



SMART CONTRACT SECURITY AUDIT

Clique

Distribution and Lock Hooks

CLIENT

Clique

LANGUAGE

Solidity

REVIEW

October 2025

Document information

Project	Clique, Distribution and Lock Hooks
Client	Clique
Platform	Not recorded in the original report
Language	Solidity
Repository	Not recorded in the original report
Commit	Not recorded in the original report
Methods	Manual code review, static analysis, and best practices evaluation
Review	18 to 28 September 2025
Fix commit	836c1c8
Report date	23 October 2025

Document history

VERSION	DATE	CHANGE
1.0	23 October 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
	C-01 Unchecked overwrite of streamIds mapping in CliqueLockHook.execute() function	8
	H-01 Incorrect Assignment of Immutable IS_FIXED_START in CliqueLockHook.sol	9
	M-01 Missing Validation of Total Proportions in Distributor.setHooks	10
	L-01 Absence of validation in BaseDistributor.setActiveWindow	11
	L-02 Distributor.setHooks allows fallback index equal to strategies.length	12
	L-03 Token Approval Not Reset in CliqueLockHook.execute	13
	L-04 Missing Validation in CliqueLockHook.setStreamPreset	13
	I-01 Unused constructor parameter in TransferHook.sol	14
	I-02 Unused import	14
	I-03 State Change Without Event	15
	G-01 Unused Error	15
	G-02 Avoid zero assignment in for loops counters for gas efficiency	15
	Disclaimer	16

Executive summary

About Clique

These contracts handle Clique's token distribution. A Distributor routes each claim through configurable hook strategies, and CliqueLockHook turns allocations into vesting streams in CliqueLock.

Summary of the review

Safe Edges reviewed Clique's distribution and streaming hook contracts (CliqueLockHook, Distributor, BaseDistributor and TransferHook) between 18 and 28 September 2025. Twelve issues were identified: one Critical, one High, one Medium, four Low and five Informational, two of which are gas optimisations. The Critical issue let anyone call CliqueLockHook.execute and overwrite a recipient's stream reference, and the High issue was an immutable variable assignment that stopped the contract compiling. Clique fixed six issues in commit 836c1c8, including the Critical and High, and acknowledged six: the Medium, all four Low and one gas issue.

Findings by severity



Findings by status

STATUS	ISSUES
Resolved	6
Acknowledged	6

Scope

Repository	Not recorded in the original report
Commit	Not recorded in the original report
Platform	Not recorded in the original report
Language	Solidity

IN-SCOPE FILES

```
CliqueLockHook
Distributor
BaseDistributor
TransferHook
```

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 code review, static analysis, and best practices evaluation.

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
C-01	Unchecked overwrite of streamIds mapping in CliqueLockHook.execute() function	Critical	● Resolved
H-01	Incorrect Assignment of Immutable IS_FIXED_START in CliqueLockHook.sol	High	● Resolved
M-01	Missing Validation of Total Proportions in Distributor.setHooks	Medium	● Acknowledged
L-01	Absence of validation in BaseDistributor.setActiveWindow	Low	● Acknowledged
L-02	Distributor.setHooks allows fallback index equal to strategies.length	Low	● Acknowledged
L-03	Token Approval Not Reset in CliqueLockHook.execute	Low	● Acknowledged
L-04	Missing Validation in CliqueLockHook.setStreamPreset	Low	● Acknowledged
I-01	Unused constructor parameter in TransferHook.sol	Informational	● Resolved
I-02	Unused import	Informational	● Resolved
I-03	State Change Without Event	Informational	● Resolved
G-01	Unused Error	Informational	● Resolved
G-02	Avoid zero assignment in for loops counters for gas efficiency	Informational	● Acknowledged

Detailed findings

C-01 Unchecked overwrite of streamIds mapping in CliqueLockHook.execute() function

Severity **Critical** Status ● Resolved

DESCRIPTION

execute() function unconditionally sets `streamIds[_root][_recipient] = streamId` after creating a stream. The function performs no access-control check preventing arbitrary callers from invoking execute(), nor does it prevent a second call from overwriting an existing streamId. As a result, an attacker (or an accidental duplicate execution by the distributor) can replace the CliqueLockHook's reference to an earlier stream for the same (root, recipient) pair.

```
function execute(address _vault, address _recipient, bytes32 _root, uint256 _allocatedamount,
bytes memory _extra)
    external
    returns (uint256 _consumed)
{
    ....
    _distributor().hookTransfer(address(this), _amount);
    token.safeApprove(LOCK, _amount);
    uint256 streamId = CliqueLock(LOCK).createStream(_streamConfig);
    @>> streamIds[_root][_recipient] = streamId;
    return _amount;
}
```

The claim() function will only point to the most-recent streamId. If overwritten, legitimate, earlier streams remain no longer addressable through the hook.

```
function claim(bytes32 _root, address _recipient) external {
    @>> uint256 streamId = streamIds[_root][_recipient];
    CliqueLock(LOCK).claim(streamId);
}
```

IMPACT

Threat actor can cause low cost state corruption and griefing attack.

ATTACK PATH

1. The attacker deploys a contract that imports IDistributor.sol and implements the minimal functions the hook calls and funds the contract

```
import "./IDistributor.sol";

contract Attacker is IDistributor {
    function configurationId(bytes32 root) external view returns (bytes32) {
        return someConfigurationId; // return a config that maps to a valid StreamPreset
    }
    function hookTransfer(address to, uint256 amount) external {
        IERC20(token).transfer(to, amount);
    }
    // implement other IDistributor functions as needed by the hook
}
```

2. Attacker calls execute on the hook using the victims tuple: `execute(victimVault, victimRecipient, victimRoot, _allocatedAmount, _extra)`. Here `_extra` decodes to `_amount = 1`.

3. The hook approves LOCK and calls `CliqueLock.createStream()`. The call succeeds (streamConfig valid) and returns `attackerStreamId`

4. Hook overwrites mapping : `streamIds[victimRoot][victimRecipient] = attackerStreamId;`

5. When the victim claim(`victimRoot, victimRecipient`), the hook forwards only the attackers streamId to `CliqueLock.claim()`. The victim receives the tiny attacker-funded amount; the original stream is no longer referenced by the hook.

RECOMMENDED MITIGATION

Consider enforcing an explicit `onlyDistributor()` modifier to ensure that only the authorized distributor contract can invoke this function.

H-01 Incorrect Assignment of Immutable IS_FIXED_START in CliqueLockHook.sol

Severity High Status ● Resolved

DESCRIPTION

The contract declares `IS_FIXED_START` as an immutable variable and initializes it to false. However, the constructor attempts to assign a new value to `IS_FIXED_START` during deployment. In Solidity, immutable variables cannot be reassigned after declaration; the compiler reserves them for one-time assignment (either at declaration or in the constructor, but not both). This results in a compile-time error, when being set to true.

IMPACT

The contract will fail to compile and cannot be deployed, blocking deployment entirely.

RECOMMENDED MITIGATION

Remove the default initialization at the declaration and assign `IS_FIXED_START` only in the constructor, for example:

```
- bool public immutable IS_FIXED_START = false;
+ bool public immutable IS_FIXED_START;

constructor(address _lock, bool _isFixedStart) {
+   IS_FIXED_START = _isFixedStart;
    ...
}
```

M-01 Missing Validation of Total Proportions in Distributor.setHooks

Severity **Medium** Status ● Acknowledged

DESCRIPTION

The setHooks function allows the admin to configure multiple strategies for a batch and assigns the HOOK role to each strategy's hook contract. While it validates that each individual strategy's proportion does not exceed $1e18$ (100%), it does not check that the sum of all strategy proportions is $1e18$.

As a result, if the total of all proportions exceeds $1e18$, the `_execute` function may attempt to allocate more tokens than the claim `_amount`. This can cause `InsufficientAllocation()` or `ConsumedAmountMismatch()` reverts, effectively preventing legitimate claims from being processed.

IMPACT

Claims can fail unexpectedly, creating a potential DoS scenario for users.

RECOMMENDED MITIGATION

Add a validation step in setHooks to ensure the sum of all strategy proportions does not exceed $1e18$.

```
function setHooks(bytes32 _configurationId, Strategy[] memory _strategies, uint256 _fallbackHook)
    external
    onlyRoles(PROJECT_ADMIN)
{
    Strategy[] memory oldStrategies = batchConfigurations[_configurationId].strategies;
    for (uint256 i = 0; i < oldStrategies.length; i++) {
        _removeRoles(oldStrategies[i].hook, HOOK);
    }
+   uint256 sum = 0;
    for (uint256 i = 0; i < _strategies.length; i++) {
        if (_strategies[i].proportion > 1 ether) {
            revert InvalidStrategies();
        }
+       sum += _strategies[i].proportion;
        _setRoles(_strategies[i].hook, HOOK);
    }
+   if (sum > 1 ether) revert InvalidStrategies();
    if (_fallbackHook > _strategies.length) {
        revert InvalidStrategies();
    }
    batchConfigurations[_configurationId].strategies = _strategies;
    batchConfigurations[_configurationId].fallbackHook = _fallbackHook;
    emit HookAdded(_configurationId);
}
```

L-01 Absence of validation in BaseDistributor.setActiveWindow

Severity Low Status Acknowledged

DESCRIPTION

The setActiveWindow function allows the admin to configure the start and end timestamps of the airdrops active window. Currently, the function does not validate that the _endTime is strictly greater than _startTime. This could potentially allow setting an invalid window (e.g., _endTime <= _startTime), which may prevent the airdrop from ever being active.

IMPACT

Misconfigured active window can cause all claims to fail due to the whenActive modifier reverting.

RECOMMENDED MITIGATION

Add a validation check to ensure _endTime > _startTime.

```
+ error InvalidActiveWindow()

function setActiveWindow(uint256 _startTime, uint256 _endTime) external onlyRoles(PROJECT_ADMIN)
{
+   if (_endTime <= _startTime) revert InvalidActiveWindow()
    activeWindowStartTime = _startTime;
    activeWindowEndTime = _endTime;
    emit ActiveWindowSet(_startTime, _endTime);
}
```

L-02 Distributor.setHooks allows fallback index equal to strategies.length

Severity **Low** Status ● Acknowledged

DESCRIPTION

The setHooks function allows the admin to configure a fallback hook for a batch by specifying its index in the strategies array. The current validation checks that the fallback index is not greater than strategies.length:

```
if (_fallbackHook > _strategies.length) revert InvalidStrategies();
```

However, this allows `_fallbackHook == strategies.length`, which is out of bounds for the array. In the `_execute` function, the fallback hook is only executed if:

```
if (_batchConfiguration.fallbackHook < _batchConfiguration.strategies.length)
```

Thus, setting `_fallbackHook == strategies.length` effectively disables the fallback hook. If other hooks do not consume the full `_amount`, `totalConsumed != _amount` and the claim will revert (`ConsumedAmountMismatch()`), potentially breaking legitimate claims.

IMPACT

Misconfigured fallback index can prevent fallback execution, potentially causing claims to revert even if all other hooks execute correctly.

RECOMMENDED MITIGATION

Consider implementing following changes:

```
- if (_fallbackHook > _strategies.length) {  
+ if (_fallbackHook >= _strategies.length) {  
    revert InvalidStrategies();  
}
```

L-03 Token Approval Not Reset in CliqueLockHook.execute

Severity **Low** Status ● Acknowledged

DESCRIPTION

The execute function in CliqueLockHook approves the LOCK contract for `_amount` tokens before creating a stream. However, for some ERC20 tokens such as USDT, setting a new allowance without first resetting it to zero will cause the transaction to revert. This is because USDT and similar tokens enforce that any non-zero allowance must first be set to zero before updating it to a new value. The current implementation does not account for this behavior, which may lead to failed executions when interacting with such tokens.

IMPACT

Transactions using tokens like USDT may revert unexpectedly during repeated executions, causing legitimate stream creations to fail.

RECOMMENDED MITIGATION

Before approving the LOCK contract, check the current allowance and reset it to zero if it is non-zero.

```
+ token.safeApprove(LOCK, 0);
_distributor().hookTransfer(address(this), _amount);
token.safeApprove(LOCK, _amount);
uint256 streamId = CliqueLock(LOCK).createStream(_streamConfig);
streamIds[_root][_recipient] = streamId;
```

L-04 Missing Validation in CliqueLockHook.setStreamPreset

Severity **Low** Status ● Acknowledged

DESCRIPTION

The setStreamPreset function allows the contract owner to set a StreamPreset struct for a given configurationId. Currently, there are no validation checks on the fields of the struct. For example, parameters such as pieceDuration, vestingDuration, or unlock percentages could be set to zero or values outside expected ranges, which may lead to incorrect vesting behavior or runtime errors when streams are created.

IMPACT

Setting invalid values could potentially cause streams to revert on creation or failed during claims.

RECOMMENDED MITIGATION

It is possible that the necessary validations are performed later during the corresponding functions; however, performing validation before setting the values is still strongly recommended, similar to CliqueLock.

```
+ error InvalidStream();

function setStreamPreset(bytes32 _configurationId, StreamPreset memory _streamPreset) external
onlyOwner {
    // Validate durations
    + if(_streamPreset.pieceDuration == 0 && _streamPreset.vestingDuration == 0) revert
    InvalidStream();
    + if (_streamPreset.startUnlockPercentage + _streamPreset.cliffUnlockPercentage > 1e18) revert
    InvalidSchedule();
    streamPresets[_configurationId] = _streamPreset;
}
```

I-01 Unused constructor parameter in TransferHook.sol

Severity Informational Status ● Resolved

DESCRIPTION

The TransferHook contract accepts `_projectAdmin` as a constructor parameter during deployment. However, this parameter is neither stored nor used within the contract, effectively making the constructor a no-op. While this does not introduce a direct security vulnerability, it can lead to confusion, and may signal incomplete or incorrect contract initialization logic.

RECOMMENDED MITIGATION

Consider implementing the following changes:

```
- constructor(address _projectAdmin) {}
+ constructor() {}
```

I-02 Unused import

Severity Informational Status ● Resolved

DESCRIPTION

Distributor.sol imports IDistributor but does not reference it anywhere. This import is redundant and can be safely removed to improve clarity and compilation efficiency.

```
- import {IDistributor} from "./IDistributor.sol";
```

I-03 State Change Without Event

Severity Informational Status ● Resolved

DESCRIPTION

Some functions modify important state variables but do not emit corresponding events. This omission makes it difficult for off-chain services to track state changes reliably.

FOUND IN CLIQUELOCKHOOK.SOL

```
function setStreamPreset(bytes32 _configurationId, StreamPreset memory _streamPreset) external
onlyOwner {
    function execute(address _vault, address _recipient, bytes32 _root, uint256 _allocatedamount,
    bytes memory _extra)
```

FOUND IN BASEDISTRIBUTOR.SOL

```
function setSigner(address _signer) external onlyRoles(PROJECT_ADMIN) {
    function setDynamicRecipient(bool _dynamicRecipient) external onlyRoles(PROJECT_ADMIN) {
    function setIsWrappedToken(bool _isWrappedToken) external onlyRoles(PROJECT_ADMIN) {
```

G-01 Unused Error

Severity Informational Status ● Resolved Category Gas

DESCRIPTION

Consider removing unused error to save gas

```
- error TransferFailed(address token, address from, address to, uint256 amount);
- error DefaultFeeNotSet();
- error TransferFailed();
```

G-02 Avoid zero assignment in for loops counters for gas efficiency

Severity Informational Status ● Acknowledged Category Gas

DESCRIPTION

To optimize gas efficiency in a for loop, it is recommended to avoid explicitly setting the counter to zero, as it is already initialized to zero by default. Removing unnecessary variable assignments will further enhance the contract's gas efficiency.

```
- for (uint256 i = 0; i < ...; ) {...}
+ for (uint256 i; i < ...; ) {...}
```

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