PassThrough audit report

Audited contract

Source code: PassThrough contract
Address: Not deployed yet

Related contracts and assumptions

Estate registry contract

During the audit we assumed that `PassThrough#estateRegistry` pointed to the ZeppelinOS-based upgradable smart contract located at `0x959e104e1a4db6317fa58f8295f586e1a978c297`, with its implementation contract located at `0xe34fcd12115bfe2fb632cd63fe7be5472b54173f`.

We also assumed that most of the time `PassThrough.sol#target` will point to that same contract, and that `operators` will rarely change it in order to fix some mistake (eg: send MANA or LAND to the `PassThrough` contract).

Implementation's source code: EstateRegistry.sol

Land registry contract

During the audit we assumed that `EstateRegistry#registry` pointed to the ZeppelinOS-based upgradable smart contract located at `0xf87e31492faf9a91b02ee0deaad50d51d56d5d4d`, with its implementation contract located at `0xf8550b0511486af9a75c63d85a01c919535f76db`.

Implementation's source code: LANDRegistry.sol

Contracts ownership

We assumed that, once deployed, `PassThrough` contracts will be owned by Decentraland, and their `operators` will be external parties.

Audit results

Payable target functions can't be called with ETH

There's no way to call a `target`'s function with ETH, as `PassThrough`'s fallback function is not payable, and its assembly `call` invocation always set the `value` as 0. In order to fix this the `fallback` function has to be `payable`, and `msg.value` should be passed to `call`.

This may have an impact for users that are changing the contract's `target` to fix some mistake. It's hard to predict if this scenario is feasible or not, but the fixes needed to avoid it are really straight forward and we recommend implementing them.

Possible accidents on Estate registry upgrades

If the `Estates` registry contract is updated with a new function for transferring LAND, such function must be blacklisted in advance in every instance of `PassThrough`. A failure to do so can result in a `Estate` being disassembled.

Our recommendation is to invert the `PassThrough` contract logic, implementing it as a whitelist instead of a blacklist. A whitelist approach has a much smaller impact if a function isn't listed, just temporarily degrading the user experience.

One thing to note is that the whitelist should only be applied when the `target` is the `estateRegistry`. Otherwise, the users won't be able to change the `target` to remedy any incident on their own, as most/all functions they need to call won't be whitelisted.

Missing checks in `constructor`

The `PassThrough` constructor doesn't validate the values of the `_estateRegistry` and `_operator` addresses. While this could lead to accidentally deploying it with wrong values, its impact would be very low, as the contract can be redeployed.

Comments

Multiple unit tests don’t check that the expected results are met

There are multiple tests that call a `PassThrough` contract without checking its result (eg: this test). This can hide possible bugs, as every non-blacklisted call will be redirected to the `target`, but no necessarily to the right function.

Missing functions in end to end tests

The following functions aren’t tested in the end to end tests:

- `safeTransferFrom(address,address,uint256,bytes)`
- `safeTransferManyFrom(address,address,uint256[],bytes)`

No test of failures in the target contract

The test suite doesn’t include cases where the call to the target contract reverts. This is specially important to be sure that error messages are correctly forwarded.

Code coverage couldn’t be computed

As part of the audit process we tried to analyze the code coverage of the contract, with no luck on computing it.

The standard tool for doing so, solidity-coverage, failed to work with this contract due to its parser being outdated. After manually changing the source code to workaround this situation, other errors arose that couldn’t be solved.

Despite not being able to quantify the coverage and with exception of the comments stated above, the contract is very well tested.