PUBLIC

Express Audit

of VU TOKEN Smart Contracts

April 24, 2018

Produced for

by

CHAINSECURITY

Table Of Content

Foreword

We first and foremost thank IMMERSIVE ENTERTAINMENT for giving us the opportunity to audit their smart contracts. This documents outlines our methodology, limitations, and results.

- ChainSecurity

Executive Summary

CHAINSECURITY has reviewed the IMMERSIVE ENTERTAINMENT smart contracts and found a critical and multiple other issues. The most important issue is that an unlimited number of tokens can be bought during the crowdsale. Any tokens above the cap can later not be withdrawn. The other important issue is the fact that not all tokens of the presale can be purchased. When trying to purchase the final tokens, the transaction will fail. These issues have to be fixed before deployment. Furthermore, CHAINSECURITY has included minor issues, recommendations and open questions.

Token Name & Symbol	VU Token, VU
Decimals	18 decimals
Phases	Presale, Crowdsale
Refund	None
Tokens issued	1,000,000,000
Minimum contribution	None
Maximum contribution	None
Token Generation	Pre-minted
Vesting	None
Pausable	Available
KYC	Whitelist
Owner Rewards	20% of VUs

Table 1: Facts about the VU token and the token sale.

System Overview

VU Tokens are being sold to those interested in participating in an exciting VR entertainment game and platform. Virtual Universe (VU) is an epic, story-driven open world game in LivingVR powered by artificial intelligence (AI), virtual reality (VR), and blockchain technologies.

In the following we describe the VU Token (VU) and its corresponding Token Sale. Table 1 gives the general overview.

Token Sale Overview

The sale consists of two parts: presale and crowdsale. The presale is internally split into three phases. The difference between the phases is the token price and the number of available tokens. The prices and caps are as described in the supplied excel sheet. Users can only withdraw their tokens after a fixed delivery date.

Token Overview

The VU Token is a standard ERC20 token and has all the required functionality.

Extra Token Features

Mass Transfer Mass Transfer reduces the gas costs of making many transfers by batching many `transfer` operations into one transaction.

Burnable Tokens can be burnt and thereby permanently destroyed. This feature is available to all token owners.

Audit Overview

Scope of the Audit

The scope of the audit is limited to the following source code files. All of these source code files were received on April 24, 2018:

File	SHA-256 checksum
crowdsale/BaseCrowdsale.sol	e6e7eb685e266002eee76515f119b8bad15e851a9abd2dfe73bf0943a42d3bca
crowdsale/ICOCrowdsale.sol	e58c20d67ebc5a18a1b02f8c1e8044a068fe1ab922ad3a05c81a1d5058677d8d
crowdsale/PresaleCrowdsale.sol	af3e0ddc131f7b199d787dde7b5ca435c6dffc8902a8fa53473ff80756df6b5d
crowdsale/Whitelist.sol	9ab9758e4d1b1ad87f2cf123bc7b5db134d7ab4d2033a8f38052a6869661eab4
token/VUToken.sol	648ce4419df3dd0d9c8afad2dd0eab7636d32a78f5444b4cff166aa323cf98a7

Depth of Audit

The scope of the security audit conducted by ChainSecurity was restricted to:

- Scan the contracts listed above for generic security issues using automated systems and manually inspect the results.
- Manual audit of the contracts listed above for security issues.

Terminology

For the purpose of this audit, we adopt the following terminology. For security vulnerabilities, we specify the *likelihood*, *impact* and *severity* (inspired by the OWASP risk rating methodology[1]).

Likelihood represents the likelihood of a security vulnerability to be encountered or exploited in the wild.

Impact specifies the technical and business related consequences of an exploit.

Severity is derived based on the likelihood and the impact calculated previously.

We categorize the findings into 4 distinct categories, depending on their severities:

- L - Low: can be considered as less important
- M - Medium: should be fixed
- H - High: we strongly suggest to fix it before release
- C - Critical: needs to be fixed before release

These severities are derived from the likelihood and the impact using the following table, following a standard approach in risk assessment.

	IMPACT		
LIKELIHOOD	High	Medium	Low
High	C	H	M
Medium	H	M	L
Low	M	L	L

[1] https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

During the audit concerns might arise or tools might flag certain security issues. If our careful inspection reveals no security impact, we label it as ✓ No Issue. Finally, if during the course of the audit process, an issue has been addressed technically, we label it as ✓ Fixed, while if it has been addressed otherwise we label it as ✓ Addressed.

Findings that are labelled as either ✓ Fixed or ✓ Addressed are resolved and therefore pose no security threat. Their severity is still listed, but just to give the reader a quick overview what kind of issues were found during the audit.

Limitations

Security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a secure smart contract. However, auditing allows to discover vulnerabilities that were overlooked during development and areas where additional security measures are necessary.

In most cases, applications are either fully protected against a certain type of attack, or they lack protection against it completely. Some of the issues may affect the entire smart contract application, while some lack protection only in certain areas. We therefore carry out a source code review trying to determine all locations that need to be fixed. Within the customer-determined timeframe, ChainSecurity has performed auditing in order to discover as many vulnerabilities as possible.

Details of the Findings

In this section we detail our findings, including both positive and negative findings.

Security Issues

In this section, we discuss our investigation into security issues. Therefore, we highlight whenever we found specific issues but also mention what vulnerability classes do not appear.

Hard Cap not enforced C ✓ Fixed

The hard cap for the crowdsale is not enforced. This is due to the fact that the `BaseCrowdsale` inherits from `AllowanceCrowdsale`, but overwrites a central part of it functionality. In the snippet below we see the `_processPurchase` function.

```
    function _processPurchase(address _beneficiary, uint _tokenAmount)
    internal
    {
        tokensSold = tokensSold.add(_tokenAmount);
        PostDeliveryCrowdsale._processPurchase(_beneficiary, _tokenAmount);
    }
```

BaseCrowdsale.sol

It does not transfer tokens directly, but instead only registers that tokens have been purchased. Hence, the allowance and the `remainingTokens()` function is never updated. Therefore, more tokens than in the hard cap can be bought. However, these tokens cannot be withdrawn later.

For a demonstration, see the test case on page 11 for an example where 480,000,000 VUs are bought even though the cap is 450,000,000 VUs.

Likelihood: High
Impact: High

Fixed: there is now a `limit` variable in the `BaseCrowdsale` contract which bounds the number of token sold at each phase.

Final token sale of presale fails M

The final token sale will likely fail in the presale. If x tokens are left and a purchase for more than x tokens is made, the purchase will fail. Note, that for the presale due to the design of the token prices and the phase caps, there will always be a residual amount of tokens that cannot be purchased. As a consequence of this, a sellout of the presale is impossible.

For the presale, the line highlighted below will fail, as it tries to enter a non-existent phase.

```
            uint tokens = _getPhaseUpperLimit(_phase).sub(_tokensSold);

            PhaseId nextPhase = PhaseId(uint(_phase) + 1);
            tokensBought = _calcTokenAmount(nextPhase, _weiAmount.sub(tokens.div(rate)), _tokensSold.add(tokens)
                );

```

PresaleCrowdsale.sol

Likelihood: Medium
Impact: Medium

In case of a sellout tokens cannot be withdrawn early L

In case the crowdsale sells out, VU Tokens will not be available before the timeout. This is by design, as the `BaseCrowdsale` defines a `deliveryTime` when tokens can be withdrawn. Hence, even though the sale is finished, the tokens cannot be withdrawn.

Likelihood: Medium
Impact: Low

Two token withdrawal necessary L

Immersive Entertainment needs to inform its customers that participate in the Token Sale that the tokens are not received immediately, but have to be withdrawn. Furthermore, Immersive Entertainment has to inform its customers that two separate token withdrawals are necessary for customers that participated in presale and crowdsale. Otherwise, unclaimed tokens might be forgotten.

Likelihood: Low
Impact: Medium

Update time constants L

The migration scripts contain a number of time constants which control start, end and delivery for both the presale and the crowdsale. These constants seem to be incorrect and do not match the values from the excel sheet. Here is an example for the presale:

```
    if (network === "main") {
        startTime = (new Date(’March␣29,␣2018␣00:00:00’)).getTime() / 1000;
        endTime = (new Date(’April␣30,␣2018␣00:00:00’)).getTime() / 1000;
```

5_deploy_presale3.js

Likelihood: Medium
Impact: Low

Recommendations / Suggestions

Minor gas saving possible

In the function `_calcTokenAmount` the result of `_getPhaseUpperLimit(_phase)` can be cached by saving it inside a local variable. Then it can also be used in line 183 instead of the expression `_tokensSold.add(tokens)`.

Minor recommendations

- The import of `BurnableToken` in `BaseCrowdsale.sol` is unnecessary.
- Consider using the `emit` keyword, available in the latest version of Solidity, as it makes the firing of events more distinguishable from function calls.
- Since the phase parameters are hardcoded in `PresaleCrowdsale.sol`, IMMERSIVE ENTERTAINMENT might as well hardcode the `phases` array as well, and give it a well-defined size.
- In a number of historical cases, token buyers have (presumably accidentally) send ERC20 tokens to token sale contracts. These sent tokens are then locked within the receiving contract if it contains no functionality to forward/spend them. IMMERSIVE ENTERTAINMENT could add such a functionality in order to avoid these locked tokens.

Open Questions

- Should unsold presale tokens be available in the crowdsale? Or burnt?
- Should unsold crowdsale tokens be automatically burnt?
- Recent commits modified some parameters, making inconsistent use of constants (compare for instance the 10^{15} in `9_deploy_test.js` and 10^{18} in `ICOCrowdsale.sol`. Is there a reason for this change?

Disclaimer

UPON REQUEST BY IMMERSIVE ENTERTAINMENT, CHAINSECURITY LTD. AGREES MAKING THIS AUDIT REPORT PUBLIC. THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND, AND CHAINSECURITY LTD. DISCLAIMS ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT. COPYRIGHT OF THIS REPORT REMAINS WITH CHAINSECURITY LTD..

Exploit for breaking the cap

```
context("ICOTestable", async () => {
    it(’buying␣more␣tokens␣than␣the␣cap’, async () => {
        assert.isTrue((await token.balanceOf(participant)).isZero());
        let initialWalletBalance = await web3.eth.getBalance(wallet);

        let ico = await ICOCrowdsaleTestable.deployed();

        await ico.buyTokens(participant, {from: participant, value:8000000000000000000000000});

        let balance = await ico.balances(participant);

        assert.isTrue(balance.eq(8000000000000000000000000 * 6000));
    })
});
```

Test case to buy more tokens than the cap