02.ASLR

List

ASLR

- **ASLR(Address Space Layout Randomization)이란?**
  - 메모리 손상 취약점 공격을 방지 하기 위한 기술 입니다.
  - 스택, 힙, 라이브러리, 등의 주소를 랜덤한 영역에 배치하여, 공격에 필요한 Target address를 예측하기 어렵게 만듭니다.
  - 프로그램이 실행 될 때 마다 각 주소들이 변경됩니다.
- **예를 들어 Return-to-libc 공격을 하기 위해서는 공유 라이브러리에서 사용하려는 함수의 주소를 알아야 합니다.**
  - 이러한 주소 값들이 프로그램이 호출 될때 마다 고정적인 주소를 가진다면 매우 쉽게 활용할 수 있습니다.
  - 하지만 ASLR의 적용으로 인해 프로그램이 호출 될때 마다 스택, 힙, 라이브러리 영역의 주소가 변경되면 공격에 어려워집니다.(불가능하지는 않습니다.)

Set ASLR

cmd
echo 0 > /proc/sys/kernel/randomize_va_space

option
0 : ASLR 해제 1 : 랜덤 스택 & 랜덤 라이브러리 설정 2 : 랜덤 스택 & 랜덤 라이브러리 & 랜덤 힙 설정

Check the protection techniques of binary files.

checksec.sh

- **Checksec.sh에서 다음과 같은 결과를 출력합니다.**
  - System-wide ASLR (kernel.randomize_va_space): On (Setting: 2)

checksec.sh --proc ASLR

```
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR
[Heap]  address: 0x1137010
[Stack] address: 0x7fffcfcd6b20
[libc]  address: 0x7f9526812830
^Z
[1]+  Stopped                 ./ASLR
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ checksec.sh --proc ASLR
* System-wide ASLR (kernel.randomize_va_space): On (Setting: 2)

  Description - Make the addresses of mmap base, heap, stack and VDSO page randomized.
  This, among other things, implies that shared libraries will be loaded to random
  addresses. Also for PIE-linked binaries, the location of code start is randomized.

  See the kernel file 'Documentation/sysctl/kernel.txt' for more details.

* Does the CPU support NX: Yes

         COMMAND    PID RELRO             STACK CANARY           NX/PaX        PIE
            ASLR  10366 Partial RELRO     Canary found           NX enabled    No PIE
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$
```

Memory map

- **다음과 같이 메모리의 변화를 확인 할 수 있습니다.**
  - "/proc/<PID>/maps" 파일을 통해 프로세스의 메모리 구조 및 주소를 확인 할 수 있습니다.
  - randomize_va_space에 2를 설정한 환경입니다.
  - 프로그램을 처음 실행했을 때와 두번째 실행했을 때의 메모리 배치가 다른 것을 확인 할 수 있습니다.

Memory area	첫번째 실행	두번째 실행
Stack	0x7ffea624c000 ~ 0x7ffea626d000	0x7ffc75b23000 ~ 0x7ffc75b44000
Heap	0x0229e000 ~ 0x022bf000	0x01198000 ~ 0x011b9000
Libc	0x7fb51e026000 ~ 0x7fb51e618000	0x7f6adb39b000 ~ 0x7f6adb98d000

/proc/self/maps

```
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR
[Heap]  address: 0x229e010
[Stack] address: 0x7ffea626a3e0
[libc]  address: 0x7fb51e046830
^Z
[1]+  Stopped                 ./ASLR
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ps -ef |grep ASLR
lazenca+   8521   6173  0 02:12 pts/4    00:00:00 ./ASLR
lazenca+   8523   6369  0 02:12 pts/18   00:00:00 grep --color=auto ASLR
lazenca0x0@ubuntu:~/Documents/Definition/protection$ cat /proc/8521/maps
00400000-00401000 r-xp 00000000 08:01 551811                             /home/lazenca0x0/Documents/Definition
/protection/ASLR/ASLR
00600000-00601000 r--p 00000000 08:01 551811                             /home/lazenca0x0/Documents/Definition
/protection/ASLR/ASLR
00601000-00602000 rw-p 00001000 08:01 551811                             /home/lazenca0x0/Documents/Definition
/protection/ASLR/ASLR
0229e000-022bf000 rw-p 00000000 00:00 0                                  [heap]
7fb51e026000-7fb51e1e6000 r-xp 00000000 08:01 655589                     /lib/x86_64-linux-gnu/libc-2.23.so
7fb51e1e6000-7fb51e3e6000 ---p 001c0000 08:01 655589                     /lib/x86_64-linux-gnu/libc-2.23.so
7fb51e3e6000-7fb51e3ea000 r--p 001c0000 08:01 655589                     /lib/x86_64-linux-gnu/libc-2.23.so
7fb51e3ea000-7fb51e3ec000 rw-p 001c4000 08:01 655589                     /lib/x86_64-linux-gnu/libc-2.23.so
7fb51e3ec000-7fb51e3f0000 rw-p 00000000 00:00 0
7fb51e3f0000-7fb51e416000 r-xp 00000000 08:01 655548                     /lib/x86_64-linux-gnu/ld-2.23.so
7fb51e5f6000-7fb51e5f9000 rw-p 00000000 00:00 0
7fb51e613000-7fb51e615000 rw-p 00000000 00:00 0
7fb51e615000-7fb51e616000 r--p 00025000 08:01 655548                     /lib/x86_64-linux-gnu/ld-2.23.so
7fb51e616000-7fb51e617000 rw-p 00026000 08:01 655548                     /lib/x86_64-linux-gnu/ld-2.23.so
7fb51e617000-7fb51e618000 rw-p 00000000 00:00 0
7ffea624c000-7ffea626d000 rw-p 00000000 00:00 0                          [stack]
7ffea635f000-7ffea6361000 r--p 00000000 00:00 0                          [vvar]
7ffea6361000-7ffea6363000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR
[Heap]  address: 0x1198010
[Stack] address: 0x7ffc75b423e0
[libc]  address: 0x7f6adb3bb830
^Z
[2]+  Stopped                 ./ASLR
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ps -ef |grep ASLR
lazenca+   8539   6173  0 02:13 pts/4    00:00:00 ./ASLR
lazenca+   8541   6369  0 02:13 pts/18   00:00:00 grep --color=auto ASLR
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ cat /proc/8539/maps
00400000-00401000 r-xp 00000000 08:01 551811                             /home/lazenca0x0/Documents/Definition
/protection/ASLR/ASLR
00600000-00601000 r--p 00000000 08:01 551811                             /home/lazenca0x0/Documents/Definition
/protection/ASLR/ASLR
00601000-00602000 rw-p 00001000 08:01 551811                             /home/lazenca0x0/Documents/Definition
/protection/ASLR/ASLR
01198000-011b9000 rw-p 00000000 00:00 0                                  [heap]
7f6adb39b000-7f6adb55b000 r-xp 00000000 08:01 655589                     /lib/x86_64-linux-gnu/libc-2.23.so
7f6adb55b000-7f6adb75b000 ---p 001c0000 08:01 655589                     /lib/x86_64-linux-gnu/libc-2.23.so
7f6adb75b000-7f6adb75f000 r--p 001c0000 08:01 655589                     /lib/x86_64-linux-gnu/libc-2.23.so
7f6adb75f000-7f6adb761000 rw-p 001c4000 08:01 655589                     /lib/x86_64-linux-gnu/libc-2.23.so
7f6adb761000-7f6adb765000 rw-p 00000000 00:00 0
7f6adb765000-7f6adb78b000 r-xp 00000000 08:01 655548                     /lib/x86_64-linux-gnu/ld-2.23.so
7f6adb96b000-7f6adb96e000 rw-p 00000000 00:00 0
7f6adb988000-7f6adb98a000 rw-p 00000000 00:00 0
7f6adb98a000-7f6adb98b000 r--p 00025000 08:01 655548                     /lib/x86_64-linux-gnu/ld-2.23.so
7f6adb98b000-7f6adb98c000 rw-p 00026000 08:01 655548                     /lib/x86_64-linux-gnu/ld-2.23.so
7f6adb98c000-7f6adb98d000 rw-p 00000000 00:00 0
7ffc75b23000-7ffc75b44000 rw-p 00000000 00:00 0                          [stack]
7ffc75b7a000-7ffc75b7c000 r--p 00000000 00:00 0                          [vvar]
7ffc75b7c000-7ffc75b7e000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$
```

Linux 폴더 정보

/proc : process의 줄임말이며, 이 디렉토리에 프로세스의 정보들이 저장됩니다.
/proc/self : 현재 실행되고 있는 프로세스의 정보가 담겨있는 디렉토리입니다.
/proc/self/maps : 현재 실행되고 있는 프로세스의 주소 맵입니다.

Example

Source code

- **아래 코드는 다음과 같은 동작을 합니다.**
  - heap, stack, libc 의 주소를 출력합니다.

ASLR Test

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

char *global = "Lazenca.0x0";

int main(){
        char *heap = malloc(100);
        char *stack[] = {"LAZENCA.0x0"};

        printf("[Heap]  address: %p\n", heap);
        printf("[Stack] address: %p\n", stack);
        printf("[libc]  address: %p\n",**(&stack + 3));
        printf("[.data] address: %p\n",global);
        gets(heap);
        return 0;
}
```

echo 0 > /proc/sys/kernel/randomize_va_space

- **다음과 같이 프로그램을 실행할 때 마다 Heap, Stack, Libc의 주소영역이 변경되지 않습니다.**

echo 0 > /proc/sys/kernel/randomize_va_space

```
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR
[Heap]  address: 0x602010
[Stack] address: 0x7fffffffe1f0
[libc]  address: 0x7ffff7a2d830
[.data] address: 0x400764
^C
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR
[Heap]  address: 0x602010
[Stack] address: 0x7fffffffe1f0
[libc]  address: 0x7ffff7a2d830
[.data] address: 0x400764
^C
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR
[Heap]  address: 0x602010
[Stack] address: 0x7fffffffe1f0
[libc]  address: 0x7ffff7a2d830
[.data] address: 0x400764
^C
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$
```

echo 1 > /proc/sys/kernel/randomize_va_space

- **다음과 같이 프로그램을 실행할 때 마다 Stack, Libc의 주소 영역이 변경됩니다.**

echo 0 > /proc/sys/kernel/randomize_va_space

```
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR
[Heap]  address: 0x602010
[Stack] address: 0x7ffc5bcf1640
[libc]  address: 0x7f37ba468830
[.data] address: 0x400764
^C
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR
[Heap]  address: 0x602010
[Stack] address: 0x7fff2ea773e0
[libc]  address: 0x7f6bd7256830
[.data] address: 0x400764
^C
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR
[Heap]  address: 0x602010
[Stack] address: 0x7ffd2f5f9ae0
[libc]  address: 0x7fba0f3b3830
[.data] address: 0x400764
^C
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$
```

echo 2 > /proc/sys/kernel/randomize_va_space

- **다음과 같이 프로그램을 실행할 때 마다 Heap, Stack, Libc의 주소 영역이 변경됩니다.**

echo 0 > /proc/sys/kernel/randomize_va_space

```
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR
[Heap]  address: 0x1ccb010
[Stack] address: 0x7ffe56b07500
[libc]  address: 0x7f896e125830
[.data] address: 0x400764
^C
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR
[Heap]  address: 0x127b010
[Stack] address: 0x7fffeb87cf70
[libc]  address: 0x7eff87928830
[.data] address: 0x400764
^C
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR
[Heap]  address: 0x179b010
[Stack] address: 0x7fff04362ea0
[libc]  address: 0x7f16e3ad6830
[.data] address: 0x400764
^C
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$
```

.data

- **앞의 예제에서 ASLR의 설정을 2로 설정하였으나 .data 영역의 주소는 변경되지 않습니다.**
- 해당 영역을 주소도 매번 새로운 주소에 할당하기 위해서는 PIE를 적용해야 합니다.
    - 해당 설명은 06.PIE 의 내용을 참고 하세요

```
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR_PIE
[Heap]  address: 0x55f20dc53010
[Stack] address: 0x7ffcbb14a3d0
[libc]  address: 0x7ff33b9a9830
[.data] address: 0x55f20ca499b4
^C
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR_PIE
[Heap]  address: 0x55fe918f7010
[Stack] address: 0x7fffe2f15390
[libc]  address: 0x7ff2eb2b8830
[.data] address: 0x55fe916ed9b4
^C
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ ./ASLR_PIE
[Heap]  address: 0x55a53fd1d010
[Stack] address: 0x7ffe38573570
[libc]  address: 0x7f450f523830
[.data] address: 0x55a53de679b4
^C
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$
```

How to detect ASLR in the "Checksec.sh" file

- **다음과 같은 방법으로 시스템의 ASLR 설정여부를 확인합니다.**
  - "/proc/1/status" 파일 내에 'PaX'단어를 검색하고 검색 결과에서 'R'이 존재하는지 확인합니다.
    - 위 조건을 모두 만족하면 "ASLR enabled"로 판단합니다.
  - "/proc/1/status" 파일 내에 'PaX'라는 단어가 없을 경우 'sysctl' 명령어를 이용해 확인합니다.
    - 'sysctl' 명령어를 통해 출력된 내용 중 "kernel.randomize_va_space ="의 값을 확인해 ASLR 설정을 판단합니다.

Checksec.sh - line 285

```
# check for system-wide ASLR support
aslrcheck() {
  # PaX ASLR support
  if !(cat /proc/1/status 2> /dev/null | grep -q 'Name:') ; then
    echo -n -e ':\033[33m insufficient privileges for PaX ASLR checks\033[m\n'
    echo -n -e '  Fallback to standard Linux ASLR check'
  fi

  if cat /proc/1/status 2> /dev/null | grep -q 'PaX:'; then
    printf ": "
    if cat /proc/1/status 2> /dev/null | grep 'PaX:' | grep -q 'R'; then
      echo -n -e '\033[32mPaX ASLR enabled\033[m\n\n'
    else
      echo -n -e '\033[31mPaX ASLR disabled\033[m\n\n'
    fi
  else
    # standard Linux 'kernel.randomize_va_space' ASLR support
    # (see the kernel file 'Documentation/sysctl/kernel.txt' for a detailed description)
    printf " (kernel.randomize_va_space): "
    if /sbin/sysctl -a 2>/dev/null | grep -q 'kernel.randomize_va_space = 1'; then
      echo -n -e '\033[33mOn (Setting: 1)\033[m\n\n'
      printf "  Description - Make the addresses of mmap base, stack and VDSO page randomized.\n"
      printf "  This, among other things, implies that shared libraries will be loaded to \n"
      printf "  random addresses. Also for PIE-linked binaries, the location of code start\n"
      printf "  is randomized. Heap addresses are *not* randomized.\n\n"
    elif /sbin/sysctl -a 2>/dev/null | grep -q 'kernel.randomize_va_space = 2'; then
      echo -n -e '\033[32mOn (Setting: 2)\033[m\n\n'
      printf "  Description - Make the addresses of mmap base, heap, stack and VDSO page randomized.\n"
      printf "  This, among other things, implies that shared libraries will be loaded to random \n"
      printf "  addresses. Also for PIE-linked binaries, the location of code start is randomized.\n\n"
    elif /sbin/sysctl -a 2>/dev/null | grep -q 'kernel.randomize_va_space = 0'; then
      echo -n -e '\033[31mOff (Setting: 0)\033[m\n'
    else
      echo -n -e '\033[31mNot supported\033[m\n'
    fi
    printf "  See the kernel file 'Documentation/sysctl/kernel.txt' for more details.\n\n"
  fi
}
```

- **먼저 "/proc/1/status" 파일을 이용해 ASLR 설정여부를 확인합니다.**
  - 하지만 해당 시스템에서는 "/proc/1/status"를 이용해 ASLR이 적용되었는지 확인 할 수 없습니다.
  - 해당 파일에 "Pax" 정보가 없기 때문입니다.
- **그렇기 때문에 다음과 같이 'sysctl' 명령어를 이용해 ASLR 설정 여부를 판단할 수 있습니다.**
  - 해당 시스템의 설정 값은 '2' 입니다.

sysctl -a | grep 'kernel.randomize_va_space = '

```
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ cat /proc/1/status |grep 'Name'
Name:        systemd
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ cat /proc/1/status |grep 'Pax'
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$ sysctl -a | grep 'kernel.randomize_va_space = '
sysctl: permission denied on key 'fs.protected_hardlinks'
sysctl: permission denied on key 'fs.protected_symlinks'
sysctl: permission denied on key 'kernel.cad_pid'
kernel.randomize_va_space = 2
sysctl: permission denied on key 'kernel.unprivileged_userns_apparmor_policy'
sysctl: permission denied on key 'kernel.usermodehelper.bset'
sysctl: permission denied on key 'kernel.usermodehelper.inheritable'
sysctl: permission denied on key 'net.ipv4.tcp_fastopen_key'
sysctl: permission denied on key 'net.ipv6.conf.all.stable_secret'
sysctl: permission denied on key 'net.ipv6.conf.default.stable_secret'
sysctl: permission denied on key 'net.ipv6.conf.ens33.stable_secret'
sysctl: permission denied on key 'net.ipv6.conf.lo.stable_secret'
lazenca0x0@ubuntu:~/Documents/Definition/protection/ASLR$
```

Related information

- N / a