CTF-League

Systems and Security Group, WEC

What is an Error?

What is an error?

- An problem present in a program which hinders proper execution.

Types ?

What is an error?

- An problem present in a program which hinders proper execution.

Types:

1. Compilation Errors
2. Runtime Errors

What is a runtime error?

- An error which occurs when a program is running.

Examples?

Example1.c

```c
int main()
{
    /* Declare the variable */
    int *ptr = NULL;

    /* Allocate memory for it */
    ptr = malloc(1000000000000);

    /* Give it a value */
    *ptr = 10;

    printf("ptr = %p, value = %d\n", ptr, *ptr);

    return 0;
}
```

Example1.c

- No error handling - A major source of runtime errors.

- Systems Programming mandates rigorous error handling!
    - Even if it fails, let it fail peacefully. Let it not take down the whole damn thing :P

Good read: On rigorous Error handling

Example2.c

```c
int a[n]; /* Assume input is taken */

for(int i = 0; i < n; i++)
{
    for(int j = i; j < n; j++)
    {
        if(a[j] > a[j+1])
            swap(a[j], a[j+1]);
    }
}
```

What does this code intend to do? What is the issue here?

Example3.c

What is happening here?

Example3.c

What is happening here?

Its an example of a class security vulnerability, the **Buffer Overflow**.

How can we catch these errors / bugs?

1. Give random inputs.

How can we catch these errors / bugs?

1. Give random inputs.
2. Dissect the program and clearly see what is happening.

Good reads:

a. Fuzz Testing

How can we catch these errors / bugs?

1. Give random inputs.
2. Dissect the program and clearly see what is happening.
    1. Catch those sneaky bastards!

Worst case, just wait for it to fail :P

Dissecting the Program aka Reverse Engineering!

Let us get started!

Introduction to gdb

- GNU Debugger
- Helps to run any program instruction by instruction.
  - Can check variables’ values.
  - Can control function calls.

Basically can do anything with your program.

Introduction to gdb

- GNU Debugger
- Helps to run any program instruction by instruction.
    - Can check variables’ values.
    - Can control function calls.

Basically can do anything with your program.

Let us analyze the first example.

Introduction to gdb

1. Starting gdb.

```
$ gdb -q <program-name>
```

2. Setting breakpoints.

```
(gdb) breakpoint <function_name> : breakpoint main

(gdb) bp <line-no> : bp 10
```

3. To go to the next instruction.

```
(gdb) ni
```

Introduction to gdb

4. Info about local variables

```
(gdb) info locals
```

5. Listing the program

```
(gdb) list

(gdb) list <line-no>
```

Practicals!

Questions?

Further Reading

Checkout the official CTF-League github repository for today's writeups and more resources.

1. GDB tutorial
2. The Live Overflow Youtube channel
3. Reverse Engineering and Binary Exploitation Series

What Next?

Thank you :-)

Contact anytime!

Adwaith Gautham, 4th Year CSE: +91-9663572932, adwait.gautham@gmail.com