Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Buffer Overflow Vulnerability in C.

Aarizkhan Shaikh (**75**) | Vaidat Patel (**67**) | Yash Mayekar (**59**)

Div – **1** | FY B.Tech

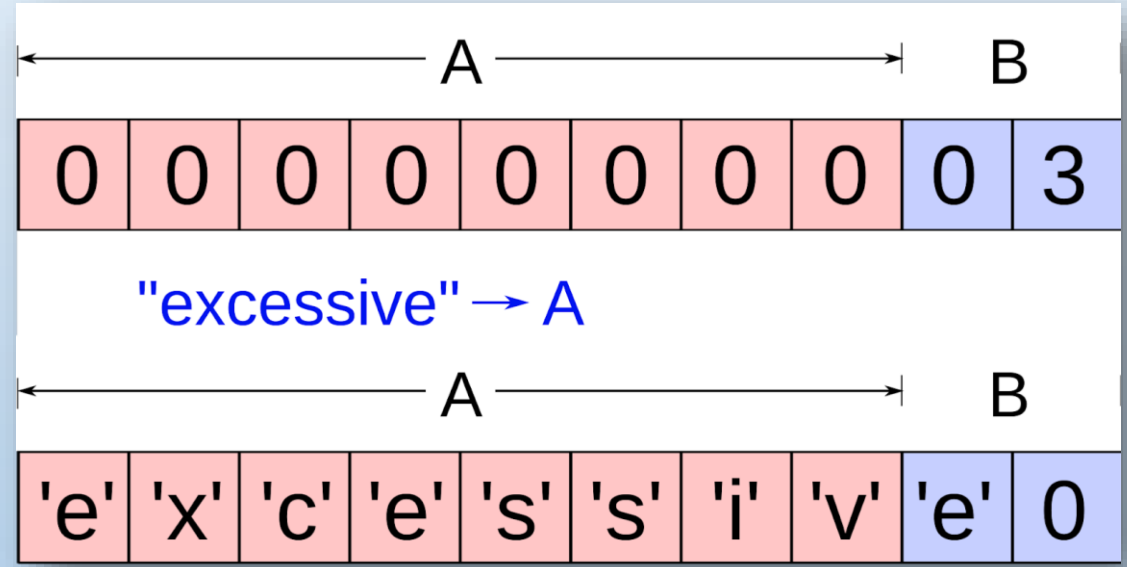
Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Agenda

- What is Buffer Overflow Vulnerability?
- What is gets()?
- Hands-on demonstration of Buffer Overflow
  - (Explanation + Exploitation)
- Alternatives, modifications of gets().
- Why secure code is important?
- Cybersecurity best practices for writing C programs.
- References.

What is BUFFER OVERFLOW Vulnerability?

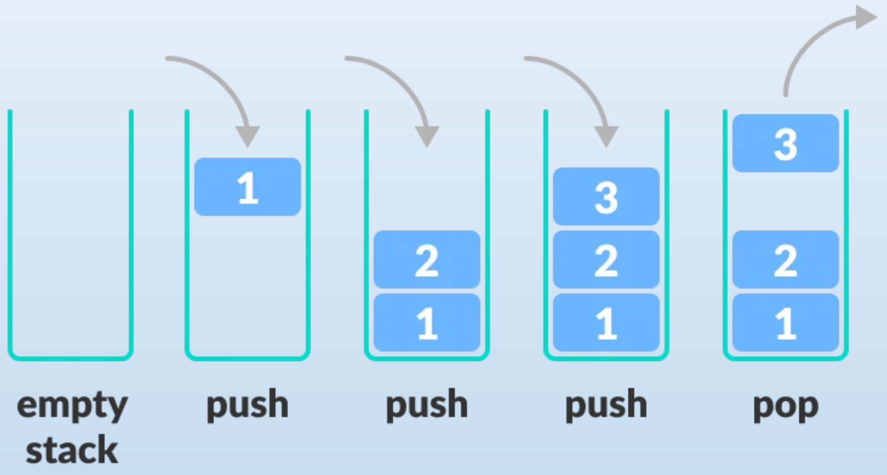
- In a sense, **Buffer is like Memory**!
- **Buffer Overflow is exceeding the memory's capacity** in the Buffer/Memory.
- A buffer overflow, also known as a buffer overrun, is a programming and **security Bug in which a program overwrites nearby memory locations** while writing data to a buffer/memory.



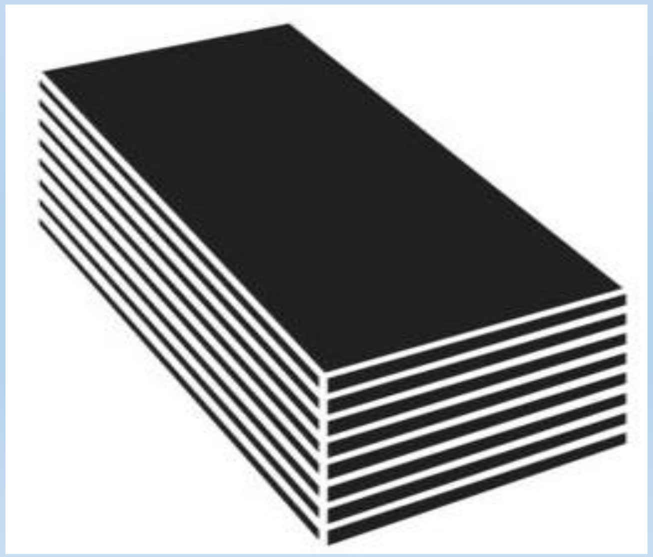
Buffer/Memory

Contd..

- The are two major types of Buffer Overflow Attacks -
- *Stack-based:*
- Buffer/memory overflow condition that occurs when the buffer/memory that is being overwritten is **allocated on the stack.**
- This generally includes **locally declared variables or (rarely) user input parameter to the function.**
- *Heap-based:*
- Buffer/memory overflow in which the buffer/memory that can be overwritten is **allocated in the heap portion of memory.**
- This generally means that the memory was **allocated using a function like malloc() or calloc().**



Stack memory

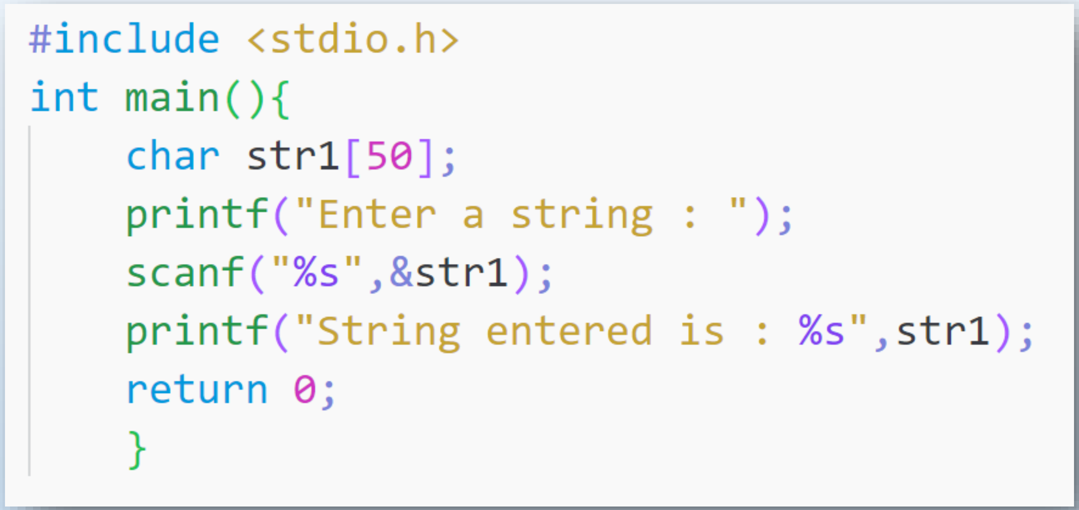


Heap memory

What is **gets()**?

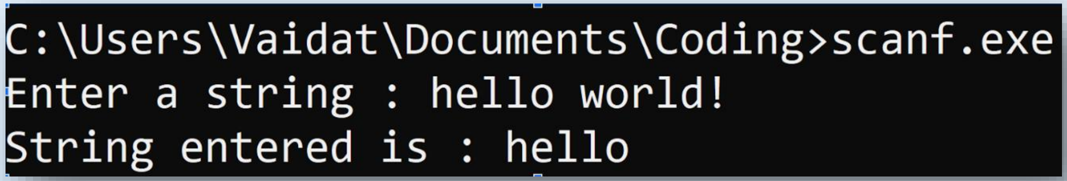
- Standard library function - <stdio.h>
- Syntax: `char *gets(char *str)`
- Takes only 1 input argument
- String input - allows space-separated strings
- Similar to **scanf(), fscanf()**
- Only problem - Buffer overflow / Array bound check missing

Practical difference b/w **gets()** and **scanf()**.



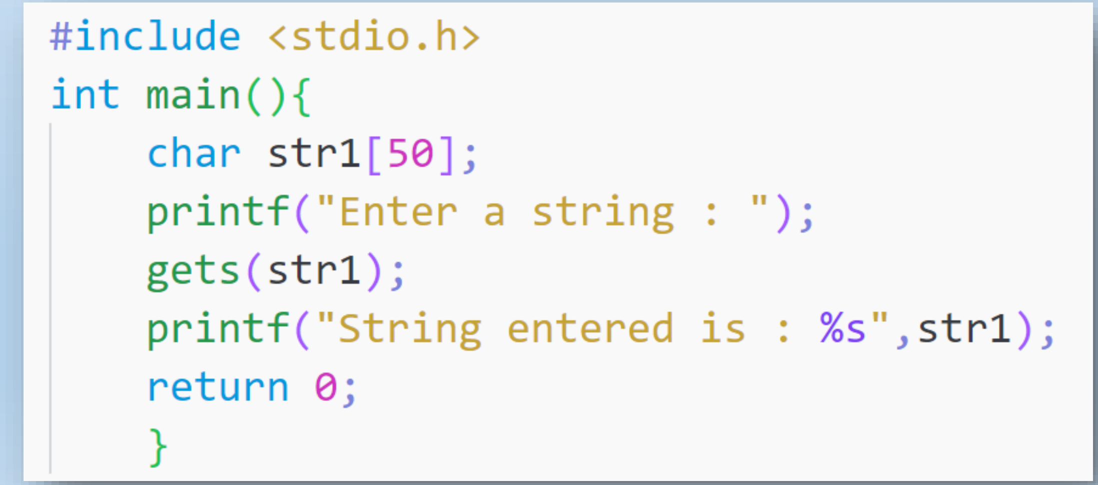
```c
#include <stdio.h>
int main(){
    char str1[50];
    printf("Enter a string : ");
    scanf("%s",&str1);
    printf("String entered is : %s",str1);
    return 0;
    }
```

Case 1: Using scanf()



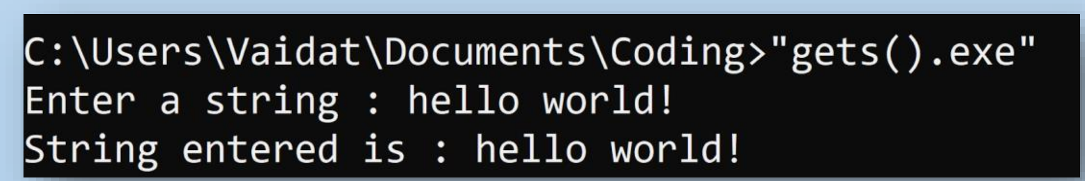
```
C:\Users\Vaidat\Documents\Coding>scanf.exe
Enter a string : hello world!
String entered is : hello
```

Output of scanf() code



```c
#include <stdio.h>
int main(){
    char str1[50];
    printf("Enter a string : ");
    gets(str1);
    printf("String entered is : %s",str1);
    return 0;
    }
```

Case 2: Using gets()



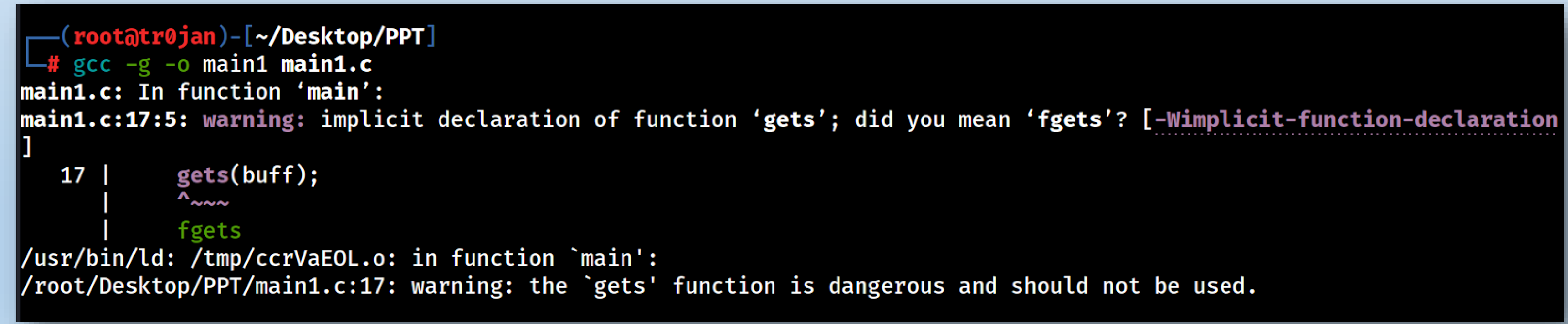
```
C:\Users\Vaidat\Documents\Coding>"gets().exe"
Enter a string : hello world!
String entered is : hello world!
```

Output of gets() code

Hands-on Demo.

Alternatives, modifications of **gets()**

- Since **gets()** doesn't know how big the buffer is the string continuously reading until it finds a newline or **encounters EOF**(**end of file**).
- The compiler throws a warning whenever we use the function gets(). This shows how vulnerable it is.
- Alts - **scanf(), fgets() & gets_s(destination, buffer_size)**



```
┌──(root@tr0jan)-[~/Desktop/PPT]
└─# gcc -g -o main1 main1.c
main1.c: In function ‘main’:
main1.c:17:5: warning: implicit declaration of function ‘gets’; did you mean ‘fgets’? [-Wimplicit-function-declaration
]
   17 |     gets(buff);
      |     ^~~~
      |     fgets
/usr/bin/ld: /tmp/ccrVaEOL.o: in function `main':
/root/Desktop/PPT/main1.c:17: warning: the `gets' function is dangerous and should not be used.
```

Terminal

Why secure code is important?

- Data breach - **Buffer overflows, XSS payloads,...**
- Code is everything.
- Security benchmarks.
- Scrutinize user inputs.
- Privileges - default-deny approach.

PS2 Independence Exploit.

- Beta version of PS2.
- Buffer overflow in BIOS of PS1 compatibility.
- Use of **homebrew software.**
- Other hardware exploitation (PS2 hard drive - HD loader).

Cybersecurity best practices for writing C programs.

- **Validate All User Inputs.**
  - Consider user input wild & random.
  - Block XSS, payload injection, etc.. by using parameter filters.
- **Complying your C code with ISO/IEC TS 17961:2013.**
  - **E.g.** Free the allocated dynamic memory after use.
- **Avoiding code with known security vulnerabilities.**
  - **E.g.** Comparing your code with non-compliant & vulnerable code
- **Don't ignore compiler warnings**
  - fgets() example.
- **Check Return values**
  - What if malloc()/calloc() returns NULL? The code will crash.
- **Code Readability**
  - Everyone should be able to read & understand your code.
  - Good practice.

Security is inversely proportional to User Experience!

References

- Warning: the gets function is Dangerous and should not be Used (knowprogram.com)
- https://stackoverflow.com/questions/1694036/why-is-the-gets-function-so-dangerous-that-it-should-not-be-used
- https://en.wikipedia.org/wiki/Morris_worm
- https://www.rapid7.com/blog/post/2019/02/19/stack-based-buffer-overflow-attacks-what-you-need-to-know/
- https://vpnoverview.com/internet-safety/business/what-is-secure-coding/
- https://en.wikipedia.org/wiki/Buffer_overflow#History
- https://en.wikipedia.org/wiki/Homebrew_(video_games)#PlayStation_2
- https://www.iso.org/standard/57853.html
- https://cwe.mitre.org/

MIT-WPU

।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Dr. Vishwanath Karad

MIT WORLD PEACE UNIVERSITY | PUNE

TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Thankyou!