EtherCamp's ProjectKudos public code audit

OPENZEPPELIN SECURITY | NOVEMBER 15, 2016

Security Audits

Following our audit of the HackerGold HKG token, we've been asked by the EtherCamp team to review their new ProjectKudos contract.

The audited contract can be found in the master branch of their GitHub repo, specifically, commit 6665fdd4b71088443a74d1ed9fda52c6a8c8b975. Main contract file is ProjectKudos.sol.

Here are our assessment and recommendations, in order of importance:

Severe

We have not found any severe security problems with the code.

Potential problems

Use latest version of Solidity

Current code is written for old versions of solc (0.4.0). With the latest storage overwriting vulnerability found on versions of solc prior to 0.4.4, there's a risk this code can be compiled with vulnerable versions. We recommend changing the solidity version pragma for the latest version (`pragma solidity ^0.4.4;`) to enforce latest compiler version to be used.

EDIT: EtherCamp fixed this problem on these commits.

Timestamp usage

(alias for **block.timestamp**) for contract logic, based on the fact that miners can perform some manipulation. In general, it's better not to rely on timestamps for contract logic. The solutions is to use **block.number** instead, and approximate dates with expected block heights.

Given the nature of the contract, we think the risk of miner manipulation is really low. The potential damage is also limited: miners could only slightly manipulate if kudos can or can't be given near the end date. We recommend the EtherCamp team to consider the potential risk and switch to **block.number** if necessary.

For more info on this topic, see this stack exchange question.

Remove unnecessary code

- ***duringVote function modifier***

  The **duringVote** function modifier is only used once. Consider removing it and adding the checks at the start of **giveKudos** function. Having unneeded extra variables and code increases risk and attack surface for contract's invariants to be broken.

  EDIT: EtherCamp fixed this problem on these commits.
- ***GrantReason enum***

  The GrantReason enum is not needed and adds extra complexity with no benefit. The same clarity can be obtained by having two uint constants called `uint constant GRANT_REASON_FACEBOOK = 0;` and `uint constant GRANT_REASON_TWITTER = 1;`. This would allow removing the grantUintToReason and grantReasonToUint functions.

  EDIT: EtherCamp fixed this problem on these commits.
- ***strToBytes function***

  The function is used to convert strings to byte arrays. It's only used to convert from project string codes into project byte32 codes. Consider removing the function and requiring caller of contract send byte32 codes directly (handle conversions in Dapp UX). Even more, consider using strings directly inside the contract, changing the `projects` mapping from `mapping(bytes32 => ProjectInfo)` to `mapping(string => ProjectInfo)` to improve code readability and reduce complexity.

  EDIT: EtherCamp fixed this problem on these commits.

Warnings

refactoring the code to improve code clarity. Some preconditions are missing in some functions, too. Consider adding preconditions to this, this, this, and this functions.

Usage of magic constants

There are several magic constants in the contract code. Some examples are:

- https://github.com/ether-camp/virtual-accelerator/blob/6665fdd4b71088443a74d1ed9fda52c6a8c8b975/contracts/ProjectKudos.sol#L171
- https://github.com/ether-camp/virtual-accelerator/blob/6665fdd4b71088443a74d1ed9fda52c6a8c8b975/contracts/ProjectKudos.sol#L302
- https://github.com/ether-camp/virtual-accelerator/blob/6665fdd4b71088443a74d1ed9fda52c6a8c8b975/contracts/ProjectKudos.sol#L315

Use of magic constants reduces code readability and makes it harder to understand code intention. We recommend extracting magic constants into contract constants.
EDIT: EtherCamp fixed this problem on these commits.

Additional Information and notes

- No uses of **send**.
- We recommend separating the implementation of ownable contracts into a different file to improve code clarity. You can use OpenZeppelin's Ownable contract for this end.
- There's a typo in the comment in line 217. The line should read "Returns votes given by specified user".

Conclusions

No severe security issues were found. Some changes were recommended to follow best practices and reduce potential attack surface.

EDIT: EtherCamp followed many of our recommendations and fixed the code based on our comments, on these commits.

of HKG. For general information about smart contract security, check out our thoughts here.

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

OpenZeppelin

Company

About us
Jobs
Blog

Contracts Library

Docs

Privacy | Terms of Use