expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

SAFUSTAKING

TOKEN OVERVIEW

Risk Findings

Severity	Found
● High	0
● Medium	0
● Low	1
● Informational	4

Centralization Risks

Owner Privileges	Description
● Can Owner Set Taxes >25% ?	Not Detected
● Owner needs to enable trading ?	Not Detected
● Can Owner Disable Trades ?	Not Detected
● Can Owner Mint ?	Not Detected
● Can Owner Blacklist ?	Not Detected
● Can Owner set Max Wallet amount ?	Not Detected
● Can Owner Set Max TX amount ?	Not Detected

TABLE OF CONTENTS

OVERVIEW

The Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result	**Passed**
KYC Verification	-
Audit Date	**14 July 2023**

CONTRACT DETAILS

Token Name: SafuStaking

Symbol: SafuStaking

Network: Ethereum

Contract Type: Staking contract

Language: Solidity

Contract Address: ---

Owner's Wallet: ---

Deployer's Wallet: ---

AUDIT METHODOLOGY

Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch , that lead to scams and rugpulls.

Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability

Tools

- DE
- Open Zeppelin
- Code Analyzer
- Solidity Code
- Compiler
- Hardhat

VULNERABILITY CHECKS

Design Logic	Passed
Compiler warnings	Passed
Private user data leaks	Passed
Timestamps dependence	Passed
Integer overflow and underflow	Passed
Race conditions & reentrancy. Cross-function race conditions	Passed
Possible delays in data delivery	Passed
Oracle calls	Passed
Front Running	Passed
DoS with Revert	Passed
DoS with block gas limit	Passed
Methods execution permissions	Passed
Economy model	Passed
Impact of the exchange rate on the logic	Passed
Malicious event log	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross-function race conditions	Passed
Safe Zepplin module	Passed

RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and acces control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

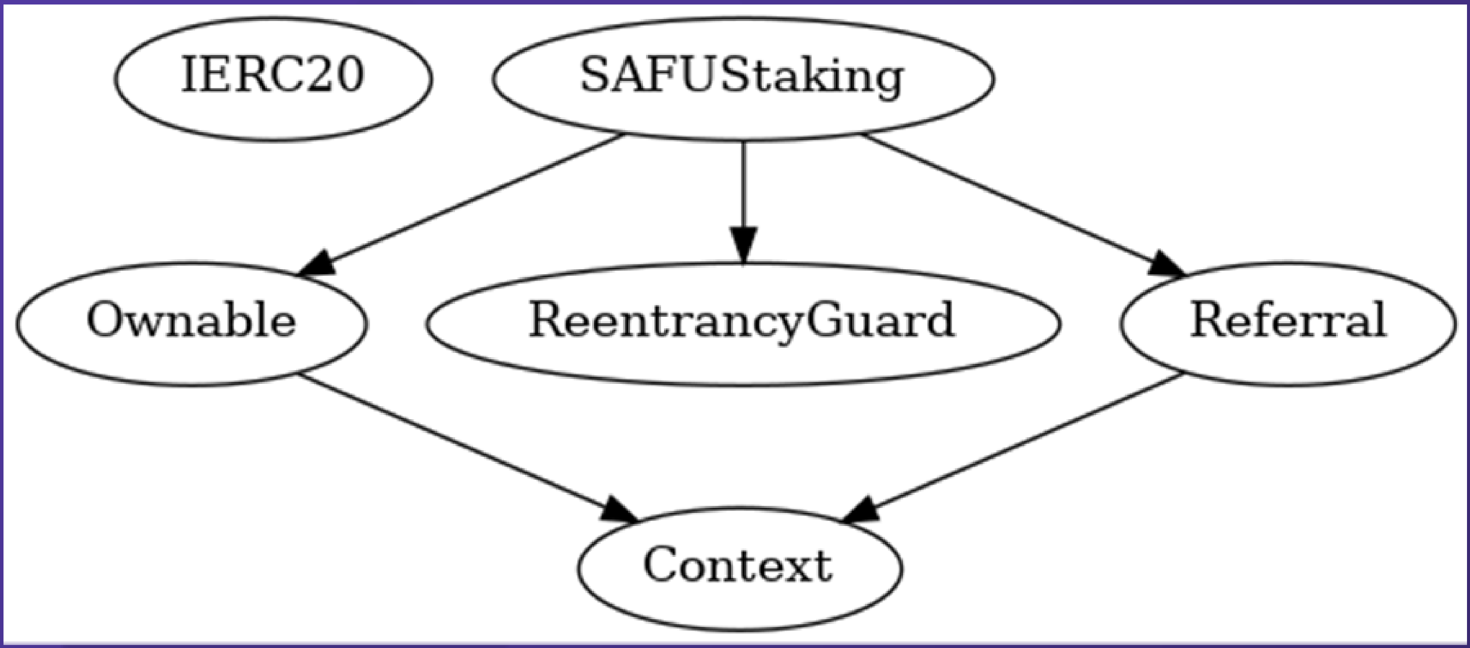
Low Risk

Issues on this level are minor details and warning that can remain unfixed.

Informational

Issues on this level are minor details and warning that can remain unfixed.

INHERITANCE TREES



UNIT TESTS

Referral system:
- referral levels updated correctly (level 1 – level 5)
- referral rewards were calculated correctly (reffered total claims and pending rewards)

  level 1 : 10%
  level 2 : 7%
  level 3 : 5%
  level 4 : 4%
  level 5 : 2%
- refferer could claim their pending tokens, which was calculated as a percentage of (total claimed + total pending rewards) of reffered user

Depositing:
- users could deposit any amount of token (except 0), the contract and user state were mostly updated correctly
- an order created with a locked amount, time and, beneficiary.

Withdraw:
- users could withdraw their locked tokens (by providing an order id) after lock time been passed.
- order pending rewards were calculated correctly and the locked amount sent to the staker.
- **some states were missing to update, like stakeOnPool**

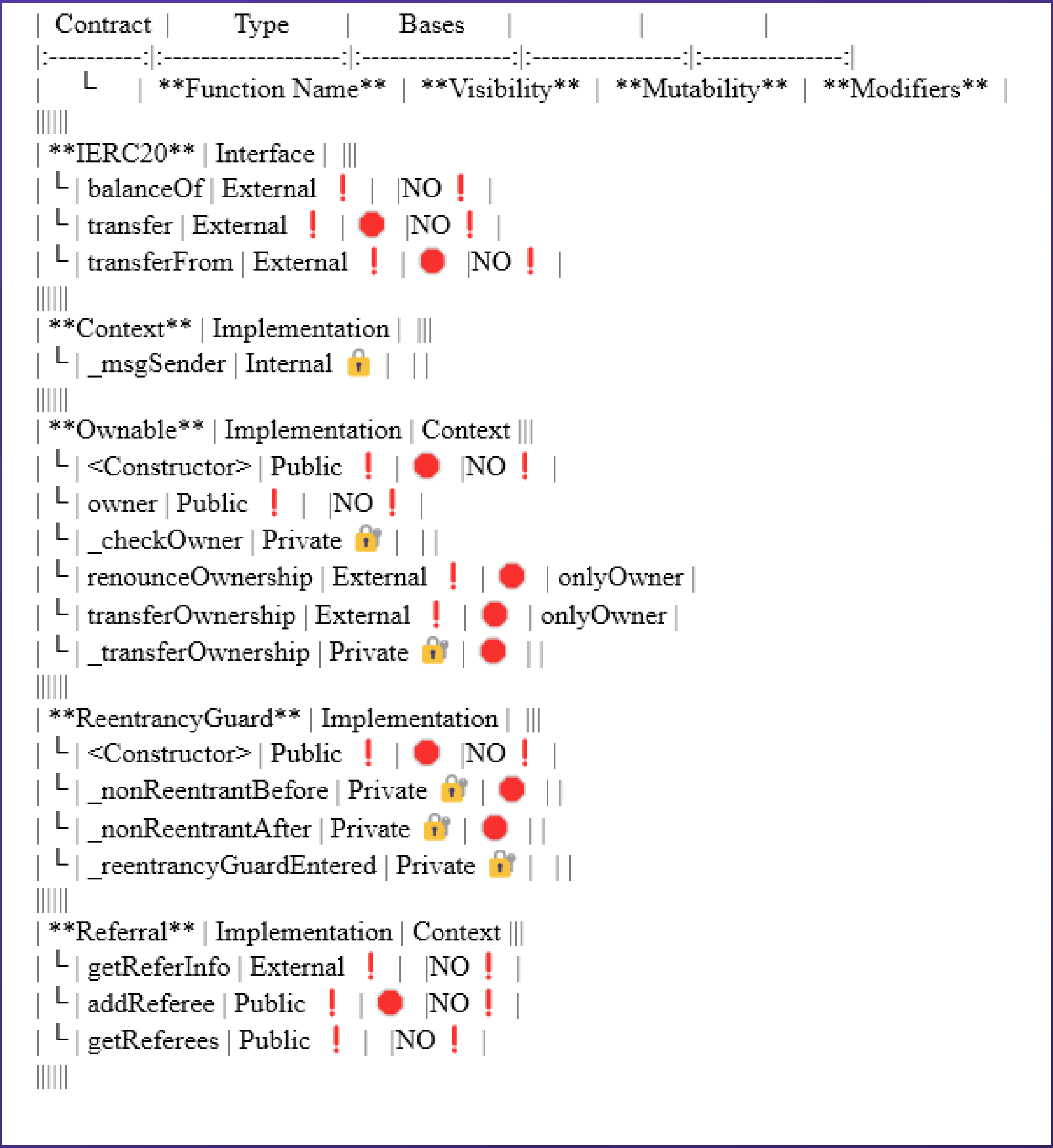
UNIT TESTS

Emergency withdraw:
– users could withdraw their locked tokens after any time, a 25% fee were charged and saved in the contract.
– order pending rewards were calculated correctly and the locked amount sent to the staker.

API:
– during testing, we had to change visibility of some variables to “public’ to be able to retrieve some variables from the contract, including but not limited to:

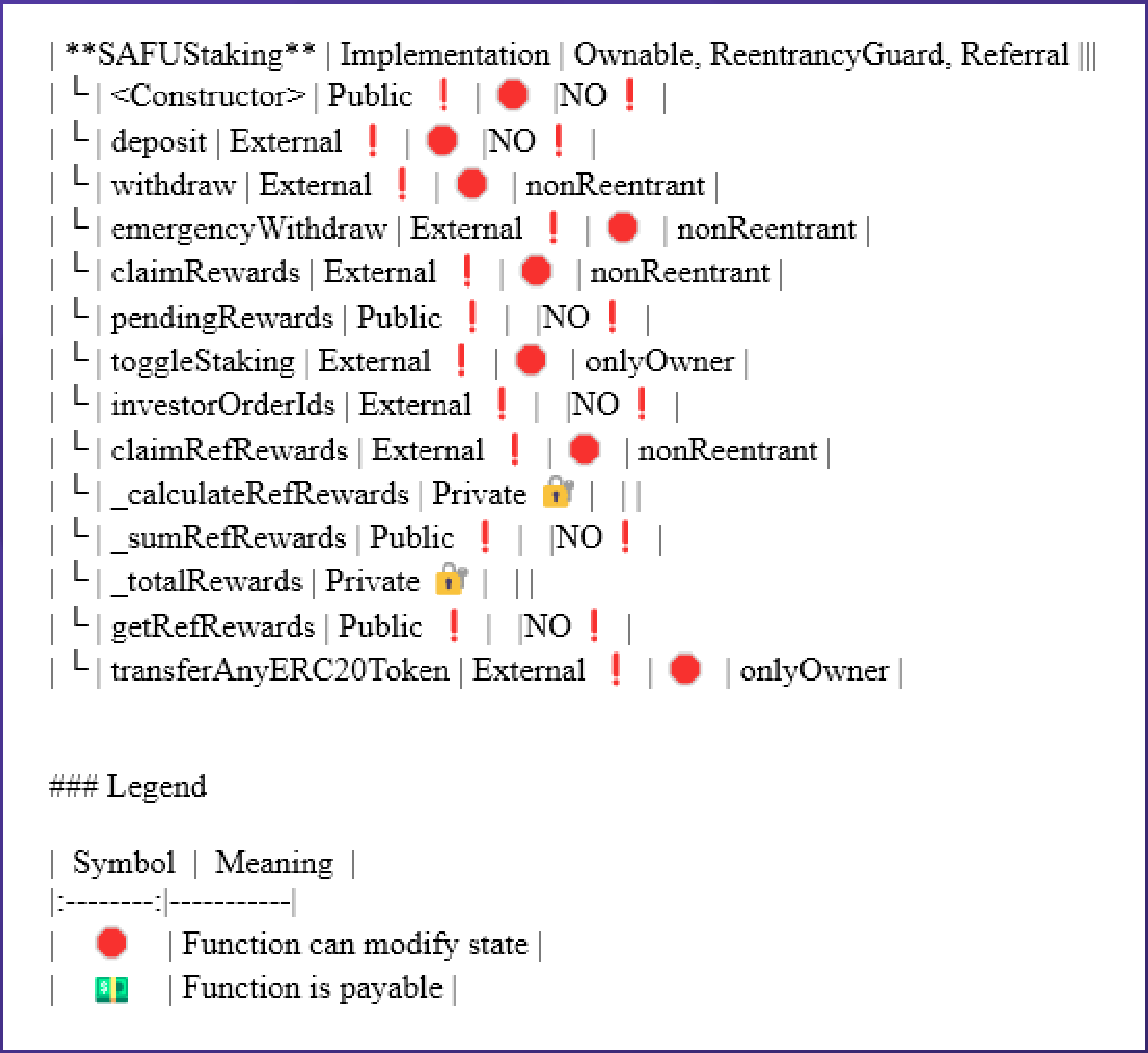
```
  mapping(address => address[]) public referrals_level_1;
  mapping(address => address[]) public referrals_level_2;
  mapping(address => address[]) public referrals_level_3;
  mapping(address => address[]) public referrals_level_4;
  mapping(address => address[]) public referrals_level_5;
```

FUNCTION DETAILS



```
| Contract |        Type        |       Bases      |                  |                 |
|:----------:|:-------------------:|:----------------:|:----------------:|:---------------:|
|     └     | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
||||||
| **IERC20** | Interface |  |||
| └ | balanceOf | External ❗ |   |NO ❗ |
| └ | transfer | External ❗ | 🛑  |NO ❗ |
| └ | transferFrom | External ❗ | 🛑  |NO ❗ |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗ | 🛑  |NO ❗ |
| └ | owner | Public ❗ |   |NO ❗ |
| └ | _checkOwner | Private 🔐 |   | |
| └ | renounceOwnership | External ❗ | 🛑  | onlyOwner |
| └ | transferOwnership | External ❗ | 🛑  | onlyOwner |
| └ | _transferOwnership | Private 🔐 | 🛑  | |
||||||
| **ReentrancyGuard** | Implementation |  |||
| └ | <Constructor> | Public ❗ | 🛑  |NO ❗ |
| └ | _nonReentrantBefore | Private 🔐 | 🛑  | |
| └ | _nonReentrantAfter | Private 🔐 | 🛑  | |
| └ | _reentrancyGuardEntered | Private 🔐 |   | |
||||||
| **Referral** | Implementation | Context |||
| └ | getReferInfo | External ❗ |   |NO ❗ |
| └ | addReferee | Public ❗ | 🛑  |NO ❗ |
| └ | getReferees | Public ❗ |   |NO ❗ |
||||||
```

FUNCTION DETAILS



```
| **SAFUStaking** | Implementation | Ownable, ReentrancyGuard, Referral |||
| └ | <Constructor> | Public ❗ | 🛑 |NO ❗ |
| └ | deposit | External ❗ | 🛑 |NO ❗ |
| └ | withdraw | External ❗ | 🛑 | nonReentrant |
| └ | emergencyWithdraw | External ❗ | 🛑 | nonReentrant |
| └ | claimRewards | External ❗ | 🛑 | nonReentrant |
| └ | pendingRewards | Public ❗ | |NO ❗ |
| └ | toggleStaking | External ❗ | 🛑 | onlyOwner |
| └ | investorOrderIds | External ❗ | |NO ❗ |
| └ | claimRefRewards | External ❗ | 🛑 | nonReentrant |
| └ | _calculateRefRewards | Private 🔐 | ||
| └ | _sumRefRewards | Public ❗ | |NO ❗ |
| └ | _totalRewards | Private 🔐 | ||
| └ | getRefRewards | Public ❗ | |NO ❗ |
| └ | transferAnyERC20Token | External ❗ | 🛑 | onlyOwner |


### Legend

| Symbol | Meaning |
|:--------:|-----------|
| 🛑 | Function can modify state |
| 💵 | Function is payable |
```

MANUAL REVIEW

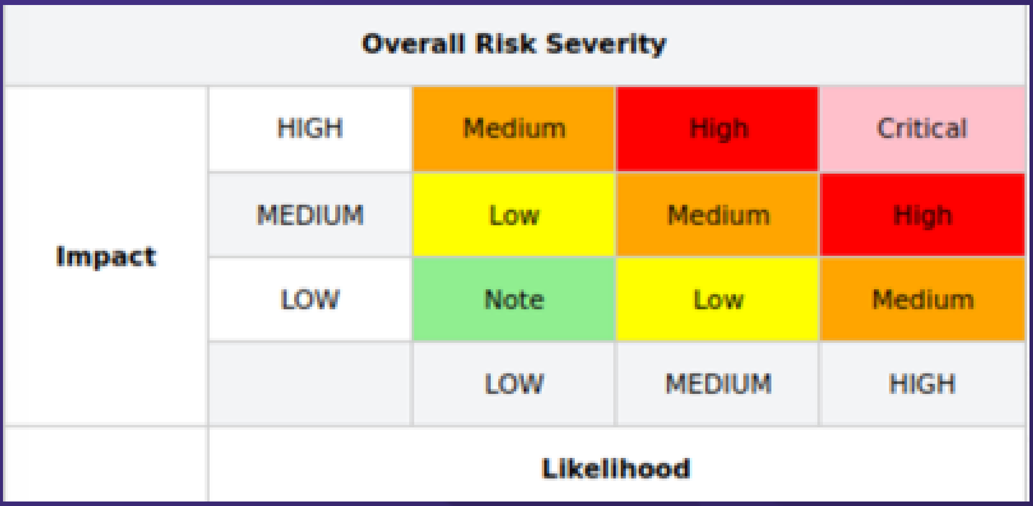
Severity Criteria

Expelee assesses the severity of disclosed vulnerabilities according to methodology based on OWASP standarts.

Vulnerabilities are dividend into three primary risk categroies:
High
Medium
Low

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious input handling
- Escalation of privileges
- Arithmetic
- Gas use



Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

INFORMATIONAL

Updating – stakeOnPool not updated
Severity: Informational
Status: Open

Overview
stakeOnPool mapping stores total staked tokens in each pool. This mapping is not updated when withdrawing or emergency withdrawing funds from the contract

Suggestion
update stakeOnPool mapping in withdraw and emergencyWithdraw functions.

INFORMATIONAL

Updating – using duration for updating rewardOnPool
Severity: Informational
Status: Open

Overview
stakeOnPool mapping uses pool identifiers (30, 180, 365) in order to update total staked tokens in each pool. But rewardOnPool is using actual durations for updating total claimed rewards of each pool

*(using 30 * 86400 etc)*
rewardOnPool[orderInfo.lockupDuration] = rewardOnPool[orderInfo.lockupDuration] + claimAvailable;

Suggestion
for API consistency, use 30, 180, 365 for updating rewardOnPool.

INFORMATIONAL

Updating – emergency withdraw
Severity: Informational
Status: Open

Overview
emergencyWithdraw function is performing all the logic of withdraw funciton plus deducting 25% fee from users. An emergencyWithdraw function should have the logic to allow users for unstaking their tokens without having to go through all the complex logic of other functions (which might be not usable due to an unkonwn bug)

Suggestion
Change the name of emergencyWithdraw function to something like “earlyUnstake’, “exit’ and create an emergencyWithdraw function to allow stakers for withdrawing their tokens freely in emergency situations without receiving rewards or paying fees.
This emergency situation can be enabled only by owner:

```
function enableEmergencyWithdraw() public onlyOwner{
        allowEmergencyWithdraw = true;
}
```

INFORMATIONAL

Updating – emergency withdraw deducting fees after unlock
Severity: Informational
Status: Open

Overview
emergencyWithdraw function is deducting 25% fee from users even if order is unlocked

Suggestion
Its suggested to only deduct 25% fee when order is still locked.

INFORMATIONAL

Updating – constant variables
Severity: Low
Status: Not Resolved

Overview
some variables are immutable and constantly being used. Its suggested to change this vairables to constant.

Suggestion
Change below variables to constant:
uint256 private _30daysPercentage = 15;
uint256 private _180daysPercentage = 35;
uint256 private _365daysPercentage = 100;

ABOUT EXPELEE

Expelee is a product-based aspirational Web3 start-up. Coping up with numerous solutions for blockchain security and constructing a Web3 ecosystem from deal making platform to developer hosting open platform, while also developing our own commercial and sustainable blockchain.

www.expelee.com

expeleeofficial	expelee
Expelee	expelee
expelee_official	expelee-co

Building the Futuristic Blockchain Ecosystem

DISCLAIMER

Building the Futuristic Blockchain Ecosystem