CSE 406
Online -1 (A - 1)

You are given the following vulnerable C program *A1.c.* Replace <param_1> , <param_2> and <param_3> in the source code with the corresponding values of Table-1.

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int foo(char *str)
{
    int arr[<param_1>];
    char buffer[<param_2>];

    /* The following statement has a buffer overflow problem */
    strcpy(buffer, str);

    return 1;
}

int main(int argc, char **argv)
{
    char str[<param_3>];
    FILE *badfile;

    badfile = fopen("badfile", "r");
    fread(str, sizeof(char), <param_3>, badfile);
    foo(str);

    printf("Try Again\n");
    return 1;
}
```

Tasks:

1. First, compile the program from the root's privilege and set its UID as shown in the lab. Do not forget to turn off address space randomization and stack protection. Also, make sure that the stack is executable while compiling the program.
2. Prepare a payload (e.g. badfile) which will cause the program to open a shell with root's privilege when executed by other users (seed).
3. Make sure that the shellcode is placed below the return address. (Figure-1). Note that during lab demonstration the payload was placed above the return address.
4. Prepare the payload in such a way that the attack will be successful with the same payload even if the buffer size is increased or decreased by 24 bytes.
5. Rename your ***exploit.py*** file with ***16050xx.py*** and submit in moodle.

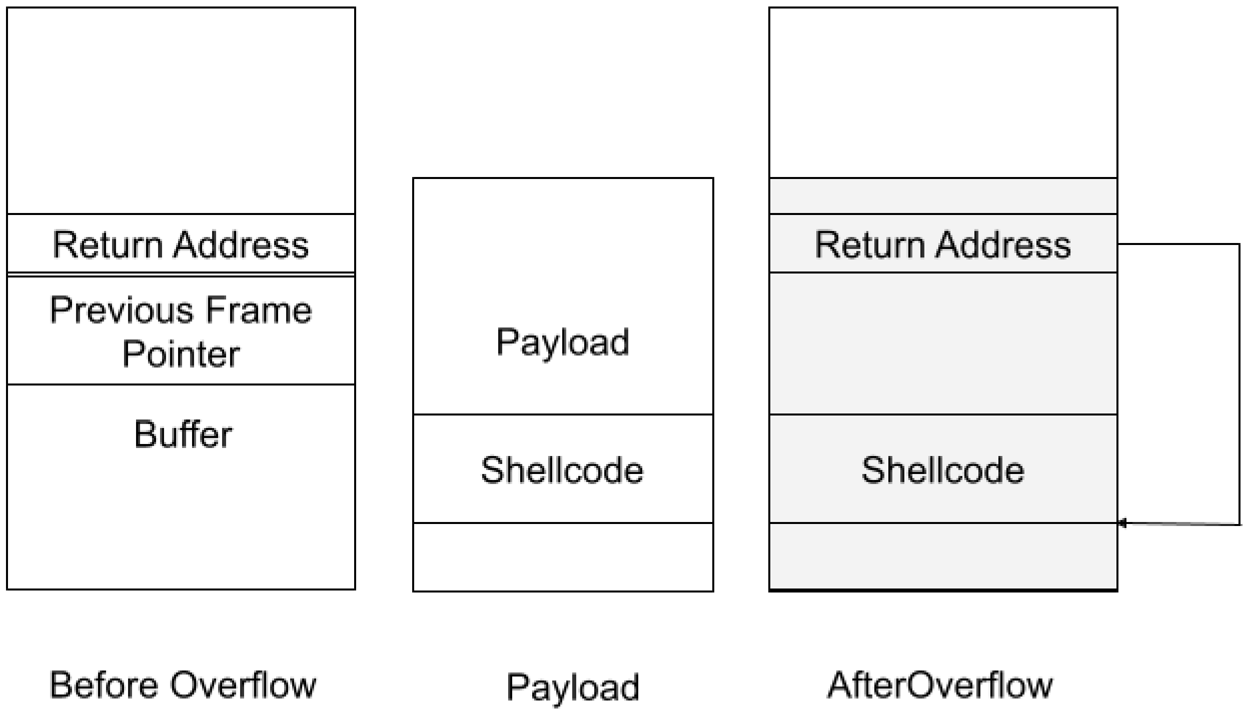

Figure-1

Student ID	param_1	param_2	param_3
1605001	40	308	528
1605002	39	573	792
1605003	35	698	913
1605004	30	627	837
1605005	31	432	643
1605006	40	669	889
1605007	20	339	539
1605008	23	730	933
1605009	31	667	878
1605010	30	781	991
1605011	23	432	635
1605012	37	419	636
1605013	38	479	697
1605014	22	318	520
1605015	25	444	649

1605016	31	707	918
1605017	34	388	602
1605018	33	780	993
1605019	35	549	764
1605020	30	495	705
1605021	27	437	644
1605022	26	699	805
1605023	29	543	752
1605024	26	408	614
1605025	22	625	827
1605026	40	548	768
1605027	24	476	680
1605028	31	312	523
1605029	26	727	933
1605030	27	575	782
0905081	25	474	679
1405059	26	718	924
1405081	40	372	592
1505004	40	400	620

Table-1