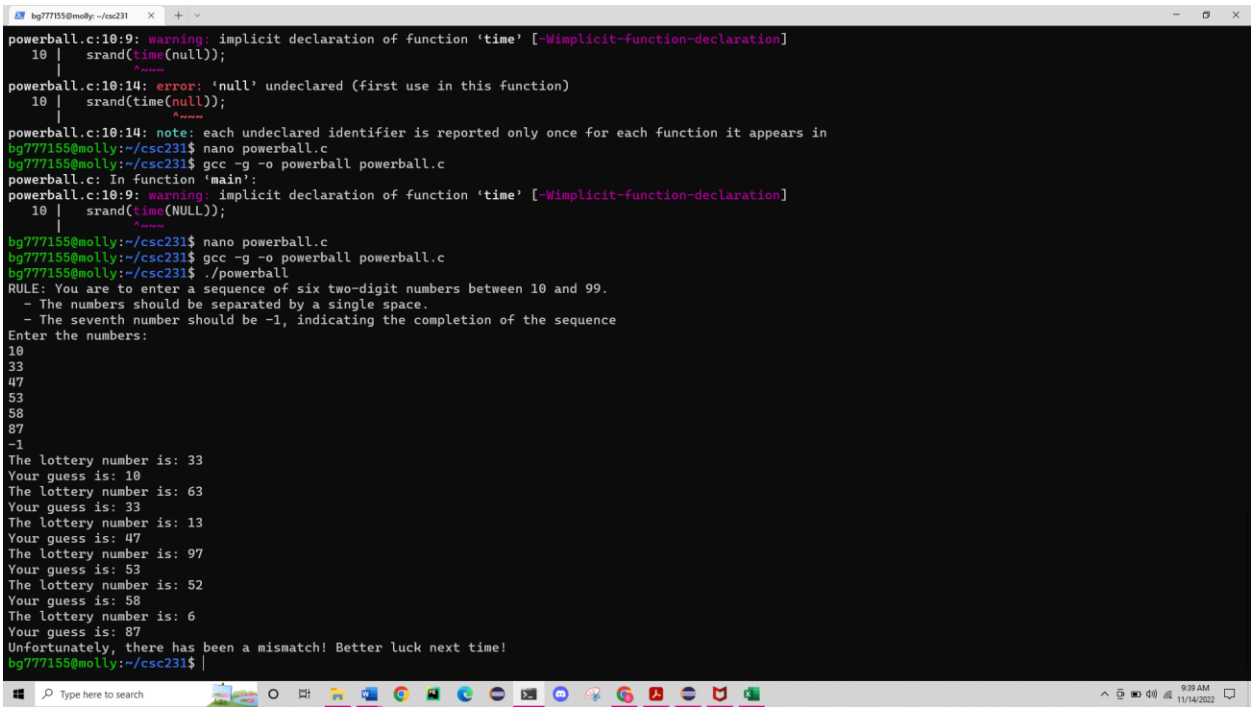
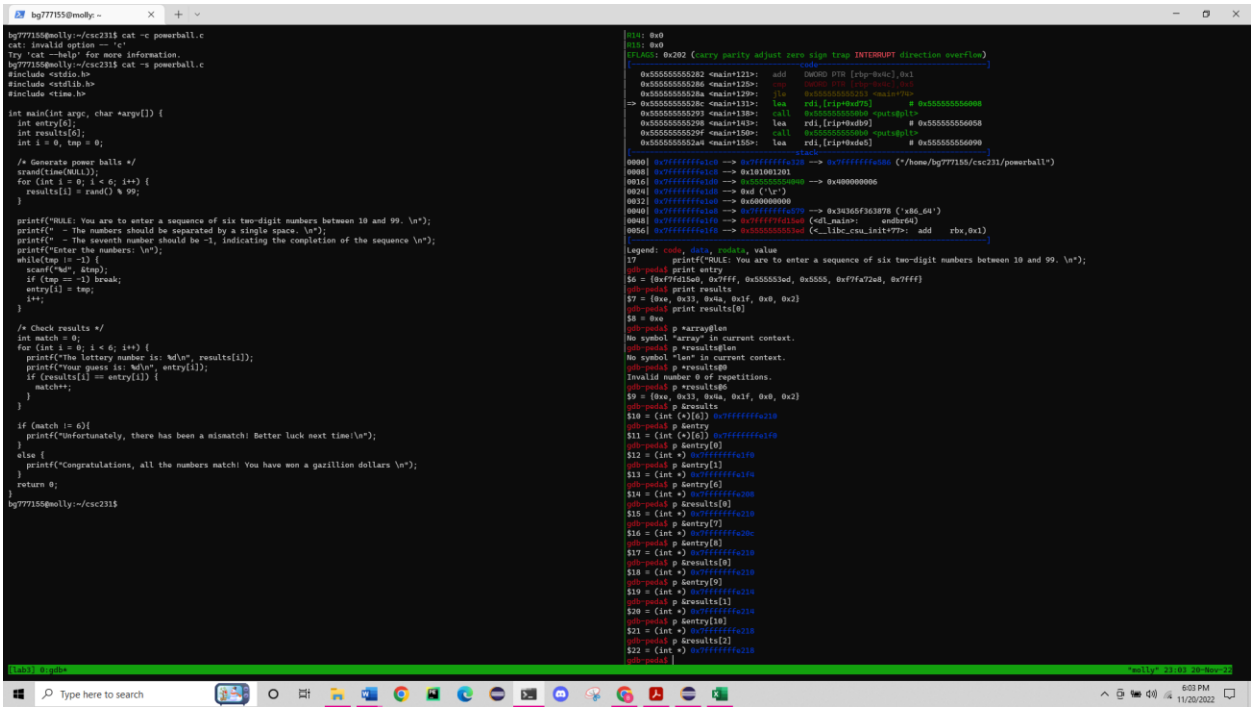


```
powerball.c:10:9: warning: implicit declaration of function ‘time’ [-Wimplicit-function-declaration]
   10 |   srand(time(null));
      |         ^~~~
powerball.c:10:14: error: ‘null’ undeclared (first use in this function)
   10 |   srand(time(null));
      |              ^~~~
powerball.c:10:14: note: each undeclared identifier is reported only once for each function it appears in
bg777155@molly:~/csc231$ nano powerball.c
bg777155@molly:~/csc231$ gcc -g -o powerball powerball.c
powerball.c: In function ‘main’:
powerball.c:10:9: warning: implicit declaration of function ‘time’ [-Wimplicit-function-declaration]
   10 |   srand(time(NULL));
      |         ^~~~
bg777155@molly:~/csc231$ nano powerball.c
bg777155@molly:~/csc231$ gcc -g -o powerball powerball.c
bg777155@molly:~/csc231$ ./powerball
RULE: You are to enter a sequence of six two-digit numbers between 10 and 99.
  - The numbers should be separated by a single space.
  - The seventh number should be -1, indicating the completion of the sequence
Enter the numbers:
10
33
47
53
58
87
-1
The lottery number is: 33
Your guess is: 10
The lottery number is: 63
Your guess is: 33
The lottery number is: 13
Your guess is: 47
The lottery number is: 97
Your guess is: 53
The lottery number is: 52
Your guess is: 58
The lottery number is: 6
Your guess is: 87
Unfortunately, there has been a mismatch! Better luck next time!
bg777155@molly:~/csc231$
```



```
bg777155@molly:~/csc231$ cat -c powerball.c
cat: invalid option -- 'c'
Try 'cat --help' for more information.
bg777155@molly:~/csc231$ cat -s powerball.c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(int argc, char *argv[]) {
  int entry[6];
  int results[6];
  int i = 0, tmp = 0;

  /* Generate power balls */
  srand(time(NULL));
  for (int i = 0; i < 6; i++) {
    results[i] = rand() % 99;
  }

  printf("RULE: You are to enter a sequence of six two-digit numbers between 10 and 99. \n");
  printf("  - The numbers should be separated by a single space. \n");
  printf("  - The seventh number should be -1, indicating the completion of the sequence \n");
  printf("Enter the numbers: \n");
  while(tmp != -1) {
    scanf("%d", &tmp);
    if (tmp == -1) break;
    entry[i] = tmp;
    i++;
  }

  /* Check results */
  int match = 0;
  for (int i = 0; i < 6; i++) {
    printf("The lottery number is: %d\n", results[i]);
    printf("Your guess is: %d\n", entry[i]);
    if (results[i] == entry[i]) {
      match++;
    }
  }

  if (match != 6){
    printf("Unfortunately, there has been a mismatch! Better luck next time!\n");
  }
  else {
    printf("Congratulations, all the numbers match! You have won a gazillion dollars \n");
  }
  return 0;
}
bg777155@molly:~/csc231$
```

```
R14: 0x0
R15: 0x0
EFLAGS: 0x202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
[ ──────────────────────────────────code──────────────────────────────────────── ]
   0x555555555282 <main+121>:   add    DWORD PTR [rbp-0x4c],0x1
   0x555555555286 <main+125>:   cmp    DWORD PTR [rbp-0x4c],0x5
   0x55555555528a <main+129>:   jle    0x555555555233 <main+78>
=> 0x55555555528c <main+131>:   lea    rdi,[rip+0xd79]        # 0x555555556008
   0x555555555293 <main+138>:   call   0x5555555550b0 <puts@plt>
   0x555555555298 <main+143>:   lea    rdi,[rip+0xdb9]        # 0x555555556058
   0x55555555529f <main+150>:   call   0x5555555550a0 <puts@plt>
   0x5555555552a4 <main+155>:   lea    rdi,[rip+0xde5]        # 0x555555556090
[ ──────────────────────────────────stack─────────────────────────────────────── ]
0000| 0x7fffffffe1c0 --> 0x7fffffffe328 --> 0x7fffffffe588 ("/home/bg777155/csc231/powerball")
0008| 0x7fffffffe1c8 --> 0x101001201
0016| 0x7fffffffe1d0 --> 0x555555554040 --> 0x400000006
0024| 0x7fffffffe1d8 --> 0xd ('\r')
0032| 0x7fffffffe1e0 --> 0x600000000
0040| 0x7fffffffe1e8 --> 0x7fffffffe670 --> 0x34365f363878 ('x86_64')
0048| 0x7fffffffe1f0 --> 0x7ffff7fd15e0 (<dl_main>:    endbr64)
0056| 0x7fffffffe1f8 --> 0x5555555553ed (<__libc_csu_init+77>:  add    rbx,0x1)
[ ──────────────────────────────────────────────────────────────────────────── ]
Legend: code, data, rodata, value
17        printf("RULE: You are to enter a sequence of six two-digit numbers between 10 and 99. \n");
gdb-peda$ print entry
$6 = {0xf7fd15e0, 0x7fff, 0x555553ed, 0x5555, 0xf7fa72e8, 0x7fff}
gdb-peda$ print results
$7 = {0xa, 0x33, 0x4a, 0x1f, 0x0, 0x2}
gdb-peda$ print results[0]
$8 = 0xa
gdb-peda$ p *array@len
No symbol "array" in current context.
gdb-peda$ p *results@len
No symbol "len" in current context.
gdb-peda$ p *results@0
Invalid number 0 of repetitions.
gdb-peda$ p *results@6
$9 = {0xa, 0x33, 0x4a, 0x1f, 0x0, 0x2}
gdb-peda$ p &results
$10 = (int (*)[6]) 0x7fffffffe210
gdb-peda$ p &entry
$11 = (int (*)[6]) 0x7fffffffe1f0
gdb-peda$ p &entry[0]
$12 = (int *) 0x7fffffffe1f0
gdb-peda$ p &entry[1]
$13 = (int *) 0x7fffffffe1f4
gdb-peda$ p &entry[6]
$14 = (int *) 0x7fffffffe208
gdb-peda$ p &results[0]
$15 = (int *) 0x7fffffffe210
gdb-peda$ p &entry[7]
$16 = (int *) 0x7fffffffe20c
gdb-peda$ p &entry[8]
$17 = (int *) 0x7fffffffe210
gdb-peda$ p &results[8]
$18 = (int *) 0x7fffffffe210
gdb-peda$ p &entry[9]
$19 = (int *) 0x7fffffffe214
gdb-peda$ p &results[1]
$20 = (int *) 0x7fffffffe214
gdb-peda$ p &entry[10]
$21 = (int *) 0x7fffffffe218
gdb-peda$ p &results[2]
$22 = (int *) 0x7fffffffe218
gdb-peda$
```

Which array is likely to grow into the other should out-of-bound indices are accessed?

Answer: As you can see above, the entry array is addressed in memory before the results array and grows into the results array when accessed out of bounds.

The exploit is performed by entering 6 single-digit or double-digit integers that are separated by spaces followed by two more filler integers separated by spaces. This fills up entry[0] through entry[7]. Since &entry[8] = &results[0], the next six numbers we enter must be the same as the first 6. This ensures that both arrays have the same 6 values since we overwrite the 6 random numbers already in the results array.