PoolTogether - LootBox and MultipleWinners Strategy

Date	November 2020

1 Executive Summary

This report presents the results of our engagement with **PoolTogether** to review the **LootBox** and **MultipleWinners Prize Strategy**.

The review was conducted by Martin Ortner and Nicholas Ward over the course of nine person-days between November 23rd and November 27th, 2020.

2 Scope

Our review focused on the following repositories:

Repository	Commit#	Focus
pooltogether-pool-contracts	`c50de1d7af67a14990543af0c2d2703eca29f0e9`	MultipleWinners PrizeStrategy (details see Appendix)
loot-box	`2cbea5a85d53c555e28791df5b264d7b32779eea`	LootBox Implementation

list of files in scope can be found in the Appendix.

2.1 Objectives

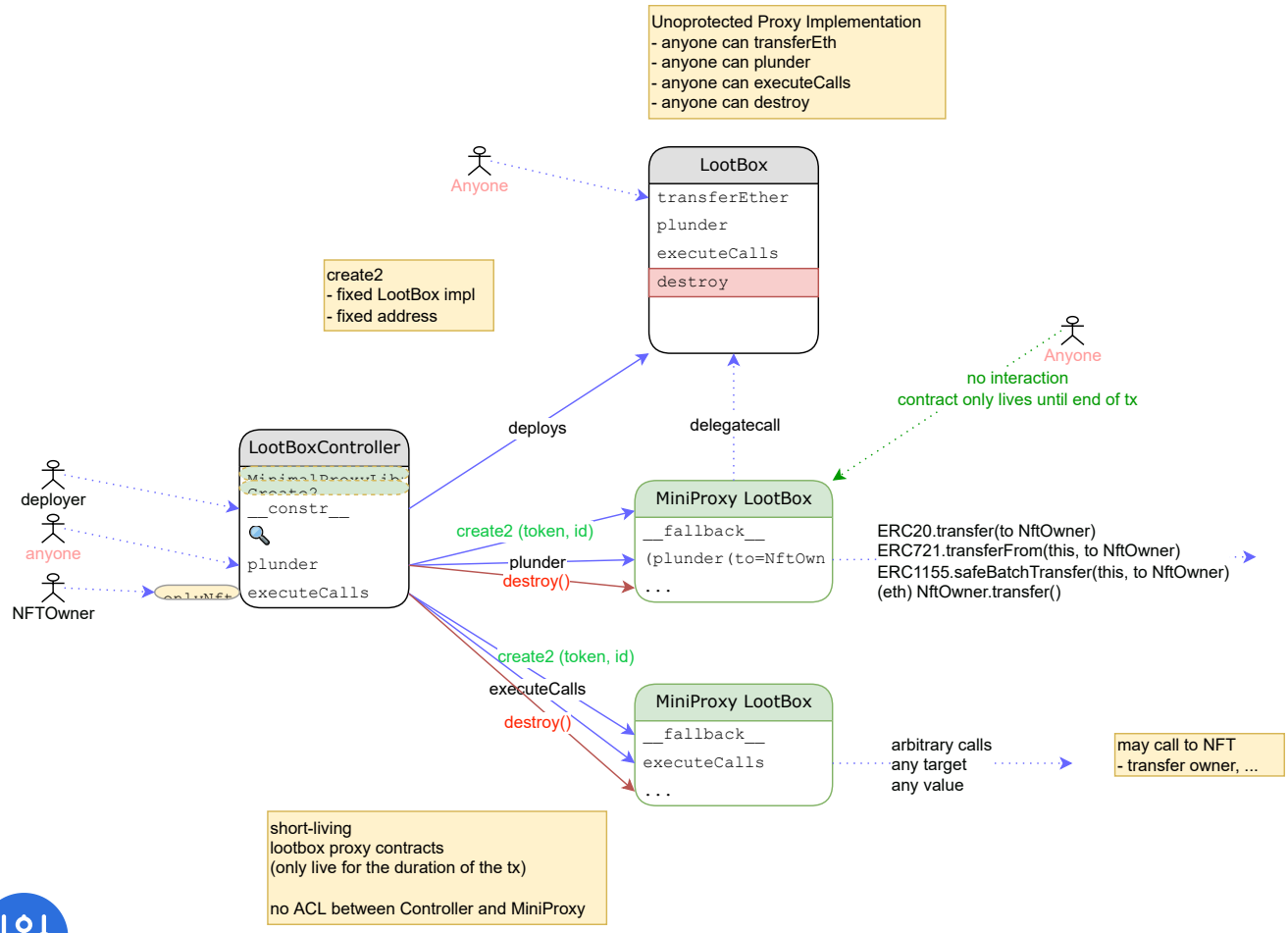
Together with the PoolTogether team, we identified the following priorities for our review:

1. Ensure that the system is implemented consistently with the intended functionality, and without unintended edge cases.
2. Identify known vulnerabilities particular to smart contract systems, as outlined in our Smart Contract Best Practices, and the Smart Contract Weakness Classification Registry.
3. Focus on the MultipleWinners Prize Strategy and LootBox.

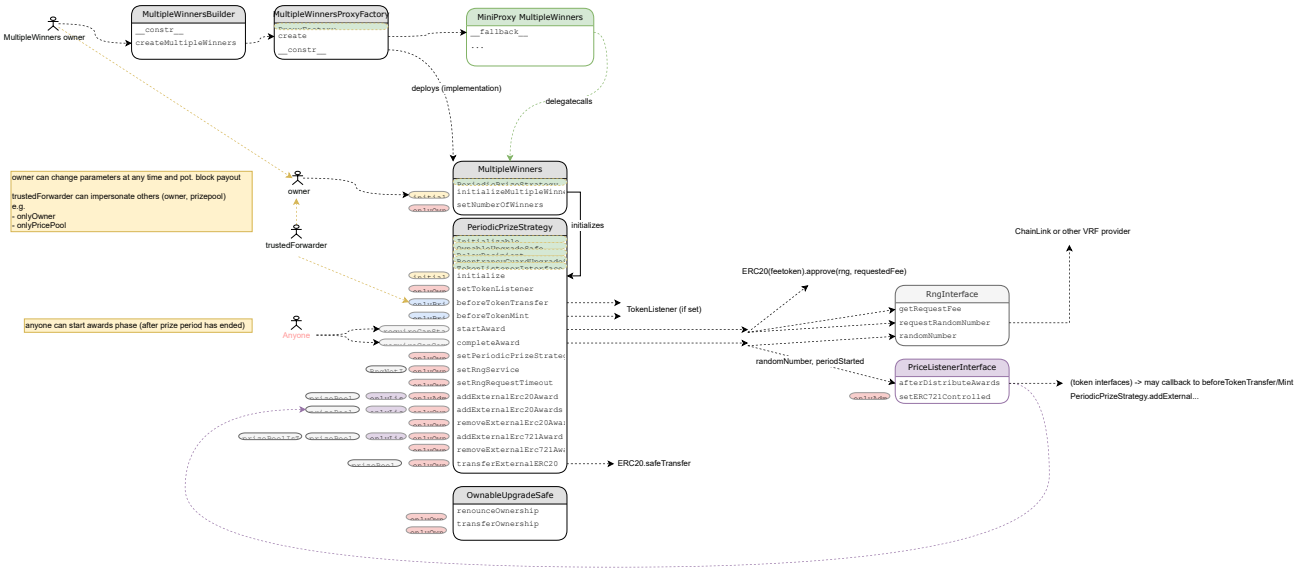
3 System Overview

The following diagrams were created during the audit and help to illustrate the connections and interdependencies for the contracts in scope, but should not serve as a substitute for documentation.

3.1 LootBox



3.2 MultipleWinners PeriodicPriceStrategy



4 Recommendations

4.1 Reconsider the use of proxy factories in the system if they are not needed

Recommendation

Various components of the system deploy proxy contracts even though there is no specific requirement for them to be proxies (non-upgradeable, no constant deployment address required).

The assumption is that the proxy factory is used to save gas on deployment, as only a minimal proxy stub is deployed that delegatecalls to the implementation. However, this additional delegatecall step will cause any interaction with the contract to consume an additional 700 gas. If eip-2929 (gas repricing) goes through, this per-use cost could increase significantly.

To reduce complexity and risks commonly associated with proxies (proxies not initialized in the same transaction, unprotected initialization, systemic complexity) as well as decreasing the interaction gas footprint it is recommended to deploy concrete contracts instead of proxies calling out to an implementation unless there is a real need for a proxy pattern.

Examples

ERC721 proxy

code/loot-box/contracts/ERC721ControlledFactory.sol:L25-L28

```
constructor () public {
  erc721ControlledInstance = new ERC721Controlled();
  erc721ControlledBytecode = MinimalProxyLibrary.minimalProxy(address(erc721
}
```

- `CounterfactualActionFactory.sol` is not directly in scope but shows the risks of using proxies with an unprotected `initialize` function that allows anyone to reinitialize the contract

code/pool/contracts/counterfactual-action/CounterfactualActionFactory.sol:L14-L19

```
function initialize(PrizePool _prizePool) external {
  require(address(_prizePool) != address(0), "CounterfactualActionFactory/pr
  depositor = new CounterfactualAction();
  prizePool = _prizePool;
}
```

4.2 Where possible, functions should accept a specific contract type rather than `address` parameters

Description

Rather than accepting address parameters and then casting to the known contract type, it is better to use the most specific type possible so the compiler can check for type safety. Typecasting inside the corpus of a function is unneeded when the type of the parameter is known beforehand.

Examples

There are more cases like this but here are some examples:

- `address erc721` -> `IERC721 erc721`

code/loot-box/contracts/LootBoxController.sol:L49-L62

```
function plunder(
  address erc721,
  uint256 tokenId,
  IERC20[] calldata erc20s,
  LootBox.WithdrawERC721[] calldata erc721s,
  LootBox.WithdrawERC1155[] calldata erc1155s
) external {
  address payable owner = payable(IERC721(erc721).ownerOf(tokenId));
  LootBox lootBoxAction = _createLootBox(erc721, tokenId);
  lootBoxAction.plunder(erc20s, erc721s, erc1155s, owner);
  lootBoxAction.destroy(owner);

  emit Plundered(erc721, tokenId, msg.sender);
}
```

code/loot-box/contracts/LootBoxController.sol:L21-L24

```
event Plundered(address indexed erc721, uint256 indexed tokenId, address inc

/// @notice Emitted when a Loot Box is executed
event Executed(address indexed erc721, uint256 indexed tokenId, address inde
```

code/loot-box/contracts/LootBoxController.sol:L70-L75

```
function executeCalls(
  address erc721,
  uint256 tokenId,
  LootBox.Call[] calldata calls
) external returns (bytes[] memory) {
  address payable owner = payable(IERC721(erc721).ownerOf(tokenId));
```

- `address indexed token` -> `IERC721 indexed token` etc. to avoid typecasts in `_withdraw*` functions

code/loot-box/contracts/LootBox.sol:L38-L44

```
event WithdrewERC20(address indexed token, uint256 amount);

/// @notice Emitted when an ERC721 token is withdrawn
event WithdrewERC721(address indexed token, uint256[] tokenIds);

/// @notice Emitted when an ERC1155 token is withdrawn
event WithdrewERC1155(address indexed token, uint256[] ids, uint256[] amount
```

- MultipleWinners

code/pool/contracts/builders/MultipleWinnersBuilder.sol:L43-L43

```
emit CreatedMultipleWinners(address(prizeStrategy), address(mw), numberOfWir
```

- `PeriodicPrizeStrategy` - `ticket`, `sponsorship`

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L141-L143

```
ticket = TicketInterface(_ticket);
rng = _rng;
sponsorship = IERC20(_sponsorship);
```

Recommendation

Review the complete codebase and, where possible, use more specific types instead of `address`.

4.3 Check codesize before adding listeners

Description

`PeriodicPrizeStrategy` allows the owner to set the `periodicPrizeStrategyListener` to any address. If the `periodicPrizeStrategyListener` is set to an address that is not `0x0` and contains no code, any call to `completeAward()` will revert because solidity checks that an external contract contains code before attempting to call any function on it. This allows the contract owner to effectively pause distribution of awards.

Recommendation

Require that any new `periodicPrizeStrategyListener` or `tokenListener` has `codesize > 0` and add an explicit `pause()` function if desired.

4.4 Stick to clear interface naming conventions

Description

Because the naming conventions used for contract interfaces are inconsistent, it can be difficult to visually distinguish interfaces from concrete contract implementations.

Examples

code/loot-box/contracts/external/pooltogether/PeriodicPrizeStrategyInterface.sol:L6-L6

```
interface PeriodicPrizeStrategyInterface {
```

code/loot-box/contracts/external/pooltogether/PeriodicPrizeStrategyListener.sol:L6-L6

```
interface PeriodicPrizeStrategyListener {
```

Recommendation

Consider indicating that a contract is an interface by prefixing the name with a capital `I` (recommended) or consistently stick to the interface naming used throughout the codebase.

4.5 LootBox - unnecessary payable

Description

`_createLootBox()` unnecessarily typecasts to `payable` before casting to a contract type.

Examples

code/loot-box/contracts/LootBoxController.sol:L90-L93

```
function _createLootBox(address erc721, uint256 tokenId) internal returns (L
  return LootBox(payable(Create2.deploy(0, _salt(erc721, tokenId), lootBoxAc
}
```

Recommendation

Remove the unnecessary `payable()`.

4.6 Clean up unused source-units in the repository

Description

Some source-units that were not in scope were found to contain trivial security issues. After reaching out to the client it was confirmed that they are not deployed anywhere on mainnet and might even be removed from the repository. It is therefore recommended to review the repository contents and remove any source files that are no longer necessary. Testing/Mockups should be moved to corresponding `./test` folders.

Examples

- unprotected initialize

code/pool/contracts/counterfactual-action/CounterfactualActionFactory.sol:L8-L19

```
contract CounterfactualActionFactory {

  CounterfactualAction public depositor;
  PrizePool public prizePool;

  function initialize(PrizePool _prizePool) external {
    require(address(_prizePool) != address(0), "CounterfactualActionFactory/
    depositor = new CounterfactualAction();
    prizePool = _prizePool;
  }
```

4.7 Rework the repository structure and clearly mark which 3rd party contracts have been modified

Description

For example, the `contracts/external/openzeppelin/ERC721.sol` has been modified from the original OpenZeppelin implementation. There is a comment about this in the contract code, but it is not immediately obvious from the folder structure that this differs from the original contract.

code/loot-box/contracts/external/openzeppelin/ERC721.sol:L14-L26

```
/**
 * @title ERC721 Non-Fungible Token Standard basic implementation
 * @dev see https://eips.ethereum.org/EIPS/eip-721.
 *
 * NOTE: This is a modified version of the OpenZeppelin ERC721 contract.  ERC
 *
 */
contract ERC721 is Context, Initializable, ERC165, IERC721, IERC721Metadata
    using SafeMath for uint256;
    using Address for address;
    using Strings for uint256;
```

5 Findings

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.

5.1 PeriodicPrizeStrategy - RNG failure can lock user funds

Critical

Description

To prevent manipulation of the `SortitionSumTree` after a requested random number enters the mempool, users are unable to withdraw funds while the strategy contract waits on a random number request between execution of `startAward()` and `completeAward()`.

If an rng request fails, however, there is no way to exit this locked state. After an rng request times out, only `startAward()` can be called, which will make another rng request and re-enter the same locked state. The rng provider can also not be updated while the contract is in this state. If the rng provider fails permanently, user funds are permanently locked.

Examples

- `requireNotLocked()` prevents transfers, deposits, or withdrawals when there is a pending award.

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L282-L285

```
function beforeTokenTransfer(address from, address to, uint256 amount, addre
  if (controlledToken == address(ticket)) {
    _requireNotLocked();
  }
```

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L528-L531

```
function _requireNotLocked() internal view {
  uint256 currentBlock = _currentBlock();
  require(rngRequest.lockBlock == 0 || currentBlock < rngRequest.lockBlock,
}
```

- `setRngService()` reverts if there is a pending or timed-out rng request

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L413-L414

```
function setRngService(RNGInterface rngService) external onlyOwner {
  require(!isRngRequested(), "PeriodicPrizeStrategy/rng-in-flight");
```

Recommendation

Instead of forcing the pending award phase to be re-entered in the event of an rng request time-out, provide an `exitAwardPhase()` function that ends the award phase without paying out the award. This will at least allow users to withdraw their funds in the event of a catastrophic failure of the rng service. It may also be prudent to allow the rng service to be updated in the event of an rng request time out.

5.2 LootBox - Unprotected selfdestruct in proxy implementation Critical

Description

When the `LootBoxController` is deployed, it also deploys an instance of `LootBox`. When someone calls `LootBoxController.plunder()` or `LootBoxController.executeCall()` the controller actually deploys a temporary proxy contract to a deterministic address using `create2`, then calls out to it to collect the loot.

The `LootBox` implementation contract is completely unprotected, exposing all its functionality to any actor on the blockchain. The most critical functionality is actually the `LootBox.destroy()` method that calls `selfdestruct()` on the implementation contract.

Therefore, an unauthenticated user can `selfdestruct` the `LootBox` proxy implementation and cause the complete system to become dysfunctional. As an effect, none of the AirDrops that were delivered based on this contract will be redeemable (Note: `create2` deploy address is calculated from the current contract address and salt). Funds may be lost.

Examples

code/loot-box/contracts/LootBoxController.sol:L28-L31

```
constructor () public {
  lootBoxActionInstance = new LootBox();
  lootBoxActionBytecode = MinimalProxyLibrary.minimalProxy(address(lootBoxAc
}
```

code/loot-box/contracts/LootBox.sol:L86-L90

```
/// @notice Destroys this contract using `selfdestruct`
/// @param to The address to send remaining Ether to
function destroy(address payable to) external {
  selfdestruct(to);
}
```

- not in scope but listed for completeness

code/pool/contracts/counterfactual-action/CounterfactualAction.sol:L7-L21

```
contract CounterfactualAction {
  function depositTo(address payable user, PrizePool prizePool, address outp
    IERC20 token = IERC20(prizePool.token());
    uint256 amount = token.balanceOf(address(this));
    token.approve(address(prizePool), amount);
    prizePool.depositTo(user, amount, output, referrer);
    selfdestruct(user);
  }

  function cancel(address payable user, PrizePool prizePool) external {
    IERC20 token = IERC20(prizePool.token());
    token.transfer(user, token.balanceOf(address(this)));
    selfdestruct(user);
  }
```

Recommendation

Enforce that only the deployer of the contract can call functionality in the contract. Make sure that nobody can destroy the implementation of proxy contracts.

5.3 Ticket duplication Major

Description

`Ticket._beforeTokenTransfer()` contains logic to update the `SortitionSumTree` from which prize winners are drawn. In the case where the `from` address is the same as the `to` address, tickets are duplicated rather than left unchanged. This allows any attacker to duplicate their tickets with no limit and virtually guarantee that they will win all awarded prizes.

code/pool/contracts/token/Ticket.sol:L71-L79

```
if (from != address(0)) {
  uint256 fromBalance = balanceOf(from).sub(amount);
  sortitionSumTrees.set(TREE_KEY, fromBalance, bytes32(uint256(from)));
}

if (to != address(0)) {
  uint256 toBalance = balanceOf(to).add(amount);
  sortitionSumTrees.set(TREE_KEY, toBalance, bytes32(uint256(to)));
}
```

This code was outside the scope of our review but was live on mainnet at the time the issue was disovered. We immediately made the client aware of the issue and an effort was made to mitigate the impact on the existing deployment.

5.4 PeriodicPriceStrategy - `trustedForwarder` can impersonate any `msg.sender` Major

Description

The `trustedForwarder` undermines the trust assumptions in the system. For example, one would assume that the access control modifier `onlyPrizePool` would only allow the configured `PrizePool` to call certain methods. However, in reality, the `trustedForwarder` can assume this position as well. The same is true for the `onlyOwnerOrListener` modifier. One would assume `msg.sender` must either be `periodicPrizeStrategyListener` or `owner` (the initial deployer) while the `trustedForwarder` can assume any of the administrative roles.

The centralization of power to allow one account to impersonate other components and roles (`owner`, `listener`, `prizePool`) in the system is a concern by itself and may give users pause when deciding whether to trust the contract system. The fact that the `trustedForwarder` can spoof events for any `msg.sender` may also make it hard to keep an accurate log trail of events in case of a security incident.

Note: The same functionality seems to be used in `ControlledToken` and other contracts which allows the `trustedForwarder` to assume any tokenholder in `ERC20UpgradeSafe`. There is practically no guarantee to `ControlledToken` holders.

Note: The `trustedForwarder` / `msgSender()` pattern is used in multiple contracts, many of which are not in the scope of this assessment.

Examples

- access control modifiers that can be impersonated

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L588-L591

```
modifier onlyPrizePool() {
  require(_msgSender() == address(prizePool), "PeriodicPrizeStrategy/only-pr
  _;
}
```

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L565-L568

```
modifier onlyOwnerOrListener() {
  require(_msgSender() == owner() || _msgSender() == address(periodicPrizeSt
  _;
}
```

- event `msg.sender` that can be spoofed because the actual `msg.sender` can be `trustedForwarder`

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L164-L164

```
emit PrizePoolOpened(_msgSender(), prizePeriodStartedAt);
```

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L340-L340

```
emit PrizePoolAwardStarted(_msgSender(), address(prizePool), requestId, lock
```

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L356-L357

```
emit PrizePoolAwarded(_msgSender(), randomNumber);
emit PrizePoolOpened(_msgSender(), prizePeriodStartedAt);
```

- `_msgSender()` implementation allows the `trustedForwarder` to impersonate any `msg.sender` address

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L541-L551

```
/// @dev Provides information about the current execution context for GSN Meta
/// @return The payable address of the message sender
function _msgSender()
  internal
  override(BaseRelayRecipient, ContextUpgradeSafe)
  virtual
  view
  returns (address payable)
{
  return BaseRelayRecipient._msgSender();
}
```

```
// File: @opengsn/gsn/contracts/BaseRelayRecipient.sol

...

   /**
     * return the sender of this call.
     * if the call came through our trusted forwarder, return the original sen
     * otherwise, return `msg.sender`.
     * should be used in the contract anywhere instead of msg.sender
     */
    function _msgSender() internal override virtual view returns (address pa
        if (msg.data.length >= 24 && isTrustedForwarder(msg.sender)) {
            // At this point we know that the sender is a trusted forwarder,
            // so we trust that the last bytes of msg.data are the verified s
            // extract sender address from the end of msg.data
            assembly {
                ret := shr(96,calldataload(sub(calldatasize(),20)))
            }
        } else {
            return msg.sender;
        }
    }
```

Recommendation

Remove the `trustedForwarder` or restrict the type of actions the forwarder can perform and don't allow it to impersonate other components in the system. Make sure users understand the trust assumptions and who has what powers in the system. Make sure to keep an accurate log trail of who performed which action on whom's behalf.

5.5 Unpredictable behavior for users due to admin front running or general bad timing Major

Description

In a number of cases, administrators of contracts can update or upgrade things in the system without warning. This has the potential to violate a security goal of the system.

Specifically, privileged roles could use front running to make malicious changes just ahead of incoming transactions, or purely accidental negative effects could occur due to unfortunate timing of changes.

In general users of the system should have assurances about the behavior of the action they're about to take.

Examples

An administrator (deployer) of `MultipleWinners` can change the number of winners in the system without warning. This has the potential to violate a security goal of the system.

- admin can change the number of winners during a prize-draw period

code/pool/contracts/prize-strategy/multiple-winners/MultipleWinners.sol:L38-L42

```
function setNumberOfWinners(uint256 count) external onlyOwner {
  __numberOfWinners = count;

  emit NumberOfWinnersSet(count);
}
```

- `PeriodicPriceStrategy` - admin may switch-out RNG service at any time (when RNG is not in inflight or timed-out)

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L413-L418

```
function setRngService(RNGInterface rngService) external onlyOwner {
  require(!isRngRequested(), "PeriodicPrizeStrategy/rng-in-flight");

  rng = rngService;
  emit RngServiceUpdated(address(rngService));
}
```

- `PeriodicPriceStrategy` - admin can effectively disable the rng request timeout by setting a high value during a prize-draw (e.g. to indefinitely block payouts)

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L420-L422

```
function setRngRequestTimeout(uint32 _rngRequestTimeout) external onlyOwner
  _setRngRequestTimeout(_rngRequestTimeout);
}
```

- `PeriodicPriceStrategy` - admin may set new tokenListener which might intentionally block token-transfers

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L175-L179

```
function setTokenListener(TokenListenerInterface _tokenListener) external or
  tokenListener = _tokenListener;

  emit TokenListenerUpdated(address(tokenListener));
}
```

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L360-L364

```
function setPeriodicPrizeStrategyListener(address _periodicPrizeStrategyList
  periodicPrizeStrategyListener = PeriodicPrizeStrategyListener(_periodicPri

  emit PeriodicPrizeStrategyListenerSet(_periodicPrizeStrategyListener);
}
```

- out of scope but mentioned as a relevant example: `PrizePool` owner can set new `PrizeStrategy` at any time

code/pool/contracts/prize-pool/PrizePool.sol:L1003-L1008

```
/// @notice Sets the prize strategy of the prize pool.  Only callable by the c
/// @param _prizeStrategy The new prize strategy
function setPrizeStrategy(address _prizeStrategy) external override onlyOwne
  _setPrizeStrategy(TokenListenerInterface(_prizeStrategy));
}
```

- a malicious admin may remove all external ERC20/ERC721 token awards prior to the user claiming them (admin front-running opportunity)

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L461-L464

```
function removeExternalErc20Award(address _externalErc20, address _prevExter
  externalErc20s.removeAddress(_prevExternalErc20, _externalErc20);
  emit ExternalErc20AwardRemoved(_externalErc20);
}
```

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L506-L510

```
function removeExternalErc721Award(address _externalErc721, address _prevExt
  externalErc721s.removeAddress(_prevExternalErc721, _externalErc721);
  delete externalErc721TokenIds[_externalErc721];
  emit ExternalErc721AwardRemoved(_externalErc721);
}
```

- the `PeriodicPrizeStrategy` `owner` (also see concerns outlined in issue 5.4) can transfer external ERC20 at any time to avoid them being awarded to users. there is no guarantee to the user.

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L517-L526

```
function transferExternalERC20(
  address to,
  address externalToken,
  uint256 amount
)
  external
  onlyOwner
{
  prizePool.transferExternalERC20(to, externalToken, amount);
}
```

Recommendation

The underlying issue is that users of the system can't be sure what the behavior of a function call will be, and this is because the behavior can change at any time.

We recommend giving the user advance notice of changes with a time lock. For example, make all system-parameter and upgrades require two steps with a mandatory time window between them. The first step merely broadcasts to users that a particular change is coming, and the second step commits that change after a suitable waiting period. This allows users that do not accept the change to withdraw immediately.

5.6 PeriodicPriceStrategy - `addExternalErc721Award` duplicate or invalid tokenIds may block award phase Medium

Description

The prize-strategy owner (or a listener) can add `ERC721` token awards by calling `addExternalErc721Award` providing the `ERC721` token address and a list of `tokenIds` owned by the prizePool.

The method does not check if duplicate `tokenIds` or `tokenIds` that are not owned by the contract are provided. This may cause an exception when `_awardExternalErc721s` calls `prizePool.awardExternalERC721` to transfer an invalid or previously transferred token, blocking the award phase.

Note: An admin can recover from this situation by removing and re-adding `ERC721` token from the awards list.

Examples

- adding `tokenIds`

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L478-L499

```
/// @notice Adds an external ERC721 token as an additional prize that can be a
/// @dev Only the Prize-Strategy owner/creator can assign external tokens,
/// and they must be approved by the Prize-Pool
/// NOTE: The NFT must already be owned by the Prize-Pool
/// @param _externalErc721 The address of an ERC721 token to be awarded
/// @param _tokenIds An array of token IDs of the ERC721 to be awarded
function addExternalErc721Award(address _externalErc721, uint256[] calldata
  // require(_externalErc721.isContract(), "PeriodicPrizeStrategy/external-erc
  require(prizePool.canAwardExternal(_externalErc721), "PeriodicPrizeStrateg

  if (!externalErc721s.contains(_externalErc721)) {
    externalErc721s.addAddress(_externalErc721);
  }

  for (uint256 i = 0; i < _tokenIds.length; i++) {
    uint256 tokenId = _tokenIds[i];
    require(IERC721(_externalErc721).ownerOf(tokenId) == address(prizePool),
    externalErc721TokenIds[_externalErc721].push(tokenId);
  }

  emit ExternalErc721AwardAdded(_externalErc721, _tokenIds);
}
```

- awarding tokens

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L248-L263

```
/// @notice Awards all external ERC721 tokens to the given user.
/// The external tokens must be held by the PrizePool contract.
/// @dev The list of ERC721s is reset after every award
/// @param winner The user to transfer the tokens to
function _awardExternalErc721s(address winner) internal {
  address currentToken = externalErc721s.start();
  while (currentToken != address(0) && currentToken != externalErc721s.end()
    uint256 balance = IERC721(currentToken).balanceOf(address(prizePool));
    if (balance > 0) {
      prizePool.awardExternalERC721(winner, currentToken, externalErc721Toke
      delete externalErc721TokenIds[currentToken];
    }
    currentToken = externalErc721s.next(currentToken);
  }
  externalErc721s.clearAll();
}
```

- transferring the tokens

code/pool/contracts/prize-pool/PrizePool.sol:L582-L606

```
/// @notice Called by the prize strategy to award external ERC721 prizes
/// @dev Used to award any arbitrary NFTs held by the Prize Pool
/// @param to The address of the winner that receives the award
/// @param externalToken The address of the external NFT token being awarded
/// @param tokenIds An array of NFT Token IDs to be transferred
function awardExternalERC721(
  address to,
  address externalToken,
  uint256[] calldata tokenIds
)
  external override
  onlyPrizeStrategy
{
  require(_canAwardExternal(externalToken), "PrizePool/invalid-external-toke

  if (tokenIds.length == 0) {
    return;
  }

  for (uint256 i = 0; i < tokenIds.length; i++) {
    IERC721(externalToken).transferFrom(address(this), to, tokenIds[i]);
  }

  emit AwardedExternalERC721(to, externalToken, tokenIds);
}
```

Recommendation

Ensure that no duplicate token-ids were provided or skip over token-ids that are not owned by prize-pool (anymore).

5.7 PeriodicPrizeStrategy - Token with callback related warnings (`ERC777` a.o.) Medium

Description

This issue is highly dependent on the configuration of the system. If an admin decides to allow callback enabled token (e.g. `ERC20` compliant `ERC777` or other `ERC721` / `ERC20` extensions) as awards then one recipient may be able to

- block the payout for everyone by forcing a revert in the callback when accepting token awards
- use the callback to siphon gas, mint gas token, or similar activities

- potentially re-enter the `PrizeStrategy` contract in an attempt to manipulate the payout (e.g. by immediately withdrawing from the pool to manipulate the 2nd `ticket.draw()`)

Examples

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L252-L263

```
function _awardExternalErc721s(address winner) internal {
  address currentToken = externalErc721s.start();
  while (currentToken != address(0) && currentToken != externalErc721s.end()
    uint256 balance = IERC721(currentToken).balanceOf(address(prizePool));
    if (balance > 0) {
      prizePool.awardExternalERC721(winner, currentToken, externalErc721Toke
      delete externalErc721TokenIds[currentToken];
    }
    currentToken = externalErc721s.next(currentToken);
  }
  externalErc721s.clearAll();
}
```

Recommendation

It is highly recommended to not allow tokens with callback functionality into the system. Document and/or implement safeguards that disallow the use of callback enabled tokens. Consider implementing means for the “other winners” to withdraw their share of the rewards independently from others.

5.8 PeriodicPrizeStrategy - unbounded external tokens linked list may be used to force a gas DoS Medium

Description

The size of the linked list of ERC20/ERC721 token awards is not limited. This fact may be exploited by an administrative account by adding an excessive number of external token addresses.

The winning user might want to claim their win by calling `completeAward()` which fails in one of the

`distribute() -> _awardAllExternalTokens() -> _awardExternalErc20s/_awardExternalErc721s`

loops if too many token addresses are configured and gas consumption hits the block gas limit (or it just gets too expensive for the user to call).

Note: an admin can recover from this situation by removing items from the list.

Examples

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L436-L448

```
/// @notice Adds an external ERC20 token type as an additional prize that can
/// @dev Only the Prize-Strategy owner/creator can assign external tokens,
/// and they must be approved by the Prize-Pool
/// @param _externalErc20 The address of an ERC20 token to be awarded
function addExternalErc20Award(address _externalErc20) external onlyOwnerOrL
  _addExternalErc20Award(_externalErc20);
}

function _addExternalErc20Award(address _externalErc20) internal {
  require(prizePool.canAwardExternal(_externalErc20), "PeriodicPrizeStrategy
  externalErc20s.addAddress(_externalErc20);
  emit ExternalErc20AwardAdded(_externalErc20);
}
```

code/pool/contracts/utils/MappedSinglyLinkedList.sol:L46-L53

```
/// @param newAddress The address to shift to the front of the list
function addAddress(Mapping storage self, address newAddress) internal {
  require(newAddress != SENTINEL && newAddress != address(0), "Invalid addre
  require(self.addressMap[newAddress] == address(0), "Already added");
  self.addressMap[newAddress] = self.addressMap[SENTINEL];
  self.addressMap[SENTINEL] = newAddress;
  self.count = self.count + 1;
}
```

- awarding the tokens loops through the linked list of configured tokens

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L248-L263

```
/// @notice Awards all external ERC721 tokens to the given user.
/// The external tokens must be held by the PrizePool contract.
/// @dev The list of ERC721s is reset after every award
/// @param winner The user to transfer the tokens to
function _awardExternalErc721s(address winner) internal {
  address currentToken = externalErc721s.start();
  while (currentToken != address(0) && currentToken != externalErc721s.end()
    uint256 balance = IERC721(currentToken).balanceOf(address(prizePool));
    if (balance > 0) {
      prizePool.awardExternalERC721(winner, currentToken, externalErc721Toke
      delete externalErc721TokenIds[currentToken];
    }
    currentToken = externalErc721s.next(currentToken);
  }
  externalErc721s.clearAll();
}
```

Recommendation

Limit the number of tokens an admin can add. Consider implementing an interface that allows the user to claim tokens one-by-one or in user-configured batches.

5.9 MultipleWinners - `setNumberOfWinners` does not enforce `count>0` Medium

Description

The constructor of `MultipleWinners` enforces that the argument `_numberOfWinners > 0` while `setNumberOfWinners` does not. A careless or malicious admin might set `__numberOfWinners` to zero to cause the `distribute()` method to throw and not pay out any winners.

Examples

- enforced in the constructor

code/pool/contracts/prize-strategy/multiple-winners/MultipleWinners.sol:L34-L34

```
require(_numberOfWinners > 0, "MultipleWinners/num-gt-zero");
```

- not enforced when updating the value at a later stage

code/pool/contracts/prize-strategy/multiple-winners/MultipleWinners.sol:L38-L42

```
function setNumberOfWinners(uint256 count) external onlyOwner {
  __numberOfWinners = count;

  emit NumberOfWinnersSet(count);
}
```

Recommendation

Require that `numberOfWinners > 0`.

5.10 LootBox - `plunder` should disallow plundering to `address(0)` Medium

Description

Anyone can call `LootboxController.plunder()` to plunder on behalf of a `tokenId` owner. If a `LootBox` received an AirDrop but no `NFT` was issued to an owner (yet) this might open up an opportunity for a malicious actor to call `plunder()` in an attempt to burn the ETH and any airdropped tokens that allow transfers to `address(0)`.

Note:

- Depending on the token implementation, transfers may or may not revert if the `toAddress == address(0)`, while burning the `ETH` will succeed.
- This might allow anyone to forcefully burn received ETH that would otherwise be available to the future beneficiary
- If the airdrop and transfer of `LootBox` ownership are not done within one transaction, this might open up a front-running window that allows a third party to burn air-dropped ETH before it can be claimed by the owner.
    - consider one component issues the airdrop in one transaction (or block) and setting the owner in a later transaction (or block). The

`owner` is unset for a short duration of time which might allow anyone to burn `ETH` held by the `LootBox` proxy instance.

Examples

- `plunder()` receiving the `owner` of an `ERC721.tokenId`

code/loot-box/contracts/LootBoxController.sol:L49-L56

```
function plunder(
  address erc721,
  uint256 tokenId,
  IERC20[] calldata erc20s,
  LootBox.WithdrawERC721[] calldata erc721s,
  LootBox.WithdrawERC1155[] calldata erc1155s
) external {
  address payable owner = payable(IERC721(erc721).ownerOf(tokenId));
```

- The modified `ERC721` returns `address(0)` if the owner is not known

code/loot-box/contracts/external/openzeppelin/ERC721.sol:L102-L107

```
 * @dev See {IERC721-ownerOf}.
 */
function ownerOf(uint256 tokenId) public view override returns (address) {
    return _tokenOwners[tokenId];
}
```

- While `withdraw[ERC20|ERC721|ERC1155]` fail with `to == address(0)`, `transferEther()` succeeds and burns the eth by sending it to `address(0)`

code/loot-box/contracts/LootBox.sol:L74-L84

```
function plunder(
  IERC20[] memory erc20,
  WithdrawERC721[] memory erc721,
  WithdrawERC1155[] memory erc1155,
  address payable to
) external {
  _withdrawERC20(erc20, to);
  _withdrawERC721(erc721, to);
  _withdrawERC1155(erc1155, to);
  transferEther(to, address(this).balance);
}
```

Recommendation

Require that the destination address `to` in `plunder()` and `transferEther()` is not `address(0)`.

5.11 PeriodicPrizeStrategy - Inconsistent behavior between award-phase modifiers and view functions Minor

Description

The logic in the `canStartAward()` function is inconsistent with that of the `requireCanStartAward` modifier, and the logic in the `canCompleteAward()` function is inconsistent with that of the `requireCanCompleteAward` modifier. Neither of these view functions appear to be used elsewhere in the codebase, but the similarities between the function names and the corresponding modifiers is highly misleading.

Examples

- `canStartAward()` is inconsistent with `requireCanStartAward`

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L377-L379

```
function canStartAward() external view returns (bool) {
  return _isPrizePeriodOver() && !isRngRequested();
}
```

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L575-L579

```
modifier requireCanStartAward() {
  require(_isPrizePeriodOver(), "PeriodicPrizeStrategy/prize-period-not-over
  require(!isRngRequested() || isRngTimedOut(), "PeriodicPrizeStrategy/rng-a
  _;
}
```

- `canCompleteAward()` is inconsistent with `requireCanCompleteAward`

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L383-L385

```
function canCompleteAward() external view returns (bool) {
  return isRngRequested() && isRngCompleted();
}
```

code/pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol:L581-L586

```
modifier requireCanCompleteAward() {
  require(_isPrizePeriodOver(), "PeriodicPrizeStrategy/prize-period-not-over
  require(isRngRequested(), "PeriodicPrizeStrategy/rng-not-requested");
  require(isRngCompleted(), "PeriodicPrizeStrategy/rng-not-complete");
  _;
}
```

Recommendation

Make the logic consistent between the view functions and the modifiers of the same name or remove the functions.

5.12 MultipleWinners - Awards can be guaranteed with a set number of tickets Minor

Description

Because additional award drawings are distributed at a constant interval in the `SortitionSumTree` by `MultipleWinners._distribute()`, any user that holds a number of tickets `>= floor(totalSupply / __numberOfWinners)` can guarantee at least one award regardless of the initial drawing.

`MultipleWinners._distribute()`:

code/pool/contracts/prize-strategy/multiple-winners/MultipleWinners.sol:L59-L65

```
uint256 ticketSplit = totalSupply.div(__numberOfWinners);
uint256 nextRandom = randomNumber.add(ticketSplit);
// the other winners receive their prizeShares
for (uint256 winnerCount = 1; winnerCount < __numberOfWinners; winnerCount++
  winners[winnerCount] = ticket.draw(nextRandom);
  nextRandom = nextRandom.add(ticketSplit);
}
```

Recommendation

Do not distribute awards at fixed intervals from the initial drawing, but instead randomize the additional drawings as well.

5.13 MultipleWinners - Inconsistent behavior compared to SingleRandomWinner Minor

Description

The `MultipleWinners` strategy carries out award distribution to the zero address if `ticket.draw()` returns `address(0)` (indicating an error condition) while `SingleRandomWinner` does not.

Examples

- `SingleRandomWinner` silently skips award distribution if `ticket.draw()` returns `address(0)`.

code/pool/contracts/prize-strategy/single-random-winner/SingleRandomWinner.sol:L8-L17

```
contract SingleRandomWinner is PeriodicPrizeStrategy {
  function _distribute(uint256 randomNumber) internal override {
    uint256 prize = prizePool.captureAwardBalance();
    address winner = ticket.draw(randomNumber);
    if (winner != address(0)) {
      _awardTickets(winner, prize);
      _awardAllExternalTokens(winner);
    }
  }
}
```

- `MultipleWinners` still attempts to distribute awards if `ticket.draw()` returns `address(0)`. This may or may not succeed depending on the implementation of the tokens included in the `externalErc20s` and `externalErc721s` linked lists.

code/pool/contracts/prize-strategy/multiple-winners/MultipleWinners.sol:L48-L57

```
function _distribute(uint256 randomNumber) internal override {
  uint256 prize = prizePool.captureAwardBalance();

  // main winner gets all external tokens
  address mainWinner = ticket.draw(randomNumber);
  _awardAllExternalTokens(mainWinner);

  address[] memory winners = new address[](__numberOfWinners);
  winners[0] = mainWinner;
```

Recommendation

Implement consistent behavior. Avoid hiding error conditions and consider throwing an exception instead.

5.14 Initialize implementations for proxy contracts and protect initialization methods Minor

Description

Any situation where the implementation of proxy contracts can be initialized by third parties should be avoided. This can be the case if the `initialize` function is unprotected or not initialized immediately after deployment.

Since the implementation contract is not meant to be used directly without a proxy delegate-calling to it, it is recommended to protect the initialization method of the implementation by initializing on deployment.

This affects all proxy implementations (the delegatecall target contract) deployed in the system.

Examples

- The implementation for `MultipleWinners` is not initialized. Even though not directly used by the system it may be initialized by a third party.

code/pool/contracts/prize-strategy/multiple-winners/MultipleWinnersProxyFactory.sol:L13-L15

```
constructor () public {
  instance = new MultipleWinners();
}
```

- The deployed `ERC721Contract` is not initialized.

code/loot-box/contracts/ERC721ControlledFactory.sol:L25-L29

```
constructor () public {
  erc721ControlledInstance = new ERC721Controlled();
  erc721ControlledBytecode = MinimalProxyLibrary.minimalProxy(address(erc721
}
```

- The deployed `LootBox` is not initialized.

code/loot-box/contracts/LootBoxController.sol:L28-L31

```
constructor () public {
  lootBoxActionInstance = new LootBox();
  lootBoxActionBytecode = MinimalProxyLibrary.minimalProxy(address(lootBoxAc
}
```

Recommendation

Initialize unprotected implementation contracts in the implementation's constructor. Protect initialization methods from being called by unauthorized parties or ensure that deployment of the proxy and initialization is performed in the same transaction.

5.15 LootBox - `transferEther` should be `internal` Minor

Description

`LootBox.transferEther()` can be `internal` as it is only called from `LootBox.plunder()` and the `LootBox` (proxy) instances are generally very short-living (created and destroyed within one transaction).

Examples

code/loot-box/contracts/LootBox.sol:L63-L67

```
function transferEther(address payable to, uint256 amount) public {
  to.transfer(amount);

  emit TransferredEther(to, amount);
}
```

Recommendation

Restrict `transferEther()`'s visibility to `internal`.

5.16 LootBox - `executeCalls` can be misused to relay calls Minor

Description

`LootBox` is deployed with `LootBoxController` and serves as the implementation for individual `create2` lootbox proxy contracts. None of the methods of the `LootBox` implementation contract are access restricted. A malicious actor may therefore use the `executeCalls()` method to relay arbitrary calls to other contracts on the blockchain in an attempt to disguise the origin or misuse the reputation of the `LootBox` contract (as it belongs to the PoolTogether project).

Note: allows non-value and value calls (deposits can be forces via `selfdestruct`)

Examples

code/loot-box/contracts/LootBox.sol:L52-L58

```
function executeCalls(Call[] calldata calls) external returns (bytes[] memor
  bytes[] memory response = new bytes[](calls.length);
  for (uint256 i = 0; i < calls.length; i++) {
    response[i] = _executeCall(calls[i].to, calls[i].value, calls[i].data);
  }
  return response;
}
```

Recommendation

Restrict access to call forwarding functionality to trusted entities. Consider implementing the `Ownable` pattern allowing access to functionality to the owner only.

Appendix 1 - Files in Scope

This audit covered the following files:

pooltogether-pool-contracts (`c50de1d7af67a14990543af0c2d2703eca29f0e9`)

File Name	SHA-1 Hash
pool/contracts/prize-strategy/multiple-winners/MultipleWinnersProxyFactory.sol	8d0a970b6fce2716d076aa46f9bfcbd6b0c2a301
pool/contracts/prize-strategy/multiple-winners/MultipleWinners.sol	cf315d981c290aee434f3569bc2ab5f0f9e777d9
pool/contracts/prize-strategy/PeriodicPrizeStrategy.sol	2bfa0f797852fde4f3af5d36fc2f963fad4f0d18
pool/contracts/prize-strategy/PeriodicPrizeStrategyListener.sol	f44f45d49e9b3c30abb71eaec1c144bd6480a0a5

File Name	SHA-1 Hash
pool/contracts/builders/MultipleWinnersBuilder.sol	a5d6e1cccc9b68882048161739efed3d3ad034a6

loot-box (`2cbea5a85d53c555e28791df5b264d7b32779eea`)

File Name	SHA-1 Hash
loot-box/contracts/LootBox.sol	c4d71463e66cd19dc6cc516753593201b69b961f
loot-box/contracts/LootBoxController.sol	432212c2ca7c857a8c5ddf3444e7f432f7f3dd0f

Appendix 2 - Artifacts

This section contains some of the artifacts generated during our review by automated tools, the test suite, etc. If any issues or recommendations were identified by the output presented here, they have been addressed in the appropriate section above.

A.2.1 Metrics Report

The following report outlines smart contract capabilities and gives an overview of the size of the audit: Solidity Metrics Report

The report was generated using vscode-solidity-metrics.

A.2.2 Surya

Surya is a utility tool for smart contract systems. It provides a number of visual outputs and information about the structure of smart contracts. It also supports querying the function call graph in multiple ways to aid in the manual inspection and control flow analysis of contracts.

Below is a complete list of functions with their visibility and modifiers:

ntracts Description Table

Contract	Type	Bases		
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
MultipleWinnersProxyFactory	Implementation	ProxyFactory		
L		Public ❗	🛑	NO❗
L	create	External ❗	🛑	NO❗
MultipleWinners	Implementation	PeriodicPrizeStrategy		
L	initializeMultipleWinners	Public ❗	🛑	initializer
L	setNumberOfWinners	External ❗	🛑	onlyOwner
L	numberOfWinners	External ❗		NO❗
L	_distribute	Internal 🔒	🛑	
PeriodicPrizeStrategy	Implementation	Initializable, OwnableUpgradeSafe, RelayRecipient, ReentrancyGuardUpgradeSafe, TokenListenerInterface		
L	initialize	Public ❗	🛑	initializer
L	_distribute	Internal 🔒	🛑	

Contract	Type	Bases		
L	currentPrize	Public ❗		NO❗
L	setTokenListener	External ❗	🛑	onlyOwner
L	estimateRemainingBlocksToPrize	Public ❗		NO❗
L	prizePeriodRemainingSeconds	External ❗		NO❗
L	_prizePeriodRemainingSeconds	Internal 🔒		
L	isPrizePeriodOver	External ❗		NO❗
L	_isPrizePeriodOver	Internal 🔒		
L	_awardTickets	Internal 🔒	🛑	
L	_awardAllExternalTokens	Internal 🔒	🛑	
L	_awardExternalErc20s	Internal 🔒	🛑	
L	_awardExternalErc721s	Internal 🔒	🛑	
L	prizePeriodEndAt	External ❗		NO❗

Contract	Type	Bases		
L	_prizePeriodEndAt	Internal 🔒		
L	beforeTokenTransfer	External ❗	🛑	onlyPrizePool
L	beforeTokenMint	External ❗	🛑	onlyPrizePool
L	_currentTime	Internal 🔒		
L	_currentBlock	Internal 🔒		
L	startAward	External ❗	🛑	requireCanStartAward
L	completeAward	External ❗	🛑	requireCanCompleteAward
L	setPeriodicPrizeStrategyListener	External ❗	🛑	onlyOwner
L	_calculateNextPrizePeriodStartTime	Internal 🔒		
L	calculateNextPrizePeriodStartTime	External ❗		NO❗
L	canStartAward	External ❗		NO❗

Contract	Type	Bases		
L	canCompleteAward	External ❗		NO❗
L	isRngRequested	Public ❗		NO❗
L	isRngCompleted	Public ❗		NO❗
L	getLastRngLockBlock	External ❗		NO❗
L	getLastRngRequestId	External ❗		NO❗
L	setRngService	External ❗	🛑	onlyOwner
L	setRngRequestTimeout	External ❗	🛑	onlyOwner
L	_setRngRequestTimeout	Internal 🔒	🛑	
L	getExternalErc20Awards	External ❗		NO❗
L	addExternalErc20Award	External ❗	🛑	onlyOwnerOrListener
L	_addExternalErc20Award	Internal 🔒	🛑	

Contract	Type	Bases		
L	addExternalErc20Awards	External ❗	🛑	onlyOwnerOrListener
L	removeExternalErc20Award	External ❗	🛑	onlyOwner
L	getExternalErc721Awards	External ❗		NO❗
L	getExternalErc721AwardTokenIds	External ❗		NO❗
L	addExternalErc721Award	External ❗	🛑	onlyOwnerOrListener
L	removeExternalErc721Award	External ❗	🛑	onlyOwner
L	transferExternalERC20	External ❗	🛑	onlyOwner
L	_requireNotLocked	Internal 🔒		
L	isRngTimedOut	Public ❗		NO❗
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		

Contract	Type	Bases		
PeriodicPrizeStrategyListener	Interface			
L	afterDistributeAwards	External ❗	🛑	NO❗
MultipleWinnersBuilder	Implementation			
L		Public ❗	🛑	NO❗
L	createMultipleWinners	External ❗	🛑	NO❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Contracts Description Table

Contract	Type	Bases		
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
LootBox	Implementation			
L	executeCalls	External ❗	🛑	NO❗
L	transferEther	Public ❗	🛑	NO❗
L	plunder	External ❗	🛑	NO❗

Contract	Type	Bases		
L	destroy	External ❗	🛑	NO❗
L	_executeCall	Internal 🔒	🛑	
L	_withdrawERC20	Internal 🔒	🛑	
L	_withdrawERC721	Internal 🔒	🛑	
L	_withdrawERC1155	Internal 🔒	🛑	
LootBoxController	Implementation			
L		Public ❗	🛑	NO❗
L	computeAddress	External ❗		NO❗
L	plunder	External ❗	🛑	NO❗
L	executeCalls	External ❗	🛑	NO❗
L	_createLootBox	Internal 🔒	🛑	
L	_salt	Internal 🔒		

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Appendix 3 - Disclosure

ConsenSys Diligence ("CD") typically receives compensation from one or more clients (the "Clients") for performing the analysis contained in these

reports (the "Reports"). The Reports may be distributed through other means, including via ConsenSys publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty.

CD makes the Reports available to parties other than the Clients (i.e., "third parties") – on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

LINKS TO OTHER WEB SITES FROM THIS WEB SITE You may, through hypertext or other computer links, gain access to web sites operated by persons other than ConsenSys and CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that ConsenSys and CD are not responsible for

the content or operation of such Web sites, and that ConsenSys and CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that ConsenSys and CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. ConsenSys and CD assumes no responsibility for the use of third party software on the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

TIMELINESS OF CONTENT The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice. Unless indicated otherwise, by ConsenSys and CD.

CONTACT

CAREERS

PRIVACY POLICY

Email*

e-mail address

→

POWERED BY CONSENSYS