EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Dubai NFT
Platform: Cross-Chain Network
Website: http://dubainfts.ae
Language: Solidity
Date: May 23rd, 2023

Table of contents

Email: audit@EtherAuthority.io

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by Dubai NFT to perform the Security audit of the Dubai NFT smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on May 23rd, 2023.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

- Dubai NFT Marketplaces are platforms where users can create, buy, sell and resell non-fungible tokens (NFTs). It is a cross-chain platform, such as NFT marketplaces, allowing users to choose different blockchains networks such as Binance Smart Chain, Ethereum, and Polygon to buy, sell, and trade NFTs.
- The Dubai NFT Contracts handle multiple contracts, and all contracts have different functions.
    - Artcom: It allows managing mint, burn, clearData, airdropTokens, withdrawal, ethToToken, pauseSale functionality.
    - Stacking: It allows setting APY and Deposit Amount.
    - Bridge: It allows a new owner address and update fees, and also sets a token address.
    - DubaiNFT: It allows update fees and token addresses
    - DubaiNfts: It allows to set BaseURI, set Development Fees, update new admin addresses, set token price, etc.
- DubaiNFT is a NFT smart contract which has functions like burn, mint, Inverst, withdrawal, airdropTokens, claim, deposit, etc.

Email: audit@EtherAuthority.io

Audit scope

Name	Code Review and Security Analysis Report for Dubai NFT Smart Contracts
Platform	**Cross-Chain Network / Solidity**
File 1	Artcom.sol
File 1 MD5 Hash	756944100572ECEA7601EF9A431CFC17
Updated File 1 MD5 Hash	E2FF772C6A77BE73E1B0D10E5B9FCA3D
File 1 Online code link	0x44e70bd21270f28a0084021bfec87d62206c65de
Updated File 1 Online code link	0xf5e696abd588eb1a8b8e1c9dcef3947e08f6f2ea
File 2	BridgeBSC.sol
File 2 MD5 Hash	E58AF8F50B3822B46269CCA15611E1F2
Updated File 2 MD5 Hash	75F01C8F132DA5D8AD8A483093FDF1DC
File 2 Online code link	0xceaf9827cca918181cb6514478c95d693a9ed9ca
Updated File 2 Online code link	0x0ef577e30695372974567c054157c3e9c17adc22
File 3	DubaiNfts.sol
File 3 MD5 Hash	0FE801BA14DAAD2F4E26BA45876482B5
Updated File 3 MD5 Hash	38BDF543BF380F2B8B50280A7F7E1DC8
File 3 Online code link	0x9b0db3098e9ada5d293c6785df8d0b7690ae9300
Updated File 3 Online code link	0xfaa293ab562784c7d513cd8ce8bda3b9959e7786
File 4	DubaiNfts_stacking.sol
File 4 MD5 Hash	59128AFC10AA9DA5D447337B16916525
Updated File 4 MD5 Hash	5B339ED668D3AB72767F69FF6AEC60B4
File 4 Online code link	0xa140e762070c8b0e90b478bfe73f630b6ea42b3b
Updated File 4 Online code link	0xe776b7a5043cabc74d6a5a46764d62ab53baf9a4
File 5	Stacking.sol
File 5 MD5 Hash	EC392DD2E034A14B419224BE406AAC0A

Email: audit@EtherAuthority.io

Updated File 5 MD5 Hash	5BDBB022FD2019F34DDA39916747D00F
File 5 Online code link	0xd25a8df97c0901fff05346d87b07f021ddfccc88
Updated File 5 Online code link	0xe6b67de50dae1f679e99cd8d0618432497aed4fb
File 6	dubaiNFT.sol
File 6 MD5 Hash	A6F2371A0DBAA68E90B8F3CD8C51CDDF
Updated File 6 MD5 Hash	0CC5864FE74B67F406389D2A5A334C8B
File 6 Online code link	0x51152bEE1fdcCeEfBBa4DB6F6a845a6068B9ecDd
Updated File 6 Online code link	0xa4EB873e9d10fC18d41978Fbe8Ac6D653Bd4326a
Audit Date	May 23rd, 2023
Revised Audit Date	May 31st, 2023

Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 Artcom.sol • Name: Artcom • Symbol: ARTCOM • Decimals: 18 • Total Supply: 5 billion • Airdrop: 10 ARTCOM • Rewards: 5% of Airdrop • Minimum Deposit: 0.0001 ether • Maximum Deposit: 3 ether • 1USD: 3 Token **Owner has control over following functions:** • Set the pause the sale. • Set the Start of the sale. • Change Price of the token. • Withdrawal token. • Set the Airdrop values. • mint and burn token.	**YES, This is valid.**
File 2 BridgeBSC.sol • Admin Fees: 4% **Owner has control over following functions:** • Set a new owner address. • Set a new token address. • Set a new fee value.	**YES, This is valid. Owner wallet's private key must be handled very securely. Because if that is compromised, then it will create problems.**
File 3 DubaiNfts.sol • Name: Dubai NFT Marketplace • Symbol: DubaiNfts	**YES, This is valid. Owner wallet's private key must be handled very securely.**

Email: audit@EtherAuthority.io

• Admin Fees: 4% • Floor Price: 0.00 ether • _base Token URI: http://18.212.58.134/metadata/ **Owner has control over following functions:** • Set a baseURI. • Set a start Sale. • Set a pause Sale. • Set a Contract Fees. • Set a new admin address. • Set a floor price. • Set a token price.	**Because if that is compromised, then it will create problems.**
File 4 DubaiNfts_stacking.sol • Tokens Per Second: 0.000001 • Current ID: 0 **Owner has control over following functions:** • Set a start Stacking. • Set a pause Stacking. • Set a token Per Second.	**YES, This is valid.**
File 5 dubaiNFT.sol • Fees: 1% • Divider: 10000 **Owner has control over following functions:** • Set a new fee value. • Set a new token address.	**YES, This is valid. Owner wallet's private key must be handled very securely. Because if that is compromised, then it will create problems.**
File 6 Stacking.sol • Current ID: 0	**YES, This is valid.**

Email: audit@EtherAuthority.io

• Minimum Deposit Amount: 100 ARTCOM • Maximum Deposit Amount: 1000 ARTCOM • APY: 1% **<u>Owner has control over following functions:</u>** • Set a _hasStart status. • Set a Deposit amount. • Set an APY amount.	

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **" Secured".** Also, these contracts do contain owner control, which does not make them fully decentralized.

Insecure	Poor secured	Secure	Well-secured

You are here

We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 2 high, 0 medium and 1 low and 6 very low level issues.

These all issues are fixed/acknowledged in the revised contract code.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Email: audit@EtherAuthority.io

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 6 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the Dubai NFT Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Dubai NFT Protocol.

The Dubai NFT team has provided unit test scripts, which helped to determine the integrity of the code in an automated way.

Code parts are not well commented on smart contracts.

Documentation

We were given a Dubai NFT Protocol smart contract code in the form of a testnet.bscscan.com web link. The hash of that code is mentioned above in the table.

As mentioned above, code parts are not well commented. But the logic is straightforward. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website: http://dubainfts.ae which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

Email: audit@EtherAuthority.io

AS-IS overview

Artcom.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	decimals	external	Passed	No Issue
8	pauseSale	external	access only Owner	No Issue
9	startICO	write	access only Owner	No Issue
10	getLatestPriceEth	read	Passed	No Issue
11	Inverst	write	Passed	No Issue
12	changePrice	external	access only Owner	No Issue
13	checkExitsAddress	read	Passed	No Issue
14	ethToToken	read	Passed	No Issue
15	withdrwal	write	Passed	No Issue
16	setDrop	write	access only Owner	No Issue
17	airdropTokens	write	Passed	No Issue
18	clearData	write	Passed	No Issue
19	symbol	external	Passed	No Issue
20	name	external	Passed	No Issue
21	totalSupply	external	Passed	No Issue
21	burnToken	external	Passed	No Issue
22	balanceOf	external	Passed	No Issue
23	transfer	external	Passed	No Issue
24	allowance	external	Passed	No Issue
25	approve	external	Passed	No Issue
26	transferFrom	external	Passed	No Issue
27	increaseAllowance	write	Passed	No Issue
28	decreaseAllowance	write	Passed	No Issue
29	mint	write	access only Owner	No Issue
30	burn	write	access only Owner	No Issue
31	_transfer	internal	Passed	No Issue
32	_mint	internal	Passed	No Issue
33	_burn	internal	Passed	No Issue
34	_approve	internal	Passed	No Issue
35	_burnFrom	internal	Passed	No Issue

Email: audit@EtherAuthority.io

Stacking.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	toggelStacking	write	access only Owner	No Issue
8	setDepositeAmount	write	Passed	No Issue
9	setAPY	write	access only Owner	No Issue
10	userInfo	internal	Passed	No Issue
11	deposite	write	Passed	No Issue
12	calclulateReward	read	Passed	No Issue
13	withdrawl	write	Passed	No Issue
14	claim	write	Passed	No Issue

DubaiNfts_stacking.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	startStacking	write	access only Owner	No Issue
8	pauseStacking	write	access only Owner	No Issue
9	setTokenPerSecond	write	access only Owner	No Issue
10	isStakeholder	read	Passed	No Issue
11	addStakeholder	internal	Passed	No Issue
12	removeStakeholder	internal	Passed	Removed
13	userInfo	internal	Passed	No Issue
14	deposite	write	Passed	No Issue
15	calclulateReward	read	Passed	No Issue
16	withdrawl	write	Owner can withdraw all funds	Refer to audit findings
17	claim	write	Passed	No Issue
18	getUserStakeIds	read	Passed	No Issue

Email: audit@EtherAuthority.io

DubaiNFT.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	nonReentrant	modifier	Passed	No Issue
8	changeFees	write	Passed	No Issue
9	changeToken	write	access only Owner	No Issue
10	createMarketItem	write	Passed	No Issue
11	createMarketSale	write	Passed	No Issue

DubaiNfts.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	supportsInterface	read	Passed	No Issue
3	tokenOfOwnerByIndex	read	Passed	No Issue
4	totalSupply	read	Passed	No Issue
5	tokenByIndex	read	Passed	No Issue
6	_beforeTokenTransfer	internal	Passed	No Issue
7	_addTokenToOwnerEnumeration	write	Passed	No Issue
8	_addTokenToAllTokensEnumeration	write	Passed	No Issue
9	_removeTokenFromOwnerEnumeration	write	Passed	No Issue
10	_removeTokenFromAllTokensEnumeration	write	Passed	No Issue
11	onlyWhitelisted	modifier	Passed	No Issue
12	addToWhiteList	write	access only Owner	No Issue
13	removeToWhiteList	write	access only Owner	No Issue
14	isWhitelisted	read	Passed	No Issue
15	onlyAdmin	modifier	Passed	No Issue
16	setBaseURI	write	access only Owner	No Issue
17	startSale	write	access only Owner	No Issue
18	pauseSale	write	access only Owner	No Issue
19	setContractFees	write	access only Owner	No Issue
20	updateAdmin	write	access only Owner	No Issue
21	_baseURI	internal	Passed	No Issue
22	setFloorPrice	write	access only Owner	No Issue

Email: audit@EtherAuthority.io

23	getBaseURI	read	Passed	No Issue
24	tokenURI	read	Passed	No Issue
25	setTokenPrice	write	Passed	No Issue
26	walletOfOwner	read	Passed	No Issue
27	mintPublic	write	Passed	No Issue
28	mint	write	Passed	No Issue
29	burn	write	Admin can burn anyone's token	Refer to audit findings
30	buy	write	Passed	No Issue

BridgeBSC.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	burn	external	Passed	No Issue
3	mint	external	Passed	No Issue
4	updateOwner	write	Passed	No Issue
5	updateToken	write	Passed	No Issue
6	updateFees	write	Passed	No Issue

Email: audit@EtherAuthority.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found in the contract code.

High Severity

(1) Subtraction overflow:

DubaiNFTs.sol

Function: buy()

```
function buy(uint256 tokenId) public payable{    infinite gas
    require(!isBlacklisted[msg.sender], "caller is backlisted");
    require(msg.value>=NFTINFO[tokenId].price," Low BNB Amount :(");
    uint256 owneFees=((uint256(100)).sub(adminFees)).sub(NFTINFO[tokenId].royalties);
    address payable  ownerAddress= payable(address(uint160(ownerOf(tokenId))));
    payable(owner()).transfer((msg.value.mul(adminFees)).div(100));
    ownerAddress.transfer((msg.value.mul(owneFees)).div(100));
    payable(NFTINFO[tokenId].creatorAddress).transfer((msg.value.mul(NFTINFO[tokenId].royalties)).div(100));
    _transfer(ownerAddress,msg.sender,tokenId);
    NFTINFO[tokenId].price=floorPrice;
    NFTINFO[tokenId].ownerAddress=payable(msg.sender);
    NFTINFO[tokenId].sell=false;
}
```

BridgeBSC.sol

Function: mint()

```
function mint(address to, uint256 tokenId,address _owner,address creator,uint256 rolyalties) external payable {
  require(msg.value>0,"Error:Low Transaction Fees");
  uint256 owneFees=((uint256(100)).sub(adminFees)).sub(rolyalties);
  payable(owner).transfer((msg.value.mul(adminFees)).div(100));
  payable(_owner).transfer((msg.value.mul(owneFees)).div(100));
  payable(creator).transfer((msg.value.mul(rolyalties)).div(100));
  token.mint(to, tokenId, rolyalties, creator);
  emit Transfer(
    msg.sender,
    to,
    tokenId,
    block.timestamp,
    Step.Mint
  );
}
```

Total of admin fees and royalties should be less than 100%.

Resolution: We suggest validating the royalties so that the total of admin fees and royalties should be less than 100%.

Email: audit@EtherAuthority.io

Status: This is fixed in the revised smart contract code.

(2) Logical vulnerability : **DubaiNFTs.sol**

Function: buy()

```
function buy(uint256 tokenId) public payable{   infinite gas
    require(!isBlacklisted[msg.sender], "caller is backlisted");
    require(msg.value>=NFTINFO[tokenId].price," Low BNB Amount :(");
    uint256 owneFees=((uint256(100)).sub(adminFees)).sub(NFTINFO[tokenId].royalties);
    address payable  ownerAddress= payable(address(uint160(ownerOf(tokenId))));
    payable(owner()).transfer((msg.value.mul(adminFees)).div(100));
    ownerAddress.transfer((msg.value.mul(owneFees)).div(100));
    payable(NFTINFO[tokenId].creatorAddress).transfer((msg.value.mul(NFTINFO[tokenId].royalties)).div(100));
    _transfer(ownerAddress,msg.sender,tokenId);
    NFTINFO[tokenId].price=floorPrice;
    NFTINFO[tokenId].ownerAddress=payable(msg.sender);
    NFTINFO[tokenId].sell=false;
}
```

In the buy function there is no check if that token is already sold or not, users can buy an already sold token even though it is not open for sale.

There is no check for the price of the token whether it is greater than 0.

Resolution: We suggest adding validation for price and if that token is already sold or not.

Status: This is fixed in the revised smart contract code.

Medium

No medium severity vulnerabilities were found in the contract code.

Low

(1) Admin can burn anyone's token: **DubaiNFTs.sol**

Admin can burn any users' tokens.

Resolution: We suggest changing the code so only token holders can burn their own tokens and not anyone else. Not even a contract creator.

Status: This is acknowledged in the revised smart contract code.

Email: audit@EtherAuthority.io

Very Low / Informational / Best practices:

(1) SafeMath Library: **DubaiNFTs.sol, BridgeBSC.sol, DubaiNfts_stacking.sol, Artcom.sol, Stacking.sol, DubaiNFT.sol**

SafeMath Library is used in this contract code, but the compiler version is greater than or equal to 0.8.0, Then it will be not required to use, solidity automatically handles overflow/underflow.

Resolution: Remove the SafeMath library and use normal math operators, It will improve code size, and less gas consumption.

Status: This is fixed in the revised smart contract code.

(2) Unused variables, Internal function:

DubaiNFTs.sol

There is a MAX_SUPPLY variable defined but not used anywhere.

DubaiNfts_stacking.sol

There are "minimumDepositeAmount" and "maximumDepositeAmount" variables defined but not used anywhere.

DubaiNfts_stacking.sol

The removeStakeholder function is defined but not used.

Resolution: Remove unused variables and unused functions from the code.

Status: This is fixed in the revised smart contract code.

(3) Owner can set 100% fees: **DubaiNFT.sol**

Function: changeFees()

```
function changeFees(uint256 _fee) public onlyOwner{    24849 gas
    fees=_fee;
}
```


Email: audit@EtherAuthority.io

The Owner can set fees upto 100%. This can cause the trust issue.

Resolution: We suggest adding some range for fees.

Status: This is fixed in the revised smart contract code.

(4) Initialized by default value: **DubaiNFTs.sol**

Function: constructor()

```
constructor() ERC721("Dubai NFT Marketplace", "DubaiNfts") {
    adminFees=4;
    floorPrice=0.00 ether;
}
```

In solidity the default value of an integer variable is 0. So no need to initialize by 0.

Resolution: We suggest removing this initialization code from the constructor to reduce gas.

Status: This is fixed in the revised smart contract code.

(5) Spelling mistake:

Artcom.sol

Function: Inverst() -> Inverst word

```
function Inverst() public payable{

    require(hasStart==true,"Sale is not started");
    require(block.timestamp<endDate,"ICO Completed");
    require(msg.value>=minimumDeposite,"Minimum Amount Not reached");
    require(msg.value<=maximumDeposite,"maximum Amount reached");
    uint256 numberOfTokens = ethToToken(msg.value);
    _transfer(owner(),msg.sender, numberOfTokens);
    soldToken = soldToken.add(numberOfTokens);
```

Spelling mistake in function name. Functions are: **Inverst()**

Email: audit@EtherAuthority.io

“Inverst” should be “**Invest**”.

Function: withdrwal() -> withdrwal word

```
function withdrwal() public onlyOwner{
        require(block.timestamp>endDate,"ICO Is Not completed yet");
        payable(owner()).transfer(address(this).balance);
    }
```

Spelling mistake in function name. Functions are: **withdrwal()**

“withdrwal” should be “**withdrawal**”.

Function: Inverst() -> Deposite word

```
function Inverst() public payable{

    require(hasStart==true,"Sale is not started");
    require(block.timestamp<endDate,"ICO Completed");
    require(msg.value>=minimumDeposite,"Minimum Amount Not reached");
    require(msg.value<=maximumDeposite,"maximum Amount reached");
    uint256 numberOfTokens = ethToToken(msg.value);
    _transfer(owner(),msg.sender, numberOfTokens);
    soldToken = soldToken.add(numberOfTokens);
```

Function: constructor() -> Deposite word

```
  uint256 public startDate=0;
  uint256 public minimumDeposite;
  uint256 public maximumDeposite;
  uint256 public soldToken;
  uint256 public tokenPerUsd;
  AggregatorV3Interface public priceFeedEth;
constructor()  {
  _name = "Artcom";
  _symbol = "ARTCOM";
  _decimals = 18;
  _totalSupply = 5000000000 * 10**18;
  _balances[msg.sender] = _totalSupply;

  emit Transfer(address(0), msg.sender, _totalSupply);
  priceFeedEth = AggregatorV3Interface(0x2514895c72f50D8bd4B4F9b1110F0D6bD2c97526);//test net
      tokenPerUsd = 3;
      minimumDeposite = 0.0001 ether;
      maximumDeposite = 3 ether;
}
```

Spelling mistake in variable and function name.

Email: audit@EtherAuthority.io

“deposite” word should be **“deposit”**.

DubaiNfts_stacking.sol

Function: claim() -> withdrawl word

```
function claim(uint256 id) public{    infinite gas
    require(hasStart==true,"Stacking is not Start yet");
    require(Stack[id].isWithdrawal==false,"Amount Already withdrawl");
    uint256 reward=calclulateReward(id);
    uint256 totalAmount=(Stack[id].amount)+(reward);
```

Function: calclulateReward() -> withdrawl word

```
function calclulateReward(uint256 id) public view returns(uint256){
    require(Stack[id].isWithdrawal==false,"Amount Already withdrawl");
    uint256 depositeTime=Stack[id].time;
    uint256 currentTime=block.timestamp;
```

Function: withdrawl() -> withdrawl word

```
function withdrawl(uint256 amount) public onlyOwner{    infinite gas
        require(rewardToken.balanceOf(address(this))>= amount.mul(1e18),'
        rewardToken.safeTransfer(msg.sender,amount.mul(1e18));
}
```

Spelling mistake in function name. Functions are: **withdrawl()** and Also in require message.

“withdrawl” should be “**withdrawal**”.

Contract : DubaiNfts_stacking -> Deposite word

```
contract DubaiNfts_stacking is Ownable {
    using SafeBEP20 for IBEP20;
    using SafeMath for uint256;
    uint256 tokenPerSecond;
    uint256 public minimumDepositeAmount;
    uint256 public maximumDepositeAmount;
    IBEP20 public immutable rewardToken;
    IERC721 public immutable stakedToken;
```


Email: audit@EtherAuthority.io

Functions: deposite(), calclulateReward() -> deposite word

```
function deposite(uint256 tokenId) public  {
    require(hasStart==true,"Stacking is not Start yet");
    require(stakedToken.ownerOf(tokenId)==msg.sender,"User can not hold this NFT");
    stakedToken.transferFrom(msg.sender,address(this),tokenId);
    userInfo(tokenId);
}
function calclulateReward(uint256 id) public view returns(uint256){
    require(Stack[id].isWithdrawal==false,"Amount Already withdrawl");
    uint256 depositeTime=Stack[id].time;
    uint256 currentTime=block.timestamp;
    uint256 reward = (currentTime.sub(depositeTime)).mul(tokenPerSecond);
    return reward;
}
```

Spelling mistake in variable and function name.

"**deposite**" word should be "**deposit**."

DubaiNFT.sol

Function: createMarketSale() -> alredy finnished word

```
function createMarketSale(    infinite gas
    address token,
    uint256 tokenId,
    uint256 amount
    ) public payable  {
        uint price = idToMarketItem[token][tokenId].price;
        bool sold = idToMarketItem[token][tokenId].sold;
        uint256 signerID = idToMarketItem[token][tokenId].tokenId;
        require(!sold , "This Sale has alredy finnished of this Item");
        if(idToMarketItem[token][tokenId].token){
```

Spelling mistake in require message

"alredy" word should be "**already**",

"finnished" word should be "**finished**".

Stacking.sol

Function: withdrawl() -> withdrawl word

Email: audit@EtherAuthority.io

```
function withdrawl(uint256 amount) public onlyOwner {    infinite gas
    require(
        stakedToken.balanceOf(address(this)) >= amount,
        "Contract balance is low"
    );
    stakedToken.safeTransfer(msg.sender, amount);
}
```

Spelling mistake in variable and function name.

Functions are: **withdrawl()** and Also in require message.

"withdrawl" should be "**withdrawal"**.

Function: constructor() -> Deposite word

```
constructor(IBEP20 _stakedToken) {
    minimumDepositeAmount = 100;
    maximumDepositeAmount = 1000;
    APY=100;
    stakedToken = IBEP20(_stakedToken);
}
```

Variables: minimumDepositeAmount, maximumDepositeAmount -> Deposite word

```
uint256 public minimumDepositeAmount;
uint256 public maximumDepositeAmount;
uint256 [] public withdrawTime=[15 minutes,30 minutes
IBEP20 public stakedToken;
struct stack {
    uint256 amount;
    address userAddress;
    uint256 withdrwalTime;
    uint256 depositeTime;
    uint256 stackId;
```

Function: calclulateReward() -> Deposite word

Email: audit@EtherAuthority.io

```
function calclulateReward(uint256 id,address useraddress) public view returns (uint256) {
    if(Stack[useraddress][id].amount>0){
    uint256 depositeTime = Stack[useraddress][id].depositeTime;
    uint256 Time = block.timestamp.sub(depositeTime);
    uint256 reward=Stack[useraddress][id].amount.mul(APY).mul(Time).div(100).div(1 days);
```

Functions: setDepositeAmount(), deposite(), userInfo() -> Deposite word

```
function setDepositeAmount(uint256 minimumAmount, uint256 maximumAmount)
    public onlyOwner
{
    maximumDepositeAmount = maximumAmount;
    minimumAmount = minimumAmount;
}

function setAPY(uint8 _APY) public onlyOwner {
    APY = _APY;
}

function userInfo(uint256 amount, uint256 WTime) internal {
    Stack[msg.sender][currentID].amount = amount;
    Stack[msg.sender][currentID].userAddress = msg.sender;
    Stack[msg.sender][currentID].withdrwalTime = block.timestamp.add(withdrawTime[WTime]);
    Stack[msg.sender][currentID].depositeTime = block.timestamp;
    Stack[msg.sender][currentID].stackId = currentID;
    Stack[msg.sender][currentID].isWithdrawal = false;
    currentID = currentID + 1;
}

function deposite(uint256 amount,uint8 withdrwalTime) public {
    require(hasStart, "Stacking is not Start yet");
    require(
        amount >= minimumDepositeAmount.mul(1e18),
        "Amount Must be Gratar than minimum Deposite Amount"
    );
    require(
        amount <= maximumDepositeAmount.mul(1e18),
        "Amount Must be less than maximum Deposite Amount"
    );
```

“deposite” word should be “**deposit**”.

“**setDepositeAmount**” should be “**setDepositAmount**”.

Resolution: Correct the spelling.

Status: This is fixed in the revised smart contract code.

(6) Owner can withdraw all funds:

DubaiNfts_stacking.sol

Email: audit@EtherAuthority.io

Function: withdrawl()

```
function withdrawl(uint256 amount) public onlyOwner{    infinite gas
        require(rewardToken.balanceOf(address(this))>= amount.mul(1e18),"Contract balance is low");
        rewardToken.safeTransfer(msg.sender,amount.mul(1e18));
}
```

Owner can withdraw all the balance of the contract by using the withdrawal function, and only the owner can call this function.

Stacking.sol

Function: withdrawl()

```
function withdrawl(uint256 amount) public onlyOwner {    infinite gas
    require(
        stakedToken.balanceOf(address(this)) >= amount,
        "Contract balance is low"
    );
    stakedToken.safeTransfer(msg.sender, amount);
}
```

Owner can withdraw all the balance of the contract by using withdrawl function, and here are only owner can call this function.

Resolution: If it is a part of the plan then disregard this issue otherwise the owner has to set charity wallet as excluded from fee.

Status: This is acknowledged in the revised smart contract code.

Email: audit@EtherAuthority.io

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

Artcom.sol

- pauseSale: Pause the sale by the owner.
- startICO: Start the sale by the owner.
- changePrice: Change Price of the token by the owner.
- withdrwal: Withdrawal token by the owner.
- setDrop: Airdrop values can be set by the owner.
- mint: The owner can create `amount` tokens and assigns them to `msg.sender`, increasing the total supply.
- burn: The owner can burn `amount` tokens and decrease the total supply.

Stacking.sol

- toggelStacking: The _hasStart status can be set by the owner.
- setDepositeAmount: Deposit amount can be set by the owner.
- setAPY: APY amount can be set by the owner.
- withdrawl: Withdrawal token by the owner.

DubaiNfts_stacking.sol

- startStacking: Start Stacking can be set by the owner.
- pauseStacking: Pause Stacking can be set by the owner.
- setTokenPerSecond: Set Token Per Second by the owner.
- deposite: Deposite token by the owner.
- withdrawl: Withdrawal token by the owner.

DubaiNFT.sol

- changeFees: A new fee value can be set by the owner.
- changeToken: A new token address can be set by the owner.

Email: audit@EtherAuthority.io

DubaiNfts.sol

- setBaseURI: The baseURI can be set by the owner.
- startSale: Start Sale can be set by the owner.
- pauseSale: Pause Sale can be set by the owner.
- setContractFees: Contract Fees can be set by the owner.
- updateAdmin: A new admin address can be set by the owner.
- setFloorPrice: Floor price can be set by the owner.
- setTokenPrice: Token price can be set by the owner.
- mint: The mint tokens by the admin.
- burn: The burn tokens by the admin.

BridgeBSC.sol

- updateOwner: A new owner address can be set by the owner.
- updateFees: A new fee value can be set by the owner.
- updateToken: A new token address can be set by the owner.

Ownable.sol

- renounceOwnership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
- transferOwnership: Current owner can transfer ownership of the contract to a new account.
- _checkOwner: Throws if the sender is not the owner.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the smart contract once its function is completed.

Email: audit@EtherAuthority.io

Conclusion

We were given a contract code in the form of a web link. And we have used all possible tests based on given objects as files. We had observed 2 high severity issues, 1 low severity issue and 6 informational issues in the smart contracts. These all issues are fixed / acknowledged in the revised contract code. **So, the smart contracts are ready for the mainnet deployment**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

The security state of the reviewed contract, based on standard audit procedure scope, is **"Secured".**

Email: audit@EtherAuthority.io

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Email: audit@EtherAuthority.io

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

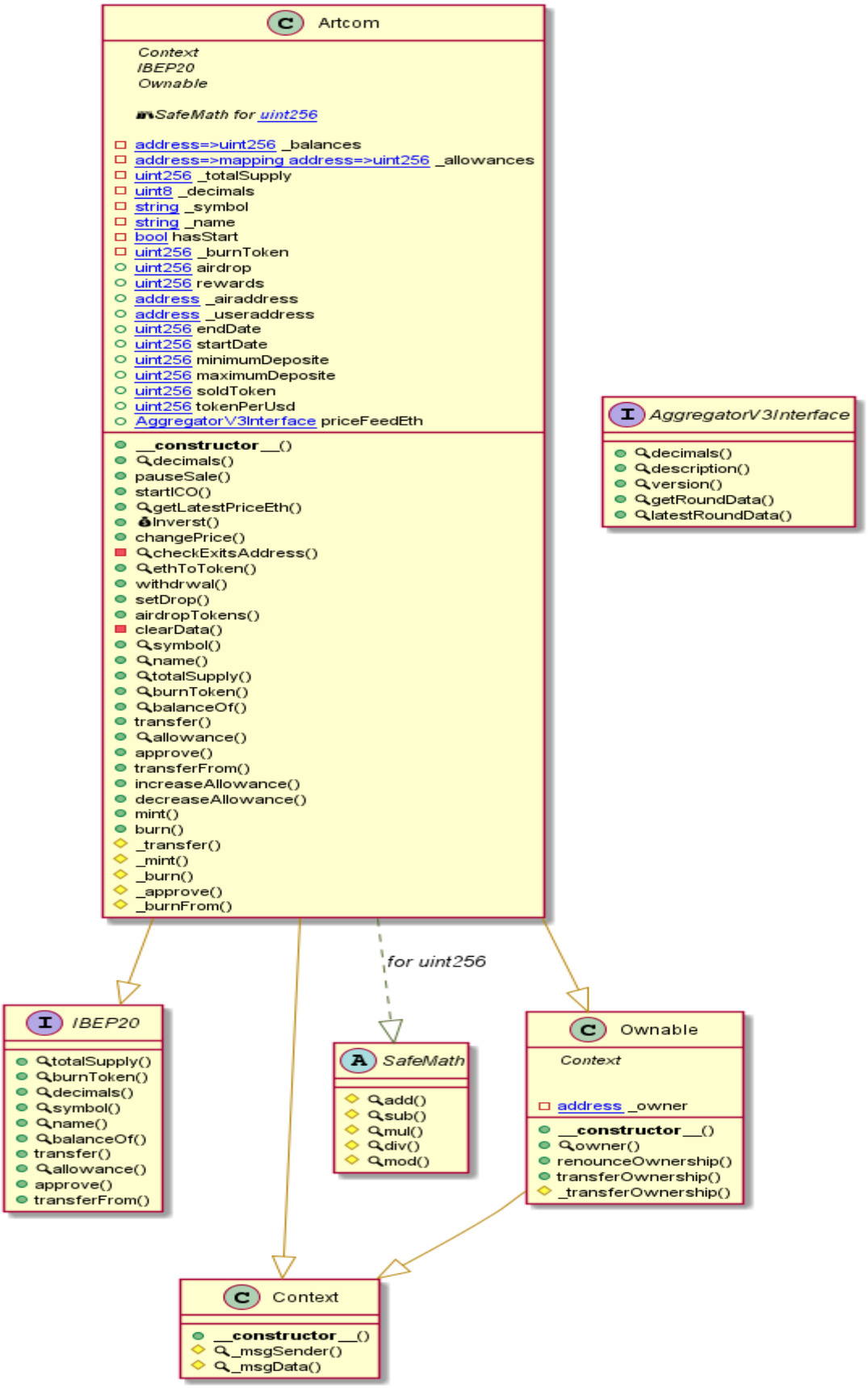
Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

Email: audit@EtherAuthority.io

Appendix

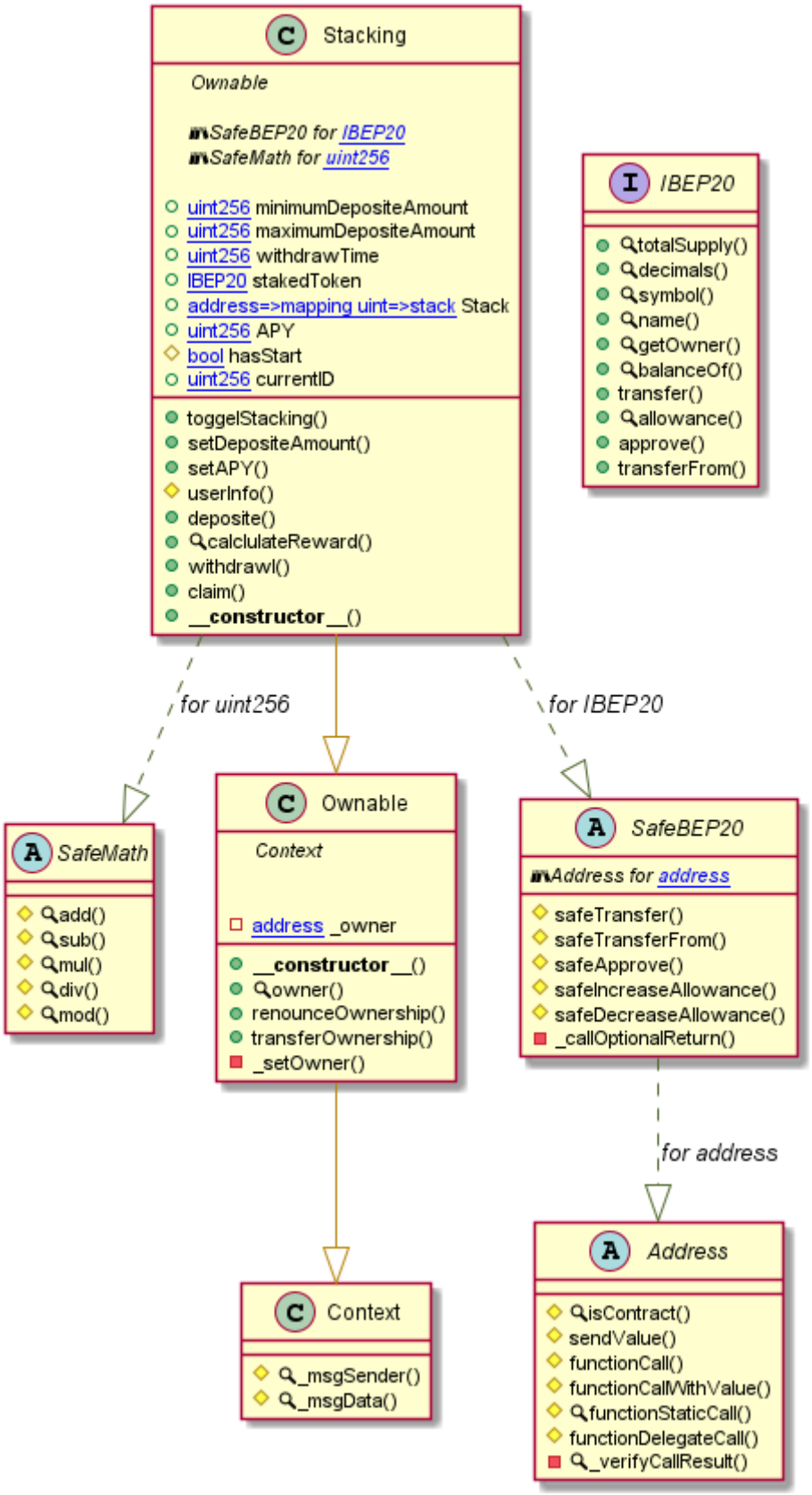
Code Flow Diagram - Dubai NFT

Artcom Diagram



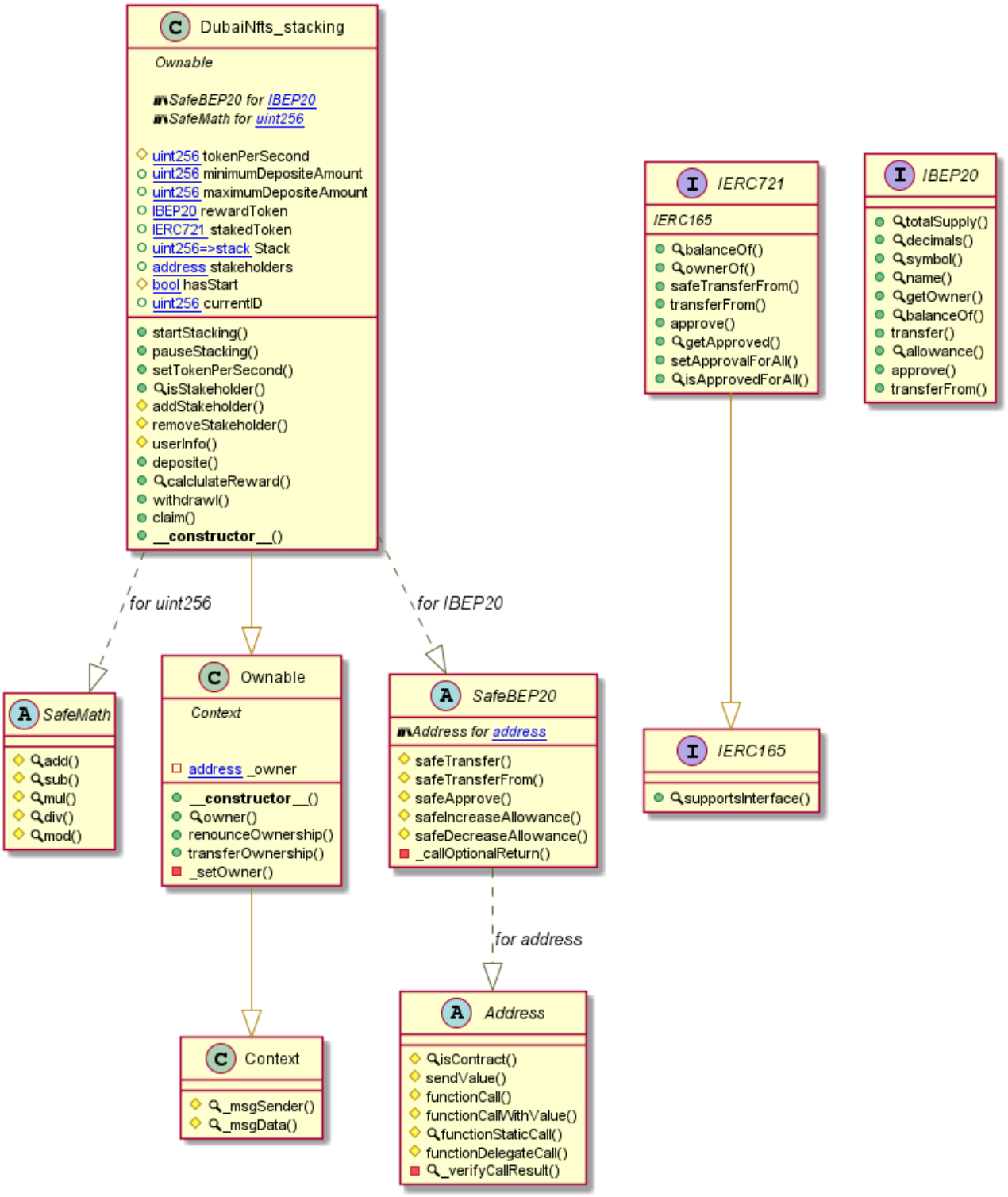
Email: audit@EtherAuthority.io

Stacking Diagram



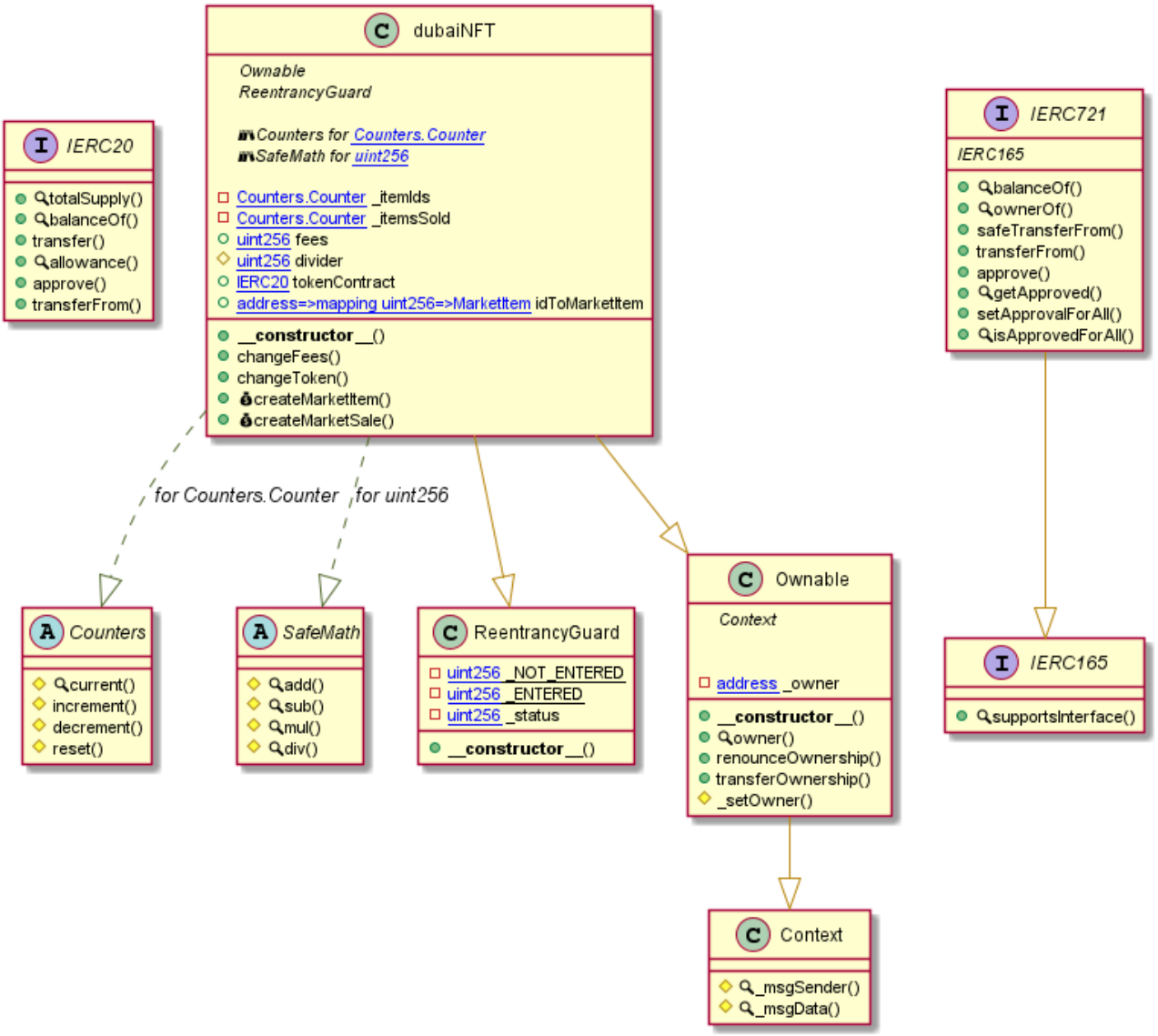
Email: audit@EtherAuthority.io

DubaiNfts_stacking Diagram



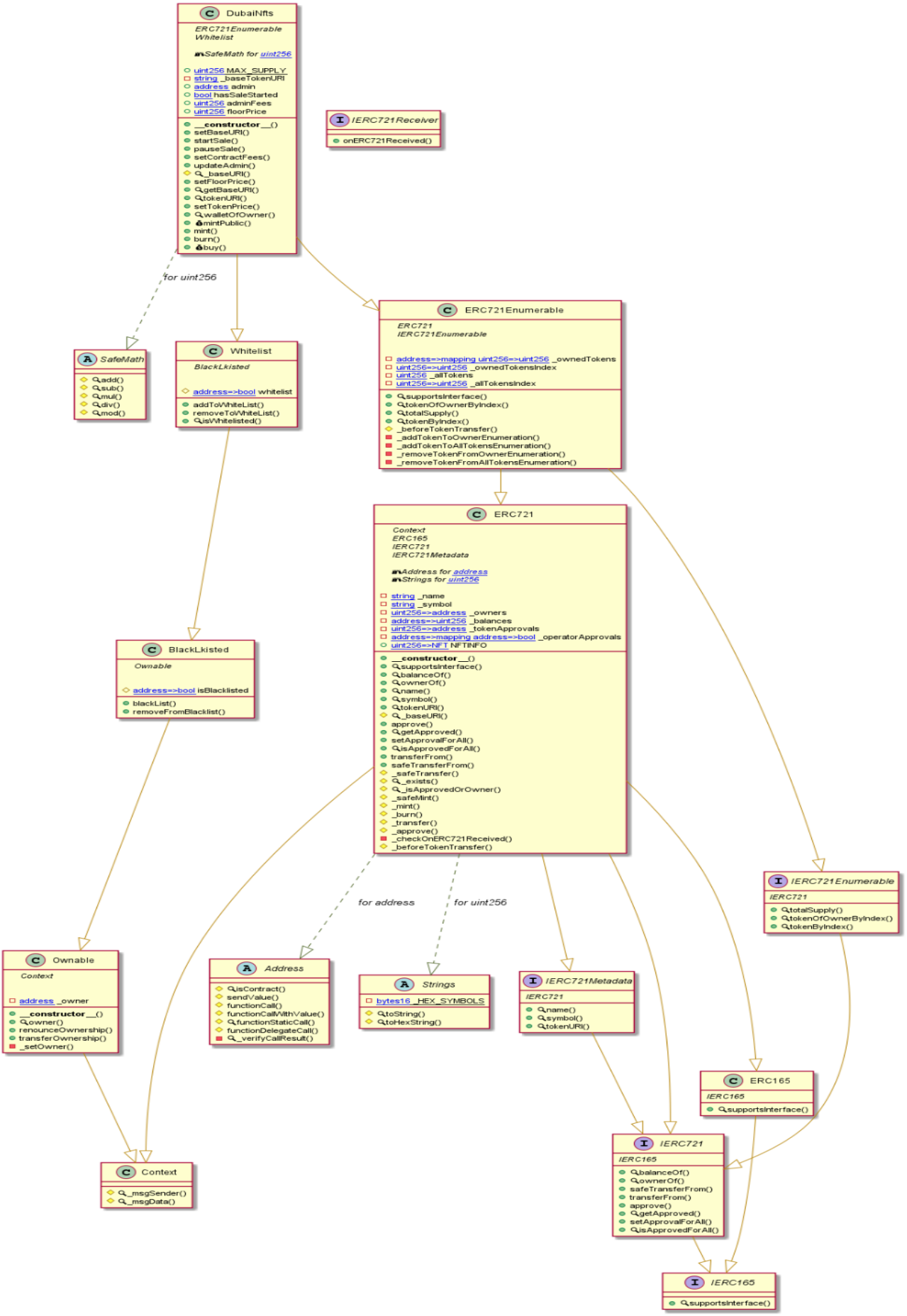
Email: audit@EtherAuthority.io

dubaiNFT Diagram



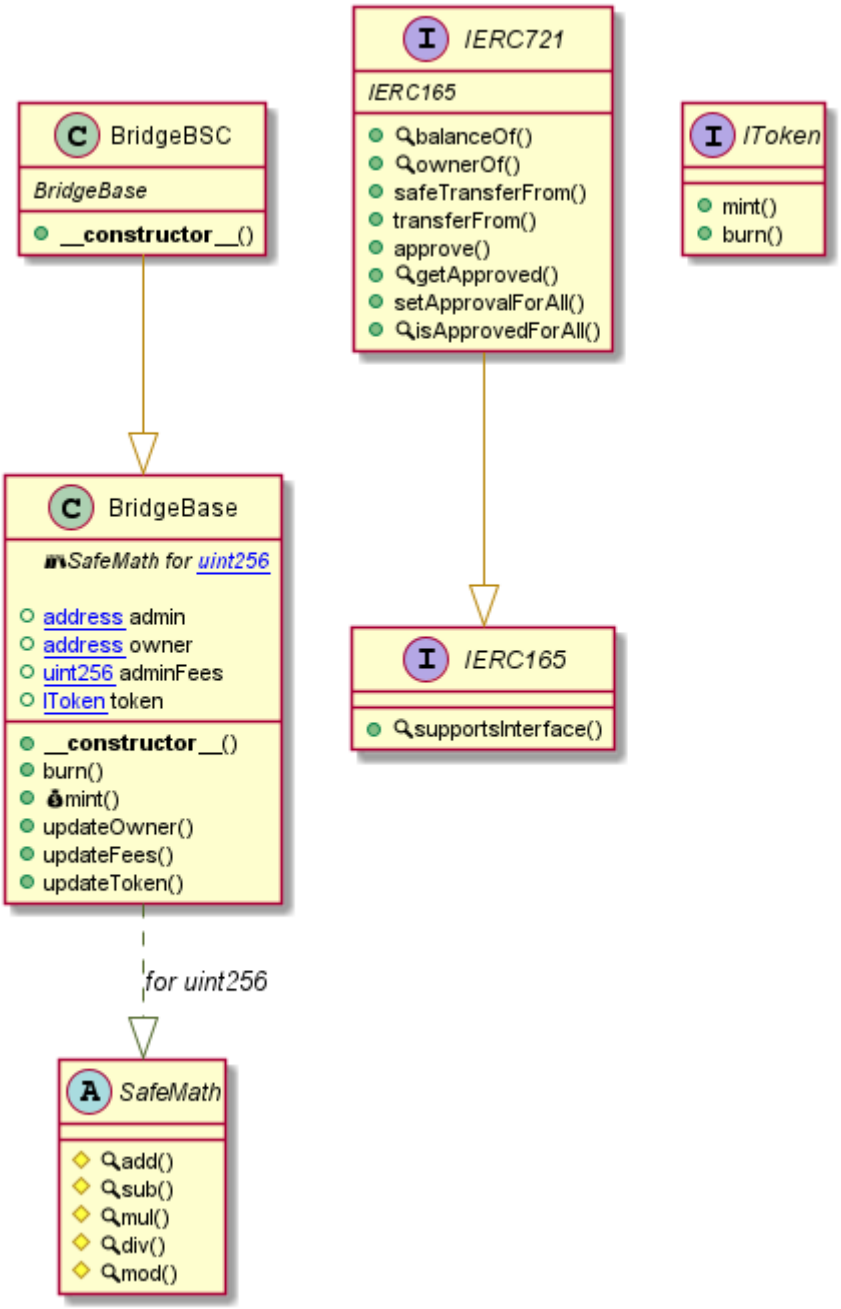
Email: audit@EtherAuthority.io

DubaiNfts Diagram



Email: audit@EtherAuthority.io

BridgeBSC Diagram



Email: audit@EtherAuthority.io

Slither Results Log

Slither log >> Artcom.sol

```
Artcom.allowance(address,address).owner (Artcom.sol#563) shadows:
        - Ownable.owner() (Artcom.sol#304-306) (function)
Artcom._approve(address,address,uint256).owner (Artcom.sol#727) shadows:
        - Ownable.owner() (Artcom.sol#304-306) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Artcom.changePrice(uint256) (Artcom.sol#461-463) should emit an event for:
        - tokenPerUsd = _tokenPerUsd (Artcom.sol#462)
Artcom.burn(uint256) (Artcom.sol#649-653) should emit an event for:
        - _burnToken = amount + _burnToken (Artcom.sol#651)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Artcom.Inverst() (Artcom.sol#446-459) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < endDate,ICO Completed) (Artcom.sol#449)
Artcom.withdrwal() (Artcom.sol#483-486) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > endDate,ICO Is Not completed yet) (Artcom.sol#484)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Artcom.Inverst() (Artcom.sol#446-459) compares to a boolean constant:
        -require(bool,string)(hasStart == true,Sale is not started) (Artcom.sol#448)
Artcom.airdropTokens(address) (Artcom.sol#494-508) compares to a boolean constant:
        -require(bool,string)(_isExist == false,Already Dropped) (Artcom.sol#502)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
Pragma version0.8.9 (Artcom.sol#10) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter Artcom.startICO(uint256)._endDate (Artcom.sol#433) is not in mixedCase
Function Artcom.Inverst() (Artcom.sol#446-459) is not in mixedCase
Parameter Artcom.changePrice(uint256)._tokenPerUsd (Artcom.sol#461) is not in mixedCase
Parameter Artcom.checkExitsAddress(address)._userAdd (Artcom.sol#465) is not in mixedCase
Parameter Artcom.ethToToken(uint256)._amount (Artcom.sol#476) is not in mixedCase
Parameter Artcom.setDrop(uint256,uint256)._airdrop (Artcom.sol#488) is not in mixedCase
Parameter Artcom.setDrop(uint256,uint256)._rewards (Artcom.sol#488) is not in mixedCase
Parameter Artcom.airdropTokens(address).ref_address (Artcom.sol#494) is not in mixedCase
Variable Artcom._airaddress (Artcom.sol#392) is not in mixedCase
Variable Artcom._useraddress (Artcom.sol#393) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (Artcom.sol#121)" inContext (Artcom.sol#111-124)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Artcom.constructor() (Artcom.sol#401-413) uses literals with too many digits:
        - _totalSupply = 5000000000 * 10 ** 18 (Artcom.sol#405)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Artcom._decimals (Artcom.sol#385) should be immutable
Artcom._name (Artcom.sol#387) should be immutable
Artcom._symbol (Artcom.sol#386) should be immutable
Artcom.maximumDeposite (Artcom.sol#397) should be immutable
Artcom.minimumDeposite (Artcom.sol#396) should be immutable
Artcom.priceFeedEth (Artcom.sol#400) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Artcom.sol analyzed (6 contracts with 84 detectors), 33 result(s) found
```

Slither log >> Stacking.sol

```
Stacking.setDepositeAmount(uint256,uint256) (Stacking.sol#529-534) should emit an event for:
        - maximumDepositeAmount = maximumAmount (Stacking.sol#532)
Stacking.setAPY(uint8) (Stacking.sol#536-538) should emit an event for:
        - APY = _APY (Stacking.sol#537)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Reentrancy in Stacking.deposite(uint256,uint8) (Stacking.sol#550-570):
        External calls:
        - stakedToken.safeTransferFrom(msg.sender,address(this),amount) (Stacking.sol#564-568)
        State variables written after the call(s):
        - userInfo(amount,withdrwalTime) (Stacking.sol#569)
                - Stack[msg.sender][currentID].amount = amount (Stacking.sol#541)
                - Stack[msg.sender][currentID].userAddress = msg.sender (Stacking.sol#542)
                - Stack[msg.sender][currentID].withdrwalTime = block.timestamp.add(withdrawTime[WTime]) (Stacking.sol#543)
                - Stack[msg.sender][currentID].depositeTime = block.timestamp (Stacking.sol#544)
                - Stack[msg.sender][currentID].stackId = currentID (Stacking.sol#545)
                - Stack[msg.sender][currentID].isWithdrawal = false (Stacking.sol#546)
        - userInfo(amount,withdrwalTime) (Stacking.sol#569)
                - currentID = currentID + 1 (Stacking.sol#547)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Stacking.claim(uint256) (Stacking.sol#591-604) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > Stack[msg.sender][id].withdrwalTime,Can not withdrwal Amount before Time) (St
acking.sol#594)
        - require(bool,string)(stakedToken.balanceOf(address(this)) >= totalAmount,Insufficent Balance) (Stacking.sol#597-600)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Stacking.claim(uint256) (Stacking.sol#591-604) compares to a boolean constant:
        -require(bool,string)(Stack[msg.sender][id].isWithdrawal == false,Amount Already withdrawl) (Stacking.sol#593)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```


Email: audit@EtherAuthority.io

```
Pragma version0.8.9 (Stacking.sol#5) allows old versions
solc-0.8.9 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) (Stacking.sol#265-275):
        - (success) = recipient.call{value: amount}() (Stacking.sol#270)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Stacking.sol#306-321):
        - (success,returndata) = target.call{value: value}(data) (Stacking.sol#317-319)
Low level call in Address.functionStaticCall(address,bytes,string) (Stacking.sol#336-344):
        - (success,returndata) = target.staticcall(data) (Stacking.sol#342)
Low level call in Address.functionDelegateCall(address,bytes,string) (Stacking.sol#358-366):
        - (success,returndata) = target.delegatecall(data) (Stacking.sol#364)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Struct Stacking.stack (Stacking.sol#511-518) is not in CapWords
Parameter Stacking.toggelStacking(bool)._hasStart (Stacking.sol#525) is not in mixedCase
Parameter Stacking.setAPY(uint8)._APY (Stacking.sol#536) is not in mixedCase
Parameter Stacking.userInfo(uint256,uint256).WTime (Stacking.sol#540) is not in mixedCase
Variable Stacking.Stack (Stacking.sol#520) is not in mixedCase
Variable Stacking.APY (Stacking.sol#521) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable Stacking.maximumDepositeAmount (Stacking.sol#508) is too similar to Stacking.minimumDepositeAmount (Stacking.sol#507)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

Stacking.minimumDepositeAmount (Stacking.sol#507) should be immutable
Stacking.stakedToken (Stacking.sol#510) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Stacking.sol analyzed (7 contracts with 84 detectors), 35 result(s) found
```

Slither log >> DubaiNfts_stacking.sol

```
DubaiNfts_stacking.setTokenPerSecond(uint8) (DubaiNfts_stacking.sol#564-566) should emit an event for:
        - tokenPerSecond = _tokenPerSecond (DubaiNfts_stacking.sol#565)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Reentrancy in DubaiNfts_stacking.deposite(uint256) (DubaiNfts_stacking.sol#599-604):
        External calls:
        - stakedToken.transferFrom(msg.sender,address(this),tokenId) (DubaiNfts_stacking.sol#602)
        State variables written after the call(s):
        - userInfo(tokenId) (DubaiNfts_stacking.sol#603)
                - Stack[currentID].tokenId = tokenId (DubaiNfts_stacking.sol#589)
                - Stack[currentID].amount = uint8(1) (DubaiNfts_stacking.sol#590)
                - Stack[currentID].userAddress = msg.sender (DubaiNfts_stacking.sol#591)
                - Stack[currentID].time = block.timestamp (DubaiNfts_stacking.sol#592)
                - Stack[currentID].stackId = currentID (DubaiNfts_stacking.sol#593)
                - Stack[currentID].isWithdrawal = false (DubaiNfts_stacking.sol#594)
        - userInfo(tokenId) (DubaiNfts_stacking.sol#603)
                - currentID = currentID + 1 (DubaiNfts_stacking.sol#596)
        - userInfo(tokenId) (DubaiNfts_stacking.sol#603)
                - stakeholders.push(_stakeholder) (DubaiNfts_stacking.sol#578)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

DubaiNfts_stacking.claim(uint256) (DubaiNfts_stacking.sol#618-627) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(rewardToken.balanceOf(address(this)) >= totalAmount,Insufficent Balance) (DubaiNfts_stacking.so
l#623)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
DubaiNfts_stacking.startStacking() (DubaiNfts_stacking.sol#555-558) compares to a boolean constant:
        -require(bool,string)(hasStart == false,Stacking Already Started) (DubaiNfts_stacking.sol#556)
DubaiNfts_stacking.pauseStacking() (DubaiNfts_stacking.sol#560-563) compares to a boolean constant:
        -require(bool,string)(hasStart == true,Stacking Already Paused) (DubaiNfts_stacking.sol#561)
DubaiNfts_stacking.deposite(uint256) (DubaiNfts_stacking.sol#599-604) compares to a boolean constant:
        -require(bool,string)(hasStart == true,Stacking is not Start yet) (DubaiNfts_stacking.sol#600)
DubaiNfts_stacking.calclulateReward(uint256) (DubaiNfts_stacking.sol#605-611) compares to a boolean constant:
        -require(bool,string)(Stack[id].isWithdrawal == false,Amount Already withdrawl) (DubaiNfts_stacking.sol#606)
DubaiNfts_stacking.claim(uint256) (DubaiNfts_stacking.sol#618-627) compares to a boolean constant:
        -require(bool,string)(Stack[id].isWithdrawal == false,Amount Already withdrawl) (DubaiNfts_stacking.sol#620)
DubaiNfts_stacking.claim(uint256) (DubaiNfts_stacking.sol#618-627) compares to a boolean constant:
        -require(bool,string)(hasStart == true,Stacking is not Start yet) (DubaiNfts_stacking.sol#619)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
Pragma version^0.8.9 (DubaiNfts_stacking.sol#7) allows old versions
solc-0.8.9 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) (DubaiNfts_stacking.sol#385-389):
        - (success) = recipient.call{value: amount}() (DubaiNfts_stacking.sol#387)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (DubaiNfts_stacking.sol#403-408):
        - (success,returndata) = target.call{value: value}(data) (DubaiNfts_stacking.sol#406)
Low level call in Address.functionStaticCall(address,bytes,string) (DubaiNfts_stacking.sol#414-418):
        - (success,returndata) = target.staticcall(data) (DubaiNfts_stacking.sol#416)
Low level call in Address.functionDelegateCall(address,bytes,string) (DubaiNfts_stacking.sol#424-428):
        - (success,returndata) = target.delegatecall(data) (DubaiNfts_stacking.sol#426)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Contract DubaiNfts_stacking (DubaiNfts_stacking.sol#533-634) is not in CapWords
Struct DubaiNfts_stacking.stack (DubaiNfts_stacking.sol#541-548) is not in CapWords
Parameter DubaiNfts_stacking.setTokenPerSecond(uint8)._tokenPerSecond (DubaiNfts_stacking.sol#564) is not in mixedCase
Parameter DubaiNfts_stacking.isStakeholder(address)._address (DubaiNfts_stacking.sol#567) is not in mixedCase
Parameter DubaiNfts_stacking.addStakeholder(address)._stakeholder (DubaiNfts_stacking.sol#575) is not in mixedCase
Parameter DubaiNfts_stacking.removeStakeholder(address)._stakeholder (DubaiNfts_stacking.sol#581) is not in mixedCase
Variable DubaiNfts_stacking.Stack (DubaiNfts_stacking.sol#550) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable DubaiNfts_stacking.maximumDepositeAmount (DubaiNfts_stacking.sol#538) is too similar to DubaiNfts_stacking.minimumDep
ositeAmount (DubaiNfts_stacking.sol#537)
Variable DubaiNfts_stacking.addStakeholder(address)._stakeholder (DubaiNfts_stacking.sol#575) is too similar to DubaiNfts_stac
king.stakeholders (DubaiNfts_stacking.sol#551)
Variable DubaiNfts_stacking.removeStakeholder(address)._stakeholder (DubaiNfts_stacking.sol#581) is too similar to DubaiNfts_s
tacking.stakeholders (DubaiNfts_stacking.sol#551)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
```


Email: audit@EtherAuthority.io

```
DubaiNfts_stacking.constructor() (DubaiNfts_stacking.sol#628-632) uses literals with too many digits:
        - tokenPerSecond = 1000000000000 (DubaiNfts_stacking.sol#629)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

DubaiNfts_stacking.maximumDepositeAmount (DubaiNfts_stacking.sol#538) should be constant
DubaiNfts_stacking.minimumDepositeAmount (DubaiNfts_stacking.sol#537) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
DubaiNfts_stacking.sol analyzed (9 contracts with 84 detectors), 48 result(s) found
```

Slither log >> dubaiNFT.sol

```
dubaiNFT.changeFees(uint256) (DubaiNft.sol#411-413) should emit an event for:
        - fees = _fee (DubaiNft.sol#412)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Reentrancy in dubaiNFT.createMarketSale(address,uint256,uint256) (DubaiNft.sol#453-488):
        External calls:
        - (tokenContract).transferFrom(msg.sender,owner(),tax) (DubaiNft.sol#466)
        - (tokenContract).transferFrom(msg.sender,idToMarketItem[token][tokenId].creator,royaltiesTax) (DubaiNft.sol#467)
        - (tokenContract).transferFrom(msg.sender,idToMarketItem[token][tokenId].owner,amount.sub(royaltiesTax).sub(tax)) (Dub
aiNft.sol#468)
        - IERC721(idToMarketItem[token][tokenId].nftContract).transferFrom(idToMarketItem[token][tokenId].owner,msg.sender,sig
nerID) (DubaiNft.sol#478)
        External calls sending eth:
        - address(owner()).transfer(tax_scope_1) (DubaiNft.sol#473)
        - idToMarketItem[token][tokenId].creator.transfer(royaltiesTax_scope_0) (DubaiNft.sol#474)
        - idToMarketItem[token][tokenId].owner.transfer((msg.value).sub(royaltiesTax_scope_0).sub(tax_scope_1)) (DubaiNft.sol#
475)
        Event emitted after the call(s):
        - MarketItemSold(tokenId,msg.sender) (DubaiNft.sol#480-483)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
Pragma version^0.8.9 (DubaiNft.sol#3) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Contract dubaiNFT (DubaiNft.sol#368-490) is not in CapWords
Parameter dubaiNFT.changeFees(uint256)._fee (DubaiNft.sol#411) is not in mixedCase
Parameter dubaiNFT.changeToken(IERC20)._newToken (DubaiNft.sol#414) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in dubaiNFT.createMarketSale(address,uint256,uint256) (DubaiNft.sol#453-488):
        External calls:
        - address(owner()).transfer(tax_scope_1) (DubaiNft.sol#473)
        - idToMarketItem[token][tokenId].creator.transfer(royaltiesTax_scope_0) (DubaiNft.sol#474)
        - idToMarketItem[token][tokenId].owner.transfer((msg.value).sub(royaltiesTax_scope_0).sub(tax_scope_1)) (DubaiNft.sol#
475)
        State variables written after the call(s):
        - idToMarketItem[token][tokenId].owner = address(msg.sender) (DubaiNft.sol#484)
        - idToMarketItem[token][tokenId].sold = false (DubaiNft.sol#486)
        - idToMarketItem[token][tokenId].isListed = false (DubaiNft.sol#487)
        Event emitted after the call(s):
        - MarketItemSold(tokenId,msg.sender) (DubaiNft.sol#480-483)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

dubaiNFT.divider (DubaiNft.sol#374) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
DubaiNft.sol analyzed (9 contracts with 84 detectors), 18 result(s) found
```

Slither log >> DubaiNfts.sol

```
DubaiNfts.tokenURI(uint256).totalSupply (DubaiNfts.sol#1424) shadows:
        - ERC721Enumerable.totalSupply() (DubaiNfts.sol#1196-1198) (function)
        - IERC721Enumerable.totalSupply() (DubaiNfts.sol#1145) (function)
DubaiNfts.walletOfOwner(address)._owner (DubaiNfts.sol#1435) shadows:
        - Ownable._owner (DubaiNfts.sol#167) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

DubaiNfts.updateAdmin(address) (DubaiNfts.sol#1403-1405) should emit an event for:
        - admin = newAdmin (DubaiNfts.sol#1404)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

DubaiNfts.setFloorPrice(uint256) (DubaiNfts.sol#1410-1412) should emit an event for:
        - floorPrice = FloorPrice (DubaiNfts.sol#1411)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

DubaiNfts.updateAdmin(address).newAdmin (DubaiNfts.sol#1403) lacks a zero-check on :
                - admin = newAdmin (DubaiNfts.sol#1404)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (DubaiNfts.sol#1099)' in ERC721._checkOnERC721Re
ceived(address,address,uint256,bytes) (DubaiNfts.sol#1092-1113) potentially used before declaration: retval == IERC721Receiver
(to).onERC721Received.selector (DubaiNfts.sol#1100)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (DubaiNfts.sol#1101)' in ERC721._checkOnERC721Re
ceived(address,address,uint256,bytes) (DubaiNfts.sol#1092-1113) potentially used before declaration: reason.length == 0 (Dubai
Nfts.sol#1102)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (DubaiNfts.sol#1101)' in ERC721._checkOnERC721Re
ceived(address,address,uint256,bytes) (DubaiNfts.sol#1092-1113) potentially used before declaration: revert(uint256,uint256)(3
2 + reason,mload(uint256)(reason)) (DubaiNfts.sol#1106)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```

```
Reentrancy in DubaiNfts.mint(address,uint256,uint256,address) (DubaiNfts.sol#1466-1475):
        External calls:
        - _safeMint(to,tokenId) (DubaiNfts.sol#1468)
                - IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (DubaiNfts.sol#1099-1109)
        State variables written after the call(s):
        - NFTINFO[tokenId].price = floorPrice (DubaiNfts.sol#1469)
        - NFTINFO[tokenId].royalties = royalties (DubaiNfts.sol#1470)
        - NFTINFO[tokenId].ownerAddress = address(to) (DubaiNfts.sol#1471)
        - NFTINFO[tokenId].creatorAddress = address(creator) (DubaiNfts.sol#1472)
        - NFTINFO[tokenId].sell = false (DubaiNfts.sol#1473)
```


Email: audit@EtherAuthority.io

```
Reentrancy in DubaiNfts.mintPublic(address,uint256,uint256) (DubaiNfts.sol#1451-1464):
        External calls:
        - _safeMint(to,tokenId) (DubaiNfts.sol#1457)
                - IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (DubaiNfts.sol#1099-1109)
        External calls sending eth:
        - address(owner()).transfer(msg.value) (DubaiNfts.sol#1456)
        State variables written after the call(s):
        - NFTINFO[tokenId].price = floorPrice (DubaiNfts.sol#1458)
        - NFTINFO[tokenId].royalties = royalties (DubaiNfts.sol#1459)
        - NFTINFO[tokenId].ownerAddress = address(to) (DubaiNfts.sol#1460)
        - NFTINFO[tokenId].creatorAddress = address(to) (DubaiNfts.sol#1461)
        - NFTINFO[tokenId].sell = false (DubaiNfts.sol#1462)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Pragma version^0.8.9 (DubaiNfts.sol#11) allows old versions
solc-0.8.9 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) (DubaiNfts.sol#474-479):
        - (success) = recipient.call{value: amount}() (DubaiNfts.sol#477)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (DubaiNfts.sol#542-553):
        - (success,returndata) = target.call{value: value}(data) (DubaiNfts.sol#551)
Low level call in Address.functionStaticCall(address,bytes,string) (DubaiNfts.sol#571-580):
        - (success,returndata) = target.staticcall(data) (DubaiNfts.sol#578)
Low level call in Address.functionDelegateCall(address,bytes,string) (DubaiNfts.sol#598-607):
        - (success,returndata) = target.delegatecall(data) (DubaiNfts.sol#605)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (DubaiNfts.sol#902) is not in mixedCase
Variable ERC721.NFTINFO (DubaiNfts.sol#758) is not in mixedCase
Parameter BlackLkisted.blackList(address)._user (DubaiNfts.sol#1319) is not in mixedCase
Parameter BlackLkisted.removeFromBlacklist(address)._user (DubaiNfts.sol#1325) is not in mixedCase
Parameter Whitelist.addToWhiteList(address[])._address (DubaiNfts.sol#1342) is not in mixedCase
Parameter Whitelist.removeToWhiteList(address[])._address (DubaiNfts.sol#1348) is not in mixedCase
Parameter Whitelist.isWhitelisted(address)._address (DubaiNfts.sol#1354) is not in mixedCase
Parameter DubaiNfts.setContractFees(uint256)._sellingFees (DubaiNfts.sol#1397) is not in mixedCase
Parameter DubaiNfts.setFloorPrice(uint256).floorPrice (DubaiNfts.sol#1410) is not in mixedCase
Parameter DubaiNfts.tokenURI(uint256)._tokenId (DubaiNfts.sol#1418) is not in mixedCase
Parameter DubaiNfts.walletOfOwner(address)._owner (DubaiNfts.sol#1435) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
                - _safeMint(to,tokenId) (DubaiNfts.sol#1457)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
DubaiNfts.sol analyzed (16 contracts with 84 detectors), 55 result(s) found
```

Slither log >> BridgeBSC.sol

```
dubaiNFT.changeFees(uint256) (DubaiNft.sol#411-413) should emit an event for:
        - fees = _fee (DubaiNft.sol#412)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Reentrancy in dubaiNFT.createMarketSale(address,uint256,uint256) (DubaiNft.sol#453-488):
        External calls:
        - (tokenContract).transferFrom(msg.sender,owner(),tax) (DubaiNft.sol#466)
        - (tokenContract).transferFrom(msg.sender,idToMarketItem[token][tokenId].creator,royaltiesTax) (DubaiNft.sol#467)
        - (tokenContract).transferFrom(msg.sender,idToMarketItem[token][tokenId].owner,amount.sub(royaltiesTax).sub(tax)) (Dub
aiNft.sol#468)
        - IERC721(idToMarketItem[token][tokenId].nftContract).transferFrom(idToMarketItem[token][tokenId].owner,msg.sender,sig
nerID) (DubaiNft.sol#478)
        External calls sending eth:
        - address(owner()).transfer(tax_scope_1) (DubaiNft.sol#473)
        - idToMarketItem[token][tokenId].creator.transfer(royaltiesTax_scope_0) (DubaiNft.sol#474)
        - idToMarketItem[token][tokenId].owner.transfer((msg.value).sub(royaltiesTax_scope_0).sub(tax_scope_1)) (DubaiNft.sol#
475)
        Event emitted after the call(s):
        - MarketItemSold(tokenId,msg.sender) (DubaiNft.sol#480-483)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
Pragma version^0.8.9 (DubaiNft.sol#3) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Contract dubaiNFT (DubaiNft.sol#368-490) is not in CapWords
Parameter dubaiNFT.changeFees(uint256)._fee (DubaiNft.sol#411) is not in mixedCase
Parameter dubaiNFT.changeToken(IERC20)._newToken (DubaiNft.sol#414) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in dubaiNFT.createMarketSale(address,uint256,uint256) (DubaiNft.sol#453-488):
        External calls:
        - address(owner()).transfer(tax_scope_1) (DubaiNft.sol#473)
        - idToMarketItem[token][tokenId].creator.transfer(royaltiesTax_scope_0) (DubaiNft.sol#474)
        - idToMarketItem[token][tokenId].owner.transfer((msg.value).sub(royaltiesTax_scope_0).sub(tax_scope_1)) (DubaiNft.sol#
475)
        State variables written after the call(s):
        - idToMarketItem[token][tokenId].owner = address(msg.sender) (DubaiNft.sol#484)
        - idToMarketItem[token][tokenId].sold = false (DubaiNft.sol#486)
        - idToMarketItem[token][tokenId].isListed = false (DubaiNft.sol#487)
        Event emitted after the call(s):
        - MarketItemSold(tokenId,msg.sender) (DubaiNft.sol#480-483)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

dubaiNFT.divider (DubaiNft.sol#374) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
DubaiNft.sol analyzed (9 contracts with 84 detectors), 18 result(s) found
```


Email: audit@EtherAuthority.io

Solidity Static Analysis

Artcom.sol

Security

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 434:18:

Gas & Economy

Gas costs:

Gas requirement of function Artcom.airdropTokens is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 494:4:

Delete dynamic array:

The "delete" operation when applied to a dynamically sized array in Solidity generates code to delete each of the elements contained. If the array is large, this operation can surpass the block gas limit and raise an OOG exception. Also nested dynamically sized objects can produce the same results.

more

Pos: 491:8:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 467:8:

Email: audit@EtherAuthority.io

Miscellaneous

Constant/View/Pure functions:

SafeMath.mod(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 254:2:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 729:4:

Stacking.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 312:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 600:16:

Gas & Economy

Email: audit@EtherAuthority.io

Gas costs:

Gas requirement of function Stacking.claim is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 597:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 226:4:

Miscellaneous

Constant/View/Pure functions:

Stacking.withdrawl(uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 589:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 603:8:

DubaiNfts_stacking.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 403:4:

Email: audit@EtherAuthority.io

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in DubaiNfts_stacking.deposite(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 599:4:

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 426:50:

Gas & Economy

Gas costs:

Gas requirement of function DubaiNfts_stacking.isStakeholder is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 567:5:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 568:8:

Miscellaneous

Email: audit@EtherAuthority.io

Constant/View/Pure functions:

DubaiNfts_stacking.withdrawl(uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 613:4:

Similar variable names:

DubaiNfts_stacking.removeStakeholder(address) : Variables have very similar names "_isStakeholder" and "_stakeholder". Note: Modifiers are currently not considered by this static analysis.

Pos: 583:12:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 623:8:

dubaiNFT.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in dubaiNFT.createMarketItem(address,uint256,uint256,bool,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 393:4:

Gas & Economy

Email: audit@EtherAuthority.io

Gas costs:

Gas requirement of function dubaiNFT.createMarketSale is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 429:4:

Miscellaneous

Constant/View/Pure functions:

IERC20.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 14:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 439:16:

DubaiNfts.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in DubaiNfts.buy(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1489:4:

Gas & Economy

Email: audit@EtherAuthority.io

Gas costs:

Gas requirement of function DubaiNfts.walletOfOwner is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1435:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1343:8:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1491:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 1311:8:

Email: audit@EtherAuthority.io

Security

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 344:6:

Gas & Economy

Gas costs:

Gas requirement of function BridgeBase.mint is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 333:2:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 357:8:

Similar variable names:

BridgeBase.mint(address,uint256,address,address,uint256) : Variables have very similar names "token" and "tokenId".
Pos: 343:6:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 66:16:

Email: audit@EtherAuthority.io

Solhint Linter

Artcom.sol

```
Artcom.sol:10:1: Error: Compiler version 0.8.7 does not satisfy the r
semver requirement
Artcom.sol:114:3: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
Artcom.sol:114:19: Error: Code contains empty blocks
Artcom.sol:295:3: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
Artcom.sol:378:1: Error: Contract has 19 states declarations but
allowed no more than 15
Artcom.sol:401:3: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
Artcom.sol:434:19: Error: Avoid to make time-based decisions in your
business logic
Artcom.sol:446:4: Error: Function name must be in mixedCase
Artcom.sol:449:16: Error: Avoid to make time-based decisions in your
business logic
Artcom.sol:484:17: Error: Avoid to make time-based decisions in your
business logic
Artcom.sol:488:67: Error: Visibility modifier must be first in list
of modifiers
Artcom.sol:494:28: Error: Variable name must be in mixedCase
```

Stacking.sol

```
Stacking.sol:466:18: Error: Parse error: missing ';' at '{'
```

DubaiNfts_stacking.sol

```
DubaiNfts_stacking.sol:506:18: Error: Parse error: missing ';' at '{'
```

dubaiNFT.sol

```
DubaiNft.sol:44:18: Error: Parse error: missing ';' at '{'
DubaiNft.sol:52:18: Error: Parse error: missing ';' at '{'
```


Email: audit@EtherAuthority.io

DubaiNfts.sol

```
DubaiNfts.sol:11:1: Error: Compiler version ^0.8.15 does not satisfy
the r semver requirement
DubaiNfts.sol:174:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
DubaiNfts.sol:452:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
DubaiNfts.sol:477:28: Error: Avoid using low level calls.
DubaiNfts.sol:551:51: Error: Avoid using low level calls.
DubaiNfts.sol:605:51: Error: Avoid using low level calls.
DubaiNfts.sol:621:17: Error: Avoid using inline assembly. It is
acceptable only in rare cases
DubaiNfts.sol:758:34: Error: Variable name must be in mixedCase
DubaiNfts.sol:763:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
DubaiNfts.sol:1105:21: Error: Avoid using inline assembly. It is
acceptable only in rare cases
DubaiNfts.sol:1133:24: Error: Code contains empty blocks
DubaiNfts.sol:1317:5: Error: Explicitly mark visibility of state
DubaiNfts.sol:1333:5: Error: Explicitly mark visibility of state
DubaiNfts.sol:1372:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
DubaiNfts.sol:1410:28: Error: Variable name must be in mixedCase
```

BridgeBSC.sol

```
BridgeBSC.sol:6:1: Error: Compiler version 0.8.16 does not satisfy
the r semver requirement
BridgeBSC.sol:316:3: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
BridgeBSC.sol:328:7: Error: Avoid to make time-based decisions in
your business logic
BridgeBSC.sol:344:7: Error: Avoid to make time-based decisions in
your business logic
BridgeBSC.sol:363:3: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
BridgeBSC.sol:363:48: Error: Code contains empty blocks
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io

Email: audit@EtherAuthority.io