



CS 426 Lab2

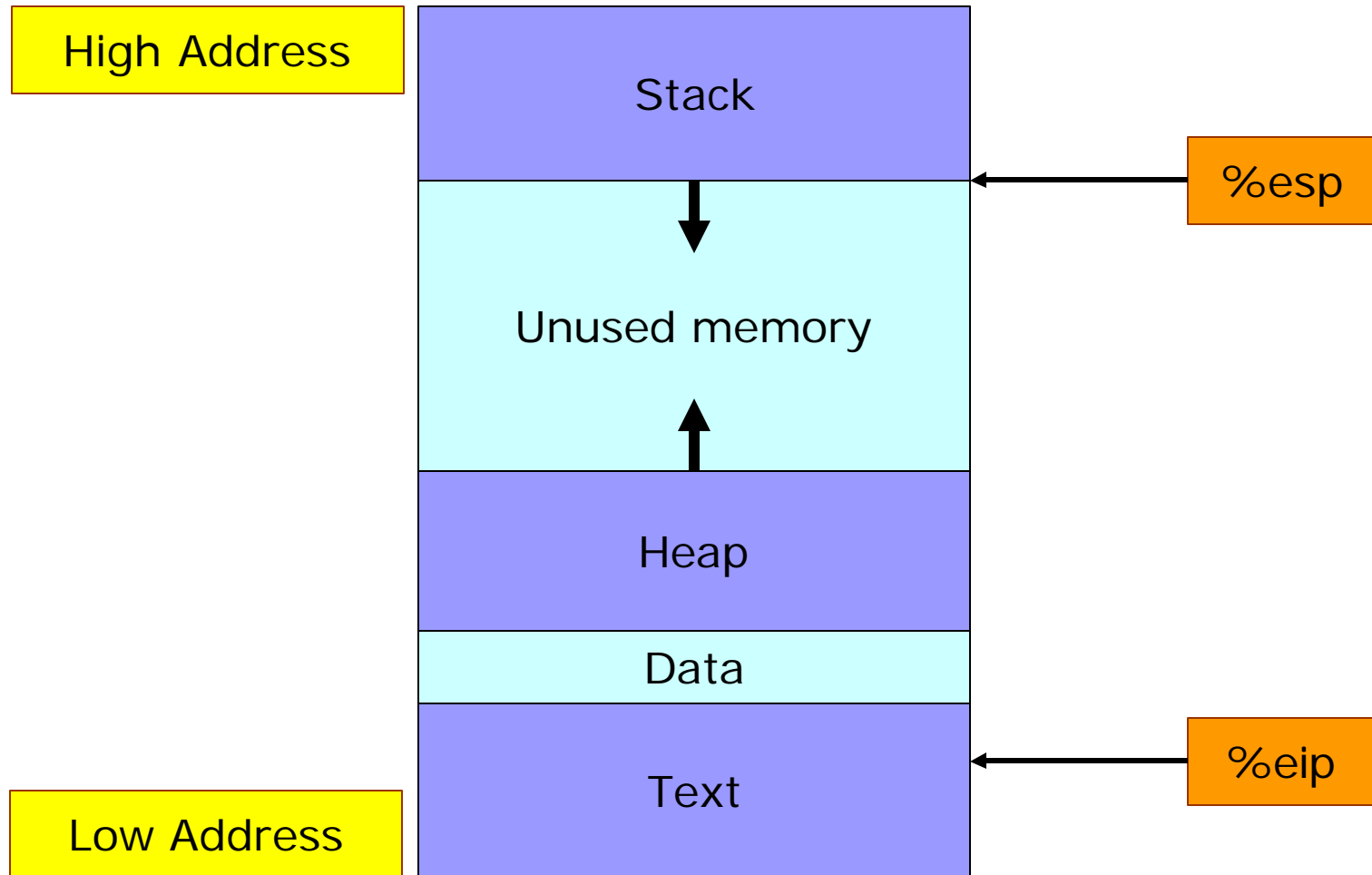
Ziqing Mao



Lab Overview

- You are asked to exploit software vulnerabilities
 - Stack overflow attacks
 - Return-to-libc attacks
 - Format string attacks
- 5 target programs
 - Write exploit programs
 - The frameworks for the exploits are provided
- The first step: clearly understand the function call mechanism and stack layout in C

Memory Layout Overview

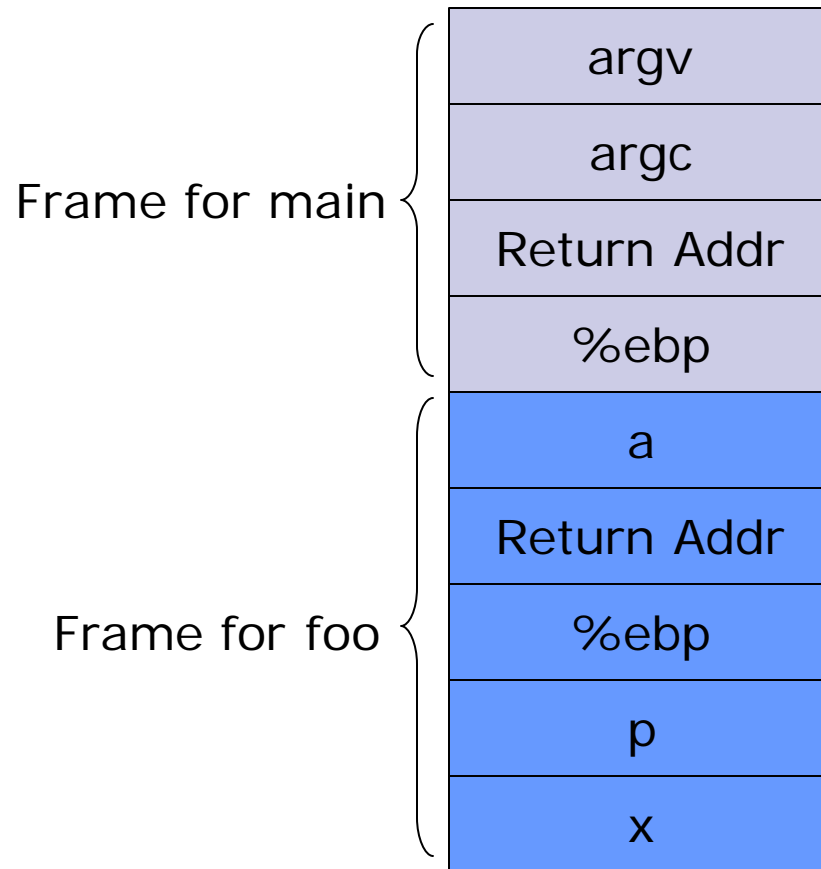


Stack Frame Layout

```
■ void foo(char* a)
{
    int p = 3;
    char x[4];

    strcpy(x, a);
}

int main(int argc, char** argv)
{
    foo("cat");
    foo("aaaabbbbccccdddd");
}
```

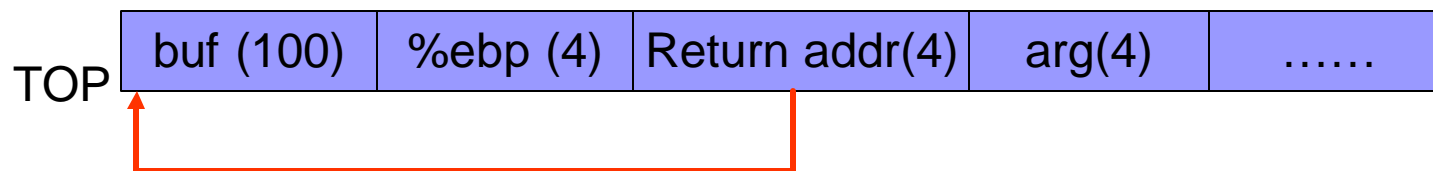


A Simple Example

- The vulnerable code

```
void foo(char* arg)
{
    char buf[100];
    strcpy(buf, arg);
}
```

- The stack layout before strcpy is revoked



- Exploit

- Put the shell code at the beginning of the buf
 - The shell code is provided in exploits/shellcode.h
- Overwrite the return address to point to buf
 - Need to find out the address of buf

Targets

- Target 1 (10 pts)
 - A program to check the correctness of the password
 - Goal: make the program accept your password
- Target 2 (20 pts)
 - A program to list files in a directory
 - Goal: to start a shell
- Target 3 (20 pts)
 - A program to check if a password is strong or weak
 - Goal: to start a shell

Return-to-libc Attack

- Basic exploit for stack overflow
 - Inject shell code into the stack
 - Overwrite the return address to point to the injected code
- Non-executable stack
 - The injected code in the stack cannot be executed
- Use the existing libc function to launch attacks
 - libc functions are loaded by the operating system
 - E.g., to launch a shell using libc: `system("/bin/sh")`

How to do that

■ How to do that

- Replace the return address with the address of the function `system()`
- Modify other appropriate portions of the stack to provide the argument `"/bin/sh"`

■ Need to find out

- The address of the function `system()`
- The address of the string `"/bin/sh"`

■ Read the reference

- http://www.infosecwriters.com/text_resources/pdf/return-to-libc.pdf



Target 4 (25 pts)

- A program to copy the user input into a local buffer
- Goal: to start a shell using return-to-libc attack

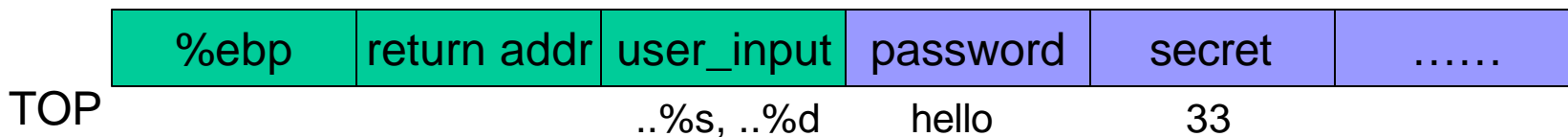
Format String Attacks

- An example

```
void func(char* user_input)
{
    int secret = 33;
    char password[] = "hello";
    printf(user_input);           //should be printf("%s", user_input)
}
```

- Exploit: func("password = %s, secret = %d")

- The stack layout when printf is revoked



- Output: password = hello, secret = 33

Format String Attacks (cont')

- The vulnerability can lead to
 - Crash the program
 - Read from an arbitrary memory address
 - Modify the value stored in an arbitrary memory address
- Target5
 - A program containing three secrets
 - Goal I: Print out the secret values (15 pts)
 - Goal II: Modify one of the secret values (10 pts)

Warming up

- C Programming

- Function call mechanism
- Stack layout
- printf

- Use `printf("%x", &var)` to print out the address of a variable
- Use GDB
- Use Google

Environment Setup

- The OS is running in a virtual machine with RedHat 6.2
- Login
 - Connect to the host machine
 - contact the TA for account info
 - ssh username@forest.cs.purdue.edu
 - Connect to the VM
 - telnet 172.16.232.128

Working in Teams

- You will work in teams of 2 people
- Find your partner!
- Send email to TA
 - zmao@cs.purdue.edu
 - Subject: cs426 team [lastname_lastname]
 - Body:
 - name1, email1
 - name2, email2
 - user, init_passwd

More Hints

- Exploits codes are short
- Several ways to exploit
- Start early
- Codes from others may **not** work
- Make a copy before you make changes
- Prepare to demo: make your exploits stable
- PSO: TA will show you how to debug, but will **not** help you to debug!
- Have fun 😊



Questions?