Umami DAO

SMART CONTRACT AUDIT

ZOKYO.

September 16th, 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.

SCORE

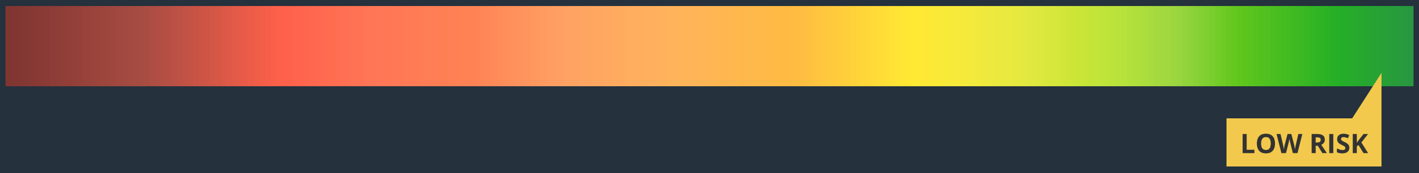
92

TECHNICAL SUMMARY

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

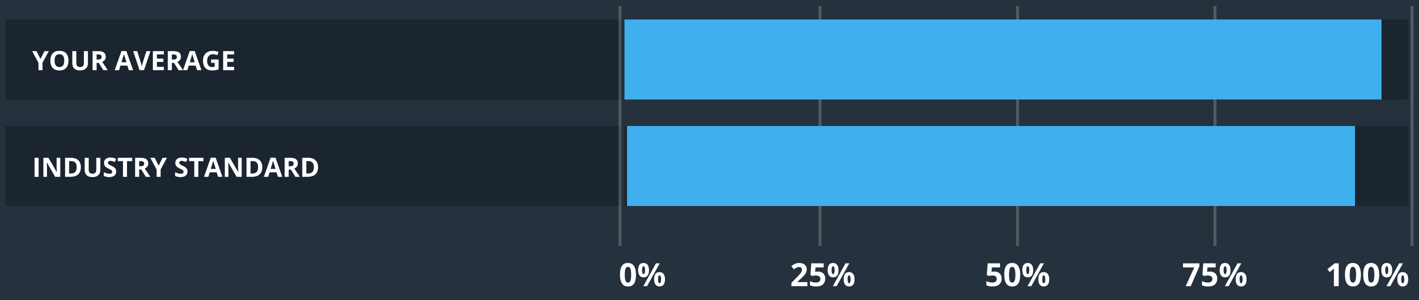
The scope of this audit was to analyze and document the Umami DAO smart contract codebase for quality, security, and correctness.

Contract Status



There were no critical issues found during the audit. (See Complete Analysis)

Testable Code



The testable code is 98.55%, which is above the industry standard of 95%. (See Complete Analysis)

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 Umami DAO 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 Umami DAO repository.

Repository - https://github.com/Arbi-s/zerotwohm-contracts

Last commit - 602c110b8cd13b872f6345221c1be6f6851bf944

Contracts under the scope:

- UMAMI.sol

Throughout the review process, Zokyo Security ensures that the 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 inefficient 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 Umami DAO 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

Zokyo team has run a deep auditing of Umami DAO's smart contracts. The contracts are well written and structured.

During the auditing process, there were some issues with medium and low severity and informational issues found. The Umami token has been deployed early in 2022. Since there are no major issues found, Umami DAO team do not plan on making any changes or any future migrations to the deployed token.

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 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

MEDIUM | RESOLVED

contract VaultOwned - in body of
setVault(address vault_) external returns (bool) onlyOwner
Too much authority for one executor that might lead to funds embezzled if not trusted

Recommendation:
Use multisig

Fix-1: according to partner's response, The Umami DAO Multisig wallet is the address currently set as the vault for the Umami token. All token operations are controlled through the DAO and through governance voting

Therefore, there's no change needs to be made

LOW | UNRESOLVED

Library Counters - in body of
increment(Counter storage counter) internal

Increment is executed without using the SafeMath library.

PS: Unless that is intended to save computation cost as overflow is almost impossible to take place !

LOW | INVALID

contract VaultOwned - in body of
setVault(address vault_) external returns (bool)
no validation done on vault_ address to make sure it is non-zero address

Fix-1: According to Partner this is intended by design. As vault_ shall be set to Zero Address when Max Supply is determined. Hence no more minting undergone by vault (zero address).

LOW | UNRESOLVED

library SafeMath - in body of
percentageAmount(uint256 total_, uint8 percentage_)
substractPercentage(uint256 total_, uint8 percentageToSub_)
logic of these functions assume the portion to be calculated from 1000 according to this div(mul(total_, percentage_), 1000);
as it is expected to be 100 also according to how percentageOfTotal is implemented.

INFORMATIONAL | UNRESOLVED

library Counters - in body of
current(Counter storage counter) internal view returns
(uint256)

No need to receive a storage argument in function. Recommendation - replace by memory

INFORMATIONAL | UNRESOLVED

contract ERC20 -
ERC20TOKEN_ERC1820_INTERFACE_ID is never used

INFORMATIONAL | UNRESOLVED

contract UMAMI - definition of
function _burnFrom(address account_, uint256 amount_) public

From a code organization point of view, this might have been intended to be private since burnFrom is already public.

	UMAMI.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 team

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

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

```
contract: UMAMI
    Initialization
        ✓ Should have correct initial values (167ms)
    mint()
        ✓ mints specified amount of token to specified address (206ms)
    sburn()
        ✓ burn specified amount of tokens from sender's account (86ms)
    burnFrom
        ✓ Should burn specified allowance from sender's account (167ms)
    increaseAllowance()
        ✓ increases allowances for specified account (151ms)
    decreaseAllowance()
        ✓ Should decrease allowances for account (84ms)
    transfer()
        ✓ transfer to account specified amount of tokens (68ms)
    PERMIT_TYPEHASH()
        ✓ permit_Typehash
    DOMAIN_SEPARATOR()
        ✓ DOMAIN_SEPARATOR
    permit()
        ✓ Should approve tokens for recipient by signing permit (109ms)
    setVault
        ✓ updates vault address (53ms)
    transferOwnership()
        ✓ Should transfer contract ownership (84ms)
    renounceOwnership
        ✓ enounceOwnership
    approve
        ✓ approve

14 passing (4s)
```

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	% UNCOVERED LINES
UMAMI.sol	98.55	81.82	100	98.61	
All files	**98.55**	**81.82**	**100**	98.61	

We are grateful to have been given the opportunity to work with the Umami DAO 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's Security Team recommends that the Umami DAO team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.