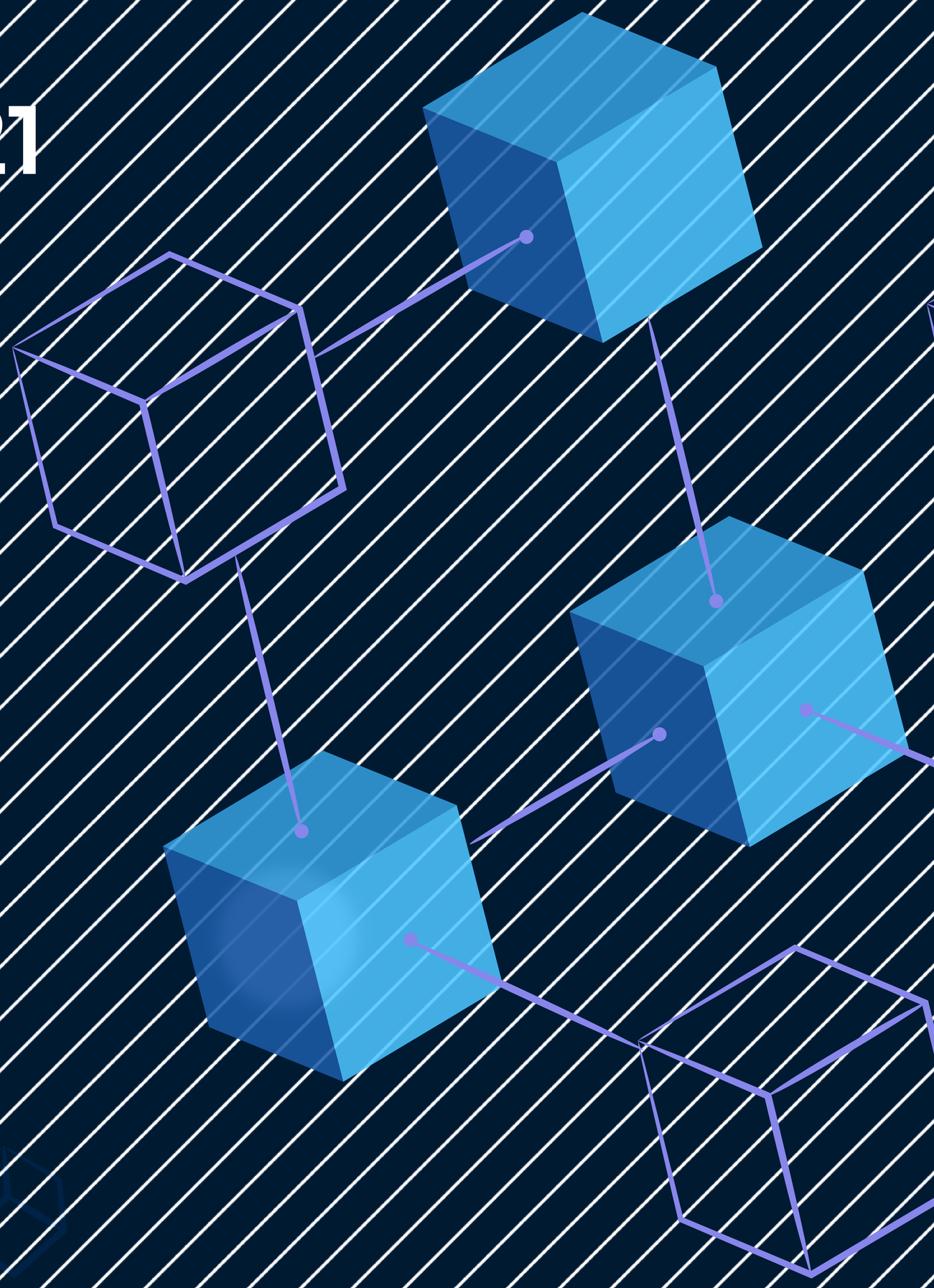
QuillAudits

Audit Report
December, 2021



For

WORTHPAD

Contents

Scope of the Audit

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

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

WorthToken.sol

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	1	0	0
Closed	0	3	3	6

WorthTokenSale.sol

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	1	0	0
Closed	1	5	4	6

WorthTokenTimeLock.sol

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	1	0	0
Closed	0	1	2	5

Introduction

During the period of **November 24, 2021 to December 13, 2021** - QuillAudits Team performed a security audit for Worthpad smart contracts.

The code for the audit was taken from following the official link: https://github.com/worthpad/worth

V	Date	Commit ID
1	November 24	f18eb432f476ab7b37bf7e1d45bdc33f10427215
2	December 10	6bb59142020a3cb6ac225137df1ded7caedbc3a9

Issues Found

A. Contract - WorthToken.sol

High severity issues

No issues were found.

Medium severity issues

1. **Centralization Risks**

Description

The role owner has the authority to
- update critical settings
- manage the list containing contracts excluding from fees
- Update the worthDVCFundWallet address
- Update the _worthDVCFundFee value

Remediation

We advise the client to handle the governance account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

- with reasonable latency for community awareness on privileged operations;
- Multisig with community-voted 3rd-party independent co-signers;
- DAO or Governance module increasing transparency and community involvement;

Status: Acknowledged

To create a decentralized solution for the management of Worthpad smart contracts, the auditee is utilizing the Gnosis Safe multisig. The Gnosis Safe requires 2 of 3 wallets to confirm every transaction in order to execute it, which prevents unauthorized access to company crypto. In future, this ownership will be transferred to the community through the WORTH DAO.

2. Funds in this contract will be locked

Description
Contract with a payable function, but without a withdrawal capacity. Since receive() external payable {} is utilized without a withdrawal function. Thus, every Ether sent to this contract will be lost.

Remediation
Remove the payable attribute or add a withdrawal function for this contract.

Status: Fixed in version 2

3. Missing Check for Reentrancy Attack

Description
Calling WorthToken.transfer() and WorthToken.transferFrom() might trigger function uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens and uniswapV2Router.addLiquidityETH , which is implemented by the third party at uniswapV2Router. If there are vulnerable external calls in uniswapV2Router , reentrancy attacks could be conducted because these two functions have state updates and event emits after external calls.

The scope of the audit would treat the third-party implementation at InutRouter02 as a black box and assume its functional correctness. However, third parties may be compromised in the real world that leads to assets lost or stolen.

Remediation
We recommend applying OpenZeppelin ReentrancyGuard library - nonReentrant modifier for the aforementioned functions to prevent reentrancy attacks.

Status: Fixed in version 2

4. Missing Range Check for Input Variable

Description

The role can set the following state variables arbitrary large or small causing potential risks in fees and anti whale :

- _maxTxAmount
- numTokensSellToAddToLiquidity
- _worthDVCFundFee
- _liquidityFee

Furthermore, we've found that the numTokensSellToAddToLiquidity can be set freely to any value which can be greater than the totalSupply. In this case, the contractTokenBalance >= numTokensSellToAddToLiquidity might never be reached, which means that swapAndLiquify might not be executed.

Remediation

We recommend setting ranges and check the above input variables.

Status: Fixed in version 2

Low severity issues

5. Bypass worthDVCFundFee and liquidityFee

Description

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision.

Therefore, the senders are able to bypass the worthDVCFundFee and liquidityAmount due to the calculation in the _tokenTransfer() function.

L548: worthDVCFundFee = amount.mul(_worthDVCFundFee).div(FEES_DIVISOR);
L522: liquidityAmount = amount.mul(_liquidityFee).div(FEES_DIVISOR);

The worthDVCFundFee = 0 if amount.mul(_worthDVCFundFee) < FEES_DIVISOR. And,
The liquidityAmount =0 if amount.mul(_liquidityFee) < FEES_DIVISOR.

In conclusion, if the amount < FEES_DIVISOR/_liquidityFee the worthDVCFundFee andliquidityAmount will equal to Zero, which means that the receivers do not pay for any extra fees.

Remediation
We recommend adding the check for amount to ensure that:
- amount.mul(_worthDVCFundFee) > FEES_DIVISOR
- amount.mul(_liquidityFee) < FEES_DIVISOR

Status: Fixed in version 2

6. Missing zero address validation

Description
We've detected missing zero address validation for newWallet in the setWorthDVCFundWallet function and _worthDVCFundWallet in the constructor.

Remediation
Consider implementing require statements where appropriate to validate all user-controlled input, including constructor, to avoid the potential for erroneous values to result in unexpected behaviors or wasted gas.

Status: Fixed in version 2

7. Improper Input Validation

Description
The setWorthDVCFundWallet does not check whether the newWallet address is similar to the current worthDVCFundWallet. Similarly, the includeInFee and excludeFromFee does not check if the account is already included or excluded from the fees.

Remediation
We recommend adding a check for the aforementioned functions, this helps to avoid the potential for erroneous values to result in unexpected behaviors or wasted gas.

Status: Fixed in version 2

Informational issues

8. Missing Events for Significant Transactions

Description
The critical settings are completely devoid of event definitions or emissions. This makes it very difficult for users or other interested parties to track important changes that take place in the system.

Thus, the missing event makes it difficult to track off-chain liquidity fee changes. An event should be emitted for significant transactions calling the following functions:

- excludeFromFee()
- includeInFee()
- setWorthDVCFundWallet()
- setLiquidityFeePercent()
- setWorthDVCFundFeePercent()
- setMinSell()
- setMaxTxAmount()
- setRouterAddress()

Remediation
We recommend emitting an event to log the update of the above functions.

Status: Fixed in version 2

9. State Variable Default Visibility

Description
The Visibility of the inSwapAndLiquify variable at line 246 is not defined. Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.
The default is internal for state variables, but it should be made explicit.

Remediation
We recommend adding the visibility for the state variable of inSwapAndLiquify. Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

Status: Fixed in version 2

10. **Public function that could be declared external**

Description
The following public functions that are never called by the contract should be declared external to save gas:

- excludeFromFee()
- includeInFee()
- setRouterAddress()
- setSwapAndLiquifyEnabled()

Remediation
Use the external attribute for functions never called from the contract.

Status: Fixed in version 2

11. **Not using delete to zero values**

Description
In the disableAllFees function and removeAllFee function, the following variables are manually replaced with Zero:

- _liquidityFee
- _worthDVCFundFee

Remediation
To simplify the code and clarify intent, consider using delete instead.

Status: Fixed in version 2

12. **Typos**

We recommend fixing the following typos:

L256: tokensIntoLiqudity should be tokensIntoLiquidity
L323: (uint 256) should be (uint256)
L402: recieve should be receive

Status: Fixed in version 2

13. Missing decimals() view function

WorthPad token decimals are not displayed in Metamask nor in the BSCScan contract page. We recommend adding the view function which aims to return this value.

Status: Fixed in version 2

B. Contract - WorthTokenSale.sol

High severity issues

1. claimDate can be changed after the sale

Description
changeClaimDate() function can be exploited by the owner to deny investors their tokens for a very long time. Due to the fact that this function also does not have any modifier/constraints and thus can be changed during as well as after the Tokensale.

Remediation
We recommend adding a condition for this function to ensure that the claimDate can’t be modified during or after the sale.

Status: Fixed in version 2. changeClaimDate() function was removed.

Medium severity issues

2. transfer() is being utilized

Description
Low-level transfer() function has been found to be used in the contract at line 280:
beneficiary.transfer(address(this).balance);
Due to the fact that .transfer() and .send() forward exactly 2,300 gas to the recipient. This hardcoded gas stipend aimed to prevent reentrancy vulnerabilities, but this only makes sense under the assumption that gas costs are constant. Recently EIP 1884 was included in the Istanbul hard fork. One of the changes included in EIP 1884 is an increase to the gas cost of the SLOAD operation, causing a contract's fallback function to cost more than 2300 gas.

```
// bad
contract Vulnerable {
    function withdraw(uint256 amount) external {
        // This forwards 2300 gas, which may not be enough if the recipient
        // is a contract and gas costs change.
        msg.sender.transfer(amount);
    }
}

// good
contract Fixed {
    function withdraw(uint256 amount) external {
        // This forwards all available gas. Be sure to check the return value!
        (bool success, ) = msg.sender.call.value(amount)("");
        require(success, "Transfer failed.");
    }
}
```

Remediation

The auditee needs to ensure that the beneficiary is not a contract .On the other hand, it's recommended to stop using .transfer() and .send() and instead use .call().

Status: Fixed in version 2

3. Centralization Risks

Description

The role owner has the authority to
- update critical settings
- execute withdrawCrypto() and withdrawTokens()

Moreover, during the sale the functions such as updateTokenPrice(), updateHardcap(), updateTokenContribution() and updateTokenAddress() can be manipulated and exploited by the owner. Especially, updateTokenAddress() function can be used to change the token address even during the token sale as well as after the token sale. This can be maliciously exploited by the owner and can lead to potential Rug Pull.

Remediation
We advise the client to handle the governance account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

- with reasonable latency for community awareness on privileged operations;
- Multisig with community-voted 3rd-party independent co-signers;
- DAO or Governance module increasing transparency and community involvement;

We recommend adding a modifier() for updateTokenPrice(), updateHardcap(), updateTokenContribution() and updateTokenAddress(), which helps to lock these functions during the sale as well as after the sale.

Status: Acknowledged
To create a decentralized solution for the management of Worthpad smart contracts, the auditee is utilizing the Gnosis Safe multisig. The Gnosis Safe requires 2 of 3 wallets to confirm every transaction in order to execute it, which prevents unauthorized access to company crypto. In future, this ownership will be transferred to the community through the WORTH DAO.

4. Business Logic

Description
As described in the code comments, the withdrawTokens() and withdrawCrypto() functions are executable after the sale.
However, the owner can call these functions during the sale or before the sale without any restrictions.

Remediation
We recommend adding the _saleEnded() modifier for these 02 functions.

Status: Fixed in version 2

5. Divisions performed before multiplication

Description

Due to the fact that Solidity truncates when dividing, performing a division before multiplication can result in truncated amounts being amplified by future calculations. There are a few instances in the codebase where division is performed mid-calculation. For examples:
L162: tokenAmount = ((amount.mul(10 ** uint256(tokenDecimal)).div(tokenPriceUsd)).mul(10 ** uint256(tokenDecimal))).div(10 ** uint256(tokenDecimal))
L185: tokenAmount = ((amount.mul(10 ** uint256(tokenDecimal)).div(tokenPriceUsd)).mul(10 ** uint256(tokenDecimal))).div(10 ** uint256(tokenDecimal))

Remediation

Being mindful of overflows, consider changing the order of operations where possible such that truncating steps are performed last to minimize any unnecessary loss of precision. We recommend changing the order of the divisions in the contract.

Status: Fixed in version 2

6. Possible reentrancy vulnerabilities

Description

There is a state change after the external call in the claimToken() token function, where the tokenExchanged[userAdd] = 0 after the Token(tokenAddr).transfer(..).

We recommend changing the order of the tokenExchanged[userAdd] = 0 before the external call of Token(tokenAddr).transfer(..) to avoid the reentrancy that might occur.

Status: Fixed in version 2

7. Unchecked transfer

Description
The return value of the following external transfer call is not checked:
L135: Token(tokenAddr).transfer(userAdd, amountToClaim);

Remediation
We recommend wrapping this operation within the require() statement.

Status: Fixed in version 2

Low severity issues

8. No error messages in require() functions

Description
There are few places in the entire codebase where the require() statement does not contain any error message. As error messages are intended to notify users about failing conditions, they should provide enough information so that appropriate corrections can be made to interact with the system. Below is a non-exhaustive list of identified instances:
- L59: require(contractUp);
- L64: require(_to != address(0));
- L69: require(saleEnded);
- L74: require(!saleEnded);

Remediation
Lack of error messages greatly damage the overall user experience, thus lowering the system's quality. Consider not only fixing the specific instances mentioned above, but also reviewing the entire codebase to make sure every error message is informative and user-friendly.

Status: Fixed in version 2

9. **startAt and endAt variables are not used**

   We discovered that the startAt and endAt variables are not used in any operation. We recommend removing these two variables or considering their use in the contract's business logic.

   Status: Fixed in version 2

10. **Lack of zero balance check**

    To avoid the potential for erroneous values to result in unexpected behaviors or wasted gas. We recommend adding a check for address(this).balance in withdrawCrypto() function, the address(this).balance should be checked if it's not empty.

    Status: Fixed in version 2

11. **Missing zero address validation**

    Description
    We've detected missing zero address validation for the following functions:
    - constructor()
    - updateUSDTBUSDaddress()
    - updateTokenAddress()
    - withdrawCrypto()

    Remediation
    Consider implementing require statements where appropriate to validate all user-controlled input, including constructor, to avoid the potential for erroneous values to result in unexpected behaviors or wasted gas.

    Status: Fixed in version 2

Informational issues

12. Missing Events for Significant Transactions

Description
The critical settings are completely devoid of event definitions or emissions. This makes it very difficult for users or other interested parties to track important changes that take place in the system.

Thus, the missing event makes it difficult to track off-chain liquidity fee changes. An event should be emitted for significant transactions calling the following functions:

- changeClaimDate()
- updateTokenAddress()
- updateTokenDecimal()
- updateUSDTBUSDaddress()
- updateTokenContribution()
- updateHardCap()
- updateTokenPrice()

Remediation
We recommend emitting an event to log the update of the above functions.

Status: Fixed in version 2

13. Public function that could be declared external

Description
The following public functions that are never called by the contract should be declared external to save gas:

- whitelistAddress()
- depositTokens()
- claimToken()
- ExchangeUSDTforToken()
- ExchangeBUSDforToken()
- toggleWhitelistStatus()

- updateTokenPrice()
- updateHardCap()
- updateTokenContribution()
- updateUSDTBUSDaddress()
- updateTokenDecimal()
- updateTokenAddress()
- withdrawTokens()
- withdrawCrypto()
- changeClaimDate()

Remediation
Use the external attribute for functions never called from the contract.

14. **Conformance to Solidity naming conventions**

As following Solidity naming convention. Functions other than constructors should use mixedCase. Examples: getBalance, transfer, verifyOwner, addMember, changeOwner.

Therefore, ExchangeUSDTforToken() and ExchangeBUSDforToken() functions are not in mixedCase. We recommend changing:
- ExchangeUSDTforToken to exchangeUSDTForToken
- ExchangeBUSDforToken to exchangeBUSDForToken

Events should be named using the CapWords style. Examples: Deposit, Transfer, Approval, BeforeTransfer, AfterTransfer.

We recommend changing:
- amountTransfered to AmountTransferred

Status: Fixed in version 2

15. Not explicitly defining maximum allowance

To favor explicitness, consider changing all instances of uint into uint256 in the entire codebase. For example:

```
L45:    event TokenTransfer(address beneficiary, uint amount);
L47:    event TokenDeposited(address indexed beneficiary, uint amount);
L48:    event UsdDeposited(address indexed beneficiary, uint amount);
L110:    for (uint i = 0; i < _recipients.length; i++) {
```

Status: Fixed in version 2

16. Irrelevant return value

toggleWhitelistStatus() function always returns true. This can be confused by the owner to assume that the whitelist variable has been toggled to true whereas in fact it could've been toggled to false. It is advised to return the whitelist status instead.

Status: Fixed in version 2

17. Ambiguous code comments

The comment above ExchangeUSDTforToken() function is inconsistent with the parameter of the function. In the comment it is stated that the parameter _amount is the "Total token to buy (in WEI)" but in fact _amount is the amount of USDT that an investor pays to buy the token. It is advised to make the comment more specific. The same is advised for the function ExchangeBUSDforToken().

Status: Fixed in version 2

C. Contract - WorthTokenTimeLock.sol

High severity issues

No issues were found.

Medium severity issues

1. **block.timestamp may not be reliable**

   The contract uses the block.timestamp as part of the calculations and time checks for the users to withdraw their token.

   Nevertheless, timestamps can be slightly altered by miners to favor them in contracts that have logics that depend strongly on them. Consider taking into account this issue and warning the users that such a scenario could happen.

   Status: Acknowledged by the auditee

2. **DoS With Block Gas Limit**

   There are two scenarios that can happen to make the depositsByWithdrawalAddress mapping growing:
   1. The lockTokens() and createMultipleLocks() are public functions without any modifiers. There a This can allow an attacker to perform Denial of Service attacks as he can lock very small amounts of tokens, thus rapidly incrementing the depositId variable and the length of ids array inside depositsByWithdrawalAddress mapping to a point such that the withdrawee is unable to withdraw tokens using the withdraw() function as it runs out of gas. This can also result in DOS attack on transferLocks() function as it iterates over the same array in the mapping

   2. In the future, the number of the tokens will be increasing naturally by the normal users. The loop in line 133, for (j=0; j<arrLength; j++) does not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully: Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can stall the complete contract at a certain point. Additionally, using unbounded loops can incur a lot of avoidable gas costs.

Remediation
Carefully test how many items at maximum you can pass to such functions to make it successful.

Status: Fixed in version 2

Low severity issues

3. No error messages in require() functions

Description
There are few places in the entire codebase where the require() statement does not contain any error message. As error messages are intended to notify users about failing conditions, they should provide enough information so that appropriate corrections can be made to interact with the system. Below is a non-exhaustive list of identified instances:
L49: require(_amount > 0);
L50: require(_unlockTime < 10000000000);
L77: require(_amounts.length > 0);
L78: require(_amounts.length == _unlockTimes.length);
L82: require(_amounts[i] > 0);
L83: require(_unlockTimes[i] < 10000000000);
L108: require(_unlockTime < 10000000000);
L109: require(!lockedToken[_id].withdrawn);
L110: require(msg.sender == lockedToken[_id].withdrawalAddress);
L121: require(!lockedToken[_id].withdrawn);
L122: require(msg.sender == lockedToken[_id].withdrawalAddress);
L150: require(block.timestamp >= lockedToken[_id].unlockTime);
L151: require(msg.sender == lockedToken[_id].withdrawalAddress);
L152: require(!lockedToken[_id].withdrawn)

Remediation
Lack of error messages greatly damage the overall user experience, thus lowering the system's quality. Consider not only fixing the specific instances mentioned above, but also reviewing the entire codebase to make sure every error message is informative and user-friendly.

Status: Fixed in version 2

4. Insufficient Input Validation

Description

The extendLockDuration() can be called by the withdrawalAddress, but it doesn't check to ensure that he/she is extending or reducing the lock duration of the Locked tokens. Therefore, it is possible that the withdrawalAddress can exploit this function to reduce his lock duration and claim the tokens instantly. For example, the withdrawalAddress might be able to reduce the unlockTime for his/her tokens from 1738430755 to 1638436714 and claim his/her locked tokens instantly.

Remediation

We recommend adding a check to ensure that the new _unlockTime is greater than then old unlockTime.

Status: Fixed in version 2

Informational issues

5. Typos

The following typo has been found:

- L188: Recievers should be Receivers

Status: Fixed in version 2

6. Conformance to Solidity naming conventions

As following Solidity naming convention. Local and State Variable Names should be in mixedCase. Examples: totalSupply, remainingSupply, balancesOf, creatorAddress, isPreSale, tokenExchangeRate.

L40: event LogWithdrawal(address SentToAddress, uint256 AmountTransferred);

Thus, we recommend changing:

- SentToAddress to sentToAddress
- AmountTransferred to amountTransferred

Status: Fixed in version 2

7. Public function that could be declared external

Description
The following public functions that are never called by the contract should be declared external to save gas:

- lockTokens()
- createMultipleLocks()
- extendLockDuration()
- transferLocks()
- withdrawTokens()

Remediation
Use the external attribute for functions never called from the contract.

Status: Fixed in version 2

8. Redundant function in interface contract

We recommend removing the approveAndCall() function in the Token interface contract as it's not being utilized in the WorthTokenTimeLock contract.

Status: Fixed in version 2

9. Incorrect code comment

As commented in line 47, "/* Public View Function */", meanwhile the lockTokens() is not a view function. Similar to line line 57, the createMultipleLocks() function is not a view function.

Therefore, we recommend fixing these comments for the aforementioned functions to increase the readability of the code.

Status: Fixed in version 2

Functional test

WorthToken.sol

Address: 0xE79E0cA86f07cf19EacB318aAa389992f3d4C130
Evn: https://testnet.bscscan.com/

Function name	Input	Output	TX	Status
setMaxTxAmount	100000000000000000000000000000000000000	true	0x0959e1a78d44e02a49fc5e9acf4d9546cb418eb3dfebeca09be3bf21afecb708	Passed
_maxTxAmount	N/A	100000000000000000000000000000000000000	call0x1dd64394E29c5988f04A8E074D0DBACd4D6147290xE79E0cA86f07cf19EacB318aAa389992f3d4C1300x7d1db4a5	Passed
setMinSell	100000000000000000000000	true	0x2abd393698ca37489a4d0173e6233152e16f37c0db6e95f333f4fa25c7d61b39	Passed
setRouterAddress	0xd389468Ea139a8Ac8856c948C8AE0c3068AF84B2	true	0x98a9cd1ef8e0f44a802afbc06e8c7b04a0cecc33f52ce335e06cf7b1396268d7	Passed
uniswapV2Router	N/A	0xd389468Ea139a8Ac8856c948C8AE0c3068AF84B2	call0x1dd64394E29c5988f04A8E074D0DBACd4D6147290xE79E0cA86f07cf19EacB318aAa389992f3d4C1300x1694505e	Passed
setWorthDVCFundWallet	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC	True	0xd0ce164a51ef0a2873e801a267e793f9cff8b387d45250bc01aa359dd5ff708b	Passed
setWorthDVCFundFeePercent	30	true	0x5fd239a7498270e52d37e51a79d87e13cb8ebc951d73473493eafdcb75e28a69	Passed

setSwapAndLiquifyEnabled	true	true	0x3cf700a87d47b2b0533055814cdf1b5afa7d4f808a1f62d840c4f312e923cfc4	Passed
transferOwnership	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC	true	0x349d9a6011de34825a27bdff3fc621bd700e113e04aa6bfcf0b22c7a90eebbec	Passed
setLiquidityFeePercent	30	true	0x88631b6d9533389eb24624aab147148d519e42bbefeb6b3739d882116ad81bc0	Passed
excludeFromFee	0x153b057d5d7262dC92099B59c975255ecE66784F	true	0x314e4fa2a926e89edbeccd1128e0ddce2a561f12766486f3255e40bb399b22e9	Passed
includeInFee	0x153b057d5d7262dC92099B59c975255ecE66784F	true	0x3b28a3f91db688906817657755 8c049822c361fdded827ae9917fa420ac1fd90	Passed
disableAllFees	N/a	true	0xf0621c11cae2b2cda1d3824a0f30542ee2a70e79c3404a9e4efeaaa56ea6a2b2	Passed
enableAllFees	N/a	true	0xc8fac310113d7af9cb15d0fb7b865ff35c0ff509fbd148cad5a3f593e363049c	Passed
transfer(_isExcludedFromFee[sender])	"0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC","10000000"	true	0x75e3369fe82d25207a0e56273c2e547bce4809e4238944608565753 7da3d6956	Passed
balanceOf	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC	10000000	call0x1dd64394E29c5988f04A8E074D0DBACd4D6147290xE79E0cA86f07cf19EacB318aAa389992f3d4C1300x70a0823100000000000000000000000060b6d91cb698f41e1ed928f9631cec6b8ff8f6cc	Passed
transfer	"0x8cF95C8750FB8E83Cc45F44232da9Dd022037e05","10000000"	true	0xcd00ba4aef5af7e8f7d85ae4667cfc24ccb5085092de12155b81d5b3f5de5f63	Passed
balanceOf	0x8cF95C8750FB8E83Cc45F44232da9Dd022037e05	9500000	call0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC0xE79E0cA86f07cf19EacB318aAa389992f3d4C1300x70a0823100000000000000000000000008cf95c8750fb8e83cc45f44232da9dd022037e05	Passed

balanceOf (contract with 25%)	0xE79E0cA86f07cf19EacB318aAa389992f3d4C130	250000	call0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC0xE79E0cA86f07cf19EacB318aAa389992f3d4C1300x70a08231000000000000000000000e79e0ca86f07cf19eacb318aaa389992f3d4c130	Passed
balanceOf (worthDVCFundFee with 25%)	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC	250000	call0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC0xE79E0cA86f07cf19EacB318aAa389992f3d4C1300x70a0823100000000000000000000060b6d91cb698f41e1ed928f9631cec6b8ff8f6cc	Passed
approve	"0x153b057d5d7262dC92099B59c975255ecE66784F","9500000"	true	0xb6f9664a84135025607faaeec59dc5862e6b3972c8b9041a14052a2221dfd52a	Passed
allowance	"0x8cF95C8750FB8E83Cc45F44232da9Dd022037e05","0x153b057d5d7262dC92099B59c975255ecE66784F"	9500000	call0x8cF95C8750FB8E83Cc45F44232da9Dd022037e050xE79E0cA86f07cf19EacB318aAa389992f3d4C1300xdd62ed3e000000000000000000000008cf95c8750fb8e83cc45f44232da9dd022037e0500000000000000000000000000153b057d5d7262dc92099b59c975255ece66784f	Passed
decreaseAllowance	"0x153b057d5d7262dC92099B59c975255ecE66784F","9300000"	true	0x9dd31763eb2e188fe0626314f659b72fb31e9dd0b48089cd44ab8da7f4733a03	Passed
allowance	"0x8cF95C8750FB8E83Cc45F44232da9Dd022037e05","0x153b057d5d7262dC92099B59c975255ecE66784F"	200000	N/A	Passed
transferFrom	"0x8cF95C8750FB8E83Cc45F44232da9Dd022037e05","0x153b057d5d7262dC92099B59c975255ecE66784F",200000	true	0xddb212246902a6eda5eadf24b96462ae1ff49c6cf403b86380fb10a44de3332a	Passed
balanceOf	0x8cF95C8750FB8E83Cc45F44	9300000	call0x153b057d5d7262dC92099B59c975255ecE66784F0xE79E0cA86f07cf19EacB318aAa389992f3d4C1300x70a08231000000000000000000000	Passed

	232da9Dd022037e05		0000008cf95c8750fb8e83cc45f44232da9dd022037e05	
balanceOf	0x153b057d5d7262dC92099B59c975255ecE66784F	190000	call0x153b057d5d7262dC92099B59c975255ecE66784F0xE79E0cA86f07cf19EacB318aAa389992f3d4C1300x70a08231000000000000000000000153b057d5d7262dc92099b59c975255ece66784f	Passed
balanceOf (of the contract with 25% fees)	0xE79E0cA86f07cf19EacB318aAa389992f3d4C130	255000	call0x153b057d5d7262dC92099B59c975255ecE66784F0xE79E0cA86f07cf19EacB318aAa389992f3d4C1300x70a08231000000000000000000000e79e0ca86f07cf19eacb318aaa389992f3d4c130	Passed
balanceOf (of the contract with 25% fees)	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC	255000	call0x153b057d5d7262dC92099B59c975255ecE66784F0xE79E0cA86f07cf19EacB318aAa389992f3d4C1300x70a08231000000000000000000000060b6d91cb698f41e1ed928f9631cec6b8ff8f6cc	Passed
approve	"0x153b057d5d7262dC92099B59c975255ecE66784F","9000000"	true	0xa9df2f275f6302a5088b5e5031d86731b1facbfaf8139667d5532695f996b966	Passed
allowance	"0x8cF95C8750FB8E83Cc45F44232da9Dd022037e05","0x153b057d5d7262dC92099B59c975255ecE66784F"	9000000	N/A	Passed
increaseAllowance	"0x153b057d5d7262dC92099B59c975255ecE66784F","300000"	true	0xb45f4ff9af91ecbec298727c6fa84b6e078e46b08178dfc42b6efe3034402cf1	Passed
allowance	"0x8cF95C8750FB8E83Cc45F44232da9Dd022037e05","0x153b057d5d7262dC92099B59c975255ecE66784F"	9300000	N/A	Passed
transfer	"0xE79E0cA86f07cf19EacB318aAa389992f3d4C	true	0x970eb652bbe9326c22eeaa7c233540a3bede3b15248984fbac72880487606376	Passed

	130","100000000000000000"			
balanceOf	0xE79E0cA86f07cf19EacB318aAa389992f3d4C130	10000000000000487500	call0x1fc96d7B1FbEfA23EA45520472B61dA0686Bc80B0xE79E0cA86f07cf19EacB318aAa389992f3d4C1300x70a08231000000000000000000000000e79e0ca86f07cf19eacb318aaa389992f3d4c130	Passed

WorthTokenSale.sol

Address: 0xAeec761bC805ac0c282b272638394d8F8bfF1C3f
Evn: https://testnet.bscscan.com/

Function name	Input	Output	TX	Status
changeClaimDate	1638074820	true	0xa982570e14cd9c28e77f93227265ce2cfd2694e3b841752c23ae7f98c02a1b21	Passed
updateTokenAddress	0xE79E0cA86f07cf19EacB318aAa389992f3d4C130	true	0x616ff75c4beb3d2a1f78fe90189100187ef6702defb537e4e85601c507acd802	Passed
updateTokenDecimal	18	true	0xf9f80546bc8339adc0d96e09372d56b9379a6d1d5f610366d9df9806d508700e	Passed
updateUSDTBUSDaddress	0x337610d27c682e347c9cd60bd4b3b107c9d34ddd, 0xed24fc36d5ee211ea25a80239fb8c4cfd80f12ee	true	0x02488ef52438aa90a20a5aa650530c899607d1013bf9239c7a91abc1a7f9d5d8	Passed
updateTokenContribution	"1","10"	true	0x98390627bf16c2ff44589e388a85c052246b5ca24548124a6fb42e8d6985a7c3	Passed
updateHardCap	200000000000000000000	true	0x633bbbd1a08a0bced115b7800d560049f9fc876c05451438310a3b9d81fc6904	Passed
updateTokenPrice	2	true	0x5c01dd829a6203eb468939804427ec9948cdd55f7a16637332885eccb64dd8ff	Passed
toggleWhitelistStatus	N/A	true	0x761693c2c75eafe198818fc09cbe45d0dd4a0adc251d9f48df6818f20f2fe86e	Passed
powerUpContract	N/A	true	0xbc91a9cc560cfd99323ec91105fcc4cf55612b0bd85940c963a6851fac82c154	Passed
whitelistAddress	["0x1fc96d7B1FbEfA23EA45520472B61dA0686Bc80B"]	true	0x7016e4f020008c39852a8cb7c2913c5a77c546253625d474b588ccfd375a7543	Passed
depositTokens	163	true	0x3b4ebf263ff6a9a3e4f47a3cb0c117e480105ec09717f68940c7c3ac8f7f938c	Passed
ExchangeBUSDforToken (From 0x1fc96d7B1FbEfA23EA4552047	1	true	0x298eda9be9bdba6842cc85cee3432cfab37cc336c5e7ac1994224669201a30fe	Passed

2B61dA0686Bc80B)				
ExchangeUSDTforToken (From 0x1fc96d7B1FbEfA23EA45520472B61dA0686Bc80B)	1	true	0xa06636095b2b5ebcd1cfceb1a97ad69bf8079e2d0cf28353ff9dbcd36b7ee987	Passed
emergencyStop	N/A	true	0xb75feb0f5ce0c3c6184d6980528ca6941272cf28d7661deb00f0bc11fc6728b5	Passed
claimTokens	N/A	true	0x8db06a0d006d9fc96466076f97bb8a4a1e28435c856891a5df65e50e1a09d696	Passed
balanceOf (beneficiary)	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC	25000000000708382	N/A	Passed
balanceOf (contract)	0xAeec761bC805ac0c282b272638394d8F8bfF1C3f	900000000000000155	N/A	Passed
withdrawTokens	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC, 0xE79E0cA86f07cf19EacB318aAa389992f3d4C130	true	0x0e9602f2dc2290727f8ba9891bad569172b631929a4d9c9677838421f1cb7325	Passed
balanceOf (contract)	0xAeec761bC805ac0c282b272638394d8F8bfF1C3f	0	N/A	Passed
balanceOf (beneficiary)	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC	880000000000708534	N/A	Passed
withdrawCrypto	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC	true	0xfadf39195543a2191c1fc585b90fb50985ba790c3f9fd017b7ad686166ffbf53	Passed
transferOwnership	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC	true	0x79c7630e6fc0dcb639beaa8d4052b90523b12fb1c8aaaab4e9d5e994e79db010	Passed

Contract: 0xB3Aec9e6BB02Da15498F3A73FEd784C0C48Dbf85

updateUSDTBUSDAddress (0x236bdA643f1476924Cd0E79023e5c2E7b03fC1D4 with decimals of 06)	0x236bdA643f1476924Cd0E79023e5c2E7b03fC1D4, 0xed24fc36d5ee211ea25a80239fb8c4cfd80f12ee	true	0x0efa19b63b2911e890dd75ddba744850b33344b18edf5575415c72fef30dde95	Passed

updateTokenContribution	1000000000000000000, 1000000000000000000	true	0x97414f0a8e3837ae858ea7c579122cda6324965ecea5ffb3f9cc47ddb6b410e2	Passed
exchangeUSDTforToken	1000000000000000000	true	0xec318855d274df645ff981c6612e5a1581b9cdd84461b4e35e94d226cb506321	FAILED

WorthTokenTimeLock.sol

Address: 0x7aB27720e14a583bB045Cf728063B0E6d4E78141
Evn: https://testnet.bscscan.com/

Function name	Input	Output	TX	Status
lockTokens	"0xE79E0cA86f07cf19EacB318aAa389992f3d4C130","0x1dd64394E29c5988f04A8E074D0DBACd4D614729","10000000","1638159000"	true	0xe07c3aa751bc21e1d4f450863079f094d320c9592521c62b943fa05ef301d13f	Passed
extendLockDurations	1, 1638159120	true	0xfc8cac7f1137e9b1c45028c848e8958d08a6102e16230020bf37e8b26d072fa8	Passed
withdrawTokens	2	true	0xa486e9034ca361638198b83f88b65aa3ac0cbd0579c00290a025c7ee23a8b564	Passed
withdrawTokens	3	true	0xfa9863429f5d72b049e4507c73ff903a70ca3eaed186fd6d7b6df9492bbce30f	Passed
createMultipleLocks	"0xE79E0cA86f07cf19EacB318aAa389992f3d4C130","0x153b057d5d7262dC92099B59c975255ecE66784F",["15000","15000"],["1638169500","1638169800"]	true	0xe40ab23ad3659f0081f1b9e9f26fee31485bc61b068a597e13c532a148090c24	Passed
getDepositDetails	5	0: address: _tokenAddress 0xE79E0cA86f07cf19EacB318aAa389992f3d4C130 1: address: _withdrawalAddress 0x153b057d5d7262dC92099B59	N/A	Passed

		c975255ecE66784F 2: uint256: _tokenAmount 15000 3: uint256: _unlockTime 1638169800 4: bool: _withdrawn false		
transferLocks	"5","0x1dd64394E29c5988f04A8E074D0DBACd4D614729"	true	0xbeef82eb4776058442debe3e4c5eb9b2228a29c1f5e80bf470551d3c4d803e4d	Passed
getDepositDetails	5	0: address: _tokenAddress 0xE79E0cA86f07cf19EacB318aAa389992f3d4C130 1: address: _withdrawalAddress 0x1dd64394E29c5988f04A8E074D0DBACd4D614729 2: uint256: _tokenAmount 15000 3: uint256: _unlockTime 1638169800 4: bool: _withdrawn false	N/A	Passed
transferOwnership	0x153b057d5d7262dC92099B59c975255ecE66784F	true	0xb2c186c9ec2513c94e36f652c3c1582478521403aa0c372b85e6b3760e0efc4e	Passed

owner	N/A	0x153b057d5d7262dC92099B59c975255ecE66784F	N/A	Passed

Automated Tests

WorthToken.sol

Slither:

```
WorthToken.addLiquidity(uint256,uint256) (WorthToken.sol#520-533) sends eth to arbitrary user
	Dangerous calls:
	- uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in WorthToken._transfer(address,address,uint256) (WorthToken.sol#439-472):
	External calls:
	- swapAndLiquify(contractTokenBalance) (WorthToken.sol#467)
		- uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
		- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (WorthToken.sol#510-516)
	External calls sending eth:
	- swapAndLiquify(contractTokenBalance) (WorthToken.sol#467)
		- uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
	State variables written after the call(s):
	- _tokenTransfer(from,to,amount) (WorthToken.sol#471)
		- _balances[sender] = _balances[sender].sub(tAmount) (WorthToken.sol#570)
		- _balances[recipient] = _balances[recipient].add(tAmount) (WorthToken.sol#571)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
WorthToken.addLiquidity(uint256,uint256) (WorthToken.sol#520-533) ignores return value by uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.
timestamp) (WorthToken.sol#525-532)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
name() should be declared external:
	- WorthToken.name() (WorthToken.sol#288-290)
symbol() should be declared external:
	- WorthToken.symbol() (WorthToken.sol#295-297)
decimal() should be declared external:
	- WorthToken.decimal() (WorthToken.sol#302-304)
totalSupply() should be declared external:
	- WorthToken.totalSupply() (WorthToken.sol#309-311)
getMinAutoLiquidityAmount() should be declared external:
	- WorthToken.getMinAutoLiquidityAmount() (WorthToken.sol#323-325)
transfer(address,uint256) should be declared external:
	- WorthToken.transfer(address,uint256) (WorthToken.sol#334-337)
allowance(address,address) should be declared external:
	- WorthToken.allowance(address,address) (WorthToken.sol#346-348)
approve(address,uint256) should be declared external:
	- WorthToken.approve(address,uint256) (WorthToken.sol#364-367)
transferFrom(address,address,uint256) should be declared external:
	- WorthToken.transferFrom(address,address,uint256) (WorthToken.sol#378-382)
increaseAllowance(address,uint256) should be declared external:
	- WorthToken.increaseAllowance(address,uint256) (WorthToken.sol#388-391)
decreaseAllowance(address,uint256) should be declared external:
	- WorthToken.decreaseAllowance(address,uint256) (WorthToken.sol#397-400)
isExcludedFromFee(address) should be declared external:
	- WorthToken.isExcludedFromFee(address) (WorthToken.sol#425-427)
excludeFromFee(address) should be declared external:
	- WorthToken.excludeFromFee(address) (WorthToken.sol#579-581)
includeInFee(address) should be declared external:
	- WorthToken.includeInFee(address) (WorthToken.sol#586-588)
setRouterAddress(address) should be declared external:
	- WorthToken.setRouterAddress(address) (WorthToken.sol#656-660)
setSwapAndLiquifyEnabled(bool) should be declared external:
	- WorthToken.setSwapAndLiquifyEnabled(bool) (WorthToken.sol#665-668)
renounceOwnership() should be declared external:
	- Ownable.renounceOwnership() (openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
	- Ownable.transferOwnership(address) (openzeppelin/contracts/access/Ownable.sol#62-65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (openzeppelin/contracts/utils/Address.sol#55-60):
        - (success) = recipient.call{value: amount}() (openzeppelin/contracts/utils/Address.sol#58)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (openzeppelin/contracts/utils/Address.sol#123-134):
        - (success,returndata) = target.call{value: value}(data) (openzeppelin/contracts/utils/Address.sol#132)
Low level call in Address.functionStaticCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#152-161):
        - (success,returndata) = target.staticcall(data) (openzeppelin/contracts/utils/Address.sol#159)
Low level call in Address.functionDelegateCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#179-188):
        - (success,returndata) = target.delegatecall(data) (openzeppelin/contracts/utils/Address.sol#186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (WorthToken.sol#46) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (WorthToken.sol#47) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (WorthToken.sol#63) is not in mixedCase
Function IUniswapV2Router01.WETH() (WorthToken.sol#83) is not in mixedCase
Parameter WorthToken.setSwapAndLiquifyEnabled(bool)._enabled (WorthToken.sol#665) is not in mixedCase
Variable WorthToken._liquidityFee (WorthToken.sol#234) is not in mixedCase
Variable WorthToken._worthDVCFundFee (WorthToken.sol#237) is not in mixedCase
Variable WorthToken._maxTxAmount (WorthToken.sol#241) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (WorthToken.sol#88) is too similar to IUniswapV2Router01.addLiq
uidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (WorthToken.sol#89)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

```
INFO:Detectors:
Address.functionCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#80-82) is never used and should be removed
Address.functionCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#90-96) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (openzeppelin/contracts/utils/Address.sol#109-115) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (openzeppelin/contracts/utils/Address.sol#123-134) is never used and should be removed
Address.functionDelegateCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#169-171) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#179-188) is never used and should be removed
Address.functionStaticCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#142-144) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (openzeppelin/contracts/utils/Address.sol#152-161) is never used and should be removed
Address.isContract(address) (openzeppelin/contracts/utils/Address.sol#27-37) is never used and should be removed
Address.sendValue(address,uint256) (openzeppelin/contracts/utils/Address.sol#55-60) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (openzeppelin/contracts/utils/Address.sol#196-216) is never used and should be removed
Context._msgData() (openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
SafeMath.div(uint256,uint256,string) (openzeppelin/contracts/utils/math/SafeMath.sol#191-200) is never used and should be removed
SafeMath.mod(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#151-153) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (openzeppelin/contracts/utils/math/SafeMath.sol#217-226) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#22-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#64-69) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#76-81) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#47-57) is never used and should be removed
SafeMath.trySub(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#35-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
WorthToken._previousLiquidityFee (WorthToken.sol#235) is set pre-construction with a non-constant function or state variable:
        - _liquidityFee
WorthToken._previousWorthDVCFundFee (WorthToken.sol#239) is set pre-construction with a non-constant function or state variable:
        - _worthDVCFundFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Pragma version0.8.9 (WorthToken.sol#11) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/access/Ownable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/security/ReentrancyGuard.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/token/ERC20/IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/utils/Address.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/utils/math/SafeMath.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Reentrancy in WorthToken.transferFrom(address,address,uint256) (WorthToken.sol#378-382):
        External calls:
        - _transfer(sender,recipient,amount) (WorthToken.sol#379)
                - uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (WorthToken.sol#510-516)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (WorthToken.sol#379)
                - uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (WorthToken.sol#435)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (WorthToken.sol#380)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (openzeppelin/contracts/utils/Address.sol#27-37) uses assembly
        - INLINE ASM (openzeppelin/contracts/utils/Address.sol#33-35)
Address.verifyCallResult(bool,bytes,string) (openzeppelin/contracts/utils/Address.sol#196-216) uses assembly
        - INLINE ASM (openzeppelin/contracts/utils/Address.sol#208-211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.8.9', '^0.8.0']
        - 0.8.9 (WorthToken.sol#11)
        - ^0.8.0 (openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (openzeppelin/contracts/security/ReentrancyGuard.sol#4)
        - ^0.8.0 (openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (openzeppelin/contracts/utils/math/SafeMath.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

```
                - uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (WorthToken.sol#495)
                - uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
        State variables written after the call(s):
        - addLiquidity(otherHalf,newBalance) (WorthToken.sol#495)
                - _allowances[owner][spender] = amount (WorthToken.sol#434)
Reentrancy in WorthToken.transferFrom(address,address,uint256) (WorthToken.sol#378-382):
        External calls:
        - _transfer(sender,recipient,amount) (WorthToken.sol#379)
                - uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (WorthToken.sol#510-516)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (WorthToken.sol#379)
                - uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (WorthToken.sol#380)
                - _allowances[owner][spender] = amount (WorthToken.sol#434)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in WorthToken._transfer(address,address,uint256) (WorthToken.sol#439-472):
        External calls:
        - swapAndLiquify(contractTokenBalance) (WorthToken.sol#467)
                - uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (WorthToken.sol#510-516)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (WorthToken.sol#467)
                - uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tAmount) (WorthToken.sol#572)
                - _tokenTransfer(from,to,amount) (WorthToken.sol#471)
Reentrancy in WorthToken.constructor(address) (WorthToken.sol#265-283):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (WorthToken.sol#270-271)
        Event emitted after the call(s):
        - Transfer(address(0),_msgSender(),TOTAL_SUPPLY) (WorthToken.sol#282)
Reentrancy in WorthToken.swapAndLiquify(uint256) (WorthToken.sol#475-498):
        External calls:
        - swapTokensForEth(half) (WorthToken.sol#489)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (WorthToken.sol#510-516)
        - addLiquidity(otherHalf,newBalance) (WorthToken.sol#495)
                - uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (WorthToken.sol#495)
                - uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (WorthToken.sol#435)
                - addLiquidity(otherHalf,newBalance) (WorthToken.sol#495)
        - SwapAndLiquify(half,newBalance,otherHalf) (WorthToken.sol#497)
INFO:Detectors:
WorthToken.allowance(address,address).owner (WorthToken.sol#346) shadows:
        - Ownable.owner() (openzeppelin/contracts/access/Ownable.sol#35-37) (function)
WorthToken._approve(address,address,uint256).owner (WorthToken.sol#430) shadows:
        - Ownable.owner() (openzeppelin/contracts/access/Ownable.sol#35-37) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
WorthToken.constructor(address)._worthDVCFundWallet (WorthToken.sol#265) lacks a zero-check on :
                - worthDVCFundWallet = _worthDVCFundWallet (WorthToken.sol#280)
WorthToken.setWorthDVCFundWallet(address).newWallet (WorthToken.sol#621) lacks a zero-check on :
                - worthDVCFundWallet = newWallet (WorthToken.sol#622)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in WorthToken._transfer(address,address,uint256) (WorthToken.sol#439-472):
        External calls:
        - swapAndLiquify(contractTokenBalance) (WorthToken.sol#467)
                - uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (WorthToken.sol#510-516)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (WorthToken.sol#467)
                - uniswapV2Router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (WorthToken.sol#525-532)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount) (WorthToken.sol#471)
                - _liquidityFee = _previousLiquidityFee (WorthToken.sol#418)
                - _liquidityFee = 0 (WorthToken.sol#412)
        - _tokenTransfer(from,to,amount) (WorthToken.sol#471)
                - _previousLiquidityFee = _liquidityFee (WorthToken.sol#409)
        - _tokenTransfer(from,to,amount) (WorthToken.sol#471)
                - _previousWorthDVCFundFee = _worthDVCFundFee (WorthToken.sol#410)
        - _tokenTransfer(from,to,amount) (WorthToken.sol#471)
                - _worthDVCFundFee = _previousWorthDVCFundFee (WorthToken.sol#419)
                - _worthDVCFundFee = 0 (WorthToken.sol#413)
Reentrancy in WorthToken.constructor(address) (WorthToken.sol#265-283):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (WorthToken.sol#270-271)
        State variables written after the call(s):
        - _isExcludedFromFee[owner()] = true (WorthToken.sol#277)
        - _isExcludedFromFee[address(this)] = true (WorthToken.sol#278)
        - uniswapV2Router = _uniswapV2Router (WorthToken.sol#274)
        - worthDVCFundWallet = _worthDVCFundWallet (WorthToken.sol#280)
Reentrancy in WorthToken.setRouterAddress(address) (WorthToken.sol#656-660):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_newPancakeRouter.factory()).createPair(address(this),_newPancakeRouter.WETH()) (WorthToken.sol#658)
        State variables written after the call(s):
        - uniswapV2Router = _newPancakeRouter (WorthToken.sol#659)
Reentrancy in WorthToken.swapAndLiquify(uint256) (WorthToken.sol#475-498):
        External calls:
        - swapTokensForEth(half) (WorthToken.sol#489)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (WorthToken.sol#510-516)
```

SOLHINT LINTER:

```
WorthToken.sol
   11:1   error    Compiler version 0.8.9 does not satisfy the ^0.5.8 semver requirement                          compiler-version
   46:5   warning  Function name must be in mixedCase                                                             func-name-mixedcase
   47:5   warning  Function name must be in mixedCase                                                             func-name-mixedcase
   63:5   warning  Function name must be in mixedCase                                                             func-name-mixedcase
   83:5   warning  Function name must be in mixedCase                                                             func-name-mixedcase
  246:5   warning  Explicitly mark visibility of state                                                            state-visibility
  265:5   warning  Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  func-visibility
  403:32  warning  Code contains empty blocks                                                                     no-empty-blocks
  431:9   warning  Error message for require is too long                                                          reason-string
  432:9   warning  Error message for require is too long                                                          reason-string
  444:9   warning  Error message for require is too long                                                          reason-string
  445:9   warning  Error message for require is too long                                                          reason-string
  447:13  warning  Error message for require is too long                                                          reason-string
  515:13  warning  Avoid to make time-based decisions in your business logic                                      not-rely-on-time
  531:13  warning  Avoid to make time-based decisions in your business logic                                      not-rely-on-time
  540:9   warning  Error message for require is too long                                                          reason-string
  541:9   warning  Error message for require is too long                                                          reason-string
```

WorthTokenSale.sol

Slither:

```
INFO:Detectors:
WorthTokenSale.withdrawCrypto(address) (WorthTokenSale.sol#279-281) sends eth to arbitrary user
        Dangerous calls:
        - beneficiary.transfer(address(this).balance) (WorthTokenSale.sol#280)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
WorthTokenSale.claimToken() (WorthTokenSale.sol#129-139) ignores return value by Token(tokenAddr).transfer(userAdd,amountToClaim) (WorthTokenSale.sol#135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
WorthTokenSale.ExchangeUSDTforToken(uint256) (WorthTokenSale.sol#148-166) performs a multiplication on the result of a division:
        -tokenAmount = ((amount.mul(10 ** uint256(tokenDecimal)).div(tokenPriceUsd)).mul(10 ** uint256(tokenDecimal))).div(10 ** uint256(tokenDecimal)) (WorthTokenSale.sol#162)
WorthTokenSale.ExchangeBUSDforToken(uint256) (WorthTokenSale.sol#171-189) performs a multiplication on the result of a division:
        -tokenAmount = ((amount.mul(10 ** uint256(tokenDecimal)).div(tokenPriceUsd)).mul(10 ** uint256(tokenDecimal))).div(10 ** uint256(tokenDecimal)) (WorthTokenSale.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in WorthTokenSale.claimToken() (WorthTokenSale.sol#129-139):
        External calls:
        - Token(tokenAddr).transfer(userAdd,amountToClaim) (WorthTokenSale.sol#135)
        State variables written after the call(s):
        - tokenExchanged[userAdd] = 0 (WorthTokenSale.sol#137)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
WorthTokenSale.constructor(address,uint256,uint256,uint256,uint256,uint256,uint256,address,address,uint256)._usdtAddr (WorthTokenSale.sol#92) lacks a zero-check on :
                - usdtAddr = _usdtAddr (WorthTokenSale.sol#101)
WorthTokenSale.constructor(address,uint256,uint256,uint256,uint256,uint256,uint256,address,address,uint256)._busdAddr (WorthTokenSale.sol#93) lacks a zero-check on :
                - busdAddr = _busdAddr (WorthTokenSale.sol#102)
WorthTokenSale.updateUSDTBUSDaddress(address,address).usdt (WorthTokenSale.sol#249) lacks a zero-check on :
                - usdtAddr = usdt (WorthTokenSale.sol#250)
WorthTokenSale.updateUSDTBUSDaddress(address,address).busd (WorthTokenSale.sol#249) lacks a zero-check on :
                - busdAddr = busd (WorthTokenSale.sol#251)
WorthTokenSale.updateTokenAddress(address).newTokenAddr (WorthTokenSale.sol#264) lacks a zero-check on :
                - tokenAddr = newTokenAddr (WorthTokenSale.sol#265)
WorthTokenSale.withdrawCrypto(address).beneficiary (WorthTokenSale.sol#279) lacks a zero-check on :
                - beneficiary.transfer(address(this).balance) (WorthTokenSale.sol#280)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
whitelistAddress(address[]) should be declared external:
        - WorthTokenSale.whitelistAddress(address[]) (WorthTokenSale.sol#109-114)
depositTokens(uint256) should be declared external:
        - WorthTokenSale.depositTokens(uint256) (WorthTokenSale.sol#119-124)
claimToken() should be declared external:
        - WorthTokenSale.claimToken() (WorthTokenSale.sol#129-139)
ExchangeUSDTforToken(uint256) should be declared external:
        - WorthTokenSale.ExchangeUSDTforToken(uint256) (WorthTokenSale.sol#148-166)
ExchangeBUSDforToken(uint256) should be declared external:
        - WorthTokenSale.ExchangeBUSDforToken(uint256) (WorthTokenSale.sol#171-189)
toggleWhitelistStatus() should be declared external:
        - WorthTokenSale.toggleWhitelistStatus() (WorthTokenSale.sol#213-220)
updateTokenPrice(uint256) should be declared external:
        - WorthTokenSale.updateTokenPrice(uint256) (WorthTokenSale.sol#225-227)
updateHardCap(uint256) should be declared external:
        - WorthTokenSale.updateHardCap(uint256) (WorthTokenSale.sol#232-234)
updateTokenContribution(uint256,uint256) should be declared external:
        - WorthTokenSale.updateTokenContribution(uint256,uint256) (WorthTokenSale.sol#240-243)
updateUSDTBUSDaddress(address,address) should be declared external:
        - WorthTokenSale.updateUSDTBUSDaddress(address,address) (WorthTokenSale.sol#249-252)
updateTokenDecimal(uint256) should be declared external:
        - WorthTokenSale.updateTokenDecimal(uint256) (WorthTokenSale.sol#257-259)
updateTokenAddress(address) should be declared external:
        - WorthTokenSale.updateTokenAddress(address) (WorthTokenSale.sol#264-266)
withdrawTokens(address,address) should be declared external:
        - WorthTokenSale.withdrawTokens(address,address) (WorthTokenSale.sol#272-274)
withdrawCrypto(address) should be declared external:
        - WorthTokenSale.withdrawCrypto(address) (WorthTokenSale.sol#279-281)
changeClaimDate(uint256) should be declared external:
        - WorthTokenSale.changeClaimDate(uint256) (WorthTokenSale.sol#286-288)
getTokenBalance(address) should be declared external:
        - WorthTokenSale.getTokenBalance(address) (WorthTokenSale.sol#297-299)
getCryptoBalance() should be declared external:
        - WorthTokenSale.getCryptoBalance() (WorthTokenSale.sol#304-306)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (openzeppelin/contracts/access/Ownable.sol#62-65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
INFO:Detectors:
Event WorthTokenSaleamountTransfered(address,address,address,uint256) (WorthTokenSale.sol#46) is not in CapWords
Parameter WorthTokenSale.whitelistAddress(address[])._recipients (WorthTokenSale.sol#109) is not in mixedCase
Parameter WorthTokenSale.depositTokens(uint256)._amount (WorthTokenSale.sol#119) is not in mixedCase
Function WorthTokenSale.ExchangeUSDTforToken(uint256) (WorthTokenSale.sol#148-166) is not in mixedCase
Parameter WorthTokenSale.ExchangeUSDTforToken(uint256)._amount (WorthTokenSale.sol#148) is not in mixedCase
Function WorthTokenSale.ExchangeBUSDforToken(uint256) (WorthTokenSale.sol#171-189) is not in mixedCase
Parameter WorthTokenSale.ExchangeBUSDforToken(uint256)._amount (WorthTokenSale.sol#171) is not in mixedCase
Parameter WorthTokenSale.withdrawTokens(address,address)._tokenAddr (WorthTokenSale.sol#272) is not in mixedCase
Parameter WorthTokenSale.changeClaimDate(uint256)._claimDate (WorthTokenSale.sol#286) is not in mixedCase
Parameter WorthTokenSale.getTokenBalance(address)._tokenAddr (WorthTokenSale.sol#297) is not in mixedCase
Modifier WorthTokenSale._contractUp() (WorthTokenSale.sol#58-61) is not in mixedCase
Modifier WorthTokenSale._saleEnded() (WorthTokenSale.sol#68-71) is not in mixedCase
Modifier WorthTokenSale._saleNotEnded() (WorthTokenSale.sol#73-76) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
WorthTokenSale.slitherConstructorVariables() (WorthTokenSale.sol#18-307) uses literals with too many digits:
        - tokenPriceUsd = 35000000000000 (WorthTokenSale.sol#26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

```
INFO:Detectors:
WorthTokenSale.claimToken() (WorthTokenSale.sol#129-139) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > claimDate,Cannot Claim Now) (WorthTokenSale.sol#132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.8.9', '^0.8.0']
        - 0.8.9 (WorthTokenSale.sol#7)
        - ^0.8.0 (openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (openzeppelin/contracts/security/ReentrancyGuard.sol#4)
        - ^0.8.0 (openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (openzeppelin/contracts/utils/math/SafeMath.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Context._msgData() (openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
SafeMath.div(uint256,uint256,string) (openzeppelin/contracts/utils/math/SafeMath.sol#191-200) is never used and should be removed
SafeMath.mod(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#151-153) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (openzeppelin/contracts/utils/math/SafeMath.sol#217-226) is never used and should be removed
SafeMath.sub(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#107-109) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (openzeppelin/contracts/utils/math/SafeMath.sol#168-177) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#22-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#64-69) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#76-81) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#47-57) is never used and should be removed
SafeMath.trySub(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#35-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.9 (WorthTokenSale.sol#7) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/access/Ownable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/security/ReentrancyGuard.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/utils/math/SafeMath.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
Reentrancy in WorthTokenSale.ExchangeBUSDforToken(uint256) (WorthTokenSale.sol#171-189):
        External calls:
        - require(bool)(Token(busdAddr).transferFrom(msg.sender,address(this),_amount)) (WorthTokenSale.sol#172)
        State variables written after the call(s):
        - balances[msg.sender] = balances[msg.sender].add(_amount) (WorthTokenSale.sol#176)
        - tokenExchanged[userAdd] += tokenAmount (WorthTokenSale.sol#186)
        - totalHardCap = totalHardCap.add(amount) (WorthTokenSale.sol#184)
        - totalTransaction = totalTransaction.add(1) (WorthTokenSale.sol#183)
Reentrancy in WorthTokenSale.ExchangeUSDTforToken(uint256) (WorthTokenSale.sol#148-166):
        External calls:
        - require(bool)(Token(usdtAddr).transferFrom(msg.sender,address(this),_amount)) (WorthTokenSale.sol#149)
        State variables written after the call(s):
        - balances[msg.sender] = balances[msg.sender].add(_amount) (WorthTokenSale.sol#153)
        - tokenExchanged[userAdd] += tokenAmount (WorthTokenSale.sol#163)
        - totalHardCap = totalHardCap.add(amount) (WorthTokenSale.sol#161)
        - totalTransaction = totalTransaction.add(1) (WorthTokenSale.sol#160)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in WorthTokenSale.ExchangeBUSDforToken(uint256) (WorthTokenSale.sol#171-189):
        External calls:
        - require(bool)(Token(busdAddr).transferFrom(msg.sender,address(this),_amount)) (WorthTokenSale.sol#172)
        Event emitted after the call(s):
        - UsdDeposited(msg.sender,_amount) (WorthTokenSale.sol#188)
Reentrancy in WorthTokenSale.ExchangeUSDTforToken(uint256) (WorthTokenSale.sol#148-166):
        External calls:
        - require(bool)(Token(usdtAddr).transferFrom(msg.sender,address(this),_amount)) (WorthTokenSale.sol#149)
        Event emitted after the call(s):
        - UsdDeposited(msg.sender,_amount) (WorthTokenSale.sol#165)
Reentrancy in WorthTokenSale.claimToken() (WorthTokenSale.sol#129-139):
        External calls:
        - Token(tokenAddr).transfer(userAdd,amountToClaim) (WorthTokenSale.sol#135)
        Event emitted after the call(s):
        - TokenTransfer(userAdd,amountToClaim) (WorthTokenSale.sol#136)
Reentrancy in WorthTokenSale.depositTokens(uint256) (WorthTokenSale.sol#119-124):
        External calls:
        - require(bool)(Token(tokenAddr).transferFrom(msg.sender,address(this),_amount)) (WorthTokenSale.sol#121)
        Event emitted after the call(s):
        - TokenDeposited(msg.sender,_amount) (WorthTokenSale.sol#122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

SOLHINT LINTER:

```
WorthTokenSale.sol
    7:1   error    Compiler version 0.8.9 does not satisfy the ^0.5.8 semver requirement                          compiler-version
   18:1   warning  Contract has 19 states declarations but allowed no more than 15                               max-states-count
   46:5   warning  Event name must be in CamelCase                                                               event-name-camelcase
   59:9   warning  Provide an error message for require                                                          reason-string
   64:9   warning  Provide an error message for require                                                          reason-string
   69:9   warning  Provide an error message for require                                                          reason-string
   74:9   warning  Provide an error message for require                                                          reason-string
   89:5   warning  Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  func-visibility
  121:9   warning  Provide an error message for require                                                          reason-string
  132:17  warning  Avoid to make time-based decisions in your business logic                                     not-rely-on-time
  134:9   warning  Error message for require is too long                                                        reason-string
  137:9   warning  Possible reentrancy vulnerabilities. Avoid state changes after transfer                      reentrancy
  142:23  warning  Visibility modifier must be first in list of modifiers                                       visibility-modifier-order
  142:32  warning  Code contains empty blocks                                                                   no-empty-blocks
  148:5   warning  Function name must be in mixedCase                                                           func-name-mixedcase
  149:9   warning  Provide an error message for require                                                         reason-string
  159:9   warning  Error message for require is too long                                                        reason-string
  171:5   warning  Function name must be in mixedCase                                                           func-name-mixedcase
  172:9   warning  Provide an error message for require                                                         reason-string
  182:9   warning  Error message for require is too long                                                        reason-string
  199:9   warning  Provide an error message for require                                                         reason-string
  207:38  warning  Avoid to make time-based decisions in your business logic                                    not-rely-on-time
  273:9   warning  Provide an error message for require                                                         reason-string
```

WorthTokenTimeLock.sol

Slither:

```
INFO:Detectors:
WorthTokenTimeLock.createMultipleLocks(address,address,uint256[],uint256[]) (WorthTokenTimeLock.sol#76-101) has external calls inside a loop: require(bool)(Token(_tokenAddress).transfe
rFrom(msg.sender,address(this),_amounts[i])) (WorthTokenTimeLock.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in WorthTokenTimeLock.withdrawTokens(uint256) (WorthTokenTimeLock.sol#149-173):
        External calls:
        - require(bool)(Token(lockedToken[_id].tokenAddress).transfer(msg.sender,lockedToken[_id].tokenAmount)) (WorthTokenTimeLock.sol#171)
        Event emitted after the call(s):
        - LogWithdrawal(msg.sender,lockedToken[_id].tokenAmount) (WorthTokenTimeLock.sol#172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
WorthTokenTimeLock.withdrawTokens(uint256) (WorthTokenTimeLock.sol#149-173) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool)(block.timestamp >= lockedToken[_id].unlockTime) (WorthTokenTimeLock.sol#150)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.8.9', '^0.8.0']
        - 0.8.9 (WorthTokenTimeLock.sol#5)
        - ^0.8.0 (openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (openzeppelin/contracts/utils/math/SafeMath.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
WorthTokenTimeLock.createMultipleLocks(address,address,uint256[],uint256[]) (WorthTokenTimeLock.sol#76-101) has costly operations inside a loop:
        - _id = ++ depositId (WorthTokenTimeLock.sol#88)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
```

```
INFO:Detectors:
WorthTokenTimeLock.lockTokens(address,address,uint256,uint256) (WorthTokenTimeLock.sol#48-67) uses literals with too many digits:
        - require(bool)(_unlockTime < 10000000000) (WorthTokenTimeLock.sol#50)
WorthTokenTimeLock.createMultipleLocks(address,address,uint256[],uint256[]) (WorthTokenTimeLock.sol#76-101) uses literals with too many digits:
        - require(bool)(_unlockTimes[i] < 10000000000) (WorthTokenTimeLock.sol#83)
WorthTokenTimeLock.extendLockDuration(uint256,uint256) (WorthTokenTimeLock.sol#107-114) uses literals with too many digits:
        - require(bool)(_unlockTime < 10000000000) (WorthTokenTimeLock.sol#108)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
lockTokens(address,address,uint256,uint256) should be declared external:
        - WorthTokenTimeLock.lockTokens(address,address,uint256,uint256) (WorthTokenTimeLock.sol#48-67)
createMultipleLocks(address,address,uint256[],uint256[]) should be declared external:
        - WorthTokenTimeLock.createMultipleLocks(address,address,uint256[],uint256[]) (WorthTokenTimeLock.sol#76-101)
extendLockDuration(uint256,uint256) should be declared external:
        - WorthTokenTimeLock.extendLockDuration(uint256,uint256) (WorthTokenTimeLock.sol#107-114)
transferLocks(uint256,address) should be declared external:
        - WorthTokenTimeLock.transferLocks(uint256,address) (WorthTokenTimeLock.sol#120-144)
withdrawTokens(uint256) should be declared external:
        - WorthTokenTimeLock.withdrawTokens(uint256) (WorthTokenTimeLock.sol#149-173)
getTotalTokenBalance(address) should be declared external:
        - WorthTokenTimeLock.getTotalTokenBalance(address) (WorthTokenTimeLock.sol#178-181)
getTokenBalanceByAddress(address,address) should be declared external:
        - WorthTokenTimeLock.getTokenBalanceByAddress(address,address) (WorthTokenTimeLock.sol#187-190)
getAllDepositIds() should be declared external:
        - WorthTokenTimeLock.getAllDepositIds() (WorthTokenTimeLock.sol#195-198)
getDepositDetails(uint256) should be declared external:
        - WorthTokenTimeLock.getDepositDetails(uint256) (WorthTokenTimeLock.sol#203-207)
getDepositsByWithdrawalAddress(address) should be declared external:
        - WorthTokenTimeLock.getDepositsByWithdrawalAddress(address) (WorthTokenTimeLock.sol#212-215)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (openzeppelin/contracts/access/Ownable.sol#62-65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
INFO:Detectors:
Context._msgData() (openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
SafeMath.div(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#135-137) is never used and should be removed
SafeMath.div(uint256,uint256,string) (openzeppelin/contracts/utils/math/SafeMath.sol#191-200) is never used and should be removed
SafeMath.mod(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#151-153) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (openzeppelin/contracts/utils/math/SafeMath.sol#217-226) is never used and should be removed
SafeMath.mul(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#121-123) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (openzeppelin/contracts/utils/math/SafeMath.sol#168-177) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#22-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#64-69) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#76-81) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#47-57) is never used and should be removed
SafeMath.trySub(uint256,uint256) (openzeppelin/contracts/utils/math/SafeMath.sol#35-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.9 (WorthTokenTimeLock.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/access/Ownable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/utils/math/SafeMath.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter WorthTokenTimeLock.lockTokens(address,address,uint256,uint256)._tokenAddress (WorthTokenTimeLock.sol#48) is not in mixedCase
Parameter WorthTokenTimeLock.lockTokens(address,address,uint256,uint256)._withdrawalAddress (WorthTokenTimeLock.sol#48) is not in mixedCase
Parameter WorthTokenTimeLock.lockTokens(address,address,uint256,uint256)._amount (WorthTokenTimeLock.sol#48) is not in mixedCase
Parameter WorthTokenTimeLock.lockTokens(address,address,uint256,uint256)._unlockTime (WorthTokenTimeLock.sol#48) is not in mixedCase
Parameter WorthTokenTimeLock.createMultipleLocks(address,address,uint256[],uint256[])._tokenAddress (WorthTokenTimeLock.sol#76) is not in mixedCase
Parameter WorthTokenTimeLock.createMultipleLocks(address,address,uint256[],uint256[])._withdrawalAddress (WorthTokenTimeLock.sol#76) is not in mixedCase
Parameter WorthTokenTimeLock.createMultipleLocks(address,address,uint256[],uint256[])._amounts (WorthTokenTimeLock.sol#76) is not in mixedCase
Parameter WorthTokenTimeLock.createMultipleLocks(address,address,uint256[],uint256[])._unlockTimes (WorthTokenTimeLock.sol#76) is not in mixedCase
Parameter WorthTokenTimeLock.extendLockDuration(uint256,uint256)._id (WorthTokenTimeLock.sol#107) is not in mixedCase
Parameter WorthTokenTimeLock.extendLockDuration(uint256,uint256)._unlockTime (WorthTokenTimeLock.sol#107) is not in mixedCase
Parameter WorthTokenTimeLock.transferLocks(uint256,address)._id (WorthTokenTimeLock.sol#120) is not in mixedCase
Parameter WorthTokenTimeLock.transferLocks(uint256,address)._receiverAddress (WorthTokenTimeLock.sol#120) is not in mixedCase
Parameter WorthTokenTimeLock.withdrawTokens(uint256)._id (WorthTokenTimeLock.sol#149) is not in mixedCase
Parameter WorthTokenTimeLock.getTotalTokenBalance(address)._tokenAddress (WorthTokenTimeLock.sol#178) is not in mixedCase
Parameter WorthTokenTimeLock.getTokenBalanceByAddress(address,address)._tokenAddress (WorthTokenTimeLock.sol#187) is not in mixedCase
Parameter WorthTokenTimeLock.getTokenBalanceByAddress(address,address)._walletAddress (WorthTokenTimeLock.sol#187) is not in mixedCase
Parameter WorthTokenTimeLock.getDepositDetails(uint256)._id (WorthTokenTimeLock.sol#203) is not in mixedCase
Parameter WorthTokenTimeLock.getDepositsByWithdrawalAddress(address)._withdrawalAddress (WorthTokenTimeLock.sol#212) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

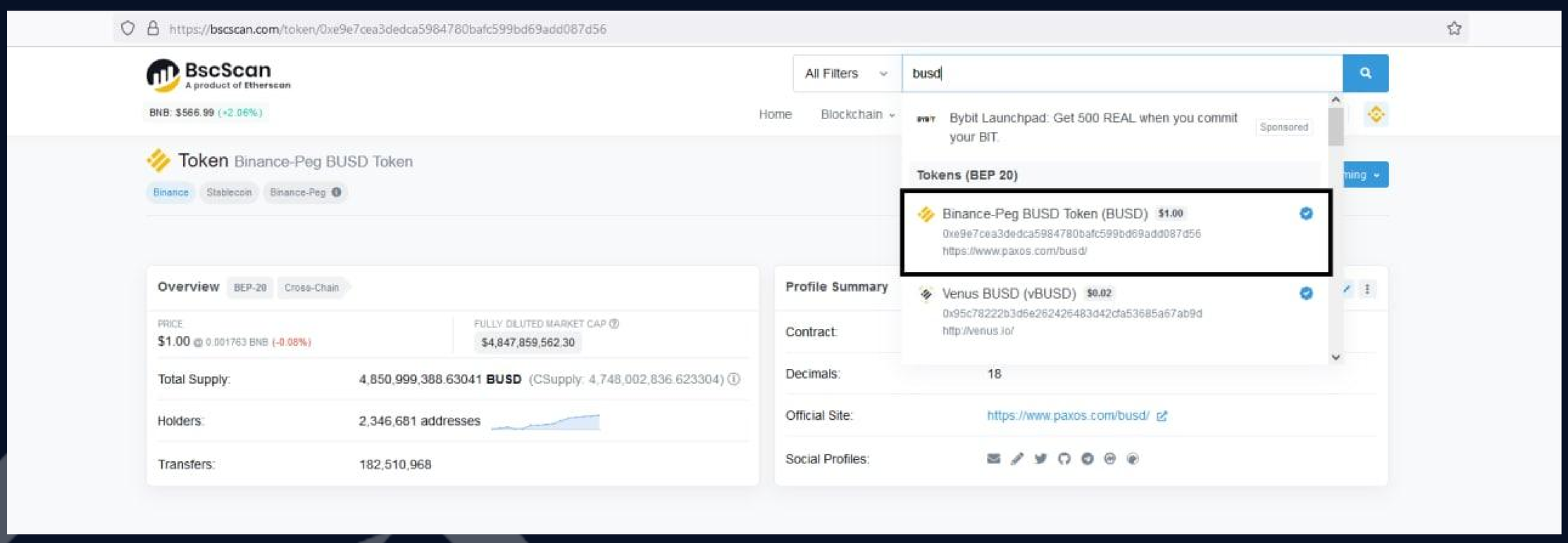
SOLHINT LINTER:

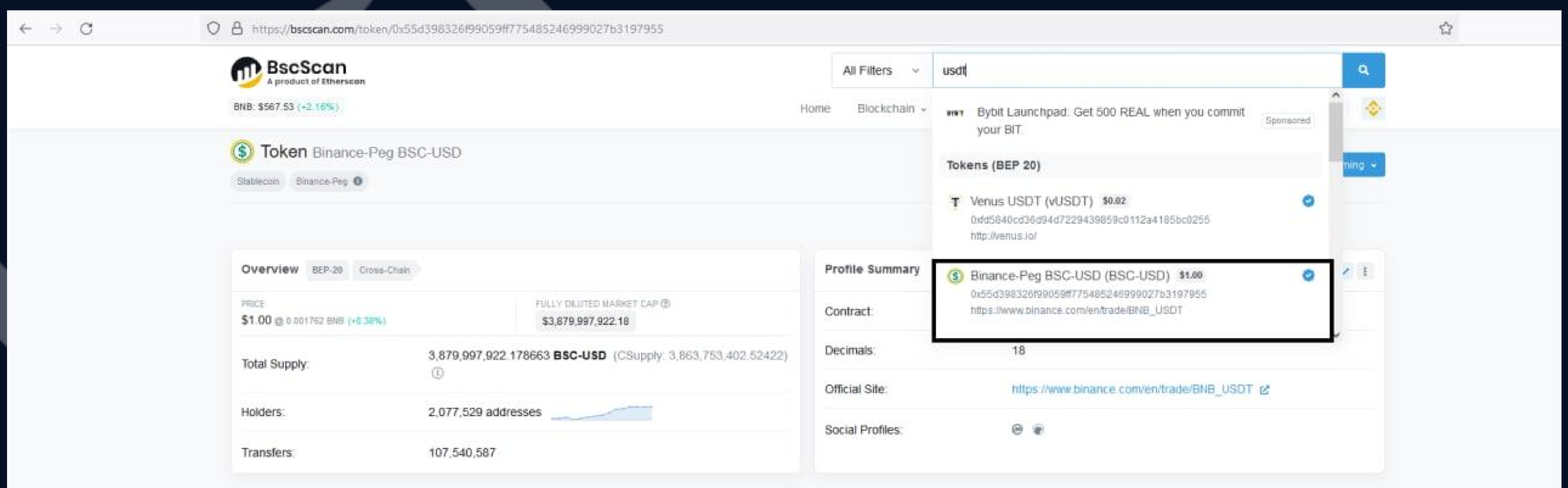
```
WorthTokenTimeLock.sol
   5:1   error    Compiler version 0.8.9 does not satisfy the ^0.5.8 semver requirement  compiler-version
  40:25  warning  Variable name must be in mixedCase                                     var-name-mixedcase
  40:48  warning  Variable name must be in mixedCase                                     var-name-mixedcase
  49:9   warning  Provide an error message for require                                   reason-string
  50:9   warning  Provide an error message for require                                   reason-string
  66:9   warning  Provide an error message for require                                   reason-string
  77:9   warning  Provide an error message for require                                   reason-string
  78:9   warning  Provide an error message for require                                   reason-string
  82:13  warning  Provide an error message for require                                   reason-string
  83:13  warning  Provide an error message for require                                   reason-string
  99:13  warning  Provide an error message for require                                   reason-string
 108:9   warning  Provide an error message for require                                   reason-string
 109:9   warning  Provide an error message for require                                   reason-string
 110:9   warning  Provide an error message for require                                   reason-string
 121:9   warning  Provide an error message for require                                   reason-string
 122:9   warning  Provide an error message for require                                   reason-string
 150:9   warning  Provide an error message for require                                   reason-string
 150:17  warning  Avoid to make time-based decisions in your business logic              not-rely-on-time
 151:9   warning  Provide an error message for require                                   reason-string
 152:9   warning  Provide an error message for require                                   reason-string
 171:9   warning  Provide an error message for require                                   reason-string
 178:64  warning  Visibility modifier must be first in list of modifiers                visibility-modifier-order
 187:92  warning  Visibility modifier must be first in list of modifiers                visibility-modifier-order
 195:38  warning  Visibility modifier must be first in list of modifiers                visibility-modifier-order
 203:51  warning  Visibility modifier must be first in list of modifiers                visibility-modifier-order
 212:79  warning  Visibility modifier must be first in list of modifiers                visibility-modifier-order
```

Closing Summary

In this report, we have considered the security of the Worthpad platform. We performed our audit according to the procedure described above.

Several of the issues were discovered, analyzed, and evaluated during the audit, which included the deployment of contracts on the BSC testnet for testing. As a result, some of these issues did not apply to the mainnet, as confirmed by the Auditee. For instance, the auditee confirmed that both the USDT and BUSD contracts used by WorthTokenSale.sol contain 18 decimal places in the mainnet, as specified below:
BUSD BEP-20 Smart Contract Address: https://bscscan.com/token/0xe9e7cea3dedca5984780bafc599bd69add087d56
Binance-Peg BSC-USD BEP-20 Smart Contract Address: https://bscscan.com/token/0x55d398326f99059ff775485246999027b3197955





In conclusion, all the issues were fixed and acknowledged by the Auditee.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the Worthpad platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. 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. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Worthpad Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
December, 2021

For

WORTHPAD

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com