Holy Shiba Pad

Smart Contract Audit Report

SYSFIXED

26 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Holy Shiba Pad	HOLYSHIB	Binance Smart Chain

Addresses

Contract address	0xD5Be0E4316194c5B87AC9420c00Bc31C00CC7E35
Contract deployer address	0x65EcDB67225Cee7d96c7E311f70593065bC322Bb

Project Website

https://www.holyshibapad.finance/

Codebase

https://bscscan.com/address/0xD5Be0E4316194c5B87AC9420c00Bc31C00CC7E35#code

SUMMARY

First meme charity utility token. 3 in 1 Token like the holy trinity. Leverage launchpad to donate 3% of the pool to charity. Every investors and project developers using Holy Shiba Pad will be donating to charity. Marketing focused to ensure fair launch buyers in profit.

Contract Summary

Documentation Quality

Holy Shiba Pad 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 Holy Shiba Pad 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 194, 216, 241, 270, 271, 378, 408, 418, 429 and 464.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 7.
- SWC-110 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 465.

CONCLUSION

We have audited the Holy Shiba Pad project which has released on January 2023 to discover issues and identify potential security vulnerabilities in Holy Shiba Pad Project. This process is used to find technical issues and security loopholes that find some common issues in the code.

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

The most common issue found in writing code on contracts that do not pose a big risk, writing on contracts is close to the standard of writing contracts in general. The low-level issues found are some arithmetic operation issues, a floating pragma is set, and out of bounds array access which the index access expression can cause an exception in case of 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.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	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
Assert Violation	SWC-110	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 Caller	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
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
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

SMART CONTRACT ANALYSIS

Started	Wednesday Jan 25 2023 14:43:05 GMT+0000 (Coordinated Universal Time)
Finished	Thursday Jan 26 2023 08:11:54 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	HOLYSHIB.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-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 194

low SEVERITY

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

Source File

- HOLYSHIB.sol

Locations

```
193   require(currentAllowance >= amount, "BEP20: transfer amount exceeds allowance");
194   _approve(sender, _msgSender(), currentAllowance - amount);
195
196   return true;
197   }
198
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 216

low SEVERITY

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

Source File

- HOLYSHIB.sol

Locations

```
215   {
216   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
217   return true;
218   }
219
220
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 241

low SEVERITY

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

Source File

- HOLYSHIB.sol

Locations

```
240   require(currentAllowance >= subtractedValue, "BEP20: decreased allowance below
zero");
241   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
242
243   return true;
244   }
245
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 270

low SEVERITY

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

Source File

- HOLYSHIB.sol

Locations

```
269   require(senderBalance >= amount, "BEP20: transfer amount exceeds balance");
270   _balances[sender] = senderBalance - amount;
271   _balances[recipient] += amount;
272
273   emit Transfer(sender, recipient, amount);
274
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 271

low SEVERITY

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

Source File

- HOLYSHIB.sol

Locations

```
270   _balances[sender] = senderBalance - amount;
271   _balances[recipient] += amount;
272
273   emit Transfer(sender, recipient, amount);
274   }
275
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 378

low SEVERITY

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

Source File

- HOLYSHIB.sol

Locations

```
377   constructor() BEP20("Holy Shiba Pad", "HOLYSHIB") {
378   _tokengeneration(msg.sender, 1e11 * 10**decimals());
379   whitelist[msg.sender] = true;
380
381   IRouter _router = IRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
382
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 408

low SEVERITY

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

Source File

- HOLYSHIB.sol

Locations

```
407   require(currentAllowance >= amount, "BEP20: transfer amount exceeds allowance");
408   _approve(sender, _msgSender(), currentAllowance - amount);
409
410   return true;
411   }
412
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 418

low SEVERITY

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

Source File

- HOLYSHIB.sol

Locations

```
417   {
418   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
419   return true;
420   }
421
422
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 429

low SEVERITY

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

Source File

- HOLYSHIB.sol

Locations

```
428   require(currentAllowance >= subtractedValue, "BEP20: decreased allowance below
zero");
429   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
430
431   return true;
432   }
433
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 464

low SEVERITY

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

Source File

- HOLYSHIB.sol

Locations

```
463   function bulkWhitelist(address[] memory accounts, bool state) external onlyOwner {
464   for (uint256 i = 0; i < accounts.length; i++) {
465   whitelist[accounts[i]] = state;
466   }
467   }
468
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 7

low SEVERITY

The current pragma Solidity directive is ""^0.8.17"". 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

- HOLYSHIB.sol

Locations

```
6
7   pragma solidity ^0.8.17;
8
9   abstract contract Context {
10   function _msgSender() internal view virtual returns (address) {
11
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 465

low SEVERITY

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

Source File

- HOLYSHIB.sol

Locations

```
464   for (uint256 i = 0; i < accounts.length; i++) {
465   whitelist[accounts[i]] = state;
466   }
467   }
468
469
```

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.