More on format strings

- printf( <string of type char*>, <optional params>);
- If the string contains anything like "...%_.%_.%_.%_.", the "%_" will be pushed as optional params.
  - [...] | saved FP | ret addr | str | param 3 | param 2 | param 1 | ...... ]
  - If the str (or buf) has "%_" in it, printf will interpret adjacent memory.
  - [...] | save fp | ret addr | buf | .... | .... | .... | .......... ]
  - Recall that %n writes the "number of bytes printed so far" to a location in memory.

Also recall printf("%Md", 10) for M.

EXAMPLE:
func calls printf.
func has a local variable buf, which the attacker can completely control
call to printf uses buf as a parameter (so pointer to buf appears as parameter)
buf has % parameters to move printf stack to addresses of return address in buf
buf uses %Md for values of M to affect number of output characters
repeat for &RET, &RET+1, &RET+2, &RET+3.

TOCTOU - Time of check to time of use

Goal: open regular files, not symlinks
int openfile(char *path) {
    struct stat s;
    if (stat(path, &s) < 0)
        return -1;
    if (!S_ISRREG(s.st_mode)) {
        error("only allowed to regular files!");
        return -1;
    }
    return open(path, O_RDONLY);
}

Issue:
Attacker can change meaning of path between stat and open
(and access files he or she shouldn’t)
Integer Overflow and Implicit Cast

Consider this code:
```c
char buf[80];
void vulnerable() {
    int len = read_int_from_network();
    char *p = read_string_from_network();
    if (len > sizeof buf) {
        error("length too large, nice try!");
        return;
    }  
    memcpy(buf, p, len);
}
```
Recall
```c
void *memcpy(void *dst, const void *src, size_t n);
typedef unsigned int size_t;
```

Issue:

If len is negative, may copy huge amounts of input into buf

Also

Consider this code:
```c
size_t len = read_int_from_network();
char *buf;
buf = malloc(len+5);
read(fd, buf, len);
```

What if len is large (e.g., len = 0xFFFFFFFF)?
Then len + 5 = 4 (on many platforms)
Result: Allocate a 4-byte buffer, then read a lot of data into that buffer.

Timing Attack - timing deviation that can be exploited by the adversary.

Password check function - timing varies based on whether a particular char is correct or not.
If an adversary can enter passwords and time how long it takes the system to check that password, the adversary can try 62 combos for 1st char, then 2nd, etc.

- $62 \times 8$ tries instead of $62^8$
- Might have to use statistics and run multiple times to get a strong signal, but even some constant $\times 62 \times 8$ is way less than $62^8$.
- Adversary can potentially learn private information this way.