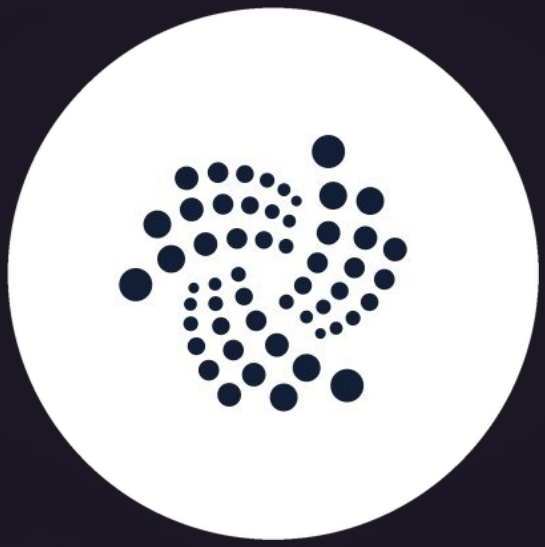
SHERLOCK

Security Review For IOTA



Collaborative Audit Prepared For: **IOTA**
Lead Security Expert(s): **ArmedGoose**
berndartmueller
g
sh1v
Date Audited: **January 20 - February 17, 2025**

Introduction

Bringing the real world to Web3 with a scalable, decentralized and programmable DLT infrastructure, IOTA is a asset-oriented programming model powered by Move. This security review focuses on chain migration and system contracts.

Scope

Repository: iotaledger/iota

Audited Commit: 46ed3db8563317fea51386d836b8896e4d4b6131

Files:

- crates/iota-config/src/genesis.rs
- crates/iota-config/src/migration_tx_data.rs
- crates/iota-framework/packages/iota-framework/Move.toml
- crates/iota-framework/packages/iota-framework/sources/address.move
- crates/iota-framework/packages/iota-framework/sources/authenticator_-state.move
- crates/iota-framework/packages/iota-framework/sources/bag.move
- crates/iota-framework/packages/iota-framework/sources/balance.move
- crates/iota-framework/packages/iota-framework/sources/bcs.move
- crates/iota-framework/packages/iota-framework/sources/borrow.move
- crates/iota-framework/packages/iota-framework/sources/clock.move
- crates/iota-framework/packages/iota-framework/sources/coin.move
- crates/iota-framework/packages/iota-framework/sources/coin_manager.move
- crates/iota-framework/packages/iota-framework/sources/config.move
- crates/iota-framework/packages/iota-framework/sources/crypto/bls12381.move
- crates/iota-framework/packages/iota-framework/sources/crypto/ecdsa_k1.move
- crates/iota-framework/packages/iota-framework/sources/crypto/ecdsa_r1.move
- crates/iota-framework/packages/iota-framework/sources/crypto/ecvrf.move
- crates/iota-framework/packages/iota-framework/sources/crypto/ed25519.move
- crates/iota-framework/packages/iota-framework/sources/crypto/groth16.move
- crates/iota-framework/packages/iota-framework/sources/crypto/group_-ops.move
- crates/iota-framework/packages/iota-framework/sources/crypto/hash.move

- crates/iota-framework/packages/iota-framework/sources/crypto/hmac.move
- crates/iota-framework/packages/iota-framework/sources/crypto/poseidon.move
- crates/iota-framework/packages/iota-framework/sources/crypto/vdf.move
- crates/iota-framework/packages/iota-framework/sources/crypto/zklogin_-verified_id.move
- crates/iota-framework/packages/iota-framework/sources/crypto/zklogin_-verified_issuer.move
- crates/iota-framework/packages/iota-framework/sources/deny_list.move
- crates/iota-framework/packages/iota-framework/sources/display.move
- crates/iota-framework/packages/iota-framework/sources/dynamic_field.move
- crates/iota-framework/packages/iota-framework/sources/dynamic_object_-field.move
- crates/iota-framework/packages/iota-framework/sources/event.move
- crates/iota-framework/packages/iota-framework/sources/hex.move
- crates/iota-framework/packages/iota-framework/sources/iota.move
- crates/iota-framework/packages/iota-framework/sources/kiosk/kiosk.move
- crates/iota-framework/packages/iota-framework/sources/kiosk/kiosk_-extension.move
- crates/iota-framework/packages/iota-framework/sources/kiosk/transfer_-policy.move
- crates/iota-framework/packages/iota-framework/sources/labeler.move
- crates/iota-framework/packages/iota-framework/sources/linked_table.move
- crates/iota-framework/packages/iota-framework/sources/object.move
- crates/iota-framework/packages/iota-framework/sources/object_bag.move
- crates/iota-framework/packages/iota-framework/sources/object_table.move
- crates/iota-framework/packages/iota-framework/sources/package.move
- crates/iota-framework/packages/iota-framework/sources/pay.move
- crates/iota-framework/packages/iota-framework/sources/priority_queue.move
- crates/iota-framework/packages/iota-framework/sources/prover.move
- crates/iota-framework/packages/iota-framework/sources/random.move
- crates/iota-framework/packages/iota-framework/sources/system_admin_-cap.move
- crates/iota-framework/packages/iota-framework/sources/table.move

- crates/iota-framework/packages/iota-framework/sources/table_vec.move
- crates/iota-framework/packages/iota-framework/sources/test/test_-scenario.move
- crates/iota-framework/packages/iota-framework/sources/test/test_utils.move
- crates/iota-framework/packages/iota-framework/sources/timelock.move
- crates/iota-framework/packages/iota-framework/sources/token.move
- crates/iota-framework/packages/iota-framework/sources/transfer.move
- crates/iota-framework/packages/iota-framework/sources/tx_context.move
- crates/iota-framework/packages/iota-framework/sources/types.move
- crates/iota-framework/packages/iota-framework/sources/url.move
- crates/iota-framework/packages/iota-framework/sources/vec_map.move
- crates/iota-framework/packages/iota-framework/sources/vec_set.move
- crates/iota-framework/packages/iota-framework/sources/versioned.move
- crates/iota-framework/packages/iota-framework/tests/address_tests.move
- crates/iota-framework/packages/iota-framework/tests/authenticator_state_-tests.move
- crates/iota-framework/packages/iota-framework/tests/bag_tests.move
- crates/iota-framework/packages/iota-framework/tests/balance_tests.move
- crates/iota-framework/packages/iota-framework/tests/bcs_tests.move
- crates/iota-framework/packages/iota-framework/tests/clock_tests.move
- crates/iota-framework/packages/iota-framework/tests/coin_manager_-tests.move
- crates/iota-framework/packages/iota-framework/tests/coin_tests.move
- crates/iota-framework/packages/iota-framework/tests/config_tests.move
- crates/iota-framework/packages/iota-framework/tests/crypto/bls12381_-tests.move
- crates/iota-framework/packages/iota-framework/tests/crypto/ecdsa_k1_-tests.move
- crates/iota-framework/packages/iota-framework/tests/crypto/ecdsa_r1_-tests.move
- crates/iota-framework/packages/iota-framework/tests/crypto/ecvrf_tests.move
- crates/iota-framework/packages/iota-framework/tests/crypto/ed25519_-tests.move

- crates/iota-framework/packages/iota-framework/tests/crypto/groth16_-tests.move
- crates/iota-framework/packages/iota-framework/tests/crypto/hash_tests.move
- crates/iota-framework/packages/iota-framework/tests/crypto/hmac_tests.move
- crates/iota-framework/packages/iota-framework/tests/crypto/poseidon_-tests.move
- crates/iota-framework/packages/iota-framework/tests/crypto/vdf_tests.move
- crates/iota-framework/packages/iota-framework/tests/crypto/zklogin_verified_-id_tests.move
- crates/iota-framework/packages/iota-framework/tests/crypto/zklogin_verified_-issuer_tests.move
- crates/iota-framework/packages/iota-framework/tests/dynamic_field_-tests.move
- crates/iota-framework/packages/iota-framework/tests/dynamic_object_field_-tests.move
- crates/iota-framework/packages/iota-framework/tests/event_tests.move
- crates/iota-framework/packages/iota-framework/tests/id_tests.move
- crates/iota-framework/packages/iota-framework/tests/iota_tests.move
- crates/iota-framework/packages/iota-framework/tests/kiosk/kiosk_borrow_-tests.move
- crates/iota-framework/packages/iota-framework/tests/kiosk/kiosk_extension_-tests.move
- crates/iota-framework/packages/iota-framework/tests/kiosk/kiosk_locked_-tests.move
- crates/iota-framework/packages/iota-framework/tests/kiosk/kiosk_test_-utils.move
- crates/iota-framework/packages/iota-framework/tests/kiosk/kiosk_tests.move
- crates/iota-framework/packages/iota-framework/tests/kiosk/policies/dummy_-policy.test.move
- crates/iota-framework/packages/iota-framework/tests/kiosk/policies/fixed_-commission_policy.test.move
- crates/iota-framework/packages/iota-framework/tests/kiosk/policies/item_-placed_policy.test.move
- crates/iota-framework/packages/iota-framework/tests/kiosk/policies/royalty_-policy.test.move

- crates/iota-framework/packages/iota-framework/tests/kiosk/policies/witness_-policy.test.move
- crates/iota-framework/packages/iota-framework/tests/kiosk/transfer_policy_-tests.move
- crates/iota-framework/packages/iota-framework/tests/linked_table_tests.move
- crates/iota-framework/packages/iota-framework/tests/object_bag_tests.move
- crates/iota-framework/packages/iota-framework/tests/object_table_tests.move
- crates/iota-framework/packages/iota-framework/tests/object_tests.move
- crates/iota-framework/packages/iota-framework/tests/package_tests.move
- crates/iota-framework/packages/iota-framework/tests/pay_tests.move
- crates/iota-framework/packages/iota-framework/tests/prover_tests.move
- crates/iota-framework/packages/iota-framework/tests/random_tests.move
- crates/iota-framework/packages/iota-framework/tests/table_tests.move
- crates/iota-framework/packages/iota-framework/tests/table_vec_tests.move
- crates/iota-framework/packages/iota-framework/tests/test_random.move
- crates/iota-framework/packages/iota-framework/tests/test_random_tests.move
- crates/iota-framework/packages/iota-framework/tests/test_scenario_tests.move
- crates/iota-framework/packages/iota-framework/tests/timelock/labeler_-test.move
- crates/iota-framework/packages/iota-framework/tests/timelock/test_label_-one.move
- crates/iota-framework/packages/iota-framework/tests/timelock/test_label_-two.move
- crates/iota-framework/packages/iota-framework/tests/timelock/timelock_-tests.move
- crates/iota-framework/packages/iota-framework/tests/timelock/timelocked_-balance_tests.move
- crates/iota-framework/packages/iota-framework/tests/token/token_actions_-tests.move
- crates/iota-framework/packages/iota-framework/tests/token/token_config_-tests.move
- crates/iota-framework/packages/iota-framework/tests/token/token_public_-actions_tests.move
- crates/iota-framework/packages/iota-framework/tests/token/token_request_-tests.move

- crates/iota-framework/packages/iota-framework/tests/token/token_test_utils.move
- crates/iota-framework/packages/iota-framework/tests/token/token_treasury_cap_tests.move
- crates/iota-framework/packages/iota-framework/tests/tx_context_tests.move
- crates/iota-framework/packages/iota-framework/tests/url_tests.move
- crates/iota-framework/packages/iota-framework/tests/vec_map_tests.move
- crates/iota-framework/packages/iota-framework/tests/vec_set_tests.move
- crates/iota-framework/packages/iota-framework/tests/verifier_tests.move
- crates/iota-framework/packages/iota-framework/tests/versioned_tests.move
- crates/iota-framework/packages/iota-system/sources/genesis.move
- crates/iota-framework/packages/iota-system/sources/iota_system.move
- crates/iota-framework/packages/iota-system/sources/iota_system_state_inner.move
- crates/iota-framework/packages/iota-system/sources/staking_pool.move
- crates/iota-framework/packages/iota-system/sources/storage_fund.move
- crates/iota-framework/packages/iota-system/sources/timelocked_staking.move
- crates/iota-framework/packages/iota-system/sources/validator.move
- crates/iota-framework/packages/iota-system/sources/validator_cap.move
- crates/iota-framework/packages/iota-system/sources/validator_set.move
- crates/iota-framework/packages/iota-system/sources/validator_wrapper.move
- crates/iota-framework/packages/iota-system/sources/voting_power.move
- crates/iota-framework/packages/iota-system/tests/delegation_tests.move
- crates/iota-framework/packages/iota-system/tests/governance_test_utils.move
- crates/iota-framework/packages/iota-system/tests/iota_system_tests.move
- crates/iota-framework/packages/iota-system/tests/rewards_distribution_tests.move
- crates/iota-framework/packages/iota-system/tests/timelocked_delegation_tests.move
- crates/iota-framework/packages/iota-system/tests/validator_set_tests.move
- crates/iota-framework/packages/iota-system/tests/validator_tests.move
- crates/iota-framework/packages/iota-system/tests/voting_power_tests.move
- crates/iota-framework/packages/stardust/sources/alias/alias.move

- crates/iota-framework/packages/stardust/sources/alias/alias_output.move
- crates/iota-framework/packages/stardust/sources/basic/basic_output.move
- crates/iota-framework/packages/stardust/sources/nft/irc27.move
- crates/iota-framework/packages/stardust/sources/nft/nft.move
- crates/iota-framework/packages/stardust/sources/nft/nft_output.move
- crates/iota-framework/packages/stardust/sources/stardust_upgrade_label.move
- crates/iota-framework/packages/stardust/sources/unlock_condition/address_-unlock_condition.move
- crates/iota-framework/packages/stardust/sources/unlock_condition/expiration_-unlock_condition.move
- crates/iota-framework/packages/stardust/sources/unlock_condition/storage_-deposit_return_unlock_condition.move
- crates/iota-framework/packages/stardust/sources/unlock_condition/timelock_-unlock_condition.move
- crates/iota-framework/packages/stardust/sources/utilities.move
- crates/iota-framework/packages/stardust/tests/alias_tests.move
- crates/iota-framework/packages/stardust/tests/basic_tests.move
- crates/iota-framework/packages/stardust/tests/nft_tests.move
- crates/iota-framework/packages/stardust/tests/unlock_condition/address_-unlock_condition_tests.move
- crates/iota-framework/src/lib.rs
- crates/iota-framework/tests/build-system-packages.rs
- crates/iota-genesis-builder/examples/build_and_compile_native_token.rs
- crates/iota-genesis-builder/examples/build_genesis_without_migration.rs
- crates/iota-genesis-builder/examples/build_stardust_genesis.rs
- crates/iota-genesis-builder/examples/build_stardust_genesis_from_s3.rs
- crates/iota-genesis-builder/examples/parse_hornet_genesis_snapshot.rs
- crates/iota-genesis-builder/examples/snapshot_add_test_outputs.rs
- crates/iota-genesis-builder/examples/snapshot_only_test_outputs.rs
- crates/iota-genesis-builder/src/genesis_build_effects.rs
- crates/iota-genesis-builder/src/lib.rs
- crates/iota-genesis-builder/src/main.rs
- crates/iota-genesis-builder/src/stake.rs

- crates/iota-genesis-builder/src/stardust/migration/executor.rs
- crates/iota-genesis-builder/src/stardust/migration/migration.rs
- crates/iota-genesis-builder/src/stardust/migration/migration_target_network.rs
- crates/iota-genesis-builder/src/stardust/migration/mod.rs
- crates/iota-genesis-builder/src/stardust/migration/tests/alias.rs
- crates/iota-genesis-builder/src/stardust/migration/tests/basic.rs
- crates/iota-genesis-builder/src/stardust/migration/tests/executor.rs
- crates/iota-genesis-builder/src/stardust/migration/tests/foundry.rs
- crates/iota-genesis-builder/src/stardust/migration/tests/mod.rs
- crates/iota-genesis-builder/src/stardust/migration/tests/nft.rs
- crates/iota-genesis-builder/src/stardust/migration/verification/alias.rs
- crates/iota-genesis-builder/src/stardust/migration/verification/basic.rs
- crates/iota-genesis-builder/src/stardust/migration/verification/created_-objects.rs
- crates/iota-genesis-builder/src/stardust/migration/verification/foundry.rs
- crates/iota-genesis-builder/src/stardust/migration/verification/mod.rs
- crates/iota-genesis-builder/src/stardust/migration/verification/nft.rs
- crates/iota-genesis-builder/src/stardust/migration/verification/util.rs
- crates/iota-genesis-builder/src/stardust/mod.rs
- crates/iota-genesis-builder/src/stardust/native_token/mod.rs
- crates/iota-genesis-builder/src/stardust/native_token/package_builder.rs
- crates/iota-genesis-builder/src/stardust/native_token/package_data.rs
- crates/iota-genesis-builder/src/stardust/native_token/package_-template/Move.toml
- crates/iota-genesis-builder/src/stardust/native_token/package_-template/sources/native_token_template.move
- crates/iota-genesis-builder/src/stardust/parse.rs
- crates/iota-genesis-builder/src/stardust/process_outputs.rs
- crates/iota-genesis-builder/src/stardust/test_outputs/alias_ownership.rs
- crates/iota-genesis-builder/src/stardust/test_outputs/delegator_outputs.rs
- crates/iota-genesis-builder/src/stardust/test_outputs/mod.rs
- crates/iota-genesis-builder/src/stardust/test_outputs/stardust_mix.rs

- crates/iota-genesis-builder/src/stardust/test_outputs/vesting_schedule_entity.rs
- crates/iota-genesis-builder/src/stardust/test_outputs/vesting_schedule_iota_-airdrop.rs
- crates/iota-genesis-builder/src/stardust/test_outputs/vesting_schedule_-portfolio_mix.rs
- crates/iota-genesis-builder/src/stardust/types/mod.rs
- crates/iota-genesis-builder/src/stardust/types/output_header.rs
- crates/iota-genesis-builder/src/stardust/types/output_index.rs
- crates/iota-genesis-builder/src/stardust/types/snapshot.rs
- crates/iota-genesis-builder/src/stardust/types/token_scheme.rs
- crates/iota-genesis-builder/src/validator_info.rs
- crates/iota-genesis-common/src/lib.rs
- crates/iota-types/src/stardust/address.rs
- crates/iota-types/src/stardust/coin_kind.rs
- crates/iota-types/src/stardust/coin_type.rs
- crates/iota-types/src/stardust/error.rs
- crates/iota-types/src/stardust/mod.rs
- crates/iota-types/src/stardust/output/alias.rs
- crates/iota-types/src/stardust/output/basic.rs
- crates/iota-types/src/stardust/output/foundry.rs
- crates/iota-types/src/stardust/output/mod.rs
- crates/iota-types/src/stardust/output/nft.rs
- crates/iota-types/src/stardust/output/unlock_conditions.rs
- crates/iota-types/src/timelock/label.rs
- crates/iota-types/src/timelock/mod.rs
- crates/iota-types/src/timelock/stardust_upgrade_label.rs
- crates/iota-types/src/timelock/timelock.rs
- crates/iota-types/src/timelock/timelocked_staked_iota.rs
- crates/iota-types/src/timelock/timelocked_staking.rs
- crates/iota/src/genesis_ceremony.rs

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
2	5	10

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue H-1: Excess Timelocked Iota are minted to Delegators when they have Timelocks for splitting

Source: https://github.com/sherlock-audit/2025-01-iota/issues/233

Summary

Any Delegator with staked Timelocks that are split will get minted more IOTA in Rebased than they hold in Stardust.

Vulnerability Detail

In Rebased Genesis, a Delegator can stake their IOTA. To do this, they must have enough migrated IOTA balance, whether Timelocked or not. The migrated objects used for staking will be marked for destruction except any surplus Timelocks.

Since the migrated Timelock is only split and a new Staked Timelock object is created for the allocation, the Delegator will get part of their Timelocked IOTA balance duplicated in Rebased.

Impact

Delegators will get part of their Timelocked IOTA balance duplicated in Rebased.

Code Snippet

https://github.com/sherlock-audit/2025-01-iota/blob/327c0d6462cc7d6a5284365a3086eecb463a0459/iota/crates/iota-genesis-builder/src/stake.rs#L386-L434 https://github.com/sherlock-audit/2025-01-iota/blob/327c0d6462cc7d6a5284365a3086eecb463a0459/iota/crates/iota-genesis-builder/src/lib.rs#L1654-L1664 https://github.com/sherlock-audit/2025-01-iota/blob/327c0d6462cc7d6a5284365a3086eecb463a0459/iota/crates/iota-framework/packages/iota-system/sources/genesis.move#L208-L225

Tool Used

Manual Review

Recommendation

Consider destroying the surplus Timelock and minting a new Timelock with the excess unstaked balance. This is the same approach used with the gas coins.

Discussion

miker83z

Fixed in https://github.com/iotaledger/iota/pull/5028

alexsporn

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

berndartmueller

Fix confirmed

gjaldon

This issue is fixed by deleting the original timelock objects after they have been split. This is done since a corresponding TimelockedStakedIota has already been created earlier.

ArmedGoose

Fix confirmed

Issue H-2: Time-locked vested reward output is misidentified by `is_timelocked_vested_reward()`, causing the migration process to fail

Source: https://github.com/sherlock-audit/2025-01-iota/issues/241

Summary

The migration process can be disrupted by creating time-locked vested reward outputs with more than two unlock conditions or native tokens, so that `is_timelocked_vested_reward()` wrongly considers them time-locked vested rewards and `try_from_stardust(..)` fails with `VestedRewardError::UnlockConditionsNumberMismatch`.

Vulnerability Detail

During `migrate_outputs(..)`, `is_timelocked_vested_reward()` is called to determine whether an output is a time-locked vested reward.

```
/// Checks if an output is a timelocked vested reward.
pub fn is_timelocked_vested_reward(
    output_id: OutputId,
    basic_output: &BasicOutput,
    target_milestone_timestamp_sec: u32,
) -> bool {
    is_vested_reward(output_id, basic_output)
        && basic_output
            .unlock_conditions()
            .is_time_locked(target_milestone_timestamp_sec)
}

/// Checks if an output is a vested reward, if it has a specific ID prefix,
/// and if it contains a timelock unlock condition.
pub fn is_vested_reward(output_id: OutputId, basic_output: &BasicOutput) -> bool {
    let has_vesting_prefix =
↪   output_id.to_string().starts_with(VESTED_REWARD_ID_PREFIX);

    has_vesting_prefix && basic_output.unlock_conditions().timelock().is_some()
}
```

If the output is a time-locked vested reward, `create_timelock_object()` converts the output to a timelock object via `try_from_stardust(..)`, which internally ensures that the output has exactly 2 unlock conditions and no native tokens:

```
if basic_output.unlock_conditions().len() != 2 {
```

```
    println!("Unlock conditions length: {:?}",
↪   basic_output.unlock_conditions().len());
    return Err(VestedRewardError::UnlockConditionsNumberMismatch);
}

if basic_output.native_tokens().len() > 0 {
    return Err(VestedRewardError::NativeTokensNotSupported);
}
```

However, those validations are missing in `is_timelocked_vested_reward()`, which is too lenient and allows outputs with more than 2 unlock conditions or native tokens to be considered time-locked vested rewards. This can be exploited to purposefully create outputs in Stardust that result in an error during migrations and disrupt the migration process.

Copy and paste the following test case to `crates/iota-types/src/unit_tests/timelock/timelock_tests.rs`. The test confirms that an output with more than two unlock conditions is wrongly considered a time-locked vested reward, and `try_from_stardust(..)` failing with `VestedRewardError::UnlockConditionsNumberMismatch`.

```
#[test]
fn timelock_from_stardust_with_native_tokens() {
    let output_id = OutputId::from_str(
        "0xb191c4bc825ac6983789e50545d5ef07a1d293a98ad974fc9498cb18123456780000",
    )
    .unwrap();

    let return_address = Ed25519Address::from(rand::random::<[u8;
↪   Ed25519Address::LENGTH]>());

    let output = BasicOutputBuilder::new_with_amount(10)
        .add_unlock_condition(AddressUnlockCondition::new(
            Ed25519Address::from_str(

↪   "0xebe40a263480190dcd7939447ee01aefa73d6f3cc33c90ef7bf905abf8728655",
            )
            .unwrap(),
        ))
        .add_unlock_condition(TimelockUnlockCondition::new(1000).unwrap())
        .add_unlock_condition(ExpirationUnlockCondition::new(return_address,
↪   1000).unwrap())
        .finish()
        .unwrap();

    assert!(is_timelocked_vested_reward(output_id, &output, 100));

    let err = try_from_stardust(output_id, &output, 100).unwrap_err();

    assert!(matches!(err, VestedRewardError::UnlockConditionsNumberMismatch));
```

```
}
```

Impact

The migration process can be disrupted.

Code Snippet

https://github.com/iotaledger/iota/blob/dd61d8a9fbb6f2c41934d375109e50b4620971e7/crates/iota-types/src/timelock/timelock.rs#L57-L66

Tool used

Manual Review

Recommendation

Consider also checking the number of unlock conditions and native tokens in `is_timelocked_vested_reward()` to prevent outputs with more than two unlock conditions or native tokens from being considered time-locked vested rewards.

Discussion

miker83z

Fixed in https://github.com/iotaledger/iota/pull/5485

alexsporn

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

berndartmueller

Fix confirmed

ArmedGoose

Fix confirmed

Issue M-1: Migration can be forced to fail by targeting parent verification

Source: https://github.com/sherlock-audit/2025-01-iota/issues/234

Summary

Any malicious user can create an Output with an Address Unlock Condition that is an *Alias Address type* but with an address pointing to a migrated NFT object in Rebased. This breaks the verify_parent() validation and causes the migration to fail.

Vulnerability Detail

The verify_parent() validation expects that an `AliasAddress` references an `Alias` object and an `NFTAddress` references an `NFT` object.

```
pub(super) fn verify_parent(
    output_id: &OutputId,
    address: &Address,
    storage: &InMemoryStorage,
) -> Result<()> {
    let object_id = ObjectID::from(stardust_to_iota_address(address)?);
    //  @audit fetch parent object from Storage
    let parent = storage.get_object(&object_id);
    match address {
        Address::Alias(address) => {
            if let Some(parent_obj) = parent {
                // @audit Parent object must be an Alias
                parent_obj
                    .to_rust::<Alias>()
                    .ok_or_else(|| anyhow!("invalid alias object for {address}"))?;
            }
        }
        Address::Nft(address) => {
            if let Some(parent_obj) = parent {
                // @audit Parent object must be an NFT
                parent_obj
                    .to_rust::<Nft>()
                    .ok_or_else(|| anyhow!("invalid nft object for {address}"))?;
            }
        }
```

In Stardust, an Output can be created with an Address Unlock Condition that points to an address of any Address Type. Given the above behavior, a malicious user can reference an NFT address but use an `AliasAddress` as the address type.

Impact

Any malicious user can force the migration to fail.

Code Snippet

https://github.com/sherlock-audit/2025-01-iota/blob/327c0d6462cc7d6a5284365a3086eecb463a0459/iota/crates/iota-genesis-builder/src/stardust/migration/verification/util.rs#L309-L343

Tool Used

Manual Review

Recommendation

Consider removing the `verify_parent()` validation.

Discussion

miker83z

Fixed in https://github.com/iotaledger/iota/pull/5428

alexsporn

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

gjaldon

`verify_parent()` is not removed, but it was modified to no longer raise an error when a parent object can not be found. It logs a warning instead.

berndartmueller

Fix confirmed

ArmedGoose

Fix confirmed

Issue M-2: Timelocked surplus coin is not fully utilized for allocations, resulting in a loss of time-locked IOTA balance for delegators

Source: https://github.com/sherlock-audit/2025-01-iota/issues/238

Summary

If the current surplus coin is large enough to cover the allocation's target amount, the remaining surplus will not be used for an allocation and results in the delegator losing part of their time-locked IOTA balance.

Vulnerability Detail

If in `pick_objects_for_allocation()` a current surplus coin sufficiently covers the required target amount`target_amount_nanos`, so that the `if` block in lines 386-425 is skipped, the `previous_surplus_coin` pointer will be set to the default `surplus_coin` (i.e., `SurplusCoin::default()`).

Consequently, if the surplus coin still had a non-zero amount remaining, it will not be used in the next validator allocation iteration. Assuming that surplus coins are always deleted (notably, this is currently not the case, as reported in https://github.com/sherlock-audit/2025-01-iota/issues/233), the remaining surplus will be lost.

Impact

Delegators will lose a part of their time-locked IOTA balance.

Code Snippet

https://github.com/iotaledger/iota/blob/46ed3db8563317fea51386d836b8896e4d4b6131/crates/iota-genesis-builder/src/stake.rs#L428

Tool used

Manual Review

Recommendation

Consider not overwriting the `previous_surplus_coin` pointer in line 428 if the surplus coin is not fully used in the current iteration so that it will be used in the next iteration.

Discussion

miker83z

Fixed in https://github.com/iotaledger/iota/pull/5028

alexsporn

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

berndartmueller

Fix confirmed

ArmedGoose

Fix confirmed

Issue M-3: `SwapSplitIterator` incorrectly clones native tokens causing the migration to fail

Source: https://github.com/sherlock-audit/2025-01-iota/issues/244

Summary

Splitting outputs with the custom `SwapSplitIterator` iterator will incorrectly clone native tokens, leading to a migration error.

Vulnerability Detail

The custom `SwapSplitIterator` iterator processes Stardust outputs, splits the output's amount in different outputs given the targets indicated for the destinations, and swaps the address unlock condition to be the destination address one, based on the given `AddressSwapSplitMap` map.

However, creating the new basic output in `swap_split_operation()` via `BasicOutputBuilder::from(basic_output)` will also copy the `native_tokens` to the new output. This results in cloned native tokens, leading to a migration error due to a mismatch between the native token's foundry minting amount and the inflated amount from cloning.

Impact

The migration fails, and outputs with native tokens cannot be split with the custom `SwapSplitIterator` iterator.

Code Snippet

https://github.com/iotaledger/iota/blob/46ed3db8563317fea51386d836b8896e4d4b6131/crates/iota-genesis-builder/src/stardust/process_outputs.rs#L680-L687

```
split_outputs.push(
    BasicOutputBuilder::from(basic_output)
        .with_amount(swap_split_amount)
        .replace_unlock_condition(AddressUnlockCondition::new(Ed25519Address::new(
            destination.to_inner(),
        )))
        .finish()
        .expect("failed to create basic output during split")
        .into(),
);
```

Tool used

Manual Review

Recommendation

Consider filtering out Stardust outputs that contain native tokens.

Discussion

miker83z

Fixed in https://github.com/iotaledger/iota/pull/5519

alexsporn

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

berndartmueller

Fix confirmed

ArmedGoose

Fix confirmed

Issue M-4: Users Unable to Withdraw Staked Funds from Validator Candidates

Source: https://github.com/sherlock-audit/2025-01-iota/issues/246

Summary

Users are unable to withdraw their staked funds from validator candidates due to an issue with how stake activation and withdrawal are handled. The protocol processes stakes immediately for pre-active validators, causing withdrawal attempts to reference a non-existent `pending_stake`.

Vulnerability Detail

- When a user stakes funds to a validator candidate, the stake is immediately added to the validator's stake pool if the validator is pre-active.
- This means that there is no `pending_stake` entry for the stake, as it has already been moved to the validator's active stake pool.
- When the user attempts to withdraw their stake in the same epoch, the protocol checks whether the stakedAsset activation epoch is greater than the current epoch.
- Since this condition holds true for stakes made in the current epoch, the system incorrectly attempts to withdraw from `pending_stake`, which does not exist.

Impact

Users who stake funds with pre-active validators are unable to withdraw their stake during the same epoch, leading to potential fund lockup until the next stake activation epoch is reached.

Code Snippet

```
if (staked_iota.stake_activation_epoch > ctx.epoch()) {
    let principal = unwrap_staked_iota(staked_iota);
    pool.pending_stake = pool.pending_stake - principal.value();

    return principal
};
```

Tool Used

Manual Review

POC

```
#[test]
#[expected_failure]
fun test_stake_withdraw_from_validator_candidate() {
    use std::debug;

    let mut scenario = test_scenario::begin(@0x0);

    set_up_iota_system_state(vector[@0x1, @0x2, @0x3]);
    let mut system_state = scenario.take_shared<IotaSystemState>();

    // add a new validator
    scenario.next_tx(@0x69);
    iota_system::iota_system::request_add_validator_candidate(
        &mut system_state,
        x"0A",
        x"0B",
        x"0C",
        x"0D",
        b"Test Validator",
        b"A test validator for Sui",
        b"https://example.com/image.png",
        b"https://example.com",
        b"/ip4/127.0.0.2/tcp/80",
        b"/ip4/127.0.0.3/udp/80",
        b"/ip4/127.0.0.4/udp/80",
        100_000,
        20_00,
        scenario.ctx()
    );

    // verify validator creation
    let val = iota_system::iota_system::candidate_validator_by_address(&mut
↪   system_state, @0x69);

    // stake with validator
    let c = iota::coin::mint_for_testing<IOTA>(50 * 100_000_000, scenario.ctx());
    scenario.next_tx(@0x99);
    iota_system::iota_system::request_add_stake(&mut system_state, c, @0x69,
↪   scenario.ctx());

    // validate stake
```

```
    let val = iota_system::iota_system::candidate_validator_by_address(&mut
↪   system_state, @0x69);
    // debug::print(val);

    // try to withdraw the stake?
    scenario.next_tx(@0x99);
    let coin_ids = scenario.ids_for_sender<StakedIota>();
    let stake = scenario.take_from_sender_by_id<StakedIota>(coin_ids[0]);

    // fails to withdraw the stake
    iota_system::iota_system::request_withdraw_stake(&mut system_state, stake,
↪   scenario.ctx());

    test_scenario::return_shared(system_state);
    // debug::print(&9000);
    scenario.end();
}
```

Recommendation

Modify the withdrawal logic to correctly identify and handle pre-active validators by ensuring that withdrawals reference the active stake pool instead of `pending_stake` when applicable. Implement a condition that differentiates between pre-active and non-pre-active validators to prevent invalid withdrawal attempts.

Discussion

alexsporn

Final commit hash on `develop` including fixes:
`5d86d45197849842100c1b34245ace3268bc01f6`

berndartmueller

Fix confirmed

ArmedGoose

Fix confirmed

sh15h4nk

Fix confirmed

Issue M-5: Validator Re-Activation Without Proper Checks

Source: https://github.com/sherlock-audit/2025-01-iota/issues/247

Summary

A removed validator can rejoin the active validator set without restrictions, potentially leading to unfair staking scenarios where previously staked users do not earn rewards despite the validator becoming active again.

Vulnerability Detail

- When a validator is removed from the active set, they are moved to `inactive_validators` to allow users to withdraw their stake.
- However, there is no check to prevent a validator from rejoining the active set simply by meeting the minimum stake requirement.
- This allows the validator to become active again while users who had previously staked with them remain in an inactive state, unable to accrue rewards.
- While users can still withdraw their stake, they miss out on rewards if the validator regains active status.

Impact

Users who staked with a validator before removal do not earn rewards when the validator becomes active again, potentially creating unfair staking scenarios.

Code Snippet

```
public(package) fun request_add_validator_candidate(
        self: &mut ValidatorSetV1,
        validator: ValidatorV1,
        ctx: &mut TxContext,
    ) {
        // The next assertions are not critical for the protocol, but they are here
↪  to catch problematic configs earlier.
        assert!(
            !is_duplicate_with_active_validator(self, &validator)
                && !is_duplicate_with_pending_validator(self, &validator),
            EDuplicateValidator
        );
        let validator_address = iota_address(&validator);
```

```
        assert!(
            !self.validator_candidates.contains(validator_address),
            EAlreadyValidatorCandidate
        );

        assert!(validator.is_preactive(), EValidatorNotCandidate);
        // Add validator to the candidates mapping and the pool id mappings so that
↪   users can start
        // staking with this candidate.
        self.staking_pool_mappings.add(staking_pool_id(&validator),
↪   validator_address);
        self.validator_candidates.add(
            iota_address(&validator),
            validator_wrapper::create_v1(validator, ctx),
        );
    }
```

POC

```
#[test]
fun test_re_add_validator() {
    use std::debug;

    // setup
    let mut scenario = test_scenario::begin(@0x0);

    set_up_iota_system_state(vector[@0x1, @0x2, @0x3, @0x4]);
    let mut system_state = scenario.take_shared<IotaSystemState>();

    // register a validator
    scenario.next_tx(@0x69);
    iota_system::iota_system::request_add_validator_candidate(
        &mut system_state,
        x"2A",
        x"2B",
        x"2C",
        x"2D",
        b"Test Validator",
        b"A test validator for Sui",
        b"https://example.com/image.png",
        b"https://example.com",
        b"/ip4/127.0.0.3/tcp/80",
        b"/ip4/127.0.0.2/udp/80",
        b"/ip4/127.0.0.4/udp/80",
        100_000,
        20_00,
        scenario.ctx()
    );
```

```
    // add min stake
    let c = iota::coin::mint_for_testing<IOTA>(2 * 1_000_000_000, scenario.ctx());
    scenario.next_tx(@0x99);
    iota_system::iota_system::request_add_stake(&mut system_state, c, @0x69,
↪   scenario.ctx());

    // accept validator
    scenario.next_tx(@0x69);
    iota_system::iota_system::request_add_validator(&mut system_state,
↪   scenario.ctx());

    // advance epoch to add the validator to active validator
    test_scenario::return_shared(system_state);
    advance_epoch(&mut scenario);
    let mut system_state = scenario.take_shared<IotaSystemState>();

    // get active validators
    let act_vals = iota_system::iota_system::active_validator_addresses(&mut
↪   system_state);
    debug::print(&act_vals);
    debug::print(&iota_system::iota_system::validator_stake_amount(&mut
↪   system_state, @0x69));

    // remove request validator
    scenario.next_tx(@0x69);
    iota_system::iota_system::request_remove_validator(&mut system_state,
↪   scenario.ctx());

    // advance epoch to make the validator invalid
    test_scenario::return_shared(system_state);
    advance_epoch(&mut scenario);
    let mut system_state = scenario.take_shared<IotaSystemState>();


    // get active validators
    let act_vals = iota_system::iota_system::active_validator_addresses(&mut
↪   system_state);
    debug::print(&act_vals);

    // try registering validator with the same address again.
    scenario.next_tx(@0x69);
    iota_system::iota_system::request_add_validator_candidate(
        &mut system_state,
        x"2A",
        x"2B",
        x"2C",
        x"2D",
        b"Test Validator",
        b"A test validator for Sui",
```

```
        b"https://example.com/image.png",
        b"https://example.com",
        b"/ip4/127.0.0.3/tcp/80",
        b"/ip4/127.0.0.2/udp/80",
        b"/ip4/127.0.0.4/udp/80",
        100_000,
        20_00,
        scenario.ctx()
    );

    // add min stake
    let c = iota::coin::mint_for_testing<IOTA>(5 * 1_000_000_000, scenario.ctx());
    scenario.next_tx(@0x99);
    iota_system::iota_system::request_add_stake(&mut system_state, c, @0x69,
↪   scenario.ctx());

    // accept the validator
    scenario.next_tx(@0x69);
    iota_system::iota_system::request_add_validator(&mut system_state,
↪   scenario.ctx());

    test_scenario::return_shared(system_state);
    advance_epoch(&mut scenario);
    let mut system_state = scenario.take_shared<IotaSystemState>();

    let act_vals = iota_system::iota_system::active_validator_addresses(&mut
↪   system_state);
    debug::print(&act_vals);
    debug::print(&iota_system::iota_system::validator_stake_amount(&mut
↪   system_state, @0x69));

    test_scenario::return_shared(system_state);
    scenario.end();
}
```

Tool Used

Manual Review

Recommendation

Introduce a check when adding a validator to the active set to ensure they are not in the inactive_validators table. If a validator was previously removed, require an explicit reactivation process to fairly handle previously staked users and ensure reward distribution aligns with validator status.

Discussion

alexsporn

Won't fix because there are no funds at risk. The staked funds are not connected to the validator address but to the staking pool which remains accessible for all users that staked, no matter if a new validator exists at the same address or not. Since the `StakedIota` objects link to a pool and are owned by the users we cannot simply move them over to the new pool.

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

Issue L-1: Any Treasury Outputs present in the snapshot will cause migration to fail

Source: https://github.com/sherlock-audit/2025-01-iota/issues/235

Summary

Every Stardust output is checked to have a corresponding created object in Rebased during migration. There are no created objects for Treasury Outputs, so this check will fail for Treasury Outputs.

Vulnerability Detail

This check will fail if there are any `Output::Treasury` in `outputs`, causing migration to fail. This happens because:

1. The snapshot parser unpacks all types of outputs including TreasuryOutput.
2. When the outputs are processed for migration, the Treasury Outputs are not filtered out.
3. When outputs are migrated, the Treasury Outputs are skipped and not included in the `migration.output_objects_map`.
4. After the outputs are migrated, the ledger state is verified. This is what calls `verify_outputs()`. At this point, there are Treasury outputs in `outputs`, but there are no Treasury Outputs in the `output_objects_map`.

Impact

This can cause migration to fail.

Note: This is a Low because there are currently no Treasury Outputs in the Ledger to be migrated, and users can no longer create them in Stardust.

Code Snippet

https://github.com/sherlock-audit/2025-01-iota/blob/327c0d6462cc7d6a5284365a3086eecb463a0459/iota/crates/iota-genesis-builder/src/stardust/parse.rs#L41
https://github.com/sherlock-audit/2025-01-iota/blob/327c0d6462cc7d6a5284365a3086eecb463a0459/iota/crates/iota-genesis-builder/src/stardust/migration/verification/mod.rs#L37-L39

Tool Used

Manual Review

Recommendation

Consider filtering out the Treasury Outputs here.

Discussion

miker83z

Fixed in https://github.com/iotaledger/iota/pull/5444

alexsporn

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

gjaldon

The fix is applied as recommended, where the treasury outputs are filtered out when verifying the outputs.

berndartmueller

Fix confirmed

ArmedGoose

Fix confirmed

Issue L-2: Hardcoded protocol parameters are used when unpacking Outputs from the Snapshot

Source: https://github.com/sherlock-audit/2025-01-iota/issues/236

Summary

Hardcoded protocol parameters are used when unpacking Outputs from the Snapshot.

Vulnerability Detail

```
Output::unpack::<_, true>(&mut self.reader, &ProtocolParameters::default())?,
```

Instead of fetching the protocol parameters from the `SnapshotParser` object, the default protocol parameters are used.

Impact

The token supply used for verification may be incorrect.

Code Snippet

https://github.com/sherlock-audit/2025-01-iota/blob/327c0d6462cc7d6a5284365a3086eecb463a0459/iota/crates/iota-genesis-builder/src/stardust/parse.rs#L41

Tool Used

Manual Review

Recommendation

Consider fetching the protocol parameters from the `SnapshotParser` object, like in total_supply().

Discussion

miker83z

Fixed in https://github.com/iotaledger/iota/pull/5465

alexsporn

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

gjaldon

The issue is addressed by extracting a protocol_parameters() method that loads the parameters from the snapshot.

berndartmueller

Fix confirmed

ArmedGoose

Fix confirmed

Issue L-3: Unnecessary mutable reference requirement restricts a view-only function from being accessed

Source: https://github.com/sherlock-audit/2025-01-iota/issues/237

Summary

The `additional_metadata` function requires passing a mutable reference to `CoinManager`as an argument despite being a read-only operation. This creates an unnecessary access restriction since blockchain data is publicly viewable.

Vulnerability Detail

The function in `iota/crates/iota-framework/packages/iota-framework/sources/coin_manager.move:216` requires `&mut CoinManager<T>` as an argument despite only performing read operations. This forces callers to be the owner of the `CoinManager` instance to access the metadata, which contradicts the public nature of blockchain data. The mutable reference requirement serves no security purpose as the function does not modify any state.

Impact

This is assessed as Info/Low severity because no current Iota modules depend on this function, and the restriction does not compromise security. The impact is limited to inconvenience for future integrations, as public blockchain data remains readable through other means regardless of this restriction.

Code Snippet

Tool Used

Manual Review

Recommendation

Change the function signature to use an immutable reference instead of a mutable one.

Discussion

miker83z

Fixed in https://github.com/iotaledger/iota/pull/5447

alexsporn

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

berndartmueller

Fix confirmed

ArmedGoose

Fix confirmed

Issue L-4: Add additional checks to `verify_basic_output()` to ensure only a `TimeLock<Balance>` has been created during migrations

Source: https://github.com/sherlock-audit/2025-01-iota/issues/239

Summary

`verify_basic_output()` lacks checks to ensure only `TimeLock<Balance>` has been created and put into `created_objects`.

Vulnerability Detail

`verify_basic_output()` verifies if the created basic output is valid. However, if the basic output is a time-locked vested reward, it is recommended to add additional checks to ensure that the other `created_objects` are empty, e.g.:

```
ensure!(
        created_objects.native_token_coin().is_err(),
        "unexpected native token coin found"
    );

    ensure!(
        created_objects.coin_manager().is_err(),
        "unexpected coin manager found"
    );

    ensure!(
        created_objects.coin_manager_treasury_cap().is_err(),
        "unexpected coin manager cap found"
    );

    ensure!(
        created_objects.package().is_err(),
        "unexpected package found"
    );
```

Impact

Missing validation unable to detect whether unexpected `created_objects` have been created as part of the migration.

Code Snippet

https://github.com/iotaledger/iota/blob/46ed3db8563317fea51386d836b8896e4d4b6131/crates/iota-genesis-builder/src/stardust/migration/verification/basic.rs#L48-L93

Tool Used

Manual Review

Recommendation

Consider adding additional checks, such as the ones listed above.

Discussion

miker83z

Fixed in https://github.com/iotaledger/iota/pull/5453

alexsporn

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

berndartmueller

Fix confirmed

ArmedGoose

Fix confirmed

Issue L-5: Outdated function comment for `attach_nft()` incorrectly mentioning `Alias` and `AliasOutput` in the context of an NFT output

Source: https://github.com/sherlock-audit/2025-01-iota/issues/240

Summary

Outdated function comment for `attach_nft()` incorrectly mentioning `Alias` and `AliasOutput` in the context of an NFT output.

Vulnerability Detail

Outdated function comment for `attach_nft()` incorrectly mentioning `Alias` and `AliasOutput` in the context of an NFT output.

Impact

Outdated function comment.

Code Snippet

https://github.com/iotaledger/iota/blob/46ed3db8563317fea51386d836b8896e4d4b6131/crates/iota-framework/packages/stardust/sources/nft/nft_output.move#L87

Tool Used

Manual Review

Recommendation

Consider updating the comment to mention `Nft` and `NftOutput` instead of referring to alias objects.

Discussion

miker83z

Fixed in https://github.com/iotaledger/iota/pull/5505

alexsporn

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

berndartmueller

Fix confirmed

ArmedGoose

Fix confirmed

Issue L-6: No validation that all Timelock Staked IOTA objects have been allocated

Source: https://github.com/sherlock-audit/2025-01-iota/issues/242

Summary

Gas and staked IOTA objects are checked that they are all allocated. However, this has not been checked for Timelock Staked IOTA objects.

Vulnerability Detail

```
assert!(gas_objects.is_empty());
assert!(staked_iota_objects.is_empty());
```

All Timelock Staked IOTA objects should be allocated.

Impact

There is no direct impact.

Code Snippet

https://github.com/sherlock-audit/2025-01-iota/blob/327c0d6462cc7d6a5284365a3086eecb463a0459/iota/crates/iota-genesis-builder/src/lib.rs#L756-L759

Tool Used

Manual Review

Recommendation

Consider adding the following validation:

```
assert!(timelock_staked_iota_objects.is_empty());
```

Discussion

miker83z

Fixed in https://github.com/iotaledger/iota/pull/5515

alexsporn

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

gjaldon

The issue has been addressed with this change.

berndartmueller

Fix confirmed

ArmedGoose

Fix confirmed

Issue L-7: `UnlockedVestingIterator` discards `native_tokens` when creating a new basic output from unlocked vesting outputs

Source: https://github.com/sherlock-audit/2025-01-iota/issues/243

Summary

The `UnlockedVestingIterator` custom iterator discards the output's `native_tokens` when creating a new basic output with the aggregated balance from the unlocked vesting outputs.

Vulnerability Detail

The `UnlockedVestingIterator` custom iterator iterates all outputs, looks for vesting outputs that can be unlocked, and stores them during the iteration. At the end of the iteration, all vesting outputs owned by a single address are merged into a unique basic output.

However, when creating the new basic output with the aggregated balance from the unlocked vesting outputs via `BasicOutputBuilder::new_with_amount`, the `native_tokens` are not considered and are discarded. Given that the vested outputs have been initially created by the Iota team via a dedicated tool, it can be assumed that such outputs do not contain any `native_tokens`, lowering the severity of this issue.

Impact

Vested outputs that contain `native_tokens` will have them discarded.

Code Snippet

https://github.com/iotaledger/iota/blob/46ed3db8563317fea51386d836b8896e4d4b6131/crates/iota-genesis-builder/src/stardust/process_outputs.rs#L517-L522

```
// create a new basic output which holds the aggregated balance from
// unlocked vesting outputs for this address
let basic = BasicOutputBuilder::new_with_amount(output_header_with_balance.balance)
    .add_unlock_condition(AddressUnlockCondition::new(address))
    .finish()
    .expect("failed to create basic output");
```

Tool used

Manual Review

Recommendation

Consider checking whether a vested output contains `native_tokens` and error out if it does.

Discussion

miker83z

This issue is fixed while fixing #241. In fact, in #241 the method `is_vested_reward` is modified to check that the basic output does not contain native tokens. Since `is_vested_reward` is used as a condition to filter the outputs used for the aggregation in `UnlockedVestingIterator`, then outputs with native tokens will be filtered out. The issue #241 is fixed in https://github.com/iotaledger/iota/pull/5485

alexsporn

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

berndartmueller

Fix confirmed

ArmedGoose

Fix confirmed

Issue L-8: Invalid CSV Header Check for AddressSwapMap

Source: https://github.com/sherlock-audit/2025-01-iota/issues/245

Summary

The header validation logic does not correctly enforce the expected conditions. The current implementation only fails if both headers do not match, instead of failing when any one header is incorrect.

Vulnerability Detail

The condition:

```
if &headers[0] != LEFT_HEADER && &headers[1] != RIGHT_HEADER
```

incorrectly checks the headers because the && operator requires both conditions to be false for the validation to fail. This means that if only one header is incorrect, the check will pass incorrectly. The correct logic should use || instead of && to ensure failure if either header does not match.

Impact

This could lead to accepting invalid CSV headers, potentially causing incorrect processing of data downstream.

Tool Used

Manual Review

Recommendation

Replace && with || to ensure that the check fails if any one of the headers is incorrect

Discussion

miker83z

Fixed in https://github.com/iotaledger/iota/pull/5494

alexsporn

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

berndartmueller

Fix confirmed

ArmedGoose

Fix confirmed

sh15h4nk

Fix confirmed

Issue L-9: Incorrect Gas Price Update for Pending Validators

Source: https://github.com/sherlock-audit/2025-01-iota/issues/248

Summary

The function `request_set_gas_price` allows both active and pending validators to submit a new gas price quote. However, since the current gas price is not updated for pending validators, they may end up using an outdated gas price when transitioning to the active state. This could cause inconsistencies in transaction fee calculations.

Vulnerability Detail

- The function verifies the validator using:

```
let verified_cap = self.validators.verify_cap(cap,
↪  ACTIVE_OR_PENDING_VALIDATOR);
```

  This means both active and pending validators can call the function.
- However, for pending validators, the current gas price remains unchanged, and only next_epoch_gas_price is updated.
- When a pending validator transitions to an active state, it should use next_epoch_gas_price, but instead, it incorrectly references current_gas_price, which remains outdated.
- The correct behavior would be to restrict gas price updates to active validators only or ensure that pending validators update `current_gas_price` appropriately.

Impact

Pending validators may start with an incorrect gas price upon activation, potentially affecting transaction fee calculations and validator rewards.

Code Snippet

```
/// A validator can call this function to submit a new gas price quote, to be
/// used for the reference gas price calculation at the end of the epoch.
public(package) fun request_set_gas_price(
    self: &mut IotaSystemStateV1,
    cap: &UnverifiedValidatorOperationCap,
    new_gas_price: u64,
) {
```

```
    // Verify the represented address is an active or pending validator, and the
↪   capability is still valid.
    let verified_cap = self.validators.verify_cap(cap, ACTIVE_OR_PENDING_VALIDATOR);
    let validator =
↪   self.validators.get_validator_mut_with_verified_cap(&verified_cap, false /*
↪   include_candidate */);

    validator.request_set_gas_price(verified_cap, new_gas_price);
}
```

Tool Used

Manual Review

Recommendation

Restrict gas price updates to active validators only Alternatively, if pending validators should be allowed to update gas prices, ensure that their current_gas_price is updated upon calling request_set_gas_price.

Discussion

alexsporn

Won't fix because the next protocol upgrade moves from validator-defined gas prices to protocol-defined gas prices. The aforementioned gas price survey will be removed from the protocol.

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

Issue L-10: Scenario: Potential Validator Manipulation of Gas Price

Source: https://github.com/sherlock-audit/2025-01-iota/issues/249

Summary

A validator can strategically set their gas price quote to influence the reference gas price calculation. By analyzing the voting power distribution and gas prices of other validators, an attacker can position their quote within a specific range to either increase or decrease the final gas price.

Vulnerability Detail

- Validators with higher gas prices contribute their voting power to the reference gas price calculation.
- A validator can compute the total voting power and determine the gas price range (x, y) where:
  - x is the gas price of the last validator contributing to 33% of voting power.
  - y is the next validator with the highest gas price.
- If there is a large gap between x and y, the attacker can choose a quote that significantly raises or lowers the gas price.
- This can lead to inefficient gas pricing, favoring validators who benefit from a specific price range.

Impact

The attacker can influence transaction costs by artificially inflating or deflating the gas price.

Code Snippet

Tool Used

Manual Review

Discussion

alexsporn

Won't fix because the next protocol upgrade moves from validator-defined gas prices to protocol-defined gas prices. The aforementioned gas price survey will be removed from the protocol.

Final commit hash on `develop` including fixes:
`1a5e629afa830e7fcd7c6168b0c91b3d1eb45b9c`

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.