情 報 保 護 學 會 誌
第23卷 第2號, 2013. 4

안드로이드 기반의 보안 프로그램의 성능 향상 방안

남 진 하*, 이 정 민**, 이 성 근***

요 약

전 세계적으로 스마트폰의 시장 점유율은 지속적으로 증가하고 있다. 국내도 해당 기기의 사용 편의성과 이를 이용할 수 있는 다양한 앱과 서비스의 출시로 개인 사용자와 기업, 그리고 정부를 포함한 공공기관에서 스마트폰 사용이 크게 늘어나고 있다. 스마트폰의 이용실태를 살펴보면 개인의 경우 인터넷 게임, SNS, 금융 서비스를 중심으로 이용하고 있으며, 기업, 정부와 공공기관도 업무 효율화에 적극적으로 활용하고 있다. 이와 같은 개인화된 기기인 스마트폰을 이용한 사용자의 이용확대와 다양한 서비스의 증가로 인해 사용자의 공인인증서와 같은 민감한 개인 정보를 포함해서 기업, 공공기관의 정보 자산이 해당 기기에 저장 되고 있는데, 개인 정보와 가치 있는 정보를 악의적인 방법을 이용해서 유출 및 수집 하여 불법적인 이득을 얻으려는 보안 위협 및 침해가 꾸준히 증가하고 있다. 이와 같은 보안 위협의 지속적인 증가로부터 스마트폰 환경에서 사용자를 안전하게 보호하기 위한 보안 프로그램이 꾸준히 출시되고 있다. 스마트폰의 경우는 기존의 PC 보다 컴퓨팅 파워가 낮고, 실행 환경이 제약적이기 때문에 성능을 고려한 보안 프로그램의 개발이 필요하다. 이에 본 고에서는 안드로이드 기반의 보안 프로그램을 개발할 때 고려 할 수 있는 성능 향상 방법을 살펴보도록 한다.

Ⅰ. 서 론

2010년부터 본격적으로 국내에 보급된 스마트폰은 개방적 플랫폼과 개인화된 서비스를 기반으로 시간과 공간에 제약 없이 게임, SNS, 금융 등 다양한 서비스를 이용할 수 있는 장점으로 가지고 있어 빠르게 사용자의 생활에 필수품으로 자리를 잡고 있다. 방송통신위원회가 2012년 하반기에 발표한 국내 스마트폰 이용 실태[1]를 보면 가입자 수는 이미 3천 2백만여 명을 넘었으며, 사용자의 무선 인터넷과 모바일 앱의 이용빈도는 48.8%로 기존의 휴대폰에서 주요 기능이었던 음성 통화의 이용빈도(34.7%)보다 약 14% 가 높으며, 이용하는 주요 이유도 다양한 응용 프로그램의 이용(66.2%)과 인터넷 수시 이용(52.4%)으로 개인의 활용 측면에서 보면 기존의 휴대폰은 통화, 문자 위주로 사용했다면, 스마트폰은 게임을 포함한 응용 프로그램과 인터넷을 중심으로 이용하려는 경향이 높은 것으로 나타났다. 또한, 기업, 정부 그리고 공공기관은 스마트폰의 전면적인 보급으로 개인의 스마트폰을 이용해 업무의 효율화를 통한 생산성 향상과 비용 절감을 목적으로 업무에 적극적으로 도입하고 있다.

스마트폰을 구성하는 요소 중에 가장 중요한 부분은 하드웨어와 소프트웨어로 나누어 볼 수 있는데, 그 중 소프트웨어에서 운영체제가 차지하는 위치는 중요하다. 스마트폰의 운영체제의 시장 점유율을 보면 다른 운영체제 보다 안드로이드와 iOS 상대적으로 높은데, 안드로이드는 세계 시장에서는 50%이상의 점유율을 가지고 있으며, 국내 시장의 점유율은 약 90%로 대부분의 국내 사용자는 안드로이드 기반의 스마트폰을 사용하고 있다. 또한, 안드로이드 플랫폼은 개방형 플랫폼과 별도의 라이선스 비용이 없는 장점 때문에 스마트폰을 포함해서 TV 셋톱박스, VOI폰 그리고 의료기기를 포함한 광범위한 범위에서 활용되고 있다.

이와 같은 개방적인 플랫폼을 탑재한 스마트폰의 이용이 지속적으로 증가하면서, 기존의 PC가 가지고 있던 보안 위협 뿐 만 아니라 자체 기기의 특징을 악용한 해

* AhnLab 융합제품개발실 (jinha.nam@ahnlab.com)
** AhnLab 융합제품개발실 (jeongmin.lee@ahnlab.com)
*** AhaLab 융합제품개발실 (sungkeun.lee@ahnlab.com)

킹과 침해가 지속적으로 증가하고 있다. 스마트폰의 보안 위협과 침해가 발생할 수 있는 영역은 다양한데 그 중에서도 사용자의 주요 정보가 저장된 단말기를 대상으로 한 악의적인 공격과 보안 위협이 증가하고 있다[2]. 단말기는 개인의 이용 측면에서 보면 주소록, 사진, 공인인증서를 포함한 민감하고 중요한 개인 정보가 저장되는 공간이며, 기업 측면에서 보면 업무의 생산성을 위해 필요한 주요 정보 자산이 저장, 관리되는 공간이다. 이러한 개인 정보와 주요 가치 있는 정보 자산이 단말기로 집중화되면서 이를 노리는 악의적인 공격 및 침해가 증가하고 있는데, 이와 같은 위협에는 해당 기기를 원격에서 제어 및 조정하거나, 응용프로그램의 실행 파일을 변조하여 동작을 변경하여 악의적은 동작을 유도하거나, 정상적인 프로그램으로 위조하여 사용자가 다운로드 하도록 유도한 이후에 원치 않는 행위를 실행하거나, 소액 결제를 유도를 통한 불법 과금 발생, 악성코드를 이용한 위치 정보를 포함한 개인 민감 정보의 도난 및 수집을 통해 개인 사생활 침해 및 추가 공격의 단서로 악용하고 있다. 또한, Wi-Fi 해킹을 통한 금융 정보를 포함한 개인 정보의 수집, 문자메시지 스니핑(sniffing)을 통한 개인 정보 훔쳐보기와 문자메시지로 전송되는 소액결제 비밀번호의 수집을 통한 원치 않는 과금 유발 등 다양한 보안 위협 및 침해가 존재한다. 이와 같은 악의적인 공격의 대부분은 악성 프로그램를 통해 발생하고 있는데, 사용자가 쉽게 응용프로그램을 사고 팔 수 있는 오픈 마켓은 자체 보안 검증 프로세스의 미비와 취약성으로 인해 게임과 도구 등으로 위장한 악의적인 공격도구가 쉽게 등록할 수 있어 이들이 유포되는 경로로 악용되고 있다.

스마트폰의 이용자들을 위협하는 다양한 보안 위협과 침해가 지속적으로 증가하고 있어 사용자를 안전하게 보호할 수 있는 보안 프로그램의 개발은 사용자 보호 측면에서 더욱 중요해 지고 있다. 운영체제 측면에서 보면 전 세계적으로 대부분의 보안 위협과 침해 사례가 안드로이드 기반으로 집중되어 있는데, 이는 해당 운영체제의 시장 점유율이 상대적으로 높아 공격할 수 있는 범위가 넓고, 개방형 플랫폼의 취약점을 이용한 공격이 가능하고, 별도의 검증 절차가 없는 마켓을 이용한 악성코드의 유포가 빠르고 쉽기 때문이다. 이러한 시장의 요구로 보안 업체는 다양한 안드로이드 기반의 보안 프로그램을 개발 및 출시 하고 있는데, 악성코드로부터 사용자를 보호하는 안티 바이러스 프로그램, 민감한 개인정보를 암화화해서 보호하는 암호 솔루션, 네트워크 연결 시에 주요 패킷을 보호하는 VPN, 프로그램의 위조와 변조를 방지하는 위변조 솔루션, 그리고 단말기의 도난 분실 등을 방지하는 MDM 솔루션등 다양한 보안 프로그램을 출시하고 있다.

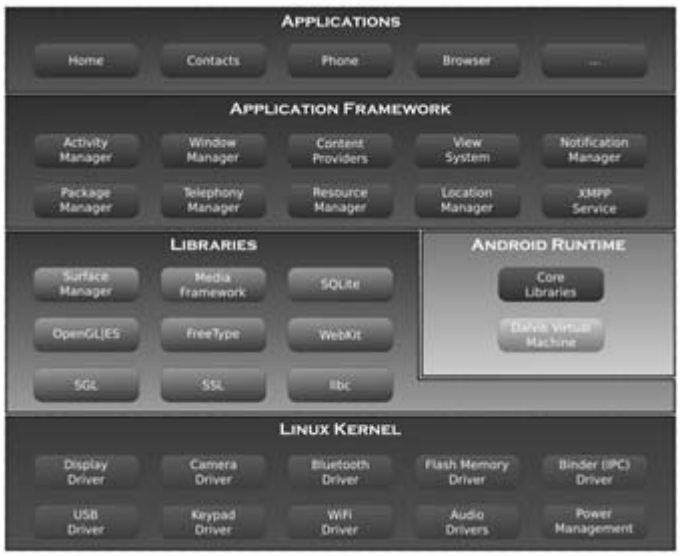
일반적으로 보안 프로그램은 다양한 보안 위협과 침해를 사전에 방어하는 것이 주요한 목적을 가지고 개발하기 때문에 프로그램의 보안 기능에 집중하여 설계하고 구현한다. 하지만, 기존의 PC 환경과 유사한 방법으로 보안 프로그램을 구현하게 되면 사용자를 안전하게 지킬 수 있는 보안 기능은 개발 할 수 있지만, 컴퓨팅 파워가 낮은 스마트폰에서 해당 프로그램이 현저하게 성능이 떨어질 수 있으며, 이는 사용자 이용 측면에서 심각한 불편을 초래할 수 있다. 스마트폰을 이용하는 사용자는 다양한 응용프로그램과 인터넷을 이용하는 것이 주요한 목적이기 때문에, 보안 프로그램을 사용할 때 다른 응용프로그램의 실행과 단말기의 이용에 심각한 제어를 받으면, 보안 프로그램의 도입을 주저할 수 있다. 그러므로, 보안 프로그램을 개발하는 개발자는 사용자를 안전하게 지키기 위한 보안성과 스마트폰의 낮은 컴퓨팅 환경을 고려한 가벼운 보안 프로그램을 개발할 필요가 있다. 이에 본 고에서는 국내에서 가장 많은 사용자를 가지고 있는 안드로이드 운영체제를 중심으로 보안 프로그램의 구현 단계에서 성능을 향상을 위해 고려할 다양한 방법을 살펴보도록 한다.

Ⅱ. 안드로이드의 구조

본 장에서는 임베디드 리눅스 커널 기반으로 Java 프로그래밍 언어를 이용해 응용프로그램을 개발하는 안드로이드의 구조[그림 1]를 살펴보도록 한다.

2.1. 안드로이드의 커널

안드로이드 커널은 스마트폰에 적합하게 수정된 리눅스 커널을 사용한다. 안드로이드용 리눅스 커널은 하드웨어의 구동에 필요한 디바이스 드라이버, 입출력에 대한 표준화 및 프로세스, 메모리 관리, 공유 라이브러리를 지원한다. 커널의 주요 특징을 살펴보면, 먼저 LMK(Low Memory Killer)를 이용해 실행 중인 프로



〔그림 1〕 안드로이드의 구조

세스를 주기적으로 모니터링 하고, 메모리가 부족한 경우에 우선 순위가 낮고, 중요하지 않다고 판단되는 프로세스를 제거하여 자원을 효율적으로 사용할 수 있다. 그리고, 안드로이드의 시스템의 통신 매카니즘을 구성하는 중요 요소로 IPC(Inter-Process Communication) 바인더를 이용하는 데, 서로 다른 프로세서에 위치한 코드나 자원을 서로 연결할 수 있으며, 응용프로그램의 대부분의 서비스가 이를 이용해서 통신한다. 또한 수정된 파워 관리(Power Management)는 단말기의 스크린 전력, 키보드와 버튼의 백라이트(Backlight) 등과 같은 처리를 단순하면서도 효율적으로 처리한다.

2.2. 라이브러리(Library)

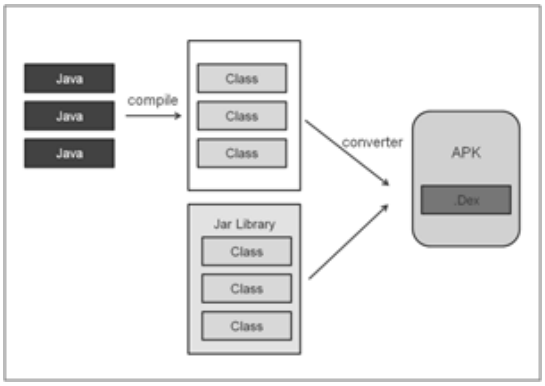
네이티브(Native)라고 불리는 라이브러리(Library) 계층은 응용프로그램을 개발하는 Java와 인터페이스를 제공하고, Open GL/ES와 같은 그래픽 라이브러리와 UI 윈도우 제어, 그리고 SQLite DB엔진과 멀티미디어 코덱(Codec), WebKit 기반의 웹 엔진을 포함한다. 라이브러리는 응용프로그램이 실행할 때, 안드로이드의 런타임(Runtime)의 코어 라이브러리를 이용해서 호출되고, 경우에 따라 리눅스 커널의 시스템 콜(System Call)을 호출한다. 이와 같은 라이브러리는 C와 C++로 개발하여 구현했는데, 이를 기반으로 성능에 영향을 미치는 동작 부분과 대용량의 메모리를 사용하는 코드를 효율적으로 처리할 수 있다.

2.3 안드로이드 런타임 (Android Runtime)

안드로이드 런타임(Android Runtime)은 달빅 버추얼 머신(Dalvik Virtual Machine)과 코어 라이브러리(Core Library)로 구성된다. 모든 안드로이드의 응용프로그램은 달빅 버추얼 머신의 인스턴스(Instance)로 동작하는데, 스마트폰과 같은 제약적인 컴퓨팅 환경에서 최소한의 환경에서도 실행될 수 있도록 별도의 파일 포맷 구조인 Dex(Dalvik Executable)의 형태로 생성되어 실행된다. Dex는 안드로이드가 제공하는 Dx컨버터 도구를 이용해서 자바의 클래스(class) 파일을 가지고 변환하여 생성된다. 이렇게 생성된 실행 가능한 코드인 Dex는 일반적인 자바 VM에서 실행 가능한 코드의 인스트럭션(Instruction)의 수보다 일반적으로 약 30%정보 적게 생성되고, 이렇게 최적화된 코드는 CPU가 처리할 작업량을 줄인다.

2.4 응용프로그램 프레임워크 (Application Framework)

응용프로그램 프레임워크(Application Framework)는 Java 기반의 응용프로그램의 인터페이스를 담당한



〔그림 2〕 안드로이드의 DEX 생성

다. 제공되는 서비스의 종류별로 살펴보면 하나의 응용프로그램이 다른 응용프로그램의 데이터의 접근이나 자신의 데이터의 공유를 할 수 있는 컨텐츠 프로바이더(Content Provider), 리소스에 대한 접근을 지원하는 리소스 매니저(Resource Manager), 응용프로그램의 생명주기를 관리하는 액티비티 매니저(Activity Manager), 모든 응용프로그램의 상태바를 관리하는 노티피케이션 매니저(Notification Manager) 등을 제공하며, 응용프로그램은 이러한 프레임워크를 이용해서 컴포넌트의 재사용을 단순화할 수 있다.

2.5. 응용프로그램 (Application)

응용프로그램(Application)은 대부분은 Java 코드로 작성하고, 필요할 경우 C/C++를 이용해 라이브러리를 개발하고 실행 중에 호출할 수 있다. 앞서 살펴본 바와 같이 일반적인 안드로이드 응용프로그램은 그림 2와 같이 Java로 생성된 클래스(Class) 파일과 Jar 라이브러리를 가지고 Dx 컨버터를 이용해서 Dex 파일을 생성한다. 이렇게 생성된 Dex 파일은 응용프로그램에 필요한 리소스(Resource) 파일과 AndroidManifest.xml과 같은 설정 파일을 가지고 압축된 파일의 형태인 apk로 묶어서 사용자에게 배포된다.

Ⅲ. 보안 프로그램의 성능 향상 방법

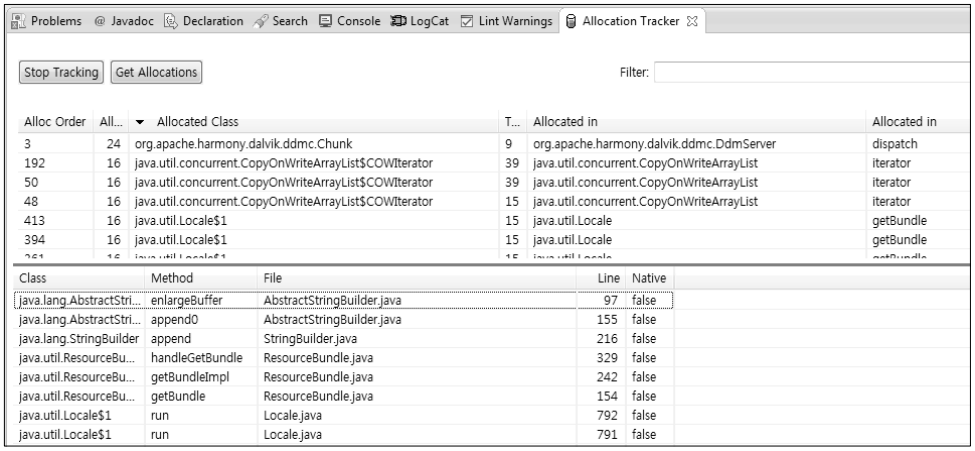
3.1 자바와 C/C++ 코드를 이용한 성능 향상

안드로이드의 응용프로그램은 달빅 가상 머신을 이용해서 코드를 실행한다. 달빅 가상 머신은 기존의 자바 코드를 컴파일하여 생성한 바이트 코드를 별도의 실행 파일인 Dex로 변환한다. 변환된 Dex는 자바 코드에 비교해서 적은 인스트럭션의 라인 수가 생성되며 이를 통해 코드의 성능을 향상시킬 수 있다. 또한 보안 프로그램이 실행하는 안드로이드 지원 버전을 2.2가 아닌 2.3 이상의 상위버전만 지원하게 되면 달빅 JIT(Just-In-Time) 컴파일러를 이용하여 네이티브 코드가 별도의 가상 머신을 이용하지 않고, 직접 CPU를 사용할 수 있어 별도의 최적화 없이도 실행 속도를 빠르게 향상 할 수 있다.

프로그램의 구현 이전 단계에서 지원하는 플랫폼의 버전을 신중히 선택할 필요가 있다. 안드로이드의 지원 API 레벨(Level)의 상위버전은 대개 하위버전 보다 성능이 향상된 API를 제공하기 때문에, 개발 할 때 상위 버전을 선택하면 성능상의 이점을 가지고 구현할 수 있다. 그러나, 혹시 하위 버전도 지원해야 프로그램은 상위 버전의 특정한 코드를 이용해서 성능이 향상된 API를 사용할 수 있는데, 이는 코드에서 Build.VERSION.SDK_INT를 이용해서 상위 버전과 하위 버전의 수행 코드를 별도로 구현하면 상위버전에서 성능이 향상된 API를 사용할 수 있다.

응용프로그램의 응답성은 사용자가 프로그램을 더 빠르게 동작하고 있는 것으로 보이게 하며, 이를 통해서 프로그램의 사용성을 개선할 수 있다. 일반적으로 안드로이드는 실행 중인 프로그램이 더 이상 동작하지 않는다고 판단하면 ANR(Application Not Responding)이 발생하게 된다. 대부분의 경우 ANR은 브로드 캐스트 리시버가 10초 이내로 결과를 반환하지 않거나, 프로그램이 5초 이상 입력에 대해 반응이 없을 경우에 발생한다. 프로그램을 구현할 때 이를 피하기 위해서는 UI 스레드(Thread)의 역할을 담당하는 메인 스레드의 작업을 최소화하고, 시간이 소요되는 네트워크 연결, 파일의 다운로드, 대용량 파일의 접근과 같은 작업이 수행할 때에는 별도의 스레드로 분리해서 처리한다.

캐쉬(Cache)는 계산할 양이 많은 경우에 효과적으로 사용할 수 있는데, 코드에서 이전에 계산한 결과값을 저장하고 이를 계산이 필요할 때 순차적으로 호출하여 사용하면 코드의 성능이 향상될 수 있다. 자바의 경우 대부분의 개발자는 해쉬맵(Hashmap)을 캐쉬로 사용하는데, 키가 integer 값일 때는 해쉬맵 보다는 SparseArrsy



〔그림 3〕 Allocation tracker

를 사용하는 것이 특정 객체의 생성을 피함으로써, 생성된 객체에 접근할 때 발생하는 접근 속도를 최소화하고, 객체의 소멸에 호출되는 가비지 컬렉션의 오버헤드를 줄일 수 있다. 또한 보안 프로그램이 특정 작업을 지속적으로 메모리에 접근할 필요가 있을 때에는 자체 캐쉬를 구현하여 자주 사용하는 데이터를 저장하면 메모리 접근시의 오버헤드를 줄일 수 있다.

보안 프로그램의 경우 C/C++로 구현된 코드의 재사용을 위해서 안드로이드의 NDK(Native Development Kit)를 이용할 수 있다. NDK는 자바 기반으로 구현한 프로그램에서 네이티브 라이브러리(Native Library)를 사용할 수 있는 개발 환경을 제공한다. 개발자가 구현한 네이티브 라이브러리는 별도의 달빅 가상 머신에서 실행하지 않고, 실제 프로세서를 직접 이용할 수 있기 때문에 실행 속도가 빠르다. 특히, 기존의 PC에서 구현된 C/C++코드를 재사용할 수 있는 장점이 있다. 또한, 특정 프로세서에 종속되어 유지보수의 비용이 증가할 수 있지만, 특정 코드의 영역에서 성능의 최적화가 가장 중요한 요소라면 해당 코드를 어셈블리 코드를 구현하여 성능을 향상 시킬 수 있다.

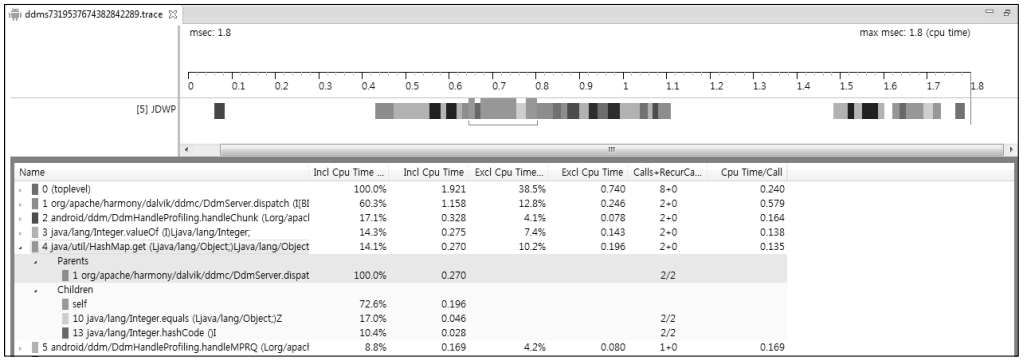
위에 살펴본 방법 이외에 코드를 설계, 구현할 때 성능을 향상 시키기 위해서는 리소스를 적게 사용하는 가벼운 데이터 구조를 사용하여 메모리를 적게 사용하거나, 특정 메소드의 호출 빈도를 최소로 함으로써 프로그램의 실행 속도를 높일 수 있으며, 실행 시간을 느리게 만드는 타입 캐스팅은 런타임에 타입이 결정되기 때문에 가급적 사용하지 않는 것이 프로그램의 성능을 향상 시킬 수 있다.

3.2 작은 데이터 타입의 사용과 객체 관리를 통한 메모리 사용

최신 안드로이드 스마트폰의 경우 RAM 사이즈가 1GB로 출시되지만, 대부분의 단말기는 512MB가 장착되어 출시하고 있다. 이러한 메모리의 제약 사항 때문에 보안 프로그램을 개발할 때 메모리를 할당 받아 사용할 때 주의가 필요하다.

대부분의 프로그램의 성능은 코드를 구현할 때 선택하는 데이터의 타입과 데이터와 명령어가 저장되는 공간을 얼마나 차지하는가가 중요하다. 보안 프로그램의 동작할 때 특정 영역의 데이터를 정렬하거나, 혹은 값을 읽어서 비교 연산을 수행하는 알고리즘을 구현할 때, 상대적으로 작은 데이터 타입을 선택하게 되면, 더 큰 데이터 타입보다 적은 명령어를 가진 코드가 실행하기 때문에 상대적으로 속도가 향상된다. 특히, 대용량의 데이터를 다루어야 할 때도 작은 타입의 데이터를 사용해 비교하고, 다른 타입의 데이터 타입이 아닌 하나의 데이터 타입을 가지고 연산을 수행하면 메모리를 적게 사용하면서 속도를 증가시킬 수 있다.

자바의 가비지 컬렉션은 객체와 배열을 생성할 때 힙(Heap) 영역에 저장한다. 객체를 new 키워드를 이용해서 생성할 때마다 힙의 메모리에 새로운 객체가 할당되므로, 객체가 더 이상 참조되지 않을 때 할당된 메모리를 해제하고 복귀시키는 것은 중요하다. 이러한 객체가 해제되지 않고 계속 힙 메모리에 상주하면 메모리 부족과 누수가 발생할 수 있다. 프로그램을 구현할 때 적은 객체의 할당을 유지하게 되면, 힙 메모리의 사용 및 누



〔그림 4〕 Trace View

수를 예방할 수 있으며, 불필요한 가비지 컬렉션의 호출이 줄어 들어 전반적인 성능 향상을 기대할 수 있다. 이러한 메모리 누수와 사용은 DDMS(Dalvik Debug Monitor Service)의 Allocation Tracker[그림 3]로 분석할 수 있다.

안드로이드는 액티비티를 화면에 표시할 때, 메인 스레드에서 응용프로그램에 해당하는 레이아웃과 비트맵 정보를 포함하는 리소스 데이터를 먼저 읽고 파싱하여 액티비티의 최상위 뷰를 표시한다. 사용자가에 보여지는 최초의 화면을 빠르게 표시하는 방법은 레이아웃의 구조를 간결하게 하거나, 비트맵과 같은 메모리 사용이 많은 객체의 수를 생성하지 않거나, 불필요할 레이아웃을 제거하면 된다. 특히, 비트맵은 액티비티가 실행되는 동안 많은 메모리를 차지하게 되므로, 프로그램의 화면을 설계할 때 사용여부를 신중하게 선택해야 한다.

3.3 배터리 사용 시간 늘리는 방법

보안 프로그램은 주기적인 보안 검사나 특정 명령을 수신하기 위해서 스마트폰이 슬립모드(Sleep)모드로 변경되어도 백그라운드에서 계속 작업을 수행할 필요가 있는데, 제약적인 스마트폰의 배터리 양을 고려해서 적은 전력을 이용해 작업을 수행해야 한다. 안드로이드 스마트폰은 일반적으로 1,250 ~ 1,500 mAh 사이의 배터리의 용량을 가지고 있다. 이러한 배터리의 용량은 프로그램의 코드 실행과 3G/4G, Wi-Fi를 이용한 데이터 전송, 모바일 네트워크나 GPS를 이용한 위치 전송, 가속센서 등을 사용할 때 상당량의 전력을 소모하게 된다.

MDM (Mobile Device Management)의 에이전트는 서버로부터 원격 명령을 수신 받거나 사용자의 최근 위치를 파악하기 위해서 빈번하게 네트워크를 사용하거나 GPS를 자주 이용한다. 네트워크를 이용해 데이터를 전송할 때 포그라운드(Foreground) 작업 보다는 백그라운드(Background)로 작업을 수행하게 되면 적은 전력을 가지고 작업을 수행할 수 있다. 그리고, 데이터의 전송 속도를 빠르게 하기 위해서 특정 파일을 압축해서 전송하거나, 필요한 파일과 통신 프로토콜을 크기를 최소화해서 전송하면 적은 양의 배터리를 가지고 동일한 작업을 수행할 수 있다. 사용자의 위치를 파악하기 위해서 GPS나 모바일 네트워크를 위해서 위치를 파악할 때에도 로케이션 리스너(Location Listener)를 자주 호출하는 것보다는 필요할 경우에만 호출하고 해제하는 것이 바람직하며, 위치 업데이트의 주기를 가능한 길게 유지하면 배터리 소모는 현저히 줄어든다.

위에서 언급한 다양한 성능 향상 방법을 이용한 프로그램의 성능을 측정하는 방법은 코드의 수행 속도를 안드로이드가 제공하는 API를 특정 동작 구간에 작성하여 실행 속도를 함수를 기반으로 측정할 수 있다. 또한, 트레이싱과 관련된 API를 이용하거나, DDMS perspective를 이용해서 성능 측정으로 위한 트레이스 파일을 생성하고, 생성된 파일을 DDMS가 제공하는 트레이스 뷰(TraceView) 도구[그림 4]를 이용하면 메소드의 실행 속도를 분석할 수 있으며, 프로그램의 병목 지점을 쉽고 빠르게 찾을 수 있다.

Ⅳ. 결 론

본 고에서는 안드로이드 기반의 보안 프로그램의 구현단계에서 성능을 향상시킬 수 있는 여러 가지 방안을 살펴보았다. 보안 프로그램을 개발할 때는 이용자를 안

전하게 보호하기 위한 보안 기능에 중점을 두고 개발을 해야 한다. 하지만, 보안 프로그램을 사용하는 이용자에게는 보안 기능 뿐만 아니라 사용성과 편의성도 중요하다. 그러므로, 프로그램의 설계와 구현할 때 강력한 보안 기능과 함께 해당 성능을 최적화할 수 있는 다양한 방안도 함께 고려하고 반영하여 보안 프로그램의 사용성과 편의성이 증진 될 것으로 기대된다.

참고문헌

[1] Web page : http://isis.kisa.or.kr/board/?pageId=060200&bbsId=3&itemId=799“ 방송 통신 위원회 2012년 하반기 ‘제 6차 스마트폰이용실태조사“.

[2] Web page : http://, http://www.ddaily.co.kr/news/news_view.php?uid=99572 “디지털 데일리“.

[3] 김현욱, “스마트폰 기술 발전 및 보안 기술 동향”, Telecommunications Review 제 21권 2호

[4] 염흥열, 장기헌, “*국내외 스마트폰 보안 표준화 동향 및 추진 전략*”, T*TA Journal No.132*

[5] Herve Guihot, *Pro Android Apps Performance Optimization*, Apress 2012, 1,17

[6] Web page : http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/Porting-Android-to-a-new-device/

〈著者紹介〉

남 진 하 (Nam JinHa)
비회원

2005년 2월 : 안동대학교 정보통신공학과 졸업(학사)
2012년 ~ 현재 : 연세대 컴퓨터공학과 석사 과정
2009년 ~ 현재 : AhnLab 융합제품개발실 선임 연구원
<관심분야> 시스템 보안, 안드로이드, 타이젠, 정보보호

이 정 민 (Lee Jeong-min)
비회원

2010년 2월 : 광운대학교 컴퓨터소프트웨어학과(학사)
2012년 2월 : 광운대학교 컴퓨터과학과(석사)
2013년 1월~현재 : AhnLab 융합제품개발실 연구원
<관심분야> 시스템보안

이 성 근 (Lee Sung-keun)
비회원

2007년 2월 : 한양사이버대학교 컴퓨터공학과(학사)
2009년 8월 : 성균관대학교 이동통신공학과(석사)
2001년 7월~현재 : AhnLab 융합제품개발실 책임연구원
<관심분야> 정보보호, 시스템보안 및 서비스 개발