

[illegible]

Name	We Love MD5
URL	https://www.attackdefense.com/challengedetails?cid=108
Type	Reverse Engineering : Static Binary Analysis

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic.

Step 1: Check the given file.

Command: ls -l

```
root@attackdefense:~# ls -l
total 968
-rwxr-xr-x 1 root root 988136 Sep 28 23:32 challenge
root@attackdefense:~#
```

Step 2: Execute it. One needs to pass the correct passcode to the binary in order to get the information.

Command: ./challenge

```
student@attackdefense:~$ ls -l
total 968
-rwxr-xr-x 1 root root 988216 Sep 28 23:30 challenge
student@attackdefense:~$
student@attackdefense:~$ ./challenge

Enter password as command line argument

challenge <password>
student@attackdefense:~$
```

Step 3: Open this binary in GDB.

```
root@attackdefense:~# gdb challenge
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
```

Print the names of the global variables used in the binary.

```
(gdb) info variables
All defined variables:

File challenge.c:
char password[33];
```

Print variables

```
(gdb) print password
$1 = "8fe219cf0c7d27adf43ede6fcad19280"
(gdb)
```

Print the list of functions

Command: info functions

```
(gdb) info functions
All defined functions:

File challenge.c:
int main(int, char **);
int print_flag(char *);
char *str2md5(const char *, int);
```

Step 4: Set disassembly flavor to Intel style (It is more user friendly and known then default ATT style).

Command: set disassembly-flavor intel

Disassemble main() function

Command: disassemble main


By looking closely on the disassembled code, it is clear that str2md5() is being called and password is being passed to it. Also, by taking a cue from the name of the lab, may be the final password is MD5 of the value stored in password variable.

```
0x0000000000400d70 <+66>:  mov     esi,0x20
0x0000000000400d75 <+71>:  lea     rdi,[rip+0x2d9384]          # 0x6da100 <password>
0x0000000000400d7c <+78>:  call    0x400b9d <str2md5>
0x0000000000400d81 <+83>:  mov     QWORD PTR [rbp-0x38],rax
0x0000000000400d85 <+87>:  cmp     DWORD PTR [rbp-0x44],0x1
0x0000000000400d89 <+91>:  jg      0x400d9e <main+112>
0x0000000000400d8b <+93>:  lea     rdi,[rip+0xb05b6]          # 0x4b1348
```

Step 5: Calculate MD5 of the password variable (8fe219cf0c7d27adf43ede6fcad19280). This gives b5afea3a5b5ddf57b3de8a4f34fff239

Try this passcode

```
student@attackdefense:~$ ./challenge b5afea3a5b5ddf57b3de8a4f34fff239
Success. Flag: 55d1774ed6d7c8252cd7bc483879d2bc
student@attackdefense:~$
```



Flag: 55d1774ed6d7c8252cd7bc483879d2bc

References:

1. GDB (<https://www.gnu.org/software/gdb/>)