Penetration Testing Report

For

“Digest”

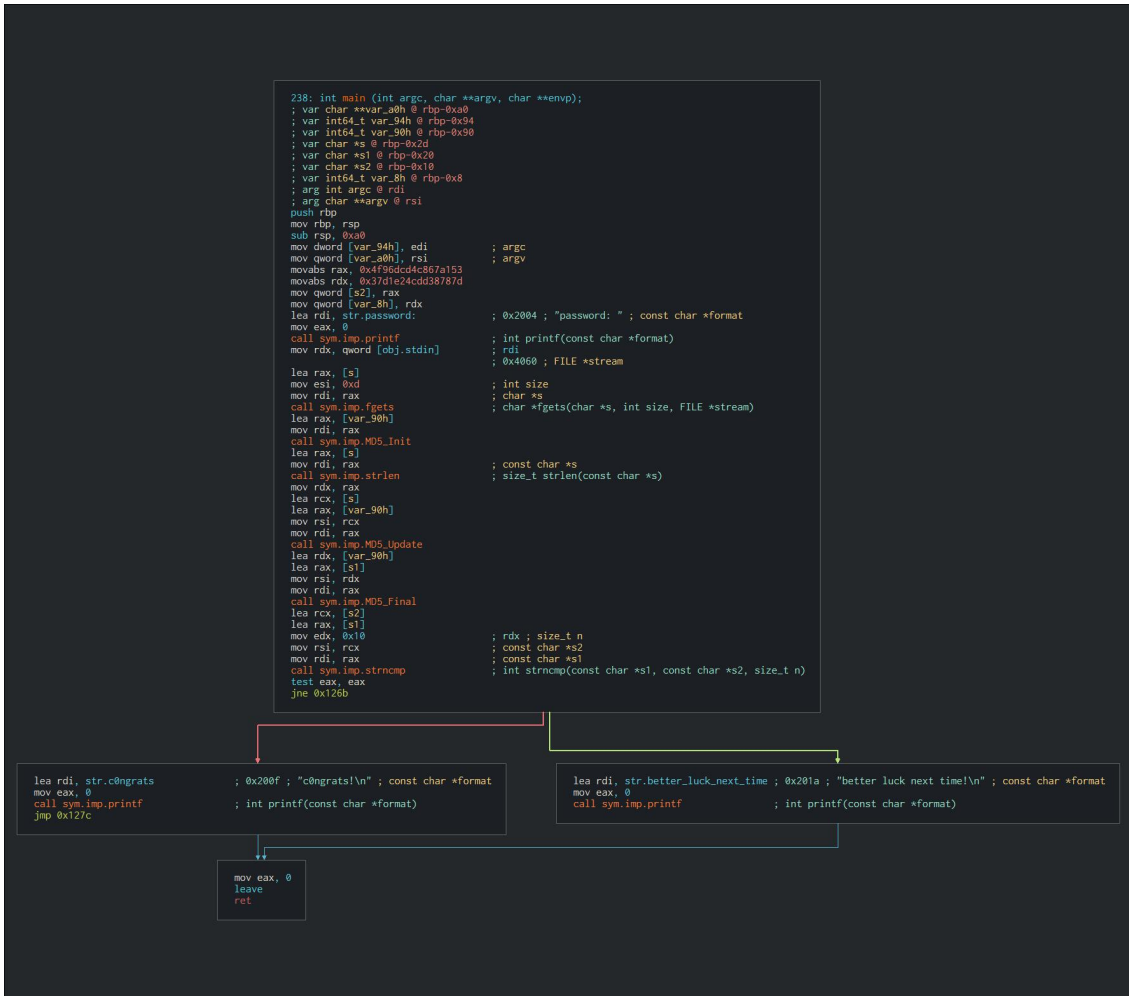
S.NO.	Title	#
1.	Challenge Category	Reverse Engineering
2.	Challenge Related Files	digest
3.	File Link / Target IP	N/A

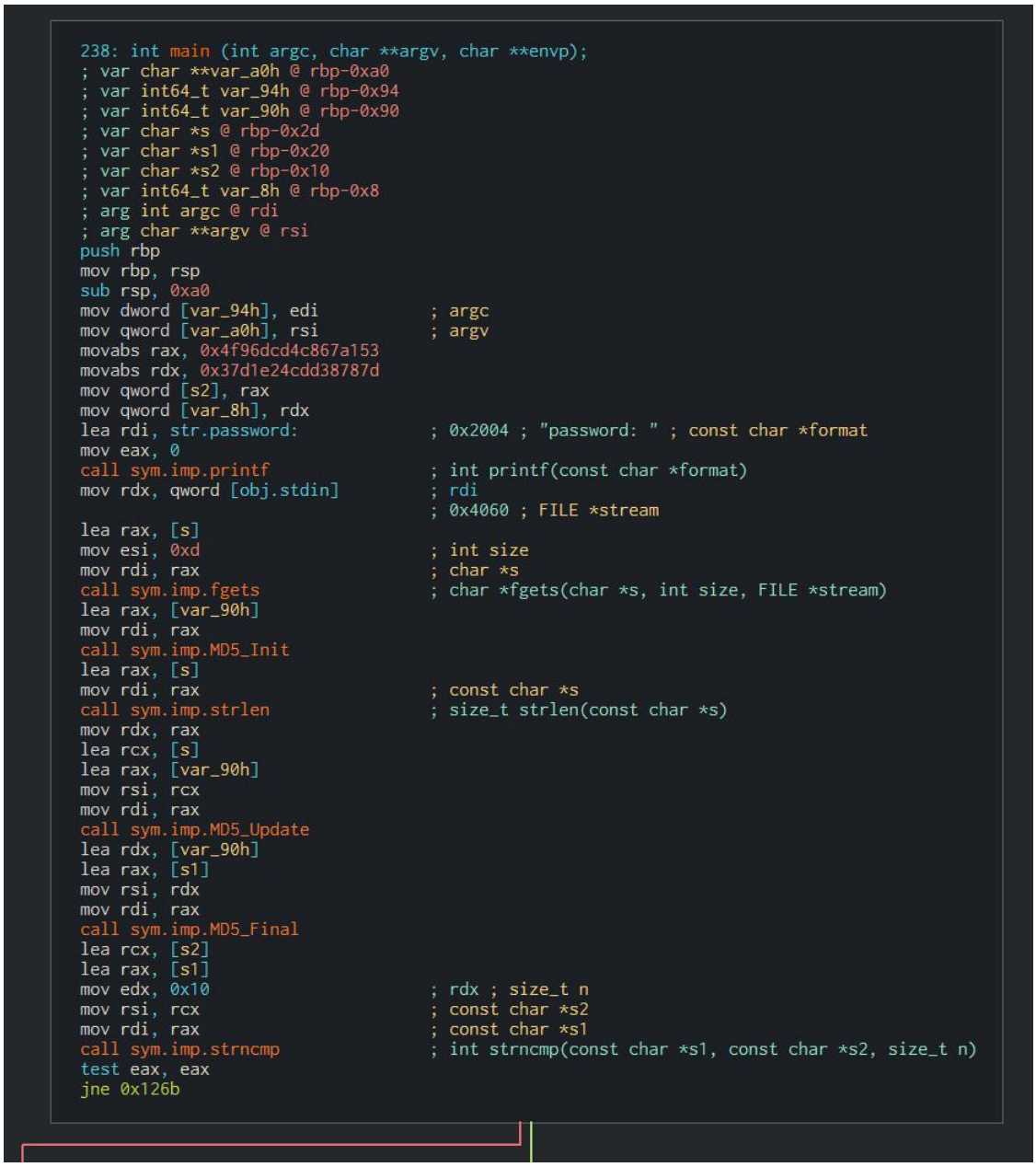
PROCEDURE

1. Analyzing the sub-routines being used & the graph view of the main function.

Functions

Name	Size	Imp.	Offset
entry.fini0	57		0x00001150
entry.init0	5		0x00001190
entry0	42		0x000010b0
main	238		0x00001195
sym.__libc_csu_fini	1		0x000012f0
sym.__libc_csu_init	93		0x00001290
sym._fini	9		0x000012f4
sym._init	23		0x00001000
sym.deregister_tm_clones	41		0x000010e0
sym.imp.MD5_Final	6		0x00001070
sym.imp.MD5_Init	6		0x00001090
sym.imp.MD5_Update	6		0x00001080
sym.imp.fgets	6		0x00001050
sym.imp.printf	6		0x00001030
sym.imp.strlen	6		0x00001060
sym.imp.strncmp	6		0x00001040
sym.register_tm_clones	57		0x00001110

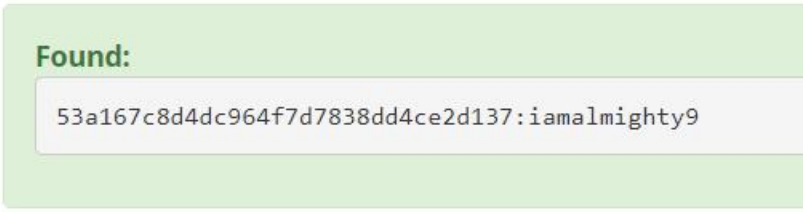




```
238: int main (int argc, char **argv, char **envp);
; var char **var_a0h @ rbp-0xa0
; var int64_t var_94h @ rbp-0x94
; var int64_t var_90h @ rbp-0x90
; var char *s @ rbp-0x2d
; var char *s1 @ rbp-0x20
; var char *s2 @ rbp-0x10
; var int64_t var_8h @ rbp-0x8
; arg int argc @ rdi
; arg char **argv @ rsi
push rbp
mov rbp, rsp
sub rsp, 0xa0
mov dword [var_94h], edi          ; argc
mov qword [var_a0h], rsi          ; argv
movabs rax, 0x4f96dcd4c867a153
movabs rdx, 0x37d1e24cdd38787d
mov qword [s2], rax
mov qword [var_8h], rdx
lea rdi, str.password:            ; 0x2004 ; "password: " ; const char *format
mov eax, 0
call sym.imp.printf               ; int printf(const char *format)
mov rdx, qword [obj.stdin]        ; rdi
                                  ; 0x4060 ; FILE *stream
lea rax, [s]
mov esi, 0xd                      ; int size
mov rdi, rax                      ; char *s
call sym.imp.fgets                ; char *fgets(char *s, int size, FILE *stream)
lea rax, [var_90h]
mov rdi, rax
call sym.imp.MD5_Init
lea rax, [s]
mov rdi, rax                      ; const char *s
call sym.imp.strlen               ; size_t strlen(const char *s)
mov rdx, rax
lea rcx, [s]
lea rax, [var_90h]
mov rsi, rcx
mov rdi, rax
call sym.imp.MD5_Update
lea rdx, [var_90h]
lea rax, [s1]
mov rsi, rdx
mov rdi, rax
call sym.imp.MD5_Final
lea rcx, [s2]
lea rax, [s1]
mov edx, 0x10                     ; rdx ; size_t n
mov rsi, rcx                      ; const char *s2
mov rdi, rax                      ; const char *s1
call sym.imp.strncmp              ; int strncmp(const char *s1, const char *s2, size_t n)
test eax, eax
jne 0x126b
```

In the beginning, we see that 2 hex values of 8 bytes each are being stored in RAX & RDX in little-endian format. Then the imports suggests that a MD5 hash is being calculated against the user input & being compared to the combined 8 bytes values stored in RAX & RDX. The only thing that needs to be done is bruteforcing the hash that is already stored. The hash is - 53a167c8d4dc964f7d7838dd4ce2d137.

2. Searching the hash in Hashes.org database.



Flags:

S.No.	Flag - No.	Flag
1.	Flag 1	HE{iamalmighty9}