CERTIK

Preliminary Comments

Worthpad

Dec 13th, 2021

Table of Contents

Summary

This report has been prepared for Worthpad to discover issues and vulnerabilities in the source code of the Worthpad project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Worthpad
Platform	BSC
Language	Solidity
Codebase	https://github.com/worthpad/Worth
Commit	f992dc35bf3fa83bb1222142b11c59affcd80a06 f18eb432f476ab7b37bf7e1d45bdc33f10427215

Audit Summary

Delivery Date	Dec 13, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	WorthToken, WorthTokenSale, WorthTokenTimeLock

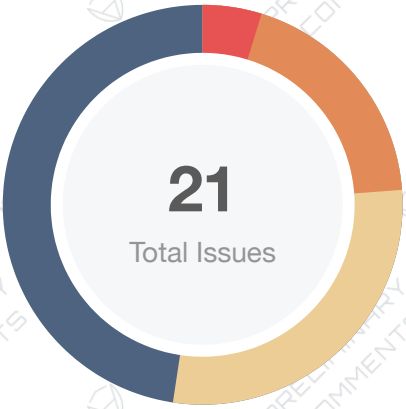
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	1	0	0	0	1	0
Major	4	0	0	4	0	0
Medium	0	0	0	0	0	0
Minor	6	2	0	2	0	2
Informational	10	3	0	2	0	5
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
WTW	projects/WorthPad/WorthToken.sol	cdd4bbc3882ad9cf806ca41d8d0174a4fa930fa15ee2ca604a45f104b43cff02
WTS	projects/WorthPad/WorthTokenSale.sol	8b50fb4010acc2a1e88f06bdb86fa32b9520cf18909cdfeaaf89880a3749b30a
WTT	projects/WorthPad/WorthTokenTimeLock.sol	b7270fee31dab2795094aaf351736c0c6b720d50e287bd9272761ba74e15786d
WTP	projects/WorthPad/WorthToken.sol	1cc506f23e08c52e4a687f9f563f4a0d58701bb2a17e1e5cd36d65aa84357aaa
WTC	projects/WorthPad/WorthTokenSale.sol	d6f12955f94decc49fbc15950e39b0d46b5e48030024e9efc5ea4cba52fad1b0
WTL	projects/WorthPad/WorthTokenTimeLock.sol	2c42790e6c52debbca27f3471b5a9ffbcd92a3db5016e5fb4f5642d26a46ced4

Findings



Critical	1 (4.76%)
Major	4 (19.05%)
Medium	0 (0.00%)
Minor	6 (28.57%)
Informational	10 (47.62%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
WTS-01	Token Price Is Totally Controlled By Owner	Centralization / Privilege	Critical	Partially Resolved
WTS-02	Centralization Risk	Centralization / Privilege	Major	Acknowledged
WTS-03	Requisite Value of ERC-20 `transferFrom()` / `transfer()` Call	Logical Issue	Minor	Resolved
WTS-04	Usage of `transfer()` for Sending BNB	Volatile Code	Minor	Pending
WTS-05	Missing Error Messages	Coding Style	Informational	Resolved
WTS-06	Unknown Implementation Of Interface `Token`	Volatile Code	Informational	Resolved
WTT-01	Requisite Value of ERC-20 `transferFrom()` / `transfer()` Call	Logical Issue	Minor	Resolved
WTT-02	Should Validate Input `_unlockTime` Larger Than Old Value For Function `extendLockDuration`	Logical Issue	Minor	Pending
WTT-03	Missing Error Messages	Coding Style	Informational	Resolved
WTT-04	Unbounded Loop	Logical Issue	Informational	Acknowledged
WTT-05	Redundant Code	Logical Issue, Gas Optimization	Informational	Pending
WTT-06	Unknown Implementation Of Interface `Token`	Volatile Code	Informational	Resolved

ID	Title	Category	Severity	Status
WTW-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
WTW-02	Centralized risk in `addLiquidity`	**Centralization / Privilege**	**Major**	Acknowledged
WTW-03	Initial Token Distribution	**Centralization / Privilege**	**Major**	Acknowledged
WTW-04	Third Party Dependencies	Volatile Code	Minor	Acknowledged
WTW-05	Potential Sandwich Attacks	Logical Issue	Minor	Acknowledged
WTW-06	Confuse Function Name `setMinSell`	Language Specific	Informational	Resolved
WTW-07	Return value not handled	Volatile Code	Informational	Acknowledged
WTW-08	Missing Event Emitting	Coding Style	Informational	Pending
WTW-09	Unused Base Contract `ReentrancyGuard`	Volatile Code	Informational	Pending

WTS-01 | Token Price Is Totally Controlled By Owner

Category	Severity	Location	Status
Centralization / Privilege	Critical	projects/WorthPad/WorthTokenSale.sol (1): 162, 185, 226	Partially Resolved

Description

Within the contract `WorthTokenSale`, the BUSD/USDT price of the token is set by the owner.
The owner has absolute control of the token price which could take advantage of the potential price scissors.

Alleviation

[Worthpad Team]: The Token Sale price can't be changed once the Token Sale starts. Added the restriction in this function in commit:
https://github.com/worthpad/worth/commit/3c812a6de7ba390109791c1007334d2563cf36bc

WTS-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	projects/WorthPad/WorthTokenSale.sol (1): 109, 286, 279, 272, 264, 257, 249, 240, 232, 225, 205, 213, 205, 197	Acknowledged

Description

In the contract `WorthToken`, the role `owner` has the authority over the following function:

- whitelistAddress
- powerUpContract
- toggleWhitelistStatus
- updateTokenPrice
- updateHardCap
- updateTokenContribution
- updateUSDTBUSDaddress
- updateTokenDecimal
- updateTokenAddress
- withdrawTokens
- withdrawCrypto
- changeClaimDate

Any compromise to the `owner` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Worthpad Team]: Worthpad team will transfer the ownership of smart contracts to a Gnosis multi-sig wallet to prevent a single point of failure. The ownership will be transferred to the Worthpad community through the WorthDAO. https://worthpad.medium.com/worth-token-sale-and-security-of-investors-funds-e8c42c946ef9

WTS-03 | Requisite Value of ERC-20 `transferFrom()` / `transfer()` Call

Category	Severity	Location	Status
Logical Issue	Minor	projects/WorthPad/WorthTokenSale.sol (1): 273, 172, 149, 121, 135	Resolved

Description

While the ERC-20 implementation does necessitate that the `transferFrom()` / `transfer()` function returns a `bool` variable yielding `true`, many token implementations do not return anything i.e. Tether (USDT) leading to unexpected halts in code execution.

Recommendation

We advise that the `SafeERC20.sol` library is utilized by OpenZeppelin to ensure that the `transferFrom()` / `transfer()` function is safely invoked in all circumstances.

Alleviation

[Worthpad Team]: WorthTokenSale contract uses SafeERC20 specification for transferFrom() / transfer() Call in commit:
https://github.com/worthpad/worth/commit/80c1de3e50b3d16960b4814f24b5fbfb269f3716
WorthTokenTimeLock contract uses SafeERC20 specification for transferFrom() / transfer() Call in commit:
https://github.com/worthpad/worth/commit/f992dc35bf3fa83bb1222142b11c59affcd80a06

WTS-04 | Usage of `transfer()` for Sending BNB

Category	Severity	Location	Status
Volatile Code	Minor	projects/WorthPad/WorthTokenSale.sol (1): 280	Pending

Description

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs. This also includes BNB transfers on the Binance Smart Chain.

Recommendation

We advise that the linked `.transfer()` and `.send()` calls are substituted with the utilization of the `sendValue()` function from the `Address.sol` implementation of OpenZeppelin either by directly importing the library or copying the linked code. We also recommend adding additional protection using Reentrancy Guard.

Alleviation

[Worthpad Team]: us `(bool success, ) = beneficiary.call{value:address(this).balance}("");` instead of `transfer()` in the commit:
https://github.com/worthpad/worth/commit/3c812a6de7ba390109791c1007334d2563cf36bc

WTS-05 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	projects/WorthPad/WorthTokenSale.sol (1): 273, 199, 172, 149, 120~121, 74, 69, 64, 59	Resolved

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise refactoring the linked requires as below with clear description:

```
14 function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
15     c = a + b;
16     require(c >= a, "SafeMath: addition overflow");
17 }
```

Alleviation

[Worthpad Team]: Fixed. Every require statement now has a message.

WTS-06 | Unknown Implementation Of Interface `Token`

Category	Severity	Location	Status
Volatile Code	Informational	projects/WorthPad/WorthTokenSale.sol (1): 10~15	Resolved

Description

As the implementation of the contract `WorthTokenTimeLock`, the function `lockTokens` could support deposit with the amount of token which implements interface `Token`. We could only know the contract supported tokens would be the implementation of ERC-1363 as the interface definition:

```
15      function approveAndCall(address spender, uint tokens, bytes memory data) external
returns (bool success);
```

But `approveAndCall` is not used within the contract `WorthTokenTimeLock`. It looks like that interface `ERC-20` is enough for use.

The token interaction could be more solidified by using the standard interface definition like ERC-1362 or ERC-20.

Recommendation

Advice to check if the contract does not need the `approveAndCall` interface in design.
Advice to use standard ERC-1363 or ERC-20 interface for `WorthTokenTimeLock` and `WorthTokenSale`.

Alleviation

[Worthpad Team]: Fixed.The function is removed and IERC20 interface is used.

WTT-01 | Requisite Value of ERC-20 `transferFrom()` / `transfer()` Call

Category	Severity	Location	Status
Logical Issue	Minor	projects/WorthPad/WorthTokenTimeLock.sol (1): 171, 99, 66	Resolved

Description

While the ERC-20 implementation does necessitate that the `transferFrom()` / `transfer()` function returns a `bool` variable yielding `true`, many token implementations do not return anything i.e. Tether (USDT) leading to unexpected halts in code execution.

Recommendation

We advise that the `SafeERC20.sol` library is utilized by OpenZeppelin to ensure that the `transferFrom()` / `transfer()` function is safely invoked in all circumstances.

Alleviation

[Worthpad Team]: WorthTokenSale contract uses SafeERC20 specification for transferFrom() / transfer() Call in commit:
https://github.com/worthpad/worth/commit/80c1de3e50b3d16960b4814f24b5fbfb269f3716
WorthTokenTimeLock contract uses SafeERC20 specification for transferFrom() / transfer() Call in commit:
https://github.com/worthpad/worth/commit/f992dc35bf3fa83bb1222142b11c59affcd80a06

WTT-02 | Should Validate Input `_unlockTime` Larger Than Old Value For Function `extendLockDuration`

Category	Severity	Location	Status
Logical Issue	Minor	projects/WorthPad/WorthTokenTimeLock.sol (1): 113	Pending

Description

Within function `extendLockDuration`, the input `_unlockTime` is not checked to make the implementation is consistent with the function name.

Recommendation

Advise to check _unlockTime is larger than lockedToken[id].unlockTime.

Alleviation

[Worthpad Team]: Fixed by adding `require(_unlockTime >= lockedToken[_id].unlockTime,"Cannot have time duration less than the previous one");` in commit: https://github.com/worthpad/worth/commit/00c462b95552c6748154e388b74398ad0db44bd2

WTT-03 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	projects/WorthPad/WorthTokenTimeLock.sol (1): 171, 150~152, 99, 77~78, 66, 121~122, 108~110, 82~83, 49~50	Resolved

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise refactoring the linked requires as below with clear description:

```
14 function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
15     c = a + b;
16     require(c >= a, "SafeMath: addition overflow");
17 }
```

Alleviation

[Worthpad Team]: Fixed. Every require statement now has a message.

WTT-04 | Unbounded Loop

Category	Severity	Location	Status
Logical Issue	Informational	projects/WorthPad/WorthTokenTimeLock.sol (1): 81	Acknowledged

Description

The `for` loop within functions `createMultipleLocks` takes the following variable `_amounts.length`, as the maximal iteration times. If the size of the array is very large, it could exceed the gas limit to execute the functions. In this case, the contract might suffer from DoS (Denial of Service) situation.

Recommendation

We recommend the team review the design and ensure this would not cause loss to the project.

Alleviation

[Worthpad Team]: This function is for locking tokens. There's no incentive for someone to execute such an attack and pay the network fee.

WTT-05 | Redundant Code

Category	Severity	Location	Status
Logical Issue, Gas Optimization	Informational	projects/WorthPad/WorthTokenTimeLock.sol (1): 35, 192~198	Pending

Description

Redundant Variable `allDepositIds` and function `getAllDepositIds`.
As the implementation of the contract, `allDepositIds` only contains the id for deposits. There only operation of this array is inserting `++depositId` for the new created `lock`.
So the elements of this array would be a sequence from 1 to `depositId`. There are no need to store this array. The `depositId` value is enough for using.

Recommendation

Advise to remove stored array `allDepositIds`

Alleviation

[Worthpad Team]: Fixed.

WTT-06 | Unknown Implementation Of Interface `Token`

Category	Severity	Location	Status
Volatile Code	Informational	projects/WorthPad/WorthTokenTimeLock.sol (1): 10~17	Resolved

Description

As the implementation of the contract `WorthTokenTimeLock`, the function `lockTokens` could support deposit with the amount of token which implements interface `Token`. We could only know the contract supported tokens would be the implementation of ERC-1363 as the interface definition:

```
15      function approveAndCall(address spender, uint tokens, bytes memory data) external
returns (bool success);
```

But `approveAndCall` is not used within the contract `WorthTokenTimeLock`. It looks like that interface `ERC-20` is enough for use.

The token interaction could be more solidified by using the standard interface definition like ERC-1362 or ERC-20.

Recommendation

Advice to check if the contract does not need the `approveAndCall` interface in design.
Advice to use standard ERC-1363 or ERC-20 interface for `WorthTokenTimeLock` and `WorthTokenSale`.

Alleviation

[Worthpad Team]: Fixed.The function is removed and IERC20 interface is used.

WTW-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	projects/WorthPad/WorthToken.sol (1): 446~448, 665, 656, 649, 642, 635, 628, 621, 607, 593, 586, 579	Acknowledged

Description

In the contract `WorthToken`, the role `owner` has the authority over the following function:

- excludeFromFee
- includeInFee
- disableAllFees
- enableAllFees
- setWorthDVCFundWallet
- setLiquidityFeePercent
- setWorthDVCFundFeePercent
- setMinSell
- setMaxTxAmount
- setRouterAddress
- setSwapAndLiquifyEnabled

The role `owner` has the privilege over the following function:

- _transfer (unrestricted _maxTxAmount)

Any compromise to the `owner` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Worthpad Team]: Worthpad team will transfer the ownership of smart contracts to a Gnosis multi-sig wallet to prevent a single point of failure. The ownership will be transferred to the Worthpad community through the WorthDAO. https://worthpad.medium.com/worth-token-sale-and-security-of-investors-funds-e8c42c946ef9

WTW-02 | Centralized risk in `addLiquidity`

Category	Severity	Location	Status
Centralization / Privilege	Major	projects/WorthPad/WorthToken.sol (1): 530	Acknowledged

Description

```
525 // add the liquidity
526 uniswapV2Router.addLiquidityETH{value: bnbAmount}(
527     address(this),
528     tokenAmount,
529     0, // slippage is unavoidable
530     0, // slippage is unavoidable
531     owner(),
532     block.timestamp
533 );
```

The `addLiquidity` function calls the `uniswapV2Router.addLiquidityETH` function with the `to` address specified as `owner()` for acquiring the generated LP tokens from the `WothToken-BNB` pool. As a result, over time the `_owner` address will accumulate a significant portion of LP tokens.If the `_owner` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

We advise the `to` address of the `uniswapV2Router.addLiquidityETH` function call to be replaced by the contract itself, i.e. `address(this)`, and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the `_owner` account is compromised. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

[Worthpad Team]: Worthpad team will transfer the ownership of smart contracts to a Gnosis multi-sig wallet to prevent a single point of failure. The ownership will be transferred to the Worthpad community through the WorthDAO. https://worthpad.medium.com/worth-token-sale-and-security-of-investors-funds-e8c42c946ef9

WTW-03 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	Major	projects/WorthPad/WorthToken.sol (1): 266	Acknowledged

Description

All of the WorthToken tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute WorthToken tokens without obtaining the consensus of the community.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

Alleviation

[Worthpad Team]: Worthpad team will transfer the ownership of smart contracts to a Gnosis multi-sig wallet to prevent a single point of failure. The ownership will be transferred to the Worthpad community through the WorthDAO. https://worthpad.medium.com/worth-token-sale-and-security-of-investors-funds-e8c42c946ef9
The tokens are going to be locked/vested in Gnosis multi-sig wallets as per the vesting schedule before the token sale starts and all the details including wallet addresses will be communicated to the community and public at large.

WTW-04 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	projects/WorthPad/WorthToken.sol (1): 268	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party PancakeSwap protocols. The scope of the audit treats 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of WorthToken requires interaction with PancakeSwap Router, PancakeSwap Pair, etc. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[Wothpad Team]: Worthpad team will constantly monitor the status of PancakeSwap contracts (3rd parties) to mitigate the side effects when unexpected activities are observed.

WTW-05 | Potential Sandwich Attacks

Category	Severity	Location	Status
Logical Issue	Minor	projects/WorthPad/WorthToken.sol (1): 528~529, 512	Acknowledged

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by backrunning (after the transaction being attacked) a transaction to sell the asset.

The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- `uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens`
- `uniswapV2Router.addLiquidityETH`

Recommendation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

Alleviation

[Worthpad Team]: We are planning to use the following strategy to alleviate this issue:

- We are planning to set `amountOutMin` at 0.1 BNB (~ $50) and reduce the numTokensSellToAddToLiquidity proportionately (~ 150,000 WORTH) based on Pre-Sale price of 1 WORTH = $0.00035.
- Other options would be implementing a price oracle or using TWAP. Our strategy will smoothen out the price fluctuations that may be caused due to auto-liquidity addition to the smart contract.

Preventing sandwich attacks is a big challenge. The practical way to prevent sandwich attacks targeting auto-liquidity functions is to keep the transaction amount of numTokensSellToAddToLiquidity small. This will lead to small and frequent transactions. This will discourage an attacker to run large sandwich attacks on auto-liquidity transactions.

WTW-06 | Confuse Function Name `setMinSell`

Category	Severity	Location	Status
Language Specific	Informational	projects/WorthPad/WorthToken.sol (1): 642	Resolved

Description

Within this function `setMinSell`, `numTokensSellToAddToLiquidity` is set for the token amount for selling when add liquidity to swap pair. Function name is not consistent with its implementation.

Recommendation

Advise to rename this function.

Alleviation

[Worthpad Team]: Function Name and corresponding Emitter have been renamed.
Function name: setNumTokensSellToAddToLiquidity
Emitter name: SetNumTokensSellToAddToLiquidityEvent

WTW-07 | Return value not handled

Category	Severity	Location	Status
Volatile Code	Informational	projects/WorthPad/WorthToken.sol (1): 525	Acknowledged

Description

The return values of function `addLiquidityETH` are not properly handled.

```
525         uniswapV2Router.addLiquidityETH{value: ethAmount}(
526             address(this),
527             tokenAmount,
528             0, // slippage is unavoidable
529             0, // slippage is unavoidable
530             owner(),
531             block.timestamp
532         );
```

Recommendation

We recommend using variables to receive the return value of the functions mentioned above and handle both success and failure cases if needed by the business logic.

Alleviation

[Worthpad Team]: Refer to WTW-05. If we use small and frequent transactions for adding auto-liquidity, thenwe won't need the return values of addLiquidityETH in the contract's business logic.

WTW-08 | Missing Event Emitting

Category	Severity	Location	Status
Coding Style	Informational	projects/WorthPad/WorthToken.sol (1): 251	Pending

Description

The linked event is not emitted:

- MinTokensBeforeSwapUpdated

Recommendation

Recommend emitting events, for all the essential state variables that are possible to be changed during runtime.

WTW-09 | Unused Base Contract `ReentrancyGuard`

Category	Severity	Location	Status
Volatile Code	Informational	projects/WorthPad/WorthToken.sol (1): 218	Pending

Description

The contract `WorthToken` has a base contract `ReentrancyGuard`, but no variable or function from `ReentrancyGuard` is used or overrode.

Recommendation

Advise to check if `ReentrancyGuard` is required base of `WorthToken`.

Alleviation

[Worthpad Team]: `nonReentrant` modifiers is added for function transfer and transferFrom in commit: https://github.com/worthpad/worth/commit/fa3c79a0fc83b00881e5f85c24c2b9765b97d922

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.