System Assessment Report
for
DEXTF Project

Version 1.1

July 22, 2020

Prepared by
[redacted] **and reviewed by** [redacted]
for Mangrovia Blockchain Solutions s.r.l.

FOR OFFICIAL USE ONLY

Document Revision History

This DEXTF System Assessment Report (SAR) is a living document that can be changed as required to reflect system, operational, or organizational changes. Modifications made to this document are recorded in the version history matrix below.

At a minimum, this document should be reviewed and assessed annually. Reviews made as part of the assessment process shall also be recorded below.

This document history shall be maintained throughout the life of the document and the associated system.

Date	Description	Version	Author
08/07/2020	Document Release	1.0	[redacted]
12/07/2020	Document Peer Review	1.0	[redacted]
22/07/2020	Document Reviewed with customer	1.1	[redacted]

Security Assessment Report Approval Signatures

I have reviewed the DEXTF Security Assessment Report and accept the analysis and findings within.

[redacted]
{Security Control Assessor Full Name}
Security Control Assessor

22/07/2020
Date

Marco Marchioro
{System Owner Full Name}
System Owner

22/07/2020
Date

Table of Contents

Overview

This document represents the *Security Assessment Report (SAR)* for DEXTF. This SAR contains the results of the security evaluation of the SET-PROTOCOL software and its implementation for DEXTF. The SAR describes the risks associated with the implementation studied during DEXTF's security assessment.

All assessment results have been analyzed to provide the system owner with an insight of the implementation of SET-PROTOCOL and any risk that could be associated to the way the protocol is used and all the features not used by DEXTF but exposed by SET-PROTOCOL standard .

1.1 Purpose

The purpose of the Security Assessment Report is to provide the system owner with a summary of the risks associated with the features not implemented into DEXTF identified by the analisys. A code review has been performed on SET-PROTOCOL to evaluate the system's implementation of, and compliance with, the organization's expectation on what a user can or cannot do.

System Overview

2.1 General System Description and Purpose

DEXTF is a project to implement a Crypto Funds Management system within the Ethereum Mainnet. It has 3 actors, DEXTF, Funds Managers (here called Managers from now on) and Funds Investors (here called Investors from now on). It should permit the Managers to set up funds composition based on a list of ERC20 tokens already present on the Ethereum Mainnet, and allows the Investors to invest into the Funds or redeem their investment any time, receving the Tokens that compose the funds they invested into. Each Fund will generate an ERC20 Token that will represent the fund itself, and the composition of the ERC20 tokens the fund set up.

Assessment Methodology

The security assessment uses a process for determining risk exposure to unwanted usage of the System, being a derivate from another project (SET-PROTOCOL), the assesment try to find any feature not implemented in the DEXTF project and interface that could lead to unwanted act from the Investors or Managers, or even DEXTF itself.

This methodology used to conduct the security assessment for DEXTF system is summarized in the following steps:

1. Analyze the SET-PROTOCOL Smart Contract suite
2. Identify all the exposed function by the Smart Contracts once deployed
3. Identify threats and/or unwanted behaviour and determine which threats are associated with which smart contract
4. Recommend corrective actions
5. Document the results

3.1 Identification of Misbehaving features

Vulnerabilities have been identified for the DEXTF through security analisys of the exposed API.

Vulnerability is an inherent weakness in an information system that can be exploited by a threat or threat agent, resulting in an undesirable impact in the protection of the confidentiality, integrity, or availability of the system (application and associated data). A vulnerability may be due to a design flaw or error in configuration or unused feature still available on the smart contracts which makes your process susceptible to malicious attacks or malicious behaviour or even casual error prone behaviour.

Whether or not a vulnerability has the potential to be exploited by a threat depends on a number of variables including (but not limited to):

- The expected behaviour performed by the users
- The possible interactions that an anonymous user (not part of the DEXTF project) could purposefully perform on the deployed contracts that could lead to an attack
- The probability of an external event or disruption on the Ethereu, mainnet platform that could lead to unexpected situations

The advatage of blockchain like ethereum is also its primary flaw, the immutability and the public aspect of each transaction expose the company to threats that could not be fixed or could not be ‘managed’ because of the nature of smart contracts and blockchain itself, its public nd transparent side also could lead to an bad advertisement or bad reputation of the project despite any action the company could take to fix it.
This is the reason why some of the results could seems ‘finnicky’ or ‘only a small flaw’, but in a distibuted environment where is not possible to ‘rollback’ or ‘fix it under the hood’, even the smallest flaw in what people epxect from the smart contract behaviour could lead to a very serious issue for the project itself.

3.3 Consideration of Threats

A threat is an adversarial force or phenomenon that could impact the availability, integrity, or confidentiality of a software suite. A threat agent is an element that provides the delivery mechanism for a threat. An entity that initiates the launch of a threat agent is referred to as a threat actor.

A threat actor might purposefully launch a threat agent (e.g. a function that allows unauthorized people to change a fund composition). However, a threat actor could be a trusted employee that acts as an agent by making an unintentional human error (e.g. access to functions that could lead to unexpected behaviour of the contract despite the unused feature associated with it).

Table 3-1. Threat Categories and Origination

Threat Origination Category	**Origination Identifier**
Threats launched purposefully	P
Threats created by unintentional human or machine errors	U
Threats caused by intrstructure service disruptions	E

Purposeful threats are launched by threat actors for a variety of reasons and the reasons may never be fully known.

Table 3-2. Potential Threats

Threat Name	Origination Identifier	Description	Typical Impact to Data or System			Impact Level	Likelihood	Risk
			Confidentiality	Integrity	Availability			
Node Package	Node.js	WARN deprecated tar.gz@1.0.7: ⚠ WARNING ⚠ tar.gz module has been deprecated and your application is vulnerable. Please use tar module instead: https://npmjs.com/tar	Medium	Medium	Low	Low	Low	Low
Input Sanitation	Vault.sol	In WithdrawTo _token should be checked if is really a token contract, as delegate call won't do the check themselves and the gas of the call could be lost if the parameter is wrong.	Low	Low	Low	Low	Medium	Low
Input Sanitation	Vault.sol	In transferFrom token should be checked if is really a token contract, as delegate call won't do the check themselves and the gas of the call could be lost if the parameter is wrong.	Low	Low	Low	Low	Medium	Low
Ether send issue	TrasnferProxy.sol and Core.sol	In multiple contracts definition in Core and TrasnferProxy seems possible to send Ether by error to the contract without a way to redeem them	Low	Low	Low	Low	Low	Medium
Reentrancy	Core.sol	BatchDepositInternal function could suffer from reentrancy due to successive delegatecall made one after the other. Noreentrant parent contract could be tested to solve the issue	LOW	Medium	Low	Medium	Low	Medium
Input Sanitation	Core.sol	In CreateSet there are input that are not sanitized, the function seems also to be	Low	Low	Medium	Low	MEdium	Low

Threat Name	Origination Identifier	Description	Typical Impact to Data or System			Impact Level	Likelihood	Risk
		available for any user (not authorizedOnly)						
Input sanitation	Core.sol	In redeemAndWithdrawTo there is no input sanitation on the delegatecall to CoreIssuanceLibrary.calculateWithdrawAndIncrementQuantities , causing possible issues	Low	Low	Low	Low	Medium	Low
Reentrancy	Core.sol	IssueInternal has no Noreentrant contract parent, having some method called before a state change, there is chance to get reentrant code	Low	Medium	Low	Low	Low	Medium

3.4 Perform Risk Analysis

The goal of determining risk exposure is to facilitate decision making on IF or HOW to fix a possible issue based on its RISK. Many of the minor issue could be for example delayed to the next release, that will include new features.

Likelihood definitions for the exploitation of vulnerabilities are found in Table 3-3.

Table 3-3. Likelihood Definitions

Likelihood	Description
Low	There is little to no chance that a threat could exploit a vulnerability and cause loss to the system or its data.
Moderate	There is a moderate chance that a threat could exploit a vulnerability and cause loss to the system or its data.
High	There is a high chance that a threat could exploit a vulnerability and cause loss to the system or its data.

Impact refers to the magnitude of potential harm that could be caused to the system (or its data) by successful exploitation. Definitions for the impact resulting from the exploitation of a vulnerability are described in Table 3-4. These values are perceived values.

Table 3-4. Impact Definitions

Impact	Description
Low	If vulnerabilities are exploited by threats, little to no loss of data would occur. Either publicy side wont' affect the company
Moderate	If vulnerabilities are exploited by threats, moderate loss of data would occur. Either publicy side will affect the company.
High	If vulnerabilities are exploited by threats, significant loss of data would occur. Either publicy side will hugely affect the company

The combination of Impact and likelihood can be reduced as in table 3-5.

Table 3-5. Risk Exposure Ratings

Likelihood	Impact		
	Low	Moderate	High
High	Low	Moderate	High
Moderate	Low	Moderate	Moderate
Low	Low	Low	Low

3.5 Document Results

Documenting the results of security review creates a record that can be reviewed for risk-based decision making and to create plans of action to mitigate risks or to change the development plan or effort.

Security Assessment Results

This section describes all security weaknesses found during testing. The following elements for each weakness are reported:

- Finding Number
- Source
- Risk
- Business Impact Statement
- Recommended Corrective Action
- Likelihood
- Impact
- Risk Level

The reader of the SAR should anticipate that the security weakness elements are described as indicated below.

- **Finding Number:** The unique number for reference.
- **Source:** The contract or library or file that cause the issue.
- **Risk:** The finding description. This is the description of the finding as defined from the vulnerability assessment, security audit report, or from the results of an internal security review.
- **Business Impact Statement:** Impact of the finding to the organization if exploited.
- **Recommended Corrective Action:** The recommended control(s) needed to remediate the finding.
- **Link to Control(s)/Test Case(s):** The control and/or test case associated with the finding.
- **Likelihood:** The likelihood that the finding will be exploited (High, Moderate, or Low).
- **Impact:** The impact to the organization if the finding is exploited (High, Moderate, or Low).
- **Risk Level:** The computed risk level associated with the finding based on the selected likelihood and impact. Please refer to Table 3-5 Risk Exposure Ratings for details.

Table 4-1. Potential Misuses Results

Contract File	Weaknesses	Workaround	Likelihood	Impact	Risk Level
Core.sol/ TrasferProxy .sol and Vault.sol	RenounceOwnership() called by mistake gives ownership to address(0) loosing forever the possibility to get it back (should only have transfer ownership)	Allow to have more than one owner or disable the function if is not a wanted feature	Low	**High**	**Low**
Core.sol	InternalTransfer allows any user to transfer their tokens to another user, is not clear what happen to the set token once the vault tokens are transferred, seems possible that some vault token could be owned by a user that have not set token (if this is not the expected behaviour). It's not clear (to be tested) if redeemandwithdraw check these properly.	Disallow internal transfer, but only after all the internal functional test pass	Low	**Medium**	**Low**
Core.sol	CreateSet is callable by anyone, seems like there is no restriction for allowed users by DEXTF. Anyone can use the project to create a set without any authorization.	An authentication system should be implemented at set level, at least createSet. Like TrasferProxy and Vault implement authorizable for contract it should be implemented for roles in the project here if this is the expected behaviour, require anyway to be tested	High	**Medium**	**Low**