CONTRACT WOLF

Blockchain Security - Smart Contract Audits

U/S

DEX with confidence

Security Assessment

April 6, 2022

Disclaimer

ContractWolf.io audits and reports should not be considered as a form of project's "advertisement" and does not cover any interaction and assessment from "project's contract" to "external contracts" such as Pancakeswap or similar.

ContractWolf does not provide any warranty on its released reports.

ContractWolf should not be used as a decision to invest into an audited project and is not affiliated nor partners to its audited contract projects.

ContractWolf provides transparent report to all its "Clients" and to its "Clients Participants" and will not claim any guarantee of bug-free code within its **SMART CONTRACT**.

ContractWolf presence is to analyze, audit and assess the client's smart contract's to find any underlying risk and to eliminate any logic and flow errors within its code.

Each company or projects should be liable to its security flaws and functionalities.

Scope of Work

UniverseSwap team agreed and provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract.

The goal of this engagement was to identify if there is a possibility of security flaws in the implementation of the contract or system.

ContractWolf will be focusing on contract issues and functionalities along with the projects claims from smart contract to their website, whitepaper and repository which has been provided by **UniverseSwap**.

Project Description

The unique and potential platform on Filecoin, UniverseSwap combines liquidity of aggregators and allows very large exchanges with minimal slippage.

Risk Level Classification

Risk Level represents the classification or the probability that a certain function or threat that can exploit vulnerability and have an impact within the system or contract.

Risk Level is computed based on CVSS Version 3.0

Level	Value	Vulnerability
Critical	9 - 10	An Exposure that can affect the contract functions in several events that can risk and disrupt the contract
High	7 - 8.9	An Exposure that can affect the outcome when using the contract that can serve as an opening in manipulating the contract in an unwanted manner
Medium	4 - 6.9	An opening that could affect the outcome in executing the contract in a specific situation
Low	0.1 - 3.9	An opening but doesn't have an impact on the functionality of the contract
Informational	0	An opening that consists of information's but will not risk or affect the contract

Auditing Approach

Every line of code along with its functionalities will undergo manual review to check its security issues, quality, and contract scope of inheritance. The manual review will be done by our team that will document any issues that there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:

    - Review of the specifications, sources, and instructions provided to ContractWolf to make sure we understand the size, scope, and functionality of the smart contract.
    - Manual review of code, our team will have a process of reading the code line-by-line with the intention of identifying potential vulnerabilities and security flaws.

2. Testing and automated analysis that includes:

    - Testing the smart contract functions with common test cases and scenarios, to ensure that it returns the expected results.

3. Best practices review, the team will review the contract with the aim to improve efficiency, effectiveness, clarifications, maintainability, security, and control within the smart contract.

4. Recommendations to help the project take steps to secure the smart contract.

Used Code from other Frameworks/Smart Contracts (Direct Imports)

Imported Packages

- SafeMath

Description

Optimization enabled: No

Decimal: 18

Symbol: UNID

Max / Total Supply: 100,000,000

Capabilities

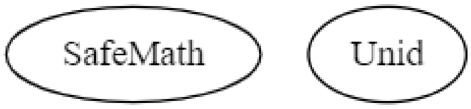
Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	1	0	0

Capabilities

Version	Solidity Versions Observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	v0.5.17		Yes	No	No

Inheritance Graph



Correct implementation of Token Standard

Tested	Verified
✓	✓

Overall Checkup (Smart Contract Security)

Tested	Verified
✓	✓

Function	Description	Exist	Tested	Verified
TotalSupply	Information about the total coin or token supply	✓	✓	✓
BalanceOf	Details on the account balance from a specified address	✓	✓	✓
Transfer	An action that transfers a specified amount of coin or token to a specified address	✓	✓	✓
TransferFrom	An action that transfers a specified amount of coin or token from a specified address	✓	✓	✓
Approve	Provides permission to withdraw specified number of coin or token from a specified address	✓	✓	✓

Verify Claims

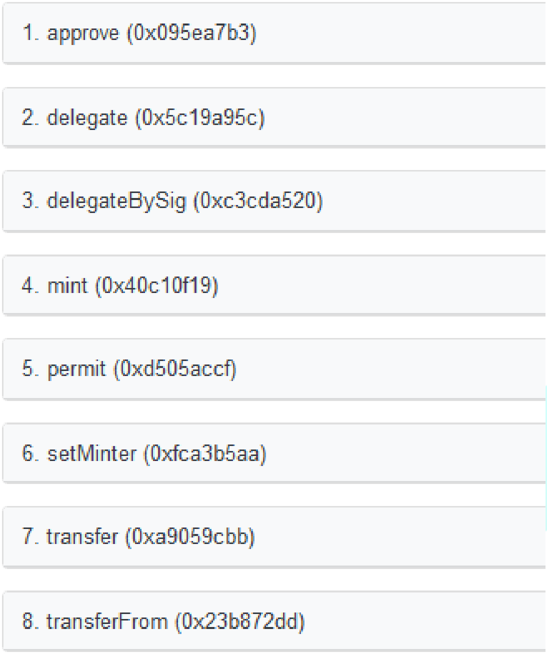
Statement	Exist	Tested	Owner
Renounce Ownership	–	–	–
Mint	✓	✓	✓
Burn	–	–	–

Block	–	–	–
Pause	–	–	–

Legend

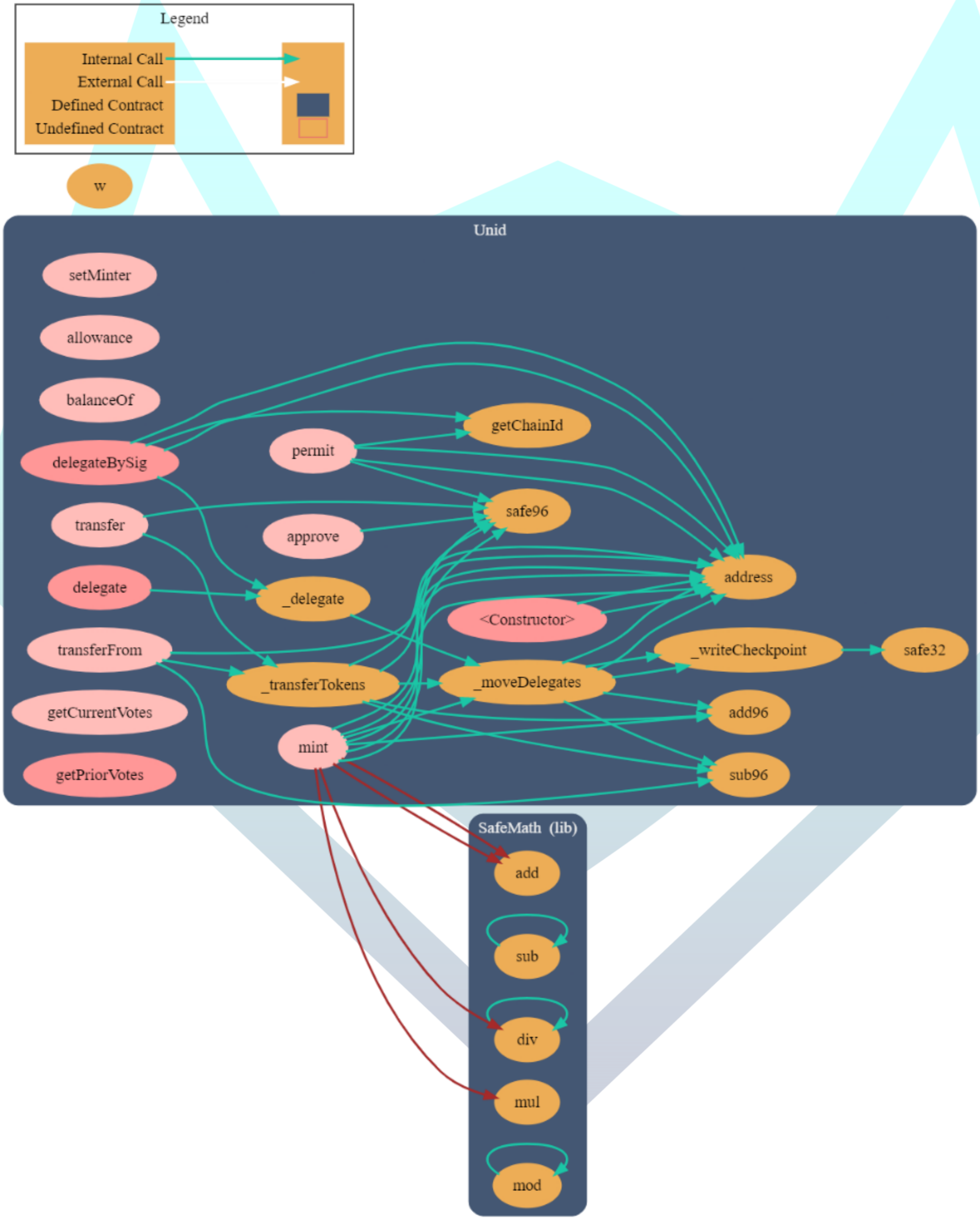
Attribute	Symbol
Verified / Can	✓
Verified / Cannot	✗
Unverified / Not checked	⚑
Not Available	–

Write Functions of Contract



1. approve (0x095ea7b3)
2. delegate (0x5c19a95c)
3. delegateBySig (0xc3cda520)
4. mint (0x40c10f19)
5. permit (0xd505accf)
6. setMinter (0xfca3b5aa)
7. transfer (0xa9059cbb)
8. transferFrom (0x23b872dd)

Call Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
w
Unid
setMinter
allowance
balanceOf
delegateBySig
permit
getChainId
transfer
approve
safe96
delegate
_delegate
address
<Constructor>
transferFrom
_writeCheckpoint
safe32
_transferTokens
_moveDelegates
getCurrentVotes
add96
mint
getPriorVotes
sub96
SafeMath (lib)
add
sub
div
mul
mod

Smart Contract Weakness Classification and Test Cases Attacks

ID	Title	Status
SWC-136	Unencrypted Private Data On-Chain	PASSED
SWC-135	Code With No Effects	PASSED
SWC-134	Message call with hardcoded gas amount	PASSED
SWC-133	Hash Collisions with Multiple Variable Length Arguments	PASSED
SWC-132	Unexpected Ether balance	PASSED
SWC-131	Presence of unused variables	PASSED
SWC-130	Right-To Left Override control character (U+202E)	PASSED
SWC-129	Typographical Error	PASSED
SWC-128	DoS With Block Gas Limit	PASSED
SWC-127	Arbitrary Jump with Function Type Variable	PASSED
SWC-126	Insufficient Gas Griefing	PASSED
SWC-125	Incorrect Inheritance Order	PASSED
SWC-124	Write to Arbitrary Storage Location	PASSED
SWC-123	Requirement Violation	PASSED
SWC-122	Lack of Proper Signature Verification	PASSED
SWC-121	Missing Protection against Signature Replay Attacks	PASSED
SWC-120	Weak Sources of Randomness from Chain Attributes	LOW ISSUE
SWC-119	Shadowing State Variables	PASSED
SWC-118	Incorrect Constructor Name	PASSED
SWC-117	Signature Malleability	PASSED
SWC-116	Block values as a proxy for time	PASSED
SWC-115	Authorization through tx.origin	PASSED
SWC-114	Transaction Order Dependence	PASSED
SWC-113	DoS with Failed Call	PASSED
SWC-112	Delegate call to Untrusted Callee	PASSED
SWC-111	Use of Deprecated Solidity Functions	PASSED
SWC-110	Assert Violation	PASSED

SWC-109	Uninitialized Storage Pointer	PASSED
SWC-108	State Variable Default Visibility	PASSED
SWC-107	Reentrancy	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	PASSED
SWC-105	Unprotected Ether Withdrawal	PASSED
SWC-104	Unchecked Call Return Value	PASSED
SWC-103	Floating Pragma	LOW ISSUE
SWC-102	Outdated Compiler Version	PASSED
SWC-101	Integer Overflow and Underflow	PASSED
SWC-100	Function Default Visibility	PASSED

Audit Result

THIS PROJECT IS AUDITED VIA LOCAL FILE AND NOT YET DEPLOYED IN LIVE NET

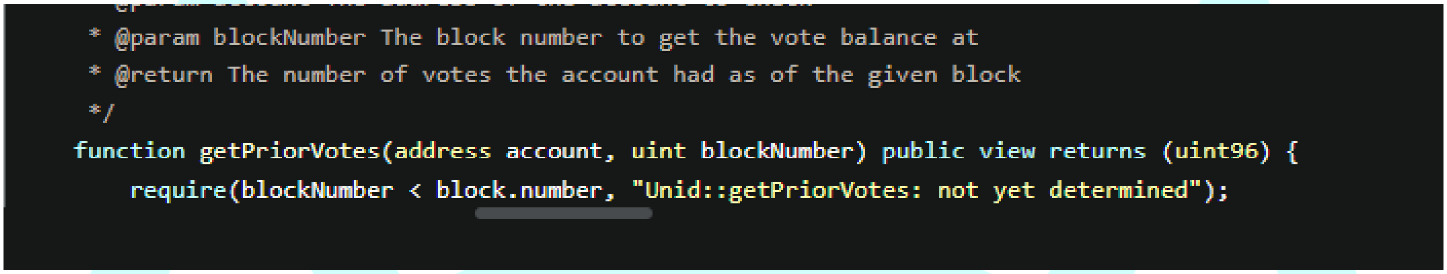
Low Issues

A floating pragma is set (SWC-103)	L: 1	UNID.sol
Weak Sources of Randomness from Chain Attributes (SWC-120)	L: 467, 540	UNID.sol

Findings

Description:

Weak Sources of Randomness from Chain Attributes (SWC-120)



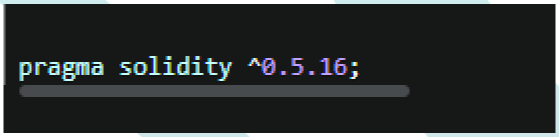
```
     * @param account The address of the account to check
     * @param blockNumber The block number to get the vote balance at
     * @return The number of votes the account had as of the given block
     */
    function getPriorVotes(address account, uint blockNumber) public view returns (uint96) {
        require(blockNumber < block.number, "Unid::getPriorVotes: not yet determined");
```

Suggestion:

Usage of block.number and block.timestamp is insecure and should be duly avoided.

Description:

A floating pragma is set (SWC-103)



```
pragma solidity ^0.5.16;
```

Suggestion:

Use specific version to ensure that the bytecode produced does not vary between builds.

Additional findings

Deprecated keywords
Outdated Contract compiler version(v0.5.16)

In an event of an upgrade, “block.number” should be replaced by “block.timestamp”

Audit Comments

- Owner can add minter address after the deployment
- Contract does not have fees/taxes
- Owner cannot change max transaction
- Owner cannot pause the contract
- Minter can mint after initial deployment
- Contract does not have a burning function
- Ownership cannot be transferred and renounced
- Owner cannot block users
- Contract is not using any antibots

CONTRACTWOLF

Blockchain Security - Smart Contract Audits