



**Name:** Try me!

**Category:** Engenharia Reversa

**File:** tryme

**Message:**

Há apenas um número que desbloqueia o chall, tente encontrá-lo!

---

### Resolução:

Neste desafio, baixamos um arquivo **tryme** que não possui extensão;

Um dos primeiros passos, especialmente em desafios de Engenharia Reversa é usar o comando *file* para saber o tipo de arquivo que estamos lidando:

```
shellt3r@lhost:~/Downloads$ file tryme
tryme: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter
/lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=7e3cb12f0fde4eb74985d535599249
47313451b0, not stripped
```

Image 1 - Comando FILE no Linux

Aqui, sabemos que o arquivo é um ELF 64 bits, ou seja, um arquivo executável.

Então, vamos executá-lo:

(Lembrando que precisamos alterar o permissionamento do arquivo para executável antes)

```
chmod +x tryme
```

```
shellt3r@lhost:~/Downloads$ chmod +x tryme
shellt3r@lhost:~/Downloads$ ./tryme
Usage: ./tryme [number]
```

Image 2 - Permissionamento e execução de arquivo ELF

O arquivo nos pede para digitar um número, como todo e bom velho Geek, colocamos o número 42:

```
shellt3r@lhost:~/Downloads$ ./tryme 42
Incorrect. Try me again!
```

Image 3 - Execução arquivo ELF

Aparentemente, temos que que “advinhar” o número em questão;

Temos duas formas de resolver esse desafio:

- Engenharia Reversa
- Programming

---

Na primeira forma usamos o segundo comando que sempre usamos para realizar Engenharia Reversa (especialmente em arquivos): *strings*

```
shellt3r@lhost:~/Downloads$ strings tryme
/lib64/ld-linux-x86-64.so.2
IG14Q
libc.so.6
puts
atoi
__cxa_finalize
__libc_start_main
_ITM_deregisterTMCloneTable
__gmon_start__
_Jv_RegisterClasses
_ITM_registerTMCloneTable
GLIBC_2.2.5
=g
AWAVA
AUATL
[]A\A]A^A_
Usage: ./tryme [number]
Incorrect. Try me again!
Success: 254e5f2c3beb1a3d03f17253c15c07f3
```

```
.*3$"  
GCC: (Debian 6.3.0-18+deb9u1) 6.3.0 20170516  
crtstuff.c  
__JCR_LIST__  
deregister_tm_clones  
__do_global_dtors_aux  
completed.6972  
__do_global_dtors_aux_fini_array_entry  
frame_dummy  
__frame_dummy_init_array_entry  
tryme.c  
__FRAME_END__  
__JCR_END__  
__init_array_end  
_DYNAMIC  
__init_array_start  
__GNU_EH_FRAME_HDR  
_GLOBAL_OFFSET_TABLE_  
__libc_csu_fini  
_ITM_deregisterTMCloneTable  
puts@@GLIBC_2.2.5  
_edata  
__libc_start_main@@GLIBC_2.2.5  
__data_start  
__gmon_start__  
__dso_handle  
__IO_stdin_used  
__libc_csu_init  
__bss_start  
main  
_Jv_RegisterClasses  
atoi@@GLIBC_2.2.5  
__TMC_END__  
_ITM_registerTMCloneTable  
__cxa_finalize@@GLIBC_2.2.5  
.symtab  
.strtab  
.shstrtab  
.interp  
.note.ABI-tag  
.note.gnu.build-id  
.gnu.hash  
.dynsym
```

```
.dynstr
.gnu.version
.gnu.version_r
.rela.dyn
.rela.plt
.init
.plt.got
.text
.fini
.rodata
.eh_frame_hdr
.eh_frame
.init_array
.fini_array
.jcr
.dynamic
.got.plt
.data
.bss
.comment
```

Se notarmos, o comando *strings* nos retorna dados legíveis por humanos. Isso porque o código compilado não está obfuscado.

Se lermos atentamente cada linha, notaremos que existem 3 linhas com o output:

```
shellt3r@localhost:~/Downloads$ strings tryme
/lib64/ld-linux-x86-64.so.2
IG14Q
libc.so.6
puts
atoi
__cxa_finalize
__libc_start_main
_ITM_deregisterTMCloneTable
_gmon_start_
_Jv_RegisterClasses
_ITM_registerTMCloneTable
GLIBC_2.2.5
=g
AWAVA
AUATL
[[A\A1AA
Usage: ./tryme [number]
Incorrect. Try me again!
Success: 254e5f2c3beb1a3d03f17253c15c07f3
;*3$"
GCC: (Debian 6.3.0-18+deb9u1) 6.3.0 20170516
crtstuff.c
__JCR_LIST__
deregister_tm_clones
__do_global_dtors_aux
completed.6972
```

Image 4 - Output para o comando strings

E aí está nossa flag: *254e5f2c3beb1a3d03f17253c15c07f3*

Acontece que ao tentarmos submeter na plataforma, obtemos que a resposta está incorreta;

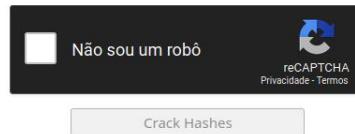
Aí entra a sacada do desafio: a flag, neste caso, é um hash (estudando um pouco sobre tipos de hashes conseguimos perceber que é um **MD5**);

Basta quebrarmos a hash então, um dos sites disponíveis para isso é o [Crack Station](#)

## Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
254e5f2c3beb1a3d03f17253c15c07f3
```



Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1\_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
254e5f2c3beb1a3d03f17253c15c07f3	md5	hacktheplanet

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Image 5 - CrackStation website

E agora, é só submetermos o resultado da Hash.

**Flag(*hacktheplanet*)**

### Pesquisas:

[encurtador.com.br/gosxU](http://encurtador.com.br/gosxU)

[encurtador.com.br/jqzSV](http://encurtador.com.br/jqzSV)