TwitVi

Smart Contract Audit Report

SYSFIXED

05 Feb 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
TwitVi	TWV	Binance Smart Chain

Addresses

Contract address	0x4392a96Fec68E162471793db631972ccAf80FE1C
Contract deployer address	0x8A0aEABF90Baa71df59114B60706e14E60E37A97

Project Website

https://twitvi.com/

Codebase

https://bscscan.com/address/0x4392a96Fec68E162471793db631972ccAf80FE1C#code

SUMMARY

TwitVi is a Web3 social networking service with GameFi functionality. Users reserve NFTs featuring bird designs; tweeting on Twitter using #TwitVi earns in-game tokens that can be used in-game or cashed in for profit. twitVi encourages people from all over the world to interact with each other.

Contract Summary

Documentation Quality

TwitVi 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 TwitVi 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 632, 654, 680, 797, 813, 835, 836, 849, 851, 866, 867, 899, 1023, 1023, 1028, 1032, 1033, 1178, 1178, 1180, 1180, 1183, 1207, 1209 and 1209.
- 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 1196 and 1197.

CONCLUSION

We have audited the TwitVi project released on February-2023 to discover issues and identify potential security vulnerabilities in TwitVi 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 TwitVi 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 floating pragma is set, a state variable visibility is not set, weak sources of randomness, tx.origin as a part of authorization control 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.

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.	PASS
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

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

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

SMART CONTRACT ANALYSIS

Started	Saturday Feb 04 2023 13:20:28 GMT+0000 (Coordinated Universal Time)
Finished	Sunday Feb 05 2023 02:07:31 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	TwitVi.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-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 632

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
631   ) internal {
632   uint256 newAllowance = token.allowance(address(this), spender) + value;
633   _callOptionalReturn(
634   token,
635   abi.encodeWithSelector(
636
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 654

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
653   );
654   uint256 newAllowance = oldAllowance - value;
655   _callOptionalReturn(
656   token,
657   abi.encodeWithSelector(
658
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 680

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
679   require(
680   nonceAfter == nonceBefore + 1,
681   "SafeERC20: permit did not succeed"
682   );
683   }
684
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 797

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
796   address owner = _msgSender();
797   _approve(owner, spender, allowance(owner, spender) + addedValue);
798   return true;
799   }
800
801
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 813

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
812   unchecked {
813   _approve(owner, spender, currentAllowance - subtractedValue);
814   }
815
816   return true;
817
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 835

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
834   unchecked {
835   _balances[from] = fromBalance - amount;
836   _balances[to] += amount;
837   }
838
839
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 836

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
835   _balances[from] = fromBalance - amount;
836   _balances[to] += amount;
837   }
838
839   emit Transfer(from, to, amount);
840
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 849

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
848
849   _totalSupply += amount;
850   unchecked {
851   _balances[account] += amount;
852   }
853
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 851

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
850   unchecked {
851   _balances[account] += amount;
852   }
853   emit Transfer(address(0), account, amount);
854
855
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 866

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
865   unchecked {
866   _balances[account] = accountBalance - amount;
867   _totalSupply -= amount;
868   }
869
870
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 867

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
866   _balances[account] = accountBalance - amount;
867   _totalSupply -= amount;
868   }
869
870   emit Transfer(account, address(0), amount);
871
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 899

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
898   unchecked {
899   _approve(owner, spender, currentAllowance - amount);
900   }
901   }
902   }
903
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1023

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1022   constructor() ERC20("TwitVi", "TWV") {
1023   _mint(owner(), 100_000_000 * (10**18));
1024
1025   taxDenominator = 100;
1026   buyTax = 0;
1027
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1023

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1022   constructor() ERC20("TwitVi", "TWV") {
1023   _mint(owner(), 100_000_000 * (10**18));
1024
1025   taxDenominator = 100;
1026   buyTax = 0;
1027
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1028

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1027   sellTax = 2;
1028   totalTax = buyTax + sellTax;
1029
1030   marketingWalletShares = 100;
1031
1032
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1032

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1031
1032   setSwapTokensLimit = totalSupply() / 1_000_000; // 0.0001% of Total Supply
1033   swapTokensAtAmount = totalSupply() / 2000; // 0.05% of Total Supply
1034   isSwapBackEnabled = true;
1035
1036
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1033

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1032   setSwapTokensLimit = totalSupply() / 1_000_000; // 0.0001% of Total Supply
1033   swapTokensAtAmount = totalSupply() / 2000; // 0.05% of Total Supply
1034   isSwapBackEnabled = true;
1035
1036   address router = getRouterAddress();
1037
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1178

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1177   if (_isAutomatedMarketMakerPair[from]) {
1178   fees = (amount * buyTax) / taxDenominator;
1179   } else if (_isAutomatedMarketMakerPair[to]) {
1180   fees = (amount * sellTax) / taxDenominator;
1181   }
1182
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1178

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1177   if (_isAutomatedMarketMakerPair[from]) {
1178   fees = (amount * buyTax) / taxDenominator;
1179   } else if (_isAutomatedMarketMakerPair[to]) {
1180   fees = (amount * sellTax) / taxDenominator;
1181   }
1182
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1180

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1179   } else if (_isAutomatedMarketMakerPair[to]) {
1180   fees = (amount * sellTax) / taxDenominator;
1181   }
1182
1183   amount = amount - fees;
1184
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1180

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1179   } else if (_isAutomatedMarketMakerPair[to]) {
1180   fees = (amount * sellTax) / taxDenominator;
1181   }
1182
1183   amount = amount - fees;
1184
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1183

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1182
1183   amount = amount - fees;
1184
1185   super._transfer(from, address(this), fees);
1186   }
1187
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1207

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1206
1207   uint256 newBalance = address(this).balance - initialBalance;
1208
1209   uint256 marketingShare = (newBalance * marketingWalletShares) / 100;
1210
1211
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1209

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1208
1209   uint256 marketingShare = (newBalance * marketingWalletShares) / 100;
1210
1211   if (marketingShare > 0) {
1212   sendBNB(marketingWallet, marketingShare);
1213
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1209

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1208
1209   uint256 marketingShare = (newBalance * marketingWalletShares) / 100;
1210
1211   if (marketingShare > 0) {
1212   sendBNB(marketingWallet, marketingShare);
1213
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1196

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1195   address[] memory path = new address[](2);
1196   path[0] = address(this);
1197   path[1] = uniswapV2Router.WETH();
1198
1199   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
1200
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1197

low SEVERITY

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

Source File

- TwitVi.sol

Locations

```
1196   path[0] = address(this);
1197   path[1] = uniswapV2Router.WETH();
1198
1199   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
1200   tokenAmount,
1201
```

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.