F Y E O

Security Assessment of the XRPL Labs Hooks Amendment

The Integrators BV

May 31, 2023
Version 1.0

Presented by:

FYEO Inc.

PO Box 147044
Lakewood CO 80214
United States

Security Level
Public

Table of Contents

Executive Summary

Overview

The Integrators BV. engaged FYEO Inc. to perform a Security Assessment of the XRPL Labs Hooks Amendment.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on January 31 - March 14, 2023, and focused on the following objectives:

- To provide the customer with an assessment of their overall security posture and any risks that were discovered within the environment during the engagement.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
- To identify potential issues and include improvement recommendations based on the results of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the FYEO Security Team took to identify and validate each issue, as well as any applicable recommendations for remediation.

Key Findings

The following issues were identified during the testing period and have since been remediated.

- FYEO-XRPL-01 – NOT_IN_BOUNDS incorrectly checks start position
- FYEO-XRPL-02 – Cast from int64 to uint32
- FYEO-XRPL-03 – Emitted transactions may be skipped
- FYEO-XRPL-04 – Pointer may not exist in slot_float
- FYEO-XRPL-05 – Pointer may not exist in slot_size
- FYEO-XRPL-06 – Pointer may not exist in slot_subfield
- FYEO-XRPL-07 – Possible access to the null pointer returned by WasmEdge_MemoryInstanceGetPointer()
- FYEO-XRPL-08 – Transaction added to emitted before all checks are done
- FYEO-XRPL-09 – Undefined Behavior in normalize_xfl
- FYEO-XRPL-10 – Wrong order of checks in slot_count
- FYEO-XRPL-11 – Wrong order of checks in slot_subarray

- FYEO-XRPL-12 – Add an overflow safeguard for float_multiply_internal_parts
- FYEO-XRPL-13 – Assertion is used to verify runtime condition
- FYEO-XRPL-14 – Handle exceptions inside hooks
- FYEO-XRPL-15 – Inconsistent order of operations
- FYEO-XRPL-16 – Inconsistent use of try catch when applying hooks
- FYEO-XRPL-17 – Incorrect check for maximum state modifications
- FYEO-XRPL-18 – Possible read of nonexistent field in emit()
- FYEO-XRPL-19 – Raw pointers are used for WasmEdge variables
- FYEO-XRPL-20 – Release flag is not specified for release configuration
- FYEO-XRPL-21 – The externref type is not processed in check_guard
- FYEO-XRPL-22 – The tecHOOK_REJECTED state is valid for executing hooks
- FYEO-XRPL-23 – Unhandled negative condition in incrementReferenceCount
- FYEO-XRPL-24 – Unprocessed write error in otxn_field() and slot()
- FYEO-XRPL-25 – slot_into may be greater than max_slots
- FYEO-XRPL-26 – ADD_TSH macro can be replaced with lambda
- FYEO-XRPL-27 – Conditions duplication in hook_float
- FYEO-XRPL-28 – Create enum for errors returned from get_stobject_length
- FYEO-XRPL-29 – Improve WasmEdge error logs
- FYEO-XRPL-30 – Incorrect order of members initialization in WasmBlkInf
- FYEO-XRPL-31 – Overlapping memory check can be optimized
- FYEO-XRPL-32 – Processing of emitted transaction in finalizeHookResult can be optimized
- FYEO-XRPL-33 – Redundant if check
- FYEO-XRPL-34 – Unclear usage of literal value in setHook()
- FYEO-XRPL-35 – Unnecessary cast to pointer
- FYEO-XRPL-36 – Unnecessary cast to pointer in getTransactionalStakeHolders
- FYEO-XRPL-37 – Unreachable condition in state_foreign
- FYEO-XRPL-38 – Unreachable return in etxn_nonce and ledger_nonce
- FYEO-XRPL-39 – Unused code
- FYEO-XRPL-40 – Variable name is the same as namespace
- FYEO-XRPL-41 – WasmBlkInf destructor can be simplified

Based on our review process, we conclude that the reviewed code implements the documented functionality.

Scope and Rules of Engagement

The FYEO Review Team performed a Security Assessment of the XRPL Labs Hooks Amendment. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

The source code was supplied through a private repository at [redacted] with the commit hash 0aaf98c46ea4506d4120d6d7c07f9573ca8faf7b.

Remediations were carried out on up to and including commit hash d81fa8c07ca81c5c98ecb7166faab8b3e4cd0eca

Files included in the code review
[redacted]/ ├── src/ │ ├── test/ │ │ ├── app/ │ │ │ ├── SetHook_test.cpp └── └── ripple/ ├── app/ ├── hook/ │ ├── impl/ │ │ └── applyHook.cpp │ ├── Guard.h ├── misc/ │ │ ├── TxQ.cpp └── tx/ ├── impl/ │ ├── Invoke.cpp │ ├── SetHook.cpp └── └── Transactor.cpp

Table 1: Scope

Technical Analyses and Findings

During the Security Assessment of the XRPL Labs Hooks Amendment, we discovered:

- 1 finding with HIGH severity rating.
- 10 findings with MEDIUM severity rating.
- 14 findings with LOW severity rating.
- 16 findings with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

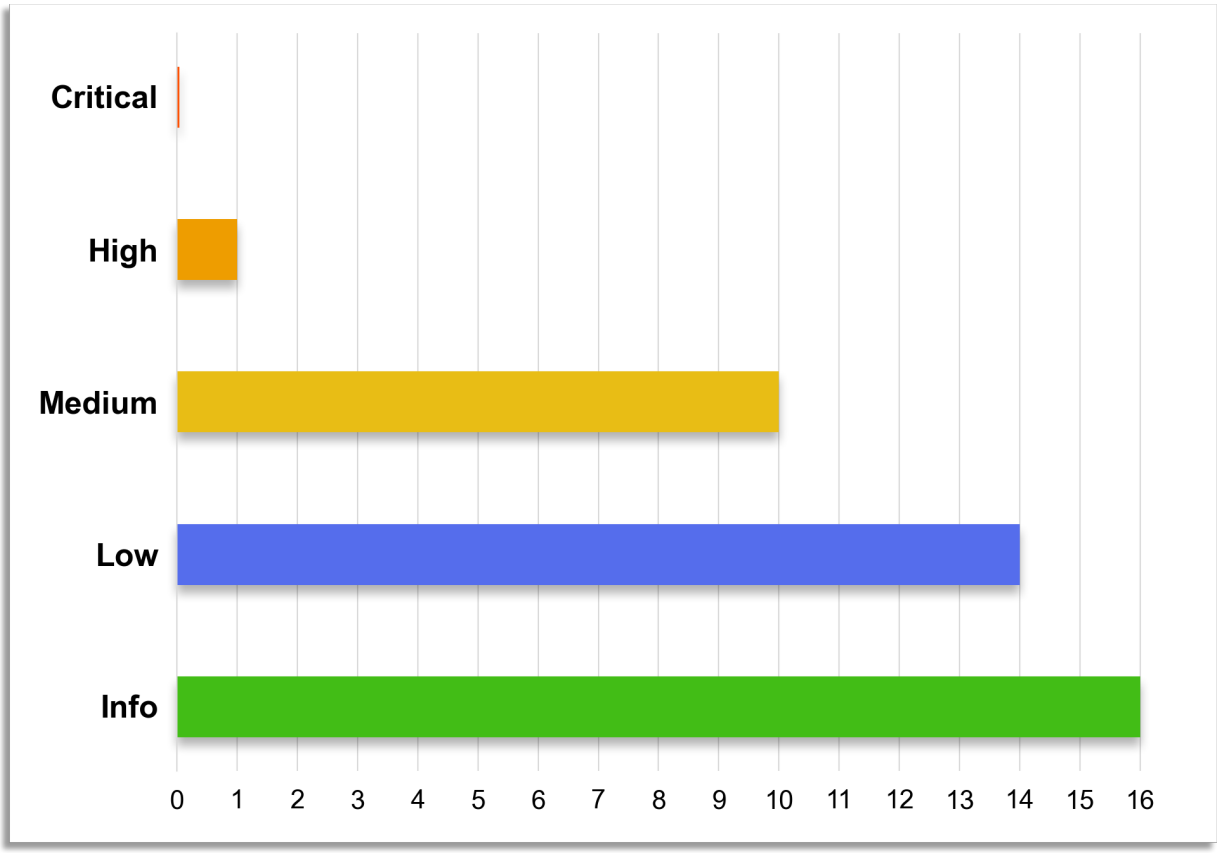

Figure 1: Findings by Severity

FINDINGS

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

Finding #	Severity	Description
FYEO-XRPL-01	High	NOT_IN_BOUNDS incorrectly checks start position
FYEO-XRPL-02	Medium	Cast from int64 to uint32
FYEO-XRPL-03	Medium	Emitted transactions may be skipped
FYEO-XRPL-04	Medium	Pointer may not exist in slot_float
FYEO-XRPL-05	Medium	Pointer may not exist in slot_size
FYEO-XRPL-06	Medium	Pointer may not exist in slot_subfield
FYEO-XRPL-07	Medium	Possible access to the null pointer returned by WasmEdge_MemoryInstanceGetPointer()
FYEO-XRPL-08	Medium	Transaction added to emitted before all checks are done
FYEO-XRPL-09	Medium	Undefined Behavior in normalize_xfl
FYEO-XRPL-10	Medium	Wrong order of checks in slot_count
FYEO-XRPL-11	Medium	Wrong order of checks in slot_subarray
FYEO-XRPL-12	Low	Add an overflow safeguard for float_multiply_internal_parts
FYEO-XRPL-13	Low	Assertion is used to verify runtime condition
FYEO-XRPL-14	Low	Handle exceptions inside hooks
FYEO-XRPL-15	Low	Inconsistent order of operations
FYEO-XRPL-16	Low	Inconsistent use of try catch when applying hooks
FYEO-XRPL-17	Low	Incorrect check for maximum state modifications
FYEO-XRPL-18	Low	Possible read of nonexistent field in emit()

FYEO-XRPL-19	Low	Raw pointers are used for WasmEdge variables
FYEO-XRPL-20	Low	Release flag is not specified for release configuration
FYEO-XRPL-21	Low	The externref type is not processed in check_guard
FYEO-XRPL-22	Low	The tecHOOK_REJECTED state is valid for executing hooks
FYEO-XRPL-23	Low	Unhandled negative condition in incrementReferenceCount
FYEO-XRPL-24	Low	Unprocessed write error in otxn_field() and slot()
FYEO-XRPL-25	Low	slot_into may be greater than max_slots
FYEO-XRPL-26	Informational	ADD_TSH macro can be replaced with lambda
FYEO-XRPL-27	Informational	Conditions duplication in hook_float
FYEO-XRPL-28	Informational	Create enum for errors returned from get_stobject_length
FYEO-XRPL-29	Informational	Improve WasmEdge error logs
FYEO-XRPL-30	Informational	Incorrect order of members initialization in WasmBlkInf
FYEO-XRPL-31	Informational	Overlapping memory check can be optimized
FYEO-XRPL-32	Informational	Processing of emitted transaction in finalizeHookResult can be optimized
FYEO-XRPL-33	Informational	Redundant if check
FYEO-XRPL-34	Informational	Unclear usage of literal value in setHook()
FYEO-XRPL-35	Informational	Unnecessary cast to pointer
FYEO-XRPL-36	Informational	Unnecessary cast to pointer in getTransactionalStakeHolders
FYEO-XRPL-37	Informational	Unreachable condition in state_foreign
FYEO-XRPL-38	Informational	Unreachable return in etxn_nonce and ledger_nonce
FYEO-XRPL-39	Informational	Unused code

FYEO-XRPL-40	**Informational**	Variable name is the same as namespace
FYEO-XRPL-41	**Informational**	WasmBlkInf destructor can be simplified

Table 2: Findings Overview

Technical Analysis

The source code has been manually validated to the extent that the state of the repository allowed. The validation includes confirming that the code correctly implements the intended functionality.

Technical Findings

General Observations

Incorporating Hooks into the XRP Ledger allows for the addition of Smart Contract functionality at layer one, enabling the customization of transaction behavior and flow. These Hooks, which are small and efficient code snippets defined on an XRPL account, can execute logic before and/or after transactions.

The concept of Hooks and their helper functions are well described in the documentation.

Hooks are represented in the WASM code. It can be compiled from C code using XRPL Hooks Builder. In the code, WASM is processed by the WasmEdge library.

The audit was focused on analyzing the Hooks framework implementation, including:

- hooks helper functions
- hooks installation
- hooks execution

Most of the identified issues are recommendations of best practices and small mistakes with low impact on security.

Several medium-risk vulnerabilities are related to:

- unsafe type casting
- emitted transaction processing
- low probability issues that cause undefined behavior

The high-risk vulnerabilities also cause undefined behavior but the probability of it is higher.

The code style is overall good, and while it does use some c-style constructions like using `0` as a null pointer or variable casting, these can easily be changed to more modern approaches.

Unit tests check different scenarios for precompiled wasm hooks that are executed in a transaction testing environment.

The overall security of the project is good, no issues regarding possible assets loss or unauthorized access were found.

NOT_IN_BOUNDS INCORRECTLY CHECKS START POSITION

Finding ID: FYEO-XRPL-01
Severity: **High**
Status: **Remediated**

Description

Start position `ptr`, with a value equal to `memory_length` and with a `len = 0` may result in out-of-bound access to the memory.

Proof of Issue

File name: src/ripple/app/hook/Macro.h

Line number: 181

```
#define NOT_IN_BOUNDS(ptr, len, memory_length)\
    ((static_cast<uint64_t>(ptr) > static_cast<uint64_t>(memory_length)) || \
     ((static_cast<uint64_t>(ptr) + static_cast<uint64_t>(len)) >
static_cast<uint64_t>(memory_length)))
```

Severity and Impact Summary

Out-of-bound access of the pointer results in Undefined Behaviour.

Recommendation

It is recommended to verify that `ptr` is always less than `memory_length`.

CAST FROM INT64 TO UINT32

Finding ID: FYEO-XRPL-02
Severity: Medium
Status: Remediated

Description

There are casts of 64-bit signed integers to 32-bit unsigned.

Proof of Issue

File name: src/ripple/app/tx/impl/Transactor.cpp
Line number: 251

```
fee += FeeUnit64{
    (uint32_t)(hookDef->getFieldAmount(sfFee).xrp().drops())
};
```

File name: src/ripple/app/tx/impl/Transactor.cpp
Line number: 295

```
FeeUnit64{(uint32_t)(hookDef-
>getFieldAmount(sfHookCallbackFee).xrp().drops())};
```

The fee calculation also does not check for overflows when accumulating in `hookExecutionFee` and in:

```
return baseFee * burden + (signerCount * baseFee) + hookExecutionFee;
```

Severity and Impact Summary

This cast between signed and unsigned and to fewer bits is dangerous.

Recommendation

It is recommended to properly convert these numbers.

EMITTED TRANSACTIONS MAY BE SKIPPED

Finding ID: FYEO-XRPL-03
Severity: Medium
Status: Remediated

Description

Injection of emitted transactions is done in the loop where elements of `emittedDir()` is processed. If one of the elements is malformed, then the processing will stop, all remaining emitted transactions will be skipped, and those already processed will be accepted.

Proof of Issue

File name: src/ripple/app/misc/impl/TxQ.cpp

Line number: 1448

```
            if (!sleItem)
            {
                // Directory node has an invalid index.  Bail out.
                JLOG(j_.fatal())
                    << "EmittedTxn processing: directory node in ledger " <<
view.seq()
                    << " has index to object that is missing: "
                    << to_string(dirEntry);
                break;
            }
```

Line number: 1461

```
            if (nodeType != ltEMITTED_TXN)
            {
                JLOG(j_.fatal())
                    << "EmittedTxn processing: emitted directory contained non
ltEMITTED_TXN type";
                break;
            }
```

Severity and Impact Summary

Only a part of the correct emitted transactions will be accepted.

Recommendation

It is recommended to skip only malformed transactions.

POINTER MAY NOT EXIST IN SLOT_FLOAT

Finding ID: FYEO-XRPL-04
Severity: Medium
Status: Remediated

Description

The access to the pointer is done before validation that the pointer exists.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 2443

```
        ripple::STAmount& st_amt =

const_cast<ripple::STBase&>(*hookCtx.slot[slot_no].entry).downcast<ripple::STA
mount>();
```

Severity and Impact Summary

The issue causes Undefined Behavior that may result in a crash.

Recommendation

It is recommended to validate the existence of a pointer before usage.

POINTER MAY NOT EXIST IN SLOT_SIZE

Finding ID: FYEO-XRPL-05
Severity: **Medium**
Status: **Remediated**

Description

The access to the pointer is done before validation that the pointer exists.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 2284

```
hookCtx.slot[slot_no].entry->add(s);
```

Severity and Impact Summary

The issue causes Undefined Behavior that may result in a crash.

Recommendation

It is recommended to validate the existence of a pointer before usage.

POINTER MAY NOT EXIST IN SLOT_SUBFIELD

Finding ID: FYEO-XRPL-06
Severity: Medium
Status: Remediated

Description

The access to the pointer is done before validation that the pointer exists.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 2366

```
        ripple::STObject& parent_obj =

const_cast<ripple::STBase&>(*hookCtx.slot[parent_slot].entry).downcast<ripple:
:STObject>();
```

Severity and Impact Summary

The issue causes Undefined Behavior that may result in a crash.

Recommendation

It is recommended to validate the existence of a pointer before usage.

POSSIBLE ACCESS TO THE NULL POINTER RETURNED BY WASMEDGE_MEMORYINSTANCEGETPOINTER()

Finding ID: FYEO-XRPL-07
Severity: **Medium**
Status: **Remediated**

Description

The `WasmEdge_MemoryInstanceGetPointer()` function may return `NULL` in the case of a failed operation.

Proof of Issue

File name: src/ripple/app/hook/Macro.h

Line number: 150

```
[[maybe_unused]] unsigned char* memory =
WasmEdge_MemoryInstanceGetPointer(memoryCtx, 0, 0);\
```

Severity and Impact Summary

Access to the null pointer results in Undefined Behaviour.

Recommendation

It is recommended to verify that `memory` pointer always exists.

TRANSACTION ADDED TO EMITTED BEFORE ALL CHECKS ARE DONE

Finding ID: FYEO-XRPL-08
Severity: Medium
Status: Remediated

Description

The `emit()` function adds a transaction to emitted queue and then performs additional checks.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 3107

```
    hookCtx.result.emittedTxn.push(tpTrans);

    auto const& txID =
        tpTrans->getID();

    if (txID.size() > write_len)
        return TOO_SMALL;

    if (NOT_IN_BOUNDS(write_ptr, txID.size(), memory_length))
        return OUT_OF_BOUNDS;
```

Severity and Impact Summary

The transaction is added to emitted queue even if the hook fails.

Recommendation

It is recommended to add a transaction to emitted after all checks are done.

UNDEFINED BEHAVIOR IN NORMALIZE_XFL

Finding ID: FYEO-XRPL-09
Severity: **Medium**
Status: **Remediated**

Description

In the case when `man` parameter is equal to `0` or `std::numeric_limits<int64_t>::min()` the function will cause Undefined Behavior by accessing out of bounds of the `power_of_ten` array.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 457, 462

```
        int32_t mo = log10(man);
        int32_t adjust = 15 - mo;

        if (adjust > 0)
        {
            man *= power_of_ten[adjust];
            exp -= adjust;
        }
        else if (adjust < 0)
        {
            man /= power_of_ten[-adjust];
            exp -= adjust;
        }
```

Severity and Impact Summary

The issue causes Undefined Behavior that may result in a crash.

Recommendation

It is recommended to check the `man` parameter before the `log10()` operation.

WRONG ORDER OF CHECKS IN SLOT_COUNT

Finding ID: FYEO-XRPL-10
Severity: Medium
Status: Remediated

Description

The access to the pointer is done before validation that the pointer exists.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 2196

```
    if (hookCtx.slot[slot_no].entry->getSType() != STI_ARRAY)
        return NOT_AN_ARRAY;

    if (hookCtx.slot[slot_no].entry == 0)
        return INTERNAL_ERROR;
```

Severity and Impact Summary

The issue causes Undefined Behavior that may result in a crash.

Recommendation

It is recommended to validate the existence of a pointer before usage.

WRONG ORDER OF CHECKS IN SLOT_SUBARRAY

Finding ID: FYEO-XRPL-11
Severity: Medium
Status: Remediated

Description

The access to the pointer is done before validation that the pointer exists.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 2296

```
    if (hookCtx.slot[parent_slot].entry->getSType() != STI_ARRAY)
        return NOT_AN_ARRAY;

    if (hookCtx.slot[parent_slot].entry == 0)
        return INTERNAL_ERROR;
```

Severity and Impact Summary

The issue causes Undefined Behavior that may result in a crash.

Recommendation

It is recommended to validate the existence of a pointer before usage.

ADD AN OVERFLOW SAFEGUARD FOR FLOAT_MULTIPLY_INTERNAL_PARTS

Finding ID: FYEO-XRPL-12
Severity: Low
Status: Remediated

Description

The result of `cpp_int` multiplication might be greater than `uint64_t`, in this case, `static_cast` will truncate the result.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 4255

```
    cpp_int mult = cpp_int(man1) * cpp_int(man2);
    mult /= power_of_ten[15];
    uint64_t man_out = static_cast<uint64_t>(mult);
```

Severity and Impact Summary

The produced result may be incorrect.

Recommendation

It is recommended to check that after casting the number is not truncated:

```
    if (mult > man_out)
        return XFL_OVERFLOW;
```

ASSERTION IS USED TO VERIFY RUNTIME CONDITION

Finding ID: FYEO-XRPL-13
Severity: Low
Status: Remediated

Description

Assertions are a mechanism that is designed for development and testing and should not be used to verify runtime conditions.

Proof of Issue

File name: src/ripple/app/tx/impl/SetHook.cpp

Line number: 1262

```
assert(hookSetObj || op == hsoNOOP);
```

Line number: 1277

```
// every case below here is guaranteed to have a populated hookSetObj
// by the assert statement above
```

Line number: 1353

```
    if (hookSetObj->get().isFieldPresent(sfHookParameters) &&
        hookSetObj->get().getFieldArray(sfHookParameters).empty())
```

Severity and Impact Summary

The issue causes Undefined Behavior that may result in a crash.

Recommendation

It is recommended to not use assertion in production code and instead verify conditions using `if` statements.

HANDLE EXCEPTIONS INSIDE HOOKS

Finding ID: FYEO-XRPL-14
Severity: Low
Status: Remediated

Description

Some of the hook functions are not handling exceptions.

Proof of Issue

For example, the exception will be thrown if `sfTransactionType` field is not present for `emitFailure`.

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 1918

```
DEFINE_HOOK_FUNCNARG(
        int64_t,
        otxn_type )
{
    HOOK_SETUP(); // populates memory_ctx, memory, memory_length, applyCtx,
hookCtx on current stack

    if (hookCtx.emitFailure)
        return safe_cast<TxType>(hookCtx.emitFailure-
>getFieldU16(sfTransactionType));

    return applyCtx.tx.getTxnType();
}
```

Severity and Impact Summary

Handling exceptions in the hooks will improve the safety of the calling code.

Recommendation

It is recommended to handle all exceptions in a hook function. For example:

```
#define HANDLE_EXCEPTION_BEGIN try {
#define HANDLE_EXCEPTION_END } catch (const std::exception&) { /*and logs*/
return INTERNAL_ERROR; }

DEFINE_HOOK_FUNCNARG(
        int64_t,
        otxn_type )
{
    HANDLE_EXCEPTION_BEGIN;
```

```
    HOOK_SETUP(); // populates memory_ctx, memory, memory_length, applyCtx,
hookCtx on current stack

    if (hookCtx.emitFailure)
        return safe_cast<TxType>(hookCtx.emitFailure-
>getFieldU16(sfTransactionType));

    return applyCtx.tx.getTxnType();
    HANDLE_EXCEPTION_END;
}
```

INCONSISTENT ORDER OF OPERATIONS

Finding ID: FYEO-XRPL-15
Severity: Low
Status: Remediated

Description

The order of the `executedHookCount_++` operation is inconsistent. In 2 cases it is run before `hook::apply()`. In 1 instance it is run after.

Proof of Issue

File name: src/ripple/app/tx/impl/Transactor.cpp
Line number: 1077

```
results.push_back(
    hook::apply(...));

executedHookCount_++;
```

File name: src/ripple/app/tx/impl/Transactor.cpp
Line number: 1176

```
executedHookCount_++;

std::map<std::vector<uint8_t>, std::vector<uint8_t>> parameters;
if (hook::gatherHookParameters(hookDef, hookObj, parameters, j_))
{
    JLOG(j_.warn())
        << "HookError[]: Failure: gatherHookParameters failed)";
    return;
}
...

try
{
    hook::HookStateMap stateMap;
    hook::HookResult callbackResult =
        hook::apply(...);
```

File name: src/ripple/app/tx/impl/Transactor.cpp
Line number: 1457

```
executedHookCount_++;

std::map<std::vector<uint8_t>, std::vector<uint8_t>> parameters;
if (hook::gatherHookParameters(hookDef, hookObj, parameters, j_))
{
    JLOG(j_.warn())
        << "HookError[]: Failure: gatherHookParameters failed)";
```

```
    return;
}

try
{
    hook::HookResult aawResult =
        hook::apply(...);
```

Severity and Impact Summary

Sometimes the `executedHookCount_` is incremented before `hook::apply()` is called. The code may also `return` between `executedHookCount_++` and `hook::apply()` - therefore incrementing this member variable without ever calling `hook::apply()`.

Recommendation

It is recommended to keep the code consistent.

INCONSISTENT USE OF TRY CATCH WHEN APPLYING HOOKS

Finding ID: FYEO-XRPL-16
Severity: Low
Status: Remediated

Description

The call to `hook::apply()` is wrapped in a try-catch block 2 out of 3 times.

Proof of Issue

File name: src/ripple/app/tx/impl/Transactor.cpp
Line number: 1059

```
results.push_back(
    hook::apply(
        hookDef->getFieldH256(sfHookSetTxnID),
        hookHash,
        ns,
        hookDef->getFieldVL(sfCreateCode),
        parameters,
        hookParamOverrides,
        stateMap,
        ctx_,
        account,
        hasCallback,
        false,
        strong,
        (strong ? 0 : 1UL),
        hook_no - 1,
        provisionalMeta));
```

Severity and Impact Summary

This call to `hook::apply()` is not wrapped in a try-catch block where it is otherwise. Exceptions should be handled the same in all instances.

Recommendation

It is recommended to keep the code consistent and add the try-catch block.

INCORRECT CHECK FOR MAXIMUM STATE MODIFICATIONS

Finding ID: FYEO-XRPL-17
Severity: Low
Status: Remediated

Description

The number of maximum state modifications is `max_state_modifications = 256` but the condition in `set_state_cache()` allows to perform `257` modifications.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 1218

```
if (modified && stateMap.modified_entry_count > max_state_modifications)
```

Severity and Impact Summary

The total number of performed modifications may be greater than the maximum allowed.

Recommendation

It is recommended to change the condition to >=.

POSSIBLE READ OF NONEXISTENT FIELD IN EMIT()

Finding ID: FYEO-XRPL-18
Severity: Low
Status: Remediated

Description

The `sfLastLedgerSequence` field is read before the `isFieldPresent` check.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 3036

```
    uint32_t tx_lls = stpTrans->getFieldU32(sfLastLedgerSequence);
    uint32_t ledgerSeq = applyCtx.app.getLedgerMaster().getValidLedgerIndex()
+ 1;
    if (!stpTrans->isFieldPresent(sfLastLedgerSequence) || tx_lls < ledgerSeq
+ 1)
```

Severity and Impact Summary

Access to a nonexistent field will result in an exception.

Recommendation

It is recommended to validate the existence of the `sfLastLedgerSequence` field before usage.

RAW POINTERS ARE USED FOR WASMEDGE VARIABLES

Finding ID: FYEO-XRPL-19
Severity: Low
Status: Remediated

Description

Raw pointers are used in `validateWasm()` and `executeWasm()` functions. Smart pointer will ensure resource safety.

Proof of Issue

File name: src/ripple/app/hook/applyHook.h

Line number: 451

```
WasmEdge_ConfigureContext* confCtx  = WasmEdge_ConfigureCreate();
WasmEdge_VMContext* vmCtx = WasmEdge_VMCreate(confCtx, NULL);
```

Line number: 485

```
WasmEdge_ConfigureContext* confCtx  = WasmEdge_ConfigureCreate();
WasmEdge_ConfigureStatisticsSetInstructionCounting(confCtx, true);
WasmEdge_VMContext* vmCtx = WasmEdge_VMCreate(confCtx, NULL);
```

Severity and Impact Summary

Possible memory leaks in the case of exceptions due to undeleted resources.

Recommendation

It is recommended to use smart pointers with a custom deleter or wrappers for `WasmEdge_ConfigureContext*` and `WasmEdge_VMContext*`.

RELEASE FLAG IS NOT SPECIFIED FOR RELEASE CONFIGURATION

Finding ID: FYEO-XRPL-20
Severity: Low
Status: Remediated

Description

Release builder uses default parameters that result in the debug build.

Proof of Issue

File name: release-builder.sh

Line number: 160

```
cmake .. -DBoost_NO_BOOST_CMAKE=ON -
DLLVM_DIR=/usr/lib64/llvm13/lib/cmake/llvm/ -
DLLVM_LIBRARY_DIR=/usr/lib64/llvm13/lib/ -
DWasmEdge_LIB=/usr/local/lib64/libwasmedge.a
```

Severity and Impact Summary

Debug build produces bigger executables and enables `assert` statements that may terminate the program if triggered.

Recommendation

It is recommended to add `-DCMAKE_BUILD_TYPE=Release` to the release configuration and check all functionality guarded by the `NDEBUG` flag.

THE EXTERNREF TYPE IS NOT PROCESSED IN CHECK_GUARD

Finding ID: FYEO-XRPL-21
Severity: Low
Status: Remediated

Description

The block type should be in one of the three categories: - empty type `0x40` - value type - signed integer

Value type includes: - Number Types - `i32`, `i64`, `f32`, `f64` - Vector Types - `v128` - Reference Types - `funcref`, `externref`

The doesn't process `externref` type that has the code `0x6F`.

Proof of Issue

File name: src/ripple/app/hook/Guard.h

Line number: 341

```
            // discard the block return type
            uint8_t block_type = hook[i];
            if ((block_type >= 0x7CU && block_type <= 0x7FU) ||
                 block_type == 0x7BU || block_type == 0x70U  ||
                 block_type == 0x7BU || block_type == 0x40U)
            {
                ADVANCE(1);
            }
            else
            {
                SIGNED_LEB();
            }
```

Severity and Impact Summary

If the block type is `externref`, the `check_guard` function may fail.

Recommendation

It is recommended to check for `0x6F` in the if statement.

References

- https://webassembly.github.io/spec/core/binary/instructions.html#binary-blocktype
- https://webassembly.github.io/spec/core/binary/types.html

THE TECHOOK_REJECTED STATE IS VALID FOR EXECUTING HOOKS

Finding ID: FYEO-XRPL-22
Severity: Low
Status: Remediated

Description

The code accepts two valid states for hook execution. These are `tesSUCCESS` and `tecHOOK_REJECTED`. The `tecHOOK_REJECTED` may not be appropriate anymore.

Proof of Issue

File name: src/ripple/app/tx/impl/Transactor.cpp
Line number: 1553

```
if (hooksEnabled && (result == tesSUCCESS || result == tecHOOK_REJECTED))
```

Severity and Impact Summary

As discussed, `tesSUCCESS` may be the only valid state that can actually occur. Still, the `tecHOOK_REJECTED` could be excluded.

Recommendation

Exclude `tecHOOK_REJECTED` from this condition.

UNHANDLED NEGATIVE CONDITION IN INCREMENTREFERENCECOUNT

Finding ID: FYEO-XRPL-23
Severity: Low
Status: Remediated

Description

The reference count may not be increased if the entry or the field doesn't exist. This scenario is not handled in the code.

Proof of Issue

File name: src/ripple/app/tx/impl/SetHook.cpp

Line number: 922

```
void incrementReferenceCount(std::shared_ptr<STLedgerEntry>& sle)
{
    if (sle && sle->isFieldPresent(sfReferenceCount))
        sle->setFieldU64(sfReferenceCount, sle->getFieldU64(sfReferenceCount)
+ 1);
}
```

Severity and Impact Summary

An unhandled negative condition may create an issue that is hard to track.

Recommendation

It is recommended to create the field if it is absent or log the error.

UNPROCESSED WRITE ERROR IN OTXN_FIELD() AND SLOT()

Finding ID: FYEO-XRPL-24
Severity: Low
Status: Remediated

Description

When the `otxn_field()` or the `slot()` function processes the default STI account, the write `-1` length will be converted to `uint32_t` resulting in the unprocessed `out of bounds memory access` error from `WasmEdge_MemoryInstanceSetData()`.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Serialization is done on:

Line number: 2109

```
    Serializer s;
    field.add(s);
```

If the `field` is `STAccount`, it may return `0` size:

File name: src/ripple/protocol/impl/STAccount.cpp

Line number: 89

```
int const size = isDefault() ? 0 : uint160::bytes;
```

The result of write operation would be an error that is not checked:

File name: src/ripple/app/hook/Macro.h

Line number: 166

```
    WasmEdge_MemoryInstanceSetData(memoryCtx, \
            reinterpret_cast<const uint8_t*>(host_src_ptr), guest_dst_ptr,
bytes_to_write);\
    bytes_written += bytes_to_write;\
```

Severity and Impact Summary

The hook will return the number of bytes written. In this case, it is `-1` which is the code of the `OUT_OF_BOUNDS` error.

Recommendation

It is recommended to verify that serialized data exists and to check the `WasmEdge_MemoryInstanceSetData()` return code.

SLOT_INTO MAY BE GREATER THAN MAX_SLOTS

Finding ID: FYEO-XRPL-25
Severity: Low
Status: Remediated

Description

If all slots are busy, then the loop will increment `slot_into` up to the `hook_api::max_slots + 1`.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 557

```
    do
    {
        slot_into = ++hookCtx.slot_counter;
    }
    while (hookCtx.slot.find(slot_into) != hookCtx.slot.end() &&
        // this condition should always be met, if for some reason, somehow it
is not
        // then we will return the final slot every time.
        hookCtx.slot_counter <= hook_api::max_slots);

    return slot_into;
```

Severity and Impact Summary

The total amount of slots may be greater than the maximum allowed.

Recommendation

It is recommended to change the condition to `hookCtx.slot_counter < hook_api::max_slots;`

ADD_TSH MACRO CAN BE REPLACED WITH LAMBDA

Finding ID: FYEO-XRPL-26
Severity: **Informational**
Status: **Remediated**

Description

It is preferable to use lambda or inline functions instead of a macro in C++.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 55

```
        #define ADD_TSH(acc, rb)\
        {\
            auto acc_r = acc;\
            if (acc_r != *otxnAcc)\
            {\
                if (tshEntries.find(acc_r) != tshEntries.end())\
                {\
                    if (tshEntries[acc_r].second == false && (rb))\
                        tshEntries[acc_r].second = true;\
                }\
                else\
                    tshEntries.emplace(acc_r, std::pair<int, bool>{upto++,
(rb)});\
            }\
        }
```

Severity and Impact Summary

Macro functions are dangerous because their use resembles that of real functions, but they have different semantics.

Recommendation

It is recommended to replace the macro with a lambda. For example:

```
auto const ADD_TSH = [&otxnAcc, &tshEntries, &upto](const AccountID& acc_r,
bool rb) {
      if (acc_r != *otxnAcc)
      {
          if (tshEntries.find(acc_r) != tshEntries.end())
          {
              tshEntries[acc_r].second |= rb;
          }
```

```
            else
                tshEntries.emplace(acc_r, std::make_pair(upto++, rb));
        }
    };
```

CONDITIONS DUPLICATION IN HOOK_FLOAT

Finding ID: FYEO-XRPL-27
Severity: **Informational**
Status: **Remediated**

Description

The check for `INVALID_FLOAT` is done twice in `get_exponent()` and also twice in `get_mantissa()`.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 331, 344

```
if (float1 < 0)
   return INVALID_FLOAT;
```

Line number: 335, 348

```
if (float1 < 0) return INVALID_FLOAT;
```

Severity and Impact Summary

Code that has no effect is typically the result of a coding error. Such code is usually optimized out of a program during compilation. However, to improve readability and ensure that logic errors are resolved, they should be identified, understood, and eliminated.

Recommendation

It is recommended to remove the duplicated conditions.

CREATE ENUM FOR ERRORS RETURNED FROM GET_STOBJECT_LENGTH

Finding ID: FYEO-XRPL-28
Severity: **Informational**
Status: **Remediated**

Description

Raw numbers are used to express different errors and described using commentaries instead of in-code structures.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 3389

```
// negative indicates error
// -1 = unexpected end of bytes
// -2 = unknown type (detected early)
// -3 = unknown type (end of function)
// -4 = excessive stobject nesting
// -5 = excessively large array or object
inline int32_t get_stobject_length (
```

Severity and Impact Summary

Named constants will improve the readability and maintainability of the code.

Recommendation

It is recommended to combine errors in an enum instead of using raw numbers.

IMPROVE WASMEDGE ERROR LOGS

Finding ID: FYEO-XRPL-29
Severity: **Informational**
Status: **Remediated**

Description

WasmEdge error message can be retrieved using `WasmEdge_ResultGetMessage()` but this is done only in `executeWasm`.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 455, 460

```
            if (!WasmEdge_ResultOK(res))
                ret = "VMLoadWasmFromBuffer failed";
            else
            {
                res = WasmEdge_VMValidate(vmCtx);
                if (!WasmEdge_ResultOK(res))
                    ret = "VMValidate failed";
            }
```

Severity and Impact Summary

Logging a specific error message will help to resolve WasmEdge-related problems.

Recommendation

It is recommended to add the message from `WasmEdge_ResultGetMessage()` in the return string.

INCORRECT ORDER OF MEMBERS INITIALIZATION IN WASMBLKINF

Finding ID: FYEO-XRPL-30
Severity: **Informational**
Status: **Remediated**

Description

The order in the initializer list for `WasmBlkInf` struct is different from the order of member declaration.

Proof of Issue

File name: src/ripple/app/hook/Guard.h

Line number: 150

Order of struct members:

```
    WasmBlkInf* parent;
    std::vector<WasmBlkInf*> children;
```

Line number: 165

Order in initializer list:

```
            children({}),
            parent(parent_),
```

Severity and Impact Summary

The `parent` member will be initialized twice.

Recommendation

It is recommended to write the initializer list in canonical order.

OVERLAPPING MEMORY CHECK CAN BE OPTIMIZED

Finding ID: FYEO-XRPL-31
Severity: **Informational**
Status: **Remediated**

Description

The `overlapping_memory()` function uses a nested loop that performs the same check, in half of the cases.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 3727

```
    for (uint64_t i = 0; i < regions.size(); i+= 2)
    {
        //...
        for (uint64_t j = 0; j < regions.size(); j+= 2)
```

Severity and Impact Summary

The program can use fewer resources for the overlapping memory check.

Recommendation

It is recommended to not iterate through already checked elements. For example:

```
    for (uint64_t i = 0; i < regions.size() - 2; i+= 2)
    {
        //...
        for (uint64_t j = i + 2; j < regions.size(); j+= 2)
```

PROCESSING OF EMITTED TRANSACTION IN FINALIZEHOOKRESULT CAN BE OPTIMIZED

Finding ID: FYEO-XRPL-32
Severity: **Informational**
Status: **Remediated**

Description

Data serialization is needed only for not emitted transactions but is done regardless.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 1671-1675

```
            std::shared_ptr<const ripple::STTx> ptr = tpTrans-
>getSTransaction();

            ripple::Serializer s;
            ptr->add(s);
            SerialIter sit(s.slice());
```

But it is used only in `if` block:

Line number: 1680

```
if (!sleEmitted)
```

Severity and Impact Summary

The program can use fewer resources for processing already emitted transactions.

Recommendation

It is recommended to move data serialization into the `if` block.

REDUNDANT IF CHECK

Finding ID: FYEO-XRPL-33
Severity: **Informational**
Status: **Remediated**

Description

The condition of this if check is redundant. The variable `canRollback` will be false in the else block.

Proof of Issue

File name: src/ripple/app/tx/impl/Transactor.cpp
Line number: 1342

```
if (canRollback) { ... }
else
{
    // this is a collect call so first check if the tsh can accept
    uint32_t tshFlags = tshAcc->getFieldU32(sfFlags);
    if (!canRollback && !(tshFlags & lsfTshCollect))
```

Severity and Impact Summary

The code can be simplified but has no impact otherwise.

Recommendation

It is recommended to remove redundant checks.

UNCLEAR USAGE OF LITERAL VALUE IN SETHOOK()

Finding ID: FYEO-XRPL-34
Severity: **Informational**
Status: **Remediated**

Description

The code uses the number 4 as the limit of iterations without proper context or explanation.

Proof of Issue

File name: src/ripple/app/tx/impl/SetHook.cpp

Line number: 1648

```
        for (int i = 0; i < 4; ++i)
        {
            if (oldHooks && i < oldHookCount)
                oldHookReserve += computeHookReserve(((*oldHooks).get())[i]);

            if (i < newHooks.size())
                newHookReserve += computeHookReserve(newHooks[i]);
        }
```

Severity and Impact Summary

Using numeric literals makes code more difficult to read and understand. The context doesn't provide enough information to verify this limitation's correctness.

Recommendation

It is recommended to use a constant instead of a literal value.

References

- https://wiki.sei.cmu.edu/confluence/display/c/DCL06-C.+Use+meaningful+symbolic+constants+to+represent+literal+values

UNNECESSARY CAST TO POINTER

Finding ID: FYEO-XRPL-35
Severity: **Informational**
Status: **Remediated**

Description

Some functions perform `dynamic_cast` which is not necessary.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 1445

```
STObject const* hookObj = dynamic_cast<STObject const*>(&hook);
```

Line number: 1478

```
STObject const* hookGrantObj = dynamic_cast<STObject const*>(&hookGrant);
```

Line number: 1588

```
auto const& hookParameterObj = dynamic_cast<STObject const*>(&hookParameter);
```

Line number: 1599

```
auto const& hookParameterObj = dynamic_cast<STObject const*>(&hookParameter);
```

Line number: 5149

```
auto const& hookObj = dynamic_cast<STObject const*>(&hook);
```

Severity and Impact Summary

Code that has no effect is typically the result of a coding error. Such code is usually optimized out of a program during compilation. However, to improve readability and ensure that logic errors are resolved, they should be identified, understood, and eliminated.

Recommendation

It is recommended to use reference directly, without casting to a pointer.

UNNECESSARY CAST TO POINTER IN GETTRANSACTIONALSTAKEHOLDERS

Finding ID: FYEO-XRPL-36
Severity: **Informational**
Status: **Remediated**

Description

The function performs `dynamic_cast` that is not necessary.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 277

```
        for (auto const& e : signerEntries)
        {
            auto const& entryObj = dynamic_cast<STObject const*>(&e);
            if (entryObj->isFieldPresent(sfAccount))
                ADD_TSH(entryObj->getAccountID(sfAccount), canRollback);
        }
```

Severity and Impact Summary

Code that has no effect is typically the result of a coding error. Such code is usually optimized out of a program during compilation. However, to improve readability and ensure that logic errors are resolved, they should be identified, understood, and eliminated.

Recommendation

It is recommended to use reference directly, without casting to a pointer. For example:

```
  for (auto const& entryObj : signerEntries)
  {
      if (entryObj.isFieldPresent(sfAccount))
          ADD_TSH(entryObj.getAccountID(sfAccount), canRollback);
  }
```

UNREACHABLE CONDITION IN STATE_FOREIGN

Finding ID: FYEO-XRPL-37
Severity: **Informational**
Status: **Remediated**

Description

An unsigned variable is checked to be greater than 0.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 1778

```
uint32_t aread_ptr
{
//...
    if (aread_ptr == 0)
    {
//...
    }
    else if (aread_ptr > 0)
    {
//...
    }
    else // unreachable
        return INVALID_ARGUMENT;
```

Severity and Impact Summary

Code that has no effect is typically the result of a coding error. Such code is usually optimized out of a program during compilation. However, to improve readability and ensure that logic errors are resolved, they should be identified, understood, and eliminated.

Recommendation

It is recommended to remove the unreachable condition.

UNREACHABLE RETURN IN ETXN_NONCE AND LEDGER_NONCE

Finding ID: FYEO-XRPL-38
Severity: **Informational**
Status: **Remediated**

Description

`etxn_nonce()` and `ledger_nonce()` functions use `WRITE_WASM_MEMORY_AND_RETURN` macro that will return the number of bytes written. The code that follows this macro will not be executed.

Proof of Issue

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 3228

```
    WRITE_WASM_MEMORY_AND_RETURN(
        write_ptr, 32,
        hash.data(), 32,
        memory, memory_length);

    return 32;
```

Line number: 3269

```
    WRITE_WASM_MEMORY_AND_RETURN(
        write_ptr, 32,
        hash.data(), 32,
        memory, memory_length);

    return 32;
```

Severity and Impact Summary

Code that has no effect is typically the result of a coding error. Such code is usually optimized out of a program during compilation. However, to improve readability and ensure that logic errors are resolved, they should be identified, understood, and eliminated.

Recommendation

It is recommended to remove the unreachable return statement.

UNUSED CODE

Finding ID: FYEO-XRPL-39
Severity: **Informational**
Status: **Remediated**

Description

Some code expressions are not used.

Proof of Issue

File name: src/ripple/app/hook/applyHook.h

Line number: 87

```
#define TER_TO_HOOK_RETURN_CODE(x)\
    (((TERtoInt(x)) << 16)*-1)
```

File name: src/ripple/app/hook/impl/applyHook.cpp

Line number: 1949

```
    auto const& txID =
        hookCtx.emitFailure
            ? applyCtx.tx.getFieldH256(sfTransactionHash)
            : applyCtx.tx.getTransactionID();
```

File name: src/ripple/app/tx/impl/Invoke.cpp

Line number: 41

```
auto& j = ctx.j;
```

Severity and Impact Summary

Unused code adds to the complexity of the codebase, making it more difficult to understand and maintain. It can also be an indication of an error.

Recommendation

It is recommended to remove unused code.

VARIABLE NAME IS THE SAME AS NAMESPACE

Finding ID: FYEO-XRPL-40
Severity: **Informational**
Status: **Remediated**

Description

The same name `hook` is used for a variable and a namespace, and both are used in the same code.

Proof of Issue

File name: src/ripple/app/hook/Guard.h

Line number: 275

```
    std::vector<uint8_t> const& hook,
```

Line number: 467

```
GUARDLOG(hook::log::CALL_ILLEGAL)
```

Line number: 769

```
    std::vector<uint8_t> const& hook,
```

Severity and Impact Summary

The problem can lead to confusion and errors in the code.

Recommendation

It is recommended to rename the variable `hook`.

WASMBLKINF DESTRUCTOR CAN BE SIMPLIFIED

Finding ID: FYEO-XRPL-41
Severity: **Informational**
Status: **Remediated**

Description

The additional logic was added to process the destruction of the `WasmBlkInf` children, but everything can be handled by the destructor alone.

Proof of Issue

File name: src/ripple/app/hook/Guard.h

Line number: 180-194

```
    void free_children(WasmBlkInf* blk)
    {
        for (WasmBlkInf* child : blk->children)
            free_children(child);
        delete blk;
    }


    ~WasmBlkInf()
    {
        // only the root is responsible for freeing
        if (is_root)
            for (WasmBlkInf* child : children)
                free_children(child);
    }
```

Severity and Impact Summary

Overcomplicated code is often difficult to understand and maintain. It can also introduce unnecessary bugs and reduce the performance of the application.

Recommendation

It is recommended to simplify the destructor:

```
    ~WasmBlkInf()
    {
        for (WasmBlkInf* child : children)
           delete child;
    }
```

Our Process

Methodology

FYEO Inc. uses the following high-level methodology when approaching engagements. They are broken up into the following phases.

Figure 2: Methodology Flow

Kickoff

The project is kicked off as the sales process has concluded. We typically set up a kickoff meeting where project stakeholders are gathered to discuss the project as well as the responsibilities of participants. During this meeting we verify the scope of the engagement and discuss the project activities. It's an opportunity for both sides to ask questions and get to know each other. By the end of the kickoff there is an understanding of the following:

- Designated points of contact
- Communication methods and frequency
- Shared documentation
- Code and/or any other artifacts necessary for project success
- Follow-up meeting schedule, such as a technical walkthrough
- Understanding of timeline and duration

Ramp-up

Ramp-up consists of the activities necessary to gain proficiency on the project. This can include the steps needed for familiarity with the codebase or technological innovation utilized. This may include, but is not limited to:

- Reviewing previous work in the area including academic papers
- Reviewing programming language constructs for specific languages
- Researching common flaws and recent technological advancements

Review

The review phase is where most of the work on the engagement is completed. This is the phase where we analyze the project for flaws and issues that impact the security posture. Depending on the project this may include an analysis of the architecture, a review of the code, and a specification matching to match the architecture to the implemented code.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review of the original protocol
2. Review of the code written for the project
3. Compliance of the code with the provided technical documentation

The review for this project was performed using manual methods and utilizing the experience of the reviewer. No dynamic testing was performed, only the use of custom-built scripts and tools were used to assist the reviewer during the testing. We discuss our methodology in more detail in the following sections.

Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues
- Poor coding practices and unsafe behavior
- Leakage of secrets or other sensitive data through memory mismanagement
- Susceptibility to misuse and system errors
- Error management and logging

This list is general and not comprehensive, meant only to give an understanding of the issues we are looking for.

Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for things such as:

- Proper implementation of the documented protocol phases
- Proper error handling
- Adherence to the protocol logical description

Reporting

FYEO Inc. delivers a draft report that contains an executive summary, technical details, and observations about the project.

The executive summary contains an overview of the engagement including the number of findings as well as a statement about our general risk assessment of the project. We may conclude that the overall risk is low but depending on what was assessed we may conclude that more scrutiny of the project is needed.

We report security issues identified, as well as informational findings for improvement, categorized by the following labels:

- Critical
- High
- Medium
- Low
- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

As we perform the audit, we may identify issues that aren't security related, but are general best practices and steps that can be taken to lower the attack surface of the project. We will call those out as we encounter them and as time permits.

As an optional step, we can agree on the creation of a public report that can be shared and distributed with a larger audience.

Verify

After the preliminary findings have been delivered, this could be in the form of the approved communication channel or delivery of the draft report, we will verify any fixes within a window of time specified in the project. After the fixes have been verified, we will change the status of the finding in the report from open to remediated.

The output of this phase will be a final report with any mitigated findings noted.

Additional Note

It is important to note that, although we did our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits along with the scope of the agreement.

While assessing the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. This is a solid baseline for severity determination.

THE CLASSIFICATION OF VULNERABILITIES

Security vulnerabilities and areas for improvement are weighted into one of several categories using, but is not limited to, the criteria listed below:

Critical – vulnerability will lead to a loss of protected assets

- This is a vulnerability that would lead to immediate loss of protected assets
- The complexity to exploit is low
- The probability of exploit is high

High - vulnerability has potential to lead to a loss of protected assets

- All discrepancies found where there is a security claim made in the documentation that cannot be found in the code
- All mismatches from the stated and actual functionality
- Unprotected key material
- Weak encryption of keys
- Badly generated key materials
- Txn signatures not verified
- Spending of funds through logic errors
- Calculation errors overflows and underflows

Medium - vulnerability hampers the uptime of the system or can lead to other problems

- Insecure calls to third party libraries
- Use of untested or nonstandard or non-peer-reviewed crypto functions
- Program crashes, leaves core dumps or writes sensitive data to log files

Low – vulnerability has a security impact but does not directly affect the protected assets

- Overly complex functions
- Unchecked return values from 3rd party libraries that could alter the execution flow

Informational

- General recommendations