SECURITY AUDIT OF

THE PARALLEL SMART CONTRACTS

Public Report

Dec 09, 2021

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
ERC20	ERC20 (BEP20 in Binance Smart Chain or *x*RP20 in other chains) tokens are blockchain-based assets that have value and can be sent and received. The primary difference with the primary coin is that instead of running on their own blockchain, ERC20 tokens are issued on a network that supports smart contracts such as Ethereum or Binance Smart Chain.

EXECUTIVE SUMMARY

This Security Audit Report prepared by Verichains Lab on Dec 09, 2021. We would like to thank the The Parallel for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the The Parallel Smart Contracts. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified some vulnerable issues in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About The Parallel Smart Contracts

The Parallel is a virtual world universe with a real-world simulation game system where players can fully create materials in the world and participate in monetization, entertainment, and networking activities.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of The Parallel Smart Contracts. It was conducted on commit 08f3fa789b50c76a1712b63c1bf79262ac4bc693 from git repository *https://github.com/theparalleldotio/*.

1.3. Audit methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Integer Overflow and Underflow
- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Explicit visibility of functions state variables (external, internal, private and public)
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

The initial review was conducted on Nov 30, 2021 and a total effort of 6 working days was dedicated to identifying and documenting security issues in the code base of the The Parallel Smart Contracts.

2.2. Contract code

The The Parallel Smart Contracts was written in Solidity language, with the required version to be ^0.8.2. The source code was written based on OpenZeppelin's library.

The provided source codes consist of three contracts which inherit some contracts from OpenZeppelin.

2.2.1. Parallel token contract

The parallel token is an ERC20 token contract. Besides the default ERC20 functions, the contract implements additional functions which avoid Whale trading to control the price of tokens in the IDO time.

In addition, the contract inherits Pausable contract. So the owner of the contract can pause or unpause the contract.

Table 2 lists some properties of the audited ParallelToken contract (as of the report writing time).

PROPERTY	VALUE
Name	Parallel Token
Symbol	PRL
Decimals	18
Total Supply	1,000,000,000 (x10^{18}) Note: the number of decimals is 18, so the total representation token will be 1,000,000,000 or 1 billion.

Table 2. The ParallelToken contract properties

2.2.2. PowerStone token contract

PowerStone is an ERC20 token contract. Currently, the contract is inheriting upgradeable contracts. Therefore, the contract can be upgradeable or changed in the future.

In addition, the contract implements the mint public function which allows owner contract can mint unlimited token. Therefore, the totalSupply value can be changed by this function.

Table 3 lists some properties of the audited PowerStone contract (as of the report writing time).

PROPERTY	VALUE
Name	PowerStone
Symbol	PS
Decimals	18

Table 3. The PowerStone contract properties

2.2.3. TokenDistribution contract

The The Parallel team uses this contract to release the token. The contract releases tokens following 8 packages. They are Dumpy, Advisors, Seed round, Private round, Public round, Team, Ecosystem + Marketing and Game rewards.

In addition, the owner contract can addLiquidity for pair of the token and busd token in the pancakeswap.

2.3. Findings

During the audit process, the audit team found some vulnerabilities in the given version of The Parallel Smart Contracts.

The Parallel fixed the code according to Verichain's draft report in commit a2b81355644a61fbf69bfea7a79935a35afaa395.

2.3.1. TokenDistribution.sol - Unsafe using transfer, transferFrom method through IERC20 interface HIGH

Currently, there are some functions in the contract that use transfer, transferFrom method to call functions from the token contract. The contract doesn't point exactly which the token contract is using. So we can't ensure that the transfer function of the token contract works exactly as expected.

For instance, the transfer function can return false with the function call failure instead of returning true or revert like ERC20 Oppenzepplin. With claim logic, the user doesn't receive anything while the amount claimed still adds.

```
function claim() public {
        require(!suspended[msg.sender], "Must be not suspended");
        uint256 amount = getClaimAmount(msg.sender);
        require(amount > 0, "Must be > 0");
        investors[msg.sender].claimed = investors[msg.sender].claimed.add(
            amount
        );
        token.transfer(msg.sender, amount);
    }
```

Snippet 1. TokenDistribution.sol Unsafe using `transfer` method in `claim` function

There are four functions that are using them. They are allocationFor, claim, useFund and withdraw functions.

RECOMMENDATION

We suggest using SafeERC20 library for IERC20 and changing all transfer, transferFrom method using in the contract to safeTransfer, safeTransferFrom which is declared in SafeERC20 library to ensure that there is no issue when transferring tokens.

UPDATES

- *Dec 09, 2021*: This recommendation has been acknowledged and fixed by the The Parallel team.

2.3.2. TokenDistribution.sol - Unsafe using addLiquidity with zero value of amountAMin and amountBMin HIGH

In the addLiquidity function, the pancakeRouter.addLiquidity method use zero value with amountAMin and amountBMin parameters. Both parameters are the minimum amount of token to provide (slippage impact). So if a whale account calls a huge transaction before addLiquidity function call, the minimum amount will be changed to a huge amount. After this function call, the whale can sell a lot of tokens to get money from the liquidity we just added.

```
function addLiquidity(uint256 amountBUSD)
        public
        onlyRole(ADDLIQUIDITY_ROLE)
    {
```

```
    (uint256 amountA, uint256 amountB, ) = pancakeRouter.addLiquidity(
      address(token),
      busd,
      1_000_000_000 * 1e18,
      amountBUSD,
      0,
      0,
      address(this),
      block.timestamp
    );
    require(amountB == amountBUSD, "Must exact BUSD Liquidity");
    require(capped.sub(allotted) >= amountA, "Must be have liquidity fund");
    allotted = allotted.add(amountA);
  }
```

RECOMMENDATION

We suggest calculating the rate of both tokens and setting accepted values to amountAMin and amountBMin parameters to reduce slippage impact.

UPDATES

- *Dec 09, 2021*: This recommendation has been acknowledged and fixed by the The Parallel team.

2.3.3. TokenDistribution.sol - Conflict require statements in addMoreAllocation function and setupLiquidity function LOW

In the addMoreAllocation function, the require statements allow for allotted.add(amount) <= capped but in the setupLiquidity function capped.sub(allotted) > 2_000_000 * 1e18. If the owner of contract add alloted too much, the contract can't setupLiquidity for the first addLiquidity.

RECOMMENDATION

We suggest adding a boolean state variable named isAddedLiquidity and changing addMoreAllocation, setupLiquidity functions like the below code:

```
function addMoreAllocation(address investor, uint256 amount)
    public
    onlyRole(DEFAULT_ADMIN_ROLE)
  {
    if (!isAddedLiquidity){
            require(allotted.add(amount) <= (capped - 2_000_000 * 1e18)`,…
   "Full out");
        }
        else{
            require(allotted.add(amount) <= capped, "Full out");
        }
        require(investors[investor].packageId != 0, "Investor not found");
        investors[investor].total = investors[investor].total.add(amount);
        allotted = allotted.add(amount);
        emit MoreAllocation(investor, amount);
    }
```

Snippet 2. TokenDistribution.sol Recommend fixing in the `addMoreAllocation` function

```
function setupLiquidity() public onlyRole(DEFAULT_ADMIN_ROLE) {
        require(
            capped.sub(allotted) > 2_000_000 * 1e18,
            "Must be have liquidity"
        );
        require(timeRelease == 0, "Must be fresh liquidity");
        token.approve(address(pancakeRouter), 1e40);
        IERC20(busd).approve(address(pancakeRouter), 1e40);
        pancakeRouter.addLiquidity(
            address(token),
            busd,
            2_000_000 * 1e18,
            200_000 * 1e18,
            0,
            0,
            address(this),
            block.timestamp
        );
        isAddedLiquidity=true;
        allotted = allotted.add(2_000_000 * 1e18);
        timeRelease = block.timestamp;
    }
```

Snippet 3. TokenDistribution.sol Recommend fixing in the `setupLiquidity` function

UPDATES

- *Dec 09, 2021*: This recommendation has been acknowledged and fixed by the The Parallel team.

2.3.4. TokenDistribution.sol - Missing check value of _timePreriod variable in the constructor LOW

The timePeriod state variable is set by 30days value. But in the constructor, The timePeriod is updated. So the default value is useless. And timePeriod can be zero which causes an issue with the div operator in getClaimAmount function.

```
function getClaimAmount(address user) public view returns (uint256) {
        if (block.timestamp < timeRelease || timeRelease == 0) return 0;
        Investor memory investor = investors[user];
        if (investor.packageId == 0) return 0;
        Package memory pack = packages[investor.packageId];
        uint256 claimable = investor.total.mul(pack.unlockPercent).div(100);
        if (block.timestamp.sub(timeRelease) > pack.lockedTime) {
            uint256 unlockAmount = investor
                .total
                .sub(claimable)
                .mul(
                    (block.timestamp.sub(timeRelease).sub(pack.lockedTime)).div(
                        timePeriod
                    )
                )
                .div((pack.vestingTime).div(timePeriod));
            claimable = claimable.add(unlockAmount);
        }
        if (claimable > investor.total) {
            claimable = investor.total;
        }
        return claimable - investor.claimed;
    }
```

Snippet 4. TokenDistribution.sol Zero value causes an issues in `getClaimAmount` function

RECOMMENDATION

We suggest adding a require statement to check the value of timePeriod in the constructor like the below code:

```
constructor(
        address _token,
        address _busd,
        address _pancakeRouter,
        uint256 _timePeriod
    ) {
        _setupRole(DEFAULT_ADMIN_ROLE, msg.sender);
        _setupRole(ADDLIQUIDITY_ROLE, msg.sender);
        capped = 1_000_000_000 * (10**18);
        token = IERC20(_token);
        busd = _busd;
        _setupInitPackage();
        pancakeRouter = IPancakeRouter02(_pancakeRouter);
        require(_timePeriod>0,"timePeriod needs to be greater than 0…
");
        timePeriod = _timePeriod;
    }
```

Snippet 5. TokenDistribution.sol Recommend fixing in the constructor

UPDATES

- *Dec 09, 2021*: This recommendation has been acknowledged and fixed by the The Parallel team.

2.4. Additional notes and recommendations

2.4.1. TokenDistribution.sol - Unnecessary usage of SafeMath library in Solidity 0.8.0+ INFORMATIVE

All safe math usage in the contract are for overflow checking, solidity 0.8.0+ already do that by default, the only usage of safemath now is to have a custom revert message which isn't the case in the auditing contracts.

RECOMMENDATION

We suggest changing all methods from SafeMath library to normal arithmetic operator in the contract for readability and gas saving.

UPDATES

- *Dec 09, 2021*: This recommendation has been acknowledged by the The Parallel team.

2.4.2. TokenDistribution.sol - Change memory variable delare to storage variable declare for gas saving INFORMATIVE

In the getClaimAmount function, we found that there are 2 variables declared with memory keyword. These variables are only used to get value. Therefore, we recommend changing memory to storage keyword for these variables for gas saving.

```
function getClaimAmount(address user) public view returns (uint256) {
        if (block.timestamp < timeRelease || timeRelease == 0) return…
 0;
        Investor memory investor = investors[user];
        if (investor.packageId == 0) return 0;
        Package memory pack = packages[investor.packageId];
        uint256 claimable = investor.total.mul(pack.unlockPercent).di…
v(100);
        if (block.timestamp.sub(timeRelease) > pack.lockedTime) {
```

Snippet 6. TokenDistribution.sol Variables should change declaring in `getClaimAmount` function

RECOMMENDATION

We suggest changing 2 variables declaration like the below code:

```
function getClaimAmount(address user) public view returns (uint256) {
        if (block.timestamp < timeRelease || timeRelease == 0) return…
 0;
        Investor storage investor = investors[user];
        if (investor.packageId == 0) return 0;
        Package storage pack = packages[investor.packageId];
        uint256 claimable = investor.total.mul(pack.unlockPercent).di…
v(100);
        if (block.timestamp.sub(timeRelease) > pack.lockedTime) {
```

Snippet 7. TokenDistribution.sol Recommend fixing in `getClaimAmount` function

UPDATES

- *Dec 09, 2021*: This recommendation has been acknowledged and fixed by the The Parallel team.

APPENDIX

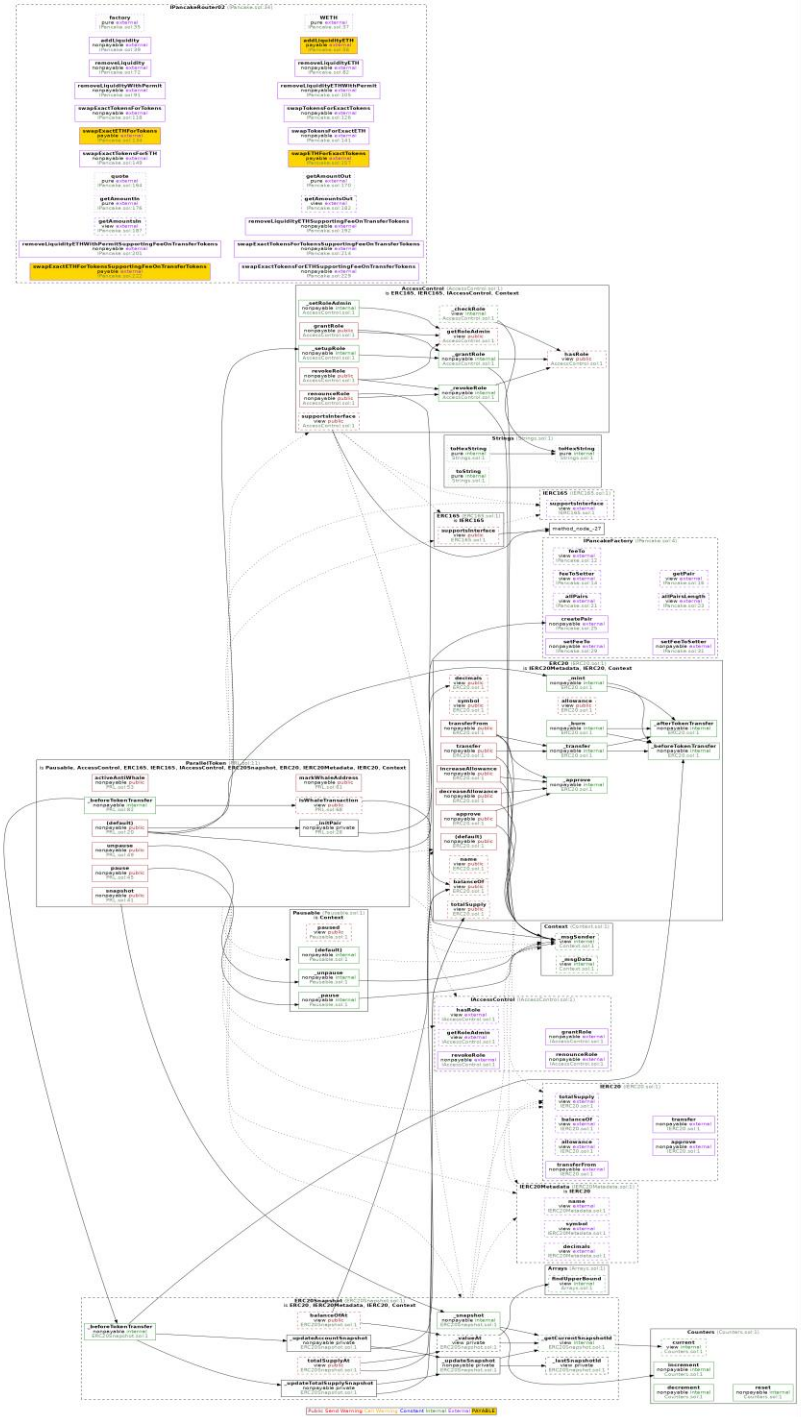
IPancakeRouter02 (IPancake.sol:34)
factory pure external IPancake.sol:35
WETH pure external IPancake.sol:37
addLiquidity nonpayable external IPancake.sol:39
addLiquidityETH payable external IPancake.sol:59
removeLiquidity nonpayable external IPancake.sol:72
removeLiquidityETH nonpayable external IPancake.sol:82
removeLiquidityWithPermit nonpayable external IPancake.sol:91
removeLiquidityETHWithPermit nonpayable external IPancake.sol:105
swapExactTokensForTokens nonpayable external IPancake.sol:118
swapTokensForExactTokens nonpayable external IPancake.sol:126
swapExactETHForTokens payable external IPancake.sol:134
swapTokensForExactETH nonpayable external IPancake.sol:141
swapExactTokensForETH nonpayable external IPancake.sol:149
swapETHForExactTokens payable external IPancake.sol:157
quote pure external IPancake.sol:164
getAmountOut pure external IPancake.sol:170
getAmountIn pure external IPancake.sol:176
getAmountsOut view external IPancake.sol:182
getAmountsIn view external IPancake.sol:187
removeLiquidityETHSupportingFeeOnTransferTokens nonpayable external IPancake.sol:192
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens nonpayable external IPancake.sol:201
swapExactTokensForTokensSupportingFeeOnTransferTokens nonpayable external IPancake.sol:214
swapExactETHForTokensSupportingFeeOnTransferTokens payable external IPancake.sol:222
swapExactTokensForETHSupportingFeeOnTransferTokens nonpayable external IPancake.sol:229
AccessControl (AccessControl.sol:1)
is ERC165, IERC165, IAccessControl, Context
_setRoleAdmin nonpayable internal AccessControl.sol:1
grantRole nonpayable public AccessControl.sol:1
_setupRole nonpayable internal AccessControl.sol:1
revokeRole nonpayable public AccessControl.sol:1
renounceRole nonpayable public AccessControl.sol:1
supportsInterface view public AccessControl.sol:1
_checkRole view internal AccessControl.sol:1
getRoleAdmin view public AccessControl.sol:1
_grantRole nonpayable internal AccessControl.sol:1
_revokeRole nonpayable internal AccessControl.sol:1
hasRole view public AccessControl.sol:1
Strings (Strings.sol:1)
toHexString pure internal Strings.sol:1
toString pure internal Strings.sol:1
toHexString pure internal Strings.sol:1
IERC165 (IERC165.sol:1)
supportsInterface view external IERC165.sol:1
ERC165 (ERC165.sol:1)
is IERC165
supportsInterface view public ERC165.sol:1
method_node_-27
IPancakeFactory (IPancake.sol:4)
feeTo view external IPancake.sol:12
feeToSetter view external IPancake.sol:14
getPair view external IPancake.sol:16
allPairs view external IPancake.sol:21
allPairsLength view external IPancake.sol:23
createPair nonpayable external IPancake.sol:25
setFeeTo nonpayable external IPancake.sol:29
setFeeToSetter nonpayable external IPancake.sol:31
ERC20 (ERC20.sol:1)
is IERC20Metadata, IERC20, Context
decimals view public ERC20.sol:1
symbol view public ERC20.sol:1
transferFrom nonpayable public ERC20.sol:1
transfer nonpayable public ERC20.sol:1
increaseAllowance nonpayable public ERC20.sol:1
decreaseAllowance nonpayable public ERC20.sol:1
approve nonpayable public ERC20.sol:1
(default) nonpayable public ERC20.sol:1
name view public ERC20.sol:1
balanceOf view public ERC20.sol:1
totalSupply view public ERC20.sol:1
_mint nonpayable internal ERC20.sol:1
allowance view public ERC20.sol:1
_burn nonpayable internal ERC20.sol:1
_transfer nonpayable internal ERC20.sol:1
_approve nonpayable internal ERC20.sol:1
_afterTokenTransfer nonpayable internal ERC20.sol:1
_beforeTokenTransfer nonpayable internal ERC20.sol:1
ParallelToken (PRL.sol:11)
is Pausable, AccessControl, ERC165, IERC165, IAccessControl, ERC20Snapshot, ERC20, IERC20Metadata, IERC20, Context
activeAntiWhale nonpayable public PRL.sol:53
markWhaleAddress nonpayable public PRL.sol:61
_beforeTokenTransfer nonpayable internal PRL.sol:82
isWhaleTransaction view public PRL.sol:68
(default) nonpayable public PRL.sol:20
_initPair nonpayable private PRL.sol:28
unpause nonpayable public PRL.sol:49
pause nonpayable public PRL.sol:45
snapshot nonpayable public PRL.sol:41
Pausable (Pausable.sol:1)
is Context
paused view public Pausable.sol:1
(default) nonpayable internal Pausable.sol:1
_unpause nonpayable internal Pausable.sol:1
_pause nonpayable internal Pausable.sol:1
Context (Context.sol:1)
_msgSender view internal Context.sol:1
_msgData view internal Context.sol:1
IAccessControl (IAccessControl.sol:1)
hasRole view external IAccessControl.sol:1
getRoleAdmin view external IAccessControl.sol:1
revokeRole nonpayable external IAccessControl.sol:1
grantRole nonpayable external IAccessControl.sol:1
renounceRole nonpayable external IAccessControl.sol:1
IERC20 (IERC20.sol:1)
totalSupply view external IERC20.sol:1
balanceOf view external IERC20.sol:1
allowance view external IERC20.sol:1
transferFrom nonpayable external IERC20.sol:1
transfer nonpayable external IERC20.sol:1
approve nonpayable external IERC20.sol:1
IERC20Metadata (IERC20Metadata.sol:1)
is IERC20
name view external IERC20Metadata.sol:1
symbol view external IERC20Metadata.sol:1
decimals view external IERC20Metadata.sol:1
Arrays (Arrays.sol:1)
findUpperBound view internal Arrays.sol:1
ERC20Snapshot (ERC20Snapshot.sol:1)
is ERC20, IERC20Metadata, IERC20, Context
_beforeTokenTransfer nonpayable internal ERC20Snapshot.sol:1
balanceOfAt view public ERC20Snapshot.sol:1
_updateAccountSnapshot nonpayable private ERC20Snapshot.sol:1
totalSupplyAt view public ERC20Snapshot.sol:1
_updateTotalSupplySnapshot nonpayable private ERC20Snapshot.sol:1
_snapshot nonpayable internal ERC20Snapshot.sol:1
_valueAt view private ERC20Snapshot.sol:1
_updateSnapshot nonpayable private ERC20Snapshot.sol:1
_getCurrentSnapshotId view internal ERC20Snapshot.sol:1
_lastSnapshotId view private ERC20Snapshot.sol:1
Counters (Counters.sol:1)
current view internal Counters.sol:1
increment nonpayable internal Counters.sol:1
decrement nonpayable internal Counters.sol:1
reset nonpayable internal Counters.sol:1
Public Send Warning Call Warning Constant Internal External PAYABLE

Image 1. Parallel token smart contract call graph

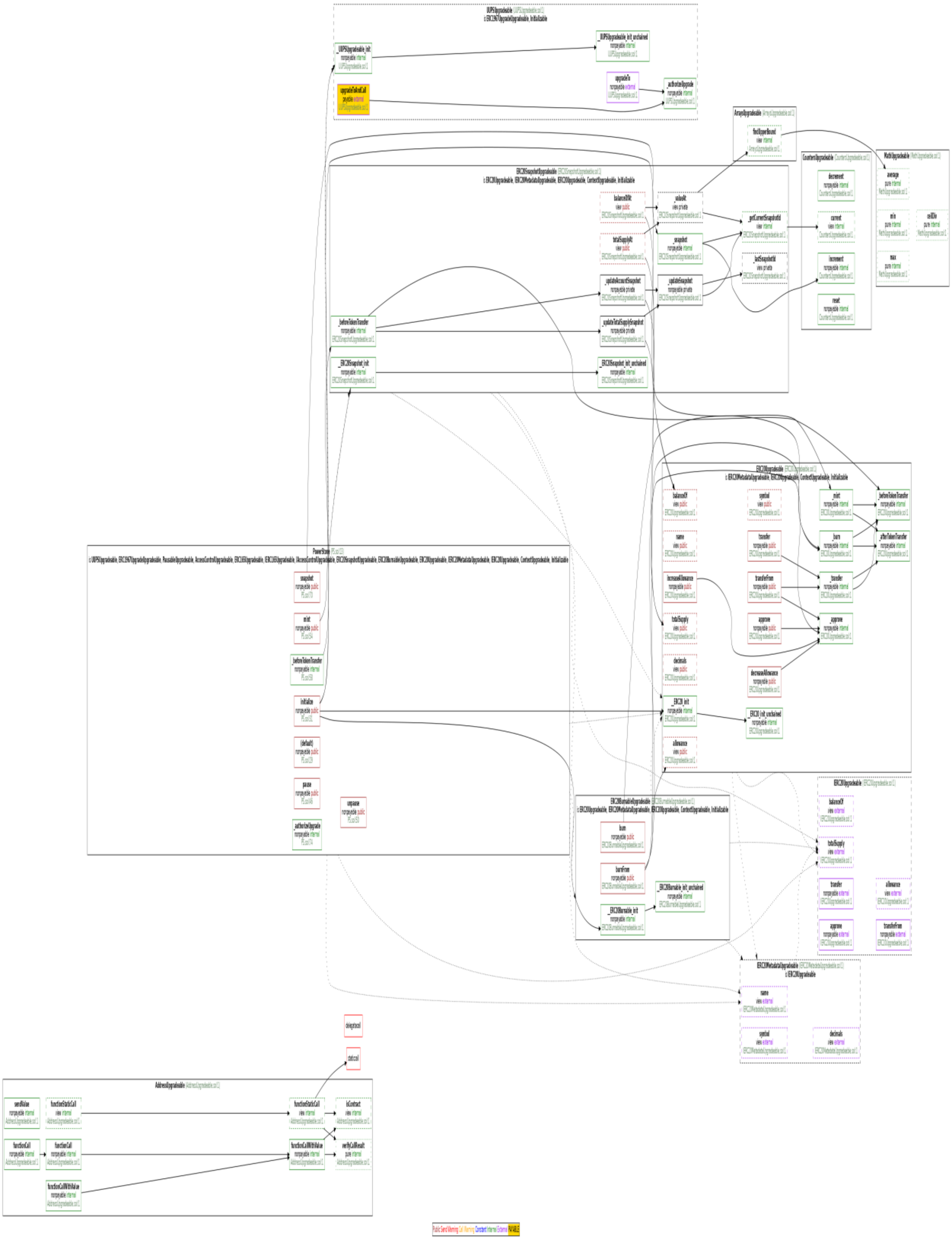
UUPSUpgradeable (UUPSUpgradeable.sol:1)
is ERC1967UpgradeUpgradeable, Initializable
__UUPSUpgradeable_init nonpayable internal UUPSUpgradeable.sol:1
__UUPSUpgradeable_init_unchained nonpayable internal UUPSUpgradeable.sol:1
upgradeToAndCall payable external UUPSUpgradeable.sol:1
upgradeTo nonpayable external UUPSUpgradeable.sol:1
_authorizeUpgrade nonpayable internal UUPSUpgradeable.sol:1
ArraysUpgradeable (ArraysUpgradeable.sol:1)
findUpperBound view internal ArraysUpgradeable.sol:1
CountersUpgradeable (CountersUpgradeable.sol:1)
decrement nonpayable internal CountersUpgradeable.sol:1
current view internal CountersUpgradeable.sol:1
increment nonpayable internal CountersUpgradeable.sol:1
reset nonpayable internal CountersUpgradeable.sol:1
MathUpgradeable (MathUpgradeable.sol:1)
average pure internal MathUpgradeable.sol:1
min pure internal MathUpgradeable.sol:1
ceilDiv pure internal MathUpgradeable.sol:1
max pure internal MathUpgradeable.sol:1
ERC20SnapshotUpgradeable (ERC20SnapshotUpgradeable.sol:1)
is ERC20Upgradeable, IERC20MetadataUpgradeable, IERC20Upgradeable, ContextUpgradeable, Initializable
balanceOfAt view public ERC20SnapshotUpgradeable.sol:1
totalSupplyAt view public ERC20SnapshotUpgradeable.sol:1
_valueAt view private ERC20SnapshotUpgradeable.sol:1
_snapshot nonpayable internal ERC20SnapshotUpgradeable.sol:1
_getCurrentSnapshotId view internal ERC20SnapshotUpgradeable.sol:1
_lastSnapshotId view private ERC20SnapshotUpgradeable.sol:1
_updateAccountSnapshot nonpayable private ERC20SnapshotUpgradeable.sol:1
_updateSnapshot nonpayable private ERC20SnapshotUpgradeable.sol:1
_updateTotalSupplySnapshot nonpayable private ERC20SnapshotUpgradeable.sol:1
_beforeTokenTransfer nonpayable internal ERC20SnapshotUpgradeable.sol:1
__ERC20Snapshot_init nonpayable internal ERC20SnapshotUpgradeable.sol:1
__ERC20Snapshot_init_unchained nonpayable internal ERC20SnapshotUpgradeable.sol:1
ERC20Upgradeable (ERC20Upgradeable.sol:1)
is IERC20MetadataUpgradeable, IERC20Upgradeable, ContextUpgradeable, Initializable
balanceOf view public ERC20Upgradeable.sol:1
name view public ERC20Upgradeable.sol:1
increaseAllowance nonpayable public ERC20Upgradeable.sol:1
totalSupply view public ERC20Upgradeable.sol:1
decimals view public ERC20Upgradeable.sol:1
__ERC20_init nonpayable internal ERC20Upgradeable.sol:1
allowance view public ERC20Upgradeable.sol:1
symbol view public ERC20Upgradeable.sol:1
transfer nonpayable public ERC20Upgradeable.sol:1
transferFrom nonpayable public ERC20Upgradeable.sol:1
approve nonpayable public ERC20Upgradeable.sol:1
decreaseAllowance nonpayable public ERC20Upgradeable.sol:1
__ERC20_init_unchained nonpayable internal ERC20Upgradeable.sol:1
_mint nonpayable internal ERC20Upgradeable.sol:1
_burn nonpayable internal ERC20Upgradeable.sol:1
_transfer nonpayable internal ERC20Upgradeable.sol:1
_approve nonpayable internal ERC20Upgradeable.sol:1
_beforeTokenTransfer nonpayable internal ERC20Upgradeable.sol:1
_afterTokenTransfer nonpayable internal ERC20Upgradeable.sol:1
PowerStone (PS.sol:13)
is UUPSUpgradeable, ERC1967UpgradeUpgradeable, PausableUpgradeable, AccessControlUpgradeable, ERC165Upgradeable, IERC165Upgradeable, IAccessControlUpgradeable, ERC20SnapshotUpgradeable, ERC20BurnableUpgradeable, ERC20Upgradeable, IERC20MetadataUpgradeable, IERC20Upgradeable, ContextUpgradeable, Initializable
snapshot nonpayable public PS.sol:70
mint nonpayable public PS.sol:54
_beforeTokenTransfer nonpayable internal PS.sol:58
initialize nonpayable public PS.sol:31
(default) nonpayable public PS.sol:29
pause nonpayable public PS.sol:46
unpause nonpayable public PS.sol:50
_authorizeUpgrade nonpayable internal PS.sol:74
ERC20BurnableUpgradeable (ERC20BurnableUpgradeable.sol:1)
is ERC20Upgradeable, IERC20MetadataUpgradeable, IERC20Upgradeable, ContextUpgradeable, Initializable
burn nonpayable public ERC20BurnableUpgradeable.sol:1
burnFrom nonpayable public ERC20BurnableUpgradeable.sol:1
__ERC20Burnable_init nonpayable internal ERC20BurnableUpgradeable.sol:1
__ERC20Burnable_init_unchained nonpayable internal ERC20BurnableUpgradeable.sol:1
IERC20Upgradeable (IERC20Upgradeable.sol:1)
balanceOf view external IERC20Upgradeable.sol:1
totalSupply view external IERC20Upgradeable.sol:1
transfer nonpayable external IERC20Upgradeable.sol:1
allowance view external IERC20Upgradeable.sol:1
approve nonpayable external IERC20Upgradeable.sol:1
transferFrom nonpayable external IERC20Upgradeable.sol:1
IERC20MetadataUpgradeable (IERC20MetadataUpgradeable.sol:1)
is IERC20Upgradeable
name view external IERC20MetadataUpgradeable.sol:1
symbol view external IERC20MetadataUpgradeable.sol:1
decimals view external IERC20MetadataUpgradeable.sol:1
delegatecall
staticcall
AddressUpgradeable (AddressUpgradeable.sol:1)
sendValue nonpayable internal AddressUpgradeable.sol:1
functionStaticCall view internal AddressUpgradeable.sol:1
functionCall nonpayable internal AddressUpgradeable.sol:1
functionCall nonpayable internal AddressUpgradeable.sol:1
functionCallWithValue nonpayable internal AddressUpgradeable.sol:1
functionStaticCall view internal AddressUpgradeable.sol:1
isContract view internal AddressUpgradeable.sol:1
functionCallWithValue nonpayable internal AddressUpgradeable.sol:1
verifyCallResult pure internal AddressUpgradeable.sol:1
Public Send Warning Call Warning Constant Internal External PAYABLE

Image 2. PowerStone token smart contract call graph

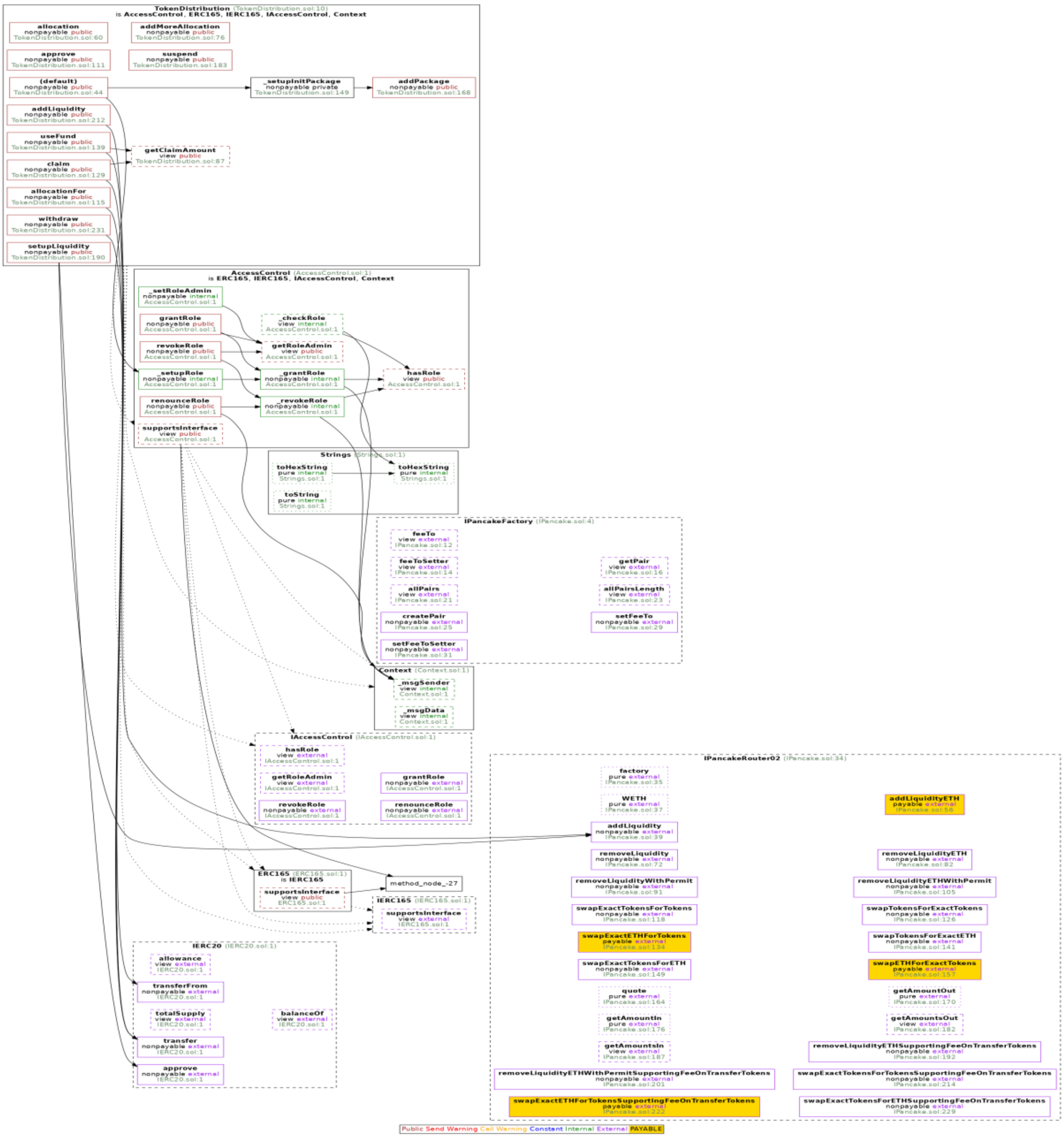
TokenDistribution (TokenDistribution.sol:10)
is AccessControl, ERC165, IERC165, IAccessControl, Context
allocation nonpayable public TokenDistribution.sol:60
addMoreAllocation nonpayable public TokenDistribution.sol:76
approve nonpayable public TokenDistribution.sol:111
suspend nonpayable public TokenDistribution.sol:183
(default) nonpayable public TokenDistribution.sol:44
_setupInitPackage nonpayable private TokenDistribution.sol:149
addPackage nonpayable public TokenDistribution.sol:168
addLiquidity nonpayable public TokenDistribution.sol:212
useFund nonpayable public TokenDistribution.sol:139
getClaimAmount view public TokenDistribution.sol:87
claim nonpayable public TokenDistribution.sol:129
allocationFor nonpayable public TokenDistribution.sol:115
withdraw nonpayable public TokenDistribution.sol:231
setupLiquidity nonpayable public TokenDistribution.sol:190
AccessControl (AccessControl.sol:1)
is ERC165, IERC165, IAccessControl, Context
_setRoleAdmin nonpayable internal AccessControl.sol:1
grantRole nonpayable public AccessControl.sol:1
_checkRole view internal AccessControl.sol:1
revokeRole nonpayable public AccessControl.sol:1
getRoleAdmin view public AccessControl.sol:1
_setupRole nonpayable internal AccessControl.sol:1
_grantRole nonpayable internal AccessControl.sol:1
hasRole view public AccessControl.sol:1
renounceRole nonpayable public AccessControl.sol:1
_revokeRole nonpayable internal AccessControl.sol:1
supportsInterface view public AccessControl.sol:1
Strings (Strings.sol:1)
toHexString pure internal Strings.sol:1
toHexString pure internal Strings.sol:1
toString pure internal Strings.sol:1
IPancakeFactory (IPancake.sol:4)
feeTo view external IPancake.sol:12
feeToSetter view external IPancake.sol:14
getPair view external IPancake.sol:16
allPairs view external IPancake.sol:21
allPairsLength view external IPancake.sol:23
createPair nonpayable external IPancake.sol:25
setFeeTo nonpayable external IPancake.sol:29
setFeeToSetter nonpayable external IPancake.sol:31
Context (Context.sol:1)
_msgSender view internal Context.sol:1
_msgData view internal Context.sol:1
IAccessControl (IAccessControl.sol:1)
hasRole view external IAccessControl.sol:1
getRoleAdmin view external IAccessControl.sol:1
grantRole nonpayable external IAccessControl.sol:1
revokeRole nonpayable external IAccessControl.sol:1
renounceRole nonpayable external IAccessControl.sol:1
ERC165 (ERC165.sol:1)
is IERC165
supportsInterface view public ERC165.sol:1
method_node_-27
IERC165 (IERC165.sol:1)
supportsInterface view external IERC165.sol:1
IERC20 (IERC20.sol:1)
allowance view external IERC20.sol:1
transferFrom nonpayable external IERC20.sol:1
totalSupply view external IERC20.sol:1
balanceOf view external IERC20.sol:1
transfer nonpayable external IERC20.sol:1
approve nonpayable external IERC20.sol:1
IPancakeRouter02 (IPancake.sol:34)
factory pure external IPancake.sol:35
WETH pure external IPancake.sol:37
addLiquidityETH payable external IPancake.sol:56
addLiquidity nonpayable external IPancake.sol:39
removeLiquidity nonpayable external IPancake.sol:72
removeLiquidityETH nonpayable external IPancake.sol:82
removeLiquidityWithPermit nonpayable external IPancake.sol:91
removeLiquidityETHWithPermit nonpayable external IPancake.sol:105
swapExactTokensForTokens nonpayable external IPancake.sol:118
swapTokensForExactTokens nonpayable external IPancake.sol:126
swapExactETHForTokens payable external IPancake.sol:134
swapTokensForExactETH nonpayable external IPancake.sol:141
swapExactTokensForETH nonpayable external IPancake.sol:149
swapETHForExactTokens payable external IPancake.sol:157
quote pure external IPancake.sol:164
getAmountOut pure external IPancake.sol:170
getAmountIn pure external IPancake.sol:176
getAmountsOut view external IPancake.sol:182
getAmountsIn view external IPancake.sol:187
removeLiquidityETHSupportingFeeOnTransferTokens nonpayable external IPancake.sol:192
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens nonpayable external IPancake.sol:201
swapExactTokensForTokensSupportingFeeOnTransferTokens nonpayable external IPancake.sol:214
swapExactETHForTokensSupportingFeeOnTransferTokens payable external IPancake.sol:222
swapExactTokensForETHSupportingFeeOnTransferTokens nonpayable external IPancake.sol:229
Public Send Warning Call Warning Constant Internal External PAYABLE

Image 3. TokenDistribution smart contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*2021-12-06*	Private Report	Verichains Lab
1.1	*2021-12-09*	Public Report	Verichains Lab

Table 4. Report versions history