Vitalik Security Audit

Skylabs VSL

Manual Code Review

Sunday, 28 August 2022

Website: https://vetter.ai/skylabs

Twitter: https://twitter.com/vetterplatform?lang=en

Telegram: https://t.me/skylabsvsl

Severity Criteria

Vitalik assesses severity of disclosed vulnerabilities according to a methodology based on OWASP standards.

Vulnerabilities are divided into 3 primary risk categories:

- Low
- Medium
- High

High-level considerations for vulnerabilities span the following key areas when conducting

assessments:

- Malicious Input Handling
- Escalation of privileges
- Arithmetic
- Gas use

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

Details

Name : Skylabs VSL Token

Symbol: VSL

Total Supply: 25,000,000,000

Contract Address:

0xcC4186DD627c3046ae191c9a52382B10ABc50784

Contract SHA256 Checksum:

5325476b140649b0ddc31c97e1c13282141053e70f93f0c706c99fae27c1b38b

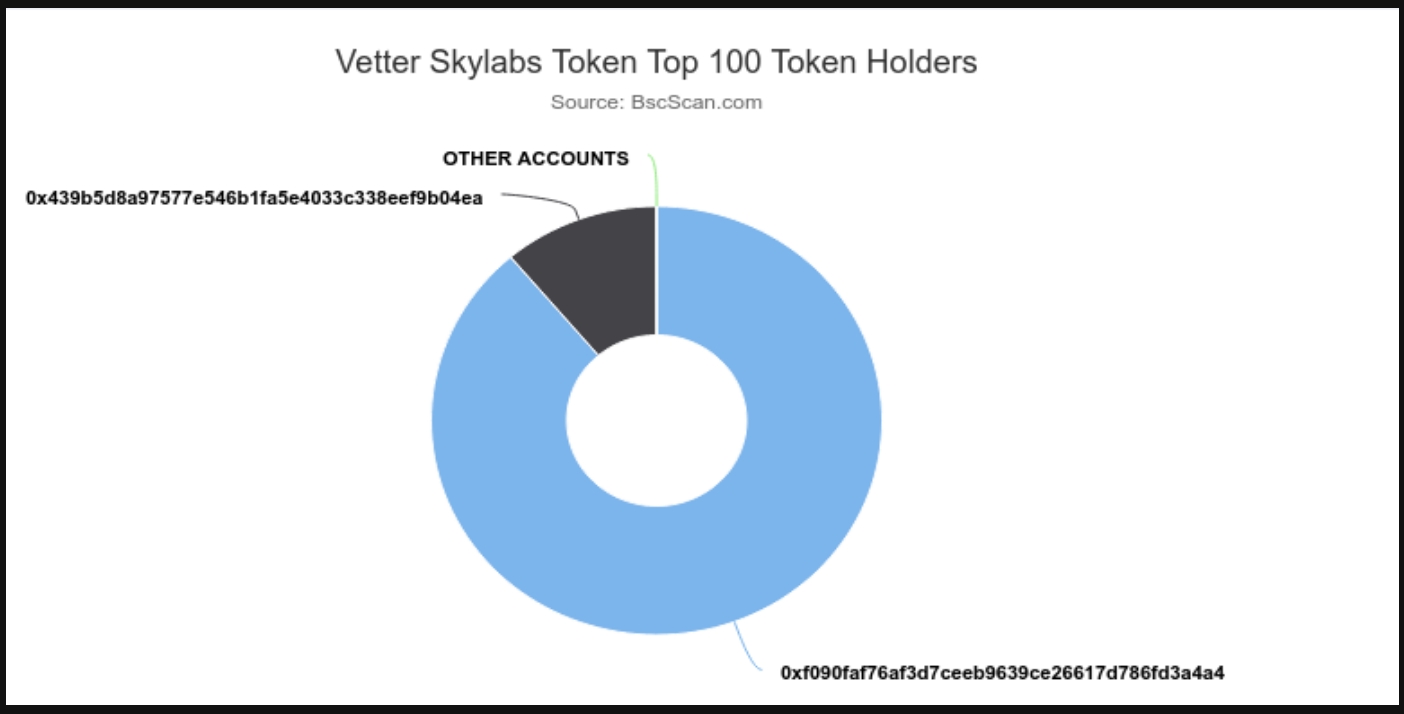
Quick Overview & Findings

Vetter's first dApp (CrowdX) was released in October 2021. As an acclaimed resource for high standards and results, the dApp is now used by thousands of investors. Vetter's ecosystem is expanding with Skylabs (VSL) Presale scheduled for August. With its in-house dev team and the trusted reputation of Vetter, launching projects that meet the highest standards, plus non-custodial staking with royalties from revenue not vaper...Skylabs is poised to lead several sectors of the blockchain.

Skylab Token Design:

Swaps taxed tokens into BNB, a portion of this BNB will be sent to staking contract and other part will be added to Liquidity pool. there is no way for owner to disable trades.

Token Holders Chart:



Skylab Findings

In process of reviewing Skylab Contract, we came up with this results:

Centralization Risks: 2 Risks (1 Low, 1 Medium)

Logical Issues: 6 Issues (2 Low, 4 medium)

Suggestions: 3 suggestions

Gas optimization: 3 gas optimizations

Findings – Centralization Risks

[M-00]-Owner of contract is able to set taxes up to 27%:

Owner of contract is able to use setAllTaxes function to increase taxes up to 27%.
This is considered a medium issue as 27% is a high tax and also because owner is able to exclude himself from paying this amount of tax

[L-01]-Owner of contract is able to exclude & include an arbitrary address from taxes:

Using excludeFromFee and includeInFee owner is able to exclude and include an arbitrary address from paying taxes.

Findings – Logical Issues

[L-00]-Potential sandwich attacks:

at lines 352 & 381 there is not output amount, someone can observe this transactions at memepool and frontrun them.

```
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount, 0, path, address(this), getTime())
```

and

```
uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(this),getTime());
```

[L-01]-Liquidity Tax is always half of Liquidity Cap:

`_liquidityCap` is 2%, while `_halfLiquidityBuyTax` or `_halfLiquiditySellTax` (which are considered liquidity tax on buy or sell) can not be more than 1%

[M-00]-Restricted view functions can cause problems in Dapps:

getStakingContract, isAllowedContract, getAllAllowedAddresses, getAllExchanges, isExchangeAddress, getExchangeID, getExchangeAddress are view functions that not everyone is able to call, remove modidfier since these are view functions and cant make any state changes, beside that, this issue can cause big problems if you want to integrate your token with a dapp later

[M-01]-Bad event emission at setupAllowedContract:

`emit AllowedContractChange(_contractAddress, true)`

if _allowOrNot is false then AllowedContractChange should be

`emit AllowedContractChange(_contractAddress, false)`

Findings – Logical Issues

[M-02]-potential mistake at emitting event:

at setupExchangeAddress event ExchangeAddressAdded is still emitted even if Exchange was already added before.

[M-03]-Potential Mistake at setupAllowedContract:

even if `_allowOrNot` is false, `allowedCount` is increased and also the input address gets added to `_allowedByID` , this might be intentional, but based on name of arrays and data structures, we think that this might be an issue.

Findings – Gas Optimization

[GO-00] - unnecessary gas usage at getTime():
instead use block.timestamp

[GO-01] - declare variables as immutable:
declare uniswapV2Router as immutable

[GO-02] - remove Address library:
remove this library as it is never used inside contract

Findings – Suggestions

[S-01]-Potential Error at changeRouterVersion due to ABI difference:

new router may not follow the ABI of Uniswap for creating pair, this is pretty much unlikely, but to make sure that there wont be any problem in future, try sending calldata of **creating pair operation** as an input parameter for new router address and then create pair with low level call to new router.

[S-02] - lack of event emition:

This 2 functions are not emitting an event:
-TransferA11BNBToAddress
-TransferBNBToAddress

[S-03] - code refactoring at _transfer can make it more readable:

there is 2 if statements that are checking if sender or receiver is address of pair or not, but they are doing pretty much the same thing, the only difference is tax difference on buy or sell, which can be set at top of other codes.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team.

This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts VITALIK to perform a security assessment.

This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors,business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as Investment advice of any sort.

This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic token sand blockchain technology.

About Vitalik Audit

we are a small yet strong auditing team, we love to read and test smart contracts, if you think you are able to audit a smart contract or you want your project to be audited reach us through one of this ways.

Telegram: t.me/VitalikAudit
Website: https://vitalik-platform.xyz
Owner: t.me/AnanCoder