



version 1 KO, August 2012

Figure 1 displays the Section Table of a PE file. The top part shows a hex dump of the section table entries, with the first row highlighted in yellow. The second part is a table representing the section table data.

Name	VirtualSize	VirtualAddress	SizeOfRawData	PointerToRawData	PointerToSectionName	Characteristics
.text	0x1000	0x1000	0x200	0x200		CODE, EXECUTE, READ
.rdata	0x1000	0x2000	0x400	0x400		INITIALIZED, READ
.data	0x1000	0x3000	0x200	0x600		DATA, READ, WRITE

The table shows the following sections:

- .text**: VirtualSize 0x1000, VirtualAddress 0x1000, SizeOfRawData 0x200, PointerToRawData 0x200. Characteristics: CODE, EXECUTE, READ.
- .rdata**: VirtualSize 0x1000, VirtualAddress 0x2000, SizeOfRawData 0x400, PointerToRawData 0x400. Characteristics: INITIALIZED, READ.
- .data**: VirtualSize 0x1000, VirtualAddress 0x3000, SizeOfRawData 0x200, PointerToRawData 0x600. Characteristics: DATA, READ, WRITE.

[illegible]

이것은 파일의 전체 정보입니다. 하지만, 많은 PE 파일은 더 많은 정보를 포함하고 있습니다. 위의 정보는 간단하고 짧게 설명한것입니다.

프로세스 로딩

섹션 테이블 분석

이곳의 위치 계산: `OptionalHeader` 의 시작위치 + `SizeOfOptionalHeader`

섹션갯수만큼(`NumberOfSections`) 섹션의 항목이 포함되어 있습니다

유요한 경계(`Alignment`)인지 검사.:
`FileAlignments, SectionAlignments`

메모리에 파일이 올라갈때(mapped) 다음을 참조 합니다.
 이미지베이스
 헤더크기
 섹션테이블



데이터 디렉토리를 분석
 데이터 디렉토리는 Optional Header 다음에 위치
 그것은 RVA and Size 이다.
 임포트 영역은 항상 두번째 항목이다.
 임포트 영역 분석
 각 지시자(descriptor)는 명시된 DLL 이름.
 명시된 DLL 이 메모리에 로드될시
 IAT and INT 가 함께 분석이 된다.
 INT(ImportNameTable)에 각각의 API 포함
 이 주소는 IAT 영역에 기록되게 된다.



엔트리포인트 에서 코드가 시작되며
코드에 존재하는 IAT 는 API 를 가리킨다.

Notes

INT Import Name Table
Null로 끝나는 리스트는 Hint/Name 구조체를 가리킵니다.

INT Import Address Table
Null로 끝나는 리스트를 가리킴
INT의 복사본
로딩후, 이것은 임포트 API를 가리키게 됩니다.

HINT
DLL의 익스포트 테이블에 포함되어 있는 인덱스
값 필요하지는 않지만, 재공통경유 검색속도가 향상됩니다.