Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.32, NO.3, Jun. 2022 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2022.32.3.477

이벤트 소싱 패턴과 블록 체인을 활용한 모바일 게임 클라이언트 메모리 조작 방지 방안

박 지 훈,[1†] 박 영 호[2‡]
[1,2]세종사이버대학교 대학원 (대학원생, 교수)

Preventing Mobile Game Client Memory Manipulation Based on Event Sourcing Patterns and Blockchain.

Jihun Park,[1†] Young-Ho Park[2‡]
[1,2]Sejong Cyber University (Graduate student, Professor)

요 약

본 연구는 클라이언트 단에서 메모리 조작의 취약점에 대응하기 위한 방법으로 이벤트 소싱 패턴과 블록체인을 활용한 방안을 제시하고자 한다. 해당 방안의 검증을 위해 메모리 조작 애플리케이션이 구동되는 방법을 분석하고 테스트 애플리케이션을 제작하여 메모리 조작 방지 방안을 적용했을 때 성능을 비교 분석하였다. 분석 결과 주요 데이터를 하나의 메모리에 저장하여 XOR 연산을 하는 방안보다 메모리의 사용량이 증가하였지만, 게임의 성능에 큰 영향을 주지 않으면서 메모리 조작 프로그램의 조작을 방지할 수 있었다.

ABSTRACT

This study aims to present a method using event sourcing patterns and blockchain as a way to cope with vulnerabilities in memory manipulation at the client level. To verify the plan, the method of running the memory operation application was analyzed, and the performance was compared and analyzed when the memory operation prevention plan was applied by fabricating a test application. As a result of the analysis, the usage of memory increased compared to the method of XOR operation by storing major data in one memory, but it was possible to prevent the operation of the memory operation program without significantly affecting the performance of the game.

Keywords: Mobile Game, Memory Hacking, Anti-Cheat, Event Sourcing, Block-Chain

I. 서 론

모바일 게임 시장의 발전에 따라 모바일 게임에 다양한 취약점이 발생하고 있다. 모바일 게임 보안 동향[1]에 따르면 모바일 게임 및 서비스를 대상으로 한 공격들이 증가하고 있고, 발생 지점에 따라 위협 요소의 분류 및 보안 기법과 한계점이 존재함을 알 수 있다.

위협 요소 중 클라이언트 단에서 발생하는 메모리 조작의 경우 공학적 지식 없이, 특정 메모리 조작 프로그램을 사용하는 것만으로도 게임 내 주요한 데이터 값을 조작할 수 있다. 게임 내 메모리 조작이 발생할 경우 개발사에 대한 사용자의 신뢰도 및 매출 하락의 피해가 발생하는데, 일례로 모바일 게임 에픽세븐에서 악의적인 유저가 게임 캐릭터의 공격력을 비정상적으로 높이는 메모리 조작을 통해 스테이지를 클리어한 뒤 상위 랭킹에 올라가는 이슈가 발생하였다[2]. 이 사건으로 에픽세븐의 유저들이 개발사에

Received(03. 02. 2022), Modified(04. 29. 2022), Accepted(05. 02. 2022)
† 주저자, everyday.devup@gmail.com
‡ 교신저자, youngho@sjcu.ac.kr(Corresponding author)

해명을 촉구하는 성명을 발표하고 환불 및 불매 운동을 펼쳐 개발사가 피해를 보았다.

클라이언트 메모리 조작은 유저의 접근성이 용이한 것에 비해, 취약점이 발생할 경우 개발사의 피해가 크기 때문에 개발사에서 다양한 방법으로 메모리 조작을 방지하려 하지만 한계점이 존재한다.

1.1 외부의 전문 게임 보안 솔루션 활용

전문 보안 솔루션은 게임의 위변조 방지, 루팅 탐지, 메모리 공격 탐지 등의 서비스를 제공하여, 개발사는 게임 내부 시스템과 컨텐츠 개발에 집중을 할 수 있다[3]. 솔루션을 활용할 경우 메모리가 조작되는 것을 탐지하여 게임을 종료시키거나 개발사에서 처리할 수 있도록 에러 콜백을 받을 수 있는데, 게임마다 시스템이 상이하기 때문에 메모리 조작 자체를 방지하는 솔루션을 받기에는 어려운 점이 있다. 보안 업체의 비용을 지불해야 하는 점과 특정 해킹 이슈가 발생했을 때 보안 솔루션에서 대응이 될 때까지 기다려야 하는 한계점도 존재한다.

1.2 개발사의 게임 내 메모리 암호화 적용

외부 보안 솔루션을 사용하더라도, 메모리 조작프로그램에 따라 보안 솔루션의 탐지를 회피하는 경우가 발생할 수 있다. 보안 솔루션을 회피할 경우를 대비하거나, 보안 솔루션을 사용하지 않았을 경우 개발사에서 메모리 조작을 방지해야 할 필요성이 있다. 개발사는 메모리 조작을 방지하기 위해 주요 게임 데이터를 XOR 연산, 분산 저장, 암호화 등의 방법을 이용한다. 하지만 보안 강도에 비례하여 게임 성능에 영향을 줄 수 있기 때문에, 강도 높은 보안을 사용하는 데 한계가 존재한다[4].

1.3 게임 서버의 데이터 검증

주요 데이터의 경우 클라이언트에서 게임 서버와의 통신 시 게임 서버에서 데이터베이스의 데이터와 비교, 검증 후 사용하는 방법으로 메모리 조작을 방지할 수 있다. 클라이언트의 값을 참조하지 않기 때문에 메모리가 조작되어도, 게임에 영향을 주지 않지만, 게임의 특성에 따라 클라이언트의 모든 데이터를 검증하는 것이 불가능할 수 있고, 데이터 검증 과정에서 데이터베이스 접근이 빈번할 경우 부하가 발생할 수 있다[5].

본 연구에서는 메모리 조작을 방지하기 위한 방법으로 클라이언트 단에서 이벤트 소싱 패턴과 블록체인을 활용한 주요 데이터 관리 방법을 제시함으로써 클라이언트 단의 게임 내 데이터 암호화 한계점을 개선하고 효율화하는 방안을 제시하고자 한다. 본 연구에서는 이벤트 소싱 패턴과 블록체인을 활용한 방안을 EBC(Event Block Chain)라 하고, EBC에서 사용하는 이벤트 블록을 EB(Event Block)라고 명명한다.

II. 관련 연구

2.1 모바일 게임 메모리 조작 프로그램

모바일 디바이스의 운영체제는 안드로이드, 아이오에스, 카이오에스 등 다양하지만, 게임 개발사에서는 많은 유저를 확보하기 위해 게임을 멀티 플랫폼으로 제공하며 계정으로 유저의 데이터를 분리한다. 폐쇄적인 아이오에스 운영체제 환경보다 오픈 소스로 제공되는 안드로이드 운영체제 환경은 해킹 프로그램에 취약한 편이다[6]. 안드로이드 및 윈도우 운영체제를 대상으로 한 치트 엔진, 게임 가디언 등의 해킹 프로그램[7,8]은 웹 검색을 통해 설치 및 사용 방법을 익힐 수 있어 유저의 접근성이 좋은 편이다. 아이오에스 기기를 소유한 유저도 에뮬레이터를 사용하여 메모리를 조작 후 플레이한 데이터를 아이오에스 기기에 접속하여 데이터를 연동할 수 있기 때문에 메모리 조작 프로그램은 플랫폼과 관계없이 대응할 필요가 있다.

2.1.1 메모리 조작 프로그램의 구동 방식

게임 가디언과 치트 엔진의 기본적인 구동 방식을 동일하며, 인터페이스와 추가적인 검색 기능의 차이만이 존재한다. 두 프로그램의 기본 구동 방식은 다음과 같다.

가. 현재 실행 중인 게임에서 UI(User Interface)에 보이거나, 유저가 알고 있는 값을 검색한다. 메모리 조작 프로그램은 유저가 검색한 값을 가진 메모리를 검색하여 메모리 리스트를 생성한다.

나. 해당 값을 증감/가감하면서 메모리 리스트에서 증감/가감/변경되지 않은 옵션을 통해 옵션에 해당하는 값을 가진 메모리만을 리스트에 남긴다.

다. 메모리 조작 프로그램의 메모리 리스트가 조작할 수 있는 수만큼 적어질 때까지 가, 나의 과정을 반복한다.

라. 메모리 조작 프로그램의 메모리 리스트에 남겨진 메모리의 값을 변경하면서 게임에 반영되는지 확인한다.

게임 가디언의 경우 XOR 된 메모리값을 검색하는 추가적인 기능을 가지고 있다. 게임 가디언은 XOR의 사용되는 키값이 인접하여 정의된다는 가정을 바탕으로, 검색하는 값을 가진 메모리에서 특정 바이트만큼 떨어진 메모리의 값으로 XOR 연산하여 검색하는 방식을 사용한다.

2.1.2 메모리 조작 대응 방식

게임 내 주요 데이터의 메모리 조작에 대응하는 방법은 게임 개발사의 정책에 따라 다를 수 있다. 구글 플레이의 출시 된 게임들을 치트 엔진을 사용하여 메모리 조작했을 때, 게임별 대응 방식을 비교하면 다음과 같은 대응 방식으로 분류할 수 있다.

가. 클라이언트 메모리 조작을 허용하는 정책으로 게임 서버에서 클라이언트의 주요 데이터를 검증하여 데이터의 이상이 있으면 게임 플레이 자체를 무효화 시키는 방법이 있다.

나. 클라이언트 메모리 조작을 탐지 및 로그를 쌓아 특정 유저의 플레이를 제한시키는 방법이 있다.

다. 클라이언트의 주요 데이터를 암호화하여 해킹 프로그램으로 조작이 어렵게 대응하는 방법이 있다.

가, 나의 대응 방식은 메모리 조작이 발생한 후의 대응 방식으로 이상 데이터를 탐지하는 방법이 중요하고, 다의 경우 주요 데이터의 암복호화 시 게임 플레이에 영향을 주지 않아야 하는 점이 중요하다.

2.1.3 게임 내 변수 암복호화에 사용되는 유료 에셋

기존의 게임 클라이언트에서 주요 데이터를 암복호화하는 방식을 분석하기 위해, 모바일 게임 제작 시 사용되는 유니티 게임 엔진의 유/무료 에셋을 확인하였다. 2021 Gaming Report[9]에 따르면 모바일 게임 제작 시 유니티 엔진을 사용하는 비율이 61%로 유니티는 대표적인 모바일 게임 엔진이라 할 수 있다. 유니티에서는 개인 또는 기업에서 판매하는 에셋을 에셋 스토어를 통해 제공하는데 본 연구에서 확인한 보안 에셋은 유니티에서 추천하는

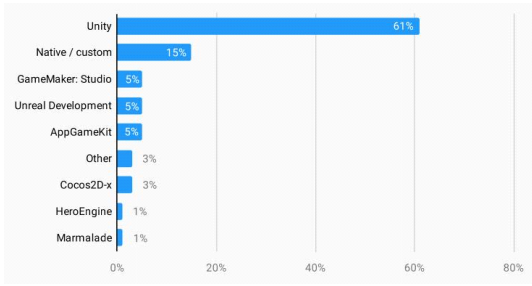

Fig. 1. Game engine share in mobile game production

Anti-Cheat Toolkit 2021[10]과 Anti Cheat Pro[11]로 두 에셋 모두 메모리 조작을 방지하는 기능을 제공하고 있다.

위의 보안 에셋은 공통으로 메모리 조작을 방지하기 위해 XOR 연산을 사용하고 있다. 주요 데이터를 정의할 때 XOR 연산에 사용할 랜덤한 키를 생성하고 해당 키를 통해 주요 데이터의 읽기/쓰기를 처리한다. 페이크 벨류에는 암호화되지 않은 값을 넣어 메모리 조작 프로그램의 탐색을 속일 수 있도록 처리되어 있다.

유료 에셋에서 데이터를 암호화할 때 XOR 연산을 사용하는 이유는, XOR 연산이 CPU 연산으로 수행 속도가 빠르고, XOR로 연산 된 값을 같은 키로 다시 XOR 연산을 하면 원래 값을 얻을 수 있는 특징이 있기 때문이다.

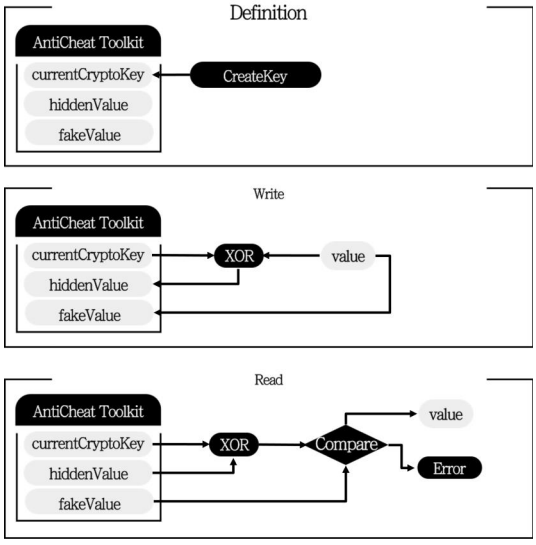

Fig. 2. AntiCheat Toolkit Flow Chart

$$event\,value \oplus random\,key = value \quad (1)$$
$$value \oplus random key = event\,value$$

게임 가디언의 경우 XOR로 암호화된 데이터를 검색하는 기능이 존재하는데, 개발자가 데이터와 키를 묶어 구조화하거나 인접하여 정의하는 경우가 많은 점을 이용하였다. XOR로 암호화된 데이터와 인접한 메모리를 순회하면서 키를 찾아 XOR로 암호화된 데이터를 조작한다. 이를 막기 위해 유료 에셋에서는 XOR 연산에 사용된 키값을 주기적으로 변경하는 방안을 제시하였지만, 키 변경 주기에 따라 검색이 가능한 시점이 존재하고, 키값 변경에 따른 부하가 발생하는 한계점이 있다.

2.2 이벤트 소싱 패턴

마틴 파울러가 제시한 이벤트 소싱 패턴은 데이터의 변경에 대한 모든 사항을 이벤트로 저장하여, 일련의 이벤트를 재생했을 때 특정 시점의 데이터를 얻는 개발 디자인 패턴이다[12]. 기존에 데이터를 저장하는 방식이 최종 데이터를 데이터베이스에 갱신하여 보관하는 반면, 이벤트 소싱 패턴은 데이터의 변화를 이벤트로 만들어 연속된 이벤트 묶음으로 데이터베이스에 보관한다[13]. EBC는 이벤트 소싱 패턴을 사용하여 게임의 주요 데이터를 관리할 경우 게임 UI에 보이는 데이터와 이벤트로 관리되는 데이터가 다르기 때문에 메모리 조작 프로그램을 통해 주요 데이터를 검색하기 어렵게 된다. 데이터가 조작된 경우 이벤트의 재생을 통해 값을 검증하고 복구할 수 있는 장점도 있다. 이벤트 소싱 패턴은 데이터를 얻기 위해 저장된 모든 이벤트를 재생하는 과정에서 이벤트의 개수에 비례하여 시간이 소요되는 단점이 있는데 스냅샷과 CQRS(Command Query Responsibility Segregation) 을 사용하여 해당 단점을 보완할 수 있다[14]. 마이크로 서비스 아키텍처를 구현하는 방법으로 사용되는 이벤트 소싱 패턴은 넷플릭스, AWS(Amazon Web Services)에서 활용되고 있다[15].

2.3 블록 체인

데이터의 무결성 확인을 위해 해시 함수와 데이터 체이닝을 사용하는 블록체인[16]을 활용하였다. 이벤트 소싱 패턴으로 만들어진 이벤트를 연결할 때 해시값을 추가하여 EB를 만들고, 연결된 EB들의 해시값을 통해 데이터의 신뢰성을 확보하였다[17, 18]. 데이터 체이닝 없이 하나의 데이터에 값을 XOR로 암복호화하는 방법의 경우 메모리 조작 프로그램으로 XOR 된 값을 변경했을 때 비교 검증할 방법이 없는 한계점이 존재하는데, EBC는 EB를 데이터 체이닝으로 구성하여 메모리 조작 프로그램이 XOR로 암호화된 EB의 값을 조작한 경우 조작된 EB를 기점으로 다음 EB의 해시값의 변경이 발생한다. 해당 특성으로 메모리 조작을 탐지할 수 있는 장점이 있다.

블록체인의 경우 블록에 데이터의 변경이 없는 형태로 구성되지만, 단말기의 제한된 메모리 환경에 최적화하기 위해 제한된 EB의 생성 및 기존 블록을 재사용할 수 있도록 구성하였다.

III. 이벤트 소싱 패턴과 블록 체인을 활용한 메모리 조작 방지 방안

EBC는 이벤트 소싱 패턴과 블록체인의 특성을 이용하여 클라이언트 단에서 메모리 조작을 방지하는 방안으로 이벤트 소싱 패턴의 이벤트를 EB로 만들고 연결하여 블록체인의 특성을 가지도록 한 디자인 패턴이라 할 수 있다.

EBC는 이벤트 소싱 패턴의 단점을 개선하기 위해 CQRS를 사용하여 최종 데이터의 갱신을 처리하는 읽기 저장소와, 이벤트가 발생했을 때 이벤트를 저장하는 EB 저장소로 구성된다. EBC가 생성될 때 XOR에 사용할 키를 읽기 저장소와 EB 저장소의 각 EB에서 생성하며, EBC가 삭제되기 전까지 키값을 유지하여 키 생성에 대한 부하를 줄였다. 읽기 저장소의 키와 EB의 키는 메모리 조작 프로그램이 값을 직접적으로 검색하는 것을 방지하기 위해

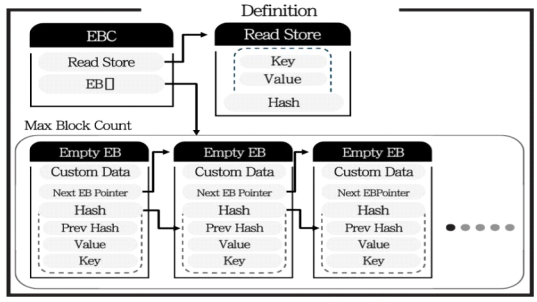

Fig. 3. Structure of EBC

XOR에 사용되는 값으로, 읽기 저장소와 EB에 값을 읽거나 쓸 때 해당 키를 사용하여 암복호화를 할 수 있도록 하였다. EB는 EBC가 생성될 때 최대 개수만큼 배열로 정의하여 연속된 메모리에 할당될 수 있도록 하였다. 이를 통해 EB 저장소의 이벤트 재생 시 캐시 미스를 줄일 수 있도록 최적화하였다.

3.1 EBC의 읽기 동작

EBC에서 데이터를 읽는 절차는 다음과 같다. 먼저 읽기 저장소의 값이 조작되었는지 확인하기 위해 키와 벨류로 해시값을 계산하여 저장된 해시값과 비교한다. 해시값을 비교했을 때 같은 경우 키와 벨류를 XOR 하여 원래의 값을 반환한다.

해시값이 달라진 경우 저장된 EB 저장소를 처음부터 재생하여 원래 값을 복구한다. 이벤트 재생을 위해 저장된 값 타입의 초기 값을 가진 임시 변수를 하나 설정한다. EB 저장소에 저장된 EB를 처음부터 재생하면서 EB 저장소에 각 EB가 조작되었는지 확인한다. 순차적으로 각 EB의 해시값을 계산하고, 다음 EB에 저장된 이전 해시값과 비교한다. 해시값을 비교했을 때 같을 경우 EB의 키와 벨류를 XOR 하여 임시 변수에 해당 값을 더한다. EB 저장소의 모든 EB를 검증하고 값을 더하여 최종적으로 임시 변수에 값이 모두 더해지면, 읽기 저장소의 값을 갱신하여 값을 복구한다. 만약 EB 저장소의 EB를 순차적으로 검증하던 중 해시값이 다른 경우에는 에러를 반환하여 조작을 탐지한다. 에러가 발생할 경우 게임의 특징에 따라 현재 게임 상태를 무효화하거나 EB 저장소의 내역을 게임 서버에 전달하여 비정상적인 유저의 판단 및 제제의 근거로 사용할 수 있다.

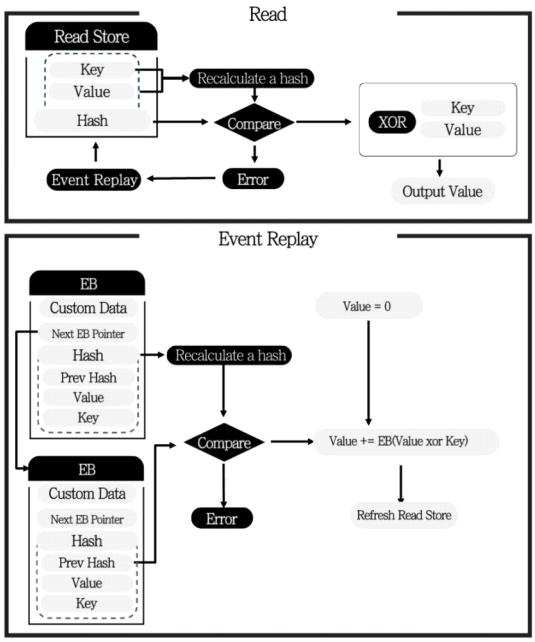

Fig. 4. Read Process And Event Play

3.2 EBC의 쓰기 동작

이벤트가 발생하면 사전에 만들어진 EB 저장소에서 빈 EB를 할당해준다. 빈 EB가 없을 경우에는 스냅샷 기능이 적용되어 빈 EB를 만든 후 할당해준다. 해당 키값과 이전 EB의 해시값, 발생한 이벤트의 값으로 해시값을 만들어 다음 EB에 저장한다. 사용자 정의 데이터에는 이벤트가 발생한 컨텐츠, 시점 등 게임 내 필요한 값을 넣어준다.

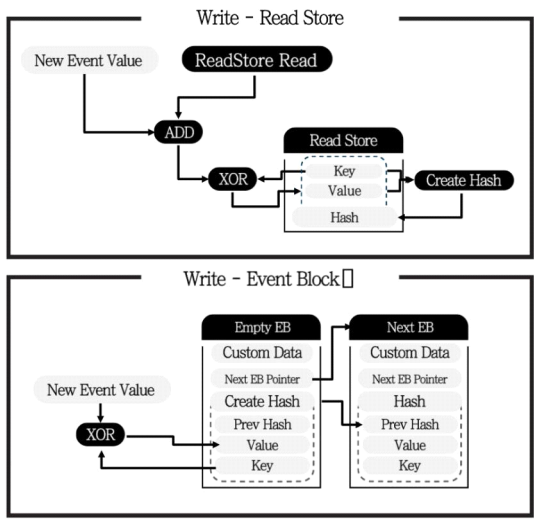

Fig. 5. Write Process

3.3 EBC의 스냅샷(Snapshot)

스냅샷이란 2개 이상의 EB를 가진 EBC에서 이벤트가 추가될 때 빈 EB가 없으면, 현재까지 쌓인 EB를 재생하여, 최종값을 가장 앞에 EB에 복사한 뒤 남은 공간을 빈 곳으로 만들고 새로운 이벤트를 추가하는 것을 말한다. 모바일 기기의 경우 메모리의 사용량이 제한되어 EB를 계속 추가할 수 없고, EB가 많아질수록 이벤트를 재생하는 데 시간이 소요되

는 단점을 보완하기 위하여 사용한다.

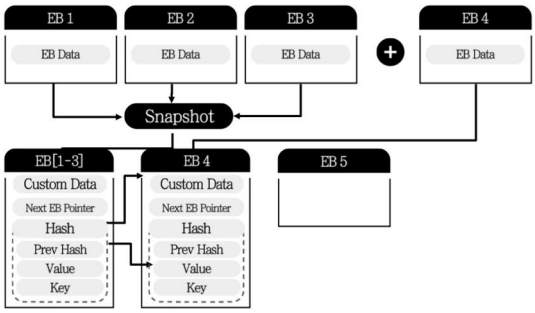

Fig. 6. Snapshot Flow Chart

IV. 성능 평가

4.1 실험 환경

메모리 조작 방지 방안에 따른 성능을 비교하기 위해 Unity 게임 엔진과 C# 언어를 사용하여 애플리케이션을 제작하였다. 메모리 조작 방지를 하지 않았을 때, Unity 에셋 스토어에서 유료로 판매되고 있는 Anti-Cheat Toolkit를 사용했을 때, EBC를 사용했을 때로 방안을 나누어 각 방법에 따라 메모리 조작 여부 및 데이터를 읽고 쓸 때의 시간 소요를 측정하였다. 블루스택 에뮬레이터에 테스트 애플리케이션을 설치하고 게임 가디언 프로그램을 사용하여 메모리 조작을 시도하였다. PC와 에뮬레이터 환경은 CPU : Intel(R) Core(TM) i5-6500 CPU @ 3.2GHz, Ram : 16.0GB, OS : Windows 10 Pro 64bit ,블루스택 5.0의 CPU 설정은 중간(2코어), 메모리 설정은 중간(2GB), 프레임 속도 60으로 설정하였다.

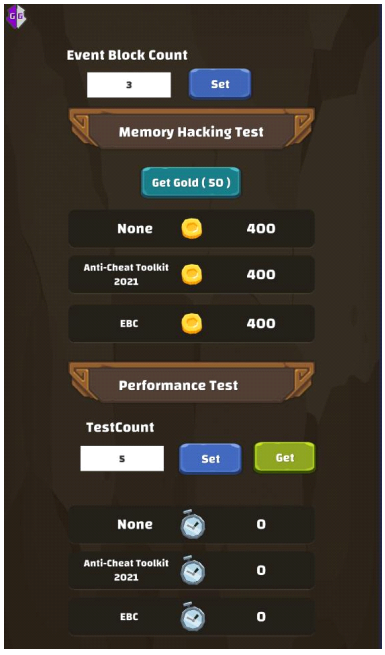

Fig. 7. Test application for EBC validation

4.2 메모리 조작 실험 결과

블루스택 에뮬레이터에서 루팅 모드를 설정한 뒤 게임 가디언을 사용하여 메모리 조작을 시도하였다. 테스트 애플리케이션의 UI에 보이는 값을 검색한 뒤 GetGold버튼을 통해 골드의 수량을 변경하면서 메모리를 검색한 뒤 검색된 메모리의 값을 조작하였다.

4.2.1 메모리 방지를 하지 않은 경우

메모리 조작 프로그램을 통해 용이하게 해당 값이 있는 메모리를 검색할 수 있었고 메모리의 값 조작도 가능하였다.

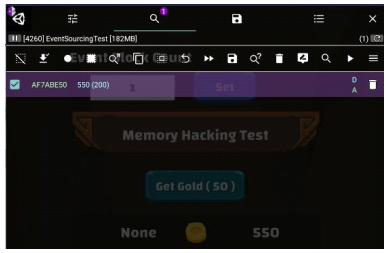

Fig. 8. Easy to search and manipulate memory

4.2.2 Anti-Cheat Toolkit을 사용한 경우

값 검색의 경우 검색된 메모리가 없었지만, 게임 가디언의 XOR 검색을 사용하여 검색할 경우 메모리를 찾을 수 있었다. Anti-Cheat Toolkit의 경우 값을 암호화하는 키값이 4바이트 떨어진 곳에 만들어지기 때문에 XOR 검색 옵션을 UI에 보이는 값 X4를 사용하여 검색하였다. 값을 조작할 때도 변경할 값 X4를 하면 XOR 된 값으로 메모리 조작이 가능하다.

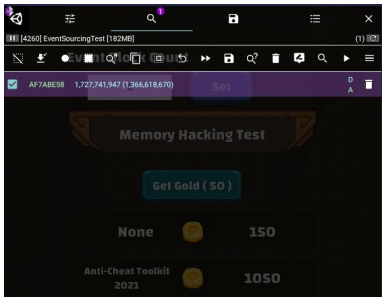

Fig. 9. XOR allows memory retrieval and manipulation

4.2.3 EBC를 사용한 경우

EBC 읽기 저장소의 있는 값은 XOR 검색을 통해 메모리를 검색할 수 있다. 하지만 Anti-Cheat Toolkit과 달리 메모리 조작이 발생하여도 메모리 조작의 탐지 및 원래 값으로 복구가 가능하였다. EB의 있는 값은 이벤트 개수에 따라 메모리 검색이 가능한 경우가 있을 수 있지만, 블록체인의 해시값 비교를 통해 EB의 데이터 무결성을 검증하여 메모리 조작을 탐지할 수 있다.

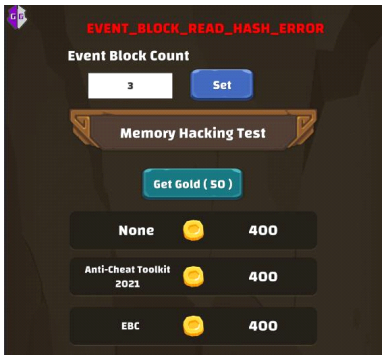

Fig. 10. If EBC is used, it is possible to detect memory manipulation and restore it to its original value.

4.2.4 메모리 조작 실험 시사점

Anti-Cheat Toolkit과 EBC 모두 값을 검색했을 때는 메모리를 찾을 수 없지만, XOR 검색을 할 경우 키값의 위치를 알고 있다면 메모리를 찾을 수 있다. Anti-Cheat Toolkit의 경우 XOR 된 값을 조작하면 실제 값도 변경되지만, EBC의 경우 XOR 된 값 외에 해시값 비교를 통해 조작 시도를 탐지할 수 있고 EB 저장소를 통해 값을 복구 할 수 있다.

4.3 읽기/쓰기 성능 평가 결과 및 시사점

블루스택 환경에서 데이터를 쓰거나 읽을 때의 소요되는 시간을 측정하여 성능을 비교하였다. 측정의 단위는 틱을 사용하여 정확도를 높였다. 1틱은 100 나노초로 1밀리초는 1000틱이다. 모바일 게임의 경우 초당 60프레임이 나올 수 있는 것을 목표로 개발을 진행하는데, 한 프레임당 16666 틱 이하로 시간이 소요되야 한다.

메모리 방지를 하지 않은 데이터보다 메모리 방지를 할 경우 데이터의 읽기/쓰기에 걸리는 시간의 차이가 30~50배 정도 차이가 나는 것을 알 수 있는데, 모든 데이터의 메모리 방지를 할 경우 성능에 영향을 줄 수 있음을 알 수 있다. 게임의 특성에 따라 중요 데이터를 선별하여 메모리 방지를 할 필요성이 있다. Anti-Cheat Toolkit을 사용할 때와 EBC를

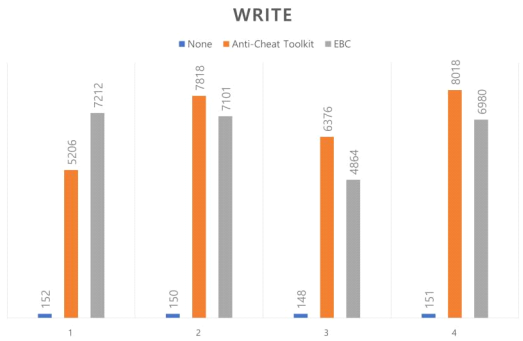

Fig. 11. Time taken for 10000 writes per one time (ticks)

Fig. 12. Time taken for 10000 reads per one time (ticks)

사용할 때 데이터를 쓸 경우에는 평균적으로 5% 정도의 성능이 향상되었고, 데이터를 읽을 경우 EBC의 성능이 Anti-Cheat Toolkit 보다 평균적으로 50%의 성능 향이 있었다. 하지만 EBC의 경우 EB의 개수만큼 메모리 사용량이 증가하기 때문에 데이터의 중요성과 게임 내 메모리 사용량을 고려하여 개수를 정할 필요가 있다. EBC는 읽기 저장소에서 자료형의 3배, EB 저장소의 각 EB마다 자료형의 3배, 다음 블록을 가리키는 포인터 4바이트의 메모리를 사용한다. int 자료형에 10개의 EB를 가진 EBC를 사용할 경우 읽기 저장소에서 12바이트, 쓰기 저장소에서 160바이트 총 172바이트로, 단일 XOR을 사용했을 때 8바이트의 20배의 메모리를 사용한다.

V. 결 론

모바일 게임의 다양한 취약점 중 하나인 메모리 조작은 사용자가 랭킹을 올리거나, 스테이지를 클리어하거나, 재화 획득을 빠르게 하는 등 사적인 이득을 얻는 데 사용되지만, 이에 따라 개발사는 일반 사용자의 신뢰를 잃고, 매출이 하락하는 등의 피해를 보게 된다. 본 연구는 메모리 조작 프로그램의 구동 방식을 분석하여, 메모리 조작 프로그램의 메모리 검색과 데이터 조작을 방지하는 방안인 EBC 시스템을 제안하였다. 기존의 방식이 데이터를 하나의 메모리에 저장하여 최종값을 갱신하는 형태로 사용했다면, EBC는 게임 내 데이터의 변동 이력을 일련의 이벤트로 관리하는 이벤트 소싱 패턴을 적용하여 메모리 조작 프로그램이 메모리값의 증감, 감소, 변동 없음의 조건으로 메모리 주소를 소거하는 방식의 검색 방법을 사용할 수 없도록 하였다. 이벤트 소싱 패턴을 적용하더라도 조작하려는 데이터의 값이 메모리에 하나만 존재할 수 있기 때문에 직접적인 값 검색이 어렵도록 모든 이벤트의 값은 XOR 암복호화를 사용하였다. 메모리 조작 프로그램 통해 게임에서 사용하는 모든 메모리를 조작하는 방법을 사용할 수 있기 때문에 블록체인을 활용하여 이벤트를 해시값을 가진 EB로 만들고 연결하였다. 연결된 EB를 통해 데이터 조작이 발생할 경우 해시값을 통해 데이터 조작 여부를 탐지할 수 있도록 하였다. 데이터의 조작을 탐지할 경우 게임 서버에 저장된 EB를 전송하여 악의적인 유저의 판단 및 제제의 근거로 사용될 수 있도록 하였다.

EBC를 사용할 경우의 성능을 평가하기 위해 암복호화를 적용하지 않은 경우, 데이터를 하나의 메모리에 저장하여 XOR로 암호화하는 유료 에셋을 사용한 경우와 비교하여 읽기, 쓰기에 대해 수행 시간을 판단하였다. 그 결과 암복호화를 적용하지 않은 경우에 비해 쓰기는 43배의 지연이 있었고, 유료 에셋보다는 5%의 성능향상이 있었다. 읽기의 경우 21배의 지연이 있었고, 유료 에셋보다는 50%의 성능 향상이 있었다. 게임에 영향을 주지 않는 데이터의 경우 유료 에셋이나 EBC를 사용하지 않는 것이 좋고, 중요한 데이터라면 EBC를 사용하는 것이 유료 에셋을 사용하는 것보다 성능 향상에 도움이 되는 것을 확인할 수 있었다. EBC의 경우 유료 에셋을 사용하는 것보다 EB가 사용하는 메모리 사용량과 개수에 비례하여 메모리의 사용량이 증가하는 단점이 있지만, 모바일 기기의 메모리 하드웨어 사양이 발전함에 따라 비례하여 상쇄된다.

References

[1] EunJin Kim, “Mobile game security trends,” Journal of The Korea Institute of information Security & Cryptology, 27(4), pp. 43-50, Aug. 2017

[2] Eun-Sang Cha and Youngsik Kim, "Performance evaluation of iocp game server and game variable obfuscation program," Journal of Korea Game Society, 19(6), pp. 71-81 Dec. 2019

[3] Appsealing, "The Complete Guide to Mobile App Security" https://www.appsealing.com/whitepapers/the-complete-guide-to-mobile-app-security/, Mar. 02. 2022

[4] Sun Myung Kang, "Security check items for safe mobile game service," Journal of The Korea Institute of information Security & Cryptology, 26(3), pp. 22-29, Jun. 2016

[5] Su In Kang and Huy Kang Kim, "How to improve online game security using intel sgx," Journal of The Korea Institute of information Security &

Cryptology, 27(4), pp. 22-26, Aug. 2017
[6] Hojun Kim and Hyoung-Kee Choi, "Cheats and its countermeasures on android mobile games," Proceedings of Symposium of the Korean Institute of communications and Information Sciences, pp. 606-607, Jun. 2017
[7] Chang Seon Lee and Jinho Yoo, "A study on improved detection signature system in hacking response of one-line games," The Jounal of Society for e-Business Studies 21(1), pp. 105-118, Feb. 2016
[8] Geon Il Heo, Cheong Il Heo, and Huy Kang Kim, "A study on mobile game security threats by analyzing malicious behavior of auto program of clash of clans," Journal of the Korea Institute of Information Security & Cryptology 25(6), pp. 1361-1376, Dec. 2015
[9] Unity, "2021 Gaming Report" https://create.unity.com/2021-game-report, Mar. 02. 2022
[10] Unity Asset Store, "Anti-Cheat Toolkit" https://assetstore.unity.com/packages/tools/utilities/anti-cheat-toolkit-2021-202695#description, Mar. 02. 2022
[11] Unity Asset Store, "Anti-Cheat Pro" https://assetstore.unity.com/packages/tools/utilities/anti-cheat-pro-117539#description, Mar. 02. 2022
[12] Sangkon Han, Jung-in Choi and Gyun Woo, "Case studies and trends in data reproduction and distributed processing using event sourcing and cqrs pattern," Journal of Korean Institute of Information Scientists and Engineers, 47(12), pp. 1101-1110, Dec. 2020
[13] Cheol Ho Jeon, Bo Min Seo, Geun Hyeok Lee, Hyeon Sig Jeon and Hyun Ju Park, "A study on the relationship between micro service architecture, event sourcing, and cqrs," Proceedings of the Korean Information Science Society Conference, pp. 230-231, Jun. 2019
[14] MinYoung Park, "Polyglot structure design to provide users with the efficiency of analysis within iot systems with eventsourcing and cqrs," Master's thesis, Hanbat National University, Aug. 2021
[15] Wookyeong Kim, "A study on how to transform monolith to microservices," Master's thesis, Graduate School of Information Sciences Soongsil University, Jun. 2020
[16] Sungbum Lee, Boohyung Lee, Sein Myung and Jong-Hyouk Lee, "Security analysis of blockchain systems : case study of cryptocurrencies," Journal of the Korea Institute of Information Security & Cryptology, 28(1), pp. 5-14, Feb. 2018
[17] Eun-Gyeom Jang, "Digital content certification and management technology based on blockchain technology," Journal of the Korea Society of Computer and Information, 26(11), pp. 121-128, Nov. 2021
[18] Hong-Seok Park and Tae-Gyu Kim, "A study on the improvement of mobile game payment using blockchain," Journal of the Korea Entertainment Industry Association, 14(3), pp. 163-171, Apr. 2020

〈저 자 소 개〉

박 지 훈 (Jihun Park) 정회원
2013년 6월~현재: 컴투스 모바일 게임 클라이언트 프로그래머
2014년 2월: 한국 산업 기술 대학교 게임공학과 학사
2020년 8월~현재: 세종 사이버대학교 정보보호학과 석사과정
〈관심분야〉 정보보호, 모바일 게임 보안

박 영 호 (Young-Ho Park) 종신회원
1990년 2월: 고려대학교 수학과 이학사
1993년 2월: 고려대학교 수학과 이학석사
1997년 2월: 고려대학교 수학과 이학박사
2002년 2월~현재: 세종사이버대학교 정보보호학과 교수
〈관심분야〉 공개키 암호, 양자내성암호, 암호 프로토콜, 암호안전성평가, 차세대인증