Mysten Validator

Audit

Presented by:

OtterSec contact@osec.io

Robert Chen r@osec.io

Michal Bochnak embe221ed@osec.io

James Wang james@osec.io

Cauê Obici caue@osec.io

Aleksandre Khokhiashvili khokho@osec.io

Contents

© 2023 Otter Audits LLC. All Rights Reserved.

Appendices

© 2023 Otter Audits LLC. All Rights Reserved.

01 | Executive Summary

Overview

Mysten Labs engaged OtterSec to perform an assessment of the `sui-verifier`, `sui-framework`, `governance`, `sui-josn-rpc`, `sui-adapter`, `collectible`, `kiosk`, `suifrens`, `suins`, and `sui.id` programs. This assessment was conducted between January 9th and May 24th, 2023. We further conducted a review of the `kiosk-extension` API PR between July 28th and August 4th, 2023. For more information on our auditing methodology, see Appendix B.

Key Findings

https://www.overleaf.com/project/645a8ecb7fc0ae05169d45a4

Over the course of this audit engagement, we produced 51 findings in total.

In particular, we reported issues around hash collisions, including one related to dynamic fields (OS-SUI-ADV-00) and another collision during the calculation of the module digest (OS-SUI-ADV-02). We identified some cases of integer overflow resulting in the RPC node failing (OS-SUI-ADV-01), an overflow during the calculation of quorum threshold (OS-SUI-ADV-09), and the possibility of entering unreachable code (OS-SUI-ADV-13). We also identified an issue related to adding the key capability during the upgrade process resulting in leaking the object id, bypassing the id leak verifier (OS-SUI-ADV-03).

We also encountered certain concerns within the SuiFrens mixing functionality, which include a scenario where users may bypass the checks that validate the countdown period before mixing their SuiFrens. This loophole is due to the public visibility of the epoch update functionality, which permits anyone to set an arbitrary countdown period (OS-SUI-ADV-04).

Another issue pertains to a typographical error within the SuiFrens mixing feature, allowing users to mint SuiFrens with an inadequate mixing limit (OS-SUI-ADV-05). Furthermore, during the process of retrieving record names, incorrect values are returned by the function (OS-SUI-ADV-06). We also identified a flaw in the initial version of the contract, involving an inaccurate check for the discount code rate (OS-SUI-ADV-18).

Additionally, the function responsible for withdrawing staked tokens and accrued rewards from staking pools rounds down the value during token calculation. This may result in the withdrawal of zero tokens when withdrawing small amounts of Sui, allowing attackers to exploit this error to prevent users from receiving their rewards OS-SUI-ADV-08), and a further issue regarding the absence of several checks while setting the default domain name (OS-SUI-ADV-19).

© 2023 Otter Audits LLC. All Rights Reserved.

We also made recommendations around replacing panic calls with debug assertions (OS-SUI-SUG-00) and removing the usage of a legacy function in the code used for deleting child objects, removed in a previous update (OS-SUI-SUG-01). We also identified some logic errors (**??**), potential risks in the new smart contracts related to the visibility of public functions (OS-SUI-SUG-16), and an erroneous signature check due to incorrect parameter passed into the signature verifying function (OS-SUI-SUG-02).

Moreover, we identified a discrepancy in the length assertion check in the batch minting and mint functionality (OS-SUI-SUG-03) and a lack of validation for the existence of a transfer policy in the kiosk module (OS-SUI-SUG-04). We also advised utilizing the has access functionality instead of directly checking for owner capability to maintain consistency with the convention followed in other functions while utilizing the same assertion (OS-SUI-SUG-05).

We also highlighted the addition of an assert statement to verify the existence of a rule in the transfer policy (OS-SUI-SUG-06) and advised adding a check for validating the existence of items in the kiosk before listing the items for purchase (OS-SUI-SUG-07).

© 2023 Otter Audits LLC. All Rights Reserved.

02 | Scope

The source code was delivered to us in a git repository at github.com/MystenLabs/sui, github.com/MystenLabs/suifrens, github.com/SuiNSdapp/sui.id, github.com/SuiNSdapp/SuiNS-C, github.com/SuiNSdapp/SuiNS-FE, and github.com/SuiNSdapp/SuiNS-BE. This audit was performed against commit 2704899, 1ee24cc where after our initial engagement, we performed reviews for additional scopes, including adding support for badges, as well as, 3ffd970, 8710fda, 4569a1b, and 80cdc79. An additional review was conducted against PR#12915

A brief description of the programs is as follows:

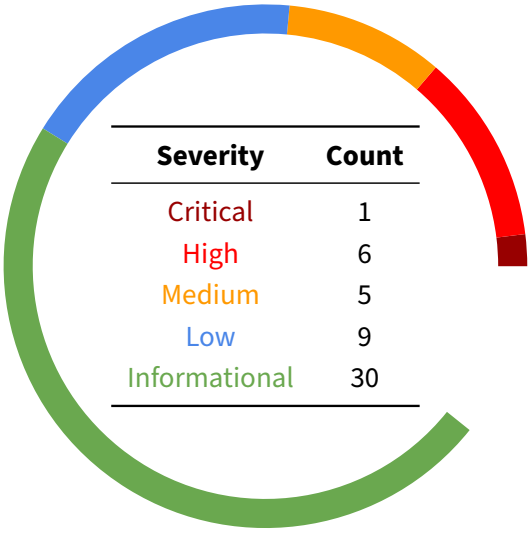
Name	Description
sui-framework	An on-chain library designed for developing smart contracts within the Sui ecosystem.
sui-system	An on-chain library voting for governance and protocol upgrades.
sui-json-rpc	An API for the interaction with Sui full node.
sui-adapter	An adapter and command line interface tool designed for local development in the Sui ecosystem.
collectible	An all-in-one package that aims to create a display, a publisher, and a transfer policy to enable kiosk trading.
kiosk	A primitive for building open, zero-fee trading platforms with a high degree of customization over transfer policies.
sui-verifier	A bytecode verifier for the Sui module.
suifrens	A contract that mints and mixes `SuiFren` objects. The application contains an accessory store where users may buy and add accessories to their `SuiFrens`, while admins may mint and reward users with badges.
suins	A next-generation identity service protocol that allows users to register, auction, buy, and sell domain names on the Sui blockchain. These on-chain modules are split into auction, controller, registrar, registry, and utils.
sui.id	A Web2 frontend that resolves a user's domain to their Sui address on the explorer, built on top of `suins` smart contracts.

© 2023 Otter Audits LLC. All Rights Reserved.

03 | Findings

Overall, we reported 51 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 do not have an immediate impact but will help mitigate future vulnerabilities.



Severity	Count
Critical	1
High	6
Medium	5
Low	9
Informational	30

© 2023 Otter Audits LLC. All Rights Reserved.

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-SUI-ADV-00	Critical	Resolved	Possible collision of hash in `hash_type_and_key`.
OS-SUI-ADV-01	High	Resolved	RPC node crashes due to an overflow in `check_total_coins`.
OS-SUI-ADV-02	High	Resolved	Hash collisions compromise permissionless upgrades.
OS-SUI-ADV-03	High	Resolved	Bypass of the `id_leak_verifier` stage of `sui-verifier` may occur.
OS-SUI-ADV-04	High	Resolved	`suifren_update_last_epoch_mixed` allows users to bypass checks during the mix.
OS-SUI-ADV-05	High	Resolved	Minting of `Suifrens` with an insufficient mixing limit may occur due to a typo during mixing.
OS-SUI-ADV-06	High	Resolved	The helper function responsible for retrieving all fields of `domain_names` returns incorrect values.
OS-SUI-ADV-07	Medium	Resolved	The `lock` permission implicitly grants `place` functionality, enabling unauthorized item placement even when the `place` permission is explicitly restricted.
OS-SUI-ADV-08	Medium	Resolved	Withdrawals from staking pools may result in rounding errors, which results in lost rewards.
OS-SUI-ADV-09	Medium	Resolved	Initializing `committee` with high values results in an overflow within the program.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-10	Medium	Resolved	The owner of `kiosk` may block the user's funds by calling `set_allow_extension`.
OS-SUI-ADV-11	Medium	Resolved	Users may set a TTL value that does not follow the maximum TTL limit.
OS-SUI-ADV-12	Low	Resolved	Incorrect checks result in the absence of verification of the invariant.
OS-SUI-ADV-13	Low	Resolved	Possibility of reaching `unreachable!()` code.
OS-SUI-ADV-14	Low	Resolved	The seed for minting does not update.
OS-SUI-ADV-15	Low	Resolved	Users may arbitrarily decrease the remaining mixes of their `SuiFrens`.
OS-SUI-ADV-16	Low	Resolved	Insufficient checks for the order of genes.
OS-SUI-ADV-17	Low	Resolved	A lack of security headers allows for clickjacking attacks.
OS-SUI-ADV-18	Low	Resolved	Erroneous checks allow the user to create an invalid discount code.
OS-SUI-ADV-19	Low	Resolved	Lack of validation in setting and retrieving default domain names.
OS-SUI-ADV-20	Low	Resolved	`profits_mut` grants unrestricted admin control over kiosk profits, posing a risk of manipulation and rendering profits data unreliable for extensions.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-00 [crit] | Dynamic Field Hash Collision

Description

In `dynamic_fields.move`, `hash_type_and_key` computes hashes based on an input `address` and an `Object`. The hash is created by concatenating three parts:

1. `parent`, `AccountAddress`, which serves as the parent of the object.
2. `k_bytes`, the data of the object being hashed. It includes the values associated with the object.
3. `k_tag_bytes`, the data type of the object. It provides information about the structure of the object.

By combining these three components, the hash uniquely identifies the object based on its parent, data, and data type.

However, due to the semi-arbitrary structure of the `Struct` and `Vector` data types, it is possible to produce a collision in the generated hashes. This collision may result in incorrect behavior for `add` and `exists_`, which do not account for the `Name` parameter when performing native functions. meanwhile, other functions call native functions using a generic parameter `Field<Name, Value>`.

Remediation

Insert the length of the `k_bytes` and `k_tag_bytes` arrays at the beginning of their values in the `hasher` object.

crates/sui-framework/src/natives/dynamic_field.rs DIFF

```diff
@@ -92,7 +92,9 @@ pub fn hash_type_and_key(
     // hash(parent || k || K)
     let mut hasher = Sha3_256::default();
     hasher.update(parent);
+    hasher.update(k_bytes.len().to_le_bytes());
     hasher.update(k_bytes);
+    hasher.update(k_tag_bytes.len().to_le_bytes());
     hasher.update(k_tag_bytes);
     let hash = hasher.finalize();
```

Patch

Fixed in 5b71cef.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-01 [high] | RPC Node Crashes Due To An Overflow

Description

In `execution_engine.rs`, `check_total_coins` calculates the total amount of coins from `&[Coin]`. However, this function does not handle the scenario where the total amount of coins exceeds the maximum u64 value. In this case, the function returns incorrect values, crashing the application.

sui-adapter/src/execution_engine.rs RUST

```rust
fn check_total_coins(coins: &[Coin], amounts: &[u64]) -> Result<(u64, u64),
    ↪  ExecutionError> {
    let total_amount: u64 = amounts.iter().sum();
    let total_coins = coins.iter().fold(0, |acc, c| acc + c.value());
    if total_amount > total_coins {
        return Err(ExecutionError::new_with_source(
            ExecutionErrorKind::InsufficientBalance,
            format!("Attempting to pay a total amount {:?} that is greater than the
    ↪  sum of input coin values {:?}", total_amount, total_coins),
        ));
    }
    Ok((total_coins, total_amount))
}
```

Remediation

Handle the scenario where the total amount of `Coins` exceeds u64 by returning an appropriate error message.

crates/sui-adapter/src/execution_engine.rs DIFF

```diff
@@ -540,15 +540,21 @@ fn check_recipients(recipients: &[SuiAddress], amounts:
    ↪  &[u64]) -> Result<(), Ex

 fn check_total_coins(coins: &[Coin], amounts: &[u64]) -> Result<(u64, u64),
    ↪  ExecutionError> {
-    let total_amount: u64 = amounts.iter().sum();
-    let total_coins = coins.iter().fold(0, |acc, c| acc + c.value());
+    let total_amount: Option<u64> = amounts.iter().try_fold(0_u64, |acc, &a|
    ↪  acc.checked_add(a));
+    let total_coins = coins.iter().try_fold(0_u64, |acc, c|
    ↪  acc.checked_add(c.value()));
+    if total_amount.is_none() || total_coins.is_none() {
+        return Err(ExecutionError::new_with_source(
+            ExecutionErrorKind::Overflow,
+            "Value overflow",
+        ));
+    }
     if total_amount > total_coins {
```

© 2023 Otter Audits LLC. All Rights Reserved.

```
            return Err(ExecutionError::new_with_source(
                ExecutionErrorKind::InsufficientBalance,
                format!("Attempting to pay a total amount {:?} that is greater than
    ↪  the sum of input coin values {:?}", total_amount, total_coins),
            ));
        }
-       Ok((total_coins, total_amount))
+       Ok((total_coins.unwrap(), total_amount.unwrap()))
    }
```

Patch

Fixed in 7485c45.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-02 [high] | Modules Digest Collision

Description

In `move_package.rs`, `compute_digest_for_modules_and_deps` calculates a unique hash according to the `modules` and `object_ids` supplied. Since some protocols depend on this hash to mandate proper upgrades, this hash must be unique. However, the current implementation lacks padding between items, which results in hash collisions.

crates/sui-types/src/move_package.rs RUST

```rust
pub fn compute_digest_for_modules_and_deps<'a>(
    modules: impl IntoIterator<Item = &'a Vec<u8>>,
    object_ids: impl IntoIterator<Item = &'a ObjectID>,
) -> [u8; 32] {
    let mut bytes: Vec<&[u8]> = modules
        .into_iter()
        .map(|x| x.as_ref())
        .chain(object_ids.into_iter().map(|obj_id| obj_id.as_ref()))
        .collect();
    // NB: sorting so the order of the modules and the order of the
↪   dependencies does not matter.
    bytes.sort();

    let mut digest = DefaultHash::default();
    for b in bytes {
        digest.update(b);
    }
    digest.finalize().digest
}
```

Proof of Concept

The provided file describes the issue through a Rust test. It demonstrates the issue by showing that the digest of a package containing two modules, `[m1, m2]`, may be equal to the digest of a package with just one module, `[m1]` if we hide the serialized `m2` in the trailing bytes of the serialized `m1`.

PoC.rs RUST

```rust
#[test]
fn package_digest_collision() {
    let module = empty_module();
    let package = vec![module.clone(), module.clone()];
    let serialized_package1: Vec<Vec<u8>> = package
        .into_iter()
        .map(|m| {
            let mut b = vec![];
            m.serialize(&mut b);
            b
```

© 2023 Otter Audits LLC. All Rights Reserved.

```
        })
        .collect();
    let digest1 =
    ↪  MovePackage::compute_digest_for_modules_and_deps(&serialized_package1,
    ↪  &vec![]);
    let serialized_package2 = vec![{
        let mut v = serialized_package1[0].clone();
        v.extend(serialized_package1[1].clone());
        v
    }];
    let digest2 =
    ↪  MovePackage::compute_digest_for_modules_and_deps(&serialized_package2,
    ↪  &vec![]);
    assert!(digest1 == digest2);
}
```

Remediation

Modify the hash computation by applying the hash function to each module and then another round of hashing to the concatenation of their respective hash outputs.

Patch

Fixed in 4ebc390.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-03 [high] | Bypass Id Leak Verifier

Description

`id_leak_verifier` in `sui-verifier` ensures the non-reusability of a UID for the Sui Object. This step of the verifier guarantees that the `id` field of Sui objects never becomes leaked. However, bypass of these checks may occur utilizing the unique Sui upgrade model. This issue arises due to the following factors:

1. It is possible to add abilities during an upgrade.
2. The verifier does not validate structures without a `Key` capability.
3. It is possible to pass objects between different versions of the same package.

Proof of Concept

The provided file describes the issue through a Sui transaction test. The scenario is as follows:

1. Publish a module containing the `Bar` type without a `Key` capability, allowing the code to `Pack` this type from any `UID`.
2. Upgrade the module, giving the `Bar` type a `Key` capability, while removing the `Pack` instruction from the code.
3. Construct a `Bar` instance using a reused `UID` with the old version of the module, and then pass it into the new version.
4. Notice that the last operation executes without any errors.

PoC.move RUST

```rust
// Copyright (c) Mysten Labs, Inc.
// SPDX-License-Identifier: Apache-2.0
//# init --addresses Test_V0=0x0 Test_V1=0x0 --accounts A
//# publish --upgradeable --sender A
module Test_V0::base {
    use sui::object;
    use sui::object::UID;
    use sui::tx_context::TxContext;
    // no key yet
    struct Foo {
        id: UID,
    }
    // no key yet
    struct Bar {
        id: UID,
    }
    public fun build_foo(ctx: &mut TxContext): Foo {
        Foo {
```

© 2023 Otter Audits LLC. All Rights Reserved.

```
            id: object::new(ctx),
        }
    }
    // works fine because Foo doesn't have key ability
    public fun build_bar_from_foo(foo: Foo): Bar {
        let Foo { id } = foo;
        Bar {
            id: id,
        }
    }
}
//# upgrade --package Test_V0 --upgrade-capability 1,1 --sender A
module Test_V1::base {
    use sui::object::UID;
    use sui::object;
    use sui::tx_context::TxContext;
    // has key which means it should be checked by id_leak_verifier
    struct Foo has key {
        id: UID,
    }
    // has key which means it should be checked by id_leak_verifier
    struct Bar has key {
        id: UID,
    }
    public fun build_foo(ctx: &mut TxContext): Foo {
        Foo {
            id: object::new(ctx),
        }
    }
    public fun build_bar_from_foo(_foo: Foo): Bar {
        // remove the violating instructions
        abort 42
    }
    public fun take_bar(bar: Bar) {
        let Bar {id} = bar;
        object::delete(id);
    }
}
//# programmable --sender A
//> 0: Test_V1::base::build_foo();
//> 1: Test_V0::base::build_bar_from_foo(Result(0));
//> 2: Test_V1::base::take_bar(Result(1));
```

Remediation

Introduce a new check during the upgrade mechanism to avoid adding a `Key` capability to an object.

Patch

Fixed in cbea73e.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-04 [high] | Arbitrary Update Of Last Epoch Mixed

Description

In `suifrens.move`, `suifren_update_last_epoch_mixed` sets the new epoch of mixed, which is intended to prevent users from frequently mixing their SuiFrens without following a countdown period. However, since the function is public, users may set an arbitrary value for `last_epoch_mixed` and bypass the checks in `capy_labs::mix`.

sources/suifrens.move RUST

```rust
public fun suifren_update_last_epoch_mixed<T>(fren: &mut SuiFren<T>, epoch:
↪ u64){
    fren.last_epoch_mixed = epoch;
}
```

sources/capy_labs.move RUST

```rust
public fun mix<T>(
    app: &mut CapyLabsApp,
    sf1: &mut SuiFren<T>,
    sf2: &mut SuiFren<T>,
    clock: &Clock,
    birth_location: vector<u8>,
    ctx: &mut TxContext
): SuiFren<T> {
    [...]
        let last_epoch_mixed_1 = suifrens::suifren_last_epoch_mixed(sf1);
        let last_epoch_mixed_2 = suifrens::suifren_last_epoch_mixed(sf2);

        let epochs_passed_1 = current_epoch - last_epoch_mixed_1;
        let epochs_passed_2 = current_epoch - last_epoch_mixed_2;

        assert!(
            current_epoch == 0 ||
                (epochs_passed_1 >= app.cool_down_period &&
                    epochs_passed_2 >= app.cool_down_period),
            EStillInCoolDownPeriod
    [...]
```

Remediation

Set the visibility of the function to `public(friend)`.

Patch

Fixed in b142be2.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-05 [high] | Mixing Over Limit Suifrens

Description

In `capy_labs.move`, `mix` ensures that users are unable to mix their `Suifrens` over a certain limit. However, a typo in the code results in the application allowing the minting of `Suifrens` with an insufficient `mixing_limit`. This issue occurs in the second `else` statement where the application borrows the value of `l1` instead of `l2` while setting the `mixing_limit` of the second `Suifrens`.

sources/capy_labs.move RUST

```rust
public fun mix<T>(
    app: &mut CapyLabsApp,
    sf1: &mut SuiFren<T>,
    sf2: &mut SuiFren<T>,
    clock: &Clock,
    birth_location: vector<u8>,
    ctx: &mut TxContext
): SuiFren<T> {
    [...]
    // Deal with mixing limits;
    {
        let l1 = mixing_limit(sf1);
        let l2 = mixing_limit(sf2);

        if (option::is_none(&l1)) {
            set_limit(sf1, app.mixing_limit - 1);
        } else {
            let limit = *option::borrow(&l1);
            assert!(limit > 0, EReachedMixingLimit);
            set_limit(sf1, limit - 1);
        };

        if (option::is_none(&l2)) {
            set_limit(sf2, app.mixing_limit - 1);
        } else {
            let limit = *option::borrow(&l1);
            assert!(limit > 0, EReachedMixingLimit);
            set_limit(sf2, limit - 1);
        };
    };
    [...]
}
```

Remediation

Borrow the value of `l2` instead of `l1` in the second `else` statement.

Patch

Fixed in b142be2.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-06 [high] | Incorrect Value In Record Name

Description

`registry::get_name_record_all_fields` retrieves `owner`, `linked_addr`, `ttl`, and `default_domain_name` from a `domain_name`.

source/tmp/registry.move RUST

```rust
public fun get_name_record_all_fields(suins: &SuiNS, domain_name: vector<u8>):
↪ (address, address, u64, String) {
    let name_record = get_name_record(suins, utf8(domain_name));
    (name_record_owner(name_record), name_record_linked_addr(name_record),
↪ name_record_ttl(name_record), name_record_default_domain_name(name_record))
}
```

This function returns erroneous values under two circumstances:

1. When the domain is a normal domain, the function returns an empty string for `default_domain_name`.
2. When the domain is a subdomain of `addr.reverse`, the function returns the default domain name without validation.

Remediation

Handle the previous scenarios and return the correct value.

Patch

This portion of the code has been temporarily removed and will be rewritten in the future.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-07 [med] | Inconsistency in Permission Logic

Description

In the current implementation, the permissions for protected actions in `kiosk_extension` appear to be ambiguous. The existing method allows the permissions value to be set to `0b10` (2), where `lock` is allowed, but `place` is not. This results in an inconsistent state, as `lock` calls `lock_internal` which in turn calls `place_internal`, bypassing the `place` permission check entirely. Thus, this allows an extension with `lock` permissions to effectively perform a `place` action regardless of the set permissions where `place_internal` was disabled.

sources/kiosk/kiosk.move RUST

```rust
/// Internal: "lock" an item disabling the `take` action.
public(friend) fun lock_internal<T: key + store>(self: &mut Kiosk, item: T) {
    df::add(&mut self.id, Lock { id: object::id(&item) }, true);
    place_internal(self, item)
}
```

Remediation

Update the permissions to include only three sets of permissions, none, `only_place`, and `only_place` with `only_lock`, respectively. This ensures that the `lock` permission implicitly grants the ability to `place`, as locking an item involves placing it first.

Patch

Fixed in b83fe70.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-08 [med] | Rounding Errors Result In Lost Accrued Rewards

Description

In `staking_pool.move`, `request_withdraw_stake` is responsible for withdrawing staked tokens and accrued rewards from a staking pool. However, the function utilizes `get_token_amount` to calculate the number of tokens to withdraw, which rounds down the token calculation.

When a user attempts to withdraw a small number of tokens, such as one SUI, the calculation will result in a withdrawal of zero tokens. An attacker may exploit this rounding error to prevent regular users of the staking pool from receiving their rewards accurately.

Proof of Concept

Consider the following scenario:

1. User A stakes 1000 SUI at an exchange rate of `1 SUI : 1 token` to a staking pool that currently has 2000 SUI and 1000 tokens (1000 SUI from rewards).
2. User B stakes 2000 SUI at an exchange rate of `2000 SUI : 1000 tokens` to the same staking pool, which changes the current pool state to 4000 SUI and 2000 tokens.
3. User B begins withdrawing their SUI one by one using split on `StakedSui`. On each withdrawal, the conversion to tokens results in `floor(1 * 1000 / 2000) = 0`. Therefore, User B does not end up withdrawing any tokens every time they withdraw 1 SUI. This process repeats until User B withdraws all of their funds.
4. At this point, the pool state is User A: 1000 staked SUI at an exchange rate of `1 SUI : 1 token`, with the pool having 2000 SUI and 2000 tokens.
5. User A attempts to withdraw their 1000 SUI, which gets converted to 1000 tokens using the original exchange rate.
6. `withdraw_rewards` ends up calculating the `total_sui_withdraw_amount` as 1000 SUI, and as a result, the rewards end up being zero.
7. User A ends up with 1000 SUI instead of 2000 SUI, losing all their accrued rewards.

Remediation

Disallow the generation of `StakedSui` objects with principal less than one `SUI`.

Patch

Fixed in cdc7dad.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-09 [med] | Potential Overflow In Threshold

Description

`commitee.rs` includes a helper function for calculating `quorum_threshold`. However, this function does not account for where the `Committee` object is initialized with voting rights, resulting in a return of an incorrect value due to an overflow.

sui-types/src/commitee.rs RUST

```rust
pub fn quorum_threshold(&self) -> StakeUnit {
    // If N = 3f + 1 + k (0 <= k < 3)
    // then (2 N + 3) / 3 = 2f + 1 + (2k + 2)/3 = 2f + 1 + k = N - f
    2 * self.total_votes / 3 + 1 // overflow possible
}
```

Remediation

Replace the current implementation of `quorum_threshold` with constants since the total voting power and quorum threshold remain fixed and are not affected by changes in the stake.

Patch

Fixed in 9bbf7b9.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-10 [med] | Blocking User Funds In Kiosk

Description

The concept of the kiosk's extensions heavily depends on utilizing the `uid_mut` function. However, the current implementation may disallow the use of this function by calling `set_allow_extensions` with `allow_extensions` set to `false`. This may result in the locking of users' funds that were transferred to the extension.

kiosk.move RUST

```rust
/// Get the mutable `UID` for dynamic field access and extensions.
/// Aborts if `allow_extensions` set to `false`.
public fun uid_mut(self: &mut Kiosk): &mut UID {
    assert!(self.allow_extensions, EExtensionsDisabled);
    &mut self.id
}
```

Remediation

Implement a fallback mechanism that enables the termination of the extensions to enhance their functionality.

Patch

This API is now deprecated, and this issue no longer exists.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-11 [med] | Absence Of Checks For Max TTL

Description

The `MAX_TTL` constant is present in `registry.move`. However, it is not verified when new domains are registered or a new TTL is assigned. In the absence of this check, users may assign invalid TTL values to their domain names.

source/tmp/registry.move RUST

```rust
public entry fun set_ttl(suins: &mut SuiNS, domain_name: vector<u8>, ttl: u64, ctx:
↪  &mut TxContext) {
    authorised(suins, domain_name, ctx);

    let domain_name = string::utf8(domain_name);
    let record = get_name_record_mut(suins, domain_name);

    *entity::name_record_ttl_mut(record) = ttl;
    event::emit(TTLChangedEvent { domain_name, new_ttl: ttl });
}
[...]
public(friend) fun set_record_internal(
    suins: &mut SuiNS,
    domain_name: String,
    owner: address,
    ttl: u64,
    ctx: &mut TxContext,
) {
    let registry = entity::registry_mut(suins);
    if (table::contains(registry, domain_name)) {
        let record = table::borrow_mut(registry, domain_name);
        *name_record_owner_mut(record) = owner;
        *name_record_ttl_mut(record) = ttl;
        *name_record_linked_addr_mut(record) = owner;
    } else new_record(suins, domain_name, owner, ttl, ctx);
}
```

Remediation

Implement proper validation checks to enforce the maximum TTL size during domain registration and when setting the TTL value.

Patch

This portion of the code has been temporarily removed and will be rewritten in the future.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-12 [low] | Absence Of Invariant Check In Governance

Description

In `voting_power.move`, `check_invariants` checks the enforcement of invariants after setting the voting power. However, the first `if` statement compares `stake_i` with itself instead of `stake_j`, not checking the invariant.

governance/voting_power.move RUST

```rust
if (stake_i > stake_i) {
    assert!(power_i >= power_j, ERelativePowerMismatch);
};
if (stake_i < stake_j) {
    assert!(power_i <= power_j, ERelativePowerMismatch);
};
```

Remediation

Compare `stake_i` with `stake_j` in the first `if` statement.

Patch

Fixed in d9d76e9.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-13 [low] | Unreachable Code

Description

`robust_value` should always return a value. However, entering the `unreachable!()` code in this function is possible due to a potential overflow in `total += s` or by passing empty items.

crates/sui-types/src/committee.rs RUST

```rust
pub fn robust_value<A, V>(
    &self,
    items: impl Iterator<Item = (A, V)>,
    threshold: StakeUnit,
) -> (AuthorityName, V)
where
    A: Borrow<AuthorityName> + Ord,
    V: Ord,
{
    debug_assert!(threshold < self.total_votes);

    let items = items
        .map(|(a, v)| (v, self.weight(a.borrow()), *a.borrow()))
        .sorted();
    let mut total = 0;
    for (v, s, a) in items {
        total += s;
        if threshold <= total {
            return (a, v);
        }
    }
    unreachable!();
}
```

Remediation

Remove `robust_value` as it is not utilized in the codebase.

Patch

Fixed in 9bbf7b9.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-14 [low] | Lack Of Seed For Minting Update

Description

In the `Mint` object, the `inner_hash` field provides a seed for minting new `suifrens`. This field should update after every mint with a new value.

sources/genesis.move RUST

```rust
public fun mint<T>(
    self: &mut Mint,
    clock: &Clock,
    birth_location: String,
    paid: &mut Coin<SUI>,
    ctx: &mut TxContext
): SuiFren<T> {
    assert!(suifrens::is_authorized<T>(&mut self.id), EMintNotAuthorized);

    handle_payment(self, paid, ctx);

    let genes = hash(&bcs::to_bytes(&vector[
        self.inner_hash,
        bcs::to_bytes(&fresh_object_address(ctx)),
        bcs::to_bytes(clock)
    ]));

    let attributes = genes::get_attributes<T>(&self.id, &genes);

    suifrens::mint<T>(&mut self.id, 0, genes, attributes, clock, birth_location,
↪   option::some(self.mixing_limit), ctx)
}
```

Remediation

Set the `self.inner_hash` value to genes at the end of `mint`.

Patch

Fixed in 5462a42.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-15 [low] | Decrease Remaining Mixes Of SuiFrens

Description

Users may arbitrarily call `decrease_remaining_mixes_by_one`. Users may supply their `SuiFrens` and reduce the remaining mix in the object without minting a new `SuiFrens`. This function also does not check if the `remaining_mix` is set to `None()`, aborting the application.

sources/genesis.move RUST

```rust
public fun decrease_remaining_mixes_by_one<T>(self: &mut SuiFren<T>) {
    let rm = option::borrow(&self.remaining_mixes);
    assert!(*rm >= 1, 0);
    option::swap(&mut self.remaining_mixes, *rm - 1);
}
```

Remediation

1. Ensure that the function may be called only during the mixing process.
2. Add a check to ensure that the `remaining_mix` is not set to `None()`.

Patch

Mysten acknowledged this issue and removed the `remaining_mixes` field from `SuiFren` and consequently `decrease_remaining_mixes_by_one`.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-16 [low] | Insufficient Gene Definition Checks

Description

`definitions_from_bcs` should ensure the correct order of genes. Otherwise, if definitions are set in an incorrect order, receiving parts of the value becomes impossible.

sources/genesis.move RUST

```rust
public fun definitions_from_bcs(bytes: vector<u8>): vector<GeneDefinition> {
    let bytes = bcs::new(bytes);
    let total = bcs::peel_vec_length(&mut bytes);
    let defs = vector::empty<GeneDefinition>();

    // Iterate over the main vector of `GeneDefinition`
    while (total > 0) {
        let name = utf8(bcs::peel_vec_u8(&mut bytes));
        let values = vector::empty<Value>();
        let val_len = bcs::peel_vec_length(&mut bytes);

        // Read each value separately.
        while (val_len > 0) {
            let (selector, val_name) = (
                bcs::peel_u8(&mut bytes),
                utf8(bcs::peel_vec_u8(&mut bytes))
            );

            vector::push_back(&mut values, Value { selector, name: val_name });
            val_len = val_len - 1;
        };

        vector::push_back(&mut defs, GeneDefinition { name, values });
        total = total - 1;
    };

    defs
}
```

Remediation

Store `prev_selector` in a variable outside the loop and check if `current_selector` is greater than `prev_selector` to ensure that `selector` is always increasing.

Patch

Fixed in 5462a42.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-17 [low] | Clickjacking

Description

The website's HTTP responses do not include the `X-Frame-Options` security header, which allows attackers to have the website in iframes within a malicious web page and perform clickjacking attacks by forcing the user to click on buttons inside the iframe and perform undesired actions.

HTTP Response headers HTTP

```
HTTP/1.1 200 OK
x-guploader-uploadid: ADPycdvnisNpDx-
    ↪  t1jfvq8n_kJRZ_UTP0i883tA0aKXLaq5drZwTLxBvmBeVoOjA0WTuT7u4qKxnQuXqBK6BC5N1Oteylg
x-goog-generation: 1682592229165780
x-goog-metageneration: 1
x-goog-stored-content-encoding: identity
x-goog-stored-content-length: 1327
x-goog-hash: crc32c=Sbaw1g==
x-goog-hash: md5=Z/GS730itI5X4wvlwHKfJQ==
x-goog-storage-class: STANDARD
accept-ranges: bytes
Content-Length: 1327
server: UploadServer
via: 1.1 google
Date: Thu, 27 Apr 2023 21:11:32 GMT
Age: 2810
Last-Modified: Thu, 27 Apr 2023 10:43:49 GMT
ETag: "67f192ef7d22b48e57e30be5c0729f25"
Content-Type: text/html
Cache-Control: public,max-age=3600
Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
Connection: close
```

Remediation

Add the header `X-Frame-Options: SAMEORIGIN` in the HTTP response to only allow same-origin iframes. However, if the application is required to be opened by external websites inside iframes, configure the `frame-ancestors` CSP directive.

Patch

The Mysten Labs team has acknowledged this finding and plans to implement a patch.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-18 [low] | Incorrect Check In Discount Code

Description

In `remove_later::deserialize_discount_code`, the current validation for the discount code rate during deserialization is incorrect, as it only checks if the rate is greater than zero or less than three characters. The intended validation should verify that the rate is greater than zero and less than three characters.

source/tmp/remove_later.move RUST

```
fun deserialize_discount_code(str: String): DiscountCode {
    [...]
    // rate cannot has more than 3 characters
    // index_of_next_colon == 0: rate is not included
    assert!((0 < index_of_next_comma || index_of_next_comma < 3),
↪   EInvalidDiscountCodeBatch);
```

Remediation

Modify the logical operator used in the validation to ensure that both checks are executed.

Patch

This portion of the code has been temporarily removed and will be rewritten in the future.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-19 [low] | Missing Check In Setting Default Domain Name

Description

In `registry.move`, `set_default_domain_name` sets the value of the new `default_domain_name`, however, several checks are missing.

1. The function does not check if `new_default_domain_name` exists or if the sender is its owner.
2. The function only permits modifying the default domain name setting for subdomains of `addr.reverse`, leaving the field empty for all other domains.
3. The default domain name still points to the same domain if the owner changes.
4. The default domain name should not be accessible through any other public functions. However, `registry::get_name_record_all_fields` returns the default domain name without validation.

source/tmp/registry.move RUST

```rust
public entry fun set_default_domain_name(
    suins: &mut SuiNS,
    new_default_domain_name: vector<u8>,
    ctx: &mut TxContext,
) {
    let reverse_label = hex::encode(address::to_bytes(sender(ctx)));
    let reverse_domain_name = make_subdomain_name(reverse_label,
↪   utf8(ADDR_REVERSE_TLD));
    let name_record = get_name_record_mut(suins, reverse_domain_name);
    let new_default_domain_name = utf8(new_default_domain_name);

    *entity::name_record_default_domain_name_mut(name_record) =
↪   new_default_domain_name;
    event::emit(DefaultDomainNameChangedEvent { domain_name:
↪   reverse_domain_name, new_default_domain_name });
}
```

Remediation

Ensure that all provided checks are properly implemented.

Patch

This portion of the code has been temporarily removed and will be rewritten in the future.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-ADV-20 [low] | Profit Manipulation by Kiosk Owner

Description

`kiosk::profits_mut` introduces a potential vulnerability due to the mutable access it grants to the `profits` field, allowing the `kiosk` owner to adjust the profits of the `kiosk`. Thus if a `kiosk` extension uses profits as a metric in its logic, it will be possible for the `Kiosk` owner to manipulate `profits` via `profits_mut` in ways that may undermine the logic of the extension.

sources/kiosk/kiosk.move RUST

```rust
public fun profits_mut(self: &mut Kiosk, cap: &KioskOwnerCap): &mut Balance<SUI> {
    assert!(has_access(self, cap), ENotOwner);
    &mut self.profits
}
```

Remediation

Update the documentation to clearly state that `profits` is mutable by the owner and should not be considered a reliable metric for extensions.

Patch

Resolved in b83fe70.

© 2023 Otter Audits LLC. All Rights Reserved.

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 anti-patterns and may lead to security issues in the future.

ID	Description
OS-SUI-SUG-00	Replace `panic!()` with `debug_assert!()` in `id_leak_verifier` for the release version.
OS-SUI-SUG-01	`is_call_safe_to_leak` contains an invalid option.
OS-SUI-SUG-02	An incorrect parameter index is utilized in `entry_points_verifier`.
OS-SUI-SUG-03	Presence of a small discrepancy of the allowed supply in `collectible.move`.
OS-SUI-SUG-04	Absence of checks for the existence of `TransferPolicy<T>`.
OS-SUI-SUG-05	`kiosk.move` contains duplicated code replaceable with `has_access`.
OS-SUI-SUG-06	Absence of checks for the existence of the added rule in `TransferPolicy`.
OS-SUI-SUG-07	Absence of checks for the existence of the item in the kiosk.
OS-SUI-SUG-08	`TreasuryCap` includes the capability to upgrade coin details in `coin.move`.
OS-SUI-SUG-09	Absence of a public function that returns `UpgradeCap` from `UpgradeTicket`.
OS-SUI-SUG-10	Absence of a check for the `init` modifier in `sui-verifier`.
OS-SUI-SUG-11	Introduce new features to allow an administrator to change configurations and users to manage accessories and badges.
OS-SUI-SUG-12	Store the last epoch mixed of `suifrens` instead of calling a helper function twice.
OS-SUI-SUG-13	Change how the seed is calculated by removing the creation of a new ID.
OS-SUI-SUG-14	Discrepancies between the attributes list used during the mint process of SuiFrens.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-15	`lor` is unnecessary and may be replaced with a simple comparison operator.
OS-SUI-SUG-16	There is a risk with exposing public functions in the registry module.
OS-SUI-SUG-17	The current CSP configuration allows for bypasses.
OS-SUI-SUG-18	`getCookie` incorrectly parses `document.cookie` while `setCookie` allows cookie attributes injection.
OS-SUI-SUG-19	Potential path traversal present in `baseImageFileName`.
OS-SUI-SUG-20	CORS allows `localhost` in the production environment.
OS-SUI-SUG-21	Image IPFS URL is not validated.
OS-SUI-SUG-22	Lack of check for the creation date in the second highest bid.
OS-SUI-SUG-23	Rounding error when executing code returns an incorrect rate.
OS-SUI-SUG-24	Valid payments may result in an abort of the program while registering.
OS-SUI-SUG-25	Correct the method of creating a new ID by removing the creation of UID.
OS-SUI-SUG-26	Incorrectly validating the domain on the client side may result in future vulnerabilities.
OS-SUI-SUG-27	Several recommendations around code style and documentation.
OS-SUI-SUG-28	Compiler Errors due to empty comments.
OS-SUI-SUG-29	Suggestions regarding ensuring adherence to coding best practices.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-00 | Replace Panic Call With Debug Assertion

Description

`id_leak_verifier` prevents the leakage of the unique IDs of Sui objects. It tracks the flow of the value extracted by the unpack bytecode and prevents it from being returned. It is written into a mutable reference, added to a vector, or passed to a function.

This prevents the reuse of IDs. However, the current implementation of `id_leak_verifier` utilizes `panic!()` calls, which may result in unexpected aborts in the release version.

Remediation

Replace the `panic!()` invocation with `debug_assert!()`.

DIFF

```diff
 fn call(verifier: &mut IDLeakAnalysis, function_handle: &FunctionHandle) ->
     ↪  PartialVMResult<()> {
@@ -337,7 +330,7 @@ fn execute_inner(
                 | Bytecode::MutBorrowGlobalGeneric(_)
                 | Bytecode::Exists(_)
                 | Bytecode::ExistsGeneric(_) => {
-            panic!("Should have been checked by global_storage_access_verifier.");
+            debug_assert!(false, "Should have been checked by
     ↪  global_storage_access_verifier.");
         }

         Bytecode::Call(idx) => {
```

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-01 | Presence Of Removed Function

Description

In `id_leak_verifier`, `is_call_safe_to_leak` is responsible for verifying if leaking an object's identifier is safe. In the code, a legacy option named `delete_child_object` exists that was utilized in the past but was removed in a previous update. However, this option still is present in the code.

Remediation

Remove the `delete_child_object` option.

DIFF

```diff
     }

     // sui::object::delete
-    (verifier.binary_view.identifier_at(m.name) == OBJECT_MODULE_NAME
+    verifier.binary_view.identifier_at(m.name) == OBJECT_MODULE_NAME
         && verifier
             .binary_view
             .identifier_at(function_handle.name)
             .as_str()
-            == "delete") ||
-    // sui::transfer::delete_child_object
-    (verifier.binary_view.identifier_at(m.name).as_str() == "transfer"
-            && verifier
-                .binary_view
-                .identifier_at(function_handle.name)
-                .as_str()
-                == "delete_child_object")
+            == "delete"
 }
```

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-02 | Erroneous Signature Check

Description

In `entry_points_verifier`, `format_signature_token` is invoked with the first parameter instead of the last parameter, resulting in an incorrect signature token format. This is due to the `TxContext` type parameter always being the last of an entry function.

Remediation

Ensure that the second parameter of `format_signature_token` is the last element of the `parameters` vector.

```diff
            SUI_FRAMEWORK_ADDRESS,
            TX_CONTEXT_MODULE_NAME,
            TX_CONTEXT_STRUCT_NAME,
-           format_signature_token(view, &parameters[0]),
+           format_signature_token(view, &parameters[parameters.len() - 1]),
        ))
    }
}
```

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-03 | Discrepancy In Allowed Supply

Description

In `collectible.move`, `batch_mint` and `mint` contain a small discrepancy in the allowed supply. Specifically, `batch_mint` utilizes the < operator in the length assertion check, which results in a maximum supply of `cap.max_supply - 1`, while `mint` utilizes the <= operator, which allows for the full `cap.max_supply` to be attained.

Remediation

Modify the length assertion check in `batch_mint` to utilize the <= operator instead of the < operator.

DIFF

```diff
        // safe to downcast since the length will never be greater than u32::MAX
        assert!(
            option::is_none(&cap.max_supply)
-           || cap.minted + (len as u32) < *option::borrow(&cap.max_supply)
+           || cap.minted + (len as u32) <= *option::borrow(&cap.max_supply)
        , ECapReached);

        assert!(
```

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-04 | Lack Of Transfer Policy Existence Check

Description

In `kiosk.move`, the docstring above `place<T>` indicates that the function needs to check whether `TransferPolicy<T>` exists. However, this validation check is not present.

kiosk.move RUST

```rust
public fun place<T: key + store>(
    self: &mut Kiosk, cap: &KioskOwnerCap, item: T
) {
    assert!(object::id(self) == cap.for, ENotOwner);
    self.item_count = self.item_count + 1;
    dof::add(&mut self.id, Item { id: object::id(&item) }, item)
}
```

Remediation

Add the `_policy: &TransferPolicy<T>` parameter to `place<T>` to ensure the existence of `TransferPolicy<T>`.

DIFF

```diff
     /// Makes sure a `TransferPolicy` exists for `T`, otherwise assets can be
     /// locked in the `Kiosk` forever.
     public fun place<T: key + store>(
-        self: &mut Kiosk, cap: &KioskOwnerCap, item: T
+        self: &mut Kiosk, cap: &KioskOwnerCap, _policy: &TransferPolicy<T>, item:
    ↪  T
     ) {
         assert!(object::id(self) == cap.for, ENotOwner);
         self.item_count = self.item_count + 1;
```

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-05 | Avoid Duplicated Code

Description

`kiosk.move` utilizes assert statements to check whether the `KioskOwnerCap` matches the `Kiosk`. Instead of invoking `has_access`, the duplicated code discovered in `has_access` is incorporated.

kiosk.move RUST

```rust
public fun has_access(self: &mut Kiosk, cap:&KioskOwnerCap): bool {
    object::id(self) == cap.for
}

public fun set_owner(
    self: &mut Kiosk, cap: &KioskOwnerCap, ctx: &TxContext
) {
    assert!(object::id(self) == cap.for, ENotOwner);
    self.owner = sender(ctx);
}
```

Remediation

Replace the duplicated code with `has_access`.

DIFF

```diff
     public fun set_owner(
         self: &mut Kiosk, cap: &KioskOwnerCap, ctx: &TxContext
     ) {
-        assert!(object::id(self) == cap.for, ENotOwner);
+        assert!(has_access(self, cap), ENotOwner);
         self.owner = sender(ctx);
     }
```

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-06 | Check Rule Existence

Description

`add_receipt` adds a rule to the `receipts` of `TransferRequest`. However, it does not check if the rule to add is present in `TransferPolicy`.

transfer_policy.move RUST

```rust
public fun add_receipt<T, Rule: drop>(
    _: Rule, request: &mut TransferRequest<T>
) {
    vec_set::insert(&mut request.receipts, type_name::get<Rule>())
}
```

Remediation

Add an assert statement utilizing `has_rule` to verify the existence of the rule.

DIFF

```diff
     /// Adds a `Receipt` to the `TransferRequest`, unblocking the request and
     /// confirming that the policy requirements are satisfied.
     public fun add_receipt<T, Rule: drop>(
-        _: Rule, request: &mut TransferRequest<T>
+        _: Rule, request: &mut TransferRequest<T>, policy: &TransferPolicy<T>
     ) {
+        assert!(has_rule<T, Rule>(policy), 1);
         vec_set::insert(&mut request.receipts, type_name::get<Rule>())
     }
```

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-07 | Check Item Existence

Description

`list` and `list_with_purchase_cap` adds items to the kiosk, allowing users to purchase them. However, it does not verify that the items exist in the kiosk.

Remediation

Check if the items exist in the kiosk.

kiosk.move DIFF

```diff
@@ -255,6 +270,7 @@ module sui::kiosk {
        self: &mut Kiosk, cap: &KioskOwnerCap, id: ID, price: u64
    ) {
        assert!(object::id(self) == cap.for, ENotOwner);
+       assert!(has_item(self, id), EItemNotFound);

@@ -299,6 +317,7 @@ module sui::kiosk {
        self: &mut Kiosk, cap: &KioskOwnerCap, id: ID, min_price: u64, ctx: &mut
    ↪   TxContext
    ): PurchaseCap<T> {
        assert!(object::id(self) == cap.for, ENotOwner);
+       assert!(has_item(self, id), EItemNotFound);
```

Patch

Fixed in 7f62c01.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-08 | Separate Upgrade Capability From Treasury Capability

Description

In `coin.move`, `TreasuryCap` obtained after the creation of the currency has the capability to update the details of the coin. Adopting a distinct `UpgradeCap` exclusively for updating the coin details instead of relying on `TreasuryCap` may be considered an advantageous approach.

This approach would facilitate a clear distinction between the capabilities needed for various actions such as minting, burning, and updating coin details.

Remediation

Separate `UpgradeCap` and `TreasuryCap` to enhance the distinction between these capabilities.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-09 | Missing Public Function

Description

In `package.move`, `authorize_upgrade` issues an `UpgradeTicket` that enables a particular upgrade to be performed utilizing `UpgradeCap`. However, the public function to return the `UpgradeCap` from an `UpgradeTicket` does not exist.

Remediation

Add a public function that returns `UpgradeCap` from `UpgradeTicket`.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-10 | Lack Of Checks For Init Function In Verifier

Description

In `sui-verifier`, `verify_init_function` checks the correctness of `init` before the deployment. However, this function does not guarantee the presence of the `entry` modifier.

Remediation

Add the correct check to ensure that the `init` function is not an `entry` function.

Patch

Fixed in 5ae2698.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-11 | Lack Of Functionalities

Description

In `genesis.move`, a new `CapyLabsApp` is created and configured with the default value during `init`. However, changing the `mixing_limit` field in the contract is impossible. A user with `AdminCap` privileges should be able to modify its value using a helper function.

sources/capy_labs.move RUST

```rust
fun init(ctx: &mut TxContext) {
    let id = object::new(ctx);
    let inner_hash = hash(&object::uid_to_bytes(&id));
    let app = CapyLabsApp {
        id,
        inner_hash,
        mixing_limit: DEFAULT_MIXING_LIMIT,
        cool_down_period: DEFAULT_COOL_DOWN_PERIOD_IN_EPOCHS,
        mixing_price : DEFAULT_MIXING_PRICE,
        profits: balance::zero<SUI>()
    };

    transfer::share_object(app)
}
```

In `accessories.move` and `badges.move`, users may mint new objects, but they are unable to be deleted once created.

Remediation

Introduce the following:

1. A new admin function that allows changing the value of `mixing_limit` in `capy_labs.move`.
2. New functions to delete minted objects and refund payments in `accessories.move` and in `badges.move`.

Patch

The first remediation recommendation has been addressed in a1a74d6.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-12 | Storing Value During Mix

Description

During `mix` in `capy_labs.move`, `last_epoch_mixed` is invoked twice for every `suifrens`. Consider storing the value of `last_epoch_mixed(sf1)` and `last_epoch_mixed(sf2)`.

sources/capy_labs.move RUST

```rust
public fun mix<T>(
        self: &mut CapyLabsApp,
        sf1: &mut SuiFren<T>,
        sf2: &mut SuiFren<T>,
        clock: &Clock,
        birth_location: String,
        paid: &mut Coin<SUI>,
        ctx: &mut TxContext
    ): SuiFren<T> {
        [...]
        // Deal with Cool Down Period Business Rules
        {
            if (option::is_some(&last_epoch_mixed(sf1))) {
                let last_epoch_mixed_1 = *option::borrow(&last_epoch_mixed(sf1));
                let epochs_passed_1 = current_epoch - last_epoch_mixed_1;
                assert!(epochs_passed_1 >= self.cool_down_period, EStillInCoolDownPeriod)
            };

            if (option::is_some(&last_epoch_mixed(sf2))) {
                let last_epoch_mixed_2 = *option::borrow(&last_epoch_mixed(sf2));
                let epochs_passed_2 = current_epoch - last_epoch_mixed_2;
                assert!(epochs_passed_2 >= self.cool_down_period, EStillInCoolDownPeriod)
            };
    [...]
```

Remediation

Store the value of `last_epoch_mixed` and reuse it in the mix function.

Patch

Fixed in 5462a42.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-13 | Change The Calculation Of Seed

Description

During `mix` in `capy_labs.move`, a new `UID` is created for the calculation and deleted once the calculation is done. This operation is unnecessary and may be replaced by invoking `tx_context::fresh_object_address`.

sources/capy_labs.move RUST

```rust
public fun mix<T>(
        self: &mut CapyLabsApp,
        sf1: &mut SuiFren<T>,
        sf2: &mut SuiFren<T>,
        clock: &Clock,
        birth_location: String,
        paid: &mut Coin<SUI>,
        ctx: &mut TxContext
    ): SuiFren<T> {
        [...]
        let id = object::new(ctx);

        // Create seed based on multiple sources: clock, fresh uid and an inner
    ↪   hash.
        let seed = hash(&bcs::to_bytes(&vector[
            self.inner_hash,
            object::id_to_bytes(&object::uid_to_inner(&id)),
            bcs::to_bytes(clock)
        ]));

        object::delete(id);
    [...]
```

Remediation

Change the method of seed calculation by utilizing `tx_context::fresh_object_address`.

Patch

Fixed in 5462a42.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-14 | Discrepancy Between Function Calls

Description

There is a difference in the source of the attributes list utilized in the `get_attributes` function calls between `capy_labs::mix` and `genesis::mint`.

The attribute `vec_map` in `capy_labs::mix` is obtained from the dynamic field of the `CapyLabsApp` object, while in `genesis::mint`, it is obtained from the dynamic field of the `Mint` object. This difference may result in unexpected behavior due to possible discrepancies in the attribute definitions within these objects.

sources/genesis.move RUST

```rust
public fun mint<T>(
    self: &mut Mint,
    [...]
): SuiFren<T> {
    [...]
    let attributes = genes::get_attributes<T>(&self.id, &genes);

    suifrens::mint<T>(&mut self.id, 0, genes, attributes, clock, birth_location,
↪   option::some(self.mixing_limit), ctx)
}
```

sources/capy_labs.move RUST

```rust
public fun mix<T>(
    self: &mut CapyLabsApp,
    [...]
): SuiFren<T> {
    [...]
    let attributes = genes::get_attributes<T>(&self.id, &genes);
    let suifren = suifrens::mint<T>(&mut self.id, new_generation, genes,
↪   attributes, clock, birth_location, option::some(self.mixing_limit), ctx);
```

Remediation

Resolve the discrepancies by utilizing consistent attribute loading.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-15 | Remove Unnecessary Function

Description

In `capy_labs.move`, `lor` takes in two arguments and returns a boolean value based on whether the first argument is less than the second argument. However, this functionality may be achieved with a simple comparison operator instead of a dedicated function.

sources/capy_labs.move RUST

```rust
fun lor(rng: u8, cmp: u8): bool {
    (rng < cmp)
}
```

Remediation

Remove the function and replace the invocation with a comparison.

Patch

Fixed in 5462a42.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-16 | Unnecessary Public Functions

Description

The public functions in `registry.move` may lead to serious security issues if an attacker obtains a mutable reference to any instance of `registry::Registry`.

In the current implementation, this threat is mitigated, as an attacker is unable to mutably borrow the original `Registry` stored as a dynamic field, nor create a new one independently. However, an unintentional future change to the registry module may inadvertently allow this to happen, potentially introducing a critical vulnerability.

contracts/sources/registry.move RUST

```rust
/// Attempts to add a new record to the registry and returns a
/// `RegistrationNFT` upon success.
public fun add_record(
    self: &mut Registry,
    domain: Domain,
    no_years: u8,
    clock: &Clock,
    ctx: &mut TxContext,
): RegistrationNFT {
```

`add_record` should not be successfully invoked from an untrusted module. Therefore, adding the `friend` visibility modifier has no drawbacks.

Remediation

Change the visibility of all public functions that accept `&mut Registry` as an argument to `friend`.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-17 | CSP Requires Strictness

Description

The current CSP correctly implements the `script-src` directive. Still, attackers may use different types of HTML injection techniques to leak data, such as a CSS injection or changing the page base URI.

CspHelmet.ts TYPESCRIPT

```typescript
const scriptSrc = [`'nonce-\${nonce}'`, "'strict-dynamic'"];
const directives = {
  scriptSrc,
};
const allowedSrc = ['/assets/', 'https://www.googletagmanager.com/'];
const scriptTags = document.getElementsByTagName('script');

if (import.meta.env.DEV) {
  allowedSrc.push('/src/index.jsx');
  directives.scriptSrcElem = ['localhost:5173'];
}

for (let i = 0; i < scriptTags.length; i++) {
  if (allowedSrc.find((src) => scriptTags[i].getAttribute('src')?.startsWith(src)))
      ↪ {
scriptTags[i].setAttribute('nonce', nonce);
  }
}
```

Remediation

Add more strict CSP directives. This includes `object-src`, `base-uri`, and `style-src`.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-18 | Incorrectly Utilized And Sanitized Cookies

Description

`getCookie` first URL decodes the `document.cookie` attribute and then searches for the cookie by its name. This allows attackers to inject cookies by sending a cookie value with an URL encoded `;` sign. This may be achieved when setting the referral cookie since its value may be attacker-controlled once it is obtained from a query parameter.

cookie.js JAVASCRIPT

```javascript
export const getCookie = (cname) => {
  let name = cname + '=';
  let decodedCookie = decodeURIComponent(document.cookie);
  let ca = decodedCookie.split(';');
  for (let i = 0; i < ca.length; i++) {
    let c = ca[i];
    while (c.charAt(0) === ' ') {
      c = c.substring(1);
    }
    if (c.indexOf(name) === 0) {
      return c.substring(name.length, c.length);
    }
  }
  return '';
};
```

Additionally, `setCookie` receives the cookie name and cookie value as parameters and sets the cookie with `document.cookie =`. However, an arbitrary cookie value may include a `;` sign and inject cookie attributes such as `SameSite`, `domain`, and `path`.

cookie.js JAVASCRIPT

```javascript
export const setCookie = (cname, cvalue, exdays) => {
  const d = new Date();
  d.setTime(d.getTime() + exdays * 24 * 60 * 60 * 1000);
  let expires = 'expires=' + d.toUTCString();
  document.cookie = `${cname}=${cvalue};${expires};path=/`;
};
```

Remediation

URL decode the cookie's value after finding it by its name. Also, URL encode the cookie name and the cookie value before adding it to the website via `document.cookie`.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-19 | Potential Path Traversal

Description

`generateImage`'s `baseImageFilePath` is attacker-controlled and may contain path traversal payloads, or paths with `../`, which allows the attacker to know if a file exists or not in the server file system.

domain-image.service.ts TYPESCRIPT

```typescript
async generateImage(
    registrationRequestDto: RegistrationRequestDtoV2,
  ): Promise<any> {
    const { domainName, expiryInEpoch, baseImageFileName, saltValue } =
      registrationRequestDto;

...

      const baseImage =
        baseImageFileName ?? (await this.pseudoRngService.choose());

...

      execFile(
        './generate_image',
        [
          `../../images/${baseImage}`,
          bgPath,
          footerPath,
          logoPath,
          domainName,
          formattedDate,
          filePath,
          salt.toString(),
        ],
```

Remediation

Add a check against path traversal payloads.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-20 | CORS Allows Localhost

Description

CORS is configured to allow `localhost` as the request origin to the backend server. This is acceptable in a development environment but must be removed when switching to a production environment.

Remediation

Remove `localhost` as an allowed origin in the production environment.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-21 | Image URL Not Validated

Description

`handleImageUpdateEvent` updates the database with information from an event emitted on-chain. However, this information, such as the image URL, is not checked if it is valid. This allows attackers to insert arbitrary values in these fields, potentially resulting in undesired vulnerabilities and exploits.

sui-subscriber.service.ts TYPESCRIPT

```typescript
handleImageUpdatedEvent = async (event: SuiEvent) => {
    const fields = event.parsedJson;
    const domainDto = {
      label: fields.domain_name.split('.')[0],
      node: fields.domain_name.split('.')[1],
      data: fields.data,
      url: fields.new_image,
      baseImageUrlUpdatePending: null,
    };
    const updatedAmount =
      await this.domainsService.upsertUpdatedImageDomainsInBulk([domainDto]);
    this.logger.verbose(updatedAmount);
    this.logger.debug(`Update ${updatedAmount} claimed domains`);
  };
}
```

One example of an undesired vulnerability is at `/domain/image/:ipfsHash`, where `ipfsHash` is checked if it exists in the database, and if so, utilizes it to build an image URL to Google storage.

domains.controller.ts TYPESCRIPT

```typescript
  @Get('image/:ipfsHash')
  async image(@Param('ipfsHash') ipfsHash: string): Promise<string> {
    const domain = await this.domainsService.findByIpfsHash(
      `ipfs://${ipfsHash}`,
    );
    if (!domain)
      throw new HttpException('Domain not recorded', HttpStatus.BAD_REQUEST);
    if (!domain.data)
      throw new HttpException(
        'Domain data not available',
        HttpStatus.BAD_REQUEST,
      );
    const decodedRegistrationRequestDto = this.domainCryptoService.decode(
      domain.data,
    );
    return await this.domainImageService.getImageUrl(
      ipfsHash,
      decodedRegistrationRequestDto,
    );
```

© 2023 Otter Audits LLC. All Rights Reserved.

```
  }
}
```

domain-image.service.ts TYPESCRIPT

```typescript
async getImageUrl(
    ipfsHash: string,
    registrationRequestDto: RegistrationRequestDtoV2,
  ): Promise<string> {
    let ipfsState = await this.ipfsStateService.find(ipfsHash);
    if (!ipfsState) {
      ipfsState = await this.ipfsStateService.insert(ipfsHash);
    }
    const fileName = `${ipfsHash}.png`;
    const cloudUrl = `https://storage.googleapis.com/suins-nft-images/${fileName}`;
...
```

If `ipfsHash` is not validated while inserted in the database, it may represent any arbitrary value in this function, which leads to path traversal while building `cloudUrl`.

Remediation

Validate if the URL is a valid IPFS URL before inserting it into the database by strictly validating the hash of the IPFS URL.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-22 | Improve Check For Second Highest Bid

Description

`auction.move` includes checks for the creation time of the highest bid to ensure that it falls within the appropriate range, but no such checks are included for the second highest bid.

source/auction/auction.move RUST

```rust
if (
    bid_detail.bid_value_mask < value
        || value < min_price
        || bid_detail.created_at_in_epoch < entry.started_at
        || entry.started_at + BIDDING_PERIOD <= bid_detail.created_at_in_epoch
) {
    // invalid bid
    // TODO: what to do now?
} else if (value > entry.highest_bid) {
    // Vickrey auction, winner pays the second highest_bid
    new_winning_bid(entry, bid_detail);
} else if (value == entry.highest_bid && bid_detail.created_at_in_ms <
    ↪  entry.winning_bid_created_at_in_ms) {
    new_winning_bid(entry, bid_detail);
} else if (value > entry.second_highest_bid) {
    // not winner, but affects second place
    new_second_highest_bid(entry, value, tx_context::sender(ctx));
} else {
    // bid does not affect auction
    // TODO: what to do now?
}
```

Remediation

Implement the same creation time check for the second highest bid as was applied to the creation time of the highest bid.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-23 | Rounding Error During Execution Of Code

Description

In `controller.move`, the formula currently utilized in `apply_discount_code` and `apply_referral_code` may produce incorrect values if the `original_fee` is not a multiple of one hundred. This may result in an inaccurate calculation of the new rate.

source/controller/controller.move RUST

```rust
fun apply_referral_code(
    config: &Configuration,
    payment: &mut Coin<SUI>,
    original_fee: u64,
    referral_code: &ascii::String,
    ctx: &mut TxContext
): u64 {
    let (rate, partner) = configuration::use_referral_code(config,
↪   referral_code);
    let remaining_fee = (original_fee / 100) * (100 - rate as u64);
    let payback_amount = original_fee - remaining_fee;
    coin_util::user_transfer_to_address(payment, payback_amount, partner, ctx);

    remaining_fee
}

// returns remaining_fee after being discounted
fun apply_discount_code(
    config: &mut Configuration,
    original_fee: u64,
    referral_code: &ascii::String,
    ctx: &mut TxContext,
): u64 {
    let rate = configuration::use_discount_code(config, referral_code, ctx);
    (original_fee / 100) * (100 - rate as u64)
}
```

Remediation

Change the formula for the rate calculation from `(original_fee / 100) * (100 - rate as u64)` to `(original_fee * (100 - rate as u64) / 100)`.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-24 | Payment Failure

Description

In `controller.move`, `register_internal` checks the user's coin value before applying discount and referral codes. This may lead to a program aborting if the valid payment exceeds the registration fee after applying the codes.

source/controller/controller.move RUST

```rust
fun register_internal(
    suins: &mut SuiNS,
    config: &mut Configuration,
    label: vector<u8>, // label has only 1 level
    owner: address,
    no_years: u64,
    secret: vector<u8>,
    payment: &mut Coin<SUI>,
    referral_code: Option<ascii::String>,
    discount_code: Option<ascii::String>,
    signature: vector<u8>,
    hashed_msg: vector<u8>,
    raw_msg: vector<u8>,
    clock: &Clock,
    ctx: &mut TxContext,
) {
    [...]
    assert!(coin::value(payment) >= registration_fee, ENotEnoughFee);

    // can apply both discount and referral codes at the same time
    if (option::is_some(&discount_code)) {
        registration_fee =
            apply_discount_code(config, registration_fee,
↪   option::borrow(&discount_code), ctx);
    };
    if (option::is_some(&referral_code)) {
        registration_fee =
            apply_referral_code(config, payment, registration_fee,
↪   option::borrow(&referral_code), ctx);
    };
    [...]
}
```

Remediation

Move the check for the value of the user's coin to after the application of discount and referral codes. This ensures that valid payments greater than the registration fee after the application of the codes are not aborted by the program.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-25 | Unnecessary Generation Of UID

Description

During `new_id` in `converter.move`, a new `UID` is generated and then deleted after the `new_id` creation. This step is unnecessary and may be replaced by invoking `tx_context::fresh_object_address`.

source/controller/controller.move RUST

```rust
public(friend) fun new_id(ctx: &mut TxContext): ID {
    let new_uid = object::new(ctx);
    let new_id = object::uid_to_inner(&new_uid);
    object::delete(new_uid);
    new_id
}
```

Remediation

Replace the step by invoking `tx_context::fresh_object_address` instead.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-26 | Incorrect Domain Validation

Description

`getDomain` incorrectly parses the domain, as any domain that ends in `sui.id` is valid according to the function:

helpers.ts TYPESCRIPT

```typescript
export const getDomain = (hostname: string): string => {
  if (!hostname.endsWith("sui.id")) return "";
  const domainTokens = hostname.split(".");
  if (domainTokens.length < 3) return "";
  // not allow move.sui.id
  if (
    domainTokens.length === 3 &&
    domainTokens[domainTokens.length - 3] === MOVE
  )
    return "";
  // we don't allow `move.sui.id`, so any domains with suffix `move.sui.id` have at
      ↪ least 4 tokens
  if (domainTokens[domainTokens.length - 3] === MOVE)
    return domainTokens.slice(0, -2).join(".");
  return domainTokens.slice(0, -1).join(".");
};
```

Remediation

Check if the domain ends with `.sui.id` or if it is equal `sui.id` to ensure that other domains ending with `sui.id` are invalid domains on this application.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-27 | Code Style And Documentation

Description

Consistent documentation and code style help prevent developer errors. Improvements may occur by:

1. Explicitly indicating time units in variables that store time durations, such as in `DEFAULT_DURATION`.
2. Correcting the incorrect comment on `name_record::has_expired`.

Remediation

Clarify time units in variables storing time durations and correct any inaccurate comments.

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-28 | Compiler Errors

Description

The program may abort during compilation if an empty comment (`/**/`) is present. This scenario must be appropriately handled to avoid program crashes.

Proof of Concept

PoC.move RUST

```rust
module test::my_module {
    #[test]
    fun test_test() {
        let _temp: u64 = 0xdeadbeef;
        /**/
    }
}
```

Note that the compiler crashed while attempting to compile this file.

Remediation

Properly handle the previously mentioned scenario to avoid crashes.

move-compiler/src/parser/lexer.rs DIFF

```diff
@@ -258,7 +258,9 @@ impl<'input> Lexer<'input> {
                        // Check if this is a documentation comment: '/**', but
    ↪   not '/***'.
                        // A documentation comment cannot be nested within another
    ↪   comment.
                        let is_doc =
-                           text.starts_with('*') && !text.starts_with("**") &&
    ↪   locs.is_empty();
+                           text.starts_with('*')
+                           && !(text.starts_with("**") || text.starts_with("*/"))
+                           && locs.is_empty();

                        locs.push((start, is_doc));
                    } else if text.starts_with("*/") {
```

© 2023 Otter Audits LLC. All Rights Reserved.

OS-SUI-SUG-29 | Code Maturity

Description

1. Use `has_access` in `kiosk::borrow` instead of directly comparing the `KioskOwnerCap` to align with the convention followed in the functions within the module ensuring code consistency.

sources/kiosk/kiosk_extension.move RUST

```rust
/// Immutably borrow an item from the `Kiosk`. Any item can be `borrow`ed
/// at any time.
public fun borrow<T: key + store>(
    self: &Kiosk, cap: &KioskOwnerCap, id: ID
): &T {
    assert!(object::id(self) == cap.for, ENotOwner);
    assert!(has_item(self, id), EItemNotFound);
    dof::borrow(&self.id, Item { id })
}
```

2. Hardcode permission bit indexes using constants like `PLACE_PERM = 0` and `LOCK_PERM = 1` to enhance code clarity and maintainability in `kiosk_extension` as shown below:

sources/kiosk/kiosk_extension.move RUST

```rust
public fun can_place<Ext: drop>(self: &Kiosk): bool {
    is_enabled<Ext>(self) && extension<Ext>(self).permissions & (1 <<
    ↪  PLACE_PERM) != 0
}

public fun can_lock<Ext: drop>(self: &Kiosk): bool {
    is_enabled<Ext>(self) && extension<Ext>(self).permissions & (1 <<
    ↪  LOCK_PERM) != 0
}
```

3. It may make sense to flag the ability to create multiple transfer policies for a given T as a security concern as this may result in bypassing the intended `TransferPolicy`.

Remediation

Implement the above-mentioned suggestions.

© 2023 Otter Audits LLC. All Rights Reserved.

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

© 2023 Otter Audits LLC. All Rights Reserved.

B | Procedure

As part of our standard auditing procedure, we split our analysis into two main sections: design and implementation.

When auditing the design of a program, we aim to ensure that the overall economic architecture is sound in the context of an on-chain program. In other words, there is no way to steal funds or deny service, ignoring any chain-specific quirks. This usually requires a deep understanding of the program's internal interactions, potential game theory implications, and general on-chain execution primitives.

One example of a design vulnerability would be an on-chain oracle that could be manipulated by flash loans or large deposits. Such a design would generally be unsound regardless of which chain the oracle is deployed on.

On the other hand, auditing the implementation of the program requires a deep understanding of the chain's execution model. While this varies from chain to chain, some common implementation vulnerabilities include reentrancy, account ownership issues, arithmetic overflows, and rounding bugs.

As a general rule of sum, implementation vulnerabilities tend to be more "checklist" style. In contrast, design vulnerabilities require a strong understanding of the underlying system and the various interactions: both with the user and cross-program.

As we approach any new target, we strive to get a comprehensive understanding of the program first. In our audits, we always approach targets with a team of auditors. This allows us to share thoughts and collaborate, picking up on details that the other missed.

While sometimes the line between design and implementation can be blurry, we hope this gives some insight into our auditing procedure and thought process.

© 2023 Otter Audits LLC. All Rights Reserved.