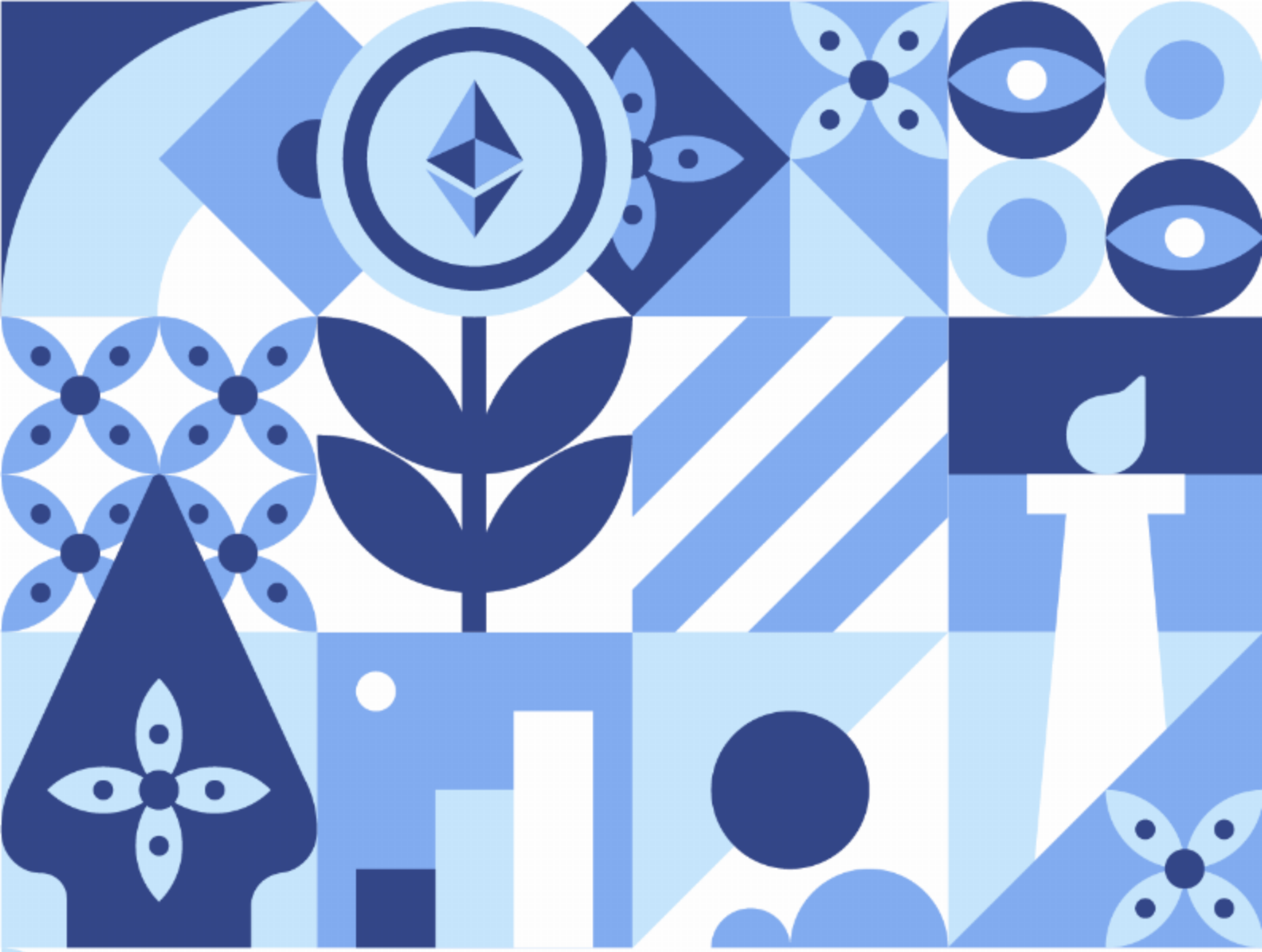

BitANT

Smart Contract Audit Report

SYSFIXED

06 Oct 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
BitANT	BitANT	Ethereum

Addresses

Contract address	0x15ee120fd69bec86c1d38502299af7366a41d1a6
Contract deployer address	0x15Ee120fD69BEc86C1d38502299af7366a41D1a6

Project Website

https://bitbtc.money/

Codebase

https://etherscan.io/address/0x15ee120fd69bec86c1d38502299af7366a41d1a6#code

SUMMARY

The BitBTC Protocol is to solve the problems of the high price of BTC and slow transfer speed, the BitBTC Protocol proposes a solution to split BTC into BitBTC on the Ethereum through smart contracts, that is,1BTC = 1 million BitBTC.BitBTC has faster transfer speed, lower transfer fees, is more suitable for micropayment, more energy saving, and more convenient participation in DeFi. BitBTC makes BTC simpler to buy everything. All exchange fees between BTC and BitBTC are used to repurchase and burn BitANT.

Contract Summary

Documentation Quality

BitANT 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 BitANT 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 988, 1007, 1029, 1062, 1064, 1085, 1086, 1111, 1113, 1226, 1773, 1781, 1902, 1902, 1913, 1913, 1913, 2224, 2274, 2278, 2390, 2393, 2394, 2401, 2405, 2528, 2528, 2529, 2529, 2529, 1781, 2224, 2278, 2390, 2393 and 2394.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 5.
- 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 2202, 2224, 2271, 2278, 2390, 2393 and 2394.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 2235, 2248, 2393 and 2396.

CONCLUSION

We have audited the BitANT project released on October 2022 to discover issues and identify potential security vulnerabilities in BitANT 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 a satisfactory result with some low-risk issues.

The issues found in the BitANT smart contract code 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 state variable visibility is not set, weak sources 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. We recommend to don't using any of those environment variables as sources of randomness and being aware that the use of these variables introduces a certain level of trust in 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	Wednesday Oct 05 2022 07:59:22 GMT+0000 (Coordinated Universal Time)
Finished	Thursday Oct 06 2022 17:25:27 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	BitANT.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged

SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" 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-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-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 988

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
987   unchecked {
988   _approve(sender, _msgSender(), currentAllowance - amount);
989   }
990
991   return true;
992
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1007

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1006   function increaseAllowance(address spender, uint256 addedValue) public virtual
returns (bool) {
1007   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
1008   return true;
1009   }
1010
1011
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1029

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1028   unchecked {
1029   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
1030   }
1031
1032   return true;
1033
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1062

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1061   unchecked {
1062   _balances[sender] = senderBalance - amount;
1063   }
1064   _balances[recipient] += amount;
1065
1066
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1064

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1063   }
1064   _balances[recipient] += amount;
1065
1066   emit Transfer(sender, recipient, amount);
1067
1068
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1085

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1084
1085   _totalSupply += amount;
1086   _balances[account] += amount;
1087   emit Transfer(address(0), account, amount);
1088
1089
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1086

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1085   _totalSupply += amount;
1086   _balances[account] += amount;
1087   emit Transfer(address(0), account, amount);
1088
1089   _afterTokenTransfer(address(0), account, amount);
1090
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1111

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1110   unchecked {
1111   _balances[account] = accountBalance - amount;
1112   }
1113   _totalSupply -= amount;
1114
1115
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 1113

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1112   }
1113   _totalSupply -= amount;
1114
1115   emit Transfer(account, address(0), amount);
1116
1117
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1226

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1225   unchecked {
1226   _approve(account, _msgSender(), currentAllowance - amount);
1227   }
1228   _burn(account, amount);
1229   }
1230
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1773

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1772   unchecked {
1773   counter._value += 1;
1774   }
1775   }
1776
1777
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1781

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1780   unchecked {
1781   counter._value = value - 1;
1782   }
1783   }
1784
1785
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1902

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1901   // (a + b) / 2 can overflow.
1902   return (a & b) + (a ^ b) / 2;
1903   }
1904
1905   /**
1906
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1902

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1901   // (a + b) / 2 can overflow.
1902   return (a & b) + (a ^ b) / 2;
1903   }
1904
1905   /**
1906
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1913

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1912   // (a + b - 1) / b can overflow on addition, so we distribute.
1913   return a / b + (a % b == 0 ? 0 : 1);
1914   }
1915   }
1916
1917
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1913

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1912   // (a + b - 1) / b can overflow on addition, so we distribute.
1913   return a / b + (a % b == 0 ? 0 : 1);
1914   }
1915   }
1916
1917
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1913

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1912   // (a + b - 1) / b can overflow on addition, so we distribute.
1913   return a / b + (a % b == 0 ? 0 : 1);
1914   }
1915   }
1916
1917
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2224

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2223   uint256 pos = _checkpoints[account].length;
2224   return pos == 0 ? 0 : _checkpoints[account][pos - 1].votes;
2225   }
2226
2227   /**
2228
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 2274

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2273   } else {
2274   low = mid + 1;
2275   }
2276   }
2277
2278
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2278

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2277
2278   return high == 0 ? 0 : ckpts[high - 1].votes;
2279   }
2280
2281   /**
2282
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2390

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2389   uint256 pos = ckpts.length;
2390   oldWeight = pos == 0 ? 0 : ckpts[pos - 1].votes;
2391   newWeight = op(oldWeight, delta);
2392
2393   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
2394
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2393

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2392
2393   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
2394   ckpts[pos - 1].votes = SafeCastUpgradeable.toUint224(newWeight);
2395   } else {
2396   ckpts.push(Checkpoint({fromBlock: SafeCastUpgradeable.toUint32(block.number),
votes: SafeCastUpgradeable.toUint224(newWeight)}));
2397
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2394

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2393   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
2394   ckpts[pos - 1].votes = SafeCastUpgradeable.toUint224(newWeight);
2395   } else {
2396   ckpts.push(Checkpoint({fromBlock: SafeCastUpgradeable.toUint32(block.number),
votes: SafeCastUpgradeable.toUint224(newWeight)}));
2397   }
2398
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 2401

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2400   function _add(uint256 a, uint256 b) private pure returns (uint256) {
2401   return a + b;
2402   }
2403
2404   function _subtract(uint256 a, uint256 b) private pure returns (uint256) {
2405
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2405

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2404   function _subtract(uint256 a, uint256 b) private pure returns (uint256) {
2405   return a - b;
2406   }
2407   uint256[47] private __gap;
2408   }
2409
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 2528

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2527   } else {
2528   super._transfer(sender, _feeCollector, amount * _fee / FEE_RATIO);
2529   super._transfer(sender, recipient, amount * (FEE_RATIO - _fee) / FEE_RATIO);
2530   }
2531   }
2532
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 2528

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2527   } else {
2528   super._transfer(sender, _feeCollector, amount * _fee / FEE_RATIO);
2529   super._transfer(sender, recipient, amount * (FEE_RATIO - _fee) / FEE_RATIO);
2530   }
2531   }
2532
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 2529

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2528   super._transfer(sender, _feeCollector, amount * _fee / FEE_RATIO);
2529   super._transfer(sender, recipient, amount * (FEE_RATIO - _fee) / FEE_RATIO);
2530   }
2531   }
2532
2533
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 2529

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2528   super._transfer(sender, _feeCollector, amount * _fee / FEE_RATIO);
2529   super._transfer(sender, recipient, amount * (FEE_RATIO - _fee) / FEE_RATIO);
2530   }
2531   }
2532
2533
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2529

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2528   super._transfer(sender, _feeCollector, amount * _fee / FEE_RATIO);
2529   super._transfer(sender, recipient, amount * (FEE_RATIO - _fee) / FEE_RATIO);
2530   }
2531   }
2532
2533
```

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

LINE 1781

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
1780   unchecked {
1781   counter._value = value - 1;
1782   }
1783   }
1784
1785
```

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

LINE 2224

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2223   uint256 pos = _checkpoints[account].length;
2224   return pos == 0 ? 0 : _checkpoints[account][pos - 1].votes;
2225   }
2226
2227   /**
2228
```

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

LINE 2278

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2277
2278   return high == 0 ? 0 : ckpts[high - 1].votes;
2279   }
2280
2281   /**
2282
```

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

LINE 2390

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2389   uint256 pos = ckpts.length;
2390   oldWeight = pos == 0 ? 0 : ckpts[pos - 1].votes;
2391   newWeight = op(oldWeight, delta);
2392
2393   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
2394
```

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

LINE 2393

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2392
2393   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
2394   ckpts[pos - 1].votes = SafeCastUpgradeable.toUint224(newWeight);
2395   } else {
2396   ckpts.push(Checkpoint({fromBlock: SafeCastUpgradeable.toUint32(block.number),
votes: SafeCastUpgradeable.toUint224(newWeight)}));
2397
```

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

LINE 2394

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2393   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
2394   ckpts[pos - 1].votes = SafeCastUpgradeable.toUint224(newWeight);
2395   } else {
2396   ckpts.push(Checkpoint({fromBlock: SafeCastUpgradeable.toUint32(block.number),
votes: SafeCastUpgradeable.toUint224(newWeight)}));
2397   }
2398
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 5

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

- BitANT.sol

Locations

```
4
5   pragma solidity ^0.8.0;
6
7   // SPDX-License-Identifier: MIT
8
9
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2202

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2201   function checkpoints(address account, uint32 pos) public view virtual returns
(Checkpoint memory) {
2202   return _checkpoints[account][pos];
2203   }
2204
2205   /**
2206
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2224

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2223   uint256 pos = _checkpoints[account].length;
2224   return pos == 0 ? 0 : _checkpoints[account][pos - 1].votes;
2225   }
2226
2227   /**
2228
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2271

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2270   uint256 mid = MathUpgradeable.average(low, high);
2271   if (ckpts[mid].fromBlock > blockNumber) {
2272   high = mid;
2273   } else {
2274   low = mid + 1;
2275
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2278

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2277
2278   return high == 0 ? 0 : ckpts[high - 1].votes;
2279   }
2280
2281   /**
2282
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2390

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2389   uint256 pos = ckpts.length;
2390   oldWeight = pos == 0 ? 0 : ckpts[pos - 1].votes;
2391   newWeight = op(oldWeight, delta);
2392
2393   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
2394
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2393

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2392
2393   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
2394   ckpts[pos - 1].votes = SafeCastUpgradeable.toUint224(newWeight);
2395   } else {
2396   ckpts.push(Checkpoint({fromBlock: SafeCastUpgradeable.toUint32(block.number),
votes: SafeCastUpgradeable.toUint224(newWeight)}));
2397
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2394

low SEVERITY

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

Source File

- BitANT.sol

Locations

```
2393   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
2394   ckpts[pos - 1].votes = SafeCastUpgradeable.toUint224(newWeight);
2395   } else {
2396   ckpts.push(Checkpoint({fromBlock: SafeCastUpgradeable.toUint32(block.number),
votes: SafeCastUpgradeable.toUint224(newWeight)}));
2397   }
2398
```

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

LINE 2235

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

- BitANT.sol

Locations

```
2234   function getPastVotes(address account, uint256 blockNumber) public view returns
(uint256) {
2235   require(blockNumber < block.number, "ERC20Votes: block not yet mined");
2236   return _checkpointsLookup(_checkpoints[account], blockNumber);
2237   }
2238
2239
```

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

LINE 2248

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

- BitANT.sol

Locations

```
2247   function getPastTotalSupply(uint256 blockNumber) public view returns (uint256) {
2248   require(blockNumber < block.number, "ERC20Votes: block not yet mined");
2249   return _checkpointsLookup(_totalSupplyCheckpoints, blockNumber);
2250   }
2251
2252
```

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

LINE 2393

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

- BitANT.sol

Locations

```
2392
2393   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
2394   ckpts[pos - 1].votes = SafeCastUpgradeable.toUint224(newWeight);
2395   } else {
2396   ckpts.push(Checkpoint({fromBlock: SafeCastUpgradeable.toUint32(block.number),
votes: SafeCastUpgradeable.toUint224(newWeight)}));
2397
```

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

LINE 2396

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

- BitANT.sol

Locations

```
2395   } else {
2396   ckpts.push(Checkpoint({fromBlock: SafeCastUpgradeable.toUint32(block.number),
votes: SafeCastUpgradeable.toUint224(newWeight)}));
2397   }
2398   }
2399
2400
```

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.