모바일 환경에서 악성코드 분석을 위한 효율적 동적 분석기법 연구

김호연*, 장성수*, 최영현*, 정태명**
*성균관대학교 전자전기컴퓨터공학과
**성균관대학교 정보통신공학부
e-mail :*{hykim, ssjang, yhchoi}@imtl.skku.ac.kr, **tmchung@ece.skku.ac.kr

A Study of Efficient Dynamic Analysis for Malware Detection in a Mobile Platform

Ho-Yeon Kim*, SeongSoo Jang*, Young-Hyun Choi*
Tai-Myoung Chung**
*Dept. of Electrical and Computer Engineering, Sungkyunkwan University
**School of Information Communication Engineering, Sungkyunkwan University

요 약

최근의 모바일 플랫폼들은 하드웨어 기술의 발달로 인해 사용가능한 프로그램이 다양화됨에 따라 악성코드의 감염 경로 또한 다양화 되고 있어, 모바일 보안의 관심도 함께 높아지고 있다. 본 논문에서는 모바일 환경에서 사용 가능한 동적 분석 방식을 제안하며, 기존에 제시된 동적 분석 기술들을 분석하여 향후 모바일 플랫폼 환경에서 동적 분석 적용 시 동적 분석 및 혼합분석 툴 개발에 기반을 마련하고자 한다.

1. 서론

가트너의 보고서에 따르면 2010년 스마트폰 OS의 보급 대수는 약 3억대로 전년대비 72.1%의 판매상승을 보였으며 이러한 모바일 플랫폼의 판매현황은 태블릿PC의 추가로 앞으로 더욱 지속될 전망이다[7].

이러한 모바일 플랫폼의 판매현황과 더불어 모바일 플랫폼의 보안위협 또한 증가하고 있다. 유통되고 있는 대부분의 모바일 플랫폼OS의 경우 UNIX를 기반으로 하여 모바일 OS로 사용하고 있다. UNIX의 경우 루트권한을 획득하지 못한다면 악성코드가 사용할 수 있는 파일의 권한이 없어 일반적으로 모바일OS의 경우 악성코드로부터의 안전지대라고 불렸다. 하지만 SMS·MMS 및 블루투스, 인터넷, 그리고 모바일 어플리케이션의 사용과 과거 수동으로 서버에서 제공하던 정보만을 수신하던 소비자가 개개인이 모두 인터넷에 참여하여 다양한 미디어 콘텐츠를 양방향으로 소모함으로 써 악성코드의 감염경로 또한 다양해졌다. 이런 다양한 침입경로를 통해 시스템의 보안을 무력화시키거나 우회하는 악성코드들이 사용자 시스템으로 침투하고 있다. 이렇게 다양한 경로로 침투하는 악성코드들은 악성코드 분석방해 기법을 적용하여 분석 시 악성코드를 발견하기가 점점 어려워지고 있다. 따라서 악성코드를 분석하는 것이 아닌 프로그램을 실행하면서 행위를 분석하는 동적 분석 기술들이 연구되고 있다. 본 논문에서는 다변화하는 보안위협으로부터 모바일 플랫폼을 보호하기위해 모바일 플랫폼에서의 동적 분석을 제안하며, 기존에 제시된 동적 분석 기술을 비교/분석하여 향후 모바일 플랫폼에서 동적 분석이 적용 될 시 기술 선택에 대하여 도움을 주고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로써 모바일 플랫폼의 특성과 분석종류, 그리고 명세화에 대해서 간략히 설명하며 3장에서는 동적 분석 기법들에 대하여 살펴본다. 4장에서는 3장에서 살펴본 동적 분석 기술들에 대하여 비교하고 5장에서 모바일 플랫폼 환경에서의 동적 분석 환경을 제안한다. 그리고 마지막으로 6장에서 본 논문의 결론과 향후 연구 계획을 소개한다.

2. 관련연구

2.1 모바일 보안 요구사항

모바일 플랫폼의 경우 기존의 PC와는 다른 특성을 갖기 때문에 모바일 플랫폼에서 구현되어지는 보안 프로그램 또한 기존의 PC에서 사용되는 프로그램과는 다른 특성을 가져야 한다.

모바일 플랫폼에 사용되어지는 프로그램들의 요구사항을 다음과 같이 정의할 수 있다[8].

- 높은 이식성 : 과거 모바일 OS의 경우 대부분 Symbian 또는 Windows Mobile이 모바일OS 시장의 대부분을 차지했으나 현재 모바일 OS 시장은 RIM사의 BlackBerry, Apple사의 iOS, Google(OHA)사의 Android 등 다양한 플랫폼과 OS가 존재한다. 또한 같은 플랫폼이라도 플랫폼 버전에 따른 호환성 유무가

다른 경우가 존재한다. 이런 다양한 환경에서 동작할 수 있는 설계능력이 요구된다.

- 적은 자원을 사용 : 하드웨어 성능의 발달로 모바일단말의 처리능력은 많은 발전을 이루었지만, 여전히 다른 플랫폼에 비하여 상대적으로 낮은 프로세스의 처리능력, 한정된 자원이라는 제약이 존재한다. 이런 제약환경에서도 사용자의 작업 및 사용 환경에 크게 영향을 미치지 않는 수행능력이 필요하다.
- 확장성 : 악성코드의 위협은 날로 진화하고 모바일OS의 주기는 기존 PC의 교체주기보다 빠르다. 이러한 환경에 부합하는 확장성 있는 구조가 요구된다.

2.2 모바일 환경

최근 출시되고 있는 모바일 플랫폼들은 대부분 권한제어를 받는다. 애플사의 iOS의 경우 UNIX를 기반으로 한다. UNIX의 경우 각 개체에 사용자, 그룹 및 기타에 대한 퍼미션을 할당하여 객체에 대한 접근권한을 제한하여 사용한다. 또한 구글(OHA)사의 Android의 경우도 Linux를 사용하며 UNIX와 마찬가지로 사용자들은 사용되는 디렉터리, I/O 인터페이스, 네트워크 자원 등을 제한받는다. 이는 악성코드를 실행하게 될 경우 악성코드가 포함된 프로세스가 실행한 사용자와 동일한 권한을 갖기 때문에 사용자에 대해 접근권한을 제한한다면 이는 보안상으로 매우 강력한 방법이 될 수 있다. 하지만 이런 권한제어의 경우 보안상으로는 강력한 기능이 될 수 있지만 사용자에 입장에서는 접근할 수 있는 영역이 제한되어서 불편함을 초래한다. 사용자는 이러한 불편함을 해소하기 위해 최고관리자권한(root, administrator)을 얻기 위해 다양한 방법을 사용하게 되는데, 이러한 경우 보안성을 기대할 수 없다. 이 경우 악성코드는 기존의 비밀번호를 사용하여 관리자권한을 얻을 수 있게 되고 악성코드 삽입 등 악의적인 명령을 수행할 수 있다.

이처럼 OS상에서 강력한 보안을 제공한다고 하더라도 사용자의 부주의로 인하여 많은 보안 위협으로부터 노출되어 있다.

2.3 정적 분석과 동적 분석

프로그램 분석은 프로그램의 분석시점에 따라 정적 분석과 동적 분석으로 나뉠 수 있다.

정적 분석의 경우 프로그램이 실행되기 전 프로그램이 실행할 때 발생할 수 있는 행위를 어림하는 분석방법으로, 프로그램이 실행되기 전 수행할 수 있기 때문에 안전성을 요하는 시스템에서 분석하기에 적합하다. 하지만 실제 프로그램을 실행하지 않고 분석하기 때문에, 실행시간에만 정확히 알 수 있는 메모리에 대한 사용이나 사용자의 다양한 입력에 대해서는 분석 시 미처 파악하지 못할 수 있는 단점이 있다. 또한 시그니쳐 기반의 패턴매칭을 통한 악성코드 판단유무를 검출하는 방법 사용 시, 신종 악성코드가 유포된다면 악성코드를 판단 할 수 없다.

동적 분석은 정적 분석과 달리 프로그램을 실행하며 해당 프로그램이 수행하는 모든 과정을 관측하는 분석 방법이다. 새로운 악성코드가 나타나더라도 악성코드 특유의 행위는 정해져 있기 때문에 동적 분석을 사용한다면 새로운 악성코드가 유포된다 하더라도 탐지가 가능하다.

최근의 악성코드들은 기하급수적으로 증가하며 이렇게 급증하는 악성코드들은 다양한 분석지연/방해기법(nop 삽입, 데이터 변경, 제어흐름 변경, 패킹 등)들을 적용하여 널리 퍼지고 있어, 악성코드가 가지고 있는 분석지연/방해기법들을 무력화 하거나 우회해야 분석이 가능하다.[1] 따라서 악성코드가 행하는 행위 자체를 분석하기 위한 동적 분석 기법과 정적 분석과 동적 분석을 혼합한 혼합분석에 대한 연구가 활발히 진행되고 있다.

2.4 명세화

동적 분석의 대상인 프로그램을 정상적인 프로그램인지, 비정상적인 프로그램인지 판단하는 기준을 명세화하는 명세화 기술이 필요하다. 이러한 명세화를 하는 기준으로는 기존에 존재하는 악성코드들의 특징을 파악하여 해당하는 행위를 수행할 경우 악성코드라고 판단할 수 있다. 예로. 명세화 방법은 악성코드들이 주로 실행하는 행위들에 대하여 위험도를 정하고 위험도의 합이 임계치를 넘게 된다면 악성 프로그램으로 판단하는 방법을 사용할 수 있다. 또 다른 방법으로는 데이터 마이닝 기법을 사용하여 발견되지 않았던 악성코드들 간의 상호관계를 분석하여 새로운 악성코드 판단 기준을 만들 수 있다. 데이터 마이닝 기법을 사용할 경우 수동적인 평가기준을 마련하지 않아도 악성코드 판별기준을 정할 수 있다. 하지만 데이터 마이닝을 사용하기 위해선 기존에 악성코드들의 상관관계를 정의해야 하며 정확하지 않은 데이터 마이닝의 경우 오탐율이 매우 높아진다.

이러한 명세화 작업의 경우 악성코드의 판단 여부를 너무 자세히 기술하게 되면 악성코드의 약간의 변형만으로도 탐지를 할 수 없으며, 이와 반대로 악성코드의 판별여부를 광범위하게 정의한다면 오탐율이 높아지므로 적은 오탐율과 높은 탐지능력을 행할 수 있는 최적의 명세화 기술이 요구된다.

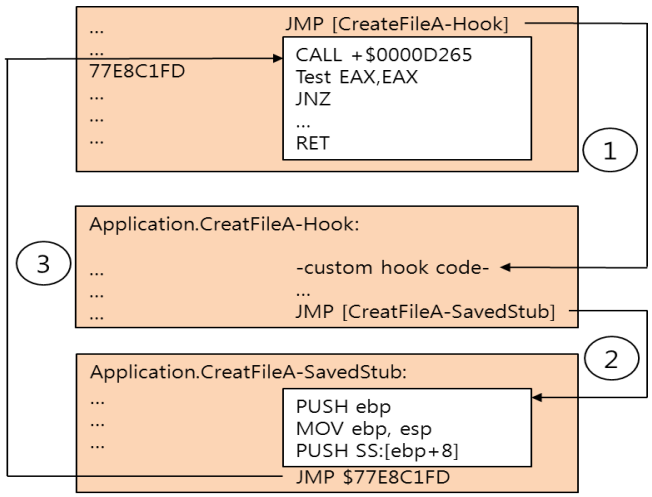
3 동적 분석 기술 분석

본 장에서는 다양한 동적 분석 기술을 살펴봄으로서 모바일 플랫폼에 적합한 동적 분석 개발을 제시한다.

3.1 API 후킹[2]

API 후킹은 악성코드가 호출하는 모든 API들을 감시하기 위해 API를 호출하는 시작부분에 임의의 코드를 삽입하여 기록하는 방법이다. 악성코드들이 수행하는 악의적인 코드들은 특정 API를 사용하기 때문에 사용하는 API들에 대하여 위험도를 기록하고 이에 대해 악성프로그램인지 정상적인 프로그램인지 판단할 수 있는 근거가 될

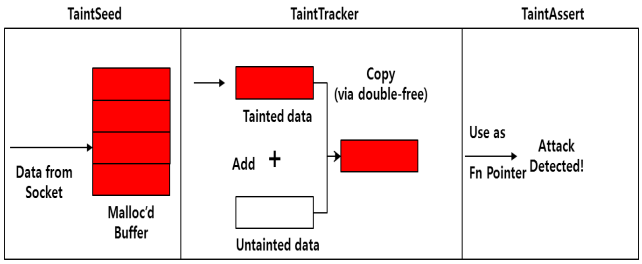
수 있다. (그림 1)은 기본적인 API 후킹의 개념도 이다.



(그림 1) API 후킹

3.2 Taint Analysis[3]

신뢰할 수 없는 데이터의 흐름을 파악하는 기법으로써 신뢰할 수 없는 데이터가 어디까지 시스템에 영향을 미치는지 알아내는 기법이다. Taint Analysis 또한 API 후킹과 마찬가지로 입력되는 데이터의 추적을 위해 데이터 뒤에 임의의 코드를 삽입하여 데이터의 경로와 tainted 데이터의 확산을 확인할 수 있다. [3]에서는 입력에 대해 비신뢰 데이터의 구별을 위한 TaintSeed와 tainted 라고 의심되는 데이터라고 확산을 추적하기 위한 TaintTracker, 그리고 taint 알람을 위한 TaintAssert라는 세 단계로 구분하여 비 신뢰 데이터에 대하여 대처하고 있다.



(그림 2) Taint Analysis

- TaintSeed : 외부에서 입력되는 모든 데이터를 비 신뢰하여 tainted라고 판단, 모든 입력에 대하여 마크하여 untainted 데이터와 tainted 데이터를 구분 짓는다.
- TaintTacker : Taint 경로를 추적하기 위하여 UCode를 세 분류로 나누어 명령어에 따라 각기 다른 행동을 취한다. 분류로는 이동명령어(MOVE, PUSH, POP 등)와 연산명령어(ADD, SUB, XOR, 등), 이 두 분류를 제외한 모든 명령어들로 분류하여 taint 데이터를 정의한다.
- TaintAssert : TaintAssert 단계는 데이터의 경로를 추적하여 데이터가 목적지에 도착했을 때, 미리 정의된 정책에 의하여 해당하는 데이터가 신뢰할 수 있는 데이터인지 판단하는 단계다. TaintAssert는 악의적인 프로그램들이 자주 사용하는 방법들을 미리 명세화 하여 해당하는 방식을 사용하였는지를 판단하여 최종적으로 프로그램의 데이터의 taint 여부를 판별한다.

3.3 Short Sequence of System Calls[4]

Sequences of Calls기법의 경우 프로세스가 행하는 시스템 콜을 감시하는 기법이다. 이 기법은 [5]에서 시스템 콜의 시퀀스 해밍길이가 짧을수록 정상적인 프로그램이며 길수록 비정상적인 프로세스라는 데에 기반을 둔다. 본 기법은 기존에 명세된 시그니쳐에 기반을 두지 않고 정상적인 프로그램의 대해서 시스템 콜의 시퀀스 해밍길이를 구하여 DB화 한다. 해당 DB를 기준으로 하여 실행중인 프로그램에서 호출하는 시스템 콜의 시퀀스 해밍길이를 구하여 미스매치 확률을 구하고 특정 미스매치율을 초과하면 비정상 프로그램이라고 판단한다.

3.4 동적 정보흐름 추적[6]

동적 정보흐름 추적기법의 경우 악성코드가 프로그램의 취약점을 스캔하여 임의의 악성코드를 삽입하는 공격을 찾아내는 기법이다. 프로세스의 행위를 추적하기 위하여 비정상적인 데이터가 생성하는 데이터들에 대하여 태깅을 수행하여 경로를 추적할 수 있다. [7]에서는 경로 추적을 위하여 세 보안정책을 적용하고 있다. 정책들로는 spurious input channels, dependencies to be tracked, 그리고 restrictions로 나누고 있다.

- Spurious input channels : 비 신뢰적인 네트워크로부터 유입되는 데이터들에 대하여 각각 태깅을 실시한다. Taint Analysis와는 다르게 안전한 채널에 대해서는 태깅을 하지 않는다.
- Dependencies to be tracked : 태깅 된 데이터들의 경로를 확인하기 위해서 확산되는 데이터들을 추적 하는 역할을 한다. 기본적으로 추적을 위해 다음과 같은 행위들을 악의적인 행위라고 간주하고 추적하게 된다. 복사, 연산, 주소참조, 주소저장이라는 행위를 프로세가 수행할 시 데이터의 흐름을 추적한다.
- Restriction : 최종적으로 태깅된 데이터들을 악의적인 데이터인지 정상적인 데이터인지를 정의된 정책에 어긋나는 지로 판별한다. 기본적인 보안정책으로는 허용되지 않은 명령어 사용 및 허용되지 않은 메모리의 사용 여부 이다.

4. 동적 분석 기술 비교

<표 1>은 각 기법에서의 동적 분석 기술에 대한 비교표 이다. 비교 목록이 의미하는 바는 다음과 같다.

- 검사 목록 : 각 기법이 수집하는 데이터
- 악성코드 판별기준 : 수집된 데이터를 사용하여 악성코드를 판단하는 기준

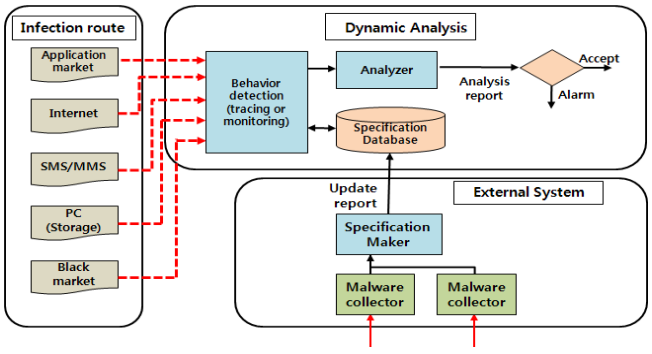
<표 1>동적 분석 기법의 비교

	검사 목록	악성코드 판별기준
API 후킹	API 호출 목록	권한 외의 API사용, API사용의 점수화
Taint Analysis	모든 외부 입력 데이터, 비 신뢰 데이터의 흐름 경로	최종 목적지에서의 데이터 테인트 여부
Short Sequence System Calls	시스템 호출 목록, 시스템 호출간의 시퀀스 해밍거리	시스템 호출 시퀀스 해밍거리간의 미스매치 율
동적 정보흐름 추적	비 신뢰 채널으로부터의 모든 입력, 입력에 대한 데이터의 흐름 경로	허용되지 않은 명령어 사용, 허용되지 않은 메모리 참조

5. 모바일 동적 분석 기법의 개발

3절과 4절에서 살펴본 동적 분석 기법들은 프로세스의 현재 행위를 분석하기 위하여 경로 추적 및 사용하는 시스템 콜, 메모리의 사용, 레지스트리의 변화 등을 메모리에 기록하며 모니터링 하기 위해 많은 시스템 자원을 할애한다. 이와 같은 오버헤드를 해결하기 위하여 모바일 플랫폼 상에서 이미 정의된 악성 행위를 기반으로 동적 분석을 수행하는 방식을 제안한다.

(그림 3)은 외부 시스템에서 악성코드 수집 시스템을 사용하여 악성코드를 수집한 후 악성코드의 특정 행위를 분석하여 명세화 한 자료를 근거로 모바일 플랫폼 상에서 동적 분석을 수행한다. 위와 같이 악성코드의 알려진 행위를 외부 시스템으로부터 업데이트를 받음으로써 알려지지 않은 악성 행위들을 기록하며 평가하는 프로세스가 줄어들어 한정된 자원을 사용하는 모바일 플랫폼에서도 동적 분석이 수행 가능하다.



(그림 3) 오버헤드를 줄이기 위한 동적 분석 방식

6. 결론

5장의 동적 분석 기법의 경우 알려진 행위에 대해서만 동적 분석을 수행하는 방식은 제로데이 공격과 같이 새로운 악성코드를 탐지할 수 없는 단점이 있다. 알려진 행위에 대해서만 분석을 수행한다는 것은 정적 분석과 달라 보이지 않으나 동적 분석 특유의 입력값에 대한 흐름의 변화 등을 감지 할 수 있다. 또한 정적 분석의 경우 패킹(packing)된 PE데이터에 대해서는 언패커(unpacker)가 필요하다. 또한 다중패킹 및 알려지지 않은 패커로 패킹 될 시 정적 분석에 어려움이 있으며 이런 언패킹 과정을 모바일 플랫폼에서 수행하기에는 많은 오버헤드가 생겨나 비효율적이다.

본 논문에서 제시하는 동적 분석의 방법의 경우 새로운 악성코드를 탐지하기 위해서는 외부 시스템에서 새로운 악성코드를 미리 탐지하여 빠른 업데이트가 필수적이다. 이를 위해서는 웹 크롤러 또는 클라이언트 허니팟과 같은 악성코드 자동 수집 기술들의 보다 많은 연구와 보다 간결한 동적 분석 연구가 필요할 것이다. 이를 위하여 본 논문을 기반으로 향후 불필요한 데이터의 동적 분석을 제외시켜 오버헤드를 줄인 동적 분석 기술에 대한 연구를 진행할 계획이다.

Acknowledgment

본 논문은 중소기업청에서 지원하는 2010년도 산학연공동기술개발사업(No.00044301)의 연구수행으로 인한 결과물임을 밝힙니다.

참고문헌

[1] 임채태 외, "최신 악성코드 기술동향 및 분석 방안 연구," 정보과학회지, Vol.28, No.11, pp.117-126, 2010.

[2] C. Willems, T. Holz, and F. Freiling, "Toward automated dynamic malware analysis using CWSandbox," Security & Privacy, IEEE, Vol.5, No.2, pp.32-39, 2007.

[3] J. Newsome, and D. Song, "Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software," Network and Distributed System Security Symposium, 2005.

[4] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," Journal of Computer Security, Vol.6, No.3, pp.151-180, 1998.

[5] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," sp, pp.0120, 1996.

[6] G. E. Suh, J. W. Lee, D. Zhang, and S. Devadas, "Secure program execution via dynamic information flow tracking," Architectural Support for Programming Languages and Operating system XI, pp.85-96, 2004.

[7] http://www.gartner.com/it/page.jsp?id=1543014, 2011

[8] http://radar.ndsl.kr/radDetail.do?cn=GTB2010080467, 2011