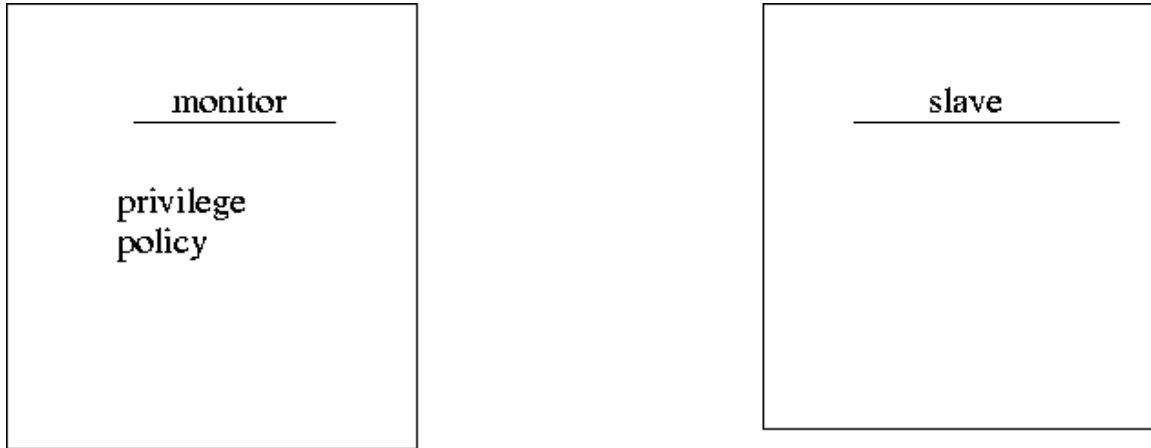


Privilege Separation

privileged program = privilege + policy (these two are bound together)



For privilege separation to be secure

- policy must go in monitor
- Separated program has a bug only if original has a bug
- If bug goes to slave (OK)
- If bug goes to monitor (??? could be worse, could be better)

Privilege separation is a good idea if done right

System solutions to buffer overflow

To exploit stack overflow:

1. find overflow in source
2. send an overflowing message to the program
3. overwrite return address with address of buffer
4. program returns to RA
5. executes shellcode
6. attacker makes system call

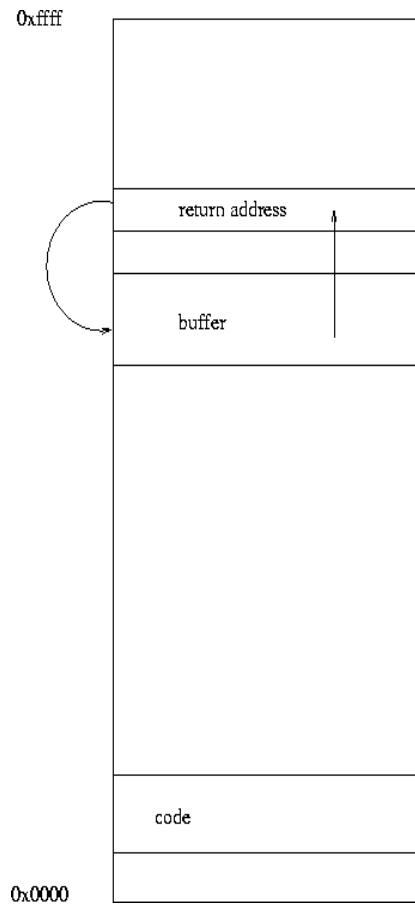
Solutions

1. static analysis: BOON
2. Ccured
3. address space randomization
PointGuard

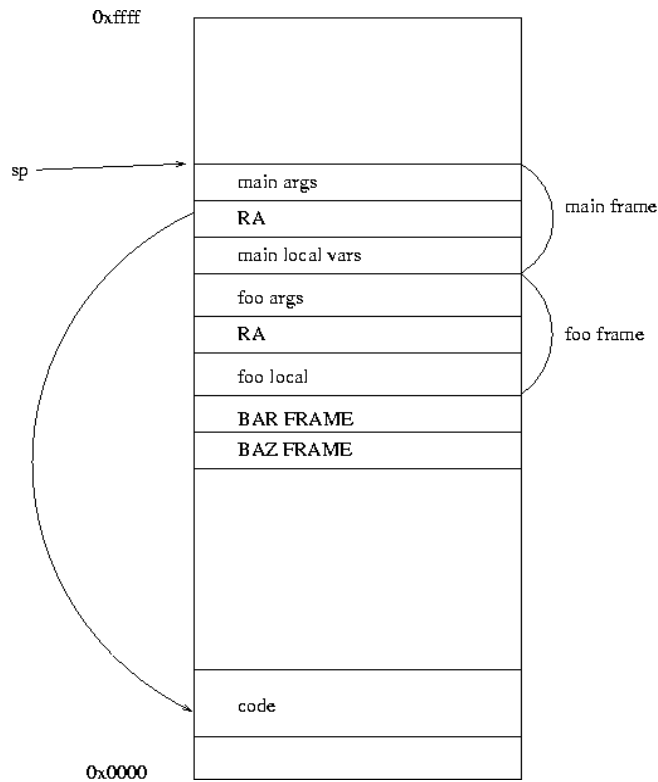
4. StackGuard

Attacker:

Needs to guess address of buffer. If attacker can run exact copy of program then he can use debugger to find the address of buffer.



A more detailed layout of the memory:



Randomization of the initial stack pointer doesn't prevent return to libc attacks. We can do more randomization:

Randomize

- stack top
- LIBC location
- code location
- data location
- inter frame padding
- rearrange functions
- rearrange basic blocks
- rearrange local variables
- rearrange data segment
- rearrange mallocs

on 32-bit system, attacker must guess ~22 bits to attack the system.

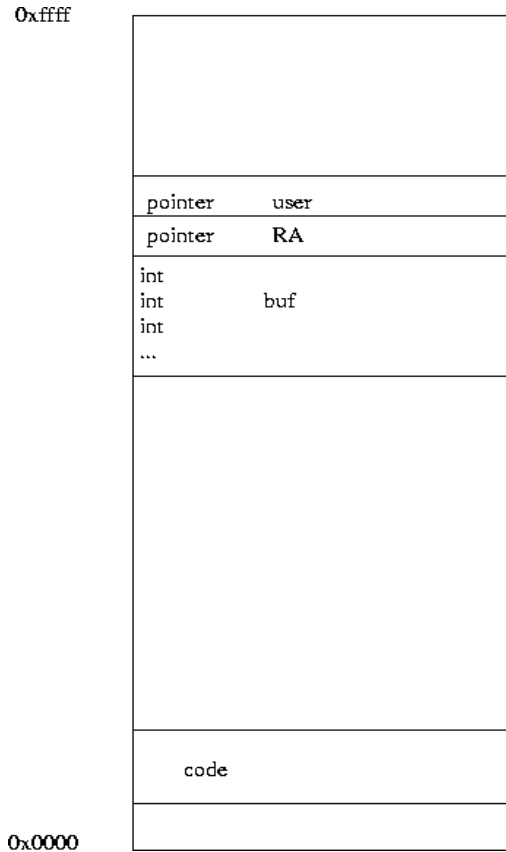
An example:

```
void foo (char * user)
```

```

{
    char buf[1024];
    strcpy(buf, usr);
    return;
}

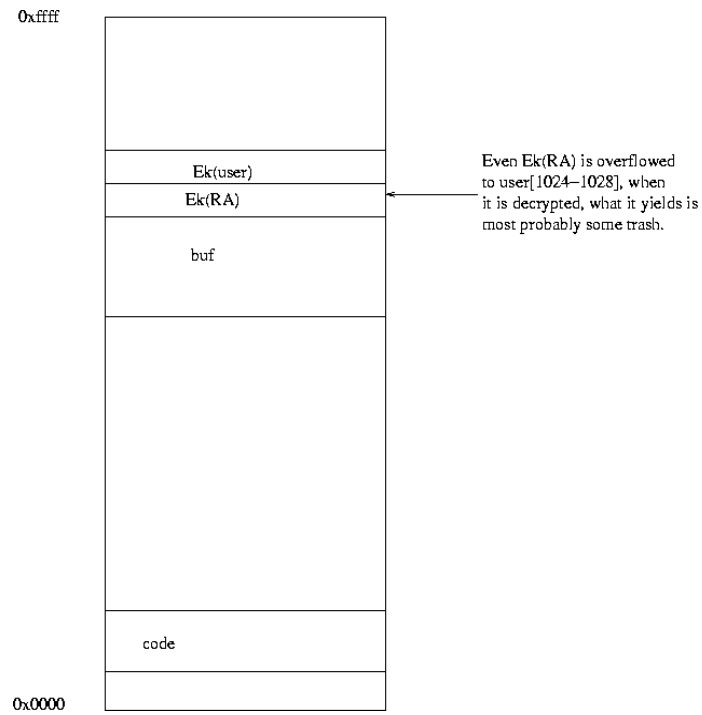
```



Add a tag to specify whether the memory word is pointer or not. For example, the user and RA should be pointer and buf is of type integer. So if a buffer overflow happens and RA is overwritten with int, the type of RA will be changed to int. When return, if we check that RA is not type of pointer, the overflow is detected.

However, the problem is that we don't have a tag to store whether it is a pointer or not. Alternative solution:

- pointers are stored in memory encrypted
- integers are not
- at startup, pick K randomly (K can be stored in a register to be secure)
- XOR



StackGuard

- at startup pick random canary
- check whether canary has been overflowed before using the RA

(Yet if one can launch a format string attack, he might be able to learn about the canary, so that he can fake the canary)

