단계적 분석 기법을 이용한 클라우드 기반 모바일 악성코드 탐지

이진아*, 민재원*, 정성민*, 정태명**
*성균관대학교 전자전기컴퓨터공학과
**성균관대학교 정보통신대학
{jnlee90,jwmin,smjung}@imtl.skku.ac.kr, tmchung@ece.skku.ac.kr

Cloud based Android Mobile Malware Detection Using Stage by Stage Analysis

Jina Lee*, Jae-Won Min*, Sung-Min Jung*, Tai-Myoung Chung**
*Dept of Electrical and Computer Engineering, Sungkyunkwan University
**School of Information Communication Engineering, Sungkyunkwan University

요 약

스마트폰의 사용이 생활에 필수적인 요소가 되었다. 스마트폰 특징의 가장 핵심적인 부분이 다양한 콘텐츠를 사용자의 취향에 맞게 선택 할 수 있다는 점이기에 스마트폰의 콘텐츠 시장 또한 빠르게 커지고 있다. 오픈 마켓인 안드로이드의 특성 상 누구나 어플리케이션을 만들어 원하는 곳에 배포할 수 있고 어플리케이션을 다운받을 수 있는 소스도 한정되어 있지 않기 때문에 스마트폰 보안을 위협하는 악의적인 어플리케이션에 노출되기 쉽다. 개인적인 정보가 저장되어 있는 핸드폰의 특징 상 악성코드에 노출 될 경우 전화번호부 유출로 인한 인한 스팸이나 피싱에서 크게는 금융정보 유출까지, 입을 수 있는 피해가 크다. 이를 방지하기 위해 클라우드 컴퓨팅을 이용해 단계적으로 악의적인 어플리케이션을 걸러 내고 클라우드 서버에 어플리케이션 실행 환경을 제공함으로써 사용자의 기기를 안전하게 보호 할 수 있는 시스템을 제안한다.

1. 서론

최근 몇 년 간 스마트폰의 사용이 폭발적으로 증가하고 있다. 스마트폰 시장이 성장하면서 그에 따른 콘텐츠 시장 또한 급격히 성장하고 있다. 한국 콘텐츠 진흥원에 따르면 2010 년 기준 콘텐츠 관련 수익 증감률은 최대 24% 성장했다[1]. 앱스토어를 통해서만 어플리케이션을 다운 받을 수 있는 아이폰의 iOS 와는 달리 안드로이드 운영체제를 사용하는 스마트폰의 경우에는 공식 마켓인 구글 플레이스토어 이외에 사용자가 원하는 웹 상의 어떤 소스에서도 어플리케이션을 다운받을 수 있다. 따라서 검증되지 않은 악의적인 프로그램을 받을 수 있는 가능성이 크다. 악성 프로그램에 대한 위협은 PC 를 사용할 때부터 항상 있어 왔지만 전화번호, 문자, 위치정보 등 개인적인 정보를 많이 다루는 스마트폰의 특성상 정보의 유출로 입는 피해의 영향력이 PC 에서 보다 더 크다고 말할 수 있다. 백신의 사용이 일반화 된 PC 에서와는 달리 스마트폰 에서 백신 어플리케이션을 이용하는 사용자는 많지 않다.

스마트폰은 개인적인 정보를 많이 담고 있고, 은행 어플리케이션을 통한 모바일뱅킹 거래도 활발하게 이루어지고 있는 데에 더해 최근에는 전자 지갑 기능으로 스마트폰이 신용카드와 같은 결제 수단 자체를 대체하려 하는 경향을 보이고 있기 때문에 모바일 단말기에 대한 보안 위협 탐지에 대한 중요성은 더욱 부각되고 있다.

따라서 본 논문에서는 스마트폰에서의 클라우드 기반 악성 어플리케이션 탐지 기법을 제안한다. 먼저 2 장에서는 프로그램을 분석하는 2 가지 방법에 대해 설명하고 3 장에서는 본 논문에서 악성코드 탐지를 위해 어떤 방법을 사용하는 지와 클라우드 서버의 활용 방법에 대해 구체적으로 설명한다. 4 장에서 제안한 시스템에 대한 한계점을 분석하고 결론을 맺는다.

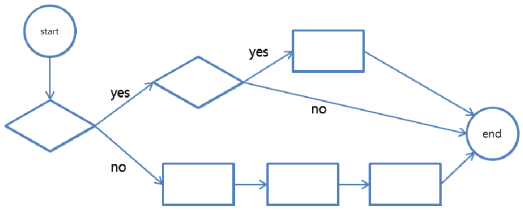
2. 관련연구

악성코드를 탐지해 내는 수많은 방법이 있지만 이것들은 크게 정적 분석(Static analysis)과 동적 분석(Dynamic analysis)으로 나누어질 수 있다[2].

정적 분석은 프로그램의 코드 자체를 분석하는 방법인 데 반해, 동적 분석은 프로그램을 실행시키면서 어플리케이션의 동적인 행동들을 감시하는 방법이다.

2.1 정적 분석

정적 분석은 어플리케이션을 실제로 실행시킬 필요가 없기 때문에 동적 분석에 비해 적은 비용으로 작



(그림 1) 데이터 흐름 그래프

업을 수행할 수 있다.

정적 분석을 실행하는 방법은 또 다시 여러 가지로 나뉘어진다. 그 중 하나는 구조적 분석(Structural analysis) 방법이다. 구조적 분석은 패턴 매칭을 이용한다. 이 방법은 프로그램의 실행이나 데이터 흐름과는 관계 없이 코드의 일부분만을 분석하는 단순한 방법이라고 할 수 있다.

제어 흐름 분석(Control flow analysis)는 제어 흐름 그래프(Control flow graph)를 통해 이루어진다. 그림 1과 같이 제어 흐름 그래프는 어떤 프로그램이 시작되며 행해질 수 있는 일들의 시퀀스를 그래프로 나타낸 것이다. 이 시퀀스가 프로그램을 정상적이지 않은 상태로 가게 만드는지 아닌지를 봐서 악성 코드를 감지한다[3].

정적 분석의 또 다른 방법으로는 데이터 흐름 그래프(Data flow graph)를 이용하는 데이터 흐름 분석(Data flow analysis)가 있다. 어떠한 함수가 실행 되었을 때 그 함수가 개인적인 데이터에 접근 해서 그 데이터를 제 3 자에게 전송하는지 데이터 흐름 그래프를 통해 알아 낼 수 있고 이것을 통해 개인정보 유출을 찾아 낼 수 있다.

2.2 동적 분석

동적 분석을 실행할 때에는 모바일 어플리케이션을 가상 환경에서 실행시켜 동적인 행동을 감시한다. 동적 분석은 주로 감염 추적(Taint tracking)과 시스템 콜 추적 방식을 사용한다.

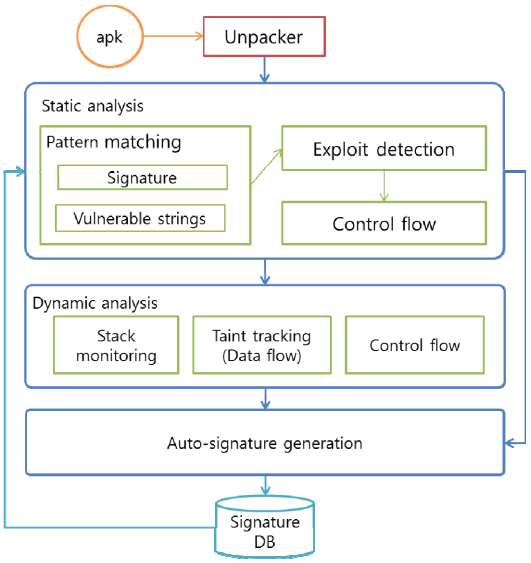
감염 추적 또는 동적 감염 분석(Dynamic taint analysis) 이라고도 하는 이 방법은 민감한 개인 정보들을 모니터링 한다. 유출되어선 안 되는 정보에 태그를 달아 이 표시된 데이터를 추적하면서 중요한 정보를 유출하는지를 관찰한다. 이 방법은 전체 시스템의 흐름을 관찰하기 때문에 높은 퍼포먼스를 요구한다[4].

시스템 콜 추적 방식은 어플리케이션에서 사용하는 시스템 콜들을 감시하는 방법이다. 이 방법은 이미 알려진 악성코드에서 발견된 행동들을 기반으로 어플리케이션을 분석한다. 악의적인 행동에는 일반적으로 복제와 전파(Replication and propagation), 개인 정보 침해(Privacy invasion), 악의적인 코드 주입(Malicious code injection), 지속적인 반복 행동(Persistent behavior) 이렇게 네 가지가 있다[5]. 복제와 전파는 바이러스에서 나타나는 일반적인 특성이다. 지속적인 반복 행동은 사용자의 입력이나 확인 없이 자동으로 프로그램을 시작하게 해 시스템의 재부팅 후에도 악성 프로그램이 살아 있는 채로 남도록 한다.

동적 분석을 이용하면 정적 분석만으로는 찾아내지 못했던 악성코드들을 찾아 낼 수는 있겠지만 동적 분석은 구현에 좀 더 많은 노력이 필요하고 전력이나 비용 또한 많이 소비한다.

3. 클라우드 기반 단계적 악성코드 탐지

스마트폰의 제한적인 자원 문제를 해결하기 위해 클라우드 컴퓨팅을 이용하는 방법은 계속해서 연구되어 온 주제이다 [6][7][8]. 본 논문에서는 어플리케이션의 분석 부분을 클라우드 서버에서 실행시킬 뿐만 아니라 씬 클라이언트 컴퓨팅 방식을 이용한다. 사용자는 의심스럽다고 생각되는 어플리케이션을 선택해 어플리케이션의 실행을 전적으로 클라우드 서버에 맡긴다[9]. 사용자의 기기와 클라우드 서버는 네트워크 연결을 통해 통신한다. 사용자가 어플리케이션에 필요한 입력을 보내면 클라우드 서버는 그에 따른 실행 결과를 출력해 보여준다. 어플리케이션의 실행을 실제 사용자의 기기가 아닌 클라우드 시스템의 가상화 환경으로 옮김으로써 클라우드의 풍부한 자원을 어플리케이션 실행에 이용할 수 있을 뿐만 아니라 악성 어플리케이션의 공격에 노출 된 경우에도 프로그램이 올려져 있던 가상 이미지만 오염되기 때문에 클라우드 시스템과 사용자의 기기는 무결성을 유지할 수 있다.



(그림 2) 악성 어플리케이션 탐지

3.1 악성 어플리케이션 탐지

본 논문에서는 악성코드 탐지를 위해 어플리케이션을 단계적으로 분석하는 방법을 이용하였다. 단계를 거쳐갈수록 연산은 복잡해지고 시간을 많이 소비하게 된다. 앞 단계에서 비교적 간단한 연산을 통해 악성 코드를 찾아내면 동적 분석과 같은 복잡한 계산은 추가로 실행하지 않음으로써 효율성이 높아지는 효과를 가져올 수 있다.

원하는 어플리케이션의 APK 파일을 다운받으면 먼저 언패커가 암호화 된 코드를 언패킹하고, 여기서

얻어진 코드를 기반으로 이후의 분석을 진행한다. 악성 어플리케이션 탐지 모듈은 크게 정적 분석과 동적 분석 단계로 나뉘어진다. 정적 분석은 또다시 두 단계로 나누어지는데, 먼저 패턴 매칭을 실행하고 여기서 아무것도 찾아내지 못하면 익스플로잇 탐지 단계로 넘어간다. 정적 분석에서 악성 어플리케이션으로 판명이 난 경우에는 APK 파일을 삭제하고, 그렇지 않으면 어플리케이션을 실행시켜 동적 분석을 진행한다. 악성 어플리케이션을 찾아내면 시그니처 자동 생성 모듈을 통해 새로운 시그니처를 생성하고 데이터베이스에 저장해 다음 번 분석에 활용할 수 있도록 한다.

첫 번째 단계에서는 어플리케이션의 실행 없이 언패커를 통해 얻은 코드를 정적 분석하는데 정적 분석 모듈에서는 가장 먼저 패턴 매칭을 이용해 프로그램을 검사한다. 여기서는 시그니처를 이용해 이전에 검사했던 악성 프로그램과 유사한 점이 있는지 비교하고, 취약 문자열을 탐지하여 쉘 명령어 사용과 시스템 폴더 접근과 같은 허가되지 않은 유저의 행동을 방지한다.

패턴 매칭에서 악의적인 행동을 탐지해 내지 못하면 다음 단계인 익스플로잇 탐지로 넘어가 어플리케이션 내에 머신코드가 존재하는지 조사한다. 만약 실행 가능한 머신코드가 발견되면 이 코드는 공격 코드일 가능성이 높다. 현재 이를 위한 다양한 정적 분석 방법이 제안되었다[10][11]. 정적 분석의 마지막 단계에서는 제어 흐름 그래프 (Control flow graph)를 만들어낸다. 시작 상태에서 시작해 모든 분기점을 포함해 종료 상태까지의 가능한 경로를 표시 해 놓은 이 그래프를 통해서 중요한 정보가 외부 네트워크를 통해 빠져나가는 지 살펴볼 수 있을 것이다[12].

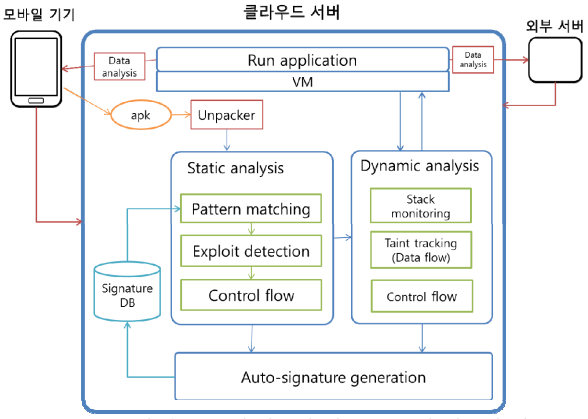
정적 분석 단계에서 어플리케이션이 안전하다고 분류 되면 동적 분석 단계에서 어플리케이션을 실제로 실행시켜 실행 중 악의적인 행동을 하는 지 한번 더 검증을 거친다. 두 번째 단계에서는 정적 분석 단계에서 제대로 분석하지 못한 복잡한 어플리케이션을 처리한다. 동적 분석은 스택 모니터링, 감염 추적, 제어 흐름 분석 세가지 방법을 이용하였다.

정적 분석 모듈에서 생성했던 제어 흐름 그래프는 동적 분석 모듈로 전달된다. 동적 분석의 제어 흐름 분석 단계에서는 어플리케이션을 실행시켜 정적 분석에서 만든 그래프를 벗어나는 실행경로가 나타나는지 살펴 새롭게 만들어진 경로가 악의적인 행동을 위한 것인지 아닌지를 판별한다.

감염 추적은 데이터가 어떻게 행동하는지를 추적한다. 전화번호부 같은 개인적인 데이터나 시스템 폴더와 같이 정상적인 어플리케이션이 접근하지 않는 데이터에 태그를 달아 놓고 이렇게 표시 된 데이터가 어플리케이션의 실행 중에 변조되거나 유출되지 않는지 살펴본다.

버퍼 오버플로우 공격은 상당히 오래 됐지만 아직까지도 가장 흔하게 사용되고 있는 공격 기법이다[13]. 이와 같이 스택에서 이루어지는 공격을 감지하기 위해 스택 모니터링을 사용한다. 스택 오버플로우로 안전하지 않은 코드를 실행시키지는 않는지, 스택 포인터가 스택이 아닌 악의적인 실행 코드를 가리키지는 않는지 체크한다.

정적 분석이나 동적 분석 단계에서 정상적인 어플리케이션으로 판별 된 어플리케이션은 실제로 사용자가 사용 할 수 있도록 한다. 그렇지 않은 경우에는 어플리케이션으로부터 시그니처를 생성하여 시그니처 데이터 베이스에 저장한다. 검사를 거듭할수록 데이터베이스에 저장된 정보가 많아지고 패턴 매칭의 정확도도 올라간다. 시그니처를 생성할 때는 시그니처 자동 생성모듈을 이용한다[14]. 시그니처 생성기로 악성 파일이 들어가면 먼저 함수들을 추출하여 이 중 시그니처로 사용 할 수 있는 것을 걸러내야 한다. 자바나 C 와 같은 높은 레벨의 언어에서 제공하는 서비스 루틴들은 악성 어플리케이션에 이용될 수는 있지만 많은 어플리케이션에서 일반적으로 사용하고 있기 때문에 이러한 함수들을 제외시켜 가며 악의적인 행동을 하는 함수들을 찾아 시그니처를 생성한다.



(그림 3) 클라우드 기반 악성코드 탐지 전체 구조

3.2 클라우드 시스템

본 논문에서는 어플리케이션에 대한 구체적인 분석은 클라우드 시스템이 처리 하도록 한다. 따라서 정적 분석뿐만 아니라 컴퓨팅 자원이 상대적으로 많이 필요한 동적 분석 방식도 추가로 사용 해 악성 코드 탐지의 효율성을 높이도록 한다.

사용자가 어플리케이션을 클라우드 서버에 설치하기로 결정하면 클라우드 서버는 먼저 사용자의 스마트폰과 동일한 환경을 만들기 위해 가상 이미지를 생성한다. 그 후 사용자가 다운 받으려고 하는 어플리케이션을 스마트폰이 아닌 클라우드 서버로 직접 다운받은 후 설치하는데, 설치 전에 악성 어플리케이션 여부를 판단하기 위해 앞에서 설명한 정적 분석 방법을 통해 어플리케이션을 진단한다. 진단 결과 안전한 어플리케이션으로 분류 된 경우 설치를 진행한 후 동적 분석으로 프로그램을 다시 한 번 진단한다. 위험한 어플리케이션으로 분류 된 경우에는 APK 파일을 삭제한다.

어플리케이션을 실행시키려면 클라우드 서버와 사용자의 스마트폰 사이에 많은 데이터 교환이 필요하

다. 또한 어플리케이션과 외부 서버와의 통신이 필요한 경우도 존재한다. 어플리케이션으로부터 만들어진 파일에 의해 사용자의 기기가 공격받거나 개인적인 정보가 밖으로 새 나가는 것을 막기 위해 클라우드 서버는 이러 한 데이터들이 안전한지에 대한 검사도 제공한다. 어플리케이션이 가상 이미지 위에 설치되어 있기 때문에 모바일 기기에서 악성 데이터가 들어오는 경우는 전체 클라우드 시스템에 큰 영향을 끼치지 않는다. 따라서 우리가 제안하는 시스템에서는 클라우드 서버에서 모바일 기기로, 클라우드에서 외부 서버로 나가는 데이터에 대한 검증만 실시한다.

어플리케이션 실행에 필요한 개인정보를 저장하거나 세이브 데이터 파일을 생성해야 하는 경우가 있다. 데이터 보호를 위해 이런 경우 파일이나 데이터는 사용자의 기기에 저장한다. 이 때 스마트폰에 저장되는 파일에 악의적인 실행 코드 등을 삽입함으로써 사용자의 기기를 공격하는 것을 막기 위해 클라우드 서버로부터 모바일 기기로 나가는 데이터들은 한번 더 필터링 과정을 거친 후 내보내도록 해 스마트폰의 오염을 막는다.

많은 어플리케이션들이 외부 서버와의 통신을 필요로 한다. 메신저 앱이나 인터넷 뱅킹, 소셜 네트워크 등의 어플리케이션은 물론이고 게임의 경우에도 순위 등록을 위해 외부 서버를 이용할 수 있다. 신뢰할 수 없는 외부의 서버로 개인정보가 유출 될 위험이 있기 때문에 바깥으로 빠져 나가는 데이터의 단속이 필요하다. 이를 위해 클라우드 시스템이 외부 서버와 통신할 때 밖으로 보내는 데이터의 종류가 무엇인지 체크한다. 연락처나 통화 기록과 같은 개인적인 정보가 바깥쪽으로 나가는 경우에는 사용자에게 어떤 데이터가 전송되려 하는지 알리고 선택할 수 있게 한다.

4. 결론

본 논문에서는 스마트폰을 악성 어플리케이션으로부터 보호하기 위해 어플리케이션을 설치 전 검증하고, 거기에 더해 프로그램 실행 환경을 클라우드 서버에 제공함으로써 사용자의 핸드폰이 악성 코드로 인해 오염되지 않도록 하는 시스템을 제안하였다. 악의적인 행동을 분석하는 데에는 클라우드 서버의 풍부한 컴퓨팅 자원을 활용하여 복잡한 계산에 대한 모바일 기기의 부담을 줄였다.

악성 어플리케이션을 탐지의 첫 번째 단계에서 시그니처를 이용하는데 이것은 잘 알려진 악성 코드를 찾아내는 데에만 사용할 수 있고 시그니처 데이터베이스에 얼마나 많은 양의 데이터가 있는지에 성능이 좌우된다. 또한 다형성 악성코드의 경우에는 기능은 변하지 않지만 실행 할 때 마다 코드의 형태가 매번 달라지기 때문에 시그니처 탐지를 우회할 수 있다. 사용자로부터 얻어지는 데이터 외에 다양한 샘플 어플리케이션의 분석을 통해 자체적으로도 시그니처 데이터베이스를 구축하는 것이 필요할 것이다.

클라우드 서버에서 스마트폰으로 데이터가 나갈 때에는 항상 데이터가 안전한지 검사를 마친 후 확인이 되면 저장을 하도록 설계되어 있기 때문에 사용자의 모바일 기기를 한번 더 안전하게 보호할 수 있다. 하지만 클라우드와 유저 사이의 데이터 전송에 거쳐야 하는 한 단계가 더 늘어나면서 어플리케이션을 사용하는 이용자 입장에서는 어플리케이션의 응답 속도가 느리다고 느낄 수 있다. 클라우드에서 모바일 기기로 나가는 데이터의 검사 시간을 줄이기 위해서는 데이터 처리 속도에 중점을 둔 탐지 모듈을 설계해야 한다. 일정 횟수만큼의 검사를 해 본 결과 깨끗한 파일만 감지된다면 이후의 데이터들은 검사를 생략 하는 등 탐지 과정을 최소화 하는 모듈에 대한 연구가 필요할 것이다.

또한 본 시스템을 실제로 구현했을 때 실제 성능 평가와 보안 분석이 필요 할 것이다.

참고문헌

[1] “이동통신사 모바일 콘텐츠 수익 증감률(2009~2010)”, http://www.kocca.kr/knowledge/internal/stat/__icsFiles/afieldfile/2011/09/26/MqoXPY2C8UOp.pdf, 한국콘텐츠진흥원, Oct, 2012

[2] Chandramohan, M., “Detection of Mobile Malware in the Wild”, IEEE, Jan 2012

[3] Enck, W., Octeau, D., McDaniel, P., Chaudhuri, S. “A Study of Android Application Security”, SEC’11, Aug 2011

[4] Enck, W., Gilbert, P., Chun, B.G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N., “TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones”, OSDI’10, Oct 2010

[5] Shun-Te Liu, “A System Call Analysis Method with MapReduce for Malware Detection”, ICPADS’11, Dec 2011

[6] Burguera, I., Zurutuza, U., Nadjm-Tehrani, S., “Crowdroid: behavior-based malware detection system for Android”, SPSM’11, 2011

[7] Byung-Gon, C., Petros, M., “Augmented Smartphone Applications through Clone Cloud Execution”, HOTOS workshop, USENIX, May, 2009

[8] Portokalidis, G., Homburg, P., Anagnostakis, K., and Bos, H., “Paranoid Android: Versatile protection for smartphones.”, In ACSAC’10, Dec, 2010

[9] Wei, T., Jun-hyung, L., Biao, S., Motaharul, I., Sangho, N., Eui-nam, H., “Multi-Platform Mobile Thin Client Architecture in Cloud Environment”, SESCE ’11, 2011

[10] Xinran, W., Yoon-Chan, J., Sencun, Z., Peng, L., “STILL: Exploit Code Detection via Static Taint and Initialization Analyses”, IEEE, Dec 2008

[11] Xinran, W., Chi-Chun, P., Peng, L., Sencun, Z., “SigFree: A Signature-Free Buffer Overflow Attack Blocker”, IEEE, Mar 2010

[12] Egele, M., Kruegel, C., Kirda, E., Vigna, G., “PiOS: Detecting Privacy Leaks in iOS Applications.”, NDSS, Feb 2011

[13] Aleph One. “Smashing the stack for fun and profit”. Phrack Magazine, 49(14), Nov. 1996

[14] Asaf, S., Eitan, M., Yuval, E., “F-Sign: Automatic, Function-Based Signature Generation for Malware”