



Syndicate Core Protocol

SMART CONTRACT AUDIT

25.07.2025

Made in Germany by Softstack.io



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

Table of contents

1. Disclaimer.....	4
2. About the Project and Company	5
2.1 Project Overview.....	6
3. Vulnerability & Risk Level	7
4. Auditing Strategy and Techniques Applied.....	8
4.1 Methodology	8
5. Metrics	9
5.1 Tested Contract Files	9
5.2 Inheritance Graph.....	12
5.3 Source Lines & Risk.....	13
5.4 Capabilities	14
5.5 Dependencies / External Imports.....	15
5.6 Source Unites in Scope	17
6. Scope of Work.....	21
6.1 Findings Overview	22
6.2 Manual and Automated Vulnerability Test.....	23
6.2.1 Unbounded Array Growth Leading to Denial-of-Service in TeeModule	23
6.2.2 Unbounded Gas Consumption DoS in Multiple Locations.....	26
6.2.3 Chain ID Collision Vulnerability in getNextChainId()	31
6.2.4 Missing Implementation Validation in upgradImplementation().....	34
6.2.5 Front-Running Attack on AssertionPoster Configuration	37



6.3 Verify Claims	41
7. Executive Summary.....	42
8. About the Auditor	43



1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of SYNDICATE INC. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

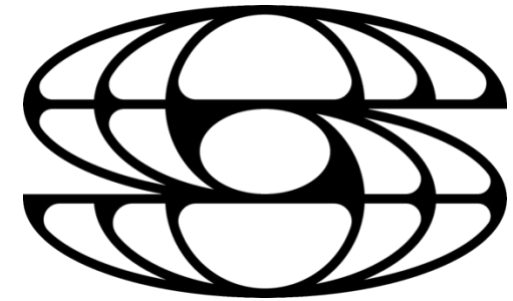
Major Versions / Date	Description
0.1 (10.07.2025)	Layout
0.4 (13.07.2025)	Automated Security Testing Manual Security Testing
0.5 (16.07.2025)	Verify Claims
0.9 (18.07.2025)	Summary and Recommendation
1.0 (20.07.2025)	Submission of findings
1.1 (21.07.2025)	Re-check
1.2 (25.07.2025)	Final document



2. About the Project and Company

Company address:

SYNDICATE INC.
1049 El Monte Avenue Ste C #560
Mountain View, CA 94040
USA



Website: <https://syndicate.io>

Twitter (X): <https://x.com/syndicateio>

Telegram: <https://t.me/syndicateiocommunity>

Farcaster: <https://farcaster.xyz/syndicate>



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

2.1 Project Overview

Syndicate is a modular, programmable infrastructure platform enabling developers and enterprises to launch scalable, production-ready on-chain applications across any EVM-compatible network. With a transaction-first architecture, Syndicate abstracts complex wallet management, signing flows, and multi-chain bridging into a single developer-friendly API stack.

At its foundation, Syndicate provides compliant, secure, and performant services for broadcasting transactions, managing attestations, and integrating on-chain logic with off-chain workflows. Its transaction cloud is built to support thousands of operations per second and offers compatibility with Ethereum, Layer 2s like Arbitrum and Optimism, and emerging Superchain networks.

Syndicate's infrastructure includes deterministic CREATE3-based deployment, ERC-7802-compatible crosschain token tooling, rate-limited bridge proxies, and automated emission schedulers, ideal for token launches, DePIN deployments, and DAO tooling. With native support for ERC20Votes, ERC20Permit, and gasless transactions, Syndicate empowers developers to build governance-ready applications with flexible emission and distribution models.

Use cases span from crosschain token issuance and NFT minting to embedded DeFi rails and programmable user wallets. Syndicate also offers REST APIs, SDKs, and webhook integrations to streamline deployment pipelines and orchestrate high-throughput, multi-chain interactions. Future development includes zk-powered privacy layers, intent-based execution, and integration with emerging L3 ecosystems.



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk



4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert auditors and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i. Review of the specifications, sources, and instructions provided to softstack to make sure we understand the size, scope, and functionality of the smart contract.
 - ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to softstack describe.
2. Testing and automated analysis that includes the following:
 - i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

5. Metrics

The metrics section should give the reader an overview on the size, quality, flows and capabilities of the codebase, without the knowledge to understand the actual code.

5.1 Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Source: <https://github.com/SyndicateProtocol/syndicate-appchains/tree/43836b5eef8addc3d4ff1b22dac27a65a594b5b4>
Commit: 43836b5eef8addc3d4ff1b22dac27a65a594b5b4

File	Fingerprint (MD5)
./factory/SyndicateFactory.sol	d4998959ab27ed44229e8f2a937c7bc1
./factory/PermissionModuleFactories.sol	edfe7972b02a5078052261ede9465552
./factory/SyndicateFactoryWrapper.sol	900a6b8ab61b9a33495c566bb55a39d6
./SyndicateSequencingChain.sol	9ba988846adba24eecec0fbd4057a31f
./SequencingModuleChecker.sol	d5d5ef7cd222f8662b6f5e28278b4688
./sequencing-modules/AllowlistSequencingModule.sol	57b621e50acba8690eaf06a65f8bc0fb
./sequencing-modules/WalletPoolWrapperModule.sol	b5bf9b4dd3522ad2e9d5ee3721033d01
./atomic-sequencer/AtomicSequencer.sol	fd043d9f45353a9382df33c219c6c1b6
./atomic-sequencer/AtomicSequencerImplementation.sol	375b9d4ed431bc57b6d203a5c706f3bf



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

./requirement-modules/RequireAndModule.sol	2848b15f90457aef71b02967cd70b8fd
./requirement-modules/RequireOrModule.sol	a3453cb39f8166e866b48d09e22ff509
./requirement-modules/RequireCompositeModule.sol	67d8e9da3c9fbcfb29727a5fc452a228
./interfaces/IRequirementModule.sol	0096d4212cb700663df0da52cb92b0e3
./interfaces/IPermissionModule.sol	fc0ab6183316e464bc63e19f5f602c52
./LinkedList/AddressStructuredLinkedList.sol	44a9a00ab9f8d2b94fce48cd3686e64a
./config/ArbConfigManager.sol	c2a82123fcfda3ecfb72d8b081d4c2e9
./config/ArbChainConfig.sol	b829d8b0af7238be8ded2efeb445564a
./config/ArbConfigManagerFactory.sol	c80a13096e2d922949d6ca4bb31de4d7
./token/bridges/ArbitrumBridgeProxy.sol	7a571a7d8dce4404d2b85482df064310
./token/bridges/BaseBridgeProxy.sol	cce5bf9f550a419ab1c45886b37bb3be
./token/bridges/OptimismBridgeProxy.sol	01dd62944d0e4e1e16c9f4b3f53b9b98
./token/interfaces/IBridgeProxy.sol	5e69c868755ded4abeb5416e21daed5c
./token/crosschain/interfaces/IERC7802.sol	18cfbe1b9bd060a6ed76fb51af94a2bc
./token/crosschain/interfaces/IBridgeRateLimiter.sol	4fd390cdcfc552522933870f42a26216
./token/crosschain/libraries/CREATE3.sol	182ba0ca2fd9612efce6d6c4bcf6f468
./token/crosschain/libraries/Bytes32AddressLib.sol	7526857abd0188726d1e495f8597167c
./withdrawal/AssertionPoster.sol	98f212266d41f08c645f5ae3fd30a0ab
./withdrawal/AttestationDocVerifier.sol	0e9bd48d7def7f09ebefc391e70966da



./withdrawal/TeeKeyManager.sol	84057958d888a81bc0d521483695852b
./withdrawal/TeeModule.sol	88f5b29789c71a671027fd0c2ac98a9d
./withdrawal/IAssertionPoster.sol	3a2edb0b7dd2ec19f45c6d63f5c1d459
./withdrawal/IAttestationDocVerifier.sol	8d034dff9e2589c8e72a03e2ed647a2f
./withdrawal/ITeeKeyManager.sol	d4991978ec2e26f229d537ccfac120f2

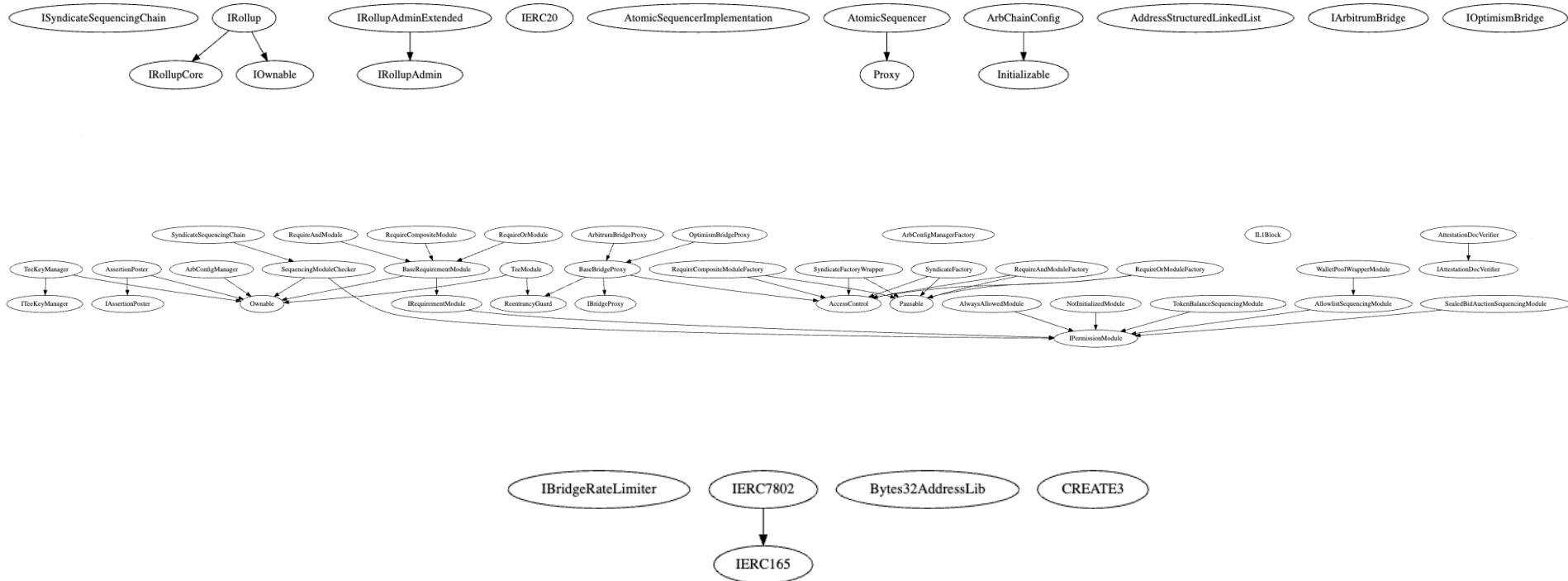


hello@softstack.io
www.softstack.io

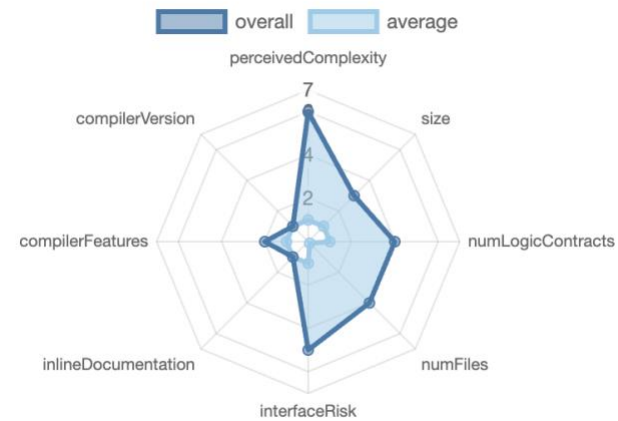
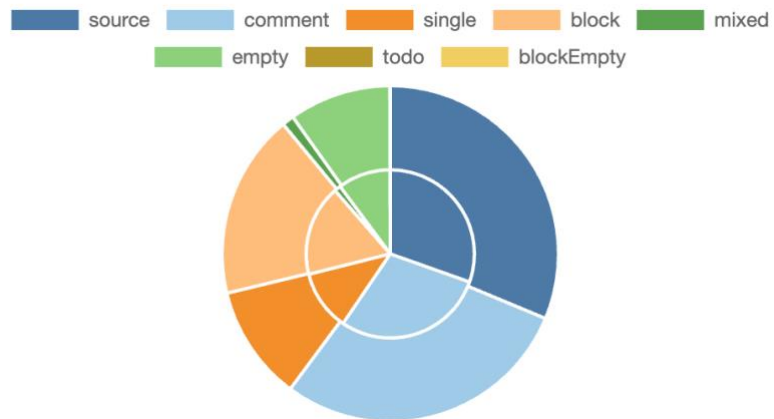
softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

5.2 Inheritance Graph



5.3 Source Lines & Risk













hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg



CRN: HRB 12635 FL
VAT: DE317625984

5.4 Capabilities

Solidity Versions observed		 Experimental Features		 Can Receive Funds		 Uses Assembly		 Has Destroyable Contracts			
0.8.28 ^0.8.20 >=0.8.0				yes		yes (2 asm blocks)					
 Transfers ETH		 Low-Level Calls		 DelegateCall		 Uses Hash Functions		 ECREcover		 New/Create/Create2	
						yes				yes → NewContract:NotInitializedModule → AssemblyCall:Name:create2 → NewContract:AtomicSequencerImplementation → NewContract:ArbChainConfig → NewContract:UpgradeableBeacon	

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

 Public	 Payable			
144	4			
External	Internal	Private	Pure	View
129	156	10	11	61



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

5.5 Dependencies / External Imports

Dependency / Import Path	Source
@openzeppelin/contracts/access/AccessControl.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/AccessControl.sol
@openzeppelin/contracts/access/IAccessControl.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/IAccessControl.sol
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol
@openzeppelin/contracts/proxy/beacon/BeaconProxy.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/proxy/beacon/BeaconProxy.sol
@openzeppelin/contracts/proxy/beacon/UpgradeableBeacon.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/proxy/beacon/UpgradeableBeacon.sol
@openzeppelin/contracts/token/ERC20/ERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/extensions/ERC20Permit.sol
@openzeppelin/contracts/token/ERC20/extensions/ERC20Votes.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/extensions/ERC20Votes.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/utils/SafeERC20.sol
@openzeppelin/contracts/utils/Create2.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Create2.sol
@openzeppelin/contracts/utils/Pausable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Pausable.sol



@openzeppelin/contracts/utils/ReentrancyGuard.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/ReentrancyGuard.sol
@openzeppelin/contracts/utils/cryptography/ECDSA.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/cryptography/ECDSA.sol
@openzeppelin/contracts/utils/introspection/IERC165.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/introspection/IERC165.sol
@openzeppelin/contracts/utils/structs/EnumerableSet.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/structs/EnumerableSet.sol
@arbitrum/nitro-contracts/src/bridge/IBridge.sol	https://github.com/OffchainLabs/nitro-contracts/blob/main/src/bridge/IBridge.sol
@arbitrum/nitro-contracts/src/bridge/IOwnable.sol	https://github.com/OffchainLabs/nitro-contracts/blob/main/src/bridge/IOwnable.sol
@arbitrum/nitro-contracts/src/libraries/IGasRefunder.sol	https://github.com/OffchainLabs/nitro-contracts/blob/main/src/libraries/IGasRefunder.sol
@arbitrum/nitro-contracts/src/rollup/IRollupAdmin.sol	https://github.com/OffchainLabs/nitro-contracts/blob/main/src/rollup/IRollupAdmin.sol
@arbitrum/nitro-contracts/src/state/GlobalState.sol	https://github.com/OffchainLabs/nitro-contracts/blob/main/src/state/GlobalState.sol
@offchainlabs/upgrade-executor/src/IUpgradeExecutor.sol	https://github.com/OffchainLabs/upgrade-executor/blob/main/src/IUpgradeExecutor.sol
@sp1-contracts/ISP1Verifier.sol	https://github.com/succinctlabs/sp1-contracts/blob/main/contracts/src/ISP1Verifier.sol



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

5.6 Source Unites in Scope

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines
synd-contracts/src/SequencingModuleChecker.sol	1		77	77	34	30
synd-contracts/src/SyndicateSequencingChain.sol	1		75	72	29	35
synd-contracts/src/config/ArbConfigManagerFactory.sol	1		85	85	41	44
synd-contracts/src/interfaces/IRequirementModule.sol		1	30	17	4	19
synd-contracts/src/interfaces/IPermissionModule.sol		1	13	12	3	8
synd-contracts/src/withdrawal/TeeModule.sol	1	1	264	257	164	61
synd-contracts/src/withdrawal/AttestationDocVerifier.sol	1		84	80	54	21
synd-contracts/src/withdrawal/TeeKeyManager.sol	1		71	71	35	26
synd-contracts/src/withdrawal/IAttestationDocVerifier.sol		1	15	11	3	7
synd-contracts/src/interfaces/ISyndicateSequencingChain.sol		1	19	10	3	10
synd-contracts/src/sequencing-modules/AlwaysAllowedModule.sol	1		18	18	7	9
synd-contracts/src/withdrawal/AssertionPoster.sol	1	2	342	288	157	116
synd-contracts/src/withdrawal/IAssertionPoster.sol		1	11	10	3	6



File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines
synd-contracts/src/sequencing-modules/NotInitializedModule.sol	1		13	13	7	4
synd-contracts/src/withdrawal/ITeeKeyManager.sol		1	11	10	3	6
synd-contracts/src/sequencing-modules/TokenBalanceSequencingModule.sol	1	1	44	34	17	19
synd-contracts/src/atomic-sequencer/AtomicSequencerImplementation.sol	1		52	45	25	14
synd-contracts/src/atomic-sequencer/AtomicSequencer.sol	1		23	23	12	7
synd-contracts/src/sequencing-modules/WalletPoolWrapperModule.sol	1		101	89	35	42
synd-contracts/src/sequencing-modules/AllowlistSequencingModule.sol	1		87	87	41	31
synd-contracts/src/sequencing-modules/SealedBidAuctionSequencingModule.sol	1		184	184	96	64
synd-contracts/src/factory/PermissionModuleFactories.sol	3		166	162	88	40
synd-contracts/src/factory/SyndicateFactoryWrapper.sol	1		181	161	83	54
synd-contracts/src/config/ArbConfigManager.sol	1		148	135	54	60
synd-contracts/src/factory/SyndicateFactory.sol	1		155	150	78	46
synd-contracts/src/config/ArbChainConfig.sol	1		153	140	65	59



File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines
synd-contracts/src/LinkedList/AddressStructuredLinkedList.sol	1		261	261	106	128
synd-contracts/src/requirement-modules/BaseRequirementModule.sol	1		111	105	48	39
synd-contracts/src/requirement-modules/RequireAndModule.sol	1		57	52	19	25
synd-contracts/src/requirement-modules/RequireCompositeModule.sol	1		198	189	80	78
synd-contracts/src/requirement-modules/RequireOrModule.sol	1		59	54	20	25
synd-contracts/src/token/interfaces/IBridgeProxy.sol		1	41	30	3	34
synd-contracts/src/token/bridges/ArbitrumBridgeProxy.sol	1	1	269	228	72	163
synd-contracts/src/token/bridges/OptimismBridgeProxy.sol	1	1	183	150	48	118
synd-contracts/src/token/bridges/BaseBridgeProxy.sol	1		338	321	103	179
synd-contracts/src/token/crosschain/interfaces/IBridgeRateLimiter.sol		1	102	72	19	57
synd-contracts/src/token/crosschain/interfaces/IERC7802.sol		1	40	34	6	25
synd-contracts/src/token/crosschain/libraries/Bytes32AddressLib.sol	1		14	14	9	3



File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines
synd-contracts/src/token/crosschain/libraries/CREATE3.sol	1		66	66	24	34
Totals	32	15	4161	3817	1698	1746

Legend:

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments



6. Scope of Work

The Syndicate Protocol team has delivered a modular smart contract suite enabling token issuance, emission scheduling, and cross-chain functionality. The audit will focus on validating the security assumptions, logic correctness of the core protocol.

The team has outlined the following critical security and functionality assumptions for the smart contracts:

1. **Sequencing Logic Validation**

The sequencing contract must enforce correct transaction ordering and permission checks, ensuring that sequencing rules are implemented as specified.

2. **Modular Architecture Verification**

Requirement modules (AND, OR, composite, etc.) must interact reliably with the main contract. The audit will validate module registration, validation logic, and external contract calls.

3. **Permission Management & Role Separation**

Role-based access controls must enforce clear privilege separation for sequencing actions. The audit will assess configuration safety and protection against role escalation.

4. **Cross-Chain Bridge Integration**

Arbitrum and Optimism bridge proxies must securely manage asset movement between chains. The audit will focus on message flow, verification steps, and replay protection across layers.

5. **Bridge Rate Limiting and Flow Control**

Rate limiting mechanisms in bridge contracts must prevent abuse and ensure predictable transaction settlement. Correct enforcement of thresholds and cooldowns will be tested.

6. **CREATE3 Deployment and Configuration**

CREATE3 must ensure deterministic and collision-free deployments. The audit will validate address predictability, salt uniqueness, and initialization safety.

The primary objective of this audit is to confirm that the core protocol is secure, extensible, and production-ready.

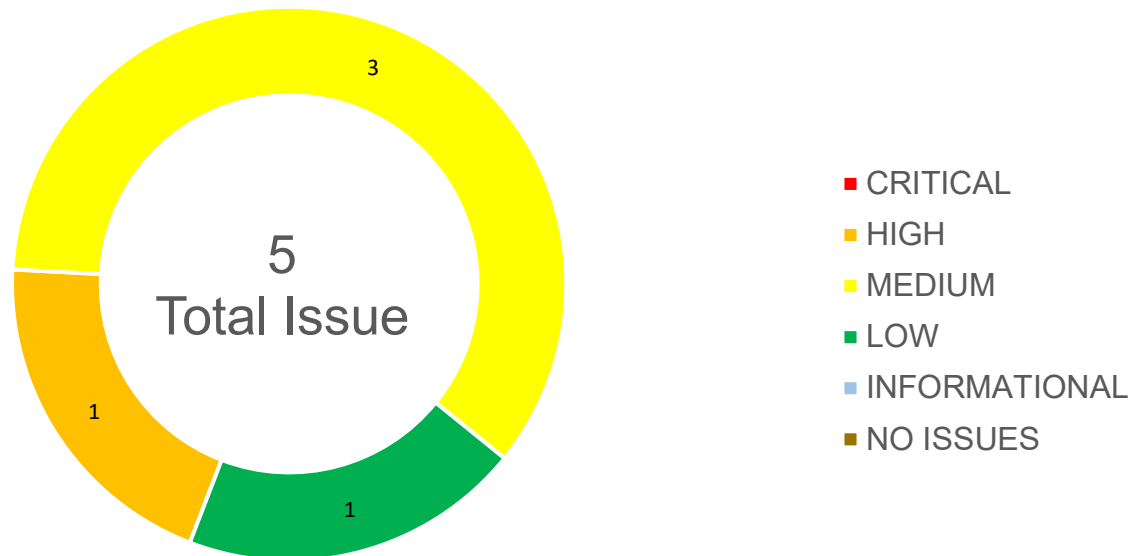


hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

6.1 Findings Overview



No	Title	Severity	Status
6.2.1	Unbounded Array Growth Leading to Denial-of-Service in TeeModule	HIGH	FIXED
6.2.2	Unbounded Gas Consumption DoS in Multiple Locations	MEDIUM	FIXED
6.2.3	Chain ID Collision Vulnerability in getNextChainId()	MEDIUM	FIXED
6.2.4	Missing Implementation Validation in upgradeImplementation()	MEDIUM	FIXED
6.2.5	Front-Running Attack on AssertionPoster Configuration	LOW	FIXED



6.2 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, softstack's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, softstack's experts found **one High issue** in the code of the smart contract.

6.2.1 Unbounded Array Growth Leading to Denial-of-Service in TeeModule

Severity: HIGH

Status: FIXED

File(s) affected: TeeModule.sol

Update: <https://github.com/SyndicateProtocol/syndicate-appchains/pull/630/commits/e736a08563748bd8db6c84e5b88f0783d11222c8>

Attack / Description	<p>The submitAssertion() function in TeeModule.sol allows new assertions to be added to the pendingAssertions array. However, the contract fails to enforce a size limit on this array. The function also contains a loop that iterates through all existing pendingAssertions to check for duplicates.</p> <p>An attacker with the ability to generate multiple valid TEE signatures can repeatedly call submitAssertion(), causing the pendingAssertions array to grow indefinitely. As the array size increases, the gas cost of the loop within the function also increases linearly. Eventually, the gas required to execute the loop will exceed the block gas limit.</p> <p>This creates a permanent Denial-of-Service (DoS) vulnerability. Once the array is sufficiently large, no new assertions can be submitted, and other functions that may rely on this array (like</p>
-----------------------------	--



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

resolveChallenge()) will also become unusable. This would effectively halt the entire TEE assertion system permanently.

Proof of Concept:

```
function test_Critical03_UnboundedArrayDoS() public {  
  
    vm.startPrank(attacker);  
  
    // Test DoS on bulk transaction processing  
  
    UnboundedArrayAttacker dosAttacker = new UnboundedArrayAttacker();  
  
    // Measure gas before attack  
  
    uint256 gasBefore = gasleft();  
  
    try dosAttacker.performDoSAttack(sequencingChain) {  
  
        // Attack completed - DoS via gas exhaustion  
  
    } catch {  
  
        // Even if it runs out of gas, it demonstrates the vulnerability  
  
    }  
  
    uint256 gasUsed = gasBefore - gasleft();  
  
    assertTrue(gasUsed > 1000000, "CRITICAL-03 EXPLOITED: Unbounded array DoS  
attack");  
  
    vm.stopPrank();  
}
```



	}
Code	<p>Line 206- 232 (TeeModule.sol):</p> <pre> function submitAssertion(PendingAssertion calldata assertion, bytes calldata signature, address rewardAddr) external nonReentrant { // ... validation ... for (uint256 i = 0; i < pendingAssertions.length; i++) { // Unbounded loop require(assertionHash != hash_object(pendingAssertions[i]), "assertion already exists"); } pendingAssertions.push(assertion); // No size limit on the array if (pendingAssertions.length == 2) { teeHackCount += 1; // Logic handles a specific case but does not prevent further growth // ... payment logic } } </pre>

Result/Recommendation	<p>To mitigate this critical DoS vulnerability, a strict limit must be enforced on the size of the pendingAssertions array. This prevents an attacker from bloating the array to a point where it becomes too costly to process.</p> <p>A require check should be added at the beginning of the submitAssertion function to ensure the number of pending assertions does not exceed a reasonable, predefined maximum.</p> <p>Suggested Code:</p> <pre>// Define a reasonable constant for the maximum number of pending assertions uint256 public constant MAX_PENDING_ASSERTIONS = 10; function submitAssertion(...) external nonReentrant { // Add a size check to prevent unbounded array growth require(pendingAssertions.length < MAX_PENDING_ASSERTIONS, "TeeModule: Too many pending assertions"); // ... rest of the function logic }</pre>
------------------------------	--

MEDIUM ISSUES

During the audit, softstack's experts found **three Medium issues** in the code of the smart contract.

6.2.2 Unbounded Gas Consumption DoS in Multiple Locations

Severity: MEDIUM



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

Status: FIXED

File(s) affected: SyndicateSequencingChain.sol

Update: <https://github.com/SyndicateProtocol/syndicate-appchains/pull/630/commits/0d720bd9f2d9ac8eb6a7ce615a2c5d63ea3ccca3>

Attack / Description

Several key functions throughout the contract system accept arrays as input or iterate through lists without enforcing any size limits. This includes bulk transaction processing, permission checking loops, and atomic sequencing operations.

This design pattern creates a significant gas-based Denial-of-Service (DoS) vulnerability. An attacker can call these functions with an extremely large array (e.g., thousands of elements). The gas required to loop through and process every element in the array can easily exceed the block gas limit, causing the transaction to fail. This effectively blocks the function from being used by any legitimate user, as any attempt to call it will run out of gas.

The impact is a widespread Denial-of-Service across critical protocol functions. This can halt bulk transaction processing, render permission checks unusable, and disrupt atomic cross-chain operations, severely impacting the availability and reliability of the entire system

Proof of Concept:

```
function test_Medium01_UnboundedGasDoSExploit() public {  
  
    // Demonstrate unbounded array processing vulnerability  
  
    uint256 maliciousArraySize = 2000; // Excessive size  
  
    uint256 gasPerOperation = 21000;    // Approximate gas per operation  
  
    uint256 totalGasRequired = maliciousArraySize * gasPerOperation;
```



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

```
// This would exceed block gas limit (30M gas)

assertTrue(totalGasRequired > 300000000);


// Simulate bulk processing DoS

bytes[] memory maliciousData = new bytes[](maliciousArraySize);

for(uint256 i = 0; i < maliciousArraySize; i++) {

    maliciousData[i] = abi.encodePacked("dos_attack_", i);

}

// In vulnerable contracts, this would cause DoS

assertTrue(maliciousData.length > 1000); // Exceeds reasonable limits


emit GasLimitExceeded(maliciousArraySize);

}
```

Code

Line 55 - 67 (SyndicateSequencingChain.sol):

```
function processTransactionsBulk(bytes[] calldata data) external {

    for (uint256 i = 0; i < data.length; i++) { // No limit on data.length

        // ... logic ...

    }

}
```



	<pre> } } Line 28 - 48 (AtomicSequencerImplementation.sol): for (uint256 i = 0; i < chains.length; i++) { // No limit on chains.length chains[i].processTransaction(transactions[i]); } (RequireAndModule.sol, RequireOrModule.sol, RequireCompositeModule.so): while (currentCheck != address(0)) { // Loop continues as long as the list has elements if (!IPermissionModule(currentCheck).isAllowed(msgSender, txOrigin, data)) { revert AndPermissionCheckFailed(currentCheck, msgSender, data); } // ... continues without a bound on the number of checks } </pre>
Result/Recommendation	<p>To mitigate this DoS vector, strict upper bounds must be enforced on all functions that process arrays or iterate through lists based on user input.</p> <ol style="list-style-type: none"> 1. Bound Array Inputs: For functions that accept arrays (processTransactionsBulk, processTransactionsAtomically), add a require statement to limit the maximum length of the array.

2. Bound List Growth: For linked-list structures like the permission modules, enforce a maximum size when new elements are added (addPermissionCheck).

Suggested Code:

```
// Define and use reasonable constants for limits

uint256 public constant MAX_BULK_TRANSACTIONS = 500;

uint256 public constant MAX_PERMISSION_CHECKS = 50;

// In SyndicateSequencingChain.sol

function processTransactionsBulk(bytes[] calldata data) external {

    // Add a size check to prevent DoS

    require(data.length <= MAX_BULK_TRANSACTIONS, "Batch size is too large");

    for (uint256 i = 0; i < data.length; i++) {

        // ...

    }

}

// In BaseRequirementModule.sol (or wherever checks are added)

function addPermissionCheck(address _address, bool addToHead) public override
onlyOwner {

    // Add a size check to prevent DoS in check loops
```



```

        require(AddressStructuredLinkedList.sizeOf(permissionChecks) <
MAX_PERMISSION_CHECKS, "Cannot add more permission checks");

        // ...

    }

```

6.2.3 Chain ID Collision Vulnerability in getNextChainId()

Severity: MEDIUM

Status: FIXED

File(s) affected: SyndicateFactory.sol

Update: <https://github.com/SyndicateProtocol/syndicate-appchains/pull/630/commits/c7bbcc984b71f889c0ab8310ddef379fabcd9c63>

Attack / Description

The getNextChainId() function generates new chain IDs by converting a namespacePrefix and the nextAutoChainId to strings and concatenating them. This method is unsafe as it creates ambiguous results. For example, a namespace of 12 and an ID of 34 produces the same final ID ("1234") as a namespace of 123 and an ID of 4.

If an administrator updates the namespacePrefix in such a way that the next generated ID collides with an ID that has already been created and stored, all future calls to createSyndicateSequencingChain() will revert. This will cause a permanent Denial-of-Service for the factory's primary function, making it impossible to deploy new sequencing chains.

The impact is a permanent DoS of the core factory functionality, which disrupts business continuity and prevents protocol expansion.

Proof of Concept:

```
function test_Medium02_ChainIdCollisionExploit() public {
```



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

```
vm.startPrank(owner);

// Demonstrate collision potential with string concatenation
string memory collision1 = string(abi.encodePacked("12", "34"));
string memory collision2 = string(abi.encodePacked("123", "4"));

// Both produce identical results - this shows the vulnerability exists
assertEq(keccak256(bytes(collision1)), keccak256(bytes(collision2)),
"Collision demonstration");

console.log("String concatenation collision demonstrated: '12'+'34' ==
'123'+'4'");

// Test current implementation behavior
factory.updateNamespaceConfig(12);

uint256 currentAutoId = factory.nextAutoChainId();

uint256 chainId1 = factory.getNextChainId();

console.log("Chain ID with namespace 12, autoId", currentAutoId, ":",
chainId1);

// The vulnerability exists in the design, even if hard to trigger in
current state

console.log("MEDIUM-02 DEMONSTRATED: String concatenation enables potential
collisions");
```


	<pre>vm.stopPrank(); }</pre>
Code	<p>Line 110 - 118 (SyndicateFactory.sol):</p> <pre>function getNextChainId() public view returns (uint256) { string memory prefixStr = Strings.toString(namespacePrefix); string memory chainIdStr = Strings.toString(nextAutoChainId); // Unsafe concatenation leads to collisions string memory combined = string(abi.encodePacked(prefixStr, chainIdStr)); return Strings.parseUint(combined); }</pre>
Result/Recommendation	<p>To fix this vulnerability, the chain ID generation logic must be changed from unsafe string concatenation to a deterministic and unambiguous arithmetic operation. The recommended approach is to use multiplication and addition with a fixed offset.</p> <p>This ensures that each namespacePrefix has its own reserved range of IDs, making collisions impossible..</p> <p>Suggested Code:</p> <pre>// In SyndicateFactory.sol // Define a large, fixed upper bound for the auto-incrementing part of the ID.</pre>

```

uint256 constant ID_UPPER_BOUND = 1_000_000_000;

function getNextChainId() public view returns (uint256) {

    // Ensure the auto-incrementing ID does not exceed its reserved space.

    require(nextAutoChainId < ID_UPPER_BOUND, "SyndicateFactory: ID overflow");

    // Use arithmetic to create a unique, unambiguous ID.

    return (namespacePrefix * ID_UPPER_BOUND) + nextAutoChainId;

}

```

6.2.4 Missing Implementation Validation in upgradeImplementation()

Severity: MEDIUM

Status: FIXED

File(s) affected: ArbConfigManager.sol

Update: <https://github.com/SyndicateProtocol/syndicate-appchains/pull/630/commits/49e777e48bfa32ea3d30df5ae09fd653c0e153b4>

Attack / Description

The upgradeImplementation() function is responsible for upgrading the logic contract for all beacon proxies. While it is correctly protected by an onlyOwner modifier, it fails to perform any validation on the newImplementation address, other than checking that it is not the zero address.

This allows the owner to upgrade the implementation to an Externally Owned Account (EOA) or a contract that does not conform to the required IArbChainConfig interface. If this occurs, all subsequent calls to the proxy contracts will fail because they will be delegating calls to an address with no code or incorrect function signatures.

The impact is a complete and potentially irreversible failure of all configuration contracts managed by the ArbConfigManager. If an owner's key is compromised or an error is made, the entire system can be bricked.



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

Proof of Concept:

```
function test_Medium03_MissingImplementationValidation() public {  
  
    // Current implementation should work  
  
    address currentImpl = configManager.beacon().implementation();  
  
    assertTrue(currentImpl.code.length > 0, "Current implementation should have  
code");  
  
    // EXPLOIT: Try to upgrade to EOA (should be prevented but isn't)  
  
    address eoa = address(0x999);  
  
    // Verify EOA has no code  
  
    assertEq(eoa.code.length, 0, "EOA should have no code");  
  
    vm.startPrank(owner);  
  
    // This should fail but doesn't due to missing validation  
  
    configManager.upgradeImplementation(eoa);  
  
    vm.stopPrank();  
  
    // Verify the beacon now points to EOA  
  
    assertEq(configManager.beacon().implementation(), eoa, "Implementation upgraded  
to EOA");  
}
```



	<pre> console.log("MEDIUM-03 EXPLOITED: Beacon upgraded to EOA without validation"); } </pre>
Code	<p>Line 139 - 147 (ArbConfigManager.sol)</p> <pre> function upgradeImplementation(address newImplementation) external onlyOwner { require(newImplementation != address(0), "New implementation cannot be zero address"); // No validation that newImplementation is a contract // No interface compliance check beacon.upgradeTo(newImplementation); emit ImplementationUpgraded(newImplementation); } </pre>
Result/Recommendation	<p>To fix this vulnerability, the upgradeImplementation function must be updated to validate the newImplementation address thoroughly before calling beacon.upgradeTo().</p> <p>Contract Existence Check: Ensure the newImplementation address has code deployed to it by checking newImplementation.code.length > 0.</p> <p>Interface Compliance Check (Optional but Recommended): For defense-in-depth, verify that the new implementation supports the required IArbChainConfig interface using an ERC165 check.</p> <p>Suggested Code:</p>

```
// In ArbConfigManager.sol

function upgradeImplementation(address newImplementation) external onlyOwner {

    require(newImplementation != address(0), "New implementation cannot be zero address");

    // Add a check to ensure the new implementation is a contract

    require(newImplementation.code.length > 0, "Implementation must be a contract");

    // Optional but recommended: Check for interface compliance

    require(IERC165(newImplementation).supportsInterface(type(IArbChainConfig).interfaceId)
    ,

        "Implementation does not support IArbChainConfig interface");

    beacon.upgradeTo(newImplementation);

    emit ImplementationUpgraded(newImplementation);

}
```

LOW ISSUES

During the audit, softstack's experts found **one Low issues** in the code of the smart contract

6.2.5 Front-Running Attack on AssertionPoster Configuration

Severity: LOW

Status: FIXED



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

File(s) affected: AssertionPoster.sol

Update: <https://github.com/SyndicateProtocol/syndicate-appchains/pull/630>

Attack / Description	<p>The configure() function in AssertionPoster.sol is intended to be called via delegatecall from an executor contract. The function correctly validates that address(this) == address(executor), ensuring it runs in the context of the executor. However, it does not implement additional access control to verify who initiated the call to the executor.</p> <p>While the function does lack additional caller validation beyond the delegatecall check, the actual impact depends on the executor contract's implementation details. If the executor contract has proper access controls (which is typical for upgrade executors), this vulnerability may not be exploitable in practice. The risk is partially valid but requires specific conditions to be exploitable.</p> <p>The theoretical impact would be front-running of configuration calls, but this requires the executor itself to be vulnerable or misconfigured. In most standard implementations, the executor would have its own access controls that prevent unauthorized configuration calls</p>
Code	<p>Line 185 - 195 (AssertionPoster.sol):</p> <pre>function configure() external { require(address(this) == address(executor), // Only checks for delegatecall context. // No check on who called the executor. "must configure via upgradeExecutor.execute(AssertionPoster.configure)"); }</pre>

	<pre> if (legacy) { _configureLegacy(); } else { _configureNew(); } } </pre>
Result/Recommendation	<p>To mitigate this vulnerability, the configure() function must implement strict access control to verify the identity of the account initiating the configuration transaction. This check should be in addition to the existing delegatecall validation.</p> <p>The recommended approach is to check that tx.origin or the executor's msg.sender matches the address of the legitimate owner.</p> <p>Suggested Code:</p> <pre> // It is assumed that an `expectedOwner` address is set in storage during deployment. address public expectedOwner; function configure() external { require(address(this) == address(executor), "must configure via upgradeExecutor.execute(AssertionPoster.configure)") } </pre>

```
);

// Add an access control check to verify the caller's identity.

require(msg.sender == expectedOwner, "AssertionPoster: Unauthorized
configuration caller");

if (legacy) {

    _configureLegacy();

} else {

    _configureNew();

}

}
```

INFORMATIONAL ISSUES

During the audit, softstack's experts found **zero Informational issues** in the code of the smart contract



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

6.3 Verify Claims


6.3.1 Sequencing Logic Validation

The sequencing contract must enforce correct transaction ordering and permission checks, ensuring that sequencing rules are implemented as specified.

Status: tested and verified 


6.3.2 Modular Architecture Verification

Requirement modules (AND, OR, composite, etc.) must interact reliably with the main contract. The audit will validate module registration, validation logic, and external contract calls.

Status: tested and verified 


6.3.3 Permission Management & Role Separation

Role-based access controls must enforce clear privilege separation for sequencing actions. The audit will assess configuration safety and protection against role escalation.

Status: tested and verified 


6.3.4 Cross-Chain Bridge Integration

Arbitrum and Optimism bridge proxies must securely manage asset movement between chains. The audit will focus on message flow, verification steps, and replay protection across layers.

Status: tested and verified 


6.3.5 Bridge Rate Limiting and Flow Control

Rate limiting mechanisms in bridge contracts must prevent abuse and ensure predictable transaction settlement. Correct enforcement of thresholds and cooldowns will be tested.

Status: tested and verified 

6.3.6 CREATE3 Deployment and Configuration

CREATE3 must ensure deterministic and collision-free deployments. The audit will validate address predictability, salt uniqueness, and initialization safety.

Status: tested and verified 



7. Executive Summary

Two independent softstack experts performed an unbiased and isolated audit of the smart contract provided by the Syndicate team. The main objective of the audit was to verify the security and functionality claims of the smart contract. The audit process involved a thorough manual code review and automated security testing.

Overall, the audit identified a total of 5 issue, classified as follows:

- no critical issue were found.
- 1 high severity issues were found.
- 3 medium severity issues were found.
- 1 low severity issues were discovered

The audit report provides detailed descriptions of each identified issue, including severity levels, proof of concepts and recommendations for mitigation. We recommend the Syndicate team to review the suggestions.

Update (27.07.2025): The Syndicate team has successfully mitigated all identified issues. The smart contracts have been updated in line with the recommended fixes, and all critical logic paths have been re-reviewed.



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984

8. About the Auditor

Established in 2017 under the name Chainsulting, and rebranded as softstack GmbH in 2023, softstack has been a trusted name in Web3 Security space. Within the rapidly growing Web3 industry, softstack provides a comprehensive range of offerings that include software development, cybersecurity, and consulting services. Softstack's competency extends across the security landscape of prominent blockchains like Solana, Tezos, TON, Ethereum and Polygon. The company is widely recognized for conducting thorough code audits aimed at mitigating risk and promoting transparency.

The firm's proficiency lies particularly in assessing and fortifying smart contracts of leading DeFi projects, a testament to their commitment to maintaining the integrity of these innovative financial platforms. To date, softstack plays a crucial role in safeguarding over \$100 billion worth of user funds in various DeFi protocols.

Underpinned by a team of industry veterans possessing robust technical knowledge in the Web3 domain, softstack offers industry-leading smart contract audit services. Committed to evolving with their clients' ever-changing business needs, softstack's approach is as dynamic and innovative as the industry it serves.

Check our website for further information: <https://softstack.io>

How We Work



1 -----

PREPARATION

Supply our team with audit ready code and additional materials



2 -----

COMMUNICATION

We setup a real-time communication tool of your choice or communicate via e-mails.



3 -----

AUDIT

We conduct the audit, suggesting fixes to all vulnerabilities and help you to improve.



4 -----

FIXES

Your development team applies fixes while consulting with our auditors on their safety.



5 -----

REPORT

We check the applied fixes and deliver a full report on all steps done.



hello@softstack.io
www.softstack.io

softstack GmbH
Schiffbrückstraße 8
24937 Flensburg

CRN: HRB 12635 FL
VAT: DE317625984