사례 연구: 보안 강화를 위한 다형성 도구(Polymorphic Tool) 개발

임을규 (imeg@hanyang.ac.kr)
한양대학교 정보통신대학
유무선 네트워크 보안 연구실

Contents

용어 설명

- PE(Portable Executable) 포맷
  - Windows 에서 사용되는 실행 파일 및 라이브러리의 구조
    - 예) exe, dll, ocx, scr 파일
  - PE 포맷의 실행 방법을 결정하는 Header가 존재
  - PE 파일에는 실행 파일이 실행되기 위한 역할을 담당하는 여러 개의 section이 존재

- Loader
  - BIOS의 bootstrap 과정과 유사
  - 원본 code의 압축 및 암호화를 해제하고 실행 시키는 역할

용어 설명

- 다형성(Polymorphism)
  - 패턴을 가지지 않는 성질
  - 다형성이 적용될 때 마다 새로운 패턴을 얻을 수 있음

- 실행 압축
  - 실행 파일의 .text section을 압축하는 기법
  - Loader에 의해 압축된 명령어들이 압축 해제되며 실행

- Instruction-set
  - Assembly language 수준의 명령어를 동일한 기능을 수행하는 다른 명령어 집합

Polymorphism of Internet Worms

- self-encryption/compression
  - limited number of decryption/decompression routines
- garbage-code insertion
  - a number of nop
- instruction-substitution
  - execess jump instructions provide a statistical clue

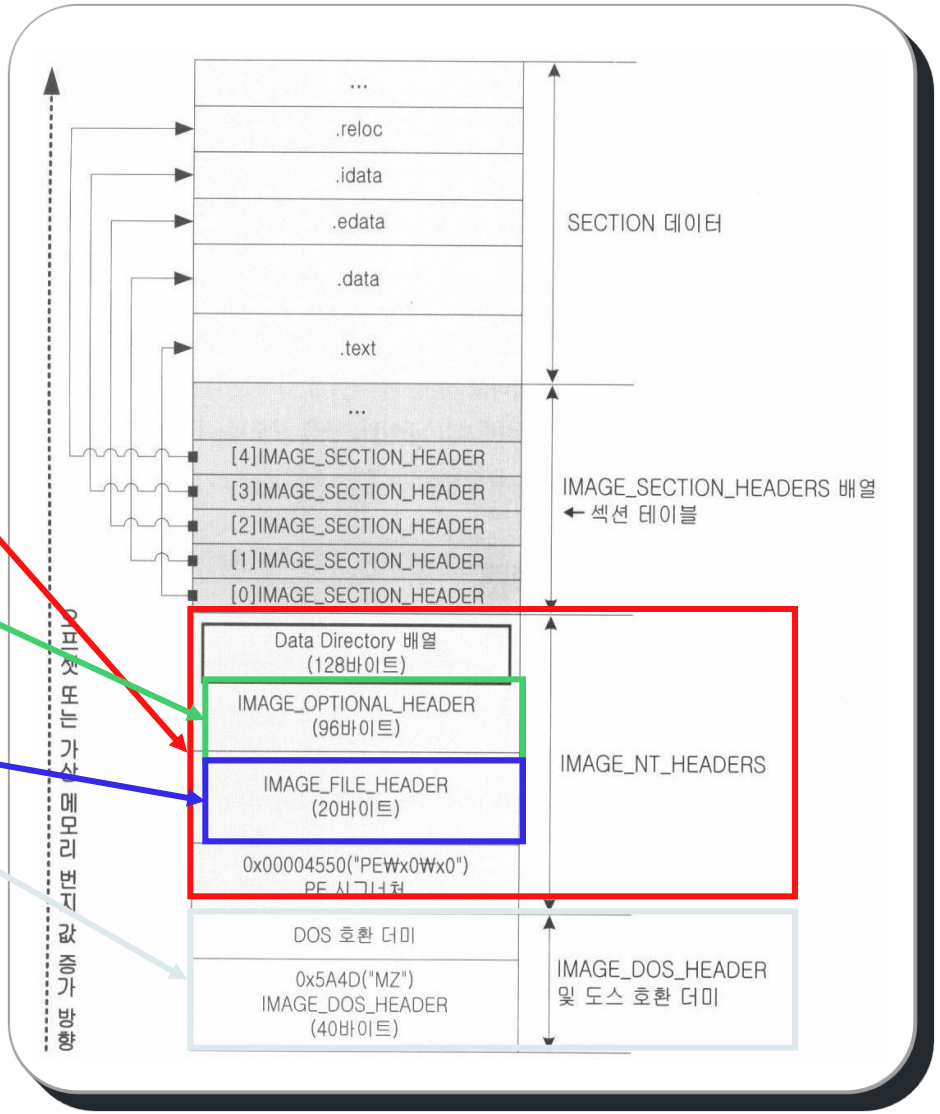
PE format의 구조

PE Format의 Header

PE 파일이 메모리에 로드될 때 필요한 정보 (MagicCode, Entry Point 등)

PE 파일의 정보 (CPU 타입, 생성시간, EXE/DLL 여부)

MS-DOS와의 호환성을 위한 DOS stub



PE : Portable Executable

PE format의 구조

- PE 파일 구조를 이루고 있는 주요 데이터 영역

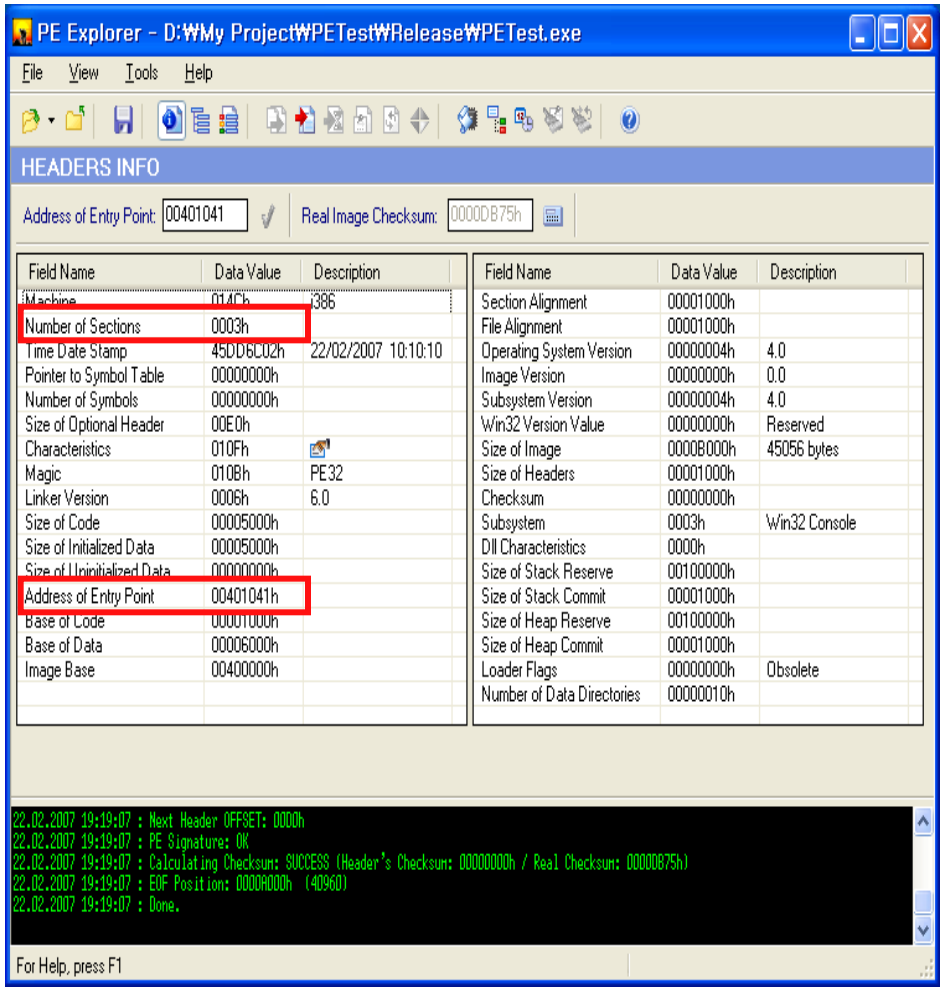
종류	이름	설 명
코드	.text	프로그램을 실행하기 위한 코드를 담고 있는 섹션이다. CPU 레지스터의 명령 포인터인 IP는 이 섹션 내에 존재하는 번지 값을 담게 된다.
데이터	.data	초기화된 전역 변수들을 담고 있는 읽고 쓰기 가능한 섹션이다.
	.rdata	읽기 전용 데이터 섹션으로서 문자열 표현이나 C++/COM 가상 함수 테이블 등이 .rdata에 배치되는 항목 중의 하나이다.
	.bss	초기화되지 않은 전역 변수들을 위한 섹션이다. 실제 PE 파일 내에서는 존재하지만 가상 주소 공간에 매핑 될 때에는 보통 .data 섹션에 병합되기 때문에 메모리상에서는 따로 존재하지 않는다.
임포트 API 정보	.idata	임포트할 DLL과 그 API들에 대한 정보를 담고 있는 섹션이다. 하지만 이 섹션은 .rdata에 병합하는 경우가 요즘 추세이다.
	.didat	지연 로딩(Delay-Loading) 임포트 데이터를 위한 섹션이다. 지연 로딩은 Windows 2000부터 지원되는 DLL 로딩의 한 방식으로 암시적인 방식과 명시적인 방식의 혼합이다. 보통 릴리즈 모드로 링크될 때 이 섹션 역시 다른 섹션에 병합된다.
익스포트 API 정보	.edata	익스포트할 API에 대한 정보를 담고 있는 섹션이다. VC++가 만들어내는 익스포트 리스트 파일인 exp 파일에서 이 섹션을 볼 수 있다.

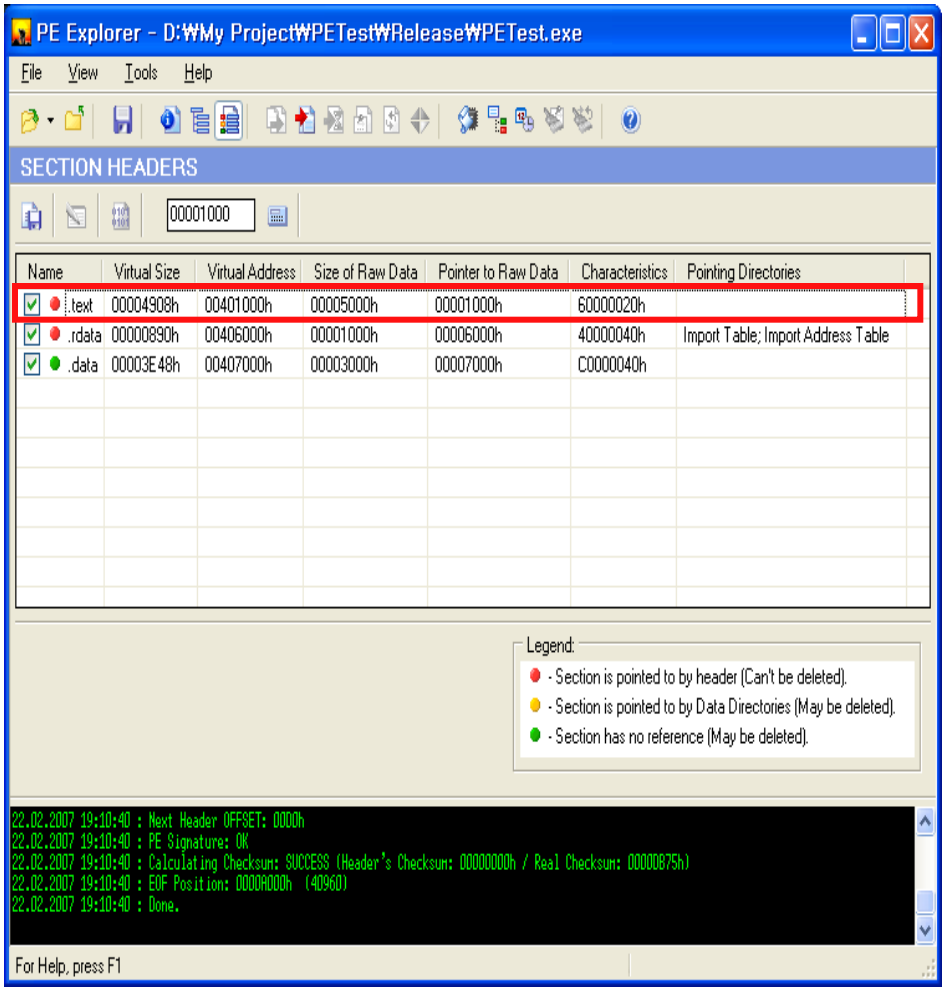
실행 압축

- 실행 압축의 원리
  - PE Format의 .text 를 압축
  - 원본 PE Format의 entry point 값을 변경
  - 압축된 파일이 실행될 때 entry point 값을 .text의 앞으로 변경

실행 압축

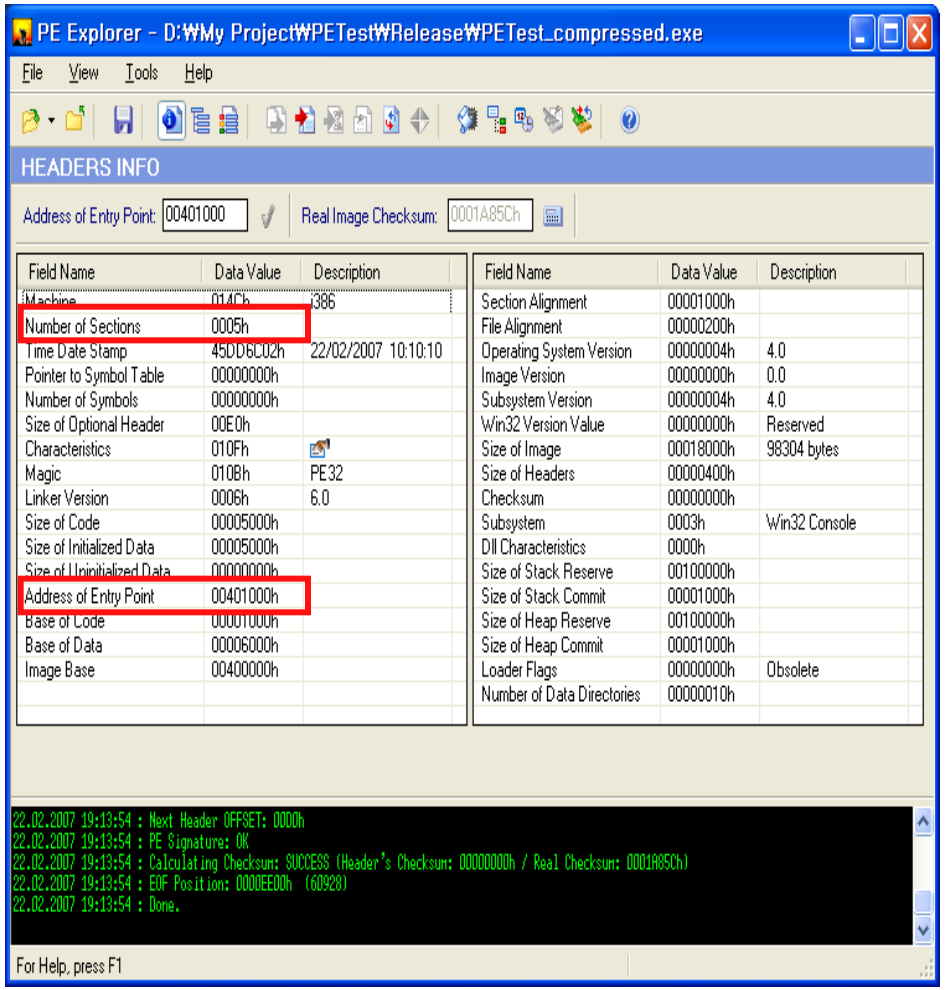
- 실행 압축 예 – ASPack 사용 전

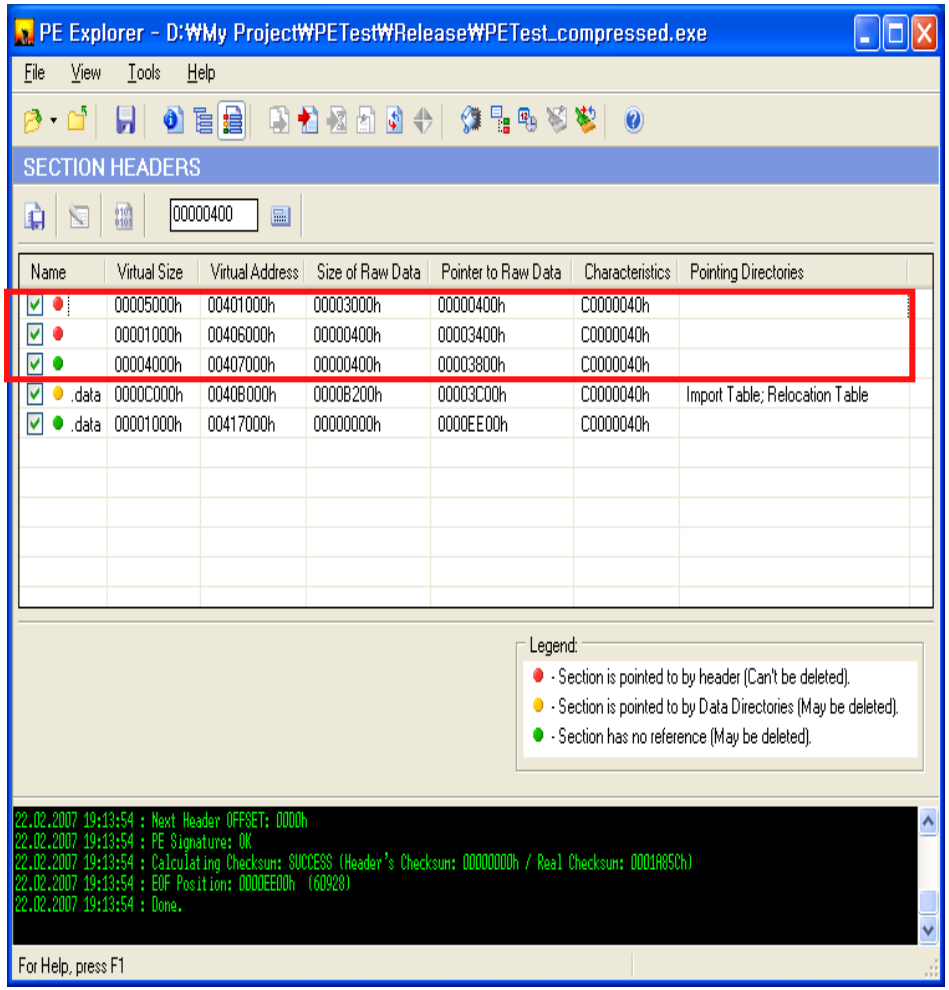




실행 압축

- 실행 압축 예 – ASPack 사용 후





실행 압축

- 실행 압축을 이용하는 악성코드
  - MyDoom, Netsky, Bagle, Agobot, Welchia, sasser, Sobig 변종 등

- 실행 압축 솔루션
  - ASPack
    - ASPACK Software, http://www.aspack.com
    - No runtime performance penalties
  - ASProtect
    - ASPACK Software, http://www.aspack.com
    - Anti-debugger protection 지원
  - UPX (The Ultimate Packer for eXecutables)
    - http://upx.sourceforge.net
    - Linux elf 파일 포맷 지원

실행 압축

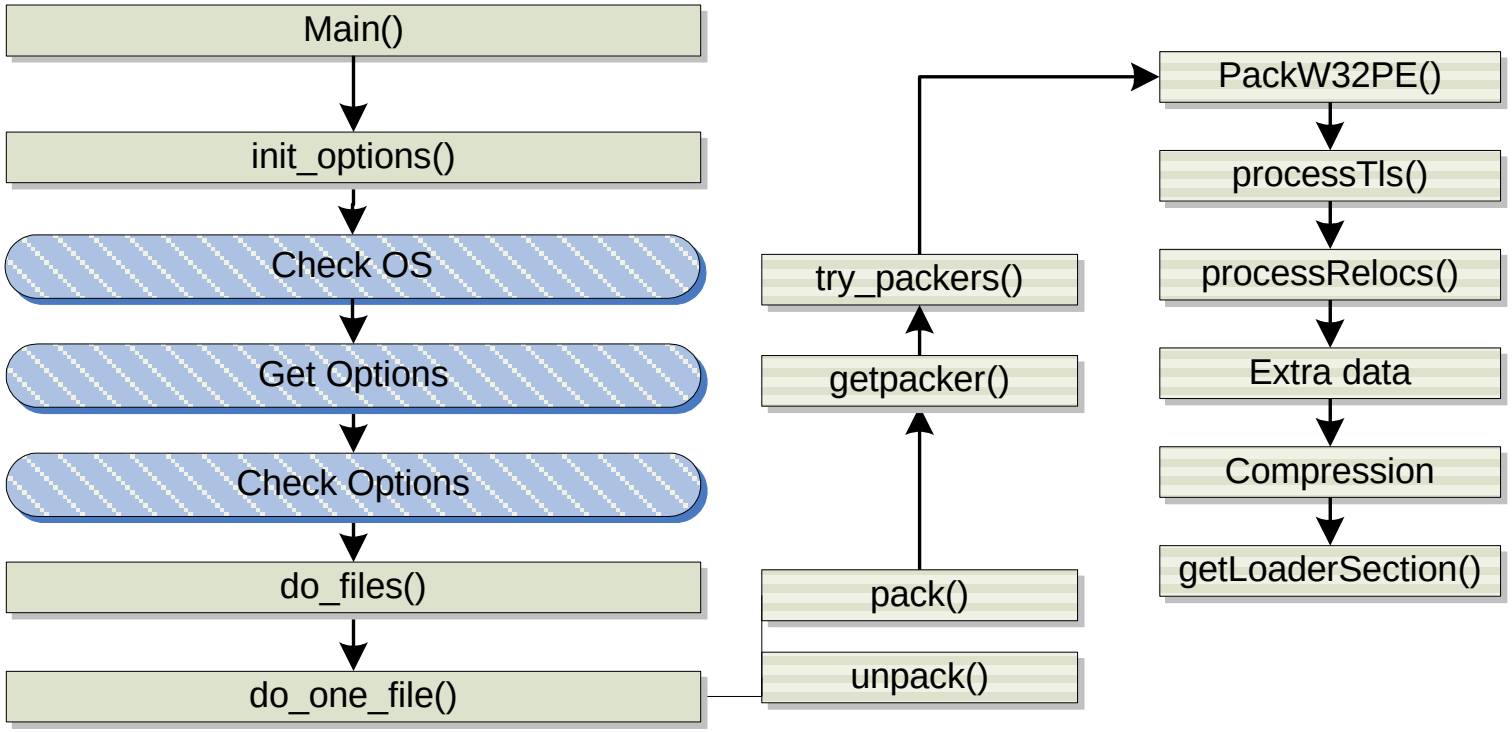
- 실행 압축 솔루션 (Cont'd)
  - 기타 솔루션
    - ExeStealth
    - FSG (Fast Small Good)
    - PECompact2
    - tElock (tHe EGOiSTE lock)

실행 압축

- PE Protecting 솔루션
  - VMProtector (Virtual Machine Protector)
    - PolyTech , http://www.vmprotect.ru/vmprotect.php
    - 프로그램의 코드를 직접 수정
    - 보호할 소스 코드의 부분을 Virtual Machine에서 실행되는 프로그램으로 변환
  - Yoda's Protector
    - http://www.codeproject.com/cpp/peprotector1.asp
    - Anti-Debugger 지원
      - IsDebuggerPresent() API를 이용하여 anti-debug 지원
    - Anti-SoftICE 지원
      - \ \ .\ NTICE와 \ \ .\ SICE 탐지기능

UPX 소스 코드 분석

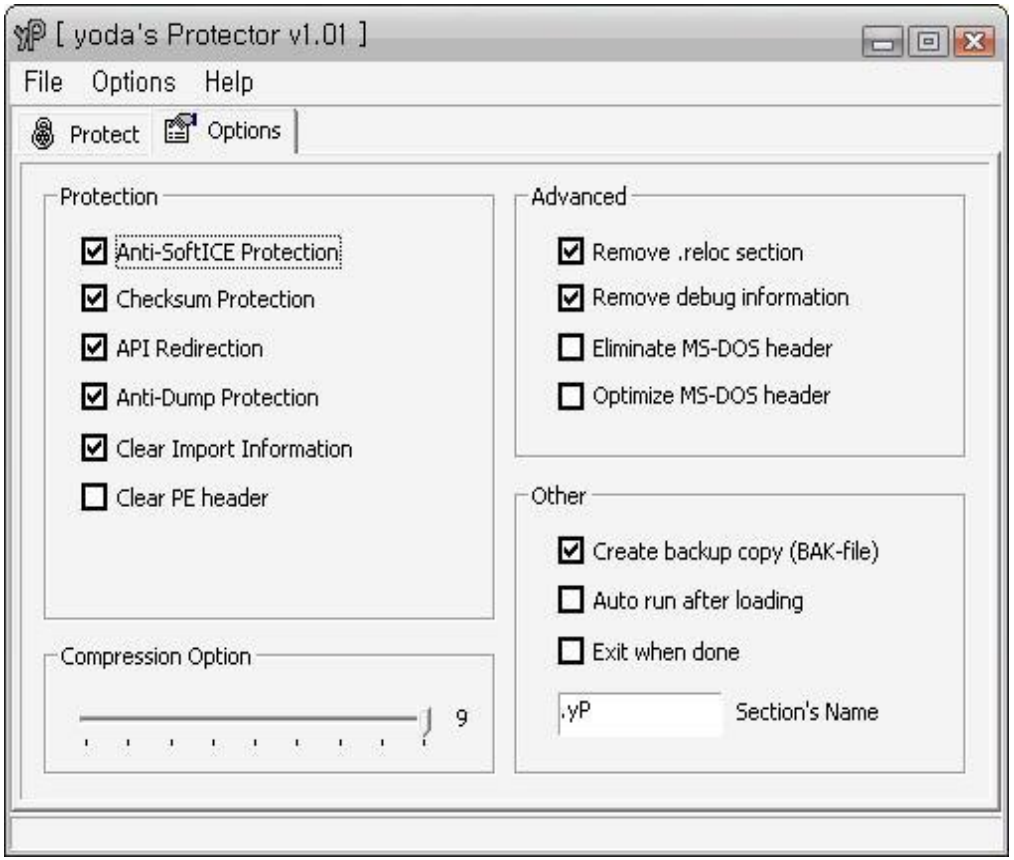
- 실행 압축 솔루션
- 소스가 공개됨



TLS : thread local storage

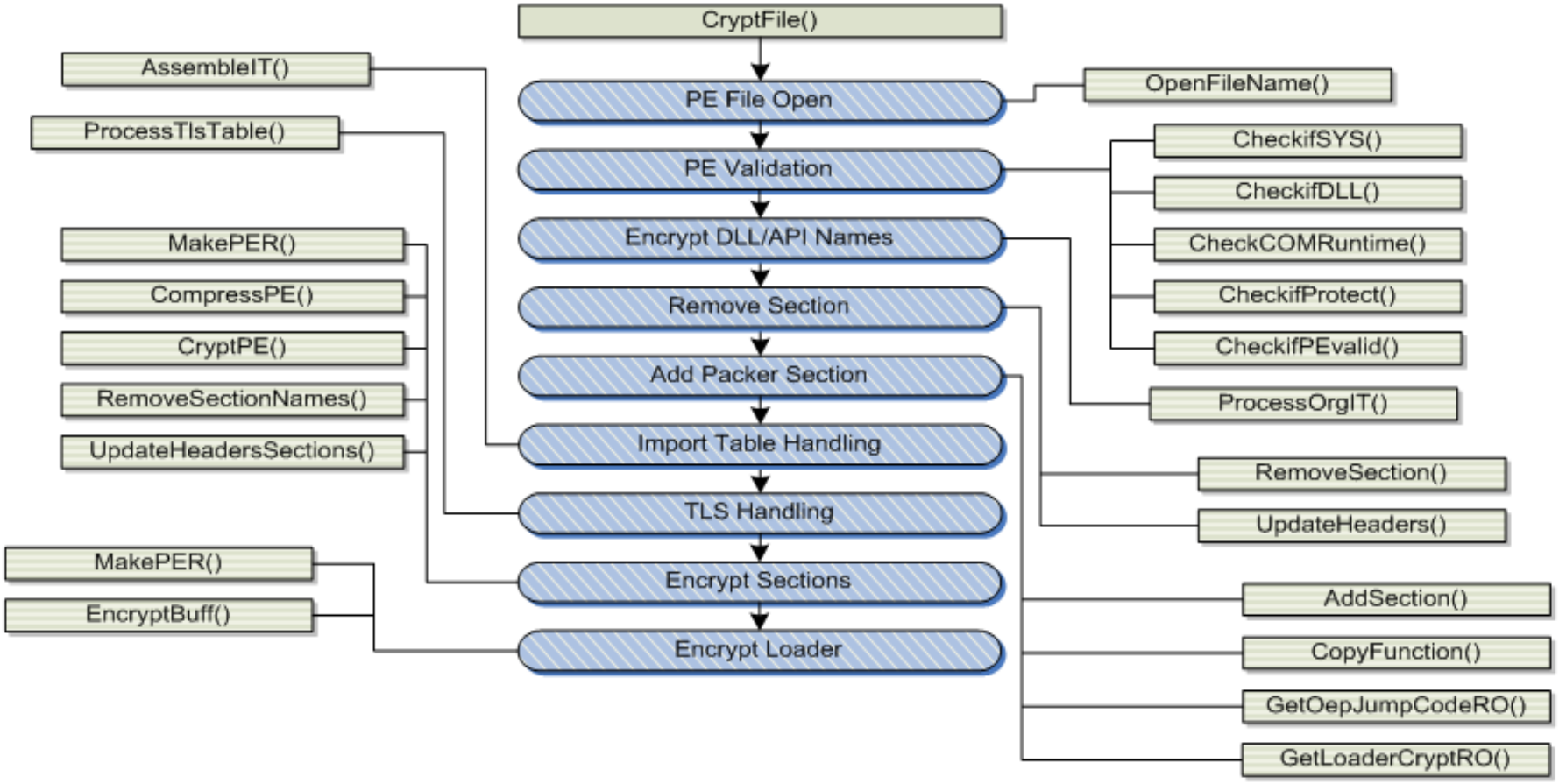
Yoda's Protector

- PE protecting solution
- Support anti-debugging



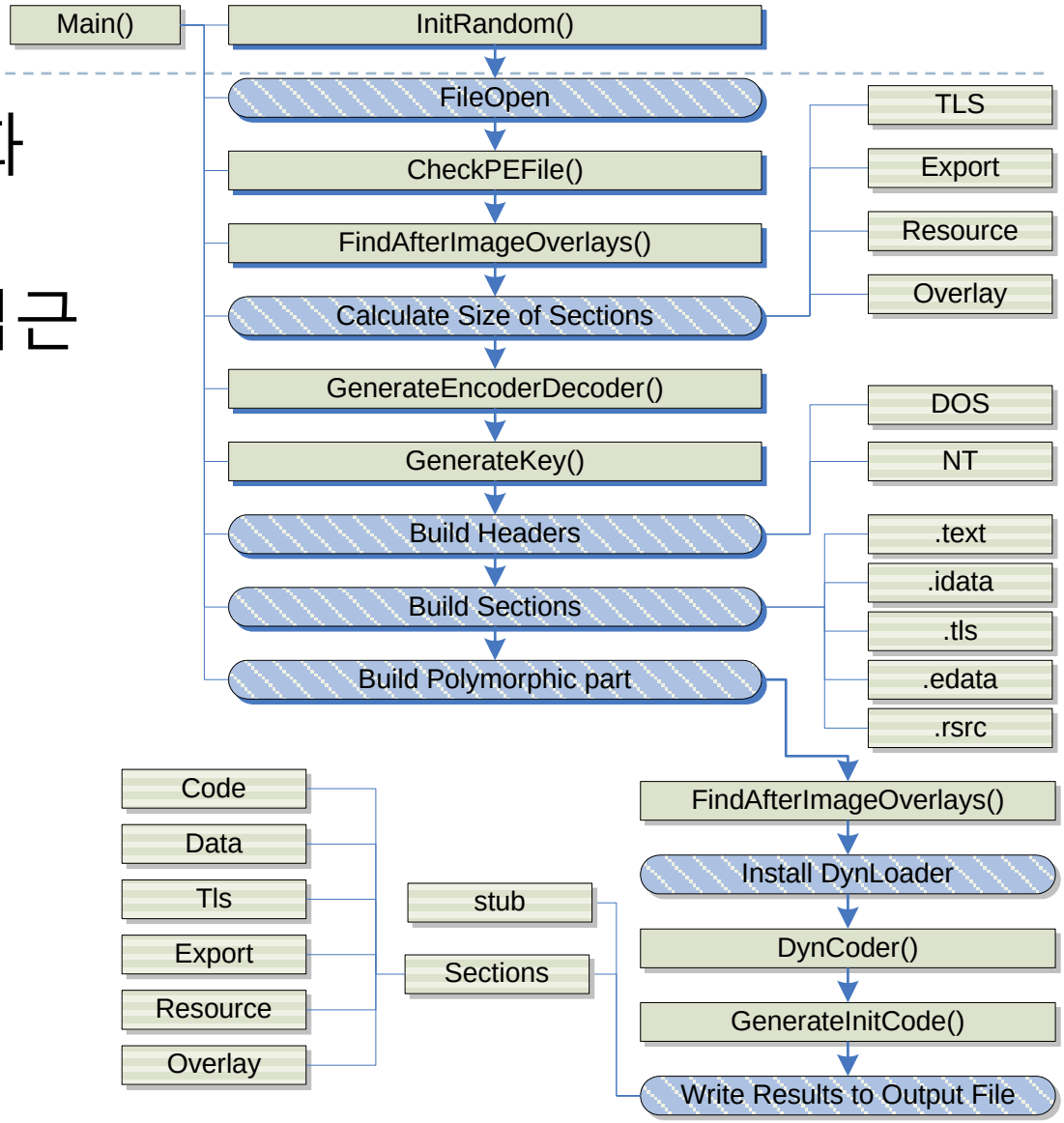
Yoda's Protector

- 소스 코드 분석



Morphine

- 실행 파일을 암호화
- http://hxdef.org/ (2008.1.21 현재 접근 안됨)



기존 방법의 문제점

- PE Format이 압축되는 과정이 항상 동일하기 때문에, 압축 후의 Signature가 동일해짐
  - Vaccine의 Pattern에 추가 되면 압축되더라도 검색이 될 수 있음
- 몇몇 솔루션에 대한 Decompressor 가 존재
  - Decompression Algorithm이 이미 분석되어 PE Format이 다형성을 갖게 되더라도 다형성 전의 원본이 분석될 수 있음

관련 ISSUE

- Loader의 다형성
  - 일반적으로 실행 압축은 .text(code section)을 압축
  - Process를 생성할 때 압축된 .text를 풀기 위해서 Loader가 필요
    - Anti-virus에서 Loader의 Signature를 검색하게 되면 detect되는 문제
    - 여러 개의 Loader를 이용하여 제한적으로 detect를 회피할 수 있음
- Anti-debugging
  - 대부분의 솔루션이 Debugger를 감지해내는 방법으로 Anti-debugging을 구현
  - Anti-debugging 루틴 자체를 crack 하여 무력화 시킬 수 있음
  - Anti-virus에서 Anti-debugging 코드 자체의 signature를 추출하면 detect 될 수 있음

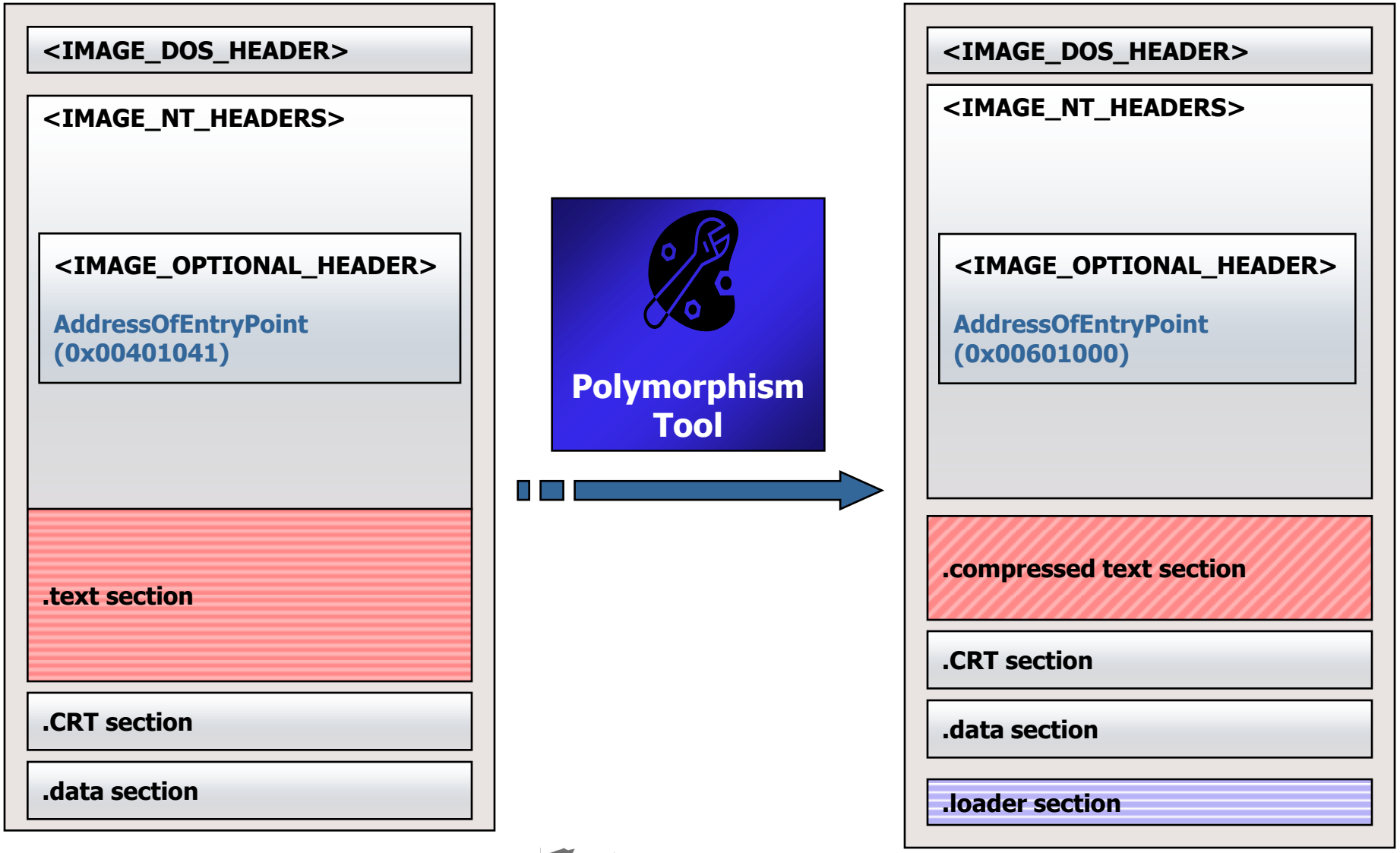
OUR APPROACH

- 실행 압축
  - PE Format의 .text 부분을 압축하기 위한 random 인자값을 이용하여 매번 다른 실행 압축 파일이 생성되도록 구성
- 암호화
  - 실행 압축과 같은 원리로 .text 부분을 암호화하고 복호화 루틴을 loader에 추가
  - 실행 파일에 key를 숨겨야 함
- Instruction Substitution
  - 치환 가능한 명령어 집합(instruction set) 사이에 random 하게 치환 적용
  - Loader에도 적용 가능

연구 내용

- 자료 분석
  - PE 구조 분석
    - 각 section 내용 분석
  - section 추가 방법 분석
  - EIP 변경 방법 분석
  - ID(Import Descriptor)/ IAT(Import Address Table) 분석
  - IAT 실시간 patch 방법 분석
- 다형성 도구 구현

다형성 도구

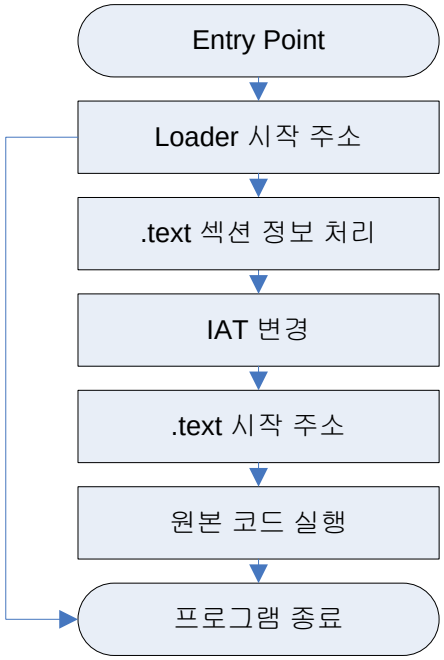

<IMAGE_DOS_HEADER>
<IMAGE_NT_HEADERS>
<IMAGE_OPTIONAL_HEADER>
AddressOfEntryPoint
(0x00401041)
.text section
.CRT section
.data section
Polymorphism
Tool
<IMAGE_DOS_HEADER>
<IMAGE_NT_HEADERS>
<IMAGE_OPTIONAL_HEADER>
AddressOfEntryPoint
(0x00601000)
.compressed text section
.CRT section
.data section
.loader section

다형성 도구의 구현

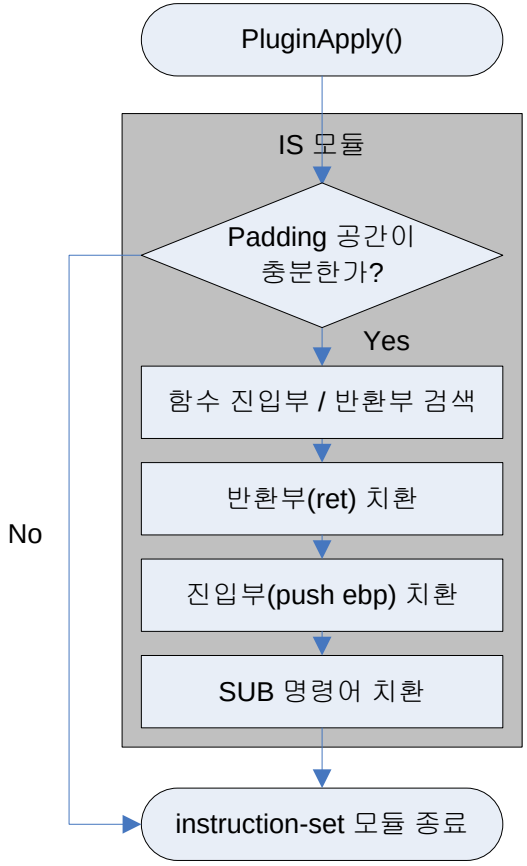
- Disassembler 구현을 이용한 명령어 치환(instruction substitution)
  - Disassembler에서 출력된 Assembly Code를 이용하여 instruction set 적용
- Compressor/Decompressor 구현
  - Decompressor의 경우, Assembly code로 구현하여 Loader 부분과 integration
- 암호화/복호화 모듈 구현 (현재 진행중)

다형성 적용 및 프로그램 실행 순서도

다형성 적용 순서

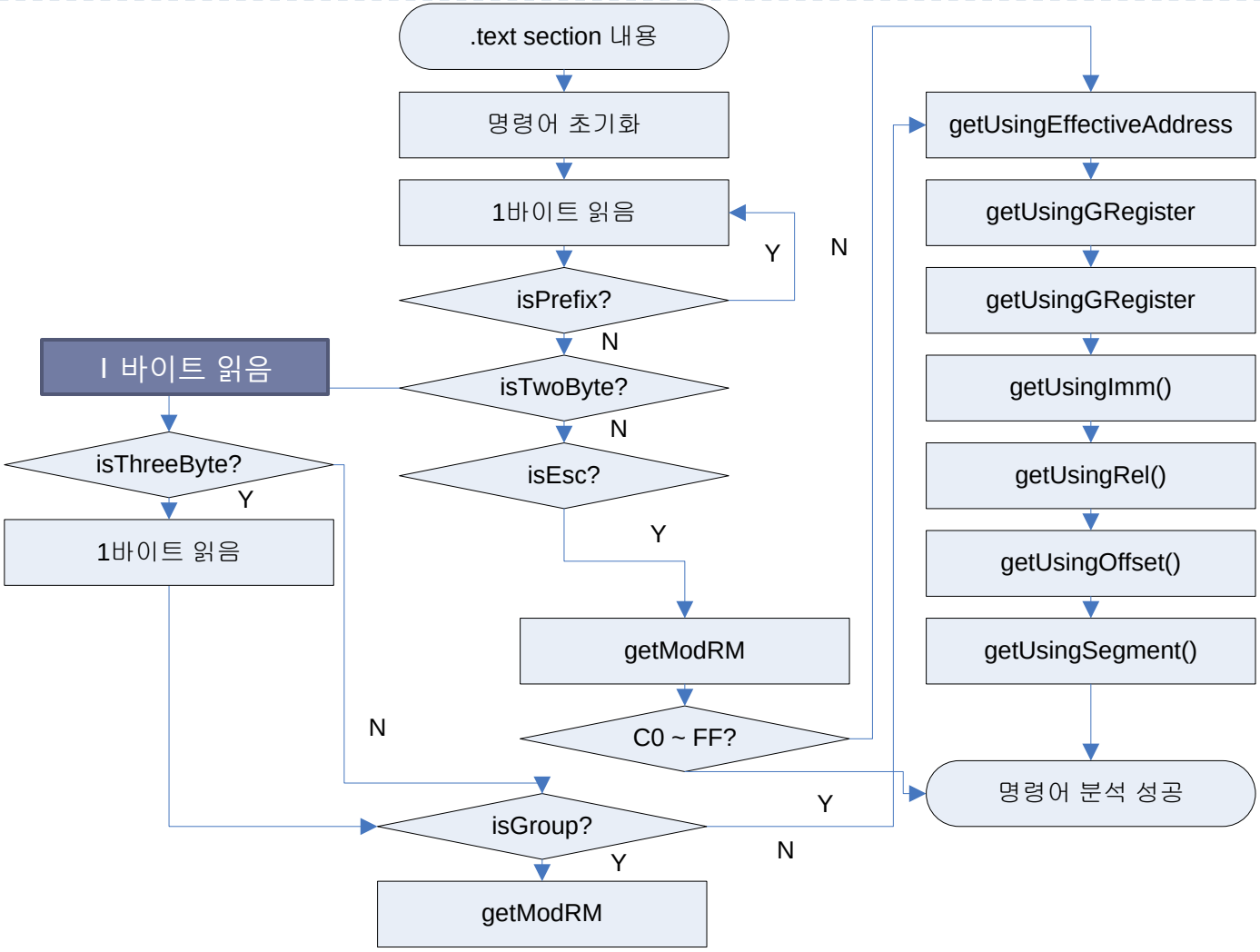


프로그램 실행 순서



Instruction substitution 적용 순서

Disassembler 순서도

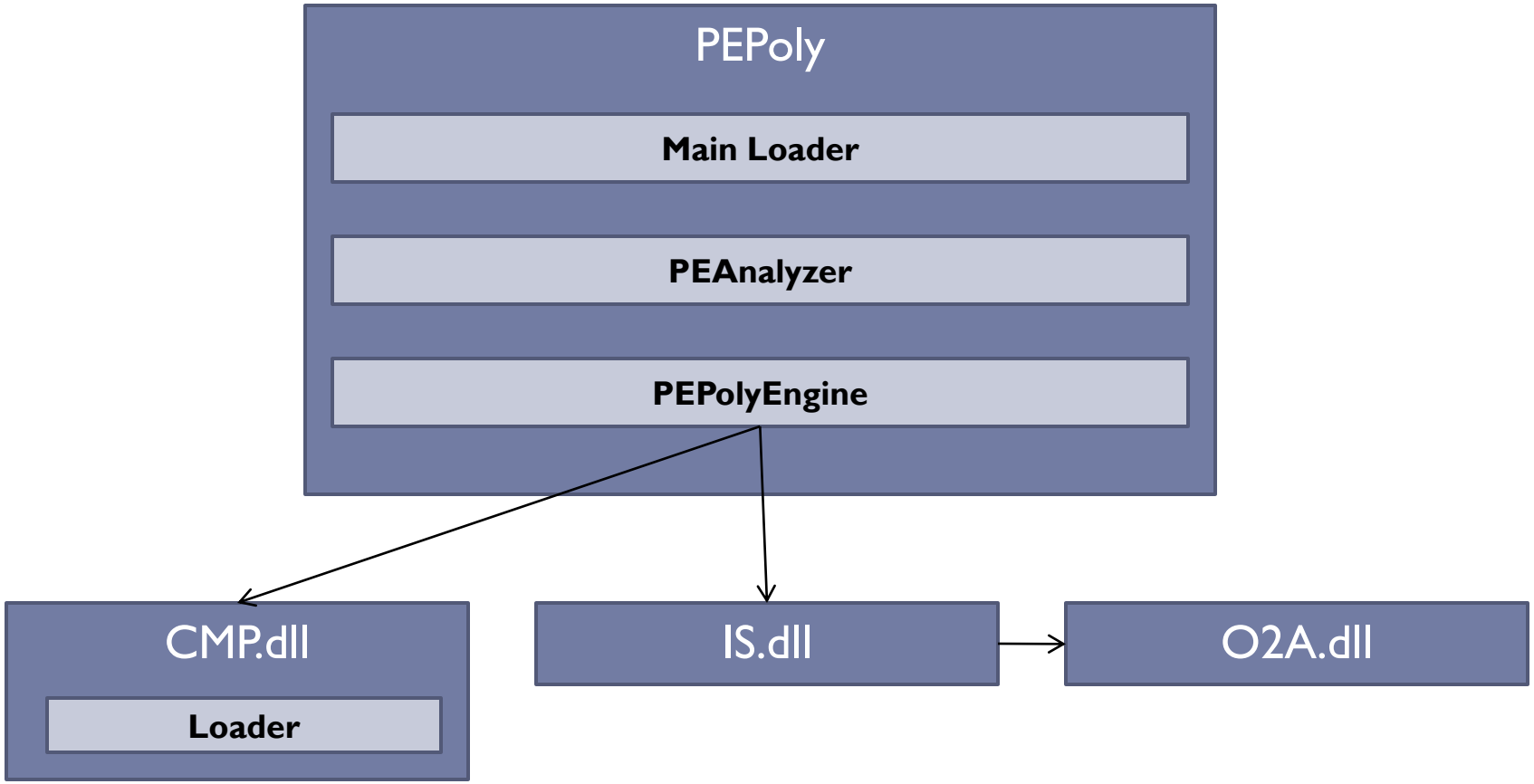


Disassembler 동작 순서

구현 결과물

- 다형성 도구(PEPoly)
  - PEPolyEngine: Polymorphism module
  - IS (IS.dll): Instruction-set module
  - O2A (O2A.dll): Disassembler module
  - CMP (CMP.dll): Compressor module

구현 결과물


PEPoly
Main Loader
PEAnalyzer
PEPolyEngine
CMP.dll
Loader
IS.dll
O2A.dll

다형성 도구의 응용

- 웜/바이러스 변종 생성 및 탐지 방법 연구
  - Zero-day 웜/바이러스 탐지
- Software Cracking을 어렵게
  - SW분석 및 debugging을 어렵게 함
  - 정품 SW의 보호

향후 연구 방향

- 구현 결과물의 한계
  - Loader 부분의 추가
  - Network Packet을 검사하는/메모리 검사하는 anti-virus 도구에 탐지됨
  - padding 부분을 이용한 명령어 치환

- 미 적용된 Instruction-set 적용 기법 연구
  - Import Lookup Table(ILT)및 IAT 변경을 통한 instruction-set 적용 방법 연구
    - 보다 많은 instruction의 변경을 위하여

향후 연구 방향 (cont'd)

- DLL 지원 연구
  - 변화되는 Image-base 에 따른 재배치 기법 연구
  - Export table 분석 및 연구
- Mobile Polymorphic Applications
  - Different mobile environments have different resource constraints

참고문헌

- Yong Tang and Shigang Chen. "An Automated Signature-Based Approach against Polymorphic Internet Worms", IEEE Transactions on Parallel and Distributed Systems, Vol. 18, No. 7, July 2007.
- Arun Lakhotia, "A Method for Detecting Obfuscated Calls in Malicious Binaries", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 31, NO. 11, NOVEMBER 2005
- A. H. Sung J. Xu, P. Chavez, and S. Mukkamala, "Static Analyzer of Vicious Executables", Proceedings of the 20th Annual Computer Security Applications Conference (ACSACi04)
- Sharath K. Udupa, Samuya K. Debray and Matias Madou, "Deobfuscation Reverse Engineering Obfuscated Code", In Proceedings of the 12th Working Conference on Reverse Engineering, 2005.
- Cullen Linn, "Obfuscation of Executable Code to Improve Resistance to Static Disassembly", In Proceedings of the 10th ACM conference on Computer and communications security, 2003.
- Benjamin Schwarz, "Disassembly of Executable Code Revisited", In Proceedings of the Ninth Working Conference on Reverse Engineering, 2002.
- Many URLs previously mentioned.

Q & A

감사합니다