Secure Code Assessment – KnabToken.sol & Crowdsale.sol

Prepared for:

Strategic Land Corporation

Assessment Date: June 2021

Table of Contents

Executive Summary

1 Secure Code Assessment Objective

Strategic Land Corporation (SLC) engaged Digital Warfare (DW) to perform a Secure Code Assessment against the code files listed in the Scope. The goal was to identifying security weaknesses in the code that may have a detrimental effect on the cryptocurrency and the company itself.

This report serves as both a baseline assessment and a roadmap for any improvements.

2 Scope

The scope of the project focused on performing a Secure Code Assessment against SLC's code files listed below. DW was provided full knowledge of the code and design features to provide for a white-box assessment.

The following table outlines the analysis types that were scoped into this project and the targets for each.

Analysis Type	Targets	Goal
Secure Code Assessment	KnabToken.sol Crowdsale.sol	Discover vulnerabilities in the code

3 Results

DW was unable to discover vulnerabilities in SLC's code. The code was well implemented and designed with security in mind.

4 Recommendations

As a result of the Secure Code Analysis, DW assembled several short-term recommendations as guidance to improve the security of their application.

4.1 Short-term Goals

- None

4.2 Long-term Goals

- Document the Policies & Procedures governing the application and its Secure Software Development Life Cycle (S-SDLC) to ensure that security is kept top of mind and the positive security processes are kept in play going forward.
- Continue conducting Secure Code Assessments on each major code release

5 Conclusions

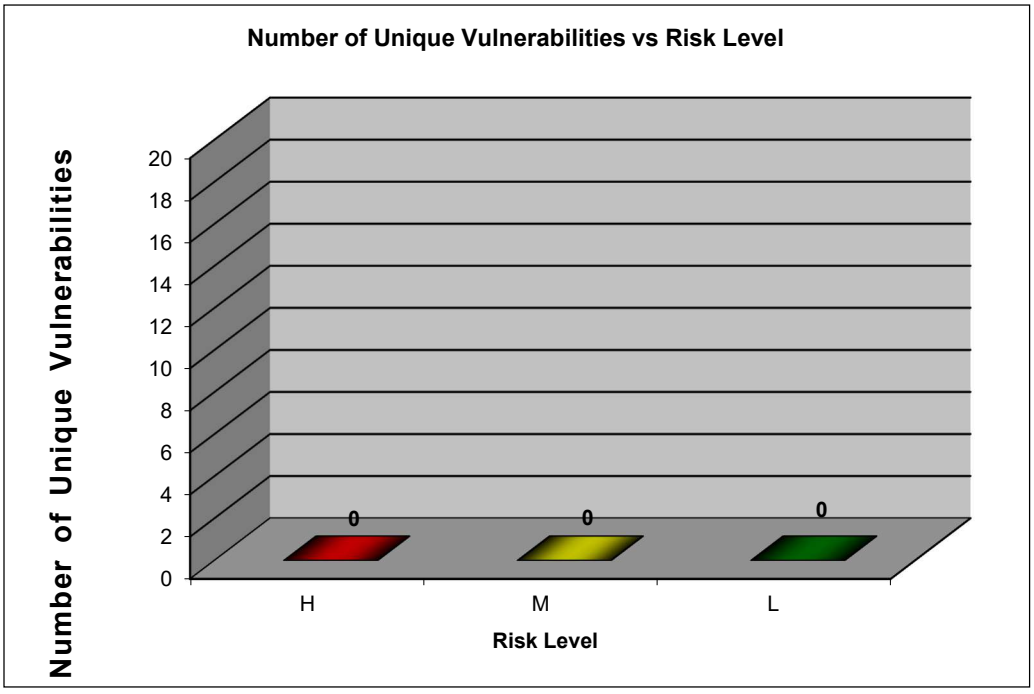
SLC has demonstrated a commitment to security within their code. It is evident that security and the principles of secure coding have been built into their S-SDLC to develop code files that are resilient to attack.

DW assessed the code files within the Scope for a variety of known and unknown attack vectors. DW errs on the side of caution and will raise all initial thoughts to be discussed with the development team. All potential issues were discussed with the development team and, at the end of the process, no issues were left that could be deemed a vulnerability.

DW recommends continuing to put security as a priority to protect the coin and its investors.

DW believes that SLC is ready to release and launch the coin.

The chart below shows a total of **0** unique vulnerabilities.



Secure Code Assessment

1 Assessment

1.1 Approach

Blockchain and Cryptocurrency technologies are the future of many industries. They are already starting to put pressure on many established companies and are driving innovation. They are however highly targeted for attack and can ruin both a company and investors if that attack is successful. Code will often have defects and vulnerabilities. These vulnerabilities are being highly targeted with cryptocurrencies due to the huge windfalls that hackers can gain from a successful attack. As such, code must be thoroughly checked to protect the company, its investors, and coin holders.

The DW approach is to start with gaining a better understanding of the internal structure of the code files. Digital Warfare starts by performing a static call graph analysis of the files, extracting their functions and their relationships. Then, a Secure Code Analysis is performed to identify potential security vulnerabilities in the contracts. To accomplish this, a methodical approach is in order, to ensure that vulnerabilities are not overlooked. Analysis is conducted by assessing the code holistically, as well as walking line-by-line through the code.

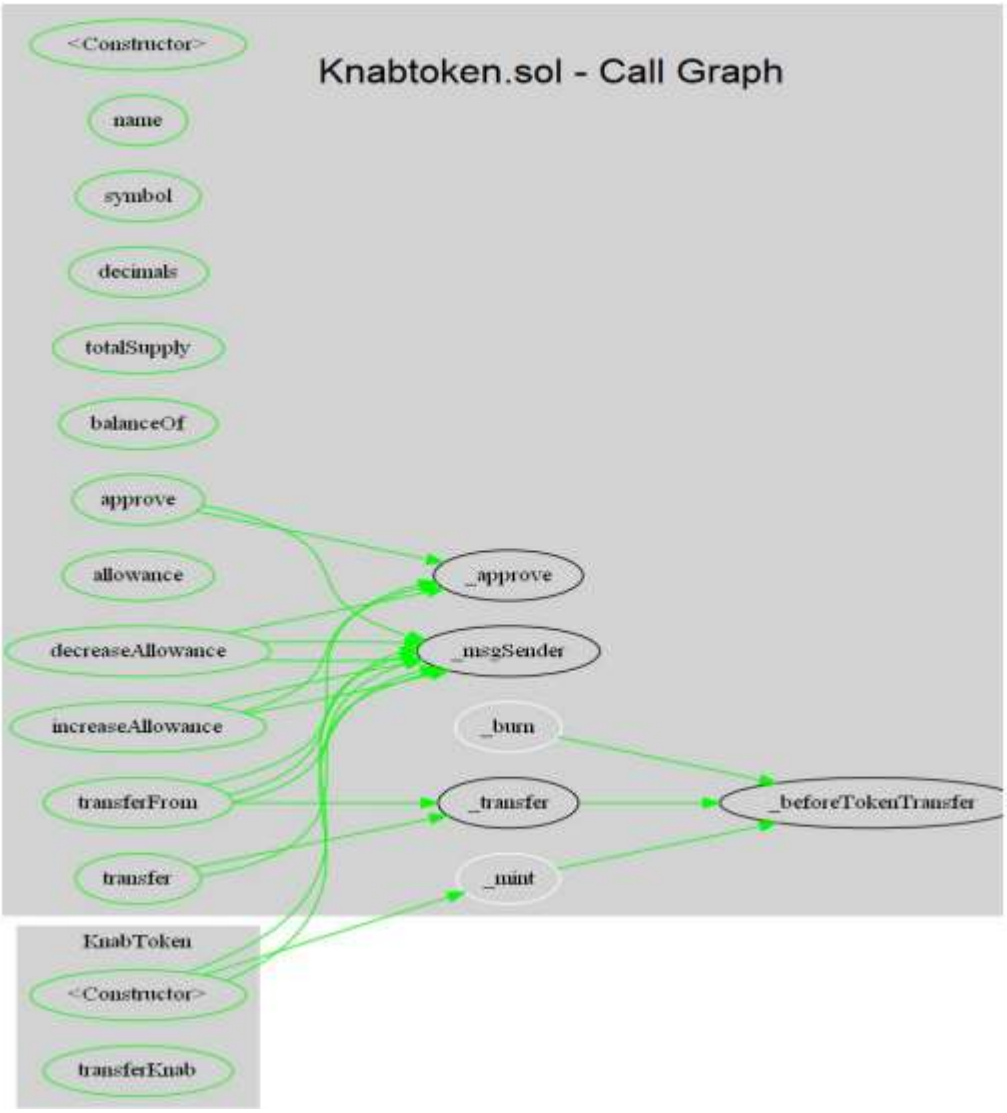
1.2 Objective

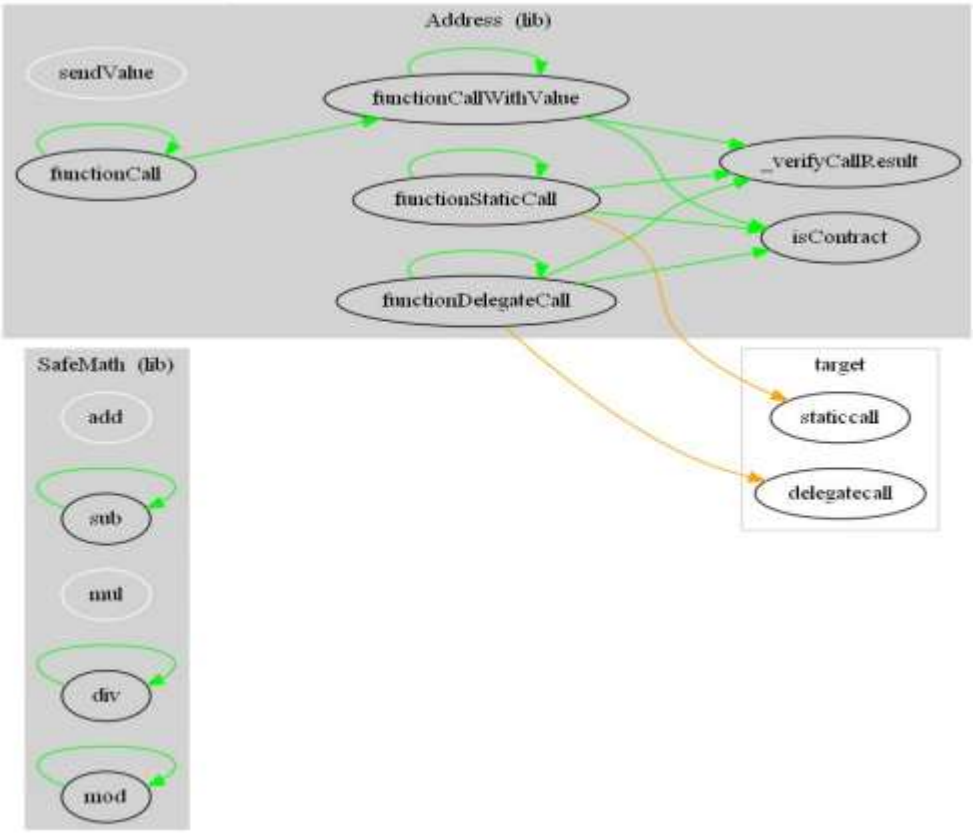
The goal of the Secure Code Analysis was to identify weaknesses present in code due. The test was performed with the objective of discovering security weaknesses in the code that would lead to a compromise in security.

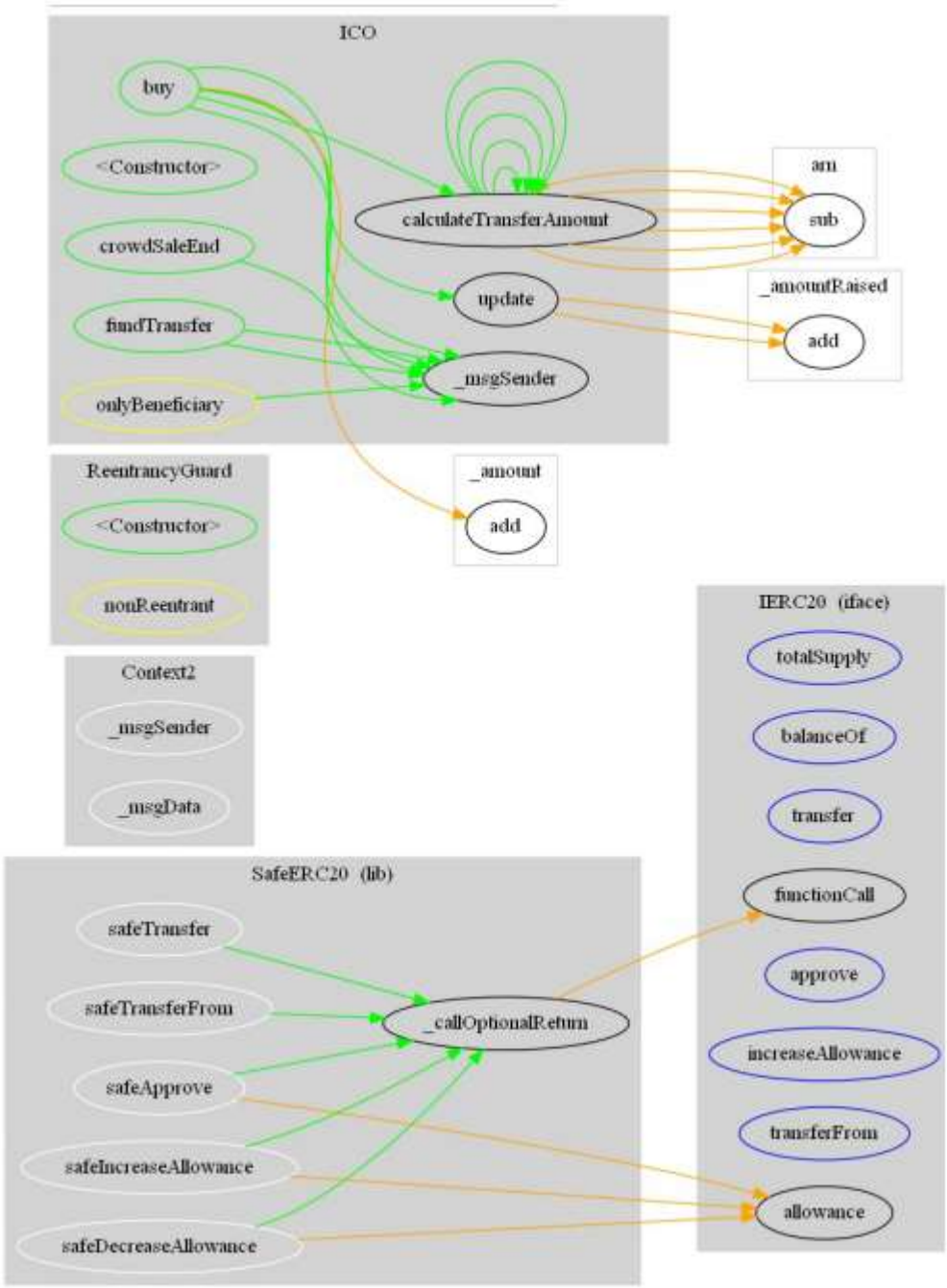
1.3 Analysis

A static call graph analysis was performed on the files to visually see the calls.

Figure 1: Showing a part of the static call graph of KnabToken.sol







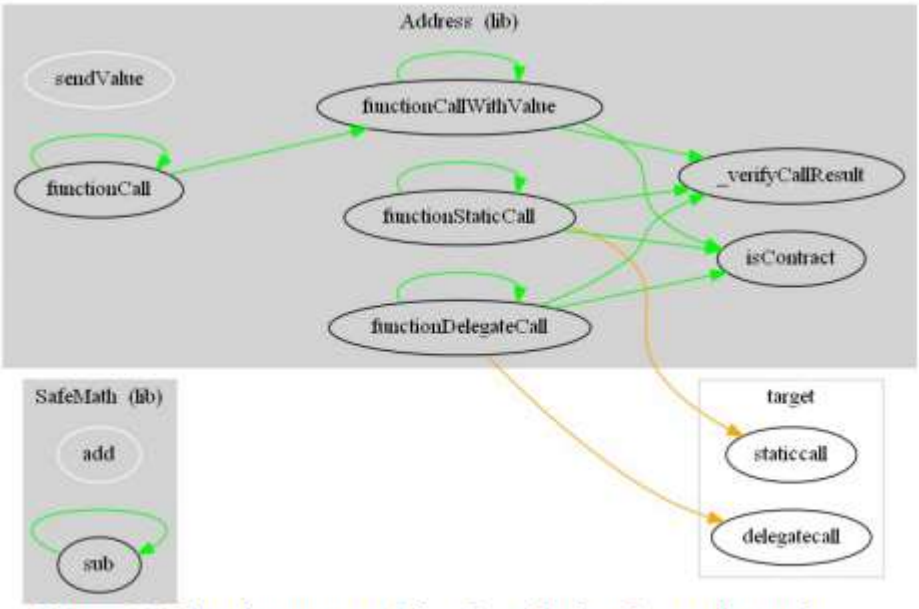


Figure 2: Showing a part of the Crowdsale token call graph

The listings were analyzed to gain a greater understanding of the code. Listing 1 and 2 represent the description of the KnabToken.sol and Crowdsale.sol files, respectively.

Listing1. KnabToken.sol functions' description

```
+ [Internal] IERC20
   - [External] totalSupply
   - [External] balanceOf
   - [External] transfer--> This is a non-constant function
   - [External] allowance
   - [External] approve--> This is a non-constant function
   - [External] transferFrom--> This is a non-constant function

+ [Internal] IERC20Metadata (IERC20)
   - [External] name
   - [External] symbol
   - [External] decimals

+  Context
   - [Internal] _msgSender
   - [Internal] _msgData

+ [Library] SafeMath
   - [Internal] add
   - [Internal] sub
   - [Internal] sub
   - [Internal] mul
   - [Internal] div
   - [Internal] div
   - [Internal] mod
   - [Internal] mod

+ [Library] Address
   - [Internal] isContract
   - [Internal] sendValue--> This is a non-constant function
   - [Internal] functionCall--> This is a non-constant function
   - [Internal] functionCall--> This is a non-constant function
   - [Internal] functionCallWithValue--> This is a non-constant function
  - [Internal] functionCallWithValue--> This is a non-constant function
   - [Internal] functionStaticCall
   - [Internal] functionStaticCall
```

```
  - [Internal] functionDelegateCall--> This is a non-constant function
  - [Internal] functionDelegateCall--> This is a non-constant function
  - [Private] _verifyCallResult

+ [Library] SafeERC20
 - [Internal] safeTransfer--> This is a non-constant function
  - [Internal] safeTransferFrom--> This is a non-constant function
  - [Internal] safeApprove--> This is a non-constant function
  - [Internal] safeIncreaseAllowance--> This is a non-constant function
  - [Internal] safeDecreaseAllowance--> This is a non-constant function
  - [Private] _callOptionalReturn--> This is a non-constant function

+  ERC20 (Context, IERC20, IERC20Metadata)
  - [Public] <Constructor>--> This is a non-constant function
  - [Public] name
  - [Public] symbol
  - [Public] decimals
  - [Public] totalSupply
  - [Public] balanceOf
  - [Public] transfer--> This is a non-constant function
  - [Public] allowance
  - [Public] approve--> This is a non-constant function
  - [Public] transferFrom--> This is a non-constant function
  - [Public] increaseAllowance--> This is a non-constant function
  - [Public] decreaseAllowance--> This is a non-constant function
  - [Internal] _transfer--> This is a non-constant function
  - [Internal] _mint--> This is a non-constant function
  - [Internal] _burn--> This is a non-constant function
  - [Internal] _approve--> This is a non-constant function
  - [Internal] _beforeTokenTransfer--> This is a non-constant function

+  KnabToken (ERC20)
  - [Public] <Constructor>--> This is a non-constant function
     - modifiers: ERC20
  - [Public] transferKnab--> This is a non-constant function
```

Listing 2. Crowdsale.sol functions' description

```
+ [Library] SafeMath
   - [Internal] add
   - [Internal] sub
   - [Internal] sub
   - [Internal] mul
   - [Internal] div
   - [Internal] div
   - [Internal] mod
   - [Internal] mod

+ [Library] Address
   - [Internal] isContract
   - [Internal] sendValue--> This is a non-constant function
   - [Internal] functionCall--> This is a non-constant function
   - [Internal] functionCall--> This is a non-constant function
   - [Internal] functionCallWithValue--> This is a non-constant function
   - [Internal] functionCallWithValue--> This is a non-constant function
   - [Internal] functionStaticCall
   - [Internal] functionStaticCall
   - [Internal] functionDelegateCall--> This is a non-constant function
   - [Internal] functionDelegateCall--> This is a non-constant function
   - [Private] _verifyCallResult

+ [Library] SafeERC20
   - [Internal] safeTransfer--> This is a non-constant function
   - [Internal] safeTransferFrom--> This is a non-constant function
   - [Internal] safeApprove--> This is a non-constant function
   - [Internal] safeIncreaseAllowance--> This is a non-constant function
   - [Internal] safeDecreaseAllowance--> This is a non-constant function
   - [Private] _callOptionalReturn--> This is a non-constant function

+ [Internal] IERC20
   - [External] totalSupply
   - [External] balanceOf
   - [External] transfer--> This is a non-constant function
   - [External] allowance
   - [External] approve--> This is a non-constant function
   - [External] increaseAllowance--> This is a non-constant function
```

```
   - [External] transferFrom--> This is a non-constant function


+  Context2
   - [Internal] _msgSender
   - [Internal] _msgData

+  ReentrancyGuard
   - [Public] <Constructor>--> This is a non-constant function

+  ICO (ReentrancyGuard, Context2)
   - [Public] <Constructor>--> This is a non-constant function
   - [Internal] calculateTransferAmount
   - [Public] buy--> This is a non-constant function
      - modifiers: nonReentrant
   - [Internal] update--> This is a non-constant function
   - [Public] fundTransfer--> This is a non-constant function
      - modifiers: onlyBeneficiary
   - [Public] crowdSaleEnd--> This is a non-constant function
      - modifiers: onlyBeneficiary
```

Several initial items were compiled and discussed with the development team. After in depth discussion and reanalysis, it was determined that no vulnerabilities are present in the 2 code files assessed.

This concludes the Secure Code Analysis.