|  |  |  |
| --- | --- | --- |
| Metadata | | Number of Rules |
| HDR | Header | 19 |
| PHT | Program Header Table | 22 |
| SHT | Section Header Table | 37 |
| STRS | String Table | 3 |
| DYN | Dynamic Section | 18 |
| NOTE | Note Section | 4 |
| SYM | Symbols Table | 15 |
| REL | Relocations Table | 3 |
| HASH | Hash Table | 2 |
| ENV | OS Environment Variables | 3 |
| Total | | 126 |

Melkor Fuzzer Rules

Specification 1.2 Violations

XXX: Semi-valid 의미 체계로 fuzz될 값

|  |  |  |  |
| --- | --- | --- | --- |
| Rule | Specification | Violation description | ELF metadata |
| hdr1 | 만약 프로그램 헤더 데이블이 존재하는 경우 시스템에 process image를 생성하는 방법을 알려준다. | PHT가 없는 실행 가능한 ELF | - HDR  e\_type = ET\_EXEC | ET\_DYN  e\_phoff = 0  e\_phentsize = 0  e\_phoff = Valid offset  e\_phnum = 0  e\_phentsize = Valid number |
| hdr2 | section header table에는 파일의 섹션을 설명하는 정보가 포함되어 있다. 모든 섹션에는 테이블 안에 항목이 있다. 링크하는 동안 사용되는 파일에는 섹션 헤더 테이블이 있어야 한다. | SHT 없이 재배치할 수 있는 파일  비어있는 SHT | - HDR  e\_type = ET\_REL  e\_shoff = 0  e\_shentsize = 0  e\_shoff = Valid offset  e\_shnum = 0  e\_shentsize = Valid number |
| hdr3 | e\_type 객체 파일 형식을 식별  ET\_LOPROC에서 ET\_HIPORC까지의 값은 프로세서별 semantic으로 예약되어 있다.  다른 값은 예약되어 있으며 필요에 따라 새로운 객체 파일 형식에 할당될 것이다. | ELF 타입이 정상값으로 설정됨(<5)  무효하고 정상적이지 않은 값(>=5)  또는 zero | - HDR  e\_type < 5  e\_type >= 5 <= ET\_HIPROC  e\_type = 0 |
| hdr4 | e\_machine 이 멤버의 값은 개별 파일에 필요한 아키텍쳐를 지정한다. | 무효한/정상적이지 않은 값을 가진 ELF\_machine | - HDR  e\_machine > 16  e\_machine = 0 |
| hdr5 | e\_entry 만약 파일에 관련된 진입점이 없을 경우, 이 멤버는 0을 보유 | Invalid entry point(0 or out of range)  some point in kernel-land | - HDR  e\_entry = XXX  e\_entry = 0 |
| hdr6 | e\_phoff 이 멤버는 프로그램 헤더 테이블의 파일 오프셋을 바이트 단위로 유지한다. 만약 파일에 프로그램 헤더 테이블이 없다면, 이 멤버는 0을 유지한다. | PHT out of bounds | - HDR  e\_phoff = XXX |
| hdr7 | e\_ehsize 이 멤버는 ELF 헤더의 크기를 바이트로 고정한다. | Random ELF header size | - HDR  e\_ehsize = XXX |
| hdr8 | e\_phentsize 이 멤버는 파일의 프로그램 헤더 테이블에 있는 한 항목의 크기(바이트)를 보유한다. 모든 항목은 크기가 동일하다.  e\_phnum 이 멤버는 프로그램 헤더 테이블의 항목 수를 보유한다. 따라서 e\_pentsize와 e\_phnum의 product은 테이블의 크기를 바이트로 제공한다.  파일에 프로그램 헤더 테이블이 없는 경우 e\_phnum은 0으로 유지된다. | 낮은 값과 높은 값의 조합  e\_phentsize to zero | - HDR  e\_phentsize = XXX  e\_phnum = XXX  e\_phentsize = 0 |
| hdr9 | -e\_shentsize 이 멤버는 섹션 헤더의 크기를 바이트 단위로 유지한다.  -섹션 헤더는 섹션 헤더 테이블의 한 항목이다. 모든 항목은 동일한 크기이다.  -e\_shnum 멤버는 섹션 헤더 테이블의 항목 수를 포함한다.따라서 e\_shentsize와 e\_shnum의 결과는 섹션 헤더 테이블의 크기(바이트)를 제공한다. 파일에 섹션 헤더 테이블이 없으면 e\_shnum은 값 0을 보유한다. | Combination of low and out of bounds  e\_shentsize to zero | - HDR  e\_shentsize = XXX  e\_shnum = XXX  e\_shentsize = 0 |
| hdr10 | e\_shstrndx 멤버는 섹션 이름 문자열 테이블과 연관된 항목의 섹션 헤더 테이블 인덱스를 보유한다.  파일에 섹션 이름 문자열 테이블이 없으면 이 멤버는 SHN\_UNDEF 값을 보유한다. | Index to zero and out of bounds | - HDR  e\_shstrndx = XXX  e\_shstrndx = 0 |
| hdr11 | EI\_CLASS 다음 바이트, e\_ident[EI\_CLASS]는 파일의 클래스 또는 용량을 식별 | ELF class with invalid/uncommon values | - HDR  e\_ident[EI\_CLASS] > ELFCLASS64  e\_ident[EI\_CLASS] = 0 |
| hdr12 | EI\_DATA Byte e\_ident[EI\_DATA]는 객체 파일에서 프로세서 특정 데이터의 데이터 인코딩을 지정한다. | ELF encoding with invalid/uncommon values | - HDR  e\_ident[EI\_DATA] > ELFDATA2MSB  e\_ident[EI\_DATA] = 0 |
| hdr13 | EI\_VERSION 바이트 e\_ident[EL\_VERSION]은 ELF 헤더 버전 번호를 지정  현재 이 값은 e\_version에 대해 위에 설명한대로 EV\_CURRENT 여야한다. | ELF version different than EV\_CURRENT | - HDR  e\_version > EV\_CURRENT  e\_version = 0  e\_ident[EI\_VERSION] > EV\_CURRENT  e\_ident[EI\_VERSION] = 0 |
| hdr14 | ELF 헤더의 e\_shoff 멤버는 파일의 시작 부분부터 섹션 헤더 테이블까지의 바이트 오프셋을 제공한다.  e\_shnum은 섹션 헤더 테이블에 포함된 항목 수를 알려준다.  e\_shentsize는 각 항목의 크기(바이트)를 제공 | SHT offset out of bound  Combination of low and high values for sh\_num and e\_shentsize | - HDR  e\_shoff = XXX  e\_shnum = XXX  e\_shentsize = XXX |
| hdr15 | SHN\_LORESERVE 이 값은예약된 인덱스 범위의 하한을 지정 | ELF file with e\_shnum = SHN\_LORESERVE | - HDR  e\_shnum = SHN\_LORESERVE |
| pht1 | p\_type 이 멤버는 이 배열 요소가 나타내는 세그먼트의 종류나 배열 요소 정보를 해석하는 방법을 알려줌 | Change the program header type with random  common / invalid values  Change the program headers type to cero | - PHT  p\_type = XXX  p\_type = 0 |
| pht2 | p\_offset 이 멤버는 세그먼트의 첫 번째 바이트가 상주하는 파일의 시작 부분에서 오프셋을 제공함 | Out of bounds  Offset unaligned | - PHT  (p\_offset = XXX) % PAGESIZE = 0  (p\_offset = XXX) % PAGESIZE != 0 |
| pht3 | p\_vaddr 이 멤버는 세그먼트의 첫 번째 바이트가 메모리에 상주하는 가상 주소를 제공함.  p\_paddr 실제 주소 지정과 관련된 시스템에서 이 멤버는 세그먼트의 실제 주소용으로 예약되어 있음 | Valid p\_vaddr but invalid p\_paddr and vice  versa  Invalid address  Some point in kernel-land  Addresses unaligned | - PHT  p\_vaddr = p\_vaddr  p\_paddr = XXX  p\_vaddr = XXX  p\_paddr = p\_paddr  (p\_vaddr = XXX) % PAGESIZE = 0  (p\_paddr = XXX) % PAGESIZE = 0  (p\_vaddr = XXX) % PAGESIZE != 0  (p\_paddr = XXX) % PAGESIZE != 0 |
| pht4 | p\_filesz 이 멤버는 세그먼트의 파일 이미지의 바이트 수를 제공함.  (이것은 0일수도 있음)  p\_memsz 이 멤버는 세그먼트의 메모리 이미지의 바이트 수를 제공한다.  (0 일수도 있음) | Bytes in memory but not in file  Bytes in file but not in memory  Loadable segment with cero bytes  A big value of bytes in memory | - PHT  p\_filesz = 0  (p\_memsz = XXX) % PAGESIZE = 0  p\_filesz = XXX  p\_memsz = 0  p\_filesz = 0  p\_memsz = 0  p\_memsz = 0xffffffff |
| pht5 | p\_align 로드할 수 있는 프로세스 세그먼트는 p\_vaddr 및 p\_offset에 대해 페이지 크기를 모듈로 합한 값을 가져야 한다.  이 멤버는 메모리와 파일에서 세그먼트가 정렬되는 값을 제공함  그렇지 않으면 p\_align은 2의 양의 정수이어야하며 p\_addr은 p\_offset과 동일해야한다. | p\_align not power of two  (3,5,7,9,10,11,12,13,14,15,17, etc.) | - PHT  p\_align = PAGESIZE - 1  p\_align = PAGESIZE +1  p\_align = XXX |
| pht6 | PT\_LOPROC-0x70000000  PT\_HIPROC- 0x7fffffff | Program headers' types between these two  constants | - PHT  p\_type >= PT\_LOPROC <= PT\_HIPROC |
| pht7 | PT\_LOAD 배열 요소는 p\_filesz 및 p\_memsz에 의해 설명된 로드 가능한 세그먼트를 지정. | More file image bytes than in memory  A big value of bytes in file image | - PHT  p\_filesz > p\_memsz  p\_filesz = 0xffffffff |
| pht8 | 필요한 경우, 패딩은 디스크립터에 대한 4바이트 정렬을 보장하기 위해 존재한다.  이러한 패딩은 namesz에 포함되지 않음. | Make the note section's size unaligned | - PHT  p\_type = PT\_NOTE  p\_filesz % 4 != 0 |
| pht9 | PT\_INTERP 배열 요소는 인터프리터로 호출할 Null 종료 경로 이르므이 위치와 크기를 지정  이 세그먼트 유형은 실행 가능 파일에 대해서만 의미가 있음  (공유 오브젝트에 대해 발생 가능)  하나의 파일에서 두 번 이상 나타날 수 없음. | Point the path to itself  Delete the PT\_INTERP program header from  the PHT  Put two more PT\_INTERP headers to the PHT | - PHT  e\_type = ET\_EXEC | ET\_DYN  p\_type != PT\_INTERP  String replacement to point to itself  Algorithm to add two PT\_INTERP |
| pht10 | 시스템이 로드 가능한 세그먼트의 메모리 이미지를 만들면 p\_flags 멤버에 지정된대로 접근 권한을 부여함  PF\_MASKPROC 마스크에 포함된 모든 비트는 프로세서 관련 의미론을 위해 예약되어 있음 | Fuzz the flags combining the common values  Add the PF\_MASKPROC to the flags | - PHT  p\_flags = XXX  p\_flags |= PF\_MASKPROC |
| pht11 | 일반적인 텍스트 세그먼트는 읽기 및 실행이 가능하지만 쓰기 권한은 없다. | Locate the text segment and set the write flag | - PHT  p\_flags |= PF\_W |
| pht12 | PT-DYNAMIC 프로그램 헤더 요소는 아래의 동적 섹션에서 설명한 .dynamic 섹션을 가리킨다. | Change the PT\_DYNAMIC header to point  somewhere else | - PHT  p\_type = PT\_DYNAMIC  p\_offset = XXX  p\_vaddr = XXX |
| pht13 | PT\_SHLIB 이 세그먼트 유형은 예약되었지만 지정되지 않은 의미를 가진다. 이 유형의 배열 요소가 포함된 프로그램은 UNIX System V용 ELF 사양을 따르지 않는다. | Add a PT\_SHLIB at the end of the PHT | - PHT  PHT[e\_phnum - 1] = PT\_SHLIB |
| pht19 | PT\_PHDR 배열 요소가 있는 경우 파일과 프로그램의 메모리 이미지 모두에서 프로그램 헤더 테이블 자체의 위치와 크기를 지정한다.  이 세그먼트의 유형은 파일에서 두 번 이상 나타날 수 없다. 또한 프로그램 헤더 테이블이 프로그램의 메모리 이미지의 일부인 경우에만 발생할 수 있다. 존재하는 경우 로드 가능한 세그먼트 항목 앞에 와야한다. | Delete the PT\_PHDR from the PHT  Create an extra PT\_PHT right after the first  one found in the PHT  Move the PT\_PHDR to the end of the PHT | - PHT  p\_type != PT\_PHDR  PHT[x].p\_type = PT\_PHDR  PHT[x+1].p\_type = PT\_PHDR  PHT[e\_phnum - 1] = PT\_PHDR |
| pht20 | 프로그램 헤더 테이블의 로드 가능한 세그먼트 항목은 p\_vaddr 구성원에 따라 오름차순으로 표시된다. | Re-order the PT\_LOAD segments in  descending order  Re-order the PT\_LOAD entries randomly | - PHT  Algorithm to re-order the PT\_LOAD entries'  p\_vaddr address (descending or randomly) |
| pht21 | PT\_INTERP 존재할 경우 로드 가능한 세그먼트 항목 앞에 와야한다. | Move the PT\_INTERP to the end of the PHT  (after the PT\_LOAD segments) | - PHT  Algorithm to relocate the PT\_INTERP to the  end of the PHT. The program header in the  end will take place where the original  PT\_INTERP is.  PHT[PT\_INTERP] = PHT[e\_phnum - 1]  PHT[e\_phnum - 1] = PT\_INTERP |
| sht1 | sh\_name 이 멤버는 섹션의 이름을 지정한다. | A sh\_name out of bounds | - SHT  sh\_name = XXX |
| sht2 | sh\_addr 섹션이 프로세스의 메모리 이미지에 나타나면, 이 멤버는 섹션의 첫번째 바이트가 있어야하는 주소를 제공한다. 그렇지 않으면 구성원에 0이 포함된다. | Invalid address  Some point in kernel-land | - SHT  sh\_addr = XXX |
| sht3 | sh\_offset 이 멤버의 값은 파일의 시작 부분에서 섹션의 첫 번째 바이트까지의 바이트 오프셋을 제공한다. SHT\_NOBITS 섹션 유형 하나는 파일에서 공백을 차지하지 않으며, sh\_offset 멤버는 파일의 개념적 배치를 찾는다. | A sh\_offset out of bounds | - SHT  sh\_offset = XXX |
| sht4 | sh\_size 이 멤버는 섹션의 크기(바이트)를 제공한다. 섹션 유형이 SHT\_NOBITS가 아닌 한, 섹션은 파일에서 sh\_size 바이트를 차지한다. 파일의 섹션은 겹칠 수 없다. 파일의 어떤 바이트도 둘 이상의 섹션에 존재하지 않는다. | Combination of low and high values | - SHT  sh\_size = XXX |
| sht5 | sh\_addralign 즉, sh\_addr의 값은 sh\_addralign의 값을 모듈로 0으로 합쳐져야한다.  현재 0과 양의 정수 2만 허용된다. 0과 1값은 섹션에 정렬 제약 조건이 없음을 의미한다. | sh\_addralign not power of two  (3,5,7,9,10,11,12,13,14,15,17, etc.)  sh\_addralign = PAGESIZE +/- 1 | - SHT  sh\_addralign = XXX  sh\_addralign = PAGESIZE - 1  sh\_addralign = PAGESIZE + 1 |
| sht6 | sh\_entsize 일부 섹션은 심볼 테이블과 같은 고정 크기 항목의 테이블을 보유한다.  이러한 섹션의 경우 이 멤버는 각 항목의 크기(바이트)를 제공한다. | Combination of low and high values  Zero | - SHT  sh\_entsize = XXX  sh\_entsize = 0 |
| sht7 | 섹션 헤더의 sh\_type 멤버는 섹션의 의미(semantic)를 지정한다. | Change the section types with random  common / invalid values | - SHT  sh\_type = XXX |
| sht8 | SHT\_LOPROC에서 SHT\_HIPROC까지 이 범위에 포함된 값은 프로세서 관련 의미를 위해 예약되었다.  SHT\_LOUSER와 SHT\_HIUSER 사이의 섹션 유형은 현재 또는 미래의 시스템 정의 섹션 유형과 충돌없이 응용 프로그램에서 사용할 수 있다. | Section types set to these four constants | - SHT  sh\_type = SHT\_LOPROC + 1  sh\_type = SHT\_HIPROC  sh\_type = SHT\_LOUSER + 1  sht\_type = SHT\_HIUSER |
| sht9 | 인덱스가 정의되지 않은 섹션 참조를 표시하더라도 인덱스 0(SHN\_UNDEF) 섹션 헤더가 있다. 이 항목에는 다음 내용이 들어있다. 섹션 헤더 테이블 항목: 인덱스 0 | SHT index 0 with random / uncommon values | - SHT  SHT[0]: sh\_\* = XXX |
| sht10 | 섹션 헤더의 sh\_flags 멤버는 섹션의 속성을 설명하는 1-비트 플래그를 포함한다. | sh\_flags with random combinations of the  valid flags  sh\_flags with invalid / undefined values | - SHT  sh\_flags =  SHF\_WRITE|SHF\_ALLOC|SHF\_EXECINS  TR|SHF\_MASKPROC  sh\_flags &= ~ SHF\_WRITE  sh\_flags &= ~ SHF\_ALLOC  sh\_flags &= ~ SHF\_EXECINSTR  sh\_flags = XXX |
| sht11 | SYN\_DYNAMIC : sh\_link = 섹션의 항목이 사용하는 문자열 테이블의 섹션 헤더 인덱스. sh\_info=0  SHT\_HASH: sh\_link= 해쉬 테이블이 적용되는 심볼 테이블의 섹션 헤더 인덱스. sh\_info=0 | sh\_link pointing to a valid SHT index but not  to a string table  sh\_link pointing to 0  sh\_link out of bounds  sh\_info with random values | - SHT  sh\_type = SHT\_DYNAMIC  sh\_link = (1 to e\_shnum - 1) !=  SHT\_STRTAB  sh\_link = XXX  sh\_info = (1 to e\_shnum - 1)  sh\_info = XXX  sh\_type = (SHT\_HASH|SHT\_GNU\_HASH)  sh\_link = (1 to e\_shnum - 1) !=  (SHT\_SYMTAB|SHT\_DYNSYM)  sh\_link = XXX  sh\_info = (1 to e\_shnum - 1)  sh\_info = XXX |
| sht12 | SHT\_REL 및 SHT\_RELA: sh\_link= 연관된 심볼 테이블의 섹션 헤더 색인.  sh\_info = 재배치가 적용되는 섹션의 섹션 헤더 인덱스이다. | sh\_link and sh\_info pointing to a valid SHT  index but not to a symbol table nor relocation  sh\_link out of bounds  sh\_link pointing to 0  sh\_info out of bounds  sh\_info pointing to 0 | - SHT  sh\_type = (SHT\_REL|SHT\_RELA)  sh\_link = (1 to e\_shnum - 1) !=  (SHT\_SYMTAB|SHT\_DYNSYM)  sh\_link = XXX  sh\_info = (1 to e\_shnum - 1)  sh\_info = XXX |
| sht13 | SHT\_SYMTAB 및 SHT\_DYNSYM: sh\_link=연관된 문자열 테이블의 섹션 헤더 인덱스.  sh\_info = 마지막 로컬 심볼의 심볼 테이블 인덱스보다 큰 하나(STB\_LOCAL). | sh\_link and sh\_info with random values  sh\_link out of bounds  sh\_link pointing to 0  sh\_info out of bounds  sh\_info pointing to 0 | - SHT  sh\_type = (SHT\_SYMTAB|SHT\_DYNSYM)  sh\_link = (1 to e\_shnum - 1) !=  (SHT\_STRTAB)  sh\_link = XXX  sh\_info = (1 to e\_shnum - 1)  sh\_info = XXX |
| sht14 | .bss 이 섹션에는 프로그램의 메모리 이미지에 기여하는 초기화되지 않은 데이터가 저장된다.  정의에 따라 시스템은 프로그램 실행을 시작할 때 데이터를 0으로 초기화한다.  섹션은 SHT\_NOBITS 섹션 유형으로 표시된 것처럼 파일 공간을 차지하지 않는다. | Fuzz its attributes  Set a random size | - SHT  sh\_type = SHT\_NOBITS  sh\_flags = XXX  sh\_size = XXX |
| sht15 | .data 및 .data1 이 섹션에는 프로그램의 메모리 이미지에 기여하는 초기화 된 데이터가 있다. | Fuzz its attributes | - SHT  sh\_type = XXX  sh\_flags = XXX  sh\_size = XXX |
| sht16 | .hash 이 섹션은 심볼 해시 테이블을 포함한다. | Fuzz its attributes | - SHT  sh\_type = (SHT\_HASH | SHT\_GNU\_HASH)  sh\_flags = XXX  sh\_size = XXX  sh\_entsize = XXX |
| sht17 | .debug 이 섹션에는 심볼릭 디버깅에 대한 정보가 있다.  내용이 지정되지는 않았다. 접두사가 있는 모든 섹션 이름  디버그는 나중에 사용하기 위해 예약되어 있다.  .line 이 섹션에는 소스 프로그램과 기계 코드간의 대응을 설명하는 심볼릭 디버깅을 위한 행 번호가 정보가 들어 있다. .debug와 마찬가지로 내용이 지정되지는 않는다. | Fuzz the attributes  Fuzz the debugging information (DWARF).  GCC adds debug\_\* | - SHT  sh\_name = ".debug\_\*"  sh\_type = XXX  sh\_flags = XXX  sh\_size = XXX  sh\_entsize = XXX  Fuzz its content. Debugging information.  DWARF. [NOT IMPLEMENTED YET] |
| sht18 | .dynamic 이 섹션에는 동적 연결 정보가 들어 있으며 SHF\_ALLOC 및 SHF\_WRITE와 같은 속성이 있다. SHF\_WRITE 비트가 설정되는지 여부는 운영 체제 및 프로세서에 의해 결정된다. | Clear those flags | - SHT  sh\_type = SHT\_DYNAMIC  sh\_flags &= ~ SHF\_ALLOC  sh\_flags &= ~ SHF\_WRITE |
| sht19 | .rodata 및 .rodata1 이 섹션은 일반적으로 프로세스 이미지 쓰기 불가능 세그먼트에 기여하는 읽기 전용 데이터를 보유한다. | Fuzz its attributes | - SHT  sh\_type = XXX  sh\_flags = XXX  sh\_size = XXX  sh\_entsize = XXX |
| sht20 | .note 이 섹션에는 포맷의 정보가 들어 있다.  필요한 경우, 패딩은 디스크립터에 대한 4바이트 정렬을 보장하기 위해 존재한다.  이러한 패딩은 namesz에 포함되지 않는다. | Fuzz its attributes  Sections's size less or equal the size of the  struct Nhdr.  Section's size unaligned | - SHT  sh\_type = SHT\_NOTE  sh\_flags = XXX  sh\_size = sizeof(Elf32\_Nhdr | Elf64\_Nhdr)  sh\_size % 4 != 0  sh\_addralign = XXX |
| sht21 | .strtab 이 섹션은 문자열을 포함하며 가장 일반적으로 심볼 테이블 항목과 연관된 이름을 나타내는 문자열이다.  파일에 심볼 문자열 테이블을 포함하는 로드 가능한 세그먼트가 있는 경우 섹션의 속성에는 SHF\_ALLOC 비트가 포함된다. 그렇지 않으면 해당 비트가 꺼진다. | Fuzz its attributes | - SHT  sh\_type = SHT\_STRTAB  sh\_flags = XXX  sh\_size = XXX  sh\_entsize = XXX  sh\_addralign = XXX |
| sht22 | .symtab 이 섹션은 심볼 테이블을 포함한다.  파일에 심볼 테이블을 포함하는 로드 가능한 세그먼트가 있으면 섹션의 속성에는 SHF\_ALLOC 비트가 포함된다. 그렇지 않으면 해당 비트가 꺼진다. | Fuzz its attributes | - SHT  sh\_type = SHT\_SYMTAB | SHT\_DYNSYM  sh\_flags = XXX  sh\_size = XXX  sh\_entsize = XXX  sh\_addralign = XXX |
| sht23 | .text 섹션은 프로그램의 “텍스트” 또는 실행 가능한 명령어를 포함한다. | Change the section type to !=  SHT\_PROGBITS and fuzz its attributes | - SHT  sh\_type != SHT\_PROGBITS  sh\_flags &= ~ SHF\_ALLOC  sh\_flags &= ~ SHF\_EXECINSTR  sh\_flags = XXX  sh\_flags = 0 |
| sht24 | .fini & .init | Delete the SHF\_EXECINSTR flag and/or  SHF\_ALLOC | - SHT  sh\_type = SHT\_PROGBITS  sh\_flags &= ~ SHF\_ALLOC  sh\_flags &= ~ SHF\_EXECINSTR  sh\_flags = XXX  sh\_flags = 0 |
| sht25 | .interp 이 섹션에는 프로그램 인터프리터의 경로 이름이 들어 있다. 파일에 섹션을 포함하는 로드 가능한 세그먼트가 있으면 섹션의 특성에는 SHF\_ALLOC 비트가 포함된다.  그렇지 않으면 해당 비트가 꺼진다. | Modify its attributes  Point the path to itself | - SHT  sh\_type = SHT\_NULL  sh\_flags &= ~ SHF\_ALLOC  sh\_flags = XXX  String replacement to point to itself |
| sht26 | .got SHT\_PROGBITS SHF\_ALLOC + SHF\_WRITE | Remove the SHF\_WRITE permission | - SHT  sh\_type = SHT\_PROGBITS  sh\_flags &= ~ SHF\_WRITE |
| sht27 | .plt SHT\_PROGBITS SHF\_ALLOC + SHF\_EXECINSTR | Remove the SHF\_EXECINSTR permission  Leave the SHF\_EXECINSTR permission but  patch the second instruction (jmp) in the PLT  to the original entrypoint. random address or  .init or .fini:  Disassembly of section .plt:  08048470 <printf@plt-0x10>:  8048470: ff 35 f8 9f 04 08 pushl  0x8049ff8  8048476: ff 25 fc 9f 04 08 jmp  \*0x8049ffc | - SHT  sh\_type = SHT\_PROGBITS  sh\_flags &= ~ SHF\_EXECINSTR  plt + 6 = jmp entrypoint (HDR.e\_entry)  plt + 6 = jmp \_init (.init.sh\_addr)  plt + 6 = jmp \_fini (.fini.sh\_addr)  plt + 6 = XXX |
| sht28 | 빈 문자열 테이블 섹션이 허용됩니다. 그 섹션의 헤더의 sh\_size 멤버는 0을 포함한다.  비어있는 인덱스는 빈 문자열 테이블에 대해 유효하지 않는다. | Modify an string table different than  e\_shstrndx so this will be an empty string table  or the offset at the end of the file with a  random size | - SHT  sh\_type = SHT\_STRTAB  shndx != e\_shstrndx  sh\_size = 0  sh\_type = SHT\_STRTAB  shndx != e\_shstrndx  sh\_offset = stat\_info.st\_size  sh\_size = 0x1337 |
| sht29 | SHT\_HASH 동적 링킹에 참여하는 모든 객체는 심볼 해시 테이블을 포함해야한다. | Delete the hash table | - SHT  sh\_type != (SHT\_HASH |  SHT\_GNU\_HASH) |
| sht30 | 오브젝트 파일에는 같은 이름을 가진 섹션이 두 개 이상 있을 수 있습니다. | Add the same section name more than two  times in the file | - SHT  sh\_name = .text | .data | .got | .got.plt | .bss |
| str1 | sh\_name 이 멤버는 섹션의 이름을 지정한다. 이 값은 섹션 헤더 문자열 테이블 섹션에 대한 인덱스이며 null로  끝나는 문자열의 위치를 제공한다.  이러한 1바이트 문자는 7비트 ASCII 문자 세트를 사용한다. 0~127의 범위를 벗어난 묹자값은 문자 인코딩에 따라 하나 이상의 바이트를 차지할 수 있다. | Fuzz the strings inside without deleting the  NULL bytes | - STR  sh\_type = SHT\_STRTAB  Chars > 0x7f |
| str2 | 문자열 테이블 섹션은 일반적으로 문자열이라고하는 null로 끝나는 문자 시퀀스를 포함한다.  인덱스 0인 첫번째 바이트는 널 문자를 보유하도록 정의된다. 마찬가지로 문자열 테이블의 마지막 바이트는 null 문자를 보유하도록 정의되어 모든 문자열에 대해 null 종료를 보장한다. | Delete some NULL bytes between strings  Change the first byte in the string table  Delete the NULL byte from the end | - STR  sh\_type = SHT\_STRTAB  string\_table[x] != 0  string\_table[0] = XXX  string\_table[sh\_size - 1] > 0x7f |
| str3 | 문자열 테이블 섹션은 일반적으로 문자열이라고하는 null로 끝나는 문자 시퀀스를 포함한다. | Fuzz every section that contains a string table  with format string vulnerability triggers | - STR  sh\_type = SHT\_STRTAB  Replace the strings with format string  vulnerability triggers such as %x or %n  without deleting the NULL byte |
| note1 | Note 섹션 – 때때로ㅓ 벤더 또는 시스템 빌더는 다른 프로그램의 적합성, 호환성등을 검사할 특별한 정보를 오브젝트 파일에 표시해야한다.  SHT\_NOTE 유형의 섹션과 PT\_NOTE 유형의 프로그램 헤더 요소를 이 용도로 사용할 수 있다.  섹션 및 프로그램 헤더 요소의 Note 정보에는 대상 프로세서의 형석으로 된 4바이트 단어의 배열인 모든 항목 수가 들어간다.  namesz 와 name 첫번째 namesz 바이트에는 항목의 소유자 또는 작성자에 대한 null 로 종료된 문자 표현이 들어 있다.  이름 충돌을 피하기 위한 공식적인 메커니즘은 없다.  관습에 따른 벤더는 “XYZ Computer Company”와 같은 고유한 이름을 식별자로 사용하며 이름이 없으면 namesz에 0이 포함된다. | namesz with a high value | - NOTE  namesz = XXX |
| note2 | name의 첫 번째 namesz 바이트에는 엔트리의 소유자 도는 작성자에 대한 null로 끝나는 문자 표현이 포함된다. | The NULL bytes in name will be overwritten  Some format strings within name | - NOTE  name = XXX |
| note3 | descsz 및 desc desc의 첫 번째 descsz 바이트는 note 디스크립터를 보유한다.  ELF는 디스크립터의 내용에 제약을 가하지 않는다.  필요한 경우 다음 note 항목에 대한 4 바이트 정렬을 보장하기 위해 패딩이 제공된다.  이러한 패딩은 descsz에 포함되지 않는다. | Fuzz the first descsz bytes in desc  descsz with a random value | - NOTE  desc = XXX  descsz = XXX |
| note4 | type 이 단어는 디스크립터의 해석을 제공한다.  각 작성자는 자체 유형을 제어한다. 단일 형식 값에 대한 여러 해석이 있을 수 있다.  따라서, 프로그램은 디스크립터를 “이해”하기 위해 이름과 타입을 모두 인식해야한다.  ELF는 디스크립터의 의미를 정의하지는 않는다. | Put in type negative values (>= 0x80000000) | - NOTE  type >= 0x80000000 |
| dyn1 | DT\_NEEDED 이 요소는 필요한 라이브러리 이름을 제공하는 널 종료 문자열의 문자열 테이블 오프셋을 보유한다. 오프셋은 DT\_STRTAB 항목에 기록된 테이블에 대한 인덱스이다.  DT\_SONAME 이 원소는 공유 객체의 이름을 제공하는 null로 끝나는 문자열의 문자열 테이블 오프셋을 갖는다. 오프셋은 DT\_STRTAB 항목에 기록된 테이블에 대한 인덱스이다.  DT\_RPATH 이 요소는 “공유 객체 의존성”에서 설명된 null로 끝나는 검색 라이브러리 검색 경로 문자열의 문자열 테이블 오프셋을 포함한다. 오프셋은 DT\_STRTAB 항목에 기록된 테이블에 대한 인덱스이다. | d\_val pointing to an out of bounds index  Pointing to an invalid name / path. | - DYN  d\_tag = DT\_NEEDED | DT\_SONAME |  DT\_RPATH  d\_val = XXX |
| dyn2 | DT\_PLTRELSZ 이 요소는 프로시저 연결 테이블과 연관된 재배치 엔트리의 총 크기(바이트)를 유지한다.  DT\_JMPREL 유형의 항목이 있으면 DT\_PLTRELSZ가 동반해야한다.  DT\_RELSZ 이 요소는 DT\_REL 재배치 테이블의 총 크기(바이트)를 유지한다.  DT\_RELASZ 이 요소는 DT\_RELA 재배치 테이블의 재배치 테이블의 총 크기(바이트)를 유지한다.  DT\_STRSZ 이 요소는 문자열 테이블의 크기(바이트)를 포함한다. | Less bytes in buffer size to trigger memory corruption vulnerabilities  High values to allocate big memory chunks  Zero | - DYN  d\_tag = DT\_PLTRELSZ | DT\_RELSZ |  DT\_RELASZ | DT\_STRSZ  d\_val = XXX  d\_val = 0 |
| dyn3 | DT\_RELAENT 이 요소는 DT\_RELA 재배치 엔트리의 크기를 바이트 단위로 유지한다.  DT\_RELENT 이 요소는 DT\_REL 재배치 엔트리의 크기를 바이트로 유지한다.  DT\_SYMENT 이 요소는 심볼 테이블 엔트리의 크기를 바이트 단위로 유지한다. | Combination of low and high values  Zero | - DYN  d\_tag = DT\_RELAENT | DT\_RELENT |  DT\_SYMENT  d\_val = XXX  d\_val = 0 |
| dyn4 | DT\_PLTGOT 이 요소는 프로시저 연결 테이블 및 global offset table과 연관된 주소를 보유한다. | Pointer to an invalid address and to some  point in kernel-land | - DYN  d\_tag = DT\_PLTGOT  d\_ptr = XXX |
| dyn5 | DT\_HASH 이 요소는 “해시 테이블”에 설명된 심볼 해시 테이블의 주소를 담고 있다. 이 해시 테이블은 DT\_SYMTAB 요소가 참조하는 심볼 테이블을 참조한다.  DT\_SYMTAB 이 요소는 파일의 32비트 클래스에 대한 Elf32\_Sym 항목이 있는 심볼 테이블의 주소를 보유한다. | DT\_HASH entry with valid values but  DT\_SYMTAB with an invalid address or to  some point in kernel-land, and vice versa. | - DYN  d\_tag = DT\_HASH | DT\_GNU\_HASH  d\_ptr = d\_ptr  d\_ptr = XXX  d\_tag = DT\_SYMTAB  d\_ptr = d\_ptr  d\_ptr = XXX |
| dyn6 | DT\_INIT 이 요소는 초기화 함수의 주소를 가지고 있다.  DT\_FINI 이 요소는 종료 함수의 주소를 담고 있다. | Make the address to point to some point in  kernel-land  Make the address to point to the executable's  entrypoint  Endless loop through DT\_INIT to DT\_FINI and  DT\_FINI to DT\_INIT | DYN  d\_tag = (DT\_INIT | DT\_FINI)  d\_ptr = XXX  d\_ptr = HDR.e\_entry  d\_tag = DT\_INIT  d\_ptr = DT\_FINI.d\_ptr  d\_tag = DT\_FINI  d\_ptr = DT\_INIT.d\_ptr |
| dyn7 | DT\_SYMBOLIC 이 요소가 공유 객체 라이브러리에 존재한다면 라이브러리내의 참조에 대한 동적 링커의 심볼 resolution 알고리즘이 변경된다. 실행 파일을 사용하여 심볼 검색을 시작하는 대신 동적 링커는 공유 객체 자체에서 시작한다. 공유 객체가 참조된 심볼을 제공하지 못하면 동적 링커는 실행 파일과 다른 공유 객체를 평소와 같이 검색한다.  DT\_TEXTREL 이 멤버의 부재는 재배치 항목이 프로그램 헤더 테이블의 세그먼트 권한으로 지정된 쓰기 불가능한 세그먼트를 수정해야한다는 것을 의미한다. 이 멤버가 있으면 하나 이상의 재배치 항목이 쓰기 불가능 세그먼트에 대한 수정을 요청할 수 있으며 동적 링커가 이에 따라 준비할 수 있다. | Replace the DT\_DEBUG for thes ones | - DYN  d\_tag = DT\_DEBUG  d\_tag = DT\_SYMBOLIC | DT\_TEXTREL |
| dyn8 | DT\_PLTREL 이 멤버는 프로시저 연결 테이블이 참조하는 재배치 엔트리의 타입을 지정한다. d\_val 멤버는 DT\_REL 또는 DT\_RELA를 적절하게 유지한다. 프로시저 연결 테이블의 모든 재배치는 동일한 재배치를 사용한다. | Set to unknown types | - DYN  d\_tag = DT\_PLTREL  d\_val != (DT\_REL | DT\_RELA) |
| dyn9 | DT\_BIND\_NOW 공유 객체 또는 실행 파일이 있으면 이 항목은 동적 링커가 프로그램에 제어를 전달하기 전에 이 항목이 들어있는 객체의 모든 재배치를 처리하도록 지시한다. 이 항목이 있으면 환경 또는 dlopen을 통해 지정될 때 이 객체에 대한 지연 바인딩을 사용하라는 지시문보다 우선시한다. | Replace the DT\_DEBUG for this one | - DYN  d\_tag = DT\_BIND\_NOW |
| dyn10 | 공유 오브젝트 이름에 위의 /usr/lib/lib2 또는 directory/file과 같이 이름의 임의의 위칠에 슬래시(/) 문자가 하나 이상있는 경우 동적 링커는 해당 문자열을 경로 이름으로 직접 사용한다.위의 lib1과 같이 이름에 슬래시가 없으면 세가지 기능이 공유 객체 경로 검색을 지정하며 다음 우선 순위를 갖는다.  첫째, 동적 배열 태그 DT\_RPATH는 콜론(:)으로 구분된 디렉토리 목록을 보유하는 문자열을 제공할 수 있다.  예를 들어, /home/dir/lib:/home.dir2/lib | Pointing to an fuzzed path | - DYN  d\_tag = DT\_RPATH | DT\_RUNPATH  d\_val = XXX |
| dyn11 | 재배치 유형은 변경할 비트와 값을 계산하는 방법을 지정한다. Intel 아키텍처는 Elf32\_Rel 재배치 항목만 사용하며 재배치 할 필드는 가중치를 유지한다. | Change the rellocation types to DT\_RELA for  32-bit files and vice versa | - DYN  d\_tag = DT\_PLTREL  d\_val = DT\_REL | DT\_RELA |
| dyn12 | DT\_PLTGOT 인텔 아키텍처에서 이 항목의 d\_ptr 멤버는 got의 첫번째 항목 주소를 제공한다. 처음 세개의 got 항목은 예약되어 있으며 두개는 프로시저 연결 테이블 정보를 저장하는데 사용된다.  테이블의 항목 0은 \_DYANMIC 심볼로 참조되는 동적 구조의 주소를 보유하도록 예약되어 있다. 이를 통해 동적 링커와 같은 프로그램은 재배치 항목을 아칙 처리하지 않고 자체 동적 구조를 찾을 수 있다. 동적 링커는 다른 프로그램에 의존하지 않고 초기화해야만 메모리 이미지를 재배치해야하기 때문에 특히 중요하다. | Patch the three addresses in GOT to point to  invalid addresses or key addresses in kernelland  $ dissector -s hydra | grep -i \_DYNAMIC  [ 191] 0x080a1f08 ... \_DYNAMIC  $ dissector -a hydra | grep -i got  [22] .got 0x080a1ff0 0x0058ff0  [23] .got.plt 0x080a1ff4 0x0058ff4  [ 13] DT\_PLTGOT 0x080a1ff4  $ objdump -s -j .got.plt hydra | head  Contents of section .got.plt:  80a1ff4 081f0a08 00000000 00000000  The first address is \_DYNAMIC. Not always  modify this one. | - DYN  d\_tag = DT\_PLTGOT  \*(d\_ptr++) = XXX  \*(d\_ptr++) = XXX  \*(d\_ptr) = XXX |
| dyn13 | DT\_JMPREL 이 항목이 있으면 d\_ptr 멤버는 프로시저 연결 테이블(plt)와 연관된 재배치 엔트리의 주소를 보유한다. 이러한 재배치 엔트리를 분리하면 지연 바인딩이 활성화되어 있는 경우 프로세스 초기화 중에 동적 링커가 이를 무시할 수 있다. 이 항목이 있으면 DT\_PLTRELSZ 및 DT\_PLTREL 유형의 관련 항목이 있어야 한다. | Leave the DT\_JMPREL entry and delete  DT\_PLTRELSZ | - DYN  d\_tag = DT\_PLTRELSZ  d\_tag != DT\_PLTRELSZ |
| dyn14 | DT\_RELA 이 요소는 재배치 테이블의 주소를 가지고 있다. 테이블의 항목에는 32비트 파일 클래스의 Elf32\_Rela와 같은 명시적인 가수가 있다. 이 요소가 있으면 동적 구조에 DT\_RELASZ 및 DT\_RELAENT 요소가 있어야한다.  DT\_REL 이 요소는 32비트 파일 클래스의 Elf32\_Rel과 같은 암시적 가수가 테이블에 있다는 점을 제외하면 DT\_RELA와 유사하다. 이 요소가 있으면 동적 구조에도 DT\_RELSZ 및 DT\_RELENT 요소가 있어야한다. | Leave the DT\_RELA entry and delete  DT\_RELASZ and/or DT\_RELAENT  Leave the DT\_REL entry and delete  DT\_RELSZ and DT\_RELENT | - DYN  d\_tag = DT\_RELASZ | DT\_RELAENT  d\_tag != DT\_RELASZ  d\_tag != DT\_RELAENT  d\_tag = DT\_RELSZ | DT\_RELENT  d\_tag != DT\_RELSZ  d\_tag != DT\_RELENT |
| dyn15 | DT\_NULL DT\_NULL 태그가 있는 항목은 \_DYNAMIC 배열의 끝을 표시한다. | Delete the NULL entry | - DYN  d\_tag != DT\_NULL |
| sym1 | Symbol Table  객체 파일의 심볼 테이블은 프로그램의 심볼 정의와 참조를 찾고 재배치하는데 필요한 정보를 담고 있다. 심볼 테이블 인덱스는 이 배열의 서브스크립트이다. 인덱스 0은 테이블의 첫 번째 항목을 지정하며 정의도지 않은 심볼 인덱스로 사용된다.  Name Value  STN UNDEF 0 | First entry different of STN\_UNDEF with  fuzzed values | - SYM  ST[0].st\_name != STN\_UNDEF  ST[0].\* = XXX |
| sym2 | st\_name 이 멤버는 심볼 파일의 문자 표현을 보유하고 있는 객체 파일의 심볼 문자열 테이블에 인덱스를 보유한다. | A st\_name out of bounds | - SYM  st\_name = XXX |
| sym3 | st\_value 이 멤버는 과녈ㄴ 심볼의 값을 제공한다. 컨텍스트에 따라 절대값, 주소ㅓ 등이 될 수 있다. | Invalid address  Some point in kernel-land  High values | - SYM  st\_value = XXX |
| sym4 | st\_size 많은 심볼은 크기가 연관되어 있다. 예를 들어, 데이터 객체의 크기는 객체에 포함된 바이트 수이다. 심볼에 크기가 없거나 알 수 없는 크기가 있으면 이 멤버는 0을 유지한다. | Combination of low and high values | - SYM  st\_size = XXX |
| sym5 | st\_shndx 모든 심볼 테이블 엔트리는 어떤 섹션과 관련하여 정의되며, 이 멤버는 관련 섹션 헤더 테이블 인덱스를 보유한다. | Index to zero and out of bounds  Set to a random but valid index within the  SHT. | - SYM  st\_shndx = 0 - HDR->e\_shnum  st\_shndx = XXX  orcSYM->st\_size = getElf\_Word(); |
| sym6 | STT\_SECTION 심볼은 섹션과 연관된다. 이 유형의 심볼 테이블 항목은 주로 재배치를 위해 존재하며 일반적으로 STB\_LOCAL 바인딩을 가진다. | Change the STB\_LOCAL binding type. | - SYM  ELF\_ST\_TYPE(st\_info) = STT\_SECTION  ELF\_ST\_BIND(st\_info) != STB\_LOCAL |
| sym7 | STT\_FILE 파일 심볼은 STB\_LOCAL 바인딩을 가지며, 섹션 인덱스는 SHN\_ABS이며, 파일에 대한 다른 STB\_LOCAL 심볼보다 앞에 위치한다.(존재할 경우) | Change the STB\_LOCAL binding type and  st\_shndx != SHN\_ABS | - SYM  ELF\_ST\_TYPE(st\_info) = STT\_FILE  ELF\_ST\_BIND(st\_info) != STB\_LOCAL  st\_shndx != SHN\_ABS |
| sym8 | 재배치 가능 파일에서 st\_value는 섹션 인덱스가 SHN\_COMMON인 심볼에 대한 정렬 제한 조건을 유지한다. | For those symbols whose st\_shndx =  SHN\_COMMON, fuzz st\_value with  inconsistent alignment values | - SYM  e\_type = ET\_REL  st\_shndx = SHN\_COMMON  st\_value != 1,2,4,8,16,32,64,128,256,512… |
| sym9 | 재배치 가능 파일에서 st\_value는 정의된 심볼에 대한 섹션 오프셋을 보유한다. 즉, st\_value는 st\_shndx가 식별하는 섹션의 시작 부분에서의 오프셋이다. | For those symbols whose st\_shndx !=  SHN\_COMMON, fuzz st\_value with values  out of bounds | - SYM  e\_type = ET\_REL  st\_shndx != SHN\_COMMON  st\_value = XXX |
| sym10 | 실행 가능 및 공유 오브젝트파일에서 st\_value는 가상 주소를 보유한다. | Invalid address  Some point in kernel-land | - SYM  e\_type = ET\_EXEC | ET\_DYN  st\_value = XXX |
| sym11 | 실행 파일에 연관된 공유 객체 중 하나에 정의된 함수에 대한 참조가 포함되어 있으면 해당 파일의 심볼 테이블 섹션에 해당 심볼에 대한 항목이 포함된다. 해당 심볼 테이블 항목의 st\_stndx 멤버에는 SHN\_UNDEF가 포함된다. 이 함수는 동적 링커에게 해당 함수의 심볼 정의가 실행 파일 자체에 포함되어 있지 않음을 알린다. | Low chances to set st\_shndx to a value  different than SHN\_UNDEF | - SYM  e\_type = ET\_EXEC | ET\_DYN  st\_shndx != SHN\_UNDEF |
| sym12 | 해당 심볼에 실행 파일의 프로시저 연결 테이블(plt)이 할당되고 해당 심볼 테이블 항목의 st\_value 멤버가 0이 아닌 경우 이 값에는 해당 프로시저 연결 테이블 항목의 첫 번째 명령의 가상 주소가 포함된다. 그렇지 않으면 st\_value 멤버에 0이 포함된다. | For those symbols with st\_shndx =  SHN\_UNDEF, set the st\_value to an invalid  pointer or at some point to kernel-land | - SYM  st\_shndx = SHN\_UNDEF  st\_value != 0  st\_value = XXX |
| sym13 | st\_info 이 멤버는 심볼의 유형과 바인딩 속성을 지정한다. 다음 코드는 값을 조작하는 방법을 보여준다.  #define ELF32\_ST\_BIND(i) ((i)>>4)  #define ELF32\_ST\_TYPE(i) ((i)&0xf)  #defineELF32\_ST\_INFO(b,t)(((b)<<4)+((t)&0xf)) | Combination of low and high values Specific combinations to escape from common symbol types and binding types | - SYM  st\_info = XXX |
| rel1 | r\_offset 이 멤버는 재배치 동작을 적용할 위치를 제공한다. 변동 가능 파일의 경우, 값은 섹션의 시작에서 재배치의 영향을 받는 저장 단위까지의 바이트 오프셋이다.  실행 파일 또는 공유 오브젝트의 경우 값은 재배치의 영향을 받는 저장 장치의 가상 주소이다. | In relocatable files r\_offset out of bounds In exec and shared objects, r\_offset will hold invalid addresses and addresses in kernelland | - REL  e\_type = ET\_REL  r\_offset = XXX  e\_type = ET\_EXEC | ET\_DYN  r\_offset = XXX |
| rel2 | r\_info 이 멤버는 재배치되어야하는 심볼 테이블 인덱스와 적용할 재배치 유형을 제공한다. 재배치 유형은 프로세서에 따라 다르다.프로세서 supplement의 텍스트가 재배치 항목의 재배치 유형 또는 심볼 테이블 인덱스를 참조하는 경우, 항목의 r\_info 구성원에 따라 ELF32\_R\_TYPE 또는 ELF32\_R\_SYM을 적용한 결과를 의미한다.  #define ELF32\_R\_SYM(i) ((i)>>8)  #define ELF32\_R\_TYPE(i) ((unsigned char)(i))  #define ELF32\_R\_INFO(s, t) (((s)<<8) + (unsigned char)(t)) | Combination of low and high values Specific combinations to escape from the macros  Make that ELF32\_R\_SYM() returns an invalid section index (its related symbol table) | - REL  r\_info = XXX  ELF\_R\_SYM(r\_info) > e\_shnum |
| rel3 | r\_addend 이 멤버는 재배치 가능 필드에 저장할 값을 계산하는데 사용되는 상수 가산치를 지정한다. | r\_addend is Sword (Signed Word). Combination of high and low values. Negative values | - REL  r\_addend >= 0x80000000  r\_addend = XXX |
| env1 | 프로세스 환경에 null이 아닌 값을 갖는 LD\_BIND\_NOW라는 변수가 들어있는 경우, 동적 링커는 제어를 프로그램에 전송하기 전에 모든 재배치를 처리한다. 예를 들어, 다음의 모든 환경 항목은 이 동작을 지정한다.  -LD\_BIND\_NOW=1  -LD\_BIND\_NOW=on  -LD\_BIND\_NOW=off  그렇지 않으면, LD\_BIND\_NOW가 환경에서 발생하지 않거나 null 값을 가진다. 동적 링커는 프로시저 연결 테이블(plt) 항목을 느리게 평가할 수 있으므로 호출되지 않은 함수에 대한 심볼 확인 및 재배치 오버 헤드가 발생하지 않는다. | Fuzz the environment variable before executing the malformed ELFs or testing the programs | - ENVIRON  export LD\_BIND\_NOW = XXX |
| env2 | 둘째. 프로세서 환경[exec(BA\_OS) 참조]에 있는 LD\_LIBRARY\_PATH라는 변수는 위와 같이 선택적으로 세미클론(;)과 다른 디렉토리 목록 뒤에 디렉토리 목록을 보유할 수 있다.  -LD\_LIBRARY\_PATH=/home/dir/lib:/home/dir2/lib:  -  LD\_LIBRARY\_PATH=/home/dir/lib:/home/dir/lib:  -LD\_LIBRARY\_PATH=/home/dir/lib:/home/dir2/lib:; | Fuzz the environment variable before executing the malformed ELFs or testing the programs | - ENVIRON  export LD\_LIBRARY\_PATH = XXX |
| hash1 | 버킷 배열은 nbucket 항목을 포함하고 체인 배열은 nchain 항목을 포함한다. 인덱스는 0에서 시작한다. | nbucket and nchain with high and low values | - HASH  nbucket = XXX  nchain = XXX |
| hash2 | 버킷과 체인 모두 심볼 테이블 인덱스를 보유한다. 체인 테이블 항목은 심볼 테이블과 유사하다. 심볼 테이블의 항목의 수는 nchain과 동일해야한다. | Take the original indexes and set them to out of bounds values | - HASH  nbucket = nbucket  nchain = nchain  bucket[1..nbucket-1] = XXX  chain[1..nchain-1] = XXX |