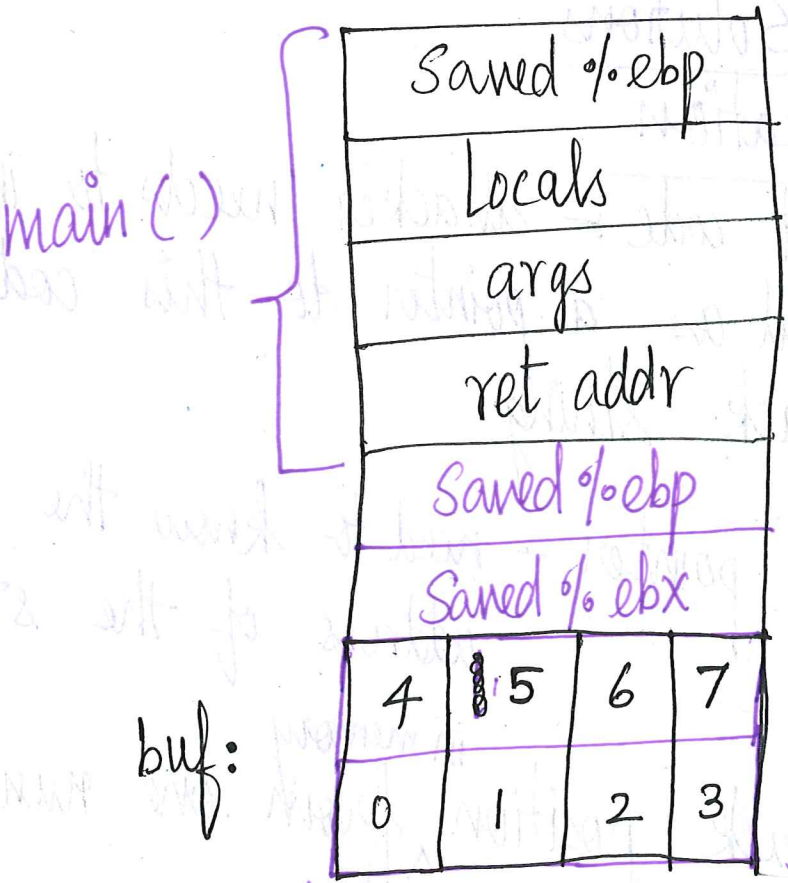
Lecture 22 - Assembly Wrap ①

Stack Smashing / Stack buffer overflow

* No bounds checking for arrays in C.
* stack - stores state information and local variables - trouble!




```
void echo()
{
    char buf[8];
    gets(buf);
    puts(buf);
}
```

Warning: (C compiler).

"The gets function is dangerous and should not be used."

Solution: use fgets.

```
fgets(buf, n, stdin);
```

(n-1) max characters to read.

Internet worm of Nov 1988 (or) Morris worm

* Robert Tappan Morris – Nov 2, 1988 – Grad stud, Cornell.
* Computer Fraud & Abuse Act
* 3 years probation, 400 hours of community service, and a $10,500 fine.

Solutions

1. Stack Randomization

To insert exploit code :- attacker needs to inject both code as well as a pointer to this code as part of the attack string.

To generate this pointer – need to know the stack address of the string.

Idea

Vary the stack position in memory from one run of the program to another.

2. Stack Corruption Detection.

gcc - stack protector (to detect buffer overruns).

idea - store a special canary value in the stack frame between any local buffer and the rest of the stack state.

bird used to detect the presence of dangerous gases in coal mines.

canary (guard value)

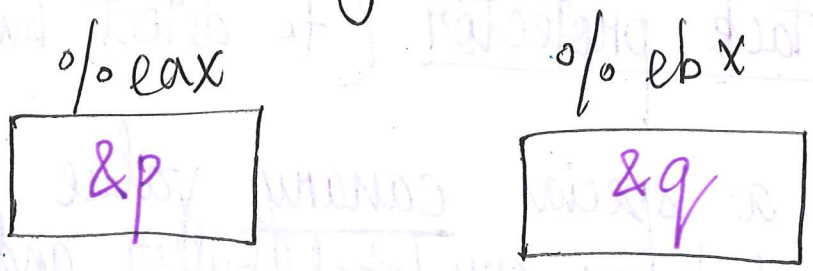
random number for each run.

check the "canary" for any buffer overflow!

END OF ASSEMBLY!

④ Assembly Review

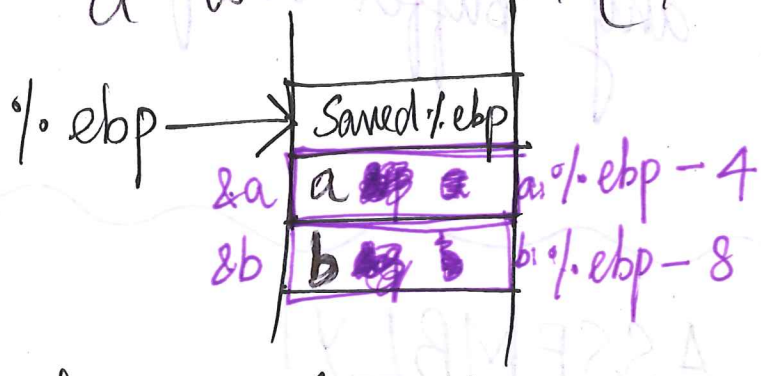
1.




```
movl (%eax), %esi
movl (%esi), %edi
movl %edi, (%ebx)
```

⇒ %esi: p ; %edi: *p ⇒ `q = *p;`

a is in -0x4(%ebp) | b is in -0x8(%ebp)

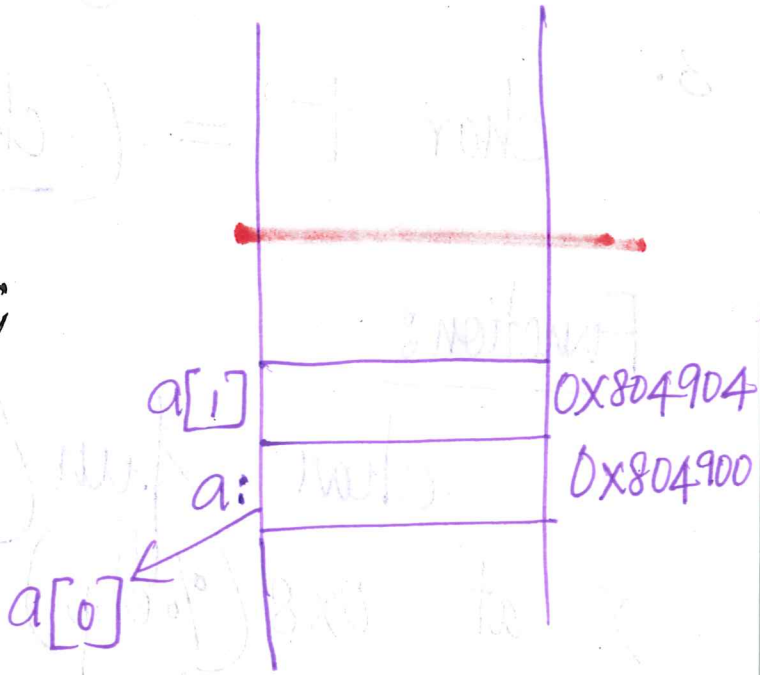


`b = *q;`

```
movl -0x4(%ebp), %eax
movl (%eax), %eax
movl %eax, -0x8(%ebp)
```

⇒ ~~`q = *p;`~~

2.
```
int a[10];
int *p = &a[7];
```

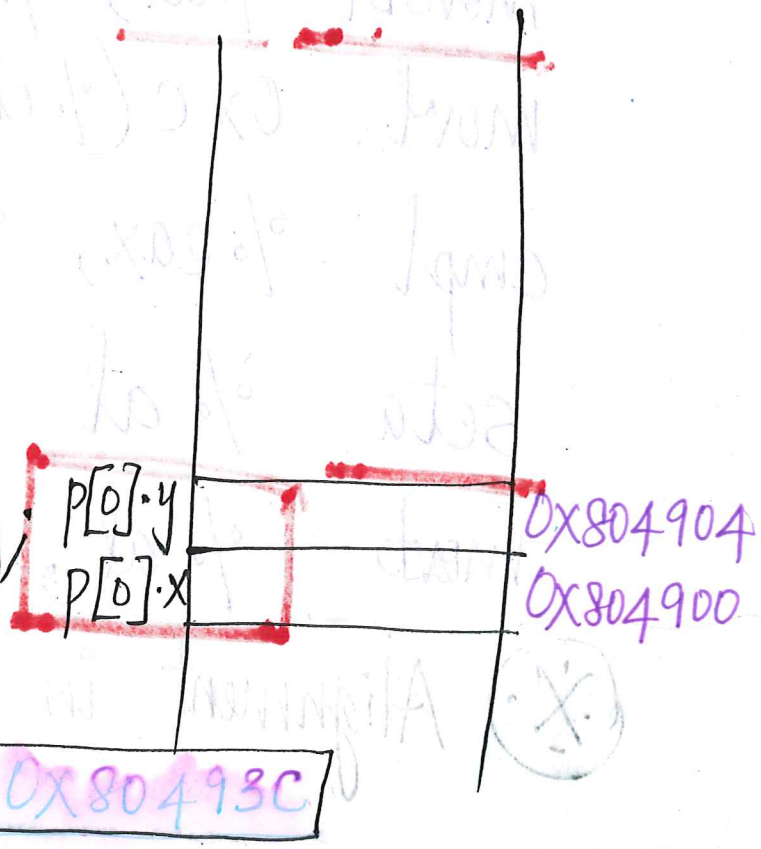


p = ?

7 X 4 = 28 = 0x1C

∴ a[7] is at 0x804900
+ 0x 1C
0x80491C

```
struct point {
    int x;
    int y;
};
struct point p[10];
int *ptr = &(p[7].y);
```



8 x 7 = 56 = 0x38

ptr = 0x804900
+ 0x 38
+ 0x 4

⇒ ptr = 0x80493C

3. `char t = (char) x > (unsigned) y;`

Function:

```
char fun(int x, int y);
```

x at 0x8(%ebp)
y at 0xc(%ebp).

Assembly:

```
movl   0x8(%ebp), %eax
movsbl %al, %edx
movl   0xc(%ebp), %eax
cmpl   %eax, %edx
seta   %al
movb   %al, -0x4(%ebp)
```

(X) Alignment in structures!