SMART CONTRACT AUDIT

ZOKYO.

May 21, 2021 | 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

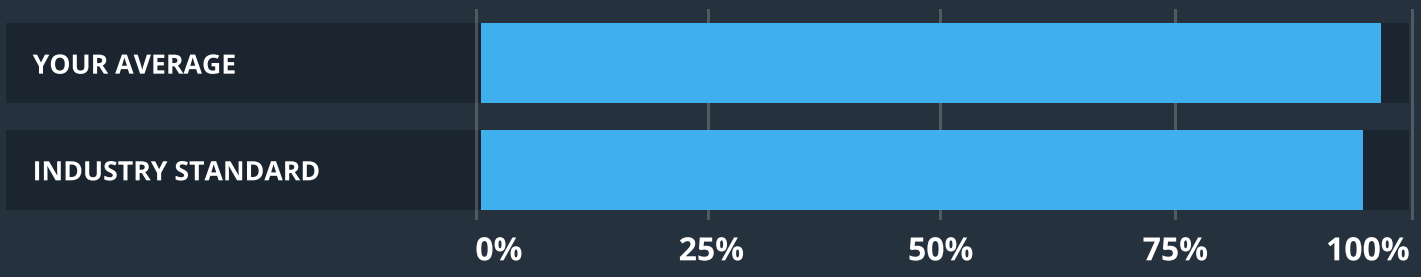
This document outlines the overall security of the DAOventures smart contracts, evaluated by Zokyo's Blockchain Security team.

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

Contract Status

There were no critical issues found during the audit.

Testable Code



Testable code is 96.99% 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 DAOventures 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 DAOventures repository – https://github.com/daoventures/DVG/commit/869834ac362a4297684f676b9e96beea14f62de4.
Last commit – 5610173f574808f5c0a00571b2b80c4d9f69bfb3.

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 DAOventures 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.	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

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. We especially want to note the good style of the code and its adaptation to the best practices.

The findings during the audit have no impact on contract performance or security, so it is fully production-ready.

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

Unlimit mint in DVGToken.sol

HIGH | RESOLVED

The contract owner can mint unconditionally tokens to any address.

Recommendation:
Add restrictions, if possible. If not, add a comment that briefly explains how users are protected from unexpected minting.

Math could fail due to negative number in DVGToken.sol

MEDIUM | RESOLVED

At function _moveDelegates, line 220 srcRepOld can be 0 or less than amount:

```
uint256 srcRepOld = srcRepNum > 0 ? checkpoints[srcRep][srcRepNum - 1].votes : 0;
```

While line 221 expects srcRepOld to be >= amount:

```
uint256 srcRepNew = srcRepOld.sub(amount);
```

Recommendation:
Depending on requirements, add 'revert' or 'if' to correctly handle possible negative value.

Potential gas overflow due to unlimited array length

MEDIUM | RESOLVED

In function massUpdatePools() there is a loop that depends on pool.length. In the case significant amount of pools, this function potentially could not be completed due to gas limits.

Recommendation:
Add a comment that confirms that no significant pool amount expected. Otherwise, add offset and limit parameters, to update pools portionally.

SPDX license identifier not provided in DVGToken.sol

LOW | RESOLVED

Trust in smart contracts can be better established if their source code is available. Since making source code available always touches on legal problems with regards to copyright, the Solidity compiler encourages the use of machine-readable SPDX license identifiers.

Recommendation:
Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.

Unnecessary ‘public’ keyword for constructor of DAOstake contract

LOW | RESOLVED

Visibility (public / external) is not needed for constructors anymore: To prevent a contract from being created, it can be marked abstract. This makes the visibility concept for constructors obsolete.

Recommendation:
Remove ‘public’ keyword from constructor.

Wrong comment in DVGToken.sol

INFORMATIONAL | RESOLVED

At line 73 there is a comment *“Delegate votes from `msg.sender` to `delegatee`”*, but actually delegates function doesn’t delegate anything and doesn’t work with msg.sender. (delegates does).

Recommendation:
Change comment to match actual function behaviour.

Same code doubled twice DAOstake.sol

INFORMATIONAL | RESOLVED

At function updatePool(uint256 _pid), lines 265 and 271 contains the same code:

```
dvg.mint(treasuryWalletAddr,
totalDVG.mul(TREASURY_WALLET_PERCENT).div(10000));
```

Recommendation:
Move this code before or below if condition.

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by DAOventures team (original test coverage)

Contract: **DAOstake for failure**
- ✓ Should fail to set wallet address with wrong params (66ms)
- ✓ Should fail to withdraw without enough balance (89ms)
- ✓ Should fail to add new pool with wrong params (94ms)

Contract: **DVGToken**

DVG token smart contract address: 0x36C02dA8a0983159322a80FFE9F24b1acfF8B570
- ✓ Should mint some DVGs in advance when deploying DVGToken smart contract (48ms)

Contract: **DAOstake for success**
- ✓ Should succeed to do setups (140ms)
- ✓ Should succeed to transfer DVG ownership
- ✓ Should succeed to add new pools (82ms)
- ✓ Should succeed to deposit (163ms)
- ✓ Should succeed to withdraw (212ms)
- ✓ Should succeed to emergency withdraw (77ms)
- ✓ Should record, mint and distribute DVGs properly (899ms)

Contract: **DVGToken**

DVG token smart contract address: 0xffa7CA1AEEEbBc30C874d32C7e22F052BbEa0429
- ✓ Should mint some DVGs in advance when deploying DVGToken smart contract (43ms)

12 passing (6s)

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts\	63.86	42.42	68.97	64.63	
DAOstake.sol	96.08	72.22	100.00	98.00	240, 379
DVGToken.sol	11.11	6.67	25.00	11.11	... 261, 262, 263
LPTokenMockup.sol	100.00	100.00	100.00	100.00	
contracts\interfaces\	100.00	100.00	100.00	100.00	
IDVGToken.sol	100.00	100.00	100.00	100.00	
All files	**63.86**	**42.42**	**68.97**	**64.63**	

Tests written by Zokyo Secured team

As part of our work assisting DAOventures 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 DAOventures contract requirements for details about issuance amounts and how the system handles these.

Contract: DAOstake

- ✓ setup (2802ms)
- ✓ should mint (478ms)
- ✓ should set treasure and community addreses (598ms)
- ✓ should set new owner to DVG (277ms)

On pools

- ✓ should create pools (2470ms)
- ✓ should add pools (1466ms)
- ✓ should NOT add pool [pool is not contract] (363ms)
- ✓ should change pool weight [without update] (514ms)
- ✓ should change pool weigth [with update] (1231ms)

On deposit/withdrow

- ✓ should deposit [0 eth] (262ms)
- ✓ should deposit (667ms)
- ✓ should deposit [second deposit by same user] (832ms)
- ✓ should withdraw (755ms)
- ✓ should withdraw [0 eth] (438ms)
- ✓ should NOT withdraw [not enought balance] (527ms)
- ✓ should call emergencyWithdraw (372ms)
- ✓ should deposit [many] (680ms)
- ✓ should withdrow (619ms)

Other

- ✓ should call pendingDVG (442ms)
- ✓ should call getMultiplier (668ms)
- ✓ should call poolLength (82ms)

Ownership

- ✓ should transfer DVG ownership (523ms)

```
Contract: DVGToken
  ✓ setup (2947ms)
  ✓ should get tokens (312ms)
  ✓ should NOT create DVGToken [Treasure address is contract] (359ms)
  On delegate
    success
      ✓ should delegate (417ms)
      ✓ should delegate x2 (373ms)
      ✓ should delegate x3 (350ms)
      ✓ should delegate x4 (309ms)
      ✓ should delegate x5 (393ms)
      ✓ should delegate with sight (511ms)
    fails
      ✓ should NOT delegate [signature expired] (385ms)
      ✓ should NOT delegate [invalid nonce] (546ms)
      ✓ should NOT delegate [invalid signature] (580ms)
  on votes
    ✓ should call getCurrentVotes (143ms)
    ✓ should NOT call getPriorVotes [block not mined] (189ms)
    ✓ should call getPriorVotes [block < first block] (183ms)
    ✓ should call getPriorVotes (823ms)

Contract: LPToken
  ✓ setup (506ms)
  ✓ should mint (413ms)

40 passing (30s)
```

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts\	96.99	83.33	100.00	96.95	
DAOstake.sol	97.06	80.56	100.00	97.00	219, 256, 379
DVGToken.sol	96.83	86.67	100.00	96.83	195, 246
LPTokenMockup.sol	100.00	100.00	100.00	100.00	
contracts\interfaces\	100.00	100.00	100.00	100.00	
IDVGToken.sol	100.00	100.00	100.00	100.00	
All files	**96.99**	**83.33**	**100.00**	**96.95**	

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

ZOKYO.