



CONTRACT  
CHECKER

Blockchain Solutions



<https://t.me/contractchecker>

[contact@contractchecker.app](mailto:contact@contractchecker.app)

[contractchecker.app](https://contractchecker.app)

Anywhere on the Blockchain

**Date:** 25.07.2022

# Smart Contract Security Audit

DIOLAUNCH TOKEN



*Harry K*

**Harry Kedelman**  
General Manager

## Audit Result

- 🕒 DIOLAUNCH TOKEN has **PASSED** the smart contract audit with below listed privileges

(Other unknown security vulnerabilities are not included in the audit responsibility scope)

Audit Result:	<b>PASSED</b>
Ownership:	Not renounced yet
KYC Verification:	<b>KYC Verified by ContractChecker</b>
Audit Date:	July 25, 2022
Audit Team:	CONTRACTCHECKER

## Findings \_ Privileges of Ownership

- ⚠️ Auto liquidity is going to an externally owned account
- ⚠️ Owner can exclude accounts from rewards
- ⚠️ Owner can exclude an account from paying fees
- ⚠️ Owner can change the fees but with limit of 25% at max
- ⚠️ Owner can change swap settings

## Important Notice for Investors

As Contract Checker team we are mainly auditing the contract code to find out how it will be functioning, and risks which are hidden in the code if any.

There are many factors must be taken into consideration before investing to a project, like: ownership status, project team approach, marketing, general market condition, liquidity, token holdings etc.

Investors must always do their own research and manage their risk considering different factors which can affect the success of a project.

## Table of Contents

<b>Audit Result</b> .....	1
<b>Findings _ Privileges of Ownership</b> .....	1
<b>Important Notice for Investors</b> .....	1
<b>SUMMARY</b> .....	3
<b>Project Summary</b> .....	3
<b>OVERVIEW</b> .....	4
<b>Auditing Approach and Applied Methodologies</b> .....	4
<b>Security</b> .....	4
<b>Sound Architecture</b> .....	4
<b>Code Correctness and Quality</b> .....	4
<b>Risk Classification</b> .....	5
<b>High level vulnerability</b> .....	5
<b>Medium level vulnerability</b> .....	5
<b>Low level vulnerability</b> .....	5
<b>Vulnerability Checklist</b> .....	5
<b>Manual Audit:</b> .....	6
<b>Smart Contract SWC Attack Test</b> .....	6
➤ <b>SWC-103: A floating pragma is set</b> .....	7
➤ <b>SWC-108: State variable visibility is not set</b> .....	7
<b>Automated Audit</b> .....	7
<b>Remix Compiler Warnings</b> .....	7
<b>Disclaimer</b> .....	8

## SUMMARY

CONTRACTCHECKER received an application for smart contract security audit of DIOLAUNCH TOKEN on July 24, 2022, from the project team to discover if any vulnerability in the source codes of the DIOLAUNCH TOKEN as well as any contract dependencies. Detailed test has been performed using Static Analysis and Manual Review techniques.

The auditing process focuses to the following considerations with collaboration of an expert team

- Functionality test of the Smart Contract to determine if proper logic has been followed throughout the whole process.
- Manually detailed examination of the code line by line by experts.
- Live test by multiple clients using Testnet.
- Analysing failure preparations to check how the Smart Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analysing the security of the on-chain data.

## Project Summary

Token Name	DIOLAUNCH TOKEN
Web Site	<a href="https://diolaunch.io/">https://diolaunch.io/</a>
Twitter	<a href="https://twitter.com/Diolaunch">https://twitter.com/Diolaunch</a>
Telegram	<a href="https://t.me/diolaunch">https://t.me/diolaunch</a>
Github	<a href="https://github.com/Diolaunch">https://github.com/Diolaunch</a>
Instagram	<a href="https://www.instagram.com/diolaunch/">https://www.instagram.com/diolaunch/</a>
Reddit	<a href="https://www.reddit.com/r/diolaunch/">https://www.reddit.com/r/diolaunch/</a>
Platform	Binance Smart Chain
Token Type	BEP20
Language	Solidity
Platforms & Tools	Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Mythril, Contract Library
Contract Address	0xA4bf0BaeF9e8ea3B4a1cE694428d61546Fb4EBbb
Contract Link	<a href="https://bscscan.com/token/0xa4bf0baef9e8ea3b4a1ce694428d61546fb4ebbb">https://bscscan.com/token/0xa4bf0baef9e8ea3b4a1ce694428d61546fb4ebbb</a>
Test Link	<a href="https://testnet.bscscan.com/address/0x6f26456100c61d3243fc6602ce1703221f416f73">https://testnet.bscscan.com/address/0x6f26456100c61d3243fc6602ce1703221f416f73</a>

## OVERVIEW

This Audit Report mainly focuses on overall security of DIOLAUNCH TOKEN smart contract. Contract Checker team scanned the contract and assessed overall system architecture and the smart contract codebase against vulnerabilities, exploitations, hacks, and back-doors to ensure its reliability and correctness.

## Auditing Approach and Applied Methodologies

Contract Checker team has performed rigorous test procedures of the project

- Code design patterns analysis in which smart contract architecture is reviewed to ensure it is structured according to industry standards and safe use of third-party smart contracts and libraries.
- Line-by-line inspection of the Smart Contract to find any potential vulnerability like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.
- Unit testing Phase, we coded/conducted custom unit tests written for each function in the contract to verify that each function works as expected.
- Automated Test performed with our in-house developed tools to identify vulnerabilities and security flaws of the Smart Contract.

The focus of the audit was to verify that the Smart Contract System is secure, resilient, and working according to the specifications. The audit activities can be grouped in the following three categories:

### Security

Identifying security related issues within each contract and the system of contract.

### Sound Architecture

Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.

### Code Correctness and Quality

A full review of the contract source code. The primary areas of focus include:

- Accuracy
- Readability
- Sections of code with high complexity
- Quantity and quality of test coverage

## Risk Classification

Vulnerabilities are classified in 3 main levels as below based on possible effect to the contract.

### High level vulnerability

Vulnerabilities on this level must be fixed immediately as they might lead to fund and data loss and open to manipulation. Any High-level finding will be highlighted with **RED** text

### Medium level vulnerability

Vulnerabilities on this level also important to fix as they have potential risk of future exploit and manipulation. Any Medium-level finding will be highlighted with **ORANGE** text

### Low level vulnerability

Vulnerabilities on this level are minor and may not affect the smart contract execution. Any Low-level finding will be highlighted with **BLUE** text

## Vulnerability Checklist

Nº	Description.	Result
1	Compiler warnings.	Passed
2	Race conditions and Re-entrancy. Cross-function race conditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed
10	Methods execution permissions.	Passed
11	Economy model.	Passed
12	The impact of the exchange rate on the logic.	Passed
13	Private user data leaks.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Uninitialized storage pointers.	Passed
17	Arithmetic accuracy.	Passed
18	Design Logic.	Passed
19	Cross-function race conditions.	Passed
20	Safe Zeppelin module.	Passed
21	Fallback function security.	Passed



## Manual Audit:

For this section the code was tested/read line by line by our developers. Additionally, Remix IDE's JavaScript VM and Kovan networks used to test the contract functionality.

## Smart Contract SWC Attack Test

SWC ID	Description	Test Result
SWC-100	Function Visibility	Passed
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	Passed
SWC-103	FloatingPragma	LOW
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed
SWC-107	Re-entrancy	Passed
SWC-108	State Variable Default Visibility	LOW
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	Passed
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed
SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-126	Insufficient Gas Griefing	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions with Multiple Variable Length Arguments	Passed
SWC-134	Message call with hardcoded gas amount	Passed
SWC-135	Code With No Effects (Irrelevant/Dead Code)	Passed
SWC-136	Unencrypted Private Data On-Chain	Passed

## ➤ SWC-103: A floating pragma is set

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.

```
5  pragma solidity >=0.5.0;
6
7  interface IUniswapV2Factory {
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29  pragma solidity >=0.6.2;
30
31
32
33
34
35
36
37
38
39
40
41  interface IUniswapV2Router01 {
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85  pragma solidity >=0.6.2;
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105  pragma solidity ^0.8.1;
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137  pragma solidity >=0.6.2;
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183  // OpenZeppelin Contracts (last updated v4.7.0) (utils/Address.sol)
184
185  pragma solidity ^0.8.1;
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408  // OpenZeppelin Contracts (last updated v4.6.0) (utils/math/SafeMath.sol)
409
410  pragma solidity ^0.8.0;
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638  // OpenZeppelin Contracts v4.4.1 (utils/Context.sol)
639
640  pragma solidity ^0.8.0;
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752  pragma solidity ^0.8.0;
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884  address public _teamAddress = 0x09E626Eb98bCF39d17163C0188F3c46135218491;
885
886
887
888  bool inSwapAndLiquify;
889
890  bool public swapAndLiquifyEnabled;
```

## ➤ SWC-108: State variable visibility is not set

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

```
881  address public _teamAddress = 0x09E626Eb98bCF39d17163C0188F3c46135218491;
882
883  bool inSwapAndLiquify;
884  bool public swapAndLiquifyEnabled;
```

## Automated Audit

### Remix Compiler Warnings

It throws warnings by Solidity's compiler. No issues found.



## Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at 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. 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 based on 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.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy all copies of this report downloaded and/or printed by you. 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 ContractChecker and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (ContractChecker) owe no duty of care towards you or any other person, nor does ContractChecker make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and ContractChecker hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, ContractChecker hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against ContractChecker, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed. If you have any doubt about the Genuity for this document, please check QR code: