MasterChef

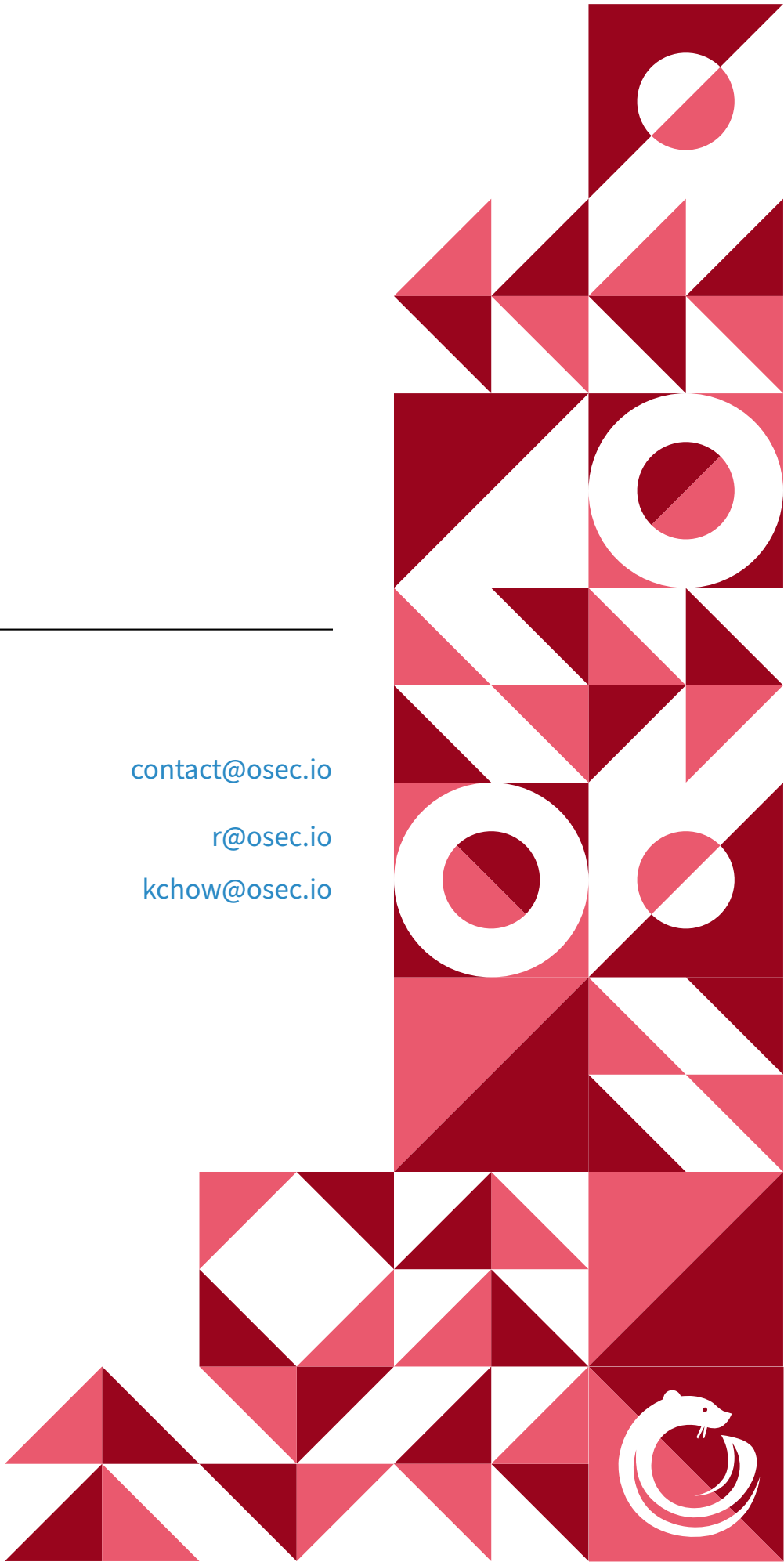
Audit

Presented by:

OtterSec contact@osec.io

Robert Chen r@osec.io

Kevin Chow kchow@osec.io



Contents

Appendices

01 | Executive Summary

Overview

PancakeSwap engaged OtterSec to perform an assessment of the `MasterChef` program. This assessment was conducted between November 8th and November 13th, 2022.

Critical vulnerabilities were communicated to the team prior to the delivery of the report to speed up remediation. After delivering our audit report, we worked closely with the team to streamline patches and confirm remediation. We delivered final confirmation of the patches November 15th, 2022.

Key Findings

Over the course of this audit engagement, we produced 5 findings total.

In particular, there was an issue with reward distribution if pools are not updated with `upkeep` (OS-MAS-ADV-00).

We also made recommendations around clean coding practices and general security recommendations. These recommendations serve to clarify the purpose and logic of functions in the program and can help prevent future security vulnerabilities stemming from misunderstanding or needlessly entangled code.

Overall, the Pancake Swap team was responsive to feedback and great to work with.

02 | Scope

The source code was delivered to us in a git repository at github.com/pancakeswap/aptos-contracts. This audit was performed against commit `430e3e9`.

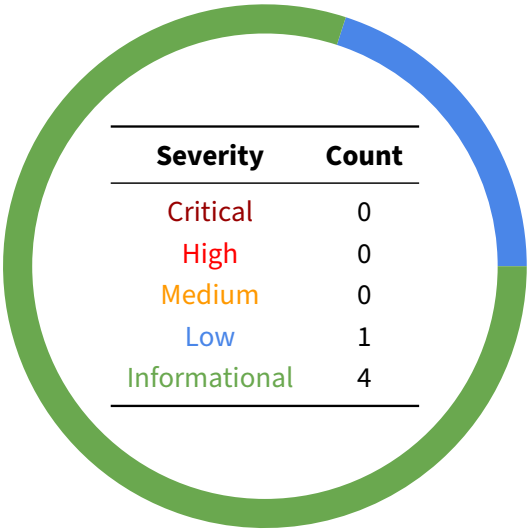
A brief description of the programs is as follows.

Name	Description
MasterChef	A staking contract for providing liquidity and rewarding those providers.

03 | Findings

Overall, we report 5 findings.

We split the findings into **vulnerabilities** and **general findings**. Vulnerabilities have an immediate impact and should be remediated as soon as possible. General findings don't have an immediate impact but will help mitigate future vulnerabilities.



Severity	Count
Critical	0
High	0
Medium	0
Low	1
Informational	4

04 | Vulnerabilities

Here we present a technical analysis of the vulnerabilities we identified during our audit. These vulnerabilities have *immediate* security implications, and we recommend remediation as soon as possible.

Rating criteria can be found in Appendix A.

ID	Severity	Status	Description
OS-MAS-ADV-00	Low	Resolved	Reward distribution can have accounting issues if there is a time gap between pool update and upkeep

OS-MAS-ADV-00 [low] [resolved] | Accounting Issues In Reward Distribution

Description

For accurate reward distribution, pools need to be updated before calling `upkeep`. However, updating is optional with `with_update` to avoid reaching a gas limit. When there is a time gap between the upkeep and pool update functions being called, there can be small accounting issues with not enough rewards.

sources/masterchef.move RUST

```
public entry fun upkeep(
    sender: &signer,
    amount: u64,
    elapsed: u64,
    with_update: bool,
)
```

Proof of Concept

If there is a one second gap in between `upkeep` and `update_pool` being called, more reward than exists is accounted for.

Remediation

We recommend adding a `last_upkeep_timestamp`.

Patch

Reward calculations is now

sources/masterchef.move RUST

```
} else if (current_timestamp <= master_chef.end_timestamp) {
    // if 'mass_update_pools' is ignored on any function which should be
    ↪  called,like 'upkeep',
    // should choose the max timestamp as 'last_reward_timestamp'.
    current_timestamp - max(pool_info.last_reward_timestamp,
    ↪  master_chef.last_upkeep_timestamp)
} else {
    master_chef.end_timestamp - max(pool_info.last_reward_timestamp,
    ↪  master_chef.last_upkeep_timestamp)
};
```

```
if (supply > 0 && total_alloc_point > 0) {
    cake_reward = ((multiplier as u128) * (((master_chef.cake_per_second
    ↪  as u128) * (cake_rate as u128) * (pool_info.alloc_point as u128))
    ↪  / (total_alloc_point as u128))) / (TOTAL_CAKE_RATE_PRECISION as
    ↪  u128);
    acc_cake_per_share = (pool_info.acc_cake_per_share) + (cake_reward *
    ↪  ACC_CAKE_PRECISION) / supply;
```

05 | General Findings

Here we present a discussion of general findings during our audit. While these findings do not present an immediate security impact, they represent antipatterns and could lead to security issues in the future.

ID	Description
OS-MAS-SUG-00	Validate asset from pool table instead of helper function
OS-MAS-SUG-01	Cake transfer should abort on violation.
OS-MAS-SUG-02	Upkeep admin can set `end_timestamp` to be `U64_MAX` and therefore rewards to zero.
OS-MAS-SUG-03	`set_pool` mutates pool reward allocations, but the order of operations can be changed to avoid overflow

OS-MAS-SUG-00 | Validate From LP Table Instead Of Helper Function

Description

Both `deposit` and `withdraw` use the function `is_valid_lp` to validate that a token exists. We recommend validating lp against `master_chef_mut.lp_to_pid` instead.

sources/masterchef.move RUST

```
assert!(is_valid_lp<CoinType>(master_chef_mut, pid),
    ↪  ERROR_INVALID_LP_TOKEN);
```

Remediation

sources/masterchef.move RUST

```
assert!(Table::contains<string::String, u64>(&master_chef.lp_to_pid,
    ↪  type_info::type_name<CoinType>()), ERROR_INVALID_LP_TOKEN);
```

OS-MAS-SUG-01 | CAKE Transfer Should Abort On Violation

Description

During CAKE distribution, `safe_transfer_cake` checks the `balance` of Cake is greater than or equal to the amount withdrawn. If this condition is violated, we recommend aborting, rather than silently lowering the amount to the balance.

some/source_masterchef.move RUST

```
fun safe_transfer_cake(
        resource_signer: &signer,
        to: address, amount: u64
    ) {
        if (amount > 0) {
            let balance = coin::balance<Cake>(RESOURCE_ACCOUNT);
            if (balance < amount) {
                amount = balance;
            };
            CAKE::transfer(resource_signer, to, amount);
        }
    }
```

Remediation

PancakeSwap acknowledged that this condition should never be violated.

OS-MAS-SUG-02 | Safeguard Upkeep Footgun Over Changing MasterChef Life

Description

An upkeep admin can set `elapsed` and therefore `end_timestamp` of MasterChef to `U64_MAX`. This would cause `cake_per_second` and reward distribution to be zero.

Remediation

Enforce a reasonable limit to elapsed, and/or allow admin to update `end_timestamp`.

sources/masterchef.move RUST

```
public entry fun upkeep(
        sender: &signer,
        amount: u64,
        elapsed: u64,
        with_update: bool,
    ) acquires MasterChef, Events {
```

sources/masterchef.move RUST

```
let new_cake_per_second = new_available_cake / (new_end_timestamp -
    ↪  current_timestamp);
```

OS-MAS-SUG-03 | Change Order Of Operations In Set Pool

Description

The function `set_pool` mutates the total allocation points, but the order can be optimized to prevent overflow.

sources/masterchef.move RUST

```
if (pool_info.is_regular) {
    master_chef_mut.total_regular_alloc_point =
     ↪  master_chef_mut.total_regular_alloc_point + alloc_point -
     ↪  pool_info.alloc_point ;
} else {
    master_chef_mut.total_special_alloc_point =
     ↪  master_chef_mut.total_special_alloc_point + alloc_point -
     ↪  pool_info.alloc_point;
};
```

Remediation

The order of operations was updated to subtract before adding.

sources/masterchef.move RUST

```
if (pool_info.is_regular) {
    master_chef_mut.total_regular_alloc_point =
     ↪  master_chef_mut.total_regular_alloc_point - pool_info.alloc_point
     ↪  + alloc_point;
} else {
    master_chef_mut.total_special_alloc_point =
     ↪  master_chef_mut.total_special_alloc_point - pool_info.alloc_point
     ↪  + alloc_point;
};
```

A | Vulnerability Rating Scale

We rated our findings according to the following scale. Vulnerabilities have immediate security implications. Informational findings can be found in the General Findings section.

Critical	Vulnerabilities that immediately lead to loss of user funds with minimal preconditions Examples: • Misconfigured authority or access control validation • Improperly designed economic incentives leading to loss of funds
High	Vulnerabilities that could lead to loss of user funds but are potentially difficult to exploit. Examples: • Loss of funds requiring specific victim interactions • Exploitation involving high capital requirement with respect to payout
Medium	Vulnerabilities that could lead to denial of service scenarios or degraded usability. Examples: • Malicious input that causes computational limit exhaustion • Forced exceptions in normal user flow
Low	Low probability vulnerabilities which could still be exploitable but require extenuating circumstances or undue risk. Examples: • Oracle manipulation with large capital requirements and multiple transactions
Informational	Best practices to mitigate future security risks. These are classified as general findings. Examples: • Explicit assertion of critical internal invariants • Improved input validation