Penetration Testing Report

For

“Strippped”

S.NO.	Title	#
1.	Challenge Category	PWN
2.	Challenge Related Files	N/A
3.	File Link / Target IP	N/A

PROCEDURE

1. You are given a binary file but most of the data is stripped.
2. Load that binary locally and try to play around check functions / how this binary actually works.
3. By hand try to assemble pseudo functions.
4. Most noticeable thing is running it with strace then you get to know that it can execute cat flag.txt but it also takes a key file to decrypt incoming commands.
5. In order to solve, we need to things cat flag.txt and encrypt that command with a key
6. While solving you will get to know that key is only one char. So we can bruteforce that key
7. Also here encryption/decryption is just doing xor only
8. So you can write your final exploit script

9.

```python
from pwn import *

def send_enc_command(io, key):
    command = "cat flag.txt"
    command = ''.join([chr(ord(i) ^ key) for i in command])
    io.sendline('1')
    io.send(command)

def execute(io):
    io.sendline('2')
    io.recvuntil('>')
    io.sendline('1')


def exploit(io, key):
    io.recvuntil('>')
    send_enc_command(io, key)
    io.recvuntil('>')
    execute(io)
    return io.recvall()

if __name__ == '__main__':
    for key in range(256):
        io = remote("localhost", 31337)
        ans = exploit(io, key)
        io.close()
        if 'HE{' in ans:
            print ans
            break
        io.close()
```

Source functions :

```
 8
 9  void fatal() {
10      puts("Something went wrong");
11      exit(0);
12  }
13
14  void print_menu() {
15      puts("Choose option:");
16      puts("1. Load command");
17      puts("2. Execute command");
18      puts("3. Exit");
19  }
20
21  int get_choice() {
22      int number;
23      scanf("%d", &number);
24      return number;
25  }
26
27  int load_key() {
28      FILE *fptr;
29      char ch;
30      fptr = fopen("key", "r");
31      if (fptr == NULL)
32          fatal();
33      ch = fgetc(fptr);
34      if (ch == EOF)
35          fatal();
36      fclose(fptr);
37      return ch;
38  }
39
40  void decrypt(char * command) {
41      int key = load_key();
42      for (int i = 0; i < strlen(command); i++)
43          command[i] ^= key;
44  }
```

```c
void load_command(char ** storage, int * pointer) {
    if (*pointer > (MAX_STORAGE - 1))
        fatal();
    char * command = malloc(MAX_BUFFER);
    for (int i = 0; i < MAX_BUFFER; i++)
        command[i] = 0;
    int length = read(0, command, MAX_BUFFER);
    if (strlen(command) < MAX_BUFFER)
        command[length - 1] = 0;
    decrypt(command);
    storage[*pointer] = command;
    *pointer += 1;
}

void execute_command(char ** storage) {
    puts("Choose command:");
    for (int i = 0; i < MAX_STORAGE; i++) {
        printf("%d. - ", i + 1);
        if (storage[i] == NULL)
            puts("Nothing here");
        else
            printf("%s\n", storage[i]);
    }
    printf("> ");

    int choice = get_choice();
    if (choice < 1 || choice > MAX_STORAGE)
        fatal();
    choice -= 1;

    if (storage[choice] == NULL)
        fatal();
    else if (!strcmp(storage[choice], "cat flag.txt")) {
        system("cat flag.txt");
        puts("");
        exit(0);
    }
    else
        fatal();
}
```

Flags:

S.No.	Flag - No.	Flag
1.	Flag 1	HE{Stripped_or_nonStriped_Pwning_is_hard}