가상화 기반의 안드로이드 루트 권한 획득 탐지

조영필*, 이하윤*, 권동현*, 최원하*, 백윤흥*
*서울대학교 전자계산학과
e-mail : ypcho@sor.snu.ac.kr

Hypervisor based Root Exploitation Monitoring in Android

Yeong-pil Cho*, Ha-yoon Yi*, Dong-hyun Kwon*, Won-ha Choi*, Yun-heung Paek
*Dept. of Electrical and Computer Engineering, Seoul National University

요 약

국내에서 가장 폭넓게 사용되는 모바일 운영체제인 안드로이드는 수 많은 악성코드에 대한 위협 속에 있다. 그 중에서 가장 위협적인 공격은 루트 권한을 획득하는 악성코드이다. 따라서 본 연구는 가상화 환경을 통해 안드로이드 시스템에서 실존하는 루트 권한 획득을 탐지하는 시스템을 소개 하고 있다. 이를 위해 CPU 제조사에서 제공하는 가상화 기반 기술을 활용하였으며 결과적으로 시스템 상에서 루트 권한으로 동작하는 프로세스를 감지할 수 있었다.

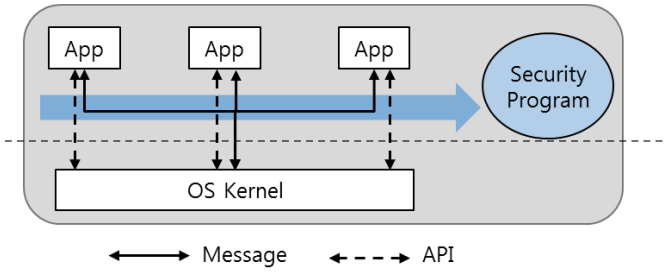
1. 서론

2013 년 영국의 iCrossing 사는 주요 국가별 모바일 운영체제의 점유율을 보고하였다 [1]. 이에 따르면, 국내 모바일 운영체제의 점유 현황은 안드로이드가 90.1%, 애플 iOS 가 9.6%, 그리고 기타 OS 가 0.3%를 차지하고 있다. 문제는 이처럼 국내 모바일 시장이 안드로이드로 편중된 상황에서 안드로이드가 모바일 악성코드의 최대 공격대상이라는 점이다. Fortinet 사의 FortiGuard Lab 은 2013 년 전체 모바일 악성코드의 목표 운영체제를 분석하였는데, 그 결과 전체의 96.5%의 악성코드가 안드로이드를 공격대상으로 삼고 있음을 밝혔다 [2].

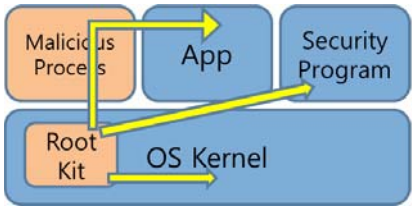
이러한 악성코드의 위협에 대응하기 위해 다양한 보안 업체에서 안티바이러스 프로그램을 개발하고 있으며, 실제 이들의 노력은 상당한 성과를 이루어, AV Comparatives 사의 조사에 따르면 많은 안티 바이러스 프로그램의 악성코드 탐지율이 90%를 상회하고 있으며 몇몇 프로그램은 99.9%에 해당하는 탐지율을 보이고 있다 [3].

일반적으로 안티바이러스 프로그램은 컴퓨터 시스템의 전체 파일시스템을 조사하여 악성코드를 탐지하는 기능을 탑재하고 있다. 그러나 이를 실행할 경우 시스템을 사용하지 못할 정도로 높은 성능 부하를 일으키기 때문에, 대부분의 사용자들은 안티바이러스 프로그램의 실시간 검사 기능을 통해 시스템의 보안성을 높이고자 한다. 그림 1 은 실시간 감시가 어떻게 이루어지는지 간략히 표현한 것이다.

그러나 이러한 동작 모델은 구조적으로 안티바이러스 프로그램이 악성코드에 대응하는데 있어 한계에 부딪히는 원인이 되었다. 안티바이러스 프로그램은 기본적으로 하나의 응용프로그램에 속한다. 따라서 시스템에 접근할 수 있는 권한은 악성코드의 목표가 되는 여타 프로그램과 동일성 상에 있다. 이러한 상황은 악성코드에 감염된 혹은 악성코드 자체가 응용프로그램에 머물고 있을 때는 문제가 되지 않는다. 그러나 루트킷 (RootKit)과 같이 운영체제의 커널 영역을 대상으로 하는 악성코드에 시스템이 감염되었을 경우 안티바이러스 프로그램을 더 이상 신뢰할 수 없게 된다. 그림 2 는 이러한 위협을 나타낸 것이다. 루트킷은 운영체제의 커널 영역에 위치하기 때문에 시스템의 어떠한 부분도 공격대상으로 삼을 수 있다. 먼저, 루트킷은 다른 악성 응용프로그램을 시스템에 주입하는 것을 목표로 할 수 있다. 만약 루트킷이 없는 상황이라면, 허가되지 않은 악성 프로그램이 시스



(그림 2) 보안 프로그램의 개념도



(그림 1) RootKit 의 위협성

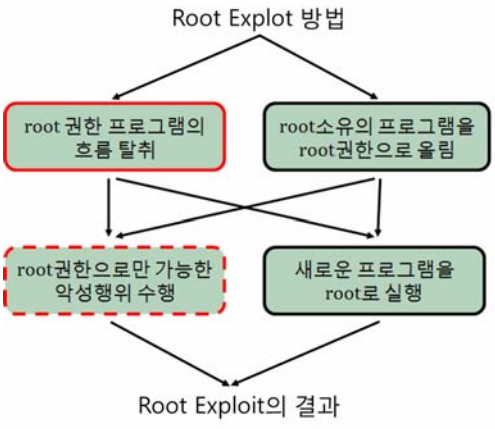
템에 설치될 때, 안티바이러스 프로그램에 의해서 차단이 될 수 있지만, 루트킷이 있을 경우 루트킷이 안티바이러스 프로그램이 악성코드 검사를 위해 사용하는 인터페이스를 우회하여 악성 프로그램을 설치하거나 심지어 안티바이러스 프로그램의 코드 혹은 데이터가 위치한 메모리 영역을 조작하여 악성코드 탐지 기능을 무력화 할 수도 있다.

이처럼 강력한 악성코드인 루트킷이 안드로이드에서 달성하고자 하는 목표는 루트 권한 획득 (Root Exploit) 이다. 안드로이드는 리눅스 커널을 기반으로 하지만, 일반적인 리눅스 보다는 향상된 보안성을 가지고 있다. 그 이유는 Dalvik VM 을 기반으로 각각의 어플리케이션들이 서로 다른 User ID 로 샌드박스화 (SandBoxing) 되어 있기 때문이다. 그러나 일단 루트킷 등에 의해서 악성 코드가 루트 권한을 획득하게 되면 기존의 안드로이드에 적용된 보안 기능들은 모두 무효화 된다. 안드로이드 상에서 이러한 위협은 실질적 이여서 Blue Coat Systems 의 보고서에 따르면 안드로이드에 대한 전체 공격에서 루트 권한 획득과 직간접적으로 관련된 공격이 58%에 달한다고 한다 [4]. 대표적인 실제 사례를 살펴보면 DroidKungFu [5], GingerMaster [6], RootSmart [7] 가 있다.

본 연구에서는 이러한 루트 권환 획득 공격에 대한 안드로이드의 취약성을 보완하는데 목표를 두고 있다. 우리는 이를 위해 하이퍼바이저를 통한 가상화 환경을 활용하였다. 이러한 모델에서 안드로이드는 하이퍼바이저 상에서 수행되는 하나의 Guest 가상머신으로 동작하기 때문에, 하이퍼바이저는 안드로이드의 동작을 실시간으로 감시할 수 있게 된다. 이러한 원리를 통해 안드로이드의 루트 권환 획득 여부를 탐지하고자 한다.

2. 루트 권한 획득 탐지 방법

루트 권한 획득을 탐지하기 위해선 악성코드가 어떠한 방식으로 루트 권환을 획득하고 또 루트 권한 획득 후 어떠한 행위를 하는지 조사할 필요성이 있다. 그림 3 은 이를 간략하게 도식화한 것이다. 먼저 악성코드가 루트 권한을 획득하는 하나의 방법은 기존에



(그림 3) 루트 권한 획득 및 목표

루트 권한으로 동작하는 프로그램의 실행 흐름을 탈취하는 것이다. 애당초 해당 프로그램이 악성 행위를 위해 만들어 진 것이 아니라면, 이러한 경우는 프로그램에 잠재된 버퍼 오버플로우와 같은 버그를 이용하는 경우가 많다. 또 다른 방식은 아직 실행되지 않은 루트 소유의 프로그램을 루트 권한으로 동작하도록 하는 것으로, 현재 안드로이드의 루트 권한 획득 프로그램에서 가장 흔히 사용되는 방법 중 하나이다.

본 연구에서는 후자의 방법을 활용한 루트 권한 획득을 감시하고자 한다. 전자의 경우를 방지하기 위해선 프로그램 코드의 무결성 검사와 같은 기술을 활용하는 것이 보다 용이하기 때문이다.

루트 소유의 프로그램을 루트 권한으로 수행하는데 있어 가장 흔히 사용되는 방식은 setuid 명령어를 사용하는 것이다. 이 명령어는 파일을 실행할 시, 현재 파일을 실행시키는 계정의 권한이 아닌, 파일 자체가 가지고 있는 권한으로 실행하는 것이다. 따라서 루트 계정으로 프로그램을 실행할 수 있는 su 명령어 파일을 루트 소유로 작성한 후, setuid 명령어를 통해 su 명령어를 사용하면, 현재 접속한 계정이 일반 유저 계정이라고 하더라도 su 명령어를 본 소유 계정인 루트 계정으로 실행할 수 있게 된다.

이러한 과정을 통해 루트 권한을 획득하면, 악성코드는 루트 권한을 통해 수행할 수 있는 악성행위를 하게 되는데 예를 들면 다른 어플리케이션의 데이터베이스 파일을 살펴보는 것이 있다. 그리고 루트 권한을 획득한 악성코드는 더 많은 악성코드를 실행하기 위해 다른 악성 프로그램을 루트 권한으로 실행하는 경우가 많이 있다.

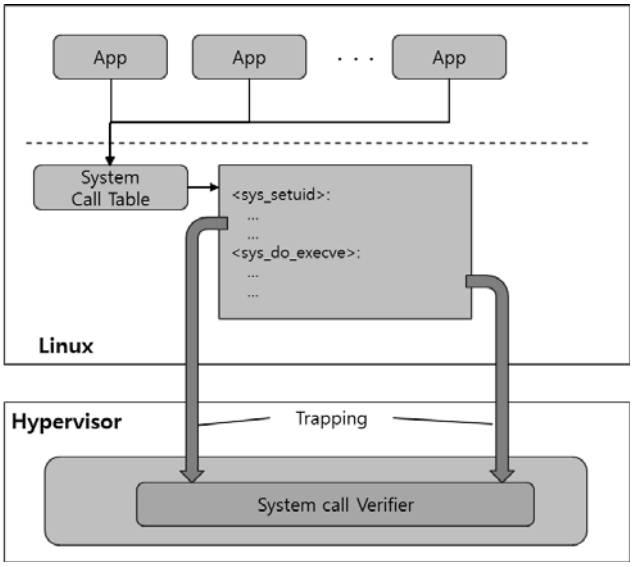
본 연구에서는 이러한 일련의 루트 권한 획득 및 이를 통한 악성 행위를 감지하기 위하여 하이퍼바이저를 사용하여 안드로이드의 동작을 감시하고자 한다. 보다 명확히 말하면, 안드로이드의 기반에 위치한 리눅스 커널의 동작을 감시할 것이다.

3. 하이퍼바이저를 통한 루트 권한 획득 탐지

하이퍼바이저는 루트 권한 획득을 탐지하기 위해서 리눅스 커널의 시스템 콜 (System Call)의 동작을 감시한다. 위에서 언급한 setuid 명령어와 루트 권한을 획득한 악성코드가 악성 프로그램을 실행하기 위하여 사용하는 exec 와 같은 명령어는 운영체제가 제공하는 API 를 거쳐서 최종적으로 커널에 위치한 시스템 콜을 호출하게 되어있다. 예를 들어 setuid 의 경우 sys_setuid 라는 시스템 콜을 호출하게 되며, exec 의 경우 do_execve 라는 시스템 콜을 호출하게 된다. 따라서 하이퍼바이저는 자신이 제공한 가상환경 상에서 동작하는 안드로이드, 보다 정확히는 리눅스 커널을 감시하다가 sys_setuid 혹은 do_execve 시스템 콜이 호출되면 해당 가상머신을 정지시킨 후, 해당 행위가 악성코드에 의한 것이 아닌지 검증을 하게 된다.

그림 4 는 이러한 시나리오를 도식화 한 것이다. 그림 상에서 App 으로 표현된 안드로이드의 어플리케이션들은 setuid 혹은 exec 를 실행하기 위하여 리눅스 커널에 위한 시스템 콜을 호출하게 된다. 이때 각각

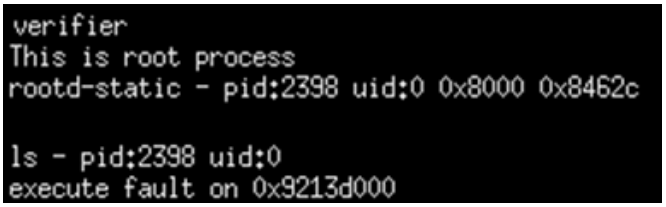
의 시스템 콜은 하이퍼바이저로 트랩 (Trap)이 걸려 시스템의 실행 흐름이 가상머신에서 하이퍼바이저로 전환된다. 여기서 시스템 콜 실행 시 하이퍼바이저로 전환되는 트랩을 걸기 위해선 다양한 매커니즘을 활용할 수 있다. 과거의 하이퍼바이저는 시스템에 가상 환경을 제공하기 위해 크게 두 갈래의 기술을 사용하였다. 첫 번째 기법은 가상 머신에서 수행할 운영체제의 코드를 가상 환경에서 동작하도록 하는 바이너리 변환 (Binary Translation) 기술이고, 두 번째 기법은 하이퍼바이저가 가상 머신의 모든 동작에 관여하여 가상 머신이 해야 할 작업들을 재현해주는 트랩 및 재현 (Trap and Emulation) 기술이다. 각각의 기술은 서로 장 단점을 가지고 있지만, 여기서 단점만을 언급하면, 바이너리 변환 기술은 장치 드라이버를 포함하여 모든 운영체제의 코드를 변환해야 하기 때문에 확장성이 저해된다는 문제가 있으며, 두 번째로 언급된 트랩 및 재현 기술은 가상 머신의 성능이 저해된다는 문제가 있다. 따라서 이러한 문제점을 해소하기 위해 인텔이나 AMD, 그리고 ARM 과 같은 CPU 설계 기업들은 하드웨어적으로 가상환경을 제공하기 위한 노력을 기울이게 되었다. 이들은 기본적으로 트랩 및 재현 기술을 기반으로 가상화 환경을 제공하는데, 하이퍼바이저 상에서 동작하는 각각의 가상 머신들이 시스템을 구성하는데 필수적으로 필요한 CPU, 메모리 등을 직접적으로 이용할 수 있도록 하고, 그 과정에서 민감한 명령어 혹은 레지스터를 접근하고자 할 때 CPU 가 직접 하이퍼바이저르 실행 권한을 넘기도록 하였다. 본 연구에서는 여기서 CPU 가 제공하는 트랩 기능을 활용하였는데, 다양한 트랩 기능들이 사용될 수 있지만 가장 대표적인 것으로는 MMU (Memory Management Unit)에서 제공하는 메모리 페이지에 대한 접근 권한에 의한 트랩이다. 예를 들면 sys_setuid 와 do_execve 의 실행 코드가 위치한 메모리 주소는 해당 커널 코드가 컴파일 된 이후에 항상 고정이 된다. 고정된 메모리 주소는 커널 컴파일 이후 System.map 파일에서 찾을 수 있다. 따라서 이렇게 고정된 메모리 페이지에 하이퍼바이저에서 실행 금지 설정을 하게 되면, 가상 머신의 리눅스 커널에서 해당 시스템 콜을 실행할 때 CPU 에 의해 강제적으로 실행 흐름이 저환된다. 이후 현재 해당 시스템 콜을 호출한 프로세스를 고려하여 허용 여부를 결정하면 된다.



(그림 5) 하이퍼바이저를 통한 감시

4. 실험 결과

그림 5 는 위에서 언급한 방법을 기반으로 하이퍼바이저에서 안드로이드 가상 머신의 시스템 콜을 감시한 결과이다. 안드로이드 시스템에서 rootd-static 이라



(그림 4) 하이퍼바이저를 통한 시스템 콜 감시

는 루트 권한 획득을 위한 가상 데몬을 실행하였으며, 해당 데몬이 리눅스의 디렉토리 내 파일의 리스트를 출력하는 ls 명령어를 루트 권한으로 실행을 한 결과이다. 그 결과 do_execve 명령어가 위치한 0x9213d000 메모리 주소에서 트랩이 걸려 하이퍼바이저로 실행 흐름이 전환 되었다. 자세한 실행 결과를 보면 ls 명령어는 2398 번의 pid 를 가지고 있는 rootd-static 데몬에 의해 실행이 되었기 때문에 2398 번의 pid 로 실행이 되었으며 uid 가 0 인 것으로 보아 루트 권한으로 실행되었음을 알 수 있었다.

Acknowledgement

본 연구는 교육과학기술부/한국과학재단 우수연구센터 육성사업(과제번호 2012-0000470), 중소기업청에서 지원하는 2012 년도 산학연공동기술개발사업(No. C0019562), 미래창조과학부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신) [No. 10047212, 1kB 이하 암호문 간의 연산을 지원하는 동형 암호 원천 기술 개발 및 응용 연구] 및 IDEC 의 지원을 받아 수행하였습니다.

참고문헌

[1] iCrossing, 2013 Mobile Market Share
[2] Fortinet, 2014 Threat Landspace Repooort
[3] AV Comparatives, 2013 Mobile Security Review
[4] Blue Coat Systems, 2013 Mobile Malware Report
[5] Jiang, X. "Security alert: New droidkungfu variant, 2011."
[6] Jiang, X. "Security alert: Gingermaster, 2011."
[7] Jiang, Xuxioan. "New RootSmart Android Malware Utilizes the GingerBreak Root Exploit." (2012).