TourUniverse Smart Contract Final Audit Report

Project Synopsis

Project Name	**TourUniverse**
Platform	Binance Smart Chain, Solidity
Github Repo	https://bscscan.com/token/0xF3dcCb92A98D0196a270FbA7a1D0125c89313e9b
Deployed Contract	Yes
Total Duration	2 Days
Timeline of Audit	**12th June 2021 to 13th June 2021**

Contract Details

Total Contract(s)	1
Name of Contract(s)	TourUniverse
Language	Solidity
Commit Hash	**Null**

Contract Vulnerabilities Synopsis

Issues	Open Issues	Closed Issues
High Severity	0	0
Medium Severity	3	1
Low Severity	0	3
Information	5	0
Total Found	8	4

Detailed Results

The contract has gone through several stages of the audit procedure that includes structural analysis, automated testing, manual code review etc.

All the issues have been explained and discussed in detail below. Along with the explanation of the issue found during the audit, the recommended way to overcome the issue or improve the code quality has also been mentioned.

A. Contract Name: TourUniverse

Medium Severity Issues

A.1 Loops are extremely costly

Line no -955, 1047

Status: Not Considered

Description:

The **TourUniverse** contract has some **for loops** in the contract that include state variables like .length of a non-memory array, in the condition of the for loops.

As a result, these state variables consume a lot more extra gas for every iteration of the for loop.

The following function includes such loops at the above-mentioned lines:

- ***includeInReward***
- ***_getCurrentSupply***

```
1047        for (uint256 i = 0; i < _excluded.length; i++) {
1048            if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) re
1049            rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1050            tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1051        }
```

Recommendation:

It's quite effective to use a local variable instead of a state variable like .length in a loop. This will be a significant step in optimizing gas usage.

For instance,

```
local_variable = excluded.length;
 for (uint256 index = 0; index < local_variable; index++) {
    if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
        rSupply = rSupply.sub(_rOwned[_excluded[i]]);
        tSupply = tSupply.sub(_tOwned[_excluded[i]]);
    }
}
```

A.2 Violation of Check_Effects_Interaction Pattern in the Withdraw function

Line no - 837-856

Status: Not Considered

Description:

As per the current design of the **TourUniverse** contract, the constructor of the contract includes an external call. However, this external call is made before updating the imperative state variable of the contract.
This approach violates the Check-Effects Interaction pattern.

```
844         IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
845          // Create a uniswap pair for this new token
846         uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
847             .createPair(address(this), _uniswapV2Router.WETH());
848 
849         // set the rest of the contract variables
850         uniswapV2Router = _uniswapV2Router;
851         //exclude owner and this contract from fee
852         _isExcludedFromFee[owner()] = true;
853         _isExcludedFromFee[address(this)] = true;
```

Therefore, as per the Solidity Guidelines, any modification of the state variables in the base contract must be performed before executing the external call.

The following function in the contract updates the state variables after making an external call at the lines mentioned below:

- ***constructor***

Recommendation:

Check Effects Interaction Pattern must be followed while implementing external calls in a function.

A.3 Absence of Input Validations

Line no - 985, 989, 993

Status: Intentional

Description:

The **TourUniverse** contract includes quite a few functions that update some crucial state variables of the contract.

However, no input validation is included in any of those functions. This might lead to an unwanted scenario where an invalid or wrong argument is passed to the function that could badly affect the expected behavior of the contract.

```
985     function setTaxFeePercent(uint256 taxFee) external onlyOwner
986         _taxFee = taxFee;
987     }
988
989     function setCharityFeePercent(uint256 charityFee) external
990         _charityFee = charityFee;
991     }
992
993     function setLiquidityFeePercent(uint256 liquidityFee) exter
994         _liquidityFee = liquidityFee;
995     }
```

Recommendation:
Arguments passed to such functions must be validated before being used to update the State variable.

A.4 No Events emitted after imperative State Variable modification

Line no -985, 989, 993

Status: Not Considered

Description:
Functions that update an imperative arithmetic state variable contract should emit an event after the updation.
The following functions modify some crucial arithmetic parameters like ***_taxFee,*** ***_charityFee*** etc in the **TourUniverse** contract but don't emit any event after that:

- ***setTaxFeePercent()***
- ***setCharityFeePercent()***
- ***setLiquidityFeePercent()***

Since there is no event emitted on updating these variables, it might be difficult to track it off chain.

***Recommendation*:**
An event should be fired after changing crucial arithmetic state variables.

Low Severity Issues

A.5 Absence of Zero Address Validation

Status: Intentional

Description:

The **TourUniverse** contract includes quite a few functions that updates some of the imperative addresses in the contract like *uniswapV2Pair* **etc**.
However, during the automated testing of the contact it was found that no Zero Address Validation is implemented on the following functions while updating the address state variables of the contract:

- ***setUniswapPair***

***Recommendation*:**

A **require** statement should be included in such functions to ensure no zero address is passed in the arguments.

A.6 External Visibility should be preferred

Status: Intentional

Description:

Those functions that are never called throughout the contract should be marked as ***external*** visibility instead of ***public*** visibility.
This will effectively result in Gas Optimization as well.

Therefore, the following function must be marked as **external** within the contract:

- ***isExcludedFromReward()***
- ***totalFees()***
- ***deliver()***
- ***reflectionFromToken()***
- ***excludeFromReward()***
- ***excludeFromFee()***
- ***includeInFee()***
- ***setSwapAndLiquifyEnabled()***
- ***isExcludedFromFee()***

Recommendation:

If the PUBLIC visibility of the above-mentioned functions is not intended, then the EXTERNAL Visibility keyword should be preferred.

A.7 Contract includes Hardcoded Addresses

Status: Intentional

Line no - 451-457

Description:

Keeping in mind the immutable nature of smart contracts, it is not considered a better practise to hardcode any address in the contract before deployment.
However, the contract does include some hardcoded addresses in the above-mentioned lines.

```
794
795     address private _charityWalletAddress = 0x1aeb122935fAEFC224f9AA70553601D537753695;
796
```

Recommendation:

of including hardcoded addresses in the contract, it would be an effective approach to initialize those addresses within the constructors at the time of deployment.

Informational

A.8 Constant declaration should be preferred

Line no - 795,798,802,803,804

Status: Not Considered

Description:

State variables that are not supposed to change throughout the contract should be declared as **constant**.

Recommendation:

The following state variables need to be declared as **constant**, unless the current contract design is intended.

- **_charityWalletAddress** at Line 795
- **_decimals** at Line 804
- **_name** at Line 802
- **_symbol** at Line 803
- **_tTotal** at Line 798

A.9 Too many Digits used

Status: Not Considered

Line no - 798,820,821

Description:

The above-mentioned lines have a large number of digits that reduces the readability of the code.

- ***_tTotal*** *at Line 798*
- ***_maxTxAmount*** *at Line 820*
- ***numTokensSellToAddToLiquidity*** *at Line 821*

Recommendation:

Ether Suffix could should be used to symbolize the 10^18 zeros.

A.10 Coding Style Issues in the Contract

Status: Not Considered

Explanation:

Code readability of a Smart Contract is largely influenced by the Coding Style issues and in some specific scenarios may lead to bugs in the future.

```
Parameter Tourniverse.setSwapAndLiquifyEnabled(bool)._enabled (contracts/TRNI.sol#1003) is not in mixedCase
Parameter Tourniverse.calculateTaxFee(uint256)._amount (contracts/TRNI.sol#1072) is not in mixedCase
Parameter Tourniverse.calculateCharityFee(uint256)._amount (contracts/TRNI.sol#1078) is not in mixedCase
Parameter Tourniverse.calculateLiquidityFee(uint256)._amount (contracts/TRNI.sol#1084) is not in mixedCase
Variable Tourniverse._taxFee (contracts/TRNI.sol#806) is not in mixedCase
Variable Tourniverse._charityFee (contracts/TRNI.sol#809) is not in mixedCase
Variable Tourniverse._liquidityFee (contracts/TRNI.sol#811) is not in mixedCase
Variable Tourniverse._maxTxAmount (contracts/TRNI.sol#820) is not in mixedCase
```

During the automated testing, it was found that the **TourUniverse** contract had quite a few code style issues.

Recommendation:

Therefore, it is highly recommended to fix the issues like naming convention, indentation, and code layout issues in a smart contract.

A.11 NatSpec Annotations must be included

Status: Not Considered

Description:
The smart contracts do not include the NatSpec annotations adequately.

Recommendation:
Cover by NatSpec all Contract methods.

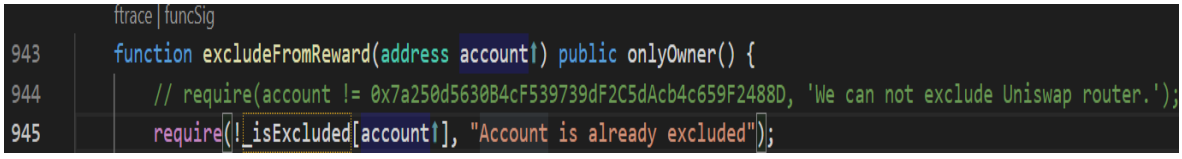
A.12 Commented codes must be wiped-out before deployment

Line no - 944

Status: Not Considered

Description:
The **TourUniverse** contract includes quite a few commented codes in the **excludeFromReward .**

This badly affects the readability of the code



Recommendation:
If these instances of code are not required in the current version of the contract, then the commented codes must be removed before deployment.

Automated Test Results

```
deliver(uint256) should be declared external:
        - Tourniverse.deliver(uint256) (contracts/TRNI.sol#917-924)
reflectionFromToken(uint256,bool) should be declared external:
        - Tourniverse.reflectionFromToken(uint256,bool) (contracts/TRNI.sol#926-935)
excludeFromReward(address) should be declared external:
        - Tourniverse.excludeFromReward(address) (contracts/TRNI.sol#943-951)
excludeFromFee(address) should be declared external:
        - Tourniverse.excludeFromFee(address) (contracts/TRNI.sol#977-979)
includeInFee(address) should be declared external:
        - Tourniverse.includeInFee(address) (contracts/TRNI.sol#981-983)
setSwapAndLiquifyEnabled(bool) should be declared external:
        - Tourniverse.setSwapAndLiquifyEnabled(bool) (contracts/TRNI.sol#1003-1006)
isExcludedFromFee(address) should be declared external:
```

```
Tourniverse.allowance(address,address).owner (contracts/TRNI.sol#884) shadows:
        - Ownable.owner() (contracts/TRNI.sol#545-547) (function)
Tourniverse._approve(address,address,uint256).owner (contracts/TRNI.sol#1113) shadows:
        - Ownable.owner() (contracts/TRNI.sol#545-547) (function)
```

```
Tourniverse._rTotal (contracts/TRNI.sol#799) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
Tourniverse._previousTaxFee (contracts/TRNI.sol#807) is set pre-construction with a non-constant function or state variable:
        - _taxFee
Tourniverse._previousCharityFee (contracts/TRNI.sol#810) is set pre-construction with a non-constant function or state variable:
        - _charityFee
Tourniverse._previousLiquidityFee (contracts/TRNI.sol#812) is set pre-construction with a non-constant function or state variable:
        - _liquidityFee
```

```
Tourniverse._charityWalletAddress (contracts/TRNI.sol#795) should be constant
Tourniverse._decimals (contracts/TRNI.sol#804) should be constant
Tourniverse._name (contracts/TRNI.sol#802) should be constant
Tourniverse._symbol (contracts/TRNI.sol#803) should be constant
Tourniverse._tTotal (contracts/TRNI.sol#798) should be constant
```