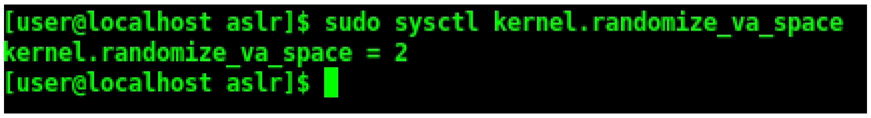
RedHat Linux Kernel Internals Laboratory Exercises

Lab 20: ASLR

Objective: In this lab you view the Linux kernel's implementation of ASLR in action.

1. Verify the current state of ASLR in your Linux VM. The options are explained in (Documentation/sysctl/kernel.txt), 2= Full randomization.

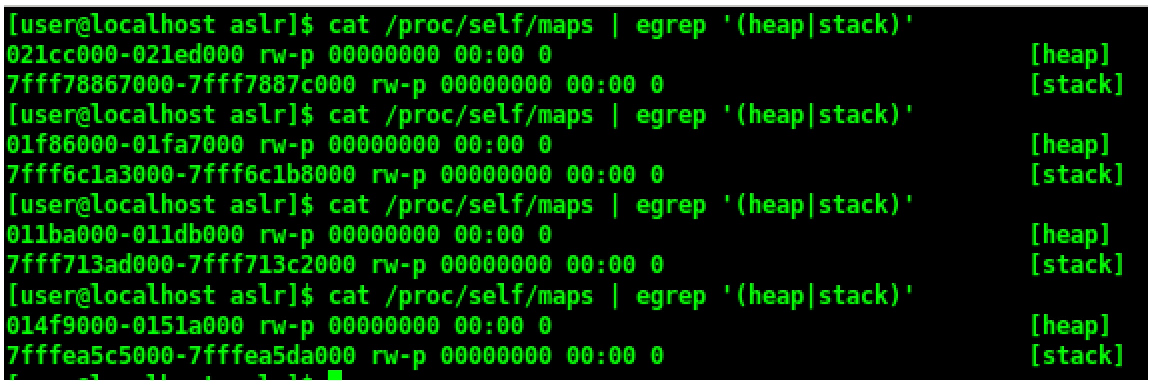


```
[user@localhost aslr]$ sudo sysctl kernel.randomize_va_space
kernel.randomize_va_space = 2
[user@localhost aslr]$ 
```

```
randomize_va_space:

This option can be used to select the type of process address
space randomization that is used in the system, for architectures
that support this feature.

0 - Turn the process address space randomization off.  This is the
    default for architectures that do not support this feature anyways,
    and kernels that are booted with the "norandmaps" parameter.

1 - Make the addresses of mmap base, 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.  This is the default if the
    CONFIG_COMPAT_BRK option is enabled.

2 - Additionally enable heap randomization.  This is the default if
    CONFIG_COMPAT_BRK is disabled.

    There are a few legacy applications out there (such as some ancient
    versions of libc.so.5 from 1996) that assume that brk area starts
    just after the end of the code+bss.  These applications break when
    start of the brk area is randomized.  There are however no known
    non-legacy applications that would be broken this way, so for most
    systems it is safe to choose full randomization.
```

2. To demonstrate the effectiveness of ASLR on user mode applications execute the following command and observe the memory values.

```
cat /proc/self/maps | egrep '(heap|stack')
```



```
[user@localhost aslr]$ cat /proc/self/maps | egrep '(heap|stack)'
021cc000-021ed000 rw-p 00000000 00:00 0                              [heap]
7fff78867000-7fff7887c000 rw-p 00000000 00:00 0                      [stack]
[user@localhost aslr]$ cat /proc/self/maps | egrep '(heap|stack)'
01f86000-01fa7000 rw-p 00000000 00:00 0                              [heap]
7fff6c1a3000-7fff6c1b8000 rw-p 00000000 00:00 0                      [stack]
[user@localhost aslr]$ cat /proc/self/maps | egrep '(heap|stack)'
011ba000-011db000 rw-p 00000000 00:00 0                              [heap]
7fff713ad000-7fff713c2000 rw-p 00000000 00:00 0                      [stack]
[user@localhost aslr]$ cat /proc/self/maps | egrep '(heap|stack)'
014f9000-0151a000 rw-p 00000000 00:00 0                              [heap]
7fffea5c5000-7fffea5da000 rw-p 00000000 00:00 0                      [stack]
```

3. Run this command several more times.

4. Based on what we have learned, how does ASLR affect modern exploitation?

5. Is this a Band-Aid solution? What about KASLR?