

모바일 게임 보안

<https://youtu.be/nlkBismP3yk>

모바일 게임 해킹 종류

- 1. 메모리 해킹
 - 앱에 관련된 데이터가 메모리상에 로드된 이후에 게임에 영향을 미치는 데이터를 인위적으로 조작하는 행위
- 2. 코드사이드 해킹
 - 직접 코드 단계까지 가서 코드를 살피며 해킹하는 방법.
- 3. 네트워크 해킹
 - 서버와 교환되는 패킷을 빼내어 다른 정보로 서버에 전송하는 행위

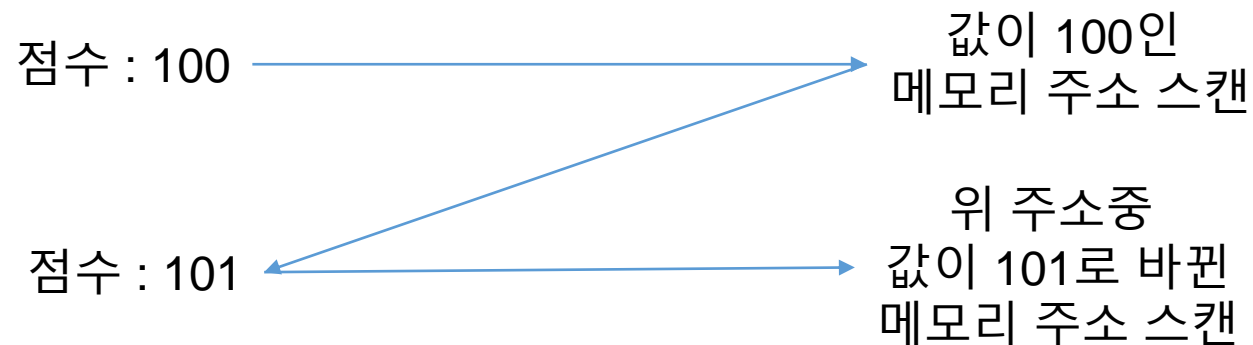
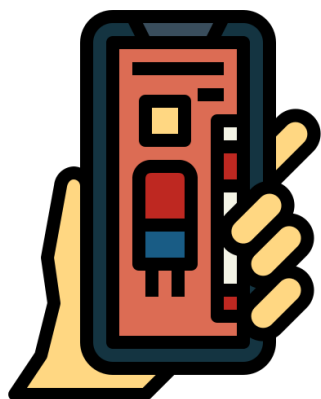
1. 메모리 해킹

- 현재 실행되고 있는 게임의 메모리를 조작하는 행위
- 메모리 치팅 툴을 이용하여 해킹을 시도함.
 - 치팅 툴은 구글 검색을 통해 쉽게 알아볼 수 있음.
 - 가상 머신의 사용도 증가하고 있음.



1. 메모리 해킹

메모리 해킹 방법



점수가 저장되는 주소를 찾을때까지 반복



1.메모리 해킹

- 1. 수치 변경 코드(암호화)

- 수치를 변경하여 저장하거나 암호화하여 저장하는 방법

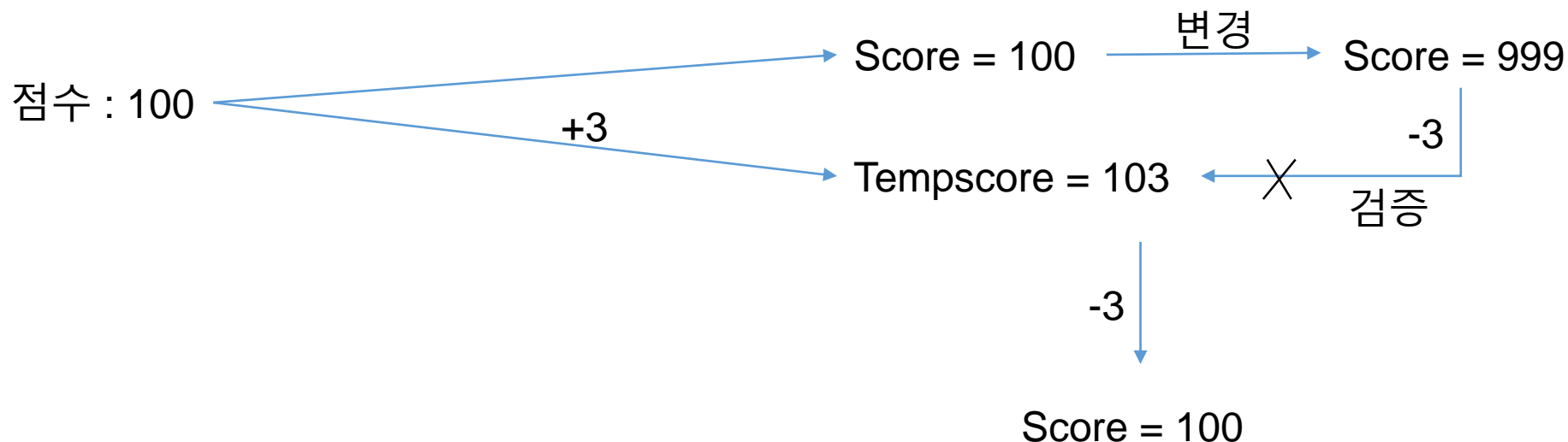
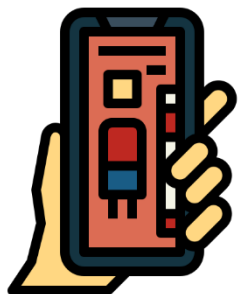
점수 : 100 $\xrightarrow{+1}$ 점수 = 101

- 스캐닝을 해도 실제값과 다르게 저장되어 찾을 수 없음.
- 적은 비용으로 큰 효과를 볼 수 있는 방법
- 이외의 비트 단위로 변경하거나 다른 문자 형태로 관리하는 방법도 있음.
- 하지만 코드를 확인할 수 있는 해커에게는 통하지 않음.

1. 메모리 해킹

• 2. 데이터 검증

- 클라이언트 자체에서 데이터를 검증하거나 서버에서 검증하는 방법
 - Ex) score 변수에 점수를 저장한다고 가정하면, temp score 변수를 하나 더 만들어 저장함.
 - Temp score 변수에는 수치 변경과 비슷하게 저장하여 score와는 다르게 저장
 - Update 할때마다 두 변수의 값을 비교하여 검증.



1.메모리 해킹

- 3. 해킹 툴 프로세스 탐지

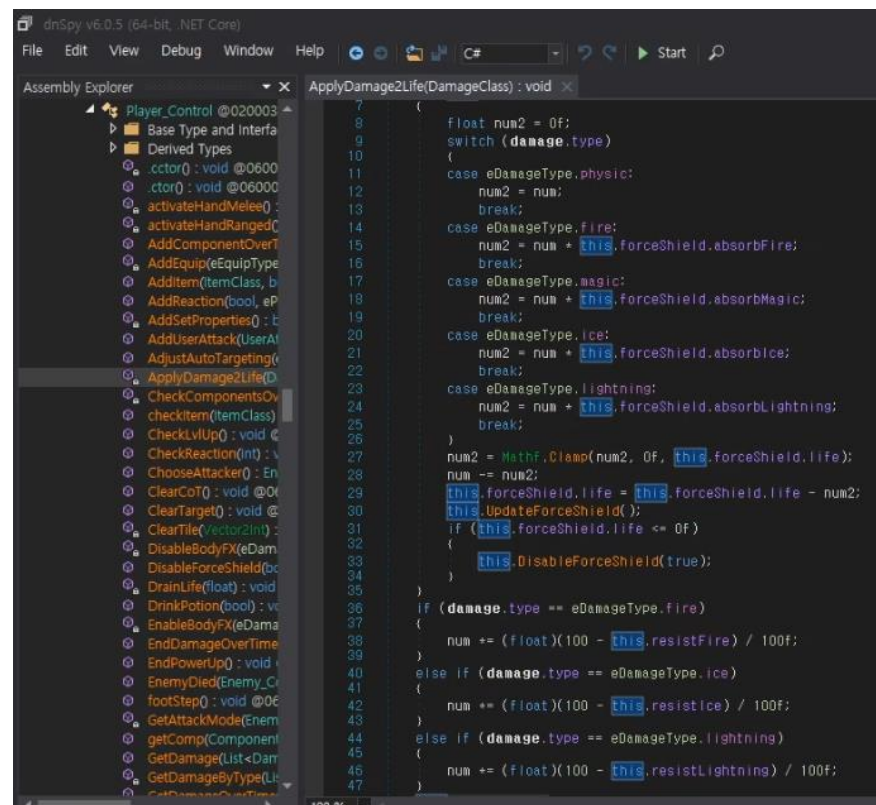
- 게임 실행 시 해킹 툴이 실행되고 있으면 게임을 종료해버리는 것.
- 게임 실행 시 확인하거나, 실행 후 중간중간 확인 하는 경우가 있다.
- 전자의 경우 중간에 해킹툴을 실행하면 발견할 수 없음.
- 후자의 경우 중간에 한번 씩 확인해야하는 로직이 필요하여, 게임 이용에 불편함이 있거나, 해킹 툴이 아님에도 게임이 종료되는 경우가 발생할 수 있음.

2. 코드사이드 해킹

- 유니티 기반의 앱은 mono, il2 두가지 방식으로 컴파일이 가능.
- Mono 방식의 컴파일은 빠른 빌드속도와 여러 플랫폼 구동이 가능한 장점이 있으나 보안에 매우 취약한 단점이 있음.
- Mono기반의 게임은 코드 난독화 및 보호솔루션이 없어 오픈된 소스코드 처럼 de파일을 쉽게 디컴파일이 가능함.

2. 코드사이드 해킹

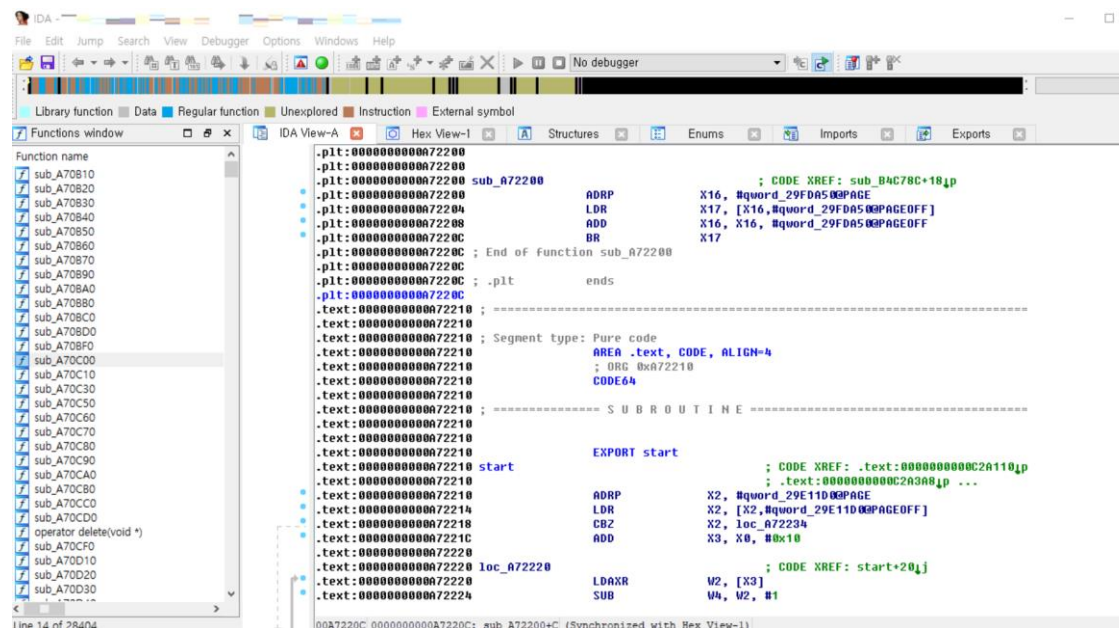
- Dnspy 프로그램을 활용하여 코드 확인 및 수정이 가능.
- Mono의 이러한 보안상 문제로 il2cpp 컴파일로 바뀌는 추세.



```
dnSpy v6.0.5 (64-bit, .NET Core)
File Edit View Debug Window Help
Assembly Explorer
Player_Control @020003
  Base Type and Interface
  Derived Types
    ctor0 : void @0600
    ctor0 : void @0600
    activateHandMelee0 :
    AddComponentOverTime :
    AddEquip(eEquipType) :
    AddItem(itemClass, b... :
    AddReaction(bool, eP... :
    AddSetProperties() : b... :
    AddUserAttack(UserAt... :
    AdjustAutoTargeting() :
    ApplyDamage2Life(Da... :
    CheckComponentsOverTime :
    CheckItem(itemClass) :
    CheckLvUp() : void :
    CheckReaction(int) :
    ChooseAttacker() : En... :
    ClearCoT() : void @0... :
    ClearTarget() : void @... :
    ClearTile(Vector2Int) :
    DisableBodyFX(eDam... :
    DisableForceShield(b... :
    DrainLife(float) : void :
    DrinkPotion(bool) : v... :
    EnableBodyFX(eDama... :
    EndDamageOverTime :
    EndPowerUp() : void :
    EnemyDied(EnemyCl... :
    footStep() : void @0... :
    GetAttackMode(Enem... :
    GetComp(Component... :
    GetDamage(List<Dam... :
    GetDamageByType(Li... :
    GetDamageOverTime :
    ApplyDamage2Life(DamageClass) : void
      7
      8 float num2 = 0f;
      9 switch (damage.type)
      10 {
      11 case eDamageType.physics:
      12   num2 = num;
      13   break;
      14 case eDamageType.fire:
      15   num2 = num + this.forceShield.absorbFire;
      16   break;
      17 case eDamageType.magic:
      18   num2 = num + this.forceShield.absorbMagic;
      19   break;
      20 case eDamageType.ice:
      21   num2 = num + this.forceShield.absorbIce;
      22   break;
      23 case eDamageType.lightning:
      24   num2 = num + this.forceShield.absorbLightning;
      25   break;
      26 }
      27 num2 = Mathf.Clamp(num2, 0f, this.forceShield.life);
      28 num -= num2;
      29 this.forceShield.life = this.forceShield.life - num2;
      30 this.UpdateForceShield();
      31 if (this.forceShield.life <= 0f)
      32 {
      33   this.DisableForceShield(true);
      34 }
      35 }
      36 if (damage.type == eDamageType.fire)
      37 {
      38   num += (float)(100 - this.resistFire) / 100f;
      39 }
      40 else if (damage.type == eDamageType.ice)
      41 {
      42   num += (float)(100 - this.resistIce) / 100f;
      43 }
      44 else if (damage.type == eDamageType.lightning)
      45 {
      46   num += (float)(100 - this.resistLightning) / 100f;
      47 }
```

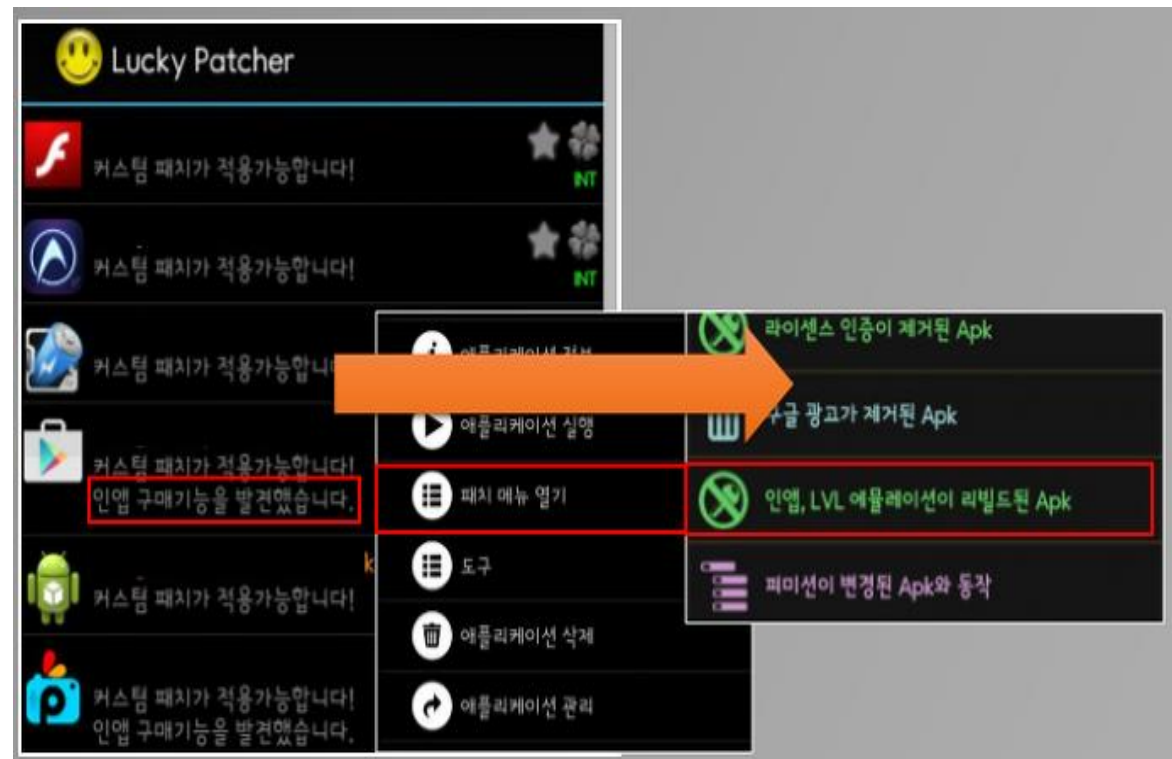
2. 코드사이드 해킹

- 유니티는 c#코드를 사용하는데 il2cpp로 빌드하게 되면 c++파일들로 바뀐다.
- 하지만 리버싱 용도로 쓰이는 IDA같은 프로그램을 활용하여 해킹
- IDA는 코드를 어셈블리어로 바꾸어주는 프로그램
 - DE파일 -> 어셈블리어 변경 후 코드 분석



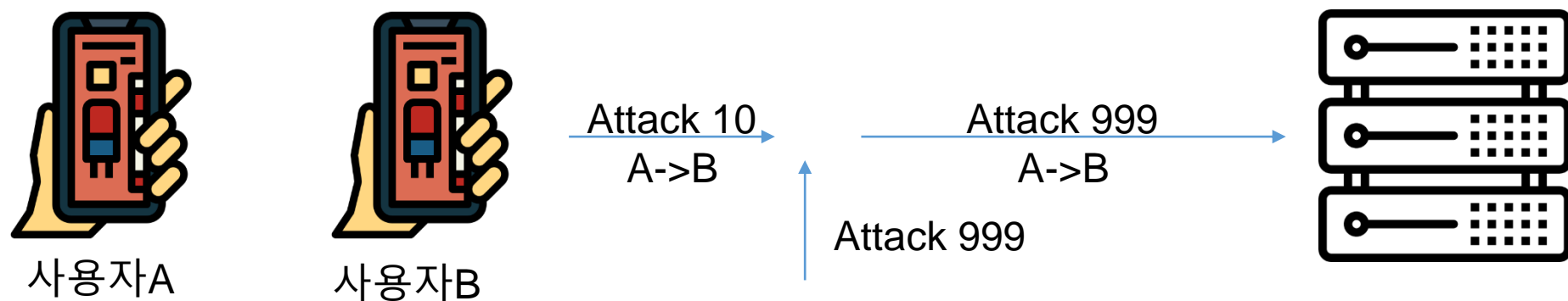
2. 코드사이드 해킹

- 광고 제거
- 인앱결제 우회
- 해킹 툴 감지 제거



3. 네트워크 해킹

- 게임중 클라이언트와 서버간의 패킷을 조작하는 해킹



- 굉장히 복잡하고 손이 많이가고 서버쪽의 보안이 클라이언트 보안보다 강력하기 때문에 어렵다.

모바일 게임 해킹의 문제점

- 해킹 난이도가 상대적으로 매우 쉬워 일반 유저도 해킹을 할 수 있음.
- 해킹 툴이 많이 개발되어 배포되는데 해킹 툴로 인한 일반 유저들 또한 개인정보가 해킹될 수 있음.
- 금전적이 문제.

Q & A

