On the Effectiveness of Address-Space Randomization

20204222 강우석

Background

Exploits

Buffer overflow

Countermeasures

- StackGuard (canary)

- StackShield

W$\oplus$X

For heap, stack, and other memory segments

W

writable

Not both

X

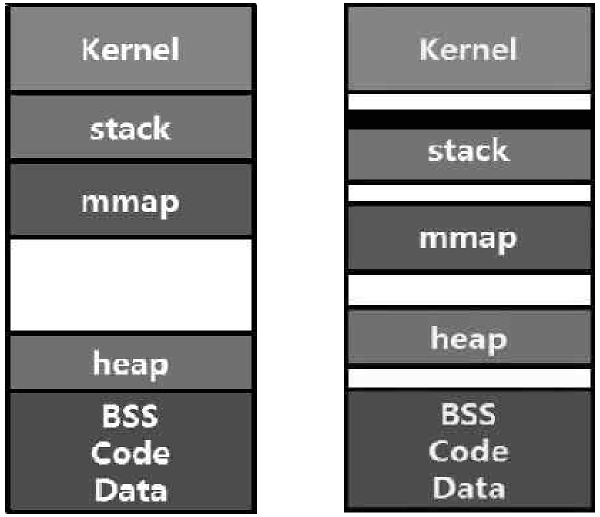
executable

RTL (return to libc)

- Libc : Standard C-language library
- Loaded into every Unix program
- Contains many useful functions

Address-Space Randomization

PaX ASLR randomizes the base address of the stack, heap, code and mmap()ed segments.



(a) ASLR disabled (b)ASLR enabled

Address-Space Randomization

```
eqkws@kali: ~/workspace/2020F_WooseokKang_CTF/bullseye
파일(F) 동작(A) 편집(E) 보기(V) 도움말(H)
eqkws@kali:~/workspace/2020F_WooseokKang_CTF/bullseye$ ldd bullseye
        linux-vdso.so.1 (0x00007ffe883e3000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fdcb04be000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fdcb069a000)
eqkws@kali:~/workspace/2020F_WooseokKang_CTF/bullseye$ ldd bullseye
        linux-vdso.so.1 (0x00007ffcf3e5e000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ff49d5a9000)
        /lib64/ld-linux-x86-64.so.2 (0x00007ff49d785000)
eqkws@kali:~/workspace/2020F_WooseokKang_CTF/bullseye$
```

Address-Space Randomization

```
eqkws@kali: ~/workspace/2020F_WooseokKang_CTF/bullseye
파일(F)  동작(A)  편집(E)  보기(V)  도움말(H)
eqkws@kali:~/workspace/2020F_WooseokKang_CTF/bullseye$ echo 0 | sudo tee /proc/sys/kernel/randomize_va_space
0
eqkws@kali:~/workspace/2020F_WooseokKang_CTF/bullseye$ ldd bullseye
        linux-vdso.so.1 (0x00007ffff7fd2000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ffff7df2000)
        /lib64/ld-linux-x86-64.so.2 (0x00007ffff7fd4000)
eqkws@kali:~/workspace/2020F_WooseokKang_CTF/bullseye$ ldd bullseye
        linux-vdso.so.1 (0x00007ffff7fd2000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ffff7df2000)
        /lib64/ld-linux-x86-64.so.2 (0x00007ffff7fd4000)
eqkws@kali:~/workspace/2020F_WooseokKang_CTF/bullseye$ ldd bullseye
        linux-vdso.so.1 (0x00007ffff7fd2000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ffff7df2000)
        /lib64/ld-linux-x86-64.so.2 (0x00007ffff7fd4000)
eqkws@kali:~/workspace/2020F_WooseokKang_CTF/bullseye$
```

How to break?

PaX ASLR

- Executable (executable code, initialized data, uninitialized data)

  16

- Mapped area (heap, dynamic libraries, thread stacks, shared memory)

  16

- Stack area (main user stack)

  24

PaX ASLR

- PaX ASLR ramdomizes only the base addresses of the three memory areas.

delta_mmap

- In PaX, each offset variable is fixed throughout a process's lifetime.

RTL Attack

Apache web server on Linux with PaX ASLR and W⊕X

```
char buf[64];
      ⋮
strcpy(buf,s); /* Overflow buffer */
```

ap_getline() in http_protocol.c

RTL Attack Scenario

1. Iterate all possible values for delta_mmap through 0 to 2^{16} = 65535.

2. For each value of delta_mmap, calculates the virtual address of usleep()

3. Create the attack buffer, and send it to the Apache web server.

4. If the connection closes immediately, continue with the next value of delta_mmap,
   otherwise, current guess for delta_mmap is correct.

RTL Attack Scenario

top of stack (higher addresses)
⋮
pointer into 64 byte buffer
`0xDEADBEEF`
address of `system()`
address of `ret` instruction
⋮
address of `ret` instruction
`0xDEADBEEF`
64 byte buffer (contains shell commands)
⋮
bottom of stack (lower addresses)

Experiments

Environment : 2.4 GHz Pentium 4 machine with PaX ASLR (for Linux kernel version 2.6.1)

protected Apache server running on a Athlon 1.8 GHz machine.

Average	Max	Min
216	810	29

(in seconds)

Improvements to ASLR

64-Bit Architecture

In 32-bit x86 machines

We can randomize the 16 of the 32 address bits

On the other hand, in 64-bit machine…

We can randomize maximum 40 address bits

$2^{40} = 1{,}099{,}511{,}627{,}776$

Randomization Frequency

How about if we randomize the address space layout of a process more frequent?

1. The address-space randomization is fixed during the duration of an attack.

2. The address-space randomization changes with each probe.

Randomization Frequency

Scenario 1.

$$\underbrace{\frac{2^n-1}{2^n}\cdot\frac{2^n-2}{2^n-1}\cdots\frac{2^n-t-1}{2^n-t}}_{\Pr[\text{first } t-1 \text{ probes fail}]}\cdot\frac{1}{2^n-t-1}=\frac{1}{2^n},$$

$$\sum_{t=1}^{2^n} t\cdot\frac{1}{2^n}=\frac{1}{2^n}\cdot\sum_{t=1}^{2^n} t=(2^n+1)/2\approx 2^{n-1}.$$

Scenario 2.

$p = 1/2^n$

$1/p = 2^n$

Conclusion

Conclusion

- For 32-bit architectures, address-space randomization is ineffective against the brute force.
- Present the effectiveness of more powerful randomization techniques.
- The most promising solution appears to be upgrading to a 64-bit architecture.

Thanks!