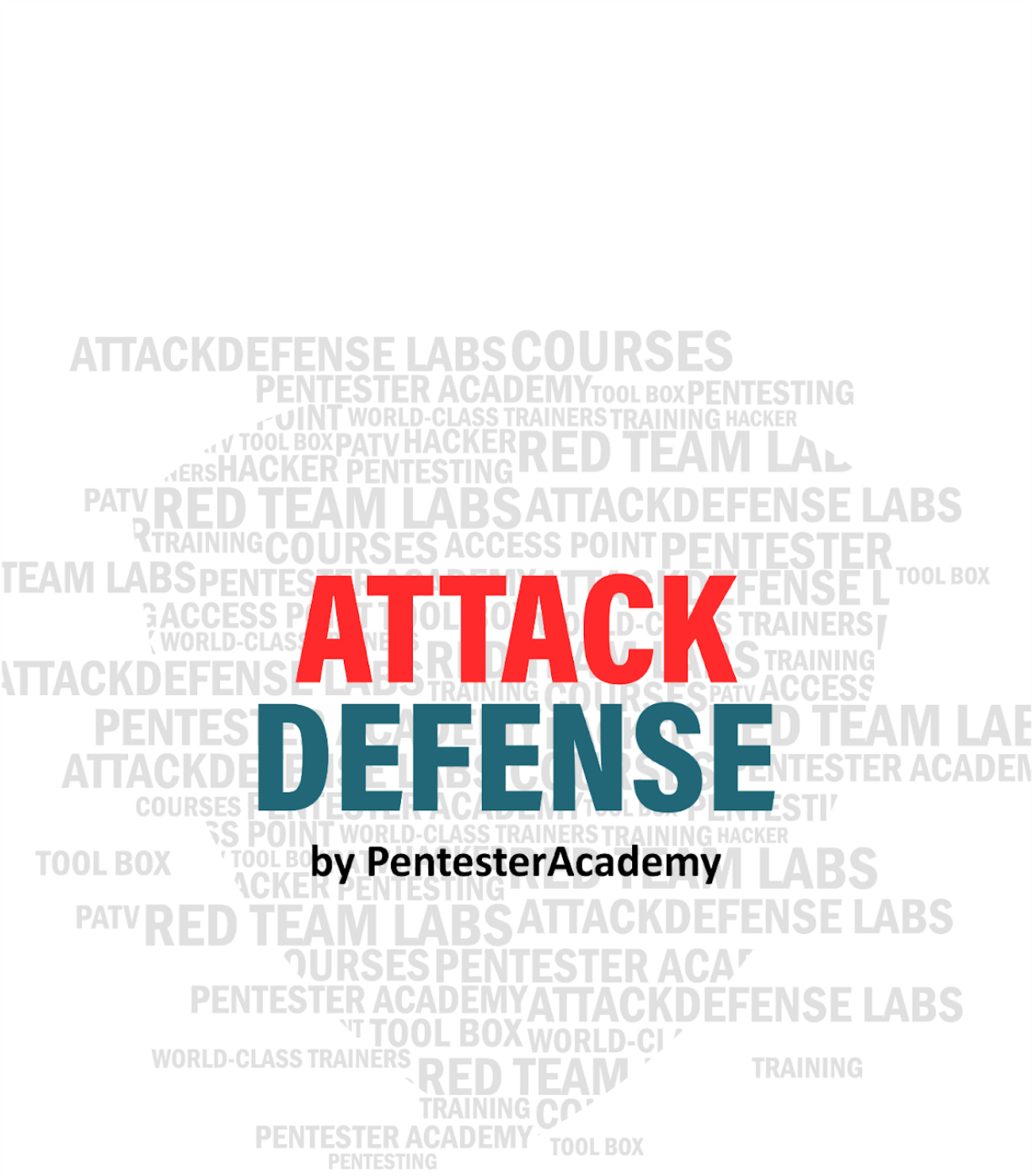
ATTACK
DEFENSE
by PentesterAcademy

Name	Recover Passcode
URL	https://www.attackdefense.com/challengedetails?cid=105
Type	Reserve 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

```
root@attackdefense:~# ./challenge

Enter password as command line argument

i.e. challenge <password>
root@attackdefense:~#
```

Step 3: Open this binary in GDB.

Command: gdb challenge

```
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.
```

Step 4: Print the names of the global variables used in the binary.

Command: info variables

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

File challenge.c:
char correct_password[13];

Non-debugging symbols:
0x0000000000000000  _nl_current_LC_CTYPE
0x0000000000000008  __libc_tsd_LOCALE
0x0000000000000010  _nl_current_LC_MONETARY
0x0000000000000018  _nl_current_LC_NUMERIC
0x0000000000000020  __libc_errno
0x0000000000000020  errno
0x0000000000000028  tcache
0x0000000000000030  tcache_shutting_down
0x0000000000000038  thread_arena
0x0000000000000040    libc_tsd_CTYPE_TOLOWER
```

Variable correct_password, looks interesting. Print it.

Command: print correct_password

```
(gdb) print correct_password
$1 = "hardbuteasy\000"
(gdb)
```

Step 5: Pass the value stored in this variable to challenge binary.

Command: run challenge hardbuteasy

It won't work. Check the assembly code. Especially, look into functions list.

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 6: 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

```
 0x0000000000400d89 <+91>:    add    rax,rdx
 0x0000000000400d8c <+94>:    mov    rcx,QWORD PTR [rax]
 0x0000000000400d8f <+97>:    lea    rax,[rbp-0x15]
 0x0000000000400d93 <+101>:   mov    edx,0xc
 0x0000000000400d98 <+106>:   mov    rsi,rcx
--Type <return> to continue, or q <return> to quit---
 0x0000000000400d9b <+109>:   mov    rdi,rax
 0x0000000000400d9e <+112>:   call   0x4004b0
 0x0000000000400da3 <+117>:   lea    rax,[rbp-0x15]
 0x0000000000400da7 <+121>:   mov    rdi,rax
 0x0000000000400daa <+124>:   call   0x400508
 0x0000000000400daf <+129>:   cmp    rax,0x9
 0x0000000000400db3 <+133>:   ja     0x400de4 <main+182>
 0x0000000000400db5 <+135>:   lea    rax,[rbp-0x15]
 0x0000000000400db9 <+139>:   mov    edx,0x9
 0x0000000000400dbe <+144>:   lea    rsi,[rip+0x2d932b]        # 0x6da0f0 <correct_password>
 0x0000000000400dc5 <+151>:   mov    rdi,rax
 0x0000000000400dc8 <+154>:   call   0x4004a8
 0x0000000000400dcd <+159>:   test   eax,eax
 0x0000000000400dcf <+161>:   jne    0x400de4 <main+182>
 0x0000000000400dd1 <+163>:   lea    rdi,[rip+0xb05b8]        # 0x4b1390
 0x0000000000400dd8 <+170>:   call   0x400cd3 <print_flag>
 0x0000000000400ddd <+175>:   mov    eax,0x0
 0x0000000000400de2 <+180>:   jmp    0x400df5 <main+199>
 0x0000000000400de4 <+182>:   lea    rdi,[rip+0xb05c6]        # 0x4b13b1
 0x0000000000400deb <+189>:   call   0x411b80 <puts>
```

Take a look at this assembly code. It is clear that only the first 9 characters of the correct_password string are being matched. Hence, we only need to provide first 9 letters.

```
(gdb) run challenge hardbutea
Starting program: /home/student/challenge challenge hardbutea
warning: Error disabling address space randomization: Operation not permitte
Success. Flag: 608ba09bfa266fd9992c7faa2b3ecc14
During startup program exited normally.
```

Correct passcode is hardbutea

Flag: 608ba09bfa266fd9992c7faa2b3ecc14

References:

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