ASSIGNMENT TITLE

SUBJECT NAME: Cryptography and Network Security

SUBJECT CODE: CS6008

MODULE: 01

NAME: R.Aadharsh

REG.NO.: 2019103604

DATE: 02/04/2022

AIM:

To crack the password for a given c code using GDB

TOOLS INVOLVED:

Linux , GDB , vim , gcc

PROBLEM DESCRIPTION:

Given a c code output that requires a password to authenticate. Our goal is to crack the password using GDB

INPUT:

Password (which we cracked)

OUTPUT:

Password Cracked

SCREENSHOT:

```
PS C:\Users\Aadharsh\downloads\Crypto> ./demo
Enter password : password
Password Cracked
```

The Process Explained In Brief

03-04-2022 **Cryptography And Network Security** **2019103604**

R.Aadharsh

1)Finding Passwords in executables using GDB:

C-Code:

```
PS C:\Users\Aadharsh\Downloads\Crypto> cat pass.cpp
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main(int argc , char *argv[]){
if(argc != 2){
printf("Invalid Arguments \n" , argv[0]);
return 0;
}

if(strcmp(argv[1] , "MyNoman") == 0){
printf("Password Cracked!\n");
}
else
printf("Failed to crack\n") ;
return 0;
}
```

Output:

```
PS C:\Users\Aadharsh\Downloads\Crypto> ./a
Invalid Arguments
PS C:\Users\Aadharsh\Downloads\Crypto> ./a MyNoman
Password Cracked!
PS C:\Users\Aadharsh\Downloads\Crypto> ./a Noman
Failed to crack
PS C:\Users\Aadharsh\Downloads\Crypto>
```

Ways to crack the password:

a)Use stings command :

```
PS C:\Users\Aadharsh\Downloads\Crypto> strings a.exe
$)0@
$d0@
=@P@
D$$t
-T1@
5T1@
$(1@
$HP@
$HP@
$HP@
$HP@
$HP@
$HP@
$HP@
$HP@
$HP@
%4a@
%0a@
%,a@
%8a@
%$a@
%<a@
%(a@
% a@
libgcc_s_dw2-1.dll
__register_frame_info
libgcj-13.dll
_Jv_RegisterClasses
__deregister_frame_info
Invalid Arguments
MyNoman
Password Cracked!
Failed to crack
Mingw runtime failure:
  VirtualQuery failed for %d bytes at address %p
  Unknown pseudo relocation protocol version %d.
  Unknown pseudo relocation bit size %d.
DeleteCriticalSection
EnterCriticalSection
ExitProcess
GetLastError
GetModuleHandleA
GetProcAddress
InitializeCriticalSection
LeaveCriticalSection
SetUnhandledExceptionFilter
TlsGetValue
VirtualProtect
VirtualQuery
__getmainargs
__p__environ
__p__fmode
__set_app_type
_cexit
_iob
_onexit
_setmode
abort
atexit
calloc
free
fwrite
memcpy
printf
puts
signal
strcmp
vfprintf
KERNEL32.dll
msvcrt.dll
```

b)Using GDB:

Assume that the coder didn’t expose the password and used the hexadecimal sum as the password

Code:

```
PS C:\Users\Aadharsh\Downloads\Crypto> cat keygen.py
#import random
#import os
s="MyNoman"
num = 0
for x in s:
        num+=ord(x)
print (num)
```

```
PS C:\Users\Aadharsh\Downloads\Crypto> python keygen.py
703
```

C code:

```
PS C:\Users\Aadharsh\Downloads\Crypto> cat pass.cpp
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main(int argc , char *argv[]){
if(argc != 2){
printf("Invalid Arguments \n" , argv[0]);
return 0;
}
int i=0,sum=0;
for(i=0;argv[1][i] != '\0'; i++){
sum+=(int)argv[1][i];
}

if(sum == 703){
printf("Password Cracked!\n");
}
else
printf("Failed to crack\n") ;
return 0;
}
```

O/P:

```
PS C:\Users\Aadharsh\Downloads\Crypto> ./p MyNoman
Password Cracked!
PS C:\Users\Aadharsh\Downloads\Crypto> ./p MyNo
Failed to crack
```

Using Strings wont work here because the password was nowhere exposed by the coder

```
$HP@
$HP@
%0a@
%,a@
%(a@
% a@
%4a@
%$a@
libgcc_s_dw2-1.dll
__register_frame_info
libgcj-13.dll
_Jv_RegisterClasses
__deregister_frame_info
Invalid Arguments
Password Cracked!
Failed to crack
Mingw runtime failure:
  VirtualQuery failed for %d bytes at address %p
  Unknown pseudo relocation protocol version %d.
  Unknown pseudo relocation bit size %d.
DeleteCriticalSection
```

Exploitation:

Disassembling the main:

```
Dump of assembler code for function main:
   0x004013b0 <+0>:     push   %ebp
   0x004013b1 <+1>:     mov    %esp,%ebp
   0x004013b3 <+3>:     and    $0xfffffff0,%esp
   0x004013b6 <+6>:     sub    $0x20,%esp
   0x004013b9 <+9>:     call   0x401a10 <__main>
   0x004013be <+14>:    cmpl   $0x2,0x8(%ebp)
   0x004013c2 <+18>:    je     0x4013e0 <main+48>
   0x004013c4 <+20>:    mov    0xc(%ebp),%eax
   0x004013c7 <+23>:    mov    (%eax),%eax
   0x004013c9 <+25>:    mov    %eax,0x4(%esp)
   0x004013cd <+29>:    movl   $0x403064,(%esp)
   0x004013d4 <+36>:    call   0x401c88 <printf>
   0x004013d9 <+41>:    mov    $0x0,%eax
   0x004013de <+46>:    jmp    0x401457 <main+167>
   0x004013e0 <+48>:    movl   $0x0,0x1c(%esp)
   0x004013e8 <+56>:    movl   $0x0,0x18(%esp)
   0x004013f0 <+64>:    movl   $0x0,0x1c(%esp)
   0x004013f8 <+72>:    jmp    0x401415 <main+101>
   0x004013fa <+74>:    mov    0xc(%ebp),%eax
   0x004013fd <+77>:    add    $0x4,%eax
   0x00401400 <+80>:    mov    (%eax),%edx
   0x00401402 <+82>:    mov    0x1c(%esp),%eax
   0x00401406 <+86>:    add    %edx,%eax
   0x00401408 <+88>:    mov    (%eax),%al
   0x0040140a <+90>:    movsbl %al,%eax
   0x0040140d <+93>:    add    %eax,0x18(%esp)
   0x00401411 <+97>:    incl   0x1c(%esp)
   0x00401415 <+101>:   mov    0xc(%ebp),%eax
   0x00401418 <+104>:   add    $0x4,%eax
   0x0040141b <+107>:   mov    (%eax),%edx
   0x0040141d <+109>:   mov    0x1c(%esp),%eax
   0x00401421 <+113>:   add    %edx,%eax
   0x00401423 <+115>:   mov    (%eax),%al
   0x00401425 <+117>:   test   %al,%al
   0x00401427 <+119>:   setne  %al
   0x0040142a <+122>:   test   %al,%al
   0x0040142c <+124>:   jne    0x4013fa <main+74>
   0x0040142e <+126>:   cmpl   $0x2bf,0x18(%esp)
   0x00401436 <+134>:   jne    0x401446 <main+150>
   0x00401438 <+136>:   movl   $0x403078,(%esp)
   0x0040143f <+143>:   call   0x401c80 <puts>
   0x00401444 <+148>:   jmp    0x401452 <main+162>
   0x00401446 <+150>:   movl   $0x40308a,(%esp)
   0x0040144d <+157>:   call   0x401c80 <puts>
   0x00401452 <+162>:   mov    $0x0,%eax
   0x00401457 <+167>:   leave
   0x00401458 <+168>:   ret
   0x00401459 <+169>:   nop
   0x0040145a <+170>:   nop
   0x0040145b <+171>:   nop
   0x0040145c <+172>:   xchg   %ax,%ax
   0x0040145e <+174>:   xchg   %ax,%ax
End of assembler dump.
```

Searching if there are any comparisions done to validate the password and when found finding the equalent hexadecimal value

```
PS C:\Users\Aadharsh> python
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit
 (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 0x2bf
703
```

Creating a python program to get the different combinations that give a hexadecimal sum of 703

Python Program :

```
PS C:\Users\Aadharsh\Downloads\Crypto> cat keygen.py
import random
import os

def getHexSum(s):
        num = 0
        for x in s:
                num+=ord(x)
        return num

key="";
while True:
        key+=random.choice("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.")
        if getHexSum(key) > 703:
                key=""
        elif getHexSum(key) == 703:
                print(key)
```

O/P:

```
PS C:\Users\Aadharsh\Downloads\Crypto> python keygen.py
q7zLlwn
gaJbGSF74
HZibj3Ta
J0Zd93q0z
cpetbF92
h5M5uJeC9
pSYmfZv
qWshZkW
lLlThnq
BmbDFYTw
wm96EJ67p
xrYaVCI9
nk6JLOaj
HN261gudP
rjIe0ZXS
nbnpjn9
jLZ15HIiO
lePrwqD
vXNIHw0k
MEBhffBu
ynTozj1
eSY4f2rp
ppJiQiA1
r1T51s9uA
9MccakXO
fYwnzSN
AJlFvOvG
Lj6lYE2GP
2k33fzCAX
nkxq4Xq
m1aJ27XtA
JOWuUm5c
D527.KGaC7B
Jx3cede9
```

So from here choosing one string as password should crack the password

```
PS C:\Users\Aadharsh\Downloads\Crypto> ./p pSYmfZv
Password Cracked!
```

Using GDB to view contents from the stack:

Code:

```
PS C:\Users\Aadharsh\downloads\Crypto> cat demo.c
#include<stdio.h>
int main() {
        char buffer[256];
        char pass[] = "password";
        printf("Enter password : ");
        scanf("%s",&buffer);
        if(strcmp(buffer,pass) == 0){
        printf("Password Cracked\n");
        }
        else{
        printf("Wrong Password \n");
        }
        return 0;
}
```

Disassembling Main:

```
(gdb) disas main
Dump of assembler code for function main:
   0x000000000040069d <+0>:     push   %rbp
   0x000000000040069e <+1>:     mov    %rsp,%rbp
   0x00000000004006a1 <+4>:     sub    $0x120,%rsp
   0x00000000004006a8 <+11>:    mov    %fs:0x28,%rax
   0x00000000004006b1 <+20>:    mov    %rax,-0x8(%rbp)
   0x00000000004006b5 <+24>:    xor    %eax,%eax
   0x00000000004006b7 <+26>:    movabs $0x64726f7773736170,%rax
   0x00000000004006c1 <+36>:    mov    %rax,-0x120(%rbp)
   0x00000000004006c8 <+43>:    movb   $0x0,-0x118(%rbp)
   0x00000000004006cf <+50>:    mov    $0x4007c4,%edi
   0x00000000004006d4 <+55>:    mov    $0x0,%eax
   0x00000000004006d9 <+60>:    callq  0x400560 <printf@plt>
   0x00000000004006de <+65>:    lea    -0x110(%rbp),%rax
   0x00000000004006e5 <+72>:    mov    %rax,%rsi
   0x00000000004006e8 <+75>:    mov    $0x4007da,%edi
   0x00000000004006ed <+80>:    mov    $0x0,%eax
   0x00000000004006f2 <+85>:    callq  0x4005a0 <__isoc99_scanf@plt>
   0x00000000004006f7 <+90>:    lea    -0x120(%rbp),%rdx
   0x00000000004006fe <+97>:    lea    -0x110(%rbp),%rax
   0x0000000000400705 <+104>:   mov    %rdx,%rsi
   0x0000000000400708 <+107>:   mov    %rax,%rdi
   0x000000000040070b <+110>:   callq  0x400580 <strcmp@plt>
   0x0000000000400710 <+115>:   test   %eax,%eax
   0x0000000000400712 <+117>:   jne    0x400720 <main+131>
   0x0000000000400714 <+119>:   mov    $0x4007dd,%edi
   0x0000000000400719 <+124>:   callq  0x400540 <puts@plt>
   0x000000000040071e <+129>:   jmp    0x40072a <main+141>
   0x0000000000400720 <+131>:   mov    $0x4007e7,%edi
   0x0000000000400725 <+136>:   callq  0x400540 <puts@plt>
   0x000000000040072a <+141>:   mov    -0x8(%rbp),%rcx
   0x000000000040072e <+145>:   xor    %fs:0x28,%rcx
   0x0000000000400737 <+154>:   je     0x40073e <main+161>
   0x0000000000400739 <+156>:   callq  0x400550 <__stack_chk_fail@plt>
   0x000000000040073e <+161>:   leaveq
   0x000000000040073f <+162>:   retq
End of assembler dump.
```

Here the call is found at main+110 , so breaking it at that point and displaying the register info

```
(gdb) break *main+110
Breakpoint 1 at 0x40070b
(gdb) run
Starting program: /home/vagrant/gdb_test/program
Enter the password : hello

Breakpoint 1, 0x000000000040070b in main ()
(gdb) info register
rax            0x7fffffffe4a0   140737488348320
rbx            0x0      0
rcx            0x0      0
rdx            0x7fffffffe490   140737488348304
rsi            0x7fffffffe490   140737488348304
rdi            0x7fffffffe4a0   140737488348320
rbp            0x7fffffffe5b0   0x7fffffffe5b0
rsp            0x7fffffffe490   0x7fffffffe490
r8             0x0      0
r9             0x0      0
r10            0x7fffffffe4a5   140737488348325
r11            0x246    582
r12            0x4005b0 4195760
r13            0x7fffffffe690   140737488348816
r14            0x0      0
r15            0x0      0
rip            0x40070b 0x40070b <main+110>
eflags         0x206    [ PF IF ]
cs             0x33     51
ss             0x2b     43
ds             0x0      0
es             0x0      0
fs             0x0      0
gs             0x0      0
```

```
(gdb) x/s $rax
0x7fffffffe4a0: "hello"
(gdb) x/s $rdx
0x7fffffffe490: "password"
```

Therefore the Password for the program is displayed

```
PS C:\Users\Aadharsh\downloads\Crypto> ./demo
Enter password : password
Password Cracked
```