

# 04.RELRO

Excuse the ads! We need some help to keep our site up.

## List

- [RELRO](#)
  - [Explanation](#)
  - [Example program](#)
    - [Source code](#)
  - [Check the protection techniques of binary files.](#)
    - [checksec.sh](#)
    - [Program header & Dynamic Section](#)
  - [Overwrite test](#)
    - [No RELRO](#)
    - [Partial RELRO](#)
    - [Full RELRO](#)
  - [Comparison of function calls](#)
    - [Partial RELRO](#)
    - [Full RELRO](#)
  - [How to detect NX in the "Checksec.sh" file](#)
    - [Binary](#)
    - [Process](#)
  - [Related information](#)

## RELRO

### Explanation

- RELRO는 RELocation Read-Only의 줄임말이며, ELF 바이너리 / 프로세스의 데이터 섹션의 보안을 강화하는 일반적인 기술입니다
- RELRO에는 Partial RELRO와 Full RELRO 두 가지 모드가 있습니다.
  - [Partial RELRO](#)
  - [Full RELRO](#)
- 아래와 같이 RELRO, Partial RELRO, Full RELRO의 차이점이 있습니다.
  - 자세한 내용은 아래 "Program header & Dynamic Section"에서 설명합니다.

## Comparison of No RELRO, Partial RELRO, Full RELRO

	No RELRO	Partial RELRO	Full RELRO
Compiler command line	gcc -Wl,-z,norelro	gcc -Wl,-z,relro	gcc -Wl,-z,relro,-z,now
Lazy binding	O	O	X
Now binding	X	X	O
Write to GOT	O	O	X
RELRO in the Program header	X	O	O
Section include in RELRO	X	INIT_ARRAY, FINI_ARRAY	INIT_ARRAY, FINI_ARRAY, PLTGOT
PLTRELSZ include in Dynamic Section	O	O	X
PLTREL include in Dynamic Section	O	O	X
JMPREL include in Dynamic Section	O	O	X
BIND_NOW include in Dynamic Section	X	X	O

## Example program

### Source code

#### RELRO.c

```
#include <stdio.h>
#include <string.h>

void main(){

    char address[16];
    size_t *pointer;
    int count = 1;

    while(count != 100)
    {
        printf("----- %d -----\n",count);
        memset(address,0,16);
        printf("Input Pointer address : ");
        fgets(address,16,stdin);

        pointer = strtol(address,0,16);
        printf("Pointer address : %p\n",pointer);

        printf("Input Pointer text : ");
        fgets(pointer,16,stdin);
        printf("Pointer text : %s\n",pointer);
        count++;
    }
    scanf("%s",address);
}
```

Check the protection techniques of binary files.

checksec.sh

No RELRO	<pre>gcc -Wl,-z,norelro -o RELRO-NoRelro RELRO.c  lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO\$ checksec.sh --file ./RELRO-NoRelro RELRO          STACK CANARY    NX          PIE          RPATH        RUNPATH      FILE No RELRO      Canary found    NX enabled  No PIE       No RPATH     No RUNPATH   ./RELRO- NoRelro lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO\$</pre>
Partial RELRO	<pre>gcc -Wl,-z,relro -o RELRO-Relro RELRO.c  lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO\$ checksec.sh --file ./RELRO-Relro RELRO          STACK CANARY    NX          PIE          RPATH        RUNPATH      FILE Partial RELRO  Canary found    NX enabled  No PIE       No RPATH     No RUNPATH   ./RELRO- Relro lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO\$</pre>
Full RELRO	<pre>gcc -Wl,-z,relro,-z,now -o RELRO-FullRelro RELRO.c  lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO\$ checksec.sh --file ./RELRO-FullRelro RELRO          STACK CANARY    NX          PIE          RPATH        RUNPATH      FILE Full RELRO     Canary found    NX enabled  No PIE       No RPATH     No RUNPATH   ./RELRO- FullRelro lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO\$</pre>

## Program header & Dynamic Section

- 다음과 같이 RELRO 적용시 "Program Header"와 "Dynamic Section"의 변화를 확인할 수 있습니다.
  - Partial RELRO를 적용하게되면 다음과 같은 변화가 발생합니다.
    - 'Program Header'에 'RELRO' 영역이 생성됩니다.
      - 해당 영역의 권한은 Read only 입니다.
    - 해당 영역에 포함되는 Section은 다음과 같습니다.
      - INIT\_ARRAY, FINI\_ARRAY
    - 즉, GOT영역을 덮어쓸수 있습니다.
  - Full RELRO를 적용하게되면 다음과 같은 변화가 발생합니다.
    - 'Program Header'에 'RELRO' 영역이 생성됩니다.
      - 해당 영역의 권한은 Read only 입니다.
    - 해당 영역에 포함되는 Section은 다음과 같습니다.
      - INIT\_ARRAY, FINI\_ARRAY, PLTGOT
    - 그리고 Section영역에서 PLTRELSZ, PLTREL, JMPREL가 제거되고, 'BIND\_NOW', 'FLAGS\_1' Section이 추가됩니다.
    - 즉, GOT영역을 덮어쓸수 없습니다.

Program header & Dynamic Section		
No RELRO	Partial RELRO	Full RELRO
<pre>./RELRO-NoRelro:      file format elf64-x86-64  Program Header:   PHDR off  0x0000000000000040 vaddr 0x000000000400040 paddr 0x000000000400040 align 2**3       filesz 0x00000000000001c0 memsz 0x00000000000001c0 flags r-x   INTERP off  0x0000000000000200 vaddr 0x000000000400200 paddr 0x000000000400200 align 2**0       filesz 0x000000000000001c</pre>	<pre>./RELRO-Relro:      file format elf64-x86-64  Program Header:   PHDR off  0x0000000000000040 vaddr 0x000000000400040 paddr 0x000000000400040 align 2**3       filesz 0x00000000000001f8 memsz 0x00000000000001f8 flags r-x   INTERP off  0x0000000000000238 vaddr 0x000000000400238 paddr 0x000000000400238 align 2**0       filesz 0x000000000000001c</pre>	<pre>./RELRO-FullRelro:  file format elf64-x86-64  Program Header:   PHDR off  0x0000000000000040 vaddr 0x000000000400040 paddr 0x000000000400040 align 2**3       filesz 0x00000000000001f8 memsz 0x00000000000001f8 flags r-x   INTERP off  0x0000000000000238 vaddr 0x000000000400238 paddr 0x000000000400238 align 2**0       filesz 0x000000000000001c</pre>

```
memsz 0x000000000000001c flags r--
  LOAD off 0x0000000000000000
vaddr 0x0000000000400000 paddr
0x0000000000400000 align 2**21
  filesz 0x00000000000000a2c
memsz 0x00000000000000a2c flags r-x
  LOAD off 0x00000000000000a30
vaddr 0x0000000000600a30 paddr
0x0000000000600a30 align 2**21
  filesz 0x00000000000000250
memsz 0x00000000000000260 flags rw-
DYNAMIC off 0x00000000000000a48
vaddr 0x0000000000600a48 paddr
0x0000000000600a48 align 2**3
  filesz 0x000000000000001d0
memsz 0x000000000000001d0 flags rw-
NOTE off 0x0000000000000021c
vaddr 0x000000000040021c paddr
0x000000000040021c align 2**2
  filesz 0x00000000000000044
memsz 0x00000000000000044 flags r--
EH_FRAME off 0x00000000000000900
vaddr 0x0000000000400900 paddr
0x0000000000400900 align 2**2
  filesz 0x00000000000000034
memsz 0x00000000000000034 flags r--
STACK off 0x00000000000000000
vaddr 0x0000000000000000 paddr
0x0000000000000000 align 2**4
  filesz 0x00000000000000000
memsz 0x0000000000000000 flags rw-
```

Dynamic Section:

```
NEEDED libc.so.6
INIT
0x0000000000400538
FINI
0x0000000000400884
INIT_ARRAY
0x0000000000600a30
INIT_ARRAYSZ
0x0000000000000008
FINI_ARRAY
0x0000000000600a38
FINI_ARRAYSZ
0x0000000000000008
GNU_HASH
0x0000000000400260
STRTAB
0x0000000000400378
SYMTAB
0x0000000000400288
STRSZ
0x000000000000008d
SYMENT
0x0000000000000018
DEBUG
0x0000000000000000
PLTGOT
0x0000000000600c20
PLTRELSZ
0x00000000000000a8
PLTREL
0x0000000000000007
JMPREL
0x0000000000400490
RELA
0x0000000000400460
RELASZ
0x0000000000000030
RELAENT
0x0000000000000018
VERNEED
0x0000000000400420
VERNEEDNUM
0x0000000000000001
VERSYM
0x0000000000400406
```

Version References:

```
required from libc.so.6:
  0x0d696917 0x00 04 GLIBC_2.7
  0x0d696914 0x00 03 GLIBC_2.4
  0x09691a75 0x00 02 GLIBC_2.2.5
```

```
memsz 0x000000000000001c flags r--
  LOAD off 0x0000000000000000
vaddr 0x0000000000400000 paddr
0x0000000000400000 align 2**21
  filesz 0x00000000000000a5c
memsz 0x00000000000000a5c flags r-x
  LOAD off 0x00000000000000e10
vaddr 0x0000000000600e10 paddr
0x0000000000600e10 align 2**21
  filesz 0x00000000000000250
memsz 0x00000000000000260 flags rw-
DYNAMIC off 0x00000000000000e28
vaddr 0x0000000000600e28 paddr
0x0000000000600e28 align 2**3
  filesz 0x000000000000001d0
memsz 0x000000000000001d0 flags rw-
NOTE off 0x00000000000000254
vaddr 0x0000000000400254 paddr
0x0000000000400254 align 2**2
  filesz 0x00000000000000044
memsz 0x00000000000000044 flags r--
EH_FRAME off 0x00000000000000930
vaddr 0x0000000000400930 paddr
0x0000000000400930 align 2**2
  filesz 0x00000000000000034
memsz 0x00000000000000034 flags r--
STACK off 0x00000000000000000
vaddr 0x0000000000000000 paddr
0x0000000000000000 align 2**4
  filesz 0x00000000000000000
memsz 0x0000000000000000 flags rw-
RELRO off 0x00000000000000e10
vaddr 0x0000000000600e10 paddr
0x0000000000600e10 align 2**0
  filesz 0x000000000000001f0
memsz 0x000000000000001f0 flags r--
```

Dynamic Section:

```
NEEDED libc.so.6
INIT
0x0000000000400570
FINI
0x00000000004008b4
INIT_ARRAY
0x0000000000600e10
INIT_ARRAYSZ
0x0000000000000008
FINI_ARRAY
0x0000000000600e18
FINI_ARRAYSZ
0x0000000000000008
GNU_HASH
0x0000000000400298
STRTAB
0x00000000004003b0
SYMTAB
0x00000000004002c0
STRSZ
0x000000000000008d
SYMENT
0x0000000000000018
DEBUG
0x0000000000000000
PLTGOT
0x0000000000601000
PLTRELSZ
0x00000000000000a8
PLTREL
0x0000000000000007
JMPREL
0x00000000004004c8
RELA
0x0000000000400498
RELASZ
0x0000000000000030
RELAENT
0x0000000000000018
VERNEED
0x0000000000400458
VERNEEDNUM
0x0000000000000001
VERSYM
0x000000000040043e
```

```
memsz 0x000000000000001c flags r--
  LOAD off 0x0000000000000000
vaddr 0x0000000000400000 paddr
0x0000000000400000 align 2**21
  filesz 0x00000000000000a3c
memsz 0x00000000000000a3c flags r-x
  LOAD off 0x00000000000000dd0
vaddr 0x0000000000600dd0 paddr
0x0000000000600dd0 align 2**21
  filesz 0x00000000000000240
memsz 0x00000000000000250 flags rw-
DYNAMIC off 0x00000000000000de8
vaddr 0x0000000000600de8 paddr
0x0000000000600de8 align 2**3
  filesz 0x000000000000001c0
memsz 0x000000000000001c0 flags rw-
NOTE off 0x00000000000000254
vaddr 0x0000000000400254 paddr
0x0000000000400254 align 2**2
  filesz 0x00000000000000044
memsz 0x00000000000000044 flags r--
EH_FRAME off 0x00000000000000910
vaddr 0x0000000000400910 paddr
0x0000000000400910 align 2**2
  filesz 0x00000000000000034
memsz 0x00000000000000034 flags r--
STACK off 0x00000000000000000
vaddr 0x0000000000000000 paddr
0x0000000000000000 align 2**4
  filesz 0x00000000000000000
memsz 0x0000000000000000 flags rw-
RELRO off 0x00000000000000dd0
vaddr 0x0000000000600dd0 paddr
0x0000000000600dd0 align 2**0
  filesz 0x00000000000000230
memsz 0x00000000000000230 flags r--
```

Dynamic Section:

```
NEEDED libc.so.6
INIT
0x0000000000400590
FINI
0x0000000000400894
INIT_ARRAY
0x0000000000600dd0
INIT_ARRAYSZ
0x0000000000000008
FINI_ARRAY
0x0000000000600dd8
FINI_ARRAYSZ
0x0000000000000008
GNU_HASH
0x0000000000400298
STRTAB
0x00000000004003d0
SYMTAB
0x00000000004002e0
STRSZ
0x000000000000008d
SYMENT
0x0000000000000018
DEBUG
0x0000000000000000
PLTGOT
0x0000000000600fa8
RELA
0x00000000004004b8
RELASZ
0x00000000000000d8
RELAENT
0x0000000000000018
BIND_NOW
0x0000000000000000
FLAGS_1
0x0000000000000001
VERNEED
0x0000000000400478
VERNEEDNUM
0x0000000000000001
VERSYM
0x000000000040045e
```

Version References:

```
required from libc.so.6:
```

```
Version References:
required from libc.so.6:
0x0d696917 0x00 04 GLIBC_2.7
0x0d696914 0x00 03 GLIBC_2.4
0x09691a75 0x00 02 GLIBC_2.2.5
```

```
0x0d696917 0x00 04 GLIBC_2.7
0x0d696914 0x00 03 GLIBC_2.4
0x09691a75 0x00 02 GLIBC_2.2.5
```

## Overwrite test

### No RELRO

- "\_\_isoc99\_scanf"의 GOT Address는 0x600c68이며, 아래와 같이 해당 영역에 값을 변경할 수 있습니다.

```
lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO$ gdb -q ./RELRO-NoRelro
Reading symbols from ./RELRO-NoRelro...(no debugging symbols found)...done.
gdb-peda$ elfsymbol __isoc99_scanf
Detail symbol info
__isoc99_scanf@reloc = 0x6
__isoc99_scanf@plt = 0x4005d0
__isoc99_scanf@got = 0x600c68
gdb-peda$ x/gx 0x600c68
0x600c68:          0x00000000004005d6
gdb-peda$ r
Starting program: /home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-NoRelro
----- 1 -----
Input Pointer address : 600c68
Pointer address : 0x600c68
Input Pointer text : AAAA
Pointer text : AAAA

----- 2 -----
Input Pointer address : ^C
Program received signal SIGINT, Interrupt.

gdb-peda$ x/gx 0x600c68
0x600c68:          0x0000000a41414141
gdb-peda$
```

- 다음과 같이 프로그램 헤더 정보와 메모리 맵을 통해 조금더 자세한 내용을 확인할 수 있습니다.
  - "\_\_isoc99\_scanf"의 주소값은 '.got.plt' 영역에 저장되어 있습니다.
    - '.got.plt' 영역의 시작 주소는 0x600c20 입니다.
  - 메모리 맵을 통해 해당 영역(0x00600000 ~ 0x00601000)에 'W' 쓰기 권한이 설정되어 있습니다.

## Check permission for memory

```
gdb-peda$ elfheader
.interp = 0x400200
.note.ABI-tag = 0x40021c
.note.gnu.build-id = 0x40023c
.gnu.hash = 0x400260
.dynsym = 0x400288
.dynstr = 0x400378
.gnu.version = 0x400406
.gnu.version_r = 0x400420
.rela.dyn = 0x400460
.rela.plt = 0x400490
.init = 0x400538
.plt = 0x400560
.plt.got = 0x4005e0
.text = 0x4005f0
.fini = 0x400884
.rodata = 0x400890
.eh_frame_hdr = 0x400900
.eh_frame = 0x400938
.init_array = 0x600a30
.fini_array = 0x600a38
.jcr = 0x600a40
.dynamic = 0x600a48
.got = 0x600c18
.got.plt = 0x600c20
.data = 0x600c70
.bss = 0x600c80
gdb-peda$ vmmmap
Start                End                  Perm                Name
0x00400000           0x00401000          r-xp                /home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-
NoRelro
0x00600000           0x00601000          rw-p                /home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-
NoRelro
0x00601000           0x00622000          rw-p                [heap]
0x00007ffff7a0d000  0x00007ffff7bcd000 r-xp                /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7bcd000  0x00007ffff7dcd000 ---p                /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dcd000  0x00007ffff7dd1000 r--p                /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dd1000  0x00007ffff7dd3000 rw-p                /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dd3000  0x00007ffff7dd7000 rw-p                mapped
0x00007ffff7dd7000  0x00007ffff7dfd000 r-xp                /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7dfd000  0x00007ffff7fdc000 rw-p                mapped
0x00007ffff7ffd000  0x00007ffff7ff8000 rw-p                mapped
0x00007ffff7ff8000  0x00007ffff7ffa000 r--p                [vvar]
0x00007ffff7ffa000  0x00007ffff7ffc000 r-xp                [vdso]
0x00007ffff7ffc000  0x00007ffff7ffd000 r--p                /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7ffd000  0x00007ffff7ffe000 rw-p                /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7ffe000  0x00007ffff7fff000 rw-p                mapped
0x00007ffff7fff000  0x00007ffff7fff000 rw-p                [stack]
0xffffffffff600000  0xffffffffff601000 r-xp                [vsyscall]
gdb-peda$
```

## Partial RELRO

- `__isoc99_scanf`의 GOT Address는 `0x601048`이며, 아래와 같이 해당 영역에 값을 변경할 수 있습니다.

```

lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO$ gdb -q ./RELRO-Relro
Reading symbols from ./RELRO-Relro...(no debugging symbols found)...done.
gdb-peda$ elfsymbol __isoc99_scanf
Detail symbol info
__isoc99_scanf@reloc = 0x6
__isoc99_scanf@plt = 0x400600
__isoc99_scanf@got = 0x601048
gdb-peda$ x/gx 0x601048
0x601048:      0x0000000000400606
gdb-peda$ r
Starting program: /home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-Relro
----- 1 -----
Input Pointer address : 601048
Pointer address : 0x601048
Input Pointer text : AAAA
Pointer text : AAAA

----- 2 -----
Input Pointer address : ^C
Program received signal SIGINT, Interrupt.

gdb-peda$ x/gx 0x601048
0x601048:      0x0000000a41414141
gdb-peda$

```

- 다음과 같이 프로그램 헤더 정보와 메모리 맵을 통해 조금더 자세한 내용을 확인할 수 있습니다.
  - '.got.plt' 영역의 시작 주소는 0x601000 입니다.
  - 메모리 맵에서 RELRO가 적용되지 않은 프로그램과 다른 부분을 확인할 수 있습니다.
    - 0x600000 ~ 0x601000 영역의 권한은 r--p 입니다.
      - 해당 영역에는 .init\_array, .fini\_array, .jcr, .dynamic, .got 헤더가 포함됩니다.
    - 0x601000 ~ 0x602000 영역의 권한은 rw-p 입니다.
      - 해당 영역에는 .got.plt, 등의 헤더가 포함됩니다.
    - 즉, 이로 인해 .got.plt 영역에 값을 변경할 수 있습니다.

## Check permission for memory

```
gdb-peda$ elfheader
.interp = 0x400238
.note.ABI-tag = 0x400254
.note.gnu.build-id = 0x400274
.gnu.hash = 0x400298
.dynsym = 0x4002c0
.dynstr = 0x4003b0
.gnu.version = 0x40043e
.gnu.version_r = 0x400458
.rela.dyn = 0x400498
.rela.plt = 0x4004c8
.init = 0x400570
.plt = 0x400590
.plt.got = 0x400610
.text = 0x400620
.fini = 0x4008b4
.rodata = 0x4008c0
.eh_frame_hdr = 0x400930
.eh_frame = 0x400968
.init_array = 0x600e10
.fini_array = 0x600e18
.jcr = 0x600e20
.dynamic = 0x600e28
.got = 0x600ff8
.got.plt = 0x601000
.data = 0x601050
.bss = 0x601060
gdb-peda$ vmmmap
Start          End            Perm          Name
0x00400000     0x00401000    r-xp         /home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-Relro
0x00600000     0x00601000    r--p         /home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-Relro
0x00601000     0x00602000    rw-p         /home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-Relro
0x00602000     0x00623000    rw-p         [heap]
0x00007ffff7a0d000 0x00007ffff7bcd000 r-xp         /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7bcd000 0x00007ffff7dcd000 ---p         /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dcd000 0x00007ffff7dd1000 r--p         /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dd1000 0x00007ffff7dd3000 rw-p         /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dd3000 0x00007ffff7dd7000 rw-p         mapped
0x00007ffff7dd7000 0x00007ffff7dfd000 r-xp         /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7fd9000 0x00007ffff7fdc000 rw-p         mapped
0x00007ffff7ff6000 0x00007ffff7ff8000 rw-p         mapped
0x00007ffff7ffa000 0x00007ffff7ffa000 r--p         [vvar]
0x00007ffff7ffa000 0x00007ffff7ffc000 r-xp         [vdso]
0x00007ffff7ffc000 0x00007ffff7ffd000 r--p         /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7ffd000 0x00007ffff7ffe000 rw-p         /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7ffe000 0x00007ffff7fff000 rw-p         mapped
0x00007ffff7fff000 0x00007ffff7fff000 rw-p         [stack]
0xffffffffff600000 0xffffffffff601000 r-xp         [syscall]
gdb-peda$
```

## Full RELRO

- 앞에서 테스트한 프로그램과 달리 GOT 영역에 값을 변경할 수 없습니다.
  - 디버거에서 '\_\_isoc99\_scanf'의 심볼 정보를 찾을 수 없습니다.
- 디스어셈블 코드에서 호출되는 함수를 분석해보겠습니다.
  - 0x4007fd 영역의 코드에서 0x4005f8 영역을 호출합니다.
  - 0x4005f8 영역의 코드에서 "rip+0x2009fa" 영역에 저장된 주소로 이동합니다.
  - "rip+0x2009fa" 영역은 0x600ff8 이며, 해당 영역에 저장된 값은 0x00007ffff7a784d0 입니다.
  - 0x00007ffff7a784d0 영역은 \_\_isoc99\_scanf 함수의 시작 주소 입니다.

```
lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO$ gdb -q ./RELRO-FullRelro
Reading symbols from ./RELRO-FullRelro...(no debugging symbols found)...done.
gdb-peda$ elfsymbol __isoc99_scanf
```



'\_\_isoc99\_scanf': no match found

gdb-peda\$ disassemble main

Dump of assembler code for function main:

```
0x0000000004006f6 <+0>:      push  rbp
0x0000000004006f7 <+1>:      mov   rbp, rsp
0x0000000004006fa <+4>:      sub   rsp, 0x30
0x0000000004006fe <+8>:      mov   rax, QWORD PTR fs:0x28
0x000000000400707 <+17>:     mov   QWORD PTR [rbp-0x8], rax
0x00000000040070b <+21>:     xor   eax, eax
0x00000000040070d <+23>:     mov   DWORD PTR [rbp-0x2c], 0x1
0x000000000400714 <+30>:     jmp   0x4007e2 <main+236>
0x000000000400719 <+35>:     mov   eax, DWORD PTR [rbp-0x2c]
0x00000000040071c <+38>:     mov   esi, eax
0x00000000040071e <+40>:     mov   edi, 0x4008a4
0x000000000400723 <+45>:     mov   eax, 0x0
0x000000000400728 <+50>:     call  0x4005c8
0x00000000040072d <+55>:     lea   rax, [rbp-0x20]
0x000000000400731 <+59>:     mov   edx, 0x10
0x000000000400736 <+64>:     mov   esi, 0x0
0x00000000040073b <+69>:     mov   rdi, rax
0x00000000040073e <+72>:     call  0x4005d0
0x000000000400743 <+77>:     mov   edi, 0x4008b4
0x000000000400748 <+82>:     mov   eax, 0x0
0x00000000040074d <+87>:     call  0x4005c8
0x000000000400752 <+92>:     mov   rdx, QWORD PTR [rip+0x2008b7]      # 0x601010 <stdin@GLIBC_2.2.5>
0x000000000400759 <+99>:     lea   rax, [rbp-0x20]
0x00000000040075d <+103>:    mov   esi, 0x10
0x000000000400762 <+108>:    mov   rdi, rax
0x000000000400765 <+111>:    call  0x4005e0
0x00000000040076a <+116>:    lea   rax, [rbp-0x20]
0x00000000040076e <+120>:    mov   edx, 0x10
0x000000000400773 <+125>:    mov   esi, 0x0
0x000000000400778 <+130>:    mov   rdi, rax
0x00000000040077b <+133>:    mov   eax, 0x0
0x000000000400780 <+138>:    call  0x4005f0
0x000000000400785 <+143>:    cdq   rax
0x000000000400787 <+145>:    mov   QWORD PTR [rbp-0x28], rax
0x00000000040078b <+149>:    mov   rax, QWORD PTR [rbp-0x28]
0x00000000040078f <+153>:    mov   rsi, rax
0x000000000400792 <+156>:    mov   edi, 0x4008cd
0x000000000400797 <+161>:    mov   eax, 0x0
0x00000000040079c <+166>:    call  0x4005c8
0x0000000004007a1 <+171>:    mov   edi, 0x4008e3
0x0000000004007a6 <+176>:    mov   eax, 0x0
0x0000000004007ab <+181>:    call  0x4005c8
0x0000000004007b0 <+186>:    mov   rdx, QWORD PTR [rip+0x200859]      # 0x601010 <stdin@GLIBC_2.2.5>
0x0000000004007b7 <+193>:    mov   rax, QWORD PTR [rbp-0x28]
0x0000000004007bb <+197>:    mov   esi, 0x10
0x0000000004007c0 <+202>:    mov   rdi, rax
0x0000000004007c3 <+205>:    call  0x4005e0
0x0000000004007c8 <+210>:    mov   rax, QWORD PTR [rbp-0x28]
0x0000000004007cc <+214>:    mov   rsi, rax
0x0000000004007cf <+217>:    mov   edi, 0x4008f9
0x0000000004007d4 <+222>:    mov   eax, 0x0
0x0000000004007d9 <+227>:    call  0x4005c8
0x0000000004007de <+232>:    add   DWORD PTR [rbp-0x2c], 0x1
0x0000000004007e2 <+236>:    cmp   DWORD PTR [rbp-0x2c], 0x64
0x0000000004007e6 <+240>:    jne   0x400719 <main+35>
0x0000000004007ec <+246>:    lea   rax, [rbp-0x20]
0x0000000004007f0 <+250>:    mov   rsi, rax
0x0000000004007f3 <+253>:    mov   edi, 0x40090c
0x0000000004007f8 <+258>:    mov   eax, 0x0
0x0000000004007fd <+263>:    call  0x4005f8
0x000000000400802 <+268>:    nop
0x000000000400803 <+269>:    mov   rax, QWORD PTR [rbp-0x8]
0x000000000400807 <+273>:    xor   rax, QWORD PTR fs:0x28
0x000000000400810 <+282>:    je    0x400817 <main+289>
0x000000000400812 <+284>:    call  0x4005c0
0x000000000400817 <+289>:    leave
0x000000000400818 <+290>:    ret
```

End of assembler dump.

```

gdb-peda$ r
Starting program: /home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-FullRelro
----- 1 -----
Input Pointer address : ^C
Program received signal SIGINT, Interrupt.

gdb-peda$ x/i 0x4005f8
0x4005f8:      jmp     QWORD PTR [rip+0x2009fa]    # 0x600ff8
gdb-peda$ x/gx 0x600ff8
0x600ff8:      0x00007ffff7a784d0
gdb-peda$ x/5i 0x00007ffff7a784d0
0x7ffff7a784d0 <__isoc99_scanf>:      push    rbx
0x7ffff7a784d1 <__isoc99_scanf+1>:      mov     r10,rdi
0x7ffff7a784d4 <__isoc99_scanf+4>:      sub     rsp,0xd0
0x7ffff7a784db <__isoc99_scanf+11>:     test   al,al
0x7ffff7a784dd <__isoc99_scanf+13>:     mov     QWORD PTR [rsp+0x28],rsi
gdb-peda$

```

- 해당 프로그램의 헤더 구성이 No RELRO, Partial RELRO 와 다릅니다.
  - 해당 프로그램의 헤더 정보에 '.rela.plt', '.got.plt' 헤더가 존재하지 않습니다.

## Check permission for memory

```
gdb-peda$ elfheader
.interp = 0x400238
.note.ABI-tag = 0x400254
.note.gnu.build-id = 0x400274
.gnu.hash = 0x400298
.dynsym = 0x4002e0
.dynstr = 0x4003d0
.gnu.version = 0x40045e
.gnu.version_r = 0x400478
.rela.dyn = 0x4004b8
.init = 0x400590
.plt = 0x4005b0
.plt.got = 0x4005c0
.text = 0x400600
.fini = 0x400894
.rodata = 0x4008a0
.eh_frame_hdr = 0x400910
.eh_frame = 0x400948
.init_array = 0x600dd0
.fini_array = 0x600dd8
.jcr = 0x600de0
.dynamic = 0x600de8
.got = 0x600fa8
.data = 0x601000
.bss = 0x601010
gdb-peda$ vmmmap
Start      End          Perm        Name
0x00400000 0x00401000  r-xp       /home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-
FullRelro
0x00600000 0x00601000  r--p       /home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-
FullRelro
0x00601000 0x00602000  rw-p       /home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-
FullRelro
0x00602000 0x00623000  rw-p       [heap]
0x00007ffff7a0d000 0x00007ffff7bcd000 r-xp       /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7bcd000 0x00007ffff7dcd000 ---p       /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dcd000 0x00007ffff7dd1000 r--p       /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dd1000 0x00007ffff7dd3000 rw-p       /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dd3000 0x00007ffff7dd7000 rw-p       mapped
0x00007ffff7dd7000 0x00007ffff7dfd000 r-xp       /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7dfd000 0x00007ffff7fdc000 rw-p       mapped
0x00007ffff7ffd000 0x00007ffff7ffe000 rw-p       mapped
0x00007ffff7ffe000 0x00007ffff7ffa000 r--p       [vvar]
0x00007ffff7ffa000 0x00007ffff7ffc000 r-xp       [vdso]
0x00007ffff7ffc000 0x00007ffff7ffd000 r--p       /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7ffd000 0x00007ffff7ffe000 rw-p       /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7ffe000 0x00007ffff7fff000 rw-p       mapped
0x00007ffff7fff000 0x00007ffff7fff000 rw-p       [stack]
0xffffffffffff60000 0xffffffffffff601000 r-xp       [vsyscall]
gdb-peda$
```

## Comparison of function calls

### Partial RELRO

- 다음과 같이 동적 라이브러리의 주소가 호출 됩니다.
  - main 함수에서 printf 함수를 사용하기 위해 메모리 주소 0x4005b0을 호출합니다.
    - 메모리 주소 0x4005b0는 ".plt" 영역입니다.
    - ".plt" 영역은 0x400590 ~ 0x400610 입니다.
  - 0x4005b0 영역의 코드는 "jmp QWORD PTR [rip+0x200a6a]" 입니다.
    - 즉, 메모리 주소 0x601020에 저장된 주소로 JUMP 합니다.
      - 메모리 주소 0x601020은 ".got.plt" 영역입니다.

- ".got.plt" 영역은 0x601000 ~ 0x601050 입니다.
- 메모리 주소 0x601020에 저장된 값은 동적 라이브러리의 주소가 아닌 '.plt' 영역입니다.
  - 이는 해당 프로그램에서 printf 함수가 호출되지 않았기 때문에 Stub 코드("printf@plt+6")의 주소 값이 저장되어 있습니다.
- 프로그램을 실행하고 printf 함수가 호출되기 시작하면 메모리 주소 0x601020(".got.plt" 영역) 영역에 동적라이브러리의 printf 함수의 시작 주소 값이 저장됩니다.
- 즉, Partial RELRO가 적용된 바이너리는 ".got.plt"영역이 Write가 가능하도록 설정되어 있기 때문에 ".got.plt" 영역에 저장된 값을 변경할 수 있습니다.

### function call of "Partial RELRO"

```
lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO$ gdb -q ./RELRO-Relro
gdb-peda$ disassemble main
Dump of assembler code for function main:
   0x000000000400716 <+0>:      push   rbp
   0x000000000400717 <+1>:      mov    rbp, rsp
   0x00000000040071a <+4>:      sub   rsp, 0x30
   0x00000000040071e <+8>:      mov   rax, QWORD PTR fs:0x28
   0x000000000400727 <+17>:     mov   QWORD PTR [rbp-0x8], rax
   0x00000000040072b <+21>:     xor   eax, eax
   0x00000000040072d <+23>:     mov   DWORD PTR [rbp-0x2c], 0x1
   0x000000000400734 <+30>:     jmp   0x400802 <main+236>
   0x000000000400739 <+35>:     mov   eax, DWORD PTR [rbp-0x2c]
   0x00000000040073c <+38>:     mov   esi, eax
   0x00000000040073e <+40>:     mov   edi, 0x4008c4
   0x000000000400743 <+45>:     mov   eax, 0x0
   0x000000000400748 <+50>:     call  0x4005b0 <printf@plt>
   0x00000000040074d <+55>:     lea  rax, [rbp-0x20]
   0x000000000400751 <+59>:     mov   edx, 0x10
   0x000000000400756 <+64>:     mov   esi, 0x0
   0x00000000040075b <+69>:     mov   rdi, rax
   0x00000000040075e <+72>:     call  0x4005c0 <memset@plt>
   0x000000000400763 <+77>:     mov   edi, 0x4008d4
   0x000000000400768 <+82>:     mov   eax, 0x0
   0x00000000040076d <+87>:     call  0x4005b0 <printf@plt>
   0x000000000400772 <+92>:     mov   rdx, QWORD PTR [rip+0x2008e7]          # 0x601060 <stdin@@GLIBC_2.2.5>
   0x000000000400779 <+99>:     lea  rax, [rbp-0x20]
   0x00000000040077d <+103>:    mov   esi, 0x10
   0x000000000400782 <+108>:    mov   rdi, rax
   0x000000000400785 <+111>:    call  0x4005e0 <fgets@plt>
   0x00000000040078a <+116>:    lea  rax, [rbp-0x20]
   0x00000000040078e <+120>:    mov   edx, 0x10
   0x000000000400793 <+125>:    mov   esi, 0x0
   0x000000000400798 <+130>:    mov   rdi, rax
   0x00000000040079b <+133>:    mov   eax, 0x0
   0x0000000004007a0 <+138>:    call  0x4005f0 <strtol@plt>
   0x0000000004007a5 <+143>:    cdq   rax
   0x0000000004007a7 <+145>:    mov   QWORD PTR [rbp-0x28], rax
   0x0000000004007ab <+149>:    mov   rax, QWORD PTR [rbp-0x28]
   0x0000000004007af <+153>:    mov   rsi, rax
   0x0000000004007b2 <+156>:    mov   edi, 0x4008ed
   0x0000000004007b7 <+161>:    mov   eax, 0x0
   0x0000000004007bc <+166>:    call  0x4005b0 <printf@plt>
   0x0000000004007c1 <+171>:    mov   edi, 0x400903
   0x0000000004007c6 <+176>:    mov   eax, 0x0
   0x0000000004007cb <+181>:    call  0x4005b0 <printf@plt>
   0x0000000004007d0 <+186>:    mov   rdx, QWORD PTR [rip+0x200889]          # 0x601060 <stdin@@GLIBC_2.2.5>
   0x0000000004007d7 <+193>:    mov   rax, QWORD PTR [rbp-0x28]
   0x0000000004007db <+197>:    mov   esi, 0x10
   0x0000000004007e0 <+202>:    mov   rdi, rax
   0x0000000004007e3 <+205>:    call  0x4005e0 <fgets@plt>
   0x0000000004007e8 <+210>:    mov   rax, QWORD PTR [rbp-0x28]
   0x0000000004007ec <+214>:    mov   rsi, rax
   0x0000000004007ef <+217>:    mov   edi, 0x400919
   0x0000000004007f4 <+222>:    mov   eax, 0x0
   0x0000000004007f9 <+227>:    call  0x4005b0 <printf@plt>
   0x0000000004007fe <+232>:    add   DWORD PTR [rbp-0x2c], 0x1
   0x000000000400802 <+236>:    cmp   DWORD PTR [rbp-0x2c], 0x64
   0x000000000400806 <+240>:    jne  0x400739 <main+35>
   0x00000000040080c <+246>:    lea  rax, [rbp-0x20]
   0x000000000400810 <+250>:    mov   rsi, rax
```

```

0x000000000400813 <+253>:      mov     edi,0x40092c
0x000000000400818 <+258>:      mov     eax,0x0
0x00000000040081d <+263>:      call   0x400600 <__isoc99_scanf@plt>
0x000000000400822 <+268>:      nop
0x000000000400823 <+269>:      mov     rax,QWORD PTR [rbp-0x8]
0x000000000400827 <+273>:      xor     rax,QWORD PTR fs:0x28
0x000000000400830 <+282>:      je     0x400837 <main+289>
0x000000000400832 <+284>:      call   0x4005a0 <__stack_chk_fail@plt>
0x000000000400837 <+289>:      leave
0x000000000400838 <+290>:      ret

```

End of assembler dump.

Reading symbols from ./RELRO-Relro...(no debugging symbols found)...done.

gdb-peda\$ elfheader

```

.interp = 0x400238
.note.ABI-tag = 0x400254
.note.gnu.build-id = 0x400274
.gnu.hash = 0x400298
.dynsym = 0x4002c0
.dynstr = 0x4003b0
.gnu.version = 0x40043e
.gnu.version_r = 0x400458
.rela.dyn = 0x400498
.rela.plt = 0x4004c8
.init = 0x400570
.plt = 0x400590
.plt.got = 0x400610
.text = 0x400620
.fini = 0x4008b4
.rodata = 0x4008c0
.eh_frame_hdr = 0x400930
.eh_frame = 0x400968
.init_array = 0x600e10
.fini_array = 0x600e18
.jcr = 0x600e20
.dynamic = 0x600e28
.got = 0x600ff8
.got.plt = 0x601000
.data = 0x601050
.bss = 0x601060

```

gdb-peda\$ x/i 0x4005b0

```

0x4005b0 <printf@plt>:      jmp     QWORD PTR [rip+0x200a6a]      # 0x601020

```

gdb-peda\$ x/gx 0x601020

```

0x601020:      0x0000000004005b6

```

gdb-peda\$ x/i 0x0000000004005b6

```

0x4005b6 <printf@plt+6>:      push   0x1

```

gdb-peda\$ r

Starting program: /home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-Relro

----- 1 -----

Input Pointer address : ^C

Program received signal SIGINT, Interrupt.

gdb-peda\$ x/gx 0x601020

```

0x601020:      0x00007ffff7a62800

```

gdb-peda\$ x/5i 0x00007ffff7a62800

```

0x7ffff7a62800 <__printf>:      sub     rsp,0xd8
0x7ffff7a62807 <__printf+7>:      test   al,al
0x7ffff7a62809 <__printf+9>:      mov     QWORD PTR [rsp+0x28],rsi
0x7ffff7a6280e <__printf+14>:      mov     QWORD PTR [rsp+0x30],rdx
0x7ffff7a62813 <__printf+19>:      mov     QWORD PTR [rsp+0x38],rcx

```

gdb-peda\$ vmmmap

Start	End	Perm	Name
0x00400000	0x00401000	r-xp	/home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-Relro
0x00600000	0x00601000	r--p	/home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-Relro
0x00601000	0x00602000	rw-p	/home/lazenca0x0/Documents/Definition/protection/RELRO/RELRO-Relro
0x00602000	0x00623000	rw-p	[heap]
0x00007ffff7a0d000	0x00007ffff7bcd000	r-xp	/lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7bcd000	0x00007ffff7dcd000	---p	/lib/x86_64-linux-gnu/libc-2.23.so

```
0x00007ffff7dcd000 0x00007ffff7dd1000 r--p      /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dd1000 0x00007ffff7dd3000 rw-p      /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dd3000 0x00007ffff7dd7000 rw-p      mapped
0x00007ffff7dd7000 0x00007ffff7dfd000 r-xp      /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7dfd000 0x00007ffff7fdc000 rw-p      mapped
0x00007ffff7ffe000 0x00007ffff7ff8000 rw-p      mapped
0x00007ffff7ff8000 0x00007ffff7ffa000 r--p      [vvar]
0x00007ffff7ffa000 0x00007ffff7ffc000 r-xp      [vdso]
0x00007ffff7ffc000 0x00007ffff7ffd000 r--p      /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7ffd000 0x00007ffff7ffe000 rw-p      /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7ffe000 0x00007ffff7fff000 rw-p      mapped
0x00007ffff7fff000 0x00007ffff800000000 rw-p      [stack]
0xffffffffff600000 0xffffffffff601000 r-xp      [syscall]
gdb-peda$
```

- 다음과 같이 아직 호출되지 않은 함수들의 GOT 값은 어떤지 확인해보겠습니다.
  - main 함수에서 scanf 함수를 사용하기 위해 메모리 주소 0x400600(".plt")을 호출합니다.
  - 0x400600 영역의 코드는 "jmp QWORD PTR [rip+0x200a42]"이며, 0x601048 영역에 저장된 주소로 이동합니다.
  - 0x601048 영역에 저장된 값은 0x400606이며, 해당 영역은 Stub 코드가 저장되어 있습니다.
  - scanf 함수가 아직 호출된 적이 없기 때문에 0x601048(".got.plt") 영역에 동적라이브러리의 scanf 함수의 시작 주소 값이 저장되어 있지 않습니다.
  - Partial RELRO에 Lazy binding을 사용하기 때문에 함수를 호출하지 않으면 동적라이브러리의 주소 값을 ".got.plt" 영역에 저장되지 않습니다.

#### GOT area of uncalled function

```
gdb-peda$ x/i 0x400600
0x400600 <__isoc99_scanf@plt>:      jmp     QWORD PTR [rip+0x200a42]      # 0x601048
gdb-peda$ x/gx 0x601048
0x601048:      0x0000000000400606
gdb-peda$ x/2i 0x0000000000400606
0x400606 <__isoc99_scanf@plt+6>:      push   0x6
0x40060b <__isoc99_scanf@plt+11>:     jmp     0x400590
gdb-peda$
```

## Full RELRO

- 다음과 같이 동적 라이브러리의 주소를 호출하게 됩니다.
  - main 함수에서 printf 함수를 사용하기 위해 메모리 주소 0x4005c8을 호출합니다.
    - 메모리 주소 0x4005c8는 ".plt.got" 영역입니다.
    - ".plt.got" 영역은 0x4005c0 ~ 0x400600입니다.
  - 0x4005c8 영역의 코드는 "jmp QWORD PTR [rip+0x2009fa]"입니다.
    - 즉, 메모리 주소 0x600fc8에 저장된 주소로 JUMP 합니다.
      - 메모리 주소 0x600fc8은 ".got" 영역입니다.
      - ".got.plt" 영역은 0x600fa8 ~ 0x601000입니다.
    - 메모리 주소 0x600fc8에 아무런 값도 저장되어 있지 않습니다.
      - 이는 해당 프로그램에서 printf 함수가 호출되지 않았기 때문입니다.
    - 프로그램을 실행하고 printf 함수가 호출되기 시작하면 메모리 주소 0x600fc8(".got" 영역) 영역에 동적라이브러리의 printf 함수의 시작 주소 값이 저장됩니다.
  - 즉, Full RELRO가 적용된 바이너리는 ".got"영역이 Read-only로 설정되어 때문에 ".got" 영역에 저장된 값을 변경할 수 없습니다.

#### function call of "Full RELRO"

```
lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO$ gdb -q ./RELRO-FullRelro
Reading symbols from ./RELRO-FullRelro...(no debugging symbols found)...done.
gdb-peda$ disassemble main
Dump of assembler code for function main:
0x00000000004006f6 <+0>:      push   rbp
0x00000000004006f7 <+1>:      mov    rbp,rbp
0x00000000004006fa <+4>:      sub    rsp,0x30
0x00000000004006fe <+8>:      mov    rax,QWORD PTR fs:0x28
0x0000000000400707 <+17>:     mov    QWORD PTR [rbp-0x8],rax
0x000000000040070b <+21>:     xor    eax,eax
0x000000000040070d <+23>:     mov    DWORD PTR [rbp-0x2c],0x1
0x0000000000400714 <+30>:     jmp   0x4007e2 <main+236>
0x0000000000400719 <+35>:     mov    eax,DWORD PTR [rbp-0x2c]
```

```

0x00000000040071c <+38>:      mov     esi,eax
0x00000000040071e <+40>:      mov     edi,0x4008a4
0x000000000400723 <+45>:      mov     eax,0x0
0x000000000400728 <+50>:      call   0x4005c8
0x00000000040072d <+55>:      lea    rax,[rbp-0x20]
0x000000000400731 <+59>:      mov     edx,0x10
0x000000000400736 <+64>:      mov     esi,0x0
0x00000000040073b <+69>:      mov     rdi,rax
0x00000000040073e <+72>:      call   0x4005d0
0x000000000400743 <+77>:      mov     edi,0x4008b4
0x000000000400748 <+82>:      mov     eax,0x0
0x00000000040074d <+87>:      call   0x4005c8
0x000000000400752 <+92>:      mov     rdx,QWORD PTR [rip+0x2008b7]      # 0x601010 <stdin@GLIBC_2.2.5>
0x000000000400759 <+99>:      lea    rax,[rbp-0x20]
0x00000000040075d <+103>:     mov     esi,0x10
0x000000000400762 <+108>:     mov     rdi,rax
0x000000000400765 <+111>:     call   0x4005e0
0x00000000040076a <+116>:     lea    rax,[rbp-0x20]
0x00000000040076e <+120>:     mov     edx,0x10
0x000000000400773 <+125>:     mov     esi,0x0
0x000000000400778 <+130>:     mov     rdi,rax
0x00000000040077b <+133>:     mov     eax,0x0
0x000000000400780 <+138>:     call   0x4005f0
0x000000000400785 <+143>:     cdqe
0x000000000400787 <+145>:     mov     QWORD PTR [rbp-0x28],rax
0x00000000040078b <+149>:     mov     rax,QWORD PTR [rbp-0x28]
0x00000000040078f <+153>:     mov     rsi,rax
0x000000000400792 <+156>:     mov     edi,0x4008cd
0x000000000400797 <+161>:     mov     eax,0x0
0x00000000040079c <+166>:     call   0x4005c8
0x0000000004007a1 <+171>:     mov     edi,0x4008e3
0x0000000004007a6 <+176>:     mov     eax,0x0
0x0000000004007ab <+181>:     call   0x4005c8
0x0000000004007b0 <+186>:     mov     rdx,QWORD PTR [rip+0x200859]      # 0x601010 <stdin@GLIBC_2.2.5>
0x0000000004007b7 <+193>:     mov     rax,QWORD PTR [rbp-0x28]
0x0000000004007bb <+197>:     mov     esi,0x10
0x0000000004007c0 <+202>:     mov     rdi,rax
0x0000000004007c3 <+205>:     call   0x4005e0
0x0000000004007c8 <+210>:     mov     rax,QWORD PTR [rbp-0x28]
0x0000000004007cc <+214>:     mov     rsi,rax
0x0000000004007cf <+217>:     mov     edi,0x4008f9
0x0000000004007d4 <+222>:     mov     eax,0x0
0x0000000004007d9 <+227>:     call   0x4005c8
0x0000000004007de <+232>:     add    DWORD PTR [rbp-0x2c],0x1
0x0000000004007e2 <+236>:     cmp    DWORD PTR [rbp-0x2c],0x64
0x0000000004007e6 <+240>:     jne    0x400719 <main+35>
0x0000000004007ec <+246>:     lea    rax,[rbp-0x20]
0x0000000004007f0 <+250>:     mov     rsi,rax
0x0000000004007f3 <+253>:     mov     edi,0x40090c
0x0000000004007f8 <+258>:     mov     eax,0x0
0x0000000004007fd <+263>:     call   0x4005f8
0x000000000400802 <+268>:     nop
0x000000000400803 <+269>:     mov     rax,QWORD PTR [rbp-0x8]
0x000000000400807 <+273>:     xor    rax,QWORD PTR fs:0x28
0x000000000400810 <+282>:     je     0x400817 <main+289>
0x000000000400812 <+284>:     call   0x4005c0
0x000000000400817 <+289>:     leave
0x000000000400818 <+290>:     ret

```

End of assembler dump.

```

gdb-peda$ elfheader
.interp = 0x400238
.note.ABI-tag = 0x400254
.note.gnu.build-id = 0x400274
.gnu.hash = 0x400298
.dynsym = 0x4002e0
.dynstr = 0x4003d0
.gnu.version = 0x40045e
.gnu.version_r = 0x400478
.rela.dyn = 0x4004b8
.init = 0x400590
.plt = 0x4005b0

```

```

.plt.got = 0x4005c0
.text = 0x400600
.fini = 0x400894
.rodata = 0x4008a0
.eh_frame_hdr = 0x400910
.eh_frame = 0x400948
.init_array = 0x600dd0
.fini_array = 0x600dd8
.jcr = 0x600de0
.dynamic = 0x600de8
.got = 0x600fa8
.data = 0x601000
.bss = 0x601010
gdb-peda$ x/i 0x4005c8
0x4005c8:      jmp     QWORD PTR [rip+0x2009fa]      # 0x600fc8
gdb-peda$ x/gx 0x600fc8
0x600fc8:      0x0000000000000000
gdb-peda$ r
Starting program: /home/lazencax0/Documents/Definition/protection/RELRO/RELRO-FullRelro
----- 1 -----
Input Pointer address : ^C
Program received signal SIGINT, Interrupt.

gdb-peda$ x/gx 0x600fc8
0x600fc8:      0x00007ffff7a62800
gdb-peda$ x/5i 0x00007ffff7a62800
0x7ffff7a62800 <__printf>:      sub     rsp,0xd8
0x7ffff7a62807 <__printf+7>:      test    al,al
0x7ffff7a62809 <__printf+9>:      mov     QWORD PTR [rsp+0x28],rsi
0x7ffff7a6280e <__printf+14>:     mov     QWORD PTR [rsp+0x30],rdx
0x7ffff7a62813 <__printf+19>:     mov     QWORD PTR [rsp+0x38],rcx
gdb-peda$ vmmmap
Start          End              Perm            Name
0x00400000     0x00401000      r-xp            /home/lazencax0/Documents/Definition/protection/RELRO/RELRO-
FullRelro
0x00600000     0x00601000      r--p            /home/lazencax0/Documents/Definition/protection/RELRO/RELRO-
FullRelro
0x00601000     0x00602000      rw-p            /home/lazencax0/Documents/Definition/protection/RELRO/RELRO-
FullRelro
0x00602000     0x00623000      rw-p            [heap]
0x00007ffff7a0d000 0x00007ffff7bcd000 r-xp            /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7bcd000 0x00007ffff7dcd000 ---p            /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dcd000 0x00007ffff7dd1000 r--p            /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dd1000 0x00007ffff7dd3000 rw-p            /lib/x86_64-linux-gnu/libc-2.23.so
0x00007ffff7dd3000 0x00007ffff7dd7000 rw-p            mapped
0x00007ffff7dd7000 0x00007ffff7dfd000 r-xp            /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7dfd000 0x00007ffff7fdc000 rw-p            mapped
0x00007ffff7ffe000 0x00007ffff7ff8000 rw-p            mapped
0x00007ffff7ff8000 0x00007ffff7ffa000 r--p            [vvar]
0x00007ffff7ffa000 0x00007ffff7ffc000 r-xp            [vdso]
0x00007ffff7ffc000 0x00007ffff7ffd000 r--p            /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7ffd000 0x00007ffff7ffe000 rw-p            /lib/x86_64-linux-gnu/ld-2.23.so
0x00007ffff7ffe000 0x00007ffff7fff000 rw-p            mapped
0x00007ffff7fff000 0x00007ffff7fff000 rw-p            [stack]
0xffffffffff600000 0xffffffffff601000 r-xp            [vsyscall]
gdb-peda$

```

• 다음과 같이 아직 호출되지 않은 함수들의 GOT 값은 어떤지 확인해보겠습니다.

- main 함수에서 scanf 함수를 사용하기 위해 메모리 주소 0x4005f8("plt.got")을 호출합니다.
- 0x4005f8 영역의 코드는 "jmp QWORD PTR [rip+0x2009fa]" 이며, 0x600f8 영역에 저장된 주소로 이동합니다.
- 0x600f8 영역에 저장된 값은 0x00007ffff7a784d0 이며, 해당 영역은 동적라이브러리의 scanf 함수의 시작 주소 값 입니다.
- Full RELRO에서는 Now binding을 사용하기 때문에 프로그램 메모리에 로드 될때 해당 프로그램에서 사용되는 모든 동적 함수의 주소를 ".got" 영역에 저장됩니다.



## GOT area of uncalled function

```
gdb-peda$ x/i 0x4005f8
0x4005f8:      jmp     QWORD PTR [rip+0x2009fa]      # 0x600ff8
gdb-peda$ x/gx 0x600ff8
0x600ff8:      0x00007ffff7a784d0
gdb-peda$
```

### Calling the printf function

- 다음과 같이 printf 함수가 호출됩니다.

	Partial RELRO		Full RELRO
main()	call 0x4005b0 <printf@plt>	main()	call 0x4005c8
.plt(printf@plt)	jmp QWORD PTR [rip+0x200a6a] # 0x601020	.plt.got(0x4005c8)	jmp QWORD PTR [rip+0x2009fa]
.got.plt(0x601020)	0x7ffff7a62800	.got(0x600fc8)	0x7ffff7a62800

## How to detect NX in the "Checksec.sh" file

### Binary

- 다음과 같은 방법으로 바이너리의 RELRO 설정여부를 확인합니다.
  - 'readelf' 명령어를 이용해 해당 파일의 프로그래 헤더와 Dynamic section 정보를 가져와 RELRO를 설정여부를 확인합니다.
  - 파일의 프로그래 헤더에 'GNU\_RELRO'가 있으면 RELRO가 적용되었다고 판단합니다.
    - 그리고 Dynamic section에 BIND\_NOW가 있으면 Full RELRO가 적용되었다고 판단합니다.
    - Dynamic section에 BIND\_NOW가 없으면 Partial RELRO가 적용되었다고 판단합니다.

### Checksec.sh - line 145

```
# check for RELRO support
if readelf -l $1 2>/dev/null | grep -q 'GNU_RELRO'; then
  if readelf -d $1 2>/dev/null | grep -q 'BIND_NOW'; then
    echo -n -e '\033[32mFull RELRO  \033[m '
  else
    echo -n -e '\033[33mPartial RELRO\033[m '
  fi
else
  echo -n -e '\033[31mNo RELRO  \033[m '
fi
```

### No RELRO

```
lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO$ readelf -l RELRO-NoRelro |grep 'GNU_RELRO'
lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO$
```

### Partial RELRO

```
lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO$ readelf -l RELRO-Relro |grep 'GNU_RELRO'
GNU_RELRO      0x0000000000000e10 0x0000000000600e10 0x0000000000600e10
lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO$ readelf -d RELRO-Relro |grep 'BIND_NOW'
lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO$
```

## Full RELRO

```
lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO$ readelf -l RELRO-FullRelro |grep 'GNU_RELRO'
GNU_RELRO      0x00000000000000dd0 0x00000000000600dd0 0x00000000000600dd0
lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO$ readelf -d RELRO-FullRelro |grep 'BIND_NOW'
0x0000000000000018 (BIND_NOW)
lazenca0x0@ubuntu:~/Documents/Definition/protection/RELRO$
```

## Process

- 다음과 같은 방법으로 프로세서의 RELRO 설정여부를 확인합니다.
  - Binary의 확인 방식과 비슷하며, 전달되는 파일의 경로가 다음과 같이 다릅니다.
    - Ex) /proc/<PID>/exe
  - 추가된 동작은 '/proc/<PID>/exe' 파일에 'Program Headers' 정보가 있는지 확인합니다.

## Checksec.sh - line 199

```
# check for RELRO support
if readelf -l $1/exe 2>/dev/null | grep -q 'Program Headers'; then
  if readelf -l $1/exe 2>/dev/null | grep -q 'GNU_RELRO'; then
    if readelf -d $1/exe 2>/dev/null | grep -q 'BIND_NOW'; then
      echo -n -e '\033[32mFull RELRO      \033[m '
    else
      echo -n -e '\033[33mPartial RELRO   \033[m '
    fi
  else
    echo -n -e '\033[31mNo RELRO      \033[m '
  fi
else
  echo -n -e '\033[31mPermission denied (please run as root)\033[m\n'
  exit 1
fi
```

## Related information

- <http://tk-blog.blogspot.jp/2009/02/relro-not-so-well-known-memory.html>