DeHacker

Code Security Assessment

JusticeDAO

Oct 8th, 2023

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.
Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	JusticeDAO
Platform	BSC
Website	https://www.justicedao.top/index_en.html
Type	Defi
Language	Solidity

Vulnerability Summary

Vulnerability Level	Total	Mitigated	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	1	0	0	1	0	0
Medium	5	0	0	5	0	0
Minor	1	0	0	1	0	0
Informational	0	0	0	0	0	0
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
EFD	donate.sol	F0E4C2F76C58916EC258F246851BEA09 1D14D4247A2FC3E18694461B1816E13B
PDO	pledge.sol	C288ED0913B1BD8F403C7FE18019A974 79A62F40B770757C07FC9F5E8DF5D285
DEJ	XOT.sol	IEE1B9E8FE3AE81FDEF261395F8B1F64A6 A291D4F1342D0618F95ED5714214D05

Findings

ID	Category	Severity	Status
GLOBAL-01	Volatile Code	Major	Acknowledged
DEJ-01	Unchecked Transfer	Medium	Acknowledged
DEJ-02	Divide Before Multiply	Minor	Acknowledged
PDO-01	Unchecked Transfer	Medium	Acknowledged
PDO-02	Reentrancy	Medium	Acknowledged
EFD-01	Unchecked Transfer	Medium	Acknowledged
EFD-02	Reentrancy Risk	Medium	Acknowledged

MAJOR

Global-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization Related Risks	Major	donate.sol pledge.sol XOT.sol	Acknowledged

Description

The owner of the ERC20PledgeToken contract has the following permissions:

- function setTokenAddress()
- function setUniswapV2Pair()
- function withDrawalToken()
- function setReceiver()

The owner of the ERC20PayToken contract has the following permissions:

- function setTokenAddress()
- function setTokenRatio()
- function setMintSwitch()
- function setRedeemSwitch()
- function setWithDrawalSwitch()
- function withDrawalToken()
- function setReceiver()

The owner of the XOT contract has the following permissions:

- function excludeFromFees()
- function excludeMultipleAccountsFromFees()
- function setSwapTokensAtAmount()
- function setSwapAndLiquifyEnabled()
- function openSwapStart()
- function addOtherPair()
- function swapAndLiquifyKOL2()
- function withDrawalToken()

The owner will have the ability to influence the operation results of the protocol.

Recommendation

This finding describes the level of decentralization of the project, and it is recommended to strengthen security and improve the degree of decentralization from the following aspects:

- It is recommended that privileged addresses use multi-signature wallet addresses.
- For modification operations that affect protocol operation stability and key business parameters, it is recommended to implement time locks.

Medium

DEJ-01 | Unchecked Transfer

Category	Severity	Location	Status
Unchecked Transfer	Medium	XOT.sol : Line 1090,1097,1190	Acknowledged

Description

Line 1090

```
function swapAndLiquify() private {
    uint256 allTokenAmount = super.balanceOf(address(this));
    swapTokensForUsdt(allTokenAmount);
    uint256 allAmount = USDT.balanceOf(address(this));
    USDT.transfer(_fundAddress, allAmount);
}
```

Line 1097

```
function swapAndLiquifyKOL() private {
    uint256 allTokenAmount = super.balanceOf(_KOLAddress);
    super._transfer(_KOLAddress,address(this), allTokenAmount);
    swapTokensForUsdt(allTokenAmount);
    uint256 allAmount = USDT.balanceOf(address(this));
    USDT.transfer(_KOLAddress, allAmount);
}
```

Line 1109

```
function withDrawalToken(address token, address _address, uint amount) external onlyOwner returns(bool){
    IERC20(token).transfer(_address, amount);
    return true;
}
```

Recommendation

Use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

Medium

PDO-01 | Unchecked Transfer

Category	Severity	Location	Status
Unchecked Transfer	Medium	pledge.sol: 55,87	Acknowledged

Description

Line 55

```
function pledge(uint _type, uint tokenAmount, uint usdtAmount, uint cycle) external returns(bool){
    require(0 < tokenAmount, 'Amount: must be > 0');

    if (cycle != 7 && cycle != 30 && cycle != 90 && cycle != 180 && cycle != 360) {
        require(true,'cycle error');
    }
    uint time = block.timestamp;
    // uint endtime = time + cycle * 24 * 60 * 60;
    uint endtime = time + cycle * 60;

    address sender = _msgSender();

    uint _amount;
    if (_type == 0) {
        IERC20(token).transferFrom(sender, receiver, tokenAmount);
        _amount = tokenAmount;
    }else {
        IERC20(_baseToken).transferFrom(sender, receiver, usdtAmount);
        IERC20(token).transferFrom(sender, receiver, tokenAmount);
        uint oldLpAmount = IERC20(uniswapV2Pair).balanceOf(address(this));
        addLiquidity(usdtAmount, tokenAmount);
        uint newLpAmount = IERC20(uniswapV2Pair).balanceOf(address(this));

        _amount = newLpAmount.sub(oldLpAmount);
    }
    pledgeData[sender].push(PledgeData(_type, _amount, cycle, 0, endtime, time));

    emit Pledge(_type, sender, _amount, cycle, 0);

    return true;
}
```

Line 87

```
function endPledge(uint key) external returns(bool){
    address sender = _msgSender();
    uint time = block.timestamp;
    require(pledgeData[sender][key].status == 0, 'Can not be released repeatedly');
    require(pledgeData[sender][key].endtime < time, 'The pledge time is not over');

    uint _type = pledgeData[sender][key]._type;
    uint _amount = pledgeData[sender][key].amount;
    if (_type == 0) {
        IERC20(token).transfer(sender, _amount);
    }else {
        IERC20(uniswapV2Pair).transfer(sender, _amount);
    }
    pledgeData[sender][key].status = 1;
    uint cycle = pledgeData[sender][key].cycle;

    emit Pledge(_type, sender, _amount, cycle, 1);

    return true;
}
```

The return value of an external transfer/transferFrom call is not checked.

Recommendation

Use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

Medium

PDO-02 | Reentrancy Risk

Category	Severity	Location	Status
Reentrancy Risk	Medium	pledge.sol: 87	Acknowledged

Description

Line 87

```
function endPledge(uint key) external returns(bool){
    address sender = _msgSender();
    uint time = block.timestamp;
    require(pledgeData[sender][key].status == 0, 'Can not be released repeatedly');
    require(pledgeData[sender][key].endtime < time, 'The pledge time is not over');

    uint _type = pledgeData[sender][key]._type;
    uint _amount = pledgeData[sender][key].amount;
    if (_type == 0) {
        IERC20(token).transfer(sender, _amount);
    }else {
        IERC20(uniswapV2Pair).transfer(sender, _amount);
    }
    pledgeData[sender][key].status = 1;
    uint cycle = pledgeData[sender][key].cycle;

    emit Pledge(_type, sender, _amount, cycle, 1);

    return true;
}
```

The `status` variable is updated after an external call, which is a potential risk of reentrancy.

Recommendation

Using ReentrancyGuard or the checks-effects-interactions pattern.

Medium

EFD-01 | Unchecked Transfer

Category	Severity	Location	Status
Unchecked Transfer	Medium	donate.sol: 88	Acknowledged

Description

Line 88

```
function redeem(uint _type, uint amount) public returns (bool) {
    if (_type == 0) {
        //redeem
        require(redeemSwitch, "Not open yet");
        require(amount >= tokenRatio * 10 ** 17, "redemption must be greater than or equal to 1,000,000");

        IERC20(tokenAddress).transferFrom(msg.sender, destroyAddress, amount);
        uint eth = amount.div(tokenRatio);
        payable(msg.sender).transfer(eth);

        mintLog[msg.sender].push(MintLog(0, eth, block.timestamp));
    } else {
        //withDrawalToken
        require(withDrawalSwitch, "Not open yet");

        require(account[msg.sender] >= amount, "Insufficient address balance");
        uint balance = IERC20(tokenAddress).balanceOf(address(this));
        require(balance >= amount, "Insufficient contract balance");

        IERC20(tokenAddress).transfer(msg.sender, amount);

        account[msg.sender] = account[msg.sender].sub(amount);
        mintLog[msg.sender].push(MintLog(1, amount, block.timestamp));
    }


    emit Redeem(_type, msg.sender, amount);
    return true;
}
```

The return value of an external transfer/transferFrom call is not checked.

Recommendation

Use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

Medium

EFD-02 | Reentrancy Risk

Category	Severity	Location	Status
Reentrancy Risk	Medium	donate.sol: 88	Acknowledged

Description

Line 88

```
function redeem(uint _type, uint amount) public returns (bool) {
    if (_type == 0) {
        //redeem
        require(redeemSwitch, "Not open yet");
        require(amount >= tokenRatio * 10 ** 17, "redemption must be greater than or equal to 1,000,000");

        IERC20(tokenAddress).transferFrom(msg.sender, destroyAddress, amount);
        uint eth = amount.div(tokenRatio);
        payable(msg.sender).transfer(eth);

        mintLog[msg.sender].push(MintLog(0, eth, block.timestamp));
    } else {
        //withDrawalToken
        require(withDrawalSwitch, "Not open yet");

        require(account[msg.sender] >= amount, "Insufficient address balance");
        uint balance = IERC20(tokenAddress).balanceOf(address(this));
        require(balance >= amount, "Insufficient contract balance");

        IERC20(tokenAddress).transfer(msg.sender, amount);

        account[msg.sender] = account[msg.sender].sub(amount);
        mintLog[msg.sender].push(MintLog(1, amount, block.timestamp));
    }


    emit Redeem(_type, msg.sender, amount);
    return true;
}
```

The `account[msg.sender]` variable is updated after an external call, which is a potential risk of reentrancy.

Recommendation

Using ReentrancyGuard or the checks-effects-interactions pattern.

Minor

DEJ-02 | Divide Before Multiply

Category	Severity	Location	Status
Divide Before Multiply	Minor	XOT.sol: 967,1045,1049	Acknowledged

Description

Line 967

```
uint256 probablyLpAmount = pairTotalAmount.mul(usdtAmount).div(pairUSDTAmount).div(1000).mul(1020);
```

Line 1045

```
amount = amount.div(100).mul(99);
```

Line 1049

```
amount = amount.div(100).mul(98);
```

This may result in a loss of accuracy.

Recommendation

Consider ordering multiplication before division. Or ensure that the lost accuracy has no impact on the business.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style

Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

- Ethereum
- Cosmos
- Eos
- Substrate

TECH STACK

- Python
- Solidity
- Rust
- c++

CONTACTS

- https://dehacker.io
- https://twitter.com/dehackerio
- https://github.com/dehacker/audits_public
- https://t.me/dehackerio
- https://blog.dehacker.io/

DeHacker

oct 2023