THE ARCADIA GROUP

NXD Protocol Audit
Security Audit Report

PREPARED FOR:

NXD Protocol

ARCADIA CONTACT INFO

Email: audits@arcadiamgroup.com

Telegram: https://t.me/thearcadiagroup

Revision history

Date	Reason	Commit
03/01/2024	Initial Audit Scope	#6269a9abfc7b741a3292e4a964557d5732d38b79
4/23/2024	Review of New Modifications	#ed5f20bb65c04ac7429e409120ebed74de100a76

Table of Contents

Executive Summary

1. Introduction and Audit Scope

NXD Protocol engaged Arcadia to perform a security audit of their core protocol smart contracts, our review of their codebase occurred in the repo **DXNhyperstructure/NXD-Protocol** on the commit hash #**6269a9abfc7b741a3292e4a964557d5732d38b79** and a subsequent re-review #**ed5f20bb65c04ac7429e409120ebed74de100a76**

a. Review Team

Van Cam Pham - Lead Security Engineer

Tony Dong - Security Engineer and Secondary Review

b. Project Background

The **NXD Protocol**, featuring the NXD cryptocurrency, is designed as a hyper-deflationary asset that serves as a derivative of the DXN Protocol's ROI. It incorporates an automated vault system that enhances its daily earnings through various strategies aimed at continuously burning NXD tokens. This process is intended to decrement the token's supply while augmenting its baseline price. Additionally, the protocol leverages its operating profits by repurchasing and staking DXN tokens. This action not only amplifies the compounding of DXN but also boosts the Ethereum (ETH) rewards necessary to support the protocol's deflationary mechanics, thereby ensuring a sustainable model for its economic ecosystem.

c. Coverage

For this audit, we performed research, test coverage, investigation, and review of NXD Protocol's main contracts as well as some interlinking DBXEN Contracts dependencies followed by issue reporting, along with mitigation and remediation instructions as outlined in this report. The following code repositories, files, and/or libraries are considered in scope for the review.

File
src/dbxen/interfaces/IBurnableToken.sol
src/dbxen/interfaces/IBurnRedeemable.sol
src/dbxen/interfaces/IDBXenViews.sol
src/dbxen/interfaces/IRankedMintingToken.sol
src/dbxen/interfaces/IStakingToken.sol
src/dbxen/DBXen.sol
src/dbxen/DBXenERC20.sol
src/dbxen/DBXenViews.sol
src/dbxen/MathX.sol
src/dbxen/XENCrypto.sol
src/interfaces/IDBXen.sol
src/interfaces/INXDProtocol.sol
src/interfaces/INXDStakingVault.sol
src/interfaces/IUniswapV2Pair.sol
src/interfaces/IUniswapV2Router01.sol
src/interfaces/IUniswapV2Router02.sol
src/interfaces/IUniswapV3SwapCallback.sol
src/interfaces/IV2Oracle.sol
src/interfaces/IV3Oracle.sol
src/libraries/OracleLibrary.sol
src/NXDERC20.sol
src/NXDProtocol.sol
src/NXDStakingVault.sol
src/TaxRecipient.sol
src/V2Oracle.sol

src/V3Oracle.sol
src/Vesting.sol

2. Audit Summary

a. Audit Methodology

Arcadia completed this security review using various methods, primarily consisting of dynamic and static analysis. This process included a line-by-line analysis of the in-scope contracts, optimization analysis, analysis of key functionalities and limiters, and reference against intended functionality.

The followings are the steps we have performed while auditing the smart contracts:

- Investigating the project and its technical architecture overview through its documentation
- Understanding the overview of the smart contracts, the functions of the contracts, the inheritance, and how the contracts interface with each other thanks to the graph created by Solidity Visual Developer
- Manual smart contract audit:
  - Review the code to find any issue that could be exploited by known attacks listed by Consensys
  - Identifying which existing projects the smart contracts are built upon and what are the known vulnerabilities and remediations to the existing projects
  - Line-by-line manual review of the code to find any algorithmic and arithmetic related vulnerabilities compared to what should be done based on the project’s documentation
  - Find any potential code that could be refactored to save gas
  - Run through the unit-tests and test-coverage if exists
- Static Analysis:

  - Scanning for vulnerabilities in the smart contracts using Static Code Analysis Software
  - Making a static analysis of the smart contracts using Slither
- Additional review: a follow-up review is done when the smart contracts have any new update. The follow-up is done by reviewing all changes compared to the audited commit revision and its impact to the existing source code and found issues.

b. Summary

There were **15** issues found, **2** of which were deemed to be 'critical', and **1** was rated as 'high'.

Severity Rating	Number of Original Occurrences	Number of Remaining Occurrences
CRITICAL	2	0
HIGH	1	0
MEDIUM	4	0
LOW	6	0
INFORMATIONAL	2	0
GAS	0	0

Findings in Manual Audit

1. Use immutable for contract fields initialized by constructor

Issue ID

NXDP-1

Status

Resolved

#632ce7ad9a0976fe6df2512ae67142823af6970c

Risk Level

Severity: Low

Code Segment

```
constructor(
       uint256 _initialSupply,
       address _initialSupplyTo,
       IERC20 _dxn,
       address _governance,
       address _vesting,
       address _devFeeTo
   ) {
       _name = block.chainid == 1? "NXD Token":"";
       _symbol = block.chainid == 1? "NXD":"";

       protocol = msg.sender;
       devFeeTo = _devFeeTo;

       _mint(_initialSupplyTo, _initialSupply);

       dxn = _dxn;

       maxSupply = _initialSupply + MAX_REWARDS_SUPPLY + MAX_DEV_ALLOC;

       taxRecipient = new TaxRecipient(msg.sender);

       governance = _governance;

       _updateTaxWhitelist(address(this), true, false);
       _updateTaxWhitelist(address(taxRecipient), true, false);
       _updateTaxWhitelist(protocol, true, true);
```

```
        _updateTaxWhitelist(_vesting, true, true);
    }
```

Description

State variables that are only initialized once in the contract constructor should be defined with the *immutable* keyword for saving storage gas costs.

Code location

```
src/NXDERC20.sol
src/NXDProtocol.sol
src/NXDStakingVault.sol
```

Reference

https://docs.soliditylang.org/en/v0.6.7/contracts.html?highlight=immutable#constant-and-immutable-state-variables

Recommendation

Define state variables initialized once in the contract constructor with the immutable keyword.

2. Cannot mint max supply of NXD

Issue ID

NXDP-2

Status

Resolved

#e6dd9a2ce85cf118493f80777a716da042c5df41

Risk Level

Severity: Critical

Code Segment

```
function mint(address account, uint256 amount) external {
       if (msg.sender != protocol) {
           revert Unauthorized();
       }
       if (totalSupply() + amount > maxSupply - MAX_DEV_ALLOC) {
           revert MaxSupply();
       }
       _mint(account, amount);
   }
```

Description

The function is expected to be able to mint until the NXD supply reaches its *maxSupply*. However, the current reverts any minting that can make the new supply greater than *maxSupply - MAX_DEV_ALLOC*. In other words, the function makes the token contract has a max supply of *maxSupply - MAX_DEV_ALLOC*

Code location

```
src/NXDERC20.sol
```

Proof of concept

```
function testMintMaxSupply() public {
       uint256 supply = nxd.totalSupply();
       uint256 maxSupply = nxd.maxSupply();
       uint256 remainToMintBeforeError = maxSupply - supply - nxd.MAX_DEV_ALLOC();
       vm.startPrank(address(nxdProtocol));
       nxd.mint(address(this), remainToMintBeforeError);
       vm.expectRevert(NXDERC20.MaxSupply.selector);
       nxd.mint(address(this), 1);
   }
```

Recommendation

Update the logic of the mint function to allow it to mint up to *maxSupply*. It is noted that updating the function to the following code may create other issues in the deposit function logic in the NXDProtocol contract.

```
function mint(address account, uint256 amount) external {
       if (msg.sender != protocol) {
           revert Unauthorized();
       }
       if (totalSupply() + amount > maxSupply) {
           revert MaxSupply();
       }
       _mint(account, amount);
}
```

3. Misleading comments in NXDERC20:_transfer

Issue ID

NXDP-3

Status

Resolved

#235de6e58e32a525ca3372a67e34ccf24c4d5c23

Risk Level

Severity: Informational

Description & Code Segment

There is no add liquidity function, and the liquidity addition only happens at *handleTax* function called at the end of the *if* scope.

```
// Send to taxRecipient to add LP
           UNISWAP_V2_ROUTER.swapExactTokensForTokensSupportingFeeOnTransferTokens(
               sellNXDAmount, amountOutMin, nxdDXNPath, address(taxRecipient),
block.timestamp
           );
```

remainingTax is 60% of tax amount

```
// We now have DXN, add liquidity to NXD/DXN pair
```

```
           uint256 remainingTax = taxAmount - sellNXDAmount; // 30%
           uint256 burnAmount = (taxAmount * 4000) / 10000; // 2% of
all tax. 2/5% of tax amount
           uint256 devFeeAmount = (taxAmount * 1000) / 10000; // 10%
of all tax. 0.5/5% of tax amount
```

Code location

```
src/NXDERC20.sol
```

Proof of concept

–

Recommendation

Update the comments to be more accurate

4. NXDERC20:_transfer: incorrect Transfer events

Issue ID

NXDP-4

Status

Resolved

#f44219818c7aac76b6291bf0937fe6de5f0ae186

Risk Level

Severity: Medium

Code Segment

```
_balances[from] -= value;
       (uint256 amountAfterTax, uint256 taxAmount) =
getAmountsAfterTax(from, to, value);
       console.log("NXDERC20: Amount after tax: %s",
amountAfterTax);
       console.log("NXDERC20: taxAmount: %s", taxAmount);
```

```
        if (taxAmount > 0) {
            // NXD burn - 2.0%
            // DXN buy and stake - 1.5%
            // LP add - 1%
            // Dev Fee - 0.5%
            _balances[address(this)] += taxAmount;
            address[] memory nxdDXNPath = new address[](2);
            nxdDXNPath[0] = address(this);
            nxdDXNPath[1] = address(dxn);

            // Sell NXD, buy DXN and stake
            uint256 sellNXDAmount = (taxAmount * 4000) / 10000; //
40% (2/5) of total tax amount, which is 1.5% buy and stake DXN +
0.5% buy DXN to add liquidity
            _approve(address(this), address(UNISWAP_V2_ROUTER),
sellNXDAmount);
            if (v2Oracle.canUpdate()) {
                v2Oracle.update();
            }
            uint256 amountOutMin = v2Oracle.consult(address(this),
sellNXDAmount);
            // - 3%
            amountOutMin = (amountOutMin * 9700) / 10000;

            // Send to taxRecipient to add LP

UNISWAP_V2_ROUTER.swapExactTokensForTokensSupportingFeeOnTransferTok
ens(
                sellNXDAmount, amountOutMin, nxdDXNPath,
address(taxRecipient), block.timestamp
            );

            console.log("NXDERC20: tax recipient received %s DXN",
dxn.balanceOf(address(taxRecipient)));

            // We now have DXN, add liquidity to NXD/DXN pair
            uint256 remainingTax = taxAmount - sellNXDAmount; // 30%
            uint256 burnAmount = (taxAmount * 4000) / 10000; // 2% of
all tax. 2/5% of tax amount
            uint256 devFeeAmount = (taxAmount * 1000) / 10000; // 10%
of all tax. 0.5/5% of tax amount
```

```
            console.log("NXDERC20: remainingTax = ", remainingTax);
            console.log("NXDERC20: burnAmount = ", burnAmount);
            console.log("NXDERC20: Expected NXD amount to add to liq=
", remainingTax - burnAmount);
            // console.log("NXDERC20: burnAmount / taxAmount = ",
(burnAmount * 1e18) / taxAmount);

            _balances[address(this)] -= remainingTax;

            // Burn 10% from tax amount
            _balances[0xDeaDbeefdEAdbeefdEadbEEFdeadbeEFdEaDbeeF] +=
burnAmount;
            emit Transfer(from,
0xDeaDbeefdEAdbeefdEadbEEFdeadbeEFdEaDbeeF, burnAmount);
            _balances[devFeeTo] += devFeeAmount;
            emit Transfer(from, devFeeTo, devFeeAmount);

            uint256 amountToTaxHandler = remainingTax - burnAmount -
devFeeAmount;

            // Send NXD to Tax recipient to add liquidity
            _balances[address(taxRecipient)] += amountToTaxHandler;
            emit Transfer(from, address(taxRecipient),
amountToTaxHandler);
            taxRecipient.handleTax();
        }
        _balances[to] += amountAfterTax;
        emit Transfer(from, to, amountAfterTax);
```

Description

The *_transfer* function is expected to emit events with correct amounts. Incorrect amounts in the Transfer events can make it hard for off-chain analytics.

- The sum of all emitted Transfer events for the from address is *amountAfterTax + burnAmount + devFeeAmount + amountToTaxHandler* is not equal to the transferred amount value

- Line 448: There is no Transfer event emission to increase the token contract balance, which would prevent tools such as etherscan from showing that this token contract has enough token balance for swapping from NXD to DXN.
- This could lead to underflow errors.

Code location

```
src/NXDERC20.sol
```

Recommendation

- Line 448: *The* balance of the NXD token contract should increase the amount set in *sellNXDAmount*, and the function should emit a *Transfer* event *from* from to the *NXD* contract
- Remove the *balance* decrease at line 480

5. TaxRecipient: unused state variable

Issue ID

NXDP-5

Status

Resolved

#fcfd561e5ac0b141f832650ac8b52b1fea3a2e70

Risk Level

Severity: Low

Code Segment

```
function setUniswapV2Pair(address _uniswapV2Pair) external {
       if (msg.sender != address(nxd)) {
           revert OnlyNXD();
       }
       uniswapV2Pair = IUniswapV2Pair(_uniswapV2Pair);
   }
```

Description

The *uniswapV2Pair* state variable can be updated but never gets used

Recommendation

Remove the *uniswapV2Pair* state variable

Code location

```
src/TaxRecipient.sol
```

6. TaxRecipient: nxd.balanceOf(address(this)) called twice

Issue ID

NXDP-6

Status

Resolved

#fc51a8a1d990a3e130e17d1113b751c52513e858

Risk Level

Severity: Low

Code Segment

```
      dxn.approve(address(UNISWAP_V2_ROUTER), ourDXNBalance);
       nxd.approve(address(UNISWAP_V2_ROUTER),
nxd.balanceOf(address(this)));
       // Add liquidity
       UNISWAP_V2_ROUTER.addLiquidity(
           address(nxd),
           address(dxn),
           nxd.balanceOf(address(this)),
           ourDXNBalance,
           0,
           0,
           address(this),
           block.timestamp
```

```
        );
```

Description

nxd.balanceOf(address(this)) is called twice in a row without any update to the NXD balance. This can create redundant gas cost

Code location

```
src/TaxRecipient.sol:handleTax
```

Recommendation

Store the returned value of the first call in a local variable and use the variable when appropriate.

7. NXDProtocol: Misleading comment

Issue ID

NXDP-7

Status

Resolved

#e9111a7e8c9e54b2716d553d52009ab6f0aecbde

Risk Level

Severity: Information

Code Segment

```
//         DXN Staking Vault
       // • 50% Buy & Burn NXD
       // • 30% Buy & Stake DXN
       // • 15% NXD Staking Vault
       // • 5% Buy & Burn DXN

// Buy DXN with 85% of ETH received. 30% to Buy & Stake DXN + 50% to Buy & Burn NXD
+ 5% to Buy & Burn DXN
uint256 ethToSwapForDXN = (address(this).balance * 8500) / 10000;
```

5/85 or 5% or 11111111 / 100000000

```
// Burn 5/85 of DXN received (= 5% of ETH received)
        uint256 dxnToBurn = (dxnAmountReceived * 11111111) /
100000000;
```

Description

Misleading comment, it’s currently “Buy DXN with 85%” a better way to describe it would be “Buy DXN & NXD with 85%”

Code location

```
src/NXDProtocol.sol: receive()
```

Recommendation

Update the comment to be more accurate

8. Users may not be able to claim referral rewards

Issue ID

NXDP-8

Status

Resolved

Risk Level

Severity: Critical

Code Segment

```
function withdrawReferralRewards() external {
        if (block.timestamp < endTime) {
            revert CSPOngoing();
        }
        uint256 amount = referredRewards[msg.sender] + referrerRewards[msg.sender];
        if (amount == 0) {
            revert NoRewards();
        }
        referrerRewards[msg.sender] = 0;
        referredRewards[msg.sender] = 0;
        nxd.mint(msg.sender, amount);
        emit ReferralRewardsWithdrawn(msg.sender, amount);
    }
```

Description

Users may not be able to claim all referral rewards even if the user has referral rewards to claim due to the following reasons:

- The *deposit* function computes referral rewards for users; however does not take the user bonus amount of the referral rewards into NXD total supply. During CSP, if users deposit DXN into the protocol contract and the minted NXD reaches *maxSupply - MAX_DEV_ALLOC*, not all referral rewards (only claimable after CSP complete) would be minted for the users as the mint function of NXD will revert

Code location

```
src/NXDProtocol.sol
```

Proof of Concept

```
function testUsersLostReferralRewards() public {
        address referrer = 0x98f6e135E44d90A2799bc9C645c02Ee7489453c5;
        vm.prank(referrer);
        nxdProtocol.setReferralCode(1234);

        address dxnWhale = 0xF5c80c305803280B587F8cabBcCdC4d9BF522AbD;
        uint256 amount = uint256(634782608695652173913044) * 1005 / 1000;
        vm.startPrank(dxnWhale);
        dxn.approve(address(nxdProtocol), amount);
        nxdProtocol.deposit(amount, 1234, true);
        assert(nxdProtocol.referredRewards(dxnWhale) > 0);
        assert(nxdProtocol.referrerRewards(referrer) > 0);

        vm.warp(endTime + 1);
        address[] memory recipients = new address[](1);
        vm.startPrank(bob);
        nxdProtocol.mintDevAlloc(recipients);
        recipients[0] = devFeeTo;
```

```
      assert(nxdProtocol.referredRewards(dxnWhale) +
nxdProtocol.referrerRewards(referrer) + nxd.totalSupply() > nxd.maxSupply());
   }
```

Recommendation

Take *userBonusAmount* into account for NXD supply during CSP. Specifically, update the condition at line 211 as the following:

```
if (nxd.totalSupply() + amountReceived + referrerAmount + userBonusAmount >
nxd.maxSupply() - nxd.MAX_DEV_ALLOC())
```

9. NXDProtocol: mintDevAlloc redundant function calls

Issue ID

NXDP-9

Status

Resolved

#e1042afcd2959aa630a51dbb74e6e3659a53419c

Risk Level

Severity: Low

Code Segment

```
if (devAlloc > 0) {
            if (devAlloc + totalNXDMinted > nxd.maxSupply()) {
                devAlloc = nxd.maxSupply() - totalNXDMinted;
            }

            nxd.mint(address(this), devAlloc);
            nxd.transfer(address(vesting), devAlloc);
            uint256 devAllocPerRecipient = devAlloc /
recipients.length;
```

```
            for (uint256 i = 0; i < recipients.length; i++) {
                vesting.setVesting(recipients[i],
devAllocPerRecipient);
            }
        }
```

Description

The two statements at lines 351 and 352 can be simplified into a single statement calling the mint function of NXD to mint NXD directly to the vesting contract.

Code location

```
src/NXDProtocol.sol
```

Recommendation

Simplify the two statements into one as follows:

```
nxd.mint(address(this), address(vesting));
```

10. NXDStakingVault: _add function gas optimizations

Issue ID

NXDP-10

Status

Resolved

#0589aab00191b8893bb0dca73bd48e198f2f165d

Risk Level

Severity: Low

Code Segment

```
        uint256 length = numPools;
        for (uint256 pid = 0; pid < length; ++pid) {
            if (poolInfo[pid].token == _token) {
                revert PoolAlreadyAdded();
            }
        }

        totalAllocPoint = totalAllocPoint + _allocPoint;

        PoolInfo storage _poolInfo = poolInfo[numPools];

        _poolInfo.token = _token;
        _poolInfo.allocPoint = _allocPoint;
        _poolInfo.accEthPerShare = 0;
        _poolInfo.withdrawable = _withdrawable;

        numPools += 1;

        emit Add(address(_token), numPools - 1, _allocPoint, _withdrawable);
```

Description

Line 124: use *length* instead of *numPools* as *numPools* is a storage variable

Line 133: use *length* instead of *numPools - 1*

Code location

```
src/NXDStakingVault.sol
```

Recommendation

In description

11. NXDStakingVault: pendingETH function does not take pending rewards into account

Issue ID

NXDP-11

Status

Resolved

Risk Level

Severity: Medium

Description

- The function calculates a staking user's possible ETH rewards when claiming rewards.
- The function, however, does not take *pendingRewards* into account. These pending rewards are added to the user's pending rewards in the *massUpdatePools* function.
- The function pendingETH is meant to be called by the front end to show to users. As the function does not consider pending rewards, the rewards users receive when claiming will be different, which may lead to user experience issues.

Code Segment

```
function pendingETH(uint256 _pid, address _user) public view returns (uint256) {
       PoolInfo storage pool = poolInfo[_pid];
       UserInfo storage user = userInfo[_pid][_user];
       uint256 accEthPerShare = pool.accEthPerShare;

       return ((user.amount * accEthPerShare) / 1e12) - user.rewardDebt;
   }
```

Code Location

```
src/NXDStakingVault.sol
```

Recommendation

- The function should calculate the increase of *accEthPerShare* by taking *pendingRewards* into account, and compute the correct rewards for users.
- The following shows an example of how it could be implemented.

```
function pendingETH(uint256 _pid, address _user) public view returns (uint256) {
       PoolInfo storage pool = poolInfo[_pid];
       UserInfo storage user = userInfo[_pid][_user];
```

```
        uint256 accEthPerShare = pool.accEthPerShare;

        uint256 tokenSupply = pool.token.balanceOf(address(this));
        if (tokenSupply == 0) {
            return 0;
        }
        uint256 ethReward = (pendingRewards * pool.allocPoint) / totalAllocPoint;
        accEthPerShare += ((ethReward * 1e12) / tokenSupply);

        return ((user.amount * accEthPerShare) / 1e12) - user.rewardDebt;
    }
```

12. NXDStakingVault: unused state variables

Issue ID

NXDP-12

Status

Resolved

#336c2ba2b4c7070a3a5bfb2786d1733fc252e0d6

Risk Level

Severity: Medium

Description

Unused state variables: *epoch*, *epochRewards*, *epochCalculationStartBlock*, *cumulativeRewardsSinceStart*, *contractStartBlock*

It is highly recommended to review the way how epochs in the reference cvault staking contract works for computing total rewards generated in an epoch, which is used for computing average staking APY: https://github.com/cVault-finance/CORE-v1/blob/feafc2e65488a0112719a92ef0bfbf2163d15319/contracts/CoreVault.sol#L82

Code Segment

-

Code Location

```
src/NXDStakingVault.sol
```

Recommendation

Review and implement the epoch mechanism for computing accumulated rewards in an epoch and how the epoch is used for computing average staking APY of cvault staking contract at:

https://github.com/cVault-finance/CORE-v1/blob/feafc2e65488a0112719a92ef0bfbf2163d15319/contracts/CoreVault.sol#L82

13. NXDStakingVault: withdraw function should be improved for better user experience

Issue ID

NXDP-13

Status

Resolved

#481e7f2ba411464a442eb0a21bc97c9c8255b3c6

Risk Level

Severity: Medium

Description

- The function reverts if the user has an existing withdrawable withdraw request
- The function shouldn't revert if there is an existing withdrawal request, which already passes the cooldown time
- Requiring the user with a pending withdrawal request to explicitly claim the withdrawable withdraw request would create an unfriendly user experience on the frontend application as it requires users to sign more than one transaction.

Code Segment

```
WithdrawalRequest storage request = withdrawalRequests[_pid][msg.sender];

       if (_amount > 0) {
           // Stop receiving rewards for this amount NOW
           user.amount = user.amount - _amount;

           if (acceptsPenalty) {
               userGetsAfterPenalty = (_amount * 7500) / 10000;
           } else {
               // 0 means Needs to wait 24 hours
               if (request.canWithdrawAfterTimestamp == 0) {
                   uint256 timestamp = block.timestamp + WITHDRAWAL_COOLDOWN;
                   withdrawalRequests[_pid][msg.sender] =
WithdrawalRequest(_amount, timestamp);
                   user.rewardDebt = (user.amount * pool.accEthPerShare) / 1e12;
                   emit WithdrawRequested(from, _amount, timestamp);
                   return;
               } else {
                   revert Cooldown();
               }
           }

           // If we are here we can withdraw

           pool.token.safeTransfer(address(to), userGetsAfterPenalty);

           // Burn penalty amount
           if (userGetsAfterPenalty < _amount) {
               nxd.transfer(0xDeaDbeefdEAdbeefdEadbEEFdeadbeEFdEaDbeeF, _amount -
userGetsAfterPenalty);
           }
       }
```

Code Location

```
src/NXDStakingVault.sol
```

Recommendation

If the user has a withdrawable withdrawal request, the contract should automatically claim the withdrawable request for the user. This makes UX better on the frontend as fewer user transactions would be required.

The following show a possible implementation

```
       WithdrawalRequest storage request = withdrawalRequests[_pid][msg.sender];
```

```
        if (_amount > 0) {
            if (block.timestamp >= request.canWithdrawAfterTimestamp) {
                withdrawCooldown(_pid);
            }
            // Stop receiving rewards for this amount NOW
            user.amount = user.amount - _amount;
            // .. the rest of the function
```

14. Vesting: setVesting function could override any existing vesting schedule if has for a user

Issue ID

NXDP-14

Status

Resolved

#23683b8963184f06f6c3135961945bc1cc7d9d89

Risk Level

Severity: High

Description

- Function overwrites any existing vesting schedule for an address.
- As this function is only callable by *mintDevAlloc* function of NXDProtocol, if *mintDevAlloc* is accidentally called more than once with the same input parameters, vesting token would be lost due to *setVesting* overwriting existing vesting schedule of an *account*

Code Segment

```
function setVesting(address user, uint256 amount) public {
        if (msg.sender != owner) {
            revert Owner();
        }
        tokenAmountToVest[user] = VestingSchedule(amount, block.timestamp);
    }
```

Code Location

```
src/Vesting.sol
```

Recommendation

Instead of overwriting any existing *VestingSchedule* at line 50, it is better to add the new vesting amount to an existing one (assuming that all addresses have a default 0 vesting amount)

The following shows a possible implementation.

```
function setVesting(address user, uint256 amount) public {
      if (msg.sender != owner) {
          revert Owner();
      }
      uint256 claimedAmount = claimed[user];
      VestingSchedule storage vesting = tokenAmountToVest[user];
      vesting.amount = vesting.amount + amount - claimedAmount;
      vesting.startTimestamp = block.timestamp;
  }
```

15. Console.log should be removed.

Issue ID

NXDP-15

Status

Resolved

#ed5f20bb65c04ac7429e409120ebed74de100a76

Risk Level

Severity: Low

Description

Console.log is used for debugging purposes and should not be part of the production code.

Code Segment

Code Location

```
All scoped files
```

Recommendation

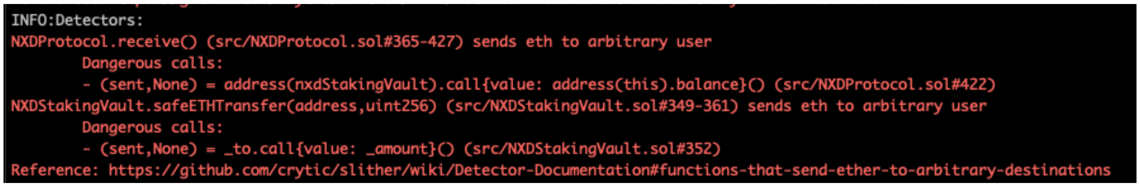
Remove console.logs

Automated Audit

Static Analysis with Slither

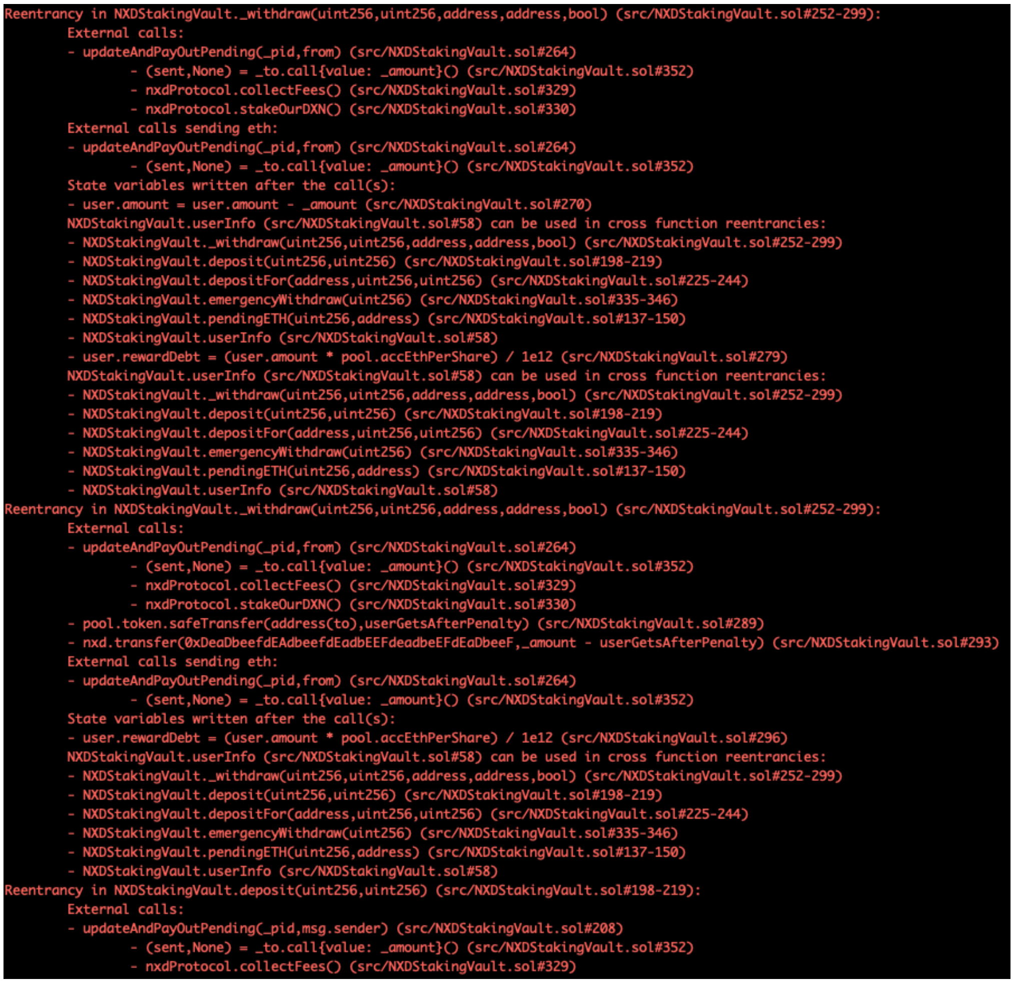
The following shows the results found by the static analysis by Slither.

- False positives and these calls are safe in the context of the contracts



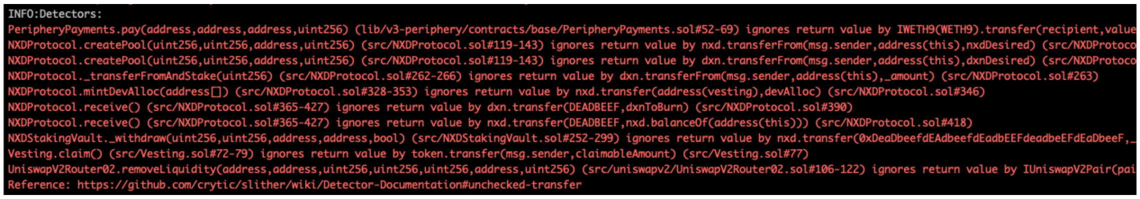
```
INFO:Detectors:
NXDProtocol.receive() (src/NXDProtocol.sol#365-427) sends eth to arbitrary user
        Dangerous calls:
        - (sent,None) = address(nxdStakingVault).call{value: address(this).balance}() (src/NXDProtocol.sol#422)
NXDStakingVault.safeETHTransfer(address,uint256) (src/NXDStakingVault.sol#349-361) sends eth to arbitrary user
        Dangerous calls:
        - (sent,None) = _to.call{value: _amount}() (src/NXDStakingVault.sol#352)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
```

- False positives reentrancy



```
Reentrancy in NXDStakingVault._withdraw(uint256,uint256,address,address,bool) (src/NXDStakingVault.sol#252-299):
        External calls:
        - updateAndPayOutPending(_pid,from) (src/NXDStakingVault.sol#264)
                - (sent,None) = _to.call{value: _amount}() (src/NXDStakingVault.sol#352)
                - nxdProtocol.collectFees() (src/NXDStakingVault.sol#329)
                - nxdProtocol.stakeOurDXN() (src/NXDStakingVault.sol#330)
        External calls sending eth:
        - updateAndPayOutPending(_pid,from) (src/NXDStakingVault.sol#264)
                - (sent,None) = _to.call{value: _amount}() (src/NXDStakingVault.sol#352)
        State variables written after the call(s):
        - user.amount = user.amount - _amount (src/NXDStakingVault.sol#270)
        NXDStakingVault.userInfo (src/NXDStakingVault.sol#58) can be used in cross function reentrancies:
        - NXDStakingVault._withdraw(uint256,uint256,address,address,bool) (src/NXDStakingVault.sol#252-299)
        - NXDStakingVault.deposit(uint256,uint256) (src/NXDStakingVault.sol#198-219)
        - NXDStakingVault.depositFor(address,uint256,uint256) (src/NXDStakingVault.sol#225-244)
        - NXDStakingVault.emergencyWithdraw(uint256) (src/NXDStakingVault.sol#335-346)
        - NXDStakingVault.pendingETH(uint256,address) (src/NXDStakingVault.sol#137-150)
        - NXDStakingVault.userInfo (src/NXDStakingVault.sol#58)
        - user.rewardDebt = (user.amount * pool.accEthPerShare) / 1e12 (src/NXDStakingVault.sol#279)
        NXDStakingVault.userInfo (src/NXDStakingVault.sol#58) can be used in cross function reentrancies:
        - NXDStakingVault._withdraw(uint256,uint256,address,address,bool) (src/NXDStakingVault.sol#252-299)
        - NXDStakingVault.deposit(uint256,uint256) (src/NXDStakingVault.sol#198-219)
        - NXDStakingVault.depositFor(address,uint256,uint256) (src/NXDStakingVault.sol#225-244)
        - NXDStakingVault.emergencyWithdraw(uint256) (src/NXDStakingVault.sol#335-346)
        - NXDStakingVault.pendingETH(uint256,address) (src/NXDStakingVault.sol#137-150)
        - NXDStakingVault.userInfo (src/NXDStakingVault.sol#58)
Reentrancy in NXDStakingVault._withdraw(uint256,uint256,address,address,bool) (src/NXDStakingVault.sol#252-299):
        External calls:
        - updateAndPayOutPending(_pid,from) (src/NXDStakingVault.sol#264)
                - (sent,None) = _to.call{value: _amount}() (src/NXDStakingVault.sol#352)
                - nxdProtocol.collectFees() (src/NXDStakingVault.sol#329)
                - nxdProtocol.stakeOurDXN() (src/NXDStakingVault.sol#330)
        - pool.token.safeTransfer(address(to),userGetsAfterPenalty) (src/NXDStakingVault.sol#289)
        - nxd.transfer(0xDeaDbeefdEAdbeefdEadbEEFdeadbeEFdEaDbeeF,_amount - userGetsAfterPenalty) (src/NXDStakingVault.sol#293)
        External calls sending eth:
        - updateAndPayOutPending(_pid,from) (src/NXDStakingVault.sol#264)
                - (sent,None) = _to.call{value: _amount}() (src/NXDStakingVault.sol#352)
        State variables written after the call(s):
        - user.rewardDebt = (user.amount * pool.accEthPerShare) / 1e12 (src/NXDStakingVault.sol#296)
        NXDStakingVault.userInfo (src/NXDStakingVault.sol#58) can be used in cross function reentrancies:
        - NXDStakingVault._withdraw(uint256,uint256,address,address,bool) (src/NXDStakingVault.sol#252-299)
        - NXDStakingVault.deposit(uint256,uint256) (src/NXDStakingVault.sol#198-219)
        - NXDStakingVault.depositFor(address,uint256,uint256) (src/NXDStakingVault.sol#225-244)
        - NXDStakingVault.emergencyWithdraw(uint256) (src/NXDStakingVault.sol#335-346)
        - NXDStakingVault.pendingETH(uint256,address) (src/NXDStakingVault.sol#137-150)
        - NXDStakingVault.userInfo (src/NXDStakingVault.sol#58)
Reentrancy in NXDStakingVault.deposit(uint256,uint256) (src/NXDStakingVault.sol#198-219):
        External calls:
        - updateAndPayOutPending(_pid,msg.sender) (src/NXDStakingVault.sol#208)
                - (sent,None) = _to.call{value: _amount}() (src/NXDStakingVault.sol#352)
                - nxdProtocol.collectFees() (src/NXDStakingVault.sol#329)
```

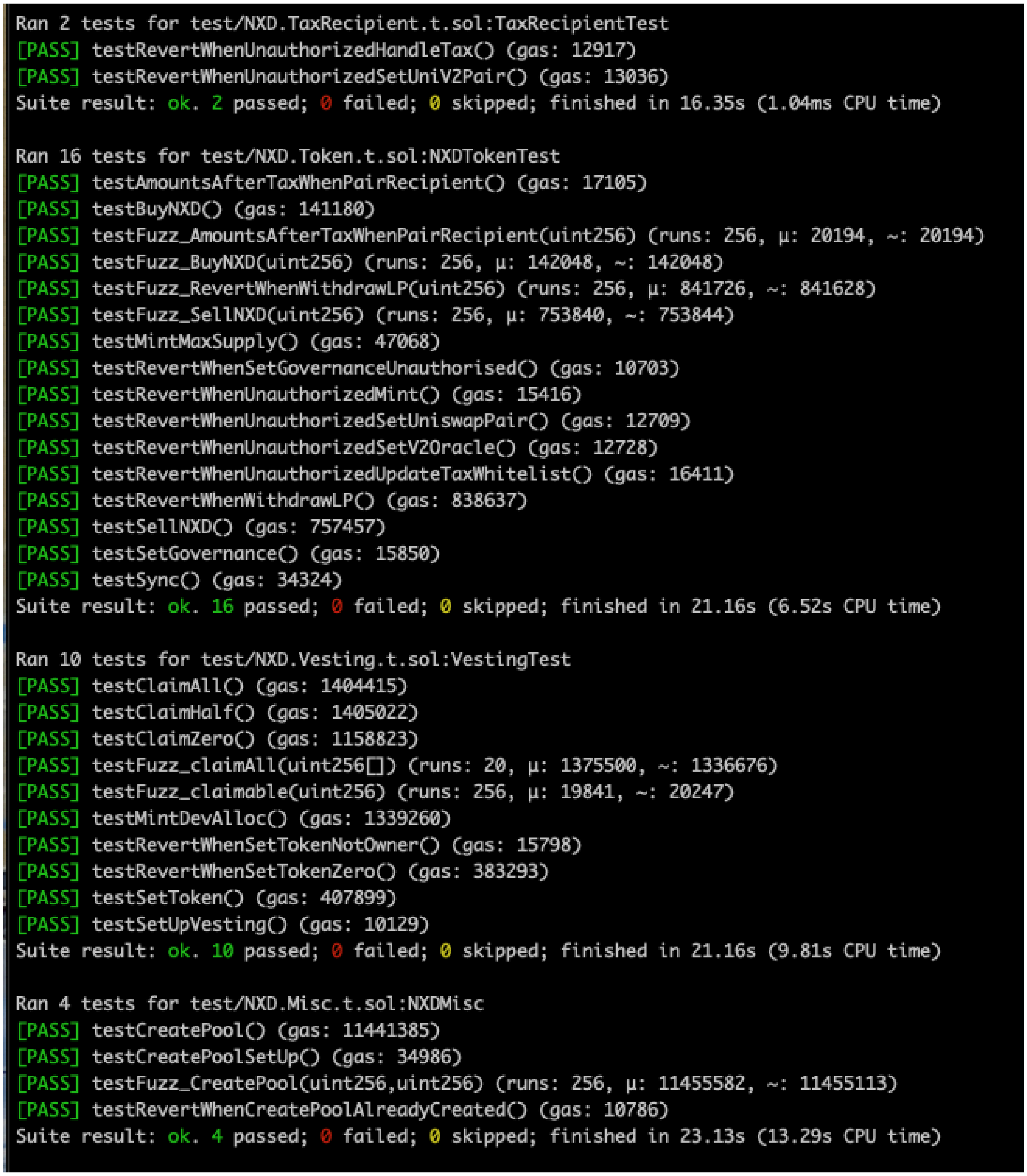
- False positives for transfer functions



```
INFO:Detectors:
PeripheryPayments.pay(address,address,address,uint256) (lib/v3-periphery/contracts/base/PeripheryPayments.sol#52-69) ignores return value by IWETH9(WETH9).transfer(recipient,value
NXDProtocol.createPool(uint256,uint256,address,uint256) (src/NXDProtocol.sol#119-143) ignores return value by nxd.transferFrom(msg.sender,address(this),nxdDesired) (src/NXDProtoco
NXDProtocol.createPool(uint256,uint256,address,uint256) (src/NXDProtocol.sol#119-143) ignores return value by dxn.transferFrom(msg.sender,address(this),dxnDesired) (src/NXDProtoco
NXDProtocol._transferFromAndStake(uint256) (src/NXDProtocol.sol#262-266) ignores return value by dxn.transferFrom(msg.sender,address(this),_amount) (src/NXDProtocol.sol#263)
NXDProtocol.mintDevAlloc(address[]) (src/NXDProtocol.sol#328-353) ignores return value by nxd.transfer(address(vesting),devAlloc) (src/NXDProtocol.sol#346)
NXDProtocol.receive() (src/NXDProtocol.sol#365-427) ignores return value by dxn.transfer(DEADBEEF,dxnToBurn) (src/NXDProtocol.sol#390)
NXDProtocol.receive() (src/NXDProtocol.sol#365-427) ignores return value by nxd.transfer(DEADBEEF,nxd.balanceOf(address(this))) (src/NXDProtocol.sol#418)
NXDStakingVault._withdraw(uint256,uint256,address,address,bool) (src/NXDStakingVault.sol#252-299) ignores return value by nxd.transfer(0xDeaDbeefdEAdbeefdEadbEEFdeadbeEFdEaDbeeF,_
Vesting.claim() (src/Vesting.sol#72-79) ignores return value by token.transfer(msg.sender,claimableAmount) (src/Vesting.sol#77)
UniswapV2Router02.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256) (src/uniswapv2/UniswapV2Router02.sol#106-122) ignores return value by IUniswapV2Pair(pai
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
```

Unit Tests

All unit tests passed

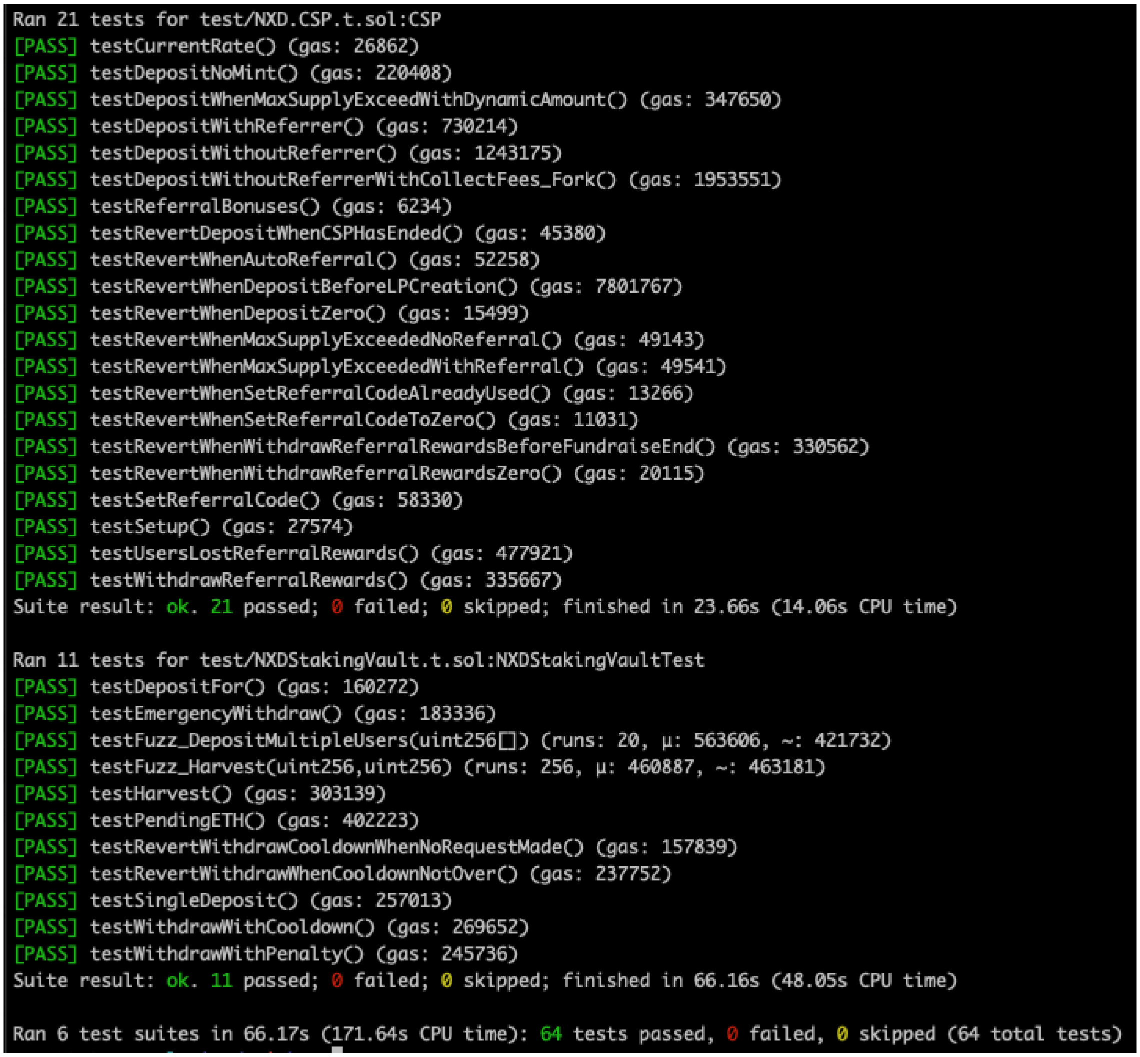


```
Ran 2 tests for test/NXD.TaxRecipient.t.sol:TaxRecipientTest
[PASS] testRevertWhenUnauthorizedHandleTax() (gas: 12917)
[PASS] testRevertWhenUnauthorizedSetUniV2Pair() (gas: 13036)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 16.35s (1.04ms CPU time)

Ran 16 tests for test/NXD.Token.t.sol:NXDTokenTest
[PASS] testAmountsAfterTaxWhenPairRecipient() (gas: 17105)
[PASS] testBuyNXD() (gas: 141180)
[PASS] testFuzz_AmountsAfterTaxWhenPairRecipient(uint256) (runs: 256, μ: 20194, ~: 20194)
[PASS] testFuzz_BuyNXD(uint256) (runs: 256, μ: 142048, ~: 142048)
[PASS] testFuzz_RevertWhenWithdrawLP(uint256) (runs: 256, μ: 841726, ~: 841628)
[PASS] testFuzz_SellNXD(uint256) (runs: 256, μ: 753840, ~: 753844)
[PASS] testMintMaxSupply() (gas: 47068)
[PASS] testRevertWhenSetGovernanceUnauthorised() (gas: 10703)
[PASS] testRevertWhenUnauthorizedMint() (gas: 15416)
[PASS] testRevertWhenUnauthorizedSetUniswapPair() (gas: 12709)
[PASS] testRevertWhenUnauthorizedSetV2Oracle() (gas: 12728)
[PASS] testRevertWhenUnauthorizedUpdateTaxWhitelist() (gas: 16411)
[PASS] testRevertWhenWithdrawLP() (gas: 838637)
[PASS] testSellNXD() (gas: 757457)
[PASS] testSetGovernance() (gas: 15850)
[PASS] testSync() (gas: 34324)
Suite result: ok. 16 passed; 0 failed; 0 skipped; finished in 21.16s (6.52s CPU time)

Ran 10 tests for test/NXD.Vesting.t.sol:VestingTest
[PASS] testClaimAll() (gas: 1404415)
[PASS] testClaimHalf() (gas: 1405022)
[PASS] testClaimZero() (gas: 1158823)
[PASS] testFuzz_claimAll(uint256[]) (runs: 20, μ: 1375500, ~: 1336676)
[PASS] testFuzz_claimable(uint256) (runs: 256, μ: 19841, ~: 20247)
[PASS] testMintDevAlloc() (gas: 1339260)
[PASS] testRevertWhenSetTokenNotOwner() (gas: 15798)
[PASS] testRevertWhenSetTokenZero() (gas: 383293)
[PASS] testSetToken() (gas: 407899)
[PASS] testSetUpVesting() (gas: 10129)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 21.16s (9.81s CPU time)

Ran 4 tests for test/NXD.Misc.t.sol:NXDMisc
[PASS] testCreatePool() (gas: 11441385)
[PASS] testCreatePoolSetUp() (gas: 34986)
[PASS] testFuzz_CreatePool(uint256,uint256) (runs: 256, μ: 11455582, ~: 11455113)
[PASS] testRevertWhenCreatePoolAlreadyCreated() (gas: 10786)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 23.13s (13.29s CPU time)
```



```
Ran 21 tests for test/NXD.CSP.t.sol:CSP
[PASS] testCurrentRate() (gas: 26862)
[PASS] testDepositNoMint() (gas: 220408)
[PASS] testDepositWhenMaxSupplyExceedWithDynamicAmount() (gas: 347650)
[PASS] testDepositWithReferrer() (gas: 730214)
[PASS] testDepositWithoutReferrer() (gas: 1243175)
[PASS] testDepositWithoutReferrerWithCollectFees_Fork() (gas: 1953551)
[PASS] testReferralBonuses() (gas: 6234)
[PASS] testRevertDepositWhenCSPHasEnded() (gas: 45380)
[PASS] testRevertWhenAutoReferral() (gas: 52258)
[PASS] testRevertWhenDepositBeforeLPCreation() (gas: 7801767)
[PASS] testRevertWhenDepositZero() (gas: 15499)
[PASS] testRevertWhenMaxSupplyExceededNoReferral() (gas: 49143)
[PASS] testRevertWhenMaxSupplyExceededWithReferral() (gas: 49541)
[PASS] testRevertWhenSetReferralCodeAlreadyUsed() (gas: 13266)
[PASS] testRevertWhenSetReferralCodeToZero() (gas: 11031)
[PASS] testRevertWhenWithdrawReferralRewardsBeforeFundraiseEnd() (gas: 330562)
[PASS] testRevertWhenWithdrawReferralRewardsZero() (gas: 20115)
[PASS] testSetReferralCode() (gas: 58330)
[PASS] testSetup() (gas: 27574)
[PASS] testUsersLostReferralRewards() (gas: 477921)
[PASS] testWithdrawReferralRewards() (gas: 335667)
Suite result: ok. 21 passed; 0 failed; 0 skipped; finished in 23.66s (14.06s CPU time)

Ran 11 tests for test/NXDStakingVault.t.sol:NXDStakingVaultTest
[PASS] testDepositFor() (gas: 160272)
[PASS] testEmergencyWithdraw() (gas: 183336)
[PASS] testFuzz_DepositMultipleUsers(uint256[]) (runs: 20, µ: 563606, ~: 421732)
[PASS] testFuzz_Harvest(uint256,uint256) (runs: 256, µ: 460887, ~: 463181)
[PASS] testHarvest() (gas: 303139)
[PASS] testPendingETH() (gas: 402223)
[PASS] testRevertWithdrawCooldownWhenNoRequestMade() (gas: 157839)
[PASS] testRevertWithdrawWhenCooldownNotOver() (gas: 237752)
[PASS] testSingleDeposit() (gas: 257013)
[PASS] testWithdrawWithCooldown() (gas: 269652)
[PASS] testWithdrawWithPenalty() (gas: 245736)
Suite result: ok. 11 passed; 0 failed; 0 skipped; finished in 66.16s (48.05s CPU time)

Ran 6 test suites in 66.17s (171.64s CPU time): 64 tests passed, 0 failed, 0 skipped (64 total tests)
```

Conclusion

Arcadia identified issues that occurred at the following repository:

- **DXNhyperstructure/NXD-Protocol** at commit #cd12c385b5f12a2f2779ebfd4b21884ec9172b6a as defined in the scope as in Section ‘Introduction and Audit Scope’

Disclaimer

While best efforts and precautions have been taken in the preparation of this document, Arcadia and the Authors assume no responsibility for errors, omissions, or damages resulting from the use of the provided information. Additionally, Arcadia would like to emphasize that the use of Arcadia's services does not guarantee the security of a smart contract or set of smart contracts and does not guarantee against attacks. One audit on its own is not enough for a project to be considered secure; that categorization can only be earned through extensive peer review and battle testing over an extended period.