OpenZeppelin

Tally SafeGuard Audit

OPENZEPPELIN SECURITY | DECEMBER 6, 2021

Security Audits

Introduction

The Tally team asked us to review and audit a set of contracts with the final goal to improve common governance contracts and give them more flexibility. We looked at the code and now publish our results.

System overview

The system relies on three main contracts:

- A `SafeGuard` contract template. This contract is the `admin` of a `Timelock` contract, and adds a layer of modular roles over the `Timelock`'s actions. This contract defines several roles that have separate responsibility and access over the state of a proposal (queueing, cancellation and execution).
- A `SafeGuardFactory` contract deploys a new `SafeGuard` and a corresponding new `Timelock` contract. Then it sets the timelock address inside the `SafeGuard` contract and registers the new `SafeGuard` into the `Registry`.
- A `Registry` which holds a list of deployed `SafeGuards` with their corresponding version number.

The `SafeGuardFactory` will basically spawn a new `SafeGuard` whenever it is called. A `SafeGuard` is a wrap around `Timelock` operations that adds different and separated roles for each action. Roles are structured through the use of OpenZeppelin's

Update: *In the PR #10, the Tally team has decided to remove the `Registry` contract. The list of deployed safeguards is now stored within the `SafeGuardFactory` contract, in the `safeGuards` enumerable set, along with their version stored in the `safeGuardVersion` mapping.*

Roles

The `SafeGuard` contract defines the following roles:

- CREATOR_ROLE is taken by the `SafeGuardFactory` which deploys this contract and is in charge of calling the `setTimelock` function.
- PROPOSER_ROLE, EXECUTOR_ROLE and CANCELER_ROLE roles are assigned to the values passed as input parameters. These roles are needed to `queue`, `execute` and `cancel` transactions respectively.
- SAFEGUARD_ADMIN_ROLE is set to the `_admin` passed as input parameter and has the power to grant any role to any address.

Update: *The `CREATOR_ROLE` role has been removed as a fix for the issue L02.*

Scope

We audited commit `b2c63a9dfc4090be13320d999e7c6c1d842625d3` of the `safeguard` repository. In scope are the smart contracts in the `contracts` directory. However, the `mocks` directory was deemed as out of scope.

Assumptions

The system is not meant to be upgradeable. The `Registry` address is set in the constructor of the `SafeGuardFactory` and each `SafeGuard` can have the `Timelock` set only once. This means that if a new `Registry` is deployed a new `SafeGuardFactory` must be deployed too. If the `Timelock` implementation changes, the old `SafeGuard`s will become obsolete, and new ones will have to be deployed.

Moreover, the system heavily relies on the implementation of a `Timelock` contract that was deemed out of scope for this audit. The team has not yet finalized the implementation of the

The codebase has been audited by two auditors during the course of one week and here we present our findings.

Critical severity

None.

High severity

[H01] ETH can be locked inside the `Timelock` contract

The `Tally` team originally based their implementations on the ground of the `GovernorBravoDelegate` Compound contract.

During the course of this audit, the `Tally` team discovered a limitation in Compound's governor where ETH sent directly to the `Timelock` is not available for use by governance proposals, and although it is not permanently stuck, requires an elaborate workaround to be retrieved.

This is because the governor implementation requires all the value of a proposal to be attached as `msg.value` by the account that triggers the execution, not using in any way the `Timelock` ETH funds.

The same issue was later identified in the `SafeGuard` implementation and the team is aware of the issue and it is in the process of fixing it.

While fixing the issue, consider using the approach adopted by the OpenZeppelin library for the same issue.

Update: *Fixed in commit* `7337db227edda83533be586135d96ddac4f5bf29`.

[H02] SafeGuardFactory can be freezed

The `Registry` contract is intended to keep track of all the `SafeGuards` that the `SafeGuardFactory` produces. It has the external `register` function which is used for this purpose.

be deployed too.

The `SafeGuardFactory` has the `createSafeGuard` function, in charge of first deploying a new `SafeGuard`, then a new `Timelock` with the address of the `SafeGuard` as `admin`, then setting the `timelock` variable of the `SafeGuard` contract and finally registering the `SafeGuard` in the registry.

The issue is that any call to `createSafeGuard` can be forced to fail by an attacker who can directly register the deterministic address of the new `SafeGuard` prior to its creation. Whenever a contract creates a `new` instance, its nonce is increased, and the address of where the new instance of the contract would be deployed can be determined by the original contract address and its nonce. Therefore, an attacker can precalculate many of the addresses where the new `SafeGuards` will be deployed and register those addresses in the `Registry` by calling the `register` function. This would result in the calls to `createSafeGuard` to revert since the `Registry` already contains the address.

To avoid having external actors calling publicly the `Register` contract, consider restricting the access to the `register` function to accept calls exclusively by the `SafeGuardFactory`.

Update: *Fixed in PR #10. The Tally team has removed the `Registry` contract.*

Medium severity

None.

Low severity

[L01] Commented out code

The `Registry` contract includes a commented out line of code. To improve readability, consider removing it from the codebase.

Update: *Fixed in PR #10 and commit `7fd27df16fc879d990d36a167a0b6e719e578558`.*

[L02] SafeGuard's admin can assign the role of creator to any address

However, by invoking the `grantRole` function of the `AccessControlEnumerable` contract in the OpenZeppelin contract library, an admin can grant this role to any address. This could cause confusion because the creator of the `SafeGuard` can only be the `SafeGuardFactory`.

Throughout the codebase, this role has been used only to restrict users from interacting with the `setTimelock` function of the `SafeGuard` contract. By design, the system ensures that `setTimelock` function can be called only once, from within the `SafeGuardFactory` contract.

Consider removing the `CREATOR_ROLE` role from the `SafeGuard` contract and using the `onlyOwner` modifier in the `setTimelock` function.

Update: *Fixed in PR #10.*

[L03] Incorrect interface definition and implementation

The `ISafeGuard` interface does not define the `queueTransactionWithDescription` function implemented in the `SafeGuard` contract, and at the same time, it defines the __abdicate, __queueSetTimelockPendingAdmin and __executeSetTimelockPendingAdmin functions but they are not implemented.

To improve correctness and consistency in the codebase, consider refactoring the `ISafeGuard` interface to match exactly the `SafeGuard` implementation.

Update: *Fixed in commit* `7fd27df16fc879d990d36a167a0b6e719e578558`.

[L04] Missing docstrings

Some of the contracts and functions in the code base lack documentation. For example, some functions in the `SafeGuard` contract.

Additionally, some docstrings use informal language, such as the one above the `setTimelock` function in the `SafeGuard` contract.

This hinders reviewers' understanding of the code's intention, which is fundamental to correctly assess not only security but also correctness. Additionally, docstrings improve readability and ease

Consider thoroughly documenting all functions (and their parameters) that are part of the contracts' public API. Functions implementing sensitive functionality, even if not public, should be clearly documented as well. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Update: *Partially fixed in PR #10. Proper docstrings have been added to various functions throughout the code base. However, in addition to the current changes, consider making the following changes:*

- *Add `description` as the `@param` in the docstring above `queueTransactionWithDescription` function*
- *Add `@param` in the docstring above the `createSafeGuard` function in `SafeGuardFactory` contract*
- *Add `@return` in docstrings above the functions in `SafeGuardFactory` contract.*

[L05] Useless or repeated code

There are places in the codebase where code is either repeated or not needed. Some examples are:

- Lines 29-32 of the `Registry` contract are useless, because the `_add` function of the `EnumerableSet` contract already performs these checks against the values already being set.
- Lines 62, 67, 73 and 78 of the `SafeGuard` contract are all repeating the same exact operation. Consider encapsulating it into an internal function to avoid duplicating code.
- Lines 62-63 and 67-68 of `SafeGuard` are repeated. Consider encapsulating them into a single internal function.
- The usage of `gasleft` to specify how much gas should be forwarded in the call of the function `executeTransaction` is unnecessary. This is because, at that point of execution, the entire gas left will be used to continue the execution. If this is not for expliciteness, consider removing the `gas` parameter from the call.

Update: *Fixed in PR #10 and commit `7fd27df16fc879d990d36a167a0b6e719e578558`.*

Notes & Additional Information

[N01] Inconsistent style

There are some places in the code base, where differences in style affect the readability, making it more difficult to understand the code. Some examples are:

- The `Registry` contract uses different styles for docstrings in the entire contract.
- The `SafeGuard` contract is emitting an event when `queueTransactionWithDescription` is called but no events are emitted in other functions dealing with transactions.
- In the `SafeGuard` contract, sometimes value is used as named parameter and sometimes _value is used.

Taking into consideration the value a consistent coding style adds to the project's readability, consider enforcing a standard coding style with help of linter tools, such as Solhint.

Update: *Fixed in PR #10 and commit `7fd27df16fc879d990d36a167a0b6e719e578558`.*

[N02] Missing license

The following contracts within the code base are missing an SPDX license identifier.

- The `ISafeGuard` interface.
- The `ITimelock` interface.
- The `SafeGuard` contract.

To silence compiler warnings and increase consistency across the codebase consider adding a license identifier. While doing it consider referring to spdx.dev guidelines.

Update: *Fixed in PR #10 and commit `7fd27df16fc879d990d36a167a0b6e719e578558`.*

[N03] OpenZeppelin Contract's dependency is not pinned

Update: *Fixed in PR #10.*

[N04] Solidity compiler version is not pinned

Throughout the code base, consider pinning the version of the Solidity compiler to its latest stable version. This should help prevent introducing unexpected bugs due to incompatible future releases. To choose a specific version, developers should consider both the compiler's features needed by the project and the list of known bugs associated with each Solidity compiler version.

Update: *Fixed in PR #10.*

[N05] Typo

At various instances throughout the code base, the word `role` is misspelled as `rol`. One such example is in the docstring within the `constructor` of the `SafeGuard` contract.

Consider correcting these typos to improve code readability.

Update: *Partially fixed in PR #10. While the spelling of `role` has been corrected, the comment "set admin role the an defined admin address" should be "set admin role to a defined admin address". Additionally, "execute" is misspelled in the `SafeGuard` contract on line 69, line 82, line 96 and line 110 and "available" is misspelled on line 70, line 83, line 97, line 111. Also, consider replacing informal words such as "gonna" in `SafeGuard` contract with formal alternatives such as "going to".*

[N06] Declare uint as uint256

There are several occurrences in the codebase where variables are declared of `uint` data type instead of `uint256`. For example, the `eta` variable in the `QueueTransactionWithDescription` event of the `SafeGuard` contract.
To favor explicitness, all instances of `uint` should be declared as `uint256`.

Update: *Fixed in PR #10 and commit `7fd27df16fc879d990d36a167a0b6e719e578558`.*

[N07] Unused import

Update: *Fixed in PR #10.*

Conclusions

One high and several other minor vulnerabilities have been found and recommendations and fixes have been suggested.

Related Posts

Beefy Zap Audit

BeefyZapRouter serves as a versatile intermediary designed to execute users' orders through routes...

Security Audits

OpenBrush Contracts Library Security Review

OpenBrush is an open-source smart contract library written in the Rust programming language and the...

Security Audits

Linea Bridge Audit

Linea is a ZK-rollup deployed on top of Ethereum. It is designed to be EVM-compatible and aims to...

Security Audits