

Yeet Security Review Report

August 3, 2025



Version 1.0

Email

audits@extropy.io

Telephone

+44 1865338228

Data Classification	Client Confidential
Client Name	Yeet
Document Title	Yeet Security Review Report
Document Version	1.0
Author	Extropy Audit Team

Contents

1	Executive Summary	1
2	Audit summary	2
2.1	Audit scope	2
2.2	Critical Audit notes	2
2.3	Issues Summary	4
2.4	Methodology	4
2.5	Approach	4
3	System Overview	5
4	Findings	7
4.1	[CRITICAL] 'accumulatedDeptRewardsYeet()' accounts for tokens under unstaking process	7
4.2	[MEDIUM] Volume is assigned to wrong epoch if no yeets are made altering the claimable amount	9
4.3	[LOW] Setting a low value for 'POT_DIVISION' helps whales to win the current round . .	10
4.4	[LOW] Blacklisting a Kodiak vault unintentionally whitelists a previously blacklisted token	11
4.5	[LOW] Arbitrary \$BERA sent to contracts are locked	12
4.6	[LOW] DOS protection may be circumvented through multiple identities	13
4.7	[INFO] NFT Boost Lookup values not adhere to docs	13
4.8	[INFO] Missing requires in '_swapNativeToTokens()'	13
4.9	[INFO] Wrong function visibility	14
4.10	[INFO] Misleading error name	15
4.11	[INFO] Not following CEI pattern may lead to cross-contract and read-only reentrancy in the future	15
4.12	[INFO] 'stakedTimes' and 'STAKING_LIMIT' names and comments are misleading	16
4.13	[INFO] Very small unstakes does not follow linear unlocking scheme when performing a 'rageQuit()'	17
5	Test Coverage	18
5.1	Solidity tests	18
A	Disclaimers	26
A.1	Client Confidentiality	26
A.2	Proprietary Information	26
B	Slither results	27

1 | Executive Summary

This report provides an overview of the security assessment conducted by Extropy.io during its participation in the bug bounty program for the Yeet project on the Immunefi platform. The primary objective of this assessment was to identify and mitigate potential security vulnerabilities within the Yeet smart contract infrastructure, ensuring the robustness and reliability of its Web3 ecosystem.

As part of the bug bounty competition, Extropy.io performed an extensive review of Yeet's smart contracts, analyzing their logic, security controls, and resistance to various attack vectors, including reentrancy attacks, privilege escalations, and denial-of-service vulnerabilities. The engagement leveraged both automated and manual auditing techniques to maximize coverage and accuracy.

Throughout the assessment, Extropy.io identified several security issues categorized by severity levels, ranging from critical to low-impact vulnerabilities. Detailed reports outlining these findings were submitted through Immunefi's platform, accompanied by suggested remediations to enhance the project's security posture. Yeet's development team has been responsive in addressing these concerns, implementing necessary fixes to mitigate risks effectively.

The participation of Extropy.io in the Immunefi bug bounty program demonstrates a commitment to fostering a secure and resilient blockchain ecosystem. By proactively identifying vulnerabilities and collaborating with project teams, Extropy.io contributes to the continuous improvement of smart contract security within the Web3 space.

This report serves as a comprehensive record of Extropy.io's findings and recommendations, emphasizing the importance of ongoing security assessments to protect user assets and maintain trust within the decentralized finance (DeFi) ecosystem.

2 | Audit summary

This audit, conducted from March 11th 2025 to March 25th 2025, employed an approach with manual review methodology. Our examination aimed to ensure the robustness and security of the Yeet codebase.

- The code is taken from the ImmuneFi Yeet competition.
- The audit was performed on commit [da15231](#) in the [audit-comp-yeet](#) repository

2.1 | Audit scope

The following contracts were audited:

Contract	LoC
StakeV2.sol	440
Zapper.sol	730
MoneyBrinter.sol	455
Yeet.sol	608
YeetGameSettings.sol	148
YeetToken.sol	29
Yeetback.sol	177
Reward.sol	207
RewardSettings.sol	55
Total	2849

2.2 | Critical Audit notes

The security review conducted by Extropy.io was performed subsequent to the pre-audit assessment carried out by ImmuneFi, the official organizer of the bug bounty program. The results of ImmuneFi's pre-audit are publicly accessible at the following link: [ImmuneFi Pre-Audit Report](#). As a result, any vulnerabilities identified in the pre-audit were omitted from the Findings section of this report to avoid redundancy and to focus exclusively on newly discovered security risks.

We also analysed the Yeet game results from the official [dAPP](#) and Berachain explorers, potentially suspicious activity was noted related to the fairness of the game. However, no concrete attack vector could be identified, and further investigation did not reveal any exploitable vulnerabilities related to the following observation.

The way the Yeet game is implemented gives some concerns regarding its fairness. Particularly in the last 11 rounds of the game (at the time of writing), the same users won the grand prize (80%) and several lottery tickets. Those appear to be under the same entity by doing basic on-chain analysis. Indeed, the user named [6655321.bera](#) won round 9 and also 7 out of 10 lottery ticket for the same round. Also he won round 10 and 2 out of 10 lottery ticket of round 11. Please note that the user keeps changing its name so it may be different when reading this document. Screenshots below.

Furthermore [6655321.bera](#) is connected with [0xfd389e863e0a9d6cdcbde645e9c066255326ad95](#) through this transfer. This new user won round 11 and 3 out of 10 lottery ticket of round 11.

6655321.🐼	🏆 WINNER ROUND IX	3.27K \$BERA 7 DAYS AGO
6655321.🐼	🎟️ LOTTERY WINNER	101 \$BERA 7 DAYS AGO
6655321.🐼	🎟️ LOTTERY WINNER	101 \$BERA 7 DAYS AGO
6655321.🐼	🎟️ LOTTERY WINNER	101 \$BERA 7 DAYS AGO
6655321.🐼	🎟️ LOTTERY WINNER	101 \$BERA 7 DAYS AGO
6655321.🐼	🎟️ LOTTERY WINNER	101 \$BERA 7 DAYS AGO
6655321.🐼	🎟️ LOTTERY WINNER	101 \$BERA 7 DAYS AGO
6655321.🐼	🎟️ LOTTERY WINNER	101 \$BERA 7 DAYS AGO
0X7579...0636	🎟️ LOTTERY WINNER	101 \$BERA 7 DAYS AGO
0X99F1...0D9B	🎟️ LOTTERY WINNER	101 \$BERA 7 DAYS AGO
0XFE2D...6032	🎟️ LOTTERY WINNER	101 \$BERA 7 DAYS AGO

0XFD38...AD95	🏆 WINNER ROUND XI	363 \$BERA 4 DAYS AGO
0X5592...691B	🎟️ LOTTERY WINNER	11.2 \$BERA 4 DAYS AGO
0X5592...691B	🎟️ LOTTERY WINNER	11.2 \$BERA 4 DAYS AGO
6655321.🐼	🎟️ LOTTERY WINNER	11.2 \$BERA 4 DAYS AGO
6655321.🐼	🎟️ LOTTERY WINNER	11.2 \$BERA 4 DAYS AGO
0XD22C...F3D0	🎟️ LOTTERY WINNER	11.2 \$BERA 4 DAYS AGO
FLOY222.🐼	🎟️ LOTTERY WINNER	11.2 \$BERA 4 DAYS AGO
BEARTARDIO.🐼	🎟️ LOTTERY WINNER	11.2 \$BERA 4 DAYS AGO
0XFD38...AD95	🎟️ LOTTERY WINNER	11.2 \$BERA 4 DAYS AGO
0XFD38...AD95	🎟️ LOTTERY WINNER	11.2 \$BERA 4 DAYS AGO
0XFD38...AD95	🎟️ LOTTERY WINNER	11.2 \$BERA 4 DAYS AGO
6655321.🐼	🏆 WINNER ROUND X	335 \$BERA 6 DAYS AGO
0X1343 029B	🎟️ LOTTERY WINNER	103 \$BERA 6 DAYS AGO

Finally 6655321.bera is also connected with 0x92ca4b8fbb5d59b40460bcdcf2322dcdea4b00f28 through this transfer. This new user won 2 out of 10 lottery ticket of round 10.

The same thing happened with the users named [ponzidestroyer2.bera](#) and [ponzidestroyer3.bera](#). Indeed 'ponzidestroyer2.bera' won rounds 4, 5, 6, 7 while 'ponzidestroyer3.bera' won round 8. Among those mentioned rounds they won several lottery tickets as well. Those addresses appear to be connected and managed by the same entity (i.e. they are transferring wins to each other [here](#)).

2.3 | Issues Summary

ID	Finding	Status
4.1	[CRITICAL] 'accumulatedDeptRewardsYeet()' accounts for tokens under unstaking process	Unresolved
4.2	[MEDIUM] Volume is assigned to wrong epoch if no yeets are made altering the claimable amount	Unresolved
4.3	[LOW] Setting a low value for 'POT_DIVISION' helps whales to win the current round	Unresolved
4.4	[LOW] Blacklisting a Kodiak vault unintentionally whitelists a previously blacklisted token	Unresolved
4.5	[LOW] Arbitrary \$BERA sent to contracts are locked	Unresolved
4.6	[LOW] DOS protection may be circumvented through multiple identities	Unresolved
4.7	[INFO] NFT Boost Lookup values not adhere to docs	Unresolved
4.8	[INFO] Missing requires in '_swapNativeToTokens()'	Unresolved
4.9	[INFO] Wrong function visibility	Unresolved
4.10	[INFO] Misleading error name	Unresolved
4.11	[INFO] Not following CEI pattern may lead to cross-contract and read-only reentrancy in the future	Unresolved
4.12	[INFO] 'stakedTimes' and 'STAKING_LIMIT' names and comments are misleading	Unresolved
4.13	[INFO] Very small unstakes does not follow linear unlocking scheme when performing a 'rageQuit()'	Unresolved

2.4 | Methodology

2.4.1 | Risk Rating

The risk rating given for issues follows the standard approach of the OWASP Foundation. We combine two factors :

- Likelihood of exploit
- Impact of Exploit

The Categories we use are *Critical*, *High*, *Medium*, *Low* and *Informational* These categories may not align with the categories used by other companies.

The informational category is used to contain suggestions for optimisation (where this is not seen as causing significant impact), or for alternative design or best practices.

2.5 | Approach

The project was assessed mainly by code inspection, with auditors working independently or together, looking for possible exploits.

Tests were written where possible to validate the issues found.

Static analysis tools were also used, and summary results are attached to this report. These tools are seen as an adjunct to code inspection.

3 | System Overview

The **Yeet project** is deployed on **Berachain**, a decentralized blockchain that is EVM-identical, meaning it can immediately adopt the latest updates from the **Ethereum Virtual Machine (EVM)** ecosystem. Unlike traditional Proof of Stake (PoS) chains, **Berachain** utilizes a **Proof of Liquidity (PoL)** consensus mechanism to overcome liquidity fragmentation and improve ecosystem engagement. The native assets on Berachain are:

- **\$BERA** used for gas fees and network security,
- **\$BGT** used for governance and earned through economic activities,
- **\$HONEY** the stablecoin

Yeet Game

The **Yeet** game follows a two-phase model: the **Bootstrap Phase** and the **Game Phase**.

- **Bootstrap Phase:** Users can buy up to **100 lottery tickets** each at a fixed price of **0.1 BERA**. During this phase, the prize pool for the current round begins to grow.
- **Game Phase:** Players can make bets ("yeets") by contributing **BERA** to the pool. The highest bettor wins **80% of the pool** (claimable later), while the remaining **20%** is allocated as a lottery pool. A key constraint of the game is that users must place a bet worth at least **0.5% of the pool's value** within **one hour** of the last bet, or the most recent bettor wins the pool. The round's duration is adjustable by the admin multisig.
- **Lottery:** A random selection of **10 lottery tickets** wins the remaining **20% of the pool**, divided equally.
- **Taxing Mechanism:** Every bet incurs a **9% tax**, where:
 - **9%** goes to **\$YEET** stakes via the Liquidity Trifecta Vault
 - **1%** contributes to **Public Good Funding** on **BeraChain**
 - **5%** goes to the **Yeet Treasury** as protocol revenue
- **\$YEET Rewards:** Participants in both the **Bootstrap** and **Game phases** can earn **\$YEET tokens** as rewards, regardless of winning the pool. A fixed amount of **\$YEET** is emitted daily, which is distributed according to the **Yeeting Rewards** model. Additionally, players can boost their **\$YEET** earnings by holding **Yeetard NFTs**.

Yeet Stake

The **Yeet Stake** system allows users to stake their **\$YEET** tokens to earn **\$BERA** rewards in real time. This staking occurs through a **claim transaction**. A portion of the **\$YEET** tokens distributed through the Yeet game are staked into **STAKEv2** automatically.

- **Unstaking:** Users can unstake their **\$YEET** tokens at any time, but this comes with certain conditions:
 - When unstaking, users will not accrue additional **\$BERA** rewards until the process completes.
 - The unstake fee starts at **50%** of the principal amount and decreases to **0%** over a period of **10 days**, incentivizing users to keep their stake active.
 - If users want to expedite the unstaking process, they can perform a **RageQuit**, but this will incur a **fee**, burning their **principal amount** staked.

Zapper

The **Zapper** contract acts as an interface for users to interact with the **Kodiak** liquidity pools. This contract facilitates the following functions:

- **Zap In:** The user can swap native **BERA** (or **wBERA**) for **\$YEET**, add liquidity to **Kodiak pools**, and deposit the liquidity tokens into the **Beradrome farm** through the `_depositIntoVault()` function.
- **Zap Out:** Users can redeem liquidity from the Kodiak vault, unstake their tokens, and swap them back for native assets.

The **Zapper** contract ensures that users can efficiently interact with the **Kodiak vault** and liquidity pools, maximizing their exposure to **\$YEET** staking opportunities.

MoneyBrinter

The **MoneyBrinter** contract manages the interaction between the **Kodiak Island tokens** and the **Beradrome Farm**.

- **Deposit Mechanism:** When **Kodiak Island tokens** are deposited into **MoneyBrinter**, they are actually sent to the **Beradrome Farm** instead of the MoneyBrinter's own ERC4626 vault. The function `_depositIntoFarm()` is specifically designed to facilitate this.
- **Redeem Function:** When a user redeems shares in MoneyBrinter through **Zapper._withdrawFromVault()**, the process calls **MoneyBrinter._withdraw()**, which in turn calls `_withdrawFromFarm()` to retrieve the **Kodiak tokens** from Beradrome.
- **Earning xKDK:** As a result of these deposits, MoneyBrinter earns **xKDK**, a token that represents additional **Kodiak** tokens that must be redeemed separately.

Other Contracts

The remaining contracts, including **Yeet.sol**, **YeetGameSettings.sol**, **YeetToken.sol**, **Yeetback.sol**, **Reward.sol**, and **RewardSettings.sol**, were also considered in the scope of the audit. However, our primary focus was on the **main contracts** governing the core functionality of the **Yeet** game, staking, and liquidity interaction, as detailed above.

4 | Findings

4.1 | [CRITICAL] 'accumulatedDeptRewardsYeet()' accounts for tokens under unstaking process

■ **Location(s):** StakeV2.sol#148

■ **Description:** The function `accumulatedDeptRewardsYeet()` computes the surplus in \$YEET token to be later distributed with `executeRewardDistributionYeet()` as the difference between the contract \$YEET balance and `totalSupply`, namely the sum of the \$YEET current staked in StakeV2 contract.

```
function accumulatedDeptRewardsYeet() public view returns (uint256) {  
    return stakingToken.balanceOf(address(this)) - totalSupply;  
}
```

However if a user calls `startUnstake()` when he wants to start the unstaking process, his principal amount is removed from `totalSupply` straight away before the vesting period ends but that amount is still in the contract \$YEET balance.

This means that `accumulatedDeptRewardsYeet()` returns an higher value than it should be and will assign it to `accRevToken0`. Then `executeRewardDistributionYeet()` may distribute part of user funds that should be returned to them after the vesting ends.

Under the circumstance that a manager calls `executeRewardDistributionYeet()` with incorrect parameters but which pass validations, both `rageQuit()` and `unstake()` will be unable to transfer back user stakes due to a lack of funds until someone (treasury or dev team) refunds stakeV2 of the necessary \$YEET.

A test is provided for that scenario. Run with `forge test --match-test test_issue8 -vvv`:

```
// SPDX-License-Identifier: MIT  
pragma solidity ^0.8.19;  
  
import "forge-std/Test.sol";  
import "forge-std/console2.sol";  
import "../src/StakeV2.sol";  
import {MockERC20} from "../mocks/MockERC20.sol";  
import {MockWETH} from "../mocks/MockWBERA.sol";  
import {IERC20} from "@openzeppelin/contracts/token/ERC20/IERC20.sol";  
import "../mocks/SimpleZapperMock.sol";  
  
contract issue8 is Test {  
    StakeV2 public stakeV2;  
    MockERC20 public token;  
    MockWETH public wbera;  
    SimpleZapperMock public mockZapper;  
  
    function setUp() public virtual {  
        token = new MockERC20("MockERC20", "MockERC20", 18);  
        wbera = new MockWETH();  
        address owner = makeAddr("owner");
```

```

    address manager = makeAddr("manager");
    mockZapper = new SimpleZapperMock(token, wbera);
    stakeV2 = new StakeV2(token, mockZapper, owner, manager, IWETH(wbera));
}

function test_issue8() public {
    // stake 50e18 out of the minted 100e18
    address userA = makeAddr("userA");
    token.mint(userA, 100 ether);
    vm.startPrank(userA);
    token.approve(address(stakeV2), 50 ether);
    stakeV2.stake(50 ether);
    vm.stopPrank();

    // fund stakeV2 with some more YEET simulating the scenario
    // of surplus fYEET in StakeV2 caused by zapper imprecisions
    address zapper = makeAddr("zapper");
    token.mint(zapper, 0.1 ether);
    vm.prank(zapper);
    token.transfer(address(stakeV2), 0.1 ether);

    /* asserting that:
       - stakeV2 has 50e18 token in stake
       - stakeV2 has 50.1e18 token in its balance
       - accumulatedDeptRewardsYeet returns 50.1e18 - 50e18 */
    assertEq(stakeV2.totalSupply(), 50 ether);
    assertEq(token.balanceOf(address(stakeV2)), 50.1 ether);
    assertEq(stakeV2.accumulatedDeptRewardsYeet(), 0.1 ether);

    // simulating that userA start the unstake process for 30e18
    // tokens out of 50e18 he deposited
    vm.prank(userA);
    stakeV2.startUnstake(30 ether);

    // asserting that totalSupply was decreased by startUnstake()
    assertEq(stakeV2.totalSupply(), 20 ether);

    // asserting that accumulatedDeptRewardsYeet returns a wrong
    // value rather than 0.1 surplus from zapper
    assertEq(stakeV2.accumulatedDeptRewardsYeet(), 0.1 ether);
}
}

```

- **Recommendation:** Consider tracking the \$YEET tokens that are currently in the unstaking process using a different counter (i.e. unstakingTokens) and change accumulatedDeptRewardsYeet() to something similar to:

```
function accumulatedDeptRewardsYeet_V2() public view returns (uint256) {
```

```
    return stakingToken.balanceOf(address(this)) - totalSupply - unstakingTokens;
}
```

■ **Status:** Unresolved.

4.2 | [MEDIUM] Volume is assigned to wrong epoch if no yeets are made altering the claimable amount

■ **Location(s):** Reward.sol

■ **Description:** The logic within `Reward.sol` allows `addYeetVolume()`, called by the `YeetGame` contract whenever a user makes a new yeet, to check whether an epoch ended or not through `_shouldEndEpoch()` and if it returns true `_endEpoch()` is called. Then `addYeetVolume()` adds the amount to the mappings `userYeetVolume[]` and `totalYeetVolume[]`

```
function addYeetVolume(address user, uint256 amount) external onlyYeetOwner {
    ...
    if (_shouldEndEpoch()) {
        _endEpoch();
    }
    userYeetVolume[currentEpoch][user] += amount;
    totalYeetVolume[currentEpoch] += amount;
}

function _shouldEndEpoch() private view returns (bool) {
    return block.timestamp >= currentEpochEnd;
}

function _endEpoch() private {
    if (currentEpoch % (DISTRIBUTION_CHANGE / EPOCH_LENGTH) == 0) {
        uint256 decayAmount = (epochRewards[currentEpoch] * DECAY_RATE) / SCALE_FACTOR;
        epochRewards[currentEpoch + 1] = epochRewards[currentEpoch] - decayAmount;
    } else {
        epochRewards[currentEpoch + 1] = epochRewards[currentEpoch];
    }

    currentEpoch++;
    currentEpochStart = currentEpochEnd;
    currentEpochEnd = _calculateEpochEnd();
    emit EpochStarted(currentEpoch, currentEpochStart, currentEpochEnd,
        epochRewards[currentEpoch]);
}
```

However a problem may arise when no yeets are made for the duration of a single epoch, which is set to be 1 day = 86400 seconds: in that case amount will be added to an incorrect epoch in the mappings `userYeetVolume[]` and `totalYeetVolume[]`.

Suppose the following:

- Epoch 10, starts on Sunday 30 March 2025 00:00:00 GMT, namely the start timestamp is 1743292800. It will end on Sunday 31 March 2025 00:00:00 GMT, namely at 1743379200.

- Epoch 11, starts at 1743379200 and ends at 1743465600
- Epoch 12, starts at 1743465600 and ends at 1743552000

Scenario:

- Current `block.timestamp` is 174333568 which corresponds to Sunday 30 March 2025 11:19:28, namely we are on epoch 10.
- UserA yeets with `amount = 1e18` and `addYeetVolume()` changes as it follows


```
userYeetVolume[10][userA_address] = 1e18;
totalYeetVolume[10] = 1e18;
```
- No yeets are made for all the remaining duration of epoch 10 and for the whole duration of epoch 11.
- Current `block.timestamp` is now 1743477096, namely it's epoch 12.
- UserB yeets with `amount = 3e18` and `addYeetVolume()` triggers `_endEpoch()` which will set


```
currentEpoch = 11 // instead of 12
currentEpochStart = 1743379200
currentEpochEnd = 1743465600 // when current timestamp it's already greater than this
```
- Then `addYeetVolume()` will set


```
userYeetVolume[11][userB_address] = 3e18;
totalYeetVolume[11] = 3e18;
```

Which is incorrect because that amount should be added to epoch 12 instead.
- Due to previous wrong assignment, `getClaimableAmount()` will return incorrect values since `totalVolume` and `userVolume` read from the affected mappings.

■ **Recommendation:** Consider addressing the scenario where no yeets are made for the duration of a whole epoch. Furthermore, consider ensuring that yeets amount are added by the YeetGame to the correct epoch.

■ **Status:** Unresolved.

4.3 | [LOW] Setting a low value for 'POT_DIVISION' helps whales to win the current round

■ **Location(s):** YeetGameSettings.sol#125

■ **Description:** In YeetGameSettings the variable `POT_DIVISION` is set through `setYeetSettings()` to be a value in the range from 10 to 1000.

```
function setYeetSettings(
    uint256 _yeetTimeSeconds,
    uint256 _potDivision,
    uint256 _taxPerYeet,
    uint256 _taxToStakers,
    uint256 _taxToPublicGoods,
    uint256 _taxToTreasury,
    uint256 _yeetbackPercentage,
    uint256 _cooldownTime,
    uint256 _bootstrapPhaseDuration,
    uint256 _minimumYeetPoint
) external onlyOwner {
    ...
}
```

```
require(_potDivision >= 10, "YeetGameSettings: potDivision must be greater than 10");
// 10% of the pot
require(_potDivision <= 1000, "YeetGameSettings: potDivision must be less than 1000");
// 0.1% of the pot
...
POT_DIVISION = _potDivision;
}
```

However setting a low value for `YeetGameSettings.POT_DIVISION` increases the chances that a whale jumps in to the current round and collects the pot every time because the minimum yeet size the next user needs to place would consist of a very large amount.

Indeed imagine the following scenario:

- ❑ current round has 100 BERA pot
- ❑ a whale spots a low value for `POT_DIVISION` for the current round and jumps into it. For example suppose `POT_DIVISION = 10`
- ❑ the whale yeet a huge amount of BERA like 166 000 BERA (around 1 million USD when writing)
- ❑ `minimumYeetPoint` for the next yeeter will be 16600 BERA (around 100k USD when writing)
- ❑ whale can with high likelihood be the last yeeter and win the round collecting all the pot.

- **Recommendation:** Consider increasing the minimum value when setting `POT_DIVISION` in `YeetGameSettings`
- **Status:** Unresolved.

4.4 | [LOW] Blacklisting a Kodiak vault unintentionally whitelists a previously blacklisted token

- **Location(s):** `Zapper.sol#367`, `Zapper.sol#377`, `Zapper.sol#401`
- **Description:** In the `Zapper.sol` contract the methods `updateSwappableTokens()` and `updateWhitelistedKodiakVault()` helps the blacklisting token and vault processes.

```
function updateSwappableTokens(address token, bool isWhitelisted) external
    override onlyOwner {
    _updateWhitelistedTokens(token, isWhitelisted);
}

function _updateWhitelistedTokens(address token, bool isWhitelisted) private {
    require(token != address(0), "Zapper: token is zero address");
    whitelistedTokens[token] = isWhitelisted;
    emit TokenWhitelisted(token, isWhitelisted);
}

function updateWhitelistedKodiakVault(address vault, bool isEnabled) external
    override onlyOwner {
    ... // other code
    whitelistedKodiakVaults[vault] = isEnabled;
    ... // other code
    if (!whitelistedTokens[token0]) {
```

```

        _updateWhitelistedTokens(token0, true);
    }
    if (!whitelistedTokens[token1]) {
        _updateWhitelistedTokens(token1, true);
    }
}

```

However they both edit the state of the mapping `_updateWhitelistedTokens[]` which can lead to a weird scenario where blacklisting a vault which involves a blacklisted token has the contrary effect to whitelist it again in `_updateWhitelistedTokens[]`.

Suppose the following scenario:

- A token named `$BAD.TOKEN` is blacklisted by the owner using the following code

```
updateSwappableTokens(bad_token_address, false)
```

This, through `_updateWhitelistedTokens()` private function

```
sets whitelistedTokens[bad_token_address] = false
```

- At this point `zapIn` and `zapOut` methods are still usable on vaults involving `$BAD.TOKEN` since the `onlyWhitelistedKodiakVaults()` modifier only checks for whitelisted vaults in the mapping `whitelistedKodiakVaults[]`
- For this reason the owner proceeds to blacklist the first vault involving `$BAD.TOKEN`, for example `wBERA/BAD.TOKEN` vault with

```
updateWhitelistedKodiakVault(wBera_BadToken_vault, false)
```

This function call sets `whitelistedKodiakVaults[wBera_BadToken_vault] = false` but at the same time whitelists again `bad_token_address` inside `whitelistedTokens[]` which was previously blacklisted due to the how `updateWhitelistedKodiakVault()` is implemented

- Additionally the owner needs some time to blacklist all the other vaults involving `$BAD.TOKEN` for example: `HONEY/BAD.TOKEN`, `USDT/BAD.TOKEN`, `USDC/BAD.TOKEN` and thus, in the meanwhile, those vaults are still being usable by users and every potential blacklisting call to `updateSwappableTokens()` is useless due to next `updateWhitelistedKodiakVault()` call.

- **Recommendation:** Consider changing blacklisting logic to ensure that if a token is blacklisted through `updateSwappableTokens()` it will not be whitelisted again in `updateWhitelistedKodiakVault()` when blacklisting one of the vaults involving it

- **Status:** Unresolved.

4.5 | [LOW] Arbitrary \$BERA sent to contracts are locked

- **Location(s):** StakeV2.sol, Zapper.sol

- **Description:** If arbitrary \$BERA are sent to StakeV2 contract through `fallback()` they are locked in the contract since there is no way for the contract owner or managers to retrieve them. Indeed they are not accounted in `accumulatedRewards` to be given out as rewards as done for \$BERA and \$wBERA in `depositReward()`.

```
fallback() external payable {}
```

Same happens in `Zapper.sol` due to

```
receive() external payable {}
```

- **Recommendation:** Consider adding a way to move out arbitrary \$BERA sent to mentioned contracts.
- **Status:** Unresolved.

4.6 | [LOW] DOS protection may be circumvented through multiple identities

- **Location(s):** StakeV2.sol#249
- **Description:** Within `startUnstake()` the implemented DOS protection ensures that an user cannot initiate more than `STAKING_LIMITS = 30` unstake processes.

```
function startUnstake(uint256 unStakeAmount) external {
    ...
    require(stakedTimes[msg.sender] < STAKING_LIMIT, "Amount must be less then
        the STAKING_LIMIT constant");
    // DOS protection https://github.com/Enigma-Dark/Yeet/issues/12
    ...
    stakedTimes[msg.sender]++;
}
```

However the same user can easily create multiple accounts and split the stake amount in several smaller stakes. At this point capping to 30 unstake processes at a time will be effective per account exclusively.

- **Recommendation:** Consider refactoring the DOS protection mechanism
- **Status:** Unresolved.

4.7 | [INFO] NFT Boost Lookup values not adhere to docs

- **Location(s):** Yeet.sol#209
- **Description:** A mismatch exists between the values in `NFTBoostWeight` page of Yeet Documentation and the `nftBoostLookup` array in `Yeet.sol`.

On the docs, the page `NFTBoostWeight`¹ describes the boosts in percentage that a player will receive in \$YEET tokens depending on the number of NFTs. In `Yeet.sol` the array `nftBoostLookup` holds all the boost values from 0 to 1500, which corresponds to 15%, namely the maximum boost a user can receive by holding 25 NFTs. Particularly if a user holds 21 NFTs the boost should be 14.1% while `nftBoostLookup` at the related position 1400 is stored instead of 1410.

- **Recommendation:** Consider changing the docs in order to match the code or vice versa
- **Status:** Unresolved.

4.8 | [INFO] Missing requires in ‘_swapNativeToTokens()’

- **Location(s):** Zapper.sol#632-662
- **Description:** Within `Zapper.sol` the function `_swapNativeToTokens()` swaps `wBERA` into `token0` and `token1` in the scenarios where no one of them is the `wBERA` token itself or just one among `token0` or `token1` is `wBERA`. The value stored in `wBeraDebt` is the converted native BERA to `wBERA` amount and represents how much can be swapped to `token0` or `token1` or both. After the swap it may happen to have some additional `wBERA` left.

¹<https://docs.yeetit.xyz/yeet/yeet-game/nftboostweight>

```
if (address(swapResult.token0) == address(wbera)) {
    swapResult.token1Debt =
        _verifyTokenAndSwap(swapData1, address(wbera),
            address(swapResult.token1), address(this));
    swapResult.token0Debt = swapResult.wBeraDebt - swapData1.inputAmount;
    swapResult.wBeraDebt = 0;
} else if (address(swapResult.token1) == address(wbera)) {
    swapResult.token0Debt =
        _verifyTokenAndSwap(swapData0, address(wbera),
            address(swapResult.token0), address(this));
    swapResult.token1Debt = swapResult.wBeraDebt - swapData0.inputAmount;
    swapResult.wBeraDebt = 0;
} else {
    swapResult.token0Debt =
        _verifyTokenAndSwap(swapData0, address(wbera),
            address(swapResult.token0), address(this));
    swapResult.token1Debt =
        _verifyTokenAndSwap(swapData1, address(wbera),
            address(swapResult.token1), address(this));
    swapResult.wBeraDebt -= (swapData0.inputAmount + swapData1.inputAmount);
}
```

However in the first scenario where token0 is already wBERA and so only swapData1.inputAmount have to be converted into token1 it's not checked that it's lower than swapResult.wBeraDebt. Even this not cause issues since the transaction will revert due to underflow it maybe more appropriate to revert with a proper error message.

Same thing happens in the third scenario where token1 is already wBERA. For the third scenario instead since both token0 and token1 are not wBERA they need to be converted into the proper token asset and the sum of swapData0.inputAmount and swapData1.inputAmount should be lower than swapResult.wBeraDebt which is not checked.

- **Recommendation:** Consider reverting with proper error messages if conditions are not met.
- **Status:** Unresolved.

4.9 | [INFO] Wrong function visibility

- **Location(s):** MoneyBrinter.sol, Yeetback.sol
- **Description:** Within MoneyBrinter.sol several functions have a public visibility while they are not used within the contract. Those includes:

- harvestKodiakRewards()
- harvestBeradromeRewards()
- compound()
- setExitFeeBasisPoints()
- setTreasury()
- setZapper()
- setStrategyManager()
- setXKdx()
- setAllocationFlagxKDK()

- ❑ `setKodiakRewards()`
- ❑ `deallocateXKDK()`
- ❑ `initiateRedeem()`
- ❑ `finalizeRedeem()`
- ❑ `updateRedeemRewardsAddress()`
- ❑ `cancelRedeem()`

Within `Yeetback.sol` several functions have a public visibility while they are not used within the contract. Those includes:

- ❑ `claimable()`
- ❑ `claim()`
- ❑ `addYeetback()`
- ❑ `getYeetsInRound()`
- ❑ `addYeetsInRound()`

■ **Recommendation:** Consider changing the visibility of the mentioned functions from `public` to `external`

■ **Status:** Unresolved.

4.10 | [INFO] Misleading error name

■ **Location(s):** `Yeet.sol#604`

■ **Description:** Within `Yeet.sol` the function `_verifyTokenIds()` reverts if the elements of the `tokenIds` array are not in ascending order or elements are duplicated.

```
function _verifyTokenIds(uint256[] memory tokenIds) internal view {
    ...

    for (uint256 i = 0; i < tokenIds.length - 1; i++) {
        if (tokenIds[i] >= tokenIds[i + 1]) {
            revert DuplicateTokenId(tokenIds[i]);
        }
    }
}
```

However if the condition is met the error name only states that the `tokenId` was a duplicated one. For example if `tokenIds = [1,2,3,7,4]` when 7 is compared against 4 the function reverts with `DuplicateTokenId` while the real issue is that `tokenIds` was not in ascending order.

■ **Recommendation:** Consider changing the error name to reflect its behavior.

■ **Status:** Unresolved.

4.11 | [INFO] Not following CEI pattern may lead to cross-contract and read-only reentrancy in the future

■ **Location(s):** `Yeet.sol#496`, `Yeet.sol#503`

■ **Description:** Within `Yeet.sol` the functions `payoutPublicGoods()` and `payoutTreasuryRevenue()` are zeroing the state variables `publicGoodsAmount` and `treasuryRevenueAmount` after the external call performed in `_payout()`

```
function _payout(uint256 amount, address payable recipient, string memory errorMessage)
    private {
        require(amount != 0, errorMessage);
        uint256 payoutAmount = amount;
        (bool success,) = recipient.call{value: payoutAmount}("");
        require(success, "Transfer failed.");
    }

    /// @notice payoutPublicGoods is a function that pays out the public goods tax
    /// to the public goods address
    function payoutPublicGoods() external onlyOwner nonReentrant {
        _payout(publicGoodsAmount, payable(publicGoodsAddress),
            "Yeet: No public goods to pay out");
        emit PublicGoodsPaidOut(publicGoodsAddress, block.timestamp, publicGoodsAmount);
        publicGoodsAmount = 0;
    }

    /// @notice payoutTreasuryRevenue is a function that pays out the treasury
    /// revenue tax to the treasury revenue address
    function payoutTreasuryRevenue() external onlyOwner nonReentrant {
        _payout(treasuryRevenueAmount, payable(treasuryRevenueAddress),
            "Yeet: No Treasury revenue to pay out");
        emit TreasuryRevenuePaidOut(treasuryRevenueAddress, block.timestamp,
            treasuryRevenueAmount);
        treasuryRevenueAmount = 0;
    }
}
```

Although it is true that the recipients of the `call()` within `_payout()` are trusted entities, such as `treasuryRevenueAddress` and `publicGoodsAddress` not following the check-effect-interaction pattern may cause other functions to read an invalid state if those addresses are compromised. Indeed the `nonReentrant` modifier does not prevent cross-contract reentrancy or read-only reentrancy from third contracts reading the state of `treasuryRevenueAddress` and `publicGoodsAddress` through their public getters.

The finding severity is Informational because no attack vector was found, but the solution is so simple that it is worth considering it to prevent any possible future updates of the protocol or the introduction of new contract relying on `treasuryRevenueAddress` and `publicGoodsAddress` values.

- **Recommendation:** Consider zeroing `publicGoodsAmount` and `treasuryRevenueAmount` after the external call performed in `_payout()` in order to follow the check-effect-interaction pattern.
- **Status:** Unresolved.

4.12 | [INFO] 'stakedTimes' and 'STAKING_LIMIT' names and comments are misleading

- **Location(s):** StakeV2.sol#89, StakeV2.sol#91
- **Description:** Within StakeV2 contract, `stakedTimes` mappings stores a counter for each address which increases every time an unstake process is initiated by an user and decreased when the unstake process actually ends in `_unstake()`. It is never used when a stake is created but name and comments does not match this behavior.

Same for the constant `STAKING_LIMIT` which is checked only in `startUnstake()`.

```
function startUnstake(uint256 unStakeAmount) external {
    ...
    require(stakedTimes[msg.sender] < STAKING_LIMIT, "Amount must be less then
        the STAKING_LIMIT constant");
    ...
    stakedTimes[msg.sender]++;
    ...
}

function _unstake(uint256 index) private {
    ...
    stakedTimes[msg.sender]--;
}
```

- **Recommendation:** Consider changing `STAKING_LIMIT` to `UNSTAKING_LIMIT` and `stakedTimes` to `unstakedTimes`.

- **Status:** Unresolved.

4.13 | [INFO] Very small unstakes does not follow linear unlocking scheme when performing a 'rageQuit()'

- **Location(s):** StakeV2.sol#310

- **Description:** Users that have staked \$YEET in StakeV2 contract have the ability to request the immediate unstake of their stake using `rageQuit()`. When computing the `unlockedAmount` and `lockedAmount` in `calculateVesting()` the following formulas are used:

```
uint256 halfAmount = vesting.amount / 2;
uint256 linearAmount = (halfAmount * timePassed) / timeTotal;
unlockedAmount = halfAmount + linearAmount;
lockedAmount = vesting.amount - unlockedAmount;
```

The variable `halfAmount` represent half of the amount that the user requested to unstake, while `timePassed` represent the difference between the current timestamp and the vesting start timestamp and `timeTotal` is the whole vesting duration. When computing `linearAmount` if the numerator is lower than the denominator, it will be zero and the computed `unlockedAmount` will be lower than it should be while `lockedAmount` will be greater than it should be.

This scenario is feasible only when `vesting.amount` is a low value and the user called `rageQuit()` shortly after having started the unstake, namely `timePassed` is also a low value.

- **Recommendation:** Consider introducing a scaling factor within the calculation ensuring that small takes

- **Status:** Unresolved.

5 | Test Coverage

5.1 | Solidity tests

```
forge test

Ran 1 test for test/Reward.Test.sol:Reward_Rewards_CappedPerUserPerEpoch
[PASS] test_shouldCapRewardsAt10Percent() (gas: 180170)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 6.60ms (1.10ms CPU
→ time)

Ran 1 test for test/Yeet.Test.sol:Yeet_getDistribution
[PASS] test_getDistribution() (gas: 32747)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 6.76ms (463.92µs CPU
→ time)

Ran 1 test for test/Yeet.Test.sol:Yeet_MinYeetAmount
[PASS] test_MinYeetAmountIsCorrect() (gas: 382812)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 6.83ms (620.29µs CPU
→ time)

Ran 2 tests for test/Reward.Test.sol:Reward_OneYeetAndClaimNextEpoch
[PASS] test_claimReturnsAmountEarned() (gas: 99564)
[PASS] test_userHasClaimsAfterEndOfEpoch() (gas: 30143)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 6.72ms (263.25µs CPU
→ time)

Ran 3 tests for test/StakeV2.test.sol:StakeV2_Ctor
[PASS] test() (gas: 209)
[PASS] test_setsTheManager() (gas: 16119)
[PASS] test_setsTheStakingToken() (gas: 11515)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 6.61ms (1.03ms CPU
→ time)

Ran 1 test for test/MockEntropy.sol:MockEntropy
[PASS] test() (gas: 165)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 6.45ms (1.03ms CPU
→ time)

Ran 1 test for test/Reward.Test.sol:Reward_StartsAtMidnight
[PASS] test_epochStartsAtMidnight() (gas: 10963)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 329.71µs (18.50µs CPU
→ time)

Ran 1 test for test/Reward.Test.sol:Reward_OneYeetAndMultipleEpochsClaims
[PASS] test_userClaimsAndYeetsInSameEpoch() (gas: 227906)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 544.00µs (209.46µs CPU
→ time)

Ran 1 test for test/mocks/MockNFTContract.sol:MockNFTContract
[PASS] test() (gas: 231)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 154.58µs (32.46µs CPU
→ time)

Ran 3 tests for test/YeetGameSettings.Test.sol:YeetGameSettingsTest
[PASS] test_ChangeYeetSettings() (gas: 57195)
[PASS] test_YeetSettingsDefaults() (gas: 14768)
```

```
[PASS] test_invalid() (gas: 14935)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 248.54µs (91.71µs CPU
→ time)

Ran 2 tests for test/YeetGameSettings.Test.sol:YeetGameSettingsTest_CooldownTime
[PASS] test_COOLDOWN_TIME_in_range() (gas: 84209)
[PASS] test_COOLDOWN_TIME_toHigh() (gas: 15952)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 244.00µs (97.50µs CPU
→ time)

Ran 2 tests for test/Reward.Test.sol:Reward_OneYeetAndMultipleEpochsClaimsAvailable
[PASS] test_userHasClaimsAfterEndOfTwoEpoch() (gas: 39463)
[PASS] test_userHasClaimsThatAccumulateAfterEndOfTwoEpoch() (gas: 217938)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 707.88µs (375.04µs CPU
→ time)

Ran 3 tests for test/YeetGameSettings.Test.sol:YeetGameSettingsTest_PotDivision
[PASS] test_POT_DIVISION_in_range() (gas: 65790)
[PASS] test_POT_DIVISION_toHigh() (gas: 17250)
[PASS] test_POT_DIVISION_toLow() (gas: 17201)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 247.46µs (107.25µs CPU
→ time)

Ran 2 tests for test/Yeet.Test.sol:Yeet_NFTBoost
[PASS] test_NFTBoost_SameIds() (gas: 65100)
[PASS] test_NFTBoost_ToManyIds() (gas: 80169)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 1.13ms (117.92µs CPU
→ time)

Ran 3 tests for test/StakeV2.test.sol:StakeV2_DepositRewards
[PASS] test() (gas: 209)
[PASS] test_depositRewards() (gas: 42051)
[PASS] test_depositRewardsEmit() (gas: 40373)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 1.89ms (1.35ms CPU
→ time)

Ran 9 tests for test/YeetGameSettings.Test.sol:YeetGameSettingsTest_TaxSettings
[PASS] test_TAX_SETTINGS_in_range() (gas: 59249)
[PASS] test_TAX_SETTINGS_sum_notEqual100() (gas: 15892)
[PASS] test_TAX_SETTINGS_taxPerYeet_max() (gas: 15327)
[PASS] test_TAX_SETTINGS_taxPerYeet_min() (gas: 15235)
[PASS] test_TAX_SETTINGS_taxToPublicGoods_max() (gas: 15453)
[PASS] test_TAX_SETTINGS_taxToStakers_max() (gas: 15380)
[PASS] test_TAX_SETTINGS_taxToStakers_min() (gas: 15353)
[PASS] test_TAX_SETTINGS_toHigh() (gas: 15349)
[PASS] test_TAX_SETTINGS_toLow() (gas: 15352)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 440.63µs (263.29µs CPU
→ time)

Ran 4 tests for test/Reward.Test.sol:Reward_PunshItChewie
[PASS] test_userCantJumpCurrentEpoch() (gas: 14795)
[PASS] test_userCantJumpToFuture() (gas: 14796)
[PASS] test_userCantJumpToPast() (gas: 101720)
[PASS] test_userSkipsReward() (gas: 66910)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 680.13µs (161.79µs CPU
→ time)

Ran 4 tests for test/YeetGameSettings.Test.sol:YeetGameSettingsTest_YeetTimeSeconds
```

```
[PASS] test_MINIMUM_YEET_TIME_SECONDS_in_range() (gas: 65723)
[PASS] test_MINIMUM_YEET_TIME_SECONDS_toHigh() (gas: 17038)
[PASS] test_MINIMUM_YEET_TIME_SECONDS_toLow30() (gas: 17090)
[PASS] test_MINIMUM_YEET_TIME_SECONDS_toLow59() (gas: 17089)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 274.92µs (130.71µs CPU
→ time)
```

```
Ran 3 tests for
→ test/RewardSettings.Test.sol:YeetGameSettingsTest_MaxCapPerWalletPerEpochFactor
[PASS] test_MAX_CAP_PER_WALLET_PER_EPOCH_FACTOR_in_range() (gas: 17390)
[PASS] test_MAX_CAP_PER_WALLET_PER_EPOCH_FACTOR_toHigh() (gas: 11683)
[PASS] test_MAX_CAP_PER_WALLET_PER_EPOCH_FACTOR_toLow() (gas: 11634)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 147.79µs (47.46µs CPU
→ time)
```

```
Ran 7 tests for test/StakeV2.test.sol:StakeV2_ExecuteRewardsDistrubution
[PASS] test() (gas: 231)
[PASS] test_accumulatedRewardsToken0() (gas: 138603)
[PASS] test_executeRewardDistribution_noRewards() (gas: 22626)
[PASS] test_executeRewardDistribution_onlyManager() (gas: 16053)
[PASS] test_executeRewardDistribution_withRewards() (gas: 271634)
[PASS] test_executeRewardDistribution_zeroTotalSupply() (gas: 110590)
[PASS] test_handleExcessToken1DepositWBERA() (gas: 113688)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 1.08ms (528.67µs CPU
→ time)
```

```
Ran 4 tests for test/NFTVesting.Test.sol:NFTVestingConstructorTest
[PASS] test() (gas: 209)
[PASS] test_SetsTheConstructorValues() (gas: 24251)
[PASS] test_TokenAmountPerNft() (gas: 8779793)
[PASS] test_throwsErrorIfBothBlacklistAndAllowList() (gas: 1592442)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 10.03ms (4.41ms CPU
→ time)
```

```
Ran 3 tests for test/StakeV2.test.sol:StakeV2_ClaimNativeRewards
[PASS] test() (gas: 209)
[PASS] test_claimNativeRewards_noRewards() (gas: 28729)
[PASS] test_claimNativeRewards_withRewards() (gas: 324330)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 841.96µs (272.71µs CPU
→ time)
```

```
Ran 2 tests for test/StakeV2.test.sol:StakeV2_HandleExcessDebt
[PASS] test_handleExcessYeetToken0() (gas: 5468365)
[PASS] test_handleExcessYeetToken1() (gas: 5468344)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 1.02ms (674.08µs CPU
→ time)
```

```
Ran 3 tests for test/StakeV2.test.sol:StakeV2_Manager
[PASS] test_addManager() (gas: 35979)
[PASS] test_onlyOwner() (gas: 12443)
[PASS] test_removeManager() (gas: 29973)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 194.13µs (82.96µs CPU
→ time)
```

```
Ran 1 test for test/Yeet.Test.sol:Yeet_NFTBoost_WorksWhenContractIsEmpty
[PASS] test_NFTBoostWhenAddressIsNull() (gas: 15286)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 2.79ms (702.71µs CPU
→ time)
```

```

Ran 4 tests for test/Reward.Test.sol:Reward_OneYeet
[PASS] test_AddVolumeToTotalYeetVolume() (gas: 11611)
[PASS] test_AddVolumeToUserYeetVolume() (gas: 11969)
[PASS] test_callingClaimWithNotingToClaimReverts() (gas: 13459)
[PASS] test_userHasZeroClaimsBeforeEndOfEpoch() (gas: 13945)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 1.47ms (71.21µs CPU
→ time)

Ran 2 tests for test/RewardSettings.Test.sol:RewardSettingsTest
[PASS] test_ChangeYeetRewardSettings() (gas: 18714)
[PASS] test_YeetRewardSettingsDefaults() (gas: 10983)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 227.75µs (32.17µs CPU
→ time)

Ran 16 tests for test/vault/VaultAdminTest.t.sol:VaultAdminTest
[PASS] testSetAllocationFlagxKDK_NotAdmin() (gas: 14764)
[PASS] testSetAllocationFlagxKDK_Valid_Admin() (gas: 24843)
[PASS] testSetExitFeeBasisPoints_NotAdmin() (gas: 14788)
[PASS] testSetExitFeeBasisPoints_Valid_Admin_Exceeds_Maximum_Fee() (gas: 15550)
[PASS] testSetExitFeeBasisPoints_Valid_Admin_ValidValue() (gas: 38782)
[PASS] testSetStrategyManager_NotAdmin() (gas: 15250)
[PASS] testSetStrategyManager_Valid_Admin() (gas: 33123)
[PASS] testSetTreasury_NotAdmin() (gas: 14837)
[PASS] testSetTreasury_Valid_Admin_ValidAddress() (gas: 22337)
[PASS] testSetTreasury_Valid_Admin_ZeroAddress() (gas: 14388)
[PASS] testSetXKdk_NotAdmin() (gas: 14884)
[PASS] testSetXKdk_Valid_Admin_ValidAddress() (gas: 39504)
[PASS] testSetXKdk_Valid_Admin_ZeroAddress() (gas: 14368)
[PASS] testSetZapper_NotAdmin() (gas: 14840)
[PASS] testSetZapper_Valid_Admin_ValidAddress() (gas: 39678)
[PASS] testSetZapper_Valid_Admin_ZeroAddress() (gas: 14370)
Suite result: ok. 16 passed; 0 failed; 0 skipped; finished in 900.79µs (347.29µs
→ CPU time)

Ran 3 tests for test/NFTVesting.Test.sol:NFTVestingAllowlist
[PASS] test_addAllowedToken() (gas: 20422)
[PASS] test_canMintAllowList() (gas: 144298)
[PASS] test_canNotMintAllowList() (gas: 15988)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 5.00ms (143.29µs CPU
→ time)

Ran 3 tests for test/Yeet.Test.sol:Yeet_PublicGoods
[PASS] test_CanUpdatePublicGoodsAddress() (gas: 21160)
[PASS] test_PayoutPublicGoods() (gas: 53442)
[PASS] test_PublicGoodsTax() (gas: 11051)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 1.16ms (61.17µs CPU
→ time)

Ran 3 tests for test/NFTVesting.Test.sol:NFTVestingBlacklist
[PASS] test_addBlacklistedToken() (gas: 20533)
[PASS] test_canMintNotBlacklistedToken() (gas: 81276)
[PASS] test_canNotMintBlacklistedToken() (gas: 14706)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 512.08µs (98.46µs CPU
→ time)

Ran 1 test for test/Yeet.Test.sol:Yeet_TestTaxes
[PASS] test_CorrectTax() (gas: 32790)

```

```

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 1.06ms (43.58µs CPU
→ time)

Ran 3 tests for test/Yeet.Test.sol:Yeet_TeamRevenue
[PASS] test_CanUpdateTeamAddress() (gas: 21226)
[PASS] test_PayoutTeamRevenue() (gas: 53412)
[PASS] test_TeamTax() (gas: 10987)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 1.07ms (60.08µs CPU
→ time)

Ran 10 tests for test/NFTVesting.Test.sol:NFTVestingClaimTest
[PASS] test() (gas: 187)
[PASS] test_ClaimAfterVestingPeriod() (gas: 79176)
[PASS] test_ClaimTwice() (gas: 94910)
[PASS] test_ClaimTwiceWillPayoutToOwner() (gas: 96839)
[PASS] test_ClaimWillPayoutToOwner() (gas: 83410)
[PASS] test_ClaimWillPayoutToOwnerForOtherAddress() (gas: 17275)
[PASS] test_ClaimedAllAfterVestingPeriod() (gas: 79164)
[PASS] test_ClaimedHalfAfterHalfVestingPeriod() (gas: 79100)
[PASS] test_ClaimedIsZero() (gas: 12044)
[PASS] test_ClaimedIsZeroForOtherAddress() (gas: 12066)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 795.83µs (460.88µs
→ CPU time)

Ran 6 tests for test/NFTVesting.Test.sol:NFTVestingClaimableTest
[PASS] test() (gas: 231)
[PASS] test_ClaimableAllAfterVestingPeriod() (gas: 13676)
[PASS] test_ClaimableHalfAfterHalfVestingPeriod() (gas: 13654)
[PASS] test_ClaimableIsZero() (gas: 12525)
[PASS] test_ClaimableMany() (gas: 130318)
[PASS] test_ClaimableTwice() (gas: 18107)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 543.42µs (240.29µs CPU
→ time)

Ran 2 tests for test/Yeet.Test.sol:Yeet_Yeetback
[PASS] test_AddYeetbackRevertsIfFeeIsNotPaid() (gas: 46777)
[PASS] test_addYeetbackToMuchFee() (gas: 310795)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 1.19ms (175.71µs CPU
→ time)

Ran 2 tests for test/StakeV2.test.sol:StakeV2_Reward
[PASS] test() (gas: 209)
[PASS] test_rewardIndex() (gas: 476249)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 4.73ms (4.21ms CPU
→ time)

Ran 3 tests for test/vault/VaultWithdrawTest.t.sol:VaultRedeemTest
[PASS] testRedeemInsufficientBalance() (gas: 18528)
[PASS] testRedeemSuccessNoFee() (gas: 106960)
[PASS] testRedeemSuccessWithFee() (gas: 172143)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 855.63µs (258.92µs CPU
→ time)

Ran 3 tests for test/vault/VaultWithdrawTest.t.sol:VaultWithdrawTest
[PASS] testWithdrawInsufficientBalance() (gas: 39175)
[PASS] testWithdrawSuccessNoFee() (gas: 110373)
[PASS] testWithdrawSuccessWithFee() (gas: 175532)

```

Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 878.79µs (301.88µs CPU
→ time)

Ran 4 tests for test/Yeetback.Test.sol:YeetbackTest

[PASS] test_AddYeetsInRound() (gas: 110303)

[PASS] test_addYeetback() (gas: 24942762)

[PASS] test_draftSameWinnerMultipleTimes() (gas: 347915)

[PASS] test_draftWinner() (gas: 25276213)

Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 21.68ms (31.10ms CPU
→ time)

Ran 2 tests for test/Yeet.Test.sol:Yeet_Claim

[PASS] test_claimEmitCorrectEvent() (gas: 549953)

[PASS] test_claimEmitCorrectEventsComplicated() (gas: 13130410)

Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 20.36ms (17.00ms CPU
→ time)

Ran 1 test for test/Reward.Test.sol:Reward_clawback

[PASS] test_clawback() (gas: 114836829)

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 43.06ms (42.77ms CPU
→ time)

Ran 1 test for test/Reward.Test.sol:Reward_runFor208weeks

[PASS] test_runFor208weeks() (gas: 115292125)

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 60.14ms (54.60ms CPU
→ time)

Ran 6 tests for test/Yeet.Test.sol:Yeet_BootstrapPhase

[PASS] test_endOfYeetTimeShouldBeTheSameForTheDurationOfTheBootstrapPhase() (gas:
→ 24824542)

[PASS] test_endTimeShoulUpdateWhenPhaseEnds() (gas: 440480)

[PASS] test_restartShouldSetCorrectEndTime() (gas: 555189)

[PASS] test_yeetDoesIncreaseMinYeetPointWhenPhaseEnds() (gas: 24827933)

[PASS] test_yeetDoesNotIncreaseMinYeetPoint() (gas: 24822487)

[PASS] test_yeetDuringBootstrapPhase() (gas: 24821191)

Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 31.32ms (30.16ms CPU
→ time)

Ran 4 tests for test/vault/VaultDepositTest.t.sol:VaultDepositTest

[PASS] testDepositWithInsufficientBalance(uint256) (runs: 256, : 67096, ~: 66514)

[PASS] testDepositWithSufficientBalance(uint256) (runs: 256, : 203065, ~: 202647)

[PASS] testMintWithInsufficientBalance(uint256) (runs: 256, : 42571, ~: 41947)

[PASS] testMintWithSufficientBalance(uint256) (runs: 256, : 203063, ~: 202645)

Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 85.63ms (132.44ms CPU
→ time)

Ran 1 test for test/Yeet.Test.sol:Yeet_ForkTestTaxes

[PASS] test_ForkCorrectTax() (gas: 350634)

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 1.14s (1.23ms CPU
→ time)

Ran 1 test for test/fork.Test.sol:ClaimRewardsTestA

[FAIL: vm.createSelectFork: could not instantiate forked environment with provider

→ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at

→ quicknode.com] setUp() (gas: 0)

Suite result: FAILED. 0 passed; 1 failed; 0 skipped; finished in 1.28s (0.00ns CPU
→ time)

```
Ran 1 test for test/zapper/Zapper_Unit_test.sol:Zapper_Unit_Test
[FAIL: vm.createFork: could not instantiate forked environment with provider
↳ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
↳ quicknode.com] setUp() (gas: 0)
Suite result: FAILED. 0 passed; 1 failed; 0 skipped; finished in 1.29s (0.00ns CPU
↳ time)
```

```
Ran 5 tests for test/NFTVesting.Test.sol:NFTVestingVerifyBlacklistCalculations
[PASS] test_RealValuesWithBlacklist() (gas: 153521579)
[PASS] test_RealValuesWithNoBlacklist() (gas: 147359487)
[PASS] test_WithAllowList() (gas: 5307487)
[PASS] test_WithBlacklist() (gas: 6226481)
[PASS] test_WithoutBlacklist() (gas: 5077595)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 1.36s (123.43ms CPU
↳ time)
```

```
Ran 1 test for
↳ test/zapper/Zapper_Unit_Test_ZapInToken0.t.sol:Zapper_Unit_Test_ZapInToken0
[FAIL: vm.createFork: could not instantiate forked environment with provider
↳ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
↳ quicknode.com] setUp() (gas: 0)
Suite result: FAILED. 0 passed; 1 failed; 0 skipped; finished in 1.28s (0.00ns CPU
↳ time)
```

```
Ran 1 test for test/zapper/ZapIn_Integration_test.t.sol:ZapIn_Integration_Test
[FAIL: vm.createFork: could not instantiate forked environment with provider
↳ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
↳ quicknode.com] setUp() (gas: 0)
Suite result: FAILED. 0 passed; 1 failed; 0 skipped; finished in 1.34s (0.00ns CPU
↳ time)
```

```
Ran 1 test for
↳ test/zapper/Zapper_Unit_Test_ZapInToken1.t.sol:Zapper_Unit_Test_ZapInToken1
[FAIL: vm.createFork: could not instantiate forked environment with provider
↳ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
↳ quicknode.com] setUp() (gas: 0)
Suite result: FAILED. 0 passed; 1 failed; 0 skipped; finished in 1.31s (0.00ns CPU
↳ time)
```

```
Ran 1 test for test/zapper/ZapOut.t.sol:ZapOut
[FAIL: vm.createFork: could not instantiate forked environment with provider
↳ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
↳ quicknode.com] setUp() (gas: 0)
Suite result: FAILED. 0 passed; 1 failed; 0 skipped; finished in 1.29s (0.00ns CPU
↳ time)
```

```
Ran 1 test for
↳ test/vault/Vault_IntegrationTest_ZeroFee.t.sol:Vault_IntegrationTest_ZeroFee
[FAIL: vm.createFork: could not instantiate forked environment with provider
↳ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
↳ quicknode.com] setUp() (gas: 0)
Suite result: FAILED. 0 passed; 1 failed; 0 skipped; finished in 1.37s (0.00ns CPU
↳ time)
```

```
Ran 1 test for test/fork.Test.sol:ClaimRewardsTest
[FAIL: vm.createSelectFork: could not instantiate forked environment with provider
↳ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
↳ quicknode.com] setUp() (gas: 0)
```

Suite result: FAILED. 0 passed; 1 failed; 0 skipped; finished in 1.33s (0.00ns CPU
→ time)

Ran 55 test suites in 1.40s (13.33s CPU time): 152 tests passed, 8 failed, 0
→ skipped (160 total tests)

Failing tests:

Encountered 1 failing test in test/fork.Test.sol:ClaimRewardsTest

[FAIL: vm.createSelectFork: could not instantiate forked environment with provider
→ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
→ quicknode.com] setUp() (gas: 0)

Encountered 1 failing test in test/fork.Test.sol:ClaimRewardsTestA

[FAIL: vm.createSelectFork: could not instantiate forked environment with provider
→ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
→ quicknode.com] setUp() (gas: 0)

Encountered 1 failing test in

→ test/vault/Vault_IntegrationTest_ZeroFee.t.sol:Vault_IntegrationTest_ZeroFee
[FAIL: vm.createFork: could not instantiate forked environment with provider
→ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
→ quicknode.com] setUp() (gas: 0)

Encountered 1 failing test in

→ test/zapper/ZapIn_Integration_test.t.sol:ZapIn_Integration_Test
[FAIL: vm.createFork: could not instantiate forked environment with provider
→ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
→ quicknode.com] setUp() (gas: 0)

Encountered 1 failing test in test/zapper/ZapOut.t.sol:ZapOut

[FAIL: vm.createFork: could not instantiate forked environment with provider
→ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
→ quicknode.com] setUp() (gas: 0)

Encountered 1 failing test in

→ test/zapper/Zapper_Unit_Test_ZapInToken0.t.sol:Zapper_Unit_Test_ZapInToken0
[FAIL: vm.createFork: could not instantiate forked environment with provider
→ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
→ quicknode.com] setUp() (gas: 0)

Encountered 1 failing test in

→ test/zapper/Zapper_Unit_Test_ZapInToken1.t.sol:Zapper_Unit_Test_ZapInToken1
[FAIL: vm.createFork: could not instantiate forked environment with provider
→ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
→ quicknode.com] setUp() (gas: 0)

Encountered 1 failing test in test/zapper/Zapper_Unit_test.sol:Zapper_Unit_Test

[FAIL: vm.createFork: could not instantiate forked environment with provider
→ bartio.rpc.berachain.com; HTTP error 403 with body: endpoint is disabled at
→ quicknode.com] setUp() (gas: 0)

Encountered a total of 8 failing tests, 152 tests succeeded

A | Disclaimers

The audit makes no statements or warranty about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purpose

A.1 | Client Confidentiality

This document contains Client Confidential information and may not be copied without written permission.

A.2 | Proprietary Information

The content of this document should be considered proprietary information. Extropy gives permission to copy this report for the purposes of disseminating information within your organisation or any regulatory agency.

B | Slither results

```

Reentrancy in Reentrance-withdrawBalance) (Reentrancy.sol#14-21) :
  External calls
  (msa.sender •call. value (userBalance [mse :sender ])) (Reentrancy.
    ↳ sol#17)
  State variables the calls):
  - userBalance[imsa (Reentrancy. sol#20)
  Reentrance. UserBalance (Reentrancy. sol#4) can be used in cross function
    ↳ reentrancies:
  Reentrance.addToBalance) (Reentrancev. sol#10-12)
  Reentrance.getBalance(address) (Reentrancy, sol#6-8)
  Reentrance.withdrawBalance) (Reentrancy.sol#14-21)
  Reentrance.withdrawBalance_fixed() (Reentrancy. sol#23-31)
  - Reentrance.wichdrawBalance_fixed_20) (Reentrancy sol#33-40)
References https://pathub.com/erytic/slither/wiki/Detector-Documentation
  #reentrancy-vulnerabilities

Deprecated standard detected THROW (Reentrancy.sol#18):
  - Usage of "throw" should be replaced with "revert"
Deprecated standard detected THROW (Reentrancy. sol#29):
  - Usage of "throw" should be replaced with "revert)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
  #deprecated-standards

Pragma version*0.4.15 (Reentrancy. sol#1) allows old versit
solc-0.4.15 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
  #incorrect-versions-of-solidity

Low level call in Reentrance.withdrawBalance() (Reentrancy. sol#14-21):
  - ! (msg. sender. call. value (userBalance [msg. sender ])) (Reentrancy
    ↳ .sol#17)
Low level call in Reentrance.withdrawBalance fixed) (Reentrancy.sol#23-31):
  - ! (msg. sender. call. value (amount)) (Reentrancy. sol#28)
Reference:
  ↳ https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function Reentrance.withdrawBalance_fixed() (Reentrancy. sol#23-31) is not in
  ↳ mixedCase
Function Reentrance.withdrawBalance_fixed_2() (Reentrancy.sol#33-40) is not in
  ↳ mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
  #conformance-to-solidity-naming-conventions

```

