Smart Contract Audit

UNBLOCKED x ELEVENUP

UNBLOCKED 1

Disclaimer

UNBLOCKED reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team. UNBLOCKED do not cover testing or auditing the integration with external contract or services (such as OpenZeppelin, Uniswap, PancakeSwap etc'...)

UNBLOCKED Audits do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. UNBLOCKED Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort. UNBLOCKED Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. UNBLOCKED's position is that each company and individual are responsible for their own due diligence and continuous security.

UNBLOCKED in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Introduction

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team, documenting any issues as there were discovered.

Scope

To ensure the correct version of the audited code, each file was hashed using the SHA256 hash function. The different hashes of the smart contracts audited are the following:

- ElevenFarming.sol

  42E0B6D4440D71849F5F057A17787975A448F3DFA554AC7526855D779226117D

- ElevenLottery.sol

  C93599B2789DF9FC6430F2A7EA34A303C8F76356EEBEB8BDFEADB89498C2D895

- ElevenStaking.sol

  646EC9F4961E384B286D11405AF21EECC6799110C7C6C3207CB96DC52113C66A

- ElevenUp.sol

  64BCF994DFA3F68DEE3EA62E6956EE3C5853447148A2C65F341CFC227F27041A

- ElevenUpNFT.sol

  5B0BE8DA61C6DC5CAFC07D2DDE940ACCBF513A1FCFD151E00A85071964CFAB12

Some libraries have been used, including :

- CloneFactory.sol

  7005EF1F2F4BF38E4DAEB6C082CB322E607B6D6C5C697B3A4C84BB6D896555A0

- PancakeOracleLibrary.sol

  95DCA38FE7B3539F62EB63283B9F39941348BF1AAC95B94CC960F204C9F986C0

- UniswapV2Library.sol

  3A364FC3BC1C8A614B982874C5B191F5593B73FB2FE496EA1C6AF707CAC4B312

The specifications of smart contracts were based on the following whitepaper:

https://docs.google.com/document/d/1vzFNOdNAUsKVwfycYANbrPqm3FNZ8iCtm44PXKS1zuk/edit#

The auditing process follows a routine series of steps :

- Compilation
- Manual Review
- MythX Static Analysis
- Code Style and best practices review
- Tests check

Compilation

contracts/ElevenFarming.sol:87:5: Warning: Visibility for constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is sufficient.
    constructor(
    ^ (Relevant source part starts here and spans across multiple lines).

contracts/ElevenUpNFT.sol:387:13: Warning: Unused local variable.
    uint80 roundID,
    ^------------^

contracts/ElevenUpNFT.sol:389:13: Warning: Unused local variable.
    uint256 startedAt,
    ^---------------^

contracts/ElevenUpNFT.sol:390:13: Warning: Unused local variable.
    uint256 timeStamp,
    ^---------------^

contracts/ElevenUpNFT.sol:391:13: Warning: Unused local variable.
    uint80 answeredInRound
    ^--------------------^

contracts/ElevenLottery.sol:140:9: Warning: Unused function parameter. Remove or comment out the variable name to silence this warning.
    address _vrfCoordinator,
    ^---------------------^

contracts/ElevenLottery.sol:141:9: Warning: Unused function parameter. Remove or comment out the variable name to silence this warning.
    address _linkToken,
    ^----------------^

contracts/ElevenLottery.sol:235:32: Warning: Unused function parameter. Remove or comment out the variable name to silence this warning.
    function fulfillRandomness(bytes32 requestId, uint256 randomness)

Some warnings was founded, we recommend that you take these warnings into consideration and delete the unused variables.

Manual Review

Each issue has an assigned severity:

- Minor issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
- Medium issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- Major issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Critical issues are directly exploitable security vulnerabilities that need to be fixed.

ElevenUpNFT.sol

1. SafeTransferFrom fonction

Minor

Be careful before each SafeTransferFrom you have to think to pass by the approve function.

Reference : function newPack, _insertUser.

Think about the implementation and integration with the application.

ElevenUpStaking.sol

1. Constructor - Initialization

Medium

No need to initialize the variables to 0 or addresses to address(0) because by default this is the case.

2. Fonction “disableStakeFor”

Medium

Add a require that checks if the staking is active or not. This will allow you to optimize gas costs, avoiding an additional call to the smart contract.

Review Manuelle

3. “updateElevenRewardAmount” Function

Medium

Add a require that checks if the amount is greater than 0. This will allow you to optimize gas costs, avoiding an additional call to the smart contract.

4. “updateElevenRewardAddress” Function

Medium

Add a require that checks if the new address is different from the current address. This will allow you to optimize gas costs, avoiding an extra call to the smart contract.

5. “updateElevenRewardPeriod” Function

Medium

Add a require that checks if the reward period is different from the current period. This will allow you to optimize gas costs, avoiding an additional call to the smart contract.

LotteryFactory.sol

1. Constructor - Initialization

Medium

No need to initialize the constructor nor to declare it since it is empty.

2. “setLibraryAddress” Function

Medium

Add a require that checks if the new address is different from the current address and is not equal to address(0). This will allow you to optimize gas costs, avoiding an extra call to the smart contract.

CloneFactory.sol

Ok

Review Manuelle

ElevenLottery.sol

1. Constructor — Medium

Add a require to check if "_refTokenId" exists.

2. “initialize” Function — Medium

Add a require to check if "_refTokenId" exists.

3. “set11upNFT” Function — Medium

Add a require to check if "_refTokenId" exists.

4. “buyTickets” Function - transferFrom — Minor

Be careful before each SafeTransferFrom you have to think to pass by the approve function.

Think about the implementation and integration with the application.

ElevenFarming.sol

1. Gas Optimization - variables initializations — Medium

Each variable assignment in Solidity costs gas. The variable "totalAllocPoint" is by default equal to 0, no need to initialize it in order to optimize the gas.

2. Constructor — Medium

Add a require that checks if "_elevenUp" is a valid address and not equal to address(0). The same for "_elevenUpPerBlock" and "_startBlock" check that it is not equal to 0. This will allow you to optimize gas costs, avoiding an extra call to the smart contract.

Code Style

1. Smart contract structure – Order of functions

Minor

Ordering helps readers identify which functions they can call and to find the constructor and fallback definitions easier.

Functions should be grouped according to their visibility and ordered:

- constructor
- receive function (if exists)
- fallback function (if exists)
- external
- public
- internal
- private

Within a grouping, place the **view** and **pure** functions last.

2. Gas Optimization - variables declarations

Medium

Remember to package your variables !

In solidity when we write a smart contract we have different places where the data is saved the one that we will be talking about is the storage. Storage: This is where all the contract state variables reside. and every time we initialize, declare or re-assign a value of a state var we spend gas. So we have to be careful with state vars and how we handle them. But more important even is that we have to pay attention to how vars are being packed.

We save state vars on the storage slot by slot, each slot has a space of 32 bytes equal to 256 bits. When we declared a uint8 (unsigned integer of 8bits), and then a uint256 (unsigned integer of 256bits) on the storage since the 8 and 256 uint can not be together on a slot that is 256 bits our contract will take 2 slots and one of them it will be just for an 8 bit. In the end, we will consume 64bits of space. But the worse part is that: when you have a entire slot and you have just a unit8 due to a lack of proper var packaging **the Ethereum virtual machine has to first convert it to uint256 to work it and the conversion cost more Gas!**

```
contract badPackaging {
 uint192 a;
 uint256 b;
}
```

```
contract GoodPackaging {
    uint8 a;
    uint16 b;
    uint88 c;
    uint64 d;
    uint40 e;
    uint40 f;
}
```

Style du code

3. Gas Optimisation - variables initializations

Medium

Every variable assignment in Solidity costs gas. When initializing variables, we often waste gas by assigning default values that will never be used.

uint256 value ; **is cheaper than** uint256 value = 0 ;

MythX Static analysis

ElevenLottery.sol

1 ISSUE For ElevenLottery.sol

Related issues

Report for @openzeppelin/contracts/token/ERC20/ERC20.sol

https://dashboard.mythx.io/#/console/analyses/318109f0-56ae-4343-bad3-3c57bc2a813b

Line	SWC Title	Severity	Short Description
149	(SWC-115) Authorization through tx.origin	Low	Use of "tx.origin" as a part of authorization control.

MythX Static analysis

ElevenFarming.sol

O ISSUE For ElevenFarming.sol

Related issues

Report for @openzeppelin/contracts/utils/Counters.sol

https://dashboard.mythx.io/#/console/analyses/0638cc0e-7226-48e6-a153-c6cbd5cdc5bb

Line	SWC Title	Severity	Short Description
41	(SWC-107) Reentrancy	Low	Write to persistent state following external call.

MythX Static analysis

Report for @openzeppelin/contracts/utils/Arrays.sol

https://dashboard.mythx.io/#/console/analyses/0638cc0e-7226-48e6-a153-c6cbd5cdc5bb

Line	SWC Title	Severity	Short Description
17	(SWC-123) Requirement Violation	Low	Requirement violation.
48	(SWC-123) Requirement Violation	Low	Requirement violation.
48	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
48	(SWC-107) Reentrancy	Low	Write to persistent state following external call.
48	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.

MythX Static analysis

ElevenStaking.sol

O ISSUE For ElevenStaking.sol

MythX Static analysis

ElevenUp.sol

O ISSUE For ElevenUpsol

MythX Static analysis

ElevenUpNFT.sol

O ISSUE For ElevenUpNFT.sol

Related issues

Report for @openzeppelin/contracts/math/Math.sol

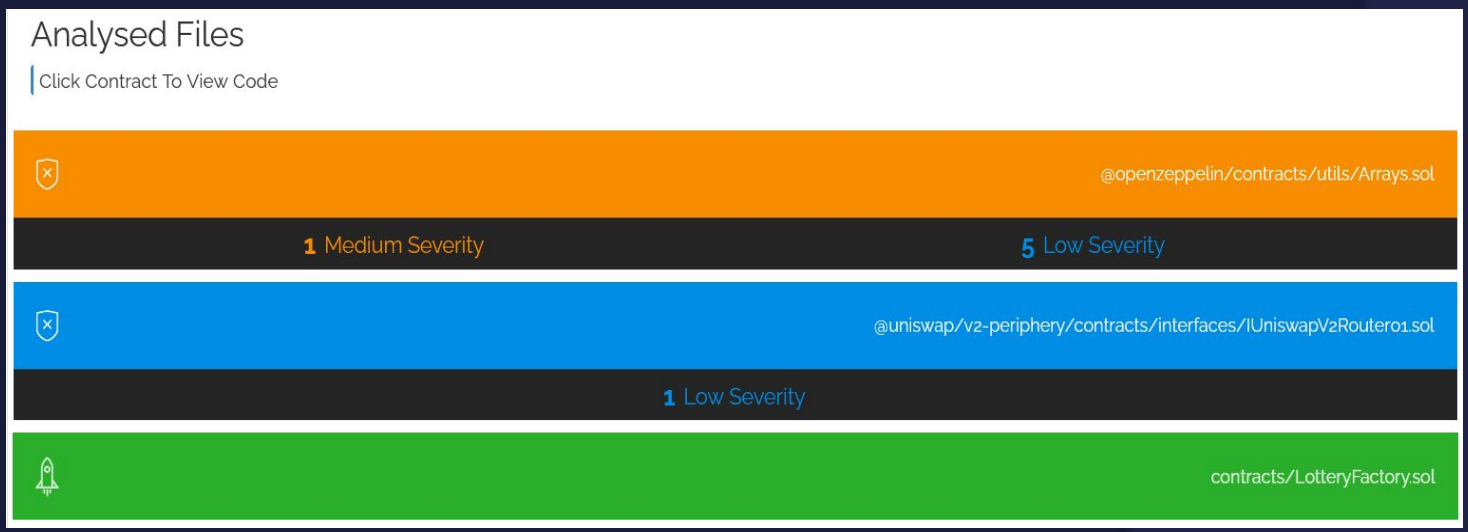
https://dashboard.mythx.io/#/console/analyses/072a75b8-186a-4e68-b656-8687199f1db5

Line	SWC Title	Severity	Short Description
32	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

MythX Static analysis

LotteryFactory.sol

O ISSUE For LotteryFactory.sol



Related issues

Report for @uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol

https://dashboard.mythx.io/#/console/analyses/5c406224-6096-44bf-8f63-0fc83d8d907e

Line	SWC Title	Severity	Short Description
96	(SWC-115) Authorization through tx.origin	Low	Use of "tx.origin" as a part of authorization control.

MythX Static analysis

Report for @openzeppelin/contracts/utils/Arrays.sol

https://dashboard.mythx.io/#/console/analyses/5c406224-6096-44bf-8f63-0fc83d8d907e

Line	SWC Title	Severity	Short Description
14	(SWC-123) Requirement Violation	Low	Requirement violation.
48	(SWC-123) Requirement Violation	Low	Requirement violation.
48	(SWC-113) DoS with Failed Call	Medium	Multiple calls are executed in the same transaction.
48	(SWC-107) Reentrancy	Low	Read of persistent state following external call.

Tests

All implemented tests passed with success.

```
11coin
  Deployment
    ✓ should deploy the ElevenCoin with a proper contract address (0ms)
    ✓ should fill the reserve with minter supply (186ms)
  Setters
    setAddressReserve method
      ✓ should only be callable by the owner (27ms)
      ✓ should not set the reserve address to the 0 address (12ms)
      ✓ should update the reserve address (37ms)
  setNFTAddress method
    ✓ should only be callable by the owner (12ms)
    ✓ should not set the NFT address to the 0 address (13ms)
    ✓ should update the NFT address (35ms)
  setVotingContract method
    ✓ should not set new voting contract if caller is not the owner (9ms)
    ✓ should not set new voting contract with Zero address (10ms)
    ✓ should set new voting contract (25ms)
  setFarmingAddress method
    ✓ should only be callable by the owner (13ms)
    ✓ should not set the farming address to the 0 address (17ms)
    ✓ should update the farming address (34ms)
  mintToFarm method
    ✓ should only be callable by the farming contract (8ms)
    ✓ should transfer to an account some 11coins tokens (43ms)
  Freeze functions
    setFrozePercentage method
      ✓ should only be callable by the owner (8ms)
      ✓ should not set the percentage above 100% (17ms)
      ✓ should set the freeze percentage (30ms)
    setFreezeTime method
      ✓ should only be callable by the owner (11ms)
      ✓ should set the freeze time (19ms)
    mintAndFreeze method
      ✓ should only be callable by the farming contract (10ms)
      ✓ should transfer to an account some 11coins tokens (49ms)
      ✓ should freeze a percentage of tokens transfered (4ms)
      ✓ should not transfer the froze part of the balance (26ms)
      ✓ should transfer the unfroze part of the balance (63ms)
      ✓ should unfreeze after the freeze period (102ms)
      ✓ should not transfer more than balance amount (7ms)
  transferFrom method
    ✓ should transfer an amount to an adress (68ms)
  Snapshot stuff
    ✓ should not allow user to query snapshotted balance if no snapshot were made (6ms)
    ✓ should not allow user to query snapshotted balance if snapshotID does not exist (6ms)
    ✓ should allow admin to take snapshot (23ms)
  Getters
    ✓ should return the tokens decimals (5ms)
    ✓ should return the tokens symbol (8ms)
    ✓ should return the tokens name (5ms)
    ✓ should return the tokens inital supply (4ms)
    ✓ should return the freeze time (3ms)
    ✓ should return the freeze percentage (3ms)
```

Tests

```
ElevenFarming
  Deployment
    ✓ should deploy the elevenFarming with a proper contract address (0ms)
  add method
    ✓ should not add pool if user is not allowed (20ms)
    ✓ should add new pool (40ms)
  set method
    ✓ should not update pool if pool id does not exist (6ms)
    ✓ should not update pool allocPoint if caller is not allowed (9ms)
    ✓ should update pool allocPoint (23ms)
  deposit method
    ✓ should not allow deposit in a pool that does not exist (7ms)
    ✓ should not allow deposit if user does not have enough balance (19ms)
    ✓ should not allow deposit if user does not approve balance (13ms)
    ✓ should deposit in farming pool (119ms)
    ✓ should increase the deposit in farming pool (235ms)
  pending11UP method
    ✓ should not return pending 11UP rewards if pool id does not exist (6ms)
    ✓ should return 0 pending 11UP rewards if no blocks have passed (6ms)
    ✓ should return pending 11UP rewards if blocks have passed (23ms)
  updatePool method
    ✓ should not update pool if pool id does not exist (9ms)
    ✓ should update pool (41ms)
  withdraw method
    ✓ should not withdraw pool if pool id does not exist (9ms)
    ✓ should not allow to withdraw more than the user stakes in pool (5ms)
    ✓ should allow user to withdraw partially from staking pool (177ms)
    ✓ should allow user to withdraw everything from staking pool (189ms)
  massUpdatePools method
    ✓ should mass update pools (27ms)
  updateMultiplier method
    ✓ should not update the bonus multiplier if caller is not allowed (5ms)
    ✓ should update bonus multiplier (20ms)
  emergencyWithdraw method
    ✓ should not allow emergency withdraw on pool that does not exist (8ms)
    ✓ should allow emergency withdraw (74ms)
    ✓ should allow emergency withdraw in case staking contract cannot pay rewards anymore (231ms)
```

Tests

```
11UPNFT
  Deployment
    ✓ should deploy the ElevenCoin with a proper contract address (0ms)
  Setters
    _setBaseURI method
      ✓ should only be callable by the owner (25ms)
      ✓ should set the token base URI (53ms)
    setExchangeToken method
      ✓ should only be callable by the owner (8ms)
      ✓ should set the token to exchange with NFT (40ms)
    setRouter method
      ✓ should only be callable by the owner (6ms)
      ✓ should set oracle contract address (37ms)
    addWhitelist method
      ✓ should only be callable by the owner (7ms)
      ✓ should not use the zero adress (6ms)
      ✓ should add an user in the whitelist (26ms)
    setPackPrice method
      ✓ should only be callable by the owner (6ms)
      ✓ should update the pack prices (56ms)
  insert users in the matrix
    _insertPresale method
      ✓ should only be callable by an admin (8ms)
      ✓ should insert only a whitelisted user (24ms)
      ✓ should insert a new user with a referral during presale (171ms)
      ✓ should not allow to call _insertReferral method if the presale is active (28ms)
      ✓ should not allow to insert in presale if the presale is inactive (52ms)
    _insertReferral method after presale
      ✓ should only be callable by a whitelisted user (7ms)
      ✓ should not get a NFT if there is not enough msg.value  (21ms)
      ✓ should insert the first user after the presale users (695ms)
      ✓ should mint a new NFT (28ms)
      ✓ setTokenURI should only be callable by an admin (13ms)
      ✓ should set the URI of the new NFT (26ms)
      ✓ should not create the URI of a nonexistent NFT (9ms)
      ✓ should not mint a new token for a referred user (10ms)
  newPack method
    ✓ should not upgrade a pack for an inexistant member (12ms)
    ✓ should not upgrade a pack if there is not enough msg.value  (16ms)
    ✓ should update the user's pack (190ms)
```

Tests

```
Getters
  balanceOf method
    ✓ should return the number of token for users (10ms)
  ownerOf method
    ✓ should return the owner of a token (10ms)
  tokenTo method
    ✓ should return the tokenId of an owner (10ms)
  totalSupply method
    ✓ should return the NFT totalSupply (5ms)
  _exists method
    ✓ should return true for an existing token (23ms)
  affiliated method
    ✓ should return if the user is referred by a specific referral (10ms)
  getReferral method
    ✓ should return the referral token Id (10ms)
  getAffiliates method
    ✓ should return the token id of a given affiliate for a given referral (21ms)
  affiliatesLength method
    ✓ should return the number of affiliates for a given referral (8ms)
  getPackLevel method
    ✓ should return the pack level for a given user (12ms)
  tokenURI method
    ✓ should revert for an nonexistant tokenID (6ms)
transfer NFT
  approve method
    ✓ should only be callable by an admin (7ms)
    ✓ should approve a  NFT transfer (29ms)
  safeTransferFrom method
    ✓ should revert a not allowed transfer (6ms)
    ✓ should revert a transfer to 0 address (8ms)
    ✓ should transfer a  NFT transfer (63ms)
```

Tests

```
ElevenStaking
  Deployment
    ✓ should deploy the elevenStaking with a proper contract address (0ms)
  addStakingPack method
    ✓ should not add staking pack with minStake greater than maxStake (12ms)
    ✓ should add unusable staking flex pack (31ms)
    ✓ should not add staking pack if caller is not allowed (15ms)
    ✓ should not update a staking pack that does not exist (6ms)
    ✓ should add staking pack 1 (33ms)
    ✓ should add staking pack 2 (34ms)
    ✓ should add staking pack 3 (29ms)
    ✓ should add staking pack 4 (32ms)
    ✓ should add staking packs for testing (65ms)
  setStakingPack method
    ✓ should update staking pack (28ms)
    ✓ should not update a staking pack with minStake greater than maxStake (6ms)
    ✓ should not update a staking pack that does not exist (7ms)
  getLengthOfStackingPackOptions method
    ✓ shoudl return the number of staking pack available (4ms)
  stake method
    ✓ should not stake a pack that does not exist (8ms)
    ✓ should not stake inside an admin pack by himself (13ms)
    ✓ should not stake below minStake of the pack (7ms)
    ✓ should not stake above maxStake of the pack (8ms)
    ✓ should not stake if staking contract cannot pay rewards (14ms)
    ✓ should not stake if user has not enough funds for staking contract (14ms)
    ✓ should not stake if user has not approved funds for staking contract (18ms)
    ✓ should not stake if user has any 11UP NFT (9ms)
    ✓ should stake Eleven tokens (266ms)
    ✓ should not allow staking in a pack that achieved max user threshold (14ms)
    ✓ should stake Eleven tokens on flex pack (280ms)
    ✓ should not stake twice 11UP tokens on flex pack (13ms)
  increaseStakeFlex method
    ✓ should increase the staking flex (184ms)
    ✓ should not increase the staking flex if user does not have enough balance (16ms)
    ✓ should not increase the staking flex if user has not approved the staking contract (11ms)
    ✓ should not update the staking flex if the staking contract does not have enough fund to pay reward (13ms)
    ✓ should not update the staking flex if the user does not have a staking in flex pack (7ms)
  unstakeFlex method
    ✓ should not allow user to unstake flex staking if the user never staked (7ms)
    ✓ should not allow user to unstake flex staking with more than it actually stakes (5ms)
    ✓ should not allow user to unstake flex staking with 0 amount (7ms)
    ✓ should not allow user to unstake flex staking if staking contract cannot pay rewards (12ms)
    ✓ should allow user to unstake flex staking (164ms)
    ✓ should allow user to fully exit a staking flex position (154ms)
  claimRewardFlex method
    ✓ should not allow user to claim flex rewards if not flex staking (7ms)
    ✓ should not allow user to claim flex rewards if staking contract cannot pay rewards (151ms)
    ✓ should allow user to claim flex rewards (118ms)
  updateStakeFlexRate method
    ✓ should not allow to update all flexible staking if caller is not allowed (6ms)
    ✓ should not allow to update all flexible staking if the x-y is not set properly - x < y (6ms)
    ✓ should not allow to update all flexible staking if the x-y is not set properly - x too high (11ms)
    ✓ should not allow to update all flexible staking if the x-y is not set properly - y too high (6ms)
    ✓ should update all flexible staking (48ms)
    ✓ should update all flexible staking but should not affect balances if done with no timestamp move (53ms)
  claimRewardLock method
    ✓ should not allow claiming rewards if user does not have lock staking (7ms)
    ✓ should not allow claiming rewards if user claims before ending of lock period (6ms)
    ✓ should not allow claiming rewards if staking contract cannot pay rewards (230ms)
    ✓ should allow user to claim rewards (41ms)
```

Tests

```
unstakeLock method
  ✓ should not allow the user to unstake a stake that does not exist (6ms)
  ✓ should not allow the user to unstake a stake before ending of lock period (6ms)
  ✓ should not allow the user to unstake a stake if staking contract cannot pay stake amount (7ms)
  ✓ should allow the user to unstake from a lock pack (197ms)
stakeFor method
  ✓ should not stakeFor if the caller is not an admin (7ms)
  ✓ should not stakeFor a pack id that does not exist (6ms)
  ✓ should not stakeFor if amount is below minStake (6ms)
  ✓ should not stakeFor if amount is below minStake (11ms)
  ✓ should not stakeFor in a pack that achieved max user threshold (6ms)
  ✓ should not stakeFor if the staking contract does not have enough balance (9ms)
  ✓ should stakeFor for a user (55ms)
  ✓ should stakeFor for a user in a flex pack (51ms)
  ✓ should not allow stakeFor for a user in a flex pack twice (8ms)
disableStakeFor method
  ✓ should allow the user to disable the stakeFor (37ms)
  ✓ should not allow the admin to stake if the function is not enabled (7ms)
updateElevenRewardAddress method
  ✓ should not update Eleven reward address if caller is not allowed (11ms)
  ✓ should not update Eleven reward address with Zero address (6ms)
  ✓ should update Eleven reward address (23ms)
updateElevenRewardAmount method
  ✓ should not update Eleven reward amount if caller is not allowed (9ms)
  ✓ should update Eleven reward amount (22ms)
updateElevenRewardPeriod method
  ✓ should not update Eleven reward period if caller is not allowed (9ms)
  ✓ should not update Eleven reward period with 0 (9ms)
  ✓ should update Eleven reward period (24ms)
getRewardForPlatformEleven method
  ✓ should not claim Eleven rewards if caller is not allowed (19ms)
  ✓ should not claim Eleven rewards if staking contract cannot pay rewards (8ms)
  ✓ should claim Eleven rewards (120ms)
```

Tests

```
  ElevenLottery
Duplicate definition of Transfer (Transfer(address,address,uint256,bytes), Transfer(address,address,uint256))
    Deployment
      ✓ should deploy the elevenLottery with a proper contract address (0ms)
      ✓ should not set the ElevenUpNFT address if the caller is not the owner (13ms)
      ✓ should set the ElevenUpNFT address (49ms)
    buyTickets method
      ✓ should revert if an user buy less than 1 ticket (7ms)
      ✓ should revert if an user did not approve the contract (9ms)
      ✓ should revert if an user don't have enough balance (12ms)
      ✓ should buy 2 tickets for an user (184ms)
      ✓ should not allow to add some bonus if the user did not approve the contract (18ms)
      ✓ should not allow to add some bonus if the user have note enough funds (13ms)
      ✓ should allow an user to add some bonus to the current lottery (103ms)
    announceWinners method
      ✓ should revert if there is less than 3 tickets sold (5ms)
      ✓ should revert if the lottery is still running (156ms)
      ✓ should revert if the contract don't have enough Link tokens (12ms)
      - should make a VRF request
      - should return random numbers
      ✓ should not allow to add bonus for a ended lottery (126ms)
    claimPrize methods
      ✓ should not allow an user who did not win the 1st prize to claim it (20ms)
      ✓ should allow the 1st prize winner to claim his prize (120ms)
      ✓ should not allow to claim the 1rst prize twice (22ms)
      ✓ should not allow an user who did not win the 2nd prize to claim it (30ms)
      ✓ should allow the 2nd prize winner to claim his prize (118ms)
      ✓ should not allow to claim the 2nd prize twice (14ms)
      ✓ should not allow an user who did not win the 3rd prize to claim it (24ms)
      ✓ should allow the 3rd prize winner to claim his prize (112ms)
      ✓ should not allow to claim the 3rd prize twice (20ms)

  LotteryFactory
Duplicate definition of Transfer (Transfer(address,address,uint256,bytes), Transfer(address,address,uint256))
    Deployment
      ✓ should deploy the elevenLottery with a proper contract address (0ms)
      ✓ should deploy the lottery Factory with a proper contract address (0ms)
    setLibraryAddress method
      ✓ should not set the libraryAddress if the caller is not the owner (11ms)
      ✓ should set the libraryAddress (22ms)
    createLottery method
      ✓ should not deploy a new lottery if the caller is the owner of the refferal Token (16ms)
      ✓ should deploy a new lottery for an user (58ms)
    cloned lottery statements
      ✓ should allow the cloned lottery to act like the original (43ms)
```

Disclosure

The report and the analysis described therein are created solely for the client and published with their consent. The scope of our review is limited to a review of the Solidity code and only the Solidity code that we indicate as part of the scope of our review in this report.

The Solidity language itself remains under development and is subject to unknown risks and defects. The review does not extend to the compiler layer, nor to other areas beyond Solidity that may present security risks. Cryptographic tokens are emerging technologies and carry high levels of technical risk and uncertainty.