
华中科技大学
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

网络空间安全学院

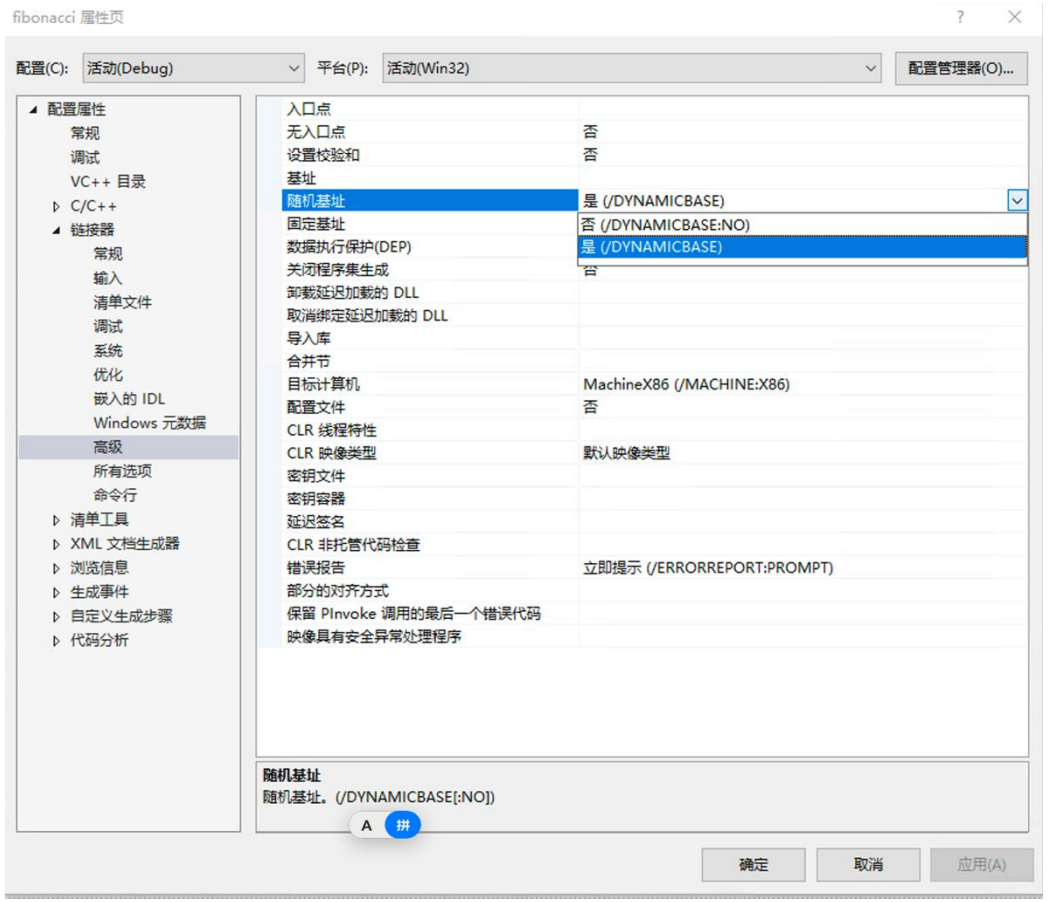

# 地址空间布局随机化-ASLR

网络空间安全学院 慕冬亮

Email : dzm91@hust.edu.cn

# 地址空间布局随机化-ASLR

- 部署Shellcode
  - 知晓堆或者栈的地址
  - 借用程序自身或者库中的代码（如jmp ESP - ROP）
- ASLR的思想
  - 栈和堆的基址是加载时随机确定的
  - 程序自身和关联库的基址是加载时随机确定的



# ASLR(Linux)

- 内存随机化：ASLR的核心思想是在程序每次启动时随机化其关键数据（如栈、堆、和库）的内存地址。

- 与其他安全技术的结合：虽然ASLR本身是一个强大的安全措施，但它通常与其他技术（如堆栈保护、执行保护（NX）等）结合使用，以提供更全面的保护。

- 系统级实现：在Linux系统中，ASLR是由内核实现的。这意味着它适用于所有运行在该系统上的应用程序。用户可以通过内核参数调整ASLR的行为。

# 地址空间布局随机化-ASLR(Linux)

- 开启/关闭ASLR
- echo 2/1/0 > /proc/sys/kernel/randomize_va_space

```
liber-MS-7D42# echo 0 > /proc/sys/kernel/randomize_va_space
liber-MS-7D42# cat /proc/sys/kernel/randomize_va_space
0
liber-MS-7D42#
```

| 级别 | 说明 |
| --- | --- |
| 0 | 关闭ASLR |
| 1 | 保留的随机化。共享库、栈、mmap() 以及 VDSO 将被随机化。 |
| 2 | 完全的随机化。在 1 的基础上，通过 brk() 分配的内存空间也将被随机化。 |

# 地址空间布局随机化-ASLR(Linux)

- 测试ASLR
- 如果想要程序地址随机化，则需要开启PIE配合使用

```
liber-MS-7D42# echo 0 > /proc/sys/kernel/randomize_va_space
liber-MS-7D42# su liber

# liber @ liber-MS-7D42 in ~/Downloads/software-security-dojo/integer-
$ ./aslr_test
stack_var: 0x7fffffffdfe0
 mmap_var: 0x7ffff7ff8000
 heap_var: 0x4ce7b0
 bss_var: 0x4c72f0
 data_var: 0x4c50f0
 text_var: 0x401775

# liber @ liber-MS-7D42 in ~/Downloads/software-security-dojo/integer-
$ ./aslr_test
stack_var: 0x7fffffffdfe0
 mmap_var: 0x7ffff7ff8000
 heap_var: 0x4ce7b0
 bss_var: 0x4c72f0
 data_var: 0x4c50f0
 text_var: 0x401775
```

```
# liber @ liber-MS-7D42 in ~/Downlo
$ ./aslr_test
stack_var: 0x7ffe17939400
 mmap_var: 0x7f5e234ac000
 heap_var: 0x1b267b0
 bss_var: 0x4c72f0
 data_var: 0x4c50f0
 text_var: 0x401775

# liber @ liber-MS-7D42 in ~/Downlo
$ ./aslr_test
stack_var: 0x7fff3278cb30
 mmap_var: 0x7f088e191000
 heap_var: 0x1d727b0
 bss_var: 0x4c72f0
 data_var: 0x4c50f0
 text_var: 0x401775

# liber @ liber-MS-7D42 in ~/Downlo
$
```

# 地址空间布局随机化-ASLR(Linux)

- 配合PIE，通过编译-no-pie ，-pie 参数控制pie的关闭/开启，pie默认开启。

```
$ gcc aslr_test.c -o aslr_pie

# liber @ liber-MS-7D42 in ~/Downloads/software-security-dojo/
$ checksec --file=aslr_pie
RELRO           STACK CANARY      NX            PIE
Full RELRO      Canary found      NX enabled    PIE enabled
```

```
$ gcc aslr_test.c -o aslr_no_pie -no-pie

# liber @ liber-MS-7D42 in ~/Downloads/software-security-dojo/
$ checksec --file=aslr_no_pie
RELRO           STACK CANARY      NX            PIE
Partial RELRO   Canary found      NX enabled    No PIE
```

# 地址空间布局随机化-ASLR(Linux)

- 使用-pie参数开启pie后，可以进行对比，程序段进行了随机化的加载。
- 程序的实际运行地址 = 程序加载基址 + 程序偏移地址

```
$ ./aslr_pie
stack_var: 0x7fffffffe130
 mmap_var: 0x7ffff7ffa000
 heap_var: 0x5555555592a0
 bss_var: 0x555555558018
 data_var: 0x555555558010
 text_var: 0x555555555209
%
```

```
$ ./aslr_no_pie
stack_var: 0x7fffffffe120
 mmap_var: 0x7ffff7ffa000
 heap_var: 0x4052a0
 bss_var: 0x404068
 data_var: 0x404060
 text_var: 0x4011f6
%
```