BIFI SECURITY ASSESSMENT REPORT

NOV. 13, 2020

시작하기 전에

- 본 문서는 블록체인 보안 전문업체 SOOHO에서 진행한 취약점 검사를 바탕으로 작성한 문서로, 보안 취약점의 발견에 초점을 두고 있습니다. 추가적으로 코드 퀄리티 및 코드 라이센스 위반 사항 등에 대해서도 논의합니다.
- 본 문서는 코드의 유용성, 코드의 안정성, 비즈니스 모델의 적합성, 비즈니스의 법적인 규제, 계약의 적합성, 버그 없는 상태에 대해 보장하거나 서술하지 않습니다. 감사 문서는 논의 목적으로만 사용됩니다.
- SOOHO는 회사 정보가 대외비 이상의 성격을 가짐을 인지하고 사전 승인 없이 이를 공개하지 않습니다.
- SOOHO는 업무 수행 과정에서 취득한 일체의 회사 정보를 누설하거나 별도의 매체를 통해 소장하지 않습니다.
- SOOHO는 스마트 컨트랙 분석에 최선을 다하였음을 밝히는 바입니다.

SOOHO 소개

SOOHO는 Audit Everything, Automatically란 슬로건으로 지속적인 보안을 위해 필요한 기술을 연구하고 서비스합니다. 자체 취약점 분석기들과 오픈소스 분석기들을 기반으로 모든 개발 생애 주기에 걸쳐 취약점들을 검사합니다. SOOHO는 자동화 도구를 연구, 개발하는 보안 분야 박사 연구원들과 탐지 결과와 컨트랙트 코드를 깊게 분석하는 화이트 해커들로 구성되어 있습니다. 보안 분야 전문성을 바탕으로 파트너 사의 컨트랙트를 알려진 취약점과 Zero-day 취약점의 위협으로부터 안전하게 만들어줍니다.

개 요

2020년 11월 13일까지 파이랩팀의 변경된 BiFi 컨트랙트에 대한 취약점 분석을 진행하였습니다. 감사 기간 동안 아래의 작업을 수행했습니다.

- SOOHO의 자체 취약점 검사기를 통한 취약점 탐지 및 결과 분석
- 보안 취약점 의심 지점에 대한 익스플로잇(Exploit) 코드 작성
- 컨트랙트 코드 모범 사례와 시큐어 코딩 가이드를 바탕으로 코드의 수정 권고 사항 작성

보안 전문가들이 컨트랙트의 취약점을 분석하였습니다. 참여한 보안 전문가는 국내외의 블록체인 해커톤 해킹 대회에서 수상을 하고 보안분야 박사 학위의 학문적 배경을 가지는 등 우수한 해킹 실력과 경험을 가지고 있습니다.

분석기 SOOHO를 통해 11월 13일까지 수집된 알려진 취약 코드 데이터 베이스의 시그니쳐를 목표 컨트랙트에서 스캐닝하였습니다. 또한 Symbolic Analysis 기반의 취약점 분석기를 이용한 검사 프로세스를 진행하였습니다.

분석 결과 **이슈는 총 1개로 심각도 순서대로 Medium 1개입니다.** 꾸준한 코드 감사를 통해 서비스의 안정을 도모하고 잠재적인 취약점에 대한 분석을 하는 것을 추천 드립니다.

분석 대상

분석 기간 동안 아래의 프로젝트를 분석하였습니다.

Project	20201022-pilab-audit
checksum	3808f009
# of Files	22
# of Lines	3,112

주요 감사 포인트 및 프로세스

BiFi은 사용자가 자신의 가상자산을 예금을 하고 해당 예금액을 담보로 다른 종류의 가상자산을 대출하는 예금/대출 DeFi 서비스입니다. 파이랩 팀은 이더리움 블록체인 기반의 스마트 컨트랙트를 이용해 시스템을 설계 및 개발하였습니다. DeFi 환경에서 주로 발생 가능한 이슈와 고려되어야 하는 사항들에 대한 분석이 진행되었습니다. 지난 BIFI 코드에서 변경된 내역을 중심으로 분석을 진행하였습니다.

예를 들어, 관리자 권한 관리, 입력값 검증, 잘못된 자동 청산, 이자 계산 시의 오류 등을 분석하였습니다. **단, 청산 요청은 정상적으로 발생하고, 특정 상황을 제외하고는 Oracle 정보는 정상임을 전제로 분석하였습니다. 또한 이자 계산 방식은 사용자들이 사전에 인지하고 있음을 전제하였습니다.**

1

취약점 자동 분석
소프트웨어 검사

2

보안 전문가의
코드 직접 분석

다음을 고려합니다.

- 보다 큰 리스크를 내포하는 코드를 중점적으로 분석합니다.
- 접근 권한의 관리가 잘 되고 있는지 검사합니다.
- 파트너 사가 의도한대로 코드가 작성되었는지 분석합니다.

3

Exploit 분석과
Poc 작성

Exploit 분석과 PoC 작성은 다음을 목표로 합니다.

- 코드 실행을 통한 동적 분석
- 발견된 취약점을 악용해 서비스에 악영향을 끼칠 수 있는지 살펴봅니다.

취약점의 심각성 척도

발견된 취약점은 심각성 척도를 기준으로 나열해서 설명합니다.

심각성 척도는 우측 OWASP의 Impact & Likelihood 기반 리스크 평가 모델을 기반으로 정해졌습니다. 해당 모델과 별개로 심각도가 부여된 이슈는 해당 결과에서 그 이유를 서술합니다.

Impact	Likelihood: Low	Likelihood: Medium	Likelihood: High
High	Medium	High	Critical
Medium	Low	Medium	High
Low	Note	Low	Medium

Severity

분석 결과

분석 결과는 심각도에 따라 Critical, High, Medium, Low, Note로 표현됩니다. Sooho는 발견된 모든 이슈에 대해서 개선하는 것을 권장합니다.

AUTHORIZATION THROUGH TX.ORIGIN (SWC-115) Medium

분석 결과에 대한 추가적인 자료 및 코멘트

File Name : `managerDataStorage.sol`
File Path : `20201022-pilab-audit/2. truffle/contracts`
`└─contracts/marketHandler/marketHandlerDataStorage`
`└─handlerDataStorage.sol`
MD5: d48fbb460ab93cbfc2e780484111b292

```
44    modifier onlyOwner {
45      require((msg.sender == owner) || (tx.origin == owner), "onlyOwner function");
46      _;
47    }
```

자세한 내용은 Smart Contract Weakness Classification and Test Cases의 SWC-115 항목을 참고하세요.

이슈 설명

`onlyOwner` modifier에서 `tx.origin`을 이용해 권한을 관리합니다. 권한 부여에 해당 변수를 사용하면 권한있는 계정이 악의적인 컨트랙트를 호출하는 경우 컨트랙트가 취약해질 수 있습니다. `tx.origin`은 이 경우, 트랜잭션의 원래 발신자(승인된 계정)를 반환하기 때문에 승인 확인을 통과하는 취약한 계약을 호출 할 수 있습니다.

검증하였습니다 - PROXY

분석 결과에 대한 추가적인 자료 및 코멘트

File Name : `callProxy.sol`
File Path : `20201022-pilab-audit/2. truffle/contracts`
`└─front/callProxy.sol`
MD5: 64ef85a8382e6be9a6c265d057c84890

설명

`callProxy.sol` 컨트랙트를 분석하였습니다. 함수의 이름과 다른 값이 출력되는 함수가 있는지를 살펴보았습니다.

검증하였습니다 - TOKEN HANDLER MODEL

분석 결과에 대한 추가적인 자료 및 코멘트

File Name : `tokenHandler.sol`
File Path : `20201022-pilab-audit/2. truffle/contracts`
`└─marketHandler/tokenHandler.sol`
MD5: 15ce858ceaf8f14143e64dc62d8648f3

설명

`tokenHandler.sol` 컨트랙트를 분석하였습니다. 최근 발생한 Akropolis 해킹과 같은 재진입 공격 혹은 토큰 오인으로 인한 피해 여지가 있는지를 집중적으로 분석하였습니다.

분석 결과

분석 결과는 심각도에 따라 Critical, High, Medium, Low, Note로 표현됩니다. Sooho는 발견된 모든 이슈에 대해서 개선하는 것을 권장합니다.

검증하였습니다 - LIQUIDATION ✓

분석 결과에 대한 추가적인 자료 및 코멘트

File Name : `liquidationManager.sol`
File Path : `20201022-pilab-audit/2. truffle/contracts`
`└── marketManager/liquidationManager.sol`
MD5: bbb9abe46848bff5e7c1565341a7829e

설명 `liquidationManager.sol` 컨트랙트를 분석하였습니다. 부분 청산 및 청산 여부 판단 기능이 올바르게 동작하는 것을 확인하였습니다.

검증하였습니다 - INTEREST MODEL ✓

분석 결과에 대한 추가적인 자료 및 코멘트

File Name : `interestModel.sol`
File Path : `20201022-pilab-audit/2. truffle/contracts`
`└── interestModel/interestModel.sol`
MD5: 09d33da59e228b11682e6ecf96b39a5f

설명 `interestModel.sol` 컨트랙트를 분석하였습니다. Interest Delta 값의 연산과 이자를 실제로 갱신하는 부분의 로직을 집중적으로 분석하였습니다. 주로 발생하는 이자의 중복 계산이나 음전 계산 등의 이슈가 없음을 확인하였습니다.

검증하였습니다 - COIN HANDLER ✓

분석 결과에 대한 추가적인 자료 및 코멘트

File Name : `coinHandler.sol`
File Path : `20201022-pilab-audit/2. truffle/contracts`
`└── marketHandler/coinHandler.sol`
MD5: 2bb0060b5c87ea25bd31193da96792b4

설명 `coinHandler.sol` 컨트랙트와 스토리지 컨트랙트를 분석하였습니다. 예금과 출금이 안전하게 수행됩니다. 또한, 예대 마진 값 계산 시 이론적으로 underflow가 발생할 수 있으나 실제로는 발생할 수 없음을 확인하였습니다. recipent의 주소가 컨트랙트일 경우에 대해서도 분석하였습니다.

검증하였습니다 - TOKEN MANAGER ✓

분석 결과에 대한 추가적인 자료 및 코멘트

File Name : `tokenManager.sol`
File Path : `20201022-pilab-audit/2. truffle/contracts`
`└── marketHandler/tokenManager.sol`
MD5: ed135d65b7c2ab415d6ed4df4d600519

설명 `tokenManager.sol` 컨트랙트를 분석하였습니다. 안전하게 동작하는 것을 확인하였습니다. 하지만, Oracle을 통해 구해진 가격이 크게 변화했을 경우에 bZx의 공격과 같은 상황이 벌어질 수 있습니다. 가격의 변화에 따른 대비책을 설계하는 것도 방법입니다.

분석 결과

분석 결과는 심각도에 따라 Critical, High, Medium, Low, Note로 표현됩니다. Sooho는 발견된 모든 이슈에 대해서 개선하는 것을 권장합니다.

검사 결과 요약 및 결론

파이랩에서 개발한 BiFi 시스템의 컨트랙트는 이해하기 쉽게 명명되고 용도와 쓰임에 따라 잘 설계되어 있습니다. 인자 검증을 생략한 부분을 제외하고는 대부분 모범 사례를 따르고 있습니다. 코드 검사 결과, **이슈는 총 1개로 심각도 순서대로 Medium 1개입니다**. 꾸준한 코드 감사를 통해 컨트랙트의 안정을 도모하고 잠재적인 취약점에 대한 분석을 하는 것을 추천드립니다.

Project	20201022-pilab-audit
checksum	3808f009
# of Files	22
# of Lines	3,112

File Tree

```
20201022-pilab-audit/2. truffle/contracts
├── front/callProxy.sol
├── interestModel/interestModel.sol
├── interfaces
│   ├── IERC20.sol
│   ├── interestModelInterface.sol
│   ├── managerDataStorageInterface.sol
│   ├── marketHandlerDataStorageInterface.sol
│   ├── marketHandlerInterface.sol
│   ├── marketManagerInterface.sol
│   ├── oracleInterface.sol
│   ├── oracleProxyInterface.sol
│   ├── safeMathInterface.sol
│   └── tokenInterface.sol
├── marketHandler
│   ├── coinHandler.sol
│   ├── marketHandlerDataStorage
│   │   └── handlerDataStorage.sol Medium
│   └── tokenHandler.sol
├── marketManager
│   ├── liquidationManager.sol
│   ├── managerDataStorage
│   │   └── managerDataStorage.sol
│   └── tokenManager.sol
├── oracle
│   ├── oracle.sol
│   └── oracleProxy.sol
├── safeMath/safeMath.sol
└── tokenStandard/token.sol
```