CERTIK

Security Assessment

Sienna AMM v

Jun 29th, 2021

Table of Contents

Summary

This report has been prepared for Sienna AMM v smart contracts, to discover issues and vulnerabilities in the source code of their Cosmwasm implementation. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

Issues ranged from medium to informational.

Multiple implementation issues were identified throughout the `shared`, `lp-token`, `factory`, and `exchange` crates which do not follow typical Rust idioms and hinder either the overall performance or readability of the code in question. More commonly identified were manual implementations of the functionality provided by the Rust standard library such as enumeration, usage of extra references, and closures were unnecessary such as through the use of the `ok_or_else` function over the `ok_or` function, unnecessary value cloning leading to extra memory usage.

Finally, the auditing team discussed all issues with the team for clarifications. The team provided a new release that alleviated all of the issues, some with remove/refactoring and others with code fixes, in commits leading up to release amm-1.0.0.

Overview

Project Summary

Project Name	Sienna AMM v
Platform	CosmosSDK
Language	Rust
Codebase	
Commit	0b2b7efce4a82227deb0327f79e3124151337810

Audit Summary

Delivery Date	Jun 29, 2021
Audit Methodology	Manual Review, Static Analysis
Key Components	

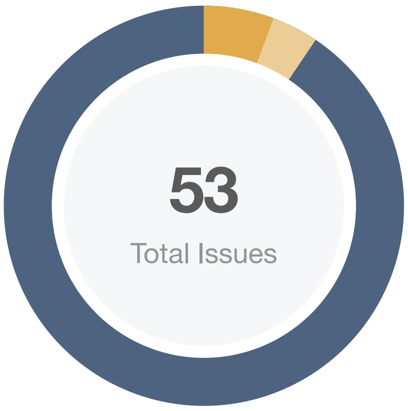
Vulnerability Summary

Vulnerability Level	Total	Pending	Partially Resolved	Resolved	Acknowledged	Declined
● Critical	0	0	0	0	0	0
● Major	0	0	0	0	0	0
● Medium	3	0	0	3	0	0
● Minor	2	0	0	2	0	0
● Informational	48	0	0	47	1	0
● Discussion	0	0	0	0	0	0

Audit Scope

ID	file	SHA256 Checksum

Findings



🟥	**Critical**	**0** (0.00%)
🟧	**Major**	**0** (0.00%)
🟨	**Medium**	**3** (5.66%)
🟨	**Minor**	**2** (3.77%)
🟦	**Informational**	**48** (90.57%)
🟩	**Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
SKE-01	Redundant Clone	Language Specific	● Informational	✓ Resolved
SKE-02	Redundant Return	Language Specific	● Informational	✓ Resolved
SKO-01	Code Structure	Language Specific	● Informational	✓ Resolved
SKO-02	Code Usability	Language Specific	● Informational	✓ Resolved
SKO-03	Proper Representation	Language Specific	● Informational	✓ Resolved
SKO-04	Proper Representation	Language Specific	● Informational	✓ Resolved
SKR-01	Code Structure	Language Specific	● Informational	✓ Resolved
SKT-01	Inefficient Implementation	Language Specific	● Informational	✓ Resolved
SKT-02	Redundant Clone	Language Specific	● Informational	✓ Resolved
SKW-01	Proper Representation	Language Specific	● Informational	✓ Resolved
SNE-01	Use Of Panic	Language Specific	● Medium	✓ Resolved
SNK-01	Unchecked Conversion	Mathematical Operations	● Medium	✓ Resolved
SNK-02	Use Of Panic	Logical Issue	● Medium	✓ Resolved
SNK-03	Use Of Panic	Logical Issue	● Minor	✓ Resolved
SNK-04	Incrementation Implementation	Language Specific	● Informational	✓ Resolved
SNK-05	Redundant Clone	Language Specific	● Informational	✓ Resolved

ID	Title	Category	Severity	Status
SNK-06	Redundant Field	Logical Issue	● Informational	✓ Resolved
SNK-07	Inefficient Looping	Language Specific	● Informational	✓ Resolved
SNK-08	Redundant Closure	Language Specific	● Informational	✓ Resolved
SNK-09	Redundant Closure	Language Specific	● Informational	ⓘ Acknowledged
SNK-10	Inefficient Looping	Language Specific	● Informational	✓ Resolved
SNK-11	Redundant Clone	Language Specific	● Informational	✓ Resolved
SNK-12	Unnecessary Re-wrapping of StdResult	Logical Issue	● Informational	✓ Resolved
SNO-01	Redundant Clone	Language Specific	● Informational	✓ Resolved
SNO-02	Redundant Field	Language Specific	● Informational	✓ Resolved
SNO-03	Unnecessary Re-wrapping of StdResult	Language Specific	● Informational	✓ Resolved
SNR-01	Redundant Clone	Language Specific	● Informational	✓ Resolved
SNT-01	Admin Change Validation Missing	Logical Issue	● Minor	✓ Resolved
SNT-02	Unnecessary Implementation	Language Specific, Logical Issue	● Informational	✓ Resolved
SNT-03	Unnecessary Implementation	Language Specific	● Informational	✓ Resolved
SNT-04	Unimplemented Functionality	Language Specific	● Informational	✓ Resolved
SNT-05	Use Of Panic	Logical Issue, Language Specific	● Informational	✓ Resolved
SNT-06	Unnecessary Re-wrapping of StdResult	Language Specific	● Informational	✓ Resolved
SNT-07	Unnecessary Implementation	Language Specific	● Informational	✓ Resolved
SNT-08	Unnecessary Implementation	Language Specific	● Informational	✓ Resolved
SNT-09	Unnecessary Implementation	Language Specific	● Informational	✓ Resolved
SNT-10	Unnecessary Implementation	Language Specific	● Informational	✓ Resolved

ID	Title	Category	Severity	Status
SNT-11	Unnecessary Binding	Language Specific	● Informational	✓ Resolved
SNT-12	Unnecessary Implementation	Language Specific	● Informational	✓ Resolved
SNT-13	Unnecessary Binding	Language Specific	● Informational	✓ Resolved
SNT-14	Code Readability	Language Specific, Coding Style	● Informational	✓ Resolved
SNT-15	Unnecessary Binding	Language Specific	● Informational	✓ Resolved
SNT-16	Call Stack	Language Specific	● Informational	✓ Resolved
SNT-17	Unnecessary Manual Implementation	Language Specific	● Informational	✓ Resolved
SNT-18	Unnecessary Implementation	Language Specific	● Informational	✓ Resolved
SNT-19	Code Readability	Language Specific, Coding Style	● Informational	✓ Resolved
SNW-01	Non Optimal Conversion	Language Specific	● Informational	✓ Resolved
SNW-02	Redundant Variable Binding	Language Specific	● Informational	✓ Resolved
SNW-03	Redundant Matching Pattern	Language Specific	● Informational	✓ Resolved
SNW-04	Code Structure	Language Specific	● Informational	✓ Resolved
SNW-05	Unnecessary Re-wrapping	Language Specific	● Informational	✓ Resolved
SNW-06	Code Structure	Language Specific	● Informational	✓ Resolved
SNW-07	Non Optimal Usage Of Sort	Language Specific	● Informational	✓ Resolved

SKE-01 | Redundant Clone

Category	Severity	Location	Status
Language Specific	● Informational	shared/src/u256_math.rs: 68	✓ Resolved

Description

Unnecessary use of `clone` on `primitive_types::U256`, which implements `Copy`.

Recommendation

Consider removing the redundant cloning.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SKE-02 | Redundant Return

Category	Severity	Location	Status
Language Specific	● Informational	shared/src/u256_math.rs: 81	✓ Resolved

Description

Unnecessary explicit `return` statement.

Recommendation

Consider removing the redundant return.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SKO-01 | Code Structure

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/state.rs: 170~174, 224~228, 347~351, 367~371	✓ Resolved

Description

The `ReadonlyConfig::from_storage` is used as a conversion function, but doesn't follow typical Rust conversion idioms.

Recommendation

Consider re-implementing the `ReadonlyConfig::from_storage` function through the `From` trait.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SKO-02 | Code Usability

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/state.rs: 452~461	✓ Resolved

Description

The `get_receiver_hash` returns an `Option<StdResult<String>>`, which makes the function difficult to use in the system.

Recommendation

Consider inverting the order of `Option` and `StdResult` so that the return type becomes `StdResult<Option<String>>`, allowing callers to short circuit on the `StdResult` of the call in the event of an `Err`, leaving the `Option<String>` to be unwrapped by the caller's implementation.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SKO-03 | Proper Representation

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/state.rs: 473	✓ Resolved

Description

The `slice_to_u128` function uses a constant of `16` to represent the size of a `u128` in bytes, which can be better clarified through the use of the `std::mem::size_of` function.

Recommendation

Consider also changing `16 byte` to `16 bytes` in the error message supplied to the `StdError::generic_error` function on L476.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SKO-04 | Proper Representation

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/state.rs: 484~490	✓ Resolved

Description

The `slice_to_u8` function uses a contract of `1` to represent the size of a `u8` in bytes, along with an if/else expression which differs from the pattern match-based approach of the `slice_to_u128` function.

Recommendation

Consider re-implementing the `slice_to_u8` function.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SKR-01 | Code Structure

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/receiver.rs: 29~34	✓ Resolved

Description

The `Snip20ReceiveMsg::to_binary` function is used as a conversion function, but doesn't follow typical Rust conversion idioms.

Recommendation

Consider re-implementing the `Snip20ReceiveMsg::to_binary` function through the `TryInto<Binary>` trait for `Snip20ReceiveMsg`.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SKT-01 | Inefficient Implementation

Category	Severity	Location	Status
Language Specific	● Informational	shared/src/asset.rs: 93~94	✓ Resolved

Description

The `create_send_msg` function creates a `Coin` instance and supplies the fields in an inefficient manner. Consider replacing `denom: denom.to_string()` with `denom: denom.clone()` since `denom` is already of type `String`.

Recommendation

Consider reducing `amount: amount` to `amount`.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SKT-02 | Redundant Clone

Category	Severity	Location	Status
Language Specific	● Informational	shared/src/asset.rs: 185	✓ Resolved

Description

The `TokenType::query_balance` function contains a redundant use of `clone` on the `exchange_addr` parameter, which is dropped without further use.

Recommendation

Consider passing the `exchange_addr` parameter by value on L185.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SKW-01 | Proper Representation

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/viewing_key.rs: 26	✓ Resolved

Description

The code represent a variable in a non optimal way.

Recommendation

Consider defining a constant variable for the `16` literal in the `ViewingKey::new` function.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNE-01 | Use Of Panic

Category	Severity	Location	Status
Language Specific	● Medium	lp-token/src/msg.rs: 222	✓ Resolved

Description

The `QueryMsg::get_validation_params` function contains a potential panic on line 222, which is unsafe due to causing the program to terminate.

Recommendation

Consider refactoring the return type of the `QueryMsg::get_validation_params` function into a `StdResult` and returning a `StdError` instead of panicking.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNK-01 | Unchecked Conversion

Category	Severity	Location	Status
Mathematical Operations	● Medium	exchange/src/contract.rs: 223, 254, 320, 400, 403, 570, 634~636	✓ Resolved

Description

The function extracts the lower 128 bits of a `Uint128` to be converted into a without checking if the value has overflowed the maximum `Uint128` value.

Recommendation

While not critical, action should be taken in the event that an overflow does occur.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNK-02 | Use Of Panic

Category	Severity	Location	Status
Logical Issue	● Medium	exchange/src/contract.rs: 463~466	✓ Resolved

Description

The linked code contains a potential panic.

Recommendation

Consider refactoring the code and opting on an error log and graceful exit.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNK-03 | Use Of Panic

Category	Severity	Location	Status
Logical Issue	● Minor	exchange/src/contract.rs: 35~37	✓ Resolved

Description

The init function contains a potential panic on line 36, which is unsafe as it will cause program termination.

Recommendation

Since the init function returns a StdResult , consider returning a StdError instead of panicking.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNK-04 | Incrementation Implementation

Category	Severity	Location	Status
Language Specific	● Informational	exchange/src/contract.rs: 402	✓ Resolved

Description

The swap function contains a manual implementation of an incrementation operation on line 402, which is unnecessary.

Recommendation

Consider refactoring line 402 to use a primitive incrementation.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNK-05 | Redundant Clone

Category	Severity	Location	Status
Language Specific	● Informational	exchange/src/contract.rs: 77	✓ Resolved

Description

The init function contains redundant usage of the clone function, which is immediately taken by reference and never consumed.

Recommendation

Consider removing the use of the clone function on line 77.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNK-06 | Redundant Field

Category	Severity	Location	Status
Logical Issue	● Informational	exchange/src/contract.rs: 103	✓ Resolved

Description

The init function contains a redundant viewing_key field name in the Config struct initialization.

Recommendation

Consider removing the redundant viewing_key field name on line 103.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNK-07 | Inefficient Looping

Category	Severity	Location	Status
Language Specific	● Informational	exchange/src/contract.rs: 174, 176, 196	Resolved

Description

The add_liquidity function performs a loop over the tokens of the deposit on line 176 while using a local mutable i variable as a loop counter, which is unnecessary.

Recommendation

Consider removing the local mutable i variable declared on line 174, the incrementation on line 196, replacing it with the enumerate function, which provides a loop counter.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNK-08 | Redundant Closure

Category	Severity	Location	Status
Language Specific	● Informational	exchange/src/contract.rs: 215	✓ Resolved

Description

The function contains a redundant closure, used to supply a value to the function which is inefficient.

Recommendation

Consider removing the closure and supplying the u256_math::sqrt function as the value supplied to the and_then function.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNK-09 | Redundant Closure

Category	Severity	Location	Status
Language Specific	● Informational	exchange/src/contract.rs: 216~221, 233~240, 245~252, 311~318, 566~568, 595~602, 606~615, 620~627, 649~656, 658~664	ⓘ Acknowledged

Description

The function contains a redundant closure via the unnecessary usage of the ok_or_else function, which is inefficient.

Recommendation

Consider replacing the use of the ok_or_else function with the ok_or function.

Alleviation

The team acknowledged the issues and opted not to alleviate in the current iteration.

SNK-10 | Inefficient Looping

Category	Severity	Location	Status
Language Specific	● Informational	exchange/src/contract.rs: 324, 326, 331	✓ Resolved

Description

The function performs a loop over the tokens of the deposit while using a local mutable i variable as a loop counter, which is unnecessary.

Recommendation

Consider removing the local mutable i variable declared and the incrementation, replacing it with the enumerate function, which provides a loop counter.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNK-11 | Redundant Clone

Category	Severity	Location	Status
Language Specific	● Informational	exchange/src/contract.rs: 350	✓ Resolved

Description

The function contains redundant usage of the clone function, which is immediately taken by reference and never consumed.

Recommendation

Consider removing the use of the clone function.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNK-12 | Unnecessary Re-wrapping of StdResult

Category	Severity	Location	Status
Logical Issue	● Informational	exchange/src/contract.rs: 480~486	✓ Resolved

Description

The function contains unnecessary re-wrapping of the StdResult returned from the call to the to_binary function.

Recommendation

Consider removing the explicit and short-circuit evaluation (?) from the call to the to_binary function.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNO-01 | Redundant Clone

Category	Severity	Location	Status
Language Specific	● Informational	factory/src/contract.rs: 82	✓ Resolved

Description

The create_exchange function performs a redundant clone on env.contract.address inside of a call to the format! macro.

Recommendation

This is unnecessary because the macro will take env.contract.address by reference. Consider removing the use of the clone function on line 82.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNO-02 | Redundant Field

Category	Severity	Location	Status
Language Specific	● Informational	factory/src/contract.rs: 169	✓ Resolved

Description

The create_ido function supplies a redundant info field name inside the IdoInitMsg struct initialization.

Recommendation

Consider refactoring info: info, to just info, in order to simplify the code.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNO-03 | Unnecessary Re-wrapping of StdResult

Category	Severity	Location	Status
Language Specific	● Informational	factory/src/contract.rs: 212~214, 223~225	✓ Resolved

Description

The function contains unnecessary re-wrapping of the StdResult returned from the call to the to_binary function.

Recommendation

Consider removing the explicit Ok() and short-circuit evaluation (?) from the call to the to_binary function.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNR-01 | Redundant Clone

Category	Severity	Location	Status
Language Specific	● Informational	exchange/src/state.rs: 53	✓ Resolved

Description

The store_config function contains unnecessary usage of the clone function on an array of type Uint128, which is unnecessary due to Uint128 implementing the Copy trait.

Recommendation

Consider removing the use of the clone function.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNT-01 | Admin Change Validation Missing

Category	Severity	Location	Status
Logical Issue	● Minor	lp-token/src/contract.rs: 294~312	✓ Resolved

Description

The change_admin function directly swaps the admin's address without performing any verification on the new admin address, which can leave the contract unrecoverable if an invalid address is supplied.

Recommendation

Consider adding an additional step to the ChangeAdmin phase, requiring the new admin address to ClaimAdmin before transfer the actual admin rights. This will also allow the old admin to send corrective ChangeAdmin messages in the event that they transfer their administrative rights to an incorrect or invalid address.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-02 | Unnecessary Implementation

Category	Severity	Location	Status
Language Specific, Logical Issue	● Informational	lp-token/src/contract.rs: 41~47	✓ Resolved

Description

The init function contains a manual implementation of Option::ok_or , which is unnecessary.

Recommendation

Consider refactoring the balance check using the Option::ok_or function.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-03 | Unnecessary Implementation

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/contract.rs: 66, 81	✓ Resolved

Description

The function contains a manual implementation of Option::ok_or , which is unnecessary.

Recommendation

Consider taking each address by reference and replacing the usage of Option::unwrap_or_else with Option::unwrap_or on L66. This will effectively changed the inferred type of the admin variable to &HumanAddress , which can be cloned where necessary on L81.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-04 | Unimplemented Functionality

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/contract.rs: 123~128, 139~149, 170	✓ Resolved

Description

The code contains unimplemented functionality.

Recommendation

Consider providing some explanation about the state of the linked code.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-05 | Use Of Panic

Category	Severity	Location	Status
Logical Issue, Language Specific	● Informational	lp-token/src/contract.rs: 226, 207~228	✓ Resolved

Description

The authenticated_queries function contains panicking and unnecessary empty viewing key check for the sake of taking time.

Recommendation

Consider refactoring lines 207-228 in order to be more legible and return a QueryAnswer::ViewingKeyError instead of panicking.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-06 | Unnecessary Re-wrapping of StdResult

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/contract.rs: 231~233	✓ Resolved

Description

The function contains unnecessary re-wrapping of the StdResult returned from the call to the to_binary function.

Recommendation

Consider removing the explicit and short-circuit evaluation (?) from the call to the to_binary function.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-07 | Unnecessary Implementation

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/contract.rs: 332~338, 347~357	✓ Resolved

Description

The try_mint function contains manual implementations of Option::ok_or , which is unnecessary.

Recommendation

Consider refactoring the balance checks using the Option::ok_or function.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-08 | Unnecessary Implementation

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/contract.rs: 361~367	✓ Resolved

Description

The try_mint function stores a HandleResponse instance in a res let binding without modifying or passing it as a function call argument before returning an explicit Ok(res) , which is unnecessary.

Recommendation

Consider removing the res let binding and placing the HandleResponse struct instantiation within the returned Ok expression.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-09 | Unnecessary Implementation

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/contract.rs: 468~475, 506~515, 518~525	✓ Resolved

Description

The function contains manual implementations of Option::ok_or , which is unnecessary.

Recommendation

Consider refactoring the balance checks using the Option::ok_or function.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-10 | Unnecessary Implementation

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/contract.rs: 487~493	✓ Resolved

Description

The try_mint function stores a HandleResponse instance in a res let binding without modifying or passing it as a function call argument before returning an explicit Ok(res) , which is unnecessary.

Recommendation

Consider removing the res let binding and placing the HandleResponse struct instantiation within the returned Ok expression.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-11 | Unnecessary Binding

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/contract.rs: 532~542	✓ Resolved

Description

The try_redeem function stores a HandleResponse instance in a res let binding without modifying or passing it as a function call argument before returning an explicit Ok(res) , which is unnecessary.

Recommendation

Consider removing the res let binding and placing the HandleResponse struct instantiation within the returned Ok expression.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-12 | Unnecessary Implementation

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/contract.rs: 563	Resolved

Description

The `try_transfer_impl` function performs a call to the `store_transfer` function with a short circuit operator (`?`) before returning an explicit `Ok(())`. Since the `try_transfer_impl` and `store_transfer` functions both return `StdResult<()>`.

Recommendation

Consider removing the explicit `Ok(())` and the short circuit operator (`?`) from the call to the `store_transfer` function, which will allow the result from the call to the `store_transfer` function to fall through as the result for the `try_transfer_impl` function.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-13 | Unnecessary Binding

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/contract.rs: 584~589	✓ Resolved

Description

The `try_transfer` function stores a `HandleResponse` instance in a `res` let binding without modifying or passing it as a function call argument before returning an explicit `Ok(res)`, which is unnecessary.

Recommendation

Consider removing the `res` let binding and placing the `HandleResponse` struct instantiation within the returned `Ok` expression.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-14 | Code Readability

Category	Severity	Location	Status
Language Specific, Coding Style	● Informational	lp-token/src/contract.rs: 601~608	✓ Resolved

Description

The `try_add_receiver_api_callback` function binds the result of a call to the `get_receiver_hash` to a local `receiver_hash` variable, but only utilizes it once in an `if let` binding, which makes the function difficult to read.

Recommendation

Consider refactoring in order to improve legibility.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-15 | Unnecessary Binding

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/contract.rs: 635~640, 649~656, 733~738, 764~769, 811~817, 843~852, 878~887, 938~989	✓ Resolved

Description

The function stores a `HandleResponse` instance in a `res` let binding without modifying or passing it as a function call argument before returning an explicit `Ok(res)`, which is unnecessary.

Recommendation

Consider removing the `res` let binding and placing the `HandleResponse` struct instantiation within the returned `Ok` expression.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-16 | Call Stack

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/contract.rs: 659, 1053	✓ Resolved

Description

The `insufficient_allowance` function is a utility function which simply calls the `StdError::generic_err` function and can needlessly increase the call stack during execution.

Recommendation

Consider adding an explicit `#[inline]` attribute to the `insufficient_allowance` function.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-17 | Unnecessary Manual Implementation

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/contract.rs: 691~695, 785~793, 799~809	✓ Resolved

Description

The `try_transfer_from_impl` function contains a manual implementation of `Option::ok_or`, which is unnecessary.

Recommendation

Consider refactoring the allowance check using the `Option::ok_or` function.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-18 | Unnecessary Implementation

Category	Severity	Location	Status
Language Specific	● Informational	lp-token/src/contract.rs: 712~721	✓ Resolved

Description

The `try_transfer_from_impl` function performs a call to the `store_transfer` function with a short circuit operator (`?`) before returning an explicit `Ok(())`. Since the `try_transfer_from_impl` and `store_transfer` functions both return `StdResult<()>`.

Recommendation

Consider removing the explicit `Ok(())` and the short circuit operator (`?`) from the call to the `store_transfer` function, which will allow the result from the call to the `store_transfer` function to fall through as the result for the `try_transfer_from_impl` function.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNT-19 | Code Readability

Category	Severity	Location	Status
Language Specific, Coding Style	● Informational	lp-token/src/contract.rs: 1041	✓ Resolved

Description

The `is_valid_symbol` function contains a manual `RangeInclusive::contains` implementation for both `3 <= len && len <= 12` and `b'A' <= byte && byte <= b'Z'`, which makes the function difficult to review.

Recommendation

Consider re-implementing the `is_valid_symbol` function using Rust idioms that more clearly convey its purpose.

Alleviation

The team fixed the issue with refactoring in commits up to release amm-1.0.0.

SNW-01 | Non Optimal Conversion

Category	Severity	Location	Status
Language Specific	● Informational	factory/src/state.rs: 60~70	✓ Resolved

Description

The Config::from_init_msg function is used as a conversion function but doesn't follow the typical Rust conversion idioms.

Recommendation

Consider re-implementing the Config::from_init_msg function under the From<InitMsg> trait for the Config struct.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNW-02 | Redundant Variable Binding

Category	Severity	Location	Status
Language Specific	● Informational	factory/src/state.rs: 109~121	✓ Resolved

Description

The pair_exists function makes use of redundant variable bindings and Option pattern matching.

Recommendation

Consider that this can be simplified.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNW-03 | Redundant Matching Pattern

Category	Severity	Location	Status
Language Specific	● Informational	factory/src/state.rs: 135	✓ Resolved

Description

The store_exchange function makes use of redundant Option pattern matching on L135.

Recommendation

Consider using is_some() instead.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNW-04 | Code Structure

Category	Severity	Location	Status
Language Specific	● Informational	factory/src/state.rs: 153	✓ Resolved

Description

Description: The storage_exchange function performs a call to the save_exchanges function with a short circuit operator (?) before returning an explicit Ok(()) . Since the storage_exchange and save_exchanges functions both return StdResult<()> .

Recommendation

Consider removing the explicit Ok(()) and the short circuit operator (?) from the call to the save_exchanges function, which will allow the result from the call to the save_exchanges function to fall through as the result of the storage_exchange function.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNW-05 | Unnecessary Re-wrapping

Category	Severity	Location	Status
Language Specific	● Informational	factory/src/state.rs: 168	Resolved

Description

The get_address_for_pair function contains unnecessary re-wrapping of a short-circuited StdResult<HumanAddr> in an explicit Ok variant, which is unnecessary.

Recommendation

Consider removing the explicit Ok and the short circuit operator (?) from the call to deps.api.human_address(&canonical) in order to allow the result to fall through as the result for the get_address_for_pair function.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNW-06 | Code Structure

Category	Severity	Location	Status
Language Specific	● Informational	factory/src/state.rs: 182	✓ Resolved

Description

The store_ido_address function performs a call to the save_exchanges function with a short circuit operator (?) before returning an explicit Ok(()) . Since the store_ido_address and save_exchanges functions both return StdResult<()> .

Recommendation

Consider removing the explicit Ok(()) and the short circuit operator (?) from the call to the save_exchanges function, which will allow the result from the call to the save_exchanges function to fall through as the result of the store_ido_address function.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

SNW-07 | Non Optimal Usage Of Sort

Category	Severity	Location	Status
Language Specific	● Informational	factory/src/state.rs: 272	✓ Resolved

Description

The generate_pair_key function explicitly implements Vec::sort through the use of Vec::sort on line 272

Recommendation

Consider replacing bytes.sort_by with bytes.sort() in order to simplify the expression.

Alleviation

The team fixed the issue in commits up to release amm-1.0.0.

Appendix

Finding Categories

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK