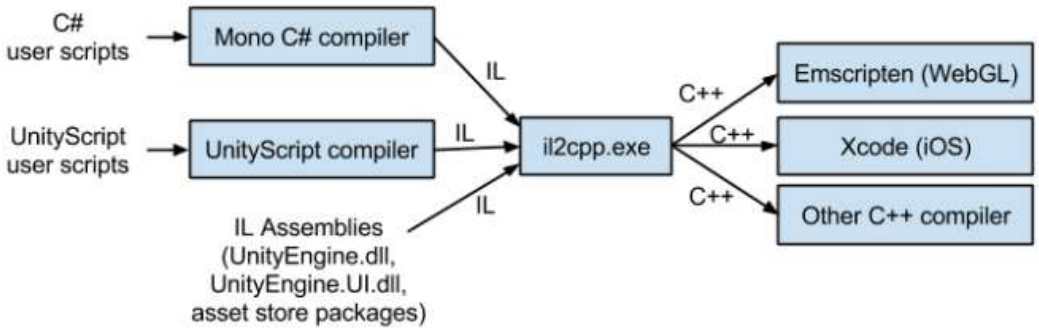

기존에 유니티에서 사용하던 모노방식은 스크립트를 IL로 만든다음 배포를 해서 디컴파일을 하면 유저가 만든 스크립트, 즉 게임의 주요한 부분들을 연산하는 코드들이 전부 노출이 된다.

이러한 부분에 대해 대응하기 위해 유니티에서 새로 만든 컴파일 방식이 il2cpp이다.

위의 그림과 같이 il2cpp.exe가 기존의 방식으로 생성된 IL파일을 cpp파일로 바꾸고, 컴파일해서 배포를 하기 때문에 모노방식처럼 유저가 배포된 파일을 가지고 바로 디컴파일 하는 것은 불가능하다.

하지만 il2cpp방식에서도 배포파일에 global-metadata.dat라는 파일이 포함되는데 이 파일과 어셈블리어 레벨로 배포된 libil2cpp.so 파일을 이용해서 심볼을 복구하는 il2cppdumper라는 툴이 있다.

이 툴의 결과로 모노에서 디컴파일을 통해 소스를 복구하는 것처럼 함수의 기능과 구현까지 모두 볼 수는 없지만 메소드의 이름과 offset을 확인할 수 있다.

물론 게임사들도 이런 툴이 존재한다는 사실을 인지하고 있다. 그래서 il2cppdumper의 입력으로 필요한 global-metadata.dat와 libil2cpp.so파일을 패킹/암호화 하여 배포하고, 언패킹/복호화하는 코드를 추가해서 배포하는 경우가 있는데, 동적으로 메모리에 해당 데이터가 로드되는 시점에 덤프를 하는 것으로 우회할 수 있다. 하지만 코드를 가상화하는 경우에는 해당 함수에서 사용하는 지역변수에 대한 심볼은 복구가 불가능하다.

즉, 유니티의 구조적인 문제 때문에 유니티로 만든 게임은 앞서 말한 여러 정보들을 공격자에게 노출 시킬 수밖에 없다.

드래곤주먹 팀에서는 현재 배포파일을 통해 파악 가능한 메소드의 이름과 offset을 통해 위협이 될 만한 공격벡터를 찾는 과정을 자동화할 수 있다고 보고 있고 이 자동 진단 툴을 개발하는 것이 단기적인 목표이다.

벡터를 찾는 과정은 정적분석과 동적분석으로 나눈다.

정적분석의 경우 게임소스에서 주로 사용하는 문자열 딕셔너리를 만들어서 그 딕셔너리와 매칭되는 메소스들을 파악하여 사용자에게 보여준다.

동적분석의 경우 현재 최우선 목표인 apk파일을 분석하려면 에뮬레이터가 필요한데, 이걸 툴에 포함을 할지, 다른 에뮬레이터를 실행하고 attach하는 방식으로 할지는 아직 정하지 않았다. 프로세스에 툴이 attach하고 나서는 앞서 정적인 부분에서 매칭되었던 메소스들과, attach된 프로세스가 동작하는 중에 생성된 인스턴스를 전부 파악하고, 어떤 prefab(게임에서 생성될 오브젝트들의 class)을 통해 인스턴스가 생성되었는지를 파악한다. 이를 통해 툴의 사용자가 게임을 직접 플레이하면서 마주쳤던 여러 오브젝트들이 어떤 prefab을 통해 생성되었는지를 알 수 있고, 오브젝트가 사용하는 함수에 대해서도 전부 파악할 수 있다.

또한 유니티의 경우 어떤 오브젝트에 다른 오브젝트가 종속되어 동작하는 구조를 많이 사용하고, 게임이 시작할 때부터 끝까지 존재하는 메인 오브젝트를 따로 두는 경우가 많아서 메인 오브젝트의 메소드를 찾고, 새로 생성하는 오브젝트를 메인 오브젝트에 종속하는 함수와 prefab을 인스턴스화 하는 함수에 후킹을 걸어서 공격자가 접근 가능한 게임의 주요한 변수들을 파악하는 방식도 연구 중에 있다.