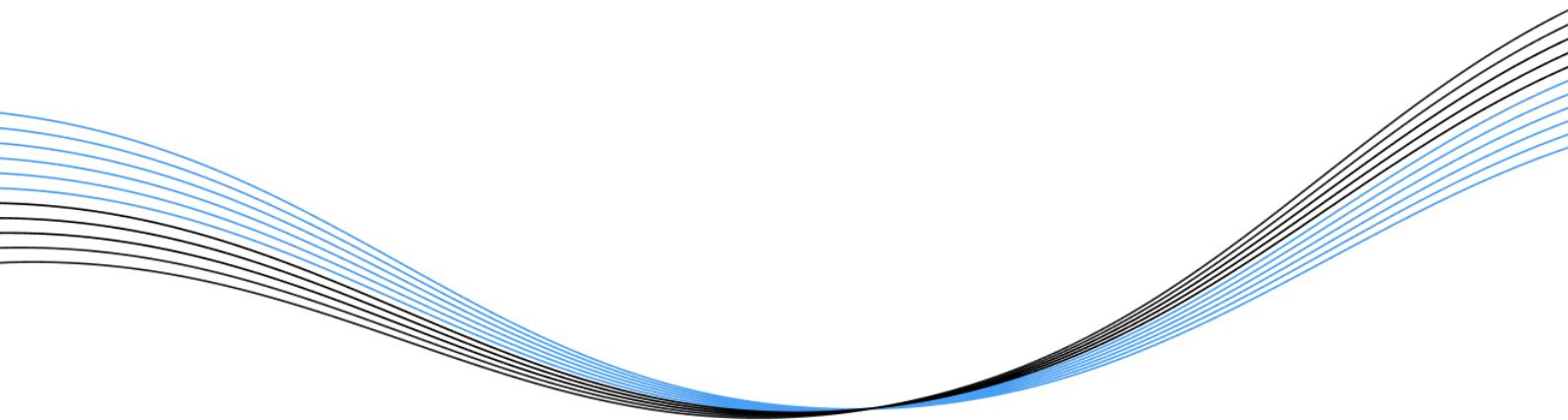


Hexlant.

© Hexlant Inc.
340 Gangnam-daero
Seoul, Republic of Korea
06242
Hexlant.com

KLAYBANK-BONDSALE SMART CONTRACT AUDIT REPORT



Audit Date
13 Jan 2022

Category
De-fi

Auditor
Hexlant Audit Team

This contract specifies that it has been validated by the Hexlant Technical Team and notifies that it has not any technical defects.

AUDIT

OVERVIEW

PUBLISHED INFORMATION

REPORT NUMBER	HEXLANT20220113_01
DATE	2022/01/13
AUDIT SCOPE	KlayKbtLpDepository

PROJECT INFORMATION

TITLE	KlayBank Bondsale
TYPE	DEFI
PLATFORM	KLAYTN
CONTRACT ADDRESS	
REPOSITORY	https://github.com/klaybank/klaybank-audit-hexlant
AUDIT COMMIT HASH	cf839c4f83c13df43f26a590d5c940b35261504b 99f2e29260b9a363e27a5e2cb350e3ba707a1df8 4a2858a1b9daf4a0431387d6a31ae07796727244 18b0ff4696d976711e81a133ad6a1ca12a784b59 c0616c26d4e9dbb7b5fed38775fc254cf664e731 LATEST 8128a7ea3339e62e5d2a25465d5795cf907dcd87

VULNERABILITY ANALYSIS

CRITICAL	0	No relevant provision
HIGH (Resolve)	1 → 0	recoverLostToken()에서 KBT token 제한
MEDIUM (Resolve)	4 → 0	register() 함수 내에 잘못된 로직 존재 외 3
LOW	0	No relevant provision

CENTRALIZED FUNCTION

register, unregisterDepositor	Function to register token and depositor allowed to deposit
initializeBondTerms	Initializer for Bond terms, totalDebt, lastDecay
setBondTerms	Function to set parameters for bond
setAdjustment	Function to set control variable adjustment
setStaking	Function to set contract for auto stake

COMPANY PROPOSAL

Hexlant는 2018년에 설립한 블록체인 기술 기업입니다. 삼성전자 출신의 보안·네트워크·소프트웨어 전문가가 스마트 컨트랙트와 블록체인 프로토콜의 보안 결함을 발견하고 블록체인 생태계의 기술 안정성을 입증하기 위해 설립하였습니다.

Hexlant는 블록체인 동작 환경을 파악하기 위해 20개 이상의 블록체인 메인넷을 직접 구축하고 있습니다. 나아가 키 보안 알고리즘 및 메인넷 모니터링 기술을 개발했습니다. 이 방식은 비트코인, 이더리움, 폴카닷, 애이다 등 헥슬란트가 보유한 모든 메인넷 플랫폼에서 적용되고 운영됩니다.

Hexlant는 위와 같은 기술 운영 경험을 바탕으로 스마트 컨트랙트 기술을 검증합니다. 스마트 컨트랙트 내 버그를 발견하는 오류 테스트 뿐만 아니라 메인넷 상황에서의 문제점을 탐지하며 서비스 관점에서 지속적으로 운영할 수 있는 블록체인 기술 가이드를 제공합니다.

Hexlant의 고객사는 컨트랙트에 대한 취약성 감사부터 오너 키 관리, 블록체인 지갑 시스템 구축 등 블록체인 기술 전반의 서비스를 제공받을 수 있습니다. 현재 200여개의 고객사가 Hexlant의 서비스를 바탕으로 블록체인 사업을 시작, 운영했으며 누적으로 관리하는 자산은 12조를 달성했습니다.

Initials for identification purposes:


For identification purposes
Hexlant.

CONTENTS

- 1. Analysis Purpose**
- 2. Vulnerability Classification**
- 3. Test Result**
- 4. Test Case**
- 5. Vulnerability Analysis**
- 6. Conclusion**

ANALYSIS PURPOSE

본 리포트는 발행된 컨트랙트 코드가 요구사항을 충분히 만족하는지, 그리고 보안의 취약점과 실제 운영하면서 발생 할 수 있는 문제들을 파악하고 해결방안을 찾기위해 분석을 수행하고 그 결과를 정리하였습니다. 이번 코드 분석은 다음과 같은 요소들을 검증하기위해 진행하였습니다.

- 구현된 기능의 정상 작동 여부
- 기능 수행 중 보안 위험성
- Off Chain에서 발생하는 문제에 대한 대비
- 컨트랙트 코드의 가독성 및 코드 완성도

VULNERABILITY CLASSIFICATION

본 취약성 검증은 오류 위험도를 아래와 같이 분류, 평가합니다.

• Critical Severity

심각성 치명적 단계는 큰 보안 결함을 뜻하며 자산 탈취 및 동결, 추가 발행 등 치명적인 문제를 야기합니다. 본 결함은 반드시 수정되어야 합니다.

• High Severity

심각성 높은 단계는 특수 조건에 의해 보안 결함이 발생할 수 있는 항목이며 수정을 강력하게 권고합니다.

• Medium Severity

심각성 중간 단계는 보안 결함은 아니나 비효율적인 컨트랙트 동작을 야기합니다. 컨트랙트를 효율적으로 동작하도록 수정을 권유하는 항목입니다.

• Low Severity

심각성 낮음 단계는 보안에는 문제가 없으나 컨트랙트 구조 개선을 위해 수정을 권유하는 항목입니다.

Severity	Issue	Status (Found)	Status (Resolve)
• CRITICAL (0)	-		
• HIGH (1)	recoverLostToken()에서 KBT token 제한	(Found - 99f2e2)	(Resolve - c0616c)
• MEDIUM (4)	register() 함수 내에 잘못된 로직 존재	(Found - 99f2e2)	(Resolve - 4a2858)
	zero address 예외 처리	(Found - 99f2e2)	(Resolve - 8128a7)
	initializeBondTerms() 유효성 검사	(Found - 99f2e2)	(Resolve - 8128a7)
	setAdjustment() 유효성 검사	(Found - 99f2e2)	(Resolve - 8128a7)
• LOW (0)	-		

TEST RESULT

Code Coverage

코드 커버리지는 작성한 테스트가 얼마만큼 컨트랙트 코드의 기능을 테스트 했는지 알 수 있는 정량적인 지표입니다.

라이브러리와 일부 컨트랙트에 구현된 기능에 대해 추가적인 호출이 진행되지 않은 경우가 존재합니다.

아래의 Coverage 지표는 위 사항을 반영한 결과입니다.

File Name	Statements	Functions	Lines
BondTreasury.sol	100%	100%	100%
BondDepository.sol	98.2%	100%	98.17%

TEST CASE

실제 적용한 테스트케이스 목록입니다. 총 45개 테스트 시나리오를 적용하여 검수했습니다.

Test Case 1. BondDepository.sol (27개의 테스트)

DEPLOY / INITIALIZE

1.1. initialize	Result	
0x0 주소를 입력 받을 시 예외처리 되는가?	PASS	FAIL
initializer가 한번만 호출되는가?	PASS	FAIL
owner 주소만이 initializeBondTerms() 를 호출할 수 있는가?	PASS	FAIL
initializeBondTerms()이 한번만 호출되는가?	PASS	FAIL

SETTERS

2.1. setBondTerms	Result	
owner가 아닌 주소에서 호출시 예외처리 되는가?	PASS	FAIL
유효하지 않은 input 입력시 예외처리 되는가?	PASS	FAIL

2.2. setAdjustment	Result	
owner가 아닌 주소에서 호출시 예외처리 되는가?	PASS	FAIL
_increment의 값이 범위를 초과할 때 예외처리 되는가?	PASS	FAIL

2.3. setStaking	Result	
owner가 아닌 주소에서 호출시 예외처리 되는가?	PASS	FAIL
staking address 가 0x0일 때 예외처리 되는가?	PASS	FAIL

DEPOSIT

3.1. deposit	Result	
_depositor 가 0x0일 때 예외처리 되는가?	PASS	FAIL

amount값이 너무 작거나 클때 예외처리 되는가?	PASS	FAIL
유저의 BondInfo 를 정확하게 업데이트 하는가?	PASS	FAIL
totalDebt와 Bond Price 를 정확하게 업데이트 하는가?	PASS	FAIL
msg.sender로부터 Treasury로 principle 토큰을 전송하고, Treasury로부터 Depository와 DAO주소로 KBT를 전송하는가?	PASS	FAIL
adjustment를 정확하게 업데이트 하는가?	PASS	FAIL
이벤트를 발생시키는가?	PASS	FAIL

REDEEM

4.1. redeem	Result	
recipient 가 0x0일 때 예외처리 되는가?	PASS	FAIL
입금 기록이 없을 때 예외처리 되는가?	PASS	FAIL
Vesting Term이 지난 후 상환 시, 유저의 BondInfo를 삭제하는가?	PASS	FAIL
Vesting Term이 지난 후 상환 시, 정확한 수량의 KBT를 recipient에게 전달하는가?	PASS	FAIL
Vesting Term이 지나기 전 상환 시, 유저의 BondInfo를 업데이트하는가?	PASS	FAIL
Vesting Term이 지나기 전 상환 시, 정확한 수량의 KBT를 recipient에게 전달하는가?	PASS	FAIL
이벤트를 발생시키는가?	PASS	FAIL

RECOVER

5.1. recoverLostToken	Result	
중요한 토큰(KBT, principle)을 전달하지 않도록 하는가?	PASS	FAIL
잘못 전송된 토큰을 DAO 주소로 보내는가?	PASS	FAIL

COMPLEX ENVIRONMENT

6.1. Multi players	Result	
여러 유저의 입금 및 상환이 제대로 작동하는가?	PASS	FAIL

Test Case 2. BondTreasury.sol (18개의 테스트)

INITIALIZE

1.1. initialize	Result	
0x0 주소를 입력 받을 시 예외처리 되는가?	PASS	FAIL
initialilzer가 한번만 호출되는가?	PASS	FAIL

REGISTER

2.1. register	Result	
0x0 주소를 입력 받을 시 예외처리 되는가?	PASS	FAIL
owner가 아닌 주소에서 호출시 예외처리 되는가?	PASS	FAIL
중복된 depository 등록시 예외처리 되는가?	PASS	FAIL
중복된 토큰 등록시 예외처리 되는가?	PASS	FAIL

DEPOSIT

3.1. deposit	Result	
등록되지 않은 depositor로부터의 입금 시 예외처리 되는가?	PASS	FAIL
등록되지 않은 토큰 입금 시 예외처리 되는가?	PASS	FAIL
0을 입력받을 시 예외처리 되는가?	PASS	FAIL
depositor로부터 principle 토큰을 받고, KBT를 전달하는가?	PASS	FAIL
okenPaidAmounts를 업데이트 하는가?	PASS	FAIL
KBT 수량 초과시 예외처리 되는가?	PASS	FAIL
이벤트를 발생시키는가?	PASS	FAIL

UNREGISTERDEPOSITOR

4.1. unregisterDepositor	Result	
owner가 아닌 주소에서 호출시 예외처리 되는가?	PASS	FAIL
등록되지 않은 depositor와 호출 시 예외처리 되는가?	PASS	FAIL
등록 최소한 depositor을 재등록시 리스트에 중복등록 되지 않는가?	PASS	FAIL

RECOVER

5.1. recoverLostToken	Result	
중요한 토큰(KBT, principle)을 전달하지 않도록 하는가?	PASS	FAIL
잘못 전송된 토큰을 DAO 주소로 보내는가?	PASS	FAIL

VULNERABILITY ANALYSIS

총 5개 항목의 취약점을 발견하였으며, 발견된 5개의 취약 사항은 아래와 같이 모두 해결되었습니다.

- **BondTreasury.sol - 03 : recoverLostToken()**에서 **KBT token 제한**
- **Status :** Found - 99f2e2 → **Reslove - c0616c**

Type	Severity	Location
Validation of address	• HIGH (Resolve)	BondTreasury.sol L97
· Description	BondTreasury는 KBT 토큰을 보유하고, <code>deposit()</code> 호출 시 <code>msg.sender</code> 에게 KBT 토큰을 전달하는 역할을 합니다. KBT 토큰이 <code>isReserveToken</code> 에 등록되지 않을 경우 <code>recoverLostToken()</code> 함수를 통해 누구나 BondTreasury가 보유한 KBT 토큰을 DAO에게 전송할 수 있습니다. BondTreasury는 KBT Token을 보유하고 있어야 함으로 KBT 토큰을 <code>recoverLostToken()</code> 에서 사용하지 못하도록 수정하는 것을 추천합니다.	
· Recommendation	<ol style="list-style-type: none">1. <code>recoverLostToken()</code> 함수에 <code>require(DAO_ != address(0) && KBT_ != address(0));</code> <code>require(_token != address(0) && _depositor != address(0));</code> 추가	
· Resolved	<code>recoverLostToken()</code> 함수에 <code>require(_token != KBT, "BondTreasury: cannot withdraw KBT");</code> 추가 (L98)	

- **BondTreasury.sol - 01 : register()** 함수 내에 잘못된 로직 존재
- **Status :** Found - 99f2e2 → **Resolve - 4a2858**

Type	Severity	Location
Validation of logic	● Medium (Resolve)	BondTreasury.sol L73
· Description :	<p><code>_register()</code> 함수는 <code>deposit()</code>을 실행할 수 있는 토큰과 <code>depositor</code>를 설정해주는 함수입니다. L73 - L82은 <code>_token</code>을 이미 등록된 <code>depositor</code>과 비교하고 <code>_token</code>이 <code>depositor</code>로 등록되어 있는 경우에만 <code>_reserveDepositors</code> 리스트에 <code>_depositor</code>을 추가합니다.</p> <p><code>_token</code>이 <code>_reserveDepositors</code> 리스트에 존재하는지 비교하는 과정은 불필요함으로 로직을 올바르게 변경하는 것을 추천합니다.</p>	
· Recommendation :	<p>1. 로직 수정</p>	
· Resolved	<p>L76에서 <code>_token</code>을 <code>_depositor</code>으로 교체하여 <code>unregisterDepositor()</code>을 통해 등록 해제됐지만 <code>_reserveDepositors</code> 리스트에 남아있는 <code>_depositor</code>가 리스트에 중복 추가되는 것을 막아주도록 수정되었다.</p>	
· Description	<p><code>_initialize()</code>, <code>register()</code> 함수가 zero address를 인자로 받을 수 있습니다.</p>	
· Recommendation	<p>1. <code>_initialize()</code>에</p> <pre>require(DAO_ != address(0) && KBT_ != address(0)) require(_token != address(0) && _depositor != address(0)) 추가</pre> <p>2. <code>register()</code>에</p> <pre>require(_token != address(0) && _depositor != address(0)) 추가</pre>	

· Resolved

L30

```
require(KBT_ != address(0), "BondTreasury: 0 address");
require(DAO_ != address(0), "BondTreasury: 0 address");
```

L66

```
require(_token != address(0) && _depositor !=
address(0),"BondTreasury:0 address");
```

➤ BondDepository.sol - 01 : initializeBondTerms() 유효성 검사

➤ Status : Found - 99f2e2 → **Resolve - 8128a7**

Type	Severity	Location
Validation of address	● Medium (Resolve)	BondDepository.sol L121

· Description : *initializeBondTerms()* 함수는 bond의 terms를 초기화해주는 함수입니다. terms를 수정하는 역할인 *setBondTerms()*함수는 수정할 *terms.maxPayout*, *terms.fee* 값이 유효한 범위인지 검사를 진행하지만, *initializeBondTerm()*함수는 유효성 검사를 진행하지 않습니다. *initializeBondTerms()* 함수에도 유효성 검사를 추가하는 것을 추천합니다.

· Recommendation :

1. *initializeBondTerms()* 함수에

```
require(_maxPayout <= 10000);
require(_fee <= 10000); 추가
```

· Resolved

L131

```
require(_maxPayout <= 10000, "BondDepository: payout cannot be
above 1 percent");
require(_fee <= 10000, "BondDepository: DAO fee cannot exceed
payout");
```

➤ **BondDepository.sol - 02 : setAdjustment()** 유효성 검사

➤ **Status :** Found - 99f2e2 → **Resolve - 8128a7**

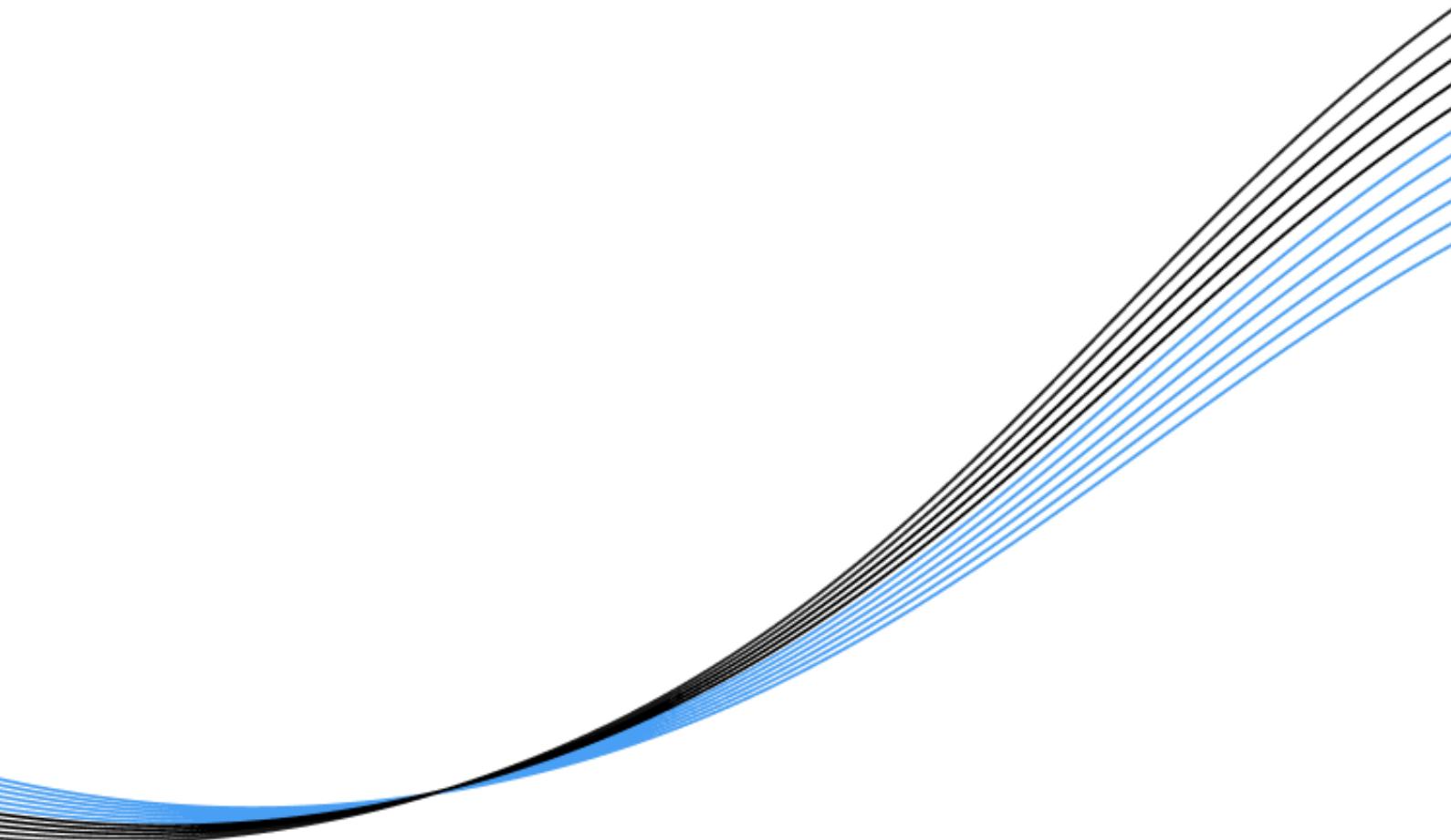
Type	Severity	Location
Validation of address	● Medium (Resolve)	BondDepository.sol L175
· Description :	<p><code>adjust()</code> 함수는 <code>terms.controlVariable</code> 값을 조절하는 역할을 합니다. <code>adjustment.add</code>가 참이고, <code>adjustment.target < terms.controlVariable</code> 이도록 <code>setAdjustment()</code>를 통해 <code>adjustment</code>를 설정하면, <code>terms.controlVariable</code>은 조건을 충족했지만 <code>adjustment.buffer</code>가 지나면 한번 업데이트 됩니다. 잘못된 업데이트가 일어나지 않도록 수정하는 것을 추천합니다.</p>	
· Recommendation :	<p>1. <code>setAdjustment()</code>에</p> <pre>require(_addition ? (_target > terms.controlVariable) : (_target < terms.controlVariable)); 추가</pre>	

· Resolved L184

```
require(_addition ? (_target > terms.controlVariable) : (_target <
    terms.controlVariable), "BondDepository: wrong target value");
```

Declare

해당 리포트는 Hexlant의 스마트 컨트랙트 보안 감사 결과를 바탕으로 작성되었습니다. 해당 리포트는 비즈니스 모델의 적합성과 법적 규제, 투자에 대한 의견을 보증하지 않습니다. 리포트에 기술한 문제점 이외에 메인넷 기술 또는 가상머신을 비롯하여 발견되지 않은 문제점이 있을 수 있습니다. 해당 리포트는 논의 목적으로만 사용됩니다.



Hexlant.

-
contact@hexlant.com
www.hexlant.com