Kin Token Audit

SEPTEMBER 11, 2017 | IN SECURITY AUDITS | BY OPENZEPPELIN SECURITY

The Kik team asked us to review and audit their Kin Token (KIN) contract. We looked at the code and now publish our results.

The audited contracts are in the kik-interactive/kin-token repository. The version used for this report is commit `3ed3a383b9304274ec22f41769716cadb854727f`.

The code is good, very straightforward, and excellently tested.

Here's our assessment and recommendations, in order of importance.

Update: *The Kik team followed most of our recommendations and updated the contract. The issues that weren't fixed do not affect public sale participants. The new commit is* `376114bf92e4a1a245bc7bb2c8c84c02885db125`._

Critical Severity

Revoking a vesting grant takes vested tokens away

The `revoke` function in `VestingTrustee` allows the owner to revoke an address's vesting grant, in the case that it is revocable. This consists of erasing the grant and transferring the remaining tokens to the owner. The function considers "remaining tokens" to be all those that haven't been unlocked and transferred yet. However, it is likely that remaining tokens should be considered to be only those which have not yet vested, regardless of whether they've been transferred or not. Consider changing `revoke` to first transfer vested tokens to the grantee, and only the remaining as a refund to the owner.

Update: *The team assured us that, although they have not implemented our suggested change, the risk is mitigated by the fact that all vesting grants will be made irrevocable as soon as possible after the sale and that revocation will be only possible by getting access to the multisig owner of the contract. There is no risk for an irrevocable grant, so there would be no problem for investors after made irrevocable.*

High Severity

Unexpected value for MAX_TOKENS_SOLD

The state variable `MAX_TOKENS_SOLD` in `KinTokenSale` represents the maximum number of tokens to be sold during the crowdsale, which is defined to be 1 trillion according to the whitepaper (chapter 6, Kin token issuance). However, the value in the contract is 512192121951 (slightly above half a trillion). Update this constant to reflect the value indicated in the whitepaper.

Update: *The team pointed out that this number represents the amount that will be offered during the sale, while the remaining half of the 1 trillion offered to the public has been sold in a presale.*_

Token ownership can be transferred while minting

The `KinTokenSale` contract allows the owner to transfer the token ownership to a different address at any moment, using the function `requestKinTokenOwnershipTransfer`. Since the token is in a *minting* state during the sale (i.e. new tokens can be created by the owner), this harms the trustlessness of the contract, as it would allow the owner to mint themselves an arbitrary number of tokens, against what was established in paper.

A comment next to the function correctly states that transferring token ownership would prevent the token sale to continue operating. If such a thing happened, the function `acceptKinTokenOwnership` would allow the sale to become the token owner again, and continue operating. However, in the meantime, an owner could end the minting phase of the token manually, violating the explicit requirement that grants be distributed before minting is ended, and that this be done after the sale period.

We recommend to remove the functions `requestKinTokenOwnershipTransfer` and `acceptKinTokenOwnership`, as they are not needed and compromise the correctness and security of the sale.

Update: *The team replied that the owner will be a multisig wallet, which mitigates the risk. Due to time constraints, the functions will be kept as the only method to pause the sale in an emergency.*

All vesting grants are revocable

It should be noted that all vesting grants are revocable (indicated by the last parameter, `true`), including the large grant given to Kik. If a grant is revoked, all of its vesting tokens are immediately transferred to the owner of `VestingTrustee`. Make sure that this is in fact desired.

Update: *The team has clarified that this is a temporary measure so that possible errors can be fixed afterwards, and that all grants will be manually made irrevocable as soon as possible.*

Medium Severity

Participation caps can be altered mid-crowdsale

The functions `setTier1Participants`, `setTier2Participants`, and `setHardParticipationCap` allow the owner of the token sale contract to modify the cap for any participant at any point in time, even after the crowdsale has started. To improve the public trust on the sale mechanics, it is recommended that once the sale starts, all caps are frozen and cannot be modified. Consider adding an `onlyBeforeSale` modifier to these functions.

Update: *The team explained that this is intentional so that potential errors in the KYC process can be corrected by changing a participant's tier at any moment.*

Possible to transfer more balance than necessary to VestingTrustee

It should be noted that it is possible to transfer more token balance than necessary to `VestingTrustee`. In that case, the only way to recover the surplus is to create a new grant that allows to immediately unlock its full amount. Although this is not a problem in the context of the trustee managed by the token sale contract, it could be inconvenient if `VestingTrustee` were to be used by itself.

Low Severity

Pending state variables should be replaced by constructor parameters

The state variables `WEI_PER_USD`, `KIN_FOUNDATION_ADDRESS`, and `KIK_ADDRESS` are meant to be replaced before deployment by the updated values (they are marked "`TODO`"). This is error prone, as one could forget to update them. Consider adding these variables as constructor parameters to enforce that they be set at construction.

***Update:** The suggestion was accepted and it will be implemented soon.*

Magic constants

Consider rewriting the operations involving the magic constant 100000000000 (0.1 trillion) in `initTokenGrants` of `KinTokenSale`. Instead of calculating 60% of 10 trillions as `60 * 100000000000`, make use of the constant `MAX_TOKENS` (10 trillions) in the operation, in order to improve clarity and maintainability. The same applies to the calculation of 30% in the same function .

***Update:** Fixed in this commit.*_

Unbounded loops

Functions `addTokenGrant` and `deleteTokenGrant` in `KinTokenSale` both iterate over the entire `tokenGrantees` storage array. If a large number of `tokenGrantees` is added, then a call to either function may end up requiring more gas than the block gas limit, making it impossible to add or remove any grantees. (See Gas Limit and Loops in the Solidity documentation.)

Consider adding a `MAX_TOKEN_GRANTEES` constant, set to a value which ensures that both `addTokenGrant` and `deleteTokenGrant` can execute without consuming the entire block gas limit. Enforce in all calls to `addTokenGrant` that the total number of `tokenGrantees` is less than `MAX_TOKEN_GRANTEES`.

Additionally, the loop in `addTokenGrant` is not needed, since there will never be a grantee address for which the previous check succeeds.

***Update:** Fixed in this commit.*_

Notes & Additional Information

- Congratulations on writing such a thorough test suite! There are 620 tests in total, with most of them covering the integration of all contracts into the token sale contract.
- The documentation of `grant` in `VestingTrustee` does not state that its user should first do a transfer of tokens to the contract. Consider making this explicit in the documentation. *(**Update:** Fixed here.)*
- There is one use of `var` in the codebase, which infers the variable type from the right hand of the assignment. We recommend avoiding this feature because in some cases it might infer a smaller integer type than the developer might think. It is best to be explicit regarding types; in this case consider replacing it for `uint256`. *(**Update:** Fixed here.)*
- The name of the constant `TOKEN_DECIMALS` is misleading. While the `decimals` constant in the `KinToken` contract is set to the number of decimal positions (18), the constant `TOKEN_DECIMALS` is set to `10**18`, i.e. one unit of the token. Consider either renaming the constant, or changing its value to 18 and updating its usage across the contract. *(**Update:** Fixed here.)*
- The comment in line 47 of `KinTokenSale`, regarding state variables `startTime` and `endTime`, mentions "blocks" but it should say "timestamps." *(**Update:** Fixed here.)*

- The documentation of `unlockVestedTokens` indicates it has a return value, but it doesn't. Remove the comment to avoid potential confusion. *(**Update:** Fixed here.)*
- Consider defining a helper function `saleEnded` to avoid the duplication in `onlyDuringSale` and `onlyAfterSale`. *(**Update:** Fixed here.)*
- The variable `tokensVested` refers to tokens that have not yet vested. Consider renaming the variable to `tokensVesting`. *(**Update:** Fixed here.)*
- `endMinting` fails silently when minting has already ended. Consider reverting the transaction instead.
- There is a typo in `Ownable`, where it says perviously it should say *previously*. *(**Update:** Fixed here.)*
- Consider using `require(isMinting)` in `onlyDuringMinting`, instead of `revert`, and the same in `onlyAfterMinting`, for more concise code. *(**Update:** Fixed here.)*

Conclusion

One critical and three high severity issues were found and explained, along with recommendations on how to fix them. Some changes were proposed to follow best practices and reduce potential attack surface.

Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the Kin Token contract. We have not reviewed the related Kin project. The above should not be construed as investment advice. For general information about smart contract security, check out our thoughts here.

Security Audits

- If you are interested in smart contract security, you can continue the discussion in our forum, or even better, join the team 🚀
- If you are building a project of your own and would like to request a security audit, please do so here.

RELATED POSTS

SECURITY AUDITS

BitClave Token Audit

The BitClave team asked us to review and audit their Consumer Activity Token (CAT) contracts. We…

READ MORE

SECURITY AUDITS

WAX Token Audit

The WAX team asked us to review and audit their WAX Token contract. We looked at the code and now…

READ MORE

SECURITY AUDITS

SuperDAO Promissory Token audit

We've been asked by the SuperDAO team to review and audit their new token code. We looked at their…

READ MORE

Email*

Get our monthly news roundup

由 reCAPTCHA 提供保护
隐私权 - 使用条款

SIGN UP