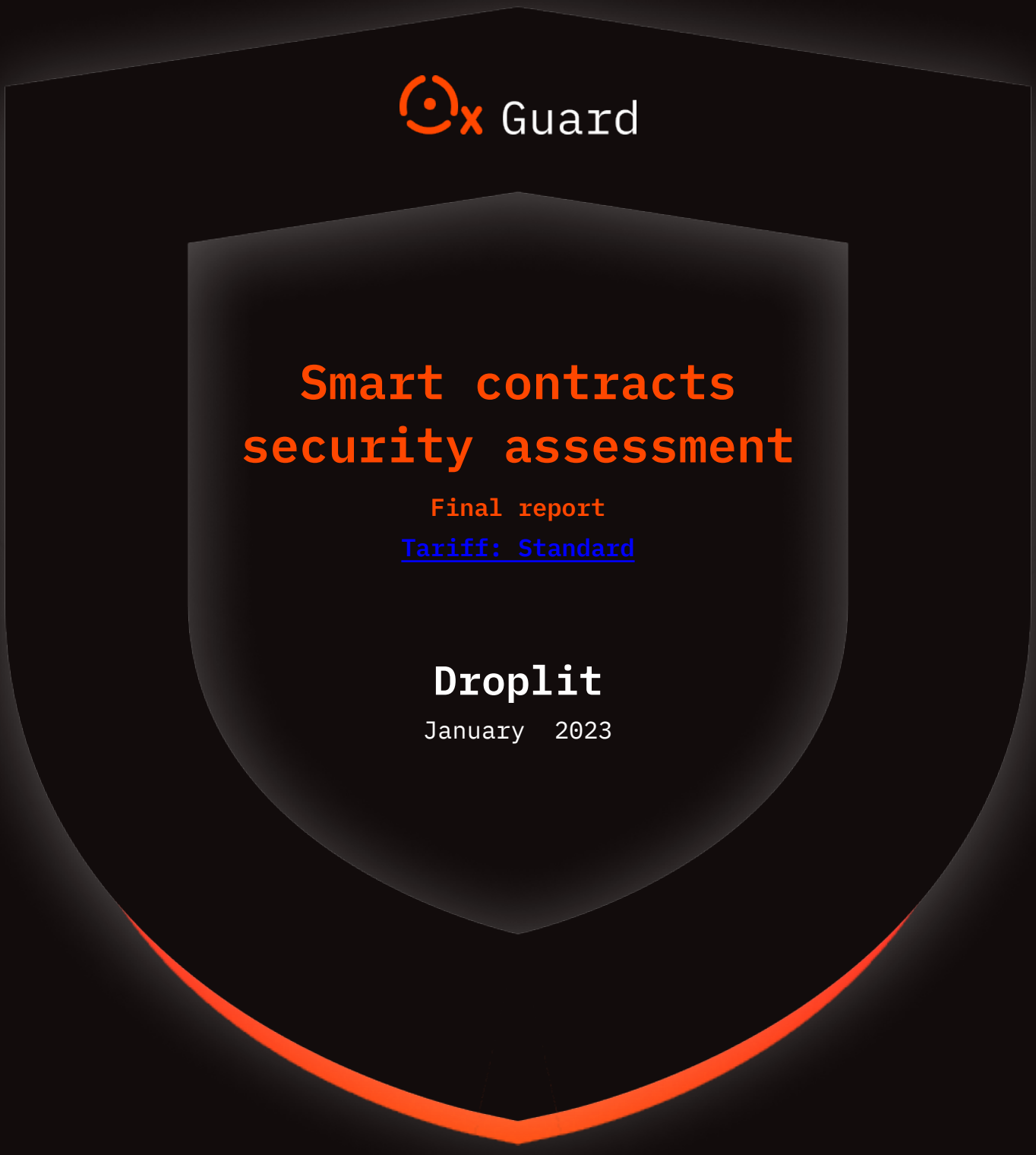

Guard

Smart contracts security assessment

Final report

Tariff: Standard

Droplit

January 2023

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for **Droplit**.

The reviewed project is a Tomb Finance fork, allowing users to farm the main DropLit (LIT) and the share dropShare (GDS) tokens. LIT and GDS tokens are ERC20-standard tokens with taxes on on transfers.

Both rewards pool (a.k.a. farms) contracts may charge a fee of up to 2% for each deposit.

The code is available in the Binance Smart Chain (BSC) network at
0x3CfEcAeEEE99C2039D2312f6097f8139Aa64Df03 (DropLit),
0x0eC381feE5A306FDdbf9CcE7B06C189798D7198A (dropShare),
0x05fdf8923D2d9a30344C538AF8fd4b7BDD3D49ae (Boardroom),
0x15CDC96634b1E488cd15c2786B9db93D850403bd (Bond),
0x358e6a3D5f1B4A612e871b7Aa4046dc2d3D71c4d (Treasury),
0x71EE586EcFba14635870B1bCfDC9EB526d2161D6 (GenesisRewardPool),
0xBDfD85D0FC253db1DD8859d472A8Fa475Db7351E (ShareRewardPool),
0xC1E455d5E1CddF5cd9620371D93656DeA1Fe30a3 (Oracle).

Report Update.

The contract's code was updated according to this report and deployed to:
0x79B2bc95344eFe31cb6a7B0Cf8A843a5eE125dFf (DropLit),
0x5Abf65C1d152244c6Bd4ad0a5eB92DB00e403BdB (dropShare),
0x9D76Db596D281897F3ce842475b7BD6Ea2580b4b (Boardroom),
0xa0f66D074efA9506F0502cF8A3A13fC891884041 (Bond),
0x6542AfE293815729ccAe051689a3c85b907f5ec0 (Treasury),
0x5c40222f13Ba0183b973dcE56f284017e53C4f8f (GenesisRewardPool),
0x79D7c1a12c4dE91C487A87602478C5bc19b3aa7c (ShareRewardPool),
0x6d149B1BeEc5bD24321C20900567D3f96B94711F (Oracle).

Name	Droplit
Audit date	2023-01-05 - 2023-01-12
Language	Solidity
Platform	Binance Smart Chain

Contracts checked

Name	Address
DropLit	0x79B2bc95344eFe31cb6a7B0Cf8A843a5eE125dFf
dropShare	0x5Abf65C1d152244c6Bd4ad0a5eB92DB00e403BdB
Boardroom	0x9D76Db596D281897F3ce842475b7BD6Ea2580b4b
Bond	0xa0f66D074efA9506F0502cF8A3A13fC891884041
Treasury	0x6542AfE293815729ccAe051689a3c85b907f5ec0
GenesisRewardPool	0x5c40222f13Ba0183b973dcE56f284017e53C4f8f
ShareRewardPool	0x79D7c1a12c4dE91C487A87602478C5bc19b3aa7c
Oracle	0x6d149B1BeEc5bD24321C20900567D3f96B94711F

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) of all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed

Floating Pragma	passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. Wrong compiler version (Oracle)
Status: Fixed

The UniswapV2OracleLibrary is designed to be compiled with a pre-0.8 compiler as it contains math operations with overflowing. `price0CumulativeLast` and `price1CumulativeLast` of Uniswap-like pairs are overflowing over time, but the `Oracle.update()` function doesn't use `unchecked{}` blocks and will permanently fail as soon as a cumulative price of the corresponding pair overflows.

```
    function update() external checkEpoch {
        (uint256 price0Cumulative, uint256 price1Cumulative, uint32 blockTimestamp) =
UniswapV2OracleLibrary.currentCumulativePrices(address(pair));
```

```
        ...
        price0Average = FixedPoint.uq112x112(uint224((price0Cumulative -
price0CumulativeLast) / timeElapsed));
        price1Average = FixedPoint.uq112x112(uint224((price1Cumulative -
price1CumulativeLast) / timeElapsed));
        ...
    }
```

Recommendation: Re-compile Oracle contract with correct compiler version or update its code to allow desired overflowing.

Medium severity issues

1. Whitelist requirements aren't mandatory (DropLit)

Status: Fixed

The `setWhiteList()` function contains a set of requirements for an address to be whitelisted, e.g. it must be a contract but not the swap router or one of the pairs. However, whitelisted in the constructor section addresses are exempt from those requirements. For example, the admin address is a whitelisted EOA.

```
    function setWhiteList(address _WhiteList) public onlyAdmin {
        require(isContract(_WhiteList) == true, "only contracts can be whitelisted");
        require(address(uniswapV2Router) != _WhiteList, "set tax to 0 if you want to
remove fee from trading");
        require(PairWBNB != _WhiteList, "set tax to 0 if you want to remove fee from
trading");
        require(PairBUSD != _WhiteList, "set tax to 0 if you want to remove fee from
trading");
        require(PairDshare != _WhiteList, "set tax to 0 if you want to remove fee from
trading");
□□whitelist[_WhiteList] = true;
    }
```

2. Whitelist requirements aren't mandatory (dropShare)

Status: Fixed

The `setWhiteList()` function contains a set of requirements for an address to be whitelisted, e.g. it must be a contract but not the swap router or one of the pairs. However, whitelisted in the constructor section addresses are exempt from those requirements. For example, the admin address is a whitelisted EOA.

```
    function setWhiteList(address _WhiteList) public onlyAdmin {
        require(isContract(_WhiteList) == true, "only contracts can be whitelisted");
        require(address(uniswapV2Router) != _WhiteList, "set tax to 0 if you want to
remove fee from trading");
        require(PairWBNB != _WhiteList, "set tax to 0 if you want to remove fee from
trading");
        require(PairBUSD != _WhiteList, "set tax to 0 if you want to remove fee from
trading");
        require(PairDshare != _WhiteList, "set tax to 0 if you want to remove fee from
trading");
□□whitelist[_WhiteList] = true;
    }
```

3. Treasure does not support all operator methods for DropLit contract (Treasury)
Status: Fixed

Treasury contract is set as the operator of DropLit token contract, but it can't use `setOracle()` function.

4. Source of rewards (GenesisRewardPool)
Status: Fixed

The contract rewards in form of share token is defined by

```
uint256 public sharesPerSecond = 0.138888888 ether;
uint256 public constant TOTAL_REWARDS = 12000 ether;
```

, but the source of these rewards is not explicitly indicated. Users may experience loss of expected reward.

5. Source of rewards (ShareRewardPool)
Status: Fixed

The contract rewards in form of share token are defined by

```
uint256 public sharesPerSecond = 0.0009384384 ether;
uint256 public constant TOTAL_REWARDS = 30000 ether;
```

, but the source of these rewards is not explicitly indicated. Users may experience a loss of expected reward.

Low severity issues

1. Lack of events (DropLit)
Status: Open

There's a general lack of events emitted in governance functions, which complicates tracking the history of changes in crucial system parameters.

2. Unused code (DropLit)
Status: Open

The SafeMath8 library is not in use. It's also outdated since it should be compiled with pre-0.8 pragma versions to avoid double-spending gas on overflow checks.

3. Parameters of addLiquidity (DropLit)
Status: Open

There's a `addLiquidity` call for a Uniswap-like router in the `start()` function that may constantly fail since the pair is already created and the amounts are fixed. Also, the deadline parameter is used incorrectly since it can't be calculated on-chain: router contract checks deadline is not smaller than the current time, i.e. setting a deadline to `block.timestamp` or greater will always pass and setting it lower than `block.timestamp` will always revert.

4. Typos (DropLit)
Status: Fixed

Typo in 'allready'.

5. Public open function (DropLit)
Status: Open

The `setPairDshare()` initializer function is open for public use. While the deployed contract is initialized correctly, any possible future code reuse may face difficulties.

6. Gas optimization (DropLit)
Status: Open

1. There're multiple unnecessary readings of data from blockchain in the `start()` function: `BUSD.balanceOf(address(this))` and `balanceOf(address(this))` should be read once to local variables.

2. The `mint()` function contains a double call of the contract's balance before and after with an unclear purpose. The internal `_mint()` function inherited from the ERC20 contract has its own safety checks.

3. Unnecessary multiplication by `MULTIPLIER` is performed in the `transfer()` and `transferFrom()` functions. It doesn't improve the accuracy of calculations, the divisor should be set to 100 instead.

7. Lack of events (dropShare)
Status: Open

There's a general lack of events emitted in governance functions, which complicates tracking the history of changes in crucial system parameters.

Recommendation: We recommend reviewing the architecture for blocking users when selling tokens. Perhaps the `_transfer()` function should track token **senders** instead of **recipients**.

8. Gas optimization (dropShare)
Status: Open

1. Unnecessary multiplication by `MULTIPLIER` is performed in the `transfer()` and `transferFrom()` functions. It doesn't improve the accuracy of calculations, the divisor should be set to 100 instead.

9. ContractGuard doesn't prevent re-entrancy (Boardroom)

Status: Open

The ContractGuard contract is designed to prevent multiple calls in the same block but it doesn't prevent re-entrancy, i.e. multiple calls in the same transaction.

```
    modifier onlyOneBlock() {
        require(!checkSameOriginReentranted(), "ContractGuard: one block, one
function");
        require(!checkSameSenderReentranted(), "ContractGuard: one block, one
function");

        _;

        _status[block.number][tx.origin] = true;
        _status[block.number][msg.sender] = true;
    }
```

10. Validation in the initialize() function (Treasury)

Status: Open

The input parameters of the `initialize()` function aren't checked in any way. The `nativePriceOne` variable is set to 1e18 regardless of the actual `decimals()` value of the native token.

11. ContractGuard doesn't prevent re-entrancy (Treasury)

Status: Open

The ContractGuard contract is designed to prevent multiple calls in the same block but it doesn't prevent re-entrancy, i.e. multiple calls in the same transaction.

```
    modifier onlyOneBlock() {
        require(!checkSameOriginReentranted(), "ContractGuard: one block, one
function");
        require(!checkSameSenderReentranted(), "ContractGuard: one block, one
function");

        _;
```

```
        _status[block.number][tx.origin] = true;
        _status[block.number][msg.sender] = true;
    }
```

12. Gas optimization (Treasury)
Status: Open

1. Checking the `targetPrice` from parameters of the `redeemBonds()` function seems unnecessary: the gas-wise way is to receive a `bondRate` parameter and check it against `getBondPremiumRate()` directly.

2. Excessive data is read from the blockchain in the `redeemBonds()` function: `getBondPremiumRate()` should receive already in-memory values of `nativePrice` and `nativePriceCeiling`.

3. Redundant code in the `redeemBonds()` function: `require(_rate > 0)` is always passed as it's already checked that `nativePrice > nativePriceCeiling`.

13. Irrelevant error message (Treasury)
Status: Fixed

There's a mistakenly copy-pasted error message in the `redeemBonds()` function:

```
  require( nativePrice > nativePriceCeiling, "Treasury: nativePrice not eligible for
bond purchase" );
```

'purchase' should be replaced by 'sale'.

14. Contract doesn't support tokens with transfer fees (GenesisRewardPool)
Status: Open

Actual transfer amounts aren't checked so the owner must not add pools with tokens with transfer commissions unless this contract is excluded from such fees (see DropLit and dropShare contracts).

15. Possible block gas limit problem (GenesisRewardPool)
Status: Open

An unlimited loop over an array of pools may cause a gas limit problem if too many pools would be

added. The owner must pay attention when adding new pools.

16. Gas optimization (GenesisRewardPool)
Status: Open

Pool duplication check is ineffective, it should be performed via mapping from the token address. The other way is to allow duplicated pools by storing individual pools balances in `PoolInfo` structure, i.e. the `updatePool()` function should not check the `pool.token.balanceOf(address(this))` but read the pool balance from the structure.

17. Gas optimization (ShareRewardPool)
Status: Open

Pool duplication check is ineffective, it should be performed via mapping from the token address. The other way is to allow duplicated pools by storing individual pools balances in `PoolInfo` structure, i.e. the `updatePool()` function should not check the `pool.token.balanceOf(address(this))` but read the pool balance from the structure.

18. Contract doesn't support tokens with transfer fees (ShareRewardPool)
Status: Open

Actual transfer amounts aren't checked so the owner must not add pools with tokens with transfer commissions unless this contract is excluded from such fees (see DropLit and dropShare contracts).

19. Possible block gas limit problem (ShareRewardPool)
Status: Open

An unlimited loop over an array of pools may cause a gas limit problem if too many pools are added. The owner must pay attention when adding new pools.

20. Gas optimization (Oracle)
Status: Open

The `getCurrentEpoch()` function and `epoch` variable of the Epoch contract have unclear functionality.

Conclusion

Droplit DropLit, dropShare, Boardroom, Bond, Treasury, GenesisRewardPool, ShareRewardPool, Oracle contracts were audited. 1 high, 5 medium, 20 low severity issues were found.
1 high, 5 medium, 2 low severity issues have been fixed in the update.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.
OxGuard retains exclusive publishing rights for the results of this audit on its website and social networks.

Static code analysis

```
Reentrancy in DropLit.start() (contracts/drop.sol#68-74):
        External calls:
        - BUSD.approve(address(uniswapV2Router),BUSD.balanceOf(address(this)))
(contracts/drop.sol#71)
        - uniswapV2Router.addLiquidity(address(this),address(BUSD),balanceOf(address(thi
s)),BUSD.balanceOf(address(this)),balanceOf(address(this)),BUSD.balanceOf(address(this))
,msg.sender,block.timestamp + 60) (contracts/drop.sol#72)
        State variables written after the call(s):
        - started = true (contracts/drop.sol#73)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1

DropLit._getPrice()._price (contracts/drop.sol#119) is a local variable never
initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables

DropLit.start() (contracts/drop.sol#68-74) ignores return value by
BUSD.approve(address(uniswapV2Router),BUSD.balanceOf(address(this))) (contracts/
drop.sol#71)
DropLit.start() (contracts/drop.sol#68-74) ignores return value by uniswapV2Router.addLi
quidity(address(this),address(BUSD),balanceOf(address(this)),BUSD.balanceOf(address(this
)),balanceOf(address(this)),BUSD.balanceOf(address(this)),msg.sender,block.timestamp +
60) (contracts/drop.sol#72)
DropLit._getPrice() (contracts/drop.sol#118-124) ignores return value by
IOracle(oracle).consult(address(this),1e18) (contracts/drop.sol#119-123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

DropLit.constructor(address,address,address,address,address,address,address)._BOND
(contracts/drop.sol#76) lacks a zero-check on :
                - BOND = _BOND (contracts/drop.sol#98)
DropLit.constructor(address,address,address,address,address,address,address)._genesisAdd
ress (contracts/drop.sol#76) lacks a zero-check on :
                - genesisAddress = _genesisAddress (contracts/drop.sol#99)
DropLit.constructor(address,address,address,address,address,address,address)._treasury
(contracts/drop.sol#76) lacks a zero-check on :
                - treasury = _treasury (contracts/drop.sol#100)
```

```
DropLit.constructor(address,address,address,address,address,address,address)._boardroom
(contracts/drop.sol#76) lacks a zero-check on :
                - boardroom = _boardroom (contracts/drop.sol#101)
DropLit.constructor(address,address,address,address,address,address,address)._shareRewar
dPool (contracts/drop.sol#76) lacks a zero-check on :
                - shareRewardPool = _shareRewardPool (contracts/drop.sol#102)
DropLit.setPairDshare(address)._pairDshare (contracts/drop.sol#145) lacks a zero-check
on :
                - PairDshare = _pairDshare (contracts/drop.sol#147)
DropLit.setAdmin(address)._admin (contracts/drop.sol#150) lacks a zero-check on :
                - admin = _admin (contracts/drop.sol#151)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

Variable 'DropLit._getPrice()._price (contracts/drop.sol#119)' in DropLit._getPrice()
(contracts/drop.sol#118-124) potentially used before declaration: uint256(_price)
(contracts/drop.sol#120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-
declaration-usage-of-local-variables

DropLit.isContract(address) (contracts/drop.sol#126-132) uses assembly
        - INLINE ASM (contracts/drop.sol#128-130)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

DropLit.setWhiteList(address) (contracts/drop.sol#135-142) compares to a boolean
constant:
        -require(bool,string)(isContract(_WhiteList) == true,only contracts can be
whitelisted) (contracts/drop.sol#136)
DropLit.transferFrom(address,address,uint256) (contracts/drop.sol#186-199) compares to
a boolean constant:
        -whitelist[sender] == true || whitelist[recipient] == true (contracts/
drop.sol#187)
DropLit.transfer(address,uint256) (contracts/drop.sol#201-216) compares to a boolean
constant:
        -whitelist[_msgSender()] == true || whitelist[recipient] == true (contracts/
drop.sol#202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-
equality

DropLit._getPrice() (contracts/drop.sol#118-124) is never used and should be removed
SafeMath8.add(uint8,uint8) (contracts/libraries/SafeMath8.sol#16-21) is never used and
```

```
should be removed
SafeMath8.div(uint8,uint8) (contracts/libraries/SafeMath8.sol#90-92) is never used and
should be removed
SafeMath8.div(uint8,uint8,string) (contracts/libraries/SafeMath8.sol#106-112) is never
used and should be removed
SafeMath8.mod(uint8,uint8) (contracts/libraries/SafeMath8.sol#126-128) is never used
and should be removed
SafeMath8.mod(uint8,uint8,string) (contracts/libraries/SafeMath8.sol#142-145) is never
used and should be removed
SafeMath8.mul(uint8,uint8) (contracts/libraries/SafeMath8.sol#64-76) is never used and
should be removed
SafeMath8.sub(uint8,uint8) (contracts/libraries/SafeMath8.sol#33-35) is never used and
should be removed
SafeMath8.sub(uint8,uint8,string) (contracts/libraries/SafeMath8.sol#47-52) is never
used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (contracts/drop.sol#4) necessitates a version too recent to be
trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/interfaces/IOracle.sol#3) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/interfaces/IUniswapV2Factory.sol#3) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/interfaces/IUniswapV2Pair.sol#3) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/interfaces/IUniswapV2Router01.sol#3) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/interfaces/IUniswapV2Router02.sol#2) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/libraries/Operator.sol#3) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/libraries/SafeMath8.sol#3) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Parameter DropLit.isContract(address)._addr (contracts/drop.sol#126) is not in
mixedCase
Parameter DropLit.setWhiteList(address)._WhiteList (contracts/drop.sol#135) is not in
mixedCase
```

```
Parameter DropLit.setPairDshare(address)._pairDshare (contracts/drop.sol#145) is not in
mixedCase
Parameter DropLit.setAdmin(address)._admin (contracts/drop.sol#150) is not in mixedCase
Parameter DropLit.setOracle(address)._oracle (contracts/drop.sol#154) is not in
mixedCase
Parameter DropLit.setTaxCollectorAddress(address)._taxCollectorAddress (contracts/
drop.sol#160) is not in mixedCase
Parameter DropLit.setTaxRate(uint256)._taxRate (contracts/drop.sol#166) is not in
mixedCase
Variable DropLit.PairDshare (contracts/drop.sol#37) is not in mixedCase
Variable DropLit.BOND (contracts/drop.sol#42) is not in mixedCase
Variable DropLit.BUSD (contracts/drop.sol#43) is not in mixedCase
Variable DropLit.PairWBNB (contracts/drop.sol#44) is not in mixedCase
Variable DropLit.PairBUSD (contracts/drop.sol#45) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/interfaces/IUniswapV2Pair.sol#34)
is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/interfaces/IUniswapV2Pair.sol#36)
is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/interfaces/
IUniswapV2Pair.sol#67) is not in mixedCase
Function IUniswapV2Router01.WETH() (contracts/interfaces/IUniswapV2Router01.sol#8) is
not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256
,address,uint256).amountADesired (contracts/interfaces/IUniswapV2Router01.sol#13) is
too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,u
int256,address,uint256).amountBDesired (contracts/interfaces/IUniswapV2Router01.sol#14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
are-too-similar

setWhiteList(address) should be declared external:
        - DropLit.setWhiteList(address) (contracts/drop.sol#135-142)
setPairDshare(address) should be declared external:
        - DropLit.setPairDshare(address) (contracts/drop.sol#145-148)
setAdmin(address) should be declared external:
        - DropLit.setAdmin(address) (contracts/drop.sol#150-152)
setOracle(address) should be declared external:
        - DropLit.setOracle(address) (contracts/drop.sol#154-157)
setTaxCollectorAddress(address) should be declared external:
```

```
        - DropLit.setTaxCollectorAddress(address) (contracts/drop.sol#160-163)
setTaxRate(uint256) should be declared external:
        - DropLit.setTaxRate(uint256) (contracts/drop.sol#166-169)
mint(address,uint256) should be declared external:
        - DropLit.mint(address,uint256) (contracts/drop.sol#171-176)
operator() should be declared external:
        - Operator.operator() (contracts/libraries/Operator.sol#18-20)
isOperator() should be declared external:
        - Operator.isOperator() (contracts/libraries/Operator.sol#27-29)
transferOperator(address) should be declared external:
        - Operator.transferOperator(address) (contracts/libraries/Operator.sol#31-33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
. analyzed (17 contracts with 78 detectors), 62 result(s) found
```

0x Guard