

Smart Contract Audit Report for ConstitutionDao (PEOPLE)



TRUSTLOOK

This audit report
donated by

[aWSB ConstitutionDAO \(\\$PEOPLE\) Community](#)

Version 1.0

Trustlook Blockchain Labs

Email: bd@trustlook.com

Project Overview

| | |
|--------------------------|-------------------------|
| Project Name | ConstitutionDao(PEOPLE) |
| Contract codebase | N/A |
| Platform | Ethereum |
| Language | Solidity |
| Submission Times | 2021.12.10 |

Report Overview

| | |
|----------------------|---------------------------|
| Report ID | TBL_20211210_00 |
| Version | 1.0 |
| Reviewer | Trustlook Blockchain Labs |
| Starting Time | 2021.12.10 |
| Finished Time | 2021.12.20 |

Disclaimer

Trustlook audit reports do not provide any warranties or guarantees on the vulnerability-free nature of the given smart contracts, nor do they provide any indication of legal compliance. The Trustlook audit process is aiming to reduce the high level risks possibly implemented in the smart contracts before the issuance of audit reports. Trustlook audit reports can be used to improve the code quality of smart contracts and are not able to detect any security issues of smart contracts that will occur in the future. Trustlook audit reports should not be considered as financial investment advice.

About Trustlook Blockchain Labs

Trustlook Blockchain Labs is a leading blockchain security team with a goal of security and vulnerability research on current blockchain ecosystems by offering industry-leading smart contracts auditing services. Please contact us for more information at (<https://www.trustlook.com/services/smart.html>) or Email (bd@trustlook.com)

The Trustlook blockchain laboratory has established a complete system test environment and methods.

| | |
|-------------------|--|
| Black-box Testing | The tester has no knowledge of the system being attacked. The goal is to simulate an external hacking or cyber warfare attack. |
| White-box Testing | Based on the level of the source code, test the control flow, data flow, nodes, SDK etc. Try to find out the vulnerabilities and bugs. |
| Gray-box Testing | Use Trustlook customized script tools to do the security testing of code modules, search for the defects if any due to improper structure or improper usage of applications. |

Introduction

By reviewing the implementation of ConstitutionDao's smart contracts, this audit report has been prepared to discover potential issues and vulnerabilities of their source code. We outline in the report about our approach to evaluate the potential security risks. Advice to further improve the quality of security or performance is also given in the report.

About ConstitutionDao

ConstitutionDao is a decentralized autonomous organization (DAO) formed in November 2021. PEOPLE is an ERC20 token created in the Juicebox platform by ConstitutionDao.

About Methodology

To evaluate the potential vulnerabilities or issues, we go through a checklist of well-known smart contracts related security issues using automatic verification tools and manual review. To discover potential logic weaknesses or project specific implementations, we thoroughly discussed with the team to understand the business model and reduce the risk of unknown vulnerabilities. For any discovered issue, we might test it on our private network to reproduce the issue to prove our findings.

The checklist of items is shown in the following table:

| Category | Type ID | Name | Description |
|----------------------|---------|----------------------|--|
| Coding Specification | CS-01 | ERC standards | The contract is using ERC standards. |
| | CS-02 | Compiler Version | The compiler version should be specified. |
| | CS-03 | Constructor Mismatch | The constructor syntax is changed with Solidity versions. Need extra attention to make the constructor function right. |
| | CS-04 | Return standard | Following the ERC20 specification, the transfer and approve |

| | | | |
|---------------------|-------|---------------------------------|--|
| | | | functions should return a bool value, and a return value code needs to be added. |
| | CS-05 | Address(0) validation | It is recommended to add the verification of <code>require(!_to!=address(0))</code> to effectively avoid unnecessary loss caused by user misuse or unknown errors. |
| | CS-06 | Unused Variable | Unused variables should be removed. |
| | CS-07 | Untrusted Libraries | The contract should avoid using untrusted libraries, or the libraries need to be thoroughly audited too. |
| | CS-08 | Event Standard | Define and use Event appropriately |
| | CS-09 | Safe Transfer | Using transfer to send funds instead of send. |
| | CS-10 | Gas consumption | Optimize the code for better gas consumption. |
| | CS-11 | Deprecated uses | Avoid using deprecated functions. |
| | CS-12 | Sanity Checks | Sanity checks when setting key parameters in the system |
| Coding Security | SE-01 | Integer overflows | Integer overflow or underflow issues. |
| | SE-02 | Reentrancy | Avoid using calls to trade in smart contracts to avoid reentrancy vulnerability. |
| | SE-03 | Transaction Ordering Dependence | Avoid transaction ordering dependence vulnerability. |
| | SE-04 | Tx.origin usage | Avoid using tx.origin for authentication. |
| | SE-05 | Fake recharge | The judgment of the balance and the transfer amount needs to use the "require function". |
| | SE-06 | Replay | If the contract involves the demands for entrusted management, attention should be paid to the non-reusability of verification to avoid replay attacks. |
| | SE-07 | External call checks | For external contracts, pull instead of push is preferred. |
| | SE-08 | Weak random | The method of generating random numbers on smart contracts requires more considerations. |
| Additional Security | AS-01 | Access control | Well defined access control for functions. |
| | AS-02 | Authentication management | The authentication management is well defined. |
| | AS-03 | Semantic Consistency | Semantics are consistent. |
| | AS-04 | Functionality checks | The functionality is well implemented. |

| | | | |
|--|-------|-----------------------|--|
| | AS-05 | Business logic review | The business model logic is implemented correctly. |
|--|-------|-----------------------|--|

The severity level of the issues are described in the following table:

| Severity | Description |
|---------------|--|
| Critical | The issue will result in asset loss or data manipulations. |
| High | The issue will seriously affect the correctness of the business model. |
| Medium | The issue is still important to fix but not practical to exploit. |
| Low | The issue is mostly related to outdated, unused code snippets. |
| Informational | This issue is mostly related to code style, informational statements and is not mandatory to be fixed. |

Audit Results

Here are the audit results of the smart contracts. Since the ConstitutionDao (PEOPLE) project was created by Juicebox platform. All related smart contracts from Juicebox were reviewed and covered in this report. Per ConstitutionDao's request, this report focuses on the influence of the renouncement of ownership of the project to address(0) and ensure the PEOPLE token will not be manipulated by Juicebox platform anymore in future.

Scope

Following files have been scanned by our internal audit tool and manually reviewed and tested by our team:

| File names | Source |
|-----------------------|---|
| Tickets.sol | https://etherscan.io/token/0x7a58c0be72be218b41c608b7fe7c5bb630736c71 |
| TicketBooth.sol | https://etherscan.io/address/0xee2ebccb7cdb34a8a822b589f9e8427c24351bfc |
| TerminalV1.sol | https://etherscan.io/address/0xd569D3CCE55b71a8a3f3C418c329A66e5f714431 |
| TerminalDirectory.sol | https://etherscan.io/address/0x46c9999a2edcd5aa177ed7e8af90c68b7d75ba46 |
| Projects.sol | https://etherscan.io/address/0x9b5a4053ffbb11ca9cd858aeee43cc95ab435418 |

Summary

| Issue ID | Severity | Location | Type ID | Status |
|-------------|----------|---|---------|--------|
| TBL_SCA_001 | Info | Tickets.sol TicketBooth.sol TerminalV1.sol TerminalDirectory.sol Projects.sol | AS-05 | open |

Details

- ID: TBL_SCA-001
- Severity: Info
- Type: AS-05 (Business logic review)
- Description:

The PEOPLE token is a ERC20 token created by Juicebox platform by following code in TicketBooth contract with address 0xee2eBCcB7CDb34a8A822b589F9E8427C24351bfc:

```
149 // Create the contract in this TerminalV1 contract in order to have mint and burn privileges.
150 // Prepend the strings with standards.
151 ticketsOf[_projectId] = new Tickets(_name, _symbol);
152
153 emit Issue(_projectId, _name, _symbol, msg.sender);
154 }
```

The PEOPLE (Tickets) token can be accessed by etherscan.io link as follows:

<https://etherscan.io/address/0x7a58c0be72be218b41c608b7fe7c5bb630736c71>

We can confirm that the owner of this contract is The TicketBooth contract by query the owner function of the PEOPLE contract using the “Read Contract” button:

7. owner

Returns the address of the current owner.

[0xee2ebccb7cdb34a8a822b589f9e8427c24351bfc](#) address

Since the PEOPLE contract is *Ownable*, which means the current owner has the right to renounce or transfer the ownership to another address. However, by reviewing the whole source code of TicketBooth, there are no such operations that can be used to modify the ownership. Therefore, we have to ensure the TicketBooth contract will not abuse the ownership of PEOPLE tokens.

Note that two privileged functions for owner of the PEOPLE contract are *print()* and *redeem()*:

```
18     function print(address _account, uint256 _amount)
19         external
20         override
21         onlyOwner
22     {
23         return _mint(_account, _amount);
24     }
25
26     function redeem(address _account, uint256 _amount)
27         external
28         override
29         onlyOwner
30     {
31         return _burn(_account, _amount);
32     }
```

We have to ensure how these functions are used in the TicketBooth contract. It is noted that function *print()* can only be called by *print()* and *unstake()* functions in TicketBooth. Similarly, function *redeem()* can only be called by *redeem()* and *stake()* functions in TicketBooth.

It is observed that the *stake()* and *unstake()* functions are fully under the control of PEOPLE token holders:

```
294     @notice
295     Stakes ERC20 tickets by burning their supply and creating an internal staked version.
296
297     @dev
298     Only a ticket holder or an operator can stake its tickets.
299
300     @param _holder The owner of the tickets to stake.
301     @param _projectId The ID of the project whos tickets are being staked.
302     @param _amount The amount of tickets to stake.
303     */
304     function stake(
305         address _holder,
306         uint256 _projectId,
307         uint256 _amount
308     )
309     external
310     override
311     requirePermissionAllowingWildcardDomain(
312         _holder,
313         _projectId,
314         Operations.Stake
315     )
```

```
351  /**
352   @notice
353   Unstakes internal tickets by creating and distributing ERC20 tickets.
354
355   @dev
356   Only a ticket holder or an operator can unstake its tickets.
357
358   @param _holder The owner of the tickets to unstake.
359   @param _projectId The ID of the project whos tickets are being unstaked.
360   @param _amount The amount of tickets to unstake.
361  */
362  function unstake(
363      address _holder,
364      uint256 _projectId,
365      uint256 _amount
366  )
367      external
368      override
369      requirePermissionAllowingWildcardDomain(
370          _holder,
371          _projectId,
372          Operations.Unstake
373      )
```

Both functions have a modifier `requirePermissionAllowingWildcardDomain()`, which requires either the msg.sender is the holder of PEOPLE token (account holder operates on his own tokens) or the related permission was granted by the holder before (account holder grants someone else permission to operate his tokens).

```
25  modifier requirePermissionAllowingWildcardDomain(
26      address _account,
27      uint256 _domain,
28      uint256 _index
29  ) {
30      require(
31          msg.sender == _account ||
32          operatorStore.hasPermission(
33              msg.sender,
34              _account,
35              _domain,
36              _index
37          ) ||
38          operatorStore.hasPermission(msg.sender, _account, 0, _index),
39          "Operatable: UNAUTHORIZED"
40      );
41      -;
42  }
```

Note that the permissions granted on any domain with index are fully set up by the account holder. Therefore, any actions operated by these functions fully depend on the token holders. Consequently, `stake()` and `unstake()` functions in TicketBooth bring no security concerns to abuse the `print()` and `redeem()` functions in the PEOPLE contract.

Before we go further to analyze the functions `print()` and `redeem()` in TicketBooth, we first need to understand the meaning of the project concept in Juicebox. Every project inside Juicebox is presented by a ERC721 token. A project can be created in the TerminalV1 contract, and the

```
>> address: 0xb1C95AC257029D11F3f64ac67b2307A426699322
```

```

219     function redeem(
220         address _holder,
221         uint256 _projectId,
222         uint256 _amount,
223         bool _preferUnstaked
224     ) external override onlyTerminal(_projectId) {
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
409
410
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
639
640
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
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
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
11
```

```

6 ▾ abstract contract TerminalUtility is ITerminalUtility {
7 ▾     modifier onlyTerminal(uint256 _projectId) {
8         require(
9             address(directory.terminalOf(_projectId)) == msg.sender,
10            "TerminalUtility: UNAUTHORIZED"
11        );
12        -;
13    }
14

```

@ Copyright 2021 Trustlook - All rights reserved

8. terminalDirectory

The direct deposit terminals.

[0x46c9999a2edcd5aa177ed7e8af90c68b7d75ba46](https://etherscan.io/address/0x46c9999a2edcd5aa177ed7e8af90c68b7d75ba46) address

The value of *terminalDirectory* in TicketBooth has no method to change in future. By using the link <https://etherscan.io/address/0x46c9999a2edcd5aa177ed7e8af90c68b7d75ba46> we can also retrieve the terminal contract of the PEOPLE as follows:

5. terminalOf

For each project ID, the juicebox terminal that the direct payment addresses are proxies for.

<input> (uint256)

Query

↳ address

[**terminalOf(uint256)** method Response]

➤ address : [0xd569d3cce55b71a8a3f3c418c329a66e5f714431](https://etherscan.io/address/0xd569d3cce55b71a8a3f3c418c329a66e5f714431)

Note that the *projectId* of People is 36, therefore we can know the current Terminal is the TerminalV1.sol at address [0xd569d3cce55b71a8a3f3c418c329a66e5f714431](https://etherscan.io/address/0xd569d3cce55b71a8a3f3c418c329a66e5f714431).

Based on the above retrieved information, we know that the *print()* and *redeem()* in TicketBooth can only be called by the current Terminal contract. Note that the terminal contract in the *terminalDirectory* can be updated by using the function *setTerminal()*.

There are three cases where a new Terminal can be updated using *setTerminal()*:

```

111 // Either:
112 // - case 1: the current terminal hasn't been set yet and the msg sender is either the projects contract or the te
113 // - case 2: the current terminal must not yet be set, or the current terminal is setting a new terminal.
114 // - case 3: the msg sender is the owner or operator and either the current terminal hasn't been set, or the curre
115 require(
116     // case 1.
117     (_currentTerminal == ITerminal(address(0)) &&
118         (msg.sender == address(projects) ||
119             msg.sender == address(_terminal))) ||
120     // case 2.
121     msg.sender == address(_currentTerminal) ||
122     // case 3.
123     ((msg.sender == _projectOwner ||
124         operatorStore.hasPermission(
125             msg.sender,
126             _projectOwner,
127             _projectId,
128             Operations.SetTerminal
129         )) &&
130         (_currentTerminal == ITerminal(address(0)) ||
131             _currentTerminal.migrationIsAllowed(_terminal))),
132     "TerminalDirectory::setTerminal: UNAUTHORIZED"
133 );

```

Case one: when the terminal hasn't been set yet which does not apply to PEOPLE's contract.
 Case two: current terminal is setting a new terminal. Case three: current project owner is setting a new terminal. For PEOPLE's contract, terminal owner is 0xd569D3CCE55b71a8a3f3C418c329A66e5f714431(Terminal V1 contract from JuiceBox) and the project owner is 0xb1C95AC257029D11F3f64ac67b2307A426699322 (Gnosis multisig from Constitution DAO).

10. ownerOf

See {IERC721-ownerOf}.

tokenId (uint256)

36

Query

↳ address

[ownerOf(uint256) method Response]

» address : [0xb1C95AC257029D11F3f64ac67b2307A426699322](#)

In the Terminal contract, only the function migrate() calls setTerminal(). However, it requires the project owner or operator with granted permission to call this function. Therefore, only the project owner or granted operator is able to set up a new terminal for PEOPLE. By transferring the ownership to address(0), no operator can be set with any permission from address(0) and therefore none can set up a new terminal after that.

```
781     function migrate(uint256 _projectId, ITerminal _to)
782     external
783     override
784     requirePermission(
785         projects.ownerOf(_projectId),
786         _projectId,
787         Operations.Migrate
788     )
```

On the other hand, if the ConstitutionDao project kept using the current TerminalV1 contract. The usage of *print()* and *redeem()* is concerning.

The function *redeem()* in TicketBooth is only called in the *redeem()* function of the TerminalV1 contract:

```
689  /**
690   * @notice
691   * Addresses can redeem their Tickets to claim the project's overflowed ETH.
692   *
693   * @dev
694   * Only a ticket's holder or a designated operator can redeem it.
695   *
696   * @param _account The account to redeem tickets for.
697   * @param _projectId The ID of the project to which the Tickets being redeemed belong.
698   * @param _count The number of Tickets to redeem.
699   * @param _minReturnedWei The minimum amount of Wei expected in return.
700   * @param _beneficiary The address to send the ETH to.
701   * @param _preferUnstaked If the preference is to redeem tickets that have been converted to ERC-20s.
702   *
703   * @return amount The amount of ETH that the tickets were redeemed for.
704   */
705  function redeem(
706      address _account,
707      uint256 _projectId,
708      uint256 _count,
709      uint256 _minReturnedWei,
710      address payable _beneficiary,
711      bool _preferUnstaked
712  )
713  external
714  override
715  nonReentrant
716  requirePermissionAllowingWildcardDomain(
717      _account,
718      _projectId,
719      Operations.Redeem
720  )
721  returns (uint256 amount)
```

The *redeem()* function can be called by the token holder or the granted operators by the holder. The function lets the token holders redeem their Tickets to claim the project's overflowed ETH. If the community decides to keep the current Terminal contract, they need to accept the redeem logic of the *redeem()* function.

The function *print()* in TicketBooth can be accessed by several functions: *printPreminedTickets()*, *printReservedTickets()*, *_distributeToTicketMods()* and *_pay()*.

For function *printPreminedTickets()*, it can only be called by the owner of the project or the

operators it sets:

```
463     function printPreminedTickets(  
464         uint256 _projectId,  
465         uint256 _amount,  
466         uint256 _currency,  
467         address _beneficiary,  
468         string memory _memo,  
469         bool _preferUnstakedTickets  
470     )  
471     external  
472     override  
473     requirePermission(  
474         projects.ownerOf(_projectId),  
475         _projectId,  
476         Operations.PrintPreminedTickets  
477     )
```

By transferring the ownership to address(0) and with no granted operators, none can have the access to this function in future.

The private function `_distributeToTicketMods()` is only called by `printReservedTickets()`:

```
934     // --- public transactions --- //  
935  
936     /**  
937     @notice  
938     Prints all reserved tickets for a project.  
939  
940     @param _projectId The ID of the project to which the reserved tickets belong.  
941  
942     @return amount The amount of tickets that are being printed.  
943     */  
944     function printReservedTickets(uint256 _projectId)  
945     public  
946     override  
947     returns (uint256 amount)
```

The function `printReservedTickets()` is responsible for printing all reserved tickets for a project. It first calculates the reserved ticket count and distributes the tickets to pre-configured ticket mods addresses, then transfers the leftover to the owner of the project. By using the current Terminal contract, the ConstitutionDao community needs to accept the logic of this function. By transferring the ownership to address(0), the reserved token will be transferred to the new owner in address(0).

The last function that uses the `print()` in TicketBooth is the private function `_pay()`. This private function is responsible for printing new tickets (PEOPLE token for this project) based on the incoming ETH payment. However, based on current configuration, the `_reservedRate` is set as 200, which can be confirmed by the metadata in current `_fundingCycle`:

3368601600

The decimal number 3368601600 is 0xC8C8C800, the second byte 0xC8 means 200 for the reserved rate. This reserved rate is used here:

```
1184 );
```

The value of `_unreservedWeightedAmount` will always be 0 when `reservedRate` is 200. This means no more PEOPLE tokens can be printed from `_pay()` function with current configuration.

Note that the function `configure()` can be used to modify the current configuration:

```
408         returns (uint256)
```

However, this function can only be called by the owner of the project or the operators it sets. By transferring the ownership to address(0) and with no granted operators, none can have access to this function in future.

From the above analysis, we can see that by transferring the ownership of ConstitutionDao project to address(0), the Juicebox platform will not be able to mint or burn PEOPLE tokens by abusing the existing contracts. However, current PEOPLE token holders can still use stake() and unstake() functions to affect the total supply of the token in TicketBooth, also the logic of the current Terminal contract must be accepted by the community if no new Terminal contract is migrated by the current owner. Also if the ownership is renounced to address(0), no new Terminal contract can be migrated for this project.

Note that the data collected and shown in this version of the report is based on the time when the report was written. It is not guaranteed that the configuration data will be kept as this until the ownership of the project was transferred to a blackhole address.

- Remediation: