SOLIDIFIED

Audit Report for Pawtocol on November 26th, 2019.

Summary

Audit Report prepared by Solidified for Pawtocol covering the UPI token smart contracts (and inherited dependencies).

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The debrief took place on November 26th, 2019, and the final results are presented here.

Audited Files

UPIToken.sol (and inherited contracts from OpenZeppelin).

Notes

The audit was performed on commit `19e42b9eba9af4c75b0af46649efa0d8fb95c4eb` of repository https://github.com/pawtocol/Token. The audit was based on the Solidity compiler `0.5.11`. Follow up performed on commit `2e54bd4937c289a4fb283f251e2513b774da7317`. The audited contract matches the one deployed at address `0x70D2b7C19352bB76e4409858FF5746e500f2B67c` (Ethereum mainnet).

Intended Behavior

The contracts implement an ERC20 compliant token with fixed supply leveraging OpenZeppelin's ERC20 implementation.

SOLIDIFIED

Audit Report for Pawtocol on November 26th, 2019.

Issues Found

Critical

No critical issues found.

Major

No Major issues found.

Minor

No minor issues found.

Notes

1. UPIToken's variables `name`, `symbol` and `decimals` overshadow variables in OpenZeppelin's `ERC20Detailed`

The private variables declared on `UPIToken` with the goal of initializing the contract in fact overshadow the ones in OpenZeppelin's implementation. This has no adverse effects to the contracts, as these variables are only not changeable later on, but three additional storage slots are used, increasing the cost of deployment.

Amended [02.12.2019]
The issue was fixed and is no longer present in commit `2e54bd4937c289a4fb283f251e2513b774da7317`.

2. Consider optimizing the minting function

`UPIToken` mints all the available supply to the deployer of the contract, and then renounces the minting role, making minting once again impossible. Consider removing `ERC20Mintable` from the implementation, and minting on UPIToken's constructor using ERC20's internal `_mint()` function instead. This will considerably reduce the bytecode size with both `ERC20Mintable` and MinterRole removed. An added benefit is a cleaner ABI, that does not include functions that are not supposed to be used after deployment

Amended [02.12.2019]
The issue was fixed and is no longer present in commit `2e54bd4937c289a4fb283f251e2513b774da7317`.

3. The Fallback function doesn't guarantee that the contract won't receive ETH

The contract includes a fallback function that reverts by default. Keep in mind that although this prevents that ether is sent to the contract through regular transactions, the contract can still receive ether from a contract being self destructed (this will not trigger the fallback function). The current behavior is exactly the same as if the fallback function was not included, except that it will show a revert reason.

Amended [02.12.2019]
The issue was fixed and is no longer present in commit `2e54bd4937c289a4fb283f251e2513b774da7317`.

4. ERC20 does not need to be imported from UPIToken

ERC20 is inherited by both ERC20Mintable and ERC20Burnable, so there is no need to import it in UPIToken.

Amended [02.12.2019]
The issue was fixed and is no longer present in commit `2e54bd4937c289a4fb283f251e2513b774da7317`.

5. Consider using the keyword `ether` to improve readability of `INITIAL_SUPPLY`

The line could be `uint256 private INITIAL_SUPPLY = 1000000000 ether;`, making it easier to read and to reason about. The INITIAL_SUPPLY variable is also written to state, but used only once, passing the value directly in the constructor will provide minor gas savings (One less storage write and one load).

Amended [02.12.2019]
The issue was fixed and is no longer present in commit `2e54bd4937c289a4fb283f251e2513b774da7317`.

Audit Report for Pawtocol on November 26th, 2019.

Closing Summary

The contracts contain no security issues that could affect their behavior. The audit revealed some opportunities for improvement, here reported as notes. The notes present no security risk to the smart contracts, and mainly refer to readability improvements and minor gas savings, and though can be fixed at Pawtocol's discretion.

The contracts were tested against known vulnerabilities such as overflows, reentrancy and others, as well as for ERC20 compliance and possible optimizations.

Follow up [02.12.2019]
All recommendations were fixed and are no longer present in commit `2e54bd4937c289a4fb283f251e2513b774da7317`. The contract has been deployed at `0x70D2b7C19352bB76e4409858FF5746e500f2B67c` (Ethereum mainnet)

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of the Pawtocol platform 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.