Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	HCP Protocol
Platform:	Binance Smart Chain
Language:	Solidity
Date:	August 10th, 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 the HCP team to perform the Security audit of the HCP 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 August 10th, 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

- The HCP protocol covers multiple contracts, and all contracts have different functions.
    - **HCP:** Contract for new registration, deposit, commission, minting, unstaking/staking NFT.
    - **A_NFT:** Contract sets BaseURI and baseExtension.
    - **B_NFT:** Sets mint contract address.
    - **C_NFT:** Contract sets supply tokens.
    - **D_NFT:** Contract for minting tokens.
    - **E_NFT:** Contract for adding addresses to Whitelist.
    - **HCToken:** Contract for pause/unpause contracts.
    - **USDTToken:** Contract for pause/unpause contracts.

- The smart contracts have functions like mintContract, changeSupply, Mint, pause, unpause, register, deposite, mintNFT, stakeNFT, unStakeNFT, withdrawToken, etc.

Audit scope

Name	Code Review and Security Analysis Report for HCP Smart Contracts
Platform	**BSC / Solidity**
File 1	HCP.sol
File 1 Initial code link	0xc14502fb7383f1d8d56bcd0e995608c876f9aeea
File 2	A_NFT.sol
File 2 Initial code link	0xfca595EB0F4b3695fE870905728b44aa930e52c9
File 3	B_NFT.sol
File 3 Initial code link	0xca65ED59F6156Cd4071C7eBDac470D1B329d92ac
File 4	C_NFT.sol
File 4 Initial code link	0x65B99fe3225D2fF78DA829327e88c4f80Da75385
File 5	D_NFT.sol
File 5 Initial code link	0x9FB50b1D5f348de2DC9a1819FDb50b3fe98a6E8a
File 6	E_NFT.sol
File 6 Initial code link	0xAE5aAA8e5B76d0E429646CD3Bfd8B6C127f87ea0
File 7	HCToken.sol
File 7 Initial code link	0xB33dA41d67A216Ebf6BE5D972128dB1d4eFEC35C
File 8	USDTToken.sol
File 8 Initial code link	0x5ff81fdf548cd8af4b3d1310675d943159a306d0
Audit Date	August 10th, 2023

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 HCP.sol • Tikcet NO: 2 • Owner Fee: 10% • Share: 4% • Multiplier: 10 (Amount to be deposited) **The owner has control over the following functions:** • Owner can register. • Owner can deposit the amount from the address. • Owner can mint NFT tokens. • Owner can withdraw the amount from the account. • Owner can Stake/unStake NFT. • Owner can renounce ownership. • Current owner can transfer the ownership.	**YES, This is valid.**
File 2 A_NFT.sol • Name: A_NFT • Symbol: ANFT • Maximum Supply: 10000 **The owner has control over the following functions:** • Mint tokens. • Set the BaseURI and baseExtension. • Added a new HCP address. • Supply tokens. • Add Whitelister addresses. • Current owner can transfer the ownership. • Owner can renounce ownership.	**YES, This is valid.**
File 3 B_NFT.sol • Name: B_NFT • Symbol: B_NFT • Maximum Supply: 10000	**YES, This is valid.**

The owner has control over the following functions: • Mint tokens. • Set the BaseURI and baseExtension. • Added a new HCP address. • Supply tokens. • Add Whitelister addresses. • Current owner can transfer the ownership. • Owner can renounce ownership.	
File 4 C_NFT.sol • Name: C_NFT • Symbol: CNFT • Maximum Supply: 10000 **The owner has control over the following functions:** • Mint tokens. • Set the BaseURI and baseExtension. • Added a new HCP address. • Supply tokens. • Add Whitelister addresses. • Current owner can transfer the ownership. • Owner can renounce ownership.	**YES, This is valid.**
File 5 D_NFT.sol • Name: D_NFT • Symbol: DNFT • Maximum Supply: 10000 **The owner has control over the following functions:** • Mint tokens. • Set the BaseURI and baseExtension. • Added a new HCP address. • Supply tokens. • Add Whitelister addresses. • Current owner can transfer the ownership. • Owner can renounce ownership.	**YES, This is valid.**

File 6 E_NFT.sol • Name: E_NFT • Symbol: ENFT • Maximum Supply: 10000 **The owner has control over the following functions:** • Mint tokens. • Set the BaseURI and baseExtension. • Added a new HCP address. • Supply tokens. • Add Whitelister addresses. • Current owner can transfer the ownership. • Owner can renounce ownership.	**YES, This is valid.**
File 7 HCToken.sol • Name: HC Token • Symbol: HCT • Decimals: 18 • Total Supply: 5 billion **The owner has control over the following functions:** • Pause / Unpause contract. • Mint tokens. • Current owner can transfer the ownership. • Owner can renounce ownership.	**YES, This is valid.**
File 8 USDTToken.sol • Name: USDT Token • Symbol: USDT • Decimals: 18 • Total Supply: 5 billion **The owner has control over the following functions:** • Pause / Unpause contract. • Mint tokens. • Current owner can transfer the ownership. • Owner can renounce ownership.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Poorly Secured"**. Also, these contracts 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 1 critical, 0 high, 0 medium, 1 low and 5 very low level issues.

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.

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	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Moderated
	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	Moderated
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: FAILED

Code Quality

This audit scope has 8 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in HCP 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 HCP Protocol.

The HCP team has not provided unit test scripts, which would have 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 HCP smart contract code in the form of a https://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.

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.

AS-IS overview

HCP.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	owner	read	Passed	No Issue
4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	register	write	Passed	No Issue
9	_updateTeamNum	write	Passed	No Issue
10	deposite	write	Missing required feature	Refer Audit Findings
11	updatePercenatage	write	Passed	No Issue
12	updateCommissionFirstTime	write	Passed	No Issue
13	updateCommission	write	Passed	No Issue
14	timeCalculationsOfUser	read	Passed	No Issue
15	updateFirstCommission	write	Passed	No Issue
16	totalFirstCommission	read	Passed	No Issue
17	getTotalCommission	read	Passed	No Issue
18	totalSecondCommission	read	Passed	No Issue
19	updateSecondCommission	write	Passed	No Issue
20	getSecondCommissionPercentage	read	Passed	No Issue
21	checkPreviousNFT	read	Passed	No Issue
22	mintNFT	write	Passed	No Issue
23	checkUpline	read	Passed	No Issue
24	withdrawAmount	write	Passed	No Issue
25	unStakeNFT	write	Critical operation lacks event log	Refer Audit Findings
26	stakeNFT	write	Critical operation lacks event log	Refer Audit Findings
27	updateUserE	write	Passed	No Issue
28	countShareUser	read	Passed	No Issue
29	globalShares	write	Passed	No Issue
30	getShareCommission	read	Passed	No Issue
31	getuserShareCommission	write	Passed	No Issue
32	getTikcet	write	Passed	No Issue
33	repurchaseTicket	write	Critical operation lacks event log	Refer Audit Findings
34	getcountofTicket	read	Passed	No Issue

35	getDepositeInfo	read	Unused variable	Refer Audit Findings
36	getCommissionInfo	read	Passed	No Issue
37	withdrawToken	external	Critical operation lacks event log	Refer Audit Findings

A_NFT.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	_setOwner	write	Passed	No Issue
7	supportsInterface	read	Passed	No Issue
8	tokenOfOwnerByIndex	read	Passed	No Issue
9	totalSupply	read	Passed	No Issue
10	tokenByIndex	read	Passed	No Issue
11	_beforeTokenTransfer	internal	Passed	No Issue
12	_addTokenToOwnerEnumeration	write	Passed	No Issue
13	_addTokenToAllTokensEnumeration	write	Passed	No Issue
14	_removeTokenFromOwnerEnumeration	write	Passed	No Issue
15	_removeTokenFromAllTokensEnumeration	write	Passed	No Issue
16	onlyHCPContract	modifier	Passed	No Issue
17	Mint	write	access only Owner	No Issue
18	totalMintedAddress	read	Passed	No Issue
19	addWhitelister	write	access only Owner	No Issue
20	walletOfOwner	read	Passed	No Issue
21	tokenURI	read	Passed	No Issue
22	whiteList	write	access only Owner	No Issue
23	setBaseURI	write	access only Owner	No Issue
24	setbaseExtension	write	access only Owner	No Issue
25	_baseURI	internal	Passed	No Issue
26	AddETAContractAddress	external	access only Owner	No Issue
27	mintContract	write	access only HCP Contract	No Issue
28	changeSupply	write	access only Owner	No Issue

B_NFT.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	_setOwner	write	Passed	No Issue
7	supportsInterface	read	Passed	No Issue
8	tokenOfOwnerByIndex	read	Passed	No Issue
9	totalSupply	read	Passed	No Issue
10	tokenByIndex	read	Passed	No Issue
11	_beforeTokenTransfer	internal	Passed	No Issue
12	_addTokenToOwnerEnumeration	write	Passed	No Issue
13	_addTokenToAllTokensEnumeration	write	Passed	No Issue
14	_removeTokenFromOwnerEnumeration	write	Passed	No Issue
15	_removeTokenFromAllTokensEnumeration	write	Passed	No Issue
16	onlyHCPContract	modifier	Passed	No Issue
17	Mint	write	access only Owner	No Issue
18	totalMintedAddress	read	Passed	No Issue
19	addWhitelister	write	access only Owner	No Issue
20	walletOfOwner	read	Passed	No Issue
21	tokenURI	read	Passed	No Issue
22	whiteList	write	access only Owner	No Issue
23	setBaseURI	write	access only Owner	No Issue
24	setbaseExtension	write	access only Owner	No Issue
25	_baseURI	internal	Passed	No Issue
26	AddETAContractAddress	external	access only Owner	No Issue
27	mintContract	write	access only HCP Contract	No Issue
28	changeSupply	write	access only Owner	No Issue

C_NFT.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	_setOwner	write	Passed	No Issue
7	supportsInterface	read	Passed	No Issue
8	tokenOfOwnerByIndex	read	Passed	No Issue

9	totalSupply	read	Passed	No Issue
10	tokenByIndex	read	Passed	No Issue
11	_beforeTokenTransfer	internal	Passed	No Issue
12	_addTokenToOwnerEnumeration	write	Passed	No Issue
13	_addTokenToAllTokensEnumeration	write	Passed	No Issue
14	_removeTokenFromOwnerEnumeration	write	Passed	No Issue
15	_removeTokenFromAllTokensEnumeration	write	Passed	No Issue
16	onlyHCPContract	modifier	Passed	No Issue
17	Mint	write	access only Owner	No Issue
18	totalMintedAddress	read	Passed	No Issue
19	addWhitelister	write	access only Owner	No Issue
20	walletOfOwner	read	Passed	No Issue
21	tokenURI	read	Passed	No Issue
22	whiteList	write	access only Owner	No Issue
23	setBaseURI	write	access only Owner	No Issue
24	setbaseExtension	write	access only Owner	No Issue
25	_baseURI	internal	Passed	No Issue
26	AddETAContractAddress	external	access only Owner	No Issue
27	mintContract	write	access only HCP Contract	No Issue
28	changeSupply	write	access only Owner	No Issue

D_NFT.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	_setOwner	write	Passed	No Issue
7	supportsInterface	read	Passed	No Issue
8	tokenOfOwnerByIndex	read	Passed	No Issue
9	totalSupply	read	Passed	No Issue
10	tokenByIndex	read	Passed	No Issue
11	_beforeTokenTransfer	internal	Passed	No Issue
12	_addTokenToOwnerEnumeration	write	Passed	No Issue
13	_addTokenToAllTokensEnumeration	write	Passed	No Issue
14	_removeTokenFromOwnerEnumeration	write	Passed	No Issue

15	_removeTokenFromAllTokensEnumeration	write	Passed	No Issue
16	onlyHCPContract	modifier	Passed	No Issue
17	Mint	write	access only Owner	No Issue
18	totalMintedAddress	read	Passed	No Issue
19	addWhitelister	write	access only Owner	No Issue
20	walletOfOwner	read	Passed	No Issue
21	tokenURI	read	Passed	No Issue
22	whiteList	write	access only Owner	No Issue
23	setBaseURI	write	access only Owner	No Issue
24	setbaseExtension	write	access only Owner	No Issue
25	_baseURI	internal	Passed	No Issue
26	AddETAContractAddress	external	access only Owner	No Issue
27	mintContract	write	access only HCP Contract	No Issue
28	changeSupply	write	access only Owner	No Issue

E_NFT.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	_setOwner	write	Passed	No Issue
7	supportsInterface	read	Passed	No Issue
8	tokenOfOwnerByIndex	read	Passed	No Issue
9	totalSupply	read	Passed	No Issue
10	tokenByIndex	read	Passed	No Issue
11	_beforeTokenTransfer	internal	Passed	No Issue
12	_addTokenToOwnerEnumeration	write	Passed	No Issue
13	_addTokenToAllTokensEnumeration	write	Passed	No Issue
14	_removeTokenFromOwnerEnumeration	write	Passed	No Issue
15	_removeTokenFromAllTokensEnumeration	write	Passed	No Issue
16	onlyHCPContract	modifier	Passed	No Issue
17	Mint	write	access only Owner	No Issue
18	totalMintedAddress	read	Passed	No Issue
19	addWhitelister	write	access only Owner	No Issue
20	walletOfOwner	read	Passed	No Issue
21	tokenURI	read	Passed	No Issue
22	whiteList	write	access only Owner	No Issue

23	setBaseURI	write	access only Owner	No Issue
24	setbaseExtension	write	access only Owner	No Issue
25	_baseURI	internal	Passed	No Issue
26	AddETAContractAddress	external	access only Owner	No Issue
27	mintContract	write	access only HCP Contract	No Issue
28	changeSupply	write	access only Owner	No Issue

HCToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	owner	read	Passed	No Issue
4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	whenNotPaused	modifier	Passed	No Issue
9	whenPaused	modifier	Passed	No Issue
10	paused	read	Passed	No Issue
11	_requireNotPaused	internal	Passed	No Issue
12	_requirePaused	internal	Passed	No Issue
13	_pause	internal	Passed	No Issue
14	_unpause	internal	Passed	No Issue
15	burn	write	Passed	No Issue
16	burnFrom	write	Passed	No Issue
17	name	read	Passed	No Issue
18	symbol	read	Passed	No Issue
19	decimals	read	Passed	No Issue
20	totalSupply	read	Passed	No Issue
21	balanceOf	read	Passed	No Issue
22	transfer	write	Passed	No Issue
23	allowance	read	Passed	No Issue
24	approve	write	Passed	No Issue
25	transferFrom	write	Passed	No Issue
26	increaseAllowance	write	Passed	No Issue
27	decreaseAllowance	write	Passed	No Issue
28	_transfer	internal	Passed	No Issue
29	_mint	internal	Passed	No Issue
30	_burn	internal	Passed	No Issue
31	_approve	internal	Passed	No Issue
32	_spendAllowance	internal	Passed	No Issue
33	_beforeTokenTransfer	internal	Passed	No Issue
34	pause	write	access only Owner	No Issue
35	unpause	write	access only Owner	No Issue

36	mint	write	access only Owner	No Issue
37	_beforeTokenTransfer	internal	Passed	No Issue

USDTToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	owner	read	Passed	No Issue
4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	whenNotPaused	modifier	Passed	No Issue
9	whenPaused	modifier	Passed	No Issue
10	paused	read	Passed	No Issue
11	_requireNotPaused	internal	Passed	No Issue
12	_requirePaused	internal	Passed	No Issue
13	_pause	internal	Passed	No Issue
14	_unpause	internal	Passed	No Issue
15	burn	write	Passed	No Issue
16	burnFrom	write	Passed	No Issue
17	name	read	Passed	No Issue
18	symbol	read	Passed	No Issue
19	decimals	read	Passed	No Issue
20	totalSupply	read	Passed	No Issue
21	balanceOf	read	Passed	No Issue
22	transfer	write	Passed	No Issue
23	allowance	read	Passed	No Issue
24	approve	write	Passed	No Issue
25	transferFrom	write	Passed	No Issue
26	increaseAllowance	write	Passed	No Issue
27	decreaseAllowance	write	Passed	No Issue
28	_transfer	internal	Passed	No Issue
29	_mint	internal	Passed	No Issue
30	_burn	internal	Passed	No Issue
31	_approve	internal	Passed	No Issue
32	_spendAllowance	internal	Passed	No Issue
33	_beforeTokenTransfer	internal	Passed	No Issue
34	pause	write	access only Owner	No Issue
35	unpause	write	access only Owner	No Issue
36	mint	write	access only Owner	No Issue
37	_beforeTokenTransfer	internal	Passed	No Issue

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

(1) Missing required feature: **HCP.sol**

There is a feature mentioned in the requirement document for upgradation of the deposit. User has to upgrade his deposit to get the remaining commission from his downlines. And that has to be done in some blocks like 88,888, otherwise the remaining commissions will go for the owner address and the owner has to withdraw that. But in code, there is no validation.

Resolution: We suggest correcting the code to fulfill the required feature.

High Severity

No high severity vulnerabilities were found in the contract code.

Medium

No medium severity vulnerabilities were found in the contract code.

Low

(1) Critical operation lacks event log: **HCP.sol**

Missing Events: Following state changing functions should emit an event.

Below is the list of events which needs events:

1. stakeNFT
2. unStakeNFT
3. repurchaseTicket
4. withdrawToken

Resolution: We suggest adding logs for the mentioned events.

Very Low / Informational / Best practices:

(1) Unused variable: **HCP.sol**

```
mapping(address => structure.DepositeInfo) public depositeInfo;
```

```
function getDepositeInfo(address _user, uint256 index) public view returns(uint256,uint256)
    return (depositeInfo[_user].time[index],depositeInfo[_user].amount[index]);
}
```

The depositeInfo is defined but not set anywhere. depositeInfo is used in getDepositeInfo which always returns 0.

```
uint256 private baseDivider = 10000000000000000000;
```

baseDivider is defined and set but not used anywhere in the code.

Resolution: We suggest removing unused variables and functions.

(2) Variables should be constant: **HCP.sol**

```
uint256 private daySecond = 86400;
uint256 private ownerPercentage = 10;
uint256 private sharePercentage = 4;
uint256 private multipler = 10000000000000000000;
```

These variables are not changing in code. So make them constant.

Resolution: We suggest declaring them as constant to save some gas.

(3) Variables should be immutable: **HCP.sol**

```
IERC20 public HCPToken;
IERC20 public USDTToken;

interANF public interANFT_;
interANF public interBNFT_;
interANF public interCNFT_;
interANF public interDNFT_;
interANF public interENFT_;
```

Variables that are defined within the constructor but further remain unchanged should be marked as immutable to save gas and to ease the reviewing process of third-parties.

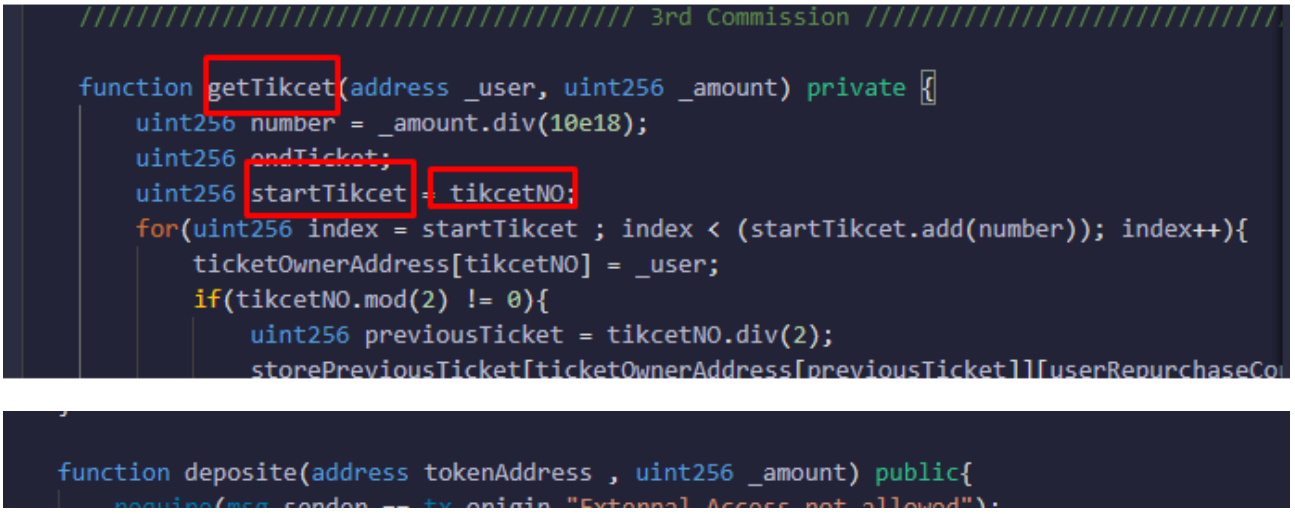
Below is the list of variables:

1. USDTToken
2. HCPToken
3. interANFT_
4. interBNFT_
5. interCNFT_
6. interDNFT_
7. interENFT_
8. defaultRefer
9. OwnerAddress.

Resolution: We suggest making them immutable to save some gas.

(4) Consider specifying function visibility to “external” instead of “public”, if that function is not being called internally. It will save some gas as well. **HCP.sol**
https://ethereum.stackexchange.com/questions/32353/what-is-the-difference-between-an-internal-external-and-public-private-function/32464

(5) Spelling mistake: **HCP.sol**



There are spelling mistakes for the word ticket and deposit at some places.
Resolution: We suggest correcting the spell for the mentioned words.

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:

HCP.sol

- register: Owner can register.
- deposite: Owner can deposit the amount from the address.
- mintNFT: Owner can mint NFT tokens.
- withdrawAmount: Owner can withdraw the amount from the account.
- unStakeNFT: Owner can unStake NFT.
- stakeNFT: Owner can Stake NFT.

A_NFT.sol

- Mint: Owner can mint tokens.
- addWhitelister: Owner can add Whitelister addresses.
- whiteList: Owner can add addresses in the Whitelist.
- setBaseURI: BaseURI can be set by the owner.
- setbaseExtension: A new base extension can be set by the owner.
- AddETAContractAddress: A new HCP Address can be set by the owner.
- mintContract: A mint contract address can be set by the HCP Contract.
- changeSupply: Supply tokens can be set by the owner.

B_NFT.sol

- Mint: Owner can mint tokens.
- addWhitelister: Owner can add Whitelister addresses.
- whiteList: Owner can add addresses in the Whitelist.
- setBaseURI: BaseURI can be set by the owner.
- setbaseExtension: A new base extension can be set by the owner.
- AddETAContractAddress: A new HCP Address can be set by the owner.
- mintContract: A mint contract address can be set by the HCP Contract.
- changeSupply: Supply tokens can be set by the owner.

C_NFT.sol

- Mint: Owner can mint tokens.
- addWhitelister: Owner can add Whitelister addresses.
- whiteList: Owner can add addresses in the Whitelist.
- setBaseURI: BaseURI can be set by the owner.
- setbaseExtension: A new base extension can be set by the owner.
- AddETAContractAddress: A new HCP Address can be set by the owner.
- mintContract: A mint contract address can be set by the HCP Contract.
- changeSupply: Supply tokens can be set by the owner.

D_NFT.sol

- Mint: Owner can mint tokens.
- addWhitelister: Owner can add Whitelister addresses.
- whiteList: Owner can add addresses in the Whitelist.
- setBaseURI: BaseURI can be set by the owner.
- setbaseExtension: A new base extension can be set by the owner.
- AddETAContractAddress: A new HCP Address can be set by the owner.
- mintContract: A mint contract address can be set by the HCP Contract.
- changeSupply: Supply tokens can be set by the owner.

E_NFT.sol

- Mint: Owner can mint tokens.
- addWhitelister: Owner can add Whitelister addresses.
- whiteList: Owner can add addresses in the Whitelist.
- setBaseURI: BaseURI can be set by the owner.
- setbaseExtension: A new base extension can be set by the owner.
- AddETAContractAddress: A new HCP Address can be set by the owner.
- mintContract: A mint contract address can be set by the HCP Contract.
- changeSupply: Supply tokens can be set by the owner.

HCToken.sol

- pause: Triggers stopped state contract can be set by the owner.
- unpause: Returns to normal state contract can be set by the owner.
- mint: Owner can mint tokens.

USDTToken.sol

- pause: Triggers stopped state contract can be set by the owner.
- unpause: Returns to normal state contract can be set by the owner.
- mint: Owner can mint tokens.

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.

Conclusion

We were given a contract code in the form of a https://testnet.bscscan.com/ web link. And we have used all possible tests based on given objects as files. We had observed 1 Critical, 1 Low and 5 Informational severity issues in the smart contracts. **So, the smart contracts are ready for the mainnet deployment after fixing the critical issue.**

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 **"Poorly 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.

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.

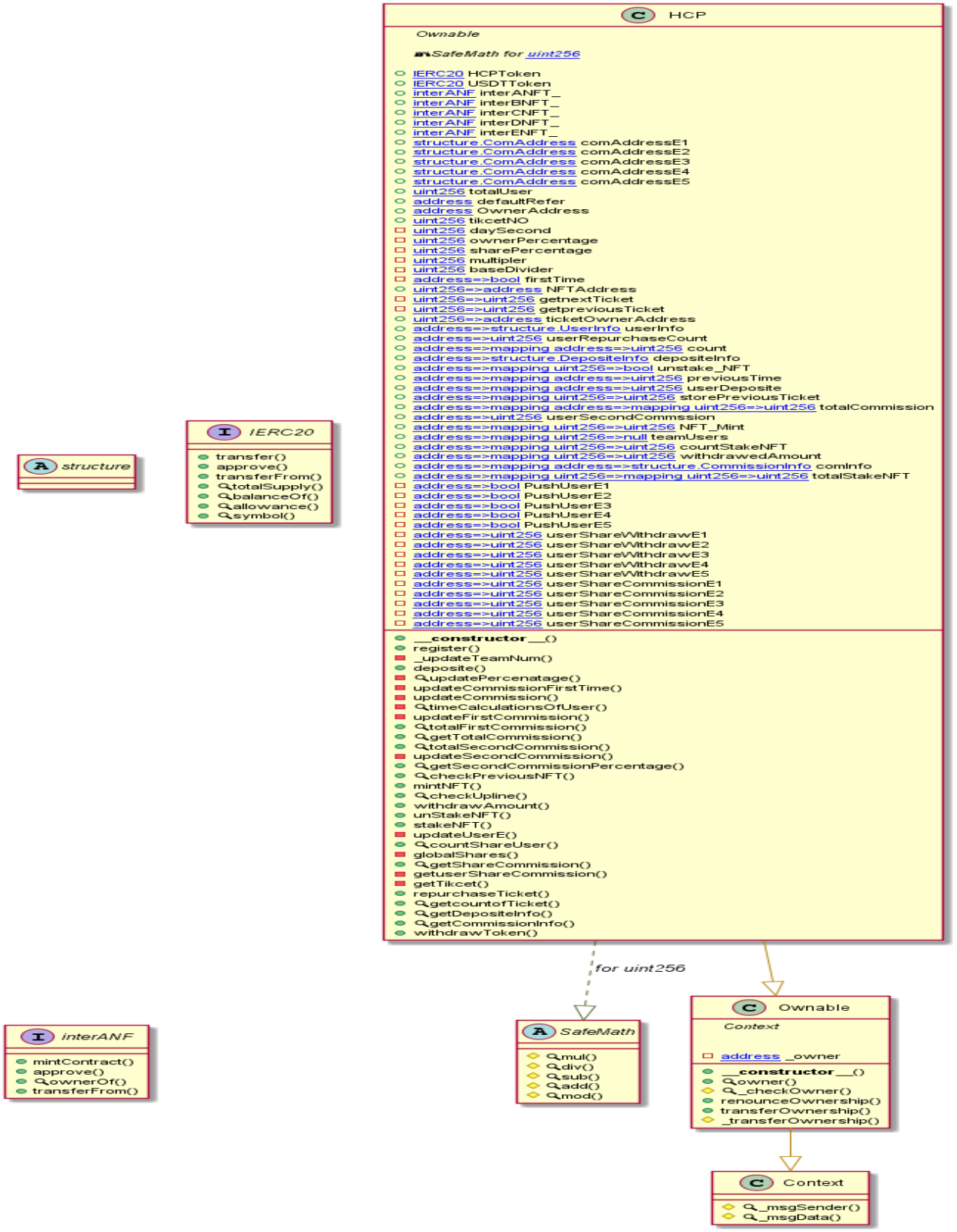
Technical Disclaimer

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.

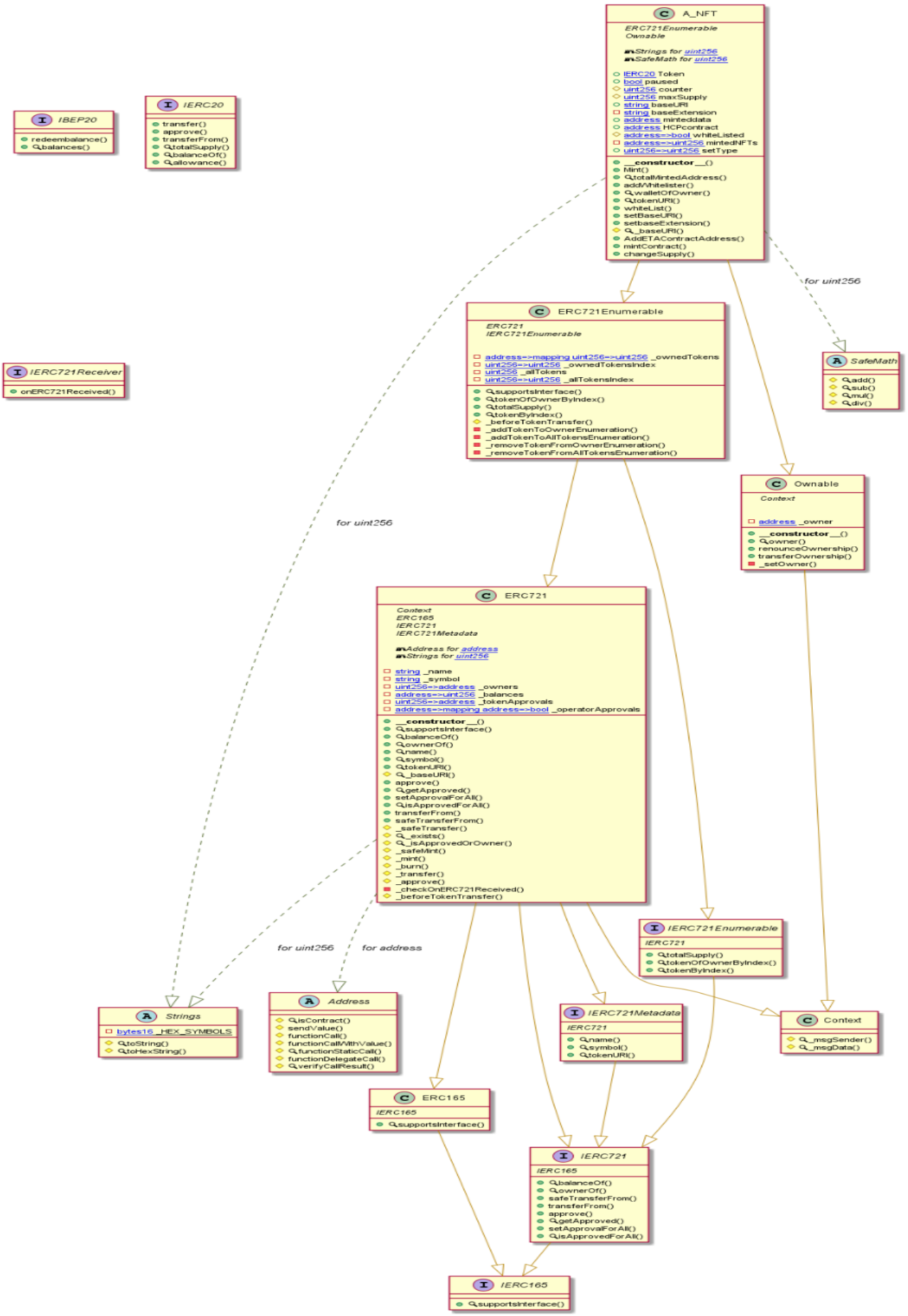
Appendix

Code Flow Diagram - HCP Protocol

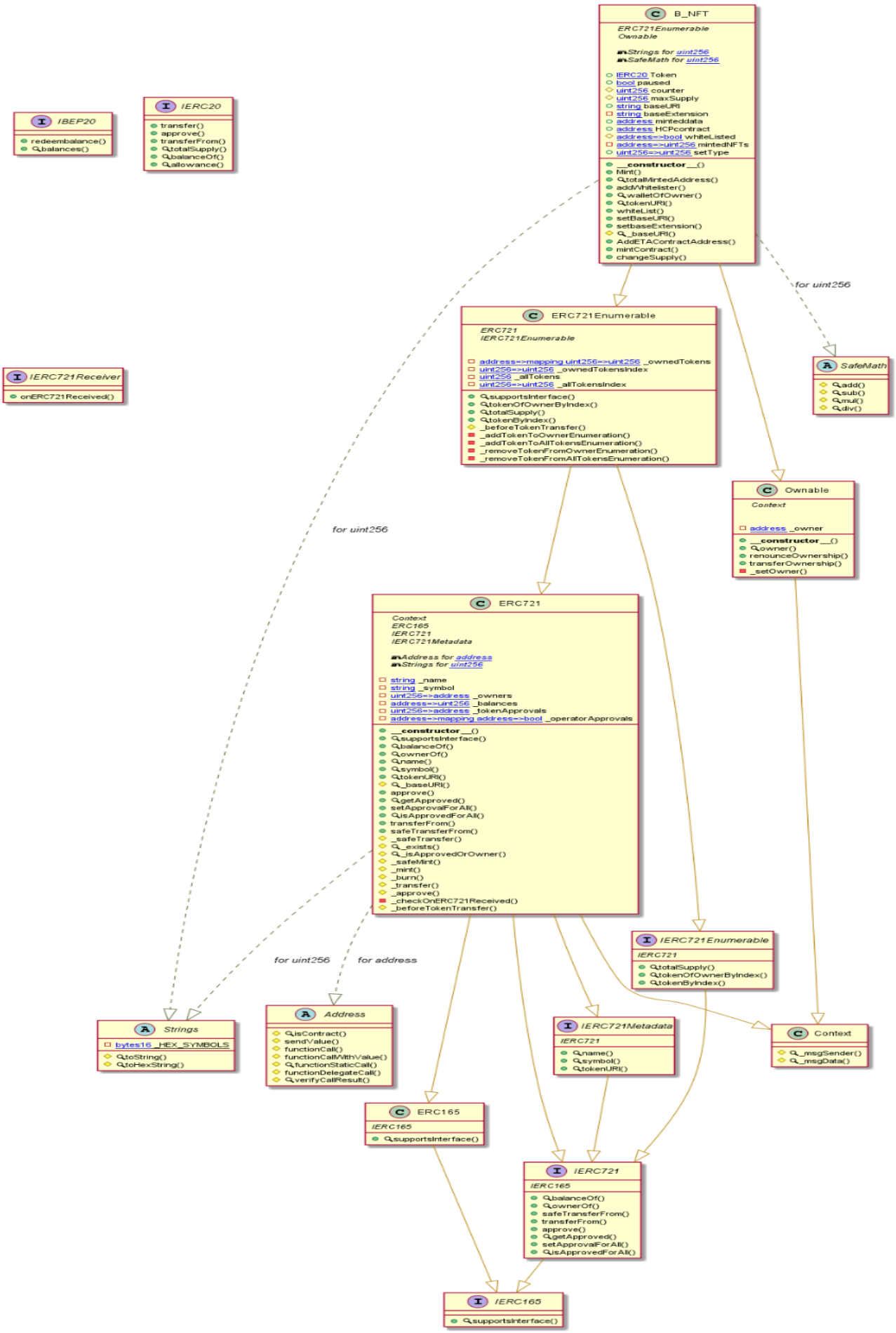
HCP Diagram



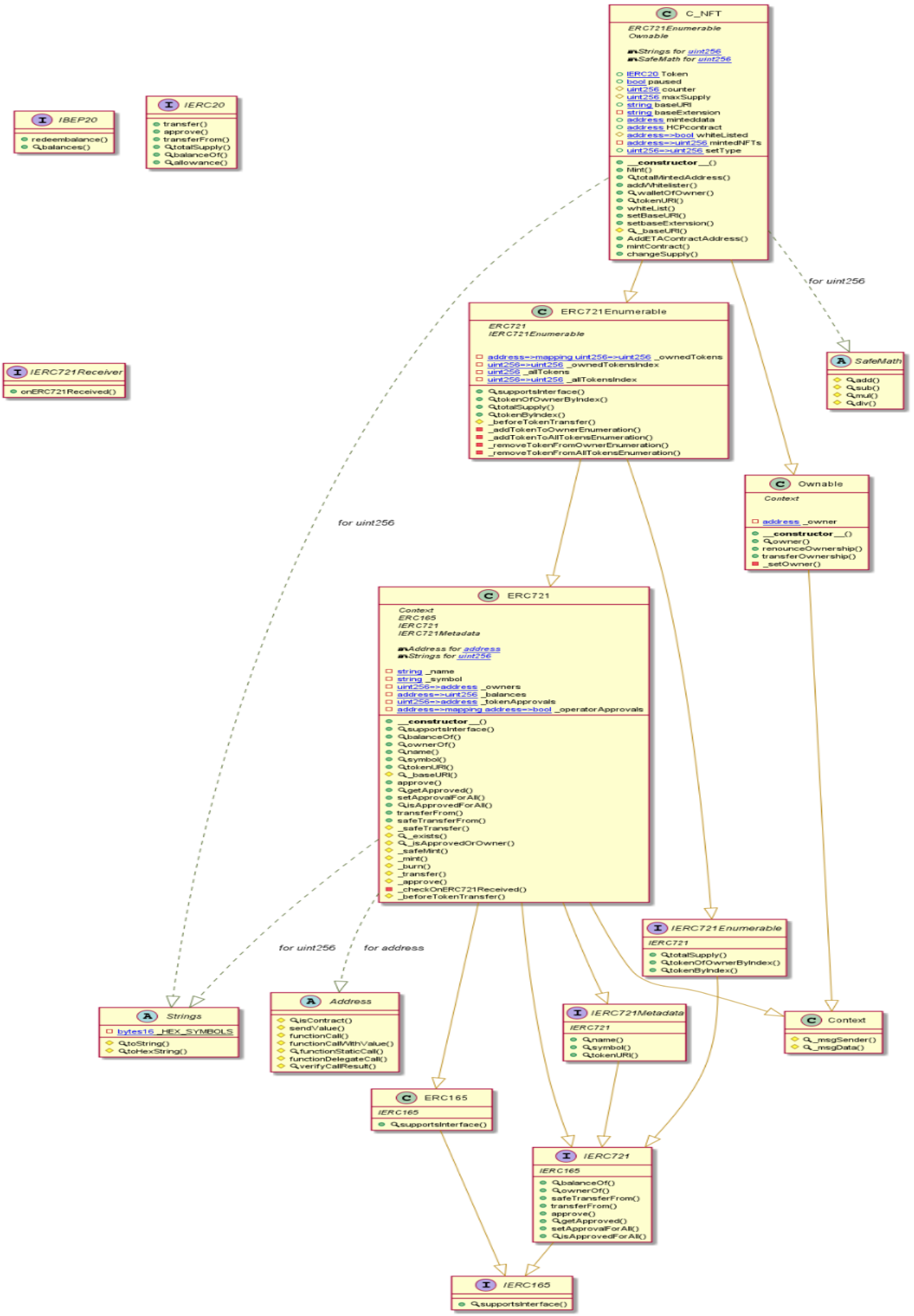
A_NFT Diagram



B_NFT Diagram

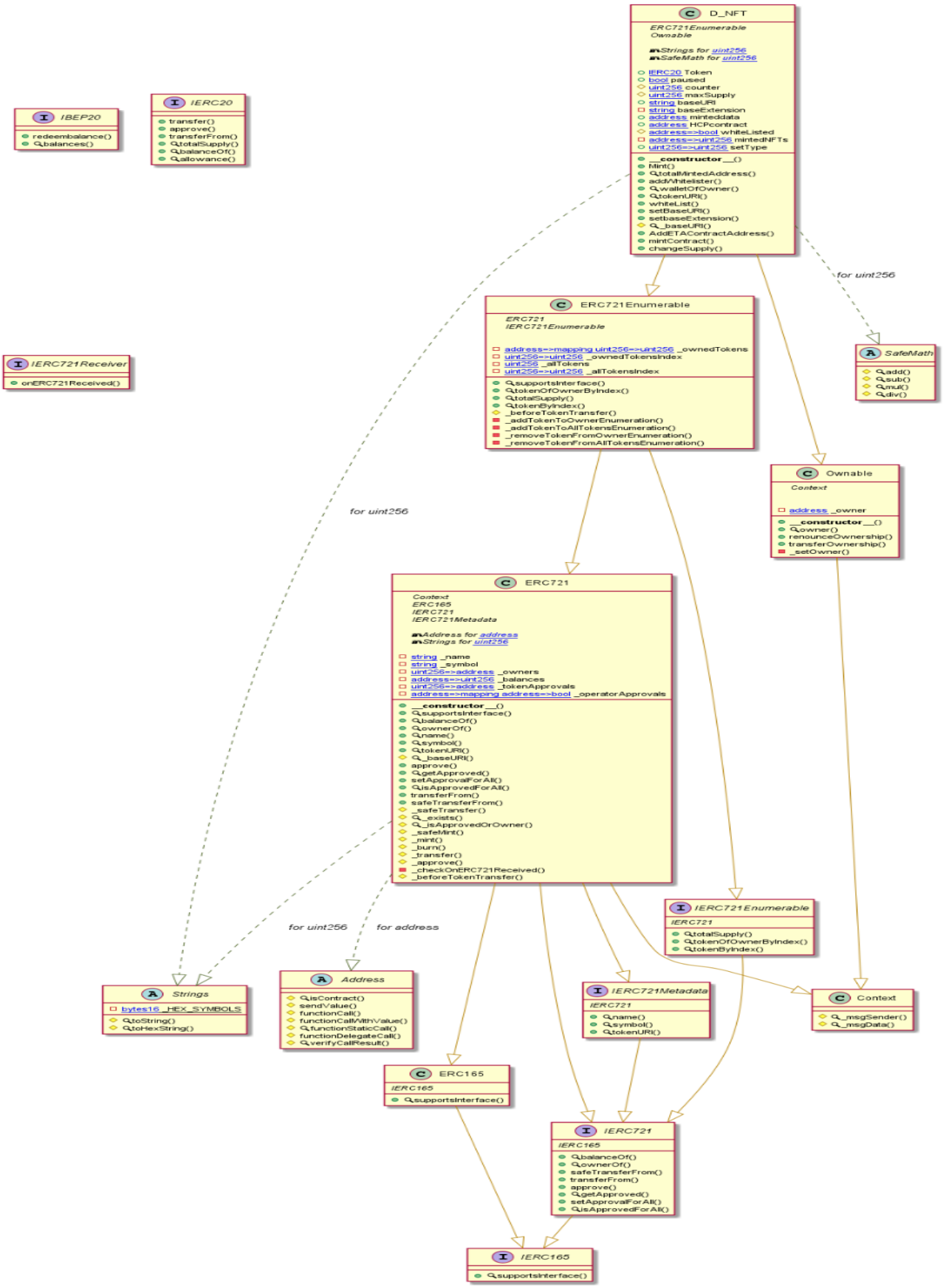


C_NFT Diagram

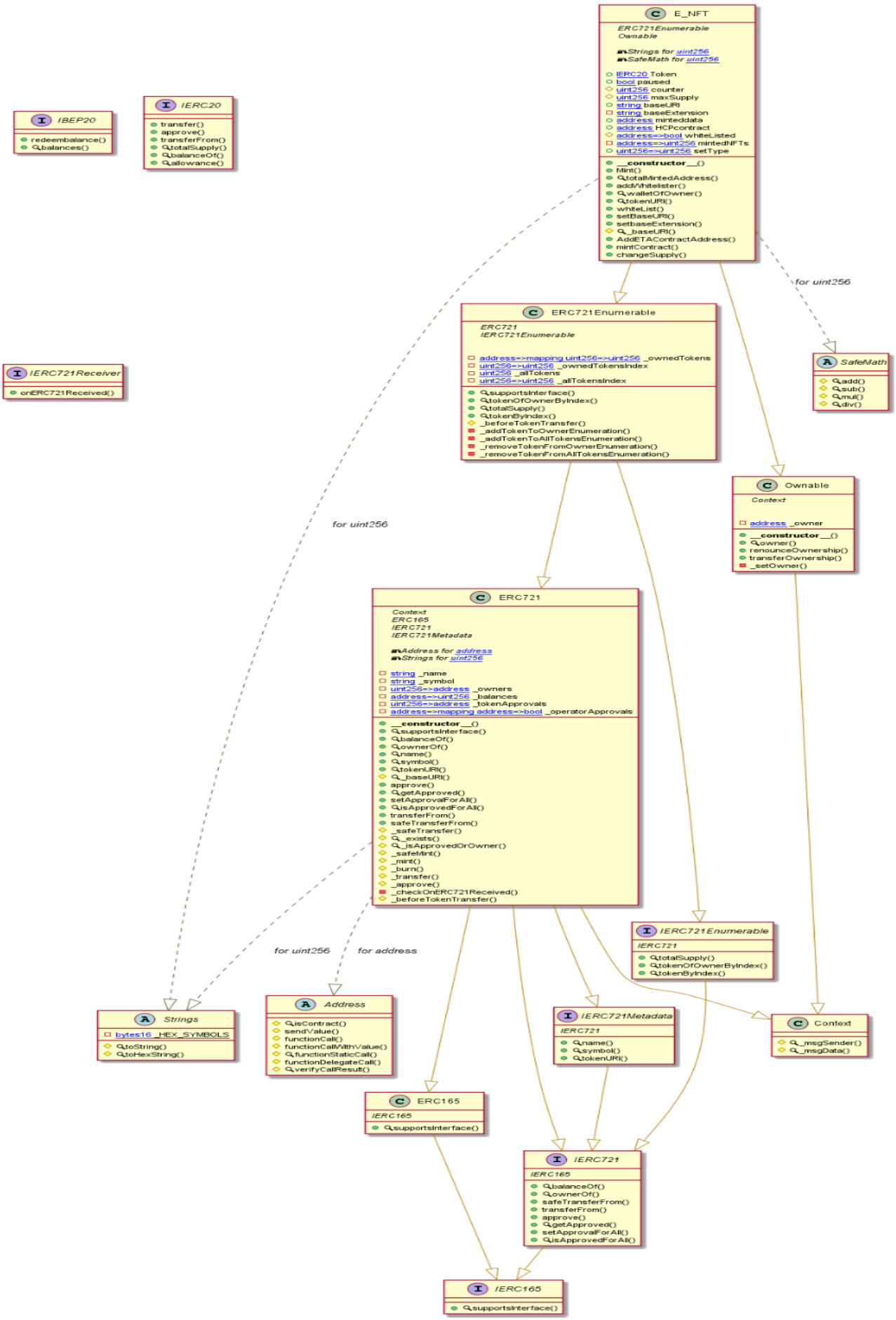


Email: audit@EtherAuthority.io

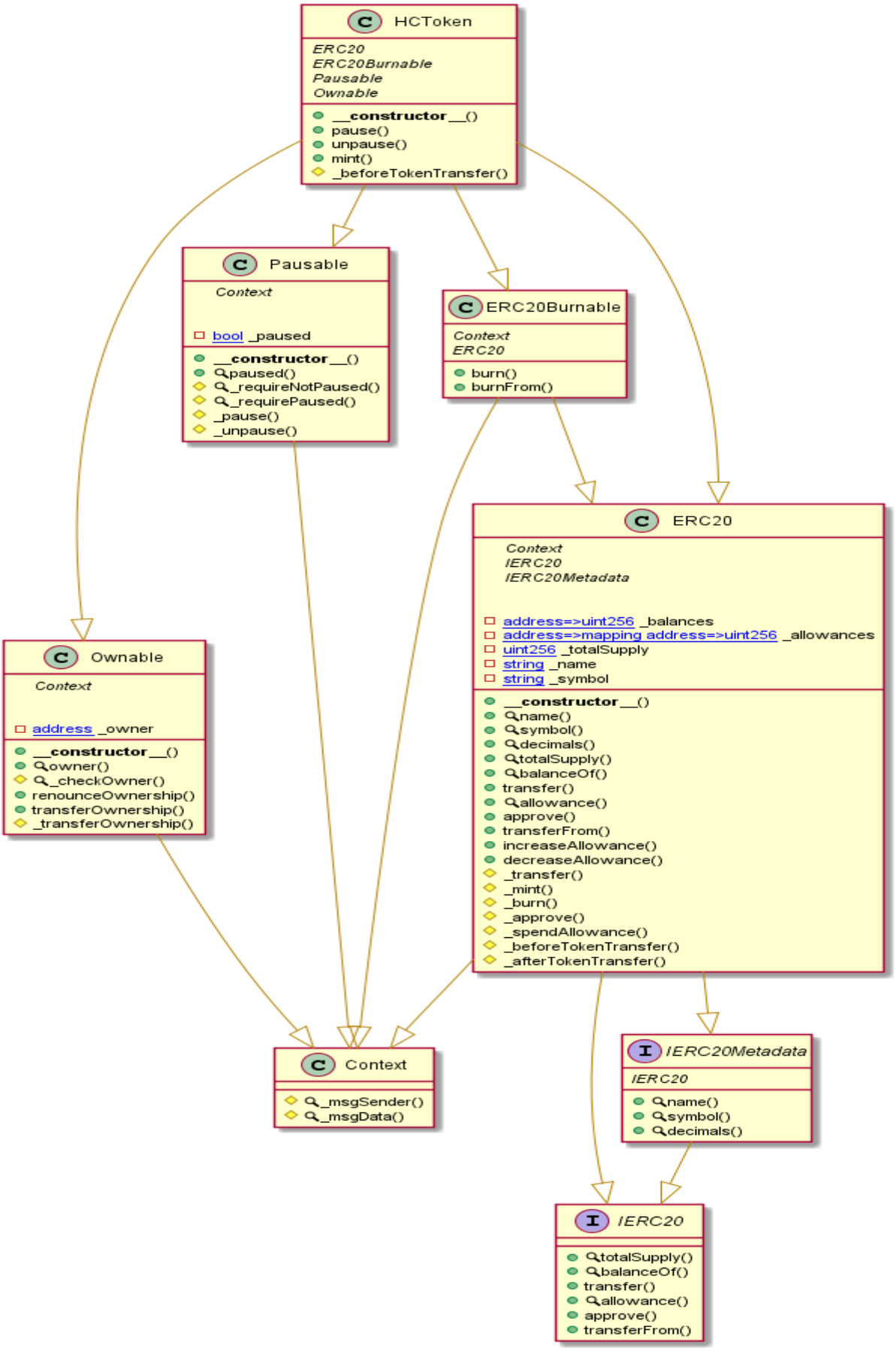
D_NFT Diagram



E_NFT Diagram

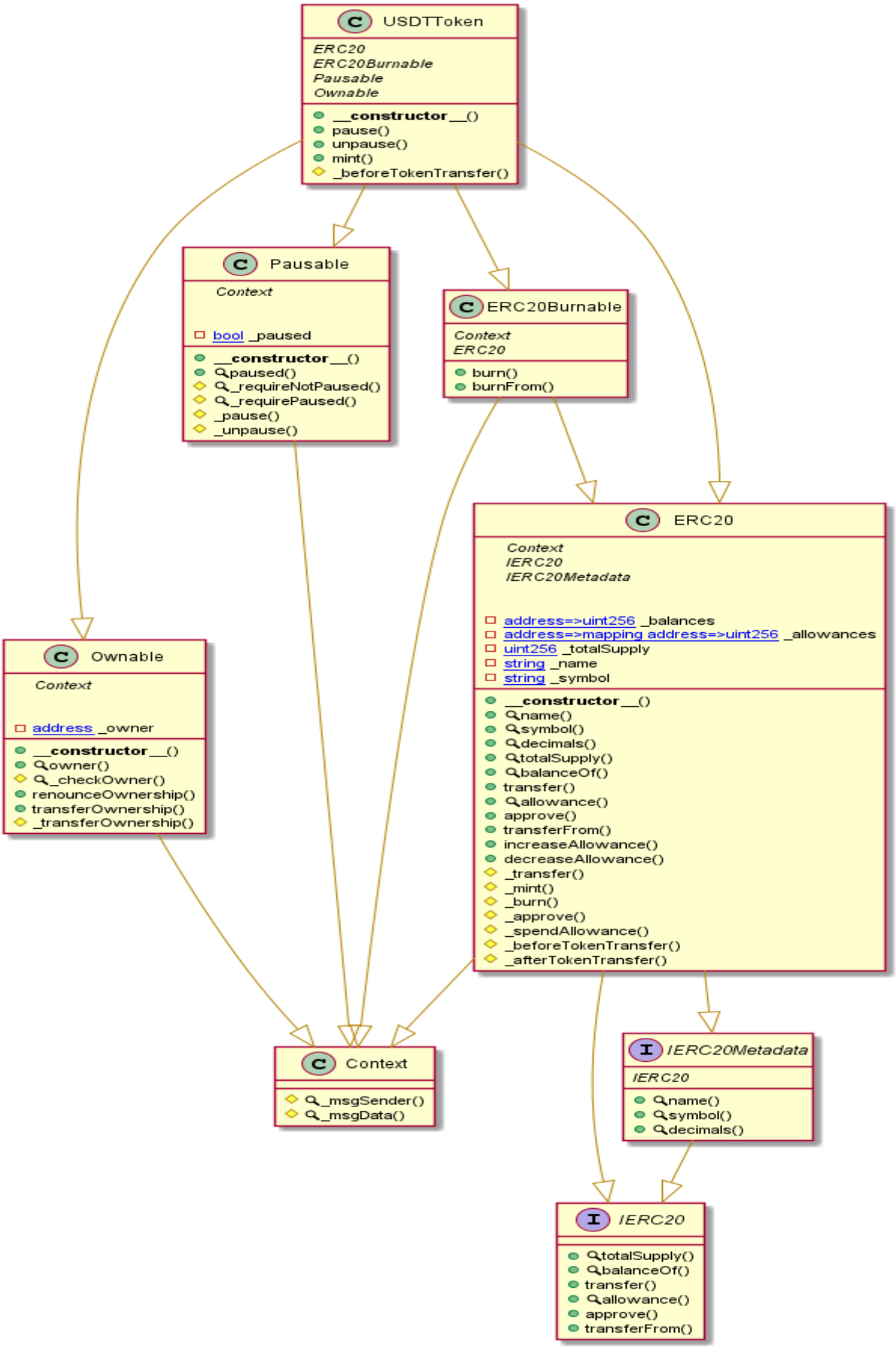


HCToken Diagram



Email: audit@EtherAuthority.io

USDTToken Diagram



Slither Results Log

Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

Slither log >> HCP.sol

```
HCP.unStakeNFT(uint256) (HCP.sol#548-562) has external calls inside a loop: interANF(NFTAddress[index]).approve(msg.sender,tot
alStakeNFT[msg.sender][index][i]) (HCP.sol#557)
HCP.unStakeNFT(uint256) (HCP.sol#548-562) has external calls inside a loop: interANF(NFTAddress[index]).transferFrom(address(t
his),msg.sender,totalStakeNFT[msg.sender][index][i]) (HCP.sol#558)
HCP.repurchaseTicket() (HCP.sol#729-752) has external calls inside a loop: USDTToken.transfer(_msgSender(),2e18) (HCP.sol#740)
HCP.repurchaseTicket() (HCP.sol#729-752) has external calls inside a loop: USDTToken.transfer(_msgSender(),4e18) (HCP.sol#742)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in HCP.deposite(address,uint256) (HCP.sol#235-266):
        External calls:
        - USDTToken.transferFrom(msg.sender,address(this),_amount) (HCP.sol#242)
        - USDTToken.transfer(OwnerAddress,per) (HCP.sol#244)
        State variables written after the call(s):
        - totalCommission[user.referrer][msg.sender][count[user.referrer][msg.sender]] = updateCommission(msg.sender,_amount)
(HCP.sol#251)
                - count[user.referrer][_user] = count[user.referrer][_user].add(1) (HCP.sol#313)
        - firstTime[msg.sender] = true (HCP.sol#249)
        - getTikcet(msg.sender,_amount) (HCP.sol#257)
                - storePreviousTicket[ticketOwnerAddress[previousTicket]][userRepurchaseCount[ticketOwnerAddress[previousTicke
t]]] = tikcetNO (HCP.sol#721)
        - getTikcet(msg.sender,_amount) (HCP.sol#257)
                - ticketOwnerAddress[tikcetNO] = _user (HCP.sol#718)
        - getTikcet(msg.sender,_amount) (HCP.sol#257)
                - tikcetNO = tikcetNO.add(1) (HCP.sol#724)
        - totalCommission[user.referrer][msg.sender][count[user.referrer][msg.sender]] = updateCommissionFirstTime(msg.sender)
 (HCP.sol#248)
        - totalCommission[user.referrer][msg.sender][count[user.referrer][msg.sender]] = updateCommission(msg.sender,_amount)
(HCP.sol#251)
        - updateFirstCommission(msg.sender,_amount) (HCP.sol#253)
                - totalCommission[msg.sender][directAddresss][count[msg.sender][directAddresss]] = total (HCP.sol#345)
        - totalCommission[user.referrer][msg.sender][count[user.referrer][msg.sender]] = updateCommissionFirstTime(msg.sender)
 (HCP.sol#248)
                - userDeposite[_user][user.referrer] = user.totalDeposit (HCP.sol#286)
                - userDeposite[user.referrer][_user] = referrUser.totalDeposit (HCP.sol#287)
        - totalCommission[user.referrer][msg.sender][count[user.referrer][msg.sender]] = updateCommission(msg.sender,_amount)
```

```
Reentrancy in HCP.deposite(address,uint256) (HCP.sol#235-266):
        External calls:
        - USDTToken.transferFrom(msg.sender,address(this),_amount) (HCP.sol#242)
        - USDTToken.transfer(OwnerAddress,per) (HCP.sol#244)
        State variables written after the call(s):
        - totalCommission[user.referrer][msg.sender][count[user.referrer][msg.sender]] = updateCommission(msg.sender,_amount)
(HCP.sol#251)
                - count[user.referrer][_user] = count[user.referrer][_user].add(1) (HCP.sol#313)
        - firstTime[msg.sender] = true (HCP.sol#249)
        - getTikcet(msg.sender,_amount) (HCP.sol#257)
                - storePreviousTicket[ticketOwnerAddress[previousTicket]][userRepurchaseCount[ticketOwnerAddress[previousTicke
t]]] = tikcetNO (HCP.sol#721)
        - getTikcet(msg.sender,_amount) (HCP.sol#257)
                - ticketOwnerAddress[tikcetNO] = _user (HCP.sol#718)
        - getTikcet(msg.sender,_amount) (HCP.sol#257)
                - tikcetNO = tikcetNO.add(1) (HCP.sol#724)
        - totalCommission[user.referrer][msg.sender][count[user.referrer][msg.sender]] = updateCommissionFirstTime(msg.sender)
 (HCP.sol#248)
        - totalCommission[user.referrer][msg.sender][count[user.referrer][msg.sender]] = updateCommission(msg.sender,_amount)
(HCP.sol#251)
        - updateFirstCommission(msg.sender,_amount) (HCP.sol#253)
                - totalCommission[msg.sender][directAddresss][count[msg.sender][directAddresss]] = total (HCP.sol#345)
        - totalCommission[user.referrer][msg.sender][count[user.referrer][msg.sender]] = updateCommissionFirstTime(msg.sender)
 (HCP.sol#248)
                - userDeposite[_user][user.referrer] = user.totalDeposit (HCP.sol#286)
                - userDeposite[user.referrer][_user] = referrUser.totalDeposit (HCP.sol#287)
        - totalCommission[user.referrer][msg.sender][count[user.referrer][msg.sender]] = updateCommission(msg.sender,_amount)
(HCP.sol#251)
                - userDeposite[_user][user.referrer] = userDeposite[_user][user.referrer].add(_amount) (HCP.sol#301)
                - userDeposite[_user][user.referrer] = 0 (HCP.sol#310)
                - userDeposite[user.referrer][_user] = 0 (HCP.sol#311)
                - userDeposite[_user][user.referrer] = userDeposite[_user][user.referrer].add(_amount) (HCP.sol#312)
        - updateFirstCommission(msg.sender,_amount) (HCP.sol#253)
                - userDeposite[_user][directAddresss] = userDeposite[_user][directAddresss].add(_amount) (HCP.sol#337)
        - getTikcet(msg.sender,_amount) (HCP.sol#257)
```

```
        - getTikcet(msg.sender,_amount) (HCP.sol#257)
                - userRepurchaseCount[ticketOwnerAddress[previousTicket]] = userRepurchaseCount[ticketOwnerAddress[previousTicket]].add(1) (HCP.sol#722)
        - updateSecondCommission(msg.sender,_amount) (HCP.sol#254)
                - userSecondCommssion[upline] = userSecondCommssion[upline].add(value) (HCP.sol#381)
        - globalShares(sharePer) (HCP.sol#256)
                - userShareCommissionE1[comAddressE1[index].userAddress] = userShareCommissionE1[comAddressE1[index].userAddress].add(sharePerUser.mul(1)) (HCP.sol#669)
        - globalShares(sharePer) (HCP.sol#256)
                - userShareCommissionE2[comAddressE2[index_scope_0].userAddress] = userShareCommissionE2[comAddressE2[index_scope_0].userAddress].add(sharePerUser.mul(3)) (HCP.sol#673)
        - globalShares(sharePer) (HCP.sol#256)
                - userShareCommissionE3[comAddressE3[index_scope_1].userAddress] = userShareCommissionE3[comAddressE3[index_scope_1].userAddress].add(sharePerUser.mul(6)) (HCP.sol#677)
        - globalShares(sharePer) (HCP.sol#256)
                - userShareCommissionE4[comAddressE4[index_scope_2].userAddress] = userShareCommissionE4[comAddressE4[index_scope_2].userAddress].add(sharePerUser.mul(10)) (HCP.sol#681)
        - globalShares(sharePer) (HCP.sol#256)
                - userShareCommissionE5[comAddressE5[index_scope_3].userAddress] = userShareCommissionE5[comAddressE5[index_scope_3].userAddress].add(sharePerUser.mul(15)) (HCP.sol#685)
Reentrancy in HCP.mintNFT(uint256) (HCP.sol#415-507):
        External calls:
        - interANFT_.mintContract(msg.sender) (HCP.sol#421)
        - interBNFT_.mintContract(msg.sender) (HCP.sol#429)
        - interCNFT_.mintContract(msg.sender) (HCP.sol#437)
        - interDNFT_.mintContract(msg.sender) (HCP.sol#445)
        - interENFT_.mintContract(msg.sender) (HCP.sol#453)
        State variables written after the call(s):
        - PushUserE1[msg.sender] = true (HCP.sol#457)
        - comAddressE1.push(structure.ComAddress(msg.sender)) (HCP.sol#454-456)
Reentrancy in HCP.mintNFT(uint256) (HCP.sol#415-507):
        External calls:
        - interANFT_.mintContract(msg.sender) (HCP.sol#421)
        - interBNFT_.mintContract(msg.sender) (HCP.sol#429)
        - interCNFT_.mintContract(msg.sender) (HCP.sol#437)
        - interDNFT_.mintContract(msg.sender) (HCP.sol#445)
        - interENFT_.mintContract(msg.sender) (HCP.sol#453)
```

```
        - interENFT_.mintContract(msg.sender) (HCP.sol#453)
        - interENFT_.mintContract(msg.sender) (HCP.sol#465)
        State variables written after the call(s):
        - PushUserE2[msg.sender] = true (HCP.sol#469)
        - comAddressE2.push(structure.ComAddress(msg.sender)) (HCP.sol#466-468)
Reentrancy in HCP.mintNFT(uint256) (HCP.sol#415-507):
        External calls:
        - interANFT_.mintContract(msg.sender) (HCP.sol#421)
        - interBNFT_.mintContract(msg.sender) (HCP.sol#429)
        - interCNFT_.mintContract(msg.sender) (HCP.sol#437)
        - interDNFT_.mintContract(msg.sender) (HCP.sol#445)
        - interENFT_.mintContract(msg.sender) (HCP.sol#453)
        - interENFT_.mintContract(msg.sender) (HCP.sol#465)
        - interENFT_.mintContract(msg.sender) (HCP.sol#477)
        State variables written after the call(s):
        - PushUserE3[msg.sender] = true (HCP.sol#481)
        - comAddressE3.push(structure.ComAddress(msg.sender)) (HCP.sol#478-480)
Reentrancy in HCP.mintNFT(uint256) (HCP.sol#415-507):
        External calls:
        - interANFT_.mintContract(msg.sender) (HCP.sol#421)
        - interBNFT_.mintContract(msg.sender) (HCP.sol#429)
        - interCNFT_.mintContract(msg.sender) (HCP.sol#437)
        - interDNFT_.mintContract(msg.sender) (HCP.sol#445)
        - interENFT_.mintContract(msg.sender) (HCP.sol#453)
        - interENFT_.mintContract(msg.sender) (HCP.sol#465)
        - interENFT_.mintContract(msg.sender) (HCP.sol#477)
        - interENFT_.mintContract(msg.sender) (HCP.sol#489)
        State variables written after the call(s):
        - PushUserE4[msg.sender] = true (HCP.sol#493)
        - comAddressE4.push(structure.ComAddress(msg.sender)) (HCP.sol#490-492)
Reentrancy in HCP.mintNFT(uint256) (HCP.sol#415-507):
        External calls:
        - interANFT_.mintContract(msg.sender) (HCP.sol#421)
        - interBNFT_.mintContract(msg.sender) (HCP.sol#429)
        - interCNFT_.mintContract(msg.sender) (HCP.sol#437)
        - interDNFT_.mintContract(msg.sender) (HCP.sol#445)
        - interENFT_.mintContract(msg.sender) (HCP.sol#453)
```

```
        - interENFT_.mintContract(msg.sender) (HCP.sol#489)
        - interENFT_.mintContract(msg.sender) (HCP.sol#501)
        State variables written after the call(s):
        - PushUserE5[msg.sender] = true (HCP.sol#505)
        - comAddressE5.push(structure.ComAddress(msg.sender)) (HCP.sol#502-504)
Reentrancy in HCP.stakeNFT(uint256,uint256) (HCP.sol#564-580):
        External calls:
        - interANF(NFTAddress[NFTNO]).transferFrom(msg.sender,address(this),NFTID) (HCP.sol#574)
        State variables written after the call(s):
        - updateUserE(msg.sender,NFTNO) (HCP.sol#579)
                - comAddressE1.push(structure.ComAddress(_user)) (HCP.sol#585-587)
        - updateUserE(msg.sender,NFTNO) (HCP.sol#579)
                - comAddressE2.push(structure.ComAddress(_user)) (HCP.sol#592-594)
        - updateUserE(msg.sender,NFTNO) (HCP.sol#579)
                - comAddressE3.push(structure.ComAddress(_user)) (HCP.sol#599-601)
        - updateUserE(msg.sender,NFTNO) (HCP.sol#579)
                - comAddressE4.push(structure.ComAddress(_user)) (HCP.sol#606-608)
        - updateUserE(msg.sender,NFTNO) (HCP.sol#579)
                - comAddressE5.push(structure.ComAddress(_user)) (HCP.sol#613-615)
        - countStakeNFT[msg.sender][NFTNO] = countStakeNFT[msg.sender][NFTNO].add(1) (HCP.sol#578)
        - totalStakeNFT[msg.sender][NFTNO][countStakeNFT[msg.sender][NFTNO]] = NFTID (HCP.sol#577)
Reentrancy in HCP.unStakeNFT(uint256) (HCP.sol#548-562):
        External calls:
        - interANF(NFTAddress[index]).approve(msg.sender,totalStakeNFT[msg.sender][index][i]) (HCP.sol#557)
        - interANF(NFTAddress[index]).transferFrom(address(this),msg.sender,totalStakeNFT[msg.sender][index][i]) (HCP.sol#558)
        State variables written after the call(s):
        - unstake_NFT[msg.sender][index] = false (HCP.sol#554)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Reentrancy in HCP.deposite(address,uint256) (HCP.sol#235-266):
        External calls:
        - USDTToken.transferFrom(msg.sender,address(this),_amount) (HCP.sol#242)
        - USDTToken.transfer(OwnerAddress,per) (HCP.sol#244)
        - HCPToken.transferFrom(msg.sender,address(this),_amount) (HCP.sol#259)
        - HCPToken.transfer(OwnerAddress,per_scope_0) (HCP.sol#261)
        Event emitted after the call(s):
        - Deposite(msg.sender,_amount) (HCP.sol#265)
Reentrancy in HCP.withdrawAmount() (HCP.sol#536-546):
        External calls:
        - USDTToken.transfer(_msgSender(),total) (HCP.sol#544)
        Event emitted after the call(s):
        - Withdraw(user,total) (HCP.sol#545)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

HCP.register(address) (HCP.sol#211-220) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(userInfo[_referral].totalDeposit > 0 || _referral == defaultRefer,invalid refer) (HCP.sol#213)
HCP._updateTeamNum(address) (HCP.sol#222-233) uses timestamp for comparisons
        Dangerous comparisons:
        - upline != address(0) (HCP.sol#226)
        - upline == defaultRefer (HCP.sol#229)
HCP.updateCommissionFirstTime(address) (HCP.sol#282-293) uses timestamp for comparisons
        Dangerous comparisons:
        - referrUser.totalDeposit > user.totalDeposit (HCP.sol#288)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

HCP.deposite(address,uint256) (HCP.sol#235-266) compares to a boolean constant:
        -firstTime[msg.sender] == false (HCP.sol#247)
HCP.getSecondCommissionPercentage(address) (HCP.sol#390-399) compares to a boolean constant:
        -NFT_Mint[_user][3] == 3 && unstake_NFT[_user][3] == true (HCP.sol#395)
HCP.getSecondCommissionPercentage(address) (HCP.sol#390-399) compares to a boolean constant:
        -NFT_Mint[_user][1] == 1 && unstake_NFT[_user][1] == true (HCP.sol#391)
HCP.getSecondCommissionPercentage(address) (HCP.sol#390-399) compares to a boolean constant:
        -NFT_Mint[_user][2] == 2 && unstake_NFT[_user][2] == true (HCP.sol#393)
HCP.getSecondCommissionPercentage(address) (HCP.sol#390-399) compares to a boolean constant:
```

```
HCP.countShareUser() (HCP.sol#622-653) compares to a boolean constant:
        -unstake_NFT[comAddressE5[index_scope_3].userAddress][9] == true (HCP.sol#648)
HCP.countShareUser() (HCP.sol#622-653) compares to a boolean constant:
        -unstake_NFT[comAddressE1[index].userAddress][5] == true (HCP.sol#629)
HCP.countShareUser() (HCP.sol#622-653) compares to a boolean constant:
        -unstake_NFT[comAddressE2[index_scope_0].userAddress][6] == true (HCP.sol#634)
HCP.countShareUser() (HCP.sol#622-653) compares to a boolean constant:
        -unstake_NFT[comAddressE3[index_scope_1].userAddress][7] == true (HCP.sol#638)
HCP.countShareUser() (HCP.sol#622-653) compares to a boolean constant:
        -unstake_NFT[comAddressE4[index_scope_2].userAddress][8] == true (HCP.sol#643)
HCP.globalShares(uint256) (HCP.sol#655-688) compares to a boolean constant:
        -unstake_NFT[comAddressE1[index].userAddress][5] == true (HCP.sol#668)
HCP.globalShares(uint256) (HCP.sol#655-688) compares to a boolean constant:
        -unstake_NFT[comAddressE2[index_scope_0].userAddress][6] == true (HCP.sol#672)
HCP.globalShares(uint256) (HCP.sol#655-688) compares to a boolean constant:
        -unstake_NFT[comAddressE3[index_scope_1].userAddress][7] == true (HCP.sol#676)
HCP.globalShares(uint256) (HCP.sol#655-688) compares to a boolean constant:
        -unstake_NFT[comAddressE4[index_scope_2].userAddress][8] == true (HCP.sol#680)
HCP.globalShares(uint256) (HCP.sol#655-688) compares to a boolean constant:
        -unstake_NFT[comAddressE5[index_scope_3].userAddress][9] == true (HCP.sol#684)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

HCP.repurchaseTicket() (HCP.sol#729-752) has costly operations inside a loop:
        - tikcetNO = tikcetNO.add(1) (HCP.sol#750)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Context._msgData() (HCP.sol#79-81) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.19 (HCP.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Contract structure (HCP.sol#7-27) is not in CapWords
Contract interANF (HCP.sol#39-44) is not in CapWords
Parameter HCP.register(address)._referral (HCP.sol#211) is not in mixedCase
Parameter HCP.deposite(address,uint256)._amount (HCP.sol#235) is not in mixedCase
Parameter HCP.updatePercenatage(uint256)._amount (HCP.sol#268) is not in mixedCase
Parameter HCP.updateCommissionFirstTime(address)._user (HCP.sol#282) is not in mixedCase
Parameter HCP.updateCommission(address,uint256)._user (HCP.sol#295) is not in mixedCase
Parameter HCP.updateCommission(address,uint256)._amount (HCP.sol#295) is not in mixedCase
Parameter HCP.timeCalculationsOfUser(address)._user (HCP.sol#324) is not in mixedCase
Parameter HCP.updateFirstCommission(address,uint256)._user (HCP.sol#330) is not in mixedCase
Parameter HCP.updateFirstCommission(address,uint256)._amount (HCP.sol#330) is not in mixedCase
Parameter HCP.totalFirstCommission(address)._user (HCP.sol#350) is not in mixedCase
```

```
Parameter HCP.getTikcet(address,uint256)._user (HCP.sol#713) is not in mixedCase
Parameter HCP.getTikcet(address,uint256)._amount (HCP.sol#713) is not in mixedCase
Parameter HCP.getcountofTicket(uint256)._Tikcet (HCP.sol#754) is not in mixedCase
Parameter HCP.getDepositeInfo(address,uint256)._user (HCP.sol#768) is not in mixedCase
Parameter HCP.getCommissionInfo(address,address)._user (HCP.sol#772) is not in mixedCase
Parameter HCP.getCommissionInfo(address,address)._directUser (HCP.sol#772) is not in mixedCase
Variable HCP.HCPToken (HCP.sol#118) is not in mixedCase
Variable HCP.USDTToken (HCP.sol#119) is not in mixedCase
Variable HCP.OwnerAddress (HCP.sol#135) is not in mixedCase
Variable HCP.NFTAddress (HCP.sol#144) is not in mixedCase
Variable HCP.unstake_NFT (HCP.sol#152) is not in mixedCase
Variable HCP.NFT_Mint (HCP.sol#159) is not in mixedCase
Variable HCP.PushUserE1 (HCP.sol#166) is not in mixedCase
Variable HCP.PushUserE2 (HCP.sol#167) is not in mixedCase
Variable HCP.PushUserE3 (HCP.sol#168) is not in mixedCase
Variable HCP.PushUserE4 (HCP.sol#169) is not in mixedCase
Variable HCP.PushUserE5 (HCP.sol#170) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
Variable HCP.userShareWithdrawE1 (HCP.sol#172) is too similar to HCP.userShareWithdrawE3 (HCP.sol#174)
Variable HCP.userShareWithdrawE1 (HCP.sol#172) is too similar to HCP.userShareWithdrawE4 (HCP.sol#175)
Variable HCP.userShareWithdrawE1 (HCP.sol#172) is too similar to HCP.userShareWithdrawE5 (HCP.sol#176)
Variable HCP.userShareWithdrawE2 (HCP.sol#173) is too similar to HCP.userShareWithdrawE3 (HCP.sol#174)
Variable HCP.userShareWithdrawE2 (HCP.sol#173) is too similar to HCP.userShareWithdrawE4 (HCP.sol#175)
Variable HCP.userShareWithdrawE2 (HCP.sol#173) is too similar to HCP.userShareWithdrawE5 (HCP.sol#176)
Variable HCP.userShareWithdrawE3 (HCP.sol#174) is too similar to HCP.userShareWithdrawE4 (HCP.sol#175)
Variable HCP.userShareWithdrawE3 (HCP.sol#174) is too similar to HCP.userShareWithdrawE5 (HCP.sol#176)
Variable HCP.userShareWithdrawE4 (HCP.sol#175) is too similar to HCP.userShareWithdrawE5 (HCP.sol#176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

HCP.updatePercenatage(uint256) (HCP.sol#268-280) uses literals with too many digits:
        - _amount >= 100000000000000000000 (HCP.sol#269)
HCP.updatePercenatage(uint256) (HCP.sol#268-280) uses literals with too many digits:
        - _amount >= 300000000000000000000 (HCP.sol#271)
HCP.updatePercenatage(uint256) (HCP.sol#268-280) uses literals with too many digits:
        - _amount >= 100000000000000000000 (HCP.sol#273)
HCP.updatePercenatage(uint256) (HCP.sol#268-280) uses literals with too many digits:
        - _amount >= 3000000000000000000 (HCP.sol#275)
HCP.mintNFT(uint256) (HCP.sol#415-507) uses literals with too many digits:
        - require(bool,string)(getTotalCommission(msg.sender) >= 10000000000000000000,Sorry commission is less than 1000K) (
HCP.sol#419)
HCP.mintNFT(uint256) (HCP.sol#415-507) uses literals with too many digits:
        - require(bool,string)(getTotalCommission(msg.sender) >= 30000000000000000000,Sorry commission is less than 3000K) (
HCP.sol#425)
HCP.mintNFT(uint256) (HCP.sol#415-507) uses literals with too many digits:
        - require(bool,string)(getTotalCommission(msg.sender) >= 100000000000000000000,Sorry commission is less than 10000K)
 (HCP.sol#433)
```

```
Variable HCP.userShareWithdrawE1 (HCP.sol#172) is too similar to HCP.userShareWithdrawE3 (HCP.sol#174)
Variable HCP.userShareWithdrawE1 (HCP.sol#172) is too similar to HCP.userShareWithdrawE4 (HCP.sol#175)
Variable HCP.userShareWithdrawE1 (HCP.sol#172) is too similar to HCP.userShareWithdrawE5 (HCP.sol#176)
Variable HCP.userShareWithdrawE2 (HCP.sol#173) is too similar to HCP.userShareWithdrawE3 (HCP.sol#174)
Variable HCP.userShareWithdrawE2 (HCP.sol#173) is too similar to HCP.userShareWithdrawE4 (HCP.sol#175)
Variable HCP.userShareWithdrawE2 (HCP.sol#173) is too similar to HCP.userShareWithdrawE5 (HCP.sol#176)
Variable HCP.userShareWithdrawE3 (HCP.sol#174) is too similar to HCP.userShareWithdrawE4 (HCP.sol#175)
Variable HCP.userShareWithdrawE3 (HCP.sol#174) is too similar to HCP.userShareWithdrawE5 (HCP.sol#176)
Variable HCP.userShareWithdrawE4 (HCP.sol#175) is too similar to HCP.userShareWithdrawE5 (HCP.sol#176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

HCP.updatePercenatage(uint256) (HCP.sol#268-280) uses literals with too many digits:
        - _amount >= 100000000000000000000 (HCP.sol#269)
HCP.updatePercenatage(uint256) (HCP.sol#268-280) uses literals with too many digits:
        - _amount >= 300000000000000000000 (HCP.sol#271)
HCP.updatePercenatage(uint256) (HCP.sol#268-280) uses literals with too many digits:
        - _amount >= 100000000000000000000 (HCP.sol#273)
HCP.updatePercenatage(uint256) (HCP.sol#268-280) uses literals with too many digits:
        - _amount >= 3000000000000000000 (HCP.sol#275)
HCP.mintNFT(uint256) (HCP.sol#415-507) uses literals with too many digits:
        - require(bool,string)(getTotalCommission(msg.sender) >= 10000000000000000000,Sorry commission is less than 1000K) (
HCP.sol#419)
HCP.mintNFT(uint256) (HCP.sol#415-507) uses literals with too many digits:
        - require(bool,string)(getTotalCommission(msg.sender) >= 30000000000000000000,Sorry commission is less than 3000K) (
HCP.sol#425)
HCP.mintNFT(uint256) (HCP.sol#415-507) uses literals with too many digits:
        - require(bool,string)(getTotalCommission(msg.sender) >= 100000000000000000000,Sorry commission is less than 10000K)
 (HCP.sol#433)
```

```
HCP.baseDivider (HCP.sol#141) is never used in HCP (HCP.sol#114-787)
HCP.getpreviousTicket (HCP.sol#146) is never used in HCP (HCP.sol#114-787)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

HCP.baseDivider (HCP.sol#141) should be constant
HCP.daySecond (HCP.sol#137) should be constant
HCP.multipler (HCP.sol#140) should be constant
HCP.ownerPercentage (HCP.sol#138) should be constant
HCP.sharePercentage (HCP.sol#139) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

HCP.HCPToken (HCP.sol#118) should be immutable
HCP.OwnerAddress (HCP.sol#135) should be immutable
HCP.USDTToken (HCP.sol#119) should be immutable
HCP.defaultRefer (HCP.sol#134) should be immutable
HCP.interANFT_ (HCP.sol#121) should be immutable
HCP.interBNFT_ (HCP.sol#122) should be immutable
HCP.interCNFT_ (HCP.sol#123) should be immutable
HCP.interDNFT_ (HCP.sol#124) should be immutable
HCP.interENFT_ (HCP.sol#125) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
HCP.sol analyzed (7 contracts with 84 detectors), 240 result(s) found
```

Slither log >> A_NFT.sol

```
A_NFT.walletOfOwner(address)._owner (A_NFT.sol#867) shadows:
        - Ownable._owner (A_NFT.sol#739) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

A_NFT.AddETAContractAddress(address) (A_NFT.sol#900-902) should emit an event for:
        - HCPcontract = HCP_Address (A_NFT.sol#901)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

A_NFT.AddETAContractAddress(address).HCP_Address (A_NFT.sol#900) lacks a zero-check on :
                - HCPcontract = HCP_Address (A_NFT.sol#901)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (A_NFT.sol#626)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (A_NFT.sol#624-640) potentially used before declaration: retval == IERC721Receiver.onERC721Re
ceived.selector (A_NFT.sol#627)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (A_NFT.sol#628)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (A_NFT.sol#624-640) potentially used before declaration: reason.length == 0 (A_NFT.sol#629)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (A_NFT.sol#628)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (A_NFT.sol#624-640) potentially used before declaration: revert(uint256,uint256)(32 + reason,
mload(uint256)(reason)) (A_NFT.sol#633)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Address.isContract(address) (A_NFT.sol#263-272) uses assembly
        - INLINE ASM (A_NFT.sol#268-270)
Address.verifyCallResult(bool,bytes,string) (A_NFT.sol#396-411) uses assembly
        - INLINE ASM (A_NFT.sol#403-406)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (A_NFT.sol#624-640) uses assembly
        - INLINE ASM (A_NFT.sol#632-634)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Address.functionCall(address,bytes) (A_NFT.sol#312-314) is never used and should be removed
Address.functionCall(address,bytes,string) (A_NFT.sol#321-323) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (A_NFT.sol#335-337) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (A_NFT.sol#344-349) is never used and should be removed
Address.functionDelegateCall(address,bytes) (A_NFT.sol#376-378) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (A_NFT.sol#385-389) is never used and should be removed
Address.functionStaticCall(address,bytes) (A_NFT.sol#356-358) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (A_NFT.sol#365-369) is never used and should be removed
Address.sendValue(address,uint256) (A_NFT.sol#289-293) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (A_NFT.sol#396-411) is never used and should be removed
Context._msgData() (A_NFT.sol#459-461) is never used and should be removed
ERC721._baseURI() (A_NFT.sol#527-529) is never used and should be removed
ERC721._burn(uint256) (A_NFT.sol#599-607) is never used and should be removed
ERC721._safeMint(address,uint256) (A_NFT.sol#577-579) is never used and should be removed
ERC721._safeMint(address,uint256,bytes) (A_NFT.sol#580-586) is never used and should be removed
SafeMath.add(uint256,uint256) (A_NFT.sol#780-784) is never used and should be removed
SafeMath.div(uint256,uint256) (A_NFT.sol#798-802) is never used and should be removed
SafeMath.mul(uint256,uint256) (A_NFT.sol#790-797) is never used and should be removed
SafeMath.sub(uint256,uint256) (A_NFT.sol#785-789) is never used and should be removed
Strings.toHexString(uint256) (A_NFT.sol#214-225) is never used and should be removed
Strings.toHexString(uint256,uint256) (A_NFT.sol#229-239) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.19 (A_NFT.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.
16
solc-0.8.19 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) (A_NFT.sol#289-293):
        - (success) = recipient.call{value: amount}() (A_NFT.sol#291)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (A_NFT.sol#344-349):
        - (success,returndata) = target.call{value: value}(data) (A_NFT.sol#347)
Low level call in Address.functionStaticCall(address,bytes,string) (A_NFT.sol#365-369):
        - (success,returndata) = target.staticcall(data) (A_NFT.sol#367)
Low level call in Address.functionDelegateCall(address,bytes,string) (A_NFT.sol#385-389):
        - (success,returndata) = target.delegatecall(data) (A_NFT.sol#387)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (A_NFT.sol#560) is not in mixedCase
Contract A_NFT (A_NFT.sol#806-916) is not in CapWords
Event A_NFTpreMaleMinting(address,address,uint256) (A_NFT.sol#833) is not in CapWords
Event A_NFTpreFemaleMinting(address,address,uint256) (A_NFT.sol#834) is not in CapWords
Event A_NFTpublicMaleMinting(address,address,uint256) (A_NFT.sol#835) is not in CapWords
Event A_NFTpublicFemaleMinting(address,address,uint256) (A_NFT.sol#836) is not in CapWords
Event A_NFTpublicEliteMinting(address,address,uint256) (A_NFT.sol#837) is not in CapWords
Function A_NFT.Mint() (A_NFT.sol#850-855) is not in mixedCase
Parameter A_NFT.addWhitelister(address[])._users (A_NFT.sol#861) is not in mixedCase
Parameter A_NFT.walletOfOwner(address)._owner (A_NFT.sol#867) is not in mixedCase
Parameter A_NFT.tokenURI(uint256)._tokenId (A_NFT.sol#878) is not in mixedCase
Parameter A_NFT.whiteList(address[])._whiteListers (A_NFT.sol#882) is not in mixedCase
Parameter A_NFT.setBaseURI(string)._newBaseURI (A_NFT.sol#888) is not in mixedCase
Parameter A_NFT.setbaseExtension(string)._newbaseExtension (A_NFT.sol#892) is not in mixedCase
Function A_NFT.AddETAContractAddress(address) (A_NFT.sol#900-902) is not in mixedCase
Parameter A_NFT.AddETAContractAddress(address).HCP_Address (A_NFT.sol#900) is not in mixedCase
Parameter A_NFT.changeSupply(uint256)._supply (A_NFT.sol#911) is not in mixedCase
Variable A_NFT.Token (A_NFT.sol#808) is not in mixedCase
Variable A_NFT.HCPcontract (A_NFT.sol#821) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

A_NFT.mintedNFTs (A_NFT.sol#824) is never used in A_NFT (A_NFT.sol#806-916)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

A_NFT.Token (A_NFT.sol#808) should be constant
A_NFT.paused (A_NFT.sol#812) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
A_NFT.sol analyzed (16 contracts with 84 detectors), 62 result(s) found
```

Slither log >> B_NFT.sol

```
B_NFT.walletOfOwner(address)._owner (B_NFT.sol#868) shadows:
        - Ownable._owner (B_NFT.sol#740) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

B_NFT.AddETAContractAddress(address) (B_NFT.sol#901-903) should emit an event for:
        - HCPcontract = HCP_Address (B_NFT.sol#902)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

B_NFT.AddETAContractAddress(address).HCP_Address (B_NFT.sol#901) lacks a zero-check on :
                - HCPcontract = HCP_Address (B_NFT.sol#902)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (B_NFT.sol#627)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (B_NFT.sol#625-641) potentially used before declaration: retval == IERC721Receiver.onERC721Re
ceived.selector (B_NFT.sol#628)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (B_NFT.sol#629)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (B_NFT.sol#625-641) potentially used before declaration: reason.length == 0 (B_NFT.sol#630)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (B_NFT.sol#629)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (B_NFT.sol#625-641) potentially used before declaration: revert(uint256,uint256)(32 + reason,
mload(uint256)(reason)) (B_NFT.sol#634)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Address.isContract(address) (B_NFT.sol#264-273) uses assembly
        - INLINE ASM (B_NFT.sol#269-271)
Address.verifyCallResult(bool,bytes,string) (B_NFT.sol#397-412) uses assembly
        - INLINE ASM (B_NFT.sol#404-407)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (B_NFT.sol#625-641) uses assembly
        - INLINE ASM (B_NFT.sol#633-635)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version^0.8.19 (B_NFT.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.
16
solc-0.8.19 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) (B_NFT.sol#290-294):
        - (success) = recipient.call{value: amount}() (B_NFT.sol#292)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (B_NFT.sol#345-350):
        - (success,returndata) = target.call{value: value}(data) (B_NFT.sol#348)
Low level call in Address.functionStaticCall(address,bytes,string) (B_NFT.sol#366-370):
        - (success,returndata) = target.staticcall(data) (B_NFT.sol#368)
Low level call in Address.functionDelegateCall(address,bytes,string) (B_NFT.sol#386-390):
        - (success,returndata) = target.delegatecall(data) (B_NFT.sol#388)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (B_NFT.sol#561) is not in mixedCase
Contract B_NFT (B_NFT.sol#807-917) is not in CapWords
Event B_NFTpreMaleMinting(address,address,uint256) (B_NFT.sol#834) is not in CapWords
Event B_NFTpreFemaleMinting(address,address,uint256) (B_NFT.sol#835) is not in CapWords
Event B_NFTpublicMaleMinting(address,address,uint256) (B_NFT.sol#836) is not in CapWords
Event B_NFTpublicFemaleMinting(address,address,uint256) (B_NFT.sol#837) is not in CapWords
Event B_NFTpublicEliteMinting(address,address,uint256) (B_NFT.sol#838) is not in CapWords
Function B_NFT.Mint() (B_NFT.sol#851-856) is not in mixedCase
Parameter B_NFT.addWhitelister(address[])._users (B_NFT.sol#862) is not in mixedCase
Parameter B_NFT.walletOfOwner(address)._owner (B_NFT.sol#868) is not in mixedCase
Parameter B_NFT.tokenURI(uint256)._tokenId (B_NFT.sol#879) is not in mixedCase
Parameter B_NFT.whiteList(address[])._whiteListers (B_NFT.sol#883) is not in mixedCase
Parameter B_NFT.setBaseURI(string)._newBaseURI (B_NFT.sol#889) is not in mixedCase
Parameter B_NFT.setbaseExtension(string)._newbaseExtension (B_NFT.sol#893) is not in mixedCase
Function B_NFT.AddETAContractAddress(address) (B_NFT.sol#901-903) is not in mixedCase
Parameter B_NFT.AddETAContractAddress(address).HCP_Address (B_NFT.sol#901) is not in mixedCase
Parameter B_NFT.changeSupply(uint256)._supply (B_NFT.sol#912) is not in mixedCase
Variable B_NFT.Token (B_NFT.sol#809) is not in mixedCase
```

```
Variable B_NFT.HCPcontract (B_NFT.sol#822) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

B_NFT.mintedNFTs (B_NFT.sol#825) is never used in B_NFT (B_NFT.sol#807-917)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

B_NFT.Token (B_NFT.sol#809) should be constant
B_NFT.paused (B_NFT.sol#813) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
B_NFT.sol analyzed (16 contracts with 84 detectors), 62 result(s) found
```

Slither log >> C_NFT.sol

```
C_NFT.walletOfOwner(address)._owner (C_NFT.sol#869) shadows:
        - Ownable._owner (C_NFT.sol#741) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

C_NFT.AddETAContractAddress(address) (C_NFT.sol#902-904) should emit an event for:
        - HCPcontract = HCP_Address (C_NFT.sol#903)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

C_NFT.AddETAContractAddress(address).HCP_Address (C_NFT.sol#902) lacks a zero-check on :
                - HCPcontract = HCP_Address (C_NFT.sol#903)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (C_NFT.sol#628)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (C_NFT.sol#626-642) potentially used before declaration: retval == IERC721Receiver.onERC721Re
ceived.selector (C_NFT.sol#629)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (C_NFT.sol#630)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (C_NFT.sol#626-642) potentially used before declaration: reason.length == 0 (C_NFT.sol#631)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (C_NFT.sol#630)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (C_NFT.sol#626-642) potentially used before declaration: revert(uint256,uint256)(32 + reason,
mload(uint256)(reason)) (C_NFT.sol#635)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Address.isContract(address) (C_NFT.sol#265-274) uses assembly
        - INLINE ASM (C_NFT.sol#270-272)
Address.verifyCallResult(bool,bytes,string) (C_NFT.sol#398-413) uses assembly
        - INLINE ASM (C_NFT.sol#405-408)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (C_NFT.sol#626-642) uses assembly
        - INLINE ASM (C_NFT.sol#634-636)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Address.functionCall(address,bytes) (C_NFT.sol#314-316) is never used and should be removed
Address.functionCall(address,bytes,string) (C_NFT.sol#323-325) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (C_NFT.sol#337-339) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (C_NFT.sol#346-351) is never used and should be removed
Address.functionDelegateCall(address,bytes) (C_NFT.sol#378-380) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (C_NFT.sol#387-391) is never used and should be removed
Address.functionStaticCall(address,bytes) (C_NFT.sol#358-360) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (C_NFT.sol#367-371) is never used and should be removed
Address.sendValue(address,uint256) (C_NFT.sol#291-295) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (C_NFT.sol#398-413) is never used and should be removed
Context._msgData() (C_NFT.sol#461-463) is never used and should be removed
ERC721._baseURI() (C_NFT.sol#529-531) is never used and should be removed
ERC721._burn(uint256) (C_NFT.sol#601-609) is never used and should be removed
ERC721._safeMint(address,uint256) (C_NFT.sol#579-581) is never used and should be removed
ERC721._safeMint(address,uint256,bytes) (C_NFT.sol#582-588) is never used and should be removed
SafeMath.add(uint256,uint256) (C_NFT.sol#782-786) is never used and should be removed
SafeMath.div(uint256,uint256) (C_NFT.sol#800-804) is never used and should be removed
SafeMath.mul(uint256,uint256) (C_NFT.sol#792-799) is never used and should be removed
SafeMath.sub(uint256,uint256) (C_NFT.sol#787-791) is never used and should be removed
Strings.toHexString(uint256) (C_NFT.sol#216-227) is never used and should be removed
Strings.toHexString(uint256,uint256) (C_NFT.sol#231-241) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.19 (C_NFT.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.
16
solc-0.8.19 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) (C_NFT.sol#291-295):
        - (success) = recipient.call{value: amount}() (C_NFT.sol#293)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (C_NFT.sol#346-351):
        - (success,returndata) = target.call{value: value}(data) (C_NFT.sol#349)
Low level call in Address.functionStaticCall(address,bytes,string) (C_NFT.sol#367-371):
        - (success,returndata) = target.staticcall(data) (C_NFT.sol#369)
Low level call in Address.functionDelegateCall(address,bytes,string) (C_NFT.sol#387-391):
        - (success,returndata) = target.delegatecall(data) (C_NFT.sol#389)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (C_NFT.sol#562) is not in mixedCase
Contract C_NFT (C_NFT.sol#808-918) is not in CapWords
Event C_NFTpreMaleMinting(address,address,uint256) (C_NFT.sol#835) is not in CapWords
Event C_NFTpreFemaleMinting(address,address,uint256) (C_NFT.sol#836) is not in CapWords
Event C_NFTpublicMaleMinting(address,address,uint256) (C_NFT.sol#837) is not in CapWords
Event C_NFTpublicFemaleMinting(address,address,uint256) (C_NFT.sol#838) is not in CapWords
Event C_NFTpublicEliteMinting(address,address,uint256) (C_NFT.sol#839) is not in CapWords
Function C_NFT.Mint() (C_NFT.sol#852-857) is not in mixedCase
Parameter C_NFT.addWhitelister(address[])._users (C_NFT.sol#863) is not in mixedCase
Parameter C_NFT.walletOfOwner(address)._owner (C_NFT.sol#869) is not in mixedCase
Parameter C_NFT.tokenURI(uint256)._tokenId (C_NFT.sol#880) is not in mixedCase
Parameter C_NFT.whiteList(address[])._whiteListers (C_NFT.sol#884) is not in mixedCase
Parameter C_NFT.setBaseURI(string)._newBaseURI (C_NFT.sol#890) is not in mixedCase
Parameter C_NFT.setbaseExtension(string)._newbaseExtension (C_NFT.sol#894) is not in mixedCase
Function C_NFT.AddETAContractAddress(address) (C_NFT.sol#902-904) is not in mixedCase
Parameter C_NFT.AddETAContractAddress(address).HCP_Address (C_NFT.sol#902) is not in mixedCase
Parameter C_NFT.changeSupply(uint256)._supply (C_NFT.sol#913) is not in mixedCase
Variable C_NFT.Token (C_NFT.sol#810) is not in mixedCase
Variable C_NFT.HCPcontract (C_NFT.sol#823) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
C_NFT.mintedNFTs (C_NFT.sol#826) is never used in C_NFT (C_NFT.sol#808-918)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

C_NFT.Token (C_NFT.sol#810) should be constant
C_NFT.paused (C_NFT.sol#814) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
C_NFT.sol analyzed (16 contracts with 84 detectors), 62 result(s) found
```

Slither log >> D_NFT.sol

```
D_NFT.walletOfOwner(address)._owner (D_NFT.sol#869) shadows:
        - Ownable._owner (D_NFT.sol#741) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

D_NFT.AddETAContractAddress(address) (D_NFT.sol#902-904) should emit an event for:
        - HCPcontract = HCP_Address (D_NFT.sol#903)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

D_NFT.AddETAContractAddress(address).HCP_Address (D_NFT.sol#902) lacks a zero-check on :
                - HCPcontract = HCP_Address (D_NFT.sol#903)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (D_NFT.sol#628)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (D_NFT.sol#626-642) potentially used before declaration: retval == IERC721Receiver.onERC721Re
ceived.selector (D_NFT.sol#629)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (D_NFT.sol#630)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (D_NFT.sol#626-642) potentially used before declaration: reason.length == 0 (D_NFT.sol#631)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (D_NFT.sol#630)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (D_NFT.sol#626-642) potentially used before declaration: revert(uint256,uint256)(32 + reason,
mload(uint256)(reason)) (D_NFT.sol#635)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Address.isContract(address) (D_NFT.sol#265-274) uses assembly
        - INLINE ASM (D_NFT.sol#270-272)
Address.verifyCallResult(bool,bytes,string) (D_NFT.sol#398-413) uses assembly
        - INLINE ASM (D_NFT.sol#405-408)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (D_NFT.sol#626-642) uses assembly
        - INLINE ASM (D_NFT.sol#634-636)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Address.functionCall(address,bytes) (D_NFT.sol#314-316) is never used and should be removed
Address.functionCall(address,bytes,string) (D_NFT.sol#323-325) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (D_NFT.sol#337-339) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (D_NFT.sol#346-351) is never used and should be removed
Address.functionDelegateCall(address,bytes) (D_NFT.sol#378-380) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (D_NFT.sol#387-391) is never used and should be removed
Address.functionStaticCall(address,bytes) (D_NFT.sol#358-360) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (D_NFT.sol#367-371) is never used and should be removed
Address.sendValue(address,uint256) (D_NFT.sol#291-295) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (D_NFT.sol#398-413) is never used and should be removed
Context._msgData() (D_NFT.sol#461-463) is never used and should be removed
ERC721._baseURI() (D_NFT.sol#529-531) is never used and should be removed
ERC721._burn(uint256) (D_NFT.sol#601-609) is never used and should be removed
ERC721._safeMint(address,uint256) (D_NFT.sol#579-581) is never used and should be removed
ERC721._safeMint(address,uint256,bytes) (D_NFT.sol#582-588) is never used and should be removed
SafeMath.add(uint256,uint256) (D_NFT.sol#782-786) is never used and should be removed
SafeMath.div(uint256,uint256) (D_NFT.sol#800-804) is never used and should be removed
SafeMath.mul(uint256,uint256) (D_NFT.sol#792-799) is never used and should be removed
SafeMath.sub(uint256,uint256) (D_NFT.sol#787-791) is never used and should be removed
Strings.toHexString(uint256) (D_NFT.sol#216-227) is never used and should be removed
Strings.toHexString(uint256,uint256) (D_NFT.sol#231-241) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.19 (D_NFT.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.
16
solc-0.8.19 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) (D_NFT.sol#291-295):
        - (success) = recipient.call{value: amount}() (D_NFT.sol#293)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (D_NFT.sol#346-351):
        - (success,returndata) = target.call{value: value}(data) (D_NFT.sol#349)
Low level call in Address.functionStaticCall(address,bytes,string) (D_NFT.sol#367-371):
        - (success,returndata) = target.staticcall(data) (D_NFT.sol#369)
Low level call in Address.functionDelegateCall(address,bytes,string) (D_NFT.sol#387-391):
        - (success,returndata) = target.delegatecall(data) (D_NFT.sol#389)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (D_NFT.sol#562) is not in mixedCase
Contract D_NFT (D_NFT.sol#808-918) is not in CapWords
Event D_NFTpreMaleMinting(address,address,uint256) (D_NFT.sol#835) is not in CapWords
Event D_NFTpreFemaleMinting(address,address,uint256) (D_NFT.sol#836) is not in CapWords
Event D_NFTpublicMaleMinting(address,address,uint256) (D_NFT.sol#837) is not in CapWords
Event D_NFTpublicFemaleMinting(address,address,uint256) (D_NFT.sol#838) is not in CapWords
Event D_NFTpublicEliteMinting(address,address,uint256) (D_NFT.sol#839) is not in CapWords
Function D_NFT.Mint() (D_NFT.sol#852-857) is not in mixedCase
Parameter D_NFT.addWhitelister(address[])._users (D_NFT.sol#863) is not in mixedCase
Parameter D_NFT.walletOfOwner(address)._owner (D_NFT.sol#869) is not in mixedCase
Parameter D_NFT.tokenURI(uint256)._tokenId (D_NFT.sol#880) is not in mixedCase
Parameter D_NFT.whiteList(address[])._whiteListers (D_NFT.sol#884) is not in mixedCase
Parameter D_NFT.setBaseURI(string)._newBaseURI (D_NFT.sol#890) is not in mixedCase
Parameter D_NFT.setbaseExtension(string)._newbaseExtension (D_NFT.sol#894) is not in mixedCase
Function D_NFT.AddETAContractAddress(address) (D_NFT.sol#902-904) is not in mixedCase
Parameter D_NFT.AddETAContractAddress(address).HCP_Address (D_NFT.sol#902) is not in mixedCase
Parameter D_NFT.changeSupply(uint256)._supply (D_NFT.sol#913) is not in mixedCase
Variable D_NFT.Token (D_NFT.sol#810) is not in mixedCase
Variable D_NFT.HCPcontract (D_NFT.sol#823) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
D_NFT.mintedNFTs (D_NFT.sol#826) is never used in D_NFT (D_NFT.sol#808-918)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

D_NFT.Token (D_NFT.sol#810) should be constant
D_NFT.paused (D_NFT.sol#814) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
D_NFT.sol analyzed (16 contracts with 84 detectors), 62 result(s) found
```

Slither log >> E_NFT.sol

```
E_NFT.walletOfOwner(address)._owner (E_NFT.sol#869) shadows:
        - Ownable._owner (E_NFT.sol#741) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

E_NFT.AddETAContractAddress(address) (E_NFT.sol#902-904) should emit an event for:
        - HCPcontract = HCP_Address (E_NFT.sol#903)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

E_NFT.AddETAContractAddress(address).HCP_Address (E_NFT.sol#902) lacks a zero-check on :
                - HCPcontract = HCP_Address (E_NFT.sol#903)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (E_NFT.sol#628)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (E_NFT.sol#626-642) potentially used before declaration: retval == IERC721Receiver.onERC721Re
ceived.selector (E_NFT.sol#629)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (E_NFT.sol#630)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (E_NFT.sol#626-642) potentially used before declaration: reason.length == 0 (E_NFT.sol#631)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (E_NFT.sol#630)' in ERC721._checkOnERC721Receive
d(address,address,uint256,bytes) (E_NFT.sol#626-642) potentially used before declaration: revert(uint256,uint256)(32 + reason,
mload(uint256)(reason)) (E_NFT.sol#635)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Address.isContract(address) (E_NFT.sol#265-274) uses assembly
        - INLINE ASM (E_NFT.sol#270-272)
Address.verifyCallResult(bool,bytes,string) (E_NFT.sol#398-413) uses assembly
        - INLINE ASM (E_NFT.sol#405-408)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (E_NFT.sol#626-642) uses assembly
        - INLINE ASM (E_NFT.sol#634-636)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Address.functionCall(address,bytes) (E_NFT.sol#314-316) is never used and should be removed
Address.functionCall(address,bytes,string) (E_NFT.sol#323-325) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (E_NFT.sol#337-339) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (E_NFT.sol#346-351) is never used and should be removed
Address.functionDelegateCall(address,bytes) (E_NFT.sol#378-380) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (E_NFT.sol#387-391) is never used and should be removed
Address.functionStaticCall(address,bytes) (E_NFT.sol#358-360) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (E_NFT.sol#367-371) is never used and should be removed
Address.sendValue(address,uint256) (E_NFT.sol#291-295) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (E_NFT.sol#398-413) is never used and should be removed
Context._msgData() (E_NFT.sol#461-463) is never used and should be removed
ERC721._baseURI() (E_NFT.sol#529-531) is never used and should be removed
ERC721._burn(uint256) (E_NFT.sol#601-609) is never used and should be removed
ERC721._safeMint(address,uint256) (E_NFT.sol#579-581) is never used and should be removed
ERC721._safeMint(address,uint256,bytes) (E_NFT.sol#582-588) is never used and should be removed
SafeMath.add(uint256,uint256) (E_NFT.sol#782-786) is never used and should be removed
SafeMath.div(uint256,uint256) (E_NFT.sol#800-804) is never used and should be removed
SafeMath.mul(uint256,uint256) (E_NFT.sol#792-799) is never used and should be removed
SafeMath.sub(uint256,uint256) (E_NFT.sol#787-791) is never used and should be removed
Strings.toHexString(uint256) (E_NFT.sol#216-227) is never used and should be removed
Strings.toHexString(uint256,uint256) (E_NFT.sol#231-241) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.19 (E_NFT.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.
16
solc-0.8.19 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) (E_NFT.sol#291-295):
        - (success) = recipient.call{value: amount}() (E_NFT.sol#293)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (E_NFT.sol#346-351):
        - (success,returndata) = target.call{value: value}(data) (E_NFT.sol#349)
Low level call in Address.functionStaticCall(address,bytes,string) (E_NFT.sol#367-371):
        - (success,returndata) = target.staticcall(data) (E_NFT.sol#369)
Low level call in Address.functionDelegateCall(address,bytes,string) (E_NFT.sol#387-391):
        - (success,returndata) = target.delegatecall(data) (E_NFT.sol#389)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (E_NFT.sol#562) is not in mixedCase
Contract E_NFT (E_NFT.sol#808-918) is not in CapWords
Event E_NFTpreMaleMinting(address,address,uint256) (E_NFT.sol#835) is not in CapWords
Event E_NFTpreFemaleMinting(address,address,uint256) (E_NFT.sol#836) is not in CapWords
Event E_NFTpublicMaleMinting(address,address,uint256) (E_NFT.sol#837) is not in CapWords
Event E_NFTpublicFemaleMinting(address,address,uint256) (E_NFT.sol#838) is not in CapWords
Event E_NFTpublicEliteMinting(address,address,uint256) (E_NFT.sol#839) is not in CapWords
Function E_NFT.Mint() (E_NFT.sol#852-857) is not in mixedCase
Parameter E_NFT.addWhitelister(address[])._users (E_NFT.sol#863) is not in mixedCase
Parameter E_NFT.walletOfOwner(address)._owner (E_NFT.sol#869) is not in mixedCase
Parameter E_NFT.tokenURI(uint256)._tokenId (E_NFT.sol#880) is not in mixedCase
Parameter E_NFT.whiteList(address[])._whiteListers (E_NFT.sol#884) is not in mixedCase
Parameter E_NFT.setBaseURI(string)._newBaseURI (E_NFT.sol#890) is not in mixedCase
Parameter E_NFT.setbaseExtension(string)._newbaseExtension (E_NFT.sol#894) is not in mixedCase
Function E_NFT.AddETAContractAddress(address) (E_NFT.sol#902-904) is not in mixedCase
Parameter E_NFT.AddETAContractAddress(address).HCP_Address (E_NFT.sol#902) is not in mixedCase
Parameter E_NFT.changeSupply(uint256)._supply (E_NFT.sol#913) is not in mixedCase
Variable E_NFT.Token (E_NFT.sol#810) is not in mixedCase
Variable E_NFT.HCPcontract (E_NFT.sol#823) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
E_NFT.mintedNFTs (E_NFT.sol#826) is never used in E_NFT (E_NFT.sol#808-918)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

E_NFT.Token (E_NFT.sol#810) should be constant
E_NFT.paused (E_NFT.sol#814) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
E_NFT.sol analyzed (16 contracts with 84 detectors), 62 result(s) found
```

Slither log >> HCToken.sol

```
Context._msgData() (HCToken.sol#9-11) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.19 (HCToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.
8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

HCToken.constructor() (HCToken.sol#444-446) uses literals with too many digits:
        - _mint(msg.sender,5000000000 * (10 ** 18)) (HCToken.sol#445)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
HCToken.sol analyzed (8 contracts with 84 detectors), 4 result(s) found
```

Slither log >> USDTToken.sol

```
Context._msgData() (USDTToken.sol#9-11) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.19 (USDTToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/
0.8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

USDTToken.constructor() (USDTToken.sol#444-446) uses literals with too many digits:
        - _mint(msg.sender,5000000000 * (10 ** 18)) (USDTToken.sol#445)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
USDTToken.sol analyzed (8 contracts with 84 detectors), 4 result(s) found
```

Solidity Static Analysis

HCP.sol

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.

more

Pos: 565:30:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in HCP.repurchaseTicket(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 729:4:

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: 326:25:

Gas costs:

Gas requirement of function HCP.totalFirstCommission 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: 350:4:

Gas costs:

Gas requirement of function HCP.checkPreviousNFT 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: 401:4:

Email: audit@EtherAuthority.io

Gas costs:

Gas requirement of function HCP.unStakeNFT 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: 548:4:

Gas costs:

Gas requirement of function HCP.stakeNFT 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: 564:4:

Gas costs:

Gas requirement of function HCP.stakeNFT 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: 564:4:

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: 647:8:

Constant/View/Pure functions:

HCP.updateUserE(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 582:4:

Similar variable names:

HCP.() : Variables have very similar names "interANFT_" and "interCNFT_". Note: Modifiers are currently not considered by this static analysis.
Pos: 205:8:

Similar variable names:

HCP.deposite(address,uint256) : Variables have very similar names "user" and "per". Note: Modifiers are currently not considered by this static analysis.
Pos: 245:12:

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: 731:8:

Result not used:

A binary operation yields a value that is not used further. This is often caused by confusing assignment (=) and comparison (==).
Pos: 576:8:

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: 56:20:

A_NFT.sol

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: 344:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 632:20:

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: 387:50:

Gas costs:

Gas requirement of function A_NFT.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: 867:4:

Gas costs:

Gas requirement of function A_NFT.whiteList 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: 882:4:

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: 862:8:

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: 883:8:

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: 851:8:

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: 345:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 633:20:

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: 388:50:

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: 863:8:

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: 884:8:

Constant/View/Pure functions:

B_NFT.walletOfOwner(address) : Is constant but potentially should not be.
Note: Modifiers are currently not considered by this static analysis.

more

Pos: 868:4:

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: 906:8:

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: 606:8:

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: 801:20:

C_NFT.sol

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: 346:4:

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: 346:4:

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: 864:8:

Constant/View/Pure functions:

C_NFT.walletOfOwner(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 869:4:

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: 759:8:

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: 736:8:

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: 802:20:

D_NFT.sol

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: 346:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 634:20:

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: 389:50:

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: 885:8:

Constant/View/Pure functions:

D_NFT.walletOfOwner(address) : Is constant but potentially should not be.
Note: Modifiers are currently not considered by this static analysis.

more

Pos: 869:4:

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: 853:8:

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: 736:8:

E_NFT.sol

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: 346:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 634:20:

Gas costs:

Gas requirement of function E_NFT.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: 869:4:

Gas costs:

Gas requirement of function E_NFT.addWhitelister 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: 863:4:

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: 885:8:

Constant/View/Pure functions:

E_NFT.walletOfOwner(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 869:4:

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: 797:8:

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: 736:8:

Gas costs:

Gas requirement of function HCToken.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: 456:4:

Gas costs:

Gas requirement of function HCToken.burn 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: 433:4:

No return:

IERC20.approve(address,uint256): Defines a return type but never explicitly returns a value.
Pos: 223:4:

Constant/View/Pure functions:

HCToken._beforeTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 460:4:

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: 411:12:

Gas costs:

Gas requirement of function USDTToken.burn 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: 433:4:

No return:

IERC20Metadata.decimals(): Defines a return type but never explicitly returns a value.
Pos: 254:4:

Similar variable names:

ERC20._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 384:28:

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: 411:12:

Solhint Linter

HCP.sol

```
Compiler version ^0.8.19 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:5
Contract name must be in CamelCase
Pos: 1:6
Contract name must be in CamelCase
Pos: 1:38
Provide an error message for require
Pos: 9:50
Provide an error message for require
Pos: 9:54
Provide an error message for require
Pos: 9:59
Provide an error message for require
Pos: 9:65
Provide an error message for require
Pos: 9:69
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:86
Error message for require is too long
Pos: 9:103
Contract has 57 states declarations but allowed no more than 15
Pos: 1:113
Variable name must be in mixedCase
Pos: 5:117
Variable name must be in mixedCase
Pos: 5:118
Variable name must be in mixedCase
Pos: 5:134
Variable name must be in mixedCase
Pos: 5:143
Variable name must be in mixedCase
Pos: 5:151
Variable name must be in mixedCase
Pos: 5:158
Variable name must be in mixedCase
Pos: 5:165
Variable name must be in mixedCase
Pos: 5:166
Variable name must be in mixedCase
Pos: 5:167
Variable name must be in mixedCase
Pos: 5:168
Variable name must be in mixedCase
Pos: 5:169
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:187
Avoid to use tx.origin
```

```
Pos: 31:211
Avoid to use tx.origin
Pos: 31:235
Possible reentrancy vulnerabilities. Avoid state changes after
transfer.
Pos: 17:247
Possible reentrancy vulnerabilities. Avoid state changes after
transfer.
Pos: 17:248
Possible reentrancy vulnerabilities. Avoid state changes after
transfer.
Pos: 17:250
Avoid making time-based decisions in your business logic
Pos: 21:284
Avoid making time-based decisions in your business logic
Pos: 21:298
Avoid making time-based decisions in your business logic
Pos: 26:325
Variable name must be in mixedCase
Pos: 46:400
Variable name must be in mixedCase
Pos: 22:414
Avoid to use tx.origin
Pos: 31:415
Error message for require is too long
Pos: 13:418
Error message for require is too long
Pos: 13:424
Error message for require is too long
Pos: 13:432
Error message for require is too long
Pos: 13:440
Error message for require is too long
Pos: 13:448
Error message for require is too long
Pos: 13:460
Error message for require is too long
Pos: 13:472
Error message for require is too long
Pos: 13:484
Error message for require is too long
Pos: 13:496
Variable name must be in mixedCase
Pos: 41:508
Avoid to use tx.origin
Pos: 31:536
Error message for require is too long
Pos: 9:539
Variable name must be in mixedCase
Pos: 25:547
Avoid to use tx.origin
Pos: 31:548
Variable name must be in mixedCase
Pos: 23:563
Variable name must be in mixedCase
Pos: 38:563
Avoid to use tx.origin
Pos: 31:564
Error message for require is too long
```

```
Pos: 9:572
Variable name must be in mixedCase
Pos: 41:581
Possible reentrancy vulnerabilities. Avoid state changes after
transfer.
Pos: 13:743
Possible reentrancy vulnerabilities. Avoid state changes after
transfer.
Pos: 17:746
Possible reentrancy vulnerabilities. Avoid state changes after
transfer.
Pos: 17:747
Possible reentrancy vulnerabilities. Avoid state changes after
transfer.
Pos: 13:749
Variable name must be in mixedCase
Pos: 31:753
```

A_NFT.sol

```
Compiler version ^0.8.19 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:267
Error message for require is too long
Pos: 9:291
Error message for require is too long
Pos: 9:344
Error message for require is too long
Pos: 9:365
Error message for require is too long
Pos: 9:385
Avoid using low level calls.
Pos: 51:386
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 17:402
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:481
Error message for require is too long
Pos: 9:498
Error message for require is too long
Pos: 9:506
Error message for require is too long
Pos: 9:522
Error message for require is too long
Pos: 9:531
Error message for require is too long
Pos: 9:532
Error message for require is too long
Pos: 9:539
Error message for require is too long
Pos: 9:552
Error message for require is too long
```

```
Pos: 5:560
Error message for require is too long
Pos: 9:566
Error message for require is too long
Pos: 9:572
Error message for require is too long
Pos: 9:581
Error message for require is too long
Pos: 9:608
Error message for require is too long
Pos: 9:609
Error message for revert is too long
Pos: 21:629
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 21:631
Code contains empty blocks
Pos: 95:641
Error message for require is too long
Pos: 9:663
Error message for require is too long
Pos: 9:676
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:743
Error message for require is too long
Pos: 9:768
Error message for require is too long
Pos: 9:794
Contract name must be in CamelCase
Pos: 1:805
Variable name must be in mixedCase
Pos: 5:807
Explicitly mark visibility of state
Pos: 5:813
Explicitly mark visibility of state
Pos: 5:814
Variable name must be in mixedCase
Pos: 5:820
Event name must be in CamelCase
Pos: 5:832
Event name must be in CamelCase
Pos: 5:833
Event name must be in CamelCase
Pos: 5:834
Event name must be in CamelCase
Pos: 5:835
Event name must be in CamelCase
Pos: 5:836
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:838
Function name must be in mixedCase
Pos: 5:849
Function name must be in mixedCase
Pos: 5:899
Variable name must be in mixedCase
Pos: 36:899
```

B_NFT.sol

```
Compiler version ^0.8.19 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:268
Error message for require is too long
Pos: 9:292
Error message for require is too long
Pos: 9:345
Error message for require is too long
Pos: 9:366
Error message for require is too long
Pos: 9:386
Avoid using low level calls.
Pos: 51:387
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 17:403
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:482
Error message for require is too long
Pos: 9:499
Error message for require is too long
Pos: 9:507
Error message for require is too long
Pos: 9:523
Error message for require is too long
Pos: 9:532
Error message for require is too long
Pos: 9:533
Error message for require is too long
Pos: 9:540
Error message for require is too long
Pos: 9:553
Error message for require is too long
Pos: 5:561
Error message for require is too long
Pos: 9:567
Error message for require is too long
Pos: 9:573
Error message for require is too long
Pos: 9:582
Error message for require is too long
Pos: 9:609
Error message for require is too long
Pos: 9:610
Error message for revert is too long
Pos: 21:630
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 21:632
Code contains empty blocks
Pos: 95:642
Error message for require is too long
Pos: 9:664
Error message for require is too long
```

```
Pos: 9:677
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:744
Error message for require is too long
Pos: 9:769
Error message for require is too long
Pos: 9:795
Contract name must be in CamelCase
Pos: 1:806
Variable name must be in mixedCase
Pos: 5:808
Explicitly mark visibility of state
Pos: 5:814
Explicitly mark visibility of state
Pos: 5:815
Variable name must be in mixedCase
Pos: 5:821
Event name must be in CamelCase
Pos: 5:833
Event name must be in CamelCase
Pos: 5:834
Event name must be in CamelCase
Pos: 5:835
Event name must be in CamelCase
Pos: 5:836
Event name must be in CamelCase
Pos: 5:837
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:839
Function name must be in mixedCase
Pos: 5:850
Function name must be in mixedCase
Pos: 5:900
Variable name must be in mixedCase
Pos: 36:900
```

C_NFT.sol

```
Compiler version ^0.8.19 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:269
Error message for require is too long
Pos: 9:293
Error message for require is too long
Pos: 9:346
Error message for require is too long
Pos: 9:367
Error message for require is too long
Pos: 9:387
Avoid using low level calls.
Pos: 51:388
```

```
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 17:404
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:483
Error message for require is too long
Pos: 9:500
Error message for require is too long
Pos: 9:508
Error message for require is too long
Pos: 9:524
Error message for require is too long
Pos: 9:533
Error message for require is too long
Pos: 9:534
Error message for require is too long
Pos: 9:541
Error message for require is too long
Pos: 9:554
Error message for require is too long
Pos: 5:562
Error message for require is too long
Pos: 9:568
Error message for require is too long
Pos: 9:574
Error message for require is too long
Pos: 9:583
Error message for require is too long
Pos: 9:610
Error message for require is too long
Pos: 9:611
Error message for revert is too long
Pos: 21:631
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 21:633
Code contains empty blocks
Pos: 95:643
Error message for require is too long
Pos: 9:665
Error message for require is too long
Pos: 9:678
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:745
Error message for require is too long
Pos: 9:770
Error message for require is too long
Pos: 9:796
Contract name must be in CamelCase
Pos: 1:807
Variable name must be in mixedCase
Pos: 5:809
Explicitly mark visibility of state
Pos: 5:815
Explicitly mark visibility of state
Pos: 5:816
Variable name must be in mixedCase
Pos: 5:822
Event name must be in CamelCase
```

```
Pos: 5:834
Event name must be in CamelCase
Pos: 5:835
Event name must be in CamelCase
Pos: 5:836
Event name must be in CamelCase
Pos: 5:837
Event name must be in CamelCase
Pos: 5:838
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:840
Function name must be in mixedCase
Pos: 5:851
Function name must be in mixedCase
Pos: 5:901
Variable name must be in mixedCase
Pos: 36:901
```

D_NFT.sol

```
Compiler version ^0.8.19 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:269
Error message for require is too long
Pos: 9:293
Error message for require is too long
Pos: 9:346
Error message for require is too long
Pos: 9:367
Error message for require is too long
Pos: 9:387
Avoid using low level calls.
Pos: 51:388
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 17:404
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:483
Error message for require is too long
Pos: 9:500
Error message for require is too long
Pos: 9:508
Error message for require is too long
Pos: 9:524
Error message for require is too long
Pos: 9:533
Error message for require is too long
Pos: 9:534
Error message for require is too long
Pos: 9:541
Error message for require is too long
Pos: 9:554
```

```
Error message for require is too long
Pos: 5:562
Error message for require is too long
Pos: 9:568
Error message for require is too long
Pos: 9:574
Error message for require is too long
Pos: 9:583
Error message for require is too long
Pos: 9:610
Error message for require is too long
Pos: 9:611
Error message for revert is too long
Pos: 21:631
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 21:633
Code contains empty blocks
Pos: 95:643
Error message for require is too long
Pos: 9:665
Error message for require is too long
Pos: 9:678
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:745
Error message for require is too long
Pos: 9:770
Error message for require is too long
Pos: 9:796
Contract name must be in CamelCase
Pos: 1:807
Variable name must be in mixedCase
Pos: 5:809
Explicitly mark visibility of state
Pos: 5:815
Explicitly mark visibility of state
Pos: 5:816
Variable name must be in mixedCase
Pos: 5:822
Event name must be in CamelCase
Pos: 5:834
Event name must be in CamelCase
Pos: 5:835
Event name must be in CamelCase
Pos: 5:836
Event name must be in CamelCase
Pos: 5:837
Event name must be in CamelCase
Pos: 5:838
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:840
Function name must be in mixedCase
Pos: 5:851
Function name must be in mixedCase
Pos: 5:901
Variable name must be in mixedCase
Pos: 36:901
```

E_NFT.sol

```
Compiler version ^0.8.19 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:269
Error message for require is too long
Pos: 9:293
Error message for require is too long
Pos: 9:346
Error message for require is too long
Pos: 9:367
Error message for require is too long
Pos: 9:387
Avoid using low level calls.
Pos: 51:388
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 17:404
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:483
Error message for require is too long
Pos: 9:500
Error message for require is too long
Pos: 9:508
Error message for require is too long
Pos: 9:524
Error message for require is too long
Pos: 9:533
Error message for require is too long
Pos: 9:534
Error message for require is too long
Pos: 9:541
Error message for require is too long
Pos: 9:554
Error message for require is too long
Pos: 5:562
Error message for require is too long
Pos: 9:568
Error message for require is too long
Pos: 9:574
Error message for require is too long
Pos: 9:583
Error message for require is too long
Pos: 9:610
Error message for require is too long
Pos: 9:611
Error message for revert is too long
Pos: 21:631
Avoid to use inline assembly. It is acceptable only in rare cases
Pos: 21:633
Code contains empty blocks
Pos: 95:643
Error message for require is too long
Pos: 9:665
```

```
Error message for require is too long
Pos: 9:678
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:745
Error message for require is too long
Pos: 9:770
Error message for require is too long
Pos: 9:796
Contract name must be in CamelCase
Pos: 1:807
Variable name must be in mixedCase
Pos: 5:809
Explicitly mark visibility of state
Pos: 5:815
Explicitly mark visibility of state
Pos: 5:816
Variable name must be in mixedCase
Pos: 5:822
Event name must be in CamelCase
Pos: 5:834
Event name must be in CamelCase
Pos: 5:835
Event name must be in CamelCase
Pos: 5:836
Event name must be in CamelCase
Pos: 5:837
Event name must be in CamelCase
Pos: 5:838
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:840
Function name must be in mixedCase
Pos: 5:851
Function name must be in mixedCase
Pos: 5:901
Variable name must be in mixedCase
Pos: 36:901
```

HCToken.sol

```
Compiler version ^0.8.19 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:20
Error message for require is too long
Pos: 9:62
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:92
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:265
```

```
Error message for require is too long
Pos: 9:326
Error message for require is too long
Pos: 9:396
Error message for require is too long
Pos: 9:397
Code contains empty blocks
Pos: 24:421
Code contains empty blocks
Pos: 24:427
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:443
```

USDTToken.sol

```
Compiler version ^0.8.19 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:20
Error message for require is too long
Pos: 9:62
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:92
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:265
Error message for require is too long
Pos: 9:345
Error message for require is too long
Pos: 9:397
Code contains empty blocks
Pos: 24:421
Code contains empty blocks
Pos: 24:427
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:443
```

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