# Contract Audit

December, 2017

New Alchemy





 

# Introduction

During December of 2017, Destiny.Games engaged New Alchemy to audit the smart contracts for their token sale for The Abyss. This engagement was technical in nature and focused on identifying flaws in code or the design of the contracts with a security impact.

Destiny.Games provided New Alchemy with read access to their code repository. After receiving New Alchemy's report, Destiny.Games made changes to the code in order to address concerns detailed in the initial audit. Destiny.Games then provided New Alchemy with the updated code to audit to ensure that the previously identified issues were remedied and that no new issues were introduced.

# Files Audited

## theAbyss.sol

The code is at the following GitHub repository:

https://github.com/theabyssportal/SmartContract/

New Alchemy initially evaluated the commit hash `b03a3227c25dc19b4f173a64049b8fe13c8ccae2`.

New Alchemy then evaluated the commit hash `3e989d2dc6cacec1bd0aca1ffcb2bed6946741b4`, which included fixes to the issues identified in the initial audit along with some new functionality.

# Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bugfree status. The audit documentation is for discussion purposes only.

# Executive Summary

A majority of the code was standard and copied from widely-used and reviewed contracts and, as a result, much of the time spent auditing pertained to the `TheAbyssCrowdsale` contract, which was short and uncomplicated. It correctly implemented widely-used and reviewed contracts for safe mathematical operations. The audit identified no security vulnerabilities.

## General Discussion

The issues identified were minor in nature, and do not affect the security of the contract. The code specifies Solidity version 0.4.18, which has only recently had a newer version of 0.4.19 released. As a result, Destiny.Games should consider updating the pragma statements to require the latest version of Solidity.

Additionally, the code implements and uses a SafeMath contract, which defines functions for safe math operations that will throw errors in the cases of integer overflow or underflows.

The simplicity of the audited contracts contributed greatly to their security. The minimalist approach in choosing which pieces of functionality to implement meant there was very little attack surface available.

There is no implementation of the soft cap of 5000 ETH that is mentioned on the website describing the crowdsale. Destiny.Games plans to manually refund all of the contributors in the case of an unsuccessful crowdsale, but this is not enforced anywhere in the contract itself. It should be noted that outside of this, Destiny.Games provided themselves with relatively low levels of permissions over the contracts. Their only privileges allow them to withdraw the funds raised by the crowdsale and pause the crowdsale to prevent users from contributing temporarily. Often times smart contracts will include functionality that allows for the owner to update the business logic behind the contract in the case of a discovered vulnerability. This has the trade-off of also allowing the contract owners to change the behavior of the contract against the wishes of the contract's contributors. Destiny.Games has chosen to not give themselves this power to replace business functionality. Instead, they chose to create functionality to temporarily pause the contract in the case of a discovered vulnerability.

## Minor Issues

### *(FIXED)* `assert` uses remaining gas

This modifier uses an assert statement which checks the transaction's gas price, and will fail if the gas price is too high. This limitation is not mentioned in the The Abyss documentation. Due to this, users may accidentally lose significant amounts of ETH; this is especially concerning because this assert only triggers when the gas price is set to a particularly high value.

## Line by line comments

### *(FIXED)* Line 99 - line 107 : `preSaleStartTime`, `preSaleEndTime`, `saleStartTime`, `saleEndTime`, `bonusWindow1EndTime`, `bonusWindow2EndTime`, and `bonusWindow3EndTime`.

**(From the code at commit hash `b03a3227c25dc19b4f173a64049b8fe13c8ccae2`)**

The values for these variables are still set to zero. It is intended that these values are set as the contract is launched. However, there exists a possibility for the user deploying the contract to make an error while entering the values of these variables. Destiny.Games should be aware of the fact that, if these dates are entered incorrectly, then they will be unable to change the values after the contract is launched. Destiny.Games can consider adding functionality to change these dates after contract launch but before the presale begins, but this functionality should not be able to be called after the presale has begun, as this could lead to inconsistent behaviors that may not be welcome by contributors.