논문 2019-1-2 http://dx.doi.org/10.29056/jsav.2019.06.02

안드로이드 환경에서 크로스 플랫폼 개발 프레임워크에 따른 모바일 앱 분포

김규식*, 전소연**, 조성제*†

Distribution of Mobile Apps Considering Cross-Platform Development Frameworks in Android Environment

Gyoosik Kim*, Soyeon Jeon**, Seong-je Cho*†

요 약

모바일 앱 개발자는 크로스 플랫폼 개발 프레임워크를 사용하여 서로 다른 모바일 플랫폼들에 구동되는 앱들을 하나의 단계로 구현할 수 있다. 공격자들 또한 크로스 플랫폼 개발 프레임워크를 사용하여 한번 작성된 악성 코드를 여러 모바일 플랫폼들 상에 바로 수행할 수 있다. 본 논문에서는 AndroZoo 사이트로부터 수집한 안드로이드 앱들을 대상으로 크로스 플랫폼 개발 프레임워크들로 작성된 정상 앱들과 악성 앱들의 비율을 연도별로 분석한다. 분석 결과, 크로스 플랫폼 개발 프레임워크들로 작성된 정상 앱들의 비율이 지속적으로 증가하여, 2018년도에는 전체 정상 앱들에서 45%를 차지한다. 크로스 플랫폼 개발 프레임워크로 작성된 악성 앱들의 비율은 2015년에는 전체 악성 앱들에서 25%를 차지하였으나 이후 그 비율이 감소하고 있다. 이러한 연구는 크로스 플랫폼 앱 개발 시에 직면할 수 있는 여러 선택 문제들을 해결하는데 기여할 수 있다.

Abstract

Using cross-platform development frameworks, mobile app developers can easily implement mobile apps for multiple platforms in one step. The frameworks also provides adversaries with the ability to write malicious code once, and then run it anywhere for other platforms. In this paper, we analyze the ratio of benign and malicious apps written by cross-platform development frameworks for Android apps collected from AndroZoo's site. The analysis results show that the percentage of benign apps written in the frameworks continues to increase, accounting for 45% of all benign apps in 2018. The percentage of malicious apps written in the frameworks accounted for 25% of all malicious apps in 2015, but that percentage has declined since then. This study provides useful information to make a suitable choice when app developers face several challenges in cross platform app development.

한글키워드 : 크로스 플랫폼 개발 프레임워크, 네이티브 앱, 하이브리드 앱, 자립형 앱, 악성 앱

keywords : cross-platform development framework, native app, hybrid app, self-contained app, malicious app

* 단국대학교 컴퓨터학과
** 단국대학교 소프트웨어학과
† 교신저자: 조성제(email: sjcho@dankook.ac.kr)
접수일자: 2019.06.02. 심사완료: 2019.06.17.
게재확정: 2019.06.20.

1. 서 론

스마트폰 플랫폼은 안드로이드, iOS, 윈도우즈

폰, BlackBerry OS와 같이 다양하다. 각 스마트폰 플랫폼은 개발 환경과 프로그래밍 언어가 달라 특정 플랫폼에 종속되어 개발된 소프트웨어(앱)는 다른 플랫폼에 그대로 사용될 수 없다. 따라서 특정 플랫폼을 위한 네이티브 앱(native app)을 서로 다른 플랫폼마다 각각 개발하는 것은 개발 비용을 크게 증가시킨다. 이러한 문제를 해결하기 위해서 최근 크로스-플랫폼 모바일 앱 개발 프레임워크(cross-platform mobile app development framework)의 사용이 증가하고 있다[1-4]. 크로스-플랫폼 모바일 앱 개발 프레임워크를 줄여서, 크로스 플랫폼 개발 프레임워크(cross-platform development framework)로 칭한다.

대표적인 크로스-플랫폼 개발 프레임워크로는 유니티(Unity), 자마린(Xamarin), Appcelerator Titanium, PhoneGap, Sencha, Cocos2d 등이 존재한다. 이러한 프레임워크들은 모바일 앱의 개발 생명주기 동안 효율성을 증대시켜 준다. 이러한 이유 때문에, 특정 플랫폼에 독립적인 크로스 플랫폼 개발 프레임워크가 주목받고 있다. Sencha나 PhoneGap 등의 크로스 플랫폼 개발 프레임워크들은 웹 기반의 Javascript, HTML5을 사용한다. 유니티는 기존 Windows나 Mac OS X와 같은 PC 기반 플랫폼에서 게임 개발에 사용되는 엔진으로, 2010년 Unity 3이 발표되면서 다양한 스마트폰 플랫폼용 모바일 앱 개발에 사용되고 있다. C# 프로그래머들은 유니티를 이용하여 새로운 프로그래밍 언어를 익힐 필요 없이 모바일 앱을 즉시 개발할 수 있으며, 작성한 C# 코드를 다양한 플랫폼 간에 재사용이 가능하다. 이들 크로스-플랫폼 개발 프레임워크로 작성된 앱들을 크로스 플랫폼 앱이라고 칭한다.

한편, 멀웨어 작성자(malware writer)들도 멀웨어 제작 시간과 비용을 최소화하면서 최대의 공격 효과를 내려고 이러한 크로스-플랫폼 개발 프레임워크를 사용한다[5-7]. 공격자들은 C#을 사용하여 악성 앱들을 더 빨리 수정하여 더 많은 모바일 플랫폼들에 전파할 수 있기 때문이다. Veil- Framework는 백신을 우회하는 실행파일을 생성하는 공격 도구들의 집합으로, 페이로드가 C#으로 개발되기도 했다.

실제 인터넷을 검색하여 보면 C# 기반의 악성코드를 제작하는 정보가 많다. C#으로 키로거(keylogger)를 작성하는 방법, C#을 사용한 Crypto-Ransomware 작성, C#으로 콘솔 기반의 트로이목마를 작성하는 튜토리얼, C#기반 Nemesis.Worm 등을 접할 수 있다. 실제 Mylonas 등은 실험을 통해 학생들이 C#을 이용하여 Windows Mobile에서는 이틀 만에 악성 앱을 제작하고, Windows Phone에서는 하루 만에 악성 앱을 제작할 수 있음을 보였다[8][9].

Lee 등의 연구[5]에 의하면, 유니티와 PhoneGap으로 작성된 안드로이드 앱들이 증가하고 있다. 그리고 잠재적으로 원하지 않는 앱(potentially unwanted app, PUA)들의 샘플이 유니티와 Cocos2d로 작성된 앱들 중에서 다수 발견되었으며, 멀웨어 샘플이 PhoneGap으로 작성된 앱들에서 많이 발견되었다. 결과적으로, 보안 연구자들이 크로스 플랫폼 개발 프레임워크로 제작된 모바일 멀웨어를 분석하고 탐지해야 하는 큰 도전과제에 직면하고 있다고 서술하였다.

이처럼, 크로스 플랫폼 개발 프레임워크로 작성되는 모바일 앱들은 많은데, 이들 프레임워크로 제작된 악성 앱이나 불법 앱에 대한 관심이나 분석 연구는 많지 않다. 이에 본 논문에서는, AndroZoo 사이트에서 수집한 앱들을 대상으로 크로스 플랫폼 개발 프레임워크로 개발된 앱들에 대한 년도 별 추이를 분석한다. 즉, 크로스 플랫폼 개발 프레임워크로 작성된 정상 앱들과 악성 앱들의 대한 연도별 비율을 파악하는 연구를 수행한다. 이러한 연구 결과는, 정상 앱들의 개발

도구 및 현황을 파악하는데 도움을 줄 수 있으며, 악성코드의 효율적 분석에 기여할 수 있다. 또한, 모바일 앱 개발자들이 크로스 플랫폼 앱 개발 시에 어떤 프레임워크를 채택해야 할지 고민할 수 있는데, 관련 고민 해결에 도움이 될 수 있는 정보를 제공하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 설명하고, 3장에서는 크로스 플랫폼 개발 프레임워크로 작성된 앱 분류 기법에 대해 설명한다. 4장에서는 크로스 플랫폼 개발 프레임워크들의 종류 및 각 프레임워크를 식별하는 특징정보에 대해 기술한다. 5장에서는 크로스 플랫폼 개발 프레임워크로 작성된 앱 분류 결과를 제시한다. 6장에서는 연구에 대한 결론을 도출하고, 향후연구 방향에 대해 서술한다.

2. 관련연구

기존 연구에서는 모바일 앱 개발 방법 및 앱 실행 방법에 따라 여러 분류 기준을 제시했다[1-4]. 그 중 [1] 연구의 경우 (1)네이티브 앱(native apps), (2)모바일 웹 앱(mobile web apps), (3)하이브리드 앱(hybrid approaches of Web and native components, 이하 하이브리드 앱), (4)자립형 프레임워크 앱(self-contained framework apps run on self-contained runtime environment)의 4가지 타입이 존재한다. 개발 방법 및 애플리케이션 실행 방법에 따른 분류는 다음과 같다.

2.1 네이티브 앱(Native Apps)

네이티브 앱은 시스템 콜 호출이 가능하기에 기기에 저장된 주소록, 파일 등의 고유 정보를 사용할 수 있으며 카메라 등의 하드웨어 또한 제어 가능하며, 웹 앱에 비하여 높은 사양의 그래픽과 성능을 자랑하면서도 구동 속도가 빠르며, 인터넷이 연결되어 있지 않아도 사용이 가능하다. 일반적으로 안드로이드 애플리케이션은 Android SDK (Software Development Kit)가 포함된 안드로이드 스튜디오를 이용해 개발하는 것이 일반적이다[10]. 안드로이드 SDK에는 포괄적인 개발 도구들이 포함되어 있다. 예를 들면 디버거, 라이브러리, QEMU 기반 에뮬레이터 및 각종 샘플코드 등이 존재한다. 또한 모바일 기기의 API 버전에 맞는 Android.jar (2019년 6월 기준 안드로이드 9.0 파이 API 레벨 28)를 제공하여 Windows, Linux, Mac OS 등 다양한 개발 환경에서 의존성을 해결하고 앱을 컴파일할 수 있도록 한다. 그 외에 성능의 최적화 및 고성능의 그래픽 작업과 널리 사용되는 C/C++ 라이브러리 사용을 위해 크로스 컴파일이 지원되는 Android NDK(Native Development Kit)를 제공하여 C++ 코드가 포함된 안드로이드 애플리케이션을 개발할 수 있도록 지원한다[11].

2.2 모바일 웹 앱(Mobile Web Apps)

모바일 웹 앱은 HTML, CSS, JavsScript로 구현되며, 런타임 환경으로 브라우저를 사용하고 모바일 플랫폼의 브라우저 지원을 활용한다[1]. 표준 기술로, 각 플랫폼의 모바일 브라우저를 사용하여 유사한 방식으로 웹 사이트들을 접근할 수 있다. 그러나 각 플랫폼 의존적인 실행코드를 포함하지 않기에, 각 플랫폼에 해당하는 애플리케이션 포멧(안드로이드: .apk, 애플: .ipa) 형태로 배포되지 않으며 한 모바일 웹 앱은 카메라, GPS 센서와 같은 디바이스, 특정 하드웨어 기능들을 사용할 수 없다. 대표적 예는 JQuery Mobile, Sencha Touch 등이 존재한다.

2.3 하이브리드 앱 (Hybrid apps)

웹과 네이티브 컴포넌트들로 구성된 하이브리드 앱(이하 하이브리드 앱)은 외부형태는 네이티브 앱이면서 실제 내부는 각 모바일 OS에 탑재된 웹킷(Webkit)을 활용하여 모바일 웹앱을 실행할 수 있다. 하이브리드 앱은 개발자가 플랫폼에 의존적인 기능(카메라 등의 장치 제어)만 네이티브 라이브러리로 제작하고 HTML과 JavaScript로 네이티브 기능을 이용할 수 있도록 했다. 애플리케이션의 네이티브 부분은 각 OS의 API를 사용하여 브라우저와 디바이스 API 간에 연결고리 역할을 하는 엔진을 통해 하이브리드 앱은 최신 디바이스가 제공해는 모든 기능을 활용할 수 있다[12]. 현재 많은 웹 기반 하이브리드 개발 프레임워크가 존재하며 Titanium Mobile과 PhoneGap, Cordova 등이 있으며 개발과 배포가 용이하다는 점 때문에 널리 사용되고 있다.

하이브리드 앱은 기존의 웹 기술을 사용하기에 크로스 사이트 스크립팅(XSS, Cross-Site Scripting)과 크로스 사이트 요청 위조(Cross-site request forgery)과 같은 웹 관련 취약점을 공유할 수 있다[13][14]. 이러한 웹 기반 취약점들이 하이브리드 앱에서 악용될 경우 자바스크립트(Java Script)를 통하여 네이티브 API와 하드웨어 및 기기정보에 접근할 수 있기에 민감정보를 탈취할 수 있으며, 따라서 더욱 위협적인 웹 공격으로 변모할 수 있다[13]. 특히 [13] 연구의 경우 Cordova 기반 앱을 리패키징하는 공격방법과 Cordova 앱을 대상으로 하는 XSS 공격방법에 대해 연구하고 발표했다.

2.4 자립형 프레임워크 앱(self-contained framework apps)

자립형 프레임워크 앱은 모바일 플랫폼에 존재하는 어떤 (웹) 환경도 재사용하지 않고, 자기 자신의 분리된 런타임 환경(their own, separate runtime environment)을 사용한다[1]. 대표적으로 모노(Mono) 프로젝트는 마이크로소프트에서 개발한 닷넷 프레임워크(.NET Framework)의 오픈소스 버전이다. 모노 프로젝트는 C#으로 작성하여 컴파일된 공통언어(CIL)이 모노 런타임(Mono Runtime) 환경에서 동작하게 하여 윈도우, macOS, 리눅스와 같은 환경에서 닷넷 프레임워크를 적용할 수 있다.

다양한 OS환경에서 동작하는 모노는 모바일 개발 플랫폼인 자마린(Xamarin)과 유니티(Unity)의 기반 프레임워크로 동작하여 모바일 환경인 iOS와 Android에서 동시에 개발 및 적용이 가능하다. 모노를 활용하는 자마린과 유니티의 특징은 각종 시스템 콜과 기반 OS(Android, iOS)의 API의 래퍼(Wrapper)를 구현하였다. 개발자는 이를 활용하여 SQLite DB, 파일 접근, 카메라 등 하드웨어 이벤트 또한 제어가 가능하며, 구동속도가 빠름과 동시에 다양한 플랫폼 환경에서도 적용 가능하다[15]. 대표적인 예로 자마린 사에서 개발한 Xamarin Evolve라는 안드로이드 앱과 iOS앱을 지원하는 앱의 경우, 약 15,000라인의 코드를 작성되었으며, iOS의 경우 이중 93%가 공유 코드였으며, 안드로이드의 경우 90%가 공유코드를 보이며 자마린의 C# 코드 재사용성의 장점을 보여주었다[16].

다른 프레임워크로는 Cocos2d-x가 존재한다. Cocos2d는 2D 게임 개발용 오픈 소스 소프트웨어 프레임워크이며, 게임과 모바일 앱, 반응형 전자책 등 GUI 기반 상호작용 소프트웨어의 개발에 사용할 수 있다[17]. Cocos2d-x는 C++ 언어로 제작하여 iOS에서 사용할 수 있는 IPA파일을 생성할 수 있고, 안드로이드에서 사용하는 APK 파일을 생성할 수 있으며, 네이티브 언어인 C++를 사용하기에 C# 래퍼(Wrapper)를 사용하는 유

니티보다 빠르다는 장점을 가지고 있다[18].

자립형 프레임워크 앱은 Native 앱과 달리 실행코드가 dll(dynamic link library)나 so(shared object)에 존재할 수 있기에 악성코드 제작자들에 의해 악용될 수 있다. 대표적인 예로는 “Trojan.Android.Banker”를 들 수 있다. 해당 악성 앱은 자마린으로 개발되었으며, 구글 플레이스토어에서 원본 앱과 비슷하게 만든 페이크 앱을 이용하여 사용자를 속인다. 또한 지속적인 원격 명령을 통해서 기기정보 탈취뿐만 아니라 문자기록, 주소록 등의 개인정보 및 카드정보까지 탈취한다[19]. 크로스 플랫폼 앱을 이용한 공격은 악성행위를 수행하는 실행코드가 dll에 존재하기에 악성코드 분석 대상인 dex를 기법을 통해서는 탐지할 수 없다.

3. 크로스플랫폼 앱 분류 기법

정상 개발자뿐만 아니라 공격자들도 멀웨어 제작시간을 최소화하면서 최대의 공격 효과를 내기 위해 크로스 플랫폼 환경에서 악성 앱을 작성한다[13][14][19]. 이러한 공격 방법에 대응하기 위해서는 앱이 작성된 플랫폼을 분류하고 그에 따른 분석 방법이 필요하다. 본 논문에서는 크로스 플랫폼을 이용한 저작권 침해 및 악성행위를 방지하기 위한 일환으로 앱이 제작된 플랫폼을 유추하는 연구를 수행한다.

구체적으로 네이티브 앱, 자립형 프레임워크 앱, 하이브리드 앱으로 분류한다. 기존 연구 [1]에서 분류한 모바일 웹 앱을 제외한 이유는 모바일 웹 앱은 안드로이드 플랫폼에서 앱 형태로 배포하지 않기에 직접 분석이 어렵기 때문이다. 자립형 프레임워크 앱과 하이브리드 앱의 경우, 세부적으로 어떠한 프레임워크를 사용하여 앱이 제작되었는가를 분석한다.

3.1 자립형 프레임워크 앱 분류 기법

자립형 프레임워크 앱은 대표적으로 모노 런타임을 사용하는 Xamarin과 Unity가 존재하며, Cocos2d-x는 Cocos2d라는 게임엔진을 사용한다. 앱을 동작하기 위해 안드로이드에서 모노 런타임과 Cocos2d를 구축하기 위해서는 자바 코드를 시작으로 하여 C/C++로 제작된 네이티브 함수를 JNI를 이용해 등록하고 호출해야 한다. 이들 엔진 및 머신에 사용하는 공유 라이브러리(so)는 개발사의 중요 정보로서 한번 빌드되고, 앱 개발 환경에서는 재빌드되지 않는다. 따라서 JNI에 매핑되는 함수의 심볼 정보는 Proguard, Dexguard, DashO와 같은 난독화 도구에 의해 변조되기 힘들다. 대표적으로 안드로이드에서 JNI 사용 예는 그림 1과 같다.

```
#include <stdio.h>
#include "First.h"
JNIEXPORT void JNICALL Java_First_foo
(JNIEnv *env, jobject obj)
{
    printf("NATIVE : foo()\n");
}
```

그림 1. C++에서 네이티브 함수의 선언
Fig. 1. declaration of a native function in C++

그림 2에서와 같이 Java에서 First 클래스의 foo라는 네이티브 함수를 등록하기 위해서는 System.loadLibrary라는 함수를 이용하여, [표 1-1]에서 보이는 바와 같이 JNIEXPORT속성과 Java_First_foo(First: 클래스 명, foo: 메소드 명) 심볼을 보유한 공유 라이브러리(so)를 로드해야 한다. 이렇기에 공유 라이브러리(so)의 심볼 정보가 변조되지 않는 한 자바코드에서 불려지는 네이티브 함수의 심볼 정보는 난독화 도구에 의해 변조되기 힘들다.

```
class First //--> 클래스 명
{
    static
    {
        System.loadLibrary("First");
    }
    native void foo(); // --> 메소드 명
    public void abc( )
    {
        First first = new First();
        first.foo();
    }
};
```

그림 2. Java에서 네이티브 함수의 등록 및 호출
Fig. 2. registering and invoking a native function in Java

표 1. dex 파일에서 메소드 접근 플래그
Table 1. access flags for a method in a dex file

Name	access flag (value)	메소드 접근제한자 및 설명
PUBLIC	0x1	visible everywhere
PRIVATE	0x2	only visible to defining class
PROTECTED	0x4	visible to package and subclasses
STATIC	0x8	does not take a this(instance) argument
FINAL	0x10	not overridable
NATIVE	0x100	implemented in native code

안드로이드의 실행파일인 dex에서는 네이티브 메소드(native method)라고 하는데 메소드 중에서도 네이티브 메소드를 구분짓기 위해서 메소드의 접근 플래그(access flag)를 활용한다. dex 파일 포맷에서 사용하는 주요 접근 플래그는 표 1과 같으며 이 중 Access flags가 native (ACC_NATIVE: 0x100)를 포함하는 메소드의 심볼 및 소속 클래스 정보(class descriptor)를 이용하여 자립형 프레임워크 앱을 분류한다.

3.2 하이브리드 앱 분류 기법

하이브리드 앱을 개발하게 해 주는 프레임워크는 네이티브 앱과 웹 앱의 장점을 함께 고려한 도구이기에, 하이브리드 앱은 네이티브 앱처럼 하드웨어의 기능들을 사용할 수 있으며 마켓에 등록이 가능하며 배포 후에도 웹만 연결되어 있다면 수정·보완이 용이하다. 이들 웹 기반 하이브리드 앱들은 기기에 API행태로 내포되어 있는 웹킷(WebKit)을 활용하는 것이 일반적이며 안드로이드의 경우 기기의 OS에 내포된 Webkit이 정의된 클래스를 상속받아 개발하도록 설계되어 있다. 그림 3은 대표적인 웹 기반 하이브리드 앱의 기반인 코르도바(Apache Cordova)의 주요 클래스인 Lorg/apache/cordova/engine/SystemWebView; 의 상속구조이다.

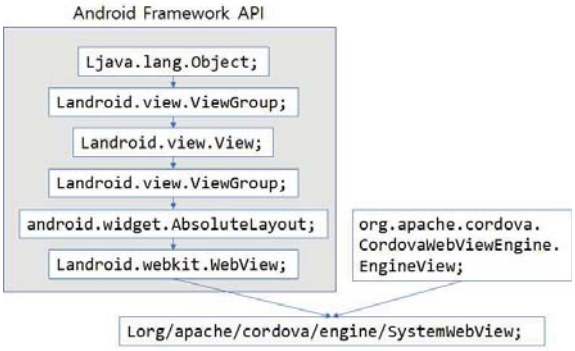

그림 3. 코르도바 SystemWebView의 상속구조
Fig. 3. inheritance structure of cordova SystemWebView

안드로이드 OS를 대상으로 한 하이브리드 앱들은 대부분 Landroid.webkit.WebView; 클래스와 Landroid.webkit.WebViewClient; 를 상속받아 웹킷의 주요 메소드들을 오버라이딩(Overriding)하여 재사용성을 증가시키고, 필요한 추가적인 메소드를 구현하여 사용한다. 따라서 WebView와 WebViewClient를 상속받는 클래스 명을 식

별하고 이를 기준으로 웹 기반 하이브리드 앱을 종류에 따라 식별할 수 있다.

하지만 상속받는 클래스는 난독화 도구에 의해 심볼이 변경(renaming)될 경우 분석이 부정확할 수 있다. 또한 상속받는 클래스의 심볼이 버전에 따라 다를 수 있다. 대표적인 예시로 Native Script가 존재한다. Native Script는 자체 렌더링 엔진을 위한 런타임을 보유하고 있으며, 버전에 따라 WebView를 상속받는 클래스 명이 상이하다. (예 :WebViewClient_vendor_1_1934982_n;, WebViewClient_vendor_1_1978533_n;) 이러한 이유로 인해 하이브리드 앱을 분류할 때도 자립형 프레임워크 앱을 분류할 때 사용한 정보인 네이티브 함수를 사용한다. 하이브리드 앱에서도 자체 렌더링 엔진을 사용하는 경우가 존재하며, 런타임을 구축하기 위해 네이티브 함수를 등록하고 실행한다. 따라서, 하이브리드 앱을 더욱 세부적으로 분류하기 위해 네이티브 함수의 심볼 정보를 활용한다.

4. 크로스플랫폼 프레임워크의 종류 및 특징정보

3장에서 분류한 바와 같이, 크로스 플랫폼 앱들을 자립형 프레임워크 앱과 하이브리드 앱, 네이티브 앱으로 분류한다. 그리고, 크로스 플랫폼 앱의 특징이 나타나지 않는 앱들의 경우 네이티브 앱 또는 미분류 상태로 분류한다.

4.1 자립형 프레임워크 앱 종류 및 특징정보

대표적인 자립형 프레임워크 앱으로 모노 런타임(mono runtime)을 사용하는 자마린 앱과 유니티 앱이 존재한다. 또한 "Write once, compile anywhere"를 표방하여 C++ 언어를 이용해 크로스 플랫폼 앱을 제작하는 qt 프레임워크로 작성된 앱도 존재한다. 또한 게임 앱은 그래픽 렌더링 작업 및 게임 처리속도의 향상이 필요하기에 네이티브 작업이 많이 필요하다. 이에 맞는 게임 개발 프레임워크인 cocos2dx, libgdx, Corona도 존재하며, 크로스플랫폼 증강현실(AR) 개발환경을 용이하게 만들어 주는 Vuforia도 존재한다. 해당 내용을 분류하면 표 2와 같다.

표 2. 자립형 프레임워크 앱의 주요 네이티브 함수(특징정보)
Table 2. major native functions (features) of self-contained framework apps

프레임워크	네이티브 함수 이름
Xamarin	Lmono/android/Runtime;init
Qt project	Lorg/qtproject/qt5/android/QtNative;keyDown Lorg/qtproject/qt5/android/QtNative;keyUp
Unity	Lcom/unity3d/player/NativeLoader;load Lcom/unity3d/player/UnityPlayer;initJni
Cocos2d-x	Lorg/cocos2dx/lib/Cocos2dxRenderer;nativeInit Lorg/cocos2dx/lib/Cocos2dxRenderer;nativeOnPause
Corona	Lcom/ansca/corona/CoronaWebView; Lcom/ansca/corona/CoronaWebView$CoronaWebViewClient; Lcom/ansca/corona/JavaToNativeShim;nativeInit
libGDX	Lcom/badlogic/gdx/physics/box2d/Body;jniApplyAngularImpulse Lcom/badlogic/gdx/physics/box2d/Body;jniSetAngularVelocity
Vuforia	Lcom/vuforia/VuforiaJNI;init Lcom/qualcomm/vuforia/VuforiaJNI;deinit

표 3의 프레임워크들은 자체 엔진을 활용하기 위해 JNI를 이용하여 자바코드에서 네이티브 함수를 등록하고 호출한다. 앞서 그림 1에서 언급한 바와 같이 자바에서 사용하는 네이티브 함수는 Java의 심볼과 C++에서 작성한 함수의 심볼이 일치해야 하기에 난독화에 의해 변조되기 어렵다.

본 논문에서 자립형 프레임워크 앱을 분류하기 위해 사용한 주요 네이티브 함수의 명은 표 2와 같으며, 본 정보들을 특징정보로 활용하여 자립형 프레임워크 앱인지 확인하고, 구체적으로 어떤 프레임워크를 이용하여 제작된 앱인지 확인했다.

표 3. 자립형 프레임워크 앱의 종류 및 특징
Table 3. types and characteristics of self-contained framework apps

프레임워크	개발언어	주요 특징	지원
Xamarin	C#(.net)	Mono 런타임 활용	Android, iOS
QT project	C++	주요 C++ 라이브러리 제공	Android, iOS
Unity	C#(.net)	Mono 런타임 활용	Android, iOS
Cocos2d-x	C++, Lua Script, JavaScript	오픈소스 2D 게임엔진	Android, iOS
Ansca Corona	Lua Script	2D 게임엔진. C, C++, Obj-C, Java의 라이브러리 또는 API를 사용할 수 있음.	Android, iOS
libGDX	Java, C. C++	자바 기반 게임 개발환경 제공	Android, iOS, BlackBerry
Vuforia	C++	증강 현실 개발 프레임워크. Unity를 위한 SDK 제공	Android, iOS

4.2 하이브리드 앱의 종류 및 특징

하이브리드 앱을 개발하는데 가장 널리 사용되는 프레임워크는 Apache Cordova이다. Apache Cordova에서 파생된 웹 기반 하이브리드 개발환경은 Onsen UI, PhoneGap, Ionic, framework7, Kendo UI 등이 존재한다. 다른 개발 환경으로는 Titanium, Appyet, react native, native script 등이 존재한다. Appyet은 경우 안드로이드만 지원하지만, 개발 방식이 기존 하이브리드 앱들과 동일하며 안드로이드 플랫폼에서 널리 사용되어 하이브리드 앱 분류에 포함했다.

이들 웹 기반 하이브리드 프레임워크의 특징은 표 4에 나타나 있다.

표 4. 하이브리드 앱의 종류 및 특징
Table 4. types and characteristics of hybrid apps

프레임워크	개발 언어	특징	지원
Apache Cordova	C#, C++, CSS, HTML, Java, JavaScript, Obj-C	많은 파생 프레임워크 존재 (Onsen UI, PhoneGap, Ionic, framework7, Kendo UI 등).	Android, iOS, BlackBerry
Appcelerator Titanium Studio	JavaScript	JavaScript로 Native UI Component API를 사용.	Android, iOS, BlackBerry, Windows
Appyet	언어 사용 X	무료 배포 및 코딩 불필요, 손쉬운 개발	Android
React Native	JavaScript	Swift, Java, Obj-C에서 작성된 Component를 결합할 수 있음. React를 사용.	Android, iOS
Native Script	JavaScript	단일 UI 작성 후, 플랫폼에 맞게 약간씩 조정. Vue.js, Angular, TypeScript 사용.	Android, iOS
Rhomobile	CCS3, HTML, JavaScript	Rhodes 오픈소스 기반 프레임워크	Android, iOS, Windows

본 연구에서는 하이브리드 앱을 분류하기 위해 WebView와 WebViewClient를 상속받는 클래스 명을 확인했다. 또한, Native Script 앱은 각 버전마다 WebView를 상속받는 클래스명이 상이하기 때문에, 네이티브 함수 정보를 활용하여 하이브리드 앱을 탐지했다. 본 논문에서 활용한 하이브리드 앱의 특징정보로 사용한 WebView, WebViewClient를 상속받는 클래스, 네이티브 함수 명은 표 5와 같다.

5. 크로스 플랫폼 앱들의 분포

본 논문에서 크로스 플랫폼 앱들의 분포를 분석하기 위해 AndroZoo[22-24]에서 제공하는 데이터를 활용하여 분석하였다. AndroZoo에서 제공하는 데이터 셋은 2014년부터 2018년까지의 데이터셋을 제공하며, 악성 앱과 정상 앱이 혼재되어 있다. 본 논문에서 사용한 앱은 총 189,800개이며, 악성 앱 31,936(26.8%)개, 정상 앱 157,864(83.2%)개 이다.

AndroZoo 데이터 셋을 구성하는 앱들의 악성 여부는 VirusTotal[25]을 활용하여 검수하였으며 이를 기반으로 malware (악성 앱, malicious apps), goodware (정상 앱, benign apps)로 분류하였다. 본 연구에서 제안한 분류기법을 사용하여 연도 별로 크로스 플랫폼 앱의 사용 빈도를 파악함으로서 크로스 플랫폼 프레임워크의 사용 경향을 분석한다. 또한 연도 별로 악성코드 중에서 크로스 플랫폼 프레임워크를 얼마나 사용하는지 파악하여, 악성코드에 대한 크로스플랫폼 별 분석 기법의 필요성을 확인한다.

5.1 정상 앱의 분류결과 및 앱 개발현황

AndroZoo에서 제공한 정상 앱(goodware) 분석결과 전체 157,864개의 앱 중에서 하이브리드 앱은 13,036(8.26%)개 자립형 프레임워크 앱은 20,284(16.29%)이며 전체 크로스 플랫폼 앱의 개수는 33,320(21.10%)개로 파악되었다. 정상 앱에 대한 자세한 분류결과는 표 6과 같다. 또한 크로스 플랫폼 앱 활용의 증가추세를 확인할 수 있는데, 2014년 크로스 플랫폼 앱의 비율의 경우 13.06% 인데 반해, 2018년의 경우 45.69%로 증가

표 5. 하이브리드 앱의 주요 특징정보
Table 5. major features of hybrid apps

프레임워크	네이티브 함수 이름 혹은 클래스 이름
Cordova	[클래스] Lorg/apache/cordova/inappbrowser/InAppBrowser$InAppBrowserClient; Lorg/apache/cordova/engine/SystemWebView;
Titanium	[클래스] Lti/modules/titanium/ui/widget/webview/TiUIWebView
Appyet	[클래스] Lcom/appyet/view/ObservableWebView; Lcom/appyet/view/observablescrollview/ObservableWebView;
React Native	[클래스] Lcom/facebook/react/views/webview/ReactWebViewManager$ReactWebView; Lcom/facebook/react/views/webview/ReactWebViewManager$ReactWebViewClient;
Native Script	[네이티브 함수] Lcom/tns/Runtime;initNativeScript Lcom/tns/AndroidJsV8Inspector;init
Rhomobile	[클래스] Lcom/rhomobile/rhodes/webview/RhoWebViewClient;

했음을 알 수 있다. 또한 하이브리드 앱의 비율은 2014년 3.23%에서 2016년 14.37%로 증가했으며 2018년 9.95%로 소폭 감소했다. 반면 자립형 프래임워크 앱의 비중은 점진적 증가한 것을 알 수 있다. 2014년 9.93%이던 앱의 비중은 2016년에 9.08%로 소폭 감소했으나, 2017년 17.71%, 2018년 35.75%로 증가하여 자립형 프레임워크의 앱의 비율이 증가하였음을 알 수 있다. 특히, 모바일 환경에서 Mono 런타임을 활용하는 경우가 점진적으로 증가하는 것을 관찰할 수 있는데, Xamarin 프레임워크의 경우 2014년 0.74%에서 2018년 6.51%로 증가하였으며, Unity의 경우 2014년 4.52%에서 2018년 23.79%로 증가하였다.

5.2 악성 앱의 분류결과 및 악성 앱 분포

AndroZoo에서 제공한 악성 앱(malware) 분석 결과 전체 31,936개의 앱 중에서 하이브리드 앱은 909(2.85%)개, 자립형 프레임워크 앱은 5,455(21.33%)이며 전체 크로스 플랫폼 앱의 개수는 6,364(19.93%)개로 파악되었다. 악성 앱에 대한 자세한 분류결과는 표 7과 같다. 정상 앱에 반해 악성 앱의 경우 2014년을 기점으로 크로스 플랫폼을 활용한 경우가 점차 줄어드는 것을 확인할 수 있는데, 2014년 17.29%이던 크로스 플랫폼 앱 비율이 2017년 4.89%, 2018년 3.12%로 감소하였음을 확인할 수 있다.

하이브리드 앱과 자립형 프레임워크 앱의 종류에 따라서 감소추세를 보인다. 악성 앱 중에서 하이브리드 앱의 비율은 2014년 7.00%, 2017년 0.54%, 2018년 0.52%로 감소하였음을 확인할 수 있다. 또한 자립형 프레임워크 앱의 비율은 2014년 17.29%, 2017년 4.32%, 2018년 2.51%로 지속적으로 감소함을 알 수 있다. 하지만 여전히 크로스 플랫폼을 이용한 악성 앱이 존재한다는 것을 알 수 있으며, 이에 대한 보안 및 저작권 보호 차원에서의 대책이 필요한 것으로 보인다.

5.3 기존 연구와 차이점

크로스 플랫폼 앱들의 분포를 파악하기 위한 연구가 기존에도 있었다[20][21]. 하지만, 기존 연구들은 실행코드를 분석하지 않았고, 압축 파일 형태로 온라인 마켓에 배포되는 APK 파일 내부의 파일명 및 디렉토리 구조들을 분석하여 크로스 플랫폼 앱들의 분포를 분석하였다. cocos2dx의 경우[20], APK 내부 특징 정보라고 할 수 있는 libcocos2dcpp.so 파일과 libunity.so 파일이 동시에 존재하는 것을 확인했다. 이 경우 dex 파일을 분석하면 구체적으로 어떤 공유 라이브러리가 실제로 사용되는지 파악할 수 있다. 따라서 본 논문에서는 바이트코드를 역공학하여 네이티브 함수와 부모 클래스(super class)의 정보를 확인하여 크로스 플랫폼 앱을 분류하였다.

6. 결론 및 향후 연구

본 논문에서는 안드로이드 앱을 개발 프레임워크에 따라 (1) 네이티브 앱(native apps), (2) 모바일 웹 앱(mobile web apps), (3) 하이브리드 앱(hybrid apps of Web and native components), (4) 자립형 프레임워크 앱(self-contained framework apps)의 4가지 타입으로 분류하였다. 이 중 크로스 플랫폼 앱인 하이브리드 앱, 자립형 프레임워크 앱에 종류에 대해 설명하였으며, 이를 분류하는 기법에 대해 서술했다. 크로스 플랫폼 앱 분류 시, 자립형 프레임워크 앱의 경우 네이티브 함수의 심볼 정보를 활용했으며, 하이브리드 앱의 경우 네이티브 함수의 심볼 정보와, 웹킷의 주요 클래스를 상속 받는 주요 클래스들의 심볼을 확인하여 크로스 플랫폼 앱을 식별하

표 6. 각 크로스 플랫폼 개발 프레임워크로 작성된 정상 앱 분포
Table 6. distribution of benign apps written with each cross-platform development framework

		2014	2015	2016	2017	2018
hybrid apps	Cordova	1036(2.13%)	2815(5.91%)	2516(8.56%)	2050(10.19%)	612(5.10%)
	React Native	0(0.00%)	1(0.00%)	76(0.26%)	112(0.56%)	38(0.32%)
	Appyet	2(0.00%)	232(0.49%)	1444(4.92%)	294(1.46%)	349(2.91%)
	Titanium	530(1.09%)	349(0.73%)	183(0.62%)	183(0.91%)	192(1.60%)
	Native Script	-	-	1(0.00%)	4(0.02%)	1(0.01%)
	rhomobile	6(0.01%)	5(0.01%)	2(0.01%)	2(0.01%)	1(0.01%)
Total no. of hybrid apps		1574(3.23%)	3402(7.14%)	4222(14.37%)	2645(13.15%)	1193(9.95%)
self-contained runtime environment apps	Corona	519(1.07%)	315(0.66%)	126(0.43%)	142(0.71%)	150(1.25%)
	Cocos2dx	1064(2.18%)	1090(2.29%)	514(1.75%)	741(3.68%)	376(3.13%)
	Unity	2202(4.52%)	2688(5.64%)	1556(5.30%)	2024(10.06%)	2854(23.79%)
	Xamarin	361(0.74%)	275(0.58%)	216(0.74%)	412(2.05%)	781(6.51%)
	Qt project	35(0.07%)	44(0.09%)	42(0.14%)	35(0.17%)	21(0.18%)
	Vuforia	18(0.04%)	23(0.05%)	15(0.05%)	6(0.03%)	1(0.01%)
	libGDX	588(1.21%)	545(1.14%)	196(0.67%)	203(1.01%)	105(0.88%)
	LongRange	-	-	1(0.00%)	-	-
Total no. of self-contained runtime environment apps		4787(9.83%)	4980(10.45%)	2666(9.08%)	3563(17.71%)	4288(35.75%)
Total no. of cross-platform apps		6361(13.06%)	8382(17.58%)	6888(23.45%)	6208(30.86%)	5481(45.69%)
Unknown or Native apps		42348(86.94%)	39281(82.41%)	22491(76.55%)	13910(69.14%)	6514(54.31%)
Total		48709	47663	29379	20118	11995

표 7. 각 크로스 플랫폼 개발 프레임워크로 작성된 악성 앱 분포
Table 7. distribution of malicious apps written with each cross-platform development framework

		2014	2015	2016	2017	2018
hybrid apps	Cordova	800(6.99%)	26(0.26%)	29(0.74%)	26(0.54%)	7(0.40%)
	React Native	-	-	2(0.05%)	-	1(0.06%)
	Appyet	-	12(0.12%)	-	-	1(0.06%)
	Titanium	1(0.01%)	4(0.04%)	-	-	-
	Native Script	-	-	-	-	-
	rhomobile	-	-	-	-	-
hybrid apps Total		801(7.00%)	42(0.42%)	31(0.79%)	26(0.54%)	9(0.52%)
self-contained runtime environment apps	Corona	32(0.28%)	32(0.32%)	5(0.13%)	7(0.15%)	0(0.0%)
	Cocos2dx	582(5.09%)	663(6.60%)	209(5.33%)	39(0.82%)	10(0.57%)
	Unity	1004(8.77%)	1486(14.79%)	364(9.28%)	143(3.00%)	28(1.60%)
	Xamarin	5(0.04%)	5(0.05%)	16(0.41%)	5(0.10%)	1(0.06%)
	Qt project	-	1(0.01%)	-	-	-
	Vuforia	2(0.02%)	1(0.01%)	-	-	-
	libGDX	354(3.09%)	357(3.55%)	87(2.22%)	12(0.25%)	5(0.29%)
	LongRange	-	-	-	-	-
Total no. of self-contained runtime environment apps		1979(17.29%)	2545(25.33%)	681(17.36%)	206(4.32%)	44(2.51%)
Total no. of cross-platform apps		2780(24.29%)	2587(25.75%)	712(18.15%)	232(4.89%)	53(3.12%)
Unknown or Native apps		8664(75.71%)	7459(74.25%)	3210(81.85%)	4539(95.14%)	1700(96.98%)
Total		11444	10046	3922	4771	1753

고 분류했다.

실험을 위해 AndroZoo 데이터 셋을 이용하였으며, 정상 앱으로 분류된 데이터 셋의 경우 크로스 플랫폼 앱이라고 파악된 개수가 전체 157,864개 중 33,320(21.10%)개로 파악되었으며, 악성 앱으로 분류된 데이터 셋의 경우 크로스플랫폼 앱이라고 파악된 앱의 개수가 전체 31,936개 중 6,364(19.93)개로 파악되었다.

향후에는 본 기법을 AndroZoo 외에 다른 데이터 셋에도 적용하여 검증하고 검증결과를 기반으로 다양한 프레임워크에 대한 분류 기법을 추가 제시하여 기법의 효용성을 높일 예정이다.

1) 이 연구는 2018년도
정부(과학기술정보통신부)의 재원으로
한국연구재단의 지원을 받아 수행된
기초연구사업임(no. 2018R1A2B2004830)
2) 본 연구는 과학기술정보통신부 및
정보통신기술진흥센터의
대학ICT연구센터육성지원사업의 연구결과로
수행되었음(IITP-2018-2015-0-00363)

참 고 문 헌

[1] Henning Heitkötter, Sebastian Hanschke, and Tim A. Majchrzak, “Evaluating cross-platform development approaches for mobile applications”, International Conference on Web Information Systems and Technologies, Springer, pp.120-138, 2012. DOI: https://doi.org/10.1007/978-3-642-36608-6_8

[2] Isabelle Dalmasso, Soumya Kanti Datta, Christian Bonnet, and Navid Nikaein, “Survey, comparison and evaluation of cross platform mobile application development tools”, IEEE 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC), pp.323-328, 2013. DOI: https://doi.org/10.1109/IWCMC.2013.6583580

[3] Spyros Xanthopoulos and Stelios Xinogalos, “A comparative analysis of cross-platform development approaches for mobile applications”, Proceedings of the 6th Balkan Conference in Informatics, ACM, pp.213-220, 2013. DOI: https://doi.org/10.1145/2490257.2490292

[4] Yonathan Aklilu Redda, “Cross platform Mobile Applications Development: Mobile Apps Mobility”, MS thesis. Institutt for datateknikk og informasjonsvitenskap, Norwegian University of Science and Technology, 2012. https://core.ac.uk/download/pdf/52105182.pdf

[5] William Lee and Xinran Wu, “Cross-platform mobile malware, write once, run everywhere”, Virus Bulletin Conference, 2015. https://www.virusbulletin.com/conference/vb2015/abstracts/cross-platform-mobile-malware-write-once-run-everywhere

[6] Santiago M. Pontiroli and F. Roberto Martinez, “The Tao of .NET and PowerShell Malware Analysis”, Virus Bulletin Conference, 2015. https://www.virusbulletin.com/conference/vb2015/abstracts/tao-net-and-powershell-malware-analysis

[7] Jaewoo Shim, Kyeonghwan Lim, Seong-je Cho, Sangchul Han, and Minkyu Park, “Static and Dynamic Analysis of Android Malware and Goodware Written with Unity Framework”, Security and Communication Networks, 2018. https://doi.org/10.1155/2018/6280768

[8] Alexios Mylonas1, Stelios Dritsas, Bill Tsoumas, and Dimitris Gritzalis, “On the feasibility of malware attacks in smartphone platforms”, International Conference on E-Business and

Telecommunications, Springer, pp.217-232, 2011. https://link.springer.com/chapter/10.1007/978-3-642-35755-8_16

[9] Alexios Mylonas1, Stelios Dritsas, Bill Tsoumas, and Dimitris Gritzalis, “Smartphone security evaluation - the malware attack case”, Proceedings of the International Conference on Security and Cryptography (SECRYPT), IEEE, pp.25-36, 2011.

[10] Google, “Android SDK”, http://developer.android.com/sdk/installing/index.html

[11] Google, “Android NDK”, https://developer.android.com/ndk

[12] IBM 소프트웨어 Throught Leadership 백서, “모바일 앱 개발 방식 비교: 네이티브, 웹, 하이브리드”, ftp://ftp.software.ibm.com/software/kr/pdf/WSW14182_Native_Web_or_Hybrid_White_Paper.pdf

[13] Naoki Kudo, Toshihiro Yamauchi, and Thomas H. Austin, “Access control mechanism to mitigate Cordova plugin attacks in hybrid applications“, Journal of Information Processing, pp.396-405, Vol.26, 2018. DOI: https://dx.doi.org/10.2197/ipsjjip.26.396

[14] Achim D. Brucker and Michael Herzberg, “On the static analysis of hybrid mobile apps”, International Symposium on Engineering Secure Software and Systems, Springer, Cham, pp.72-88, 2016. DOI: https://doi.org/10.1007/978-3-319-30806-7_5

[15] Gerald Versluis, “A Brief History of Xamarin”, Xamarin. Forms Essentials, Apress, Berkeley, CA, pp.3-18, 2017. DOI: https://doi.org/10.1007/978-1-4842-3240-8_1

[16] Matias Martinez and Sylvain Lecomte, “Towards the quality improvement of cross-platform mobile applications”, 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft), pp.184-188, 2017. DOI: https://doi.org/10.1109/MOBILESoft.2017.30

[17] Ceferino Gabriel R. Ramirez, Jason B. Almonte, Reynaldo R. Tugade, and Rowel O. Atienza, “Implementation of a digital game-based learning environment for elementary Education”, 2010 2nd International Conference on Education Technology and Computer, IEEE, Vol. 4, pp.208-212, 2010. DOI: https://doi.org/10.1109/ ICETC.2010.5529699

[18] Jinseok Seo and Hun Choi, “A Case Study on One-Source Multi-Platform Mobile Game Development Using Cocos2d-x”, International Journal of Engineering and Applied Sciences (IJEAS), Vol. 3, Issue 11, pp.80-84, Nov. 2016. https://www.ijeas.org/download_data/IJEAS0311032.pdf

[19] Sonali Kothari Tidke, Pravin P. Karde, and Vilas Thakare, “Detection and Prevention of Android Malware Thru Permission Analysis”, 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), IEEE, pp.1-6, Aug. 2019. DOI: https://doi.org/10.1109/ICCUBEA.2018.8697490

[20] 전소연, 김병철, 조성제, “안드로이드에서 교차 플랫폼 모바일 앱 개발 프레임워크 및 악성 앱 동향”, 한국소프트웨어감정평가학회 춘계학술대회, May 2019.

[21] 오지환, 이명건, 조성제, 한상철, “주요 안드로이드 마켓에서 앱 카테고리 및 개발 도구에 따른 악성 앱 분포”, 정보과학회논문지, 46권 2호, pp.109-115, 2019. DOI: https://doi.org/10.5626/JOK.2019.46.2.109

[22] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon, “Androzoo: Collecting millions of Android apps for the research community”, 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR), IEEE, pp.468-471, 2016. DOI: https://doi.org/10.1109/MSR.2016.056

[23] AndroZoo home, Université du Luxembourg, https://androzoo.uni.lu/

[24] Li Li, Jun Gao, M´ed´eric Hurier, Pingfan Kong, Tegawend´e F. Bissyand´e, Alexandre Bartel, Jacques Klein, and Yves Le Traon, “Androzoo++: Collecting millions of android apps and their metadata for the research community”, arXiv preprint arXiv:1709.05281, 2017. https://arxiv.org/pdf/1709.05281.pdf
[25] VirusTotal (free virus, malware and URL online scanning service). https://www.virustotal.com/gui/home/upload

저 자 소 개

김규식(Gyoosik Kim)

2016.2 단국대학교 응용컴퓨터학과 졸업
2018.2 단국대학교 컴퓨터학과 졸업
<주관심분야> 시스템 보안, 시스템 소프트웨어, 임베디드 시스템, 안드로이드 프레임워크

전소연(Soyeon Jeon)

2016.2 단국대학교 소프트웨어학과 학사과정
<주관심분야> 안드로이드 보안, 데이터마이닝, 머신러닝

조성제(Seong-je Cho)

1989.2 서울대학교 컴퓨터공학과 졸업
1991.2 서울대학교 컴퓨터공학과 공학석사
1996.8 서울대학교 컴퓨터공학과 공학박사
2009.02-2010.02 Univ. of Cincinnati 방문교수
1997.3-현재 : 단국대학교 컴퓨터학과/소프트웨어학과 교수
<주관심분야> 소프트웨어 지적 재산 보호, 시스템 및 모바일 보안, 악성코드 분석, 시스템소프트웨어