Universe Token

Smart Contract Audit Report

SYSFIXED

28 Jul 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Universe Token	UVT	Binance Smart Chain

Addresses

Contract address	0x196eb1d21c05cc265ea0a1479e924e7983467838
Contract deployer address	0x656cBB61B2EC36470DaddF7349BC6b232962B875

Project Website

https://www.uvtoken.com/

Codebase

https://bscscan.com/address/0x196eb1d21c05cc265ea0a1479e924e7983467838#code

SUMMARY

UVTOKEN is committed to providing a safe, convenient, and efficient decentralized digital asset platform for the public through its unique technology, allowing everyone to use their digital assets anytime, anywhere, with confidence and convenience, and enriching the application scenarios of blockchain technology and massive digital assets to promote business progress and social development.

Contract Summary

Documentation Quality

Universe Token provides a very good documentation with standard of solidity base code.

- The technical description is provided clearly and structured and also dont have any high risk issue.

Code Quality

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by Universe Token with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-100 SWC-108 | Explicitly define visibility for all state variables on lines 1146, 1147 and 1164.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 151, 165, 180, 181, 194, 206, 221, 235, 249, 263, 279, 302, 325, 351, 564, 587, 620, 622, 643, 644, 669, 671, 720, 1126, 1126, 1150, 1150, 1280, 1286, 1292, 1298, 1304, 1486 and 1524.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 19.
- SWC-110 SWC-123 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 1140, 1141, 1359, 1360, 1487, 1490, 1525 and 1527.

CONCLUSION

We have audited the Universe Token project released on July 2022 to discover issues and identify potential security vulnerabilities in Universe Token Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

The security audit report provides satisfactory results with low-risk issues.

The Universe Token smart contract code issues do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are arithmetic operation issues, a state variable visibility is not set, a public state variable with array type causing reachable exception by default, and out-of-bounds array access in which the index access expression can cause an exception in case an invalid array index value is used. The current pragma Solidity directive is ""^0.8.0"". Specifying a fixed compiler version is recommended to ensure that the bytecode produced does not vary between builds. It is best practice to set the visibility of state variables explicitly. The default visibility for "ischargetransactionfee" is internal. Other possible visibility settings are public and private.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	PASS
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

Typographical Error	SWC-129	A typographical error can occur for example when the intent of a defined operation is to sum a number to a variable.	PASS
Override control character	SWC-130	Malicious actors can use the Right-To-Left-Override unicode character to force RTL text rendering and confuse users as to the real intent of a contract.	PASS
Unused variables	SWC-131 SWC-135	Unused variables are allowed in Solidity and they do not pose a direct security issue.	PASS
Unexpected Ether balance	SWC-132	Contracts can behave erroneously when they strictly assume a specific Ether balance.	PASS
Hash Collisions Variable	SWC-133	Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The transfer() and send() functions forward a fixed amount of 2300 gas.	PASS
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	PASS

SMART CONTRACT ANALYSIS

Started	Wednesday Jul 27 2022 16:20:20 GMT+0000 (Coordinated Universal Time)
Finished	Thursday Jul 28 2022 09:40:22 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	UvtToken.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.	low	acknowledged
SWC-110	PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 151

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
150   unchecked {
151   uint256 c = a + b;
152   if (c < a) return (false, 0);
153   return (true, c);
154   }
155
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 165

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
164   if (b > a) return (false, 0);
165   return (true, a - b);
166   }
167   }
168
169
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 180

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
179   if (a == 0) return (true, 0);
180   uint256 c = a * b;
181   if (c / a != b) return (false, 0);
182   return (true, c);
183   }
184
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 181

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
180   uint256 c = a * b;
181   if (c / a != b) return (false, 0);
182   return (true, c);
183   }
184   }
185
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 194

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
193   if (b == 0) return (false, 0);
194   return (true, a / b);
195   }
196   }
197
198
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 206

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
205   if (b == 0) return (false, 0);
206   return (true, a % b);
207   }
208   }
209
210
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 221

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
220   function add(uint256 a, uint256 b) internal pure returns (uint256) {
221   return a + b;
222   }
223
224   /**
225
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 235

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
234   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
235   return a - b;
236   }
237
238   /**
239
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 249

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
248   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
249   return a * b;
250   }
251
252   /**
253
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 263

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
262   function div(uint256 a, uint256 b) internal pure returns (uint256) {
263   return a / b;
264   }
265
266   /**
267
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 279

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
278   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
279   return a % b;
280   }
281
282   /**
283
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 302

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
301   require(b <= a, errorMessage);
302   return a - b;
303   }
304   }
305
306
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 325

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
324   require(b > 0, errorMessage);
325   return a / b;
326   }
327   }
328
329
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 351

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
350   require(b > 0, errorMessage);
351   return a % b;
352   }
353   }
354   }
355
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 564

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
563   address owner = _msgSender();
564   _approve(owner, spender, allowance(owner, spender) + addedValue);
565   return true;
566   }
567
568
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 587

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
586   unchecked {
587   _approve(owner, spender, currentAllowance - subtractedValue);
588   }
589
590   return true;
591
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 620

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
619   unchecked {
620   _balances[from] = fromBalance - amount;
621   }
622   _balances[to] += amount;
623
624
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 622

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
621   }
622   _balances[to] += amount;
623
624   emit Transfer(from, to, amount);
625
626
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 643

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
642
643   _totalSupply += amount;
644   _balances[account] += amount;
645   emit Transfer(address(0), account, amount);
646
647
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 644

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
643   _totalSupply += amount;
644   _balances[account] += amount;
645   emit Transfer(address(0), account, amount);
646
647   _afterTokenTransfer(address(0), account, amount);
648
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 669

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
668   unchecked {
669   _balances[account] = accountBalance - amount;
670   }
671   _totalSupply -= amount;
672
673
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 671

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
670   }
671   _totalSupply -= amount;
672
673   emit Transfer(account, address(0), amount);
674
675
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 720

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
719   unchecked {
720   _approve(owner, spender, currentAllowance - amount);
721   }
722   }
723   }
724
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1126

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
1125
1126   uint256 private _tTotal = 2000000000 * 10**18;
1127
1128   uint256 public _lpFee = 3;
1129   uint256 private _previousLpFee = _lpFee;
1130
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1126

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
1125
1126   uint256 private _tTotal = 2000000000 * 10**18;
1127
1128   uint256 public _lpFee = 3;
1129   uint256 private _previousLpFee = _lpFee;
1130
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1150

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
1149
1150   uint256 private numTokensSellToAddToLiquidity = 5 * 10**18;
1151   uint256[] private nftIds_;
1152   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
1153   event SwapAndLiquifyEnabledUpdated(bool enabled);
1154
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1150

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
1149
1150   uint256 private numTokensSellToAddToLiquidity = 5 * 10**18;
1151   uint256[] private nftIds_;
1152   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
1153   event SwapAndLiquifyEnabledUpdated(bool enabled);
1154
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1280

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
1279   function getTransferAmount(uint256 _amount) private view returns (uint256) {
1280   uint256 allFree =
_amount.mul(_lpFee.add(_liquidityFee).add(_bulkFee).add(_nftFee)).div(10**2);
1281   return _amount.sub(allFree);
1282   }
1283
1284
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1286

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
1285   return _amount.mul(_lpFee).div(
1286   10**2
1287   );
1288   }
1289
1290
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1292

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
1291   return _amount.mul(_liquidityFee).div(
1292   10**2
1293   );
1294   }
1295
1296
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1298

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
1297   return _amount.mul(_bulkFee).div(
1298   10**2
1299   );
1300   }
1301
1302
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1304

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
1303   return _amount.mul(_nftFee).div(
1304   10**2
1305   );
1306   }
1307
1308
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1486

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
1485   uint total = getLpTotoalSUpply();
1486   for(uint256 i = 0; i < _lpUsers.length; i++){
1487   uint usertotal = getuserLpSUpply(_lpUsers[i]);
1488   uint value_ =_value.mul(usertotal).div(total);
1489   if(usertotal >0){
1490
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1524

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- UvtToken.sol

Locations

```
1523   require(_value > 0,"The transfer amount cannot be 0");
1524   for(uint256 i = 0; i < _nftUsers.length; i++){
1525   uint256 freevalue =getNftFree(_nftUsers[i],tokenId_ ,_value);
1526   if(freevalue >0){
1527   _transfer(sender_, _nftUsers[i], freevalue);
1528
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 19

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- UvtToken.sol

Locations

```
18
19   pragma solidity ^0.8.0;
20
21   /**
22   * @dev Interface of the ERC20 standard as defined in the EIP.
23
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1146

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "inSwapAndLiquify" is internal. Other possible visibility settings are public and private.

Source File

- UvtToken.sol

Locations

```
1145   IPancakePair public  _iPancakePair;
1146   bool inSwapAndLiquify;
1147   bool ischargetransactionfee = true;
1148   bool public swapAndLiquifyEnabled = true;
1149
1150
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1147

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "ischargetransactionfee" is internal. Other possible visibility settings are public and private.

Source File

- UvtToken.sol

Locations

```
1146   bool inSwapAndLiquify;
1147   bool ischargetransactionfee = true;
1148   bool public swapAndLiquifyEnabled = true;
1149
1150   uint256 private numTokensSellToAddToLiquidity = 5 * 10**18;
1151
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1164

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "NftconTract_" is internal. Other possible visibility settings are public and private.

Source File

- UvtToken.sol

Locations

```
1163   event Ischargetransactionfee(bool _enabled);
1164   UvtnftToken NftconTract_;
1165
1166   uint256 public TokenIdfor1_ = 2022070701;
1167   uint256 public TokenIdfor2_ = 2022070702;
1168
```

SWC-110 | PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.

LINE 1140

low SEVERITY

The public state variable "_nftUsers" in "UvtToken" contract has type "address[]" and can cause an exception in case of use of invalid array index value.

Source File

- UvtToken.sol

Locations

```
1139
1140   address[] public _nftUsers;
1141   address[] public _lpUsers;
1142   mapping (address => bool) private _isExcludedLpUsers;
1143   IPancakeRouter02 public  uniswapV2Router;
1144
```

SWC-110 | PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.

LINE 1141

low SEVERITY

The public state variable "_lpUsers" in "UvtToken" contract has type "address[]" and can cause an exception in case of use of invalid array index value.

Source File

- UvtToken.sol

Locations

```
1140   address[] public _nftUsers;
1141   address[] public _lpUsers;
1142   mapping (address => bool) private _isExcludedLpUsers;
1143   IPancakeRouter02 public  uniswapV2Router;
1144   address public  uniswapV2Pair;
1145
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1359

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- UvtToken.sol

Locations

```
1358   address[] memory path = new address[](2);
1359   path[0] = address(this);
1360   path[1] = uniswapV2Router.WETH();
1361
1362   _approve(address(this), address(uniswapV2Router), tokenAmount);
1363
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1360

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- UvtToken.sol

Locations

```
1359   path[0] = address(this);
1360   path[1] = uniswapV2Router.WETH();
1361
1362   _approve(address(this), address(uniswapV2Router), tokenAmount);
1363
1364
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1487

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- UvtToken.sol

Locations

```
1486   for(uint256 i = 0; i < _lpUsers.length; i++){
1487   uint usertotal = getuserLpSUpply(_lpUsers[i]);
1488   uint value_ =_value.mul(usertotal).div(total);
1489   if(usertotal >0){
1490   _transfer(sender_, _lpUsers[i], value_);
1491
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1490

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- UvtToken.sol

Locations

```
1489   if(usertotal >0){
1490   _transfer(sender_, _lpUsers[i], value_);
1491   alluserBalance.add(usertotal);
1492   }
1493   }
1494
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1525

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- UvtToken.sol

Locations

```
1524   for(uint256 i = 0; i < _nftUsers.length; i++){
1525   uint256 freevalue =getNftFree(_nftUsers[i],tokenId_ ,_value);
1526   if(freevalue >0){
1527   _transfer(sender_, _nftUsers[i], freevalue);
1528   }
1529
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1527

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- UvtToken.sol

Locations

```
1526   if(freevalue >0){
1527   _transfer(sender_, _nftUsers[i], freevalue);
1528   }
1529   }
1530   emit TransferArray(_value ,sender_ ,tokenId_);
1531
```

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 Sysfixed'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 Sysfixed 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 is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

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.

This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.