SMART CONTRACT AUDIT

DAO Maker

PROJECT: DAOMAKER

METHODOLOGY

Main tests list:

- Best code practices
- ERC20/BEP20 compliance (if applicable)
- Logical bugs
- General Denial Of Service(DOS)
- Locked ether
- Private data leaks
- Using components with known vulns
- Weak PRNG
- Unsed vars
- Uncheked call return method
- Code with no effects
- Function visibility
- Use of deprecated functions
- Authorization issues
- Re-entrancy
- Arithmetic Over/Under Flows
- Hidden Malicious Code
- External Contract Referencing
- Short Address/ Parameter Attack
- Race Conditions / Front Running
- Uninitialized Storage Pointers
- Floating Points and Precision
- Signatures Replay
- Pool Asset Security (backdoors in the underlying ERC-20)

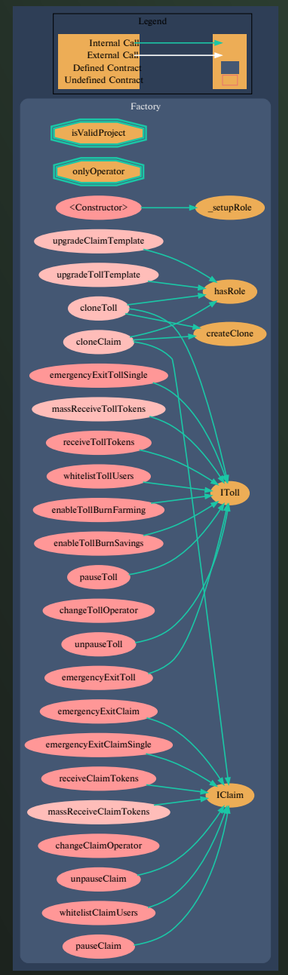
STRUCTURE OF CONTRACT
FACTORY.SOL

Contrcutor lacks for 0 address check

Contract methods analysis

upgradeTollTemplate(address newTemplate)
Method lacks for 0 address check

upgradeClaimTemplate(address newTemplate)
Method lacks for 0 address check



Pic. 1.1.
Factory.sol

```
cloneToll(
  uint256 _start,
  address _token,
  address _operator,
  uint256 _tollFee,
  uint256 _distributionRate,
  uint256 _fullDistributionSeconds,
  uint64[] calldata _distributionDelays,
  uint64[] calldata _distributionPercents,
  bool _isBurnableToken,
  uint8 strategy
)
```

Operator who has an ability to clone tolls can assign toll operator, which can be any third party. We suggest that toll operators should be assigned only by default admin(owner), otherwise this may lead to problems with access and may cause financial loss if external operator's account will be compromised. Another option is to remove all cash withdrawals posibilities from toll operator

```
cloneClaim(
  address _operator,
  uint256 _start,
  uint64[] calldata _releasePeriods,
  uint64[] calldata _releasePercents,
  address _token,
  uint8 strategy
)
```

Operator who has an ability to clone claims can assign claim operator, which can be any third party. We suggest that claim operators should be assigned only by default admin(owner), otherwise this may lead to problems with access and may cause financial loss if external operator's account will be compromised. Another option is to remove all cash withdrawals posibilities from claim operator

```
whitelistTollUsers(
  address _toll,
  address[] memory _users,
  uint128[] memory _amounts
)
```

Vulnerbilities not detected

enableTollBurnFarming(
address _toll,
uint256 _farmId,
address _farm,
address _farmManager,
address _burnReceiver,
uint256 _percent
)

Toll operator has an ability to provide farm address and burnReceiver address. Tokens will be transferred to that address, so if any malicious address provided there will be tokens lack. We suggest that farm and burnReceiver addresses should be used as global variables and only owner of this contract must have an ability to change them

enableTollBurnSavings(
address _toll,
address _burnReceiver,
uint256 _percent
)

Toll operator has an ability to provide burnReceiver address. Tokens will be transferred to that address, so if any malicious address provided there will be tokens lack. We suggest that burnReceiver address should be used as global variable and only owner of this contract must have an ability to change it.

pauseToll(address _toll)

Vulnerabilities not detected

unpauseToll(address _toll)

Vulnerabilities not detected

emergencyExitToll(
address _toll,
address _receiver
)

Toll operators have an ability to transfer all users tokens to the desired account and this is a serious issue.
All methods that can perform any emergency cash withdrawals should be performed only by owner, or at least tokens shouls be transferred to hardcoded owner's address.

emergencyExitTollSingle(
address _toll,
address _investor,
address _receiver
)

Toll operators have an ability to transfer certain user's tokens to the desired account and this is a serious issue.
All methods that can perform any emergency cash withdrawals should be performed only by owner, or at least tokens shouls be transferred to hardcoded owner's address.

changeTollOperator(
address _toll,
address _newOperator
)
We suggest that only only person who has an OPERATOR ROLE should be able to change toll's operator in order to prevent security risks.

massReceiveTollTokens(
address _toll,
address[] calldata _receivers
)
Vulnerabilities not detected

whitelistClaimUsers(
address _claim,
address[] memory _users,
uint128[] memory _amounts
)
Vulnerabilities not detected

pauseClaim(address _claim)
Vulnerabilities not detected

unpauseClaim(address _claim)
Vulnerabilities not detected

emergencyExitClaim(
address _claim,
address _receiver
)
Claim operators have an ability to transfer all users tokens to the desired account and this is a serious issue. All methods that can perform any emergency cash withdrawals should be performed only by owner, or at least tokens shouls be transferred to hardcoded owner's address. Method lacks checks for 0 addresses.

emergencyExitClaimSingle(
address _claim,
address _investor,
address _receiver
)
Claim operators have an ability to transfer certain user's tokens to the desired account and this is a serious issue. All methods that can perform any emergency cash withdrawals should be performed only by owner, or at least tokens shouls be transferred to hardcoded owner's address. Method lacks checks for 0 addresses.

```
massReceiveClaimTokens(
  address _claim,
  address[] calldata _receivers
)
```
Vulnerabilities not detected

```
receiveTollTokens(
  address _toll,
  uint256 _amount
)
```
Vulnerabilities not detected

```
receiveClaimTokens(
  address _claim
)
```
Vulnerabilities not detected

STRUCTURE OF CONTRACT
TOLL.SOL

Contract methods analysis

addWhitelistedUsers(
  address[] calldata _usersArray,
  uint128[] calldata _amountsArray
)
Vulnerabilities not detected

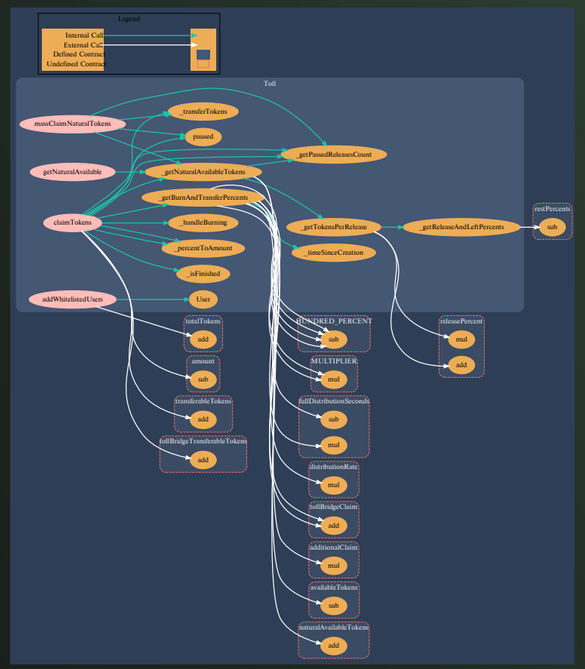
massClaimNaturalTokens(address[] calldata receivers)
Vulnerabilities not detected

claimTokens(
  address receiver,
  uint256 amount
)
Critical logic part, recommended to add nonReentrant modifier

getNaturalAvailable(address user)
Vulnerbilities not detected

_getBurnAndTransferPercents(address user)
Vulnerbilities not detected

_getNaturalAvailableTokens(address user)
Vulnerbilities not detected



Pic. 1.2.

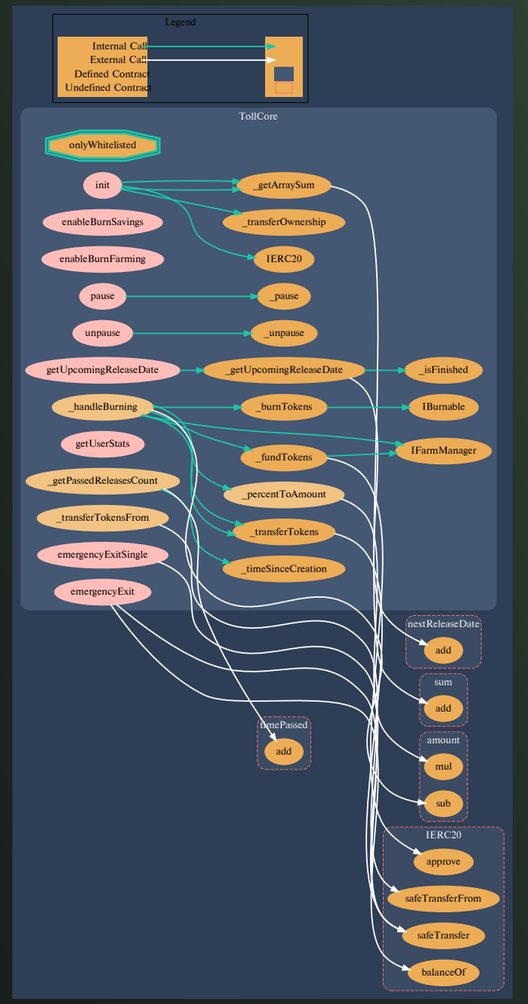
Toll.sol

STRUCTURE OF CONTRACT
TOLLCORE.SOL

Contract methods analysis

```
init(
  uint256 _start,
  address _token,
  uint256 _tollFee,
  uint256 _distributionRate,
  uint256 _fullDistributionSeconds,
  uint64[] calldata _distributionDelays,
  uint64[] calldata _distributionPercents,
  bool _isBurnableToken,
  address _burnValley,
  uint8 _strategy
 )
```

Recommened to use initializer modifier from Initializable.sol contract by openzepplein



Pic. 1.3.
TollCore.sol

enableBurnSavings(
  address _burnReceiver,
  uint256 _percent
)
Vulnerabilities not detected

enableBurnFarming(
  uint256 _farmId,
  address _farm,
  address _farmManager,
  address _burnReceiver,
  uint256 _percent
)
Vulnerabilities not detected

pause()
Vulnerabilities not detected

unpause()
Vulnerabilities not detected

emergencyExit(address receiver)
Vulnerabilities not detected

emergencyExitSingle(address investor, address receiver)
Vulnerabilities not detected

getUserStats(address user)
Vulnerabilities not detected

getUpcomingReleaseDate()
Vulnerabilities not detected

_getPassedReleasesCount()
recommended to cache distributionDelaysSum into local variable to optimize gas usage

_percentToAmount(uint256 amount, uint256 percent)
Vulnerabilities not detected

_timeSinceStart()
Vulnerabilities not detected

_handleBurning(uint256 amount)
Vulnerabilities not detected

_isFinished()
Vulnerabilities not detected

_transferTokens(address to, uint256 amount)
Vulnerabilities not detected

_transferTokensFrom(address from, address to, uint256 amount)
Vulnerabilities not detected

_burnTokens(uint256 amount)
Vulnerabilities not detected

_fundTokens(uint256 amount)
recommended to use SafeApprove

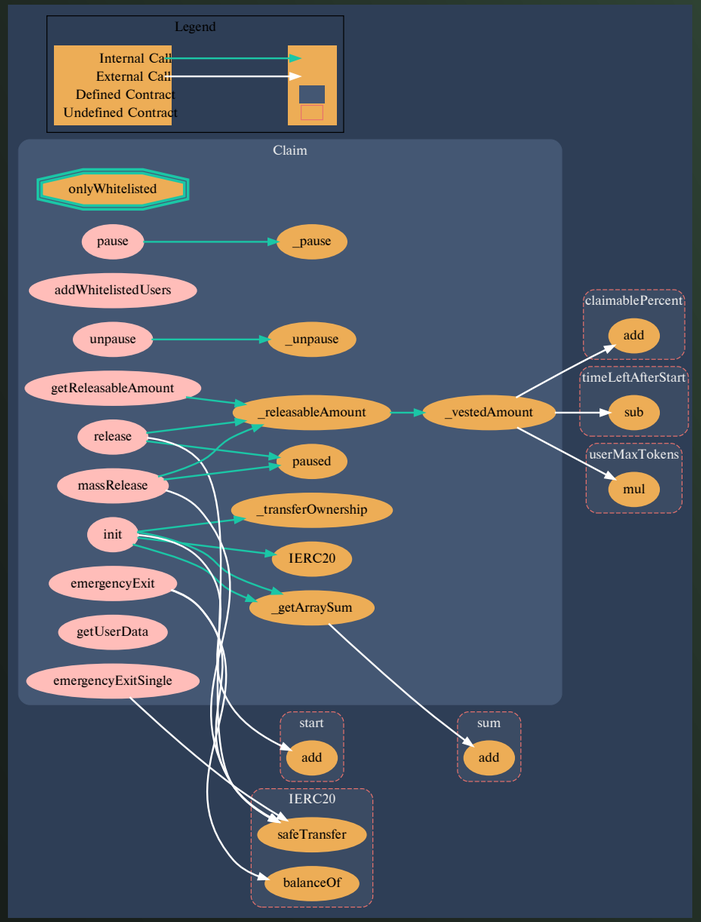
_buildArraySum(uint64[] calldata diffArray)
Vulnerabilities not detected

STRUCTURE OF CONTRACT
CLAIM.SOL

Contract methods analysis

init(
uint256 _start,
uint64[] calldata _releasePeriods,
uint64[] calldata _releasePercents,
address _token,
uint8 _strategy
)
Recommended to use initializer modifier from Initializable.sol contract by openzepplein

addWhitelistedUsers(
address[] calldata _usersArray,
uint128[] calldata _amountsArray
)
Vulnerabilities not detected



Pic. 1.4.
Claim.sol

pause()
Vulnerabilities not detected

unpause()
Vulnerabilities not detected

emergencyExit(address receiver)
Vulnerabilities not detected

emergencyExitSingle(address investor, address receiver)
Vulnerabilities not detected

release(address receiver)
Critical logic part, recommended to add nonReentrant modifier

massRelease(address[] calldata receivers)
Vulnerabilities not detected

getUserData(address user)
Vulnerabilities not detected

getReleasableAmount(address user)
Vulnerabilities not detected

_releasableAmount(address receiver)
Vulnerabilities not detected

_vestedAmount(address receiver)
Vulnerabilities not detected

_getPassedReleasesCount()
recommended to cache releasePeriodsSum into local variable to optimize gas usage

_buildArraySum(uint64[] calldata diffArray)
Vulnerabilities not detected

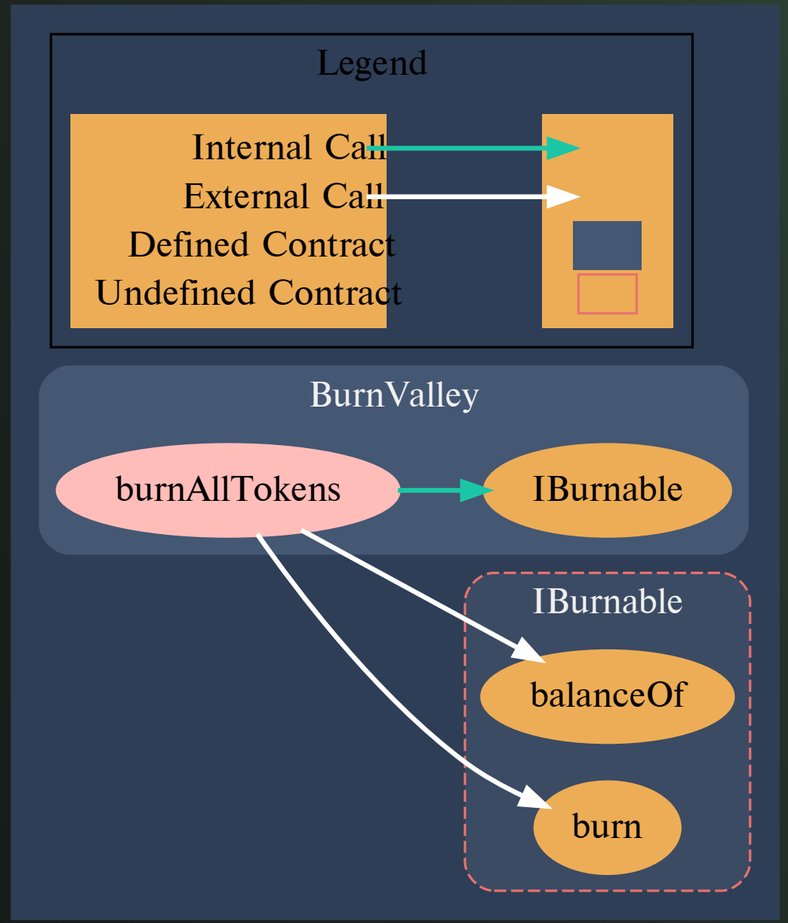
_timeSinceStart()
Vulnerabilities not detected

_isFinished()
Vulnerabilities not detected

STRUCTURE OF CONTRACT
BURNVALLEY.SOL

Contract methods analysis

burnAllTokens(address _token)
Vulnerabilities not detected



Pic. 1.5.
BurnValley.sol

GET IN TOUCH

info@smartstate.tech
smartstate.tech