NFTANIA V5

Smart Contract Security Audit

Prepared by ShellBoxes

June 16th, 2022 - June 30th, 2022

Shellboxes.com

contact@shellboxes.com

Document Properties

Client	NFTANIA
Version	1.0
Classification	Public

Scope

The NFTANIA V5 Contract in the NFTANIA V5 Repository

Files	MD5 Hash
crowdsaleWithLiquidity.sol	697e96be4087a9e849793101090bbc16
liquidityLocker.sol	ec43340464ed2e4161abccdfcc101cac
NftaniaAddliquidity.sol	144526d6174f1fa5a5a40a47e2bb05fe
NftaniaAirdropV5.sol	532e2320858a6ef506e11dc8524c63f1
NftaniaTokenWithRoyalites.sol	7c4110df77950e5b6cf4b9311e4b0c45
vestinglocker.sol	2e18b45cdd7de836ae944ba152a15a24

Contacts

COMPANY	EMAIL
ShellBoxes	contact@shellboxes.com

Contents

1 Introduction

NFTANIA engaged ShellBoxes to conduct a security assessment on the NFTANIA V5 beginning on June 16th, 2022 and ending June 30th, 2022. 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.

In response to the initial report, the NFTANIA team addressed all of the issues raised and put in place the necessary corrections and updates. This audit concluded that the version of the contracts that have passed the audit are ready to go live on the mainnet.

1.1 About NFTANIA

Nftania, introduces NFTs as universal tokenizers that can create decentralized ecosystems by tokenizing virtually any asset attribute and not just rarity; these tokens thrive in an ecosystem that provides provenance and proof of "ownership" of these tokens to its members. In principle, these tokens could be tied to any value vector of an asset that NFTs are able to capture then propagate throughout the network.

Issuer	NFTANIA
Website	https://www.nftania.com/
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

ShellBoxes 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 ShellBoxes 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.

Impact				
	High	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low
		High	Medium	Low

Likelihood

2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the NFTANIA V5 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 , 6 medium-severity, 9 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
Missing Transfer Verification	MEDIUM	Fixed
Loss Precision Due To Division	MEDIUM	Fixed
Avoid using .transfer() to transfer Ether	MEDIUM	Fixed
Missing Transfer Verification	MEDIUM	Fixed
Missing Transfer Verification	MEDIUM	Fixed
Loss Precision Due To Division	MEDIUM	Fixed
Missing Address Verification	LOW	Fixed
Floating Pragma	LOW	Fixed
Missing Address Verification	LOW	Fixed
Missing Value Verification	LOW	Fixed
Missing Address Verification	LOW	Fixed
Missing Value Verification	LOW	Fixed
Floating Pragma	LOW	Fixed
Missing Address Verification	LOW	Fixed
Missing Value Verification	LOW	Fixed

3 Finding Details

A crowdsaleWithLiquidity.sol

A.1 Missing Transfer Verification [MEDIUM]

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: crowdsaleWithLiquidity.sol

```
239 function withdrawTokens (IERC20 token , address to , uint256 amount )
        ↪ public onlyOwner {
240     uint256 erc20balance = token.balanceOf (address(this)) ;
241     require (amount <= erc20balance,"balance is low") ;
242     token.transfer(to,amount);
243     emit WithdrawTokens (to, amount);
244 }
```

Risk Level:

Likelihood – 2

Impact - 4

Recommendation:

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

Status - Fixed

The Nftania team has fixed the issue by using the safeERC20 Implementation.

A.2 Loss Precision Due To Division [MEDIUM]

Description:

In the getRemainingTokens function, the remainingTokens is divided by 10e18, the issue here is that if we have the amount less than 10e18 the result will be equal to 0 due to a loss of precision. The same issue goes for the getShares function.

Code:

Listing 2: crowdsaleWithLiquidity.sol

```
187 function getRemainingTokens() public view returns (uint256
        ↪ _remainingTokens){
188     _remainingTokens = remainingTokens / 10**18;
189     return (_remainingTokens);
190 }
```

Listing 3: crowdsaleWithLiquidity.sol

```
323 function getShares(uint256 paymentAmount) internal view returns (uint256
        ↪  poolShare, uint256 revenueShare ) {
324     poolShare = paymentAmount * poolPercent / 1000;
325     revenueShare = paymentAmount - poolShare;
326     return (poolShare, revenueShare);
327 }
```

Risk Level:

Likelihood – 2
Impact - 3

Recommendation:

- It is recommended to return the balance of the contract and in the front(dAPP) you can divide it by 10e18.
- It's recommended to verify if remainingTokens is greater than 10e18.
- It's recommended to verify if paymentAmount * poolPercent is greater than 1000.

Status - Fixed

The Nftania team has fixed the issue by verifying that remainingTokens is greater than 10e18 and paymentAmount * poolPercent is greater than 1000.

A.3 Avoid using .transfer() to transfer Ether [MEDIUM]

Description:

Although transfer() and send() are recommended as a security best-practice to prevent reentrancy attacks because they only forward 2300 gas, the gas repricing of opcodes may break deployed contracts.

Code:

Listing 4: crowdsaleWithLiquidity.sol

```
318 function forwardFunds(uint256 _revenueShare) internal {
319     revenueWallet.transfer(_revenueShare);
320 }
```

Risk Level:

Likelihood – 2
Impact - 3

Recommendation:

Consider using .call{value: ...}("") instead, without hard-coded gas limits along with checks-effects-interactions pattern or reentrancy guards for reentrancy protection.

Status - Fixed

The Nftania team has fixed the issue by using .call{ value: ... }("") instead of .transfer().

B NftaniaAddliquidity.sol

B.1 Missing Transfer Verification [MEDIUM]

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 5: NftaniaAddliquidity.sol

```
22 function addLiquidityETH (address _token, uint tokenAmount, uint
       ↪ EthAmount, address _beneficiary ) external payable
23     returns (uint amountToken, uint amountETH, uint amountliquidity,
           ↪ uint totalLiquidity, address _pairAddress) {
24     token = _token;
25     beneficiary = _beneficiary;
26     IERC20(token).transferFrom(msg.sender, address(this), tokenAmount);
           ↪ // add tokens to liquidty creation contract
27     IERC20(token).approve(ROUTER, tokenAmount); // approve router
           ↪ contract to spend tokens

29     (amountToken, amountETH, amountliquidity) =
30     IUniswapV2Router(ROUTER).addLiquidityETH {value:EthAmount} (
31       token, // token Address
32       tokenAmount, // tokens amount to be added
33       0, // min tokens to be added
```

```
34          0, // min tokens to be added
35          beneficiary, // liquidity tokens recieving address
36          block.timestamp+120 // Deadline for liquidty addition
37      );

39      pairAddress = IUniswapV2Factory(FACTORY).getPair(token, WETH);
40      totalLiquidity = IERC20(pairAddress).balanceOf(beneficiary);
41      emit LiquidityAdded(amountToken, amountETH, amountliquidity,
            ↪ totalLiquidity, pairAddress);
42      return (amountToken, amountETH, amountliquidity, totalLiquidity,
            ↪ pairAddress);
43  }
```

Risk Level:

Likelihood – 2
Impact - 4

Recommendation:

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

Status - Fixed

The Nftania team has fixed the issue by using the safeERC20 Implementation.

B.2 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible.In the addLiquidityETH function, the contract should verify that `_token` and `_beneficiary` are different from address(0).

Code:

Listing 6: NftaniaAddliquidity.sol

```
22 function addLiquidityETH (address _token, uint tokenAmount, uint
       ↪ EthAmount, address _beneficiary ) external payable
23     returns (uint amountToken, uint amountETH, uint amountliquidity,
           ↪ uint totalLiquidity, address _pairAddress) {
24     token = _token;
25     beneficiary = _beneficiary;
26     IERC20(token).transferFrom(msg.sender, address(this), tokenAmount);
           ↪ // add tokens to liquidty creation contract
27     IERC20(token).approve(ROUTER, tokenAmount);
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is recommended to make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Nftania Team has fixed the issue by adding a require statement to verify that _token and _beneficiary are different from the address(0).

B.3 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8. Contracts should be deployed using the same compiler version.

Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 7: NftaniaAddliquidity.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8;
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Fixed

The Nftania Team has fixed the issue by locking the pragma version to 0.8.15.

C NftaniaAirdropV5.sol

C.1 Missing Transfer Verification [MEDIUM]

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 8: NftaniaAirdropV5.sol

```
260     function withdrawTokens (IERC20 token , address to , uint256 amount
           ↪ ) public onlyOwner {
261         uint256 erc20balance = token.balanceOf (address(this)) ;
262         require (amount <= erc20balance,"balance is low") ;
263         token.transfer(to,amount);
264         emit WithdrawTokens (to, amount);
265     }
```

Risk Level:

Likelihood – 2
Impact - 4

Recommendation:

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

Status - Fixed

The Nftania team has fixed the issue by using the safeERC20 Implementation.

D NftaniaTokenWithRoyalites.sol

D.1 Loss Precision Due To Division [MEDIUM]

Description:

In the calculateRoyalty function, the amount * royaltyRate is divided by 1000, the issue here is that if we have the amount less than 10e18 the result will be equal to 0 due to a loss of precision.

Code:

Listing 9: NftaniaTokenWithRoyalites.sol

```
249 function calculateRoyalty (address from, address to, uint256 amount)
        ↪ internal view
250     returns (uint256 royaltyAmount, uint256 transferAmount){
251     if (royaltyExemptTo[to]){
252         royaltyAmount = 0;
253         transferAmount = amount;
254     }
255     else if (royaltyExemptFrom[from]){
256         royaltyAmount = 0;
257         transferAmount = amount;
258     }
259     else{
260         royaltyAmount = amount * royaltyRate / 1000;
261         transferAmount = amount - royaltyAmount;
262     }
263     return (royaltyAmount, transferAmount);
264 }
```

Risk Level:

Likelihood – 2

Impact - 3

Recommendation:

It's recommended to verify if amount * royaltyRate is greater than 1000.

Status - Fixed

The Nftania Team has fixed the issue by adding a require statement to verify that amount * royaltyRate is greater than 1000.

D.2 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible.In the constructor, the contract should verify that _royaltyWallet is different from the address(0).

Code:

Listing 10: NftaniaTokenWithRoyalites.sol

```
112 constructor(uint _tokensReleaseDate, address _ADDR_TEAM_lOCKER, address
        ↪ _ADDR_TEAM, address _ADDR_INVESTORS_lOCKER, address
        ↪ _ADDR_INVESTORS, address _royaltyWallet, uint256 _royaltyRate)
113 ERC20("Nftania NFT2.0", "NFT2") ERC20Permit("Nftania NFT2.0") {
114     ADDR_TEAM_lOCKER = _ADDR_TEAM_lOCKER;
115     ADDR_INVESTORS_lOCKER = _ADDR_INVESTORS_lOCKER;
116     ADDR_TEAM = _ADDR_TEAM;
117     ADDR_INVESTORS = _ADDR_INVESTORS;
118     royaltyWallet = _royaltyWallet;
119     royaltyRate = _royaltyRate ;
120     require(ADDR_TEAM_lOCKER != address(0),"NftaniaToken: Address is the
            ↪  zero address");
121     require(ADDR_INVESTORS_lOCKER != address(0),"NftaniaToken: Address
            ↪ is the zero address");
```

```
122     require(ADDR_TEAM != address(0),"NftaniaToken: Address is the zero
            ↪ address");
123     require(ADDR_INVESTORS != address(0),"NftaniaToken: Address is the
            ↪ zero address");
```

Risk Level:

Likelihood – 2
Impact - 4

Recommendation:

It is recommended to make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Nftania Team has fixed the issue by adding a require statement to verify that _royaltyWallet is different from the address(0).

D.3 Missing Value Verification [LOW]

Description:

Certain functions lack a safety check in the values, the values of the arguments should be verified to allow only the ones that go with the contract's logic. In the constructor of the contract, the royaltyRate variable is not verified to be less than 100, also, the tokensReleaseDate and the newTokenReleaseDate variables are not verified to be greater than now.

Code:

Listing 11: NftaniaTokenWithRoyalites.sol

```
112 constructor(uint _tokensReleaseDate, address _ADDR_TEAM_LOCKER, address
        ↪ _ADDR_TEAM, address _ADDR_INVESTORS_LOCKER, address
        ↪ _ADDR_INVESTORS, address _royaltyWallet, uint256 _royaltyRate)
```

```
113 ERC20("Nftania NFT2.0", "NFT2") ERC20Permit("Nftania NFT2.0") {
114     ADDR_TEAM_LOCKER = _ADDR_TEAM_LOCKER;
115     ADDR_INVESTORS_LOCKER = _ADDR_INVESTORS_LOCKER;
116     ADDR_TEAM = _ADDR_TEAM;
117     ADDR_INVESTORS = _ADDR_INVESTORS;
118     royaltyWallet = _royaltyWallet;
119     royaltyRate = _royaltyRate ;
```

Listing 12: NftaniaTokenWithRoyalites.sol

```
174 function updateTokenReleaseDate(uint newTokenReleaseDate) onlyOwner
    ↪ freezeNotExpired public {
175     require (newTokenReleaseDate < tokensReleaseDate,"Nftania Token:
        ↪ Token release date cannot be delayed");
176     tokensReleaseDate= newTokenReleaseDate;
177     emit TokenReleaseDateUpdate (newTokenReleaseDate);
178 }
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

Consider verifying the `_royaltyRate` to be less than 100, also, verifying the tokensRelease-Date and the newTokenReleaseDate variables to be greater than now.

Status - Fixed

The Nftania Team has fixed the issue by adding require statements to verify the values of the arguments.

E vestinglocker.sol

E.1 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible.The contract should verify that `_beneficiaryWallet` and `_token` are different from the address(0).

Code:

Listing 13: vestinglocker.sol

```
79 constructor(uint256 _amount, address _beneficiaryWallet, string memory
       ↪ _beneficiaryName) {
80     owner = msg.sender;
81     beneficiaryWallet = _beneficiaryWallet;
82     beneficiaryName = _beneficiaryName;
83     setInstalmentsTimes();
84     setInstalmentsAmounts(_amount);
85 }
```

Listing 14: vestinglocker.sol

```
90 function setToken (IERC20 _token) public {
91     require(owner == msg.sender,"This function can be called by owner
           ↪ only");
92     token = _token;
93 }
```

Risk Level:

Likelihood – 1

Impact - 3

Recommendation:

It is recommended to make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Nftania Team has fixed the issue by adding a require statement to verify that _token and _beneficiary are different from the address(0).

E.2 Missing Value Verification [LOW]

Description:

Certain functions lack a safety check in the values, the values of the arguments should be verified to allow only the ones that go with the contract's logic. In the setInstalmentsAmounts function, the _amount is not verified to be greater than 0.

Code:

Listing 15: vestinglocker.sol

```
106 function setInstalmentsAmounts(uint256 _amount) internal {
107     uint256 instalment = (_amount * 10**18 )/5;
108     instalmentsAmount[1] = instalment;
109     instalmentsAmount[2] = instalment;
110     instalmentsAmount[3] = instalment;
111     instalmentsAmount[4] = instalment;
112     instalmentsAmount[5] = instalment;
113 }
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

Consider verifying the _amount to be different from 0.

Status - Fixed

The Nftania team has fixed the issue by adding a require statement to verify that _amount to be different from 0.

E.3 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.0. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 16: vestinglocker.sol

```
62 // SPDX-License-Identifier: MIT
63 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Fixed

The Nftania Team has fixed the issue by locking the pragma version to 0.8.15.

F liquidityLocker.sol

F.1 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible.The contract should verify that `_beneficiaryWallet` and `_token` are different from the address(0).

Code:

Listing 17: liquidityLocker.sol

```
76 constructor(string memory _beneficiaryName, uint256 _unlockTime, address
       ↪  _beneficiaryWallet) {
77     owner = msg.sender;
78     beneficiaryWallet = _beneficiaryWallet;
79     beneficiaryName = _beneficiaryName;
80     unlockTime = _unlockTime;
81 }
```

Listing 18: liquidityLocker.sol

```
86 function setToken (IERC20 _token) public {
87     require(owner == msg.sender,"This function can be called by owner
           ↪ only");
88     token = _token;
89 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is recommended to make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Nftania Team has fixed the issue by adding a require statement to verify that _token and _beneficiary are different from the address(0).

F.2 Missing Value Verification [LOW]

Description:

Certain functions lack a safety check in the values, the values of the arguments should be verified to allow only the ones that go with the contract's logic. In the constructor, the _unlockTime variable is not verified to be greater than now.

Code:

Listing 19: liquidityLocker.sol

```
76 constructor(string memory _beneficiaryName, uint256 _unlockTime, address
       ↪  _beneficiaryWallet) {
77     owner = msg.sender;
78     beneficiaryWallet = _beneficiaryWallet;
79     beneficiaryName = _beneficiaryName;
80     unlockTime = _unlockTime;
81 }
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

Consider verifying the `_unlockTime` to be greater than `now`.

Status - Fixed

The Nftania Team has fixed the issue by adding a `require` statement to verify that `_unlockTime` is greater than `block.timestamp`.

F.3 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma `0.8.0`. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 20: liquidityLocker.sol

```
62 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Fixed

The Nftania Team has fixed the issue by locking the pragma version to 0.8.15.

4 Static Analysis (Slither)

Description:

ShellBoxes 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:

```
NftaniaNFT2.withdrawTokens(IERC20,address,uint256) (
    ↪ NftaniaTokenWithRoyalites.sol#212-217) ignores return value by
    ↪ token.transfer(to,amount) (NftaniaTokenWithRoyalites.sol#215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unchecked-transfer

ERC20Permit.constructor(string).name (../openzeppelin-contracts/
    ↪ contracts/token/ERC20/extensions/draft-ERC20Permit.sol#35)
    ↪ shadows:
        - ERC20.name() (../openzeppelin-contracts/contracts/token/ERC20/
            ↪ ERC20.sol#61-63) (function)
        - IERC20Metadata.name() (../openzeppelin-contracts/contracts/
            ↪ token/ERC20/extensions/IERC20Metadata.sol#16) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #local-variable-shadowing

NftaniaNFT2.updateRoyalty(uint256) (NftaniaTokenWithRoyalites.sol
    ↪ #220-223) should emit an event for:
        - royaltyRate = _royaltyRate (NftaniaTokenWithRoyalites.sol#222)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-events-arithmetic
```

```
NftaniaNFT2.constructor(uint256,address,address,address,address,address,
    ↪ uint256)._ADDR_TEAM_lOCKER (NftaniaTokenWithRoyalites.sol#112)
    ↪ lacks a zero-check on :
                - ADDR_TEAM_lOCKER = _ADDR_TEAM_lOCKER (
                    ↪ NftaniaTokenWithRoyalites.sol#114)
NftaniaNFT2.constructor(uint256,address,address,address,address,address,
    ↪ uint256)._ADDR_INVESTORS_lOCKER (NftaniaTokenWithRoyalites.sol
    ↪ #112) lacks a zero-check on :
                - ADDR_INVESTORS_lOCKER = _ADDR_INVESTORS_lOCKER (
                    ↪ NftaniaTokenWithRoyalites.sol#115)
NftaniaNFT2.constructor(uint256,address,address,address,address,address,
    ↪ uint256)._ADDR_TEAM (NftaniaTokenWithRoyalites.sol#112) lacks a
    ↪ zero-check on :
                - ADDR_TEAM = _ADDR_TEAM (NftaniaTokenWithRoyalites.sol
                    ↪ #116)
NftaniaNFT2.constructor(uint256,address,address,address,address,address,
    ↪ uint256)._ADDR_INVESTORS (NftaniaTokenWithRoyalites.sol#112)
    ↪ lacks a zero-check on :
                - ADDR_INVESTORS = _ADDR_INVESTORS (
                    ↪ NftaniaTokenWithRoyalites.sol#117)
NftaniaNFT2.constructor(uint256,address,address,address,address,address,
    ↪ uint256)._royaltyWallet (NftaniaTokenWithRoyalites.sol#112) lacks
    ↪  a zero-check on :
                - royaltyWallet = _royaltyWallet (
                    ↪ NftaniaTokenWithRoyalites.sol#118)
NftaniaNFT2.withdraw(uint256,address).receivingWallet (
    ↪ NftaniaTokenWithRoyalites.sol#205) lacks a zero-check on :
                - receivingWallet.transfer(amount) (
                    ↪ NftaniaTokenWithRoyalites.sol#207)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation
```

```
Variable 'ECDSA.tryRecover(bytes32,bytes).r (../openzeppelin-contracts/
    ↪ contracts/utils/cryptography/ECDSA.sol#59)' in ECDSA.tryRecover(
    ↪ bytes32,bytes) (../openzeppelin-contracts/contracts/utils/
    ↪ cryptography/ECDSA.sol#54-83) potentially used before declaration
    ↪ : r = mload(uint256)(signature + 0x20) (../openzeppelin-contracts
    ↪ /contracts/utils/cryptography/ECDSA.sol#76)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #pre-declaration-usage-of-local-variables

Reentrancy in NftaniaNFT2.withdrawTokens(IERC20,address,uint256) (
    ↪ NftaniaTokenWithRoyalites.sol#212-217):
        External calls:
        - token.transfer(to,amount) (NftaniaTokenWithRoyalites.sol#215)
        Event emitted after the call(s):
        - WithdrawTokens(to,amount) (NftaniaTokenWithRoyalites.sol#216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32
    ↪ ) (../openzeppelin-contracts/contracts/token/ERC20/extensions/
    ↪ draft-ERC20Permit.sol#40-59) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline,ERC20Permit:
            ↪ expired deadline) (../openzeppelin-contracts/contracts/
            ↪ token/ERC20/extensions/draft-ERC20Permit.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #block-timestamp

Address.isContract(address) (../openzeppelin-contracts/contracts/utils/
    ↪ Address.sol#26-36) uses assembly
        - INLINE ASM (../openzeppelin-contracts/contracts/utils/Address.
            ↪ sol#32-34)
Address.verifyCallResult(bool,bytes,string) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#195-215) uses assembly
```

```
        - INLINE ASM (../openzeppelin-contracts/contracts/utils/Address.
            ↪ sol#207-210)
ECDSA.tryRecover(bytes32,bytes) (../openzeppelin-contracts/contracts/
    ↪ utils/cryptography/ECDSA.sol#54-83) uses assembly
        - INLINE ASM (../openzeppelin-contracts/contracts/utils/
            ↪ cryptography/ECDSA.sol#64-68)
        - INLINE ASM (../openzeppelin-contracts/contracts/utils/
            ↪ cryptography/ECDSA.sol#75-78)
ECDSA.tryRecover(bytes32,bytes32,bytes32) (../openzeppelin-contracts/
    ↪ contracts/utils/cryptography/ECDSA.sol#112-124) uses assembly
        - INLINE ASM (../openzeppelin-contracts/contracts/utils/
            ↪ cryptography/ECDSA.sol#119-122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

NftaniaNFT2.addFromExempt(address) (NftaniaTokenWithRoyalites.sol
    ↪ #237-240) compares to a boolean constant:
        -require(bool,string)(royaltyExemptFrom[targetAddress] == false,
            ↪ this address is already (From) exempted) (
            ↪ NftaniaTokenWithRoyalites.sol#238)
NftaniaNFT2.addToExempt(address) (NftaniaTokenWithRoyalites.sol#243-246)
    ↪  compares to a boolean constant:
        -require(bool,string)(royaltyExemptTo[targetAddress] == false,
            ↪ this address is already (To) exempted) (
            ↪ NftaniaTokenWithRoyalites.sol#244)
NftaniaNFT2._beforeTokenTransfer(address,address,uint256) (
    ↪ NftaniaTokenWithRoyalites.sol#285-297) compares to a boolean
    ↪ constant:
        -paused() == false (NftaniaTokenWithRoyalites.sol#286)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #boolean-equality

Different versions of Solidity is used:
        - Version used: ['0.8.0', '^0.8.0']
```

```
- 0.8.0 (NftaniaTokenWithRoyalites.sol#61)
- ^0.8.0 (../openzeppelin-contracts/contracts/access/Ownable.sol
  ↪ #3)
- ^0.8.0 (../openzeppelin-contracts/contracts/security/Pausable.
  ↪ sol#3)
- ^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/ERC20.
  ↪ sol#3)
- ^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/IERC20.
  ↪ sol#3)
- ^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
  ↪ extensions/ERC20Burnable.sol#3)
- ^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
  ↪ extensions/ERC20Snapshot.sol#3)
- ^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
  ↪ extensions/IERC20Metadata.sol#3)
- ^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
  ↪ extensions/draft-ERC20Permit.sol#3)
- ^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
  ↪ extensions/draft-IERC20Permit.sol#3)
- ^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/utils/
  ↪ SafeERC20.sol#3)
- ^0.8.0 (../openzeppelin-contracts/contracts/utils/Address.sol
  ↪ #3)
- ^0.8.0 (../openzeppelin-contracts/contracts/utils/Arrays.sol#3)
- ^0.8.0 (../openzeppelin-contracts/contracts/utils/Context.sol
  ↪ #3)
- ^0.8.0 (../openzeppelin-contracts/contracts/utils/Counters.sol
  ↪ #3)
- ^0.8.0 (../openzeppelin-contracts/contracts/utils/cryptography/
  ↪ ECDSA.sol#3)
- ^0.8.0 (../openzeppelin-contracts/contracts/utils/cryptography/
  ↪ draft-EIP712.sol#3)
- ^0.8.0 (../openzeppelin-contracts/contracts/utils/math/Math.sol
  ↪ #3)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #different-pragma-directives-are-used

Address.functionCall(address,bytes) (../openzeppelin-contracts/contracts
    ↪ /utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#89-95) is never used and should be
    ↪ removed
Address.functionCallWithValue(address,bytes,uint256) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#108-114) is never used and
    ↪ should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (../
    ↪ openzeppelin-contracts/contracts/utils/Address.sol#122-133) is
    ↪ never used and should be removed
Address.functionDelegateCall(address,bytes) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#168-170) is never used and should be
    ↪ removed
Address.functionDelegateCall(address,bytes,string) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#178-187) is never used and
    ↪ should be removed
Address.functionStaticCall(address,bytes) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#141-143) is never used and should be
    ↪ removed
Address.functionStaticCall(address,bytes,string) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#151-160) is never used and
    ↪ should be removed
Address.isContract(address) (../openzeppelin-contracts/contracts/utils/
    ↪ Address.sol#26-36) is never used and should be removed
Address.sendValue(address,uint256) (../openzeppelin-contracts/contracts/
    ↪ utils/Address.sol#54-59) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#195-215) is never used and should be
    ↪ removed
```

```
Context._msgData() (../openzeppelin-contracts/contracts/utils/Context.
    ↪ sol#20-22) is never used and should be removed
Counters.decrement(Counters.Counter) (../openzeppelin-contracts/
    ↪ contracts/utils/Counters.sol#31-37) is never used and should be
    ↪ removed
Counters.reset(Counters.Counter) (../openzeppelin-contracts/contracts/
    ↪ utils/Counters.sol#39-41) is never used and should be removed
ECDSA.recover(bytes32,bytes) (../openzeppelin-contracts/contracts/utils/
    ↪ cryptography/ECDSA.sol#99-103) is never used and should be
    ↪ removed
ECDSA.recover(bytes32,bytes32,bytes32) (../openzeppelin-contracts/
    ↪ contracts/utils/cryptography/ECDSA.sol#131-139) is never used and
    ↪  should be removed
ECDSA.toEthSignedMessageHash(bytes32) (../openzeppelin-contracts/
    ↪ contracts/utils/cryptography/ECDSA.sol#201-205) is never used and
    ↪  should be removed
ECDSA.tryRecover(bytes32,bytes) (../openzeppelin-contracts/contracts/
    ↪ utils/cryptography/ECDSA.sol#54-83) is never used and should be
    ↪ removed
ECDSA.tryRecover(bytes32,bytes32,bytes32) (../openzeppelin-contracts/
    ↪ contracts/utils/cryptography/ECDSA.sol#112-124) is never used and
    ↪  should be removed
Math.ceilDiv(uint256,uint256) (../openzeppelin-contracts/contracts/utils
    ↪ /math/Math.sol#38-41) is never used and should be removed
Math.max(uint256,uint256) (../openzeppelin-contracts/contracts/utils/
    ↪ math/Math.sol#12-14) is never used and should be removed
Math.min(uint256,uint256) (../openzeppelin-contracts/contracts/utils/
    ↪ math/Math.sol#19-21) is never used and should be removed
SafeERC20._callOptionalReturn(IERC20,bytes) (../openzeppelin-contracts/
    ↪ contracts/token/ERC20/utils/SafeERC20.sol#87-97) is never used
    ↪ and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (../openzeppelin-contracts
    ↪ /contracts/token/ERC20/utils/SafeERC20.sol#44-57) is never used
    ↪ and should be removed
```

```
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (../openzeppelin
    ↪ -contracts/contracts/token/ERC20/utils/SafeERC20.sol#68-79) is
    ↪ never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (../openzeppelin
    ↪ -contracts/contracts/token/ERC20/utils/SafeERC20.sol#59-66) is
    ↪ never used and should be removed
SafeERC20.safeTransfer(IERC20,address,uint256) (../openzeppelin-
    ↪ contracts/contracts/token/ERC20/utils/SafeERC20.sol#20-26) is
    ↪ never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (../
    ↪ openzeppelin-contracts/contracts/token/ERC20/utils/SafeERC20.sol
    ↪ #28-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version0.8.0 (NftaniaTokenWithRoyalites.sol#61) necessitates a
    ↪ version too recent to be trusted. Consider deploying with
    ↪ 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/access/Ownable
    ↪ .sol#3) necessitates a version too recent to be trusted. Consider
    ↪  deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/security/
    ↪ Pausable.sol#3) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
    ↪ ERC20.sol#3) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
    ↪ IERC20.sol#3) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
    ↪ extensions/ERC20Burnable.sol#3) necessitates a version too recent
    ↪  to be trusted. Consider deploying with 0.6.12/0.7.6
```

```
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
   ↪ extensions/ERC20Snapshot.sol#3) necessitates a version too recent
   ↪  to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
   ↪ extensions/IERC20Metadata.sol#3) necessitates a version too
   ↪ recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
   ↪ extensions/draft-ERC20Permit.sol#3) necessitates a version too
   ↪ recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
   ↪ extensions/draft-IERC20Permit.sol#3) necessitates a version too
   ↪ recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
   ↪ utils/SafeERC20.sol#3) necessitates a version too recent to be
   ↪ trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/utils/Address.
   ↪ sol#3) necessitates a version too recent to be trusted. Consider
   ↪ deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/utils/Arrays.
   ↪ sol#3) necessitates a version too recent to be trusted. Consider
   ↪ deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/utils/Context.
   ↪ sol#3) necessitates a version too recent to be trusted. Consider
   ↪ deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/utils/Counters
   ↪ .sol#3) necessitates a version too recent to be trusted. Consider
   ↪  deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/utils/
   ↪ cryptography/ECDSA.sol#3) necessitates a version too recent to be
   ↪  trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/utils/
   ↪ cryptography/draft-EIP712.sol#3) necessitates a version too
   ↪ recent to be trusted. Consider deploying with 0.6.12/0.7.6
```

```
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/utils/math/
    ↪ Math.sol#3) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (../openzeppelin-
            ↪ contracts/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (../openzeppelin-contracts/contracts/utils/Address.sol
    ↪ #122-133):
        - (success,returndata) = target.call{value: value}(data) (../
            ↪ openzeppelin-contracts/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (../
    ↪ openzeppelin-contracts/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (../openzeppelin
            ↪ -contracts/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string)
    ↪ (../openzeppelin-contracts/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (../
            ↪ openzeppelin-contracts/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Parameter NftaniaNFT2.updateRoyalty(uint256)._royaltyRate (
    ↪ NftaniaTokenWithRoyalites.sol#220) is not in mixedCase
Variable NftaniaNFT2.ADDR_TEAM_lOCKER (NftaniaTokenWithRoyalites.sol#81)
    ↪  is not in mixedCase
Variable NftaniaNFT2.ADDR_TEAM (NftaniaTokenWithRoyalites.sol#82) is not
    ↪  in mixedCase
```

```
Variable NftaniaNFT2.ADDR_INVESTORS_lOCKER (NftaniaTokenWithRoyalites.
    ↪ sol#86) is not in mixedCase
Variable NftaniaNFT2.ADDR_INVESTORS (NftaniaTokenWithRoyalites.sol#87)
    ↪ is not in mixedCase
Function ERC20Permit.DOMAIN_SEPARATOR() (../openzeppelin-contracts/
    ↪ contracts/token/ERC20/extensions/draft-ERC20Permit.sol#72-74) is
    ↪ not in mixedCase
Variable ERC20Permit._PERMIT_TYPEHASH (../openzeppelin-contracts/
    ↪ contracts/token/ERC20/extensions/draft-ERC20Permit.sol#27-28) is
    ↪ not in mixedCase
Function IERC20Permit.DOMAIN_SEPARATOR() (../openzeppelin-contracts/
    ↪ contracts/token/ERC20/extensions/draft-IERC20Permit.sol#58) is
    ↪ not in mixedCase
Variable EIP712._CACHED_DOMAIN_SEPARATOR (../openzeppelin-contracts/
    ↪ contracts/utils/cryptography/draft-EIP712.sol#30) is not in
    ↪ mixedCase
Variable EIP712._CACHED_CHAIN_ID (../openzeppelin-contracts/contracts/
    ↪ utils/cryptography/draft-EIP712.sol#31) is not in mixedCase
Variable EIP712._HASHED_NAME (../openzeppelin-contracts/contracts/utils/
    ↪ cryptography/draft-EIP712.sol#33) is not in mixedCase
Variable EIP712._HASHED_VERSION (../openzeppelin-contracts/contracts/
    ↪ utils/cryptography/draft-EIP712.sol#34) is not in mixedCase
Variable EIP712._TYPE_HASH (../openzeppelin-contracts/contracts/utils/
    ↪ cryptography/draft-EIP712.sol#35) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

ERC20Permit._PERMIT_TYPEHASH (../openzeppelin-contracts/contracts/token/
    ↪ ERC20/extensions/draft-ERC20Permit.sol#27-28) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #state-variables-that-could-be-declared-constant

addFreezeExempt(address) should be declared external:
```

```
        - NftaniaNFT2.addFreezeExempt(address) (NftaniaTokenWithRoyalites
            ↪ .sol#143-145)
removeFreezeExempt(address) should be declared external:
        - NftaniaNFT2.removeFreezeExempt(address) (
            ↪ NftaniaTokenWithRoyalites.sol#148-150)
checkIfFreezeExempt(address) should be declared external:
        - NftaniaNFT2.checkIfFreezeExempt(address) (
            ↪ NftaniaTokenWithRoyalites.sol#153-155)
updateTokenReleaseDate(uint256) should be declared external:
        - NftaniaNFT2.updateTokenReleaseDate(uint256) (
            ↪ NftaniaTokenWithRoyalites.sol#174-178)
earlyReleaseTokens() should be declared external:
        - NftaniaNFT2.earlyReleaseTokens() (NftaniaTokenWithRoyalites.sol
            ↪ #181-185)
isTokenReleased() should be declared external:
        - NftaniaNFT2.isTokenReleased() (NftaniaTokenWithRoyalites.sol
            ↪ #188-190)
releaseTokens() should be declared external:
        - NftaniaNFT2.releaseTokens() (NftaniaTokenWithRoyalites.sol
            ↪ #194-197)
withdraw(uint256,address) should be declared external:
        - NftaniaNFT2.withdraw(uint256,address) (
            ↪ NftaniaTokenWithRoyalites.sol#205-209)
withdrawTokens(IERC20,address,uint256) should be declared external:
        - NftaniaNFT2.withdrawTokens(IERC20,address,uint256) (
            ↪ NftaniaTokenWithRoyalites.sol#212-217)
updateRoyalty(uint256) should be declared external:
        - NftaniaNFT2.updateRoyalty(uint256) (NftaniaTokenWithRoyalites.
            ↪ sol#220-223)
addFromExempt(address) should be declared external:
        - NftaniaNFT2.addFromExempt(address) (NftaniaTokenWithRoyalites.
            ↪ sol#237-240)
addToExempt(address) should be declared external:
```

```
        - NftaniaNFT2.addToExempt(address) (NftaniaTokenWithRoyalites.sol
            ↪ #243-246)
pause() should be declared external:
        - NftaniaNFT2.pause() (NftaniaTokenWithRoyalites.sol#275-278)
unpause() should be declared external:
        - NftaniaNFT2.unpause() (NftaniaTokenWithRoyalites.sol#279-282)
snapshot() should be declared external:
        - NftaniaNFT2.snapshot() (NftaniaTokenWithRoyalites.sol#300-302)
renounceOwnership() should be declared external:
        - NftaniaNFT2.renounceOwnership() (NftaniaTokenWithRoyalites.sol
            ↪ #305-307)
        - Ownable.renounceOwnership() (../openzeppelin-contracts/
            ↪ contracts/access/Ownable.sol#53-55)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (../openzeppelin-contracts/
            ↪ contracts/token/ERC20/ERC20.sol#131-134)
        - NftaniaNFT2.approve(address,uint256) (NftaniaTokenWithRoyalites
            ↪ .sol#310-312)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (../openzeppelin-contracts/
            ↪ contracts/access/Ownable.sol#61-64)
name() should be declared external:
        - ERC20.name() (../openzeppelin-contracts/contracts/token/ERC20/
            ↪ ERC20.sol#61-63)
symbol() should be declared external:
        - ERC20.symbol() (../openzeppelin-contracts/contracts/token/ERC20
            ↪ /ERC20.sol#69-71)
decimals() should be declared external:
        - ERC20.decimals() (../openzeppelin-contracts/contracts/token/
            ↪ ERC20/ERC20.sol#86-88)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (../openzeppelin-contracts/
            ↪ contracts/token/ERC20/ERC20.sol#112-115)
transferFrom(address,address,uint256) should be declared external:
```

```
        - ERC20.transferFrom(address,address,uint256) (../openzeppelin-
            ↪ contracts/contracts/token/ERC20/ERC20.sol#149-163)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (../openzeppelin-
            ↪ contracts/contracts/token/ERC20/ERC20.sol#196-204)
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (../openzeppelin-contracts/
            ↪ contracts/token/ERC20/extensions/ERC20Burnable.sol#19-21)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (../openzeppelin-
            ↪ contracts/contracts/token/ERC20/extensions/ERC20Burnable.
            ↪ sol#34-41)
balanceOfAt(address,uint256) should be declared external:
        - ERC20Snapshot.balanceOfAt(address,uint256) (../openzeppelin-
            ↪ contracts/contracts/token/ERC20/extensions/ERC20Snapshot.
            ↪ sol#105-109)
totalSupplyAt(uint256) should be declared external:
        - ERC20Snapshot.totalSupplyAt(uint256) (../openzeppelin-contracts
            ↪ /contracts/token/ERC20/extensions/ERC20Snapshot.sol
            ↪ #114-118)
permit(address,address,uint256,uint256,uint8,bytes32,bytes32) should be
    ↪ declared external:
        - ERC20Permit.permit(address,address,uint256,uint256,uint8,
            ↪ bytes32,bytes32) (../openzeppelin-contracts/contracts/
            ↪ token/ERC20/extensions/draft-ERC20Permit.sol#40-59)
nonces(address) should be declared external:
        - ERC20Permit.nonces(address) (../openzeppelin-contracts/
            ↪ contracts/token/ERC20/extensions/draft-ERC20Permit.sol
            ↪ #64-66)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external

Lquiditylocker.constructor(string,uint256,address)._beneficiaryWallet (
    ↪ liquidityLocker.sol#76) lacks a zero-check on :
```

```
            - beneficiaryWallet = _beneficiaryWallet (liquidityLocker.
                ↪ sol#78)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

Lquiditylocker.getInfo() (liquidityLocker.sol#91-97) uses timestamp for
    ↪ comparisons
        Dangerous comparisons:
        - unlockTime > block.timestamp (liquidityLocker.sol#95)
Lquiditylocker.unlockTokens() (liquidityLocker.sol#99-105) uses
    ↪ timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= unlockTime,liquidity
            ↪ Lock: Current time is before unlock time) (liquidityLocker
            ↪ .sol#100)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #block-timestamp

Address.isContract(address) (../openzeppelin-contracts/contracts/utils/
    ↪ Address.sol#26-36) uses assembly
        - INLINE ASM (../openzeppelin-contracts/contracts/utils/Address.
            ↪ sol#32-34)
Address.verifyCallResult(bool,bytes,string) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (../openzeppelin-contracts/contracts/utils/Address.
            ↪ sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

Address.functionCall(address,bytes) (../openzeppelin-contracts/contracts
    ↪ /utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#108-114) is never used and
    ↪ should be removed
```

```
Address.functionDelegateCall(address,bytes) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#168-170) is never used and should be
    ↪ removed
Address.functionDelegateCall(address,bytes,string) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#178-187) is never used and
    ↪ should be removed
Address.functionStaticCall(address,bytes) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#141-143) is never used and should be
    ↪ removed
Address.functionStaticCall(address,bytes,string) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#151-160) is never used and
    ↪ should be removed
Address.sendValue(address,uint256) (../openzeppelin-contracts/contracts/
    ↪ utils/Address.sol#54-59) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (../openzeppelin-contracts
    ↪ /contracts/token/ERC20/utils/SafeERC20.sol#44-57) is never used
    ↪ and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (../openzeppelin
    ↪ -contracts/contracts/token/ERC20/utils/SafeERC20.sol#68-79) is
    ↪ never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (../openzeppelin
    ↪ -contracts/contracts/token/ERC20/utils/SafeERC20.sol#59-66) is
    ↪ never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (../
    ↪ openzeppelin-contracts/contracts/token/ERC20/utils/SafeERC20.sol
    ↪ #28-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version^0.8.0 (liquidityLocker.sol#62) necessitates a version too
    ↪  recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
    ↪ IERC20.sol#3) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
```

```
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
    ↪ utils/SafeERC20.sol#3) necessitates a version too recent to be
    ↪ trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/utils/Address.
    ↪ sol#3) necessitates a version too recent to be trusted. Consider
    ↪ deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (../openzeppelin-
            ↪ contracts/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (../openzeppelin-contracts/contracts/utils/Address.sol
    ↪ #122-133):
        - (success,returndata) = target.call{value: value}(data) (../
            ↪ openzeppelin-contracts/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (../
    ↪ openzeppelin-contracts/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (../openzeppelin
            ↪ -contracts/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string)
    ↪ (../openzeppelin-contracts/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (../
            ↪ openzeppelin-contracts/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Parameter Lquiditylocker.setToken(IERC20)._token (liquidityLocker.sol
    ↪ #86) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions
```

```
Variable Lquiditylocker.getInfo()._benefeciaryWallet (liquidityLocker.
    ↪ sol#92) is too similar to Lquiditylocker.constructor(string,
    ↪ uint256,address)._beneficiaryWallet (liquidityLocker.sol#76)
Variable Lquiditylocker.getInfo()._benefeciaryWallet (liquidityLocker.
    ↪ sol#92) is too similar to Lquiditylocker.unlockTokens().
    ↪ _beneficiaryWallet (liquidityLocker.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #variable-names-are-too-similar

setToken(IERC20) should be declared external:
        - Lquiditylocker.setToken(IERC20) (liquidityLocker.sol#86-89)
getInfo() should be declared external:
        - Lquiditylocker.getInfo() (liquidityLocker.sol#91-97)
unlockTokens() should be declared external:
        - Lquiditylocker.unlockTokens() (liquidityLocker.sol#99-105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external

NftaniaAirdrop.withdrawTokens(IERC20,address,uint256) (NftaniaAirdropV5
    ↪ (1).sol#260-265) ignores return value by token.transfer(to,amount
    ↪ ) (NftaniaAirdropV5 (1).sol#263)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unchecked-transfer

NftaniaAirdrop.withdraw(uint256,address).receivingWallet (
    ↪ NftaniaAirdropV5 (1).sol#253) lacks a zero-check on :
                - receivingWallet.transfer(amount) (NftaniaAirdropV5 (1).
                    ↪ sol#255)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

Reentrancy in NftaniaAirdrop.getAirdrop(bytes32,uint8,bytes32,bytes32,
    ↪ uint256,address,uint256) (NftaniaAirdropV5 (1).sol#148-177):
```

```
        External calls:
        - deliverTokens(to,amount) (NftaniaAirdropV5 (1).sol#171)
              - returndata = address(token).functionCall(data,SafeERC20:
                 ↪  low-level call failed) (../openzeppelin-contracts/
                 ↪ contracts/token/ERC20/utils/SafeERC20.sol#92)
              - IERC20(tokenAddress).safeTransferFrom(owner(),to,
                 ↪ amountWithDecimal) (NftaniaAirdropV5 (1).sol#182)
              - (success,returndata) = target.call{value: value}(data)
                 ↪ (../openzeppelin-contracts/contracts/utils/Address.
                 ↪ sol#131)
        External calls sending eth:
        - deliverTokens(to,amount) (NftaniaAirdropV5 (1).sol#171)
              - (success,returndata) = target.call{value: value}(data)
                 ↪ (../openzeppelin-contracts/contracts/utils/Address.
                 ↪ sol#131)
        Event emitted after the call(s):
        - AirdropClaim(to,amount,airdropID,block.timestamp) (
           ↪ NftaniaAirdropV5 (1).sol#174)
        - NewAirdropStatus(totalAirdropedTokens,remainingTokens / 10 **
           ↪ 18,block.timestamp) (NftaniaAirdropV5 (1).sol#175)
Reentrancy in NftaniaAirdrop.withdrawTokens(IERC20,address,uint256) (
   ↪ NftaniaAirdropV5 (1).sol#260-265):
        External calls:
        - token.transfer(to,amount) (NftaniaAirdropV5 (1).sol#263)
        Event emitted after the call(s):
        - WithdrawTokens(to,amount) (NftaniaAirdropV5 (1).sol#264)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #reentrancy-vulnerabilities-3

NftaniaAirdrop.constructor(uint256,uint256,address,uint256) (
   ↪ NftaniaAirdropV5 (1).sol#127-136) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_startDate > block.timestamp,Nftania
           ↪ Airdrop: Start date is before current time) (
```

```
            ↪ NftaniaAirdropV5 (1).sol#130)
NftaniaAirdrop.getAirdrop(bytes32,uint8,bytes32,bytes32,uint256,address,
    ↪ uint256) (NftaniaAirdropV5 (1).sol#148-177) uses timestamp for
    ↪ comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= startDate,Airdrop has
            ↪ not started yet) (NftaniaAirdropV5 (1).sol#154)
        - require(bool,string)(block.timestamp <= endDate,Airdrop has
            ↪ already finished) (NftaniaAirdropV5 (1).sol#155)
NftaniaAirdrop.setDates(uint256,uint256) (NftaniaAirdropV5 (1).sol
    ↪ #242-250) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_startDate > block.timestamp,Nftania
            ↪ Airdrop: Start date is before current time) (
            ↪ NftaniaAirdropV5 (1).sol#244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #block-timestamp

Address.isContract(address) (../openzeppelin-contracts/contracts/utils/
    ↪ Address.sol#26-36) uses assembly
        - INLINE ASM (../openzeppelin-contracts/contracts/utils/Address.
            ↪ sol#32-34)
Address.verifyCallResult(bool,bytes,string) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (../openzeppelin-contracts/contracts/utils/Address.
            ↪ sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

NftaniaAirdrop.getAirdrop(bytes32,uint8,bytes32,bytes32,uint256,address,
    ↪ uint256) (NftaniaAirdropV5 (1).sol#148-177) compares to a boolean
    ↪  constant:
        -require(bool,string)(nonce == true,Nftania Airdrop: This airdrop
            ↪  is already minted) (NftaniaAirdropV5 (1).sol#163)
```

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

Different versions of Solidity is used:
        - Version used: ['0.8.0', '^0.8.0']
        - 0.8.0 (NftaniaAirdropV5 (1).sol#90)
        - ^0.8.0 (../openzeppelin-contracts/contracts/access/Ownable.sol
            ↪ #3)
        - ^0.8.0 (../openzeppelin-contracts/contracts/security/Pausable.
            ↪ sol#3)
        - ^0.8.0 (../openzeppelin-contracts/contracts/security/
            ↪ ReentrancyGuard.sol#3)
        - ^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/IERC20.
            ↪ sol#3)
        - ^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/utils/
            ↪ SafeERC20.sol#3)
        - ^0.8.0 (../openzeppelin-contracts/contracts/utils/Address.sol
            ↪ #3)
        - ^0.8.0 (../openzeppelin-contracts/contracts/utils/Context.sol
            ↪ #3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #different-pragma-directives-are-used

Address.functionCall(address,bytes) (../openzeppelin-contracts/contracts
    ↪ /utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#108-114) is never used and
    ↪ should be removed
Address.functionDelegateCall(address,bytes) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#168-170) is never used and should be
    ↪ removed
Address.functionDelegateCall(address,bytes,string) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#178-187) is never used and
    ↪ should be removed
```

```
Address.functionStaticCall(address,bytes) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#141-143) is never used and should be
    ↪ removed
Address.functionStaticCall(address,bytes,string) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#151-160) is never used and
    ↪ should be removed
Address.sendValue(address,uint256) (../openzeppelin-contracts/contracts/
    ↪ utils/Address.sol#54-59) is never used and should be removed
Context._msgData() (../openzeppelin-contracts/contracts/utils/Context.
    ↪ sol#20-22) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (../openzeppelin-contracts
    ↪ /contracts/token/ERC20/utils/SafeERC20.sol#44-57) is never used
    ↪ and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (../openzeppelin
    ↪ -contracts/contracts/token/ERC20/utils/SafeERC20.sol#68-79) is
    ↪ never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (../openzeppelin
    ↪ -contracts/contracts/token/ERC20/utils/SafeERC20.sol#59-66) is
    ↪ never used and should be removed
SafeERC20.safeTransfer(IERC20,address,uint256) (../openzeppelin-
    ↪ contracts/contracts/token/ERC20/utils/SafeERC20.sol#20-26) is
    ↪ never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version0.8.0 (NftaniaAirdropV5 (1).sol#90) necessitates a version
    ↪  too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/access/Ownable
    ↪ .sol#3) necessitates a version too recent to be trusted. Consider
    ↪  deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/security/
    ↪ Pausable.sol#3) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
```

```
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/security/
    ↪ ReentrancyGuard.sol#3) necessitates a version too recent to be
    ↪ trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
    ↪ IERC20.sol#3) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
    ↪ utils/SafeERC20.sol#3) necessitates a version too recent to be
    ↪ trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/utils/Address.
    ↪ sol#3) necessitates a version too recent to be trusted. Consider
    ↪ deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/utils/Context.
    ↪ sol#3) necessitates a version too recent to be trusted. Consider
    ↪ deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (../openzeppelin-
            ↪ contracts/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (../openzeppelin-contracts/contracts/utils/Address.sol
    ↪ #122-133):
        - (success,returndata) = target.call{value: value}(data) (../
            ↪ openzeppelin-contracts/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (../
    ↪ openzeppelin-contracts/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (../openzeppelin
            ↪ -contracts/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string)
    ↪ (../openzeppelin-contracts/contracts/utils/Address.sol#178-187):
```

```
        - (success,returndata) = target.delegatecall(data) (../
            ↪ openzeppelin-contracts/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Event NftaniaAirdropairdropDateUpdate(uint256,uint256) (NftaniaAirdropV5
    ↪  (1).sol#122) is not in CapWords
Parameter NftaniaAirdrop.setTokenAddress(address)._tokenAddress (
    ↪ NftaniaAirdropV5 (1).sol#142) is not in mixedCase
Parameter NftaniaAirdrop.setDates(uint256,uint256)._startDate (
    ↪ NftaniaAirdropV5 (1).sol#242) is not in mixedCase
Parameter NftaniaAirdrop.setDates(uint256,uint256)._endDate (
    ↪ NftaniaAirdropV5 (1).sol#242) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

getAirdrop(bytes32,uint8,bytes32,bytes32,uint256,address,uint256) should
    ↪  be declared external:
        - NftaniaAirdrop.getAirdrop(bytes32,uint8,bytes32,bytes32,uint256
            ↪ ,address,uint256) (NftaniaAirdropV5 (1).sol#148-177)
getDetails() should be declared external:
        - NftaniaAirdrop.getDetails() (NftaniaAirdropV5 (1).sol#195-197)
setDates(uint256,uint256) should be declared external:
        - NftaniaAirdrop.setDates(uint256,uint256) (NftaniaAirdropV5 (1).
            ↪ sol#242-250)
withdraw(uint256,address) should be declared external:
        - NftaniaAirdrop.withdraw(uint256,address) (NftaniaAirdropV5 (1).
            ↪ sol#253-257)
withdrawTokens(IERC20,address,uint256) should be declared external:
        - NftaniaAirdrop.withdrawTokens(IERC20,address,uint256) (
            ↪ NftaniaAirdropV5 (1).sol#260-265)
pause() should be declared external:
        - NftaniaAirdrop.pause() (NftaniaAirdropV5 (1).sol#279-282)
unpause() should be declared external:
```

```
        - NftaniaAirdrop.unpause() (NftaniaAirdropV5 (1).sol#284-287)
renounceOwnership() should be declared external:
        - NftaniaAirdrop.renounceOwnership() (NftaniaAirdropV5 (1).sol
            ↪ #290-292)
        - Ownable.renounceOwnership() (../openzeppelin-contracts/
            ↪ contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (../openzeppelin-contracts/
            ↪ contracts/access/Ownable.sol#61-64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external

VestingLocker.constructor(uint256,address,string)._beneficiaryWallet (
    ↪ vestinglocker.sol#79) lacks a zero-check on :
                - beneficiaryWallet = _beneficiaryWallet (vestinglocker.
                    ↪ sol#81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

VestingLocker.getInstallmentInfo(uint256) (vestinglocker.sol#120-132)
    ↪ uses timestamp for comparisons
        Dangerous comparisons:
        - instalmentTime > block.timestamp (vestinglocker.sol#125)
VestingLocker.releaseInstalment(uint256) (vestinglocker.sol#134-141)
    ↪ uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= instalmentsTime[
            ↪ instalmentId],TokenTimelock: current time is before
            ↪ instalmemt time) (vestinglocker.sol#135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #block-timestamp

Address.isContract(address) (../openzeppelin-contracts/contracts/utils/
    ↪ Address.sol#26-36) uses assembly
```

```
        - INLINE ASM (../openzeppelin-contracts/contracts/utils/Address.
            ↪ sol#32-34)
Address.verifyCallResult(bool,bytes,string) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (../openzeppelin-contracts/contracts/utils/Address.
            ↪ sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

VestingLocker.releaseInstalment(uint256) (vestinglocker.sol#134-141)
    ↪ compares to a boolean constant:
        -require(bool,string)(instalmentsReleased[instalmentId] == false,
            ↪ TokenTimelock: this instalment is already released) (
            ↪ vestinglocker.sol#136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #boolean-equality

Address.functionCall(address,bytes) (../openzeppelin-contracts/contracts
    ↪ /utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#108-114) is never used and
    ↪ should be removed
Address.functionDelegateCall(address,bytes) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#168-170) is never used and should be
    ↪ removed
Address.functionDelegateCall(address,bytes,string) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#178-187) is never used and
    ↪ should be removed
Address.functionStaticCall(address,bytes) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#141-143) is never used and should be
    ↪ removed
Address.functionStaticCall(address,bytes,string) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#151-160) is never used and
    ↪ should be removed
```

```
Address.sendValue(address,uint256) (../openzeppelin-contracts/contracts/
    ↪ utils/Address.sol#54-59) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (../openzeppelin-contracts
    ↪ /contracts/token/ERC20/utils/SafeERC20.sol#44-57) is never used
    ↪ and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (../openzeppelin
    ↪ -contracts/contracts/token/ERC20/utils/SafeERC20.sol#68-79) is
    ↪ never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (../openzeppelin
    ↪ -contracts/contracts/token/ERC20/utils/SafeERC20.sol#59-66) is
    ↪ never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (../
    ↪ openzeppelin-contracts/contracts/token/ERC20/utils/SafeERC20.sol
    ↪ #28-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version^0.8.0 (vestinglocker.sol#64) necessitates a version too
    ↪ recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
    ↪ IERC20.sol#3) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
    ↪ utils/SafeERC20.sol#3) necessitates a version too recent to be
    ↪ trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/utils/Address.
    ↪ sol#3) necessitates a version too recent to be trusted. Consider
    ↪ deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#54-59):
```

```
        - (success) = recipient.call{value: amount}() (../openzeppelin-
            ↪ contracts/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (../openzeppelin-contracts/contracts/utils/Address.sol
    ↪ #122-133):
        - (success,returndata) = target.call{value: value}(data) (../
            ↪ openzeppelin-contracts/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (../
    ↪ openzeppelin-contracts/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (../openzeppelin
            ↪ -contracts/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string)
    ↪ (../openzeppelin-contracts/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (../
            ↪ openzeppelin-contracts/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Parameter VestingLocker.setToken(IERC20)._token (vestinglocker.sol#90)
    ↪ is not in mixedCase
Parameter VestingLocker.setInstalmentsAmounts(uint256)._amount (
    ↪ vestinglocker.sol#106) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

Variable VestingLocker.getLockInfo()._benefeciaryWallet (vestinglocker.
    ↪ sol#115) is too similar to VestingLocker.constructor(uint256,
    ↪ address,string)._beneficiaryWallet (vestinglocker.sol#79)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #variable-names-are-too-similar

setToken(IERC20) should be declared external:
        - VestingLocker.setToken(IERC20) (vestinglocker.sol#90-93)
getLockInfo() should be declared external:
```

```
        - VestingLocker.getLockInfo() (vestinglocker.sol#115-118)
getInstallmentInfo(uint256) should be declared external:
        - VestingLocker.getInstallmentInfo(uint256) (vestinglocker.sol
            ↪ #120-132)
releaseInstalment(uint256) should be declared external:
        - VestingLocker.releaseInstalment(uint256) (vestinglocker.sol
            ↪ #134-141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external

NftaniaAddLiquidity.addLiquidityETH(address,uint256,uint256,address) (
    ↪ NftaniaAddliquidity.sol#22-43) ignores return value by IERC20(
    ↪ token).transferFrom(msg.sender,address(this),tokenAmount) (
    ↪ NftaniaAddliquidity.sol#26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unchecked-transfer

NftaniaAddLiquidity.addLiquidityETH(address,uint256,uint256,address) (
    ↪ NftaniaAddliquidity.sol#22-43) ignores return value by IERC20(
    ↪ token).approve(ROUTER,tokenAmount) (NftaniaAddliquidity.sol#27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

NftaniaAddLiquidity.addLiquidityETH(address,uint256,uint256,address).
    ↪ _token (NftaniaAddliquidity.sol#22) lacks a zero-check on :
                - token = _token (NftaniaAddliquidity.sol#24)
NftaniaAddLiquidity.addLiquidityETH(address,uint256,uint256,address).
    ↪ _beneficiary (NftaniaAddliquidity.sol#22) lacks a zero-check on :
                - beneficiary = _beneficiary (NftaniaAddliquidity.sol#25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

Reentrancy in NftaniaAddLiquidity.addLiquidityETH(address,uint256,
    ↪ uint256,address) (NftaniaAddliquidity.sol#22-43):
```

```
        External calls:
        - IERC20(token).transferFrom(msg.sender,address(this),tokenAmount
            ↪ ) (NftaniaAddliquidity.sol#26)
        - IERC20(token).approve(ROUTER,tokenAmount) (NftaniaAddliquidity.
            ↪ sol#27)
        - (amountToken,amountETH,amountliquidity) = IUniswapV2Router(
            ↪ ROUTER).addLiquidityETH{value: EthAmount}(token,
            ↪ tokenAmount,0,0,beneficiary,block.timestamp + 120) (
            ↪ NftaniaAddliquidity.sol#29-37)
        External calls sending eth:
        - (amountToken,amountETH,amountliquidity) = IUniswapV2Router(
            ↪ ROUTER).addLiquidityETH{value: EthAmount}(token,
            ↪ tokenAmount,0,0,beneficiary,block.timestamp + 120) (
            ↪ NftaniaAddliquidity.sol#29-37)
        State variables written after the call(s):
        - pairAddress = IUniswapV2Factory(FACTORY).getPair(token,WETH) (
            ↪ NftaniaAddliquidity.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-2

Reentrancy in NftaniaAddLiquidity.addLiquidityETH(address,uint256,
    ↪ uint256,address) (NftaniaAddliquidity.sol#22-43):
        External calls:
        - IERC20(token).transferFrom(msg.sender,address(this),tokenAmount
            ↪ ) (NftaniaAddliquidity.sol#26)
        - IERC20(token).approve(ROUTER,tokenAmount) (NftaniaAddliquidity.
            ↪ sol#27)
        - (amountToken,amountETH,amountliquidity) = IUniswapV2Router(
            ↪ ROUTER).addLiquidityETH{value: EthAmount}(token,
            ↪ tokenAmount,0,0,beneficiary,block.timestamp + 120) (
            ↪ NftaniaAddliquidity.sol#29-37)
        External calls sending eth:
        - (amountToken,amountETH,amountliquidity) = IUniswapV2Router(
            ↪ ROUTER).addLiquidityETH{value: EthAmount}(token,
```

```
            ↪ tokenAmount,0,0,beneficiary,block.timestamp + 120) (
            ↪ NftaniaAddliquidity.sol#29-37)
        Event emitted after the call(s):
        - LiquidityAdded(amountToken,amountETH,amountliquidity,
            ↪ totalLiquidity,pairAddress) (NftaniaAddliquidity.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

Different versions of Solidity is used:
        - Version used: ['^0.8', '^0.8.0']
        - ^0.8 (NftaniaAddliquidity.sol#4)
        - ^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/IERC20.
            ↪ sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #different-pragma-directives-are-used

Pragma version^0.8 (NftaniaAddliquidity.sol#4) is too complex
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
    ↪ IERC20.sol#3) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Parameter NftaniaAddLiquidity.addLiquidityETH(address,uint256,uint256,
    ↪ address)._token (NftaniaAddliquidity.sol#22) is not in mixedCase
Parameter NftaniaAddLiquidity.addLiquidityETH(address,uint256,uint256,
    ↪ address).EthAmount (NftaniaAddliquidity.sol#22) is not in
    ↪ mixedCase
Parameter NftaniaAddLiquidity.addLiquidityETH(address,uint256,uint256,
    ↪ address)._beneficiary (NftaniaAddliquidity.sol#22) is not in
    ↪ mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions
```

```
NftaniaCrowdsale.callNftaniaAddLiquidity(uint256,uint256) (
    ↪ crowdsaleWithLiquidity.sol#350-357) sends eth to arbitrary user
        Dangerous calls:
        - (amountToken,amountETH,amountliquidity,totalLiquidity,
            ↪ pairAddress) = INftaniaAddLiquidity(NftaniaAddLiquidity).
            ↪ addLiquidityETH{value: EthAmount}(tokenAddress,tokenAmount
            ↪ ,EthAmount,LiquidityPoolLocker) (crowdsaleWithLiquidity.
            ↪ sol#353-354)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #functions-that-send-ether-to-arbitrary-destinations

NftaniaCrowdsale.withdrawTokens(IERC20,address,uint256) (
    ↪ crowdsaleWithLiquidity.sol#239-244) ignores return value by token
    ↪ .transfer(to,amount) (crowdsaleWithLiquidity.sol#242)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unchecked-transfer

NftaniaCrowdsale.callNftaniaAddLiquidity(uint256,uint256) (
    ↪ crowdsaleWithLiquidity.sol#350-357) ignores return value by
    ↪ IERC20(tokenAddress).approve(NftaniaAddLiquidity,tokenAmount) (
    ↪ crowdsaleWithLiquidity.sol#352)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

NftaniaCrowdsale.withdraw(uint256,address).receivingWallet (
    ↪ crowdsaleWithLiquidity.sol#232) lacks a zero-check on :
                - receivingWallet.transfer(amount) (crowdsaleWithLiquidity
                    ↪ .sol#234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

Reentrancy in NftaniaCrowdsale.buyTokens(uint256) (
    ↪ crowdsaleWithLiquidity.sol#279-310):
```

```
External calls:
- deliverTokens(msg.sender,amount) (crowdsaleWithLiquidity.sol
    ↪ #302)
        - IERC20(tokenAddress).safeTransferFrom(owner(),to,amount
            ↪ * 10 ** 18) (crowdsaleWithLiquidity.sol#314)
        - returndata = address(token).functionCall(data,SafeERC20:
            ↪  low-level call failed) (../openzeppelin-contracts/
            ↪ contracts/token/ERC20/utils/SafeERC20.sol#92)
        - (success,returndata) = target.call{value: value}(data)
            ↪ (../openzeppelin-contracts/contracts/utils/Address.
            ↪ sol#131)
- autoAddLiquidity() (crowdsaleWithLiquidity.sol#304)
        - IERC20(tokenAddress).approve(NftaniaAddLiquidity,
            ↪ tokenAmount) (crowdsaleWithLiquidity.sol#352)
        - (amountToken,amountETH,amountliquidity,totalLiquidity,
            ↪ pairAddress) = INftaniaAddLiquidity(
            ↪ NftaniaAddLiquidity).addLiquidityETH{value:
            ↪ EthAmount}(tokenAddress,tokenAmount,EthAmount,
            ↪ LiquidityPoolLocker) (crowdsaleWithLiquidity.sol
            ↪ #353-354)
External calls sending eth:
- deliverTokens(msg.sender,amount) (crowdsaleWithLiquidity.sol
    ↪ #302)
        - (success,returndata) = target.call{value: value}(data)
            ↪ (../openzeppelin-contracts/contracts/utils/Address.
            ↪ sol#131)
- forwardFunds(revenueShare) (crowdsaleWithLiquidity.sol#303)
        - revenueWallet.transfer(_revenueShare) (
            ↪ crowdsaleWithLiquidity.sol#319)
- autoAddLiquidity() (crowdsaleWithLiquidity.sol#304)
        - (amountToken,amountETH,amountliquidity,totalLiquidity,
            ↪ pairAddress) = INftaniaAddLiquidity(
            ↪ NftaniaAddLiquidity).addLiquidityETH{value:
            ↪ EthAmount}(tokenAddress,tokenAmount,EthAmount,
```

```
                ↪ LiquidityPoolLocker) (crowdsaleWithLiquidity.sol
                ↪ #353-354)
        Event emitted after the call(s):
        - LiquidityAdded(amountToken,amountETH,amountliquidity,
            ↪ totalLiquidity,pairAddress) (crowdsaleWithLiquidity.sol
            ↪ #355)
                - autoAddLiquidity() (crowdsaleWithLiquidity.sol#304)
        - PurchaseOrder(msg.sender,amount,poolShare,revenueShare,
            ↪ walletPurchases[msg.sender],walletBalance,purchaseOrderID,
            ↪ block.timestamp) (crowdsaleWithLiquidity.sol#307)
        - StatusUpdate(weiRaised,poolRaised,revenueRaised,totalTokensSold
            ↪ ,remainingTokens / 10 ** 18) (crowdsaleWithLiquidity.sol
            ↪ #308)
Reentrancy in NftaniaCrowdsale.callNftaniaAddLiquidity(uint256,uint256)
    ↪ (crowdsaleWithLiquidity.sol#350-357):
        External calls:
        - IERC20(tokenAddress).approve(NftaniaAddLiquidity,tokenAmount) (
            ↪ crowdsaleWithLiquidity.sol#352)
        - (amountToken,amountETH,amountliquidity,totalLiquidity,
            ↪ pairAddress) = INftaniaAddLiquidity(NftaniaAddLiquidity).
            ↪ addLiquidityETH{value: EthAmount}(tokenAddress,tokenAmount
            ↪ ,EthAmount,LiquidityPoolLocker) (crowdsaleWithLiquidity.
            ↪ sol#353-354)
        External calls sending eth:
        - (amountToken,amountETH,amountliquidity,totalLiquidity,
            ↪ pairAddress) = INftaniaAddLiquidity(NftaniaAddLiquidity).
            ↪ addLiquidityETH{value: EthAmount}(tokenAddress,tokenAmount
            ↪ ,EthAmount,LiquidityPoolLocker) (crowdsaleWithLiquidity.
            ↪ sol#353-354)
        Event emitted after the call(s):
        - LiquidityAdded(amountToken,amountETH,amountliquidity,
            ↪ totalLiquidity,pairAddress) (crowdsaleWithLiquidity.sol
            ↪ #355)
```

```
Reentrancy in NftaniaCrowdsale.withdrawTokens(IERC20,address,uint256) (
    ↪ crowdsaleWithLiquidity.sol#239-244):
        External calls:
        - token.transfer(to,amount) (crowdsaleWithLiquidity.sol#242)
        Event emitted after the call(s):
        - WithdrawTokens(to,amount) (crowdsaleWithLiquidity.sol#243)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

NftaniaCrowdsale.constructor(uint256,uint256,uint256,uint256,uint256,
    ↪ uint256,uint256,uint256,uint256) (crowdsaleWithLiquidity.sol
    ↪ #132-163) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_startDate >= block.timestamp,
            ↪ NftaniaCrowdsale: opening time is before current time) (
            ↪ crowdsaleWithLiquidity.sol#150)
NftaniaCrowdsale.updateDates(uint256,uint256) (crowdsaleWithLiquidity.
    ↪ sol#269-276) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_startDate >= block.timestamp,
            ↪ NftaniaCrowdsale: opening time is before current time) (
            ↪ crowdsaleWithLiquidity.sol#270)
NftaniaCrowdsale.buyTokens(uint256) (crowdsaleWithLiquidity.sol#279-310)
    ↪  uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= startDate,Nftania
            ↪ Crowdsale: Sale has not start yet) (crowdsaleWithLiquidity
            ↪ .sol#283)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #block-timestamp

Address.isContract(address) (../openzeppelin-contracts/contracts/utils/
    ↪ Address.sol#26-36) uses assembly
```

```
        - INLINE ASM (../openzeppelin-contracts/contracts/utils/Address.
          ↪ sol#32-34)
Address.verifyCallResult(bool,bytes,string) (../openzeppelin-contracts/
   ↪ contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (../openzeppelin-contracts/contracts/utils/Address.
          ↪ sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.0', '^0.8.0']
        - 0.8.0 (crowdsaleWithLiquidity.sol#72)
        - ^0.8.0 (../openzeppelin-contracts/contracts/access/Ownable.sol
          ↪ #3)
        - ^0.8.0 (../openzeppelin-contracts/contracts/security/Pausable.
          ↪ sol#3)
        - ^0.8.0 (../openzeppelin-contracts/contracts/security/
          ↪ ReentrancyGuard.sol#3)
        - ^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/IERC20.
          ↪ sol#3)
        - ^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/utils/
          ↪ SafeERC20.sol#3)
        - ^0.8.0 (../openzeppelin-contracts/contracts/utils/Address.sol
          ↪ #3)
        - ^0.8.0 (../openzeppelin-contracts/contracts/utils/Context.sol
          ↪ #3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #different-pragma-directives-are-used

Address.functionCall(address,bytes) (../openzeppelin-contracts/contracts
   ↪ /utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (../openzeppelin-
   ↪ contracts/contracts/utils/Address.sol#108-114) is never used and
   ↪ should be removed
```

```
Address.functionDelegateCall(address,bytes) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#168-170) is never used and should be
    ↪ removed
Address.functionDelegateCall(address,bytes,string) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#178-187) is never used and
    ↪ should be removed
Address.functionStaticCall(address,bytes) (../openzeppelin-contracts/
    ↪ contracts/utils/Address.sol#141-143) is never used and should be
    ↪ removed
Address.functionStaticCall(address,bytes,string) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#151-160) is never used and
    ↪ should be removed
Address.sendValue(address,uint256) (../openzeppelin-contracts/contracts/
    ↪ utils/Address.sol#54-59) is never used and should be removed
Context._msgData() (../openzeppelin-contracts/contracts/utils/Context.
    ↪ sol#20-22) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (../openzeppelin-contracts
    ↪ /contracts/token/ERC20/utils/SafeERC20.sol#44-57) is never used
    ↪ and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (../openzeppelin
    ↪ -contracts/contracts/token/ERC20/utils/SafeERC20.sol#68-79) is
    ↪ never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (../openzeppelin
    ↪ -contracts/contracts/token/ERC20/utils/SafeERC20.sol#59-66) is
    ↪ never used and should be removed
SafeERC20.safeTransfer(IERC20,address,uint256) (../openzeppelin-
    ↪ contracts/contracts/token/ERC20/utils/SafeERC20.sol#20-26) is
    ↪ never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version0.8.0 (crowdsaleWithLiquidity.sol#72) necessitates a
    ↪ version too recent to be trusted. Consider deploying with
    ↪ 0.6.12/0.7.6
```

```
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/access/Ownable
    ↪ .sol#3) necessitates a version too recent to be trusted. Consider
    ↪  deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/security/
    ↪ Pausable.sol#3) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/security/
    ↪ ReentrancyGuard.sol#3) necessitates a version too recent to be
    ↪ trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
    ↪ IERC20.sol#3) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/token/ERC20/
    ↪ utils/SafeERC20.sol#3) necessitates a version too recent to be
    ↪ trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/utils/Address.
    ↪ sol#3) necessitates a version too recent to be trusted. Consider
    ↪ deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../openzeppelin-contracts/contracts/utils/Context.
    ↪ sol#3) necessitates a version too recent to be trusted. Consider
    ↪ deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (../openzeppelin-
    ↪ contracts/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (../openzeppelin-
            ↪ contracts/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (../openzeppelin-contracts/contracts/utils/Address.sol
    ↪ #122-133):
        - (success,returndata) = target.call{value: value}(data) (../
            ↪ openzeppelin-contracts/contracts/utils/Address.sol#131)
```

```
Low level call in Address.functionStaticCall(address,bytes,string) (../
    ↪ openzeppelin-contracts/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (../openzeppelin
            ↪ -contracts/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string)
    ↪ (../openzeppelin-contracts/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (../
            ↪ openzeppelin-contracts/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Parameter INftaniaAddLiquidity.addLiquidityETH(address,uint256,uint256,
    ↪ address).EthAmount (crowdsaleWithLiquidity.sol#80) is not in
    ↪ mixedCase
Parameter NftaniaCrowdsale.intilizeAddresses(address,address,address,
    ↪ address)._tokenAddress (crowdsaleWithLiquidity.sol#167) is not in
    ↪  mixedCase
Parameter NftaniaCrowdsale.intilizeAddresses(address,address,address,
    ↪ address)._LiquidityPoolLocker (crowdsaleWithLiquidity.sol#168) is
    ↪  not in mixedCase
Parameter NftaniaCrowdsale.intilizeAddresses(address,address,address,
    ↪ address)._NftaniaAddLiquidity (crowdsaleWithLiquidity.sol#169) is
    ↪  not in mixedCase
Parameter NftaniaCrowdsale.intilizeAddresses(address,address,address,
    ↪ address)._revenueWallet (crowdsaleWithLiquidity.sol#170) is not
    ↪ in mixedCase
Parameter NftaniaCrowdsale.updatePurchaseLimits(uint256,uint256,uint256)
    ↪ ._minPurchase (crowdsaleWithLiquidity.sol#247) is not in
    ↪ mixedCase
Parameter NftaniaCrowdsale.updatePurchaseLimits(uint256,uint256,uint256)
    ↪ ._maxPurchase (crowdsaleWithLiquidity.sol#247) is not in
    ↪ mixedCase
Parameter NftaniaCrowdsale.updatePurchaseLimits(uint256,uint256,uint256)
    ↪ ._walletLimit (crowdsaleWithLiquidity.sol#247) is not in
```

```
    ↪ mixedCase
Parameter NftaniaCrowdsale.updateTokenPresalePrice(uint256).
    ↪ _tokenPresalePrice (crowdsaleWithLiquidity.sol#260) is not in
    ↪ mixedCase
Parameter NftaniaCrowdsale.updateDates(uint256,uint256)._startDate (
    ↪ crowdsaleWithLiquidity.sol#269) is not in mixedCase
Parameter NftaniaCrowdsale.updateDates(uint256,uint256)._endDate (
    ↪ crowdsaleWithLiquidity.sol#269) is not in mixedCase
Parameter NftaniaCrowdsale.forwardFunds(uint256)._revenueShare (
    ↪ crowdsaleWithLiquidity.sol#318) is not in mixedCase
Parameter NftaniaCrowdsale.callNftaniaAddLiquidity(uint256,uint256).
    ↪ EthAmount (crowdsaleWithLiquidity.sol#350) is not in mixedCase
Variable NftaniaCrowdsale.NftaniaAddLiquidity (crowdsaleWithLiquidity.
    ↪ sol#103) is not in mixedCase
Variable NftaniaCrowdsale.LiquidityPoolLocker (crowdsaleWithLiquidity.
    ↪ sol#105) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

Reentrancy in NftaniaCrowdsale.buyTokens(uint256) (
    ↪ crowdsaleWithLiquidity.sol#279-310):
        External calls:
        - forwardFunds(revenueShare) (crowdsaleWithLiquidity.sol#303)
              - revenueWallet.transfer(_revenueShare) (
                  ↪ crowdsaleWithLiquidity.sol#319)
        External calls sending eth:
        - deliverTokens(msg.sender,amount) (crowdsaleWithLiquidity.sol
            ↪ #302)
              - (success,returndata) = target.call{value: value}(data)
                  ↪ (../openzeppelin-contracts/contracts/utils/Address.
                  ↪ sol#131)
        - forwardFunds(revenueShare) (crowdsaleWithLiquidity.sol#303)
              - revenueWallet.transfer(_revenueShare) (
                  ↪ crowdsaleWithLiquidity.sol#319)
```

```
        - autoAddLiquidity() (crowdsaleWithLiquidity.sol#304)
                - (amountToken,amountETH,amountliquidity,totalLiquidity,
                   ↪ pairAddress) = INftaniaAddLiquidity(
                   ↪ NftaniaAddLiquidity).addLiquidityETH{value:
                   ↪ EthAmount}(tokenAddress,tokenAmount,EthAmount,
                   ↪ LiquidityPoolLocker) (crowdsaleWithLiquidity.sol
                   ↪ #353-354)
        Event emitted after the call(s):
        - LiquidityAdded(amountToken,amountETH,amountliquidity,
           ↪ totalLiquidity,pairAddress) (crowdsaleWithLiquidity.sol
           ↪ #355)
                - autoAddLiquidity() (crowdsaleWithLiquidity.sol#304)
        - PurchaseOrder(msg.sender,amount,poolShare,revenueShare,
           ↪ walletPurchases[msg.sender],walletBalance,purchaseOrderID,
           ↪ block.timestamp) (crowdsaleWithLiquidity.sol#307)
        - StatusUpdate(weiRaised,poolRaised,revenueRaised,totalTokensSold
           ↪ ,remainingTokens / 10 ** 18) (crowdsaleWithLiquidity.sol
           ↪ #308)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-4

intilizeAddresses(address,address,address,address) should be declared
    ↪ external:
        - NftaniaCrowdsale.intilizeAddresses(address,address,address,
           ↪ address) (crowdsaleWithLiquidity.sol#166-180)
getRemainingTokens() should be declared external:
        - NftaniaCrowdsale.getRemainingTokens() (crowdsaleWithLiquidity.
           ↪ sol#187-190)
getDetails() should be declared external:
        - NftaniaCrowdsale.getDetails() (crowdsaleWithLiquidity.sol
           ↪ #212-214)
getPurchaseLimits() should be declared external:
        - NftaniaCrowdsale.getPurchaseLimits() (crowdsaleWithLiquidity.
           ↪ sol#217-219)
```

```
getTokenPresalePrice() should be declared external:
        - NftaniaCrowdsale.getTokenPresalePrice() (crowdsaleWithLiquidity
            ↪ .sol#227-229)
withdraw(uint256,address) should be declared external:
        - NftaniaCrowdsale.withdraw(uint256,address) (
            ↪ crowdsaleWithLiquidity.sol#232-236)
withdrawTokens(IERC20,address,uint256) should be declared external:
        - NftaniaCrowdsale.withdrawTokens(IERC20,address,uint256) (
            ↪ crowdsaleWithLiquidity.sol#239-244)
updatePurchaseLimits(uint256,uint256,uint256) should be declared
    ↪ external:
        - NftaniaCrowdsale.updatePurchaseLimits(uint256,uint256,uint256)
            ↪ (crowdsaleWithLiquidity.sol#247-257)
updateTokenPresalePrice(uint256) should be declared external:
        - NftaniaCrowdsale.updateTokenPresalePrice(uint256) (
            ↪ crowdsaleWithLiquidity.sol#260-266)
updateDates(uint256,uint256) should be declared external:
        - NftaniaCrowdsale.updateDates(uint256,uint256) (
            ↪ crowdsaleWithLiquidity.sol#269-276)
buyTokens(uint256) should be declared external:
        - NftaniaCrowdsale.buyTokens(uint256) (crowdsaleWithLiquidity.sol
            ↪ #279-310)
manualAddLiquidity(uint256) should be declared external:
        - NftaniaCrowdsale.manualAddLiquidity(uint256) (
            ↪ crowdsaleWithLiquidity.sol#339-342)
pause() should be declared external:
        - NftaniaCrowdsale.pause() (crowdsaleWithLiquidity.sol#371-374)
unpause() should be declared external:
        - NftaniaCrowdsale.unpause() (crowdsaleWithLiquidity.sol#376-379)
renounceOwnership() should be declared external:
        - NftaniaCrowdsale.renounceOwnership() (crowdsaleWithLiquidity.
            ↪ sol#382-384)
        - Ownable.renounceOwnership() (../openzeppelin-contracts/
            ↪ contracts/access/Ownable.sol#53-55)
```

```
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (../openzeppelin-contracts/
          ↪ contracts/access/Ownable.sol#61-64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #public-function-that-could-be-declared-external
. analyzed (48 contracts with 75 detectors), 311 result(s) found
```

Conclusion:

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

5 Conclusion

In this audit, we examined the design and implementation of NFTANIA V5 contract and discovered several issues of varying severity. NFTANIA team addressed all the issues raised in the initial report and implemented the necessary fixes.

The present code base is well-structured and ready for the mainnet.

For a Contract Audit, contact us at contact@shellboxes.com