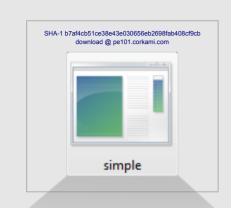
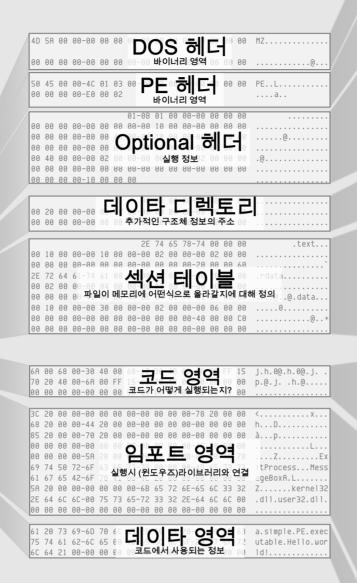
PE 이해







I ASCII 덤프 항목(Fields) 고정 시그니쳐(MZ) PE 헤더 주소 **1** 0x40 고정 시그니쳐(P,E,'0','0') 'PE', 0, 0 프로세서정보 : ARM/MIPS/Intel/. 0x14c [intel 386] Machine 50 45 00 00-4C 01 03 00-00 00 00-00 00 00 00 PE..L.... NumberOfSections 섹션의 갯수 2 SizeOfOptionalHeader 섹션 테이블의 위치와 관련있음(=Optional 헤더의크기) **②** 0xe0 0x102 [32b EXE] Characteristics 실행시 코드 시작주소(진입점) 5 AddressOfEntryPoint .. OB 01 00 00-00 00 00 00 파일이 메모리에 로딩될때의 할당되는 메모리 영역의 (시작)주소 0x400000 ImageBase SectionAlignment 0x1000 메모리 로딩시, 메모리 블럭크기(이값 단위로 메모리 할당) 2 메로이에 로딩시, 매핑되는 파일의 블럭크기(이라 단위로 파일을 위어들임) 2 FileAlignment 0x200 00 00 00 00-00 00 00 00-<mark>04 00</mark> 00 00-00 00 00 00 요구되는 윈도우 버전 MajorSubsystemVersion | 4 [NT 4 or later] 요구되는 전체 메모리크기 SizeOfImage 0x4000 SizeOfHeaders 총 헤더의 크기 🔞 0x200 2 [GUI] 드라이버, GUI, 커맨드 라인 . Subsystem NumberOfRvaAndSizes 16 데이타 디렉토리의 갯수 ④ ...00 00 00 00-00 00 00 00 임포트영역의 RVA^{*} 4 섹션 테이블(Section Table) Name VirtualSize VirtualAddress SizeOfRawData PointerToRawData Characteristics <u>2E 72 64 61-74 61 00 00</u>-00 10 00 00-<mark>00 20 00 00</mark> CODE EXECUTE READ .text .rdata 0x1000 0x2000 INITIALIZED READ 00 02 00 00-<mark>00 04 00 00</mark>-00 00 00 00-00 00 00 00 각 섹션의 PointerToRawData 위치(Offset) 에서 SizeofRawData 크기만큼 읽어 들인다 이것은 ImageBase + VirtualAddress 메모리 영역에 <mark>VirtualSize</mark> 크기만큼 로드되며, characteristics 속성을 포함하게 된다. C 언어 형식 push 0x403000 push 0x403017 6A 00 68 00-30 40 00 68-17 30 40 00-6A 00 FF 15 j.h.0@.h.0@.j. 70 20 40 00-<mark>6A 00 FF 15</mark>-68 20 40 00 p.@.j. .h.@. call [0x402070] →MessageBox(0, "Hello World!", "a simple PE executable", 0); call [0x402068] 임포트 구조체 디스크립터(descriptors) 3C 20 00 00-00 00 00 00-00 00 00 00-78 20 00 00 C............... 68 20 00 00-<mark>44 20 00 00</mark>-00 00 00 00-00 00 00 00 | h...D....... 85 20 00 00-70 20 00 00-00 00 00 00-00 00 00 da...p.... 00 00 00 00-00 00 00 00-00 00 00 00-4C 20 00 00 메모리주소 0x402068 는 kernel32.dll 의 ExitProcess 를 가리키게 됩니다. \rightarrow 0x204c, 0^{IA1} 메모리주소 0x40<mark>2070</mark> 는 user32.dll 의 <mark>MessageBoxA 를</mark> 가리키게 됩니다. 69 74 50 72-6F 63 65 73-73 00 00 00-4D 65 73 73 | itProcess...Mess 61 67 65 42-6F 78 41 00-4C 20 00 00-00 00 00 00 **5A 20 00 00-00 00 00 00-6B 65 72 6E-65 6C 33 32 Z......**kernel32 0x2085 -user32.dll 2E 64 6C 6C-00 75 73 65-72 33 32 2E-64 6C 6C 00 .dll.user32.dll. 61 20 73 69-6D 70 6C 65-20 50 45 20-65 78 65 63 a.simple.PE.exec a simple PE executable\0 75 74 61 62-6C 65 00 48-65 6C 6C 6F-20 77 6F 72 utable.Hello.wor 이것은 파일의 전체 정보 입니다. 하지만, 많은 PE 파일은 더 많은 정보를 포함하고 있습니다. 위의 정보는 간단하고 짧게 설명한것입니다.

번역 <Daniel, Choi>

