Compile-Time Memory Corruption Mitigations

4Developers 2018, Gdańsk

Andrzej Dyjak

🤝

whoami

Preludium

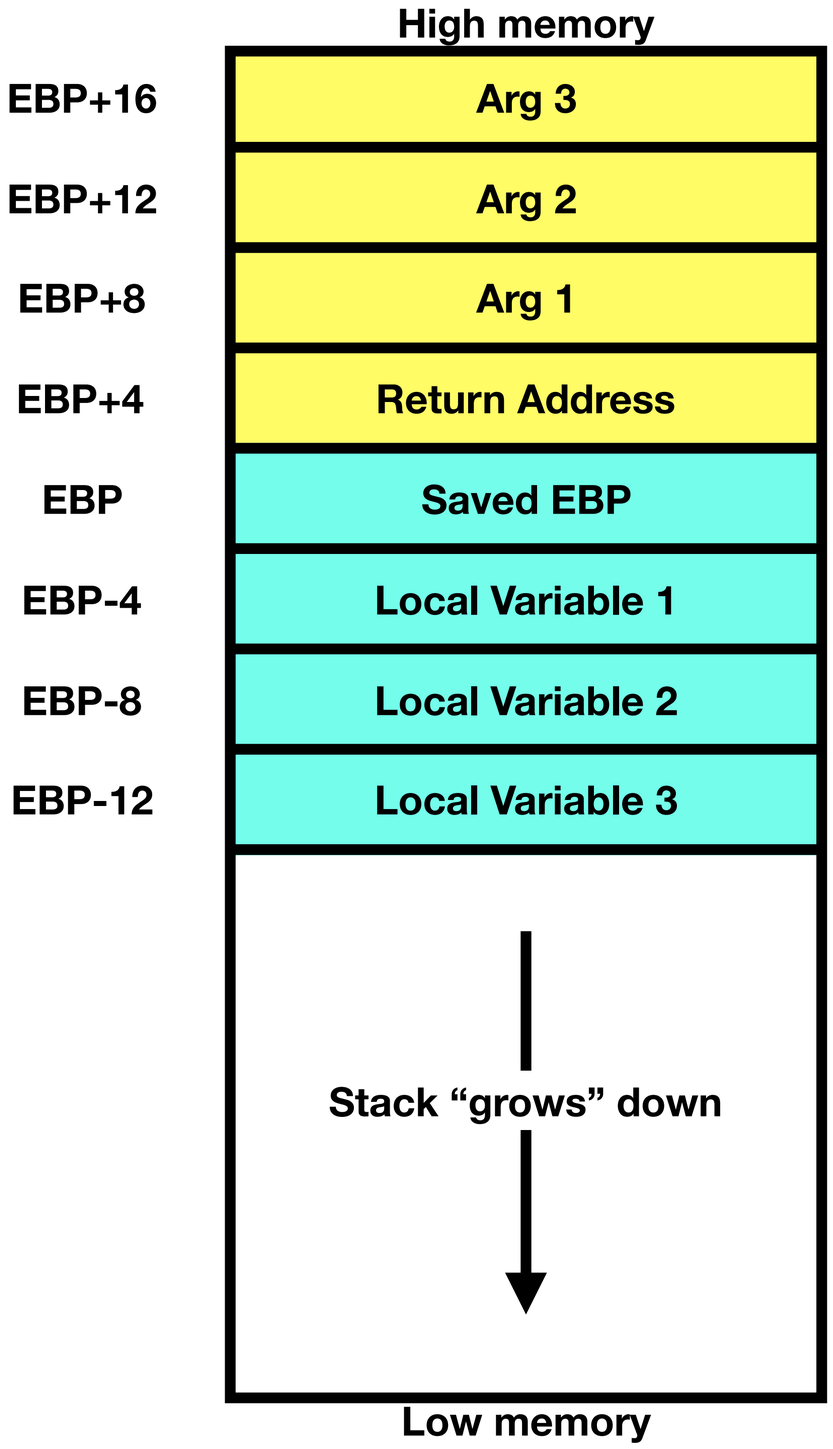
- What are compile-time mitigations?
- Why this topic might be of interest to you?
- What compilers will we focus on?
- What architecture will we talk about?

Memory Corruption Bugs

- What are they?
- How can they be exploited?
- Why are they important?

Standard stuff (10+ years)

- Stack Smashing Protector (`-fstack-protector, -fstack-protector-all, -fstack-protector-strong, -fstack-protector-explicit`)
- Fortify Source (`-D_FORTIFY_SOURCE=[1|2] -O2`)
- Format Strings (`-Wformat, -Wformat-overflow=[1|2], -Wformat-security`)
- Position Independent *, ASLR (`-fPIE, -fPIC` for gcc, `-pie` for ld)
- Relocations Read-Only (`-Wl,-z,relro (partial);` **`-Wl,-z,relro,-z,now (full)`**)


High memory
EBP+16
Arg 3
EBP+12
Arg 2
EBP+8
Arg 1
EBP+4
Return Address
EBP
Saved EBP
EBP-4
Local Variable 1
EBP-8
Local Variable 2
EBP-12
Local Variable 3
Stack “grows” down
Low memory

Stack-Smashing Protector (ProPolice)

- Uses stack canaries to detect an overflow in which case it aborts the program
- Provides safer stack structure
  - Makes sure that arguments (on the stack, `$rbp+N`) have no arrays or pointers (?)
  - Re-orders the stack so arrays are put before other local variables regardless of the source definition
- `-fstack-protector`, `-fstack-protector-all`, **`-fstack-protector-strong`**, `-fstack-protector-explicit`

```
ad@poligon:~/research/compilers/samples$ cat overflow.c
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
        char *ptr = NULL;
        int n = 128;
        char buf[1337];

        strcpy(buf, argv[1]);

        return 0;
}
```

without stack protector

```
(gdb) disass main
Dump of assembler code for function main:
   0x00000000004004b2 <+0>:     push   rbp
   0x00000000004004b3 <+1>:     mov    rbp,rsp
   0x00000000004004b6 <+4>:     sub    rsp,0x560
   0x00000000004004bd <+11>:    mov    DWORD PTR [rbp-0x554],edi
   0x00000000004004c3 <+17>:    mov    QWORD PTR [rbp-0x560],rsi
   0x00000000004004ca <+24>:    mov    QWORD PTR [rbp-0x8],0x0
   0x00000000004004d2 <+32>:    mov    DWORD PTR [rbp-0xc],0x80
   0x00000000004004d9 <+39>:    mov    rax,QWORD PTR [rbp-0x560]
   0x00000000004004e0 <+46>:    add    rax,0x8
   0x00000000004004e4 <+50>:    mov    rdx,QWORD PTR [rax]
   0x00000000004004e7 <+53>:    lea    rax,[rbp-0x550]
   0x00000000004004ee <+60>:    mov    rsi,rdx
   0x00000000004004f1 <+63>:    mov    rdi,rax
   0x00000000004004f4 <+66>:    call   0x4003b0 <strcpy@plt>
=> 0x00000000004004f9 <+71>:    mov    eax,0x0
   0x00000000004004fe <+76>:    leave
   0x00000000004004ff <+77>:    ret
End of assembler dump.
Python Exception <class 'UnicodeEncodeError'> 'ascii' codec can't encode character '\u27a4' in position 12: ordinal not in range(128):
(gdb) x/x $rbp-0x8
0x7fffffffdc58: 0x41414141
Python Exception <class 'UnicodeEncodeError'> 'ascii' codec can't encode character '\u27a4' in position 12: ordinal not in range(128):
(gdb) x/x $rbp-0xc
0x7fffffffdc54: 0x41414141
Python Exception <class 'UnicodeEncodeError'> 'ascii' codec can't encode character '\u27a4' in position 12: ordinal not in range(128):
(gdb) r `python -c 'print "A"*2000'`
```

1

2

3

with stack protector

```
(gdb) disass main
Dump of assembler code for function main:
   0x0000000000400522 <+0>:     push   rbp
   0x0000000000400523 <+1>:     mov    rbp,rsp
   0x0000000000400526 <+4>:     sub    rsp,0x570
   0x000000000040052d <+11>:    mov    DWORD PTR [rbp-0x564],edi
   0x0000000000400533 <+17>:    mov    QWORD PTR [rbp-0x570],rsi
   0x000000000040053a <+24>:    mov    rax,QWORD PTR fs:0x28
   0x0000000000400543 <+33>:    mov    QWORD PTR [rbp-0x8],rax
   0x0000000000400547 <+37>:    xor    eax,eax
   0x0000000000400549 <+39>:    mov    QWORD PTR [rbp-0x558],0x0
   0x0000000000400554 <+50>:    mov    DWORD PTR [rbp-0x55c],0x80
   0x000000000040055e <+60>:    mov    rax,QWORD PTR [rbp-0x570]
   0x0000000000400565 <+67>:    add    rax,0x8
   0x0000000000400569 <+71>:    mov    rdx,QWORD PTR [rax]
   0x000000000040056c <+74>:    lea    rax,[rbp-0x550]
   0x0000000000400573 <+81>:    mov    rsi,rdx
   0x0000000000400576 <+84>:    mov    rdi,rax
   0x0000000000400579 <+87>:    call   0x400410 <strcpy@plt>
=> 0x000000000040057e <+92>:    mov    eax,0x0
   0x0000000000400583 <+97>:    mov    rcx,QWORD PTR [rbp-0x8]
   0x0000000000400587 <+101>:   xor    rcx,QWORD PTR fs:0x28
   0x0000000000400590 <+110>:   je     0x400597 <main+117>
   0x0000000000400592 <+112>:   call   0x400420 <__stack_chk_fail@plt>
   0x0000000000400597 <+117>:   leave
   0x0000000000400598 <+118>:   ret
End of assembler dump.
Python Exception <class 'UnicodeEncodeError'> 'ascii' codec can't encode character '\u27a4' in position 12: ordinal not in range(128):
(gdb) x/d $rbp-0x558
0x7fffffffd988: 0
Python Exception <class 'UnicodeEncodeError'> 'ascii' codec can't encode character '\u27a4' in position 12: ordinal not in range(128):
(gdb) x/d $rbp-0x55c
0x7fffffffd984: 128
Python Exception <class 'UnicodeEncodeError'> 'ascii' codec can't encode character '\u27a4' in position 12: ordinal not in range(128):
(gdb) x/x $rbp-0x8
0x7fffffffded8: 0x41424344
Python Exception <class 'UnicodeEncodeError'> 'ascii' codec can't encode character '\u27a4' in position 12: ordinal not in range(128):
(gdb) r `python -c 'print "A"*1352 + "DCBA"'`
```

1

2

3

4

5

Fortify Source

- Simple idea: try to pre-compute size of buffers for functions prone to overflows (e.g. `strcpy()`, `memcpy()`, etc)
- But people already use those and education is hard… so substitute them at compile time with checks included alternatives (e.g. `strcpy()` will be substituted with `__strcpy_chk()`)
- Some additional voodoo like checking if `%n` for format strings is RO
- Overhead is small = easy to accept
- `-D_FORITFY_SOURCE=[1|2]` **`-O2`**

```
ad@poligon:~/research/compilers/samples$ cat overflow2.c
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
        char buf[1337];
        char buf2[2000];

        int i;
        for(i=0; i<sizeof(buf2); i++) {
                /* why memset() if you can for loop
                   && in the same time show your audience what your curly brace style is
                   no new lines it's how I roll */
                buf2[i] = 'A';
        }

        strcpy(buf, buf2);

        return 0;
}
```

```
ad@poligon:~/research/compilers/samples$ gcc-8.2 -ggdb overflow2.c -o overflow2
ad@poligon:~/research/compilers/samples$ ./overflow2
Segmentation fault (core dumped)
```

```
ad@poligon:~/research/compilers/samples$ gcc-8.2 -ggdb -D_FORTIFY_SOURCE=2 overflow.c -o overflow-fs2
ad@poligon:~/research/compilers/samples$ ./overflow-fs2
Segmentation fault (core dumped)
```

(without -O2)

```
ad@poligon:~/research/compilers/samples$ gcc-8.2 -ggdb -O2 -D_FORTIFY_SOURCE=2 overflow2.c -o overflow2-fs
ad@poligon:~/research/compilers/samples$ ./overflow2-fs
*** buffer overflow detected ***: ./overflow2-fs terminated
======= Backtrace: =========
/lib/x86_64-linux-gnu/libc.so.6(+0x777e5)[0x7fbeab4d77e5]
/lib/x86_64-linux-gnu/libc.so.6(__fortify_fail+0x5c)[0x7fbeab57915c]
/lib/x86_64-linux-gnu/libc.so.6(+0x117160)[0x7fbeab577160]
/lib/x86_64-linux-gnu/libc.so.6(+0x1164b2)[0x7fbeab5764b2]
./overflow2-fs[0x400441]
/lib/x86_64-linux-gnu/libc.so.6(__libc_start_main+0xf0)[0x7fbeab480830]
./overflow2-fs[0x400479]
======= Memory map: ========
00400000-00401000 r-xp 00000000 fd:01 1832248                    /home/ad/research/compilers/samples/overflow2-fs
00600000-00601000 rw-p 00000000 fd:01 1832248                    /home/ad/research/compilers/samples/overflow2-fs
01304000-01325000 rw-p 00000000 00:00 0                          [heap]
7fbeab24a000-7fbeab260000 r-xp 00000000 fd:01 1975               /lib/x86_64-linux-gnu/libgcc_s.so.1
7fbeab260000-7fbeab45f000 ---p 00016000 fd:01 1975               /lib/x86_64-linux-gnu/libgcc_s.so.1
7fbeab45f000-7fbeab460000 rw-p 00015000 fd:01 1975               /lib/x86_64-linux-gnu/libgcc_s.so.1
7fbeab460000-7fbeab620000 r-xp 00000000 fd:01 1980               /lib/x86_64-linux-gnu/libc-2.23.so
7fbeab620000-7fbeab820000 ---p 001c0000 fd:01 1980               /lib/x86_64-linux-gnu/libc-2.23.so
7fbeab820000-7fbeab824000 r--p 001c0000 fd:01 1980               /lib/x86_64-linux-gnu/libc-2.23.so
7fbeab824000-7fbeab826000 rw-p 001c4000 fd:01 1980               /lib/x86_64-linux-gnu/libc-2.23.so
7fbeab826000-7fbeab82a000 rw-p 00000000 00:00 0
7fbeab82a000-7fbeab850000 r-xp 00000000 fd:01 1978               /lib/x86_64-linux-gnu/ld-2.23.so
7fbeaba45000-7fbeaba48000 rw-p 00000000 00:00 0
7fbeaba4e000-7fbeaba4f000 rw-p 00000000 00:00 0
7fbeaba4f000-7fbeaba50000 r--p 00025000 fd:01 1978               /lib/x86_64-linux-gnu/ld-2.23.so
7fbeaba50000-7fbeaba51000 rw-p 00026000 fd:01 1978               /lib/x86_64-linux-gnu/ld-2.23.so
7fbeaba51000-7fbeaba52000 rw-p 00000000 00:00 0
7fff7afa6000-7fff7afc7000 rw-p 00000000 00:00 0                  [stack]
7fff7afee000-7fff7aff1000 r--p 00000000 00:00 0                  [vvar]
7fff7aff1000-7fff7aff3000 r-xp 00000000 00:00 0                  [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0          [vsyscall]
Aborted (core dumped)
```

```
$rip   : 0x00007ffff7a42428  →  <raise+56> cmp rax, 0xfffffffffffff000
$r8    : 0x0000000000000004
$r9    : 0x0000000000000000
$r10   : 0x0000000000000008
$r11   : 0x0000000000000246
$r12   : 0x000000000000005e
$r13   : 0x00007fffffffcd68  →  0x00007ffff7de6ac6  →  <_dl_fixup+214> mov r8, rax
$r14   : 0x00007fffffffcd68  →  0x00007ffff7de6ac6  →  <_dl_fixup+214> mov r8, rax
$r15   : 0x0000000000000002
$eflags: [carry PARITY adjust ZERO sign trap INTERRUPT direction overflow resume virtualx86 identification]
$es: 0x0000  $ds: 0x0000  $gs: 0x0000  $cs: 0x0033  $ss: 0x002b  $fs: 0x0000
──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────[ stack ]──
0x00007fffffffcbd8│+0x00: 0x00007ffff7a4402a  →  <abort+362> mov rdx, QWORD PTR fs:0x10  ←$rsp
0x00007fffffffcbe0│+0x08: 0x0000000000000020
0x00007fffffffcbe8│+0x10: 0x0000000000000000
0x00007fffffffcbf0│+0x18: 0x0000000000000000
0x00007fffffffcbf8│+0x20: 0x0000000000000000
0x00007fffffffcc00│+0x28: 0x0000000000000000
0x00007fffffffcc08│+0x30: 0x0000000000000000
0x00007fffffffcc10│+0x38: 0x0000000000000000
──────────────────────────────────────────────────────────────────────────────────────────────────────[ code:i386:x86-64 ]──
   0x7ffff7a4241e <raise+46>       mov    eax, 0xea
   0x7ffff7a42423 <raise+51>       movsxd rdi, ecx
   0x7ffff7a42426 <raise+54>       syscall
 → 0x7ffff7a42428 <raise+56>       cmp    rax, 0xfffffffffffff000
   0x7ffff7a4242e <raise+62>       ja     0x7ffff7a42450 <__GI_raise+96>
   0x7ffff7a42430 <raise+64>       repz   ret
   0x7ffff7a42432 <raise+66>       nop    WORD PTR [rax+rax*1+0x0]
   0x7ffff7a42438 <raise+72>       test   ecx, ecx
   0x7ffff7a4243a <raise+74>       jg     0x7ffff7a4241b <__GI_raise+43>
──────────────────────────────────────────────────────────────────────────────────────────────────────────[ threads ]──
[#0] Id 1, Name: "overflow2-fs", stopped, reason: SIGABRT
──────────────────────────────────────────────────────────────────────────────────────────────────────────────[ trace ]──
[#0] 0x7ffff7a42428 → Name: __GI_raise(sig=0x6)
[#1] 0x7ffff7a4402a → Name: __GI_abort()
[#2] 0x7ffff7a847ea → Name: __libc_message(do_abort=0x2, fmt=0x7ffff7b9c49f "*** %s ***: %s terminated\n")
[#3] 0x7ffff7b2615c → Name: __GI___fortify_fail(msg=<optimized out>)
[#4] 0x7ffff7b24160 → Name: __GI___chk_fail()
[#5] 0x7ffff7b234b2 → Name: __strcpy_chk(dest=0x7fffffffcf50 "", src=0x7fffffffd490 'A' <repeats 2000 times>, destlen=0x539)
[#6] 0x400441 → Name: strcpy(__src=0x7fffffffd490 'A' <repeats 2000 times>, __dest=0x7fffffffcf50 "")
[#7] 0x400441 → Name: main(argc=<optimized out>, argv=<optimized out>)
──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
Python Exception <class 'UnicodeEncodeError'> 'ascii' codec can't encode character '\u27a4' in position 12: ordinal not in range(128):
(gdb) r `python -c 'print "A"*2000'`
```

```
ad@poligon:~/research/compilers/samples/glibc-2.28/debug$ ls
Depend                fdelt_chk.c           gets_chk.c            poll_chk.c            sprintf_chk.c         tst-backtrace3.c      tst-lfschk5.cc        vswprintf_chk.c       wmemcpy_chk.c
Makefile              fgets_chk.c           getwd_chk.c           ppoll_chk.c           stack_chk_fail.c      tst-backtrace4.c      tst-lfschk6.cc        vwprintf_chk.c        wmemmove_chk.c
Versions              fgets_u_chk.c         longjmp_chk.c         pread64_chk.c         stack_chk_fail_local.c tst-backtrace5.c     tst-longjmp_chk.c     warning-nop.c         wmempcpy_chk.c
asprintf_chk.c        fgetws_chk.c          mbsnrtowcs_chk.c      pread_chk.c           stpcpy_chk.c          tst-backtrace6.c      tst-longjmp_chk2.c    wcpcpy_chk.c          wmemset_chk.c
backtrace-tst.c       fgetws_u_chk.c        mbsrtowcs_chk.c       printf_chk.c          stpncpy_chk.c         tst-chk1.c            tst-longjmp_chk3.c    wcpncpy_chk.c         wprintf_chk.c
backtrace.c           fortify_fail.c        mbstowcs_chk.c        read_chk.c            strcat_chk.c          tst-chk2.c            tst-ssp-1.c           wcrtomb_chk.c         xtrace.sh
backtracesyms.c       fprintf_chk.c         memcpy_chk.c          readlink_chk.c        strcpy_chk.c          tst-chk3.c            ttyname_r_chk.c       wcscat_chk.c
backtracesymsfd.c     fread_chk.c           memmove_chk.c         readlinkat_chk.c      strncat_chk.c         tst-chk4.cc           vasprintf_chk.c       wcscpy_chk.c
catchsegv.sh          fread_u_chk.c         mempcpy_chk.c         readonly-area.c       strncpy_chk.c         tst-chk5.cc           vdprintf_chk.c        wcsncat_chk.c
chk_fail.c            fwprintf_chk.c        memset_chk.c          realpath_chk.c        swprintf_chk.c        tst-chk6.cc           vfprintf_chk.c        wcsncpy_chk.c
confstr_chk.c         getcwd_chk.c          noophooks.c           recv_chk.c            test-stpcpy_chk.c     tst-lfschk1.c         vfwprintf_chk.c       wcsnrtombs_chk.c
dprintf_chk.c         getdomainname_chk.c   obprintf_chk.c        recvfrom_chk.c        test-strcpy_chk.c     tst-lfschk2.c         vprintf_chk.c         wcsrtombs_chk.c
execinfo.h            getgroups_chk.c       pcprofile.c           segfault.c            tst-backtrace.h       tst-lfschk3.c         vsnprintf_chk.c       wcstombs_chk.c
explicit_bzero_chk.c  gethostname_chk.c     pcprofiledump.c       snprintf_chk.c        tst-backtrace2.c      tst-lfschk4.cc        vsprintf_chk.c        wctomb_chk.c
```

Format Strings

- Warn about unsafe usage of functions that make use of format strings (`printf()`, `scanf()`, etc)
- It **does not actively protect** against anything, if the developer does not deal with the warnings then *all is lost*
- `-Wformat, -Wformat-overflow=[1|2], -Wformat-security`

```
ad@poligon:~/research/compilers/samples$ cat format.c
#include <stdio.h>

int main(int argc, char *argv[])
{
        printf(argv[1]);

        return 0;
}
```

```
ad@poligon:~/research/compilers/samples$ gcc-8.2 -ggdb -Wformat format.c -o format
ad@poligon:~/research/compilers/samples$ ./format %s | xxd
00000000: 9b17 6181 fc7f
```

```
ad@poligon:~/research/compilers/samples$ gcc-8.2 -ggdb -Wformat -Wformat-security format.c -o format-sec
format.c: In function 'main':
format.c:5:2: warning: format not a string literal and no format arguments [-Wformat-security]
  printf(argv[1]);
  ^~~~~~
ad@poligon:~/research/compilers/samples$ ./format-sec %s | xxd
00000000: 8f77 136f fc7f                           .w.o..
```

Position Independent *

- Long time ago (~2000) pipacs 🦉 invented ASLR and after that nothing was the same…
- In order to make use of ASLR we need to properly compile our code (both executables and shared libraries)
- `-fpic, fPIC, -fpie, -fPIE (gcc); -pie (ld)`

```
ad@poligon:~/research/compilers/samples$ cat hello.c
#include <stdio.h>

void hello()
{
        printf("hello() @ %x\n", &hello);
}

int main(int argc, char *argv[])
{
        hello();
        return 0;
}
```

```
ad@poligon:~/research/compilers/samples$ gcc-8.2 -ggdb hello.c -o hello
ad@poligon:~/research/compilers/samples$ ./checksec.sh/checksec --file hello
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable  FILE
No RELRO        No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   68 Symbols      No      0               2            hello
```

1

```
ad@poligon:~/research/compilers/samples$ ./hello
hello() @ 4004b2
ad@poligon:~/research/compilers/samples$ ./hello
hello() @ 4004b2
ad@poligon:~/research/compilers/samples$ ./hello
hello() @ 4004b2
```

2

```
ad@poligon:~/research/compilers/samples$ gcc-8.2 -ggdb -fPIE -pie hello.c -o hello-pie
ad@poligon:~/research/compilers/samples$ ./checksec.sh/checksec --file hello-pie
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable  FILE
No RELRO        No canary found   NX enabled    PIE enabled     No RPATH   No RUNPATH   69 Symbols      No      0               2            hello-pie
```

1

```
ad@poligon:~/research/compilers/samples$ ./hello-pie
hello() @ 22280685
ad@poligon:~/research/compilers/samples$ ./hello-pie
hello() @ 4e87f685
ad@poligon:~/research/compilers/samples$ ./hello-pie
hello() @ 83f9d685
```

2

Relocations Read-Only

- Premise: relocations are a target for overwrites during exploitation-phase
- Solution: make relocations them Read-Only
- `-Wl,-z,relro (partial)`; **`-Wl,-z,relro,-z,now (full)`**

```
ad@poligon:~/research/compilers/samples$ ./checksec.sh/checksec --file hello-none
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable  FILE
No RELRO        No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   68 Symbols      No      0               2            hello-none

ad@poligon:~/research/compilers/samples$ readelf -l hello-none

Elf file type is EXEC (Executable file)
Entry point 0x4003e0
There are 8 program headers, starting at offset 64

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  PHDR           0x0000000000000040 0x0000000000400040 0x0000000000400040
                 0x00000000000001c0 0x00000000000001c0  R E    8
  INTERP         0x0000000000000200 0x0000000000400200 0x0000000000400200
                 0x000000000000001c 0x000000000000001c  R      1
      [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
  LOAD           0x0000000000000000 0x0000000000400000 0x0000000000400000
                 0x00000000000006d4 0x00000000000006d4  R E    200000
  LOAD           0x00000000000006d8 0x00000000006006d8 0x00000000006006d8
                 0x0000000000000220 0x0000000000000228  RW     200000
  DYNAMIC        0x00000000000006e8 0x00000000006006e8 0x00000000006006e8
                 0x00000000000001d0 0x00000000000001d0  RW     8
  NOTE           0x000000000000021c 0x000000000040021c 0x000000000040021c
                 0x0000000000000020 0x0000000000000020  R      4
  GNU_EH_FRAME   0x0000000000000584 0x0000000000400584 0x0000000000400584
                 0x000000000000003c 0x000000000000003c  R      4
  GNU_STACK      0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000000000 0x0000000000000000  RW     10

 Section to Segment mapping:
  Segment Sections...
   00
   01     .interp
   02     .interp .note.ABI-tag .hash .dynsym .dynstr .gnu.version .gnu.version_r .rela.dyn .rela.plt .init .plt .plt.got .text .fini .rodata .eh_frame_hdr .eh_frame
   03     .init_array .fini_array .dynamic .got .got.plt .data .bss
   04     .dynamic
   05     .note.ABI-tag
   06     .eh_frame_hdr
   07
```

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   68 Symbols      No      0               2            hello-partial
```

```
ad@poligon:~/research/compilers/samples$ readelf -l hello-partial

Elf file type is EXEC (Executable file)
Entry point 0x400410
There are 9 program headers, starting at offset 64

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  PHDR           0x0000000000000040 0x0000000000400040 0x0000000000400040
                 0x00000000000001f8 0x00000000000001f8  R E    8
  INTERP         0x0000000000000238 0x0000000000400238 0x0000000000400238
                 0x000000000000001c 0x000000000000001c  R      1
      [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
  LOAD           0x0000000000000000 0x0000000000400000 0x0000000000400000
                 0x0000000000000704 0x0000000000000704  R E    200000
  LOAD           0x0000000000000e18 0x0000000000600e18 0x0000000000600e18
                 0x0000000000000220 0x0000000000000228  RW     200000
  DYNAMIC        0x0000000000000e28 0x0000000000600e28 0x0000000000600e28
                 0x00000000000001d0 0x00000000000001d0  RW     8
  NOTE           0x0000000000000254 0x0000000000400254 0x0000000000400254
                 0x0000000000000020 0x0000000000000020  R      4
  GNU_EH_FRAME   0x00000000000005b4 0x00000000004005b4 0x00000000004005b4
                 0x000000000000003c 0x000000000000003c  R      4
  GNU_STACK      0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000000000 0x0000000000000000  RW     10
  GNU_RELRO      0x0000000000000e18 0x0000000000600e18 0x0000000000600e18
                 0x00000000000001e8 0x00000000000001e8  R      1

 Section to Segment mapping:
  Segment Sections...
   00
   01     .interp
   02     .interp .note.ABI-tag .hash .dynsym .dynstr .gnu.version .gnu.version_r .rela.dyn .rela.plt .init .plt .plt.got .text .fini .rodata .eh_frame_hdr .eh_frame
   03     .init_array .fini_array .dynamic .got .got.plt .data .bss
   04     .dynamic
   05     .note.ABI-tag
   06     .eh_frame_hdr
   07
   08     .init_array .fini_array .dynamic .got
```

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable  FILE
Full RELRO      No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   66 Symbols      No      0               2            hello-full
```

```
ad@poligon:~/research/compilers/samples$ readelf -l hello-full

Elf file type is EXEC (Executable file)
Entry point 0x400400
There are 9 program headers, starting at offset 64

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  PHDR           0x0000000000000040 0x0000000000400040 0x0000000000400040
                 0x00000000000001f8 0x00000000000001f8  R E    8
  INTERP         0x0000000000000238 0x0000000000400238 0x0000000000400238
                 0x000000000000001c 0x000000000000001c  R      1
      [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
  LOAD           0x0000000000000000 0x0000000000400000 0x0000000000400000
                 0x00000000000006f4 0x00000000000006f4  R E    200000
  LOAD           0x0000000000000e00 0x0000000000600e00 0x0000000000600e00
                 0x0000000000000210 0x0000000000000218  RW     200000
  DYNAMIC        0x0000000000000e10 0x0000000000600e10 0x0000000000600e10
                 0x00000000000001c0 0x00000000000001c0  RW     8
  NOTE           0x0000000000000254 0x0000000000400254 0x0000000000400254
                 0x0000000000000020 0x0000000000000020  R      4
  GNU_EH_FRAME   0x00000000000005a4 0x00000000004005a4 0x00000000004005a4
                 0x000000000000003c 0x000000000000003c  R      4
  GNU_STACK      0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000000000 0x0000000000000000  RW     10
  GNU_RELRO      0x0000000000000e00 0x0000000000600e00 0x0000000000600e00
                 0x0000000000000200 0x0000000000000200  R      1

 Section to Segment mapping:
  Segment Sections...
   00
   01     .interp
   02     .interp .note.ABI-tag .hash .dynsym .dynstr .gnu.version .gnu.version_r .rela.dyn .init .plt .plt.got .text .fini .rodata .eh_frame_hdr .eh_frame
   03     .init_array .fini_array .dynamic .got .data .bss
   04     .dynamic
   05     .note.ABI-tag
   06     .eh_frame_hdr
   07
   08     .init_array .fini_array .dynamic .got
```

Recent stuff (~5 years)

- `-fsanitize*`
- `-mmitigate-rop (wtf` 🤔`)`
- `-fstack-clash-protection`
- `-fcf-protection=[full|branch|return|none]`
- There are some flags that caught my interest for edge cases but I need to do further research on them

Summary

- There are numerous compiler flags related to security and it's useful to know them (both for your compiler and target architecture)
- There are tradeoffs (e.g. noticeable execution slow down for `-fstack-protector-all`) so you should know how these mitigations work
- Landscape haven't changed that much in the last decade

https://dyjak.me

Twitter: @andrzejdyjak

Github: @dyjakan