Password:

crackme0x04:

1. the sum of every digit equals to 15: 78,555
2. 78eee
3. 7r1,5ee

Q1.

break \**address*

break *linenum*

In break command, the \* indicates the number after it is an address. Without \* mark, the break command will use the number as linenum in the current source file.

Q2.

use the follow command in gdb:

x /s *address*

Q3.

the decimal value of 0x149a is the correct answer, since the format string in scanf is %d.

Q4.

eax = 0xffffcf60

edx = 0x1ec

Q5.

Result is 0x246, stored in 0xffffcf60

Q6.

1st para:[ebp-0x4], the starting address where the input string is stored.

2nd para: [ebp-0xc]=0x52b24, value stored in $ebp-0xc which is 0x52b24

Q7.

Function shift creates a new char array whose value of every digit is the result of every digit of the second parameter subtracting 3, and adds '\0' to the end of the new char array. Then print it in the stdout and return.

Q8.

Q9.

Function check reads every digit of the input string into decimal number and calculate the sum of them. If the sum equals 15, print "Password OK!" and then exit the program in the normal way; If not, print "Passwork Incorrect!" and then return.

The following is the source code:

Q10.

sscanf(input\_str[i],"$d",ebp-0x4);

0x080485b8<+48>: lea eax,[ebp-0x4]

0x080485bb<+51>: mov DWORD PTR [esp+0x8],eax//3rd para:ebp-0x4

0x080485bf<+55>: mov DWORD PTR [esp+0x4],0x804873d//2nd para:"%d"

0x080485c7<+63>: lea eax,[ebp-0xd]

0x080485ca<+66>: mov DWORD PTR[esp],eax//1st para:ebp-0xd,input\_str[i]

0x080485cd<+69>: call 0x80483c8 <sscanf@plt>

Q11.

the number equals to the length of your input string.

Q12.

the max unsigned integer can be stored in eax(0xffffffff), because jae assumes the value is unsigned so an unsigned comparison is performed.

Q13.

Function parell reads the whole input string into one single decimal number. If the number is even, it prints "Password OK!" and exit the program in the normal way; Otherwise, it returns.

Q14.

The argument is the starting address where the input string is stored.

Q15.

In crackme, it uses ebp to accessing passed in argument, for example $ebp+0x8; While in bomblab, it uses esp to accessing passed in argument and it doesn't push the ebp into the stack as well.

Q16.

Compare the input string with the strings stored at esp+ebx\*4+0x14, where ebx is the index and initial value is 0. If they are equal, break the loop and jump to the address <func\_game+68>;Otherwise,increase ebx by 1. If ebx is smaller than 3, go back to the start point of the loop. Otherwise, break the loop and jump to the address <func\_game+73>.

Q17.

Compare two string. If they are equal, return 1;

Q18:

Set the sign flag(SF) if the 32nd bit of $esp+0x18 is set. Jump to 0x804924d if sign flag(SF) is clear.

Q19:

The ebx is used for index in the loop.

Q20:

[base + index\*scale + disp]

So base is esp, index is ebx, scale is 4, offset is 0x14, respectively.