

# Security Audit

## of SMART VALOR's Smart Contracts

October 18, 2018

Produced for



by



# Table Of Content

Foreword . . . . .	1
Executive Summary . . . . .	1
Audit Overview . . . . .	2
1. Scope of the Audit . . . . .	2
2. Depth of Audit . . . . .	2
3. Terminology . . . . .	2
Limitations . . . . .	4
System Overview . . . . .	5
1. Token Sale Overview . . . . .	5
2. Token Overview . . . . .	5
3. Extra Token Features . . . . .	5
Best Practices in SMART VALOR's project . . . . .	6
1. Hard Requirements . . . . .	6
2. Soft Requirements . . . . .	6
Security Issues . . . . .	7
1. Locked tokens   . . . . .	7
2. Gas-dependent Reentrancy  . . . . .	7
3. Reentrancy with constant gas  . . . . .	7
4. Reentrant method call  . . . . .	7
5. Division influences Transfer Amount  . . . . .	7
6. Division Before Multiplication  . . . . .	7
7. Locked Ether  . . . . .	7
8. Transactions May Affect Ether Amount  . . . . .	7
9. Transactions May Affect Ether Receiver  . . . . .	7
10. Transactions May Affect Ether Transfer  . . . . .	7

11. Unhandled Exception	✓ No Issue	8
12. Unrestricted Selfdestruct	✓ No Issue	8
13. Unrestricted ether flow	✓ No Issue	8
14. Unrestricted write to storage	✓ No Issue	8
15. Unsafe Call to Untrusted Contract	✓ No Issue	8
16. Unsafe Dependence On Block Information	✓ No Issue	8
17. Unsafe Dependence On Block Gas	✓ No Issue	8
18. Use Of Origin	✓ No Issue	8
19. Delegatecall dependent on user input	✓ No Issue	8
Trust Issues		9
1. Unsold tokens will be burned manually	 ✓ Acknowledged	9
Design Issues		10
1. Outdated compiler version	 ✓ Fixed	10
2. Unused write to storage	✓ No Issue	10
Recommendations / Suggestions		11
Disclaimer		12

# Foreword

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

– ChainSecurity

## Executive Summary

The SMART VALOR smart contracts have been analyzed under different aspects, with a variety of tools for automated security analysis of Ethereum smart contracts and expert manual review.

Overall, we found that SMART VALOR employs very good coding practices and has clean, well-documented code. No critical vulnerabilities were identified during the audit, but nonetheless minor issues were raised which were successfully mitigated by SMART VALOR.

# 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 October 16, 2018:

File	SHA-256 checksum
ValorToken.sol	70eac1cadecef5fea05ea0f592fe8066457689e6dab93168af6e5b7e04f10315

## 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<sup>1</sup>).

**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:

-  Low: can be considered as less important
-  Medium: should be fixed
-  High: we strongly suggest to fix it before release
-  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.

LIKELIHOOD	IMPACT		
	High	Medium	Low
High			
Medium			
Low			

<sup>1</sup>[https://www.owasp.org/index.php/OWASP\\_Risk\\_Rating\\_Methodology](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**. 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 by improving documentation or further specification, we label it as **✓ Addressed**. Finally, if an issue is meant to be fixed in the future without immediate changes to the code, we label it as **✓ Acknowledged**.

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.

# System Overview

Token Name & Symbol	VALOR TOKEN, VALOR
Decimals	18 decimals
Phases	No
Tokens issued	100 Million
Token Type	ERC20
Pre-Minted	Yes
Burnable	Yes
Vesting	No
Pausable	No

Table 1: Facts about the VALOR token and the Token Sale.

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

## Token Sale Overview

SMART VALOR will pre-mint all 100 million VALOR tokens and these will be distributed to three different wallets at the token generation event. Funds at each of the wallet addresses have a dedicated future use. Below the initial recipients are listed with a corresponding percentage of the total supply to be received:

- Future Development Fund (FDF), 26%
- Company multisig cold wallet (CMCW), 55%
- Employees pool (EP), 19%

VALOR TOKENS will be sent to investors separately by SMART VALOR after the token generation event. The initial minting and the transfer of these funds can be observed by listening to corresponding `Transfer` events from `address(0)`.

## Token Overview

VALOR TOKENS are ERC20 standard based tokens and fully implement the interface with all corresponding functionality.

## Extra Token Features

**Burnable** Tokens can be burnt, a feature which, according to the white paper, will be used by SMART VALOR to burn unsold tokens. This feature is also available to all token owners, offering a way to burn their own tokens.



# Best Practices in SMART VALOR's project

Projects of good quality follow best practices. In doing so, they make audits more meaningful, by allowing efforts to be focused on subtle and project-specific issues rather than the fulfillment of general guidelines.

Avoiding code duplication is a good example of a good engineering practice which increases the potential of any security audit.

We now list a few points that should be enforced in any good project that aims to be deployed on the Ethereum blockchain. The corresponding box is ticked when SMART VALOR's project fitted the criterion when the audit started.

## Hard Requirements

These requirements ensure that the SMART VALOR's project can be audited by CHAINSECURITY.

- ☒ The code is provided as a Git repository to allow the review of future code changes.
- ☒ Code duplication is minimal, or justified and documented.
- ☒ Libraries are properly referred to as package dependencies, including the specific version(s) that are compatible with SMART VALOR's project. No library file is mixed with SMART VALOR's own files.
- ☒ The code compiles with the latest Solidity compiler version. If SMART VALOR uses an older version, the reasons are documented.
- ☒ There are no compiler warnings, or warnings are documented.

## Soft Requirements

Although these requirements are not as important as the previous ones, they still help to make the audit more valuable to SMART VALOR.

- ☒ There are migration scripts.
- ☒ There are tests.
- ☒ The tests are related to the migration scripts and a clear separation is made between the two.
- ☒ The tests are easy to run for CHAINSECURITY, using the documentation provided by SMART VALOR.
- ☒ The test coverage is available or can be obtained easily.
- ☒ The output of the build process (including possible flattened files) is not committed to the Git repository.
- ☒ The project only contains audit-related files, or, if not possible, a meaningful separation is made between modules that have to be audited and modules that CHAINSECURITY should assume correct and out of scope.
- ☒ There is no dead code.
- ☒ The code is well documented.
- ☒ The high-level specification is thorough and allow a quick understanding of the project without looking at the code.
- ☒ Both the code documentation and the high-level specification are up to date with respect to the code version CHAINSECURITY audits.
- ☒ There are no getter functions for public variables, or the reason why these getters are in the code is given.
- ☒ Function are grouped together according either to the Solidity guidelines<sup>2</sup>, or to their functionality.

---

<sup>2</sup><https://solidity.readthedocs.io/en/latest/style-guide.html#order-of-functions>

# Security Issues

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

## Locked tokens

VALOR TOKENS can be accidentally transferred to the VALOR TOKEN contract address. Numerous historic cases have shown that accidental ERC20 transfers to token contracts occur frequently. Currently, such ERC20 tokens would be locked as there is no functionality to claim or handle them in any way.

**Likelihood:** Medium

**Impact:** Low

**Acknowledged:** SMART VALOR is aware that this is an issue in the Ethereum ecosystem, but will not make changes to the ValoToken contract for several reasons. Firstly, although several standards have been proposed to mitigate this, none of them have reached the proper consensus and adoption to be considered a solid standard. Secondly, implementing such a mechanism would involve big changes to the codebase and contradict the "do less do well" design philosophy. Lastly, solving this would require another trusted role to be introduced.

## Gas-dependent Reentrancy

Calls into external contracts that receive all remaining gas must not be followed by writes to storage.

## Reentrancy with constant gas

Ether transfers (such as send/transfer) must not be followed by writes to storage.

## Reentrant method call

Method calls must not be followed by state changes.

## Division influences Transfer Amount

The use of division to calculate the amount of transferred ether may be incorrect due to integer rounding.

## Division Before Multiplication

The use of division before multiplication may result in incorrect final results due to integer rounding.

## Locked Ether

Contracts that may receive ether must also allow users to extract the deposited ether from the contract.

## Transactions May Affect Ether Amount

The amount of ether transferred must not be influenced by other transactions.

## Transactions May Affect Ether Receiver

The receiver of ether transfers must not be influenced by other transactions.

## Transactions May Affect Ether Transfer

Ether transfers whose execution can be manipulated by other transactions must be inspected for unintended behavior.

#### **Unhandled Exception** ✓ No Issue

The return value of statements that may return error values must be explicitly checked.

#### **Unrestricted Selfdestruct** ✓ No Issue

The execution of selfdestruct statements must be restricted to an authorized set of users.

#### **Unrestricted ether flow** ✓ No Issue

The execution of ether flows should be restricted to an authorized set of users.

#### **Unrestricted write to storage** ✓ No Issue

Contract fields that can be modified by any user must be inspected.

#### **Unsafe Call to Untrusted Contract** ✓ No Issue

The target of a call instruction can be manipulated by an attacker.

#### **Unsafe Dependence On Block Information** ✓ No Issue

Security-sensitive operations must not depend on block information.

#### **Unsafe Dependence On Block Gas** ✓ No Issue

Security-sensitive operations must not depend on gas-related information.

#### **Use Of Origin** ✓ No Issue

The origin statement must not be used.

#### **Delegatecall dependent on user input** ✓ No Issue

The target and arguments provided to delegatecall must be sanitized.

# Trust Issues

The issues described in this section are not security issues but describe functionality which is not fixed inside the smart contract and hence requires additional trust into SMART VALOR, including in SMART VALOR's ability to deal with such powers appropriately.

**Unsold tokens will be burned manually**



✓ Acknowledged

As described in the specification all unsold tokens will be burned by SMART VALOR. However, as the burning of the tokens is not automatically triggered upon a certain event like is often the case with e.g. ICO finalization, the users need to trust SMART VALOR to fulfill its promise to indeed do so. Users and investors can easily audit if this happens by monitoring for the emission of corresponding Burn events.

**Acknowledged:** SMART VALOR states that the rationale behind this design is that the tokensale is not handled only through a smart contract. SMART VALOR further suggests that a strategy to automatically burn tokens should rely upon a trusted oracle which would be equivalent to directly trusting the company that reports the number of sold tokens.

# Design Issues

The points listed here are general recommendations about the design and style of SMART VALOR's project. They highlight possible ways for SMART VALOR to further improve the code.

## Outdated compiler version

Unless an explicit reason is given, the newest compiler version should be used. The current implementation uses `pragma solidity ^0.4.24`, which suffers from the following critical bug<sup>3</sup>:

Higher order bits in the exponent are not properly cleaned before the EXP opcode is applied if the type of the exponent expression is smaller than 256 bits and not smaller than the type of the base. In that case, the result might be larger than expected if the exponent is assumed to lie within the value range of the type. Literal numbers as exponents are unaffected as are exponents or bases of type `uint256`.

SMART VALOR needs to be aware that the VALOR TOKEN contract may be affected on Line 19:

```
uint256 internal constant VALOR = 10 ** uint256(decimals);
```

If SMART VALOR chooses to stay with the outdated compiler version, the bytecode generated by the compiler should be checked that it's not affected by this issue.

**Fixed:** SMART VALOR acknowledged the criticality of the bug in the outdated 0.4.24 version and updated the pragma directive to use the latest compiler version, which is 0.4.25.

## Unused write to storage

Writes to storage should be used by the contract, otherwise they are unnecessary.

<sup>3</sup><https://etherscan.io/solcbuginfo?a=ExpExponentCleanup>

## Recommendations / Suggestions

- ✓ To ensure that the value of `totalSupply_` is always correct and to avoid errors on deployment should e.g. the percentage distribution across `employeePoolSupply`, `futureDevFundSupply` and `companyWalletSupply` be changed manually, the following check can be introduced in the constructor:

```
assert(INITIAL_SUPPLY == employeePoolSupply + futureDevFundSupply
      + companyWalletSupply);
```

**Post-audit comment:** SMART VALOR has fixed some of the issues above and is aware of all the implications of those points which were not addressed. Given this awareness, SMART VALOR has to perform no more code changes with regards to these recommendations.

## Disclaimer

UPON REQUEST BY SMART VALOR, 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..