BLOCKCHAIN PROTOCOL AUDIT

zokyo

July 30th 2024 | v. 1.0

Security Audit Score

PASS

Zokyo Security has concluded that these smart contracts passed a security audit.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the Shido repository:
Repo: https://github.com/ShidoGlobal/ShidoChain

Last commit - `00222ee19e141b64ce44cad35af9fe9c8d0582f6`

During the audit, Zokyo Security ensured 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, efficiently using resources without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the most recent vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of Shido smart contract/s. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01 Due diligence in assessing the overall code quality of the codebase.

02 Cross-comparison with other, similar smart contract/s by industry leaders.

03 Thorough manual review of the codebase line by line.

Executive Summary

Zokyo team conducted a comprehensive security audit of the Shido Proof-of-Stake EVM and WASM blockchain from April 8, 2024 to July 19, 2024. The audit aimed to identify potential vulnerabilities and weaknesses within the blockchain's codebase, EVM and WASM integrations, and overall security posture.

We are pleased to report that no critical vulnerabilities were discovered during the audit. However, we identified several medium and low-severity issues, as well as opportunities for enhancing Shido's security posture. This report presents our detailed findings and actionable recommendations to address these issues.

Scope and Methodology

The scope of the audit included:

Comprehensive code review: We thoroughly examined the Shido blockchain's codebase, focusing on the Go, JavaScript, and Solidity components.

In-depth EVM and WASM analysis: We meticulously assessed the security implications of integrating EVM and WASM technologies into the Shido blockchain, paying particular attention to the unique challenges and opportunities presented by this hybrid architecture.

Thorough vulnerability mapping: We conducted extensive vulnerability mapping, referencing industry best practices and known attack vectors specific to Proof-of-Stake, EVM, and WASM systems, to identify potential areas of weakness.

Our methodology involved:

Static code analysis: We used automated tools to scan the codebase for common vulnerabilities and coding errors.

Manual code review: Our experienced security engineers manually reviewed the codebase to identify potential security flaws that automated tools could not detect.

Threat modeling: We constructed threat models to identify potential attack scenarios specific to the Shido blockchain's unique architecture and assess their likelihood and impact.

Penetration testing: We attempted to exploit potential vulnerabilities to assess their severity and impact within the context of the Shido Proof-of-Stake EVM and WASM environment.

STRUCTURE AND ORGANIZATION OF THE DOCUMENT

For the ease of navigation, the following sections are arranged from the most to the least critical ones. Issues are tagged as “Resolved” or “Unresolved” or “Acknowledged” depending on whether they have been fixed or addressed. Acknowledged means that the issue was sent to the Shido team and the Shido team is aware of it, but they have chosen to not solve it. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Client or remains disregarded by the Client. 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

FINDINGS SUMMARY

#	Title	Risk	Status
1	Incorrect Access Control via Malicious CosmWasm Contract	Medium	Acknowledged
2	IClear Text Logging of Sensitive Credentials (client/export.go)	Medium	Acknowledged
3	DoS in CometBFT Block Sync (github.com/cometbft/cometbft/blocksync)	Medium	Acknowledged
4	DoS in Cosmos SDK Crisis Module (github.com/cosmos/cosmos-sdk/x/crisis)	Medium	Acknowledged
5	Uncontrolled Resource Consumption in go-jose (github.com/dvsekhvalnov/jose2go)	Medium	Acknowledged
6	Consensus Issues in go-ethereum (github.com/ethereum/go-ethereum)	Medium	Acknowledged
7	GoBoolExpressions	Low	Acknowledged
8	GoDeprecation	Low	Acknowledged
9	GoDfaConstantCondition	Low	Acknowledged
10	GoDfaErrorMayBeNotNil	Low	Acknowledged
11	GoDfaNilDereference	Low	Acknowledged
12	GoExportedFuncWithUnexportedType	Low	Acknowledged
13	GoImportUsedAsName	Low	Acknowledged
14	GoReservedWordUsedAsName (x/vesting/types/schedule.go: 261, 282, 304)	Low	Acknowledged

#	Title	Risk	Status
15	GoSwitchMissingCasesForIotaConsts (ethereum/eip712/eip712_legacy.go: 381, rpc/types/utils.go: 198)	Low	Acknowledged
16	GoUnhandledErrorResult	Low	Acknowledged
17	GoUnusedConst	Low	Acknowledged
18	GoUnusedExportedFunction	Low	Acknowledged
19	GoUnusedExportedType	Low	Acknowledged
20	GoUnusedFunction	Low	Acknowledged
21	GoUnusedGlobalVariable	Low	Acknowledged
22	GoVetCopyLock	Low	Acknowledged
23	GoVetLostCancel	Low	Acknowledged
24	JsonSchemaCompliance (client/docs/config.json)	Low	Acknowledged
25	PackageJsonMismatchedDependency (contracts/package.json: 10)	Low	Acknowledged
26	GoBuildTag (rpc/namespaces/ethereum/debug/trace.go: 18)	Low	Acknowledged
27	GoCommentStart	Low	Acknowledged
28	GoDirectComparisonOfErrors (rpc/websockets.go, server/json_rpc.go, wallets/usbwallet/ledger.go)	Low	Acknowledged
29	GoErrorStringFormat (wallets/usbwallet/ledger.go: 140)	Low	Acknowledged
30	GoMixedReceiverTypes	Low	Acknowledged
31	GoNameStartsWithPackageName	Low	Acknowledged
32	GoPreferNilSlice	Low	Acknowledged

#	Title	Risk	Status
33	GoRedundantConversion	Low	Acknowledged
34	GoUnsortedImport (app/ante/utils/interfaces.go)	Low	Acknowledged
35	GoVarAndConstTypeMayBeOmitted	Low	Acknowledged

MEDIUM-1 | ACKNOWLEDGED

Incorrect Access Control via Malicious CosmWasm Contract

Description: The system is vulnerable to incorrect access control through the deployment and use of a malicious CosmWasm contract via IBC interactions.
Impact: This could lead to unauthorized access to sensitive data, loss of funds, or unexpected minting of tokens.
Likelihood: Low to moderate, as it requires the ability to deploy and execute malicious contracts.

Recommendation:

Upgrade github.com/cosmos/ibc-go/v7/modules/core/keeper to version 7.4.0 or higher to address this vulnerability.

MEDIUM-2 | ACKNOWLEDGED

IClear Text Logging of Sensitive Credentials (client/export.go)

Impact: Unsanitized input from sensitive credentials is logged in clear text, potentially exposing private keys.
Likelihood: Moderate, as malicious actors could access this information with access to the logs.

Recommendation:

Sanitize or hash sensitive information before logging. Consider using a secure logging mechanism that encrypts or obscures sensitive data.
Code Snippet:

```
export.go:80 `keyS := strings.ToUpper(hexutil.Encode(privB)[2:])`
export.go:82 `fmt.Println(keyS)`
```

MEDIUM-3 | ACKNOWLEDGED

DoS in CometBFT Block Sync (github.com/cometbft/cometbft/blocksync)

Description: A malicious peer can cause a DoS by sending a block with a very high LastCommit round, leading to excessive memory usage and potential crashes.
Impact: This vulnerability can disrupt the normal operation of the Shido blockchain by preventing it from syncing with the network.
Likelihood: Low to moderate, as it requires a malicious peer with knowledge of the block sync mechanism.

Recommendation:
Upgrade github.com/cometbft/cometbft to the latest version to mitigate this issue.

MEDIUM-4 | ACKNOWLEDGED

DoS in Cosmos SDK Crisis Module (github.com/cosmos/cosmos-sdk/x/crisis)

Description: The chain does not halt when an invariant check fails on a Cosmos SDK network, and a transaction is sent to the x/crisis module.
Impact: This can lead to a denial-of-service (DoS) attack, in which an attacker repeatedly triggers invariant failures to disrupt the network.
Likelihood: Moderate, as it requires knowledge of the invariant checks and the ability to craft malicious transactions.

Recommendation:
There is no fixed version for github.com/cosmos/cosmos-sdk/x/crisis. Consider implementing a custom solution to handle invariant failures gracefully, such as logging the error, notifying administrators, and potentially pausing block production until the issue is resolved.

MEDIUM-5 | ACKNOWLEDGED

Uncontrolled Resource Consumption in go-jose (github.com/dvsekhvalnov/jose2go)

Description: The library is vulnerable to uncontrolled resource consumption due to improper input validation, which could lead to excessive memory or CPU usage.
Impact: An attacker could exploit this vulnerability to cause a denial-of-service (DoS) attack.
Likelihood: Low, as it requires the ability to send specially crafted requests to the application.

Recommendation:
Upgrade github.com/dvsekhvalnov/jose2go to the latest version to mitigate this issue.

MEDIUM-6 | ACKNOWLEDGED

Consensus Issues in go-ethereum (github.com/ethereum/go-ethereum)

Description: Several vulnerabilities related to consensus mechanisms in go-ethereum were identified, including:

- Integer overflow in the Ethash consensus engine (CVE-2021-42219, CWE-400O)
- Potential for state inconsistency in certain consensus scenarios (CVE-2020-26240, CWE-682)

Impact: These vulnerabilities could lead to forks in the blockchain or other consensus-related issues.
Likelihood: Low to moderate, as they often require specific conditions or complex attacks to exploit.

Recommendation:
Upgrade github.com/ethereum/go-ethereum to the latest version, which includes fixes for these consensus-related vulnerabilities.

GoBoolExpressions

The codebase contains multiple instances of boolean expressions that are always true or false. These do not pose a security risk but can impact code readability and maintainability.

Files:

- app/ante/cosmos/setup_test.go: 64, 93
- app/ante/evm/setup_test.go: 47, 76
- app/ante/utils/setup_test.go: 44, 73
- app/upgrades/v1/upgrades_test.go: 39, 40
- x/evm/handler_test.go: 82, 124
- x/evm/keeper/setup_test.go: 80, 81, 87, 88
- x/feemarket/keeper/setup_test.go: 54, 55
- x/vesting/keeper/utils_test.go: 55
- app/ante/setup_test.go: 41, 46
- app/app.go: 177
- app/app.go: 178
- crypto/hd/utils_test.go: 76
- rpc/backend/backend_suite_test.go: 85
- testutil/integration/network/setup.go: 86
- x/erc20/migrations/v3/types/params.go: 49
- x/erc20/migrations/v3/types/params.go: 50
- x/evm/statedb/statedb.go: 279
- x/evm/types/params.go: 69
- x/evm/types/params.go: 70
- x/evm/types/params.go: 73
- x/feemarket/migrations/v4/types/params.go: 79
- x/feemarket/types/params.go: 77

Recommendation:

Review and simplify these expressions where possible. For example, in app/ante/cosmos/setup_test.go, the expression checkTx is always false. This could be simplified by removing the conditional check or setting checkTx to true if needed.

LOW-2 | ACKNOWLEDGED

GoDeprecation

The codebase uses several deprecated functions and types. While this does not necessarily indicate a vulnerability, it is good practice to update to newer versions to benefit from potential improvements and bug fixes.
Files: Multiple files throughout the codebase.

Recommendation:
Update the codebase to use the latest versions of libraries and frameworks. Specifically, deprecated functions like EmitEvents and EmitEvent should be replaced with their recommended alternatives. For example, in app/ante/cosmos/fees.go, app/ante/evm/eth.go, and other files, replace ctx.EventManager().EmitEvents(events) with ctx.EventManager().EmitTypedEvent(&events).

LOW-3 | ACKNOWLEDGED

GoDfaConstantCondition

Some conditional checks are always true or false due to constant values. This can be simplified to improve code readability.
Files:

- app/app.go: 1239
- precompiles/vesting/events.go: 53
- rpc/websockets.go: 477
- x/wasm/ibctesting/chain.go: 562

Recommendation:
Remove or refactor these unnecessary conditional checks. For example, in app/app.go:1239, the condition if storeUpgrades != nil is always false and can be removed.

LOW-4 | ACKNOWLEDGED

GoDfaErrorMayBeNotNil

There are instances where errors are not explicitly checked after function calls, potentially leading to unexpected behavior.
Files:

- crypto/hd/utils_test.go: 147
- ethereum/eip712/eip712.go: 14
- testutil/tx/eip712.go: 61
- x/wasm/keeper/querier_test.go: 854

Recommendation:
Add error checks after these function calls and handle the errors appropriately (e.g., log the error, return an error, or take corrective action). For example, in crypto/hd/utils_test.go:147, add a check for err != nil after privateKey.ToECDSA()

LOW-5 | ACKNOWLEDGED

GoDfaNilDereference

Some method calls could dereference nil pointers, leading to runtime errors.
Files:

- rpc/backend/chain_info.go: 208
- rpc/backend/node_info.go: 95
- x/wasm/keeper/keeper_test.go: 706, 987
- x/wasm/module_test.go: 581

Recommendation:
Add nil checks for the variables (e.g., err) before calling methods on them (e.g., err.Error()). For example, in rpc/backend/chain_info.go:208, add a check like if err != nil { ... } before logging err.Error().

LOW-6 | ACKNOWLEDGED

GoExportedFuncWithUnexportedType

Some exported functions return unexported types, which can make it difficult for external packages to use them.
Files:

- x/wasm/ibc_integration_test.go: 232
- x/wasm/relay_pingpong_test.go: 332

Recommendation:
Either export the relevant types (captureAckTestContractEngine and hit) or refactor the code to return values of exported types. This would improve the usability of these functions for external packages.

LOW-7 | ACKNOWLEDGED

GoImportUsedAsName

Variables are named the same as imported packages, which can lead to confusion.
Files:

- app/ante/evm/utils_test.go: 683
- testutil/abci.go: 66, 94, 121, 144, 170
- app/app.go: 382
- rpc/backend/chain_info.go: 44, 254
- rpc/websockets.go: 234, 257, 295, 461
- x/evm/keeper/state_transition_benchmark_test.go: 246, 282, 317
- x/evm/keeper/state_transition_test.go: 257, 649
- x/evm/types/params_test.go: 59
- rpc/backend/client_test.go: 142, 169, 220
- rpc/websockets.go: 318
- rpc/backend/utils.go: 251
- rpc/namespaces/ethereum/eth/filters/api.go: 412
- server/start.go: 238, 327

- x/evm/keeper/keeper_test.go: 54
- x/evm/keeper/statedb.go: 56
- x/wasm/keeper/querier_test.go: 767

Recommendation:

Rename the variables to avoid conflicts with the imported package names. For example, in app/ante/evm/utils_test.go:683, rename the tx variable to something like ethTx.

LOW-8 | ACKNOWLEDGED

GoReservedWordUsedAsName (x/vesting/types/schedule.go: 261, 282, 304)

Recommendation:

Recommendation: Rename the min variable to avoid using the reserved word min. You could use minAmount or minimumAmount instead.

LOW-9 | ACKNOWLEDGED

GoSwitchMissingCasesForIotaConsts (ethereum/eip712/eip712_legacy.go: 381, rpc/types/utils.go: 198)

Recommendation:

Add missing case statements for all possible values of the iota constants in the switch statements, or add a default case to handle any unexpected values. This will make the code more robust and prevent potential errors if new values are added to the iota constants in the future.

GoUnhandledErrorResult

In some instances, errors are not explicitly checked after function calls, potentially leading to unexpected behavior.
Files:

- app/tps_counter_test.go: 26, 73
- crypto/hd/algorithm_test.go: 75, 77
- indexer/kv_indexer.go: 52, 178, 191
- rpc/websockets.go: 324
- server/start.go: 507
- x/erc20/keeper/ibc_callbacks_integration_test.go: 285, 546
- x/erc20/keeper/integration_test.go: 43, 55, 67
- x/erc20/keeper/mint_test.go: 29
- x/erc20/keeper/msg_server_test.go: 79, 461, 1059, 1148, 1149
- x/erc20/keeper/params_test.go: 11
- x/erc20/keeper/proposals_test.go: 139, 281
- x/erc20/keeper/token_pairs.go: 27
- x/evm/keeper/statedb.go: 59
- x/wasm/ioutils/ioutil.go: 23
- x/wasm/keeper/genesis_test.go: 142, 143, 638
- x/wasm/keeper/keeper.go: 597, 633, 653, 671, 792, 810, 853, 927
- x/wasm/migrations/v3/store.go: 58
- x/wasm/types/iavl_range_test.go: 66

Recommendation:
Add error checks after these function calls and handle the errors appropriately (e.g., log the error, return an error, or take corrective action). For example, in app/tps_counter_test.go:26, the error returned by tpc.start(ctx) should be checked and handled.

LOW-11 | ACKNOWLEDGED

GoUnusedConst

Several unused constants are declared in the codebase.
Files: app/params/weights.go, app/upgrades/v1/constants.go, precompiles/distribution/errors.go, rpc/apis.go, server/flags/flags.go, x/erc20/types/events.go, x/evm/types/events.go, x/feemarket/types/keys.go, x/wasm/alias.go, x/wasm/types/test_fixtures.go

Recommendation:
Remove unused constants to improve code clarity. For example, in app/params/weights.go, several constants like DefaultWeightMsgSend and DefaultWeightMsgMultiSend are declared but not used.

LOW-12 | ACKNOWLEDGED

GoUnusedExportedFunction

Some exported functions are not used within the codebase.
Files: Multiple files throughout the codebase.

Recommendation:
Consider removing these functions if they are not part of the public API. For example, the function NewTestnetCmd in client/testnet.go is not used and could be removed.

LOW-13 | ACKNOWLEDGED

GoUnusedExportedType

Some exported types are not used within the codebase
Files:

- x/wasm/keeper/genesis_test.go: 685
- x/wasm/keeper/staking_test.go: 76

Recommendation:
Consider removing these types if they are not part of the public API.

LOW-16 | ACKNOWLEDGED

GoUnusedFunction

Some functions are declared but not used within the codebase
Files: x/wasm/relay_test.go: 811

Recommendation:
Remove unused functions to improve code clarity.

LOW-17 | ACKNOWLEDGED

GoUnusedGlobalVariable

Several unused global variables are declared in the codebase
Files: Multiple files throughout the codebase.

Recommendation:
Remove unused global variables to improve code clarity. For example, the variable isGenesistxn in app/ante/evm/fee_checker.go is not used and could be removed.

LOW-18 | ACKNOWLEDGED

GoVetCopyLock

Locks are passed by value in some cases, which can lead to unexpected behavior.
Files:

- ibc/module_test.go: 32, 52, 72, 89, 104, 119, 134, 149, 165
- x/erc20/keeper/proposals_test.go: 109, 254, 343

Recommendation:
Pass locks by reference to ensure proper synchronization. For example, in ibc/module_test.go, change the receiver of the OnChanOpenInit function from (m MockIBCModule) to (m *MockIBCModule).

LOW-19 | ACKNOWLEDGED

GoVetLostCancel

The cancel function should be called, not discarded, to avoid a context leak.
Files: rpc/backend/evm_query_client_test.go: 143, 149

Recommendation:
Call the cancelFn function after using the context to prevent resource leaks.

LOW-20 | ACKNOWLEDGED

JsonSchemaCompliance (client/docs/config.json)

Description:

- Missing required property 'paths' (line 2)
- Property 'apis' is not allowed (line 8)

Recommendation:
Add the missing paths property and remove the disallowed apis property to comply with the JSON schema.

LOW-21 | ACKNOWLEDGED

PackageJsonMismatchedDependency (contracts/package.json: 10)

Recommendation:
Install the missing @openzeppelin/contracts package using npm install @openzeppelin/contracts.

LOW-22 | ACKNOWLEDGED

GoBuildTag (rpc/namespaces/ethereum/debug/trace.go: 18)

Description: Build tag // +build go1.5 can be removed.

Recommendation:
Remove the build tag as it is no longer necessary.

LOW-23 | ACKNOWLEDGED

GoCommentStart

Several comments do not follow the recommended format.
Files: Multiple files throughout the codebase.

Recommendation:
Update comments to adhere to the recommended format for better readability and consistency. For example, in app/app.go, change the comment on line 167 to // ProposalsEnabled....

LOW-24 | ACKNOWLEDGED

GoDirectComparisonOfErrors (rpc/websockets.go, server/json_rpc.go, wallets/usbwallet/ledger.go)

Recommendation:
Use errors.Is() for error comparison to handle wrapped errors correctly. For example, in rpc/websockets.go, change err == http.ErrServerClosed to errors.Is(err, http.ErrServerClosed).

LOW-25 | ACKNOWLEDGED

GoErrorStringFormat (wallets/usbwallet/ledger.go: 140)

Recommendation:
Rewrite the error message not to be capitalized and not end with punctuation. For example, change it to "Ledger version 1.5.0 or higher is required for EIP-712 signing (found version v%d.%d.%d)".

LOW-26 | ACKNOWLEDGED

GoMixedReceiverTypes

Several structs have methods defined on both value and pointer receivers.
Files: Multiple files throughout the codebase.

Recommendation:
Choose a consistent receiver type (either value or pointer) for each struct to improve code readability and maintainability.

LOW-27 | ACKNOWLEDGED

GoNameStartsWithPackageName

Some names start with the package name, which is not recommended.
Files:

- app/ante/evm/interfaces.go: 19
- precompiles/vesting/types.go: 27
- rpc/backend/backend.go: 27
- x/wasm/alias.go: 24, 96, 211

Recommendation:
Rename the types or interfaces to follow Go naming conventions and avoid conflicts.

LOW-28 | ACKNOWLEDGED

GoPreferNilSlice

Empty slices are declared using literals ([]byte{} or []string{}) instead of nil.
Files: Multiple files throughout the codebase.

Recommendation:
Use nil to declare empty slices for better performance and consistency.

LOW-29 | ACKNOWLEDGED

GoRedundantConversion

Redundant type conversions are present in the code.
Files: Multiple files throughout the codebase.

Recommendation:
Remove unnecessary type conversions to improve code readability and performance.

LOW-30 | ACKNOWLEDGED

GoUnsortedImport (app/ante/utils/interfaces.go)

Recommendation:
Sort the imports in the file alphabetically to improve readability and maintainability.

GoVarAndConstTypeMayBeOmitted

The type of some variables and constants can be omitted, as Go can infer them.
Files:

- app/ante/sigverify.go: 18
- app/test_helpers.go: 42
- ibc/testing/app.go: 30
- x/evm/statedb/mock_test.go: 16
- x/evm/statedb/statedb_test.go: 17, 18, 19, 20, 21
- x/wasm/keeper/recurse_test.go: 142

Recommendation:
Remove the explicit type declarations for these variables and constants to simplify the code.

CONTRACTS		
Transaction Ordering	Check for any logic that could be exploited by manipulating the order of transactions	PASSED
Timestamp Reliance	Identify areas where code depends on timestamps, and consider potential vulnerabilities related to time manipulation	PASSED
Integer Calculations	Scrutinize integer operations to prevent overflow or underflow errors	PASSED
Rounding Errors	Look for places where rounding might lead to unintended consequences	PASSED
Denial of Service (DoS)	Examine if the code has weaknesses that could be exploited to disrupt service or cause unexpected resource consumption	PASSED
Access Controls	Verify that access to sensitive functions and data is properly restricted and that permissions are implemented correctly	PASSED
Centralization	Assess whether any single entity or component has excessive control, creating a potential point of failure	PASSED
Business Logic Integrity	Ensure that the code accurately reflects the intended business rules and specifications	PASSED
Code Duplication	Minimize redundant code to reduce the attack surface and improve maintainability	PASSED
Gas Usage	Analyze gas consumption patterns to predict potential DoS scenarios or unexpected cost implications	PASSED
Token Minting	If applicable, verify that token minting mechanisms are secure and cannot be abused	PASSED
Call Return Values	Ensure that return values from external calls are always checked for errors	PASSED
Capability Flow	Carefully track how capabilities are transferred and used to prevent unauthorized actions	PASSED
Witness Type Usage	If applicable, verify that witness types are used correctly to enforce constraints	PASSED

We are grateful for the opportunity to work with the Shido 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 Shido team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo