw to someone else
✓ relative withdraw (71ms)

Contract: Duration
✓ 3 months (127ms)
✓ 6 months (130ms)
✓ 9 months (118ms)
✓ 12 months (123ms)
✓ 15 months (123ms)
✓ 18 months (123ms)
✓ 21 months (136ms)
✓ 24 months (134ms)
✓ 27 months (125ms)
✓ 30 months (123ms)
✓ 33 months (120ms)
✓ 36 months (125ms)

Contract: Extra
✓ factor 0 (132ms)
✓ factor 1 (226ms)
✓ factor 2 (226ms)
✓ factor 3 (233ms)
✓ factor 4 (221ms)
✓ factor 5 (233ms)
✓ factor 6 (230ms)
✓ factor 7 (221ms)
✓ factor 8 (222ms)
✓ factor 9 (665ms)

Contract: Staking
with social token
✓ creator → xCreator (43ms)
with staking plan
✓ staking details
✓ reconfigure protected
deposit & withdraw
✓ no wait + release half
✓ no wait + release all
✓ wait start + release half
✓ wait start + release all
✓ wait halfway + release half

```
    ✓ wait halfway + release all
    ✓ wait stop + release half
    ✓ wait stop + release all
  multi user
    ✓ scenario 1 (131ms)
    ✓ scenario 2 (115ms)
    ✓ scenario 3a (212ms)
    ✓ scenario 3b (219ms)
    ✓ scenario 4 (413ms)
Contract: Vested airdrop
    ✓ restricted access to admin function (60ms)
    ✓ can enable & disable airdrop
    ✓ cannot unlock for disabled airdrop
  with airdrop enabled
    ✓ unlock all at once
    ✓ apply schedule (615ms)
Contract: $00 Token
    ✓ Check social token (103ms)
  Votes counting
    ✓ support token and xtoken (241ms)
Contract: $Crea Token
    ✓ check
  with collection
  Check state
    ✓ Creator registry (148ms)
    ✓ Creator token
    ✓ Creator xToken
  Metadata
  registry
    ✓ Authorized (67ms)
    ✓ Protected (44ms)
  creatorToken
    ✓ Authorized
    ✓ Protected
  xCreatorToken
    ✓ Authorized
    ✓ Protected
  Transfer ownership
  Creator registry
    ✓ Protected
    ✓ Authorized
```

```
Creator token
  ✓ Protected
  ✓ Authorized (204ms)
Claiming
  ✓ protected against invalid proof and replay (153ms)
Delegation
  ✓ delegate on creator affect xcreator
  ✓ delegation hook is protected
  ✓ delegation on xcreator is disabled
ERC1363
transferAndCall
  ✓ without data
  ✓ with data
  ✓ with reverting hook
transferFromAndCall
  ✓ without data
  ✓ with data
  ✓ with reverting hook
approveAndCall
  ✓ without data
  ✓ with data
  ✓ with reverting hook
131 passing (2m)
```

FILE	% STMTS	% BRANCH	% FUNCS	% LINES/uncovered lines
dao\	55	100	23.08	55
P00lsDAO.sol	55	100	23.08	55
P00lsTimelock.sol	100	100	100	100
finance\auction\	98	93.75	92.86	98.11
Auction.sol	95.83	100	87.5	96
AuctionManager.sol	100	87.5	100	100
finance\locking\	93.51	93.75	91.3	92.5
Locking.sol	93.51	93.75	91.3	92.5
finance\staking\	100	78.57	100	100
Escrow.sol	100	78.57	100	100

finance\vesting\	100	100	100	100
VestedAirdrops.sol	100	100	100	100
tokens\	97.47	87.5	93.02	96.34
P00lsCreatorRegistry.sol	95.83	100	85.71	95.83
P00lsTokenBase.sol	100	100	100	100
P00lsTokenCreator.sol	100	100	100	100
P00lsTokenXCreator.sol	96.88	66.67	94.12	94.29
interfaces.sol	100	100	100	100
RegistryOwnable.sol	78.57	50	78.57	72.22 12,25,26,67,68

The overall unit-test coverage was checked to be sufficient for the industry standard, its correctness was verified and it was verified to cover all critical and crucial cases for the contracts set.

We are grateful to have been given the opportunity to work with the Pools 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 Pools team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.