SMART CONTRACT AUDIT

ZOKYO.

November 11th 2022 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

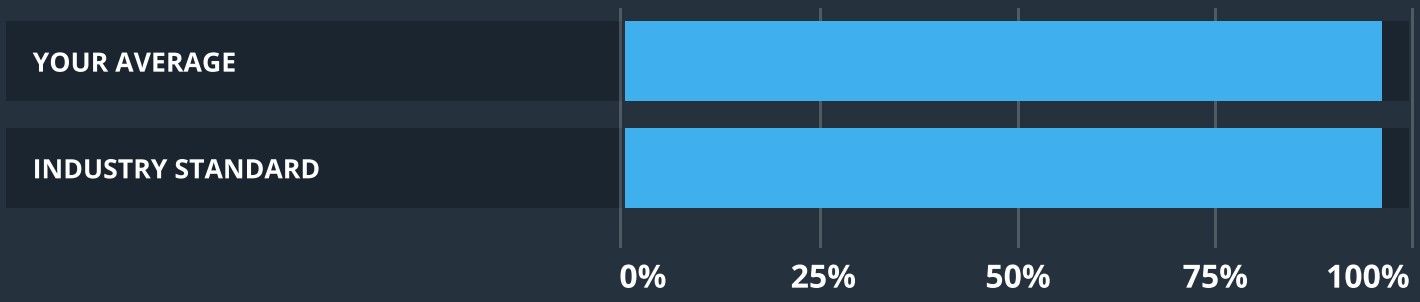
TECHNICAL SUMMARY

This document outlines the overall security of the Symbiosis smart contracts evaluated by the Zokyo Security team.

The scope of this audit was to analyze and document the Symbiosis smart contract codebase for quality, security, and correctness.

Contract Status

Testable Code



The testable code is 95%, which is corresponds the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the NEAR network's fast-paced and rapidly changing environment, we at Zokyo recommend that the Symbiosis team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contracts was taken from the Symbiosis repository.
Repo: git: https://github.com/symbiosis-finance/near-contracts-audit
Contracts are written on Rust and are prepared for the NEAR blockchain

Initial commit:
dev branch, f70cda0cd54dc498dcabd3d609b78153a8637591
Final commit:
dev branch, 1333ce515b0a611e0b14b25d2ce24153e6b9c81e

Within the scope of this audit, Zokyo auditors reviewed the following contract(s):

contracts/bridge
- bridge/src/lib.rs
- bridge/src/receive_request.rs
- bridge/src/mpc.rs
- bridge/src/oracle_request.rs
- bridge/src/storage.rs
- bridge/src/transmit_request.rs
- bridge/src/transmitter.rs

contracts/portal
- portal/src/ft_receiver.rs
- portal/src/lib.rs
- portal/src/storage.rs
- portal/src/pause.rs
- portal/src/synthesize.rs
- portal/src/unsynthesize.rs
- portal/src/token.rs

contracts/metarouter
- bridge/src/lib.rs
- bridge/src/meta_route.rs
- bridge/src/external_call.rs
- bridge/src/ft_receiver.rs
- bridge/src/util.rs

AUDITING STRATEGY AND TECHNIQUES APPLIED

Throughout the review process, Zokyo Security ensures that the contract:

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of resources, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of Symbiosis smart contracts. To do so, the code was reviewed line by line by our smart contract developers, who documented any issues as they were discovered. A part of this work included writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.
2	Cross-comparison with other, similar smart contracts by industry leaders.
3	Testing contract logic against common and uncommon attack vectors.
4	Thorough manual review of the codebase, line by line.

EXECUTIVE SUMMARY

The Symbiosis contracts represent the NEAR side of the Symbiosis bridge. The core bridge contract was created for the relayer to call other contracts (when triggered from the Relayer side). The portal mints and burns wrapped tokens for swaps, and Metarouter is used for calling other NEAR contracts (any contracts at all).

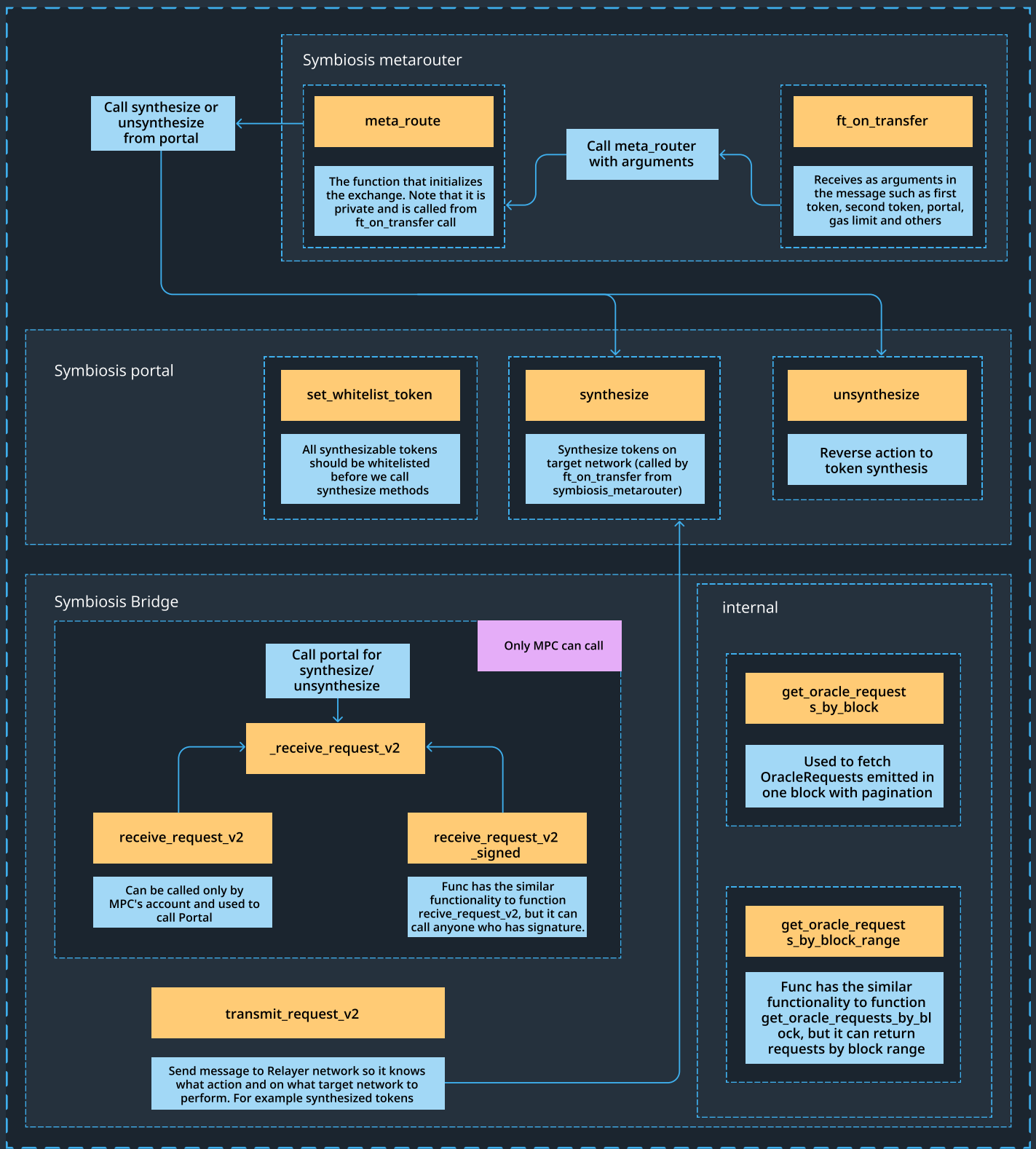
Zokyo Security checked the codebase and deeply analized the business logic of the contracts. The team also provided a set of unit and scenario tests, checked existing tests coverage, and provided extensive exploratory testing of contracts. As a result of the performed business logic analysis, the team of auditors prepared the detailed scheme of actions, roles, and processes within the contracts' system. The team has also checked protocol's sim and sandbox tests.

The overall code quality is good, it matches Rust best practices and has a sufficient suite of tests prepared. Nevertheless, the team has provided extensive checks against the roles and access control system, funds flow, gas usage, cross contract calls structure, and other apects regarding both business logic and NEAR-related best practices. The team detected 2 Medium-level problems (with roles management and NEAR Rust types structure), which were fixed by the Symbiosis team.

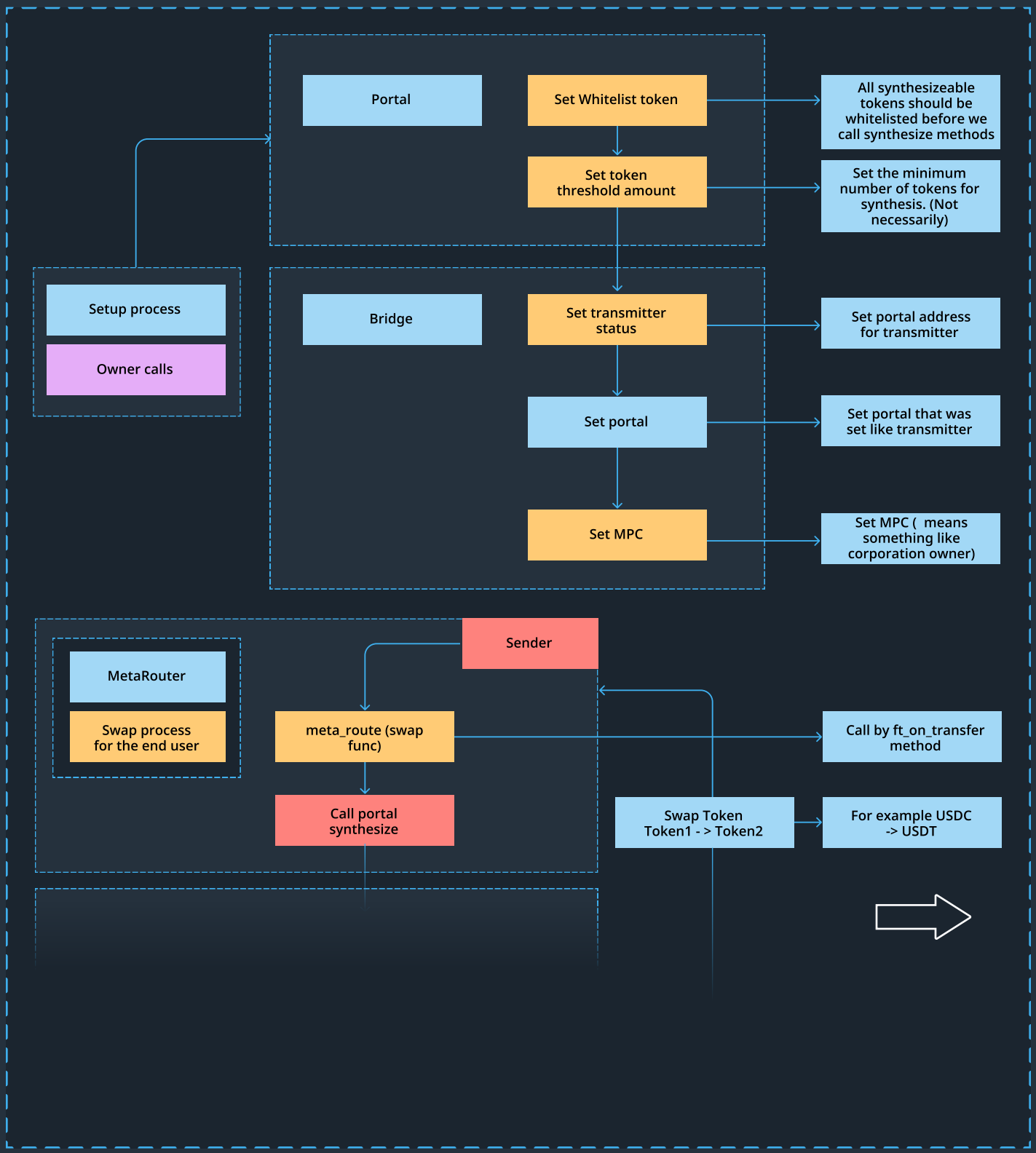
The main issue present in the project was the problem with the dependencies tree preparation. It influences the compilation and launch of modules connected to the Sandbox tests prepared by the team. Therefore, the team of auditors dedicated a lot of time to researching the reasons of the problem. Tests were verified by the Symbiosys team that prepared proofs that tests could be run on it side. Yet, Zokyo Security struggled with running tests on our side. The issue is connected to the work of the package tree dependencies building engine, which is out of the scope of the project and is connected purely to the NEAR toolkit. Thus, after several testing steps, the issue was resolved and the team verified tests suite running. Nevertheless, the common recommendation for the Symbiosis team is to pay attention on the proper package management sanitizing and provide an alternative testing line with the machines with different OSes.

Overall, the security of the project can be evaluated as High, the code conforms standard security checklist, and user flow of the protocol was checked against the auditors' list of potential problems and loopholes.

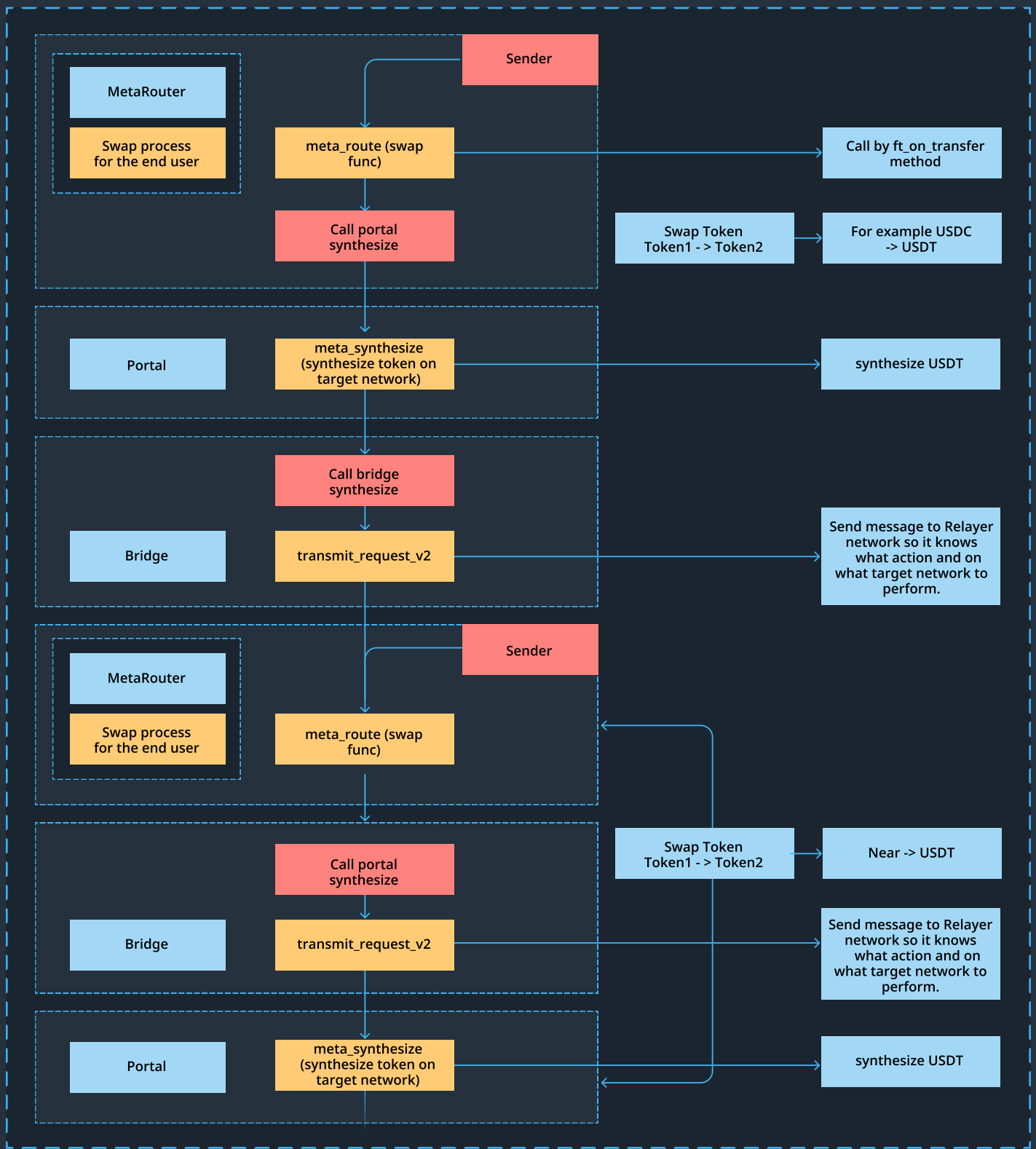
PROTOCOL OVERVIEW



Protocol Core Flow



Protocol Core Flow



STRUCTURE AND ORGANIZATION OF THE DOCUMENT

For the ease of navigation, sections are arranged from the most to the least critical one. Issues are tagged as “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Customer’s side or remains disregarded by the Customer. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

MEDIUM-1 | RESOLVED

Lost role check on a portal set

Line. 78, symbiosis-bridge/src/lib.rs, function ‘set_portal’

Missing role for the method. Therefore, the method might generate some problems since an attacker can change the portal account in any time. An incorrect portal address will break the work of the receive request functionality since it relies on the internal portal state. The issue directly impacts the main funcionality but is marked as Medium since the owner of the protocol can still change the address back.

Recommendation:
Add an additional check depending on who can change the portal address

MEDIUM-2 | RESOLVED

Wrong type in params

Line. 109,121, symbiosis-portal/src/synthesize.rs, function ‘revert_synthesize’

The NEAR request requires all money-related types to be passed as a string (U128 to be exact), and sdk relies on this type. Currently, this error will appear only at runtime since it’s not related to the compilation stage.

Recommendation:
Pass the amount as NEAR U128 instead of standard u128 type

LOW-1 | VERIFIED

Value overflow is possible

Line. 190,208 symbiosis-metarouter/src/meta_router.rs, function ‘balance_callback’

A value overflow can be used to attack a smart contract

Recommendation:
It is necessary to check the value of the ‘second_amout_in’ variable

From the client:
The Symbiosis team stated that it is a private method that can be called only by the metarouter contract and that it fetches balances of the previously whitelisted tokens. Thus, they verify the current behaviour.

INFO-1 | RESOLVED

Dependency issue

This issue is connected to dependency problems in NEAR packages. This issue is not connected with the security of contracts, thus it is marked as informational. However, this case may cause troubles for the environment preparation and further development. The dependent ‘workspaces’ package (tests target) requires another package that is called ‘near-jsonrpc-client’ and unfortunately it has some floats. Since each package of the project has its own list of required packages, there can almost always be cases when they require the same package but of different versions. Usually, there is no problem with different versions, except for the current case. The cargo module can increase the package version in within the major version of it because the maintenance team of each package cannot change the type or API of their crate. So the version within a major release should be interchangeable. But crates.io does not have it as a hard restriction. The audited project has 2 versions of near-primitives, which is 0.13.0 and 0.14.0 and both of them can be replaced by each other. But because of Cargo's ability to increase versions within the major one, it updates the ‘near-jsonrpc-client’ package to the 4.0.0-beta.0 version, which requires `near-primitives` 0.14.0.

In the case explained above, the project is compilable and tests work fine. But in some cases, Cargo can pick up not the 4.0.0-beta.0 version of Near RPC, but the one before it, 4.0.0. And it’s crucial for the project because it requires near-primitives of a higher version, 0.15.0.

And since 0.15.0 is not compatible with the previous versions of this package (0.13.0 and 0.14.0), the project becomes uncompilable.

The issue is caused by the work of a package manager engine. The most possible reason is that it cannot distinguish 4.0.0-beta.0 and 4.0.0 because of its internal parser functionality. One of the auditors suggests that since '0' and '0-beta.0' sub-versions look the same for Rust, it picks whatever it needs.

If this problems appears, you need to hardly set version of near RPC in Cargo.lock file.
Here is the correct version:

```
[[package]]
name = "near-jsonrpc-client"
version = "0.4.0-beta.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum =
"bba462f54bc35289a1013ed3a2ebfa67cc8b12699f81c12dd67687f200c7b871"
```

The issue is marked as informational and is mentioned as a warning for the future environments preparation. Therefore, it is recommended for the team to take it into consideration and properly sanitize the Cargo.lock file.

INFO-2 | RESOLVED

Failing test environment

The team of auditors was facing a big problem with running sim tests that were dedicated to the dev team for at least one month. Those tests use a new NEAR test feature that is called 'sandbox'. Regular tests are based on direct calls to deployed contracts in local blockchain nodes. Sandbox uses an external process for a full blockchain node with a validator. And the environment described in Symbiosis project didn't work on several different auditor's workstations probably because of the flows in packages management or in parallel work between nodes, but it's conjectured.

Error states, that the process refused connection to local sandbox node. Example:
called `Result::unwrap()` on an `Err` value: Failed to connect to RPC service http://
localhost:19396 within 10 seconds:
TransportError(SendError(PayloadSendError(reqwest::Error { kind: Request, url: Url { scheme:
"http", cannot_be_a_base: false, username: "", password: None, host:
Some(Domain("localhost")), port: Some(19396), path: "/", query: None, fragment: None },
source: hyper::Error(Connect, ConnectError("tcp connect error", Os { code: 111, kind:
ConnectionRefused, message: "Connection refused" })) })))
thread 'test_bridge_set_mpc_signed' panicked at 'called `Result::unwrap()` on an `Err` value:
Failed to connect to RPC service http://localhost:19396 within 10 seconds:
TransportError(SendError(PayloadSendError(reqwest::Error { kind: Request, url: Url { scheme:
"http", cannot_be_a_base: false, username: "", password: None, host:
Some(Domain("localhost")), port: Some(19396), path: "/", query: None, fragment: None },
source: hyper::Error(Connect, ConnectError("tcp connect error", Os { code: 111, kind:
ConnectionRefused, message: "Connection refused" })) }))', symbiosis-tests/tests/
suite.rs:29:50

This issue is present on several different machines, but others still show this:

`Error: Action #0: CompilationError(PrepareError(Instantiate))`

And none of the tests work because of it. Auditors tried to run sim tests sequentially - one by one in seperately built modules. This helps, and sometimes, there are successful runs, but runtime errors still appear.
Note: there were no problems with sandbox tests in clear environment, they appear after all Symbiosis dependencies are applied.
More info in the issue below.
From the client:
The Symbiosis team verifies that all tests can be run on their local machines - Macbooks with M2 and Intel i9 chips.
Post-audit:
The team of auditors was able to launch sandbox tests within the Symbiosis environment. The environment was tested on several machines until the issue was resolved after the package engine was able to build the correct dependancy tree with NEAR packages. Thus, it is highly recommended for the Symbiois team to pay attention to the proper packages dependencies sanitizing since there is a chance to get a similar issue in the future.

INFO-3 | RESOLVED

As for the clarification for the sim test problem mentioned above (Info-2), Zokyo Security wants to clarify the steps they took.
The team faced two possible explanations of why sandbox sim tests can fail to run. The first is an error in connection to the sandbox node and the second one is a strange error without any additional info beyond it. Since the second one is more common, we spend additional time on trying to reanimate sim tests.
The problem appears on different hardware starting from some Intel-based laptops to desktop platforms (Ryzen 5 CPU on B550 chipset, Intel Core i5-10XXXF on some H series chipset). In case the problem was connected with the sandbox itself, one of othe auditors spent some time on recereating a test suite (the core functionality for setuping the test env) to downgrade it to be able to use old library - 'near-skd-sim' version 4.0.0-pre-9 (latest for the moment). And it produced a bit more clear error message:

Default sandbox error output:
`Error: Action #0: CompilationError(PrepareError(Instantiate))`

NEAR sim sdk output:
ExecutionOutcome { logs: [], receipt_ids: [
`8W2ztqWuBVUMK9i1msS9rVDWEW1DJEfbDEpfTEBqPxvz`,
], burnt_gas: 2428064099776, tokens_burnt: 242806409977600000000, status: Failure(Action #0: PrepareError: Error happened during instantiation.) }

Therefore, the research narrowed the problem to the instantiation problem. It lead to the issue already described in the NEAR documentation:

https://docs.rs/near-vm-errors/2.2.0/near_vm_errors/enum.PrepareError.html
Error happened during instantiation.
This might indicate that start function trapped, or module isn't instantiable and/or unlinkable.

So now, the team of auditors knew that this error occurs when trying to start a VM. At this point, it was hard to tell if tests had any influence on it. Thus, auditors investigated how this error throws in the source code of the VM.

Nearcore sources states:
https://github.com/near/nearcore/blob/05e932f80642be7b1c7893fb090f48dd097df6a4/runtime/near-vm-runner/src/prepare.rs#L495

In this file, we can see some cases of the error and how tests cover this case:

```
// nothing can be imported from non-"env" module for now.
    let r =
        parse_and_prepare_wat(r#"(module (import "another_module" "memory" (memory
1 1)))"#);
    assert_matches!(r, Err(PrepareError::Instantiate));

    let r = parse_and_prepare_wat(r#"(module (import "env"
"gas" (func (param i32))))"#);
    assert_matches!(r, Ok(_));
```

Therefore, it's most likely to be some kind of a linkage error.

From the client:
The Symbiosis team verifies that all tests can be run on their local machines - Macbooks with M2 and Intel i9 chips.

Post-audit:
Zokyo Security was able to launch sandbox tests within the Symbiosis environment. The environment was tested on several machines until the issue was resolved after the package engine was able to build the correct dependancy tree with the Near packages. Thus, it is highly recommended for the Symbiois team to pay attention for the proper packages dependencies sanitizing since there is a chance to get a similar issue in the future.

INFO-4 | RESOLVED

Unnecessary late initialization

Line. 196 symbiosis-metarouter/src/meta_router.rs, function ‘balance_callback’

Checks for late initializations that can be replaced by a let statement with an initializer.
The issue is marked as informational as it refers to code best practices and performance.

Recommendation:
Declare ‘second_promise’ like this

let second_promise = if args.second_function_name.len() > 0 {

INFO-5 | VERIFIED

Gas test problem

Line. 90, symbiosis-metarouter/src/ft_receiver.rs, function ‘ft_on_transfer’

92,symbiosis-metarouter/src/meta_router.rs,function ‘storage_deposit_lazy’

159, symbiosis-portal/src/unsynthesize.rs, function 'meta_unsynthesize'

When writing unit tests, there is a problem with insufficient amount of gas to call the second callback in these lines. We assume that the problem occurs only in the mocked test version of the network. This error occurs due to the difference in the behavior of blockchain gas calculations with regard to the testnet version.

From the client:
The Symbiosus team verified the chosen approach: on tge Symbiosi testing stage, they always attach the maximum gas amount (300_000_000_000_000) - it is required due to the size of the promise call chain.

INFO-6 | RESOLVED

Redundant clone

Line. 124, 223, 238 contracts/symbiosis-metarouter/src/meta_route.rs

It is not always possible for the compiler to eliminate useless allocations and deallocations generated by redundant clone()s.
The issue is marked as informational as it refers to code best practices and performance.

Recommendation:
Delete clones

INFO-7 | RESOLVED

Unnecessary late initialization

Line. 2219,239 contracts/symbiosis-portal/src/unsynthesize.rs

Checks for late initializations that can be replaced by a let statement with an initializer.
The i is marked as informational as it refers to code best practices and performance.

Recommendation:
Declare ‘transfer_promise_1’ and ‘transfer_promise_2’ like this
let transfer_promise_1 = if storage_balance == 0 {

INFO-8 | RESOLVED

Redundant clone

Line. 50,52 contracts/symbiosis-portal/src/lib.rs

It is not always possible for the compiler to eliminate useless allocations and deallocations generated by redundant clone()s.
The issue is marked as informational as it refers to code best practices and performance.

Recommendation:
Delete clones

INFO-9 | RESOLVED

Error during instantiation.

Lines. 1,2 contracts/symbiosis-portal/src/synthesize.rs
Lines. 1,7,8 contracts/symbiosis-portal/src/unsynthesize.rs
Lines. 4 contracts/symbiosis-bridge/src/receive_request.rs
Lines. 1,2 packages/symbiosis-core/src/evm_calldata/unsynthesize.rs
Lines. 1,2 packages/symbiosis-core/src/evm_calldata/revert_burn.rs
Lines. 1,2 packages/symbiosis-core/src/evm_calldata/mint_synthetic_token.rs
Lines. 1,2 packages/symbiosis-core/src/evm_calldata/meta_unsynthesize.rs
Lines. 1,2 packages/symbiosis-core/src/evm_calldata/meta_mint_synthetic_token.rs

The error occurred during the instantiation (refer to Info-2 and Info-3). It was reproduced every time on the testnet after using any functions from the symbiosis-portal and symbiosis-bridge contracts. The issue reproduces only during the build of those contracts that include ‘ethabi’ crate(use ethabi::*;). Since symbiosis-core includes an ‘ethabi’ crate, all the contracts that include symbiosis-core have this issue too.
The error is reproducible on ordinary intel and amd architectures, though it looks like it is not reproducible on M1 chips. In spite of the problem being connected to the Near environment and its cross-platform compiler, it should be mentioned in the report. Due to the issue being present, it may have unpredictable behavior which may influence the mainnet deployment. The issue will not be treated as unresolved, though it should be present as a notice for the Symbiosis team.

Recommendation:
The best option is to contact the NEAR team and check any builds out of MacOS machines with M1 architecture (since the team reported that they have this environment), since it looks like it is the single working configuration. The issue is mostly connected with the “ethabi” package. Zokyo Security recommends paying attention for the proper sanitizing of the packages tree.

Post-audit:
The team of auditors was able to avoid the package problem within the Symbiosis environment. The environment was tested on several machines until the issue was resolved after the package engine was able to build the correct dependancy tree with the NEAR packages. Thus, it is highly recommended for the Symbiois team to pay attention for the proper packages dependencies sanitizing since a similar issue migh occur in the future.

	contracts/bridge	contracts/portal
Correct Accounts Usage Flow	Pass	Pass
Access Management Hierarchy	Pass	Pass
Types Conformity, Over/Under Flows	Pass	Pass
Unexpected Tokens	Pass	Pass
Cross-contract calls	Pass	Pass
Public Visibility Methods Access	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Incorrect Parameters Attack	Pass	Pass
Unchecked Return Values	Pass	Pass
Race Conditions/Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Unsafe Rust Code	Pass	Pass
Floating Points and Precision	Pass	Pass
Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying token)	Pass	Pass

	contracts/metarouter
Correct Accounts Usage Flow	Pass
Access Management Hierarchy	Pass
Types Conformity, Over/Under Flows	Pass
Unexpected Tokens	Pass
Cross-contract calls	Pass
Public Visibility Methods Access	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Incorrect Parameters Attack	Pass
Unchecked Return Values	Pass
Race Conditions/Front Running	Pass
General Denial Of Service (DOS)	Pass
Unsafe Rust Code	Pass
Floating Points and Precision	Pass
Authentication	Pass
Signatures Replay	Pass
Pool Asset Security (backdoors in the underlying token)	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Security

As a part of our work assisting Symbiosis in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Near testing framework.

The tests were based on the functionality of the code, as well as a review of the Symbiosis contract requirements for details about issuance amounts and how the system handles these.

contracts/symbiosis-bridge/src/mpc.rs
test_mpc_set_wrong
test_mpc_set_success
test_bridge_view_getters
contracts/symbiosis-bridge/src/transmitter.rs
test_transmit_request_unit
test_oracle_by_range_request_unit
contracts/symbiosis-portal/src/ft_receiver.rs
test_transfer_msg_serialize
test_migrate
test_set_meta_router
contracts/symbiosis-portal/src/pause.rs
test_pause_unpause
contracts/symbiosis-portal/src/synthesize.rs
test_revert_synthesize
contracts/symbiosis-portal/src/token.rs
test_set_unset_whitelist_token
test_set_unset_token_threshold
contracts/symbiosis-portal/src/unsynthesize.rs
test_synthesize_unsynthesize
test_get_unsynthesize_state
test_revert_burn_request
test_meta_unsynthesize (failing via incorrect Gas usage, only test env)
test_meta_unsynthesize_internal
contracts/symbiosis-metarouter/src/external_call.rs
test_external_call
contracts/symbiosis-metarouter/src/lib.rs
setup_contract

contracts/symbiosis-metarouter/src/meta_route.rs
test_native_meta_route
test_meta_route
test_balance_callback
test_balance_callback_second_msq
contracts/symbiosis-metarouter/src/util.rs
test_parse_token_receiver_synth_caller_skip
test_parse_token_receiver_msq_empty

The team of auditors performed several rounds of testing, including the preparation of unit tests for the core user flows, manual testing over the locally deployed contracts, and checking the whole set of Sandbox tests prepared by the Symbiosis team.

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by the Symbiosis team

As a part of our work assisting the Symbiosis team in verifying the correctness of their contract code, we have checked the full set of Sandbox and unit tests prepared by the Symbiosis team.

It needs to be mentioned that the original code has a significant original coverage with testing scenarios provided by the Symbiosis team. All of them were also carefully checked by the team of auditors.

contracts/ft/src/lib.rs
test_new
test_default
test_transfer
contracts/symbiosis-metarouter/src/ft_receiver.rs
test_on_transfer_msg
contracts/symbiosis-metarouter/src/utils.rs
test_replace_amount_with_space
test_replace_amount
test_parse_token_receiver
contracts/symbiosis-portal/src/ft_receiver.rs
test_transfer_msg_serialize
symbiosis_tests/tests/test_bridge_mpc.rs
test_bridge_set_mpc
test_bridge_set_mpc_signed
test_bridge_set_mpc_signed_error
symbiosis_tests/tests/test_bridge_oracle_request.rs
symbiosis_tests/tests/test_bridge_oracle_request.rs
symbiosis_tests/tests/test_bridge_receive_request.rs
test_bridge_receive_request_v2
test_bridge_receive_request_v2_error
test_bridge_receive_request_v2_signed
test_bridge_receive_request_v2_signed_error
symbiosis_tests/tests/test_bridge_transmit_request.rs
test_bridge_transmit_request_v2
symbiosis_tests/tests/test_bridge_transmitter.rs
test_bridge_set_transmitter
test_bridge_set_transmitter
test_portal_synthesize

We are grateful for the opportunity to work with the Symbiosis team.

The statements made in this document should not be interpreted as an investment or legal advice, nor should its authors be held accountable for the decisions made based on them.

Zokyo Security recommends the Symbiosis team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.