Ekta Portal NFT Smart Contract Final Audit Report

Executive Summary

Project Name: EKTA Portal NFT

Overview: The Ekta Portal NFT Collection (10,000 NFTs) is an additional collection for general Ekta users to have a chance to own an Ekta Portal Node (a Portal NFT can be used to claim 1 physical portal) besides the previous 969 OG Portal Airdrop collection (which was a free NFT distribution to early investors).

Timeline: 15th september 2022 to 23 september,2022

Method: Manual Review, Functional Testing, Automated Testing, etc.

Audit Scope : The scope of this audit was to analyse **Ekta Portal NFT** codebase for quality, security, and correctness.
https://github.com/airdropgames/EKTA-NFT-Collection/commit/31e31cfbba3e327efabf8546b308762229a4e90b

Initial Commit hash : 31e31cfbba3e327efabf8546b308762229a4e90b

Fixed In: 12eedfe1b7cf5c748f84621fef6bd5f9c71fa2f1

Number of security issues per severity.

TYPE	HIGH	MEDIUM	LOW	INFORMATIONAL
OPEN	0	0	0	0
ACKNOWLEDGED	0	1	0	0
Partially Resolved	0	0	0	0
CLOSED	1	0	0	1

Check Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle,Solhint, Mythril, Slither, Solidity statistic analysis.

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Issues Found – Code Review / Manual Testing

High Severity Issues

H.1 Centralization Related Risks

Description
Any compromise to the owner's privileged accounts may allow a hacker to take advantage of this authority and manipulate the Ekta portal NFT contract.
The batchMint function is controlled by the owner and its given the right to mint any amount of token at any time.

Recommendation:
The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the Ekta Team to carefully manage the privileged account's private key to avoid any potential risks of being hacked.

In general, we strongly recommend centralized privileges or roles in the EKTA NFT contract to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively for private sale we recommend removing the rights to the owner of minting any amount of token and the privilege should be enabled once the private sale ends.

But considering that the ending of sale is again handled by the owner of the contract.

Status: Resolved

Medium Severity Issues

M.1 DDOS attack in BatchMint and BatchTransfer

Description
When the for loop is run for a bigger range of values in the BatchMint and BatchTransfer then it raises a scenario of DDOS to the system.
When the gas prices are high then the lower range/relaistic range of minting and transfer will fail.

Recommendation:
Compute the current gas price when the batch mint and batch transfer call is made and the range of for loop should be handled accordingly.
This could be done with the **GASPRICE** opcode proposed in EIP-1559. Until EIP-1559 is implemented, it is not straightforward to compute the current gas price without an external oracle such as ETH Gas Station. However, such oracles could be DDoSed as we have seen on Feb 23rd, 2021.

Status: Acknowledged

Low Severity Issues

None

Informational Issues

I.1 Recommendations and Gas optimizations

- For test stake use smock Library.
- Gas optimization
    - _calculateFee should remove the safeMath wrapper in div.
    - The pre-increment operation is cheaper (about 5 GAS per iteration) so use ++i instead of i++ or i+= 1 in for loop. We recommend to use pre-increment in all the for loops.
    - Instead of using the && operator in a single require statement to check multiple conditions,using multiple require statements with 1 condition per require statement will save 3 GAS per &&. We recommend to implement in all the contracts.
    - In for loop the default value initilization to 0 should not be there remove from all the for loop
    - != 0 costs 6 less GAS compared to > 0 for unsigned integers in require statements with the optimizer enabled. We recommend to use !=0 instead of > 0 in all the contracts.
    - In the EVM, there is no opcode for non-strict inequalities (>=, <=) and two operations are performed (> + =.) Consider replacing >= with the strict counterpart >. Recommend to follow the inequality with a strict one.
    - Explicit set of values to there default state at the time of declaration is wastage of gas units.
    - All the public functions which are not used internally needs to be converted to external
- When the state gets updated the event should always get fired. We recommend to fire the events for all the state changes.

Status: **Partially Resolved**

Functional Tests (Rinkeby)

Contracts

- Test OG TOKEN Contract - 0x85568A5aeFFa0Ed889cF7c0311bc67674Ad3B6df
- EktaPortalNft Contract - 0xc6A29cb58904dE0C67D7c8F149A9D5359D62b5Bd

Transactions

- setIsMintingPaused to **false**
  https://rinkeby.etherscan.io/tx/0x2ddde22561d8b0959b518f5da71fc43919879981cf2df1013a637e72c9c440ef
- 100 token batch transfer by owner 2 times in private sale - Owner should not be allowed to mint tokens in private sale.
  - https://rinkeby.etherscan.io/tx/0x08f232b33b7c7852d8a1ff44ed7d46acaa41b5d33cba930a3fb83f3ffb0c8e4f
  - https://rinkeby.etherscan.io/tx/0x72eed6968db1718bd53962ade6142032233aa5751a03766c5f11d84ad6b9e5f3
- setPrice to 10 WEI
  https://rinkeby.etherscan.io/tx/0xa496156c40157945f5cb9200830b49e2a5ff13e1b7ba08121d50ba1961ceffe5
- withdraw all amount
  https://rinkeby.etherscan.io/tx/0x65362965af22c63e6f79c44aa42b006d4f66528403a1e05c2524bfee4b45e503
- setApprovalForAll for portalContract
  https://rinkeby.etherscan.io/tx/0x5c0410b450d1587eaacf979d8cca98843b5762b17b8ea88bfbf9a7db2d39488d
- batchTransfer 1,2,3,4 tokenID's
  https://rinkeby.etherscan.io/tx/0x267d632334610adfc447eaf4e36ce4a0b6b2284bd244c4753c69452d4231d0d8
- setPublicMintingStartTimestamp to `1663878429`
  https://rinkeby.etherscan.io/tx/0x02444686b974c0078bc60cf5963c4d35f197b9bd3ec80e2009c7c13922308 71e
- BatchMint 1000 tokens trigger DOS attack
  https://rinkeby.etherscan.io/tx/0x5ae79e4d785e653599801d6e19216bbba2434209d54441ce86b885292254b1cb
- batchMintForRandomUser/nonWhiteList in public sale is allowed to mint
  https://rinkeby.etherscan.io/tx/0x0d0fad1c83e03af96aa2bf69e5de187b28c1c1a7255924cdb497205081 4e25f7
- setPublicMintingStartTimestamp set again to 1663890320 and mode to private sale mode
  https://rinkeby.etherscan.io/tx/0xdb6ce942621a37a7d79557ca4a418379adaed05d932a7880cb12e61f23970a2e

- batchMintForRandomUser/nonWhiteList in private sale is not allowed to mint https://rinkeby.etherscan.io/tx/0x3e31c009f57dd1eacb8338e669a10dc18f39a74ec186a058d82a6e84699991fd
- View Methods works **fine** for NFT portal contract
    - `getMyGranterTokenIds`
    - `isGranterTokenIdUsed`
    - `getGranterTokenIds`
    - `getAddressClaimCount`
    - `getAddressMintCount`
    - `tokenURI`

QuillAudits

Ekta Portal NFT Audit Report

Automated Tests
Slither:

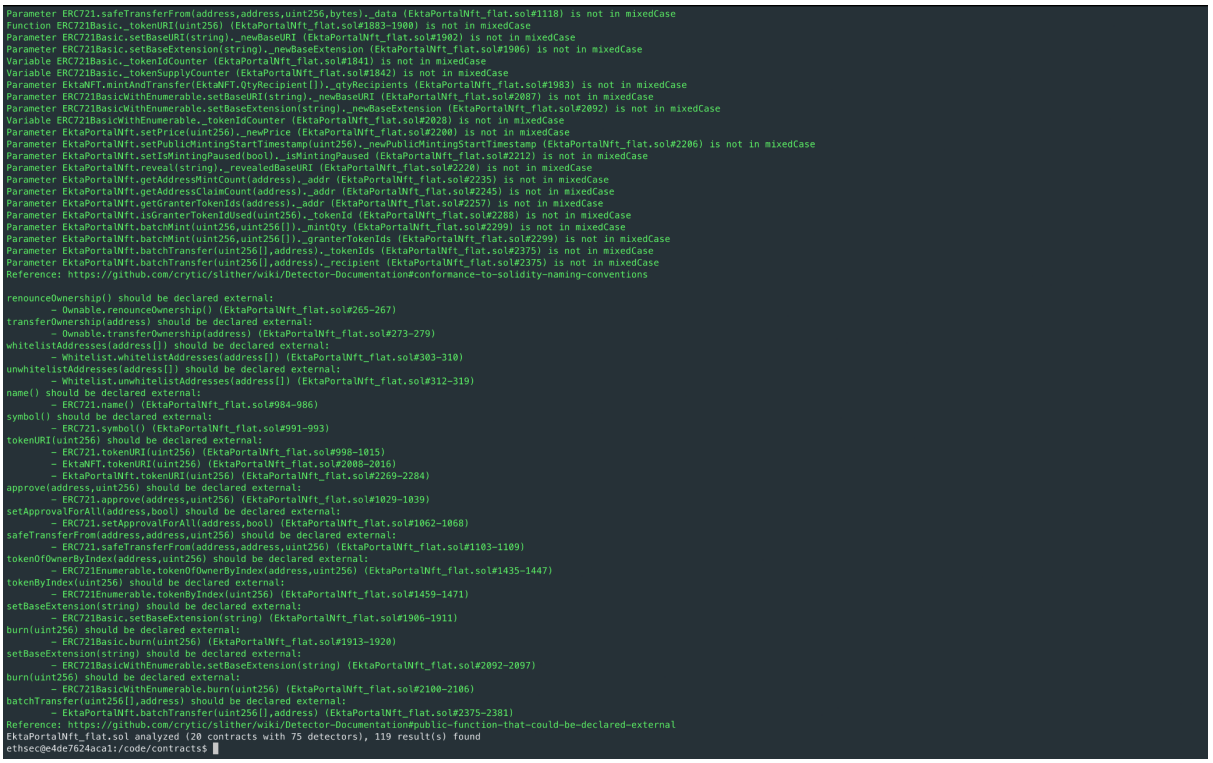
```
ethsec@e4de7624aca1:/code/contracts$ slither EktaPortalNft_flat.sol
Compilation warnings/errors on EktaPortalNft_flat.sol:
Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNL
ICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> EktaPortalNft_flat.sol



EktaPortalNft._getGranterTokenIds(address) (EktaPortalNft_flat.sol#2157-2197) uses a dangerous strict equality:
        - userGrantNftBalance == 0 (EktaPortalNft_flat.sol#2165)
EktaPortalNft._getGranterTokenIds(address) (EktaPortalNft_flat.sol#2157-2197) uses a dangerous strict equality:
        - balanceCounter == userGrantNftBalance (EktaPortalNft_flat.sol#2182)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

EktaPortalNft.batchMint(uint256,uint256[]).i (EktaPortalNft_flat.sol#2330) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721._checkOnERC721Received(address,address,uint256,bytes) (EktaPortalNft_flat.sol#1340-1370) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (EktaPortalNft_flat.sol#
1347-1366)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

ERC721Basic.constructor(string,string,string,uint256)._name (EktaPortalNft_flat.sol#1848) shadows:
        - ERC721._name (EktaPortalNft_flat.sol#905) (state variable)
ERC721Basic.constructor(string,string,string,uint256)._symbol (EktaPortalNft_flat.sol#1849) shadows:
        - ERC721._symbol (EktaPortalNft_flat.sol#908) (state variable)
EktaNFT.constructor(string,string,string,string,uint256)._name (EktaPortalNft_flat.sol#1936) shadows:
        - ERC721._name (EktaPortalNft_flat.sol#905) (state variable)
EktaNFT.constructor(string,string,string,string,uint256)._symbol (EktaPortalNft_flat.sol#1937) shadows:
        - ERC721._symbol (EktaPortalNft_flat.sol#908) (state variable)
ERC721BasicWithEnumerable.constructor(string,string,string,uint256)._name (EktaPortalNft_flat.sol#2034) shadows:
        - ERC721._name (EktaPortalNft_flat.sol#905) (state variable)
ERC721BasicWithEnumerable.constructor(string,string,string,uint256)._symbol (EktaPortalNft_flat.sol#2035) shadows:
        - ERC721._symbol (EktaPortalNft_flat.sol#908) (state variable)
EktaPortalNft.constructor(string,string,string,string,uint256,uint256,uint256,address)._name (EktaPortalNft_flat.sol#2137) shadows:
        - ERC721._name (EktaPortalNft_flat.sol#905) (state variable)
EktaPortalNft.constructor(string,string,string,string,uint256,uint256,uint256,address)._symbol (EktaPortalNft_flat.sol#2138) shadows:
        - ERC721._symbol (EktaPortalNft_flat.sol#908) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

EktaPortalNft.constructor(string,string,string,string,uint256,uint256,uint256,uint256,address)._freeAccessGranter (EktaPortalNft_flat.sol#2145) lacks a zero-check on :
                - freeAccessGranter = _freeAccessGranter (EktaPortalNft_flat.sol#2151)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

EktaPortalNft._getGranterTokenIds(address) (EktaPortalNft_flat.sol#2157-2197) has external calls inside a loop: i <= granterNft.totalSupply() (EktaPortalNft_flat.sol#2170)
EktaPortalNft._getGranterTokenIds(address) (EktaPortalNft_flat.sol#2157-2197) has external calls inside a loop: granterNft.ownerOf(i) != _addr (EktaPortalNft_flat.sol#2175)
EktaPortalNft.batchMint(uint256,uint256[]) (EktaPortalNft_flat.sol#2299-2364) has external calls inside a loop: require(bool,string)(granterNft.ownerOf(_granterTokenIds[i]) == msg.sender,You don't own this granter
NFT) (EktaPortalNft_flat.sol#2335-2338)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (EktaPortalNft_flat.sol#1354)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (EktaPortalNft_flat.sol#1340-1370) potent
ially used before declaration: retval == IERC721Receiver.onERC721Received.selector (EktaPortalNft_flat.sol#1355)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (EktaPortalNft_flat.sol#1356)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (EktaPortalNft_flat.sol#1340-1370) potent
ially used before declaration: reason.length == 0 (EktaPortalNft_flat.sol#1357)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (EktaPortalNft_flat.sol#1356)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (EktaPortalNft_flat.sol#1340-1370) potent
ially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (EktaPortalNft_flat.sol#1363)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

EktaPortalNft.batchMint(uint256,uint256[]) (EktaPortalNft_flat.sol#2299-2364) uses timestamp for comparisons
        Dangerous comparisons:
        - isPublicMintingPeriod = publicMintingStartTimestamp <= block.timestamp (EktaPortalNft_flat.sol#2345-2346)
        - require(bool,string)(isPublicMintingPeriod || isWhitelisted(msg.sender) || addressClaimCount[msg.sender] > 0,You are not whitelisted) (EktaPortalNft_flat.sol#2347-2352)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

QuillAudits

Ekta Portal NFT Audit Report

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (EktaPortalNft_flat.sol#349-359) uses assembly
        - INLINE ASM (EktaPortalNft_flat.sol#355-357)
Address.verifyCallResult(bool,bytes,string) (EktaPortalNft_flat.sol#555-575) uses assembly
        - INLINE ASM (EktaPortalNft_flat.sol#567-570)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (EktaPortalNft_flat.sol#1340-1370) uses assembly
        - INLINE ASM (EktaPortalNft_flat.sol#1362-1364)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.9', '^0.8.0', '^0.8.9']
        - ^0.8.0 (EktaPortalNft_flat.sol#5)
        - ^0.8.0 (EktaPortalNft_flat.sol#50)
        - ^0.8.0 (EktaPortalNft_flat.sol#115)
        - ^0.8.0 (EktaPortalNft_flat.sol#188)
        - ^0.8.0 (EktaPortalNft_flat.sol#214)
        - ^0.8.9 (EktaPortalNft_flat.sol#294)
        - ^0.8.0 (EktaPortalNft_flat.sol#326)
        - ^0.8.0 (EktaPortalNft_flat.sol#582)
        - ^0.8.0 (EktaPortalNft_flat.sol#611)
        - ^0.8.0 (EktaPortalNft_flat.sol#638)
        - ^0.8.0 (EktaPortalNft_flat.sol#673)
        - ^0.8.0 (EktaPortalNft_flat.sol#834)
        - ^0.8.0 (EktaPortalNft_flat.sol#866)
        - ^0.8.0 (EktaPortalNft_flat.sol#893)
        - ^0.8.0 (EktaPortalNft_flat.sol#1397)
        - ^0.8.0 (EktaPortalNft_flat.sol#1587)
        - ^0.8.9 (EktaPortalNft_flat.sol#1834)
        - 0.8.9 (EktaPortalNft_flat.sol#1925)
        - ^0.8.9 (EktaPortalNft_flat.sol#2021)
        - 0.8.9 (EktaPortalNft_flat.sol#2111)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (EktaPortalNft_flat.sol#408-413) is never used and should be removed
Address.functionCall(address,bytes,string) (EktaPortalNft_flat.sol#421-427) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (EktaPortalNft_flat.sol#440-452) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (EktaPortalNft_flat.sol#460-476) is never used and should be removed
Address.functionDelegateCall(address,bytes) (EktaPortalNft_flat.sol#520-530) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (EktaPortalNft_flat.sol#538-547) is never used and should be removed
Address.functionStaticCall(address,bytes) (EktaPortalNft_flat.sol#484-495) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (EktaPortalNft_flat.sol#503-512) is never used and should be removed
Address.sendValue(address,uint256) (EktaPortalNft_flat.sol#377-388) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (EktaPortalNft_flat.sol#555-575) is never used and should be removed
Context._msgData() (EktaPortalNft_flat.sol#205-207) is never used and should be removed
Counters.reset(Counters.Counter) (EktaPortalNft_flat.sol#41-43) is never used and should be removed
ERC721._baseURI() (EktaPortalNft_flat.sol#1022-1024) is never used and should be removed
SafeMath.div(uint256,uint256) (EktaPortalNft_flat.sol#1738-1740) is never used and should be removed
SafeMath.div(uint256,uint256,string) (EktaPortalNft_flat.sol#1794-1803) is never used and should be removed
SafeMath.mod(uint256,uint256) (EktaPortalNft_flat.sol#1754-1756) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (EktaPortalNft_flat.sol#1820-1829) is never used and should be removed
SafeMath.mul(uint256,uint256) (EktaPortalNft_flat.sol#1724-1726) is never used and should be removed
SafeMath.sub(uint256,uint256) (EktaPortalNft_flat.sol#1710-1712) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (EktaPortalNft_flat.sol#1771-1780) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (EktaPortalNft_flat.sol#1605-1615) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (EktaPortalNft_flat.sol#1659-1668) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (EktaPortalNft_flat.sol#1675-1684) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (EktaPortalNft_flat.sol#1638-1652) is never used and should be removed
SafeMath.trySub(uint256,uint256) (EktaPortalNft_flat.sol#1622-1631) is never used and should be removed
Strings.toHexString(uint256) (EktaPortalNft_flat.sol#151-162) is never used and should be removed
Strings.toHexString(uint256,uint256) (EktaPortalNft_flat.sol#167-181) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (EktaPortalNft_flat.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (EktaPortalNft_flat.sol#50) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (EktaPortalNft_flat.sol#115) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (EktaPortalNft_flat.sol#188) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (EktaPortalNft_flat.sol#214) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.9 (EktaPortalNft_flat.sol#294) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (EktaPortalNft_flat.sol#326) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (EktaPortalNft_flat.sol#582) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (EktaPortalNft_flat.sol#611) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (EktaPortalNft_flat.sol#638) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (EktaPortalNft_flat.sol#673) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (EktaPortalNft_flat.sol#834) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (EktaPortalNft_flat.sol#866) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (EktaPortalNft_flat.sol#893) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (EktaPortalNft_flat.sol#1397) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (EktaPortalNft_flat.sol#1587) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.9 (EktaPortalNft_flat.sol#1834) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (EktaPortalNft_flat.sol#1925) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.9 (EktaPortalNft_flat.sol#2021) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (EktaPortalNft_flat.sol#2111) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (EktaPortalNft_flat.sol#377-388):
        - (success) = recipient.call{value: amount}() (EktaPortalNft_flat.sol#383)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (EktaPortalNft_flat.sol#460-476):
        - (success,returndata) = target.call{value: value}(data) (EktaPortalNft_flat.sol#472-474)
Low level call in Address.functionStaticCall(address,bytes,string) (EktaPortalNft_flat.sol#503-512):
        - (success,returndata) = target.staticcall(data) (EktaPortalNft_flat.sol#510)
Low level call in Address.functionDelegateCall(address,bytes,string) (EktaPortalNft_flat.sol#538-547):
        - (success,returndata) = target.delegatecall(data) (EktaPortalNft_flat.sol#545)
Low level call in EktaPortalNft.withdraw(uint256) (EktaPortalNft_flat.sol#2228-2232):
        - (sent) = address(owner()).call{value: amount}() (EktaPortalNft_flat.sol#2230)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter Whitelist.isWhitelisted(address)._address (EktaPortalNft_flat.sol#299) is not in mixedCase
Parameter Whitelist.whitelistAddresses(address[])._addresses (EktaPortalNft_flat.sol#303) is not in mixedCase
Parameter Whitelist.unwhitelistAddresses(address[])._addresses (EktaPortalNft_flat.sol#312) is not in mixedCase
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (EktaPortalNft_flat.sol#1118) is not in mixedCase
Function ERC721Basic._tokenURI(uint256) (EktaPortalNft_flat.sol#1883-1900) is not in mixedCase
Parameter ERC721Basic.setBaseURI(string)._newBaseURI (EktaPortalNft_flat.sol#1902) is not in mixedCase
Parameter ERC721Basic.setBaseExtension(string)._newBaseExtension (EktaPortalNft_flat.sol#1906) is not in mixedCase
Variable ERC721Basic._tokenIdCounter (EktaPortalNft_flat.sol#1841) is not in mixedCase
Variable ERC721Basic._tokenSupplyCounter (EktaPortalNft_flat.sol#1842) is not in mixedCase
Parameter EktaNFT.mintAndTransfer(EktaNFT.QtyRecipient[])._qtyRecipients (EktaPortalNft_flat.sol#1983) is not in mixedCase
Parameter ERC721BasicWithEnumerable.setBaseURI(string)._newBaseURI (EktaPortalNft_flat.sol#2087) is not in mixedCase
Parameter ERC721BasicWithEnumerable.setBaseExtension(string)._newBaseExtension (EktaPortalNft_flat.sol#2092) is not in mixedCase
Variable ERC721BasicWithEnumerable._tokenIdCounter (EktaPortalNft_flat.sol#2028) is not in mixedCase
Parameter EktaPortalNft.setPrice(uint256)._newPrice (EktaPortalNft_flat.sol#2200) is not in mixedCase
Parameter EktaPortalNft.setPublicMintingStartTimestamp(uint256)._newPublicMintingStartTimestamp (EktaPortalNft_flat.sol#2206) is not in mixedCase
Parameter EktaPortalNft.setIsMintingPaused(bool)._isMintingPaused (EktaPortalNft_flat.sol#2212) is not in mixedCase
Parameter EktaPortalNft.reveal(string)._revealedBaseURI (EktaPortalNft_flat.sol#2220) is not in mixedCase
Parameter EktaPortalNft.getAddressMintCount(address)._addr (EktaPortalNft_flat.sol#2235) is not in mixedCase
Parameter EktaPortalNft.getAddressClaimCount(address)._addr (EktaPortalNft_flat.sol#2245) is not in mixedCase
Parameter EktaPortalNft.getGranterTokenIds(address)._addr (EktaPortalNft_flat.sol#2257) is not in mixedCase
Parameter EktaPortalNft.isGranterTokenIdUsed(uint256)._tokenId (EktaPortalNft_flat.sol#2288) is not in mixedCase
Parameter EktaPortalNft.batchMint(uint256,uint256[])._mintQty (EktaPortalNft_flat.sol#2299) is not in mixedCase
Parameter EktaPortalNft.batchMint(uint256,uint256[])._granterTokenIds (EktaPortalNft_flat.sol#2299) is not in mixedCase
Parameter EktaPortalNft.batchTransfer(uint256[],address)._tokenIds (EktaPortalNft_flat.sol#2375) is not in mixedCase
Parameter EktaPortalNft.batchTransfer(uint256[],address)._recipient (EktaPortalNft_flat.sol#2375) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```



```
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (EktaPortalNft_flat.sol#1118) is not in mixedCase
Function ERC721Basic._tokenURI(uint256) (EktaPortalNft_flat.sol#1883-1900) is not in mixedCase
Parameter ERC721Basic.setBaseURI(string)._newBaseURI (EktaPortalNft_flat.sol#1902) is not in mixedCase
Parameter ERC721Basic.setBaseExtension(string)._newBaseExtension (EktaPortalNft_flat.sol#1906) is not in mixedCase
Variable ERC721Basic._tokenIdCounter (EktaPortalNft_flat.sol#1841) is not in mixedCase
Variable ERC721Basic._tokenSupplyCounter (EktaPortalNft_flat.sol#1842) is not in mixedCase
Parameter EktaNFT.mintAndTransfer(EktaNFT.QtyRecipient[])._qtyRecipients (EktaPortalNft_flat.sol#1983) is not in mixedCase
Parameter ERC721BasicWithEnumerable.setBaseURI(string)._newBaseURI (EktaPortalNft_flat.sol#2087) is not in mixedCase
Parameter ERC721BasicWithEnumerable.setBaseExtension(string)._newBaseExtension (EktaPortalNft_flat.sol#2092) is not in mixedCase
Variable ERC721BasicWithEnumerable._tokenIdCounter (EktaPortalNft_flat.sol#2028) is not in mixedCase
Parameter EktaPortalNft.setPrice(uint256)._newPrice (EktaPortalNft_flat.sol#2200) is not in mixedCase
Parameter EktaPortalNft.setPublicMintingStartTimestamp(uint256)._newPublicMintingStartTimestamp (EktaPortalNft_flat.sol#2206) is not in mixedCase
Parameter EktaPortalNft.setIsMintingPaused(bool)._isMintingPaused (EktaPortalNft_flat.sol#2212) is not in mixedCase
Parameter EktaPortalNft.reveal(string)._revealedBaseURI (EktaPortalNft_flat.sol#2220) is not in mixedCase
Parameter EktaPortalNft.getAddressMintCount(address)._addr (EktaPortalNft_flat.sol#2235) is not in mixedCase
Parameter EktaPortalNft.getAddressClaimCount(address)._addr (EktaPortalNft_flat.sol#2245) is not in mixedCase
Parameter EktaPortalNft.getGranterTokenIds(address)._addr (EktaPortalNft_flat.sol#2257) is not in mixedCase
Parameter EktaPortalNft.isGranterTokenIdUsed(uint256)._tokenId (EktaPortalNft_flat.sol#2288) is not in mixedCase
Parameter EktaPortalNft.batchMint(uint256,uint256[])._mintQty (EktaPortalNft_flat.sol#2290) is not in mixedCase
Parameter EktaPortalNft.batchMint(uint256,uint256[])._granterTokenIds (EktaPortalNft_flat.sol#2299) is not in mixedCase
Parameter EktaPortalNft.batchTransfer(uint256[],address)._tokenIds (EktaPortalNft_flat.sol#2375) is not in mixedCase
Parameter EktaPortalNft.batchTransfer(uint256[],address)._recipient (EktaPortalNft_flat.sol#2375) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (EktaPortalNft_flat.sol#265-267)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (EktaPortalNft_flat.sol#273-279)
whitelistAddresses(address[]) should be declared external:
        - Whitelist.whitelistAddresses(address[]) (EktaPortalNft_flat.sol#303-310)
unwhitelistAddresses(address[]) should be declared external:
        - Whitelist.unwhitelistAddresses(address[]) (EktaPortalNft_flat.sol#312-319)
name() should be declared external:
        - ERC721.name() (EktaPortalNft_flat.sol#984-986)
symbol() should be declared external:
        - ERC721.symbol() (EktaPortalNft_flat.sol#991-993)
tokenURI(uint256) should be declared external:
        - ERC721.tokenURI(uint256) (EktaPortalNft_flat.sol#998-1015)
        - EktaNFT.tokenURI(uint256) (EktaPortalNft_flat.sol#2008-2016)
        - EktaPortalNft.tokenURI(uint256) (EktaPortalNft_flat.sol#2269-2284)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (EktaPortalNft_flat.sol#1029-1039)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (EktaPortalNft_flat.sol#1062-1068)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (EktaPortalNft_flat.sol#1103-1109)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721Enumerable.tokenOfOwnerByIndex(address,uint256) (EktaPortalNft_flat.sol#1435-1447)
tokenByIndex(uint256) should be declared external:
        - ERC721Enumerable.tokenByIndex(uint256) (EktaPortalNft_flat.sol#1459-1471)
setBaseExtension(string) should be declared external:
        - ERC721Basic.setBaseExtension(string) (EktaPortalNft_flat.sol#1906-1911)
burn(uint256) should be declared external:
        - ERC721Basic.burn(uint256) (EktaPortalNft_flat.sol#1913-1920)
setBaseExtension(string) should be declared external:
        - ERC721BasicWithEnumerable.setBaseExtension(string) (EktaPortalNft_flat.sol#2092-2097)
burn(uint256) should be declared external:
        - ERC721BasicWithEnumerable.burn(uint256) (EktaPortalNft_flat.sol#2100-2106)
batchTransfer(uint256[],address) should be declared external:
        - EktaPortalNft.batchTransfer(uint256[],address) (EktaPortalNft_flat.sol#2375-2381)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
EktaPortalNft_flat.sol analyzed (20 contracts with 75 detectors), 119 result(s) found
ethsec@e4de7624aca1:/code/contracts$
```

Results

There were 75 results uncovered via Slither for the EKTA Portal NFT contract, and we checked through all of them and found them to be false positives.

Closing Summary

In this report, we have considered the security of the **EKTA Portal NFT**. We performed our audit according to the procedure described above.

One high, one medium and one informational issue are found in the Initial audit and the EKTA NFT Team has fixed the high severity Issue.

Disclaimer:

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the **EKTA Portal NFT** Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the **EKTA Portal NFT** Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.