PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Uniglo

23 August 2022

paladinsec.co

info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Paladin is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Paladin may, at its discretion, claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for Uniglo's token and vesting contracts on the Ethereum network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	Uniglo
URL	https://uniglo.io/
Network	Ethereum
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`UnigloERC20`	`0x87Fb5a2E712e3eE57607c32C43Ab177B0A234e0F`	MATCH
`UnigloVesting`	`0xD4E431db86c843F21C3bfD2Fcb8e97f10461fc42`	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	0	-	-	-
Medium	2	2	-	-
Low	2	1	-	1
Informational	14	5	1	8
Total	**18**	**8**	**1**	**9**

Classification of Issues

Severity	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 Global

ID	Severity	Summary	Status
01	INFO	Lack of events	ACKNOWLEDGED
02	INFO	Lack of indexing for event parameters	ACKNOWLEDGED

1.3.2 UnigloERC20

ID	Severity	Summary	Status
03	MEDIUM	Requirement for account set can be bypassed with `transferFrom`	✓ RESOLVED
04	MEDIUM	`swapAndTransfer` reenters into `transfer`, which causes the swap fees to be stuck in the contract as they are wiped without distribution	✓ RESOLVED
05	LOW	Precision error for fees causes `totalFee` to potentially be slightly higher than the actual total fees	✓ RESOLVED
06	LOW	Accounting in slippage factor	ACKNOWLEDGED
07	INFO	Validation missing for `swapAndLiquify`	✓ RESOLVED
08	INFO	Modularize logic: Same logic under transfer functions	ACKNOWLEDGED
09	INFO	Only owner can remove liquidity	✓ RESOLVED
10	INFO	Recipient excluded from fee	ACKNOWLEDGED
11	INFO	`increaseRouterAllowance` function is not required	✓ RESOLVED
12	INFO	`console` import not required	ACKNOWLEDGED
13	INFO	Unnecessary calls can be avoided	ACKNOWLEDGED
14	INFO	Typographical errors	ACKNOWLEDGED

1.3.3 UnigloVesting

ID	Severity	Summary	Status
15	INFO	`safeTransfer` should be used	✓ RESOLVED
16	INFO	Actual transfer of contribution tokens does not take place within contract	✓ RESOLVED
17	INFO	No zero length check for adding and removing contributors	ACKNOWLEDGED
18	INFO	Typographical errors	PARTIAL

2 Findings

2.1 Global Issues

The global issues within this section apply to the contracts within scope as a whole.

2.1.1 Issues & Recommendations

Issue #01	Lack of events
Severity	INFORMATIONAL
Description	Essential function calls should be captured with events. This allows for off-chain components to filter and track the activities
Recommendation	Add in events for key function calls.
Resolution	ACKNOWLEDGED

Issue #02	Lack of indexing for event parameters
Severity	INFORMATIONAL
Description	Essential identifying parameters within events should be marked as `indexed`. This allows for off-chain components to filter events only including these values.
Recommendation	Add indices to the key variables within the events you might want to filter on.
Resolution	ACKNOWLEDGED

2.1 UnigloERC20

`UnigloERC20` is an ERC20-based contract which allows the transfer of tokens with fees. The fees collected are used for marketing, treasury, liquidation and burn. Uniglo token is also added as a pair with ETH on Uniswap, where the owner can add in the liquidity. All the tokens are minted to the recipient address provided at the time of deployment.

The treasury and marketing address can withdraw the fees collected by calling the `trasnferAndSwap`. The tokens would be dependent on the gloShare which has been set under the contract.

The owner can call `swapAndLiquify` to add the liquidity to the Uniswap Pair. The tokens are first swapped for ETH tokens, and then the liquidity is added to the Pool.

2.1.1 Privileged Functions

- `setAccounts`
- `increaseRouterAllowance`
- `setFeeRates`
- `setGloRate`
- `excludeFromFee`
- `swapAndLiquify`
- `swapAndTransfer`
- `renounceOwnership`
- `transferOwnership`

2.1.2 Issues & Recommendations

Issue #03	Requirement for account set can be bypassed with `transferFrom`
Severity	MEDIUM SEVERITY
Location	Line 124~

```
function transferFrom(address from, address to, uint256
amount) public virtual override returns (bool) {
    require(feeRatesSet, "Error: Fees have not been set");

    address spender = _msgSender();

    _spendAllowance(from, spender, amount);

    if(!excludedFromFee[from]) {
        uint256 burnFee = amount * burnRate /
PERCENTAGE_FACTOR;
        uint256 marketingFee = amount * marketingRate /
PERCENTAGE_FACTOR;
        uint256 treasuryFee = amount * treasuryRate /
PERCENTAGE_FACTOR;
        uint256 liquidityFee = amount * liquidityRate /
PERCENTAGE_FACTOR;

        uint256 totalFee = amount * totalRate /
PERCENTAGE_FACTOR;

        marketingGloBalance += marketingFee;
        treasuryGloBalance += treasuryFee;
        liquidityBalance += liquidityFee;

        _transfer(from, to, amount - totalFee);
        _transfer(from, address(this), totalFee - burnFee);

        _burn(from, burnFee);
    } else {
        _transfer(from, to, amount);
    }

    return true;
}
```

Description	Only the `feeRatesSet` requirement is added to `transferFrom`. The `transfer` function includes the requirement for checking on fees and account settings. Users can bypass the provision of account set by using the `transferFrom` function.
Recommendation	Consider adding the in account set check to `transferFrom` as well in line with the `transfer` function: `require(accountsSet, "Error: Accounts have not been set");`
Resolution	RESOLVED The check for `Accounts` has been added under `transferFrom`.

Issue #04	**`swapAndTransfer` reenters into `transfer`, which causes the swap fees to be stuck in the contract as they are wiped without distribution**
Severity	MEDIUM SEVERITY
Location	Line 159
Description	With the current implementation, the checks-effects-interactions pattern is not adhered to within the `swapAndTransfer` function. The marketing and treasury balance is set to 0 at the end of the function, and this might cause issues on reentrancy coming from the `transfer` functions as those update both the balances. These increments are lost and stuck in the contract forever.
Recommendation	Consider following the check-effects-interactions pattern and move the balance set to the start of the function.
Resolution	RESOLVED The setting of market and treasury balance has been moved to above the `transfer` call.

Issue #05	**Precision error for fees causes `totalFee` to potentially be slightly higher than the actual total fees**
Severity	LOW SEVERITY

Location	Lines 102-107 and Lines 132-137
	`uint256 burnFee = amount * burnRate / PERCENTAGE_FACTOR;`
	`uint256 marketingFee = amount * marketingRate / PERCENTAGE_FACTOR;`
	`uint256 treasuryFee = amount * treasuryRate / PERCENTAGE_FACTOR;`
	`uint256 liquidityFee = amount * liquidityRate / PERCENTAGE_FACTOR;`
	
	`uint256 totalFee = amount * totalRate / PERCENTAGE_FACTOR;`

Description	`totalFee` can be calculated by summing the `burnFee`, `marketingFee`, `treasuryFee` and `liquidityFee`. This will avoid precision errors.
Recommendation	Consider summing the fees to calculate the `totalFee`.
Resolution	RESOLVED Logic has been added to get the `totalFee` by summing up the `burnFee`, `marketingFee`, `treasuryFee` and `liquidityFee`.

Issue #06	Accounting in slippage factor
Severity	LOW SEVERITY

Description	The contract contains a number of instances where slippage factor is not accounted for. We have consolidated these issues into a single issue in an effort to keep the report size reasonable. Adding in slippage factor would help protect from front run and price manipulation:
	
	Lines 177, 196, 220
	`swapExactTokensForETHSupportingFeeOnTransferTokens(ethShare, 0, path, caller, block.timestamp);`
	
	In the current implementation, `amountOutMin` is being passed as zero, thus manipulation is possible.
	
	Line 226
	`addLiquidityETH{value: newEthBalance}(address(this), otherHalf, 0, 0, owner(), block.timestamp);`
	
	In the current implementation, `amountTokenMin` and `amountETHMin` are being passed as zero, thus manipulation is possible.

Recommendation	Swap: Consider using `getAmountsOut` from Uniswap Router to get a probable value of amount out from the swap and adding a slippage factor with the same, under an accepted value. Liquidity: Before adding liquidity, call `getReserves` under the Pair contract to get the token reserves and pass to the `quote` function within the Uniswap router to get the right amount for adding liquidity. These calculations must strictly be made off-chain, as doing them on-chain has no effect as a frontrun transaction has already happened at that point. The minimum should therefore be passed into the functions and forwarded to the swap minimum parameters.
Resolution	ACKNOWLEDGED The client mentions that this may lead to revert transactions so this is kept as 0.

Issue #07	Validation missing for `swapAndLiquify`
Severity	INFORMATIONAL
Location	Line 206
Description	If the `liquidityBalance` is 0, then the function call is not required. This will avoid any unnecessary calls to the function.
Recommendation	Consider adding a check for `liquidityBalance`: `require(liquidityBalance > 0, "Error: No amount available");`
Resolution	RESOLVED A check for `liquidityBalance` has been added.

Issue #08	Modularize logic: Same logic under `transfer` functions
Severity	INFORMATIONAL
Location	Lines 95-152
Description	Both `transfer` and `transferFrom` function have the same logic to collect fees before the transfer. This logic can be extracted and defined only once to improve code quality and avoid repetitions.
Recommendation	Consider overriding the `_transfer` function and adding the logic for the fees within this function.
Resolution	ACKNOWLEDGED

Issue #09	Only owner can remove liquidity
Severity	INFORMATIONAL
Location	Line 226
Description	Within the function `swapAndLiquify`, the owner is the one who adds liquidity. If the owner is a contract, ensure that functionality is added to be able to transfer LP tokens and remove liquidity.
Recommendation	Consider adding functions for the owner to manage liquidity for the pair in case it is a contract. Consider carefully documenting that the liquidity is held by the project team
Resolution	RESOLVED The client has indicated they understand this. Users should still be careful as the owner can withdraw the liquidity at any point.

Issue #10	Recipient excluded from fee
Severity	INFORMATIONAL
Location	Line 95 and 124
Description	The recipient under the functions `transfer` and `transferFor` can also be checked for `excludedFromFee`.
Recommendation	Consider adding a check to verify if the recipients are excluded from the fee or not.
Resolution	ACKNOWLEDGED

Issue #11	`increaseRouterAllowance` function is not required
Severity	INFORMATIONAL

Location	Line 231
	`function increaseRouterAllowance(uint256 amount) external onlyOwner returns (bool) {`
	`    _approve(address(this), UNISWAP_ROUTER, allowance(address(this), UNISWAP_ROUTER) + amount);`
	`    return true;`
	`}`

Description	This function is not required as the Uniswap router is already approved for the maximum value under constructor.
Recommendation	Consider removing the function.
Resolution	RESOLVED The client has indicated they wish to retain this function.

Issue #12	**console import not required**
Severity	INFORMATIONAL
Location	Line 11 `import "hardhat/console.sol";`
Description	This import might have been used for testing purpose but was added to production on accident.
Recommendation	Consider removing the `import` statement as this is not required under the contract.
Resolution	ACKNOWLEDGED

Issue #13	**Unnecessary calls can be avoided**
Severity	INFORMATIONAL

Description	Line 114 and 144
	`_transfer(sender, address(this), totalFee - burnFee);`
	A check can be added to transfer only when `totalFee > 0`.
	
	Line 116 and 146
	`_burn(sender, burnFee);`
	A check can be added to burn only when `burnFee > 0`.

Recommendation	Consider adding the above if-clauses.
Resolution	ACKNOWLEDGED

Issue #14	Typographical errors
Severity	INFORMATIONAL
Description	We have consolidated several typographical errors into a single issue in an effort to keep the report size reasonable.

Line 16

```
address public constant UNISWAP_ROUTER =
0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D
```

This can be defined as `IUniswapV2Router02` to avoid casting throughout the contract.

Line 28

```
uint256 public glo_share; // Equals 7.5%
```

This comment is not in line with the code as the limitation of 7.5 is not added.

Line 30

```
address public immutable GLO_ETH_PAIR;
```

This can be defined as `IUniswapV2Pair` to avoid casting throughout the contract.

Line 54

```
_approve(address(this), UNISWAP_ROUTER, ~uint256(0));
```

`type(uint256).max` can be used instead of bitwise operation for approving max value. This will make the code cleaner.

Line 174 and 217

```
path[1] = IUniswapV2Router02(UNISWAP_ROUTER).WETH();
```

Value of WETH can be stored in an immutable variable to save gas cost.

	Line 177, 196, 220 and 226 `swapExactTokensForETHSupportingFeeOnTransferTokens(.., .., .., .., block.timestamp);` Using `type(uint256).max` for `timestamp` would help to reduce the gas cost.
Recommendation	Consider fixing the typographical errors.
Resolution	ACKNOWLEDGED

2.1 UnigloVesting

`UnigloVesting` provides the functionality for the contributors to claim GLO tokens as per the tokens accrued with respect to the vesting time.

The owner can add and remove contributors before initializing the vesting. The contributors can claim the accrued GLO tokens from the contract. Vesting is a linear-style vesting for a 30 days period. The accrued GLO tokens are calculated based on the number of days elapsed under the total vesting period.

Under the current implementation, the owner adds in the contributor addresses and amounts. There is no actual transfer of tokens from the contributors.

2.1.1 Privileged Functions

- `addContributors`
- `removeContributors`
- `initializeVesting`
- `renounceOwnership`
- `transferOwnership`

2.1.2 Issues & Recommendations

Issue #15	`safeTransfer` should be used
Severity	INFORMATIONAL
Location	Line 76 `IERC20(GLO).transfer(caller, amountToClaim);`
Description	Instead of using `transfer`, the code can be modified to use `safeTransfer` for handling potential tokens which would not return a boolean or return false. This issue has been marked as informational since GLO is not such a token at the moment and this issue therefore does not affect users.
Recommendation	Consider using `safeTransfer`.
Resolution	RESOLVED The `safeTransfer` call has been added.

Issue #16	Actual transfer of contribution tokens does not take place within contract
Severity	INFORMATIONAL
Description	Under the current implementation, the owner adds in the contributor addresses and amounts. There is no actual transfer of tokens from the contributors. This issue has been raised solely to inform the team of this behavior. This should not affect users as it is most likely desired.
Recommendation	Do check in the logic or functionality for the actual transfer of tokens to the vesting contract.
Resolution	RESOLVED The client understands this and the transfer of tokens will take place from the contributors only.

Issue #17	No zero length check for adding and removing contributors
Severity	INFORMATIONAL
Location	Lines 34 and 47
Description	There is no zero length check present for the arrays under the functions `addContributors` and `removeContributors`.
Recommendation	Consider adding a zero length check for the contributors, to avoid unnecessary calls.
Resolution	ACKNOWLEDGED

Issue #18	Typographical errors
Severity	INFORMATIONAL

Description	We have consolidated several typographical errors into a single issue in an effort to keep the report size reasonable.
	
	Line 14
	`address public immutable GLO;`
	
	This can be defined as `IERC20` to avoid casting throughout the contract.
	
	Line 16
	`uint256 constant public CHECKPOINT = 1 minutes; // Should be set to 1 days.`
	
	The comment and value assigned are mismatched.
	
	Line 25
	`constructor(address _glo)`
	
	This can be defined as `IERC20` to avoid casting through out the contract.
	
	Line 39
	`for(uint256 i = 0; i < accounts.length; i++)`
	
	`accounts.length` can be added to a local variable to save gas.
	
	Line 68
	`function claim() public`
	
	This function can be declared as `external`.

Recommendation	Consider fixing the typographical errors.
Resolution	PARTIALLY RESOLVED `CHECKPOINT` has been changed to days and the `claim` function has been made `external`.


PALADIN
BLOCKCHAIN SECURITY