

## SHY

We are given a binary that asks for a passcode

```
(docx@kali)-[~/shy]...  
$ ./shy  
Enter the secret passcode to unlock: 1234  
Wrong passcode.
```

and since we cant find it we try gdb on the binary

```
(docx@kali)-[~/shy]  
$ gdb shy  
GNU gdb (Debian 16.3-5) 16.3  
Copyright (C) 2024 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.  
For bug reporting instructions, please see:  
<https://www.gnu.org/software/gdb/bugs/>.  
Find the GDB manual and other documentation resources online at:  
  <http://www.gnu.org/software/gdb/documentation/>.  
  
For help, type "help".  
Type "apropos word" to search for commands related to "word" ...  
Reading symbols from shy ...  
(No debugging symbols found in shy)  
(gdb) run  
Starting program: /home/docx/shy/shy  
[Thread debugging using libthread_db enabled]  
Using host libthread_db library "/usr/lib/x86_64-linux-gnu/libthread_db.so.1".  
Debugger detected. Terminating.  
[Inferior 1 (process 36081) exited with code 01]  
(gdb) █
```

which shows that it has detected a Debugger and it Terminates automatically

so we have to try ghidra on it

import the file and do the default analysis

below the success string we find a function which checks for the flag

```

LAB_00101201
00101201 8b 45 ec    MOV     EAX,dword ptr [RBP + local_1c]
00101204 3b 45 d4    CMP     EAX,dword ptr [RBP + local_34]
00101207 7c 9f      JL      LAB_001011a8
00101209 48 8b 45 d8 MOV     RAX,qword ptr [RBP + local_30]
0010120d 48 8d 15    LEA     RDX,[s_Success!_Here_is_your_flag:_%s_00102020] = "Success!
0c 0e 00 00
00101214 48 89 c6    MOV     RSI,RAX
00101217 48 89 d7    MOV     RDI=>s_Success!_Here_is_your_flag:_%s_00102020... = "Success!
0010121a b8 00 00    MOV     EAX,0x0
00 00
0010121f e8 2c fe    CALL    <EXTERNAL>::printf                                int printf(
ff ff
00101224 90          NOP
00101225 48 83 c4 30 ADD     RSP,0x30
00101229 5b          POP     RBX
0010122a 41 5c      POP     R12
0010122c 5d          POP     RBP
0010122d c3          RET

```

```

*****
*                               *
*                               *
*****
undefined FUN_0010122e()
undefined  ⚠ <UNASSIGNED> <RETURN>
undefined4  Stack[-0xc]:4 local_c                                XREF[3]: 00101:
00101:
00101:
undefined8  Stack[-0x20]:8 local_20                             XREF[1,1]: 00101:
00101:
undefined8  Stack[-0x28]:8 local_28                             XREF[2]: 00101:
00101:
undefined4  Stack[-0x2f]:4 local_2f                             XREF[2,1]: 00101:
00101:
00101:

```

we find an statement which checks the value entered to a hex value 0x7a69

```

}
local_c = 0;
printf("Enter the secret passcode to unlock: ");
iVar1 = __isoc23_scanf(&DAT_001020ae,&local_c);
if (iVar1 == 1) {
    if (local_c == 0x7a69) {

```

so we decode that hex value to decimal

```

(docx@kali)-[~/shy]
$ python
Python 3.13.11 (main, Dec 8 2025, 11:43:54) [GCC 15.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more
>>> 0x7a69
31337
>>>

```

which gives the number 31337 which gets us the flag

```

(docx@kali)-[~/shy]
$ ./shy
Enter the secret passcode to unlock: 31337
Success! Here is your flag: SGCTF{y0u_bypa55ed_m3}

```

SGCTF{y0u\_bypa55ed\_m3}