Introduction

ENEE 457

Computer Systems Security

Fall 2020

Dana Dachman-Soled

Includes material from Prof. Michelle Mazurek and Prof. Dave Levin

- Normally, we care about **correctness**
  - Does software achieve desired behavior?
- Security is a kind of correctness
  - Does software prevent **undesired** behavior?

The key difference is the adversary!

What are undesired behaviors?

- Reveals info that users want to hide
  - Corporate secrets, private data, PII
  - *Privacy/Confidentiality*
- Modifies info or functionality
  - Destroy records, change data mid-processing, install unwanted software
  - *Integrity*
- Deny access to data or service
  - Crash website, DoS,
  - *Fairness*

Why are attacks so common?

- Systems are complex, people are limited
- Many attacks exploit a ***vulnerability***
  - A ***software defect*** that can be manipulated to yield an undesired behavior
- Software defects come from:
  - Flaws in design
  - Bugs in implementation

Why are attacks so common?

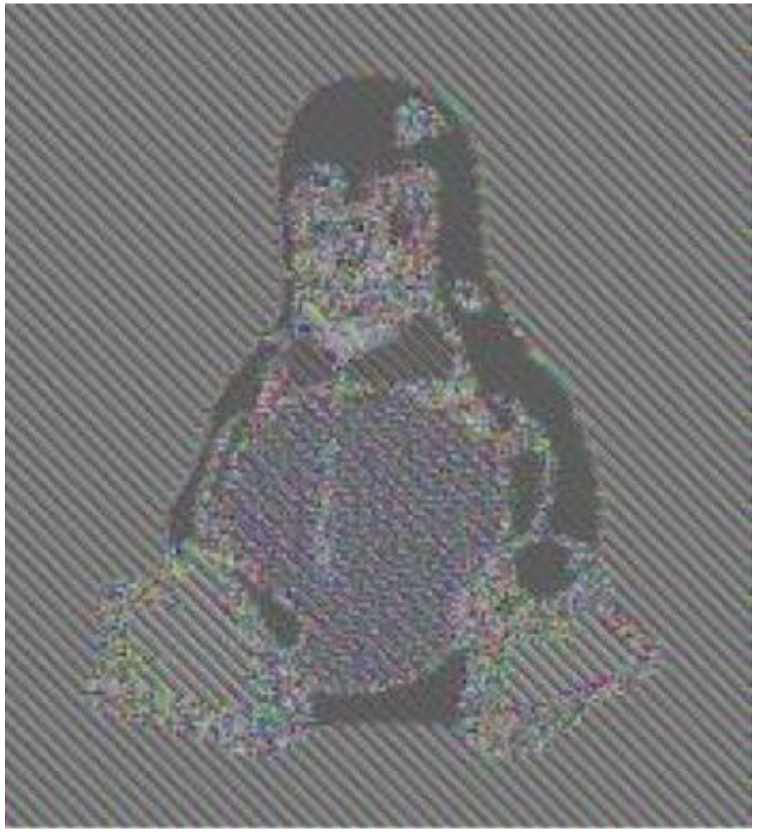
- Normal users avoid bugs
- Adversaries look for them to exploit

Why are attacks so common?

- Because it’s profitable
  - (Or attackers think it is)
- Because complex systems are only as strong as their weakest link

Steps toward more security….

- Eliminate bugs or design flaws, or make them harder to exploit
  - Think like an attacker!
- Deeply understand systems we build
- Be mindful of user-controlled inputs



Today’s agenda

- What is security
- C Refresher (Pointers, Memory Allocation)
- Case Study: Heartbleed Attack
- Course Survey

What’s in this course?

- Software and Web security
- Crypto
- Network security
- Special Topics (Bitcoin, Side-Channels, and more)

Software security

Memory safety

Malware

Web security

Static analysis

Design principles

What's in this course?

- Software and Web security
- Crypto
- Network security
- Special Topics (Bitcoin, Side-Channels, and more)

Applied crypto

- What it is (medium-high level)
- How to use it responsibly

Black-box approach

Designing protocols that use crypto

Authentication

Public Key/Symmetric Key

What’s in this course?

- Software and Web Security
- Crypto
- Network security
- Special Topics (Bitcoin, Side-Channels, and more)

Network security

- How to build secure networked systems

Attacks on TCP, DNS, Packet Sniffing

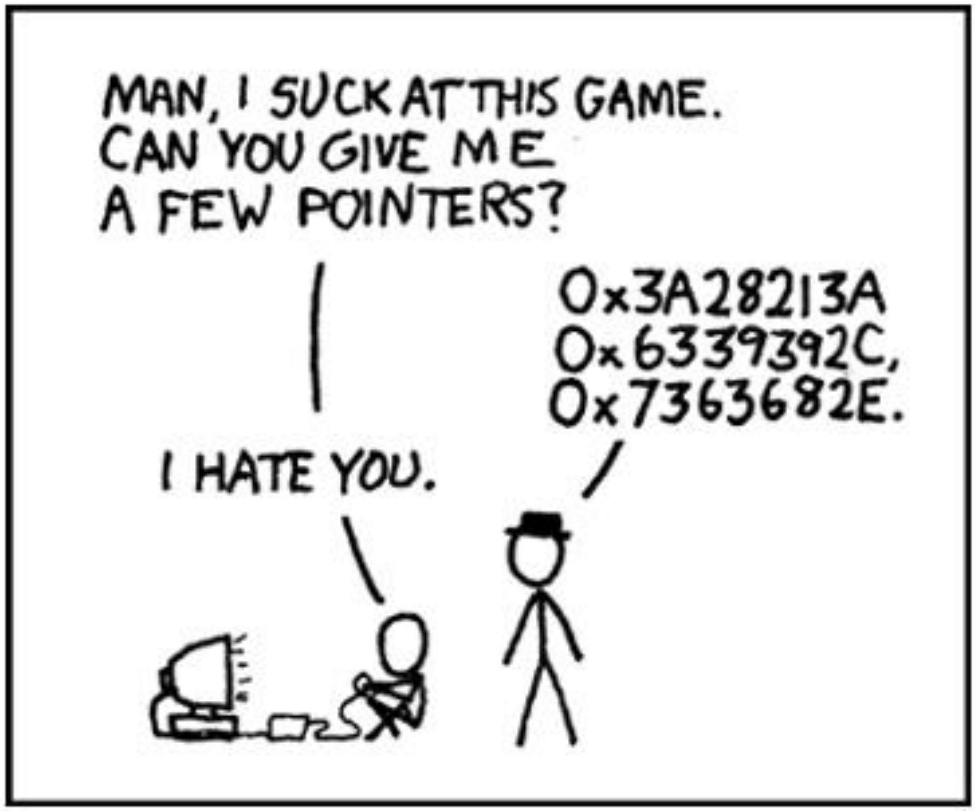
Anonymity

What's in this course?

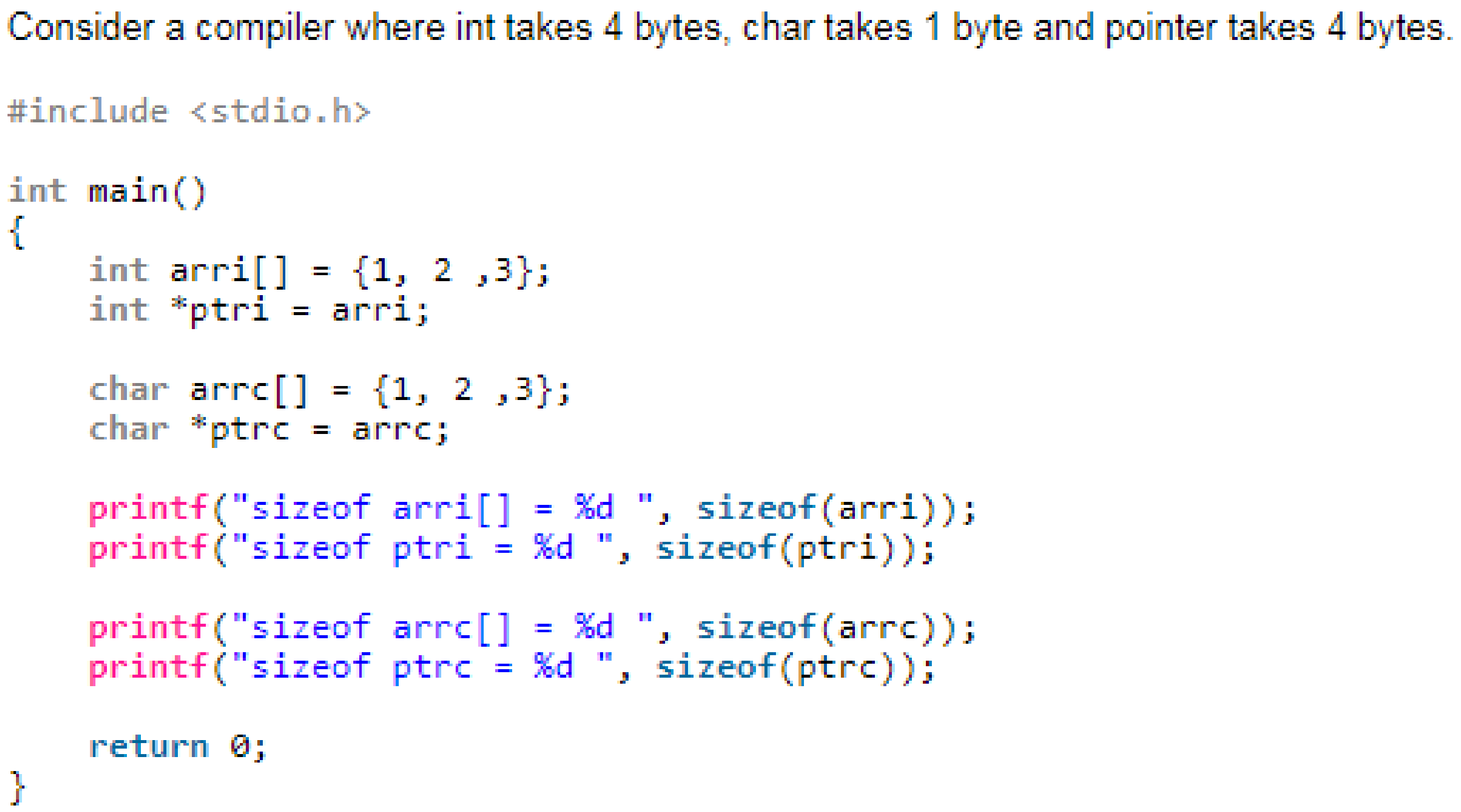
- Software and Web security
- Crypto
- Network security
- Special Topics (will include some or all of):
  - Bitcoin/Blockchain
  - Adversarial Machine Learning
  - Password Hashing
  - Side-Channel Attacks
  - Differential Privacy

First Topic: Buffer Overflows

Review:
Pointers and Memory Allocation in C


MAN, I SUCK AT THIS GAME. CAN YOU GIVE ME A FEW POINTERS?
0x3A28213A 0x6339392C, 0x7363682E.
I HATE YOU.

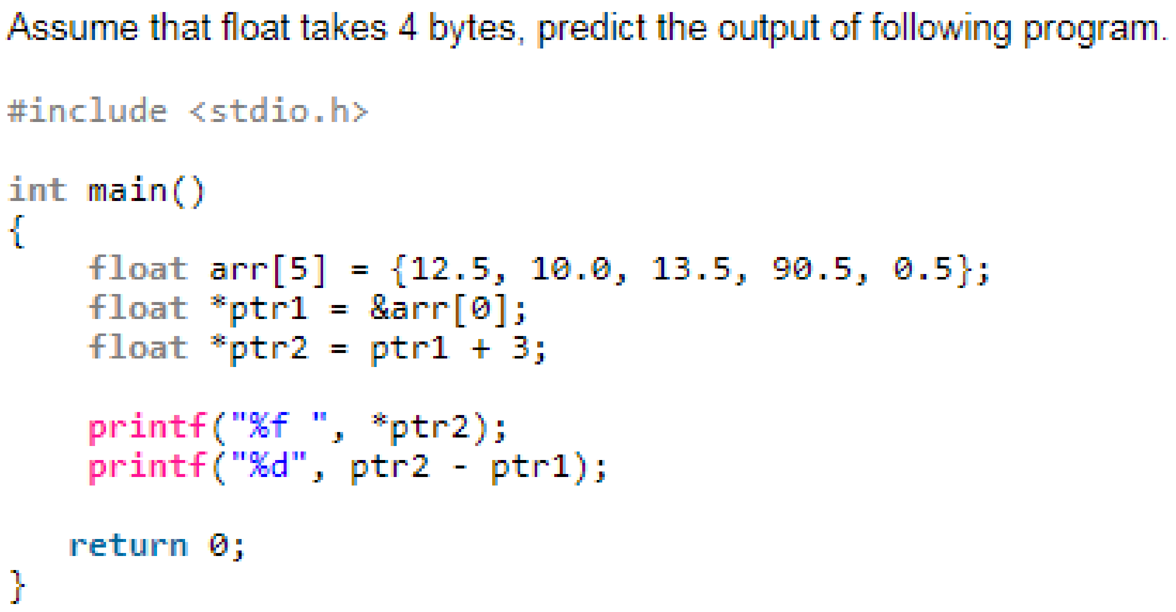
Review:
Pointers and Memory Allocation in C



Consider a compiler where int takes 4 bytes, char takes 1 byte and pointer takes 4 bytes.

```
#include <stdio.h>

int main()
{
    int arri[] = {1, 2 ,3};
    int *ptri = arri;

    char arrc[] = {1, 2 ,3};
    char *ptrc = arrc;

    printf("sizeof arri[] = %d ", sizeof(arri));
    printf("sizeof ptri = %d ", sizeof(ptri));

    printf("sizeof arrc[] = %d ", sizeof(arrc));
    printf("sizeof ptrc = %d ", sizeof(ptrc));

    return 0;
}
```

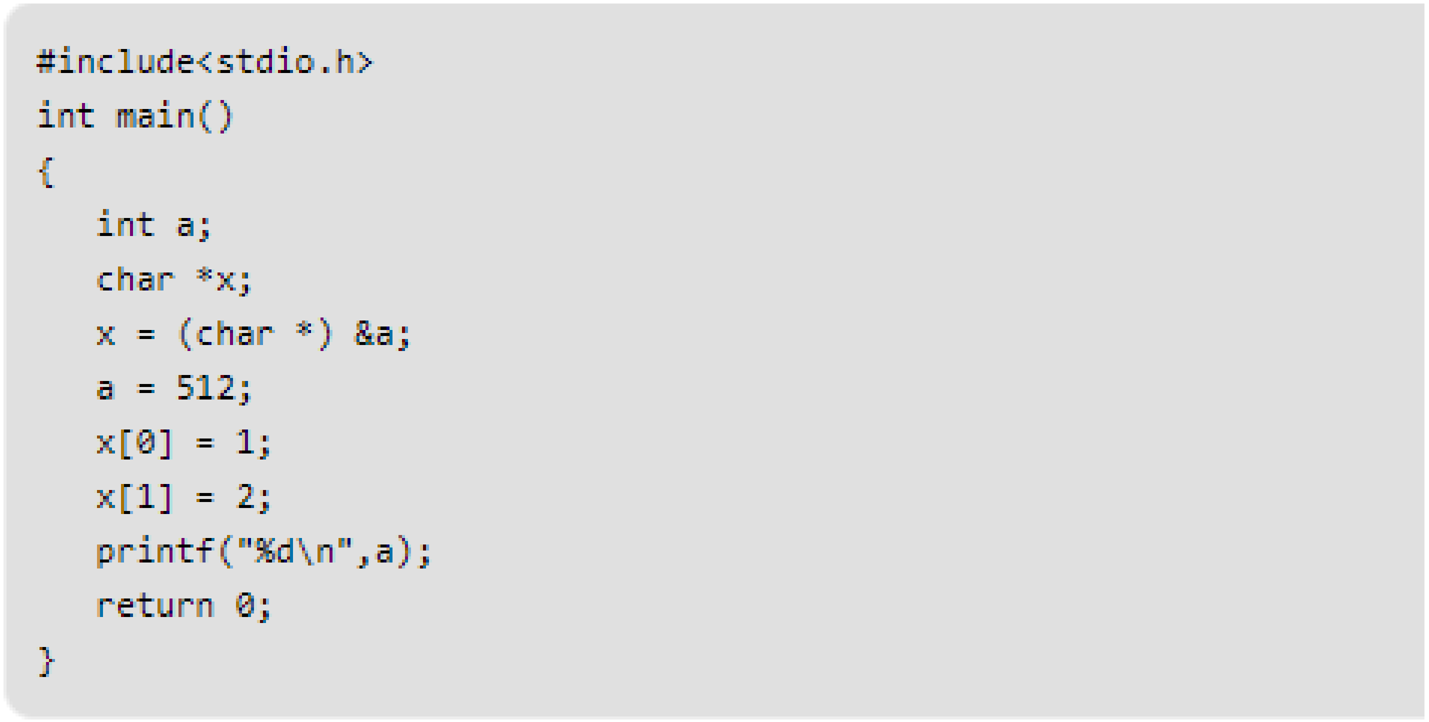
Review:
Pointers and Memory Allocation in C



Assume that float takes 4 bytes, predict the output of following program.

```c
#include <stdio.h>

int main()
{
    float arr[5] = {12.5, 10.0, 13.5, 90.5, 0.5};
    float *ptr1 = &arr[0];
    float *ptr2 = ptr1 + 3;

    printf("%f ", *ptr2);
    printf("%d", ptr2 - ptr1);

   return 0;
}
```

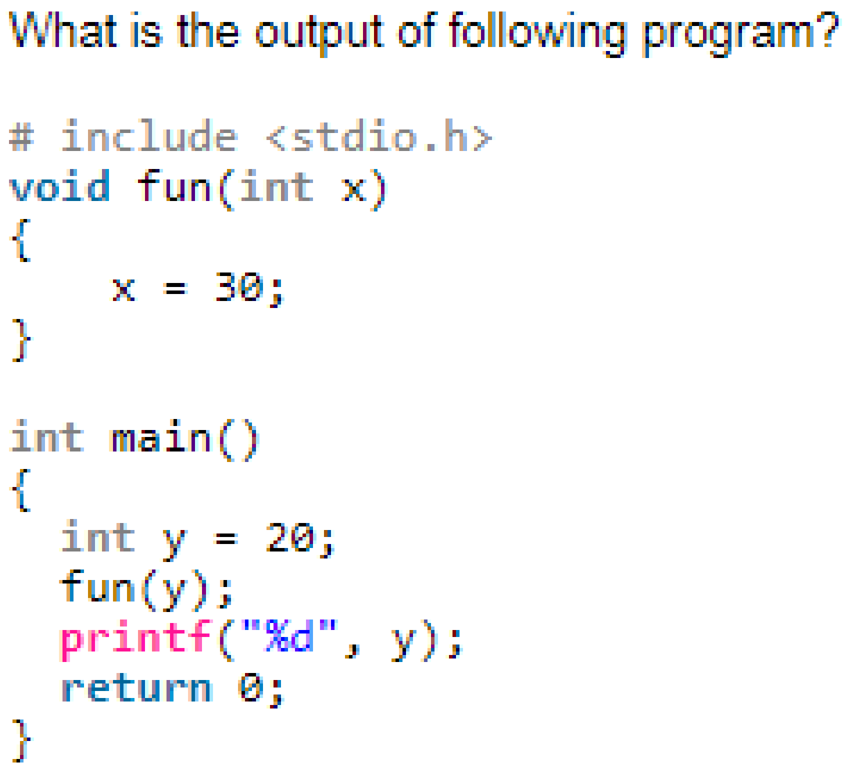
Review:
Pointers and Memory Allocation in C



```c
#include<stdio.h>
int main()
{
   int a;
   char *x;
   x = (char *) &a;
   a = 512;
   x[0] = 1;
   x[1] = 2;
   printf("%d\n",a);
   return 0;
}
```

What is the output? Assume *little-endian* processor.
The **least significant** byte (the "little end") of the data is placed at the byte with the lowest address. The rest of the data is placed in order in the next three bytes in memory.

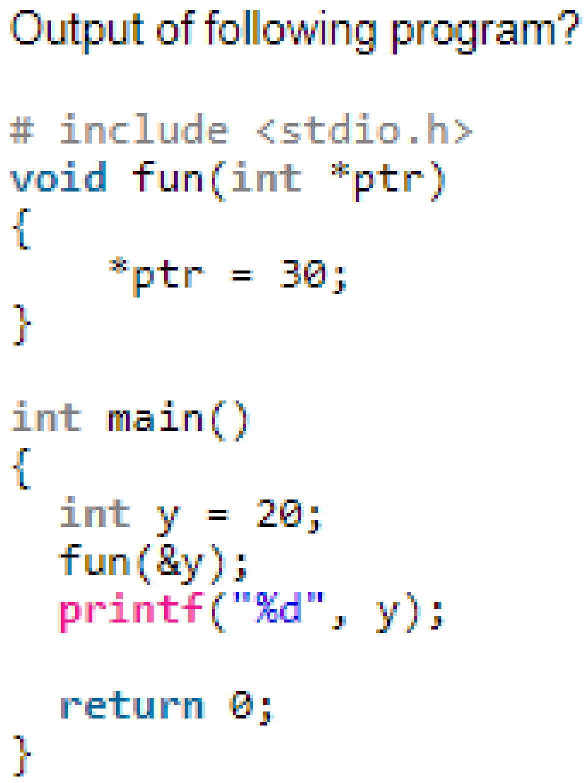
Review:
Pointers and Memory Allocation in C



What is the output of following program?

```
# include <stdio.h>
void fun(int x)
{
    x = 30;
}

int main()
{
  int y = 20;
  fun(y);
  printf("%d", y);
  return 0;
}
```

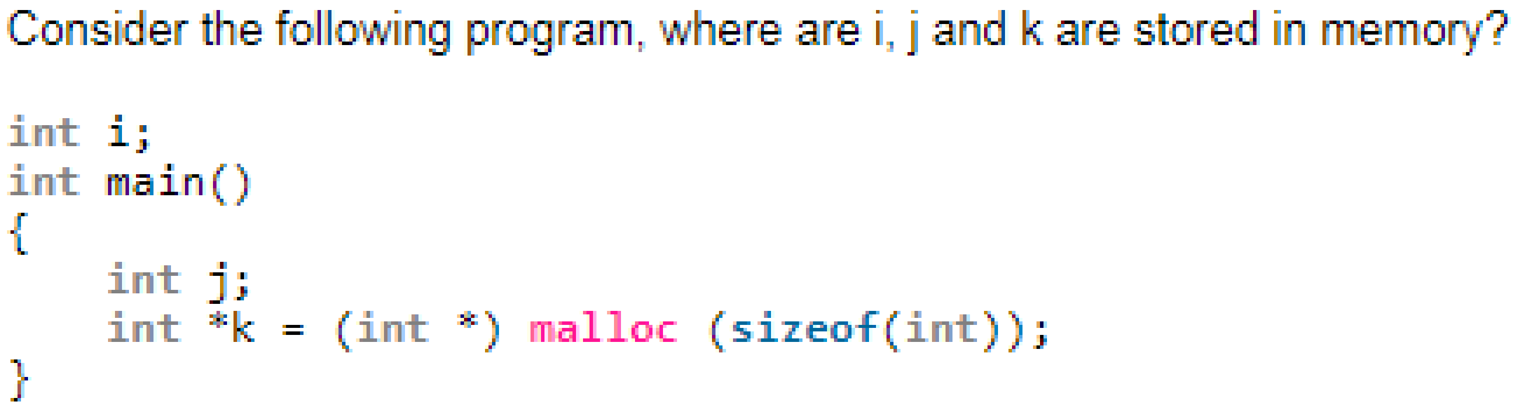
Review:
Pointers and Memory Allocation in C



Output of following program?

```
# include <stdio.h>
void fun(int *ptr)
{
    *ptr = 30;
}

int main()
{
  int y = 20;
  fun(&y);
  printf("%d", y);

  return 0;
}
```

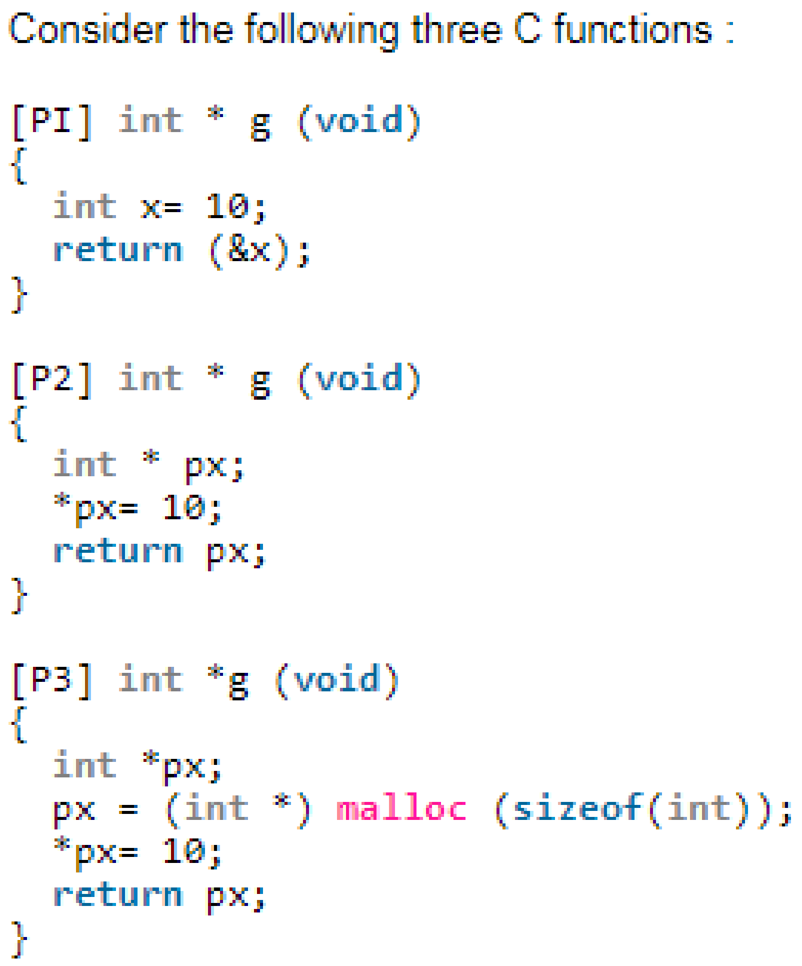
Review:
Pointers and Memory Allocation in C



Consider the following program, where are i, j and k are stored in memory?

```
int i;
int main()
{
    int j;
    int *k = (int *) malloc (sizeof(int));
}
```

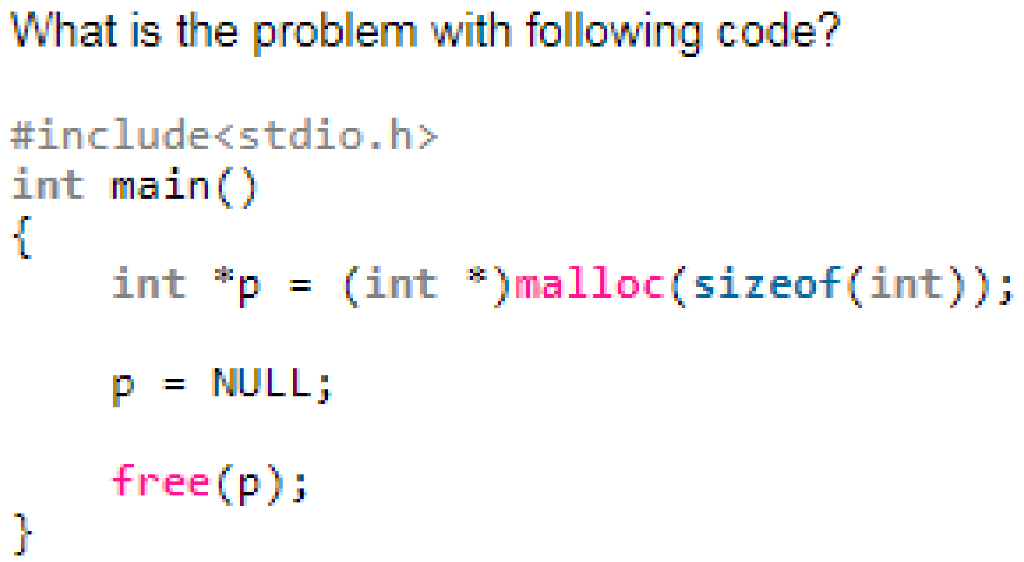
Review:
Pointers and Memory Allocation in C



Consider the following three C functions :

```
[PI] int * g (void)
{
  int x= 10;
  return (&x);
}

[P2] int * g (void)
{
  int * px;
  *px= 10;
  return px;
}

[P3] int *g (void)
{
  int *px;
  px = (int *) malloc (sizeof(int));
  *px= 10;
  return px;
}
```

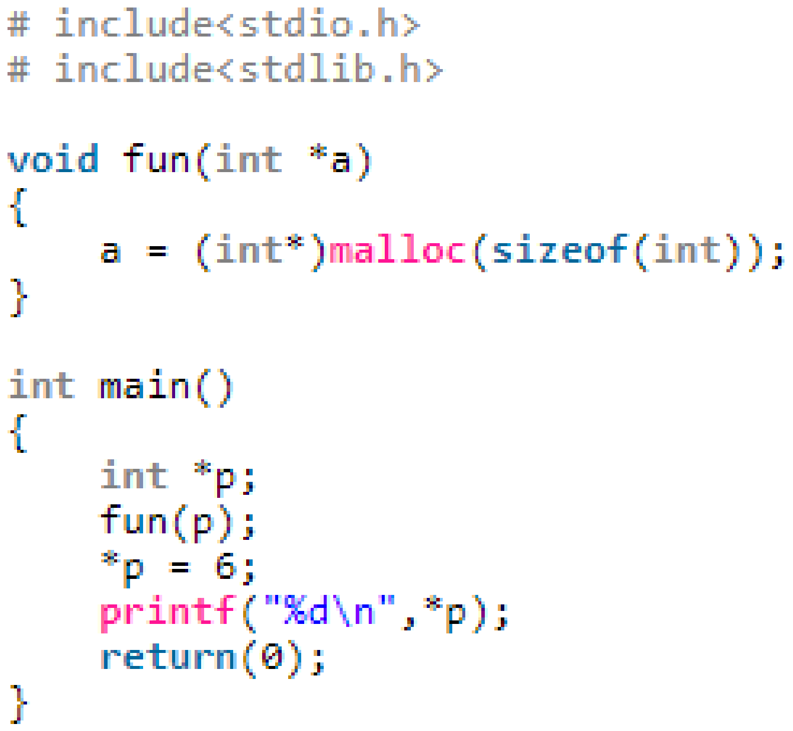
Review:
Pointers and Memory Allocation in C



What is the problem with following code?

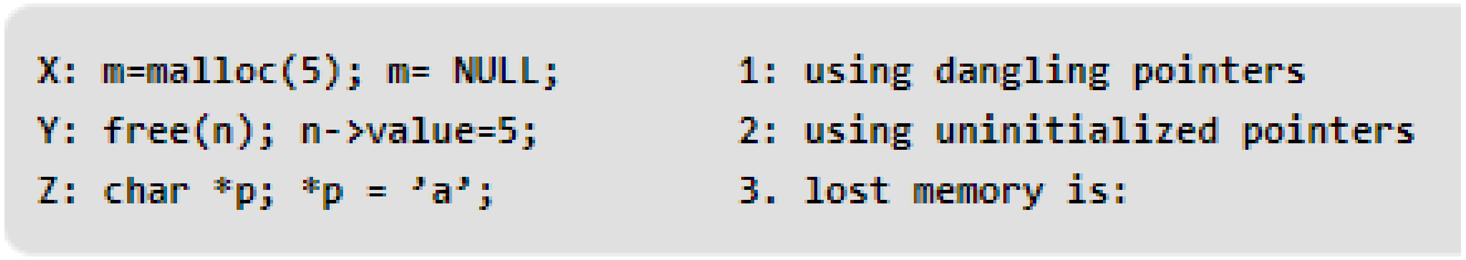
```c
#include<stdio.h>
int main()
{
    int *p = (int *)malloc(sizeof(int));

    p = NULL;

    free(p);
}
```

Review:
Pointers and Memory Allocation in C



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

void fun(int *a)
{
    a = (int*)malloc(sizeof(int));
}

int main()
{
    int *p;
    fun(p);
    *p = 6;
    printf("%d\n",*p);
    return(0);
}
```

Review:
Pointers and Memory Allocation in C



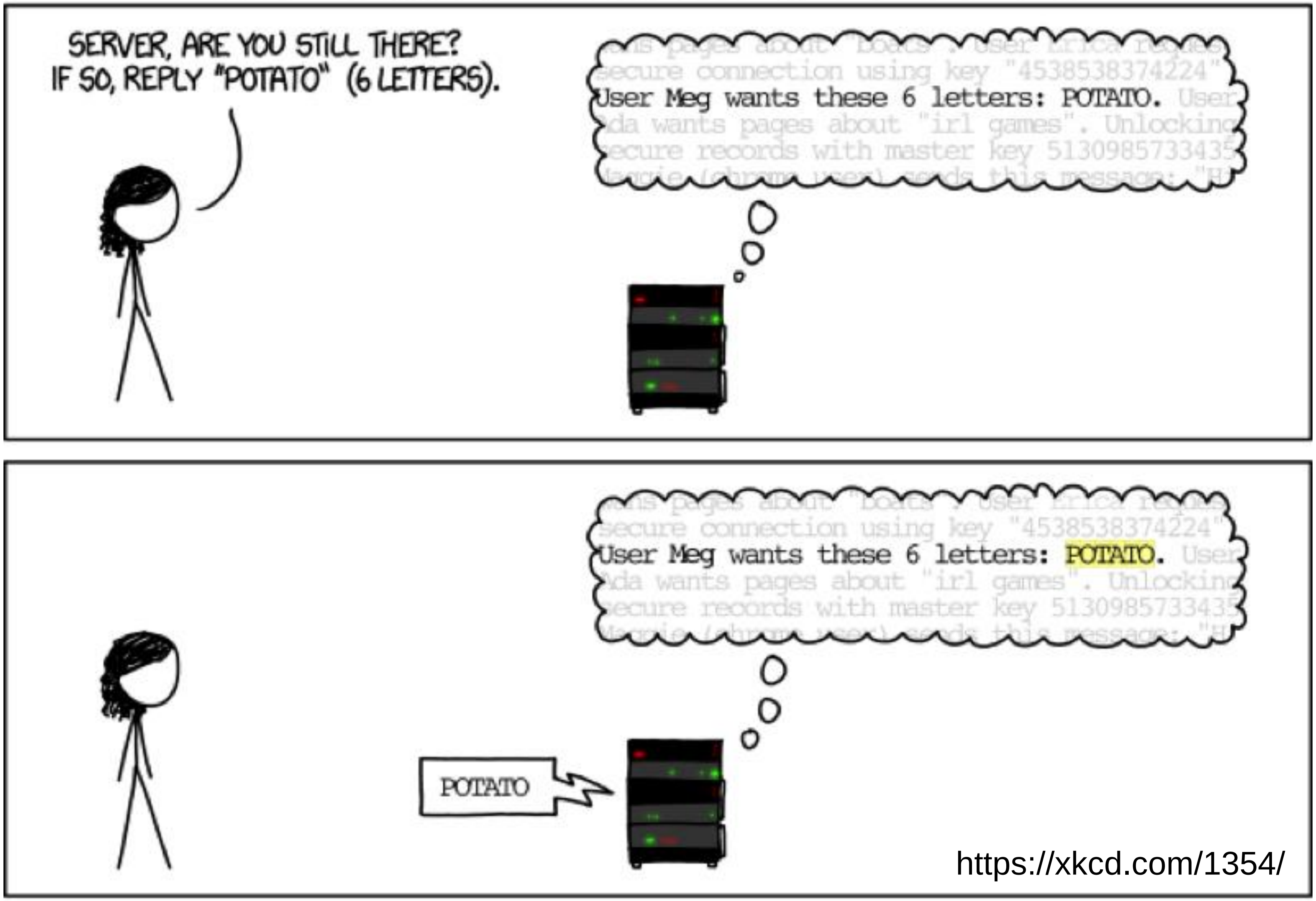
```
X: m=malloc(5); m= NULL;          1: using dangling pointers
Y: free(n); n->value=5;           2: using uninitialized pointers
Z: char *p; *p = 'a';             3. lost memory is:
```

Case study: Heartbleed

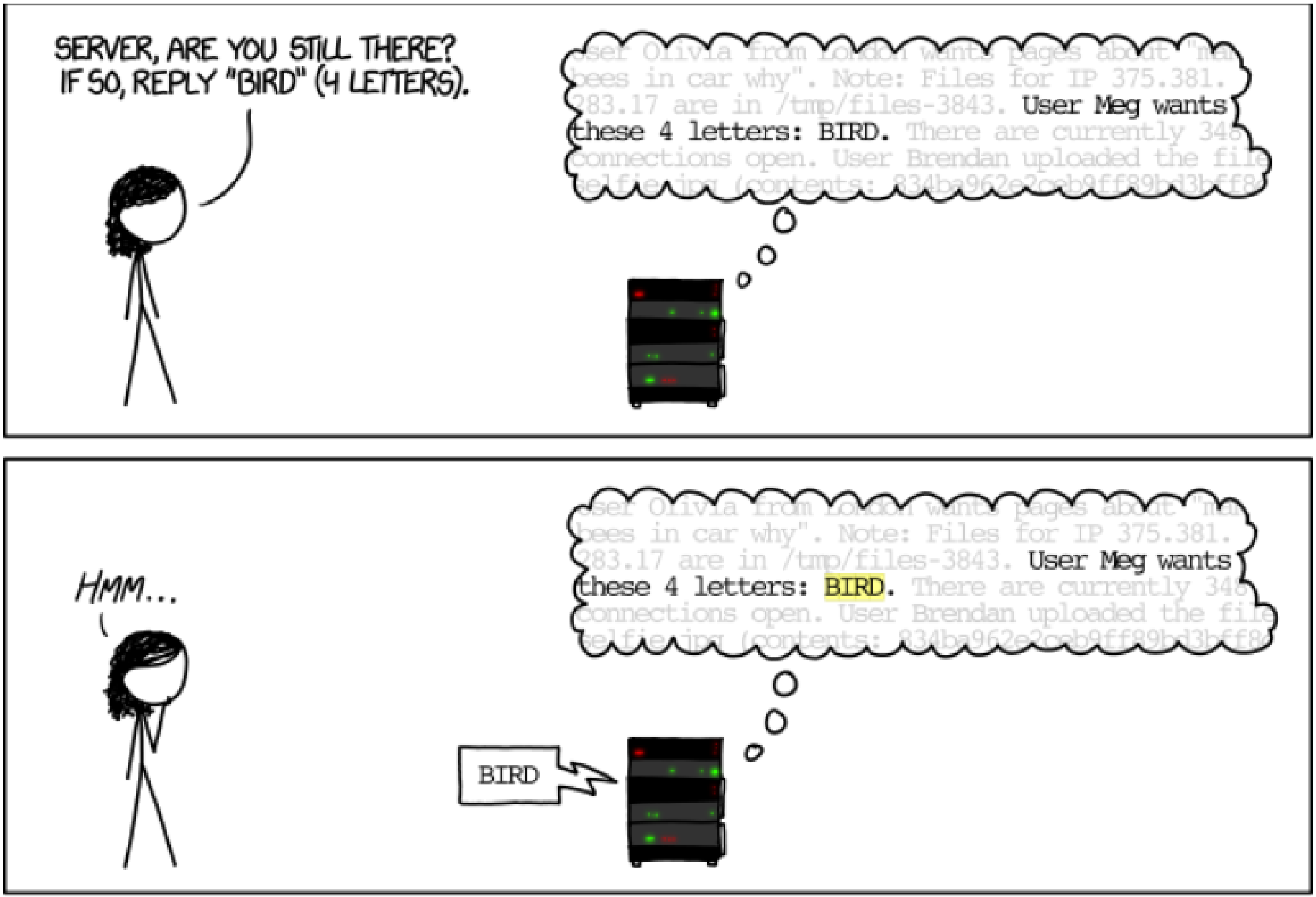
- SSL is the main protocol for secure (encrypted) online communication

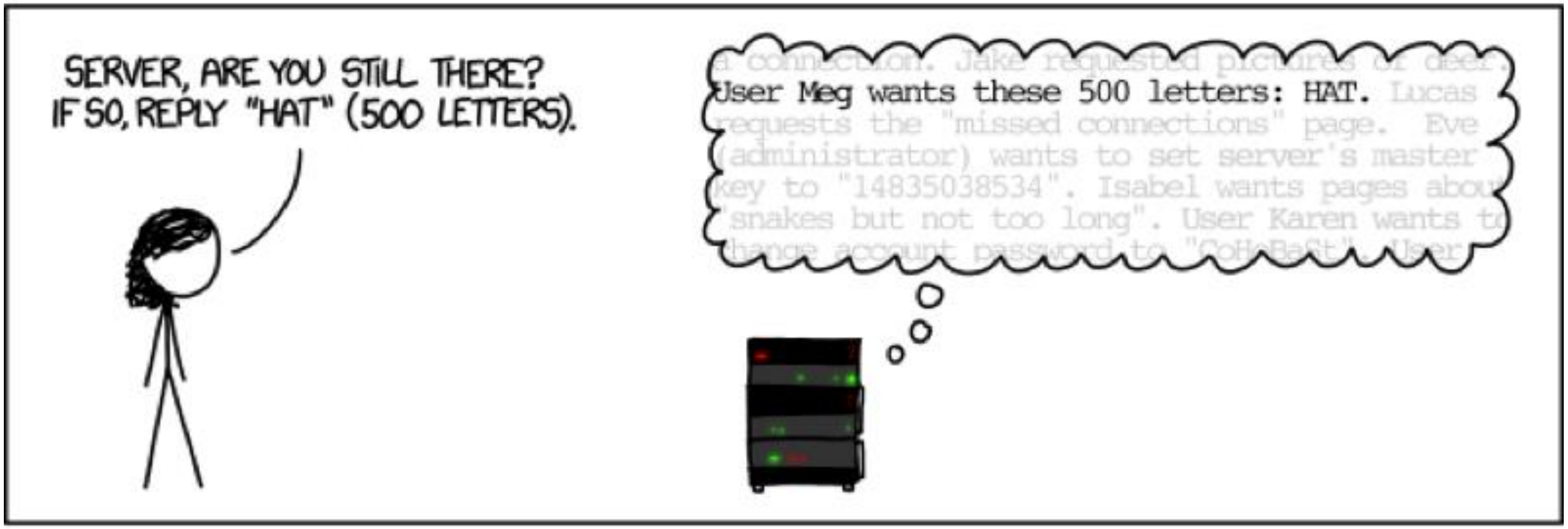
- Heartbleed was a vulnerability in the most popular SSL server

HOW THE HEARTBLEED BUG WORKS:

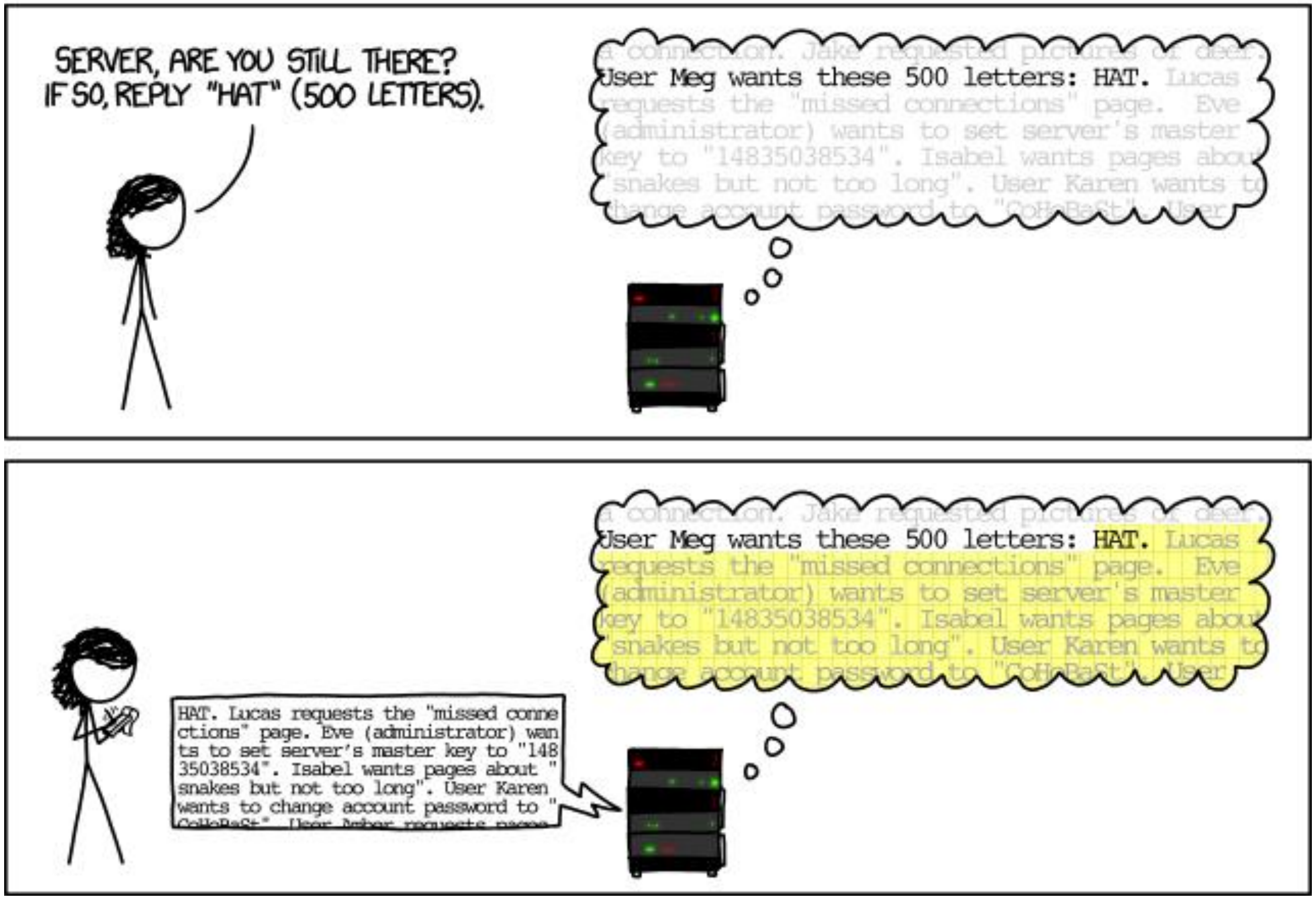

SERVER, ARE YOU STILL THERE? IF SO, REPLY "POTATO" (6 LETTERS).
secure connection using key "4538538374224"
User Meg wants these 6 letters: POTATO. User
Ada wants pages about "irl games". Unlocking
secure records with master key 5130985733435
POTATO
secure connection using key "4538538374224"
User Meg wants these 6 letters: POTATO. User
Ada wants pages about "irl games". Unlocking
secure records with master key 5130985733435

https://xkcd.com/1354/


SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).
User Olivia from London wants pages about "ma
bees in car why". Note: Files for IP 375.381.
283.17 are in /tmp/files-3843. User Meg wants
these 4 letters: BIRD. There are currently 346
connections open. User Brendan uploaded the file
selfie.jpg (contents: 834ba962e2ceb9ff89bd3bff8
HMM...
BIRD


SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
a connection. Jake requested pictures of deer. User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User

Heartbleed: A Closer Look at Buffer Read Overflow


SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
a connection. Jake requested pictures of deer. User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User
HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User Amber requests pages

Case study: Heartbleed

- SSL is the main protocol for secure (encrypted) online communication
- Malformed packet allows you to see server memory
  - Passwords, keys, emails, visitor logs …..
- Fix: Don't let the user tell you how much data to send back!
  - This is a *design* flaw

Heartbleed: A Closer Look at Buffer Read Overflow

- Read Overflow: A bug that permits reading past the end of a buffer.

Read integer

Read message

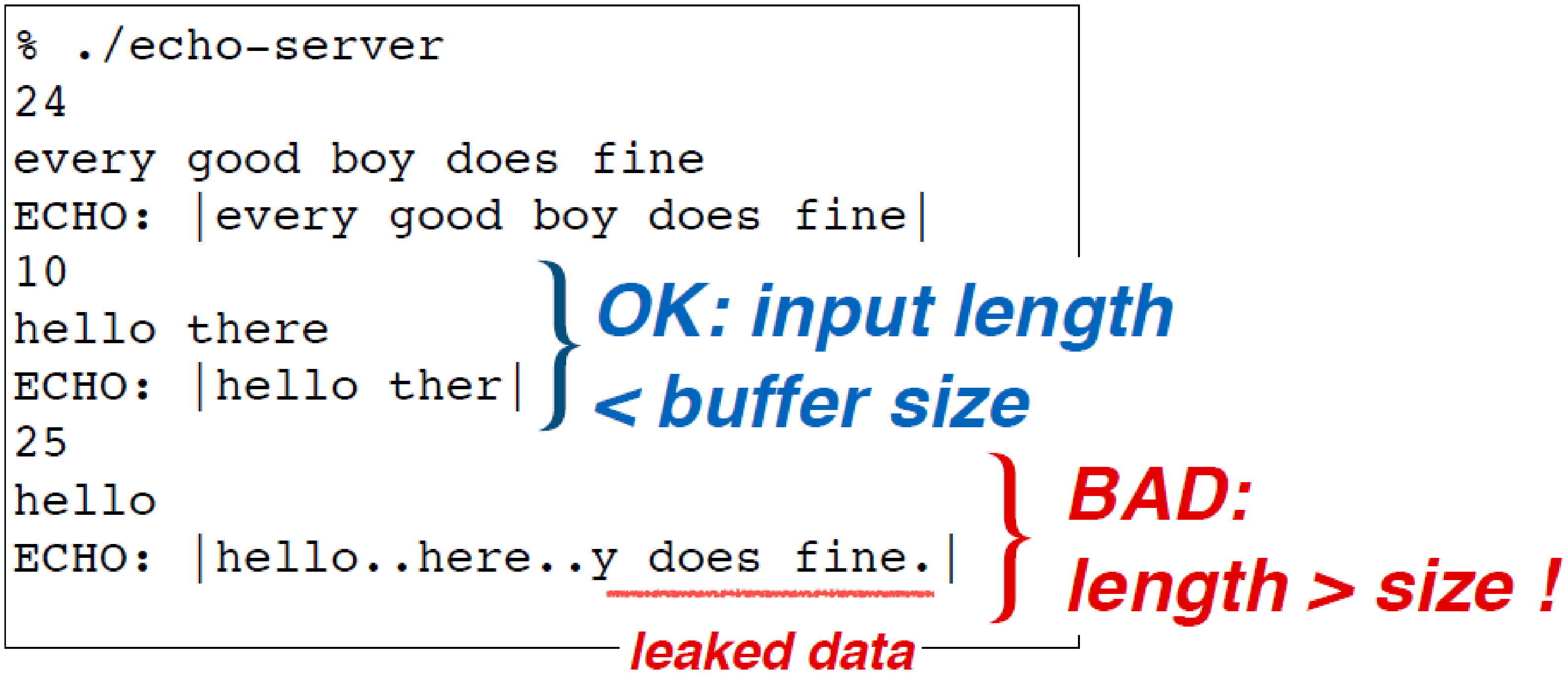
Echo back (partial) message

```
int main() {
 char buf[100], *p;

 while (1) {
  p = fgets(buf,sizeof(buf),stdin);
  len = atoi(p);
  p = fgets(buf,sizeof(buf),stdin);
  for (i=0; i<len; i++) {
   if (!iscntrl(buf[i]))
    putchar(buf[i]);
   else putchar('.');
  }
  printf("\n");
 }
 ...
```

len may exceed actual message length!

Heartbleed: A Closer Look at Buffer Read Overflow

- Sample Output:



```
% ./echo-server
24
every good boy does fine
ECHO: |every good boy does fine|
10
hello there
ECHO: |hello ther|
25
hello
ECHO: |hello..here..y does fine.|
```