SOLIDIFIED

Audit Report for Props Protocol - March 26, 2021

Summary

Audit Report prepared by Solidified covering the Props Protocol smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on 22 March 2021, and the results are presented here.

Update: Fixes were reviewed on 26 March 2021

Audited Files

The source code has been supplied in the form GitHub repositories:

https://github.com/propsproject/props-protocol

Commit number: `ce4b8e872fd1975a664128e71c106c2c6a5d59ac`

https://github.com/propsproject/props-token-distribution/tree/

Commit number: `b54af5fdd375536f9abe5870cb8b9899e24a6a43`

The scope of the audit was limited to the following files:

```
contracts
├── AppProxyFactoryL1.sol
├── AppProxyFactoryL2.sol
├── IAppProxyFactoryBridged.sol
├── IPropsProtocol.sol
├── PropsProtocol.sol
├── bridge
│   ├── AppProxyFactoryBridgeL1.sol
│   ├── AppProxyFactoryBridgeL2.sol
│   ├── GovernanceBridgeL1.sol
│   └── GovernanceBridgeL2.sol
├── governance
│   ├── GovernorAlpha.sol
│   └── Timelock.sol
├── staking
│   ├── IStaking.sol
│   └── Staking.sol
├── tokens
│   ├── app-points
```

```
│   │   ├── AppPointsCommon.sol
│   │   ├── AppPointsL1.sol
│   │   ├── AppPointsL2.sol
│   │   └── IAppPoints.sol
│   └── props
│       ├── IPropsTokenL1.sol
│       ├── IPropsTokenL2.sol
│       ├── IRPropsToken.sol
│       ├── ISPropsToken.sol
│       ├── PropsTokenL2.sol
│       ├── RPropsToken.sol
│       └── SPropsToken.sol
└── utils
    ├── MetaTransactionProvider.sol
    └── MinimalProxyFactory.sol


props-token-distribution/contracts
└── token
    ├── ERC865Token.sol
    ├── IERC865.sol
    ├── PropsTimeBasedTransfers.sol
    ├── PropsToken.sol
    ├── PropsTokenDetailed.sol
    └── PropsTokenLib.sol
```

Intended Behavior

The smart contracts implement a loyalty-points staking protocol. Users can stake their tokens to earn both, Props tokens and reward tokens for different apps that integrate with the protocol. App reward distribution is based on a user-specified weight per app.

Code Complexity and Test Coverage

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does equate to a higher risk. Certain bugs are more easily detected in unit testing than a security audit and vice versa. It is, therefore, more likely that undetected issues remain if the test coverage is low or non-existent.

Criteria	Status	Comment
Code complexity	Medium-High	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	High	-

SOLIDIFIED

Audit Report for Props Protocol - March 26, 2021

Test coverage report:

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	98.36	86.49	94.23	98.4	
AppProxyFactoryL1.sol	100	100	100	100	
AppProxyFactoryL2.sol	100	100	100	100	
IAppProxyFactoryBridged.sol	100	100	100	100	
IPropsProtocol.sol	100	100	100	100	
PropsProtocol.sol	98.01	85.71	92.31	98.04	302,313,324
contracts/staking/	100	86.36	100	100	
IStaking.sol	100	100	100	100	
Staking.sol	100	86.36	100	100	
contracts/tokens/app-points/	100	83.33	100	100	
AppPointsCommon.sol	100	70	100	100	
AppPointsL1.sol	100	100	100	100	
AppPointsL2.sol	100	100	100	100	
IAppPoints.sol	100	100	100	100	
contracts/tokens/props/	100	87.5	100	100	
IPropsTokenL1.sol	100	100	100	100	
IPropsTokenL2.sol	100	100	100	100	
IRPropsToken.sol	100	100	100	100	
ISPropsToken.sol	100	100	100	100	
RPropsToken.sol	100	87.5	100	100	
contracts/utils/	96	58.33	100	96.55	
MetaTransactionProvider.sol	96	58.33	100	96.55	179
All files	98.84	83.58	97.25	98.89	

SOLIDIFIED

Audit Report for Props Protocol - March 26, 2021

Issues Found

Solidified found that the Props protocol contracts contain no critical issue, 1 major issue, 4 minor issues, in addition to 4 informational notes.

We recommend all issues are amended, while the notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	Permissioned roles concentrate a lot of privileges	Major	Acknowledged
2	Staking.sol: Divisions before multiplication lead to decreased precision.	Minor	Resolved
3	PropsProtocol.sol: stakeAsDelegate() will always fail for unstaking	Minor	Resolved
4	PropsProtocol.sol: transferGuardianship() and transferControl() lag zero checks	Minor	Resolved
5	Timelock.sol: Pausing of contract cannot be undone	Minor	Resolved
6	Staking.sol: Not all parameter changes are covered by events	Note	-
7	SPropsToken.sol: Domain separator could be precalculated to safe gas	Note	-
8	SPropsToken.sol: approve() should revert if transfers are not allowed	Note	-

Critical Issues

No critical issues have been found.

Major Issues

1. Permissioned roles concentrate a lot of privileges

The Props protocol implementation assigns key privileges to certain roles, including `controller`, `admin`, and `guardian`. This introduces a certain degree of centralization, which may be a security risk in the case of key compromise.

The Props protocol team acknowledges this fact and provides clear documentation of the privileged assigns to these roles:
https://github.com/propsproject/props-protocol/blob/master/docs/RolesAndPermissions.md
The team also states the state their intention to employ a multisig wallet for these addresses.

Recommendation

Use a multisig wallet and have clear protocols in place for key management, including protocols for key permission assignments, usage, revocation, and a key compromise protocol. In addition, consider providing a road map to future decentralization of the protocol.

Minor Issues

2. Staking.sol: Divisions before multiplication lead to decreased precision.

There are two places in this contract where integer divisions are performed before multiplications in calculations. Reversing these operations would lead to improved precision:

Line 54:
`rewardsDuration = uint256(1e18).div(_dailyRewardEmission).mul(1 days);`

Line 334:
`rewardsDuration = uint256(1e18).div(_dailyRewardEmission).mul(1 days);`

Recommendation

Reverse the order of the operations.

Update: Resolved

3. PropsProtocol.sol: stakeAsDelegate() will always fail for unstaking

The `stakeAsDelegate()` will always revert in `_stake()` in the case of unstaking, even though the comments indicate unstaking as a delegate should be possible. This is due to the check in line 720:

`require(_msgSender() == _from, "Unauthorized");`

Recommendation

Consider allowing unstaking as a delegate or change the comments.

Update: Desired behavior. Resolved by updating documentation.

4. PropsProtocol.sol: transferGuardianship() and transferControl() lag zero checks

The `transferGuardianship()` and `transferControl()` functions allow `address(0)` as parameters. However, renouncing these roles in the current implementation would break the protocol.

Recommendation

Consider adding a zero check to disallow `address(0)` as a parameter.

Update: Resolved

5.Timelock.sol: Pausing of contract cannot be undone

The `TimeLock` contract can only be stopped by the pause function. There is no `unpause()` equivalent to enable the contract again. This means that once paused the time locking functionality will be disabled forever.

Recommendation

Implement an `unpause()` function or change the function and error messages to reflect halting rather than pausing.

Update: Desired behavior. Resolved by renaming functionality to halt.

Informative Notes

6. Staking.sol: Not all parameter changes are covered by events

In some cases, key protocol parameters can be changed without an event being emitted. It is best practice to emit events in such cases consistently to allow off-chain protocol monitoring. Examples of these are functions `changeDailyRewardEmission()` and `changeRewardsDistribution()`.

Recommendation

Consider adding even types and emit these events when appropriate.

7. SPropsToken.sol: Domain separator could be precalculated to safe gas

The `domainSeparator` used in `delegateBySig()` could be precalculated in the initializer to reduce gas costs.

Recommendation

Pre-calculate the value in the initializer.

8. SPropsToken.sol: approve() should revert if transfers are not allowed

The `approve()` function is implemented despite transfers not being allowed. This is inconsistent with expected behavior.

Recommendation

Change the implementation to simply revert.

Audit Report for Props Protocol - March 26, 2021

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of Props Protocol or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.