



SMART CONTRACT SECURITY AUDIT

Clique

Distribution and Vesting Contracts

CLIENT

Clique

PLATFORM

Ethereum

LANGUAGE

Solidity

REVIEW

March 2025

DOCUMENT

Document information

Project	Clique, Distribution and Vesting Contracts
Client	Clique
Platform	Ethereum
Language	Solidity
Repository	clique-external/b3-contracts
Commit	Not recorded in the original report
Methods	Manual Analysis, Functional Testing, Automated Testing
First review	17 to 20 March 2025
Updated code received	20 March 2025

Document history

VERSION	DATE	CHANGE
1.0	March 2025	Original report issued View original report
1.1	October 2026	Reformatted into the Safe Edges standard template. Findings unchanged

This report was reformatted into the Safe Edges standard template in October 2026. Findings, severities, recommendations and code references are unchanged from the original report, which is preserved unchanged in the [Safe Edges Audit Portfolio on GitHub](#).

CONTENTS

Contents

01	Executive summary	3
02	Scope	4
03	Methodology	5
04	Severity and status	6
05	Findings summary	7
06	Detailed findings	8
	M-01 Incorrect totalPieces calculation in CliqueLock:calculateClaimableAmount can lead to uneven vesting amount distribution	8
	M-02 Anyone can initiate claims for any stream using just the streamId in CliqueLock:claim	8
	M-03 CliqueDistributorManager:createDistributor should explicitly check for signature non-malleability	9
	L-01 Excessive Ether sent to Distributor:claim is not refunded	9
	L-02 Missing fee validation checks in Distributor:setFee	9
	L-03 CliqueLock:_createStream lacks multiple validations before creating a stream	10
	I-01 Duplicate import of ECDSA in CliqueDistributorManager.sol	10
	I-02 Missing check for signature usage in CliqueDistributorManager:createDistributor	10
	I-03 Use SafeERC20 for secure token transfers	11
	I-04 Missing address(0) checks for critical state variables	11
	I-05 Remove unused custom errors to optimize gas usage	11
A	Appendix: vulnerability checklist	12
	Disclaimer	13

Executive summary

About Clique

These contracts form part of the Clique token distribution and management system, providing secure and efficient mechanisms for token distribution and vesting.

Summary of the review

Safe Edges reviewed Clique's token distribution and vesting contracts (b3-contracts) on Ethereum between 17 and 20 March 2025. Eleven issues were identified: three Medium, three Low and five Informational. The Clique team fixed five and acknowledged six, including two of the three Medium issues.

Findings by severity



Findings by status

STATUS	ISSUES
Resolved	5
Acknowledged	6

Scope

Repository	clique-external/b3-contracts
Commit	Not recorded in the original report
Platform	Ethereum
Language	Solidity

Only the code at the commit above was reviewed. Changes made after this commit are not covered by this report.

Methodology

Methods recorded for this engagement: Manual Analysis, Functional Testing, Automated Testing.

Manual review

Auditors read the in-scope code line by line, tracing control flow, state changes and trust boundaries against the intended behaviour described by the client.

Threat modelling

We map privileged roles, external dependencies and value flows, then test the assumptions the code makes about each of them.

Functional and automated testing

Where tooling exists for the language and chain, we run static analysis and targeted tests to confirm suspected issues and check edge cases.

Fix review

After the client addresses the findings, we review the updated code and record the final status of each issue.

Severity and status

Severity levels

SEVERITY	DEFINITION
Critical	Direct loss or permanent lock of funds, or full compromise of the protocol, with few or no preconditions. Must be fixed before deployment.
High	Significant loss of funds, broken core functionality or manipulation of critical state under realistic conditions. Should be fixed before deployment.
Medium	Limited loss or degraded functionality, or issues that need specific conditions or privileged access to exploit. Should be fixed.
Low	Minor impact, low likelihood, or deviations from best practice. Fix when practical.
Informational	Code quality, documentation, gas and style observations with no direct security impact.

Status definitions

STATUS	DEFINITION
Resolved	The issue was fixed and the fix was verified.
Partially resolved	The risk was reduced but not fully removed.
Acknowledged	The client accepted the issue without changing the code.
Open	The issue was not addressed at the time of the report.
Not recorded	The original report did not record a resolution status for this issue.

Findings summary

ID	TITLE	SEVERITY	STATUS
M-01	Incorrect totalPieces calculation in CliqueLock:calculateClaimableAmount can lead to uneven vesting amount distribution	Medium	● Resolved
M-02	Anyone can initiate claims for any stream using just the streamId in CliqueLock:claim	Medium	● Acknowledged
M-03	CliqueDistributorManager:createDistributor should explicitly check for signature non-malleability	Medium	● Acknowledged
L-01	Excessive Ether sent to Distributor:claim is not refunded	Low	● Acknowledged
L-02	Missing fee validation checks in Distributor:setFee	Low	● Resolved
L-03	CliqueLock:_createStream lacks multiple validations before creating a stream	Low	● Resolved
I-01	Duplicate import of ECDSA in CliqueDistributorManager.sol	Informational	● Resolved
I-02	Missing check for signature usage in CliqueDistributorManager:createDistributor	Informational	● Acknowledged
I-03	Use SafeERC20 for secure token transfers	Informational	● Acknowledged
I-04	Missing address(0) checks for critical state variables	Informational	● Acknowledged
I-05	Remove unused custom errors to optimize gas usage	Informational	● Resolved

Detailed findings

M-01 Incorrect totalPieces calculation in CliqueLock:calculateClaimableAmount can lead to uneven vesting amount distribution

Severity **Medium** Status ● Resolved

DESCRIPTION

The function calculates totalPieces as follows:

This works correctly when $\text{vestingDuration} \% \text{pieceDuration} \neq 0$. However, when vestingDuration is exactly divisible by pieceDuration, an extra piece is added unnecessarily, causing uneven vesting distribution. For example:

- vestingDuration = 13 months, pieceDuration = 3 months → totalPieces = 5 (correct)
- vestingDuration = 12 months, pieceDuration = 3 months → totalPieces = 5 (incorrect, should be 4)

As a result, the last vesting period may receive a disproportionately large share of the vested amount.

IMPACT

Leads to uneven vesting, potentially delaying or accelerating fund distribution unfairly.

RECOMMENDATION

Before adding 1 to totalPieces, check if $\text{vestingDuration} \% \text{pieceDuration} == 0$.

M-02 Anyone can initiate claims for any stream using just the streamId in CliqueLock:claim

Severity **Medium** Status ● Acknowledged

DESCRIPTION

The claim function takes only streamId as an argument and fetches the recipient dynamically:

However, it does not check if the caller (msg.sender) is the intended recipient. This allows a malicious actor to claim another user's stream.

IMPACT

Attackers can steal vested tokens by providing another user's streamId.

RECOMMENDATION

Ensure that only the rightful recipient can initiate the claim:

M-03 CliqueDistributorManager:createDistributor should explicitly check for signature non-malleability

Severity **Medium** Status ● Acknowledged

DESCRIPTION

The function recovers the signer from a hash and signature but does not ensure that the s field is in the lower half order. This allows signature malleability.

IMPACT

Attackers can manipulate valid signatures to create alternate valid signatures.

RECOMMENDATION

Either use OpenZeppelin's ECDSA library or add a check to enforce s being in the lower half:

L-01 Excessive Ether sent to Distributor:claim is not refunded

Severity **Low** Status ● Acknowledged

DESCRIPTION

Users must send a fee to claim tokens. If they send excess Ether, it is not refunded.

IMPACT

Users may lose excess Ether when claiming tokens.

RECOMMENDATION

Ensure the exact fee is sent:

L-02 Missing fee validation checks in Distributor:setFee

Severity **Low** Status ● Resolved

DESCRIPTION

The function lacks constraints on the fee range.

RECOMMENDATION

Define minFee and maxFee limits.

L-03 CliqueLock:_createStream lacks multiple validations before creating a stream

Severity Low Status Resolved

DESCRIPTION

The function does not validate `amount > 0`, `startTime < cliffTime < endTime`, or `block.timestamp < startTime`.

IMPACT

Allows creation of invalid or spam streams.

RECOMMENDATION

consider adding a the validation check in the function

I-01 Duplicate import of ECDSA in CliqueDistributorManager.sol

Severity Informational Status Resolved

DESCRIPTION

The ECDSA library is imported twice, making the code redundant.

RECOMMENDATION

Remove one import

I-02 Missing check for signature usage in CliqueDistributorManager:createDistributor

Severity Informational Status Acknowledged

DESCRIPTION

The function relies on deadline for replay protection but does not prevent multiple uses of the same signature within that timeframe.

IMPACT

Allows signature reuse, leading to spam distributor creation.

RECOMMENDATION

Include a nonce in the signature hash to prevent replay attacks.

I-03 Use SafeERC20 for secure token transfers

Severity **Informational** Status ● Acknowledged

DESCRIPTION

It is recommended to use OpenZeppelin's SafeERC20 library for secure transfers.

I-04 Missing address(0) checks for critical state variables

Severity **Informational** Status ● Acknowledged

DESCRIPTION

Certain functions do not check for address(0), which could introduce errors.

- Found in Distributor.sol
- Found in CliqueDistributorManager.sol

RECOMMENDATION

Validate addresses before assignment.

I-05 Remove unused custom errors to optimize gas usage

Severity **Informational** Status ● Resolved

DESCRIPTION

Unused custom errors increase contract size and gas costs.

- Found in CliqueDistributorManager.sol
- Found in Distributor.sol

RECOMMENDATION

Remove these errors if they are not used anywhere in the contract

APPENDIX A

Vulnerability checklist

Vulnerability classes checked during this review, as listed in the original report.

✓ Arbitrary write to storage	✓ Upgradeable safety
✓ Centralization of control	✓ Using inline assembly
✓ Ether theft	✓ Style guide violation
✓ Improper or missing events	✓ Parallel Execution safety
✓ Logical issues and flaws	✓ UTXO Model Verification
✓ Arithmetic Computations Correctness	✓ FuelVM Opcodes
✓ Race conditions/front running	✓ Cross-Chain Interactions
✓ Re-entrancy	✓ Modular Design
✓ Malicious libraries	✓ Access Control Vulnerabilities
✓ Address hardcoded	✓ Denial of Service (DoS)
✓ Divide before multiply	✓ Oracle Manipulation
✓ Integer overflow/underflow	✓ Signature Replay Attacks
✓ ERC's conformance	✓ Improper Handling of External Calls
✓ Missing Zero Address Validation	✓ Proxy Storage Collision
✓ Revert/require functions	✓ Use of Deprecated Functions

Disclaimer

This report is based on the code at the commit listed in the Scope section and on information the client provided during the review. It reflects Safe Edges' assessment at the time of the review and is not a guarantee that the code is free of vulnerabilities.

Changes made after the reviewed commit, deployment configuration, off-chain components and third-party dependencies are outside the scope of this report unless stated otherwise.

Security is ongoing. We recommend continued testing, monitoring and bug bounty coverage alongside any audit.

This report is not investment advice and is not an endorsement of the project, its team or any token.



Safe Edges is a Web3 security firm and official security partner for the Fuel, ENI and XI EcoChain ecosystems. We work across smart contract audits, Rust and WASM reviews, ZK circuit review, AI and MCP security, and penetration testing.

200+

PROJECTS AUDITED

\$4B+

TVL SECURED

0

POST-AUDIT HACKS