CimpleDAO

Smart Contract Security Audit

Prepared by ShellBoxes

March 4th, 2022 - May 13th, 2022

Shellboxes.com

contact@shellboxes.com

Document Properties

Client	CimpleDAO
Target	CimpleDAO Smart Contracts
Version	1.0
Classification	Public

Scope

Files	MD5 Hash
CimpleDAO.sol	8f4fd9e465e15770f8244e82a95243ae
NFT.sol	14b66b766121e47ee6e5df836d60f64b
NFTUtils.sol	d4faea4bd332360a58881005f9722121
VoteUtils.sol	605e4b118fe5d4b42ecf4ba8fc4104d6

Contacts

VERSION	COMPANY	EMAIL
0.1	ShellBoxes	contact@shellboxes.com

Contents

1 Introduction

CimpleDAO engaged ShellBoxes to conduct a security assessment on the CimpleDAO beginning on March 4th, 2022 and ending May 13th, 2022. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About CimpleDAO

CiMPLE is an easy way for people to multiply their crypto assets utilizing previously complex and relatively unknown strategies. There are 9 CiMPLE strategies to help users multiply their crypto assets. The primary CiMPLE strategy is to leverage against existing tokens to buy more tokens when the price dips and repay those loans when the price goes back up, netting additional tokens and allowing for a larger purchase on the next dip. This can compound over time and result in a multiple of the original token balance. Our platform will make it simple enough for users to utilise the power of Defi seamlessly.

Issuer	CimpleDAO
Website	`http://www.cimple.cc/`
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

ShellBoxes used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart

contracts and can quickly detect code that does not comply with security best practices.

1.2.1 Risk Methodology

Vulnerabilities or bugs identified by ShellBoxes are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.

Impact				
	High	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low
		High	Medium	Low

Likelihood

2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the CimpleDAO implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include 3 critical-severity, 1 high-severity, 1 medium-severity, 10 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
User Can Burn Anyone's Tokens	CRITICAL	Acknowledged
Voter Can Vote Infinitely	CRITICAL	Acknowledged
User Can Mint All the Supply	CRITICAL	Acknowledged
Take all the referral fees	HIGH	Acknowledged
Votables Can Add Users The votable list	MEDIUM	Acknowledged
For Loop Over Dynamic Array	LOW	Acknowledged
Missing Address Verification	LOW	Acknowledged
Renounce Ownership	LOW	Acknowledged
Floating Pragma	LOW	Acknowledged
Renounce Ownership	LOW	Acknowledged
Floating Pragma	LOW	Acknowledged
Missing Address Verification	LOW	Acknowledged
Floating Pragma	LOW	Acknowledged
For Loop Over Dynamic Array	LOW	Acknowledged

Missing Address Verification	LOW	Acknowledged

3 Finding Details

A CimpleDAO.sol

A.1 User Can Burn Anyone's Tokens [CRITICAL]

Description:

The following functions are missing a critical verification in the address argument. Concerning the payFeeByToken function, the user can provide any address to be the spender, and therefore he can burn any amount of any user's tokens. For the createStake function, the user can provide any address to be the staker, which means another user's Cimple tokens will be burned.

Code:

Listing 1: CimpleDAO.sol

```
265 function createStake(address staker, uint256 _stake) public payable
        ↪ returns (bool) {
266     require(_stake <= balanceOf(staker, Cimple), 'Error stake amount
            ↪ must be >= holding amount of Cimple Token');

268     (bool _isStakeholder, uint256 s) = isStakeholder(staker);
269     if(!_isStakeholder) addStakeholder(staker, block.timestamp);
270     else {
271         StakeHolder memory stakeholder = stakeholders[s];
272         uint256 rewardOfCMPG = 0;
273         (rewardOfCMPG, , ) = calculateReward(staker, stakeholder.
                ↪ holdTimeStamp, block.timestamp);
274         _mint(staker, CMPG, rewardOfCMPG, "0x000");
275         tokenSupply[CMPG] = tokenSupply[CMPG].add(rewardOfCMPG);

277         stakeholders[s].holdTimeStamp = block.timestamp;
278     }
```

```
279     _burn(staker, Cimple, _stake);
280     _mint(staker, stCimple, _stake, "0x000");
281     tokenSupply[stCimple] = tokenSupply[stCimple].add(_stake);
282     tokenSupply[Cimple] = tokenSupply[Cimple].sub(_stake);
283     tokenBurn[Cimple] = tokenBurn[Cimple].add(_stake);
284     _addOrUpdateUserInfo(staker);
285     emit StakingCimpleToken(staker, stCimple, _stake);
286     return true;
287 }
```

Listing 2: CimpleDAO.sol

```
413 function payFeeByToken(address spender, uint256 cimpleAmount) public
        ↪ returns(bool) {
414     require(cimpleAmount > 0, "Error, amount of pay must be > 0");
415     _burn(spender, Cimple, cimpleAmount);
416     tokenSupply[Cimple] = tokenSupply[Cimple].sub(cimpleAmount);
417     tokenBurn[Cimple] = tokenBurn[Cimple].add(cimpleAmount);
418     _addOrUpdateUserInfo(spender);
419     emit PayFee(spender, Cimple, cimpleAmount);
420     return true;
421 }
```

Risk Level:

Likelihood – 5
Impact - 5

Recommendation:

The staker and spender addresses should be replaced with msg.sender to make sure that the user can only spend/burn his own tokens.

Status – Acknowledged

The Cimple team has acknowledged the risk.

A.2 Voter Can Vote Infinitely [CRITICAL]

Description:

The requestVoteAction function use an address coming from the arguments to identify the voter, which leaves the possibility of someone calling the function with different addresses and therefore getting an infinite number of votes.

Code:

Listing 3: CimpleDAO.sol

```
74 function requestVoteAction (address voteaddress, uint256 vote_id ,bool
       ↪ proposal) external votable returns (bool ) {
75     bool vd = voteUtils.voteAction(voteaddress, vote_id, proposal);
76     return vd;
77 }
```

Risk Level:

Likelihood – 5
Impact - 4

Recommendation:

The voteaddress should be replaced with msg.sender to make sure that the user can only vote for himself.

Status - Acknowledged

The Cimple team has acknowledged the risk.

A.3 Take all the referral fees [HIGH]

Description:

Any user can call the createAccount function to create a new account with any address and put his address as referralID, in addition to that he can also update the existent accounts putting his address as referredByID. That will make him get a percentage of all the fees paid by the referrals.

Code:

Listing 4: CimpleDAO.sol

```
203 function createAccount (address userAddress, string memory referralID)
        ↪ public payable returns(bool) {
204     (bool _isUser, ) = isUser(userAddress);
205     if(!_isUser) {
206         usersInfo.push(UserDetail(userAddress,balanceOf(userAddress,
                ↪ Cimple), balanceOf(userAddress, stCimple), balanceOf(
                ↪ userAddress, CMPG), referralID, ""));
207         emit CreatedNewUser(userAddress, referralID);
208     }
209     return true;
210 }
```

Listing 5: CimpleDAO

```
211 function updateUserInfo (address userAddress, string memory referredByID
        ↪ ) public payable returns (bool) {
212     (bool _isUser, uint256 s) = isUser(userAddress);
213     if(_isUser) {
214         UserDetail storage tempUser = usersInfo[s];
215         tempUser.cimpleValue = balanceOf(userAddress, Cimple);
216         tempUser.stCimpleValue = balanceOf(userAddress, stCimple);
217         tempUser.CMPGValue = balanceOf(userAddress, CMPG);
218         tempUser.referredBy = referredByID;
```

```
219         emit UpdatedUserInfo(userAddress, tempUser.cimpleValue, tempUser.
                ↪ stCimpleValue, tempUser.CMPGValue, tempUser.referralID,
                ↪ referredByID);
220     }
221     return true;
222 }
```

Risk Level:

Likelihood – 4
Impact - 3

Recommendation:

It is recommended to use msg.sender instead of the userAddress argument to ensure that the user can only create/modify his account.

Status - Acknowledged

The Cimple team has acknowledged the risk.

A.4 Votables Can Add Users The votable list [MEDIUM]

Description:

The users that are included in the votablelist can include other users by transferring enough tokens to them to meet the condition, and they will be added once the _addOrUpdateUserInfo function gets called. Also, the mintRoleList users can also include anyone in votablelist using the mint function.

Code:

Listing 6: CimpleDAO.sol

```
111 function mint(address account, uint256 id, uint256 amount) public
        ↪ mintable {
```

```
112     _mint(account, id, amount, "0x000");
113     tokenSupply[id] = tokenSupply[id].add(amount);
114     _addOrUpdateUserInfo(account);
115 }
```

Listing 7: CimpleDAO.sol

```
180 function _addOrUpdateUserInfo(address userAddress) internal {
181     (bool _isUser, uint256 s) = isUser(userAddress);
182     if(_isUser){
183         UserDetail storage tempUser = usersInfo[s];
184         tempUser.cimpleValue = balanceOf(userAddress, Cimple);
185         tempUser.stCimpleValue = balanceOf(userAddress, stCimple);
186         tempUser.CMPGValue = balanceOf(userAddress, CMPG);
187         if ((tempUser.CMPGValue*100)/tokenSupply[CMPG] > 10) {
188             votablelist[userAddress] = true;
189         }
190         else {
191             votablelist[userAddress] = false;
192         }
193         if ((tempUser.CMPGValue*100)/tokenSupply[CMPG] > 1) {
194             votecreatablelist[userAddress] = true;
195         }
196         else {
197             votecreatablelist[userAddress] = false;
198         }
199     }else{
200         usersInfo.push(UserDetail(userAddress, 0, 0, 0 , "", ""));
201     }
202 }
```

Risk Level:

Likelihood – 3

Impact - 3

Recommendation:

It is recommended to call the _addOrUpdateUserInfo function in the _afterTokenTransfer to ensure that the votablelist gets updated after every transfer.

Status - Acknowledged

The Cimple team has acknowledged the risk.

A.5 For Loop Over Dynamic Array [LOW]

Description:

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial-of-Service. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Code:

Listing 8: CimpleDAO.sol

```
83      function multipleAddressesToMintableRoleList(address[] memory
            ↪ addresses) public onlyOwner {
84          for(uint256 i =0; i < addresses.length; i++) {
85              singleAddressToMintableRoleList(addresses[i]);
86          }
87      }
```

Listing 9: CimpleDAO.sol

```
118  function mintBatch(address to, uint256[] memory ids, uint256[] memory
         ↪ amounts, bytes memory data) public onlyOwner {
119      _mintBatch(to, ids, amounts, data);
120      for(uint256 i = 0; i < ids.length; i++){
```

```
121         tokenSupply[ids[i]] = tokenSupply[ids[i]].add(amounts[i]);
122     }
123     _addOrUpdateUserInfo(to);
124 }
```

Listing 10: CimpleDAO.sol

```
174 function isUser(address _address) public view returns(bool, uint256) {
175    for (uint256 s = 0; s < usersInfo.length; s += 1){
176        if (_address == usersInfo[s].userAddress) return (true, s);
177    }
178    return (false, 0);
179 }
```

Listing 11: CimpleDAO.sol

```
426 function claimFirstCimpleForNFT(address _userAddress, uint256 _tokenID)
      ↪ public payable returns(bool){
427     (uint256[] memory _tokenIDs, uint256[] memory _prices, ) = nftUtils.
          ↪ filterNftDetail(_userAddress);
428     bool _flag = false;
429     uint256 _selectedTokenPrice;
430     for (uint256 i = 0; i < _tokenIDs.length; i += 1){
431         if(_tokenID == _tokenIDs[i]){
432             _flag = true;
433             _selectedTokenPrice = _prices[i];
434         }
435     }
```

Risk Level:

Likelihood – 3

Impact - 2

Recommendation:

Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Status - Acknowledged

The Cimple team has acknowledged the risk.

A.6 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, some of the contract's functionality may become inaccessible.

Code:

Listing 12: CimpleDAO.sol

```
66 constructor(address _nftUtils, address _voteUtils) ERC1155("") {
67     deployedStartTimeStamp = block.timestamp;
68     nftUtils = NFTUtils(_nftUtils);
69     voteUtils = VoteUtils(_voteUtils);
70 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is recommended to undertake further validation on the user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Status - Acknowledged

The Cimple team has acknowledged the risk.

A.7 Renounce Ownership [LOW]

Description:

Typically, the contract's owner is the account that deploys the contract. As a result, the owner can perform certain privileged activities. The renounceOwnership function is used in smart contracts to renounce ownership. However, if the contract's ownership has never been transferred before renouncing it, it will never have an Owner, which may result in a denial of service.

Code:

Listing 13: CimpleDAO.sol

```
9 contract CimpleDAO is ERC1155, Ownable {
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method will require two or more users to sign the transaction. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

Status - Acknowledged

The Cimple team has acknowledged the risk.

A.8 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.0. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 14: CimpleDAO.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Acknowledged

The Cimple team has acknowledged the risk.

B NFT.sol

B.1 User Can Mint All the Supply [CRITICAL]

Description:

The user can mint NFTs as much as he wants without any condition or an amount of Ether that he needs to pay. Thus, anyone can mint all the tokens available in the supply.

Code:

Listing 15: NFT.sol

```
35  function mintTo(address recipient)
36      public
37      payable
38      returns (uint256)
39  {
40      uint256 tokenId = currentTokenId.current();
41      require(tokenId < TOTAL_SUPPLY, "Max supply reached");
42      currentTokenId.increment();
43      uint256 newItemId = currentTokenId.current();
44      _safeMint(recipient, newItemId);
45      return newItemId;
46  }
```

Risk Level:

Likelihood – 2
Impact - 3

Recommendation:

Restrict the mint access to a specific entity or add some conditions to prevent unlimited minting.

Status - Acknowledged

The Cimple team has acknowledged the risk

B.2 Renounce Ownership [LOW]

Description:

Typically, the contract's owner is the account that deploys the contract. As a result, the owner can perform certain privileged activities. The renounceOwnership function is used in smart contracts to renounce ownership. However, if the contract's ownership has never been transferred before renouncing it, it will never have an Owner, which may result in a denial of service.

Code:

Listing 16: NFT.sol

```
9 contract NFT is ERC721Enumerable, PullPayment, Ownable
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method will require two or more users to sign the transaction. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

Status - Acknowledged

The Cimple team has acknowledged the risk.

B.3 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.0. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version that may introduce issues in the contract system.

Code:

Listing 17: NFT.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Acknowledged

The Cimple team has acknowledged the risk.

C NFTUtils.sol

C.1 Missing Address Verification [LOW]

Description:

Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Code:

Listing 18: NFTUtils.sol

```
24 constructor(address nftaddress) {
25     CiMPLENFTaddress = nftaddress;
26     owner = msg.sender;
27 }
```

Listing 19: NFTUtils

```
29 function changeOwner(address _newOwner) public {
30     require(msg.sender == owner, 'failed');
31     owner = _newOwner;
32 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is recommended to undertake further validation on the user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Status - Acknowledged

The Cimple team has acknowledged the risk.

C.2 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.0. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version that may introduce issues in the contract system.

Code:

Listing 20: NFTUtils.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Acknowledged

The Cimple team has acknowledged the risk.

D VoteUtils.sol

D.1 For Loop Over Dynamic Array [LOW]

Description:

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial-of-Service. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Code:

Listing 21: VoteUtils.sol

```
66 function _checkVoteAddress(address p_voteaddress, address[] memory
       ↪ p_voters) private pure returns (bool ) {
67     bool checkflag = false;
68     for (uint i = 0; i < p_voters.length; i++)
69     {
70         if (p_voters[i] == p_voteaddress) {
71             checkflag = true;
72         }
73     }
74     return checkflag;
75 }
```

Risk Level:

Likelihood – 3

Impact - 2

Recommendation:

Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Status - Acknowledged

The Cimple team has acknowledged the risk.

D.2 Missing Address Verification [LOW]

Description:

Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Code:

Listing 22: VoteUtils.sol

```
29 function changeOwner(address _newOwner) public {
30     require(msg.sender == owner, 'failed');
31     owner = _newOwner;
32 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is recommended to undertake further validation on the user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Status - Acknowledged

The Cimple team has acknowledged the risk.

4 Best Practices

BP.1 Unnecessary variable initialization:

Description:

When a variable is declared in solidity, it gets initialized with its type's default value. Thus, there is no need to initialize a variable with the default value.

Code:

Listing 23: CimpleDAO.sol

```
13 uint256 public constant Cimple = 0;
```

Listing 24: CimpleDAO.sol

```
43 uint256 public votablelistcounter = 0;
```

Listing 25: CimpleDAO.sol

```
45 uint256 public votecreatablelistcounter = 0;
```

Listing 26: CimpleDAO.sol

```
274 uint256 rewardOfCMPG = 0;
```

Listing 27: CimpleDAO.sol

```
294 uint256 rewardOfCMPG = 0;
```

Listing 28: CimpleDAO.sol

```
428 bool _flag = false;
```

Listing 29: VoteUtils.sol

```
67 bool checkflag = false;
```

BP.2 Payable Functions That Do Not Require ETH transfer:

Description:

Functions that do not require ETH transfer should not be declared as payable.

Code:

Listing 30: CimpleDAO.sol

```
71 function requestVoteProposal (address createaddress, string memory des,
       ↪ uint256 endtime, uint256 starttime) public payable votecreatable
       ↪ {
72     voteUtils.makeproposal(createaddress, des, endtime, starttime);
73 }
```

Listing 31: CimpleDAO.sol

```
203 function createAccount (address userAddress, string memory referralID)
       ↪ public payable returns(bool) {
204     (bool _isUser, ) = isUser(userAddress);
205     if(!_isUser) {
206         usersInfo.push(UserDetail(userAddress,balanceOf(userAddress,
               ↪ Cimple), balanceOf(userAddress, stCimple), balanceOf(
               ↪ userAddress, CMPG), referralID, ""));
207         emit CreatedNewUser(userAddress, referralID);
208     }
209     return true;
210 }
```

Listing 32: CimpleDAO.sol

```
211 function updateUserInfo (address userAddress, string memory referredByID
       ↪ ) public payable returns (bool) {
212     (bool _isUser, uint256 s) = isUser(userAddress);
213     if(_isUser) {
```

```
214         UserDetail storage tempUser = usersInfo[s];
215         tempUser.cimpleValue = balanceOf(userAddress, Cimple);
216         tempUser.stCimpleValue = balanceOf(userAddress, stCimple);
217         tempUser.CMPGValue = balanceOf(userAddress, CMPG);
218         tempUser.referredBy = referredByID;
219         emit UpdatedUserInfo(userAddress, tempUser.cimpleValue, tempUser.
                ↪ stCimpleValue, tempUser.CMPGValue, tempUser.referralID,
                ↪ referredByID);
220     }
221     return true;
222 }
```

Listing 33: CimpleDAO.sol

```
267 function createStake(address staker, uint256 _stake) public payable
        ↪ returns (bool) {
268     require(_stake <= balanceOf(staker, Cimple), 'Error stake amount
            ↪ must be >= holding amount of Cimple Token');
269
270     (bool _isStakeholder, uint256 s) = isStakeholder(staker);
271     if(!_isStakeholder) addStakeholder(staker, block.timestamp);
272     else {
```

Listing 34: CimpleDAO.sol

```
290 function removeStake(address unstaker, uint256 _stake) public payable
        ↪ returns ( bool ) {
291     require(_stake <= balanceOf(unstaker, stCimple), 'Error, unstake
            ↪ amount must be >= holding amount of stCimple Token.');
292     (bool _isStakeholder, uint256 s) = isStakeholder(unstaker);
293     StakeHolder memory stakeholder = stakeholders[s];
294     uint256 rewardOfCMPG = 0;
```

Listing 35: CimpleDAO.sol

```
426 function claimFirstCimpleForNFT(address _userAddress, uint256 _tokenID)
        ↪ public payable returns(bool){
```

```
427     (uint256[] memory _tokenIDs, uint256[] memory _prices, ) = nftUtils.
            ↪ filterNftDetail(_userAddress);
428     bool _flag = false;
```

Listing 36: CimpleDAO.sol

```
450 function claimRewardStreamingForNFT(address _address, uint256 _tokenID)
        ↪ public payable returns( bool) {
451     (uint256 _cimpleIR, ,) = calculateCimpleIR(block.timestamp);
452     (uint256 _rate, uint256 _mintTimestamp) = nftUtils.
            ↪ calculateStreamingRateForNFT(_address, _tokenID, block.
            ↪ timestamp, _cimpleIR);
453     if(_rate > 0) {
```

Listing 37: NFTUtils.sol

```
45 function setNftAwardList(uint256 _tokenID, bool _flag) external payable
       ↪ isOwner {
46     nftAwardList[_tokenID] = _flag;
47 }
```

Listing 38: NFTUtils.sol

```
50 function increaseNftAwardListCount() external payable isOwner {
51     nftAwardListCount++;
52 }
```

BP.3 Two Loops Can Be Reduced to One

Description:

The _mintBatch and _burnBatch functions in the ERC1155 implementations contain a loop over the ids array. Here in the mintBatch and the burnBatch functions a loop over ids is added to the logic, meanwhile the logic can be implemented using one loop.

Code:

Listing 39: CimpleDAO.sol

```
116 function mintBatch(address to, uint256[] memory ids, uint256[] memory
        ↪ amounts, bytes memory data) public onlyOwner {
117     _mintBatch(to, ids, amounts, data);
118     for(uint256 i = 0; i < ids.length; i++){
119         tokenSupply[ids[i]] = tokenSupply[ids[i]].add(amounts[i]);
120     }
121     _addOrUpdateUserInfo(to);
122 }
```

Listing 40: CimpleDAO.sol

```
129 function burnBatch(address from, uint256[] memory ids, uint256[] memory
        ↪ amounts) public onlyOwner {
130     _burnBatch(from, ids, amounts);
131     for(uint256 i = 0; i < ids.length; i++)
```

BP.4 Loop Can Be Avoided Using a Mapping:

Description:

The loop here is unnecessary, a mapping can be used instead to find out if a user has already created an account. The code in this case will be more efficient and consumes less gaz.

Code:

Listing 41: CimpleDAO.sol

```
172 function isUser(address _address) public view returns(bool, uint256) {
173    for (uint256 s = 0; s < usersInfo.length; s += 1){
174        if (_address == usersInfo[s].userAddress) return (true, s);
175    }
176    return (false, 0);
177 }
```

BP.5 Add A Break When Address Is Found:

Description:

There is a missing break after finding the p_voteaddress to reduce the number of iterations.

Code:

Listing 42: VoteUtils.sol

```
66 function _checkVoteAddress(address p_voteaddress, address[] memory
       ↪ p_voters) private pure returns (bool ) {
67     bool checkflag = false;
68     for (uint i = 0; i < p_voters.length; i++)
69     {
70         if (p_voters[i] == p_voteaddress) {
71             checkflag = true;
72         }
73     }
74     return checkflag;
75 }
```

5 Static Analysis (Slither)

Description:

ShellBoxes expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

Results:

```
Compilation warnings/errors on contracts/CimpleDAO.sol:
Warning: Contract code size is 36349 bytes and exceeds 24576 bytes (a
   ↪ limit introduced in Spurious Dragon). This contract may not be
   ↪ deployable on mainnet. Consider enabling the optimizer (with a
   ↪ low "runs" value!), turning off revert strings, or using
   ↪ libraries.
 --> contracts/CimpleDAO.sol:9:1:
  |
9 | contract CimpleDAO is ERC1155, Ownable { // [+] Renounce Ownership
  | ^ (Relevant source part starts here and spans across multiple lines)
    ↪ .

CimpleDAO.calculateCimpleIR(uint256) (contracts/CimpleDAO.sol#136-171)
   ↪ performs a multiplication on the result of a division:
       -temp = additionalDailyRatePerYear[i - 1] * 175 / 100 (contracts/
          ↪ CimpleDAO.sol#155)
       -temp = temp * 9810837779 / 1e10 (contracts/CimpleDAO.sol#157)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #divide-before-multiply
```

```
CimpleDAO.isStakeholder(address) (contracts/CimpleDAO.sol#241-246) uses
    ↪ a dangerous strict equality:
        - _address == stakeholders[s].holderAddress (contracts/CimpleDAO.
            ↪ sol#243)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dangerous-strict-equalities

Contract locking ether found:
        Contract CimpleDAO (contracts/CimpleDAO.sol#9-475) has payable
            ↪ functions:
         - CimpleDAO.requestVoteProposal(address,string,uint256,uint256)
             ↪ (contracts/CimpleDAO.sol#70-72)
         - CimpleDAO.createAccount(address,string) (contracts/CimpleDAO.
             ↪ sol#201-208)
         - CimpleDAO.updateUserInfo(address,string) (contracts/CimpleDAO.
             ↪ sol#209-220)
         - CimpleDAO.createStake(address,uint256) (contracts/CimpleDAO.
             ↪ sol#265-287)
         - CimpleDAO.removeStake(address,uint256) (contracts/CimpleDAO.
             ↪ sol#288-314)
         - CimpleDAO.payFee() (contracts/CimpleDAO.sol#399-412)
         - CimpleDAO.claimFirstCimpleForNFT(address,uint256) (contracts/
             ↪ CimpleDAO.sol#422-444)
         - CimpleDAO.claimRewardStreamingForNFT(address,uint256) (
             ↪ contracts/CimpleDAO.sol#445-473)
        But does not have a function to withdraw the ether
Contract locking ether found:
        Contract NFTUtils (contracts/NFTUtils.sol#7-178) has payable
            ↪ functions:
         - NFTUtils.setNftAwardList(uint256,bool) (contracts/NFTUtils.sol
             ↪ #45-47)
         - NFTUtils.increaseNftAwardListCount() (contracts/NFTUtils.sol
             ↪ #50-52)
```

```
        - NFTUtils.addNFTUsersInfo(address,uint256,uint256) (contracts/
            ↪ NFTUtils.sol#80-85)
        But does not have a function to withdraw the ether
Contract locking ether found:
        Contract VoteUtils (contracts/VoteUtils.sol#4-97) has payable
            ↪ functions:
        - VoteUtils.makeproposal(address,string,uint256,uint256) (
            ↪ contracts/VoteUtils.sol#33-37)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #contracts-that-lock-ether

Reentrancy in CimpleDAO.createStake(address,uint256) (contracts/
    ↪ CimpleDAO.sol#265-287):
        External calls:
        - _mint(staker,CMPG,rewardOfCMPG,0x000) (contracts/CimpleDAO.sol
            ↪ #274)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,
                    ↪ amount,data) (node_modules/@openzeppelin/contracts/
                    ↪ token/ERC1155/ERC1155.sol#476-484)
        State variables written after the call(s):
        - _burn(staker,Cimple,_stake) (contracts/CimpleDAO.sol#279)
                - _balances[id][from] = fromBalance - amount (node_modules
                    ↪ /@openzeppelin/contracts/token/ERC1155/ERC1155.sol
                    ↪ #349)
        - stakeholders[s].holdTimeStamp = block.timestamp (contracts/
            ↪ CimpleDAO.sol#277)
Reentrancy in CimpleDAO.createStake(address,uint256) (contracts/
    ↪ CimpleDAO.sol#265-287):
        External calls:
        - _mint(staker,CMPG,rewardOfCMPG,0x000) (contracts/CimpleDAO.sol
            ↪ #274)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,
                    ↪ amount,data) (node_modules/@openzeppelin/contracts/
```

```
                ↪ token/ERC1155/ERC1155.sol#476-484)
        - _mint(staker,stCimple,_stake,0x000) (contracts/CimpleDAO.sol
            ↪ #280)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,
                    ↪ amount,data) (node_modules/@openzeppelin/contracts/
                    ↪ token/ERC1155/ERC1155.sol#476-484)
        State variables written after the call(s):
        - _mint(staker,stCimple,_stake,0x000) (contracts/CimpleDAO.sol
            ↪ #280)
                - _balances[id][to] += amount (node_modules/@openzeppelin/
                    ↪ contracts/token/ERC1155/ERC1155.sol#280)
        - tokenSupply[stCimple] = tokenSupply[stCimple].add(_stake) (
            ↪ contracts/CimpleDAO.sol#281)
        - tokenSupply[Cimple] = tokenSupply[Cimple].sub(_stake) (
            ↪ contracts/CimpleDAO.sol#282)
    Reentrancy in CimpleDAO.removeStake(address,uint256) (contracts/
        ↪ CimpleDAO.sol#288-314):
        External calls:
        - _mint(unstaker,Cimple,_stake,0x000) (contracts/CimpleDAO.sol
            ↪ #296)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,
                    ↪ amount,data) (node_modules/@openzeppelin/contracts/
                    ↪ token/ERC1155/ERC1155.sol#476-484)
        - _mint(unstaker,CMPG,rewardOfCMPG,0x000) (contracts/CimpleDAO.
            ↪ sol#300)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,
                    ↪ amount,data) (node_modules/@openzeppelin/contracts/
                    ↪ token/ERC1155/ERC1155.sol#476-484)
        State variables written after the call(s):
        - _mint(unstaker,CMPG,rewardOfCMPG,0x000) (contracts/CimpleDAO.
            ↪ sol#300)
                - _balances[id][to] += amount (node_modules/@openzeppelin/
                    ↪ contracts/token/ERC1155/ERC1155.sol#280)
        - removeStakeholder(unstaker) (contracts/CimpleDAO.sol#303)
```

```
                - stakeholders[s] = stakeholders[stakeholders.length - 1]
                    ↪ (contracts/CimpleDAO.sol#254)
                - stakeholders.pop() (contracts/CimpleDAO.sol#255)
        - stakeholders[s].holdTimeStamp = block.timestamp (contracts/
            ↪ CimpleDAO.sol#305)
        - tokenSupply[CMPG] = tokenSupply[CMPG].add(rewardOfCMPG) (
            ↪ contracts/CimpleDAO.sol#301)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-1

ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,
    ↪ uint256[],uint256[],bytes).response (node_modules/@openzeppelin/
    ↪ contracts/token/ERC1155/ERC1155.sol#498) is a local variable
    ↪ never initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,
    ↪ uint256,bytes).response (node_modules/@openzeppelin/contracts/
    ↪ token/ERC1155/ERC1155.sol#476) is a local variable never
    ↪ initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,
    ↪ uint256,bytes).reason (node_modules/@openzeppelin/contracts/token
    ↪ /ERC1155/ERC1155.sol#480) is a local variable never initialized
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,
    ↪ uint256[],uint256[],bytes).reason (node_modules/@openzeppelin/
    ↪ contracts/token/ERC1155/ERC1155.sol#503) is a local variable
    ↪ never initialized
CimpleDAO.claimFirstCimpleForNFT(address,uint256)._selectedTokenPrice (
    ↪ contracts/CimpleDAO.sol#425) is a local variable never
    ↪ initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-local-variables

CimpleDAO.claimRewardStreamingForNFT(address,uint256) (contracts/
    ↪ CimpleDAO.sol#445-473) ignores return value by nftUtils.
    ↪ addNFTUsersInfo(_address,_tokenID,block.timestamp) (contracts/
```

```
    ↪ CimpleDAO.sol#454)
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,
    ↪ uint256,bytes) (node_modules/@openzeppelin/contracts/token/
    ↪ ERC1155/ERC1155.sol#467-486) ignores return value by
    ↪ IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,
    ↪ data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155
    ↪ .sol#476-484)
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,
    ↪ uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/
    ↪ token/ERC1155/ERC1155.sol#488-509) ignores return value by
    ↪ IERC1155Receiver(to).onERC1155BatchReceived(operator,from,ids,
    ↪ amounts,data) (node_modules/@openzeppelin/contracts/token/ERC1155
    ↪ /ERC1155.sol#497-507)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

NFTUtils.changeOwner(address) (contracts/NFTUtils.sol#29-32) should emit
    ↪  an event for:
        - owner = _newOwner (contracts/NFTUtils.sol#31)
VoteUtils.changeOwner(address) (contracts/VoteUtils.sol#29-32) should
    ↪ emit an event for:
        - owner = _newOwner (contracts/VoteUtils.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-events-access-control

NFTUtils.constructor(address).nftaddress (contracts/NFTUtils.sol#24)
    ↪ lacks a zero-check on :
                - CiMPLENFTaddress = nftaddress (contracts/NFTUtils.sol
                    ↪ #25)
NFTUtils.changeOwner(address)._newOwner (contracts/NFTUtils.sol#29)
    ↪ lacks a zero-check on :
                - owner = _newOwner (contracts/NFTUtils.sol#31)
VoteUtils.changeOwner(address)._newOwner (contracts/VoteUtils.sol#29)
    ↪ lacks a zero-check on :
```

```
                - owner = _newOwner (contracts/VoteUtils.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address
    ↪ ,uint256,uint256,bytes).response (node_modules/@openzeppelin/
    ↪ contracts/token/ERC1155/ERC1155.sol#476)' in ERC1155.
    ↪ _doSafeTransferAcceptanceCheck(address,address,address,uint256,
    ↪ uint256,bytes) (node_modules/@openzeppelin/contracts/token/
    ↪ ERC1155/ERC1155.sol#467-486) potentially used before declaration:
    ↪  response != IERC1155Receiver.onERC1155Received.selector (
    ↪ node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol
    ↪ #477)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address
    ↪ ,uint256,uint256,bytes).reason (node_modules/@openzeppelin/
    ↪ contracts/token/ERC1155/ERC1155.sol#480)' in ERC1155.
    ↪ _doSafeTransferAcceptanceCheck(address,address,address,uint256,
    ↪ uint256,bytes) (node_modules/@openzeppelin/contracts/token/
    ↪ ERC1155/ERC1155.sol#467-486) potentially used before declaration:
    ↪  revert(string)(reason) (node_modules/@openzeppelin/contracts/
    ↪ token/ERC1155/ERC1155.sol#481)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,
    ↪ address,uint256[],uint256[],bytes).response (node_modules/
    ↪ @openzeppelin/contracts/token/ERC1155/ERC1155.sol#498)' in
    ↪ ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,
    ↪ address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/
    ↪ contracts/token/ERC1155/ERC1155.sol#488-509) potentially used
    ↪ before declaration: response != IERC1155Receiver.
    ↪ onERC1155BatchReceived.selector (node_modules/@openzeppelin/
    ↪ contracts/token/ERC1155/ERC1155.sol#500)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,
    ↪ address,uint256[],uint256[],bytes).reason (node_modules/
    ↪ @openzeppelin/contracts/token/ERC1155/ERC1155.sol#503)' in
    ↪ ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,
```

```
    ↪ address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/
    ↪ contracts/token/ERC1155/ERC1155.sol#488-509) potentially used
    ↪ before declaration: revert(string)(reason) (node_modules/
    ↪ @openzeppelin/contracts/token/ERC1155/ERC1155.sol#504)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #pre-declaration-usage-of-local-variables

Reentrancy in CimpleDAO.claimFirstCimpleForNFT(address,uint256) (
    ↪ contracts/CimpleDAO.sol#422-444):
        External calls:
        - _mint(_userAddress,Cimple,_rewardCimpleAmount,0x000) (contracts
            ↪ /CimpleDAO.sol#436)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,
                    ↪ amount,data) (node_modules/@openzeppelin/contracts/
                    ↪ token/ERC1155/ERC1155.sol#476-484)
        State variables written after the call(s):
        - tokenSupply[Cimple] = tokenSupply[Cimple].add(
            ↪ _rewardCimpleAmount) (contracts/CimpleDAO.sol#437)
        - _addOrUpdateUserInfo(_userAddress) (contracts/CimpleDAO.sol
            ↪ #438)
                - tempUser.cimpleValue = balanceOf(userAddress,Cimple) (
                    ↪ contracts/CimpleDAO.sol#182)
                - tempUser.stCimpleValue = balanceOf(userAddress,stCimple)
                    ↪  (contracts/CimpleDAO.sol#183)
                - tempUser.CMPGValue = balanceOf(userAddress,CMPG) (
                    ↪ contracts/CimpleDAO.sol#184)
                - usersInfo.push(UserDetail(userAddress,0,0,0,,)) (
                    ↪ contracts/CimpleDAO.sol#198)
        - _addOrUpdateUserInfo(_userAddress) (contracts/CimpleDAO.sol
            ↪ #438)
                - votablelist[userAddress] = true (contracts/CimpleDAO.sol
                    ↪ #186)
                - votablelist[userAddress] = false (contracts/CimpleDAO.
                    ↪ sol#189)
```

```
        - _addOrUpdateUserInfo(_userAddress) (contracts/CimpleDAO.sol
            ↪ #438)
                - votecreatablelist[userAddress] = true (contracts/
                    ↪ CimpleDAO.sol#192)
                - votecreatablelist[userAddress] = false (contracts/
                    ↪ CimpleDAO.sol#195)
    Reentrancy in CimpleDAO.claimRewardStreamingForNFT(address,uint256) (
        ↪ contracts/CimpleDAO.sol#445-473):
        External calls:
        - nftUtils.addNFTUsersInfo(_address,_tokenID,block.timestamp) (
            ↪ contracts/CimpleDAO.sol#454)
        - nftUtils.setNFTUsersInfoByIndex(s,block.timestamp) (contracts/
            ↪ CimpleDAO.sol#460)
        - _mint(_address,Cimple,_amountOfReward,0x000) (contracts/
            ↪ CimpleDAO.sol#465)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,
                    ↪ amount,data) (node_modules/@openzeppelin/contracts/
                    ↪ token/ERC1155/ERC1155.sol#476-484)
        State variables written after the call(s):
        - tokenSupply[Cimple] = tokenSupply[Cimple].add(_amountOfReward)
            ↪ (contracts/CimpleDAO.sol#466)
        - _addOrUpdateUserInfo(_address) (contracts/CimpleDAO.sol#468)
                - tempUser.cimpleValue = balanceOf(userAddress,Cimple) (
                    ↪ contracts/CimpleDAO.sol#182)
                - tempUser.stCimpleValue = balanceOf(userAddress,stCimple)
                    ↪  (contracts/CimpleDAO.sol#183)
                - tempUser.CMPGValue = balanceOf(userAddress,CMPG) (
                    ↪ contracts/CimpleDAO.sol#184)
                - usersInfo.push(UserDetail(userAddress,0,0,0,,)) (
                    ↪ contracts/CimpleDAO.sol#198)
        - _addOrUpdateUserInfo(_address) (contracts/CimpleDAO.sol#468)
                - votablelist[userAddress] = true (contracts/CimpleDAO.sol
                    ↪ #186)
```

```
            - votablelist[userAddress] = false (contracts/CimpleDAO.
                ↪ sol#189)
        - _addOrUpdateUserInfo(_address) (contracts/CimpleDAO.sol#468)
            - votecreatablelist[userAddress] = true (contracts/
                ↪ CimpleDAO.sol#192)
            - votecreatablelist[userAddress] = false (contracts/
                ↪ CimpleDAO.sol#195)
Reentrancy in CimpleDAO.createStake(address,uint256) (contracts/
    ↪ CimpleDAO.sol#265-287):
    External calls:
    - _mint(staker,CMPG,rewardOfCMPG,0x000) (contracts/CimpleDAO.sol
        ↪ #274)
        - IERC1155Receiver(to).onERC1155Received(operator,from,id,
            ↪ amount,data) (node_modules/@openzeppelin/contracts/
            ↪ token/ERC1155/ERC1155.sol#476-484)
    State variables written after the call(s):
    - tokenSupply[CMPG] = tokenSupply[CMPG].add(rewardOfCMPG) (
        ↪ contracts/CimpleDAO.sol#275)
Reentrancy in CimpleDAO.createStake(address,uint256) (contracts/
    ↪ CimpleDAO.sol#265-287):
    External calls:
    - _mint(staker,CMPG,rewardOfCMPG,0x000) (contracts/CimpleDAO.sol
        ↪ #274)
        - IERC1155Receiver(to).onERC1155Received(operator,from,id,
            ↪ amount,data) (node_modules/@openzeppelin/contracts/
            ↪ token/ERC1155/ERC1155.sol#476-484)
    - _mint(staker,stCimple,_stake,0x000) (contracts/CimpleDAO.sol
        ↪ #280)
        - IERC1155Receiver(to).onERC1155Received(operator,from,id,
            ↪ amount,data) (node_modules/@openzeppelin/contracts/
            ↪ token/ERC1155/ERC1155.sol#476-484)
    State variables written after the call(s):
    - tokenBurn[Cimple] = tokenBurn[Cimple].add(_stake) (contracts/
        ↪ CimpleDAO.sol#283)
```

```
        - _addOrUpdateUserInfo(staker) (contracts/CimpleDAO.sol#284)
              - tempUser.cimpleValue = balanceOf(userAddress,Cimple) (
                 ↪ contracts/CimpleDAO.sol#182)
              - tempUser.stCimpleValue = balanceOf(userAddress,stCimple)
                 ↪  (contracts/CimpleDAO.sol#183)
              - tempUser.CMPGValue = balanceOf(userAddress,CMPG) (
                 ↪ contracts/CimpleDAO.sol#184)
              - usersInfo.push(UserDetail(userAddress,0,0,0,,)) (
                 ↪ contracts/CimpleDAO.sol#198)
        - _addOrUpdateUserInfo(staker) (contracts/CimpleDAO.sol#284)
              - votablelist[userAddress] = true (contracts/CimpleDAO.sol
                 ↪ #186)
              - votablelist[userAddress] = false (contracts/CimpleDAO.
                 ↪ sol#189)
        - _addOrUpdateUserInfo(staker) (contracts/CimpleDAO.sol#284)
              - votecreatablelist[userAddress] = true (contracts/
                 ↪ CimpleDAO.sol#192)
              - votecreatablelist[userAddress] = false (contracts/
                 ↪ CimpleDAO.sol#195)
Reentrancy in CimpleDAO.mint(address,uint256,uint256) (contracts/
   ↪ CimpleDAO.sol#110-114):
        External calls:
        - _mint(account,id,amount,0x000) (contracts/CimpleDAO.sol#111)
              - IERC1155Receiver(to).onERC1155Received(operator,from,id,
                 ↪ amount,data) (node_modules/@openzeppelin/contracts/
                 ↪ token/ERC1155/ERC1155.sol#476-484)
        State variables written after the call(s):
        - tokenSupply[id] = tokenSupply[id].add(amount) (contracts/
           ↪ CimpleDAO.sol#112)
        - _addOrUpdateUserInfo(account) (contracts/CimpleDAO.sol#113)
              - tempUser.cimpleValue = balanceOf(userAddress,Cimple) (
                 ↪ contracts/CimpleDAO.sol#182)
              - tempUser.stCimpleValue = balanceOf(userAddress,stCimple)
                 ↪  (contracts/CimpleDAO.sol#183)
```

```
            - tempUser.CMPGValue = balanceOf(userAddress,CMPG) (
               ↪ contracts/CimpleDAO.sol#184)
            - usersInfo.push(UserDetail(userAddress,0,0,0,,)) (
               ↪ contracts/CimpleDAO.sol#198)
      - _addOrUpdateUserInfo(account) (contracts/CimpleDAO.sol#113)
            - votablelist[userAddress] = true (contracts/CimpleDAO.sol
               ↪ #186)
            - votablelist[userAddress] = false (contracts/CimpleDAO.
               ↪ sol#189)
      - _addOrUpdateUserInfo(account) (contracts/CimpleDAO.sol#113)
            - votecreatablelist[userAddress] = true (contracts/
               ↪ CimpleDAO.sol#192)
            - votecreatablelist[userAddress] = false (contracts/
               ↪ CimpleDAO.sol#195)
Reentrancy in CimpleDAO.mintBatch(address,uint256[],uint256[],bytes) (
   ↪ contracts/CimpleDAO.sol#115-121):
      External calls:
      - _mintBatch(to,ids,amounts,data) (contracts/CimpleDAO.sol#116)
            - IERC1155Receiver(to).onERC1155BatchReceived(operator,
               ↪ from,ids,amounts,data) (node_modules/@openzeppelin/
               ↪ contracts/token/ERC1155/ERC1155.sol#497-507)
      State variables written after the call(s):
      - tokenSupply[ids[i]] = tokenSupply[ids[i]].add(amounts[i]) (
         ↪ contracts/CimpleDAO.sol#118)
      - _addOrUpdateUserInfo(to) (contracts/CimpleDAO.sol#120)
            - tempUser.cimpleValue = balanceOf(userAddress,Cimple) (
               ↪ contracts/CimpleDAO.sol#182)
            - tempUser.stCimpleValue = balanceOf(userAddress,stCimple)
               ↪  (contracts/CimpleDAO.sol#183)
            - tempUser.CMPGValue = balanceOf(userAddress,CMPG) (
               ↪ contracts/CimpleDAO.sol#184)
            - usersInfo.push(UserDetail(userAddress,0,0,0,,)) (
               ↪ contracts/CimpleDAO.sol#198)
      - _addOrUpdateUserInfo(to) (contracts/CimpleDAO.sol#120)
```

```
            - votablelist[userAddress] = true (contracts/CimpleDAO.sol
              ↪ #186)
            - votablelist[userAddress] = false (contracts/CimpleDAO.
              ↪ sol#189)
        - _addOrUpdateUserInfo(to) (contracts/CimpleDAO.sol#120)
            - votecreatablelist[userAddress] = true (contracts/
              ↪ CimpleDAO.sol#192)
            - votecreatablelist[userAddress] = false (contracts/
              ↪ CimpleDAO.sol#195)
Reentrancy in CimpleDAO.payFee() (contracts/CimpleDAO.sol#399-412):
    External calls:
    - _mint(msg.sender,Cimple,cimpleCountForValue,0x000) (contracts/
      ↪ CimpleDAO.sol#407)
        - IERC1155Receiver(to).onERC1155Received(operator,from,id,
          ↪ amount,data) (node_modules/@openzeppelin/contracts/
          ↪ token/ERC1155/ERC1155.sol#476-484)
    State variables written after the call(s):
    - tokenSupply[Cimple] = tokenSupply[Cimple].add(
      ↪ cimpleCountForValue) (contracts/CimpleDAO.sol#408)
    - _addOrUpdateUserInfo(msg.sender) (contracts/CimpleDAO.sol#409)
        - tempUser.cimpleValue = balanceOf(userAddress,Cimple) (
          ↪ contracts/CimpleDAO.sol#182)
        - tempUser.stCimpleValue = balanceOf(userAddress,stCimple)
          ↪  (contracts/CimpleDAO.sol#183)
        - tempUser.CMPGValue = balanceOf(userAddress,CMPG) (
          ↪ contracts/CimpleDAO.sol#184)
        - usersInfo.push(UserDetail(userAddress,0,0,0,,)) (
          ↪ contracts/CimpleDAO.sol#198)
    - _addOrUpdateUserInfo(msg.sender) (contracts/CimpleDAO.sol#409)
        - votablelist[userAddress] = true (contracts/CimpleDAO.sol
          ↪ #186)
        - votablelist[userAddress] = false (contracts/CimpleDAO.
          ↪ sol#189)
    - _addOrUpdateUserInfo(msg.sender) (contracts/CimpleDAO.sol#409)
```

```
            - votecreatablelist[userAddress] = true (contracts/
              ↪ CimpleDAO.sol#192)
            - votecreatablelist[userAddress] = false (contracts/
              ↪ CimpleDAO.sol#195)
Reentrancy in CimpleDAO.removeStake(address,uint256) (contracts/
   ↪ CimpleDAO.sol#288-314):
    External calls:
    - _mint(unstaker,Cimple,_stake,0x000) (contracts/CimpleDAO.sol
      ↪ #296)
        - IERC1155Receiver(to).onERC1155Received(operator,from,id,
          ↪ amount,data) (node_modules/@openzeppelin/contracts/
          ↪ token/ERC1155/ERC1155.sol#476-484)
    State variables written after the call(s):
    - tokenBurn[stCimple] = tokenBurn[stCimple].add(_stake) (
      ↪ contracts/CimpleDAO.sol#299)
    - tokenSupply[stCimple] = tokenSupply[stCimple].sub(_stake) (
      ↪ contracts/CimpleDAO.sol#297)
    - tokenSupply[Cimple] = tokenSupply[Cimple].add(_stake) (
      ↪ contracts/CimpleDAO.sol#298)
Reentrancy in CimpleDAO.removeStake(address,uint256) (contracts/
   ↪ CimpleDAO.sol#288-314):
    External calls:
    - _mint(unstaker,Cimple,_stake,0x000) (contracts/CimpleDAO.sol
      ↪ #296)
        - IERC1155Receiver(to).onERC1155Received(operator,from,id,
          ↪ amount,data) (node_modules/@openzeppelin/contracts/
          ↪ token/ERC1155/ERC1155.sol#476-484)
    - _mint(unstaker,CMPG,rewardOfCMPG,0x000) (contracts/CimpleDAO.
      ↪ sol#300)
        - IERC1155Receiver(to).onERC1155Received(operator,from,id,
          ↪ amount,data) (node_modules/@openzeppelin/contracts/
          ↪ token/ERC1155/ERC1155.sol#476-484)
    State variables written after the call(s):
    - _addOrUpdateUserInfo(unstaker) (contracts/CimpleDAO.sol#307)
```

```
            - tempUser.cimpleValue = balanceOf(userAddress,Cimple) (
                ↪ contracts/CimpleDAO.sol#182)
            - tempUser.stCimpleValue = balanceOf(userAddress,stCimple)
                ↪  (contracts/CimpleDAO.sol#183)
            - tempUser.CMPGValue = balanceOf(userAddress,CMPG) (
                ↪ contracts/CimpleDAO.sol#184)
            - usersInfo.push(UserDetail(userAddress,0,0,0,,)) (
                ↪ contracts/CimpleDAO.sol#198)
        - _addOrUpdateUserInfo(unstaker) (contracts/CimpleDAO.sol#307)
            - votablelist[userAddress] = true (contracts/CimpleDAO.sol
                ↪ #186)
            - votablelist[userAddress] = false (contracts/CimpleDAO.
                ↪ sol#189)
        - _addOrUpdateUserInfo(unstaker) (contracts/CimpleDAO.sol#307)
            - votecreatablelist[userAddress] = true (contracts/
                ↪ CimpleDAO.sol#192)
            - votecreatablelist[userAddress] = false (contracts/
                ↪ CimpleDAO.sol#195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-2

Reentrancy in CimpleDAO.claimFirstCimpleForNFT(address,uint256) (
    ↪ contracts/CimpleDAO.sol#422-444):
        External calls:
        - _mint(_userAddress,Cimple,_rewardCimpleAmount,0x000) (contracts
            ↪ /CimpleDAO.sol#436)
            - IERC1155Receiver(to).onERC1155Received(operator,from,id,
                ↪ amount,data) (node_modules/@openzeppelin/contracts/
                ↪ token/ERC1155/ERC1155.sol#476-484)
        - nftUtils.setNftAwardList(_tokenID,true) (contracts/CimpleDAO.
            ↪ sol#440)
        - nftUtils.increaseNftAwardListCount() (contracts/CimpleDAO.sol
            ↪ #441)
        Event emitted after the call(s):
```

```
        - ClaimRewardFirstCimpleForNFT(_userAddress,_tokenID,
            ↪ _rewardCimpleAmount) (contracts/CimpleDAO.sol#442)
Reentrancy in CimpleDAO.claimRewardStreamingForNFT(address,uint256) (
    ↪ contracts/CimpleDAO.sol#445-473):
        External calls:
        - nftUtils.addNFTUsersInfo(_address,_tokenID,block.timestamp) (
            ↪ contracts/CimpleDAO.sol#454)
        - nftUtils.setNFTUsersInfoByIndex(s,block.timestamp) (contracts/
            ↪ CimpleDAO.sol#460)
        - _mint(_address,Cimple,_amountOfReward,0x000) (contracts/
            ↪ CimpleDAO.sol#465)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,
                    ↪ amount,data) (node_modules/@openzeppelin/contracts/
                    ↪ token/ERC1155/ERC1155.sol#476-484)
        Event emitted after the call(s):
        - ClaimRewardStreamingForNFT(_address,_tokenID,_amountOfReward) (
            ↪ contracts/CimpleDAO.sol#469)
        - TransferSingle(operator,address(0),to,id,amount) (node_modules/
            ↪ @openzeppelin/contracts/token/ERC1155/ERC1155.sol#281)
                - _mint(_address,Cimple,_amountOfReward,0x000) (contracts/
                    ↪ CimpleDAO.sol#465)
Reentrancy in CimpleDAO.createStake(address,uint256) (contracts/
    ↪ CimpleDAO.sol#265-287):
        External calls:
        - _mint(staker,CMPG,rewardOfCMPG,0x000) (contracts/CimpleDAO.sol
            ↪ #274)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,
                    ↪ amount,data) (node_modules/@openzeppelin/contracts/
                    ↪ token/ERC1155/ERC1155.sol#476-484)
        Event emitted after the call(s):
        - TransferSingle(operator,from,address(0),id,amount) (
            ↪ node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155
            ↪ .sol#352)
```

```
            - _burn(staker,Cimple,_stake) (contracts/CimpleDAO.sol
              ↪ #279)
Reentrancy in CimpleDAO.createStake(address,uint256) (contracts/
   ↪ CimpleDAO.sol#265-287):
        External calls:
        - _mint(staker,CMPG,rewardOfCMPG,0x000) (contracts/CimpleDAO.sol
          ↪ #274)
            - IERC1155Receiver(to).onERC1155Received(operator,from,id,
              ↪ amount,data) (node_modules/@openzeppelin/contracts/
              ↪ token/ERC1155/ERC1155.sol#476-484)
        - _mint(staker,stCimple,_stake,0x000) (contracts/CimpleDAO.sol
          ↪ #280)
            - IERC1155Receiver(to).onERC1155Received(operator,from,id,
              ↪ amount,data) (node_modules/@openzeppelin/contracts/
              ↪ token/ERC1155/ERC1155.sol#476-484)
        Event emitted after the call(s):
        - StakingCimpleToken(staker,stCimple,_stake) (contracts/CimpleDAO
          ↪ .sol#285)
        - TransferSingle(operator,address(0),to,id,amount) (node_modules/
          ↪ @openzeppelin/contracts/token/ERC1155/ERC1155.sol#281)
            - _mint(staker,stCimple,_stake,0x000) (contracts/CimpleDAO
              ↪ .sol#280)
Reentrancy in CimpleDAO.payFee() (contracts/CimpleDAO.sol#399-412):
        External calls:
        - _mint(msg.sender,Cimple,cimpleCountForValue,0x000) (contracts/
          ↪ CimpleDAO.sol#407)
            - IERC1155Receiver(to).onERC1155Received(operator,from,id,
              ↪ amount,data) (node_modules/@openzeppelin/contracts/
              ↪ token/ERC1155/ERC1155.sol#476-484)
        Event emitted after the call(s):
        - PayFee(msg.sender,Cimple,cimpleCountForValue) (contracts/
          ↪ CimpleDAO.sol#410)
Reentrancy in CimpleDAO.removeStake(address,uint256) (contracts/
   ↪ CimpleDAO.sol#288-314):
```

```
        External calls:
        - _mint(unstaker,Cimple,_stake,0x000) (contracts/CimpleDAO.sol
            ↪ #296)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,
                    ↪ amount,data) (node_modules/@openzeppelin/contracts/
                    ↪ token/ERC1155/ERC1155.sol#476-484)
        - _mint(unstaker,CMPG,rewardOfCMPG,0x000) (contracts/CimpleDAO.
            ↪ sol#300)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,
                    ↪ amount,data) (node_modules/@openzeppelin/contracts/
                    ↪ token/ERC1155/ERC1155.sol#476-484)
        Event emitted after the call(s):
        - TransferSingle(operator,address(0),to,id,amount) (node_modules/
            ↪ @openzeppelin/contracts/token/ERC1155/ERC1155.sol#281)
                - _mint(unstaker,CMPG,rewardOfCMPG,0x000) (contracts/
                    ↪ CimpleDAO.sol#300)
        - UnstakingCimpleToken(unstaker,Cimple,_stake) (contracts/
            ↪ CimpleDAO.sol#308)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

CimpleDAO.calculateCimpleIR(uint256) (contracts/CimpleDAO.sol#136-171)
    ↪ uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deployedStartTimeStamp < _currentTimeStamp
            ↪ ,Error, selected date is lower than token publish date) (
            ↪ contracts/CimpleDAO.sol#137)
        - usedYearCount > 30 (contracts/CimpleDAO.sol#148)
        - usedYearCount < uint256(1) (contracts/CimpleDAO.sol#151)
        - i <= usedYearCount (contracts/CimpleDAO.sol#154)
        - cimpleIR >= 1e18 (contracts/CimpleDAO.sol#167)
CimpleDAO.isStakeholder(address) (contracts/CimpleDAO.sol#241-246) uses
    ↪ timestamp for comparisons
        Dangerous comparisons:
```

```
        - s < stakeholders.length (contracts/CimpleDAO.sol#242)
        - _address == stakeholders[s].holderAddress (contracts/CimpleDAO.
            ↪ sol#243)
CimpleDAO.totalStakes() (contracts/CimpleDAO.sol#258-264) uses timestamp
    ↪  for comparisons
        Dangerous comparisons:
        - s < stakeholders.length (contracts/CimpleDAO.sol#260)
CimpleDAO.totalRewards() (contracts/CimpleDAO.sol#315-321) uses
    ↪ timestamp for comparisons
        Dangerous comparisons:
        - s < stakeholders.length (contracts/CimpleDAO.sol#317)
CimpleDAO.calculateReward(address,uint256,uint256) (contracts/CimpleDAO.
    ↪ sol#322-351) uses timestamp for comparisons
        Dangerous comparisons:
        - _nowTimeStamp > _holdTimeStamp && _holdTimeStamp >=
            ↪ deployedStartTimeStamp (contracts/CimpleDAO.sol#323)
        - _holdPeriodDayCount > 0 (contracts/CimpleDAO.sol#335)
        - index < _holdPeriodDayCount (contracts/CimpleDAO.sol#336)
CimpleDAO._calculateDailySupplyOfCMPG(uint256) (contracts/CimpleDAO.sol
    ↪ #365-377) uses timestamp for comparisons
        Dangerous comparisons:
        - _days >= 1 (contracts/CimpleDAO.sol#370)
        - index < _days (contracts/CimpleDAO.sol#371)
CimpleDAO._calculateDistributionPercentOfCMPG(address) (contracts/
    ↪ CimpleDAO.sol#378-387) uses timestamp for comparisons
        Dangerous comparisons:
        - stakeholders.length > 0 && _isStakeHolder (contracts/CimpleDAO.
            ↪ sol#380)
CimpleDAO.getStakeHolders() (contracts/CimpleDAO.sol#388-398) uses
    ↪ timestamp for comparisons
        Dangerous comparisons:
        - s < stakeholders.length (contracts/CimpleDAO.sol#392)
CimpleDAO.claimRewardStreamingForNFT(address,uint256) (contracts/
    ↪ CimpleDAO.sol#445-473) uses timestamp for comparisons
```

```
        Dangerous comparisons:
        - _rate > 0 (contracts/CimpleDAO.sol#448)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #block-timestamp

Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#201-221) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.
            ↪ sol#213-216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

CimpleDAO.claimFirstCimpleForNFT(address,uint256) (contracts/CimpleDAO.
    ↪ sol#422-444) compares to a boolean constant:
        -require(bool,string)(nftUtils.getNftAwardList(_tokenID) == false
            ↪  && _flag,This user is not available for this rewards.) (
            ↪ contracts/CimpleDAO.sol#432)
CimpleDAO.claimRewardStreamingForNFT(address,uint256) (contracts/
    ↪ CimpleDAO.sol#445-473) compares to a boolean constant:
        -_isNftUser == false (contracts/CimpleDAO.sol#453)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #boolean-equality

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1']
        - ^0.8.0 (contracts/CimpleDAO.sol#2)
        - ABIEncoderV2 (contracts/CimpleDAO.sol#3)
        - ^0.8.0 (contracts/NFTUtils.sol#2)
        - ^0.8.0 (contracts/VoteUtils.sol#2)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol
            ↪ #4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/
            ↪ ERC1155.sol#4)
```

```
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/
          ↪ IERC1155.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/
          ↪ IERC1155Receiver.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/
          ↪ extensions/IERC1155MetadataURI.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol
          ↪ #4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol
          ↪ #4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/
          ↪ introspection/ERC165.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/
          ↪ introspection/IERC165.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/
          ↪ SafeMath.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #different-pragma-directives-are-used

CimpleDAO.singleAddressToMintableRoleList(address) (contracts/CimpleDAO.
    ↪ sol#87-94) has costly operations inside a loop:
        - totalMintRoleList ++ (contracts/CimpleDAO.sol#91)
CimpleDAO.removeAddressFromMintableRoleList(address) (contracts/
    ↪ CimpleDAO.sol#100-106) has costly operations inside a loop:
        - totalMintRoleList -- (contracts/CimpleDAO.sol#104)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #costly-operations-inside-a-loop

Address.functionCall(address,bytes) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#85-87) is never used and should be
    ↪ removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#95-101) is never used and should be
    ↪ removed
```

```
Address.functionCallWithValue(address,bytes,uint256) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#114-120) is never used
    ↪ and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (
    ↪ node_modules/@openzeppelin/contracts/utils/Address.sol#128-139)
    ↪ is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#174-176) is never used and should be
    ↪ removed
Address.functionDelegateCall(address,bytes,string) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#184-193) is never used
    ↪ and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#147-149) is never used and should be
    ↪ removed
Address.functionStaticCall(address,bytes,string) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#157-166) is never used
    ↪ and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts
    ↪ /utils/Address.sol#60-65) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#201-221) is never used and should be
    ↪ removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.
    ↪ sol#21-23) is never used and should be removed
NFTUtils.removeNFTUsersInfo(address,uint256) (contracts/NFTUtils.sol
    ↪ #87-94) is never used and should be removed
SafeMath.div(uint256,uint256,string) (node_modules/@openzeppelin/
    ↪ contracts/utils/math/SafeMath.sol#191-200) is never used and
    ↪ should be removed
SafeMath.mod(uint256,uint256,string) (node_modules/@openzeppelin/
    ↪ contracts/utils/math/SafeMath.sol#217-226) is never used and
    ↪ should be removed
```

```
SafeMath.sub(uint256,uint256,string) (node_modules/@openzeppelin/
    ↪ contracts/utils/math/SafeMath.sol#168-177) is never used and
    ↪ should be removed
SafeMath.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts/
    ↪ utils/math/SafeMath.sol#22-28) is never used and should be
    ↪ removed
SafeMath.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/
    ↪ utils/math/SafeMath.sol#64-69) is never used and should be
    ↪ removed
SafeMath.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts/
    ↪ utils/math/SafeMath.sol#76-81) is never used and should be
    ↪ removed
SafeMath.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts/
    ↪ utils/math/SafeMath.sol#47-57) is never used and should be
    ↪ removed
SafeMath.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts/
    ↪ utils/math/SafeMath.sol#35-40) is never used and should be
    ↪ removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version^0.8.0 (contracts/CimpleDAO.sol#2) allows old versions
Pragma version^0.8.0 (contracts/NFTUtils.sol#2) allows old versions
Pragma version^0.8.0 (contracts/VoteUtils.sol#2) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/
    ↪ Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155
    ↪ /ERC1155.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155
    ↪ /IERC1155.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155
    ↪ /IERC1155Receiver.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155
    ↪ /extensions/IERC1155MetadataURI.sol#4) allows old versions
```

```
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address
    ↪ .sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context
    ↪ .sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/
    ↪ introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/
    ↪ introspection/IERC165.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/
    ↪ SafeMath.sol#4) allows old versions
solc-0.8.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Variable CimpleDAO.calculateReward(address,uint256,uint256)._stakeholder
    ↪  (contracts/CimpleDAO.sol#322) is too similar to CimpleDAO.
    ↪ stakeholders (contracts/CimpleDAO.sol#41)
Variable CimpleDAO.removeStakeholder(address)._stakeholder (contracts/
    ↪ CimpleDAO.sol#251) is too similar to CimpleDAO.stakeholders (
    ↪ contracts/CimpleDAO.sol#41)
Variable CimpleDAO.addStakeholder(address,uint256)._stakeholder (
    ↪ contracts/CimpleDAO.sol#247) is too similar to CimpleDAO.
    ↪ stakeholders (contracts/CimpleDAO.sol#41)
Variable CimpleDAO.testCalculateReward(address,uint256)._stakeholder (
    ↪ contracts/CimpleDAO.sol#352) is too similar to CimpleDAO.
    ↪ stakeholders (contracts/CimpleDAO.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #variable-names-are-too-similar

CimpleDAO.oneSecondTimeStamp (contracts/CimpleDAO.sol#23) is never used
    ↪ in CimpleDAO (contracts/CimpleDAO.sol#9-475)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-state-variable
```

```
CimpleDAO.votablelistcounter (contracts/CimpleDAO.sol#43) should be
    ↪ constant
CimpleDAO.votecreatablelistcounter (contracts/CimpleDAO.sol#45) should
    ↪ be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #state-variables-that-could-be-declared-constant
contracts/CimpleDAO.sol analyzed (14 contracts with 75 detectors), 96
    ↪ result(s) found

ERC721._checkOnERC721Received(address,address,uint256,bytes) (
    ↪ node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol
    ↪ #394-416) ignores return value by IERC721Receiver(to).
    ↪ onERC721Received(_msgSender(),from,tokenId,data) (node_modules/
    ↪ @openzeppelin/contracts/token/ERC721/ERC721.sol#401-412)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).
    ↪ retval (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.
    ↪ sol#401)' in ERC721._checkOnERC721Received(address,address,
    ↪ uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721
    ↪ /ERC721.sol#394-416) potentially used before declaration: retval
    ↪ == IERC721Receiver.onERC721Received.selector (node_modules/
    ↪ @openzeppelin/contracts/token/ERC721/ERC721.sol#402)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).
    ↪ reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.
    ↪ sol#403)' in ERC721._checkOnERC721Received(address,address,
    ↪ uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721
    ↪ /ERC721.sol#394-416) potentially used before declaration: reason.
    ↪ length == 0 (node_modules/@openzeppelin/contracts/token/ERC721/
    ↪ ERC721.sol#404)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).
    ↪ reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.
    ↪ sol#403)' in ERC721._checkOnERC721Received(address,address,
```

```
    ↪ uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721
    ↪ /ERC721.sol#394-416) potentially used before declaration: revert(
    ↪ uint256,uint256)(32 + reason,mload(uint256)(reason)) (
    ↪ node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#409)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #pre-declaration-usage-of-local-variables

Reentrancy in Escrow.withdraw(address) (node_modules/@openzeppelin/
    ↪ contracts/utils/escrow/Escrow.sol#58-66):
        External calls:
        - payee.sendValue(payment) (node_modules/@openzeppelin/contracts/
            ↪ utils/escrow/Escrow.sol#63)
        Event emitted after the call(s):
        - Withdrawn(payee,payment) (node_modules/@openzeppelin/contracts/
            ↪ utils/escrow/Escrow.sol#65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

ERC721._checkOnERC721Received(address,address,uint256,bytes) (
    ↪ node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol
    ↪ #394-416) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/token/ERC721/
            ↪ ERC721.sol#408-410)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#201-221) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.
            ↪ sol#213-216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1']
        - ^0.8.0 (contracts/NFT.sol#2)
```

```
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol
          ↪ #4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/
          ↪ PullPayment.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/
          ↪ ERC721.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/
          ↪ IERC721.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/
          ↪ IERC721Receiver.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/
          ↪ extensions/ERC721Enumerable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/
          ↪ extensions/IERC721Enumerable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/
          ↪ extensions/IERC721Metadata.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol
          ↪ #4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol
          ↪ #4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol
          ↪ #4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol
          ↪ #4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/escrow/
          ↪ Escrow.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/
          ↪ introspection/ERC165.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/
          ↪ introspection/IERC165.sol#4)
    Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
       ↪ #different-pragma-directives-are-used
```

```
Address.functionCall(address,bytes) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#85-87) is never used and should be
    ↪ removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#95-101) is never used and should be
    ↪ removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#114-120) is never used
    ↪ and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (
    ↪ node_modules/@openzeppelin/contracts/utils/Address.sol#128-139)
    ↪ is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#174-176) is never used and should be
    ↪ removed
Address.functionDelegateCall(address,bytes,string) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#184-193) is never used
    ↪ and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#147-149) is never used and should be
    ↪ removed
Address.functionStaticCall(address,bytes,string) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#157-166) is never used
    ↪ and should be removed
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#201-221) is never used and should be
    ↪ removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.
    ↪ sol#21-23) is never used and should be removed
Counters.decrement(Counters.Counter) (node_modules/@openzeppelin/
    ↪ contracts/utils/Counters.sol#32-38) is never used and should be
    ↪ removed
Counters.reset(Counters.Counter) (node_modules/@openzeppelin/contracts/
    ↪ utils/Counters.sol#40-42) is never used and should be removed
```

```
ERC721._baseURI() (node_modules/@openzeppelin/contracts/token/ERC721/
    ↪ ERC721.sol#105-107) is never used and should be removed
ERC721._burn(uint256) (node_modules/@openzeppelin/contracts/token/ERC721
    ↪ /ERC721.sol#303-317) is never used and should be removed
PullPayment._asyncTransfer(address,uint256) (node_modules/@openzeppelin/
    ↪ contracts/security/PullPayment.sol#71-73) is never used and
    ↪ should be removed
Strings.toHexString(address) (node_modules/@openzeppelin/contracts/utils
    ↪ /Strings.sol#72-74) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils
    ↪ /Strings.sol#41-52) is never used and should be removed
Strings.toHexString(uint256,uint256) (node_modules/@openzeppelin/
    ↪ contracts/utils/Strings.sol#57-67) is never used and should be
    ↪ removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version^0.8.0 (contracts/NFT.sol#2) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/
    ↪ Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/
    ↪ PullPayment.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/
    ↪ ERC721.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/
    ↪ IERC721.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/
    ↪ IERC721Receiver.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/
    ↪ extensions/ERC721Enumerable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/
    ↪ extensions/IERC721Enumerable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/
    ↪ extensions/IERC721Metadata.sol#4) allows old versions
```

```
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address
    ↪ .sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context
    ↪ .sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/
    ↪ Counters.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings
    ↪ .sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/escrow/
    ↪ Escrow.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/
    ↪ introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/
    ↪ introspection/IERC165.sol#4) allows old versions
solc-0.8.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity
contracts/NFT.sol analyzed (16 contracts with 75 detectors), 43 result(s
    ↪ ) found


Contract locking ether found:
        Contract NFTUtils (contracts/NFTUtils.sol#7-178) has payable
            ↪ functions:
         - NFTUtils.setNftAwardList(uint256,bool) (contracts/NFTUtils.sol
            ↪ #45-47)
         - NFTUtils.increaseNftAwardListCount() (contracts/NFTUtils.sol
            ↪ #50-52)
         - NFTUtils.addNFTUsersInfo(address,uint256,uint256) (contracts/
            ↪ NFTUtils.sol#80-85)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #contracts-that-lock-ether
```

```
NFTUtils.changeOwner(address) (contracts/NFTUtils.sol#29-32) should emit
    ↪  an event for:
        - owner = _newOwner (contracts/NFTUtils.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-events-access-control

NFTUtils.constructor(address).nftaddress (contracts/NFTUtils.sol#24)
    ↪ lacks a zero-check on :
                - CiMPLENFTaddress = nftaddress (contracts/NFTUtils.sol
                    ↪ #25)
NFTUtils.changeOwner(address)._newOwner (contracts/NFTUtils.sol#29)
    ↪ lacks a zero-check on :
                - owner = _newOwner (contracts/NFTUtils.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

NFTUtils.removeNFTUsersInfo(address,uint256) (contracts/NFTUtils.sol
    ↪ #87-94) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version^0.8.0 (contracts/NFTUtils.sol#2) allows old versions
solc-0.8.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity
contracts/NFTUtils.sol analyzed (2 contracts with 75 detectors), 7
    ↪ result(s) found


Contract locking ether found:
        Contract VoteUtils (contracts/VoteUtils.sol#4-97) has payable
            ↪ functions:
         - VoteUtils.makeproposal(address,string,uint256,uint256) (
             ↪ contracts/VoteUtils.sol#33-37)
```

```
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #contracts-that-lock-ether

VoteUtils.changeOwner(address) (contracts/VoteUtils.sol#29-32) should
    ↪ emit an event for:
        - owner = _newOwner (contracts/VoteUtils.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-events-access-control

VoteUtils.changeOwner(address)._newOwner (contracts/VoteUtils.sol#29)
    ↪ lacks a zero-check on :
                - owner = _newOwner (contracts/VoteUtils.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

Pragma version^0.8.0 (contracts/VoteUtils.sol#2) allows old versions
solc-0.8.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity
contracts/VoteUtils.sol analyzed (1 contracts with 75 detectors), 5
    ↪ result(s) found
```

Conclusion:

Most of the vulnerabilities found by the analysis have already been addressed by the smart contract code review.

6 Conclusion

In this audit, we examined the design and implementation of CimpleDAO contract and discovered several issues of varying severity. CimpleDAO team has acknowledged all the issues raised in the initial report. Shellboxes' auditors advised CimpleDAO Team to maintain a high level of vigilance and to keep those findings in mind in order to avoid any future complications.

For a Contract Audit, contact us at contact@shellboxes.com