SCOUTCHAIN SECURITY ASSESSMENT REPORT

JAN. 14 - JAN. 18, 2019

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 컨트랙트 분석에 최선을 다하였음을 밝히는 바입니다.

SOOHO 소개

SOOHO는 Audit Everything이란 슬로건으로 지속적인 보안을 위해 필요한 기술을 연구하고 서비스 합니다. 자체 취약점 분석기인 Aegis와 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

2019년 1월 14일에서 1월 18일까지 SOOHO는 ScoutChain의 스마트 컨트랙트에 대한 취약점 분석을 진행하였습니다. 감사 기간 동안 아래의 작업을 수행했습니다.

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

총 3명의 보안 전문가가 ScoutChain 컨트랙트의 취약점을 분석하였습니다. 참여한 보안 전문가는 Defcon, Nuit du Hack, 화이트햇, SamsungCTF 등 국내외의 해킹 대회에서 수상을 하고 보안분야 박사 학위의 학문적 배경을 가지는 등 우수한 해킹 실력과 경험을 가지고 있습니다.

SOOHO를 통해 알려진 약 3,000개 취약 코드 시그니쳐를 ScoutChain 컨트랙트에서 스캐닝하였습니다. 또한 이더리움 커뮤니티에서 주로 사용하는 유용한 보안 도구인 Oyente, Mythril, Osiris 등을 이용해 보다 복합적인 보안 취약점 검사 프로세스를 진행하였습니다.

총 2개의 취약점이 발견되었습니다. 발견된 취약점은 심각도 순서대로 Note 2 입니다. 또한, 대부분의 코드가 개발 모범 사례를 모두 준수하게 따르는 것으로 파악되었습니다. 모든 취약점은 배포된 컨트랙트의 소스 코드 기준 모두 해결되었음이 확인되었습니다. 꾸준한 코드 감사를 통해 Scoutchain 서비스의 안정을 도모하고 잠재적인 취약점에 대한 분석을 하는 것을 추천 드립니다.

분석 대상

1월 14일에서 1월 18일 동안 아래의 프로젝트를 분석하였습니다.

프로젝트명	scoutTkn	백서 정보	v1.1	컨트랙트 주소	0xb862C
Commit	3f72f9e	MD5	b1d08c0		
# of Files	9				
# of Lines	591				

주요 감사 포인트 및 프로세스

ScoutChain은 신뢰기반 참여형 탈중앙화 구인구직 플랫폼입니다. ScoutChain Token (이하 'SCT')는 스카우트체인 플랫폼에 참여할 수단이 되는 토큰입니다. SCT는 ERC20 표준에 맞춰 개발되었습니다. 추가로 특정 계정의 동결과 전체 토큰 거래 정지 기능이 존재합니다. 이에 따라 지갑의 소유권과 ERC 토큰에서 발생 가능한 취약점, 운영 과정에서 발생할 수 있는 해킹 시나리오에 대해 취약한지를 위주로 검증하였습니다.

예를 들어, 관리자가 아닌 임의의 유저가 토큰을 mint/burn 할 수 있는지, 검증 과정을 의도적으로 우회할 수 있는지, 레이스 컨디션에 대한 대비가 되어 있는지, 트랜잭션의 성공/실패에 대해 모두 잘 처리되는지, 업그레이드 시에 메모리 corruption이 발생하는지 등의 시나리오가 이에 해당됩니다. **단, 관리자에 의한 내부 해킹은 발생하지 않음을 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

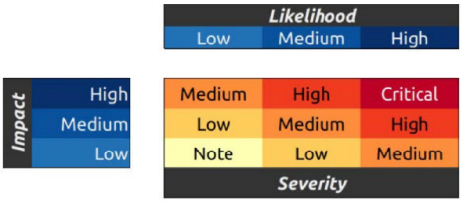
Exploit 분석과 PoC 작성은 다음을 목표로 합니다.

- 코드 실행을 통한 동적 분석
- 발견된 취약점의 악용을 통해 실제 금전적인 이득을 얻을 수 있는지 살펴봅니다. (e.g., 무한 인출)
- 발견된 취약점을 악용해 토큰 서비스에 악영향을 끼칠 수 있는지 살펴봅니다. (e.g., Mint)

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.



분석 결과

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

REDUNDANT CONDITION Note

분석 결과에 대한 추가적인 자료 및 코멘트

대상 파일 : SCTtoken.sol
파일 위치 : scoutTkn/contracts
└── SCTtoken.sol
MD5 : 495cccf1b1fb639532a2b5670b070035

```
95    function freeze(uint256 _value) public  returns (bool success) {
96      require(_balances[msg.sender] >= _value && _value > 0);
97
98      _balances[msg.sender] = SafeMath.sub(_balances[msg.sender], _value);
```

```
104   function unfreeze(uint256 _value) public returns (bool success) {
105     require(freezeOf[msg.sender] >= _value && _value > 0);
106
107     freezeOf[msg.sender] = SafeMath.sub(freezeOf[msg.sender], _value);
```

취약점이 아닌 가스 최적화 관점에서의 분석 결과입니다.

취약점 설명

freeze 함수의 조건문에 포함된 _balances[msg.sender] >= _value와 unfreeze 함수의 조건문에 포함된 freezeOf[msg.sender] >= _value 조건은 불필요합니다. 왜냐하면 각각의 아래에서 SafeMath를 통해 해당 조건은 방지되기 때문입니다. 따라서 불필요한 가스 사용을 줄이기 위해서 각 조건을 삭제하기를 추천드립니다.

OWNERSHIP CAN BE RELEASED Note

분석 결과에 대한 추가적인 자료 및 코멘트

대상 파일 : Ownable.sol
파일 위치 : scoutTkn/contracts/helper
└── Ownable.sol
MD5 : e1bc392b03129cad5bfe90f8003141b8

```
10  contract SCTtoken is ERC20, ERC20Detailed , Pausable{
```

```
10  contract Pausable is Ownable {
```

```
50      function renounceOwnership() public onlyOwner {
51          emit OwnershipTransferred(_owner, address(0));
52          _owner = address(0);
53      }
```

설명

SCTtoken은 Pausable을 상속받습니다. 또한, Pausable은 Ownable를 상속받습니다. Ownable에 정의되어 있는 함수 renounceOwnership은 _owner의 값을 초기화합니다. 이는 onlyOwner를 통해 접근 권한이 관리되고 있는 계정의 동결과 전체 토큰 거래 정지 기능을 영영 제어할 수 없게 합니다. 즉, SCTtoken에서 해당 함수를 사용할 수 없도록 재정의하거나 해당 함수를 Ownable에서 삭제 해야합니다.

추가 분석 결과

추가 분석 결과는 취약점은 발견되지 않았지만 취약점 분석 과정에서 중점적으로 살펴본 이슈들에 대한 내용을 포함하고 있습니다.

TOKEN WILL NOT MINT OR BURN ✓

분석 결과에 대한 추가적인 자료 및 코멘트

대상 파일 : SCTtoken.sol
파일 위치 : scoutTkn/contracts
└── SCTtoken.sol
MD5 : 495cccf1b1fb639532a2b5670b070035

```
10  contract SCTtoken is ERC20, ERC20Detailed, Pausable {
```

```
152     function _mint(address account, uint256 value) internal {
153         require(account != address(0));
154
155         _totalSupply = _totalSupply.add(value);
156         _balances[account] = _balances[account].add(value);
157         emit Transfer(address(0), account, value);
158     }
```

설명 백서의 22p 기준으로 SCT는 "추가로 발행되지 않습니다." 실제로, SCTtoken이 상속받는 ERC20에는 토큰의 추가 발행 기능에 해당되는 _mint 함수가 존재하지만 internal로 선언되었고 다른 함수에서 호출하지 않아 토큰이 추가 발행되지 않습니다. 따라서 백서 내용과 동일하게 구현되었습니다.

올바른 발행량 ✓

분석 결과에 대한 추가적인 자료 및 코멘트

대상 파일 : SCTtoken.sol
파일 위치 : scoutTkn/contracts
└── SCTtoken.sol
MD5 : 495cccf1b1fb639532a2b5670b070035

설명 SCTtoken.sol의 발행량이 백서에 명시된 양과 동일합니다.

총 10억개의 SCT가 발행됩니다.

안전합니다 ✓

분석 결과에 대한 추가적인 자료 및 코멘트

대상 파일 : SCTtoken.sol
파일 위치 : scoutTkn/contracts
└── SCTtoken.sol
MD5 : 495cccf1b1fb639532a2b5670b070035

설명 SCTtoken.sol의 주요 함수들이 적절한 접근 권한을 가지고 있습니다.

토큰 거래 정지 기능과 계정 정지 기능, 다른 유저로 권한 전달하는 기능 들에 적용되었습니다.

추가 분석 결과

추가 분석 결과는 취약점은 발견되지 않았지만 취약점 분석 과정에서 중점적으로 살펴본 이슈들에 대한 내용을 포함하고 있습니다.

안전합니다 ✓

분석 결과에 대한 추가적인 자료 및 코멘트

대상 파일 : `ERC20.sol`
파일 위치 : `scoutTkn/contracts/helper`
`└── ERC20.sol`
MD5 : 38f48ffa4f7efa4c53e4a11beb48e5aa

설명 `ERC20.sol`은 올바른 접근 권한을 가지고 모든 연산이 안전합니다.

안전합니다 ✓

분석 결과에 대한 추가적인 자료 및 코멘트

대상 파일 : `ERC20Detailed.sol`
파일 위치 : `scoutTkn/contracts/helper`
`└── ERC20Detailed.sol`
MD5 : c495589974b833f34dbe55c9c97daa3d

설명 `ERC20Detailed.sol`은 올바른 접근 권한을 가지고 모든 연산이 안전합니다.

안전합니다 ✓

분석 결과에 대한 추가적인 자료 및 코멘트

대상 파일 : `IERC20.sol`
파일 위치 : `scoutTkn/contracts/helper`
`└── IERC20.sol`
MD5 : f522419ba826d20c38fdc472447a8e75

설명 `IERC20.sol`은 올바른 접근 권한을 가지고 모든 연산이 안전합니다.

안전합니다 ✓

분석 결과에 대한 추가적인 자료 및 코멘트

대상 파일 : `Pausable.sol`
파일 위치 : `scoutTkn/contracts/helper`
`└── Pausable.sol`
MD5 : 916946c4ef28c9c5e3c7f5e9bdb33c93

설명 `Pausable.sol`은 올바른 접근 권한을 가지고 모든 연산이 안전합니다.

검증하였습니다 - MYTHRIL ✓

분석 결과에 대한 추가적인 자료 및 코멘트

설명 Mythril을 통해 발견한 취약점에 대해 모두 분석하였습니다. 분석 결과, 의미있는 취약점은 발견할 수 없었습니다.

검사 결과 요약 및 결론

ScoutChain의 컨트랙트 코드는 이해하기 쉽게 명명되고 용도와 쓰임에 따라 잘 설계되어 있습니다. 특히, 백서의 내용을 기반으로 하는 충실한 구현이 돋보였습니다. 취약점 검사 결과, 총 2개의 취약점이 발견되었습니다. 발견된 취약점은 심각도 순서대로 Note 2 입니다. 분석한 코드는 대부분 안전했으나 꾸준한 코드 감사를 통해 ScoutChain 서비스의 안정을 도모하고 잠재적인 취약점에 대한 분석을 하는 것을 추천드립니다.

프로젝트명	scoutTkn
Commit	2cbc587
# of Files	9
# of Lines	591

파일 구조

```
scoutTkn
└── contracts
    ├── Migrations.sol
    ├── SCTtoken.sol Note
    └── helper
        ├── ERC20.sol
        ├── ERC20Detailed.sol
        ├── IERC20.sol
        ├── Ownable.sol Note
        ├── Pausable.sol
        └── SafeMath.sol
```