모바일 핸드셋의 정보 보호 현황 및 전망

문 성 원*

1. 서 론

모바일 프로세서와 무선 통신의 성능이 발전함에 따라서 핸드셋 고유의 기능인 통화 및 문자는 물론 PC 환경처럼 다양한 어플리케이션을 설치하고 그 서비스를 이용할 수 있는 환경이 마련되었다. 또한 탭 북과 같이 기존의 PC와 비슷한 성능을 유지하면서도 Mobility를 제공하는 제품 또한 활성화 되고 있다. 그 결과, 모바일 장치에서도 데스크 탑에서만 느낄 수 있었던 프로세스와 통신 성능, 그리고 다양한 어플리케이션에 대한 경험을 할 수 있는 환경이 실현되고 있다.

이처럼 스마트 폰이나 탭 북을 통한 손안의 PC 환경이 실현됨에 따라서 정보 보호의 중요성은 점점 강조되고 있다. 과거 Feature Phone 개발 환경에서는 제조사에서 제작한 인증된 어플리케이션만 실행 가능했었지만, 스마트폰의 경우에는 누구나 어플리케이션을 개발하고 실행할 수 있는 환경이 구축된다. 이런 환경에서는 Device 자체와 설치되는 어플리케이션의 안정성이 모두 중요하다.

그 뿐만 아니라, 최근에는 모바일 OS로서 Embedded Linux 기반의 운영 체제가 대다수를 이끌고 있으면 앞으로도 Linux를 기반으로 한 운영체제가 주를 이루고 있다. 그런데 Linux는 오픈소스로서 Kernel과 같은 핵심 코드가 공개되고 있기 때문에 Linux를 기반으로 하는 Smart Phone에서는 시스템을 더 철저히 분석하고 정보보호를 강화할 수 있도록 검토해야 한다.

모바일 디바이스는 다음과 같은 특성 때문에 정보 보호의 중요성이 더 중요하다.

첫 번째로 핸드셋은 Mobility를 제공하기 때문에 PC 환경에 비해서 사용자에 밀접하게 종속된 장치이다. 특히, 사용자의 위치 정보를 수집하고 이 정보를 이용하는 어플리케이션이 풍부하다. 두 번째로 개인의 사생활과 관련된 많은 정보가 저장된다. 연락처, 사진, 동영상, 사용자가 구매한 컨텐츠, 어플리케이션 그리고 계정 정보 등 사용자와 관련된 주요한 정보를 저장하고 있다. 세 번째로는 모바일을 통한 결재 시스템이 확산되는 추세에 있다는 점이다. 모바일 뱅킹 시스템은 물론 문자를 통한 개인 인증 및 결제가 가능하기 때문에 정보 보호의 중요성이 더욱 강조되고 있다.

2012년 말 여러 통계 조사에 따르면 스마트폰에 사용되는 OS의 점유율은 Android가 70% 이상을 차지했다고 한다. Android OS뿐만 아니라 최근에 신규로 발표되고 있는 Tizen, FireFox, Ubuntu 역시 Embedded Linux를 기반으로 하고

※ 교신저자(Corresponding Author) : 문성원, 주소 : 서울시 금천구 가산동 517-4번지 LG전자 MC-C연구소, 전화: +82-10-4715-3137, E-mail : sungwon.moon@lge.com
* LG전자 MC연구소 선임연구원

있어서 Open OS에 대한 안정성 확보가 매우 중요하다고 볼 수 있다.

1장에서는 모바일 폰에서 필수적으로 개발되어야 하는 Secure Bootloader에 대해서 설명하고 2장에서는 ARM 사의 Trust Zone에 대해서 소개한다. Trust Zone의 경우에는 Cortex™-A부터 소개가 되었으나 실제 디바이스 개발에 적용되기 시작한 것은 비교적 최근이며 아직 활성화가 이루어진 상태는 아니다. 3장에서는 Linux 커널 및 이를 기반으로 하는 안드로이드 플랫폼에서 제안하고 있는 Security 기술에 대해서 설명한다. 4장에서는 결론과 스마트 폰에서의 정보 보호 기술의 향후 전망에 대해서 예상해 본다.

2. 디바이스 정보 보호 기술

2.1 Secure Boot

Embedded Linux를 기반으로 하는 Android 디바이스를 개발하는데 있어서 Chipset에서 제공하는 Secure Bootloader를 활성화하는 것은 매우 중요한 개발 항목 중의 하나이다. 제조업체에서는 여러 이해 관계자들의 요구 사항에 따라서 개발된 SW 플랫폼이 안전하게 동작할 수 있도록 시스템의 무결성을 보장할 수 있어야 한다. 초창기의 Android 폰에서는 Custom ROM이라는 명칭으로 해당 폰에서 동작할 수 있는 비공식적인 SW 이미지가 발행되어 사용되기도 했었다. 결국, Secure Boot 기능 없이 다양한 Flashing 기법이 제공된 결과였다.

그러나 공식 SW가 아닌 경우, 공식적인 인증 절차를 거치지 않는 SW이기 때문에 디바이스의 오동작을 일으킬 확률이 높다. 이로 인해서 제조사의 서비스 비용 높아지고, 사용자 또한 피해를 입을 수 있다. 또한 비공식 악성 SW가 사용자 모르게 설치될 수 있으므로 시스템이 외부 공격에 더 취약할 수 있다.

이러한 이유로 어떤 식으로든지 시스템의 무결성을 제공하는 것은 매우 중요하다. 주로 Hash나 RSA알고리즘을 이용해서 원본 이미지에 대해서 Signature 데이터를 특정 메모리 영역에 생성하고 부팅 시에 각 단계에서 해당 Signature를 체크하는 방식을 이용하고 있다.

그림 1에서 볼 수 있는 것처럼, 칩셋 제조사(Qualcomm, Nvidia)에서 BOOT ROM을 개발할 때 자신의 Boot Loader를 안전하게 인증할 수 있는 방안을 마련한다. 이 때, 부트 로더는 여러 단계로 이루어질 수 있으며 Android Boot loader 이미지까지의 무결성을 제공해야 한다. 그 후에 제조사에서는 Android의 부트로더부터 Kernel 그리고 안드로이드 주요 컴포넌트에 대해서 비정상적인 SW의 수정을 막을 수 있는 방안을 제공한다.

그림 2에서는 부트 로더에서 Kernel 이미지의 무결성을 체크하기 위한 인증서 생성 방법의 한 예를 보여주고 있다. 빌드 과정을 통해서 커널 이미지가 생성되면 빌드의 마지막 단계에서 제조사의 인증서 생성 툴을 이용해서 부팅시 무결성 체

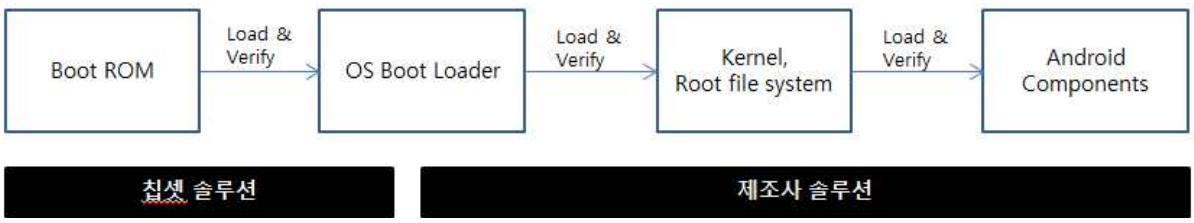


그림 1. Secure Boot 순서도

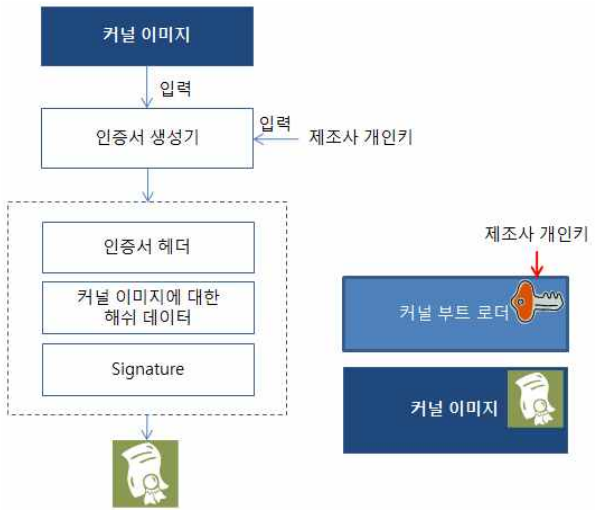


그림 2. Kernel 이미지에 대한 인증서 생성기

크를 위해서 사용될 인증서를 생성한다. 해쉬 알고리즘을 이용하여 Kernel 이미지에 대한 해쉬값을 생성하고 RSA를 이용하여 제조사에서 생성한 개인키를 이용해서 Signature를 생성하고 이에 대한 인증서 헤더를 생성해서 붙인다. 이렇게 생성된 인증서는 Kernel 이미지에 첨부되며 폰이 부팅될 때마다 커널 부트로더가 미리 생성한 제조사의 공개키를 이용해서 미리 생성된 인증서를 이용하여 Kernel 이미지의 무결성을 검증한다.

2.2 Trust Zone

대표적인 Embedded 칩셋 제조업체인 ARM 사에서는 Cortex™-A부터 TrustZone이라고 하는 TEE(Trusted Execution Environment)를 제공하고 있다.

TrustZone 기술은 모바일 결제, DRM 그리고 Web OS와 같이 정보 보호가 중요한 어플리케이션이 안전하게 실행될 수 있도록 시스템 및 하드웨어적인 환경을 제공하는 것이 목표이다. 기본적으로 ARM 칩에 TrustZone이 포팅되어 별도의 HW 모듈의 추가 없이 하드웨어 기반의 TEE 환경을 제공할 수 있다.

그림 3은 TrustZone의 하드웨어 아키텍처이다. 단일 프로세서 상에서 가상화(virtualization)를 통해서 시스템 리소스를 Normal World와 Secure World 두 개의 모드로 분리된다. Monitor Mode는 Secure World에서 동작하는 프로세스로 현재의 Context를 결정하는 역할을 한다. AMBA (Advanced Microcontroller Bus Architecture) 3 AXITM bus 라고 하는 하드웨어 로직은 Normal Mode의 컴포넌트가 Secure World의 리소스에 접근하지 못하도록 차단한다.

이를 통해서 DRM이나 결제와 같은 보안과 직결되는 중요한 프로그램의 실행과 일반적인 어플리케이션의 실행을 분리할 수 있다. Secure World의 어플리케이션은 Normal World의 resource에 대한 접근이 가능하지만 Normal World의 어플리케이션은 Secure World의 리소스를 사용할 수 없도록 설계된다.

그림 4는 주변 장치에 대해서 TrustZone을 연동 방법에 대해서 설명하고 있다. APB bus는 AXI bus에 비해서 간단하고 소모 전류가 적기 때문에 ARM 시스템은 APB(Advanced Peripheral Bus) bus를 이용해서 주변장치를 컨트롤 한다. AXI-to-APB 브리지가 AXI bus와 APB bus를 연결하는 중간자 역할을 수행하여 AXI transaction을 APB 주변 장치를 선택할 수 있도록 해

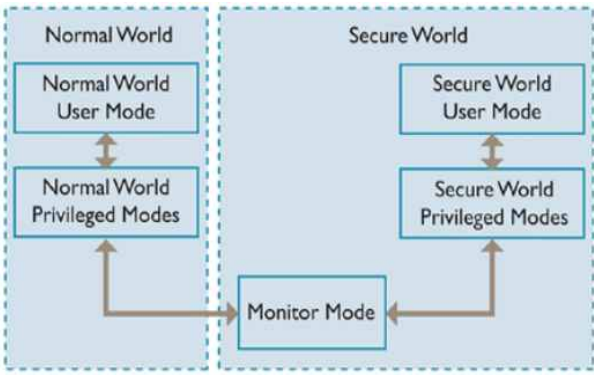


그림 3. TrustZone Hardware Architecture

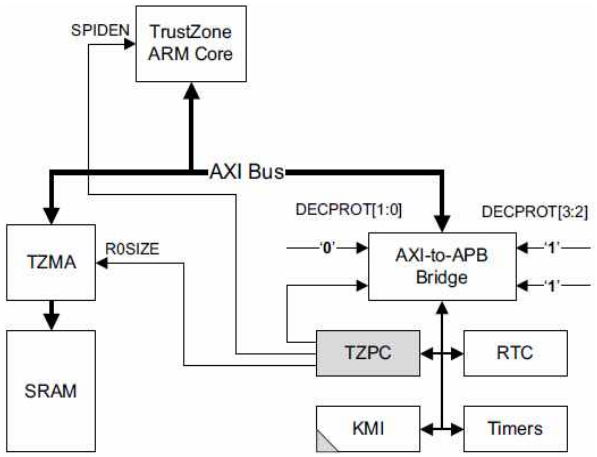


그림 4. TZPC(TrustZone Protection Controller)

석한다. 해당 버스에 장착되어 있는 주변장치 각각에 대한 TZPCDECPROT를 가지고 있어서 해당 장치가 Secure Mode 인지 Non-Secure Mode 인지 설정하는 역할을 한다. AXI-to-APB 브리지는 Normal World의 작업이 Secure 장치로 실행되는 경우에는 해당 작업을 거절한다. TZPC는 run-time에 동적으로 Security status를 변경하는 역할을 수행한다.

그림 4에서 회색으로 표기된 TZPC는 항상 Secure 모드 주변 장치이고 KMI(Keboard and Mouse Interface)는 두 모드로 변환될 수 있는 장치를 나타낸다. 나머지 흰색으로 그려진 RTC, Timers, SRAM 등은 항상 Normal 모드 장치이다. 즉, 본 예제에서는 KMI 디바이스만 두 모드로 전환할 수 있는 장치이다. 일반적인 경우에는 Normal 모드의 장비로 사용되다가 Secure World의 주요 Application(패스워드 입력 등)이 실행될 때에는 Secure Mode로 동작한다.

지금까지 TrustZone의 하드웨어 장치 기법에 대해서 설명했지만, Software 측면에서 TrustZone을 이용하는 OS 또한 연동이 되어야 TrustZone HW를 이용할 수 있다. 그림 5는 ARM사에서 제안하는 소프트웨어 아키텍쳐의 사용 예이다. 그림에서 볼 수 있는 것처럼, Normal World의 OS외에 별도로 가상화된 Secure World에서 동작하는 OS를 구성한다. Monitor의 역할은 두 가상 프로세서간의 Context를 관리하는 것이다. Monitor는 TrustZone driver의 Secure Monitor Call(SMC) 혹은 각 디바이스의 HW 예외 처리 요청에 의해서 동작한다. 두 모드 간의 Context를 관리하는 Monitor 모드는 안전하게 보호되어야하기 때문에 Secure World에서 실행되어야 한다.

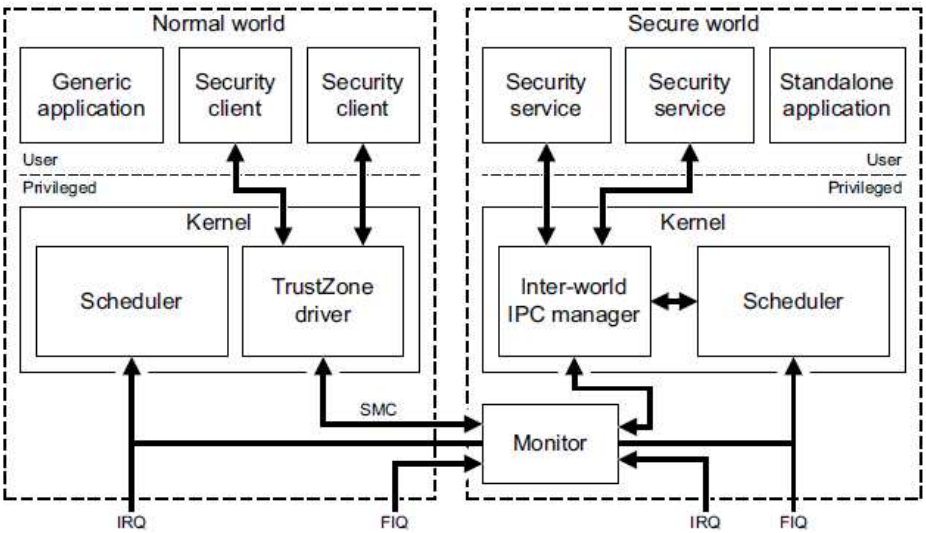


그림 5. Secure World를 가지는 가능한 SW Architecture

3. Android Platform Security

구글에서는 Android 어플리케이션 생태계를 구축하기 위해는 Security가 매우 중요하다는 것을 인식하고 있고 안전한 어플리케이션 실행 환경을 구축하기 위해서 노력하고 있다. 본 장에서는 Android 어플리케이션 구조에 대해서 간단히 알아보고 Android에서 제공하고 있는 Security 서비스에 대해서 설명한다.

3.1 Linux Kernel Security

Android는 Linux Kernel을 기반으로 하는 OS이므로 Linux에서 제공하는 Security 모델을 활용하고 있다. 사용자 기반의 Permissions Model, Process Isolation, 프로세스 간의 안전한 통신을 위한 Secure IPC를 제공한다. 그리고 linux 커널의 불필요한 부분을 삭제하는 것을 포함한다.

3.2 Application Security

안드로이드 어플리케이션의 주요 항목은 다음과 같은 요소가 있다.

• AndroidManifest.xml: 모든 어플리케이션은 하나의 Manifest 파일을 가지고 있으며, 해당 어플리케이션이 어떤 기능을 수행할 것인지에 대해서 기술하는 파일이다. 또한 어플리케이션이 접근할 수 있는 특정 시스템 리소스에 대한 Permissions이 기술된다.

• Activities: 어플리케이션의 Entry Point이며 핸드셋에서 특정 화면의 기능 단위이다. 일반적으로 사용자를 위한 User Interface를 포함하는 어플리케이션이다. 하나의 어플리케이션 화면에 하나의 Activity가 존재한다.

• Services: 서비스의 개념은 Android 어플리케이션에서 백 그라운드로 동작하는 기능이다. Services는 해당 어플리케이션의 영역에서만 동작하거나 다른 어플리케이션의 Context에서도 동작이 가능하다. 서비스의 주요 사용 예는 MP3 플레이어 어플리케이션이다. 음악 플레이어에서는 음악 파일의 디코딩과 플레이를 Service 형태로 제공하고 있어서 사용자가 Activity를 종료하더라도 백 그라운드 형태로 음악을 들을 수 있다.

• Broadcast Receiver: Broadcast Receiver는 OS나 다른 어플리케이션에 의해서 Intent 라고 하는 IPC를 통해서 이벤트가 발생할 때, 생성되는 객체이다. Low battery 메시지 같은 것이 실제 예가 될 수 있으며, 이를 처리하고자 하는 Android 어플리케이션은 이 Broadcast 이벤트를 받기 위한 Broadcast Receiver를 등록해야 한다.

앞에서 설명한 안드로이드 어플리케이션의 Security를 보장하기 위해서 Sandbox 개념을 도입하였다. 이는 Linux의 Process Isolation과 비슷한 개념이지만, Process에 대한 Isolation은 아니고 Android의 어플리케이션 단위의 개념이다. 시스템이 모든 Android 어플리케이션에 대해서 UID를 지정하고 각 어플리케이션에 대해서 메모리 공간을 확보한다. 기본적으로는 어플리케이션은 서로 통신할 수 없고 Operating System에 대해서 제한적인 접근 권한만을 가지고 있다.

어플리케이션의 OS에 대한 접근 권한은 AndroidManifest.xml 파일의 데이터를 이용한다. 사용자가 Google Play로부터 어플리케이션을 다운로드 요청을 하면, 그 어플리케이션 apk에 포함되어 있는 AndroidManifest.xml 파일의 Permissions을 사용자에게 보여주고 어플을 설치할 것인지 말 것인지에 대해서 의사 결정할 수 있도록 도와준다(그림 6).

최근에는 실제 어플리케이션이 설치되고 나면 사용자에 보고된 권한에 비해서 과도한 Resources를 허용되는 경우가 보고되고 있다. 허용되지

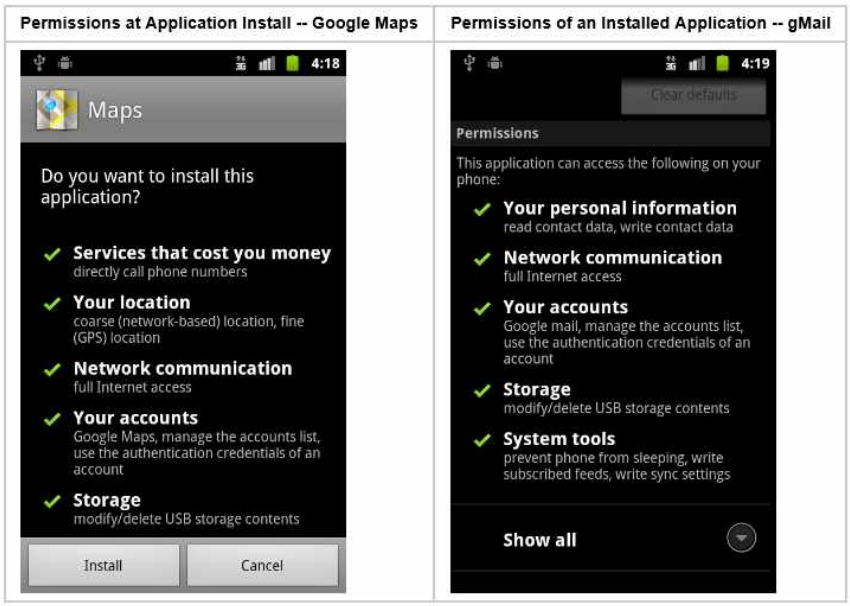


그림 6. 어플리케이션 Permissions

않은 SMS의 사용이나 데이터 망을 접근하는 경우 금전적으로도 피해를 줄 수도 있다. 그 뿐만 아니라 심지어 사용자에게 모든 권한에 대해서 Manifest 파일에 작성하여 사용자에게 보고했다고 하더라도, 사용자는 어플리케이션을 설치할 때 요구하는 권한을 자세히 살펴보지 않는 경우가 많다. 따라서 사용자에게 좀 더 어필이 가능한 Permissions 정책이 필요할 것으로 보인다.

3.3 기타 Security 기능

File system에 대한 Encryption 이나 VPN 과 같은 Security를 위한 추가 기능을 제공한다. Android 3.0 이상에서는 사용자가 파일 시스템 전체에 대해서 Encrypt/Decrypt 할 수 있는 기능을 제공한다. Linux Kernel에서 제공하고 있는 AES와 같은 대칭키 알고리즘을 이용해서 파일 시스템 전체에 대해서 암호화 기능을 제공한다.

PPTP, L2TP, IPsec 기반의 VPN 기능을 제공한다. 특히, Android 4.0에서는 VpnService 라고 하는 클래스를 제공하여 Third-party VPN solutions을 사용할 수 있도록 하였다.

마지막으로 Credential 저장소를 지원한다. 이는 미리 정의된 인증서를 저장할 수 있는 공간이다. Android 4.0 이 후에는 미리 생성되어 있는 인증서 삭제가 가능하며 사용자가 새로 생성한 인증서를 추가하는 것도 가능하다.

4. 결론 및 향후 전망

본 문서에서는 모바일 디바이스에서 고려되고 있는 주요 Security Feature에 대해서 살펴보았다. Boot ROM 기반의 Secure Boot 기능은 대부분은 스마트폰에 대해서는 적용이 되고 있고 필수적인 Feature로 고려되고 있다. 제 3의 개발자 집단에서는 Secure Boot를 적용할 경우, Custom ROM을 사용할 수 없다는 것에 대해서 불만을 표출한 바 이에 대한 Unlock 방안도 적용되고 있지만, 이에 따른 부작용도 분명히 있을 수 있기 때문에 정책적인 해결책도 필요할 것으로 보인다.

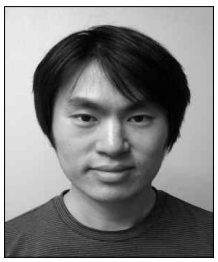
TrustZone은 ARM 칩에서는 오래 전부터 지원하고 있으나, 아직 이를 제대로 활용할 수 있는

Software Architecture는 활성화 되어 있지 않은 상태이다. 앞으로 핸드셋을 이용한 결제 시스템 및 뱅킹 시스템이 더욱 활성화 될 것이기 때문에 TrustZone을 제대로 활용할 수 있는 OS의 개발이 필수적이라고 할 수 있겠다.

OS 측면에 있어서는 디바이스 리소스의 접근 권한에 대한 고민이 필요하다. 현재로서는 사용자에게 접근 권한에 대한 허용 여부를 허락을 받고 있으나 일반 사용자 중에서 어플리케이션의 권한에 대해서 관심을 가지는 사람이 많지 않다. 그러므로 접근 권한에 대한 경각심을 가질 수 있도록 좀 더 명확한 방법이 필요할 것으로 보인다.

참 고 문 헌

[1] http://www.arm.com/products/processors/technologies/trustzone.php
[2] ARM Security Technology -Building a Secure System using TrustZone® Technology
[3] 백광호, 강동호, 김기영, “SEE 분야의 연구 및 기술 동향,” 전자통신동향분석, 2007년
[4] Enck, William, Machigar Ongtang, and Patrick McDaniel. “Understanding android security.” Security & Privacy, IEEE 7.1 (2009): 50-57.
[5] T. Alves and D. Felton, “Trustzone: Integrated Hardware and Software Security,” ARM white paper, July 2004.
[6] T. Halfhill, “ARM Dons Armor: TrustZone Security Extensions Strengthen ARMv6 Architecture,” Microprocessor Report, 2003.
[7] http://source.android.com/tech/security/



문 성 원

• 1998년~2003년 고려대학교 물리/컴퓨터 학사
• 2003년~2005년 고려대학교 전산학 석사
• 2005년~현재 LG전자 MC연구소 선임연구원
• 관심분야 : 정보보호, 모바일 플랫폼, Embedded Linux, Android Security, Linux Security