PE Format 조작을 통한 소프트웨어 크랙 방지 기술

김태형* · 장종욱*

*동의대학교

How to Prevent Software crack for Control PE

Tae-hyoung Kim* · Jong-uk Jang**

*Dong Eui Universit

E-mail : fingersnoop@gmail.com

요 약

과거에는 소프트웨어 보안이 크게 중요하지 않게 생각해왔다. 그러나 소프트웨어를 공격하는 기술은 시대를 넘어 빠르게 성장하고 있으며 이로 인한 소프트웨어 산업의 성장은 감소하고 저작권자의 이익은 점점 감소하고 있다. 그래서 본 연구에서는 PE 포맷 조작을 통해 소프트웨어 크랙을 방지하는 것을 제안한다. 보통 해커는 프로그램을 정적으로 먼저 분석을 해서 1차적인 정보를 얻는데 PE 포맷의 약간의 조작만으로 정적 분석을 방해할 수 있다. 그리고 PE 포맷 조작을 통해 해당 프로그램에 여러 가지 보안 코드가 삽입 가능하며 이를 통해 해커들의 디버거를 이용한 접근이나 동적 분석을 방해 할 수 있다.

ABSTRACT

In the past, People thought that software security was not important. but Skills of attacking software has growing up in fast, software crack fall down software industry growth and profit of copyright holder was declined. So I propose software crack prevention for changing PE Format. Hackers can analyze program in static. As we change the PE format, we can prevent static analysis. As I insert anti - debugging code the exe file, the program is protected from dynamic analysis.

키워드

크랙,소프트웨어 크랙, 소프트웨어 공격, 악성 코드

Ⅰ. 서 론

PE 포맷은 Windows Portable Executable의 약자이다. Windows NT 시리즈에서는 파일을 실행하기 위해선 Windows의 정해진 규칙을 따라야 한다. 그 규칙은 몇 가지의 단계가 있다. 특정한 파일을 실행하면 PE 로더가 파일을 메모리에 일정한 규칙에 따라 매핑을 한다. 과거에는 소프트웨어 보안이 크게 중요하지 않게 생각해왔다. 그러나 소프트웨어를 공격하는 기술은 시대를 넘어 빠르게 성장하고 있으며 이로 인한 소프트웨어 산업의 성장은 감소하고 저작권자의 이익은 점점 감소하고 있다. 그래서 본 연구에서는 PE 포맷 조작을 통해 소프트웨어 크랙을 방지하는 것을 제안한다. 보통 해커는 프로그램을 정적으로 먼저 분석을 해서 1차적인 정보를 얻는데 PE 포맷의 약간의 조작만으로 정적 분석을 방해할 수 있다. 그리고 PE 포맷 조작을 통해 해당 프로그램에 여러 가지 보안 코드가 삽입 가능하며 이를 통해 해커들의 디버거를 이용한 접근이나 동적 분석을 방해 할 수 있다. 그리고 API 리다이렉션이나 코드 가상화를 통해 해커로부터의 소프트웨어의 공격을 막을 수 있다.

II. 본 론

PE 포맷은 다음과 같다. DOS HEADER와 DOS STUB 그리고 NT HEADER 그리고 SECTION HEADER 및 SECTION으로 구성되어 있다.

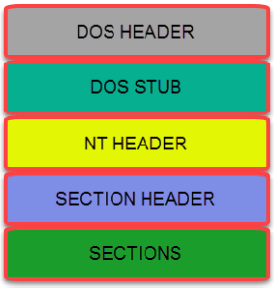

그림 1. PE structure

2.1 PE 구조

2.1.1) DOS_HEADER

도스 헤더는 도스를 위해 만들어졌다. e_magic 필드 그리고 e_lfanew 필드가 가장 중요한 필드이다. e_magic 필드는 "MZ"의 값이 들어가 있으며 이 값은 Mark Zbikowski의 약자며, MZ는 윈도우 실행파일이라는 의미이다. 그리고 e_lfanew 필드는 nt_header로 넘어가는 첫 번째 오프셋의 점프 주소를 의미한다.

2.1.2) DOS_STUB

DOS STUB은 지금은 사용되지 않는 필드이며 더미 코드이다.

2.1.3) NT_HEADER

NT_HEADER는 signature와 file header 그리고 optional header로 구성되어 있다. signature는 "PE"를 의미하며 PE파일이라는 뜻이다. file header의 number of sections라는 필드는 섹션의 추가와 삭제의 가장 중요한 필드이다. Optional header의 address of entry point라는 필드와 Image Base 필드 그리고 Section Alignment 필드와 File Alignment 필드와 Size of Image필드 그리고 Size of Header 필드는 모두 PE를 조작하는데 가장 중요한 필드들이다. 윈도우에서 특정 프로그램을 실행할 시 메모리에서의 Image Base값 주소는 보통 0x00400000h로 설정되어 있는데 이 값은 동적으로 바뀐다. Section Alignment는 섹션에서의 가장 최소 단위를 뜻하며, File Alignment는 파일상태에서의 섹션의 가장 최소 단위를 의미한다. Size of Header 필드는 헤더의 총 크기를 의미한다.

2.1.4) SECTION HEADER

섹션 헤더는 VirtualSize와 VirtualAddress, Size of rawdata 그리고 PointerToRawData필드로 이루어져 있으며 이들 모두 섹션헤더에서 굉장히 중요한 필드들이다. Virtual Size는 전체 섹션 크기에서 유효한 사이즈를 의미하며 VirtualAddress는 RVA(상대적 가상 주소)를 의미하며 이것은 메모리에서 섹션의 시작 주소를 의미한다. Size of rawdata필드는 파일인 상태에서의 유효한 데이터들의 수를 의미하고 PointerToRawData는 파일에서의 섹션 시작 주소를 의미한다.

2.2 패킹 메커니즘

프로그램이 시작하기 전에 우리는 코드를 만들고 컴파일하고 링킹을 하는 과정을 거친다. 아래 그림은 기계어로 바뀐 최종 코드의 결과물이다.

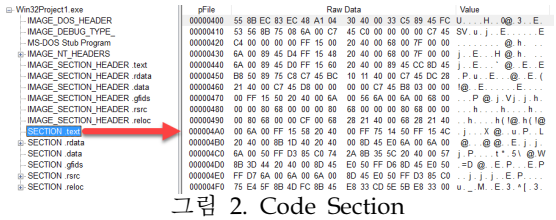

그림 2. Code Section

그러나 해커들은 디버거를 통해 기계어 상태의 코드를 동적으로 분석하여 Original Code를 얻어낼 수 있다. 그래서 본 논문에서는 해커들의 이러한 공격을 효과적으로 방지하기 위해 저 코드 섹션을 압축하고 암호화를 해야 한다. 그리고 우리는 해당 안티디버깅 코드를 삽입하여 해커들로부터 디버거의 접근을 막을 수 있다.

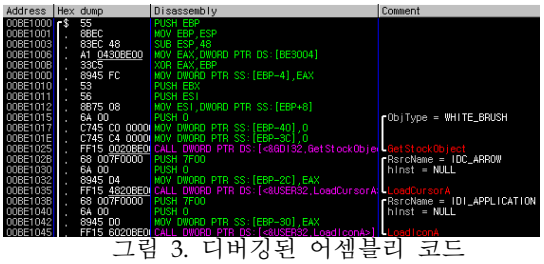

그림 3. 디버깅된 어셈블리 코드

우리는 이러한 패킹 과정을 통해 프로세스 메모리에 접근하고, 동적 분석을 막을 수 있다.

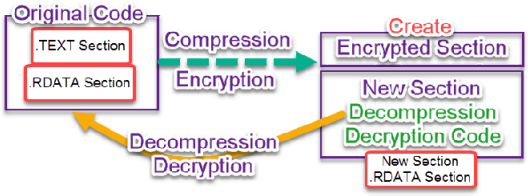

그림 4. 패킹 메커니즘

패킹을 하기 전에 첫 번째로 우리는 섹션을 하나 새로 추가를 해야 하고 number of sections의 필드의 값을 늘어난 섹션의 수만큼 변경해 줘야 한다. 그리고 우린 .rsrc섹션을 제거를 해줘야 한다. 해당 섹션은 dll들이 메모리에 매핑 될 시 동일한 주소로 메핑 될 때 발생하는 충돌을 방지하기 위해 Relocation할 주소 테이블이 존재하는데 이 섹션을 제거해 줌으로써, 정적인 Base 주소를 얻어 패킹을 하는데 편의를 줄 수 있다. 그리고

우리는 섹션이 추가된 크기만큼 Size of Image를 Section Alignment의 배수로 증가시켜 줘야한다. 두 번째로 추가된 섹션에 대한 Section Header를 작성해 줘야한다. 그리고 추가된 Section Header를 넣기 위한 추가 공간을 만들어야 하고 추가된 공간의 File Alignment의 배수만큼 각 Section Header의 PointerToRawData의 값들을 증가시켜 줘야 한다. 세 번째로 새롭게 추가된 섹션을 위한 새로운 .rdata 섹션을 만들어 줘야하며 이 방법 또한 위와 같은 방법으로 추가해 줘야 한다. 정리하면 기존의 압축 프로그램은 압축과 해제에 있어 이를 압축하고 해제하는 프로그램이 존재해야 하며 이 프로그램이 없을 때는 압축을 하거나 해제할 수 없다. 그러나 패킹을 하게 되면 압축된 상태에서 실행이 가능하며 이를 실행 압축이라고 한다. 압축되고 암호화된 프로그램을 실행하면 이를 해제하고 복호화를 하는 코드가 실행되고 이를 원래 프로그램으로 재 복구하여 프로그램이 실행되는 것이다. 이 때 재 복구하는 과정에서 해커의 공격을 막는 코드가 삽입되게 되며 해커의 정적 분석과 동적 분석을 막는 원리이다. 그림 4를 보면 프로그램의 원본 오리지널 기계어 코드 섹션과 .Rdata섹션을 압축 및 암호화하여 새로운 섹션에 저장하고, 이를 해제하고 복호화와 해커의 공격을 맞는 각종 안티 디버깅 코드를 삽입하는 섹션을 따로 만든다. 이 섹션을 위해 새로운 .Rdata섹션을 만들어 줘야하며 프로그램의 정상 작동을 위해 Address of Entry Point주소를 압축 해제와 복호화를 하는 섹션으로 돌리게 되면 최종적으로 패킹이 완료 된다. 프로그램 시작 전에서의 모습은 원본 코드에서 바로 실행되는 것이 아닌 압축 해제와 복호화를 시작하는 새로운 섹션에서 시작되며 압축 해제와 복호화를 전부 진행하여 원본 코드를 복구한 뒤 원본 코드로 시작되는 섹션으로 점프를 시켜주면 원본 코드가 실행되게 되고 이를 종료하면 다시 패킹된 상태로 돌아간다. 압축 해제와 복호화를 하는 섹션에서 해커의 공격을 막는 보안 코드가 삽입되게 된다.

III. 결론

결국 패킹을 한 후, 우리는 패킹된 프로그램을 시작하면 원본 코드는 압축 및 암호화되어 새로운 섹션에 숨겨져 있기 때문에 기존 원본 코드에 대한 모든 정보는 정적으로 분석이 불가능하게 되고, 숨겨져 있는 각종 보안 코드에 의해 동적 분석을 하는데 있어 해커들의 불편함을 감수하게 한다. 그러나 이 방법은 해커들에게도 또 다른 공격 방법을 제공할 수 있는 빌미를 줄 수 있다. 이 방법을 통해 프로그램에 악성코드를 삽입할 수 있으며 이러한 문제점들은 아직도 앞으로 해결해야 할 우리의 숙제이다.

ACKNOWLEDGMENT

이 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 지역신산업선도인력양성사업 성과임
(NRF-2016H1D5A1910985)