

SMART CONTRACT AUDIT REPORT

For Recurring Income Token

Prepared By: Kishan Patel

Prepared For: REIT Finance Labs

Prepared on: 21/10/2021

Table of Content

- Disclaimer
- Overview of the audit
- Attacks made to the contract
- Good things in smart contract
- Critical vulnerabilities found in the contract
- Medium vulnerabilities found in the contract
- Low severity vulnerabilities found in the contract
- Summary of the audit

- **Disclaimer**

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

- **Overview of the audit**

The project has 6 files. It contains approx 1300 lines of Solidity code. All the functions and state variables are well commented using the natspec documentation, but that does not create any vulnerability.

- **Attacks made to the contract**

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

- **Over and under flows**

An overflow happens when the limit of the type variable `uint256`, 2^{256} , is exceeded. What happens is that the value resets to zero instead of incrementing more. On the other hand, an underflow happens when you try to subtract 0 minus a number bigger than 0. For example, if you subtract $0 - 1$ the result will be $= 2^{256}$ instead of -1 . This is quite dangerous.

This contract **does** check for overflows and underflows by using OpenZeppelin's SafeMath to mitigate this attack, but all the functions have strong validations, which prevented this attack.

- **Short address attack**

If the token contract has enough amount of tokens and the buy function doesn't check the length of the address of the sender, the ethereum's virtual machine will just add zeros to the transaction until the address is complete.

Although this contract **is not vulnerable** to this attack, but there are some point where users can mess themselves due to this (Please see below). It is highly recommended to call functions after checking validity of the address.

- **Visibility & Delegate call**

It is also known as, The Parity Hack, which occurs while misuse of Delegate call.

No such issues found in this smart contract and visibility also properly addressed. There are some places where there is no visibility defined. Smart Contract will assume "Public" visibility if there is no visibility defined. It is good practice to explicitly define the visibility, but again, the contract is not prone to any vulnerability due to this in this case.

- **Reentrancy / TheDAO hack**

Reentrancy occurs in this case: any interaction from a contract (A) with another contract (B) and any transfer of ethereum hands over control to that contract (B).

This makes it possible for B to call back into A before this interaction is completed.

Use of “require” function in this smart contract mitigated this vulnerability.

- **Forcing Ethereum to a contract**

While implementing “selfdestruct” in smart contract, it sends all the ethereum to the target address. Now, if the target address is a contract address, then the fallback function of target contract does not get called. And thus Hacker can bypass the “Required” conditions. Here, the Smart Contract’s balance has never been used as guard, which mitigated this vulnerability.

- **Good things in smart contract**

- **SafeMath library:-**

- ❖ **Filename:- All files**

- You are using SafeMath library it is a good thing. This protects you from underflow and overflow attacks.

```
5  
6 import "../base/SafeMath.sol";  
7 import "../interfaces/ICommitter.sol";  
8 import "../interfaces/ITreasury.sol";
```

- **Compiler version is fixed:-**

- ❖ **Filename:- All Files**

=> In this file you have put “pragma solidity 0.6.12;” which is a good way to define compiler version.

=> Solidity source files indicate the versions of the compiler they can be compiled with. Pragma solidity ^0.6.12; // bad: compiles 0.6.12 and above
pragma solidity 0.6.12; //good: compiles 0.6.12 only

=> If you put(>=) symbol then you are able to get compiler version 0.6.12 and above. But if you don’t use(^/>=) symbol then you are able to use only 0.6.12 version. And if there are some changes come in the compiler and you use the old version then some issues may come at deploy time.

=> Try to use latest version of solidity.

- **Good required condition in functions:-**

- ❖ **Filename:- Comitter.sol**

- Here you are checking that treasury transfer is active or not, duration should be between 3 to 12 and msg.value should be bigger or equal to minComitmentValue.

```

85 //users become a investor with this function
86 function enterEtherComitment(uint duration,uint[] memory date,uint[] memory d
87 (bool isAllowed,)=reitAdminIns.isTreasuryTransferAllowed());
88 require(!isAllowed,"Treasury will be transfered");
89 uint txFeeRate = feeManagerInstance.getFeeRate();
90 require(duration >= 3,"Duration must be higher then 2");
91 require(duration <= 12,"Duration must be lower then 13");
92 require(msg.value >= minComitmentValue,"Comit value must be higher the

```

- Here you are checking that correct commitment should be paid, commitment should not be already paid, Payment is not overdue, commitment state should not be failed, payment is in correct order or not, payment should accepted before 4 days.

```

143 //Installment payment function
144 function enterPayment(uint256 commitmentIndex,uint256 paymentIndex,uint256 cla
145 uint txFeeRate = feeManagerInstance.getFeeRate();
146 // uint paymentDate = TimeCalculation.addMonths(accountCommitsEther[msg.se
147 uint paymentDate = block.timestamp + 600 * paymentIndex;//test 10 min
148 require(
149 msg.value - (accountCommitsEther[msg.sender][commitmentIndex].amount
150 ,"You should pay same or higher comitment"
151 );
152 require(accountCommitsEther[msg.sender][commitmentIndex].payedCount <= pay
153 require(paymentDate + 4 days > block.timestamp, "Payment overdue");
154 require(accountCommitsEther[msg.sender][commitmentIndex].state == 0, "Stat
155 require(
156 paymentIndex == accountCommitsEther[msg.sender][commitmentIndex].payed
157 ,"Payments must be made in the correct order."
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
184
185
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
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

```

- Here you are checking that feeManager address can only call this function.

```

221 //Set minimum value for comitment
222 function setMinComitValueValue(uint val)external override{
223 require(msg.sender == feeManager,"Only Fee Manager can call this function");
224 minComitmentValue = val;
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

```

- Here you are checking that only owner can call this function, treasuryAdr is valid and proper address and treasuryAddress is not already set.

```

226 //Set treasury address for connection
227 function setTreasuryAddress(address payable treasuryAdr)external {
228 require(treasuryAdr != address(0),"Address(0)");
229 require(msg.sender == owner,"Only owner can call this function");
230 require(treasuryAddress == address(0),"Treasury address already set");
231 treasuryAddress = treasuryAdr;
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

```

- Here you are checking that only owner can call this function and commite state is not failed.

```

234
235 ▾ function updateCommitState(address acc,uint indexFirst) external {
236     require(accountCommitsEther[acc][indexFirst].state == 0,"State Fail");
237     require(msg.sender == owner,"Only owner can call this function");
238     accountCommitsEther[acc][indexFirst].state = 1;

```

- Here you are checking that only treasuryAddress can call this method.

```

267
268 ▾ function changeRewardClaimedStatus(address acc,uint commitIndex,uint paymentIndex) external {
269     require(msg.sender == treasuryAddress,"Only Treasury Contract can call this method");
270     accountCommitsEther[acc][commitIndex].defiProtocolValues[paymentIndex].isClaimed = true;
271     accountCommitsEther[acc][commitIndex].defiProtocolValues[paymentIndex].claimTime = block.timestamp;

```

❖ Filename:- ReitCoin.sol

- Here you are checking that treasuryAdr address is valid and proper, and contractOwner address can only call this function.

```

55
56 ▾ function setTreasuryAddress(address treasuryAdr)external {
57     require(treasuryAdr != address(0),"Address(0)");
58     require(msg.sender == contractOwner,"Only owner can call this function");
59     treasuryAddress = treasuryAdr;

```

- Here you are checking that only treasuryAddress address can only call this function.

```

80
81
82 ▾ function mint(address account,uint amount)external override{
83     require(msg.sender == treasuryAddress,"Only treasury contract can call this function");
84     _mint(account,amount);

```

- Here you are checking that sender and recipient addresses value are valid and proper.

```

101
102 ▾ function _transfer(address sender, address recipient, uint256 amount) internal virtual {
103     require(sender != address(0), "ERC20: transfer from the zero address");
104     require(recipient != address(0), "ERC20: transfer to the zero address");
105

```

- Here you are checking that account address value is valid and proper.

```

112
113 ▾ function _mint(address account, uint256 amount) internal virtual {
114     require(account != address(0), "ERC20: mint to the zero address");
115

```

```

121
122
123 ▾ function _burn(address account, uint256 amount) internal virtual {
124     require(account != address(0), "ERC20: burn from the zero address");
125

```

- Here you are checking that owner and spender addresses value are valid and proper.

```
133 ▾   function _approve(address owner, address spender, uint256 amount) internal virtual {
134     require(owner != address(0), "ERC20: approve from the zero address");
135     require(spender != address(0), "ERC20: approve to the zero address");
136 }
```

❖ Filename:- ReitGCoin.sol

- Here you are checking that treasuryAdr is proper and valid address and only contract owner can call this function.

```
53
54 ▾   function setTreasuryAddress(address treasuryAdr) external {
55     require(treasuryAdr != address(0), "Address(0)");
56     require(msg.sender == contractOwner, "Only owner can call this function");
57     treasuryAddress = treasuryAdr;
58 }
```

- Here you are checking that only owner of contract or treasuryAddress can call this function and _totalSupply should not be bigger than _maxSupply.

```
85
86 ▾   function mint(address account, uint amount) external override {
87     require(
88         msg.sender == treasuryAddress
89         ||
90         msg.sender == contractOwner, "Only treasury contract or owner can call
91     );
92     require(_maxSupply > _totalSupply, "Reached to maximum reitg supply");
93     _mint(account, amount);
94 }
```

- Here you are checking that sender and recipient addresses value are valid and proper.

```
106 ▾  function _transfer(address sender, address recipient, uint256 amount) internal virtual {
107     require(sender != address(0), "ERC20: transfer from the zero address");
108     require(recipient != address(0), "ERC20: transfer to the zero address");
109     _transfer(sender, recipient, amount);
110 }
```

- Here you are checking that account address value is valid and proper.

```
117
118 ▾  function _mint(address account, uint256 amount) internal virtual {
119     require(account != address(0), "ERC20: mint to the zero address");
120     _mint(account, amount);
121 }
```

```
126 }
127
128 ▾  function _burn(address account, uint256 amount) internal virtual {
129     require(account != address(0), "ERC20: burn from the zero address");
130     _burn(account, amount);
131 }
```

- Here you are checking that owner and spender addresses value are valid and proper.

```
137
138 ▾  function _approve(address owner, address spender, uint256 amount) internal virtual {
139     require(owner != address(0), "ERC20: approve from the zero address");
140     require(spender != address(0), "ERC20: approve to the zero address");
141     _approve(owner, spender, amount);
142 }
```

❖ Filename:- FeeManager.sol

- Here you are checking that only comitterAddress or depositorAddress can call this function.

```
56 //update collected txfee every commit payment or deposit
57 function txFeeProcess(uint txFee) external override{
58     //only depo and comitter
59     require(msg.sender == address(comitterAddress) || msg.sender == address(depositorAddress));
60     totalCollectedTxFee += txFee;
```

- Here you are checking that only treasuryAddress can call this function.

```
63 function managementFeeProcess(uint fee) external override {
64     //only treasury
65     require(msg.sender == treasuryAddress,"Only owner can call this function");
66     totalCollectedManagementFee += fee;
```

- Here you are checking that comitterAdr should be valid and proper, comitterAddress should be not already set.

```
68 //Set comitter address for connection
69 function setComitterAddress(address payable comitterAdr)external onlyAdmins{
70     require(comitterAdr != address(0),"Address(0)");
71     require(comitterAddress == address(0),"Comitter address already set");
72     comitterAddress = comitterAdr;
```

- Here you are checking that depositorAdr value is valid and proper, depositorAddress is not already set.

```
75 //Set depositor address for connection
76 function setDepositorAddress(address payable depositorAdr)external onlyAdmins{
77     require(depositorAdr != address(0),"Address(0)");
78     require(depositorAddress == address(0),"Depositor address already set");
79     depositorAddress = depositorAdr;
```

- Here you are checking that treasuryAdr value is valid and proper, treasuryAddress is not already set.

```
82 //Set treasury address for connection
83 function setTreasuryAddress(address payable treasuryAdr)external onlyAdmins {
84     require(treasuryAdr != address(0),"Address(0)");
85     require(treasuryAddress == address(0),"Treasury address already set");
86     treasuryAddress = treasuryAdr;
```

- Here you are checking that acc should be valid and proper address, treasuryAddress can call this function.

```
89 //send txfeereward to user
90 function sendTxFeeReward(uint value,address payable acc) external override{
91     require(acc!=address(0),"Adress(0)");
92     require(msg.sender == treasuryAddress);
93     uint feeVal = value * feeRate / 1000;
```

❖ Filename:- Depositor.sol

- Here you are checking that treasury transfer is active or not and msg.value should be bigger or equal to minDepositValue.

```

56 //Deposit function for user
57 function enterDeposit(uint256 claimDate,uint[] memory dateForBuyBack) external
58 (bool isAllowed,)=reitAdminIns.isTreasuryTransferAllowed();
59 require(!isAllowed,"Treasury will be transfered");
60 uint txFeeRate = feeManagerInstance.getFeeRate();
61 require(msg.value >= minDepositValue ,"Deposit value must be higher or equal

```

- Here you are checking that only owner can call this function, treasuryAdr is valid and proper address and treasuryAddress is not already set.

```

97 //Treasury Connection
98 function setTreasuryAddress(address payable treasuryAdr)external {
99 require(treasuryAdr != address(0),"Address(0)");
100 require(msg.sender == owner,"Only owner can call this function");
101 require(treasuryAddress == address(0),"Treasury address already set");
102 treasuryAddress = treasuryAdr;

```

- Here you are checking that only owner can call this function, treasuryAdr is valid and proper address and treasuryAddress is not already set.

```

97 //Treasury Connection
98 function setTreasuryAddress(address payable treasuryAdr)external {
99 require(treasuryAdr != address(0),"Address(0)");
100 require(msg.sender == owner,"Only owner can call this function");
101 require(treasuryAddress == address(0),"Treasury address already set");
102 treasuryAddress = treasuryAdr;

```

- Here you are checking that only feeManager address can call this function.
Please check val should be bigger than 0.

```

105 //Set minimum value for enter deposit
106 function setMinDepositValue(uint val)external override{
107 require(msg.sender == feeManager,"Only Fee Manager can call this function")
108 minDepositValue = val;

```

- Here you are checking that only treasuryAddress can call this method.

```

136 }
137 // if user claim his/her reward this function will update their state
138 function changeRewardClaimedStatus(address acc,uint index,bool isClaim,uint re
139 require(msg.sender == treasuryAddress,"Only Treasury Contract can call thi
140 account[0]]Deposits[acc][index] isClaimed = isClaim;

```

- **Critical vulnerabilities found in the contract**

=> No Critical vulnerabilities found

- **Medium vulnerabilities found in the contract**

=> **No Medium vulnerabilities found**

- **Low severity vulnerabilities found**

- **7.1: Suggestions to add code validations:-**

- => You have implemented required validation in contract.

- => There are some place where you can improve validation and security of your code.

- => These are all just suggestion it is not bug.

- ❖ **Filename:- ReitAdmin.sol**

- ✚ **Function: - allowSetLimit, allowTreasuryTransfer,**

```
78 //SetSign allowance
79 function allowSetLimit(bool newState)external onlyAdmins{
80     setLimitSigns[msg.sender]=newState;
81 }
```

```
170 /* TREASURY TRANSFER ALLOWANCE */
171 function allowTreasuryTransfer(bool state)external onlyAdmins{
172     treasuryAllowance[msg.sender]=state;
173 }
```

- In allowSetLimit function you need to check that msg.sender has not already same value set as newState. If newState value is already same for msg.sender then don't run this function.

- In allowTreasuryTransfer you need to check that same state value is not set to msg.sender if it is same then don't run this function.

○ 7.2: Uncheck return response of transfer method:-

=> I have found that you are transferring fund to address using a transfer methods.

=> It is always good to check the return value or response from a function call.

=> Here are some functions where you forgot to check a response.

=> I suggest, if there is a possibility then please check the response.

❖ Filename:- Comitter.sol

✚ Function: - enterEtherCommitment, enterPayment

```
104
105     treasuryAddress.transfer(msg.value-feeValue);
106     treasury.collateralValueProcess(collateralValue,monthlyCollateralAmount);
107
108
109
110
111
112
113
114
115
116
117     treasury.buyBackAdd(commitValHolder.commitmentBuyBackValue);
118     treasury.sendReitToUser(msg.sender,commitmentValue);
119     // treasury.setMonthlyCollateral(monthlyCollateralAmount);
120
121     feeManager.transfer(feeValue);
122     feeManagerInstance.txFeeProcess(feeValue);|
123
124     (accDefiDebtSecValuesHolder memory newWinHolder = accDefiDebtSecValuesHolder)
125
126
127     // uint paymentBuyBackValue = amount.mul(70).div(100).div(24);
128
129     treasuryAddress.transfer(msg.value-feeValue);
130
131     accDefiDebtSecValuesHolder memory newWinHolder = accDefiDebtSecValuesHolder
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148     accountCommitsEther[msg.sender][commitmentIndex].defiProtocolValues.push(n
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
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200     defiComitterTotalEthAtIndex[lendingProtocolIndex] = defiComitterTotalEthA
201     feeManager.transfer(feeValue);
202     feeManagerInstance.txFeeProcess(feeValue);
203
204     (accDefiDebtSecValuesHolder memory newWinHolder = accDefiDebtSecValuesHolder)
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
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
```

- Here you are calling transfer method 4 times. It is good to check that the transfer is successfully done or not.

❖ Filename:- FeeManager.sol

✚ Function: - sendTxFeeReward

```
94     uint txFeeReward = feeVal * feeRePaymentRate / txFeeStabil;
95     totalCollectedTxFee = totalCollectedTxFee.sub(txFeeReward);
96     acc.transfer(txFeeReward);
97
98     (accDefiDebtSecValuesHolder memory newWinHolder = accDefiDebtSecValuesHolder)
99
100
101
102
103
104
105
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
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
184
185
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
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
```

- Here you are calling transfer method 1 time. It is good to check that the transfer is successfully done or not.

○ 7.3: Approve given more allowance:-

=> I have found that in approve function user can give more allowance to a user beyond their balance.

=> It is necessary to check that user can give allowance less or equal to their amount.

=> There is no validation about user balance. So it is good to check that a user not set approval wrongly.

❖ Filename:- ReitCoin.sol

+ Function: - _approve

```
132
133 ▾ function _approve(address owner, address spender, uint256 amount) internal vi
134     require(owner != address(0), "ERC20: approve from the zero address");
135     require(spender != address(0), "ERC20: approve to the zero address");
136
137     _allowances[owner][spender] = amount;
138     emit Approval(owner, spender, amount);
```

- Here you can check that balance of owner should be bigger or equal to amount value.

❖ Filename:- ReitGCoin.sol

+ Function: - _approve

```
137
138 ▾ function _approve(address owner, address spender, uint256 amount) internal vi
139     require(owner != address(0), "ERC20: approve from the zero address");
140     require(spender != address(0), "ERC20: approve to the zero address");
141
142     _allowances[owner][spender] = amount;
143     emit Approval(owner, spender, amount);
144 )
```

- Here you can check that balance of owner should be bigger or equal to amount value.

• Summary of the Audit

Overall, the code is written with all validation and all security is implemented. Code is performs well and there is no way to steal fund from this contract.

Please try to check the address and value of token externally before sending to the solidity code.

Our final recommendation would be to pay more attention to the visibility of the functions , hardcoded address and mapping since it's quite important to define who's supposed to executed the functions and to follow best practices regarding the use of assert, require etc. (which you are doing ;)).

- **Good Point:** You are paying attention to the visibility of the functions. Contract is nit perfect size with good validation. Code performance and quality is good. Address validation and value validation is done properly.
- **Suggestions:** Please try to add suggested code validations, check return response of transfer method, check user balance in approve function.