NuCypher WorkLock

Security Assessment (Summary)

January 22, 2020

Prepared For:
MacLane Wilkison | *NuCypher*
maclane@nucypher.com

Prepared By:
David Pokora | *Trail of Bits*
david.pokora@trailofbits.com

Michael Colburn | *Trail of Bits*
michael.colburn@trailofbits.com

Changelog:

January 22, 2020:	Initial summary draft
January 27, 2020:	Revised summary language
February 12, 2020:	Copyedited

Assessment Summary

During the week of January 13 to January 17, 2020, Trail of Bits performed an assessment of the NuCypher WorkLock token distribution protocol, initially working from commit hash `b115d3c848c4235e3903ac5c8d5ebebab18d3e4d` of the NuCypher repository. Changes introduced to WorkLock during the assessment were also re-evaluated from commit hash `7a60009d631b31da6f9eabef32dc7cf56f4ebda0`.

Throughout this assessment, we sought to answer various questions about the security of the protocol, which generally yielded positive results. Our manual review did not identify any major concerns regarding the WorkLock platform. A non-exhaustive list of the approaches taken and their results include:

- Attempts to overwrite a previous bid with a new bid were not possible as bids are aggregated in a mapping per `msg.sender`.
- Arithmetic surrounding ETH-to-token distribution ratios could not be manipulated by canceling a bid, as the deposited ETH count and ETH supply will not change after bidding has closed; rather, the proportionate amount of tokens will be considered "unclaimed" and burnable.
- Prior to the refactor, transfer operations and accounting of balances within the `PreallocationEscrow` yielded positive security results, with appropriate arithmetic and control flow in place. For instance, balances were checked appropriately when withdrawing, and the Solidity assembly block in `receiveApproval` was observed to be non-problematic. Similarly, after refactoring, the `StakingEscrow` was observed to properly handle arithmetic surrounding balance-related operations such as deposits.
- Reviewing the logic in WorkLock's `claim` method yielded positive results. Claiming appropriately requires that the bidding period has ended. Prior to the removal of the `PreallocationEscrow`, multiple claims could not be made for the same user, as a `PreallocationEscrow` is linked to the user and their funds, and the existence of a previously linked `PreallocationEscrow` is checked appropriately. Other methods such as `cancelBid` and `refund` appropriately checked the existence of a `PreallocationEscrow` so users could not withdraw a disproportionate amount of tokens or ETH. After removal of the `PreallocationEscrow`, these checks were appropriately replaced with checks against a boolean which is similarly set when tokens are claimed, etc.
- Investigation of arithmetic and `require` statements in the `refund` codepath found refund calculation to be sound. The refund is calculated as the delta of work done, accounting for previous refunds. Work measurements are appropriately updated and disabled upon completion of all work.

- Analysis of special cases in a WorkLock instance, such as not funding the contract with tokens before the ending bid date, was also found to be non-problematic. In such a case, canceling the bid will simply burn your tokens (zero) and return your ETH as intended.

Although the assessment yielded positive results regarding vulnerability, Trail of Bits recommends incorporating the following changes related to code quality and documentation:

- WorkLock's constructor would benefit from additional comments that indicate the significance of the `require` statements. (WorkLock.sol#L77-L86)
- The `require` statement in WorkLock's `bid` method could use less ambiguous language to indicate why the `require` may fail. (WorkLock.sol#L148-L149)
- When canceling a bid using WorkLock's `cancelBid`, if the end bid date has not passed, the ETH supply will be decreased. If the end bid date has been reached, tokens will instead be moved to a separate `unclaimedTokens` balance, to be burned later. The rationale behind this decision is to maintain ETH-to-token ratios for token distribution arithmetic, which could be better documented for external parties. Similarly, additional comments clarifying the arithmetic within the `refund` method may be beneficial. (WorkLock.sol#L166-L170)
- The use of SafeMath's `sub` operation at the end of the `getRemainingWork` method is unnecessary due to arithmetic checks that precede it. (WorkLock.sol#L138-L141)
- Resolve the TODO comment within `cancelBid` and add appropriate user documentation if necessary (WorkLock.sol#L160). This will ensure users are aware of methods an attacker may employ to game the system, such as scenarios previously mentioned in issue #1508.

Overall, NuCypher has taken reasonable approaches to ensuring security, employing the use of static analyzers such as Slither, property testing through Echidna, and otherwise unit testing to uncover any remaining low hanging fruit. The simplicity of the WorkLock protocol's contracts promotes a simplistic state machine that is relatively easy to follow and minimizes the possibility of trapping state transitions.