META-UTOPIA LAND

Smart Contract Audit Report

SYSFIXED

03 Mar 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
META-UTOPIA LAND	LAND2	Binance Smart Chain

Addresses

Contract address	0x9131066022b909c65edd1aaf7ff213dacf4e86d0
Contract deployer address	0x3D2DB0847cd0f04597f026402E24A8a029b40175

Project Website

https://www.meta-utopia.io/

Codebase

https://bscscan.com/address/0x9131066022b909c65edd1aaf7ff213dacf4e86d0#code

SUMMARY

The idea of building a perfect island, a utopia, would be difficult to imagine if it was not for MetaFI. MetaFI is short for Metaverse-Finance, which describes finance integration in the Metaverse. When blockchain gave birth to cryptocurrency, it was a sign that it was possible to fractionalize and financialize in the Metaverse. To realize the idea of Meta-Utopia, we must understand how MetaFI can connect the island, the cities, and its citizens. A free economy must drive a free society, a creator and contributor system that rewards pretty. The topics in this Gitbook are keys to building this system. EVERYONE wants to be part of the Metaverse in the future. Blockchain provides the trust and transparency that is needed. With the introduction of Web 3.0, the Semantic Web, and natural life infrastructures, technology now exists to make connectivity possible. The potential of the Power of the Community will be applied in a decentralized manner. Governance will be conducted through a DAO or Distributed Autonomous Organization, and its economy will run on DeFI Protocols. With these rules in place, we can now discuss creating a system and building a Utopia in the Metaverse.

Contract Summary

Documentation Quality

META-UTOPIA LAND 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 META-UTOPIA LAND 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 1039, 1044, 1058 and 1059.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 356, 386, 422, 424, 445, 446, 471, 473, 525, 1052, 1053, 1054, 1055, 1056, 1063, 1075, 1075, 1197, 1197, 1203, 1205, 1206, 1208, 1208, 1208, 1231, 1231, 1232, 1232, 1238, 1238, 1241, 1241, 1251, 1251 and 1253.

- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 10, 101, 129, 155, 575, 655, 689, 852, 903, 1016 and 1035.
- 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 1050 and 1206.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 1130.

CONCLUSION

We have audited the META-UTOPIA LAND project released on March 2022 to discover issues and identify potential security vulnerabilities in META-UTOPIA LAND 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 META-UTOPIA LAND 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 some arithmetic operation issues, a floating pragma is set, 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 which the index access expression can cause an exception in case of the use of an invalid array index value. 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. State variable visibility is not set, It is best practice to set the visibility of state variables explicitly. The default visibility for "_stakeAddress" is internal. Other possible visibility settings are public and private. Use of "tx.origin" as a part of authorization control, The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

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.	ISSUE FOUND
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 Mar 02 2022 05:10:33 GMT+0000 (Coordinated Universal Time)
Finished	Thursday Mar 03 2022 11:59:39 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	LAND2.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-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	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-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	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-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 356

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
355   address owner = _msgSender();
356   _approve(owner, spender, _allowances[owner][spender] + addedValue);
357   return true;
358   }
359
360
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 386

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
385   unchecked {
386   _approve(owner, spender, currentAllowance - subtractedValue);
387   }
388
389   return true;
390
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 422

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
421   unchecked {
422   _balances[from] = fromBalance - amount;
423   }
424   _balances[to] += amount;
425
426
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 424

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
423   }
424   _balances[to] += amount;
425
426   emit Transfer(from, to, amount);
427
428
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 445

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
444
445   _totalSupply += amount;
446   _balances[account] += amount;
447   emit Transfer(address(0), account, amount);
448
449
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 446

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
445   _totalSupply += amount;
446   _balances[account] += amount;
447   emit Transfer(address(0), account, amount);
448
449   _afterTokenTransfer(address(0), account, amount);
450
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 471

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
470   unchecked {
471   _balances[account] = accountBalance - amount;
472   }
473   _totalSupply -= amount;
474
475
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 473

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
472   }
473   _totalSupply -= amount;
474
475   emit Transfer(account, address(0), amount);
476
477
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 525

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
524   unchecked {
525   _approve(owner, spender, currentAllowance - amount);
526   }
527   }
528   }
529
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1052

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1051   0,
1052   1000 * 1e18,
1053   1300 * 1e18,
1054   1500 * 1e18,
1055   1800 * 1e18,
1056
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1053

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1052   1000 * 1e18,
1053   1300 * 1e18,
1054   1500 * 1e18,
1055   1800 * 1e18,
1056   2000 * 1e18
1057
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1054

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1053   1300 * 1e18,
1054   1500 * 1e18,
1055   1800 * 1e18,
1056   2000 * 1e18
1057   ];
1058
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1055

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1054   1500 * 1e18,
1055   1800 * 1e18,
1056   2000 * 1e18
1057   ];
1058   IPancakeRouter02 uniswapV2Router02;
1059
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1056

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1055   1800 * 1e18,
1056   2000 * 1e18
1057   ];
1058   IPancakeRouter02 uniswapV2Router02;
1059   IPancakePair uniswapV2Pair;
1060
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1063

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1062   constructor(address swapV2Router, address BUSD) ERC20("META-UTOPIA LAND", "LAND")
{
1063   _mint(msg.sender, 2100000 * 1e18);
1064   _swapV2Router = swapV2Router;
1065   _BUSD = BUSD;
1066   uniswapV2Router02 = IPancakeRouter02(_swapV2Router);
1067
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1075

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1074
1075   _startTime = block.timestamp - (block.timestamp % 86400);
1076   }
1077
1078   function setTokenKeepingAddress(address tokenKeepingAddress)
1079
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1075

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1074
1075   _startTime = block.timestamp - (block.timestamp % 86400);
1076   }
1077
1078   function setTokenKeepingAddress(address tokenKeepingAddress)
1079
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1197

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1196   _isNoProtected = (block.timestamp >
1197   (_startTime + _proctectedDayAmount.length * 86400));
1198
1199   if (_isNoProtected) {
1200   return true;
1201
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1197

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1196   _isNoProtected = (block.timestamp >
1197   (_startTime + _proctectedDayAmount.length * 86400));
1198
1199   if (_isNoProtected) {
1200   return true;
1201
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1203

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1202
1203   _addressBuyAmount5[buyer] = _addressBuyAmount5[buyer] + amount;
1204   uint256 canBuyTotalAmount = 0;
1205   for (uint256 index = 0; index < _proctectedDayAmount.length; index++) {
1206   canBuyTotalAmount = canBuyTotalAmount + _proctectedDayAmount[index];
1207
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1205

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1204   uint256 canBuyTotalAmount = 0;
1205   for (uint256 index = 0; index < _proctectedDayAmount.length; index++) {
1206   canBuyTotalAmount = canBuyTotalAmount + _proctectedDayAmount[index];
1207   if (
1208   (_startTime + (index + 1) * 86400) > block.timestamp &&
1209
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1206

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1205   for (uint256 index = 0; index < _proctectedDayAmount.length; index++) {
1206   canBuyTotalAmount = canBuyTotalAmount + _proctectedDayAmount[index];
1207   if (
1208   (_startTime + (index + 1) * 86400) > block.timestamp &&
1209   _addressBuyAmount5[buyer] > canBuyTotalAmount
1210
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1208

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1207   if (
1208   (_startTime + (index + 1) * 86400) > block.timestamp &&
1209   _addressBuyAmount5[buyer] > canBuyTotalAmount
1210   ) {
1211   return false;
1212
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1208

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1207   if (
1208   (_startTime + (index + 1) * 86400) > block.timestamp &&
1209   _addressBuyAmount5[buyer] > canBuyTotalAmount
1210   ) {
1211   return false;
1212
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1208

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1207   if (
1208   (_startTime + (index + 1) * 86400) > block.timestamp &&
1209   _addressBuyAmount5[buyer] > canBuyTotalAmount
1210   ) {
1211   return false;
1212
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1231

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1230   );
1231   uint256 burnAmount = (amount * 3) / 100;
1232   uint256 stakeAmount = (amount * 2) / 100;
1233   super._burn(sender, burnAmount);
1234   super._transfer(sender, _tokenKeepingAddress, stakeAmount);
1235
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1231

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1230   );
1231   uint256 burnAmount = (amount * 3) / 100;
1232   uint256 stakeAmount = (amount * 2) / 100;
1233   super._burn(sender, burnAmount);
1234   super._transfer(sender, _tokenKeepingAddress, stakeAmount);
1235
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1232

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1231   uint256 burnAmount = (amount * 3) / 100;
1232   uint256 stakeAmount = (amount * 2) / 100;
1233   super._burn(sender, burnAmount);
1234   super._transfer(sender, _tokenKeepingAddress, stakeAmount);
1235   super._transfer(
1236
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1232

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1231   uint256 burnAmount = (amount * 3) / 100;
1232   uint256 stakeAmount = (amount * 2) / 100;
1233   super._burn(sender, burnAmount);
1234   super._transfer(sender, _tokenKeepingAddress, stakeAmount);
1235   super._transfer(
1236
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1238

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1237   recipient,
1238   amount - burnAmount - stakeAmount
1239   );
1240   } else {
1241   uint256 balance99 = (balanceOf(sender) * 9999999999) / 1e10;
1242
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1238

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1237   recipient,
1238   amount - burnAmount - stakeAmount
1239   );
1240   } else {
1241   uint256 balance99 = (balanceOf(sender) * 9999999999) / 1e10;
1242
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1241

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1240   } else {
1241   uint256 balance99 = (balanceOf(sender) * 9999999999) / 1e10;
1242
1243   if (amount > balance99) {
1244   amount = balance99;
1245
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1241

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1240   } else {
1241   uint256 balance99 = (balanceOf(sender) * 9999999999) / 1e10;
1242
1243   if (amount > balance99) {
1244   amount = balance99;
1245
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1251

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1250
1251   uint256 burnAmount = (amount * 5) / 100;
1252   super._burn(sender, burnAmount);
1253   super._transfer(sender, recipient, amount - burnAmount);
1254   }
1255
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1251

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1250
1251   uint256 burnAmount = (amount * 5) / 100;
1252   super._burn(sender, burnAmount);
1253   super._transfer(sender, recipient, amount - burnAmount);
1254   }
1255
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1253

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1252   super._burn(sender, burnAmount);
1253   super._transfer(sender, recipient, amount - burnAmount);
1254   }
1255   }
1256   }
1257
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 10

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

- LAND2.sol

Locations

```
9
10   pragma solidity ^0.8.0;
11
12   /**
13   * @dev Interface of the ERC20 standard as defined in the EIP.
14
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 101

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

- LAND2.sol

Locations

```
100
101   pragma solidity ^0.8.0;
102
103   /**
104   * @dev Interface for the optional metadata functions from the ERC20 standard.
105
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 129

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

- LAND2.sol

Locations

```
128
129   pragma solidity ^0.8.0;
130
131   /**
132   * @dev Provides information about the current execution context, including the
133
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 155

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

- LAND2.sol

Locations

```
154
155   pragma solidity ^0.8.0;
156
157   /**
158   * @dev Implementation of the {IERC20} interface.
159
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 575

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

- LAND2.sol

Locations

```
574
575   pragma solidity ^0.8.0;
576
577   /**
578   * @dev Contract module which provides a basic access control mechanism, where
579
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 655

low SEVERITY

The current pragma Solidity directive is ""^0.5.0"". 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

- LAND2.sol

Locations

```
654
655   pragma solidity ^0.5.0;
656
657   interface IPancakeFactory {
658   event PairCreated(
659
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 689

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- LAND2.sol

Locations

```
688
689   pragma solidity ^0.6.2;
690
691   interface IPancakeRouter01 {
692   function factory() external pure returns (address);
693
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 852

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- LAND2.sol

Locations

```
851
852   pragma solidity ^0.6.2;
853
854   interface IPancakeRouter02 is IPancakeRouter01 {
855   function removeLiquidityETHSupportingFeeOnTransferTokens(
856
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 903

low SEVERITY

The current pragma Solidity directive is ""^0.5.0"". 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

- LAND2.sol

Locations

```
902
903   pragma solidity ^0.5.0;
904
905   interface IPancakePair {
906   event Approval(
907
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1016

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

- LAND2.sol

Locations

```
1015
1016   pragma solidity ^0.8.0;
1017
1018   abstract contract IUserParent2 {
1019   function getParent(address addr) public view virtual returns (address) {}
1020
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1035

low SEVERITY

The current pragma Solidity directive is ""=0.4.22<0.9.0"". 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

- LAND2.sol

Locations

```
1034
1035   pragma solidity >=0.4.22 <0.9.0;
1036
1037   contract LAND2 is ERC20, Ownable, IUserParent2 {
1038   mapping(address => bool) public _noBurnAddress;
1039
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1039

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1038   mapping(address => bool) public _noBurnAddress;
1039   address _stakeAddress = address(0);
1040   address public _swapV2Pair;
1041   address private _swapV2Router;
1042
1043
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1044

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1043   address private _BUSD;
1044   address topAddress;
1045   address public _tokenKeepingAddress;
1046   mapping(address => address) public _addressParentInfo;
1047   mapping(address => uint256) public _addressBuyAmount5;
1048
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1058

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "uniswapV2Router02" is internal. Other possible visibility settings are public and private.

Source File

- LAND2.sol

Locations

```
1057   ];
1058   IPancakeRouter02 uniswapV2Router02;
1059   IPancakePair uniswapV2Pair;
1060   uint256 public _startTime;
1061
1062
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1059

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "uniswapV2Pair" is internal. Other possible visibility settings are public and private.

Source File

- LAND2.sol

Locations

```
1058   IPancakeRouter02 uniswapV2Router02;
1059   IPancakePair uniswapV2Pair;
1060   uint256 public _startTime;
1061
1062   constructor(address swapV2Router, address BUSD) ERC20("META-UTOPIA LAND", "LAND")
       {
1063
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 1130

low SEVERITY

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

Source File

- LAND2.sol

Locations

```
1129   }
1130   return size > 0 && addr != tx.origin;
1131   }
1132
1133   function burn(uint256 amount) public virtual override {
1134
```

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

LINE 1050

low SEVERITY

The public state variable "_proctectedDayAmount" in "LAND2" contract has type "uint256[6]" and can cause an exception in case of use of invalid array index value.

Source File

- LAND2.sol

Locations

```
1049
1050   uint256[6] public _proctectedDayAmount = [
1051   0,
1052   1000 * 1e18,
1053   1300 * 1e18,
1054
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1206

low SEVERITY

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

Source File

- LAND2.sol

Locations

```
1205   for (uint256 index = 0; index < _proctectedDayAmount.length; index++) {
1206   canBuyTotalAmount = canBuyTotalAmount + _proctectedDayAmount[index];
1207   if (
1208   (_startTime + (index + 1) * 86400) > block.timestamp &&
1209   _addressBuyAmount5[buyer] > canBuyTotalAmount
1210
```

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.