실행시 구조분석을 통한 악성코드 탐지기법 분석

오형근*, 김은영, 이철호
국가보안기술연구소

The analysis of malicious code detection techniques through structure analysis at runtime

HyungGeun Oh*, EunYoung Kim, CheolHo Lee
National Security Research Institute

요 약

본 고에서는 악성 프로그램을 탐지하기 위해 특정 프로그램 실행시 해당 파일 구조를 분석하여 악성 프로그램을 탐지하기 위해 Michael Weber, Matthew Schmid, Michael Schatz와 David Geyer에 의해 제안된 실행코드 탐지 방식을 분석하고 있다. 제안된 방식에서는 기존 방법에서 사용하고 있는 악성 프로그램의 시그니처 분석을 통한 탐지 방법과 다르게 윈도우 PE 파일 형태의 파일 구조를 가지고 있는 실행 프로그램의 문맥 분석을 통해 알려지지 않은 악성 프로그램을 탐지함을 목적으로 하고 있다. 제안된 방식에서는 특히 PEAT(Portable Executable Analysis Toolkit)라는 이동 실행 분석 툴 킷을 개발, 사용함으로써 악성프로그램을 탐지 하고 있는데 이 툴 킷은 PE 파일 구조를 가진 임의의 애플리케이션에 대해 악성코드의 존재 여부를 밝힐 수 있는 실행시 구조적 특징을 이용한다.

1. 서론

컴퓨터와 통신 기술의 발달은 고도의 정보 통신망을 구축하는데 큰 공헌을 하였으며 정부기관을 비롯하여 금융권, 기업체 그리고 개인에 이르기까지 각종 정보를 공유함으로써 사회 전반에 대해 큰 변화를 가져왔다. 그러나 인터넷(Internet) 등의 컴퓨터 통신망을 통한 정보의 위조, 변조 및 유출을 비롯한 각종 불법 행위로 인해 많은 문제점이 대두되는 등 정보화로 야기되는 역기능이 심각한 사회 문제로 등장하고 있다. 특히, 한 통계조사에 따르면 악성프로그램의 증가는 2조 6천억이라는 엄청난 사회적 비용을 발생시키고 있으며 2003년도 초에 발생한 1.25 인터넷 대란에서 볼 수 있듯이 국가정보 인프라 마비라는 비상사태를 초래시킬 수 있음을 확인한 바 있다. 또한, 우리는 백신 프로그램을 사용했으나 그 효과면에서 만족스럽지 않다는 것을 확인하였다. 이것은 기존 수동적 형태의 악성 프로그램 탐지 기법의 한계를 드러내는 것으로 보다 능동적으로 악성 프로그램에 대처하기 위한 새로운 탐지 시스템이 필요함을 나타내고 있다. 따라서, 이와 관련된 탐지 기술에 대해 살펴보고 실행시 구조 분석을 통해 악성코드를 탐지하는 PEAT에 대해 분석하도록 한다.

2. 악성코드 탐지 기술 현황

바이러스, 웜, 트랩도어 또는 백 도어 등의 다양한 형태로 존재하는 악성코드는 악성 프로그램, 악성실행코드 등 여러 가지 이름으로 불리고 있으며 그 종류도 IT 기술이 발전함에 따라 다양한 형태가 지속적으로 등장하고 있다. 이와 같이 그 종류의 다양성, 실행 환경의 상이함, 일반 사용자의 보안 인식의 부재, 악성코드를 쉽게 제작할 수 있는 악성코드 제작 도구 개발 등으로 인해 악성코드 탐지 기술 개발에 많은 어려움이 따르고 있다. 이와 같은 어려움에도 불구하고 현재 많은 백신 업체들에서 악성프로그램을 탐지하기 위한 탐지 기술을 개발하고 있으며 좋은 성능의 탐지 엔진이 개발되어 있다. 그러나, 이러한 탐지 엔진은 알려진 악성코드에 대해서만 탐지가 가능하며 알려지지 않은 악성코드에 대한 탐지는 불가능하다는 단점이 있다. 이러한 방식은 악성코드 출현 후 피해가 발생하고 나서 백신 업체에 의해 분석 및 대응 패치가 보급된다는 점에서 그 한계가 있다. 본 장에서는 현재 백신 업체 등에서 개발 보급하여 사용되고 있는 여러 가지 탐지 기술 등에 대해 살펴보도록 하겠다.

2.1 바이러스 지문 검사법

백신 프로그램이 바이러스를 진단하는 방법들 중의 한 가지로서 사람들을 구분할 때 지문으로 구분하듯이 컴퓨터 바이러스가 가지고 있는 독특한 문자열인 '패턴'을 통해서 바이러스를 검색하는 방법이다. 따라서, 바이러스의 패턴을 많이 확보할수록 보다 많은 바이러스를 진단할 확률이 높은 것으로 간주된다. 이러한 지문 검사법을 수행하기 위한 방법은 다시 크게 2가지로 구분된다. 순차적 문자열 검사법과 특정 문자열 검사 방법이 그것이다. 순차적 문자열 검사법은 바이러스 실행 코드에 대해 코드 처음부터 끝까지 실행코드를 따라 가면서 진단하는 방법으로 검색 속도가 빠르다는 장점이 있다. 하지만 이러한 방법은 변형 바이러스를 탐지하는데 있어서 특정 문자열 검사법보다 탐지율이 떨어진다는 단점을 가지고 있다. 반면에 특정 문자열 검사법은 파일을 전체 혹은 일부를 검색해 특정 문자가 있는지 검사하는 방식이다. 이 방법은 검색 속도가 다소 떨어지지만 변형 바이러스를 쉽게 파악할 수 있다는 장점이 있다.

2.2 휴리스틱 검사법

전통적으로 백신 제품의 솔루션은 시그니처 기반의 스캐닝 (signature-based scanning) 기법이다. 이 기법은 오랫동안 백신 제품의 최적의 솔루션으로 평가받고 있다. 그러나

시그너쳐 기반의 스캐닝 기법은 알려지지 않은 악성코드를 탐지하기에는 역부족이다. 따라서 시그너쳐 기반의 스캐닝 기법을 좀더 보완하기 위하여 알려지지 않은 바이러스를 탐지하기 위한 방법으로 휴리스틱 스캐닝 기법이 개발되고 있다. 휴리스틱 스캐닝 기법은 시그너쳐 기반의 스캐닝 기법과 유사하다. 그러나 휴리스틱 스캐닝 기법은 시그너쳐 기반의 스캐닝 기법과 같이 어떤 특정 시그너쳐만을 찾는 방법 대신에, 보통의 응용 프로그램에서 발견할 수 없는 어떤 특별한 명령이나 행위를 찾는다. 따라서 이러한 휴리스틱 기법을 이용하여 구현된 탐지 엔진은 바이러스, 웜 또는 트로이안 등 알려지지 않은 새로운 악성행위를 탐지하는데 이용될 수 있다. 휴리스틱 스캐닝 기법은 다음과 같이 크게 두 가지로 구분할 수 있다. 가중치 기반 시스템(Weight-based systems)와 규칙 기반 시스템(Rule-based systems)이 있다.

2.3 CRC 검사법

CRC(Cyclic Redundancy Check)는 시리얼 전송에서 데이터의 신뢰성을 검증하기 위한 에러 검출 방법의 일종이다. CRC에 의한 방법은 높은 신뢰도를 확보하며 에러 검출을 위한 오버헤드가 적고, 랜덤 에러나 버스트 에러를 포함한 에러 검출에 매우 좋은 성능을 가지고 있는 것을 특징으로 한다. 이러한 CRC 방법에는 보통 2가지 종류가 사용되는데, 원칩 마이크로 프로세서와 같이 간단한 용도에서는 CRC-16이 사용되고, 이보다 더욱 정확한 에러 검출이 필요한 경우에는 CRC-32를 사용한다. CRC 검사법은 오진율이 낮다는 장점을 가지고 있지만 1 byte만 변형되어도 진단하지 못함으로써 변형된 악성코드에 대한 탐지가 불가능하다는 단점을 가지고 있다.

2.4 가상 실행 검사법

백신 프로그램 안에 가상의 컴퓨터 환경을 만든 후 파일이 동작하는 기능 하나하나를 분석해서 악성코드로서의 특정한 동작을 하느냐 하지 않느냐로 구분하여 악성코드 존재 여부를 찾아내는 검사법이다. 일종의 시뮬레이션(simulation, 모의 실험) 기법이라고도 한다. 즉, 가상실행엔진을 이용하여 악성코드를 실행해 가면서 추적하는 방식이며, 기존의 패턴 매치를 이용한 방법에서 해결하지 못했던 복잡한 암호화 및 다형성 방법에까지 폭넓게 적용 가능하다는 장점이 있다.

2.5 특정 위치 진단법

이 진단법은 파일 하나하나의 내부를 분석해서 특수한 문자열을 찾아내는 방법으로 사전에 악성코드에 대한 특징을 분석하여 데이터베이스에 악성코드만이 가지고 있는 대한 특정 문자열을 가지고 있다가 해당 문자열이 특정 위치에 존재하면 악성코드로 진단한다. 예컨대 주소 0000번지에 'VBS'라는 단어가 있으면 해당 파일이 악성코드에 감염되었다는 것을 알아내는 식이다. 이 방법은 기존에 특정 파일의 코드 전체를 처음부터 끝까지 찾는 방식을 개선한 것으로 특정 문자열 코드가 특정 위치에만 존재한다는 사실을 알아낸 후 이를 이용해 개발된 방식이다. 대표적인 국내 백신인 V3 Warp 엔진에서 사용되고 있는 방식으로 기존 방식에 비해 스캔 효율이 향상되었다는 장점을 가지고 있다.

3. 면역 시스템

악성프로그램에 의한 피해를 최소화 할 수 있는 방법 중의 하나가 바로 바이러스 엔진 업데이트인데, 사용자들의 미숙함으로 인해 그 피해가 더욱 늘어나고 있다. 이러한 피해를 줄이기 위한 방법으로 자동 면역 시스템 기술(Automatic Immune System Technology, AIST)이 있다.

자동 면역 시스템 기술은 신종 악성프로그램을 탐지하고 그 정보를 내부 네트워크로 연결된 컴퓨터들이 다같이 공유할 수 있도록 하여 바이러스의 확산을 미연에 방지할 수 있도록 하는 시스템이다.

제안되고 있는 면역 시스템들은 면역계통들의 역할을 컴퓨터 보안 시스템에 적용함으로써 기존 보안 시스템의 한계에서 벗어나 능동적이고 항구적인 보안 체계를 갖추고자 하는 의도에서 자연 면역 시스템의 컴퓨터 보안 시스템에 대한 적용 방법이 연구되었다. 우리가 오늘날 자연 면역학 체계를 컴퓨터 보안 문제에 적용하고 있는 네 가지 예가 있는데 그것은 바로 호스트 기반의 침입 탐지 기법, 네트워크 기반의 침입 탐지 시스템, 분류할 수 있는 변화 탐지 알고리즘 및 취약점 감소를 위한 차이점 소개 방법 등이 그것이다. 다음에서는 컴퓨터 시스템에서의 면역 체계 구성을 위해 제안 및 연구되고 있는 방안들에 대해 살펴보도록 한다.

3.1 IBM 면역 시스템

악성프로그램이 가지는 커다란 특징들 중에 하나는 인터넷이나 전자메일 등을 통해 빠르게 전파된다는 점이다. 이러한 형태의 악성프로그램들에 대응하기 위한 방법으로써 IBM에서는 자연 세계의 면역 시스템이 가지는 특징들에 관심을 가지고 이에 대해 연구하기 시작했다. 인간의 면역 체계는 이질적인 유기체를 발견하게 되면 이 침입자를 탐지하고 퇴치하는 항체를 생성하는 점을 이용해 이를 바이러스 퇴치에 적용하게 된 것이다. 면역 체계는 미래에 똑같은 악성프로그램이 침입할 경우를 대비해 그 악성프로그램에 관한 충분한 정보를 기억하도록 한다. 일단 최초의 침입을 물리친 면역체계는 이 항체의 일부를 남겨두어서 미래에 똑같은 악성프로그램이 침입했을 때 빠른 속도로 대처한다. 백신 프로그램은 악성프로그램 전체를 인식하지 않고 그들의 시그니쳐만을 인식하는 "항체"를 가지고 있다. IBM의 연구 목표는 프로그램의 유추능력을 향상시켜서 악성프로그램을 빠르게 확인할 수 있고, 하나의 컴퓨터뿐만 아니라 컴퓨터 네트워크 전체에서도 확인이 가능하도록 하는 시스템을 개발하는 것이다. IBM의 시스템은 새로운 악성프로그램의 시그니쳐와 복구 방법을 자동적으로 생성한다고 했는데, 실험 결과에 의해 나타난 긍정적 오류(False Positive)가 상당히 높았다. 여기서 긍정적 오류란 악성프로그램이 아닌 경우에도 악성프로그램이라고 탐지하는 경우를 말하는데 현재의 기술로는 상용화에 만족할 만한 수준이 되지 못하고 있다.

3.2 시만텍의 디지털 면역 시스템(Digital Immune System)

면역시스템을 안티 바이러스 기술에 최초로 적용한 기업이 시만텍이다. 시만텍은 디지털 면역 시스템(Digital Immune System)에 대하여 IBM과 수년동안 공동연구를 해오고 있다. 시만텍과의 수년간 공동 개발 끝에 상용화된 시만텍 디지털 면역 시스템은 바이러스 감지와 진송 기술인 검색 및 송부(Scan & Delivery) 기술과 자동 백신 개발 기술인 시만텍 안티 바이러스 자동복구(SARA : Symantec Anti-virus Research Automation) 기술로 구성된 폐쇄 순환 자동 시스템(closed-looped automated system)을 가지고 있다. 검색 및 송부 기술(Scan & Delivery)은 기업체 전산 시스템 내에서 바이러스로 손상된 파일이나 감염이 의심스러운 파일을 발견해 이를 시만텍 안티 바이러스 연구소로 자동 전송시키고 새로운 바이러스 백신을 E-Mail로 받아들이는 역할을 담당한다. 시만텍 안티 바이러스 연구소는 전송된 바이러스를 시만텍 안티 바이러스 자동 복구 기술에

의해 자동적으로 백신을 개발, 처음 바이러스를 보고한 기업으로 다시 전송한다. 또한 노턴 안티 바이러스 확장 엔진 기술을 이용, 전 세계 모든 시스템으로 전파해 다른 기업 전산 시스템이 같은 바이러스에 감염되는 것을 방지하게 하고 있다. 디지털면역시스템(DIS)은 다음과 같이 작동된다.

- 데스크 탑, 서버 그리고 게이트웨이 등에서 새로운 위협을 탐지
- 분석을 위해 의심되는 파일은 시만텍 보안 대응 센터(Symantec Security Response)로 전송
- 탐지 보고된 정보의 관리
- 탐지 보고된 바이러스 정보의 안전한 전송과 새로운 탐지 정책의 안전한 배포
- 전송 정보에 대한 처리
- DB 검사와 환경설정
- 탐지 바이러스 정보 필터링
- False Positive의 감소

3.3 UNM(University of New Mexico)의 면역 시스템

UNM에서는 컴퓨터 면역 시스템(Computer Immune System)에 대한 연구가 활발히 진행되고 있다. Stephanie Forrest 교수와 Geoff Hunsicker, Ken Ingham, Hajime noue, Anil Somayaji, Christy Warrender 등의 학생들이 연구하면서 다수의 논문과 연구 실적을 발표하고 있다. 여기에서는 주로 면역 시스템을 이용한 네트워크 기반 침입탐지에 대하여 연구하고 있는데 주로 면역 시스템을 이용한 네트워크 기반 침입 탐지에 대해 연구하고 있다. 특히 pH(process Homeostasis, 프로세스 항상성)라는 비정상적 프로그램 행위를 탐지하고 해당 프로그램의 실행을 방지하는 개념의 리눅스 커널 익스텐션을 개발하였다. 해당 리눅스 커널에서는 다른 프로그램들에 의해 만들어진 시스템 콜의 패턴 프로파일을 자동적으로 만들어서 그와 같은 이상 행위를 탐지한다.

4. PEAT 탐지 기법 분석

PEAT(Portable Executable Analysis Tookit)는 실행시 윈도우즈 PE 파일 구조를 분석함으로써 악성여부를 판단할 수 있도록 Cigital사에서 근무하고 있는 Michael Weber, Matthew Schmid, Michael Schatz와 George Mason 대학의 David Geyer가 제안한 툴 킷이다. 이 툴은 악성코드의 존재 여부를 밝히기 위해 윈도우즈 PE 파일 포맷 형태를 가지고 있는 파일들의 실행시 구조적 특징을 분석할 수 있도록 해 준다. 이 툴이 사용되기 위한 전제 조건은 우선 특정 프로그램이 컴파일 될 때 단지 하나의 바이너리 코드로 컴파일 되어야 한다는 것과 코드 시작 부분부터 끝까지 구조적 특징들에 대해 동질성을 가져야 한다는 점이다. 이러한 동질성이 훼손되었다는 것은 곧 바이너리 코드가 변조되었다는 것을 의미한다. PEAT 툴은 다음의 세 가지 항목으로 구성되어 있다.

- 시각화(Visualization)
- 정적 검사(Static Check)
- 자동화된 통계학적 분석(Automated Statistical Analysis)

4.1 시각화(Visualization)

PEAT는 PE 파일들의 다음과 같은 몇 가지 특징들을 그래프식으로 나타내어 준다.

- PE 파일에 있어서의 바이트 영역이 코드, 패딩, ASCII data 혹은 랜덤 바이트 값들을 포함하고 있을 가능성
- 점프, 콜 혹은 레지스터 접근 등과 같은 연산을 수행하는 명령어들에 대한 어드레스 오프셋(address offsets)
- 어떠한 행동을 가르키는 것이 알려진 명령어 패턴

PEAT에서 제공하는 이러한 비주얼 툴 킷을 이용해서 사용자들은 디스어셈블된 코드 리스트를 볼 수가 있다. 또한 ASCII 스트링을 식별하기 위해 사용자는 주어진 영역 내에서 모든 바이트 값들에 대한 ASCII 설명을 볼 수 있다. 이러한 비주얼 툴들을 통해 변형되지 않은 완전한 프로그램과 변형된 프로그램 영역을 식별할 수 있도록 하고 있다.

4.2 정적 검사(Static checks)

PEAT는 악성코드에 대한 정보를 얻기 위해 윈도우즈 PE 파일에 대한 정적 검사를 수행한다. 이러한 정적 검사에서는 우선 PE 파일의 헤더로부터 확보된 실행코드의 엔트리 포인트 주소를 검사한다. 그리고 나서 엔트리 포인트 다음의 위조된 명령어를 호출하는 "위조된 호출(bogus call)"를 찾는다.

실행코드 상에서의 명령어 순서는 메모리에서 명령어들의 주소를 결정하기 위해 악성코드들이 사용하는 기본적인 방법이다. 이것은 명령어 포인터 레지스터 EIP가 CALL 명령어를 스택에 쓰기 때문인데 바이러스는 즉시 EIP를 가로채어 사용한다. 이와 같이 명령어 순서가 불분명할 수 있다는 성질 때문에 PEAT는 PE 파일이 디스어셈블된 이후에 명령어들(bogus call)의 존재를 파악하여 명령어 순서 이상시 사용자에게 경고를 하게 된다.

마지막으로 PEAT는 DLL 라이브러리들을 PE 파일들의 임포트 테이블에 넣을 것인지를 결정하고 그 각각의 이름과 컨텐츠를 기록한다. 또한 PEAT는 DLL 라이브러리에 있는 함수를 호출하는 실행코드들에 있는 모든 명령어들을 찾고 해당 라이브러리 및 함수 이름과 명령어들의 위치를 기록한다. 이것은 정상적인 코드 수행 범위를 벗어나 이상행위 수행 여부를 결정하기 위한 선행 단계이다.

4.3 통계학적 분석(Statistical analysis)

PEAT는 또한 PE 파일 내에서 비정상적인 영역을 식별하기 위해 통계학적인 방법을 사용하는 PE 파일에 대한 분석 기능을 제공하고 있다. 사용자는 실행코드로부터 아래와 같은 항목을 뽑아 낼 수 있다.

- 명령어 빈도
- 명령어 패턴
- 레지스터 오프셋(offset)
- 점프와 콜 오프셋(offset)
- opcode 값들에 대한 엔트로피
- 코드와 ASCII 확률

이와 같은 값들은 프로그램내에서 어떤 의심스러운 영역을 식별하는데 유용하게 사용될 수 있다. 다음에서는 입력 데이터로서 사용되는 특징들에 적용시킬 수 있는 통계학적인 접근법에 대해 분석해 본다. PEAT가 PE 파일에 대한 자동화된 분석을 수행할 때, PEAT는 PE 파일의 각 섹션에 대해 반복적인 분석 작업을 수행한다. 그 섹션은 명령어들로 디스어셈블 되며, 그리고 나서 고정된 명령어 숫자들에 상응하는 n 개의 연속적인 disjoint 윈도우들로 분할된 뒤 각 윈도우에 대해 아래와 같은 데이터 포인트 리스트를 계산한다.

$$X = (x_1, x_2, \ldots, x_n)$$

이 리스트로부터 아래와 같은 또 다른 데이터 포인트 리스트를 만든다.

$$Y = (y_1, y_2, \ldots, y_{n-1})$$

이 값(y_i)은 $y_i = x_{i+1} - x_i$를 이용해 계산된다.

다음으로 PEAT는 각 항들에 대해 위의 연산을 반복적으로 수행하고 각각의 항 X에 존재하는 데이터 포인트가 X에 있는 나머지 데이터 포인트들에 대해 확률 범위에서 벗어나는지를 결정한 뒤 항 i에 대해 X/x_i에 대한 평균과 표준 편차를 계산한다. 그리고 x_i가 평균에 대한 두 개의 표준 편차 내에 존재하는지를 결정한다. 만약 표준 편차 안에 존재하지 않는다면 각 항은 비정상으로 판명되며 이러한 방법은 섹션에 있는 다른 항들과 비교하여 비정상적인 엔트로피를 가지고 있는 항들에 대한 리스트를 만들어 낸다. 비슷한 절차가 Y 데이터 포인터들에 대한 항들에 적용된다. 예를 들어 만약 공통적인 명령어 패턴들이 섹션 안에 있는 어떠한 포인트를 계속해서 유지하다가 갑자기 사라진다면 이러한 행위들은 이상 행위로 기록된다. X와 Y 데이터 포인트를 모두 사용하는 것은 처음의 절반은 정상인 반면 다음의 절반에 악성 코드가 있어서 X 포인트가 섹션에서 비정상적인 영역을 찾는 것이 불충분할 수 있기 때문이다.

가. 명령어 빈도(Instruction Frequencies)

명령어들에 대한 빈도 시험을 통해 악성코드를 탐지하기 위한 전제조건은 우선, 모든 바이러스는 어셈블리 언어로 작성되며 호스트 애플리케이션들은 상위 레벨 언어로 컴파일된다는 점이다. 그리고 이와 같은 전제조건을 기반으로 하여 어셈블리 프로그램들과 컴파일된 코드에서 발생하는 명령어 인식 방법에 관한 연구 및 컴파일 코드에서는 빈번하게 나타나고 어셈블리 언어에서 드물게 나타나는 명령어 형태에 관한 명령어 식별 연구가 진행되고 있다. 이러한 연구 결과물들을 통해 비정상적인 윈도우들을 찾기 위해 그 명령어 빈도가 계산되는 명령어 리스트를 만들어 내고 있다.

따라서 통계학적 분석 기법을 사용하여 PE 파일의 섹션이 감염 후 어셈블된 악성 코드는 컴파일된 코드 명령어의 비정상적인 생성(발생 빈도 높음)을 통해 발견할 수 있다.

나. 명령어 패턴(Instruction Patterns)

명령어 패턴 시험을 위한 기본 접근 방법은 명령어 빈도를 시험하기 위한 아이디어와 매우 유사하다. 제안된 방식에서는 MS 비주얼 C++ 컴파일러로부터 어셈블리 언어 출력시 명령어 패턴들에 대한 리스트를 만든다. 그러한 명령어 패턴 리스트로부터 산출한 명령어 패턴들의 빈도는 그와 같은 패턴들의 비정상적인 출현을 통해 악성 어셈블리 코드에 감염되었다는 것을 발견할 수 있다.

다. 레지스터 오프셋을 통한 메모리 접근(Memory Access via Register Offsets)

악성코드를 탐지하기 위한 또 다른 접근방법은 일반적인 애플리케이션들과 악성 코드는 각각 서로 다른 어떤 레지스터들을 사용할 것이라는 것이다. 특히, 기본적인 포인터 레지스터 EBP는 공통적으로 스택에 있는 로컬 변수들에 접근하기 위한 참조 포인터로서 일반적인 애플리케니션들에 의해 사용된다 그러나 악성 프로그램들은 그들이 메모리의 어디에 있는지, 그들이 어한 기능을 수행하기 위해 공통적으로 필요한 정보들은 무엇인지 그리고 알려지지 않은 악성코드 실행을 통한 코드 확산을 조절하기 위해 참조 포인트를 사용한다. 이와 같이 PEAT는 레지스터를 통해 메모리에 액세스하였을 때 사용되는 레지스터 오프셋 값들에 대한 통계학적인 분석을 사용하고 있다.

라. 점프와 콜간 거리(Jump and Call Distances)

애플리케이션에 포함된 함수들의 순서를 파악함으로써 악성코드에 감염된 애프리케이션들을 탐지할 수 있다. 함수들간의 제어권은 CALL과 RET 명령어들을 통해 넘겨지게 되는데, 점프 명령어들은 단일 함수 내에서 이러한 control flow를 바꾸게 된다. 따라서 일반적인 애플리케이션일 경우 점프 명령어 사이에 이동한 거리는 상대적으로 작으며 call 명령어들 사이에 이동한 거리는 상대적으로 크다. 그러나 악성 애플리케이션들은 비정상적으로 call과 jump 시 이동거리가 크다. PEAT는 이러한 점을 사용하여 악성코드를 식별한다.

마. 바이트 타입 확률(Byte-Type Probabilities)

악성코드를 탐지하기 위한 방법에 있어서 PEAT가 사용하는 마지막 접근방법은 ASCII 데이터, 패딩 또는 코드를 구성하는 각 항들에 대한 발생 확률을 이용하는 것이다. 악성코드를 탐지하기 위해 사용되는 다른 방법들과 함께 이러한 byte-type 정보는 악성행위를 판단하기 위해 사용되는 다른 접근 방법들과 함께 사용될 수 있다.

5. 결론

제안된 방식은 알려지지 않은 악성코드를 탐지해 내기 위해 PE 파일 형태를 가지는 파일들에 대해 명령어 구조, 사용 빈도 등을 분석하여 악성 여부를 판별하고 있다. 이와 같은 방식은 기존에 사용하였던 패턴 매칭을 이용한 방식이나 행위 추적 및 면역 체계와 다른 접근 방법을 사용하고 있다.

그러나 기존 상용 방식 및 연구 되고 있는 알려지지 않은 악성코드 탐지 방식에서와 마찬가지로 행위 및 명령어 구조 등에 대한 사전 분석이 면밀히 이루어지지 않는다면 악성 행위 발생시 이를 탐지해 내지 못하거나 혹은 정상 행위도 악성으로 판별할 수 있는 긍정적 오류의 가능성이 그대로 존재한다.

다만 기존 방식과 다르게 코드 수행이 이루어지고 난 이후 탐지하는 방식이 아닌 코드 실행 직전에 해당 명령어 분석을 수행함으로써 사용자 컴퓨터에서의 코드 실행 안전성을 제공한다는 장점이 있다.

[참고문헌]

[1] Michael Weber, Matthew Schmid, Michael Schatz and David Geyer, "A Toolkit for Detecting and Analyzing Malicioius Software", ACSAC'02, IEEE, 2003

[2] Matt Pietrek, "Peering Indside the PE:A Tour of the Win32 Portable Executable File Format", MSDN Library

[3] 오형근, "악성코드 대응 기술 동향", 공군정보통신 통권 제7호, 공군본부, 2003

[4] 오형근, 배병철, 김은영, 박중길, "실행시간 악성실행코드 탐지 시스템 설계", 한국정보과학회 춘계학술발표논문집(A) 제30권 제1호, 2003

[5] symantec 사, http://www.symantec.com