ZakumiFi

Smart Contract Audit Report

SYSFIXED

28 Oct 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
ZakumiFi	ZakumiFi	Binance Smart Chain

Addresses

Contract address	0x2efdff1e566202f82e774bb7add18c56cbb9427d
Contract deployer address	0xC18A15C32784a81FbdF45E5C286716Dabb970c7d

Project Website

https://zakumi.io/

Codebase

https://bscscan.com/address/0x2efdff1e566202f82e774bb7add18c56cbb9427d#code

SUMMARY

ZakumiFi is a DeFi Layer 1 ecosystem that supports users' comprehensive access to Dapps, NFT World, GameFi projects, Staking, Yield farming services, swap, and wallet protocols in the most straightforward way.

Contract Summary

Documentation Quality

ZakumiFi 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 ZakumiFi 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 132, 142, 150, 169, 171, 183, 184, 198, 200, 484, 484, 485, 550, 557, 600, 600, 605, 605, 629, 629, 631, 642, 643, 658, 686, 689, 689, 694, 694 and 702.
- 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 648, 649, 676 and 677.

CONCLUSION

We have audited the ZakumiFi project released on October 2022 to discover issues and identify potential security vulnerabilities in ZakumiFi 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 issues found in the ZakumiFi 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 and out-of-bounds array access. The index access expression can cause an exception in case 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	Thursday Oct 27 2022 01:39:54 GMT+0000 (Coordinated Universal Time)
Finished	Friday Oct 28 2022 05:51:20 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	ZakumiFi.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-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-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 132

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
131   unchecked {
132   _approve(sender, _msgSender(), currentAllowance - amount);
133   }
134   }
135
136
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 142

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
141   function increaseAllowance(address spender, uint256 addedValue) public virtual
returns (bool) {
142   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
143   return true;
144   }
145
146
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 150

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
149   unchecked {
150   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
151   }
152
153   return true;
154
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 169

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
168   unchecked {
169   _balances[sender] = senderBalance - amount;
170   }
171   _balances[recipient] += amount;
172
173
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 171

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
170   }
171   _balances[recipient] += amount;
172
173   emit Transfer(sender, recipient, amount);
174
175
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 183

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
182
183   _totalSupply += amount;
184   _balances[account] += amount;
185   emit Transfer(address(0), account, amount);
186
187
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 184

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
183   _totalSupply += amount;
184   _balances[account] += amount;
185   emit Transfer(address(0), account, amount);
186
187   _afterTokenTransfer(address(0), account, amount);
188
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 198

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
197   unchecked {
198   _balances[account] = accountBalance - amount;
199   }
200   _totalSupply -= amount;
201
202
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 200

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
199   }
200   _totalSupply -= amount;
201
202   emit Transfer(account, address(0), amount);
203
204
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 484

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
483
484   _mint(owner(), 6e7 * (10 ** 18));
485   swapTokensAtAmount = totalSupply() / 5000;
486   }
487
488
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 484

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
483
484   _mint(owner(), 6e7 * (10 ** 18));
485   swapTokensAtAmount = totalSupply() / 5000;
486   }
487
488
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 485

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
484   _mint(owner(), 6e7 * (10 ** 18));
485   swapTokensAtAmount = totalSupply() / 5000;
486   }
487
488   receive() external payable {
489
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 550

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
549   function updateFeeShares(uint256 _marketingFeeShare, uint256 _liquidityFeeShare)
external onlyOwner {
550   require(_marketingFeeShare + _liquidityFeeShare == 100, "Fee shares must add up to
100");
551   marketingShare = _marketingFeeShare;
552   liquidityShare = _liquidityFeeShare;
553   emit FeeSharesUpdated(marketingShare, liquidityShare);
554
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 557

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
556   function updateMarketingWalletShares(uint256 _marketingShare1, uint256
_marketingShare2) external onlyOwner {
557   require(_marketingShare1 + _marketingShare2 == 100, "Marketing fee shares must add
up to 100");
558   marketingShare1 = _marketingShare1;
559   marketingShare2 = _marketingShare2;
560   }
561
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 600

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
599   if(liquidityShare > 0) {
600   uint256 liquidityTokens = contractTokenBalance * liquidityShare / 100;
601   swapAndLiquify(liquidityTokens);
602   }
603
604
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 600

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
599   if(liquidityShare > 0) {
600   uint256 liquidityTokens = contractTokenBalance * liquidityShare / 100;
601   swapAndLiquify(liquidityTokens);
602   }
603
604
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 605

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
604   if(marketingShare > 0) {
605   uint256 marketingTokens = contractTokenBalance * marketingShare / 100;
606   swapAndSendMarketing(marketingTokens);
607   }
608
609
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 605

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
604   if(marketingShare > 0) {
605   uint256 marketingTokens = contractTokenBalance * marketingShare / 100;
606   swapAndSendMarketing(marketingTokens);
607   }
608
609
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 629

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
628   }
629   uint256 fees = amount * _totalFees / 100;
630
631   amount = amount - fees;
632
633
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 629

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
628   }
629   uint256 fees = amount * _totalFees / 100;
630
631   amount = amount - fees;
632
633
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 631

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
630
631   amount = amount - fees;
632
633   super._transfer(from, address(this), fees);
634   }
635
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 642

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
641   function swapAndLiquify(uint256 tokens) private {
642   uint256 half = tokens / 2;
643   uint256 otherHalf = tokens - half;
644
645   uint256 initialBalance = address(this).balance;
646
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 643

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
642   uint256 half = tokens / 2;
643   uint256 otherHalf = tokens - half;
644
645   uint256 initialBalance = address(this).balance;
646
647
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 658

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
657
658   uint256 newBalance = address(this).balance - initialBalance;
659
660   uniswapV2Router.addLiquidityETH{value: newBalance}(
661   address(this),
662
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 686

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
685
686   uint256 newBalance = address(this).balance - initialBalance;
687
688   if (marketingShare1 > 0) {
689   uint256 marketing1BNB = newBalance * marketingShare1 / 100;
690
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 689

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
688   if (marketingShare1 > 0) {
689   uint256 marketing1BNB = newBalance * marketingShare1 / 100;
690   sendBNB(payable(marketingWallet1), marketing1BNB);
691   }
692
693
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 689

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
688   if (marketingShare1 > 0) {
689   uint256 marketing1BNB = newBalance * marketingShare1 / 100;
690   sendBNB(payable(marketingWallet1), marketing1BNB);
691   }
692
693
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 694

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
693   if (marketingShare2 > 0) {
694   uint256 marketing2BNB = newBalance * marketingShare2 / 100;
695   sendBNB(payable(marketingWallet2), marketing2BNB);
696   }
697
698
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 694

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
693   if (marketingShare2 > 0) {
694   uint256 marketing2BNB = newBalance * marketingShare2 / 100;
695   sendBNB(payable(marketingWallet2), marketing2BNB);
696   }
697
698
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 702

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
701   function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner{
702   require(newAmount > totalSupply() / 100000, "SwapTokensAtAmount must be greater
than 0.001% of total supply");
703   swapTokensAtAmount = newAmount;
704   }
705   }
706
```

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

- ZakumiFi.sol

Locations

```
647   address[] memory path = new address[](2);
648   path[0] = address(this);
649   path[1] = uniswapV2Router.WETH();
650
651   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
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

- ZakumiFi.sol

Locations

```
648   path[0] = address(this);
649   path[1] = uniswapV2Router.WETH();
650
651   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
652   half,
653
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 676

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
675   address[] memory path = new address[](2);
676   path[0] = address(this);
677   path[1] = uniswapV2Router.WETH();
678
679   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
680
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 677

low SEVERITY

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

Source File

- ZakumiFi.sol

Locations

```
676   path[0] = address(this);
677   path[1] = uniswapV2Router.WETH();
678
679   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
680   tokenAmount,
681
```

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.