게임 무결성 제공

IT융합공학부 권혁동

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

CryptoCraft LAB

1. 메모리 조작

- 싱글 게임의 경우 모든 연산이 로컬에서 이루어짐
  - 단, 유저가 데이터를 조작하여 게임을 진행해도 피해가 없음

- 온라인 게임의 경우 서버 연산, 클라이언트 연산 두 종류로 나뉘어짐
  - 서버 연산: 연산을 서버에서 진행하여 서버 부담이 크지만 신뢰도 상승
  - 클라이언트 연산: **서버 부담을 줄이지만 조작의 위험 존재**

- 모바일 게임은 서버 부담을 줄이기 위해 **결과 값만 전송**하는 일이 잦음

1. 메모리 조작

- 메모리 조작을 통해 수정된 값을 입력
  - 전투 등의 **결과만을 서버**로 전송
  - 중간 값에 대한 로그가 남아있다면 검출 가능하나 없다면 불가능

- 재화 구매로 정상적인 플레이를 하는 유저들의 불만 초래

1. 메모리 조작

- 97년도에 출시된 에디터가 2018년에 출시된 게임에 적용된 모습
- 보안 요소를 고려하지 않은 경우 서비스 제공에 문제 발생

1. 메모리 조작

1. 공격자가 인게임에 표시된 공격력, 대미지 등을 확인
2. 에디터에 확인한 값을 입력
3. 다시 게임 플레이를 진행하고 입력한 값을 검사
4. 에디터가 값을 찾아내면 해당 값이 저장된 주소 값을 반환
5. 반환 된 주소 값에 원하는 값을 입력하여 메모리를 조작
6. 조작한 값이 사용되어 다음 플레이 시에 영향을 줌

2. 보안 기법

- 변수 쪼개기
  - 변수를 저장할 때 하나의 변수에 저장하지 않고 이를 나누어서 저장
  - 원래대로 복구하기 위해서는 함수를 통해 변수를 합쳐서 복구 가능
  - 32비트의 변수 -> 8비트 4개의 변수

- 예시
  - 저장할 값: 1300
  - 32비트 변수에 저장: 00000000000000000000010100010100 (1300)
  - 8비트 변수 4개에 저장: 00000000(0) / 00000000(0) / 00000101(5) / 00010100(20)
  - 32비트 변수에 저장 시 1300으로 검색하면 검출이 가능하나 쪼개서 저장하면 불가능

2. 보안 기법

- 마스킹
  - 변수를 하나의 변수에 저장하되 무작위로 생성되는 마스크를 씌워서 저장
  - 원래대로 복구하기 위해서는 함수를 통해 마스크를 벗겨서 복구 가능

- 예시
  - 저장할 값: 2500
  - 마스크: 01010101010101010101010101010101
  - 32비트 변수에 저장: 00000000000000000000100111000100 (2500)
  - 마스킹을 거친 값: 01010101010101010101110010010001 (1431657617)
  - 마스킹을 거치게 되면 값을 알아볼 수 없으므로 원본 값 추적이 어려움

2. 보안 기법

- Arm Trustzone
  - 트러스트 존을 사용하여 연산
  - 공격자가 위변조된 값을 사용할 경우 트러스트 존에서 감지 가능

- 서버 연산
  - 모든 연산을 서버 측에 위임
  - 가장 신뢰할 수 있는 방법이나 서버의 부담이 가중

3. 결론

- 게임 보안은 공격자가 유리하기 때문에 핵 문제에서 벗어나기 어려움
- 다만, 에디터 같은 로컬 변수 수정은 온라인 게임에서 불가능 해야함
  - 최근 발생한 사고는 보안에 대한 의식이 매우 결여된 것에서 발생한 사고
- 보안 방어 측이 항상 불리하긴 하지만 대비책은 준비해야 한다