

PriceAl Smart Contract Audit Report



27 Jan 2023



TABLE OF CONTENTS

Audited Details

- Audited Project
- Blockchain
- Addresses
- Project Website
- Codebase

Summary

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

Conclusion

Audit Results

Smart Contract Analysis

- Detected Vulnerabilities

Disclaimer

About Us



AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain	
PriceAl	PRICE	Binance Smart Chain	

Addresses

Contract address 0xb69edbD0527F448b650676C7085E15a7422791c5	
Contract deployer address	0x8134b687be5752eFF8361B663030420D47648bfF

Project Website

https://priceai.ai/

Codebase

https://bscscan.com/address/0xb69edbD0527F448b650676C7085E15a7422791c5#code



SUMMARY

PriceAI is a revolutionary platform that uses advanced AI and zkSNARKs technology for privacy-preserving DeFi price feeds. Our platform features include advanced data privacy and control and complete financial autonomy. Join the revolution in decentralized finance.

Contract Summary

Documentation Quality

PriceAI 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.

• Standart solidity basecode and rules are already followed with PriceAI with the discovery of several low issues.

Test Coverage

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

Audit Findings Summary

- SWC-100 SWC-108 | Explicitly define visibility for all state variables on lines 191.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 164, 242, 314, 316, 324, 325, 336, 337, 342, 344, 348, 375, 378, 379, 392, 398 and 412.
- SWC-110 | It is recommended to use use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 354 and 355.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 259.



CONCLUSION

We have audited the PriceAI project released on January 2023 to discover issues and identify potential security vulnerabilities in PriceAI 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 PriceAI 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-level issues found are some arithmetic operation issues, a state variable visibility is not set, the use of "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 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.	ISSUE FOUND
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operationsISSUEshould be safe from overflows and underflows.FOUN	
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	
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.	
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	
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.	
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	
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.	
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	Thursday Jan 26 2023 23:56:47 GMT+0000 (Coordinated Universal Time)		
Finished	Friday Jan 27 2023 19:43:27 GMT+0000 (Coordinated Universal Time)		
Mode	Standard		
Main Source File	PRICE.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

SYSFIXED

SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	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 164

Iow SEVERITY

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

Source File

- PRICE.sol

Locations

163 uint256 private swapThreshold; 164 uint256 constant public _totalSupply = 5e8 * 10**18; 165 uint256 constant public transferfee = 0; 166 uint256 constant public fee_denominator = 10000; 167 168



SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 242

Iow SEVERITY

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

Source File

- PRICE.sol

```
241 if (_allowances[sender][msg.sender] != type(uint256).max) {
242 __allowances[sender][msg.sender] -= amount;
243 }
244
245 return _transfer(sender, recipient, amount);
246
```



SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 314

Iow SEVERITY

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

Source File

- PRICE.sol

```
313
314 balance[from] -= amount;
315 uint256 amountAfterFee = (takeFee) ? takeTaxes(from, is_buy(from, to),
is_sell(from, to), amount) : amount;
316 balance[to] += amountAfterFee;
317 emit Transfer(from, to, amountAfterFee);
318
```



SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 316

Iow SEVERITY

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

Source File

- PRICE.sol

```
315 uint256 amountAfterFee = (takeFee) ? takeTaxes(from, is_buy(from, to),
is_sell(from, to), amount) : amount;
316 balance[to] += amountAfterFee;
317 emit Transfer(from, to, amountAfterFee);
318
319 return true;
320
```



SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 324

Iow SEVERITY

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

Source File

- PRICE.sol

```
323
324 balance[from] -= amount;
325 balance[to] += amount;
326 return true;
327 }
328
```



SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 325

Iow SEVERITY

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

Source File

- PRICE.sol

```
324 balance[from] -= amount;
325 balance[to] += amount;
326 return true;
327 }
328
329
```



SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 336

Iow SEVERITY

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

Source File

- PRICE.sol

```
335 uint256 fee;
336 if (isbuy) fee = buyTaxes.marketing + buyTaxes.rewards;
337 else if (issell) fee = sellTaxes.marketing + sellTaxes.rewards;
338 else fee = transferfee;
339
340
```



SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 337

Iow SEVERITY

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

Source File

- PRICE.sol

```
336 if (isbuy) fee = buyTaxes.marketing + buyTaxes.rewards;
337 else if (issell) fee = sellTaxes.marketing + sellTaxes.rewards;
338 else fee = transferfee;
339
340 if (fee == 0) return amount;
341
```



SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 342

Iow SEVERITY

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

Source File

- PRICE.sol

```
341
342 uint256 feeAmount = amount * fee / fee_denominator;
343 if (feeAmount > 0) {
344 balance[address(this)] += feeAmount;
345 emit Transfer(from, address(this), feeAmount);
346
```



SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 344

Iow SEVERITY

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

Source File

- PRICE.sol

```
343 if (feeAmount > 0) {
344 balance[address(this)] += feeAmount;
345 emit Transfer(from, address(this), feeAmount);
346
347 }
348
```



SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 348

Iow SEVERITY

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

Source File

- PRICE.sol

```
347 }
348 return amount - feeAmount;
349 }
350
351 function internalSwap(uint256 contractTokenBalance) internal inSwapFlag {
352
```



SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 375

Iow SEVERITY

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

Source File

- PRICE.sol

```
374 uint256 rewardsBNB = 0;
375 uint256 totalTax = sellTaxes.marketing + sellTaxes.rewards;
376
377 if (totalTax > 0) {
378 marketingBNB = bnbInContract * sellTaxes.marketing / totalTax;
379
```



SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 378

Iow SEVERITY

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

Source File

- PRICE.sol

```
377 if (totalTax > 0) {
378 marketingBNB = bnbInContract * sellTaxes.marketing / totalTax;
379 rewardsBNB = bnbInContract - marketingBNB;
380 }
381
382
```



SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 379

Iow SEVERITY

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

Source File

- PRICE.sol

```
378 marketingBNB = bnbInContract * sellTaxes.marketing / totalTax;
379 rewardsBNB = bnbInContract - marketingBNB;
380 }
381
382 bool success;
383
```



SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 392

Iow SEVERITY

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

Source File

- PRICE.sol

```
391 function updateBuyFeeAmount(uint256 _marketingFee, uint256 _rewardsFee) external
onlyOwner {
392 require((_marketingFee + _rewardsFee) < maxBuyFee, "Total should be less
maxBuyFee");
393 buyTaxes.marketing = _marketingFee;
394 buyTaxes.rewards = _rewardsFee;
395 }
396</pre>
```



SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 398

Iow SEVERITY

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

Source File

- PRICE.sol

```
397 function updateSellFeeAmount(uint256 _marketingFee, uint256 _rewardsFee) external
onlyOwner {
398 require((_marketingFee + _rewardsFee) < maxSellFee, "Total should be less
maxSellFee");
399 sellTaxes.marketing = _marketingFee;
400 sellTaxes.rewards = _rewardsFee;
401 }
402</pre>
```





SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 412

Iow SEVERITY

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

Source File

- PRICE.sol

```
411 require(!isTradingEnabled, "Trading already enabled");
412 swapThreshold = (balanceOf(lpPair)) / 100000;
413 isTradingEnabled = true;
414 }
415
416
```



SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 191

Iow SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "inSwap" is internal. Other possible visibility settings are public and private.

Source File

- PRICE.sol

```
190 bool public LiquidityAdded = false;
191 bool inSwap;
192
193 modifier inSwapFlag {
194 inSwap = true;
195
```



SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 259

Iow SEVERITY

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

Source File

- PRICE.sol

```
258 bool isLimited = ins != owner()
259 && out != owner() && tx.origin != owner() // any transaction with no direct
interaction from owner will be accepted
260 && msg.sender != owner()
261 && !liquidityAdd[ins] && !liquidityAdd[out] && out != DEAD && out != address(0) &&
out != address(this);
262 return isLimited;
263
```





SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 354

Iow SEVERITY

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

Source File

- PRICE.sol

```
353 address[] memory path = new address[](2);
354 path[0] = address(this);
355 path[1] = swapRouter.WETH();
356
357 if (_allowances[address(this)][address(swapRouter)] != type(uint256).max) {
358
```



SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 355

Iow SEVERITY

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

Source File

- PRICE.sol

```
354 path[0] = address(this);
355 path[1] = swapRouter.WETH();
356
357 if (_allowances[address(this)][address(swapRouter)] != type(uint256).max) {
358 _allowances[address(this)][address(swapRouter)] = type(uint256).max;
359
```



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.