BIT GAME VERSE TOKEN

Smart Contract Audit Report

SYSFIXED

02 Oct 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
BIT GAME VERSE TOKEN	BGVT	Binance Smart Chain

Addresses

Contract address	0xa03110800894b3ccf8723d991d80875561f96777
Contract deployer address	0xCEeE3D2DDC69Ac88aF00d12C2b0820eD21F360Ee

Project Website

https://bgverse.io/

Codebase

https://bscscan.com/address/0xa03110800894b3ccf8723d991d80875561f96777#code

SUMMARY

BGV launchpool offers a platform where a user can stake BGV tokens for earning token of other projects by spending no additional cost , that is, for absolutely free. The number of tokens generated depends on the number of tokens subscribed by the user to the pool and the total amount of BGV tokens staked in the pool. Usually a period of 30 days is allotted to the user to earn the new tokens. Every three seconds, The tokens earned by the user are calculated so that the user can harvest the pending rewards or amount at any point of time . BGV aims to become a benchmark for all the DEX platform. That is why we are highly committed to provide value, fairness, And innovation to the decentralized financial system. Through us, anyone can make and generate passive income by sitting in the comfort of their homes.

Contract Summary

Documentation Quality

BIT GAME VERSE 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 BIT GAME VERSE TOKEN with the discovery of several low issues.

Test Coverage

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

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 156, 156, 157, 157, 159, 159, 160, 160, 161, 161, 162, 162, 287, 293, 303, 335, 350, 352, 371, 372, 376, 379, 381, 385, 388, 390, 428, 428, 429, 429, 430, 430, 432, 432, 432, 455, 461, 462, 463, 466, 466, 466, 477, 483, 486, 487, 489, 521, 532, 537, 573, 579, 583, 586, 587, 595, 604, 604, 607, 607, 608, 614, 615, 615, 616, 623, 623, 668, 673, 673, 681, 691, 697, 697, 698, 698, 702, 702 and 352.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 14.
- 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 351, 352, 352, 484, 484, 486, 487, 648, 649, 682 and 692.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 164 and 573.

CONCLUSION

We have audited the BIT GAME VERSE TOKEN project released on October 2022 to discover issues and identify potential security vulnerabilities in the BIT GAME VERSE 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 BIT GAME VERSE 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 some arithmetic operation issues, a floating pragma is set, the potential use of "block.number" as a source of randomness, 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 ""^0.8.7"". 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. The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

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.	PASS
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.	ISSUE FOUND
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	Saturday Oct 01 2022 11:59:29 GMT+0000 (Coordinated Universal Time)
Finished	Sunday Oct 02 2022 01:32:32 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	BitGameVerse.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-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-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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	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-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-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 156

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
155
156   uint256 private _tTotal = 70000000000 * 10**_decimals;
157   uint256 private _rTotal = (MAX - (MAX % _tTotal));
158
159   uint256 public swapTokensAtAmount = 700_000_000 * 10**_decimals;
160
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 156

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
155
156   uint256 private _tTotal = 70000000000 * 10**_decimals;
157   uint256 private _rTotal = (MAX - (MAX % _tTotal));
158
159   uint256 public swapTokensAtAmount = 700_000_000 * 10**_decimals;
160
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 157

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
156   uint256 private _tTotal = 70000000000 * 10**_decimals;
157   uint256 private _rTotal = (MAX - (MAX % _tTotal));
158
159   uint256 public swapTokensAtAmount = 700_000_000 * 10**_decimals;
160   uint256 public maxBuyLimit = 700_000_000 * 10**_decimals;
161
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 157

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
156   uint256 private _tTotal = 70000000000 * 10**_decimals;
157   uint256 private _rTotal = (MAX - (MAX % _tTotal));
158
159   uint256 public swapTokensAtAmount = 700_000_000 * 10**_decimals;
160   uint256 public maxBuyLimit = 700_000_000 * 10**_decimals;
161
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 159

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
158
159   uint256 public swapTokensAtAmount = 700_000_000 * 10**_decimals;
160   uint256 public maxBuyLimit = 700_000_000 * 10**_decimals;
161   uint256 public maxSellLimit = 700_000_000 * 10**_decimals;
162   uint256 public maxWalletLimit = 700_000_000 * 10**_decimals;
163
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 159

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
158
159   uint256 public swapTokensAtAmount = 700_000_000 * 10**_decimals;
160   uint256 public maxBuyLimit = 700_000_000 * 10**_decimals;
161   uint256 public maxSellLimit = 700_000_000 * 10**_decimals;
162   uint256 public maxWalletLimit = 700_000_000 * 10**_decimals;
163
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 160

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
159   uint256 public swapTokensAtAmount = 700_000_000 * 10**_decimals;
160   uint256 public maxBuyLimit = 700_000_000 * 10**_decimals;
161   uint256 public maxSellLimit = 700_000_000 * 10**_decimals;
162   uint256 public maxWalletLimit = 700_000_000 * 10**_decimals;
163
164
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 160

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
159   uint256 public swapTokensAtAmount = 700_000_000 * 10**_decimals;
160   uint256 public maxBuyLimit = 700_000_000 * 10**_decimals;
161   uint256 public maxSellLimit = 700_000_000 * 10**_decimals;
162   uint256 public maxWalletLimit = 700_000_000 * 10**_decimals;
163
164
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 161

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
160   uint256 public maxBuyLimit = 700_000_000 * 10**_decimals;
161   uint256 public maxSellLimit = 700_000_000 * 10**_decimals;
162   uint256 public maxWalletLimit = 700_000_000 * 10**_decimals;
163
164   uint256 public genesis_block=block.number;
165
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 161

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
160   uint256 public maxBuyLimit = 700_000_000 * 10**_decimals;
161   uint256 public maxSellLimit = 700_000_000 * 10**_decimals;
162   uint256 public maxWalletLimit = 700_000_000 * 10**_decimals;
163
164   uint256 public genesis_block=block.number;
165
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 162

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
161   uint256 public maxSellLimit = 700_000_000 * 10**_decimals;
162   uint256 public maxWalletLimit = 700_000_000 * 10**_decimals;
163
164   uint256 public genesis_block=block.number;
165   uint256 private deadline;
166
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 162

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
161   uint256 public maxSellLimit = 700_000_000 * 10**_decimals;
162   uint256 public maxWalletLimit = 700_000_000 * 10**_decimals;
163
164   uint256 public genesis_block=block.number;
165   uint256 private deadline;
166
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 287

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
286   require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");
287   _approve(sender, _msgSender(), currentAllowance - amount);
288
289   return true;
290   }
291
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 293

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
292   function increaseAllowance(address spender, uint256 addedValue) public returns
(bool) {
293   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
294   return true;
295   }
296
297
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 303

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
302   require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below
zero");
303   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
304
305   return true;
306   }
307
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 335

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
334   uint256 currentRate = _getRate();
335   return rAmount / currentRate;
336   }
337
338   //@dev kept original RFI naming -> "reward" as in reflection
339
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 350

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
349   require(_isExcluded[account], "Account is not excluded");
350   for (uint256 i = 0; i < _excluded.length; i++) {
351   if (_excluded[i] == account) {
352   _excluded[i] = _excluded[_excluded.length - 1];
353   _tOwned[account] = 0;
354
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 352

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
351   if (_excluded[i] == account) {
352   _excluded[i] = _excluded[_excluded.length - 1];
353   _tOwned[account] = 0;
354   _isExcluded[account] = false;
355   _excluded.pop();
356
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 371

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
370   function _reflectRfi(uint256 rRfi, uint256 tRfi) private {
371   _rTotal -= rRfi;
372   totFeesPaid.rfi += tRfi;
373   }
374
375
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 372

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
371   _rTotal -= rRfi;
372   totFeesPaid.rfi += tRfi;
373   }
374
375   function _takeLiquidity(uint256 rLiquidity, uint256 tLiquidity) private {
376
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 376

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
375   function _takeLiquidity(uint256 rLiquidity, uint256 tLiquidity) private {
376   totFeesPaid.liquidity += tLiquidity;
377
378   if (_isExcluded[address(this)]) {
379   _tOwned[address(this)] += tLiquidity;
380
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 379

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
378   if (_isExcluded[address(this)]) {
379   _tOwned[address(this)] += tLiquidity;
380   }
381   _rOwned[address(this)] += rLiquidity;
382   }
383
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 381

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
380   }
381   _rOwned[address(this)] += rLiquidity;
382   }
383
384   function _takeMarketing(uint256 rMarketing, uint256 tMarketing) private {
385
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 385

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
384   function _takeMarketing(uint256 rMarketing, uint256 tMarketing) private {
385   totFeesPaid.marketing += tMarketing;
386
387   if (_isExcluded[address(this)]) {
388   _tOwned[address(this)] += tMarketing;
389
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 388

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
387   if (_isExcluded[address(this)]) {
388   _tOwned[address(this)] += tMarketing;
389   }
390   _rOwned[address(this)] += rMarketing;
391   }
392
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 390

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
389   }
390   _rOwned[address(this)] += rMarketing;
391   }
392
393
394
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 428

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
427
428   s.tRfi = (tAmount * temp.rfi) / 100;
429   s.tMarketing = (tAmount * temp.marketing) / 100;
430   s.tLiquidity = (tAmount * temp.liquidity) / 100;
431   s.tTransferAmount =
432
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 428

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
427
428   s.tRfi = (tAmount * temp.rfi) / 100;
429   s.tMarketing = (tAmount * temp.marketing) / 100;
430   s.tLiquidity = (tAmount * temp.liquidity) / 100;
431   s.tTransferAmount =
432
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 429

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
428   s.tRfi = (tAmount * temp.rfi) / 100;
429   s.tMarketing = (tAmount * temp.marketing) / 100;
430   s.tLiquidity = (tAmount * temp.liquidity) / 100;
431   s.tTransferAmount =
432   tAmount -
433
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 429

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
428   s.tRfi = (tAmount * temp.rfi) / 100;
429   s.tMarketing = (tAmount * temp.marketing) / 100;
430   s.tLiquidity = (tAmount * temp.liquidity) / 100;
431   s.tTransferAmount =
432   tAmount -
433
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 430

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
429   s.tMarketing = (tAmount * temp.marketing) / 100;
430   s.tLiquidity = (tAmount * temp.liquidity) / 100;
431   s.tTransferAmount =
432   tAmount -
433   s.tRfi -
434
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 430

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
429   s.tMarketing = (tAmount * temp.marketing) / 100;
430   s.tLiquidity = (tAmount * temp.liquidity) / 100;
431   s.tTransferAmount =
432   tAmount -
433   s.tRfi -
434
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 432

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
431   s.tTransferAmount =
432   tAmount -
433   s.tRfi -
434   s.tMarketing -
435   s.tLiquidity;
436
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 432

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
431   s.tTransferAmount =
432   tAmount -
433   s.tRfi -
434   s.tMarketing -
435   s.tLiquidity;
436
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 432

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
431   s.tTransferAmount =
432   tAmount -
433   s.tRfi -
434   s.tMarketing -
435   s.tLiquidity;
436
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 455

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
454   {
455   rAmount = tAmount * currentRate;
456
457   if (!takeFee) {
458   return (rAmount, rAmount, 0, 0, 0);
459
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 461

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
460
461   rRfi = s.tRfi * currentRate;
462   rMarketing = s.tMarketing * currentRate;
463   rLiquidity = s.tLiquidity * currentRate;
464
465
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 462

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
461   rRfi = s.tRfi * currentRate;
462   rMarketing = s.tMarketing * currentRate;
463   rLiquidity = s.tLiquidity * currentRate;
464
465   rTransferAmount =
466
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 463

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
462   rMarketing = s.tMarketing * currentRate;
463   rLiquidity = s.tLiquidity * currentRate;
464
465   rTransferAmount =
466   rAmount -
467
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 466

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
465   rTransferAmount =
466   rAmount -
467   rRfi -
468   rMarketing -
469   rLiquidity ;
470
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 466

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
465   rTransferAmount =
466   rAmount -
467   rRfi -
468   rMarketing -
469   rLiquidity ;
470
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 466

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
465   rTransferAmount =
466   rAmount -
467   rRfi -
468   rMarketing -
469   rLiquidity ;
470
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 477

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
476   (uint256 rSupply, uint256 tSupply) = _getCurrentSupply();
477   return rSupply / tSupply;
478   }
479
480   function _getCurrentSupply() private view returns (uint256, uint256) {
481
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 483

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
482   uint256 tSupply = _tTotal;
483   for (uint256 i = 0; i < _excluded.length; i++) {
484   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply)
485   return (_rTotal, _tTotal);
486   rSupply = rSupply - _rOwned[_excluded[i]];
487
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 486

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
485   return (_rTotal, _tTotal);
486   rSupply = rSupply - _rOwned[_excluded[i]];
487   tSupply = tSupply - _tOwned[_excluded[i]];
488   }
489   if (rSupply < _rTotal / _tTotal) return (_rTotal, _tTotal);
490
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 487

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
486   rSupply = rSupply - _rOwned[_excluded[i]];
487   tSupply = tSupply - _tOwned[_excluded[i]];
488   }
489   if (rSupply < _rTotal / _tTotal) return (_rTotal, _tTotal);
490   return (rSupply, tSupply);
491
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 489

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
488   }
489   if (rSupply < _rTotal / _tTotal) return (_rTotal, _tTotal);
490   return (rSupply, tSupply);
491   }
492
493
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 521

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
520   require(
521   balanceOf(to) + amount <= maxWalletLimit,
522   "You are exceeding maxWalletLimit"
523   );
524   }
525
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 532

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
531   require(
532   balanceOf(to) + amount <= maxWalletLimit,
533   "You are exceeding maxWalletLimit"
534   );
535   }
536
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 537

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
536   if (coolDownEnabled) {
537   uint256 timePassed = block.timestamp - _lastSell[from];
538   require(timePassed >= coolDownTime, "Cooldown enabled");
539   _lastSell[from] = block.timestamp;
540   }
541
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 573

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
572   !_isExcludedFromFee[recipient] &&
573   block.number <= genesis_block + deadline;
574
575   valuesFromGetValues memory s = _getValues(tAmount, takeFee, isSell, useLaunchTax);
576
577
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 579

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
578   //from excluded
579   _tOwned[sender] = _tOwned[sender] - tAmount;
580   }
581   if (_isExcluded[recipient]) {
582   //to excluded
583
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 583

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
582   //to excluded
583   _tOwned[recipient] = _tOwned[recipient] + s.tTransferAmount;
584   }
585
586   _rOwned[sender] = _rOwned[sender] - s.rAmount;
587
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 586

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
585
586   _rOwned[sender] = _rOwned[sender] - s.rAmount;
587   _rOwned[recipient] = _rOwned[recipient] + s.rTransferAmount;
588
589   if (s.rRfi > 0 || s.tRfi > 0) _reflectRfi(s.rRfi, s.tRfi);
590
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 587

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
586   _rOwned[sender] = _rOwned[sender] - s.rAmount;
587   _rOwned[recipient] = _rOwned[recipient] + s.rTransferAmount;
588
589   if (s.rRfi > 0 || s.tRfi > 0) _reflectRfi(s.rRfi, s.tRfi);
590   if (s.rLiquidity > 0 || s.tLiquidity > 0) {
591
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 595

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
594   address(this),
595   s.tLiquidity + s.tMarketing
596   );
597   }
598   if (s.rMarketing > 0 || s.tMarketing > 0) _takeMarketing(s.rMarketing,
s.tMarketing);
599
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 604

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
603   function swapAndLiquify(uint256 contractBalance, Taxes memory temp) private
lockTheSwap {
604   uint256 denominator = (temp.liquidity +
605   temp.marketing
606   ) * 2;
607   uint256 tokensToAddLiquidityWith = (contractBalance * temp.liquidity) /
denominator;
608
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 604

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
603   function swapAndLiquify(uint256 contractBalance, Taxes memory temp) private
lockTheSwap {
604   uint256 denominator = (temp.liquidity +
605   temp.marketing
606   ) * 2;
607   uint256 tokensToAddLiquidityWith = (contractBalance * temp.liquidity) /
denominator;
608
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 607

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
606   ) * 2;
607   uint256 tokensToAddLiquidityWith = (contractBalance * temp.liquidity) /
denominator;
608   uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
609
610   uint256 initialBalance = address(this).balance;
611
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 607

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
606   ) * 2;
607   uint256 tokensToAddLiquidityWith = (contractBalance * temp.liquidity) /
denominator;
608   uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
609
610   uint256 initialBalance = address(this).balance;
611
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 608

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
607   uint256 tokensToAddLiquidityWith = (contractBalance * temp.liquidity) /
denominator;
608   uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
609
610   uint256 initialBalance = address(this).balance;
611
612
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 614

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
613
614   uint256 deltaBalance = address(this).balance - initialBalance;
615   uint256 unitBalance = deltaBalance / (denominator - temp.liquidity);
616   uint256 bnbToAddLiquidityWith = unitBalance * temp.liquidity;
617
618
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 615

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
614   uint256 deltaBalance = address(this).balance - initialBalance;
615   uint256 unitBalance = deltaBalance / (denominator - temp.liquidity);
616   uint256 bnbToAddLiquidityWith = unitBalance * temp.liquidity;
617
618   if (bnbToAddLiquidityWith > 0) {
619
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 615

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
614   uint256 deltaBalance = address(this).balance - initialBalance;
615   uint256 unitBalance = deltaBalance / (denominator - temp.liquidity);
616   uint256 bnbToAddLiquidityWith = unitBalance * temp.liquidity;
617
618   if (bnbToAddLiquidityWith > 0) {
619
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 616

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
615   uint256 unitBalance = deltaBalance / (denominator - temp.liquidity);
616   uint256 bnbToAddLiquidityWith = unitBalance * temp.liquidity;
617
618   if (bnbToAddLiquidityWith > 0) {
619   // Add liquidity to pancake
620
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 623

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
622
623   uint256 marketingAmt = unitBalance * 2 * temp.marketing;
624   if (marketingAmt > 0) {
625   payable(marketingWallet).sendValue(marketingAmt);
626   }
627
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 623

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
622
623   uint256 marketingAmt = unitBalance * 2 * temp.marketing;
624   if (marketingAmt > 0) {
625   payable(marketingWallet).sendValue(marketingAmt);
626   }
627
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 668

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
667   function updateCooldown(bool state, uint256 time) external onlyOwner {
668   coolDownTime = time * 1 seconds;
669   coolDownEnabled = state;
670   }
671
672
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 673

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
672   function updateSwapTokensAtAmount(uint256 amount) external onlyOwner {
673   swapTokensAtAmount = amount * 10**_decimals;
674   }
675
676   function updateIsBlacklisted(address account, bool state) external onlyOwner {
677
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 673

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
672   function updateSwapTokensAtAmount(uint256 amount) external onlyOwner {
673   swapTokensAtAmount = amount * 10**_decimals;
674   }
675
676   function updateIsBlacklisted(address account, bool state) external onlyOwner {
677
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 681

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
680   function bulkIsBlacklisted(address[] memory accounts, bool state) external
onlyOwner {
681   for (uint256 i = 0; i < accounts.length; i++) {
682   _isBlacklisted[accounts[i]] = state;
683   }
684   }
685
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 691

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
690   function bulkupdateAllowedTransfer(address[] memory accounts, bool state) external
onlyOwner {
691   for (uint256 i = 0; i < accounts.length; i++) {
692   allowedTransfer[accounts[i]] = state;
693   }
694   }
695
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 697

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
696   function updateMaxTxLimit(uint256 maxBuy, uint256 maxSell) external onlyOwner {
697   maxBuyLimit = maxBuy * 10**decimals();
698   maxSellLimit = maxSell * 10**decimals();
699   }
700
701
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 697

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
696   function updateMaxTxLimit(uint256 maxBuy, uint256 maxSell) external onlyOwner {
697   maxBuyLimit = maxBuy * 10**decimals();
698   maxSellLimit = maxSell * 10**decimals();
699   }
700
701
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 698

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
697   maxBuyLimit = maxBuy * 10**decimals();
698   maxSellLimit = maxSell * 10**decimals();
699   }
700
701   function updateMaxWalletlimit(uint256 amount) external onlyOwner {
702
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 698

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
697   maxBuyLimit = maxBuy * 10**decimals();
698   maxSellLimit = maxSell * 10**decimals();
699   }
700
701   function updateMaxWalletlimit(uint256 amount) external onlyOwner {
702
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 702

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
701   function updateMaxWalletlimit(uint256 amount) external onlyOwner {
702   maxWalletLimit = amount * 10**decimals();
703   }
704
705   function updateRouterAndPair(address newRouter, address newPair) external onlyOwner
{
706
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 702

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
701   function updateMaxWalletlimit(uint256 amount) external onlyOwner {
702   maxWalletLimit = amount * 10**decimals();
703   }
704
705   function updateRouterAndPair(address newRouter, address newPair) external onlyOwner
{
706
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 352

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
351   if (_excluded[i] == account) {
352   _excluded[i] = _excluded[_excluded.length - 1];
353   _tOwned[account] = 0;
354   _isExcluded[account] = false;
355   _excluded.pop();
356
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 14

low SEVERITY

The current pragma Solidity directive is ""^0.8.7"". 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

- BitGameVerse.sol

Locations

```
13   // SPDX-License-Identifier: UNLICENSE
14   pragma solidity ^0.8.7;
15
16
17
18
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 351

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
350   for (uint256 i = 0; i < _excluded.length; i++) {
351   if (_excluded[i] == account) {
352   _excluded[i] = _excluded[_excluded.length - 1];
353   _tOwned[account] = 0;
354   _isExcluded[account] = false;
355
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 352

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
351   if (_excluded[i] == account) {
352   _excluded[i] = _excluded[_excluded.length - 1];
353   _tOwned[account] = 0;
354   _isExcluded[account] = false;
355   _excluded.pop();
356
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 352

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
351   if (_excluded[i] == account) {
352   _excluded[i] = _excluded[_excluded.length - 1];
353   _tOwned[account] = 0;
354   _isExcluded[account] = false;
355   _excluded.pop();
356
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 484

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
483   for (uint256 i = 0; i < _excluded.length; i++) {
484   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply)
485   return (_rTotal, _tTotal);
486   rSupply = rSupply - _rOwned[_excluded[i]];
487   tSupply = tSupply - _tOwned[_excluded[i]];
488
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 484

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
483   for (uint256 i = 0; i < _excluded.length; i++) {
484   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply)
485   return (_rTotal, _tTotal);
486   rSupply = rSupply - _rOwned[_excluded[i]];
487   tSupply = tSupply - _tOwned[_excluded[i]];
488
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 486

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
485   return (_rTotal, _tTotal);
486   rSupply = rSupply - _rOwned[_excluded[i]];
487   tSupply = tSupply - _tOwned[_excluded[i]];
488   }
489   if (rSupply < _rTotal / _tTotal) return (_rTotal, _tTotal);
490
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 487

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
486   rSupply = rSupply - _rOwned[_excluded[i]];
487   tSupply = tSupply - _tOwned[_excluded[i]];
488   }
489   if (rSupply < _rTotal / _tTotal) return (_rTotal, _tTotal);
490   return (rSupply, tSupply);
491
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 648

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
647   address[] memory path = new address[](2);
648   path[0] = address(this);
649   path[1] = router.WETH();
650
651   _approve(address(this), address(router), tokenAmount);
652
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 649

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
648   path[0] = address(this);
649   path[1] = router.WETH();
650
651   _approve(address(this), address(router), tokenAmount);
652
653
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 682

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
681   for (uint256 i = 0; i < accounts.length; i++) {
682   _isBlacklisted[accounts[i]] = state;
683   }
684   }
685
686
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 692

low SEVERITY

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

Source File

- BitGameVerse.sol

Locations

```
691   for (uint256 i = 0; i < accounts.length; i++) {
692   allowedTransfer[accounts[i]] = state;
693   }
694   }
695
696
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 164

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- BitGameVerse.sol

Locations

```
163
164   uint256 public genesis_block=block.number;
165   uint256 private deadline;
166
167   address public deadWallet = 0x000000000000000000000000000000000000dEaD;
168
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 573

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- BitGameVerse.sol

Locations

```
572   !_isExcludedFromFee[recipient] &&
573   block.number <= genesis_block + deadline;
574
575   valuesFromGetValues memory s = _getValues(tAmount, takeFee, isSell, useLaunchTax);
576
577
```

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.