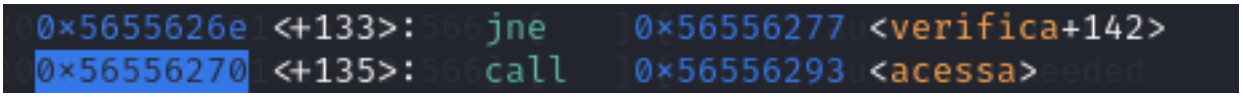
Truques no Debugger

(OBVIAMENTE, COMEÇAMOS COM O RUN NO DEBUGGER DO GDB)
+ Setamos um breakpoint na main, que é a função principal

```
break main
```

→ O princípio é alterar de uma vez o fluxo natural do programa e apontá-lo para a acessa




```
(gdb) i r
eax            0x565562d3       1448436435
ecx            0xffffd370       -11408
edx            0xffffd390       -11376
ebx            0xf7e1dff4       -136192012
esp            0xffffd350       0xffffd350
ebp            0xffffd358       0xffffd358
esi            0x56556330       1448436528
edi            0xf7ffcba0       -134231136
eip            0x565562e2       0x565562e2 <main+15>
eflags         0x286            [ PF SF IF ]
cs             0x23             35
ss             0x2b             43
ds             0x2b             43
es             0x2b             43
fs             0x0              0
gs             0x63             99
```

→ Acima está destacado o endereço normal de eip

```
set $eip = 0x56556270
```

[tbm poderia ser set $pc = ...]

```
(gdb) set $eip = 0x56556270
(gdb) i r
eax            0x565562d3       1448436435
ecx            0xffffd370       -11408
edx            0xffffd390       -11376
ebx            0xf7e1dff4       -136192012
esp            0xffffd350       0xffffd350
ebp            0xffffd358       0xffffd358
esi            0x56556330       1448436528
edi            0xf7ffcba0       -134231136
eip            0x56556270       0x56556270 <verifica+135>
eflags         0x286            [ PF SF IF ]
cs             0x23             35
ss             0x2b             43
ds             0x2b             43
es             0x2b             43
fs             0x0              0
gs             0x63             99
```

→ O endereço setado agora aponta direto para a função que desejamos acessar

→ Para executar com sucesso, basta agora dar o continue:

```
c
```

+ O objetivo agora é descobrir qual a senha que o programa compara
→ Para deletar os breakpoints anteriores:

```
d
```

```
disas verifica
```

→ Vamos por nosso breakpoint no nosso último JNE (Jump Not Equal) pra que possamos ver a execução do programa no momento da comparação para ver com o que ele está comparando

```
   0x5655626c <+131>:   test   %eax,%eax
--Type <RET> for more, q to quit, c to continue without paging--
   0x5655626e <+133>:   jne    0x56556277 <verifica+142>
   0x56556270 <+135>:   call   0x56556293 <acessa>
   0x56556275 <+140>:   jmp    0x56556289 <verifica+160>
```

```
b* 5655626e
```

```
run
```

<passamos uma senha aleatória>
→ Vamos examinar a memória do registrador EIP no formato de string

```
x/20s $eip
```

```
0x56557000 <_fp_hw>:    "\003"
0x56557002 <_fp_hw+2>:  ""
0x56557003 <_fp_hw+3>:  ""
0x56557004 <_IO_stdin_used>:    "\001"
0x56557006 <_IO_stdin_used+2>:  "\002"
0x56557008:     "Entre com a senha: "
0x5655701c:     "123"
0x56557020:     "Essa senha foi desativada!"
0x5655703b:     "pr0t3g1d0"
0x56557045:     "Acesso Negado"
(gdb)
0x56557053:     "Bem vindo! "
0x5655705f:     "id; sh"
0x56557066:     "Sem argumentos! "
0x56557077:     ""
0x56557078:     "\001\033\003;P"
0x5655707e:     ""
0x5655707f:     ""
0x56557080:     "\t"
0x56557082:     ""
0x56557083:     ""
0x56557084:     "\250\357\377\377\230"
```

→ A senha é pr0t3g1d0