

# Format String Attack



## Format String

```
/* printf_example.c */  
  
#include <stdio.h>  
  
int main()  
{  
    int i=1, j=2, k=3;  
  
    printf("Hello World \n");  
    printf("Print one number: %d\n", i);  
    printf("Print two numbers: %d, %d\n", i, j);  
    printf("Print three numbers: %d, %d, %d\n", i, j, k);  
}
```

```
int printf(const char *format, ...);
```

foo(int a) ehpt 8

foo(2, 3) X

void foo(int a, ...);

foo(5, 4)

## Function With Varying Length of Arguments

```
#include <stdio.h>
#include <stdarg.h>

// Function with variable number of arguments
int myprint(int Narg, ... )
{
    va_list ap;
    int i;

    va_start(ap, Narg);

    for(i=0; i<Narg; i++) {
        // Print optional argument of type int
        printf("%d ", va_arg(ap, int));
        // Print optional argument of type double
        printf("%f ", va_arg(ap, double));
    }
    printf("\n");

    va_end(ap);
}

int main() {
    myprint(1, 2, 3.5);
    myprint(2, 3, 4.5, 4, 5.5);

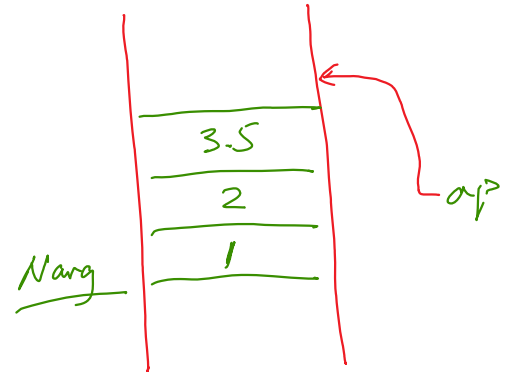
    return 1;
}
```

va\_arg

2  
3.5

optional arguments

va\_start (



# How printf() Accesses Optional Arguments

format specifier

```
#include <stdio.h>

int main()
{
    int id=100, age=25; char *name = "Bob Smith";
    printf("\n ID: %d, Name: %s, Age: %d\n", id, name, age);
}
```

va\_arg(ap, int)  
va\_arg(ap, (char\*))  
va\_arg(ap, int)

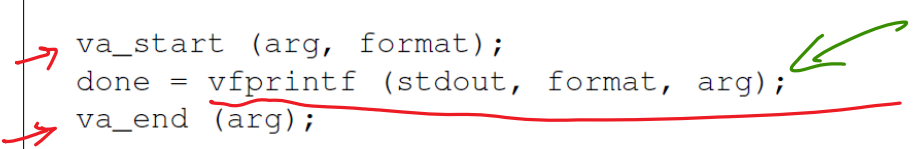
Parameter	Meaning	Passed as
<u>%d</u>	decimal ( <u>int</u> )	value
%u	unsigned decimal (unsigned int)	value
%x	hexadecimal (unsigned int)	value
%s	string ((const) (unsigned) char *)	reference
%n	number of bytes written so far, (* int)	reference

# How Format String Works

```
int __printf (const char *format, ...)
{
    va_list arg;
    int done;

    va_start (arg, format);
    done = vfprintf (stdout, format, arg);
    va_end (arg);

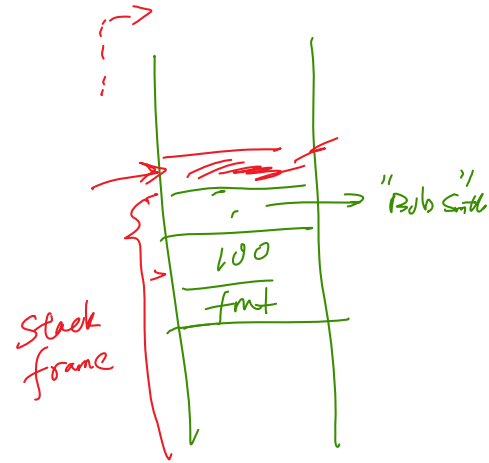
    return done;
}
```



## printf() With Missing Arguments

```
/* arg_mismatch.c */  
  
#include <stdio.h>  
  
int main()  
{  
    int id=100, age=25; char *name = "Bob Smith";  
  
    // Age parameter missing  
    printf("ID: %d, Name: %s, Age: %d\n", id, name);  
}
```

ID: 100, Name: Bob Smith, Age:



## Untrusted User Input Becomes Format String

Example 1:

```
printf(user_input)
```

printf("%s\n", user\_input)

Example 2:

```
| sprintf(format, "%s %s", user_input, " : %d");  
| printf(format, program_data);
```

Example 3:

```
| sprintf(format, "%s %s", getenv("PWD"), " : %d");  
| printf(format, program_data);
```

## A Vulnerable Program

Set-UID  
program

```
#include <stdio.h>
```

```
void fmtstr()
```

```
{
```

```
    char input[100];
```

```
    printf("Please enter a string: ");
```

```
    fgets(input, sizeof(input)-1, stdin);
```

```
    printf(input);
```

```
}
```

```
int main(int argc, char *argv[])
```

```
{
```

```
    fmtstr();
```

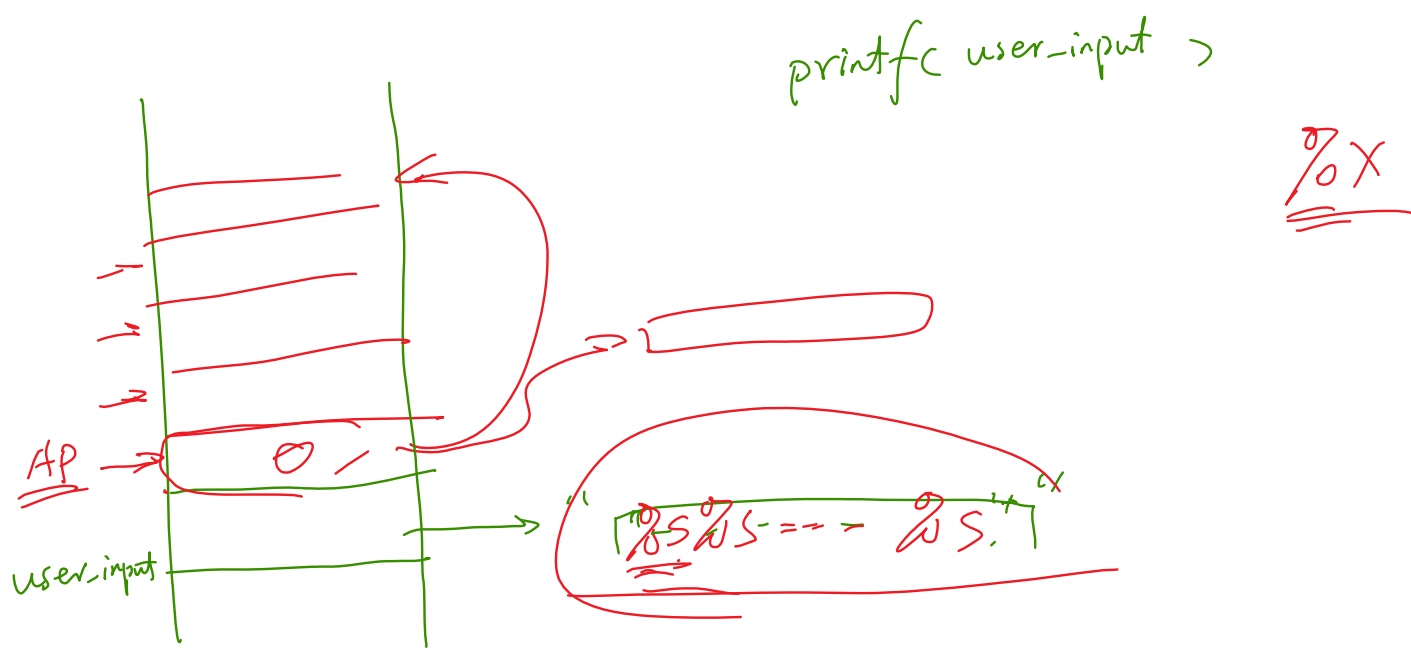
```
    return 0;
```

```
}
```

printf( " \_\_\_\_\_" )

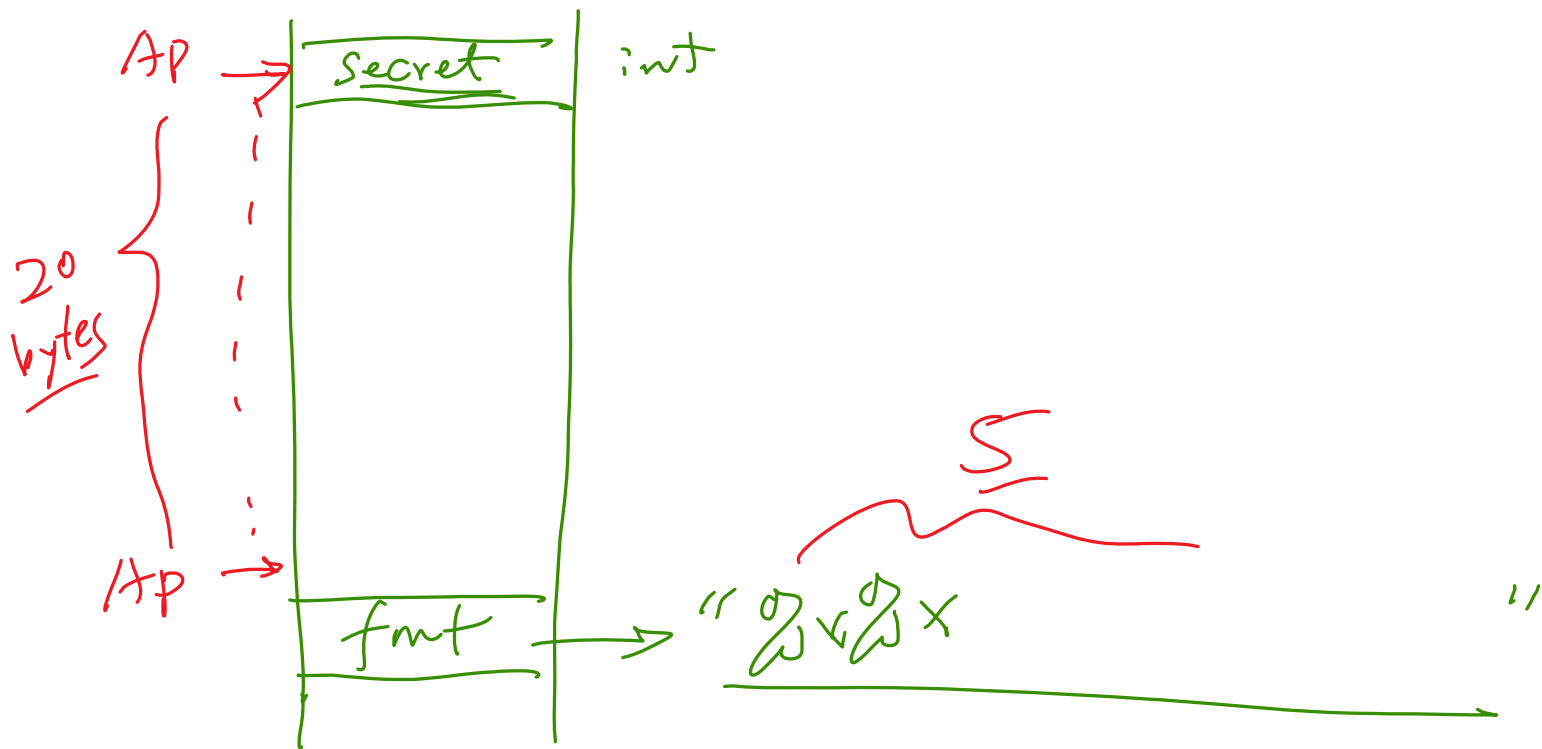


Crash the Program



# Print Out Secret Value

Question: How do you print out some secret valued stored on the stack?



## Print Out Secret Message at Specific Address

Question: How do you print out some secret valued stored at address 0xaabbccdd?

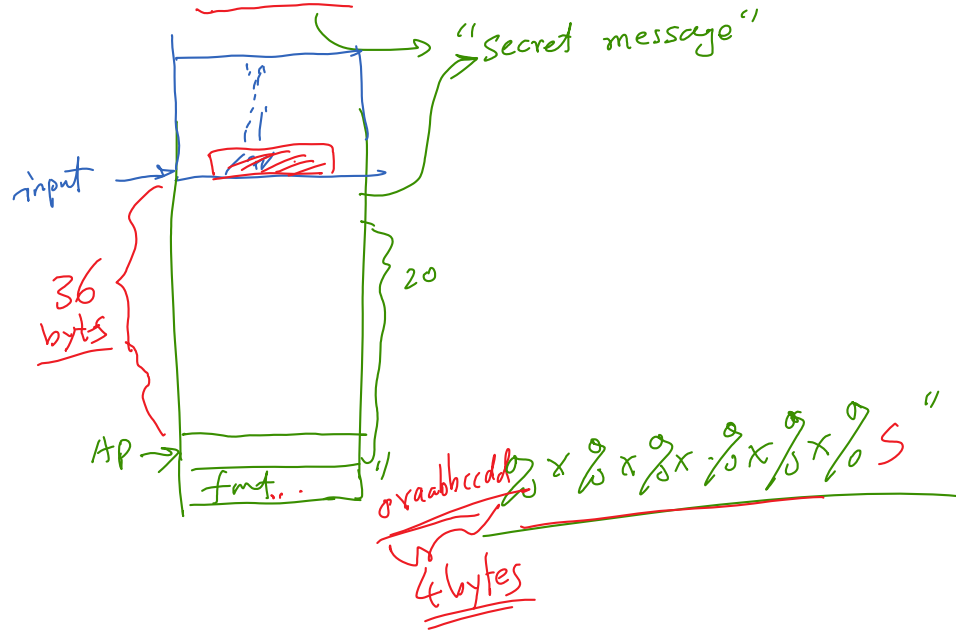
```
#include <stdio.h>

void fmtstr()
{
    char input[100];

    printf("Please enter a string: ");
    fgets(input, sizeof(input)-1, stdin);

    printf(input);
}

int main(int argc, char *argv[])
{
    fmtstr();
    return 0;
}
```



# Attack via Memory Modification



**SYRACUSE  
UNIVERSITY**  
**ENGINEERING  
& COMPUTER  
SCIENCE**

## Modify Memory at Specific Address

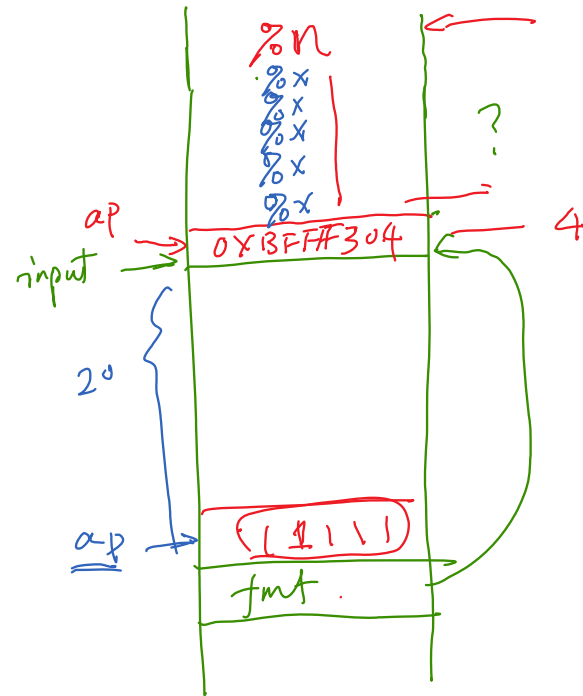
Question: How do you change the data stored at address `0xBFFF304` with some value (any value is fine)?

```
void main()
{
    int count=0;
    printf("Hello%n", &count);
}
```

address

5

0xBFFF304 %d%d%d%d%d%n



## Modify Memory with Specific Value (Approach 1)

Question: How do you modify the data stored at address `0xBFFF304` with value `0x66887799`?

### ❖ Precision and width modifiers

```
printf("%.5d", 10);
```

```
printf("%5d", 10);
```

```
printf("%d", 10);
```

10

00010  
~~~~~  
~~~~~10

32-bit int: at most 8 bytes for hex  
More than 8 characters for decimal

1000  
0x66887799

%x%x%x%x%x%x n  
⇓  
%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x n  
32 96 1000

## Modify Memory with Specific Value (Approach 2, much faster)

Question: How do you modify the data stored at address 0xBFFF304 with value 0x66887799?

*Shellcode*

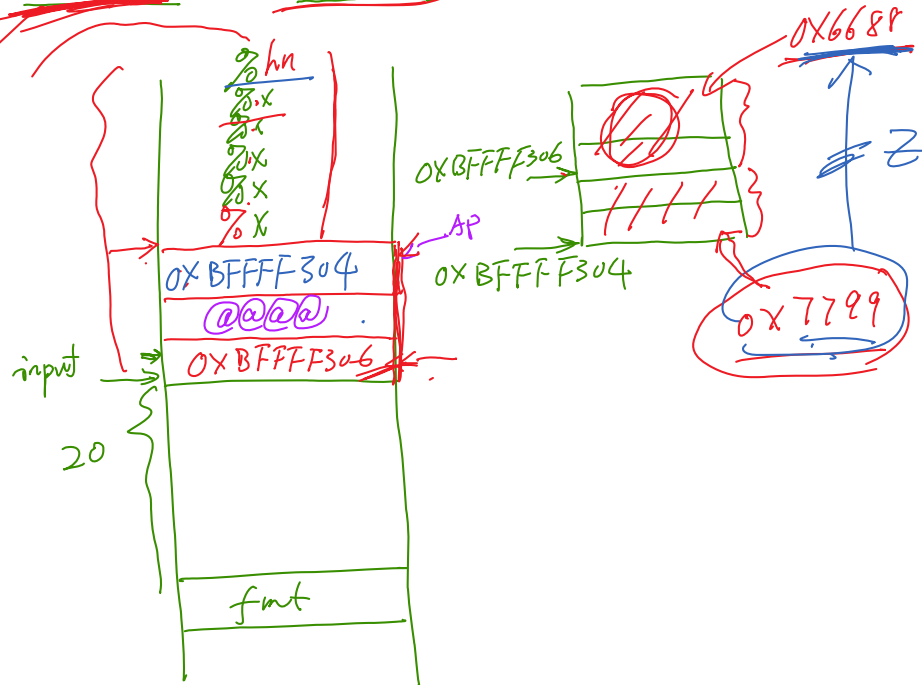
### ❖ Length modifiers for %n

```
#include <stdio.h>
void main()
{
    int a, b, c;
    a = b = c = 0x11223344;

    printf("12345%n\n", &a);
    printf("The value of a: 0x%x\n", a);
    printf("12345%hn\n", &b);
    printf("The value of b: 0x%x\n", b);
    printf("12345%hhn\n", &c);
    printf("The value of c: 0x%x\n", c);
}
```



*%...x*



# Code Injection



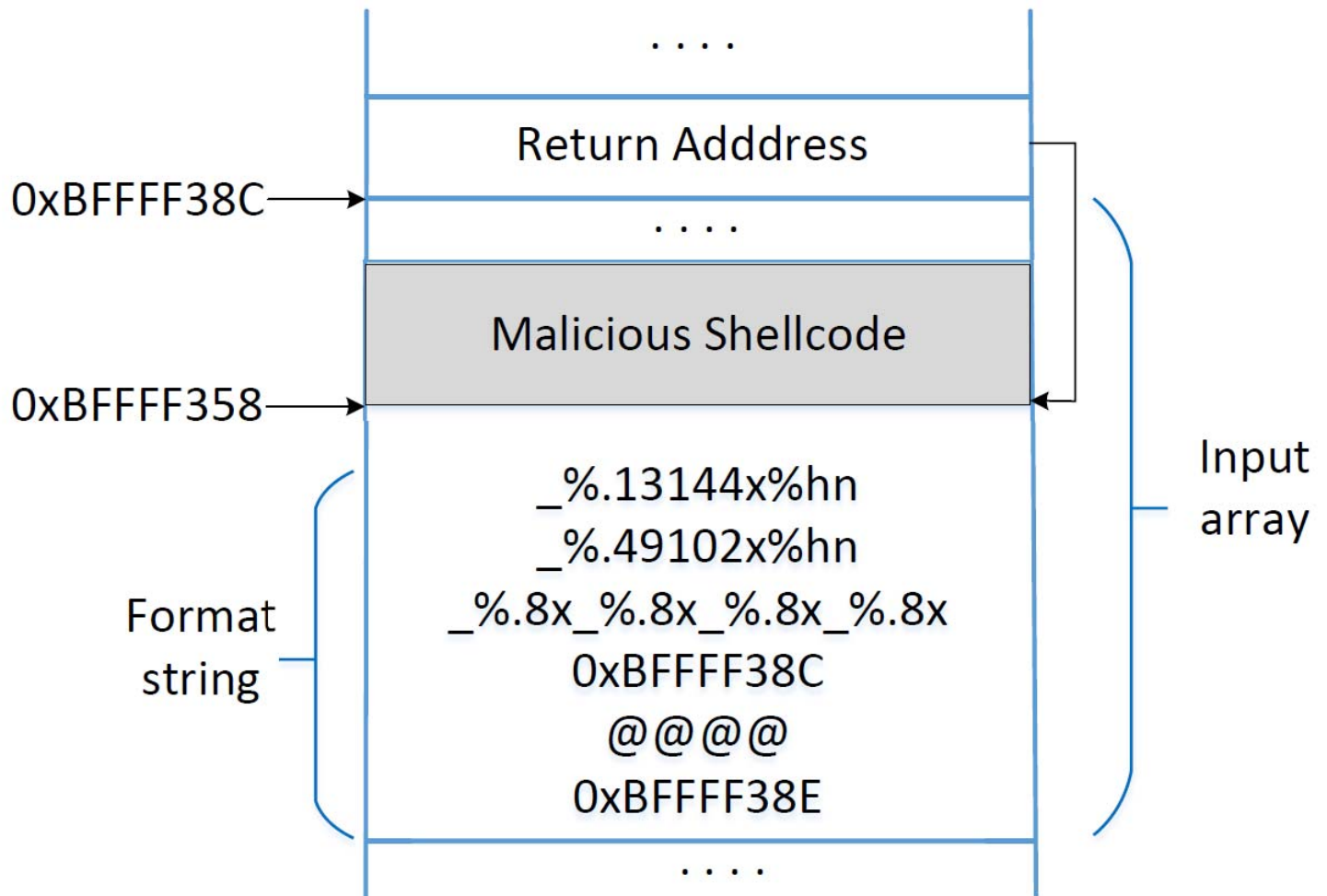
**SYRACUSE  
UNIVERSITY**  
**ENGINEERING  
& COMPUTER  
SCIENCE**



# Code Injection

Question: How to use format string vulnerability to jump to injected shellcode?

# Constructing the Format String



# Format-String Attack Demo



# Summary

- ❖ How format string works
- ❖ Format string vulnerability
- ❖ Exploiting the vulnerability
  - Crash program
  - Steal secret
  - Modify data
  - Code injection