제30회 한국정보처리학회 추계학술발표대회 논문집 제15권 제2호 (2008. 11)

악성 코드의 실행 압축 해제 기술

정구현, 추의진, 이주석, 이희조
고려대학교 컴퓨터·전파 통신공학과
e-mail:ghjeong@korea.ac.kr

Unpacking Malwares

Guhyeon Jeong, Euijin Choo, Joosuk Lee, Heejo Lee
Div. of Computer & Communication Engineering, Korea University

요 약

최근 많은 악성 코드 제작자들이 실행 압축 기술을 악용하고 있다. 악성코드 제작자들은 자신의 악성 코드가 탐지되는 것을 어렵게 만들기 위해 기존의 알려진 실행 압축 알고리즘보다는 새로운 실행 압축 알고리즘을 개발하여 사용하고 암호화 기법 등을 이용하여 분석하기 어렵게 만들고 있다. 본 논문에서는 기존의 실행 압축 해제 기술에 대한 연구들을 분석하여 기존 기술의 한계점을 극복하기 위한 방안과 앞으로의 연구 방향을 제시한다.

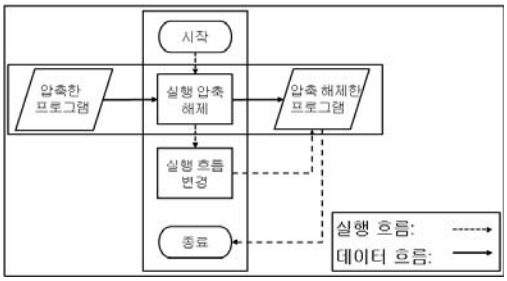
1. 서론

실행 압축 기술은 저장 공간을 효율적으로 사용하기 위해서 실행파일을 압축하여 저장하는 기술이다. 저장 공간의 제약이 많았던 과거에 이를 극복하기 위해 사용했던 실행 압축 기술이 최근 많은 악성 코드 제작자들에 의해 악용되고 있다. 악성코드 제작자들은 자신의 악성 코드가 탐지되는 것을 어렵게 만들기 위해 실행 압축 알고리즘과 암호화 기법 등을 사용하여 악성 코드를 분석하기 어렵게 만들고 있다. 예를 들어, Sasser, AgoBot, Nimda 등의 변종 악성 코드들이 실제로 실행압축을 사용하고 있다.[7] UPX[8], yoda Crypter[9] 등의 실행 압축 도구들은 인터넷을 통해서 쉽게 구할 수 있다. 많은 사람들이 전문적 지식이 없이도 쉽게 사용할 수 있어 그 위험성이 더 크다. 이에 따라 실행 압축 해제 기술에 대한 연구의 필요성이 점차 커지고 있다. 본 논문에서는 기존의 실행 압축 해제 기술 연구들의 조사를 통하여 오리지널 엔트리 포인트의 탐색과 해제에 걸리는 시간 그리고 해제 종료 시점의 결정에 대한 추가적인 연구가 보다 높은 성능의 압축 해제 기술을 만들 수 있을 것이라는 사실을 보인다. 또한 기존의 기술들을 체계적으로 정리하여 연구자들에게 보다 뚜렷한 연구 방향을 제시할 것이다.

2. 악성 코드 실행 압축 기술

악성코드 제작자들은 코드를 분석하기 어렵도록 만들기 위해서 악성 코드 실행 압축 기술을 사용한다. 이 절에서는 실행 압축 기술에 대해서 자세히 기술한다.

실행 압축의 기본 원리는 다음과 같다. 우선 실제로 작동할 프로그램을 압축하여 데이터 형식으로 저장하고 있는 프로그램을 작성한다. 그 후, 그 압축을 해제하는 루틴을 프로그램에 함께 포함하여 실행 압축 프로그램을 완성한다. 이후에 실제로 사용자가 이 실행 압축 프로그램을 실행하면 아래 그림1. 과 같이 작동한다. 우선 실행 압축 해제 루틴이 이전에 압축해 두었던 원래의 프로그램의 압축을 해제한다. 그 결과로 압축이 해제된 원래의 프로그램이 생성된다. 그 이후에 프로그램의 실행 흐름을 원래 프로그램의 시작 지점으로 바꾼다.



(그림 1) 실행 압축 프로그램의 실행

이와 같은 기본 원리를 이용하여 많은 악성 코드 제작자들이 악성 코드의 분석을 어렵게 하기 위해서 파일 보호 프로그램을 사용한다. yoda Crypter, PE Crypter 등이 자주 사용되고 있다. 이러한 도구들이 사용하는 파일 보호 기법에는 XOR연산을 이용해 코드의 해석을 불가능하게 만드는 암호화 기술, 실제 같은 일을 하는 다른 코드로 치환하여 패턴을 사용해 어셈블리어 수준이 아닌 좀 더 높은 수준으로 분석해 주는 툴을 무력화시키는 코드치환 기술, 디버거로 분석할 경우에 프로그램 자체를 비정상 종료

·본 연구는 지식경제부 및 정보통신연구진흥원의 대학IT연구센터 지원사업의 연구결과로 수행되었음" (IITA-2008-C1090-0801-0016)

하는 디버거 무력화 기술, 함수 호출을 몇 단계 거치도록 하여 분석을 어렵게 하는 API Redirection 기술 등이 있다.[1]

위와 같이 실행 압축 기술을 악용할 경우 악성 코드를 분석하는 시간이 더욱 오래 걸리기 때문에 그 시간에 걸쳐 피해가 계속 누적될 수 있다. 그러므로 실행 압축 해제 기술의 연구는 필수적이다.

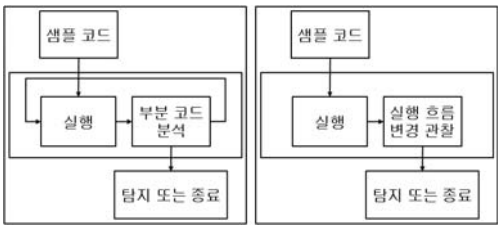
3. 악성 코드 탐지를 위한 실행 압축 해제 기술

악성 코드 제작자들이 실행 압축 기술을 악용함에 따라서 실행 압축 해제 기술에 대한 연구도 활발하고 다양하게 이루어지고 있다. 이 절에서는 실행 압축 해제의 원리와 기존의 실행 압축 해제 기술들에 대해서 설명한다.

실행 압축 해제 기술의 가장 큰 핵심은 오리지널 엔트리 포인트(Original Entry Point)를 찾는 것이다. 오리지널 엔트리 포인트는 실행 압축을 한 원래의 프로그램의 진입점을 말한다. 실행 압축 프로그램을 실행하면 실행 압축 해제를 위한 루틴을 실행한 후에 결국은 원래의 프로그램의 진입점으로 실행 흐름이 바뀌어야 한다. 그러므로 실행 흐름이 오리지널 엔트리 포인트로 바뀌는 순간 그 이후에 실행되는 코드가 실행 압축을 하기 이전의 원래의 프로그램이다.

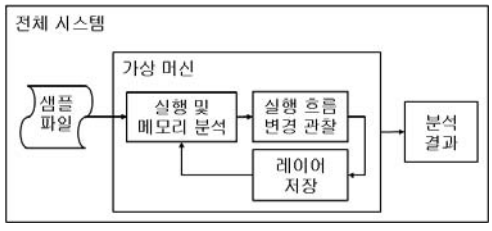
초기에는 실행 압축 해제를 위해서 기존 도구에 플러그인을 덧붙여 사용하여 사용자가 직접 조작하는 형태의 방식을 이용하였다. 대표적으로 Ollydbg[2] 플러그인들이 있다. Ollydbg 플러그인들은 실행 압축 해제에 매우 유용한 도구이다. 예를 들어, OllyScript는 Ollydbg를 사용함에 있어 어셈블리 언어와 비슷한 형식의 스크립트를 이용하여 자동화할 수 있기 때문에, 오리지널 엔트리 포인트를 반복적으로 찾아야 하는 실행 압축 해제 기술에 유용하다.

위와 같은 방식의 기술들을 이용하여 실행 압축을 해제하기 위해서는 많은 시간과 비용을 투자해야하기 때문에 좀 더 자동화된 실행 압축 해제 기술에 대한 연구가 진행되기 시작했다. PolyUnpack[3]의 경우 실행 압축 프로그램은 결국 원래의 프로그램을 실행한다는 점에 착안하여 실행 압축한 코드를 연속적으로 일부분씩 실행하여 실행한 코드를 분석하는 방법을 사용하였다. 또 다른 기술로는 OmniUnpack[4]이 있다. 이 기술은 실행 압축된 코드를 해제하는데 다음의 원리를 사용한다. 메모리에 기록되는 모든 내용들과 명령어 포인터를 추적하여 만약 그 명령어 포인터가 이전에 기록된 메모리의 주소를 가리킨다면 실행 압축 해제 과정이 프로그램에서 일어난 것이라고 판단할 수 있다. 이런 방식의 기술들을 이용하여 기존에 비해 더 효율적인 실행 압축 해제를 할 수 있게 되었다.



(그림 2) PolyUnpack(좌측)과 OmniUnpack(우측)의 구조

최근에는 가상 머신을 이용한 실행 압축 해제 기술에 대한 연구도 이루어지고 있다. 가상 머신을 이용할 경우 그렇지 않은 환경에 비해서 속도는 다소 느릴 수 있지만 악성 코드로부터 시스템이 감염되는 것을 막을 수 있고 프로그램이 작동하는 동안 전체 시스템을 관찰할 수 있다는 장점이 있기 때문이다. 이와 관련된 연구로는 Renovo[5]가 있다. Renovo는 기본적으로 실행 흐름이 프로그램의 실행 후 작성된 메모리 영역으로 점프하는 지 아닌지를 모니터 한다. 메모리에 프로그램이 로드되면 메모리 맵을 만들고 맵을 “clean” 상태로 생성한다. 프로그램이 메모리 쓰기 작업을 수행을 할 때마다 그 메모리 지역을 “dirty”로 표시한다. 또한 숨겨진 레이어가 있는 지 없는 지를 판별하고 숨겨진 원래의 코드와 오리지널 엔트리 포인트를 추출한다. 여러 겹으로 숨겨진 레이어를 색출하기 위해서 대비하여 현재의 레이어로부터 추출된 정보들에 대하여 덤핑을 수행하며 그 이후에 메모리 맵을 다시 “clean”으로 초기화 시킨다. 즉, Renovo 는 여러 번 실행 압축이 수행된 경우에도 중간 단계의 코드와 데이터를 모두 추적할 수 있다. 이와 같이 가상 머신을 이용한 실행 압축 해제 기술로 인해 전체 시스템 규모의 관찰이 가능해졌다.



(그림 3) Renovo의 구조

지금까지 살펴본 기술들은 동적인 분석 방법을 사용하고 있다. 이러한 기술들의 장점은 실제로 실행하는 코드를 얻을 수 있다는 점이다. 하지만 악성 코드 제작자가 특정 조건에서만 악의적 행위를 하도록 작성해두었다면 실제로

실행한 흐름을 바탕으로 얻은 코드에서는 악성 코드의 흔적을 발견하기 어렵다. 이러한 점을 보완하기 위해서 Andreas Moser 등은 그들의 논문[6]에서 프로그램의 실행흐름을 인위적으로 조작하여 발생 가능한 여러 가지 실행 흐름을 탐색하고 이를 통한 악성 코드 분석을 시도하였다. 그 결과, 알려지지 않은 프로그램에 대해서 기존의 동적 분석 기반의 연구들보다 더 많은 행위 정보를 얻을 수 있고 실제로 악성 코드를 탐지할 수 있음을 보였다.

4. 실행 압축 해제 기술의 연구 방향

3절에서 소개한 것과 같이 지금까지 실행 압축 해제 기술에 대한 연구들이 진행되어 왔다. 하지만 그와 동시에 악성 코드 제작자들의 실행 압축 기법들도 함께 발전하고 있다. 특히, 최근의 악성 코드들은 일부의 코드를 바꾸면서 스스로 변화하는 형태의 다형성 바이러스 형태를 취하고 있어 알려진 실행 압축 알고리즘에 의존한 압축 해제 방식을 따르는 기존의 실행 압축 자동화 도구들을 속이기 쉽다. 따라서 알려지지 않은 알고리즘에 대응할 수 있는 실행 압축해제 기술에 대한 연구가 필요하다. 효과적인 실행 압축 해제를 하기 위해서 아래와 같은 사항들을 고려해야 한다. 첫째, 정확한 오리지널 엔트리 포인트를 찾아야 한다. 오리지널 엔트리 포인트를 찾으면 원래의 프로그램을 복원할 수 있기 때문이다. 둘째로, 오리지널 엔트리 포인트뿐만 아니라 실제 코드가 어디서 종료되는지 판단할 수 있어야 한다. 실제로 실행 압축 해제에 걸리는 시간은 예측할 수 없는 문제이기 때문이다. 마지막으로 속도의 측면에서도 개선이 필요하다. 가상머신을 이용하여 악성 코드를 분석하는 방법은 클라이언트 사용자 시스템과는 분리된 독립된 환경에서 실행함으로써 대상코드가 악성행위를 수행하여 시스템에 악영향을 끼치는 것을 방지할 수 있다. 또한, 시스템에 직접 실행하며 탐지를 수행하는 도중에 시스템에 악성행위가 실시되어 시스템이 동작하지 못하는 상태가 될 위험 없이 가상 머신에서 실행함으로써 간단하게 새로운 이미지를 올리는 것으로 대체할 수 있다. 가상머신을 빠르게 초기화하며 실행 압축 해제를 수행할 수 있으므로 속도 면에서도 큰 장점을 갖는다. 기존의 가상 머신 기반의 실행 압축 해제 도구들은 오리지널 엔트리 포인트와 종료 시점을 판단 할 수 없어 실행 압축 해제를 하는 데에 있어서 성능의 한계를 보였다. 이러한 점들을 보완하여 두 시점(오리지널 엔트리 포인트와 종료 시점)을 정확히 판단하고 해당 시간동안의 메모리 상태를 얻어낼 수 있다면 알려지지 않은 악성 코드들에 대해서도 범용적으로 사용할 수 있는 실행 압축 해제 자동화 도구를 구현할 수 있을 것이다.

5. 결론

많은 악성 코드 제작자들이 실행 압축 기술을 악용하여 악성 코드를 분석하기 어렵게 만들고 있다. 이에 따라 실행 압축 해제 기술에 대한 많은 연구가 진행되고 있지만 성능의 한계가 있다. 본 논문에서는 오리지널 엔트리 포인트의 탐색과 해제에 걸리는 시간 그리고 해제 종료 시점의 결정에 대한 추가적인 연구가 필요함을 보였다. 또한 기존의 기술들에 대한 체계적인 분석을 바탕으로 기존 연구들의 한계를 극복할 수 있는 연구 방향을 제시하였다.

참고문헌

[1] 국가사이버안전센터(2004). “악성코드 분석을 위한 실행압축 해제 기법”. http://www.ncsc.go.kr/.

[2] Ollydbg. http://www.ollydbg.de/.

[3] P. Royal, M. Halpin, D. Dagon, R. Edmonds, and W.Lee. "PolyUnpack: Automating the Hidden-Code Extraction of Unpack-Executing Malware". In Proceedings of 2006 annual Computer Security Applications Conference(ACSAC), pages 289-300, 2006. IEEE Computer Society.

[4] Martignoni. L, Christodorecu. M, and Jha. S. "Omni-Unpack: Fast, Generic, and Safe Unpacking of Malware". In Proceedings of 2007 annual Computer Security Applications Conference(ACSAC), pages 431-441, 2007. IEEE Computer Society.

[5] M. G. Kang, P. Poosankam, and H. Yin. "Renovo: A hidden code extractor for packed executables". In Proceedings of the 5th ACM Workshop on Recurring Malcode(WORM'07), Oct. 2007.

[6] A. Moser, C. Kruegel, and E. Kirda. "Exploring multiple execution paths for malware analysis". In Proceedings of the 2007 IEEE Symposium on Security and Privacy(Oakland'07), May 2007.

[7] 안철수연구소. http://home.ahnlab.com/.

[8] UPX. http://upx.sourceforge.net/.

[9] yoda Crpyter. http://y0da.cjb.net/.