META ARENA SECURITY ASSESSMENT REPORT

01. 17 - 02. 04, 2022

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 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

2022년 01월 17일에서 02월 04일까지 SOOHO는 META ARENA 프로젝트 컨트랙트에 대한 취약점 분석을 진행하였습니다. 감사 기간 동안 아래의 작업을 수행했습니다.

- SOOHO의 자체 취약점 검사기를 통한 취약점 탐지 및 결과 분석
- 공개된 분석기 Mythril의 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 META ARENA 프로젝트 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 취약 코드 시그니쳐를 META ARENA 프로젝트 컨트랙트에서 스캐닝하였습니다. 또한 이더리움 커뮤니티에서 주로 사용하는 유용한 보안 도구인 Mythril을 이용해 보다 복합적인 보안 취약점 검사 프로세스를 진행하였습니다. 추가적으로 SOOHO의 VeriSmart를 이용해 Arithmetic 연산에 대해 형식 검증하였습니다.

발견된 취약점은 심각도 Note 2 입니다. 꾸준한 코드 감사를 통해 서비스의 안정을 도모하고 잠재적인 취약점에 대한 분석을 하는 것을 추천 드립니다.

분석 대상

01/17부터 02/04까지 분석을 위해 전달받은 프로젝트는 다음과 같습니다.

프로젝트명	klaymetacontract.zip
MD5	59dc75d4...
# of Files	15
# of Lines	4585

주요 감사 포인트 및 프로세스

META ARENA 프로젝트는 KIP-7, KIP-17 프로토콜을 활용한 게임 & NFT 플랫폼입니다. 이에 따라 프로토콜, 게임 로직에서 발생될 수 있는 취약점과 업그레이드 시에 발생할 수 있는 해킹 시나리오에 대해 취약한지를 위주로 검증할 예정입니다.

예를 들어, 관리자가 아닌 임의의 유저가 검증 과정에 참여할 수 있는지, 토큰을 mint/burn 할 수 있는지, 레이스 컨디션에 대한 대비가 되어 있는지, 트랜잭션의 성공/실패에 대해 모두 잘 처리되는지 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.

Likelihood		
Low	Medium	High

Impact			
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.

가스 소모 최적화 Note

분석 결과에 대한 추가적인 자료 및 코멘트

대상 파일 : META.sol, MST.sol, NFT.sol
파일 위치 : klaymeta/assets/

```
└── META.sol (MD5: da30958c...)
└── MST.sol  (MD5: b0302340...)
└── NFT.sol  (MD5: a8f55b24...)
```

Meta.sol, MST.sol

```
90      uint8 private _decimals = 18;
```

```
97      bytes4 private _KIP7_RECEIVED = 0x9d188c22;
```

NFT.sol

```
226     // Token name
227     string private _name;
228
229     // Token symbol
230     string private _symbol;
```

설명 constant state variable은 일반적인 state variable보다 가스 소모가 적기 때문에 고정된 상태 변수에 대해 constant 키워드를 사용 고려

(RESOLVED) 난수 예측성

KlayMeta에서는 랜덤값 연산에 사용되는 해시에 대해 공격자가 분포를 파악하기 어렵게 하도록 빈번하게 변경하고 있음을 확인함.

대상 파일 : Summon.sol
파일 위치 : klaymeta/avatar/

```
└── Summon.sol
```

MD5 : 20a011eac7ae983ec9baaed5f1aa3744

```
500     function getRand(uint min, uint max) internal returns (uint) {
501         require(min >= 0 && min <= max);
502
503         uint256 r = uint256(keccak256(abi.encodePacked(randomHash, OracleLike(oracle).hash(), randomNonce, block.timestamp)));
504         randomNonce = randomNonce + 1;
505
506         return r.mod(max - min + 1).add(min);
507     }
```

요약

- Keccak256과 같은 해시 함수는 정해진 블록을 기준으로 입력에 대해 Uniform 한 확률 분포를 갖는 출력을 갖도록 설계
- Keccak 등과 같은 해시 함수를 통해 얻은 결과 값으로 % (Modulo) 연산을 하게 되면, 해당 범위에 대해 Uniform 한 확률 분포를 갖을 수 있다는 가정이 깨지게 되어 특정 구간에서 확률 분포가 기울어진 결과가 나올 수 있음
- 공격자는 확률 분포가 높은 숫자로 발생되는 시점을 예측하여 높은 능력치의 아바타를 높은 확률로 Summon 할 수 있음

분석 결과

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

랜덤 정의

난수 r 은 다음과 같이 정의 (getRand 기준)

$$H_1 = keccak, H_2 = blockhash, \oplus = concat$$

$$r = H_1(randomhash \oplus oraclehash \oplus randomNonce \oplus block.timestamp)$$

- random hash의 경우, summon 또는 specialSummon의 행위가 발생될 때 변경 (연산 가능)

$$randomHash = H_2(N_{block} - 1) + msg.sender$$

- oracle hash의 경우 경우, 초기 상태를 기반으로 설정 (연산 가능)

$$oracleHash = H_1(H_2(N_{block} - 1) + msg.sender)$$

- randomNonce의 경우, 1씩 증가 (상태값 확인 가능)
- block.timestamp (매 초마다 변경)

공격 시나리오

1. 공격자는 다른 사용자의 액션에 의해 hash가 변경되지 않고 미리 계산된 값을 유효하게 사용하기 위해 사용자들이 활동이 적인 시간을 기준으로 시작
2. 미리 계산된 값과 timestamp를 연산하여 미래에 높은 분포가 나타나는 시간대를 찾고 해당 시간에서 summon
3. 만약 공격자가 찾은 시간 이전에 hash가 변경되면 (1) 부터 다시 시작

```
[2022-01-24 16:32:15] 1e937edc1e4d0720e0df524037d4c6881118124e85160d71dd2ca57cdaf21873 => 39
[2022-01-24 16:32:16] ae87932b6949dd947761870001807a68d592e5b3e4eb5f33e6279a964c6437a9 => 1
[2022-01-24 16:32:17] 563b7494a0e3313f60e5b7278164a15762f78d4be66c3a2558c9e20cb813b9a0 => 56
[2022-01-24 16:32:18] ba4592cdfaae9fdad4b66aecd55e11a18bc6904509fe6822679da379a82adb7b => 95
[2022-01-24 16:32:19] 65a7137317533ef8923873fe6169c5c5df24957243297a83f0df8b6f89bae97e => 98
[2022-01-24 16:32:20] 6b364cf3caa8a4cd518dae06f587b31163c630a7ee2575e235938c47d1fa3397 => 99
        [-] found it
[2022-01-24 16:32:21] 7feafe0b312af953872856db66e1f18b146dc211b048b3ac44c586fc82ff878e => 58
[2022-01-24 16:32:22] 858e71fde49278f000c12cbdd140d9dff56c481dc315e2de7dd12fe4c3de8c65 => 9
[2022-01-24 16:32:23] e39b15d68df8a33c1501b1e0f0740f770c02af6af51ff3a0499f6df4106de5fc => 32
[2022-01-24 16:32:24] 33d553be248deccf9f284f5975b86ded7267c4bba86e576d57854fddb21a3efa => 86
[2022-01-24 16:32:25] aa9a8f337acdb65c15e199a9ae9b53a2c8db37423ce61a3801dc7ef8f77b14db => 67
[2022-01-24 16:32:26] 3d56c3739a6acea4104d2b2b0d3a72139e8b5a37f6267b1d5dfca9f36d427812 => 38
```

보안 방안

- 솔리디티 단독으로 랜덤성이 높은 랜덤 생성기를 만드는 것은 현실적으로 불가능
- 문제점이 발생되는 이유는 random hash가 변경되는 기간이 공격자가 원하는 값의 분포를 얻기 충분한 시간
- 따라서, random hash 들을 주기적으로 변경하여 공격자의 예측 기간을 줄이고 공격 비용을 높이는 것을 추천

분석 결과

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

함수 재진입 Note

분석 결과에 대한 추가적인 자료 및 코멘트

대상 파일 : Swap.sol
파일 위치 : klaymeta/
└── Swap.sol (MD5: 1aa84fc0...)

```
 99          if (_token1 == address(0))
100          {
101              (bool sent, ) = (msg.sender).call.value(afterBalance.sub(beforeBalance))("");
102              require(sent, "faild to send ether");
103          }
```

설명

msg.sender에 대해 raw level call을 수행할 때, 만약 msg.sender가 contract라면 함수 재진입 가능성이 존재합니다. swap 함수 재진입을 보호하는 코드를 추가하거나 raw level call이 아닌 transfer 를 통해 함수에 재진입을 할 수 없도록 하는 방법을 권장합니다.