BLOCKHAT
SECURITY

Trip Foundation

Smart Contract Security Audit

Prepared by BlockHat

April 3rd, 2023 - April 5th, 2023

BlockHat.io

contact@blockhat.io

Document Properties

Client	Trip Foundation
Version	0.1
Classification	Public

Scope

The Trip Foundation Contract in the Trip Foundation Repository

Repo	Owner
https://bscscan.com/address/0xf4E9FeDE7FAB2C061b709C479BCBe9B726379341#code	0xf4E9FeDE7FAB2C061b709C479BCBe9B726379341

Files	MD5 Hash
tripfoundation.sol	8f0649699eac6b3fd5cf56bbf9231ee4

Contacts

COMPANY	CONTACT
BlockHat	contact@blockhat.io

Contents

1 Introduction

Trip Foundation engaged BlockHat to conduct a security assessment on the Trip Foundation beginning on April 3rd, 2023 and ending April 5th, 2023. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About Trip Foundation

-

Issuer	Trip Foundation
Website	`https://tripfoundation.io/`
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

BlockHat used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

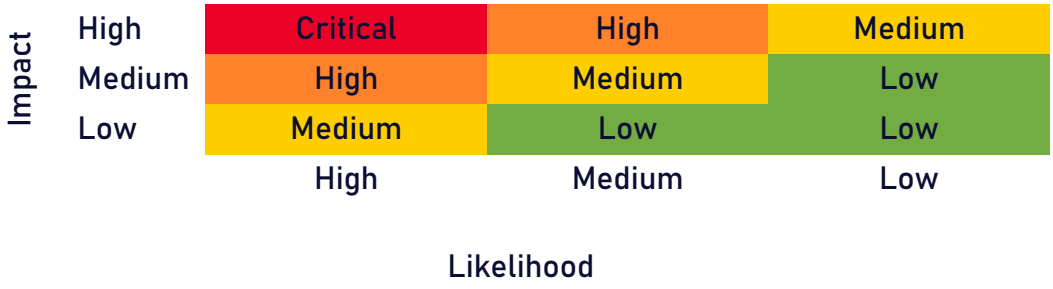
1.2.1 Risk Methodology

Vulnerabilities or bugs identified by BlockHat are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.



2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the Trip Foundation implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include , 5 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
Missing Transfer Verification	LOW	Acknowledged
Planno check	LOW	Acknowledged
Missing values check	LOW	Acknowledged
Missing Wallet addresses check	LOW	Acknowledged
Missing address verification	LOW	Acknowledged

3 Finding Details

A tripfoundation.sol

A.1 Missing Transfer Verification [LOW]

Description:

The ERC20 standard token implementation functions return the transaction status as a Boolean. It is a good practice to check for the return status of the function call to ensure that the transaction was executed successfully. It is the developer's responsibility to enclose these function calls with require() to ensure that, when the intended ERC20 function call returns false, the caller transaction also fails.

Code:

Listing 1: TripFoundation.sol

```
351     function withdrawprofit() public {

353         address _customerAddress = msg.sender;
354     uint256 withdrawamt;

356         User storage user = users[msg.sender];
357         require(user.cashoutuser == false);

359     withdrawamt = payouts_[_customerAddress];

361         uint256 contractBal = token.balanceOf(address(this));
362     require(contractBal > withdrawamt);

364     payoutsTo_[_customerAddress] = payoutsTo_[_customerAddress].add(
            ↪ withdrawamt);

366         token.transfer(msg.sender, withdrawamt);
```

```
367          payouts_[_customerAddress] = 0;

369          // fire event
370          emit onWithdraw(_customerAddress, withdrawamt);
371      }
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Use the safeTransfer function from the safeERC20 Implementation, or put the transfer call inside an assert or require to verify that it returned true.

Status - Acknowledged

A.2 Planno check [LOW]

Description:

The function accepts a uint256 planno and an address referrer as arguments, but there is no check to ensure that the provided planno is within the expected range (1-6).

Code:

Listing 2: TripFoundation.sol

```
353      function withdrawprofit() public {

355          address _customerAddress = msg.sender;
356     uint256 withdrawamt;

358          User storage user = users[msg.sender];
359          require(user.cashoutuser == false);
```

```
361     withdrawamt = payouts_[_customerAddress];

363         uint256 contractBal = token.balanceOf(address(this));
364     require(contractBal > withdrawamt);

366     payoutsTo_[_customerAddress] = payoutsTo_[_customerAddress].add(
            ↪ withdrawamt);

368         token.transfer(msg.sender, withdrawamt);
369          payouts_[_customerAddress] = 0;

371         // fire event
372         emit onWithdraw(_customerAddress, withdrawamt);
373       }
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

To fix this, you could add a require statement to ensure planno is within the expected range:

Status - Acknowledged

A.3 Missing values check [LOW]

Description:

1) withdrawprofit() It doesn't check if the withdrawamt is greater than zero, which could allow a user to withdraw zero tokens and still trigger the onWithdraw event. 2) If the _amountOfTokens argument is set to 0, the user would be able to call the selltokens function without actually selling any tokens. This could lead to unintended consequences,

such as the user receiving payouts or triggering other functions in the contract without actually providing any value or contributing to the contract.

Code:

Listing 3: TripFoundation.sol

```
353     function withdrawprofit() public {

355         address _customerAddress = msg.sender;
356     uint256 withdrawamt;

358         User storage user = users[msg.sender];
359         require(user.cashoutuser == false);

361     withdrawamt = payouts_[_customerAddress];

363         uint256 contractBal = token.balanceOf(address(this));
364     require(contractBal > withdrawamt);

366     payoutsTo_[_customerAddress] = payoutsTo_[_customerAddress].add(
            ↪ withdrawamt);

368         token.transfer(msg.sender, withdrawamt);
369          payouts_[_customerAddress] = 0;

371         // fire event
372         emit onWithdraw(_customerAddress, withdrawamt);
373     }
```

Listing 4: TripFoundation.sol

```
373     function selltokens(uint256 _amountOfTokens) onlybelievers () public
            ↪  {
374         address _customerAddress = msg.sender;
375          User storage user = users[msg.sender];
```

```
376         require(user.cashoutuser == false);

378         require(_amountOfTokens <= tokenBalanceLedger_[_customerAddress])
                ↪ ;
379         uint256 _tokens = _amountOfTokens;
380         uint256 _Bnb = tokensToBnb_(_tokens);
381         uint256 _taxedBnb = _Bnb;

383         tokenSupply_ = SafeMath.sub(tokenSupply_, _tokens);
384         tokenBalanceLedger_[_customerAddress] = SafeMath.sub(
                ↪ tokenBalanceLedger_[_customerAddress], _tokens);

386         payouts_[_customerAddress] += _taxedBnb;

388         emit onTokenSell(_customerAddress, _tokens, _taxedBnb);
389     }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

add a check to ensure that withdrawamt is greater than zero.

Status - Acknowledged

A.4 Missing Wallet addresses check [LOW]

Description:

The constructor of the smart contract accepts 5 addresses as arguments for pancake, marketing, trip, development and project wallets and . However, there is no check for the input

values to ensure that none of these addresses are set to 0x0 or the zero address. This may lead to unexpected behavior or even security vulnerabilities.

Code:

Listing 5: TripFoundation.sol

```
321  constructor(address payable pancake, address payable marketing, address
       ↪  payable trip, address payable development, address payable
       ↪ project, address tokenAddr) public {
322   require(!isContract(development) && isContract(tokenAddr));
323         token = IERC20(tokenAddr);

325    pancakeWallet  = pancake;
326    marketingWallet = marketing;
327    tripWallet   = trip;
328    developmentWallet = development;
329    projectWallet  = project;


332   tokenBalanceLedger_[developmentWallet] = SafeMath.add(
         ↪ tokenBalanceLedger_[developmentWallet], developmentsupply);
333   emit Transfer(address(this), developmentWallet, developmentsupply);



337  }
```

Risk Level:

Likelihood – 1

Impact - 2

Recommendation:

It is recommended to add a check at the beginning of the constructor to ensure that none of the addresses passed as arguments are set to 0x0 or the zero address.

Status - Acknowledged

A.5 Missing address verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument transfer function should include a zero-address test for the _toAddress

Code:

Listing 6: TripFoundation.sol

```
397     function transfer(address _toAddress, uint256 _amountOfTokens) public
            ↪  returns(bool) {
398         address _customerAddress = msg.sender;

400         require( _amountOfTokens <= tokenBalanceLedger_[_customerAddress
                ↪ ]);

402         uint256 _taxedTokens = _amountOfTokens;

404         tokenSupply_ = tokenSupply_;

406         tokenBalanceLedger_[_customerAddress] = SafeMath.sub(
                ↪ tokenBalanceLedger_[_customerAddress], _amountOfTokens);
407         tokenBalanceLedger_[_toAddress] = SafeMath.add(
                ↪ tokenBalanceLedger_[_toAddress], _taxedTokens);


410         emit Transfer(_customerAddress, _toAddress, _taxedTokens);
```

```
411             return true;
412         }
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

It is recommended to verify that the address provided in the arguments is different from the address(0) .

Status - Acknowledged

4 Best Practices

BP.1 Duplicated Referral Mappings

Description:

The developer has duplicated referral mappings up to 30 levels, which can cause code redundancy and increase the likelihood of errors. A better approach would be to use a single mapping with a level identifier as the key, reducing code duplication and improving code maintainability

Code:

Listing 7: TripFoundation.sol

```
238   mapping(address => address) internal referralLevel1Address;
239      mapping(address => address) internal referralLevel2Address;
240      mapping(address => address) internal referralLevel3Address;
241      mapping(address => address) internal referralLevel4Address;
242      mapping(address => address) internal referralLevel5Address;
243      mapping(address => address) internal referralLevel6Address;
244      mapping(address => address) internal referralLevel7Address;
245      mapping(address => address) internal referralLevel8Address;
246      mapping(address => address) internal referralLevel9Address;
247      mapping(address => address) internal referralLevel10Address;
248
249   mapping(address => address) internal referralLevel11Address;
250      mapping(address => address) internal referralLevel12Address;
251      mapping(address => address) internal referralLevel13Address;
252      mapping(address => address) internal referralLevel14Address;
253      mapping(address => address) internal referralLevel15Address;
254   mapping(address => address) internal referralLevel16Address;
255      mapping(address => address) internal referralLevel17Address;
256      mapping(address => address) internal referralLevel18Address;
257      mapping(address => address) internal referralLevel19Address;
```

```
258     mapping(address => address) internal referralLevel20Address;
259
260   mapping(address => address) internal referralLevel21Address;
261     mapping(address => address) internal referralLevel22Address;
262     mapping(address => address) internal referralLevel23Address;
263     mapping(address => address) internal referralLevel24Address;
264     mapping(address => address) internal referralLevel25Address;
265   mapping(address => address) internal referralLevel26Address;
266     mapping(address => address) internal referralLevel27Address;
267     mapping(address => address) internal referralLevel28Address;
268     mapping(address => address) internal referralLevel29Address;
269     mapping(address => address) internal referralLevel30Address;
```

BP.2 Redundant Referral Level Checking

Description:

The developer is creating new variables to store the value of each referral level, which can lead to redundancy in the code and unnecessarily increase gas costs. Instead, the developer can directly access the storage for each referral level or use a memory variable to retrieve the values, reducing code complexity and improving gas efficiency.

Code:

Listing 8: TripFoundation.sol

```
435         chkLv2 = referralLevel1Address[_referredBy];
436         chkLv3 = referralLevel2Address[_referredBy];
437         chkLv4 = referralLevel3Address[_referredBy];
438         chkLv5 = referralLevel4Address[_referredBy];
439         chkLv6 = referralLevel5Address[_referredBy];
440         chkLv7 = referralLevel6Address[_referredBy];
441         chkLv8 = referralLevel7Address[_referredBy];
442         chkLv9 = referralLevel8Address[_referredBy];
443         chkLv10 = referralLevel9Address[_referredBy];
```

```
444
445     chkLv11 = referralLevel10Address[_referredBy];
446        chkLv12 = referralLevel11Address[_referredBy];
447           chkLv13 = referralLevel12Address[_referredBy];
448           chkLv14 = referralLevel13Address[_referredBy];
449           chkLv15 = referralLevel14Address[_referredBy];
450     chkLv16 = referralLevel15Address[_referredBy];
451        chkLv17 = referralLevel16Address[_referredBy];
452           chkLv18 = referralLevel17Address[_referredBy];
453           chkLv19 = referralLevel18Address[_referredBy];
454           chkLv20 = referralLevel19Address[_referredBy];
455
456     chkLv21 = referralLevel20Address[_referredBy];
457        chkLv22 = referralLevel21Address[_referredBy];
458           chkLv23 = referralLevel22Address[_referredBy];
459           chkLv24 = referralLevel23Address[_referredBy];
460           chkLv25 = referralLevel24Address[_referredBy];
461     chkLv26 = referralLevel25Address[_referredBy];
462        chkLv27 = referralLevel26Address[_referredBy];
463           chkLv28 = referralLevel27Address[_referredBy];
464           chkLv29 = referralLevel28Address[_referredBy];
465           chkLv30 = referralLevel29Address[_referredBy];
```

BP.3 Public functions can be external

Description:

Functions with a public scope that are not called inside the contract should be declared external to reduce the gas fees

Code:

Listing 9: TripFoundation.sol

```
1243     function totalPayouts(address _customerAddress) view public returns(
             ↪ uint256) {
1244         return payoutsTo_[_customerAddress];
1245     }
1246 
1247      function totalavailPayouts(address _customerAddress) view public
             ↪ returns(uint256) {
1248         return payouts_[_customerAddress];
1249     }
1250 
1251     function totalBnbBalance() public view returns(uint) {
1252         return address(this).balance;
1253     }
```

Listing 10: TripFoundation.sol

```
1243     function capitabalance(address _customerAddress) public view returns
             ↪ (uint256){
1244         return capitaBalanceLedger_[_customerAddress];
1245     }
1246 
1247     function getUserCapitaWithdrawn(address userAddress) public view
             ↪ returns (uint256) {
1248   return users[userAddress].withdrawnCapita;
1249   }
```

Listing 11: TripFoundation.sol

```
1288   function getUserAllWithdrawn(address userAddress) public view returns
           ↪ (uint256 unilevel_withdrawn,uint256 referral_withdrawn,uint256
           ↪ reward_withdrawn ) {
1289    unilevel_withdrawn = users[userAddress].withdrawnUNILEVEL;
1290    referral_withdrawn = users[userAddress].withdrawnReferral;
1291    reward_withdrawn = users[userAddress].withdrawnReward;
1292   }
```

Listing 12: TripFoundation.sol

```
1298  function getUserTotalReward(address userAddress) public view returns (
          ↪ uint256) {
1299   return users[userAddress].RankingReward;
1300  }
1301
1302     function getUserRanking(address userAddress) public view returns (
           ↪ bool[12] memory achivement) {
1303   return users[userAddress].Reward_achivement;
1304  }
1305
1306
1307  function getUserCheckpoint(address userAddress) public view returns(
          ↪ uint256) {
1308   return users[userAddress].checkpoint;
1309  }
1310
1311  function getUserReferrer(address userAddress) public view returns(
          ↪ address) {
1312   return users[userAddress].referrer;
1313  }
1314
1315  function getUserDownlineCount(address userAddress) public view returns(
          ↪ uint256[5] memory referrals) {
1316    return (users[userAddress].levels);
1317  }
```

Listing 13: TripFoundation.sol

```
1327  function getUserReferralTotalBonus(address userAddress) public view
          ↪ returns(uint256) {
1328   return users[userAddress].totalBonus;
1329  }
1330
1331
```

```
1332     function getUserteambusiness(address userAddress) public view
             ↪ returns(uint256) {
1333     return users[userAddress].teambusiness;
1334     }
1335
1336
1337     function getUserReferralWithdrawn(address userAddress) public view
             ↪ returns(uint256) {
1338     return users[userAddress].totalBonus.sub(users[userAddress].bonus);
1339     }
1340
1341     function getUserAvailable(address userAddress) public view returns(
             ↪ uint256) {
1342     return getUserReferralBonus(userAddress).add(getUserDividends(
             ↪ userAddress));
1343     }
1344
1345     function getUserAmountOfDeposits(address userAddress) public view
             ↪ returns(uint256) {
1346     return users[userAddress].deposits.length;
1347     }
1348
1349     function getUserTotalDeposits(address userAddress) public view returns(
             ↪ uint256 amount) {
1350     for (uint256 i = 0; i < users[userAddress].deposits.length; i++) {
1351       amount = amount.add(users[userAddress].deposits[i].amount);
1352     }
1353     }
```

Listing 14: TripFoundation.sol

```
1375     function getSiteInfo() public view returns(uint256 _totalInvested,
             ↪ uint256 _totalBonus) {
1376     return(totalInvested, totalRefBonus);
1377     }
```

Listing 15: TripFoundation.sol

```
1419     function sellPrice() public view returns(uint256) {
1420         if(tokenSupply_ == 0){
1421             return tokenPriceInitial_ - tokenPriceDecremental_;
1422         } else {
1423             uint256 _Bnb = tokensToBnb_(1e18);
1424             uint256 _taxedBnb = _Bnb;
1425             return _taxedBnb;
1426         }
1427     }
1428
1429
1430     function buyPrice() public view returns(uint256) {
1431         if(tokenSupply_ == 0){
1432             return tokenPriceInitial_ + tokenPriceIncremental_;
1433         } else {
1434             uint256 _Bnb = tokensToBnb_(1e18);
1435             return _Bnb;
1436         }
1437     }
```

BP.4 uninitialized totalRefBonus

Description:

The variable totalRefBonus is never initialized and It is used in getSiteInfo() , this varibable will be 0 all the time The variable totalRefBonus is always initialized to 0 and is never updated throughout the execution of the smart contract. As a result, it will always be 0 when accessed in the getSiteInfo() function.

Code:

Listing 16: TripFoundation.sol

```
1375    function getSiteInfo() public view returns(uint256 _totalInvested,
            ↪ uint256 _totalBonus) {
1376    return(totalInvested, totalRefBonus);
1377   }
```

BP.5 Error message in require function

Description:

The require function should also include an error message that explains why the transaction failed. Without an error message, it may be difficult for the user to understand why their transaction failed, which can lead to confusion and frustration.

Code:

Listing 17: TripFoundation.sol

```
343         require(myTokens() > 0);
```

Listing 18: TripFoundation.sol

```
357         require(user.cashoutuser == false);
```

Listing 19: TripFoundation.sol

```
362    require(contractBal > withdrawamt);
```

Listing 20: TripFoundation.sol

```
381         require(_amountOfTokens <= tokenBalanceLedger_[_customerAddress])
                ↪ ;
```

Listing 21: TripFoundation.sol

```
747       require(value <= token.allowance(msg.sender, address(this)));
```

Listing 22: TripFoundation.sol

```
885         require(capitaBalanceLedger_[msg.sender] != 0);
```

Listing 23: TripFoundation.sol

```
942             require(_initialsupply >= totalAmount);
```

5 Static Analysis (Slither)

Description:

Block Hat expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

Results:

```
TripFoundation.withdrawprofit() (tripfoundation.io.sol#353-373) ignores
    ↪ return value by token.transfer(msg.sender,withdrawamt) (
    ↪ tripfoundation.io.sol#368)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unchecked-transfer

TripFoundation.totalRefBonus (tripfoundation.io.sol#184) is never
    ↪ initialized. It is used in:
        - TripFoundation.getSiteInfo() (tripfoundation.io.sol#1378-1380)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-state-variables

TripFoundation.getUserDividends(address) (tripfoundation.io.sol#970-993)
    ↪  performs a multiplication on the result of a division:
        -share = user.deposits[i].tokenamount.mul(user.deposits[i].
            ↪ percent).div(PLANPER_DIVIDER) (tripfoundation.io.sol#980)
        -totalAmount = totalAmount.add(share.mul(to.sub(from)).div(
            ↪ TIME_STEP)) (tripfoundation.io.sol#984)
TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
    ↪ #995-1243) performs a multiplication on the result of a division:
```

```
        -share = downline.deposits[i].tokenamount.mul(downline.deposits[i
            ↪ ].percent).div(PLANPER_DIVIDER) (tripfoundation.io.sol
            ↪ #1014)
        -UNILEVELshare = share.mul(UNILEVEL_PERCENTS[level - 1]).div(
            ↪ PERCENTS_DIVIDER) (tripfoundation.io.sol#1018)
TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
    ↪ #995-1243) performs a multiplication on the result of a division:
        -UNILEVELshare_scope_11 = share_scope_8.mul(UNILEVEL_PERCENTS[
            ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
            ↪ #1060)
        -totalUNILEVELAmount = totalUNILEVELAmount.add(
            ↪ UNILEVELshare_scope_11.mul(to_scope_10.sub(from_scope_9)).
            ↪ div(TIME_STEP)) (tripfoundation.io.sol#1064)
TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
    ↪ #995-1243) performs a multiplication on the result of a division:
        -share_scope_2 = downline.deposits[i_scope_0].tokenamount.mul(
            ↪ downline.deposits[i_scope_0].percent).div(PLANPER_DIVIDER)
            ↪  (tripfoundation.io.sol#1035)
        -UNILEVELshare_scope_5 = share_scope_2.mul(UNILEVEL_PERCENTS[
            ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
            ↪ #1039)
TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
    ↪ #995-1243) performs a multiplication on the result of a division:
        -UNILEVELshare_scope_5 = share_scope_2.mul(UNILEVEL_PERCENTS[
            ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
            ↪ #1039)
        -totalUNILEVELAmount = totalUNILEVELAmount.add(
            ↪ UNILEVELshare_scope_5.mul(to_scope_4.sub(from_scope_3)).
            ↪ div(TIME_STEP)) (tripfoundation.io.sol#1043)
TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
    ↪ #995-1243) performs a multiplication on the result of a division:
        -share_scope_20 = downline.deposits[i_scope_18].tokenamount.mul(
            ↪ downline.deposits[i_scope_18].percent).div(PLANPER_DIVIDER
            ↪ ) (tripfoundation.io.sol#1099)
```

```
        -UNILEVELshare_scope_23 = share_scope_20.mul(UNILEVEL_PERCENTS[
            ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
            ↪ #1103)
    TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
        ↪ #995-1243) performs a multiplication on the result of a division:
        -UNILEVELshare_scope_23 = share_scope_20.mul(UNILEVEL_PERCENTS[
            ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
            ↪ #1103)
        -totalUNILEVELAmount = totalUNILEVELAmount.add(
            ↪ UNILEVELshare_scope_23.mul(to_scope_22.sub(from_scope_21))
            ↪ .div(TIME_STEP)) (tripfoundation.io.sol#1107)
    TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
        ↪ #995-1243) performs a multiplication on the result of a division:
        -share_scope_32 = downline.deposits[i_scope_30].tokenamount.mul(
            ↪ downline.deposits[i_scope_30].percent).div(PLANPER_DIVIDER
            ↪ ) (tripfoundation.io.sol#1141)
        -UNILEVELshare_scope_35 = share_scope_32.mul(UNILEVEL_PERCENTS[
            ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
            ↪ #1145)
    TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
        ↪ #995-1243) performs a multiplication on the result of a division:
        -share_scope_14 = downline.deposits[i_scope_12].tokenamount.mul(
            ↪ downline.deposits[i_scope_12].percent).div(PLANPER_DIVIDER
            ↪ ) (tripfoundation.io.sol#1076)
        -UNILEVELshare_scope_17 = share_scope_14.mul(UNILEVEL_PERCENTS[
            ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
            ↪ #1080)
    TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
        ↪ #995-1243) performs a multiplication on the result of a division:
        -UNILEVELshare_scope_35 = share_scope_32.mul(UNILEVEL_PERCENTS[
            ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
            ↪ #1145)
        -totalUNILEVELAmount = totalUNILEVELAmount.add(
            ↪ UNILEVELshare_scope_35.mul(to_scope_34.sub(from_scope_33))
```

```
            ↪ .div(TIME_STEP)) (tripfoundation.io.sol#1149)
    TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
       ↪ #995-1243) performs a multiplication on the result of a division:
           -UNILEVELshare_scope_17 = share_scope_14.mul(UNILEVEL_PERCENTS[
              ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
              ↪ #1080)
           -totalUNILEVELAmount = totalUNILEVELAmount.add(
              ↪ UNILEVELshare_scope_17.mul(to_scope_16.sub(from_scope_15))
              ↪ .div(TIME_STEP)) (tripfoundation.io.sol#1084)
    TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
       ↪ #995-1243) performs a multiplication on the result of a division:
           -share_scope_38 = downline.deposits[i_scope_36].tokenamount.mul(
              ↪ downline.deposits[i_scope_36].percent).div(PLANPER_DIVIDER
              ↪ ) (tripfoundation.io.sol#1161)
           -UNILEVELshare_scope_41 = share_scope_38.mul(UNILEVEL_PERCENTS[
              ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
              ↪ #1165)
    TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
       ↪ #995-1243) performs a multiplication on the result of a division:
           -UNILEVELshare_scope_41 = share_scope_38.mul(UNILEVEL_PERCENTS[
              ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
              ↪ #1165)
           -totalUNILEVELAmount = totalUNILEVELAmount.add(
              ↪ UNILEVELshare_scope_41.mul(to_scope_40.sub(from_scope_39))
              ↪ .div(TIME_STEP)) (tripfoundation.io.sol#1169)
    TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
       ↪ #995-1243) performs a multiplication on the result of a division:
           -share_scope_26 = downline.deposits[i_scope_24].tokenamount.mul(
              ↪ downline.deposits[i_scope_24].percent).div(PLANPER_DIVIDER
              ↪ ) (tripfoundation.io.sol#1121)
           -UNILEVELshare_scope_29 = share_scope_26.mul(UNILEVEL_PERCENTS[
              ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
              ↪ #1125)
```

```
TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
   ↪ #995-1243) performs a multiplication on the result of a division:
       -UNILEVELshare_scope_29 = share_scope_26.mul(UNILEVEL_PERCENTS[
          ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
          ↪ #1125)
       -totalUNILEVELAmount = totalUNILEVELAmount.add(
          ↪ UNILEVELshare_scope_29.mul(to_scope_28.sub(from_scope_27))
          ↪ .div(TIME_STEP)) (tripfoundation.io.sol#1129)
TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
   ↪ #995-1243) performs a multiplication on the result of a division:
       -share_scope_56 = downline.deposits[i_scope_54].tokenamount.mul(
          ↪ downline.deposits[i_scope_54].percent).div(PLANPER_DIVIDER
          ↪ ) (tripfoundation.io.sol#1221)
       -UNILEVELshare_scope_59 = share_scope_56.mul(UNILEVEL_PERCENTS[
          ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
          ↪ #1225)
TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
   ↪ #995-1243) performs a multiplication on the result of a division:
       -UNILEVELshare_scope_59 = share_scope_56.mul(UNILEVEL_PERCENTS[
          ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
          ↪ #1225)
       -totalUNILEVELAmount = totalUNILEVELAmount.add(
          ↪ UNILEVELshare_scope_59.mul(to_scope_58.sub(from_scope_57))
          ↪ .div(TIME_STEP)) (tripfoundation.io.sol#1229)
TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
   ↪ #995-1243) performs a multiplication on the result of a division:
       -share_scope_50 = downline.deposits[i_scope_48].tokenamount.mul(
          ↪ downline.deposits[i_scope_48].percent).div(PLANPER_DIVIDER
          ↪ ) (tripfoundation.io.sol#1201)
       -UNILEVELshare_scope_53 = share_scope_50.mul(UNILEVEL_PERCENTS[
          ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
          ↪ #1205)
TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
   ↪ #995-1243) performs a multiplication on the result of a division:
```

```
        -share_scope_44 = downline.deposits[i_scope_42].tokenamount.mul(
            ↪ downline.deposits[i_scope_42].percent).div(PLANPER_DIVIDER
            ↪ ) (tripfoundation.io.sol#1181)
        -UNILEVELshare_scope_47 = share_scope_44.mul(UNILEVEL_PERCENTS[
            ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
            ↪ #1185)
TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
    ↪ #995-1243) performs a multiplication on the result of a division:
        -UNILEVELshare_scope_53 = share_scope_50.mul(UNILEVEL_PERCENTS[
            ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
            ↪ #1205)
        -totalUNILEVELAmount = totalUNILEVELAmount.add(
            ↪ UNILEVELshare_scope_53.mul(to_scope_52.sub(from_scope_51))
            ↪ .div(TIME_STEP)) (tripfoundation.io.sol#1209)
TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
    ↪ #995-1243) performs a multiplication on the result of a division:
        -UNILEVELshare_scope_47 = share_scope_44.mul(UNILEVEL_PERCENTS[
            ↪ level - 1]).div(PERCENTS_DIVIDER) (tripfoundation.io.sol
            ↪ #1185)
        -totalUNILEVELAmount = totalUNILEVELAmount.add(
            ↪ UNILEVELshare_scope_47.mul(to_scope_46.sub(from_scope_45))
            ↪ .div(TIME_STEP)) (tripfoundation.io.sol#1189)
TripFoundation.tokensToBnb_(uint256) (tripfoundation.io.sol#1477-1493)
    ↪ performs a multiplication on the result of a division:
        -_etherReceived = (SafeMath.sub((((tokenPriceInitial_ + (
            ↪ tokenPriceDecremental_ * (_tokenSupply / 1e18))) -
            ↪ tokenPriceDecremental_) * (tokens_ - 1e18)),(
            ↪ tokenPriceDecremental_ * ((tokens_ ** 2 - tokens_) / 1e18)
            ↪ ) / 2) / 1e18) (tripfoundation.io.sol#1480-1491)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #divide-before-multiply

TripFoundation.buyPrice() (tripfoundation.io.sol#1430-1437) uses a
    ↪ dangerous strict equality:
```

```
        - tokenSupply_ == 0 (tripfoundation.io.sol#1431)
TripFoundation.sellPrice() (tripfoundation.io.sol#1419-1427) uses a
    ↪ dangerous strict equality:
        - tokenSupply_ == 0 (tripfoundation.io.sol#1420)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dangerous-strict-equalities

Contract locking ether found:
        Contract TripFoundation (tripfoundation.io.sol#148-1504) has
            ↪ payable functions:
         - TripFoundation.fallback() (tripfoundation.io.sol#338-340)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #contracts-that-lock-ether

Reentrancy in TripFoundation.invest(uint256,address) (tripfoundation.io.
    ↪ sol#707-878):
        External calls:
        - token.safeTransferFrom(msg.sender,projectWallet,projectfee) (
            ↪ tripfoundation.io.sol#766)
        - token.safeTransferFrom(msg.sender,developmentWallet,
            ↪ developmentfee) (tripfoundation.io.sol#770)
        - token.safeTransferFrom(msg.sender,tripWallet,tripfee) (
            ↪ tripfoundation.io.sol#774)
        - token.safeTransferFrom(msg.sender,marketingWallet,marketingfee)
            ↪  (tripfoundation.io.sol#778)
        - token.safeTransferFrom(msg.sender,pancakeWallet,pancakefee) (
            ↪ tripfoundation.io.sol#782)
        State variables written after the call(s):
        - user.referrer = referrer (tripfoundation.io.sol#795)
        - users[upline].levels[i] = users[upline].levels[i].add(1) (
            ↪ tripfoundation.io.sol#801)
        - users[upline_scope_0].bonus = users[upline_scope_0].bonus.add(
            ↪ amount) (tripfoundation.io.sol#822)
```

```
        - users[upline_scope_0].totalBonus = users[upline_scope_0].
            ↪ totalBonus.add(amount) (tripfoundation.io.sol#823)
        - users[upline_scope_0].teambusiness = users[upline_scope_0].
            ↪ teambusiness.add(value) (tripfoundation.io.sol#831)
        - users[upline_scope_0].levelbusiness[i_scope_3] = users[
            ↪ upline_scope_0].levelbusiness[i_scope_3].add(value) (
            ↪ tripfoundation.io.sol#838)
        - users[upline_scope_0].RankingReward = _rankingtokens (
            ↪ tripfoundation.io.sol#844)
        - users[upline_scope_0].Reward_achivement[i_scope_3] = true (
            ↪ tripfoundation.io.sol#845)
        - user.checkpoint = block.timestamp (tripfoundation.io.sol#862)
    Reentrancy in TripFoundation.invest(uint256,address) (tripfoundation.io.
        ↪ sol#707-878):
        External calls:
        - token.safeTransferFrom(msg.sender,projectWallet,projectfee) (
            ↪ tripfoundation.io.sol#766)
        - token.safeTransferFrom(msg.sender,developmentWallet,
            ↪ developmentfee) (tripfoundation.io.sol#770)
        - token.safeTransferFrom(msg.sender,tripWallet,tripfee) (
            ↪ tripfoundation.io.sol#774)
        - token.safeTransferFrom(msg.sender,marketingWallet,marketingfee)
            ↪  (tripfoundation.io.sol#778)
        - token.safeTransferFrom(msg.sender,pancakeWallet,pancakefee) (
            ↪ tripfoundation.io.sol#782)
        - token.safeTransferFrom(msg.sender,address(this),finalvalue) (
            ↪ tripfoundation.io.sol#868)
        State variables written after the call(s):
        - user.deposits.push(Deposit(time,percent,value,_amountOfTokens,
            ↪ block.timestamp)) (tripfoundation.io.sol#870)
    Reentrancy in TripFoundation.withdrawprofit() (tripfoundation.io.sol
        ↪ #353-373):
        External calls:
```

```
        - token.transfer(msg.sender,withdrawamt) (tripfoundation.io.sol
            ↪ #368)
        State variables written after the call(s):
        - payouts_[_customerAddress] = 0 (tripfoundation.io.sol#369)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-1

TripFoundation.transferFrom(address,address,uint256) (tripfoundation.io.
    ↪ sol#1394-1402) contains a tautology or contradiction:
        - require(bool)(tokenBalanceLedger_[_from] >= _amount && allowed[
            ↪ _from][msg.sender] >= _amount && _amount >= 0) (
            ↪ tripfoundation.io.sol#1396)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #tautology-or-contradiction

TripFoundation.invest(uint256,address).percent (tripfoundation.io.sol
    ↪ #712) is a local variable never initialized
TripFoundation.getUserDividends(address).totalAmount (tripfoundation.io.
    ↪ sol#973) is a local variable never initialized
TripFoundation.getUserUNILEVELIncome(address).totalUNILEVELAmount (
    ↪ tripfoundation.io.sol#997) is a local variable never initialized
TripFoundation.invest(uint256,address).value (tripfoundation.io.sol#710)
    ↪  is a local variable never initialized
TripFoundation.invest(uint256,address).time (tripfoundation.io.sol#711)
    ↪ is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-local-variables

TripFoundation.withdrawcapita() (tripfoundation.io.sol#881-926) ignores
    ↪ return value by user.withdrawnCapita.add(ttamount) (
    ↪ tripfoundation.io.sol#894)
TripFoundation.withdrawcapita() (tripfoundation.io.sol#881-926) ignores
    ↪ return value by user.withdrawnCapita.add(ttamount) (
    ↪ tripfoundation.io.sol#907)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

TripFoundation.constructor(address,address,address,address,address,
    ↪ address).pancake (tripfoundation.io.sol#320) lacks a zero-check
    ↪ on :
                - pancakeWallet = pancake (tripfoundation.io.sol#324)
TripFoundation.constructor(address,address,address,address,address,
    ↪ address).marketing (tripfoundation.io.sol#320) lacks a zero-check
    ↪  on :
                - marketingWallet = marketing (tripfoundation.io.sol#325)
TripFoundation.constructor(address,address,address,address,address,
    ↪ address).trip (tripfoundation.io.sol#320) lacks a zero-check on :
                - tripWallet = trip (tripfoundation.io.sol#326)
TripFoundation.constructor(address,address,address,address,address,
    ↪ address).project (tripfoundation.io.sol#320) lacks a zero-check
    ↪ on :
                - projectWallet = project (tripfoundation.io.sol#328)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

Reentrancy in TripFoundation.invest(uint256,address) (tripfoundation.io.
    ↪ sol#707-878):
        External calls:
        - token.safeTransferFrom(msg.sender,projectWallet,projectfee) (
            ↪ tripfoundation.io.sol#766)
        - token.safeTransferFrom(msg.sender,developmentWallet,
            ↪ developmentfee) (tripfoundation.io.sol#770)
        - token.safeTransferFrom(msg.sender,tripWallet,tripfee) (
            ↪ tripfoundation.io.sol#774)
        - token.safeTransferFrom(msg.sender,marketingWallet,marketingfee)
            ↪  (tripfoundation.io.sol#778)
        - token.safeTransferFrom(msg.sender,pancakeWallet,pancakefee) (
            ↪ tripfoundation.io.sol#782)
```

```
        State variables written after the call(s):
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
                - RefUser[senderAddress][dataId].refUserAddress =
                    ↪ refUserAddress (tripfoundation.io.sol#424)
                - RefUser[senderAddress][dataId].refLevel = refLevel (
                    ↪ tripfoundation.io.sol#425)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
                - chkLv10 = referralLevel9Address[_referredBy] (
                    ↪ tripfoundation.io.sol#446)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
                - chkLv11 = referralLevel10Address[_referredBy] (
                    ↪ tripfoundation.io.sol#448)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
                - chkLv12 = referralLevel11Address[_referredBy] (
                    ↪ tripfoundation.io.sol#449)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
                - chkLv13 = referralLevel12Address[_referredBy] (
                    ↪ tripfoundation.io.sol#450)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
                - chkLv14 = referralLevel13Address[_referredBy] (
                    ↪ tripfoundation.io.sol#451)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
                - chkLv15 = referralLevel14Address[_referredBy] (
                    ↪ tripfoundation.io.sol#452)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
```

```
            - chkLv16 = referralLevel15Address[_referredBy] (
                ↪ tripfoundation.io.sol#453)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - chkLv17 = referralLevel16Address[_referredBy] (
                ↪ tripfoundation.io.sol#454)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - chkLv18 = referralLevel17Address[_referredBy] (
                ↪ tripfoundation.io.sol#455)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - chkLv19 = referralLevel18Address[_referredBy] (
                ↪ tripfoundation.io.sol#456)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - chkLv2 = referralLevel1Address[_referredBy] (
                ↪ tripfoundation.io.sol#438)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - chkLv20 = referralLevel19Address[_referredBy] (
                ↪ tripfoundation.io.sol#457)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - chkLv21 = referralLevel20Address[_referredBy] (
                ↪ tripfoundation.io.sol#459)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - chkLv22 = referralLevel21Address[_referredBy] (
                ↪ tripfoundation.io.sol#460)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - chkLv23 = referralLevel22Address[_referredBy] (
                ↪ tripfoundation.io.sol#461)
```

```
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
          ↪ .io.sol#813)
            - chkLv24 = referralLevel23Address[_referredBy] (
              ↪ tripfoundation.io.sol#462)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
          ↪ .io.sol#813)
            - chkLv25 = referralLevel24Address[_referredBy] (
              ↪ tripfoundation.io.sol#463)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
          ↪ .io.sol#813)
            - chkLv26 = referralLevel25Address[_referredBy] (
              ↪ tripfoundation.io.sol#464)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
          ↪ .io.sol#813)
            - chkLv27 = referralLevel26Address[_referredBy] (
              ↪ tripfoundation.io.sol#465)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
          ↪ .io.sol#813)
            - chkLv28 = referralLevel27Address[_referredBy] (
              ↪ tripfoundation.io.sol#466)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
          ↪ .io.sol#813)
            - chkLv29 = referralLevel28Address[_referredBy] (
              ↪ tripfoundation.io.sol#467)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
          ↪ .io.sol#813)
            - chkLv3 = referralLevel2Address[_referredBy] (
              ↪ tripfoundation.io.sol#439)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
          ↪ .io.sol#813)
            - chkLv30 = referralLevel29Address[_referredBy] (
              ↪ tripfoundation.io.sol#468)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
          ↪ .io.sol#813)
```

```
            - chkLv4 = referralLevel3Address[_referredBy] (
                ↪ tripfoundation.io.sol#440)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - chkLv5 = referralLevel4Address[_referredBy] (
                ↪ tripfoundation.io.sol#441)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - chkLv6 = referralLevel5Address[_referredBy] (
                ↪ tripfoundation.io.sol#442)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - chkLv7 = referralLevel6Address[_referredBy] (
                ↪ tripfoundation.io.sol#443)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - chkLv8 = referralLevel7Address[_referredBy] (
                ↪ tripfoundation.io.sol#444)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - chkLv9 = referralLevel8Address[_referredBy] (
                ↪ tripfoundation.io.sol#445)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - referralCount_[senderAddress] ++ (tripfoundation.io.sol
                ↪ #422)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - referralLevel10Address[_customerAddress] =
                ↪ referralLevel9Address[_referredBy] (tripfoundation.
                ↪ io.sol#539)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
```

```
            - referralLevel11Address[_customerAddress] =
                ↪ referralLevel10Address[_referredBy] (tripfoundation
                ↪ .io.sol#547)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - referralLevel12Address[_customerAddress] =
                ↪ referralLevel11Address[_referredBy] (tripfoundation
                ↪ .io.sol#555)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - referralLevel13Address[_customerAddress] =
                ↪ referralLevel12Address[_referredBy] (tripfoundation
                ↪ .io.sol#563)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - referralLevel14Address[_customerAddress] =
                ↪ referralLevel13Address[_referredBy] (tripfoundation
                ↪ .io.sol#571)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - referralLevel15Address[_customerAddress] =
                ↪ referralLevel14Address[_referredBy] (tripfoundation
                ↪ .io.sol#579)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - referralLevel16Address[_customerAddress] =
                ↪ referralLevel15Address[_referredBy] (tripfoundation
                ↪ .io.sol#587)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
            ↪ .io.sol#813)
            - referralLevel17Address[_customerAddress] =
                ↪ referralLevel16Address[_referredBy] (tripfoundation
                ↪ .io.sol#595)
```

```
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
           ↪ .io.sol#813)
               - referralLevel18Address[_customerAddress] =
                  ↪ referralLevel17Address[_referredBy] (tripfoundation
                  ↪ .io.sol#603)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
           ↪ .io.sol#813)
               - referralLevel19Address[_customerAddress] =
                  ↪ referralLevel18Address[_referredBy] (tripfoundation
                  ↪ .io.sol#611)
        - distributeRef(referrer,msg.sender,_newReferral) (tripfoundation
           ↪ .io.sol#813)
               - referralLevel1Address[_customerAddress] = _referredBy (
                  ↪ tripfoundation.io.sol#433)

Variable TripFoundation.getUserUNILEVELIncome(address).share_scope_32 (
   ↪ tripfoundation.io.sol#1141) is too similar to TripFoundation.
   ↪ getUserUNILEVELIncome(address).share_scope_38 (tripfoundation.io.
   ↪ sol#1161)
Variable TripFoundation.getUserUNILEVELIncome(address).share_scope_14 (
   ↪ tripfoundation.io.sol#1076) is too similar to TripFoundation.
   ↪ getUserUNILEVELIncome(address).share_scope_44 (tripfoundation.io.
   ↪ sol#1181)
Variable TripFoundation.getUserUNILEVELIncome(address).share_scope_50 (
   ↪ tripfoundation.io.sol#1201) is too similar to TripFoundation.
   ↪ getUserUNILEVELIncome(address).share_scope_56 (tripfoundation.io.
   ↪ sol#1221)
Variable TripFoundation.getUserUNILEVELIncome(address).to_scope_10 (
   ↪ tripfoundation.io.sol#1058) is too similar to TripFoundation.
   ↪ getUserUNILEVELIncome(address).to_scope_16 (tripfoundation.io.sol
   ↪ #1078)
Variable TripFoundation.getUserUNILEVELIncome(address).to_scope_22 (
   ↪ tripfoundation.io.sol#1101) is too similar to TripFoundation.
   ↪ getUserUNILEVELIncome(address).to_scope_28 (tripfoundation.io.sol
```

```
    ↪ #1123)
Variable TripFoundation.getUserUNILEVELIncome(address).to_scope_22 (
    ↪ tripfoundation.io.sol#1101) is too similar to TripFoundation.
    ↪ getUserUNILEVELIncome(address).to_scope_52 (tripfoundation.io.sol
    ↪ #1203)
Variable TripFoundation.getUserUNILEVELIncome(address).to_scope_28 (
    ↪ tripfoundation.io.sol#1123) is too similar to TripFoundation.
    ↪ getUserUNILEVELIncome(address).to_scope_58 (tripfoundation.io.sol
    ↪ #1223)
Variable TripFoundation.getUserUNILEVELIncome(address).to_scope_34 (
    ↪ tripfoundation.io.sol#1143) is too similar to TripFoundation.
    ↪ getUserUNILEVELIncome(address).to_scope_40 (tripfoundation.io.sol
    ↪ #1163)
Variable TripFoundation.getUserUNILEVELIncome(address).to_scope_10 (
    ↪ tripfoundation.io.sol#1058) is too similar to TripFoundation.
    ↪ getUserUNILEVELIncome(address).to_scope_40 (tripfoundation.io.sol
    ↪ #1163)
Variable TripFoundation.getUserUNILEVELIncome(address).to_scope_16 (
    ↪ tripfoundation.io.sol#1078) is too similar to TripFoundation.
    ↪ getUserUNILEVELIncome(address).to_scope_46 (tripfoundation.io.sol
    ↪ #1183)
Variable TripFoundation.getUserUNILEVELIncome(address).to_scope_34 (
    ↪ tripfoundation.io.sol#1143) is too similar to TripFoundation.
    ↪ getUserUNILEVELIncome(address).to_scope_46 (tripfoundation.io.sol
    ↪ #1183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #variable-names-are-too-similar

TripFoundation.distributeRef(address,address,bool) (tripfoundation.io.
    ↪ sol#429-704) uses literals with too many digits:
        - chkLv2 != 0x0000000000000000000000000000000000000000 (
            ↪ tripfoundation.io.sol#474)
TripFoundation.distributeRef(address,address,bool) (tripfoundation.io.
    ↪ sol#429-704) uses literals with too many digits:
```

```
TripFoundation.invest(uint256,address) (tripfoundation.io.sol#707-878)
    ↪ uses literals with too many digits:
        - referralLevel1Address[msg.sender] != 0
            ↪ x0000000000000000000000000000000000000000 (tripfoundation.
            ↪ io.sol#808)
TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
    ↪ #995-1243) uses literals with too many digits:
        - users[userAddress].teambusiness >= 100000 * 10 ** 18 (
            ↪ tripfoundation.io.sol#1156)
TripFoundation.getUserUNILEVELIncome(address) (tripfoundation.io.sol
    ↪ #995-1243) uses literals with too many digits:
        - users[userAddress].teambusiness >= 500000 * 10 ** 18 (
            ↪ tripfoundation.io.sol#1196)
TripFoundation.slitherConstructorVariables() (tripfoundation.io.sol
    ↪ #148-1504) uses literals with too many digits:
        - _initialsupply = 450000000 * 10 ** 18 (tripfoundation.io.sol
            ↪ #186)
TripFoundation.slitherConstructorVariables() (tripfoundation.io.sol
    ↪ #148-1504) uses literals with too many digits:
        - developmentsupply = 50000000 * 10 ** 18 (tripfoundation.io.sol
            ↪ #188)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #too-many-digits

TripFoundation.stakingTime_ (tripfoundation.io.sol#233) is never used in
    ↪  TripFoundation (tripfoundation.io.sol#148-1504)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-state-variable

TripFoundation.developmentsupply (tripfoundation.io.sol#188) should be
    ↪ constant
TripFoundation.name (tripfoundation.io.sol#154) should be constant
TripFoundation.symbol (tripfoundation.io.sol#155) should be constant
```

```
TripFoundation.totalRefBonus (tripfoundation.io.sol#184) should be
    ↪ constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #state-variables-that-could-be-declared-constant

totalSupply() should be declared external:
        - BEP20.totalSupply() (tripfoundation.io.sol#139)
        - TripFoundation.totalSupply() (tripfoundation.io.sol#1259-1261)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (tripfoundation.io.sol#140)
        - TripFoundation.allowance(address,address) (tripfoundation.io.
            ↪ sol#1413-1416)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (tripfoundation.io.
            ↪ sol#141)
        - TripFoundation.transferFrom(address,address,uint256) (
            ↪ tripfoundation.io.sol#1394-1402)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (tripfoundation.io.sol#142)
        - TripFoundation.approve(address,uint256) (tripfoundation.io.sol
            ↪ #1406-1411)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (tripfoundation.io.sol#143)
        - TripFoundation.transfer(address,uint256) (tripfoundation.io.sol
            ↪ #399-414)
withdrawprofit() should be declared external:
        - TripFoundation.withdrawprofit() (tripfoundation.io.sol#353-373)
selltokens(uint256) should be declared external:
        - TripFoundation.selltokens(uint256) (tripfoundation.io.sol
            ↪ #378-394)
withdrawcapita() should be declared external:
        - TripFoundation.withdrawcapita() (tripfoundation.io.sol#881-926)
withdraw() should be declared external:
        - TripFoundation.withdraw() (tripfoundation.io.sol#928-962)
```

```
   Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
      ↪ #public-function-that-could-be-declared-external
   tripfoundation.io.sol analyzed (5 contracts with 78 detectors), 552
      ↪ result(s) found
```

Conclusion:

Most of the vulnerabilities found by the analysis have already been addressed by the smart contract code review. w

6 Conclusion

In this audit, we examined the design and implementation of Trip Foundation contract and discovered issues of minor severity.Trip Foundation team addressed all issues raised in the initial report and implemented the necessary fixes, while classifying the rest as with low-probability of occurrence. BlockHat' auditors advised Trip Foundation Team to maintain a high level of vigilance and to keep those findings in mind in order to avoid any future complications.

BLOCKHAT
SECURITY

For a Contract Audit, contact us at contact@blockhat.io