jamk.fi

# Software exploitation

## Assignment1

Mikael Romanov

Assignment1
Maaliskuu 2018
Tekniikan ja liikenteen ala
Insinööri (AMK), tieto- ja viestintätekniikan tutkinto-ohjelma
Kyberturvallisuus

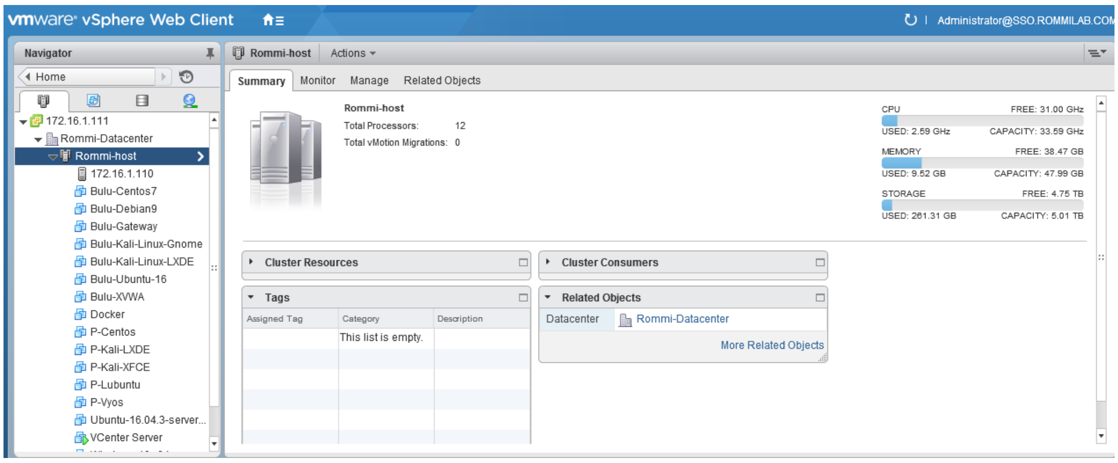
## 1 Table of Contents

## 1 Introduction

This assignments goal was to create customized input to a maze game by reviewing the written code. The maze had levels which were printed out as progressed. The game had 7 levels that had different kind of challenges. The maze gradually increased in difficulty.

## 2 Testing platform

I used my DL380 G7 server with VMware ESXI 6.0 which had enough horse power to go around. All though this assignment didn't need any kind of resources, Kali Linux worked like a charm (Figure 1)



Rommi-host
Total Processors: 12
Total vMotion Migrations: 0

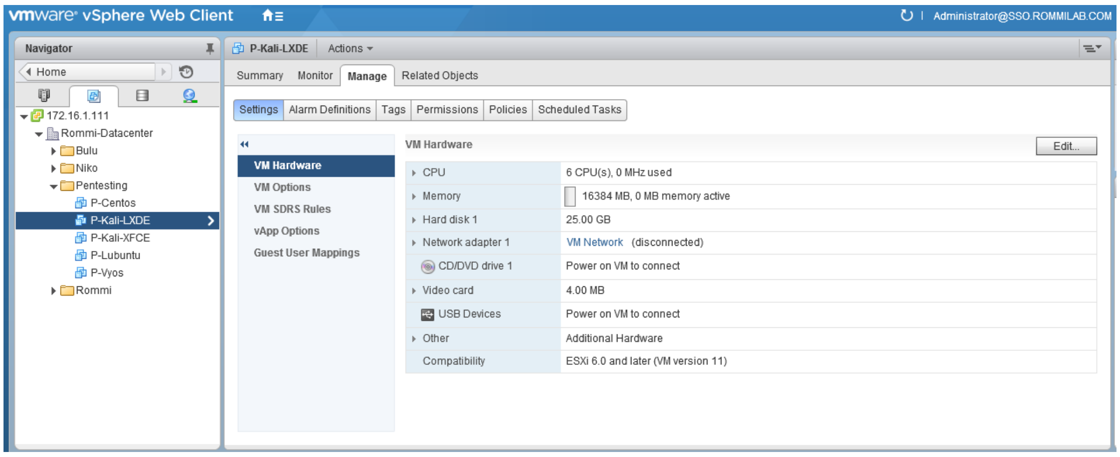
CPU FREE: 31.00 GHz
USED: 2.59 GHz CAPACITY: 33.59 GHz
MEMORY FREE: 38.47 GB
USED: 9.52 GB CAPACITY: 47.99 GB
STORAGE FREE: 4.75 TB
USED: 261.31 GB CAPACITY: 5.01 TB

| Assigned Tag | Category | Description |
|---|---|---|
| | This list is empty. | |

| | |
|---|---|
| Datacenter | Rommi-Datacenter |

Figure 1 VMware esxi6.0

I reserved ridiculous amount of resources for the KALI vm and they were following(Figure 2)



vmware vSphere Web Client | Administrator@SSO.ROMMILAB.COM

Navigator | Home

```
172.16.1.111
  Rommi-Datacenter
    Bulu
    Niko
    Pentesting
      P-Centos
      P-Kali-LXDE
      P-Kali-XFCE
      P-Lubuntu
      P-Vyos
    Rommi
```

P-Kali-LXDE | Actions

Summary | Monitor | Manage | Related Objects

Settings | Alarm Definitions | Tags | Permissions | Policies | Scheduled Tasks

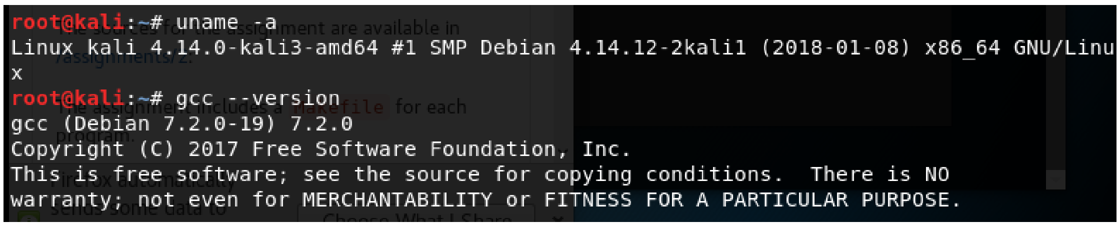
VM Hardware | VM Options | VM SDRS Rules | vApp Options | Guest User Mappings

VM Hardware | Edit...

| | |
|---|---|
| CPU | 6 CPU(s), 0 MHz used |
| Memory | 16384 MB, 0 MB memory active |
| Hard disk 1 | 25.00 GB |
| Network adapter 1 | VM Network (disconnected) |
| CD/DVD drive 1 | Power on VM to connect |
| Video card | 4.00 MB |
| USB Devices | Power on VM to connect |
| Other | Additional Hardware |
| Compatibility | ESXi 6.0 and later (VM version 11) |

Figure 2 resources

As seen below I used Kali Linux 4.14.0-kali3-amd64, with gcc version 7.2.0.(Figure3) I cloned wrong machine from my library, it was supposed to be LXDE but was instead the vanilla version which has too much clutter on the screen for my taste. Oh well.

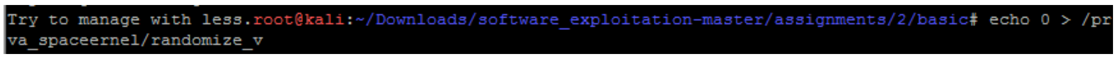


```
root@kali:~# uname -a
Linux kali 4.14.0-kali3-amd64 #1 SMP Debian 4.14.12-2kali1 (2018-01-08) x86_64 GNU/Linu
x
root@kali:~# gcc --version
gcc (Debian 7.2.0-19) 7.2.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Figure 3 versions

I used the make file to compile the program and also ***gcc -g maze.c*** for debugging purposes. Thee make file was simple to us to compile the file, it only required the command ***make.*** The MAKEFILE used also gcc to compile the program.

After compiling the program I turned ASLR off, because it is hard to know where the memory is allocated next. ASLR can be set off by echoing ***echo 0 > /proc/sys/kernel/randomize_va_space*** (Figure 4)

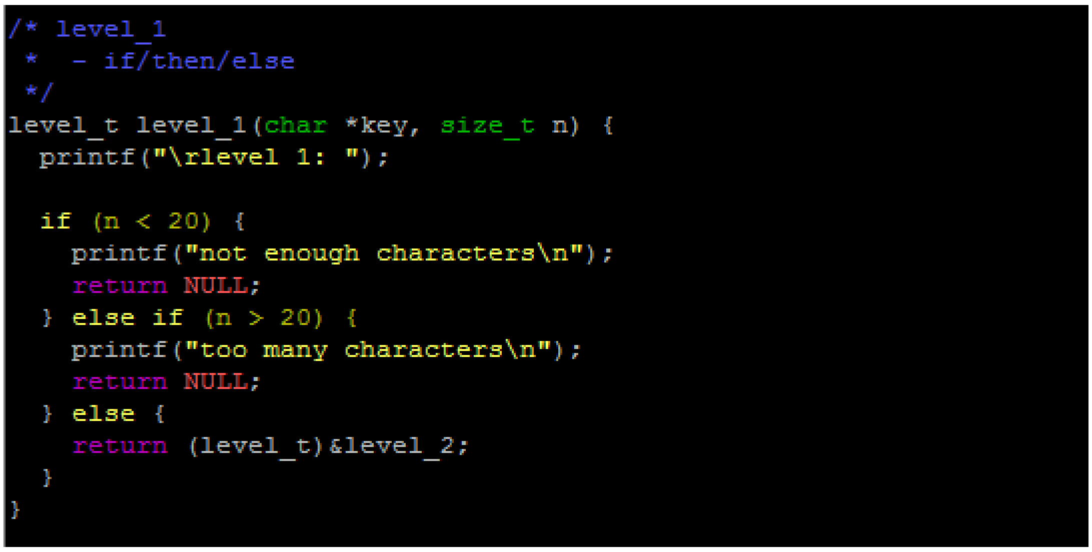


```
Try to manage with less.root@kali:~/Downloads/software_exploitation-master/assignments/2/basic# echo 0 > /pr
va_spaceernel/randomize_v
```

Figure 4 aslr off

## 3 Testing

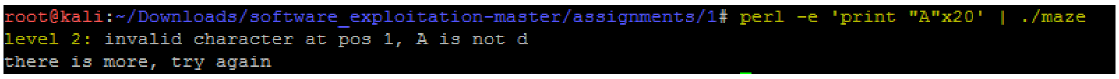
I started my testing by reviewing the code (maze.c). I found out that to pass level 1 the input needs to be exactly 20 characters long. (Figure5)



```
/* level_1
 *   - if/then/else
 */
level_t level_1(char *key, size_t n) {
  printf("\rlevel 1: ");

  if (n < 20) {
    printf("not enough characters\n");
    return NULL;
  } else if (n > 20) {
    printf("too many characters\n");
    return NULL;
  } else {
    return (level_t)&level_2;
  }
}
```

Figure 5 firts level

So when I saw this I tested with perl by echoing “A” character 20 times with command ***perl -e ‘print “A”x20’ | ./maze*** and got to level 2 (Figure6)

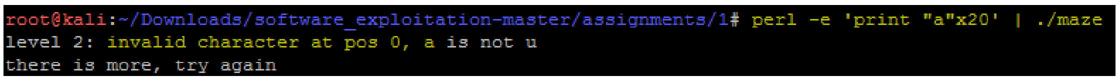


```
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e 'print "A"x20' | ./maze
level 2: invalid character at pos 1, A is not d
there is more, try again
```

Figure 6 first input

I then started experimenting with the input and somehow ignored the hint that was telling me that the character at index position[1] is not A.

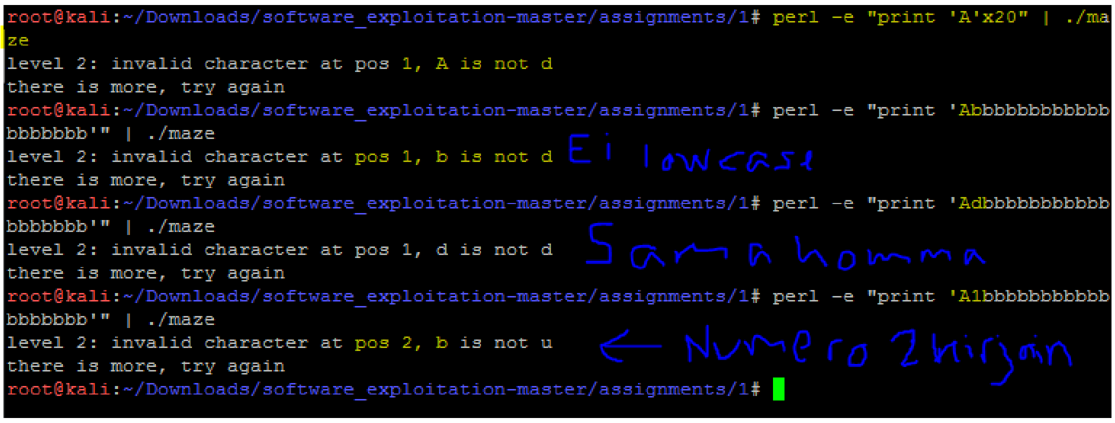
So by ignoring the hint I then tested with lowercase “a” the output was same although I then saw that the index position changed to [0] (Figure7)



```
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e 'print "a"x20' | ./maze
level 2: invalid character at pos 0, a is not u
there is more, try again
```

Figure 7 second test input

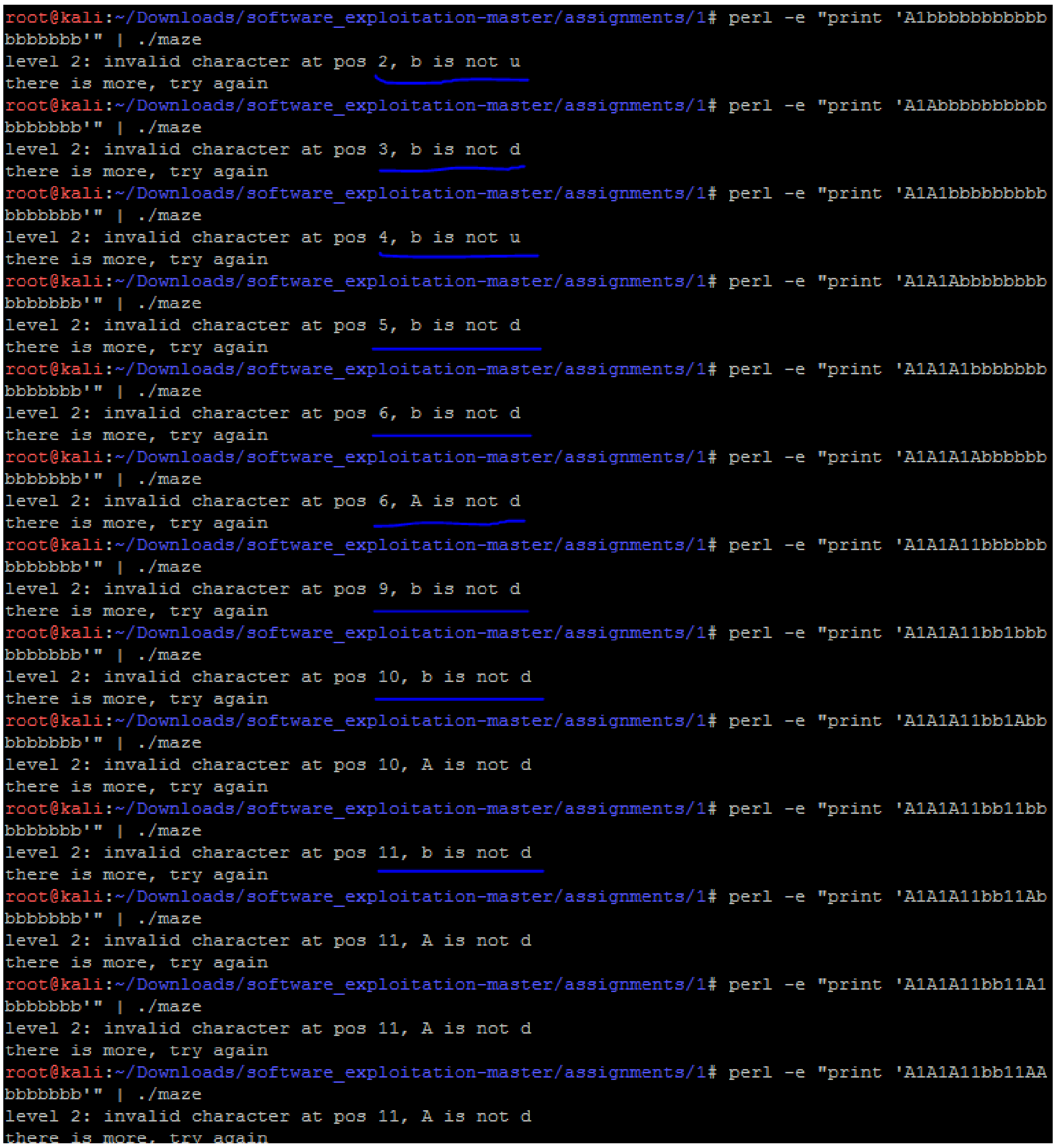
I should’ve reviewed the code before but I soon realized that the level 2 requires a pattern. A pattern that is constructed of lowercase, uppercase, and a one special character. I could’ve write a fast python script to do all the typing, but I was too eager to test out if my logic was correct. I knew that the first character is an uppercase character. (Figure8)



```
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A'x20" | ./ma
ze
level 2: invalid character at pos 1, A is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'Abbbbbbbbbbbb
bbbbbbb'" | ./maze
level 2: invalid character at pos 1, b is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'Adbbbbbbbbbbb
bbbbbbb'" | ./maze
level 2: invalid character at pos 1, d is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1bbbbbbbbbbb
bbbbbbb'" | ./maze
level 2: invalid character at pos 2, b is not u
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1#
```

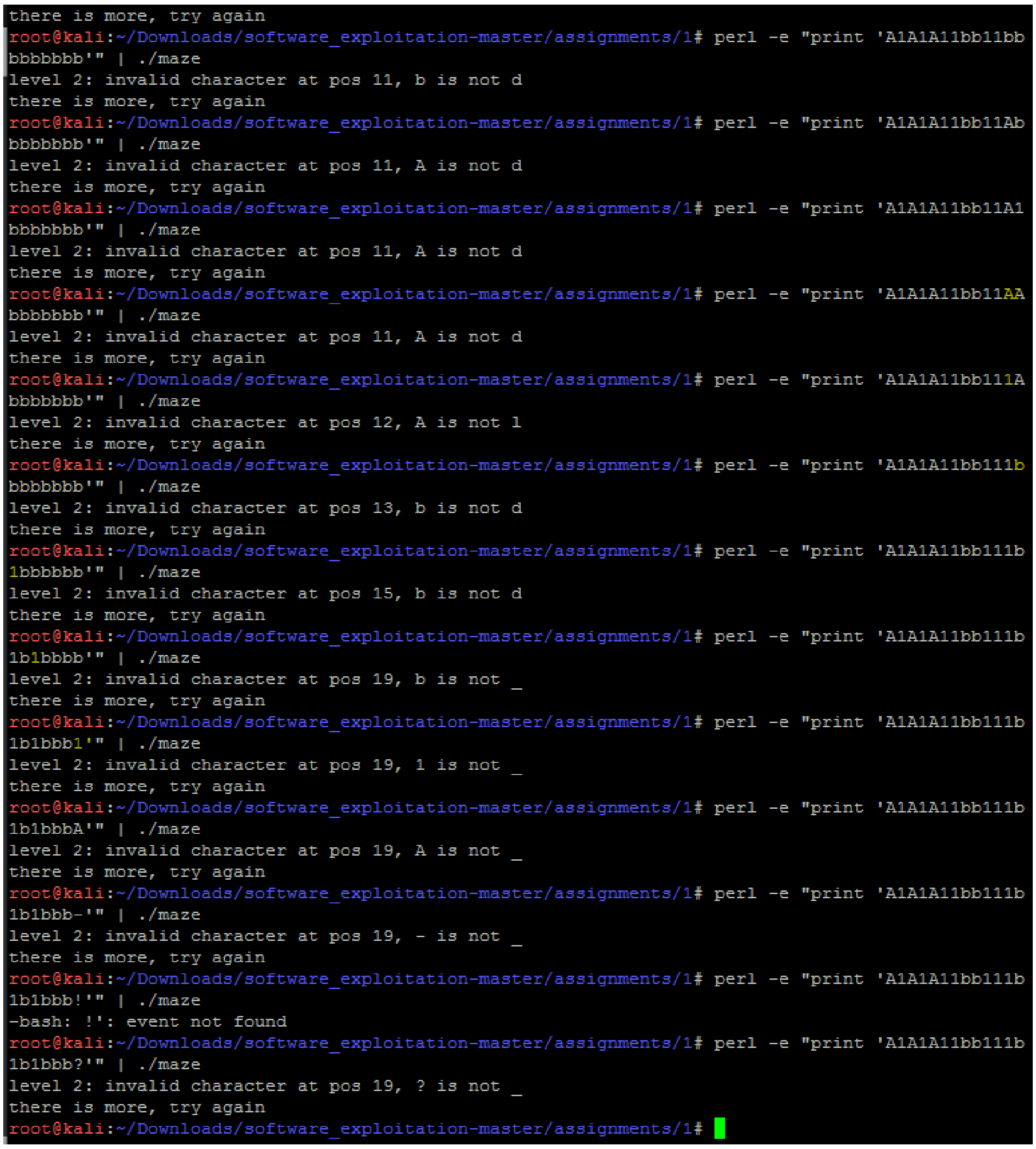
Figure 8 figuring out

I was correct! Then I started doing same things to the following characters one by one. (Figure 9)



```
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1bbbbbbbbbbb
bbbbbbb'" | ./maze
level 2: invalid character at pos 2, b is not u
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1Abbbbbbbbbb
bbbbbbb'" | ./maze
level 2: invalid character at pos 3, b is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1bbbbbbbbb
bbbbbbb'" | ./maze
level 2: invalid character at pos 4, b is not u
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1Abbbbbbbb
bbbbbbb'" | ./maze
level 2: invalid character at pos 5, b is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A1bbbbbbb
bbbbbbb'" | ./maze
level 2: invalid character at pos 6, b is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A1Abbbbbb
bbbbbbb'" | ./maze
level 2: invalid character at pos 6, A is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bbbbbb
bbbbbbb'" | ./maze
level 2: invalid character at pos 9, b is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb1bbb
bbbbbbb'" | ./maze
level 2: invalid character at pos 10, b is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb1Abb
bbbbbbb'" | ./maze
level 2: invalid character at pos 10, A is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb11bb
bbbbbbb'" | ./maze
level 2: invalid character at pos 11, b is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb11Ab
bbbbbbb'" | ./maze
level 2: invalid character at pos 11, A is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb11A1
bbbbbbb'" | ./maze
level 2: invalid character at pos 11, A is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb11AA
bbbbbbb'" | ./maze
level 2: invalid character at pos 11, A is not d
there is more, try again
```

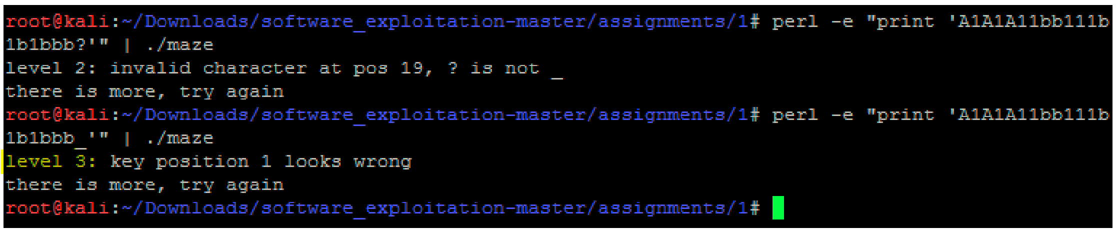
Figure 9 onebyone



```
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb11bb
bbbbbbb'" | ./maze
level 2: invalid character at pos 11, b is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb11Ab
bbbbbbb'" | ./maze
level 2: invalid character at pos 11, A is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb11A1
bbbbbbb'" | ./maze
level 2: invalid character at pos 11, A is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb11AA
bbbbbbb'" | ./maze
level 2: invalid character at pos 11, A is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb111A
bbbbbbb'" | ./maze
level 2: invalid character at pos 12, A is not l
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb111b
bbbbbbb'" | ./maze
level 2: invalid character at pos 13, b is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb111b
1bbbbbb'" | ./maze
level 2: invalid character at pos 15, b is not d
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb111b
1b1bbbb'" | ./maze
level 2: invalid character at pos 19, b is not _
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb111b
1b1bbb1'" | ./maze
level 2: invalid character at pos 19, 1 is not _
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb111b
1b1bbbA'" | ./maze
level 2: invalid character at pos 19, A is not _
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb111b
1b1bbb-'" | ./maze
level 2: invalid character at pos 19, - is not _
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb111b
1b1bbb!'" | ./maze
-bash: !': event not found
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb111b
1b1bbb?'" | ./maze
level 2: invalid character at pos 19, ? is not _
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1#
```

Figure 10 almost at level3

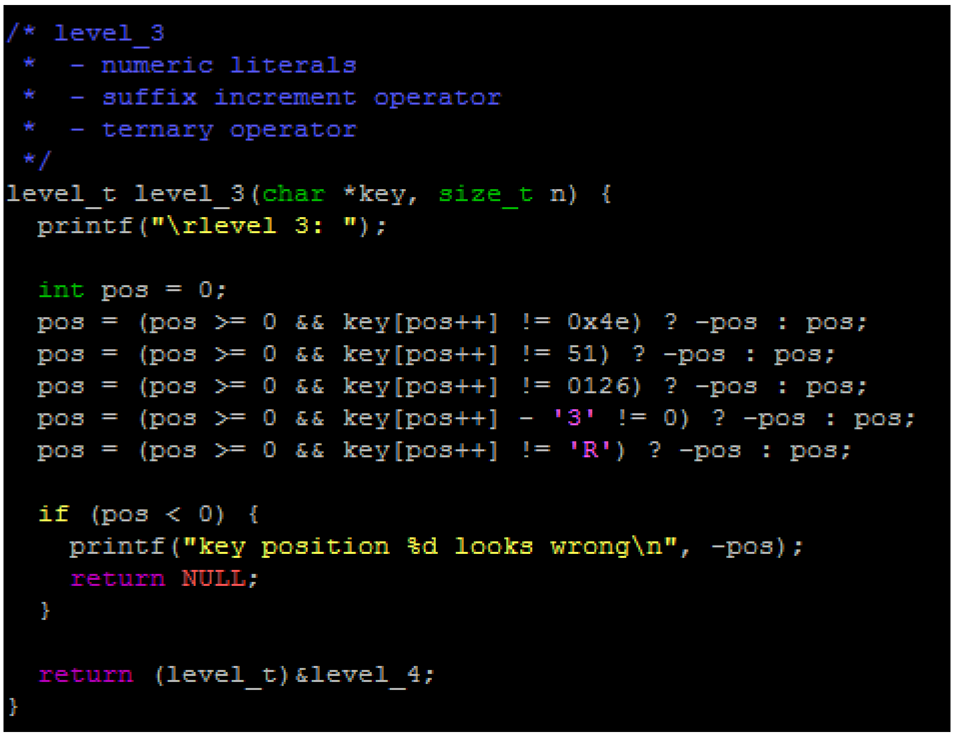
The last character was took me little time to test if it was actually _ and it was (Figure 11)



```
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb111b
1b1bbb?'" | ./maze
level 2: invalid character at pos 19, ? is not _
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1# perl -e "print 'A1A1A11bb111b
1b1bbb_'" | ./maze
level 3: key position 1 looks wrong
there is more, try again
root@kali:~/Downloads/software_exploitation-master/assignments/1#
```

Figure 11 level 3

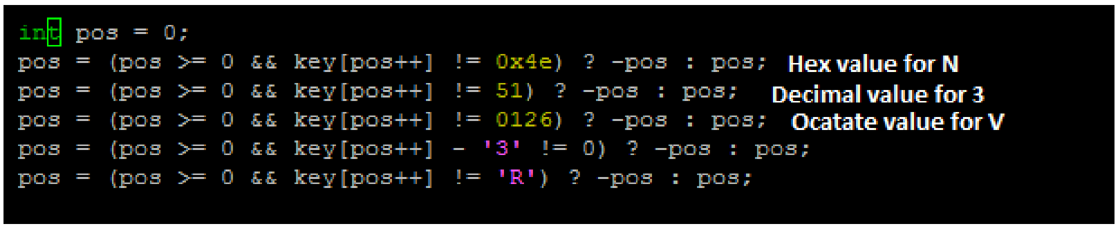
After I reached level 3 I started reviewing the script of what was going on the part three, this time I saw the hint but was baffled why it said position 1. The first character couldn't be "A", I was sure about it. First looking at the level 3 in thought to myself what are these values here(Figure 12)



```
/* level_3
 *  - numeric literals
 *  - suffix increment operator
 *  - ternary operator
 */
level_t level_3(char *key, size_t n) {
  printf("\rlevel 3: ");

  int pos = 0;
  pos = (pos >= 0 && key[pos++] != 0x4e) ? -pos : pos;
  pos = (pos >= 0 && key[pos++] != 51) ? -pos : pos;
  pos = (pos >= 0 && key[pos++] != 0126) ? -pos : pos;
  pos = (pos >= 0 && key[pos++] - '3' != 0) ? -pos : pos;
  pos = (pos >= 0 && key[pos++] != 'R') ? -pos : pos;

  if (pos < 0) {
    printf("key position %d looks wrong\n", -pos);
    return NULL;
  }

  return (level_t)&level_4;
}
```

Figure 12 level 3 logic

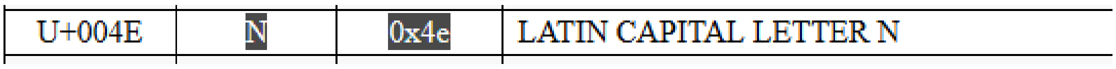
After little time I understood what those were, the statements were pretty simple to figure out. The first value was a hex-value the second was decimal value and the third was octate value. (Figure13)



```
int pos = 0;
pos = (pos >= 0 && key[pos++] != 0x4e) ? -pos : pos;  Hex value for N
pos = (pos >= 0 && key[pos++] != 51) ? -pos : pos;   Decimal value for 3
pos = (pos >= 0 && key[pos++] != 0126) ? -pos : pos;  Ocatate value for V
pos = (pos >= 0 && key[pos++] - '3' != 0) ? -pos : pos;
pos = (pos >= 0 && key[pos++] != 'R') ? -pos : pos;
```

Figure 13 figured values

So by the rules of these statements the first letter of the sentence was uppercase "N" (14)



| | | | |
|---|---|---|---|
| U+004E | N | 0x4e | LATIN CAPITAL LETTER N |

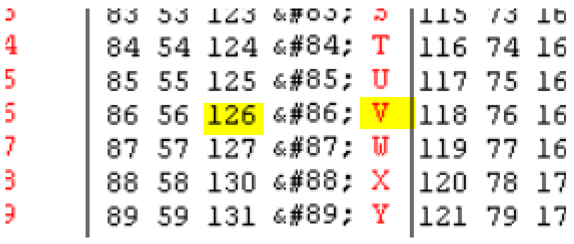
Figure 14 hex value

Second value was the number "3" (Figure15)

```
51 33 063 &#51; 3
```

Figure 15 decimal value for 3

Third value was uppercase letter “V”( Figure16)

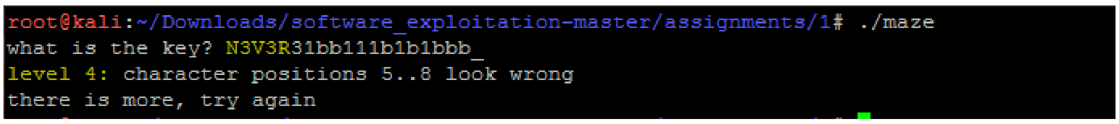


```
84 54 124 &#84; T |116 74 16
85 55 125 &#85; U |117 75 16
86 56 126 &#86; V |118 76 16
87 57 127 &#87; W |119 77 16
88 58 130 &#88; X |120 78 17
89 59 131 &#89; Y |121 79 17
```

Figure 16 octate value for V

Fourth value “R” and the fifth value “3”

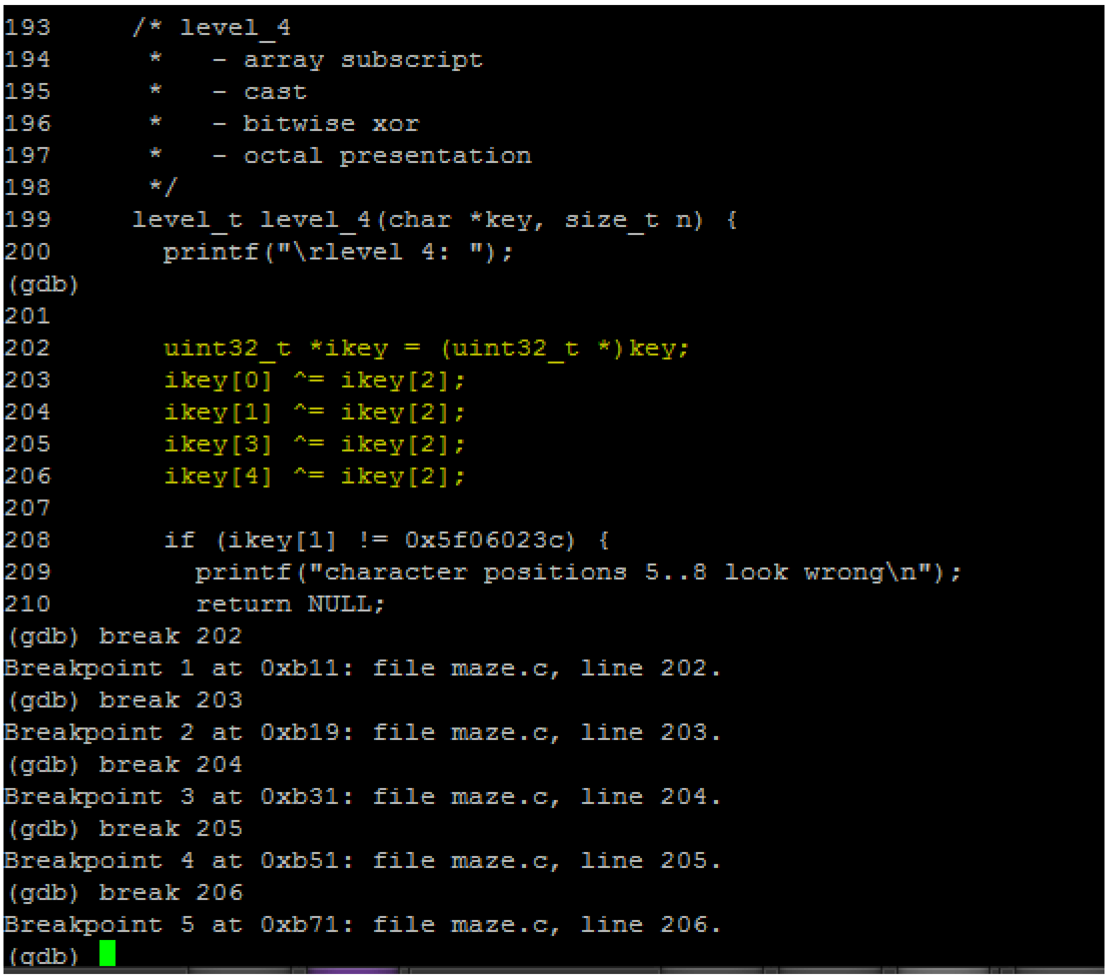
I then tested this logic and got to level 4 (Figure17)



```
root@kali:~/Downloads/software_exploitation-master/assignments/1# ./maze
what is the key? N3V3R31bb111b1b1bbb_
level 4: character positions 5..8 look wrong
there is more, try again
```

Figure 17 testing to get to level 4

I got stuck to the level 4, I understood how the new value of ikey was xorred against the ikey[2]. I tried long time by xorring the values together and by debugging, but some how I missed something (Figure 18)

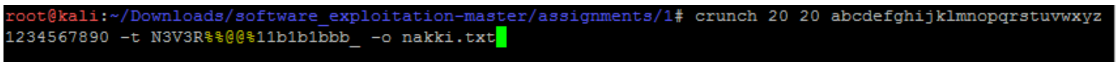


```
193     /* level_4
194      *    - array subscript
195      *    - cast
196      *    - bitwise xor
197      *    - octal presentation
198      */
199     level_t level_4(char *key, size_t n) {
200       printf("\rlevel 4: ");
(gdb)
201
202       uint32_t *ikey = (uint32_t *)key;
203       ikey[0] ^= ikey[2];
204       ikey[1] ^= ikey[2];
205       ikey[3] ^= ikey[2];
206       ikey[4] ^= ikey[2];
207
208       if (ikey[1] != 0x5f06023c) {
209         printf("character positions 5..8 look wrong\n");
210         return NULL;
(gdb) break 202
Breakpoint 1 at 0xb11: file maze.c, line 202.
(gdb) break 203
Breakpoint 2 at 0xb19: file maze.c, line 203.
(gdb) break 204
Breakpoint 3 at 0xb31: file maze.c, line 204.
(gdb) break 205
Breakpoint 4 at 0xb51: file maze.c, line 205.
(gdb) break 206
Breakpoint 5 at 0xb71: file maze.c, line 206.
(gdb)
```

Figure 18

Allthough I missed something out doesn't mean I can get forward in this game. So I can maneuver the things go from a different approach. This approach isn't what the assignment was looking for but I have try it.

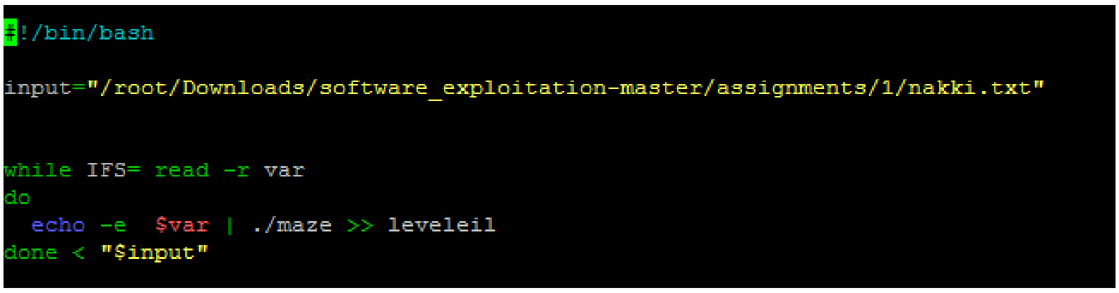
First I made a wordlist with crunch that only changed the 5-8 index values to the corresponding value (Figure 19) I outputted all the different possibilities to a file called nakki.txt



```
root@kali:~/Downloads/software_exploitation-master/assignments/1# crunch 20 20 abcdefghijklmnopqrstuvwxyz
1234567890 -t N3V3R%%@@%11b1b1bbb_ -o nakki.txt
```

Figure 19 wordlist

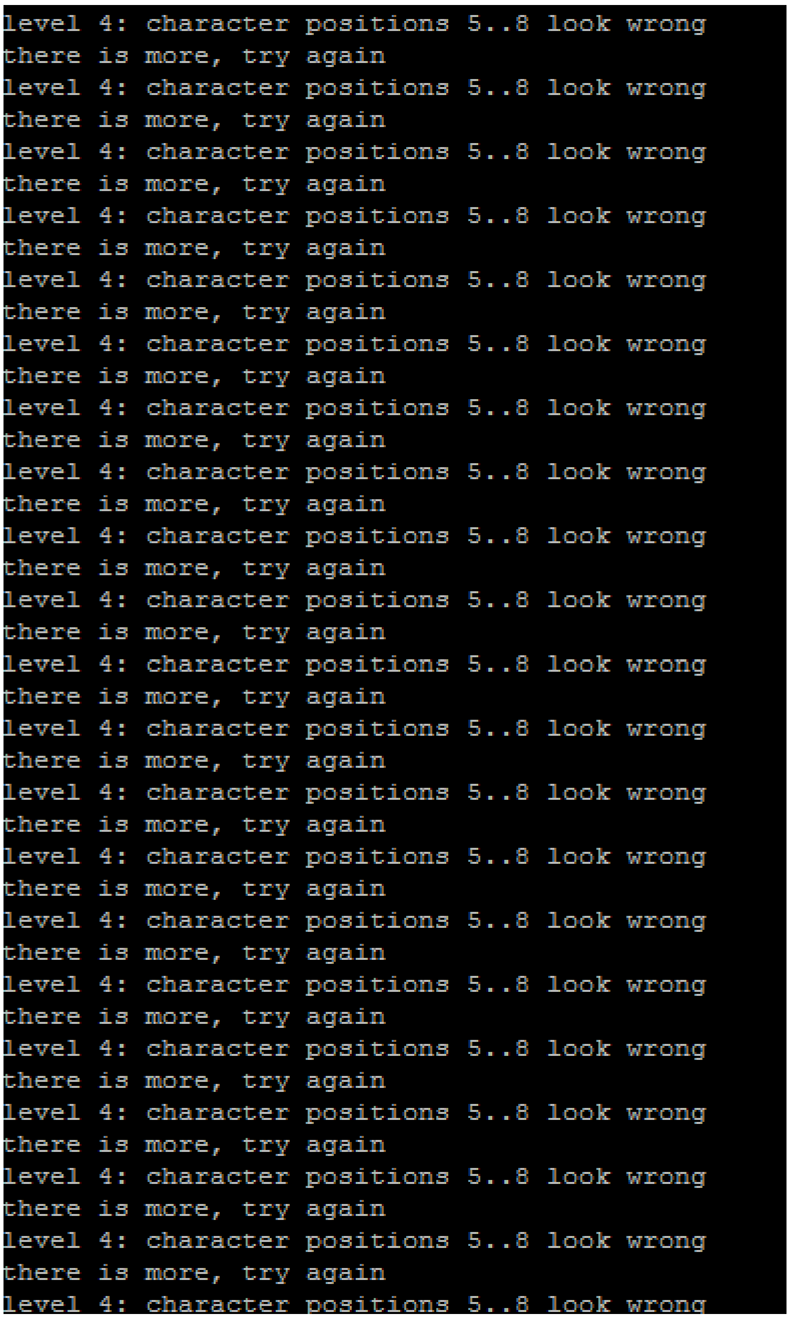
Then I just made a small shell script that goes through all the lines in the file and pushes them into the ./maze program and the output is then saved to a file (Figure 20)



```
#!/bin/bash

input="/root/Downloads/software_exploitation-master/assignments/1/nakki.txt"


while IFS= read -r var
do
  echo -e  $var | ./maze >> leveleil
done < "$input"
```

Figure 20 shell script

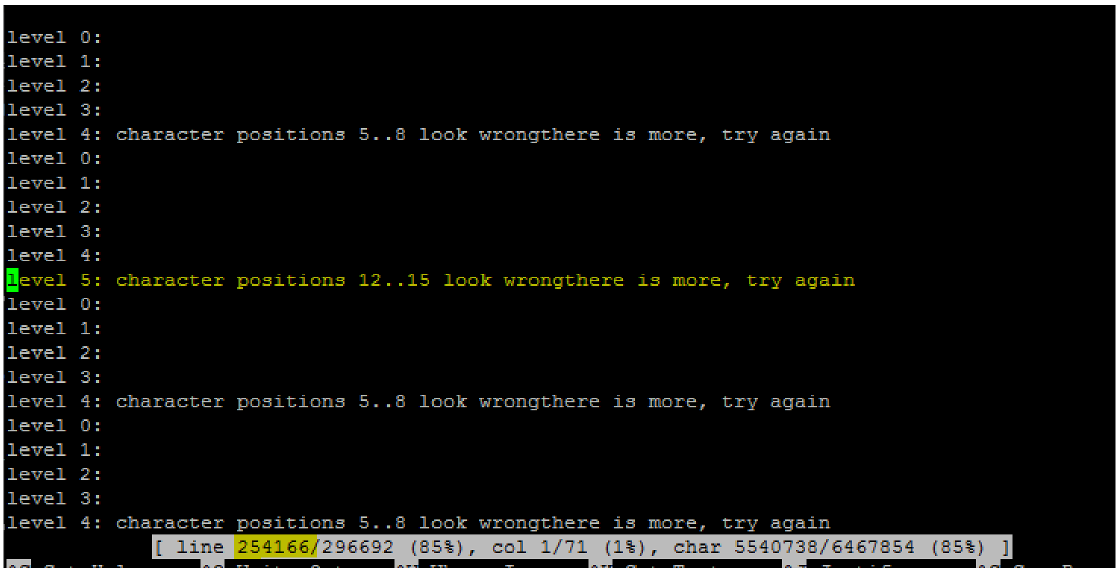
Then I booted up the shell script after giving it +x permission. I started following the file leveleil with tail -f (Figure 21)



```
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
there is more, try again
level 4: character positions 5..8 look wrong
```

Figure 21 tail -f

After a minute or so I opened the leveleil file with nano and searched for level 5: and found it. (Figure 22)

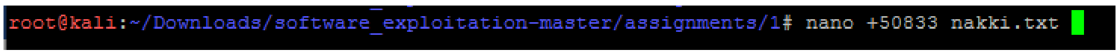


```
level 0:
level 1:
level 2:
level 3:
level 4: character positions 5..8 look wrongthere is more, try again
level 0:
level 1:
level 2:
level 3:
level 4:
level 5: character positions 12..15 look wrongthere is more, try again
level 0:
level 1:
level 2:
level 3:
level 4: character positions 5..8 look wrongthere is more, try again
level 0:
level 1:
level 2:
level 3:
level 4: character positions 5..8 look wrongthere is more, try again
                    [ line 254166/296692 (85%), col 1/71 (1%), char 5540738/6467854 (85%) ]
```

Figure 22 Level 5

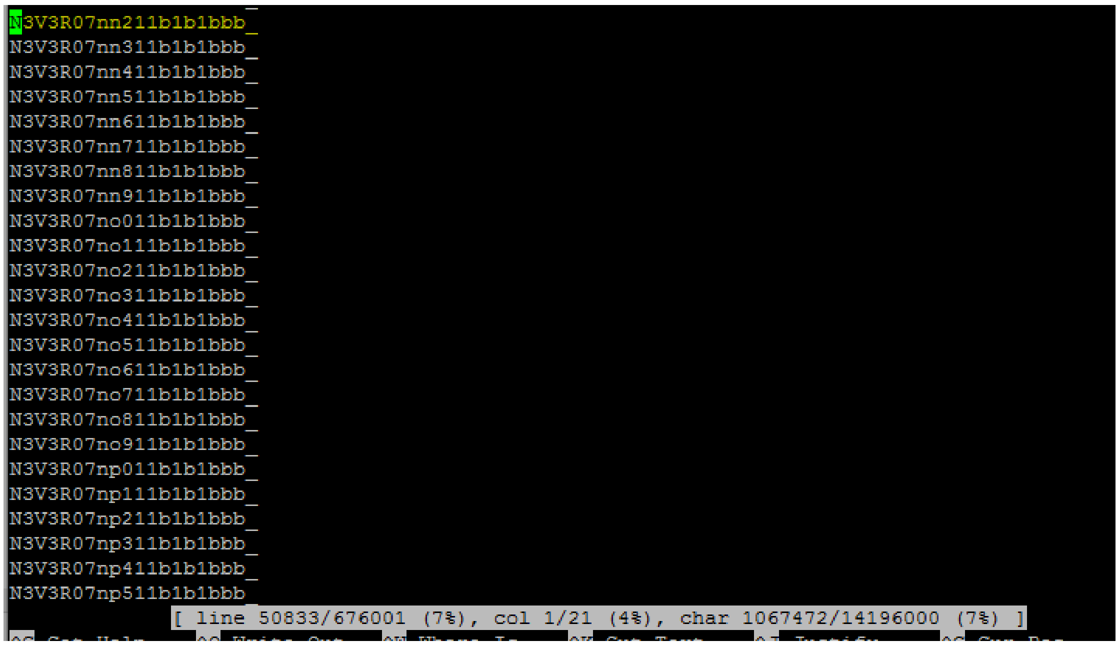
Now only thing to find is the passphrase that was in use by checking out the line where the level0 is and comparing it to the password file.

So the location should be (254166-1)/5 in the position of 50833 in our password file (Figure 24) the line can be accessed via (Figure 23)



```
root@kali:~/Downloads/software_exploitation-master/assignments/1# nano +50833 nakki.txt
```

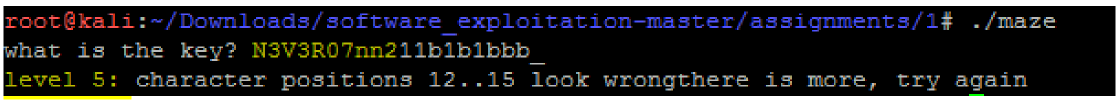
Figure 23 nano



```
N3V3R07nn211b1b1bbb_
N3V3R07nn311b1b1bbb_
N3V3R07nn411b1b1bbb_
N3V3R07nn511b1b1bbb_
N3V3R07nn611b1b1bbb_
N3V3R07nn711b1b1bbb_
N3V3R07nn811b1b1bbb_
N3V3R07nn911b1b1bbb_
N3V3R07no011b1b1bbb_
N3V3R07no111b1b1bbb_
N3V3R07no211b1b1bbb_
N3V3R07no311b1b1bbb_
N3V3R07no411b1b1bbb_
N3V3R07no511b1b1bbb_
N3V3R07no611b1b1bbb_
N3V3R07no711b1b1bbb_
N3V3R07no811b1b1bbb_
N3V3R07no911b1b1bbb_
N3V3R07np011b1b1bbb_
N3V3R07np111b1b1bbb_
N3V3R07np211b1b1bbb_
N3V3R07np311b1b1bbb_
N3V3R07np411b1b1bbb_
N3V3R07np511b1b1bbb
                    [ line 50833/676001 (7%), col 1/21 (4%), char 1067472/14196000 (7%) ]
```

Figure 24 password

and as calculated correctly the location was where it was supposed to be and now I have the level 5 via dirty gaming. ☺ (Figure 25)



```
root@kali:~/Downloads/software_exploitation-master/assignments/1# ./maze
what is the key? N3V3R07nn211b1b1bbb_
level 5: character positions 12..15 look wrongthere is more, try again
```

Figure 25 Level 5

Even though this wasn't the point of the exercise I exploited the software from another point of view when I didn't manage forward.

## 4 Conclusion

This assignment was fun and really nice type of challenge even though I didn't get to the finish goal. It didn't take me long time to get to the level 4, but then the progress stopped because I was missing something. This assignment was very good practice and I learned something new about C. These kind of assignments are great because they make you think about the problem and when you get it feels great. Even if used on a cheap way to bypass as last resort.