ATTACK DEFENSE

by PentesterAcademy

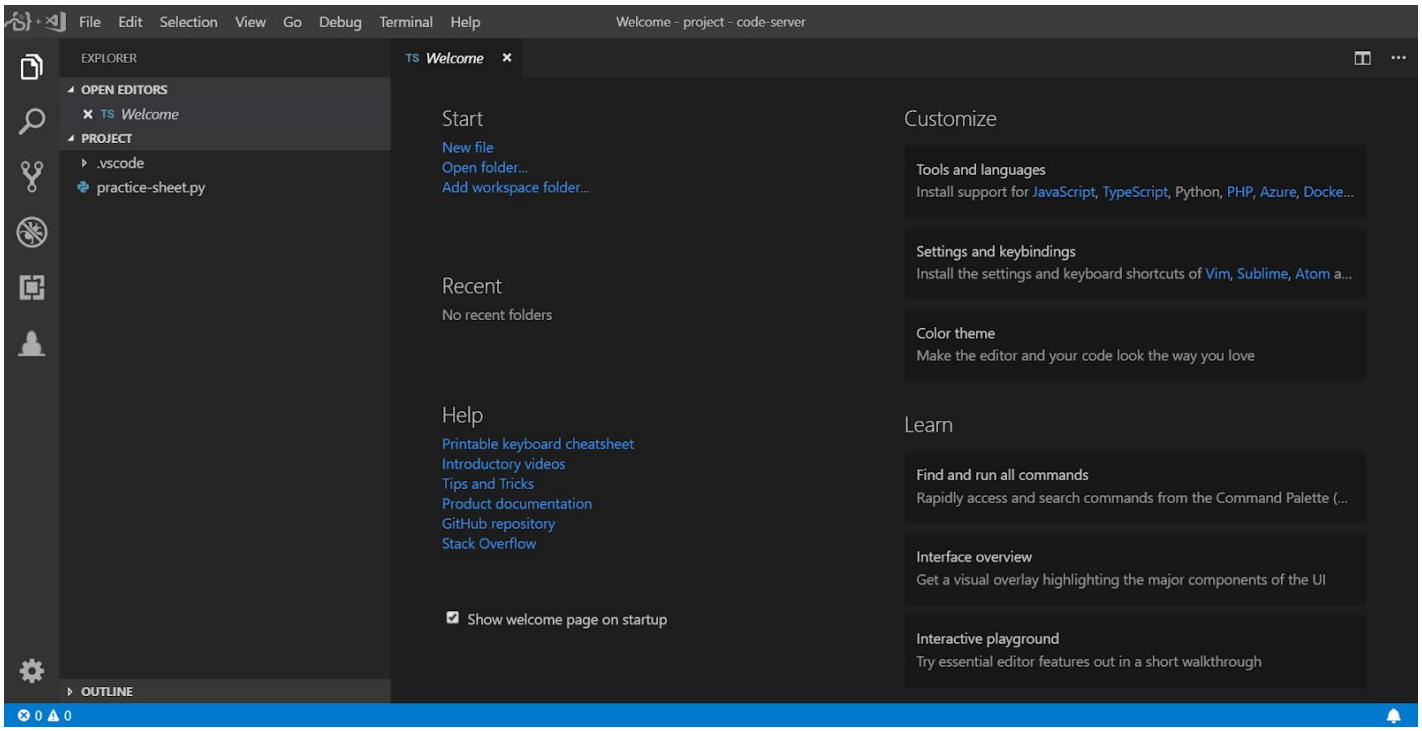
Name	Automating GDB with Python I
URL	https://attackdefense.com/challengedetails?cid=1210
Type	Offensive Python : Debugging

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. Also, note that the IP addresses and domain names might be different in your lab.

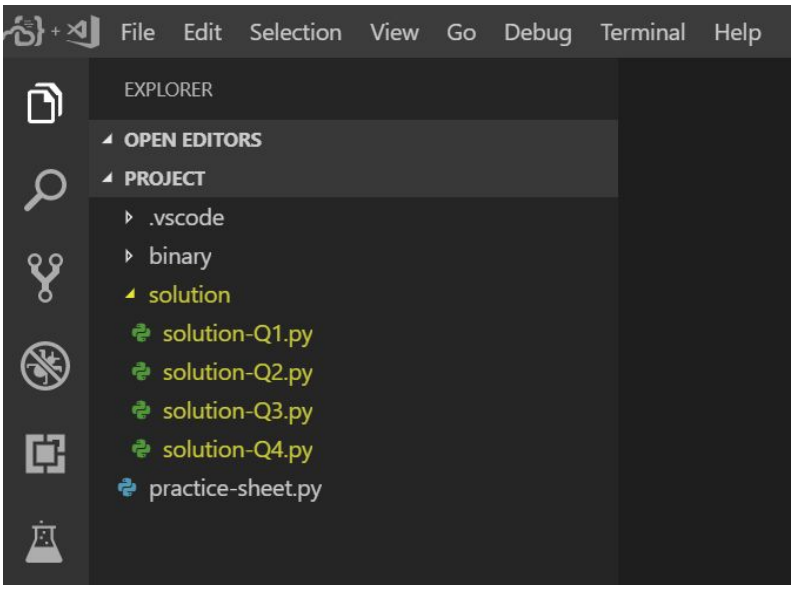
Objective: Use Python pygdbmi library to find all possible values of the exit status codes for all permitted inputs.

Solution:

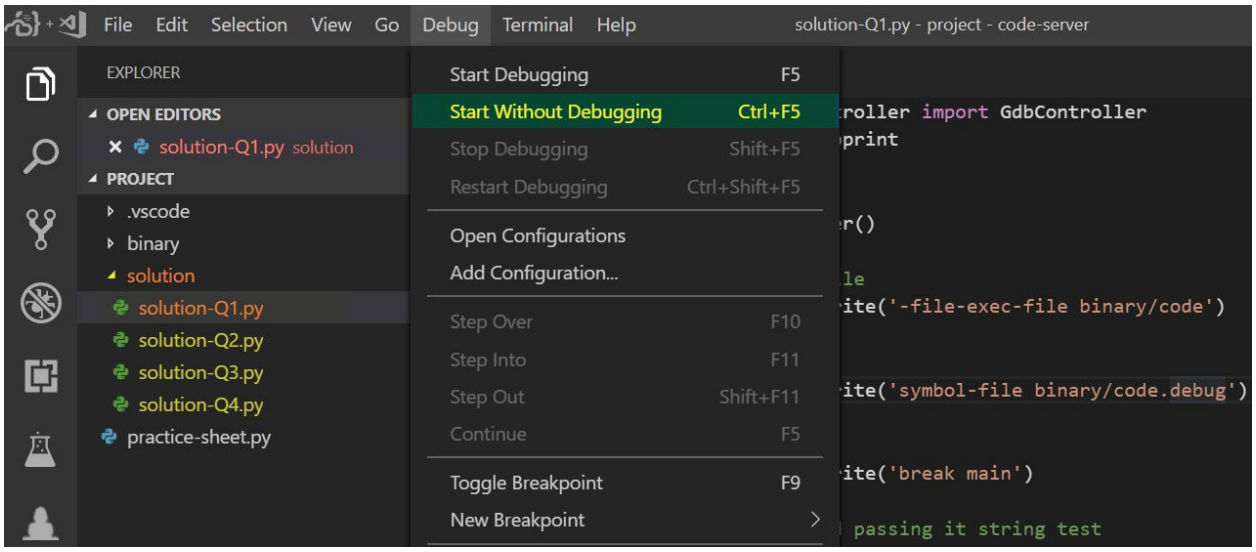
Landing Page:



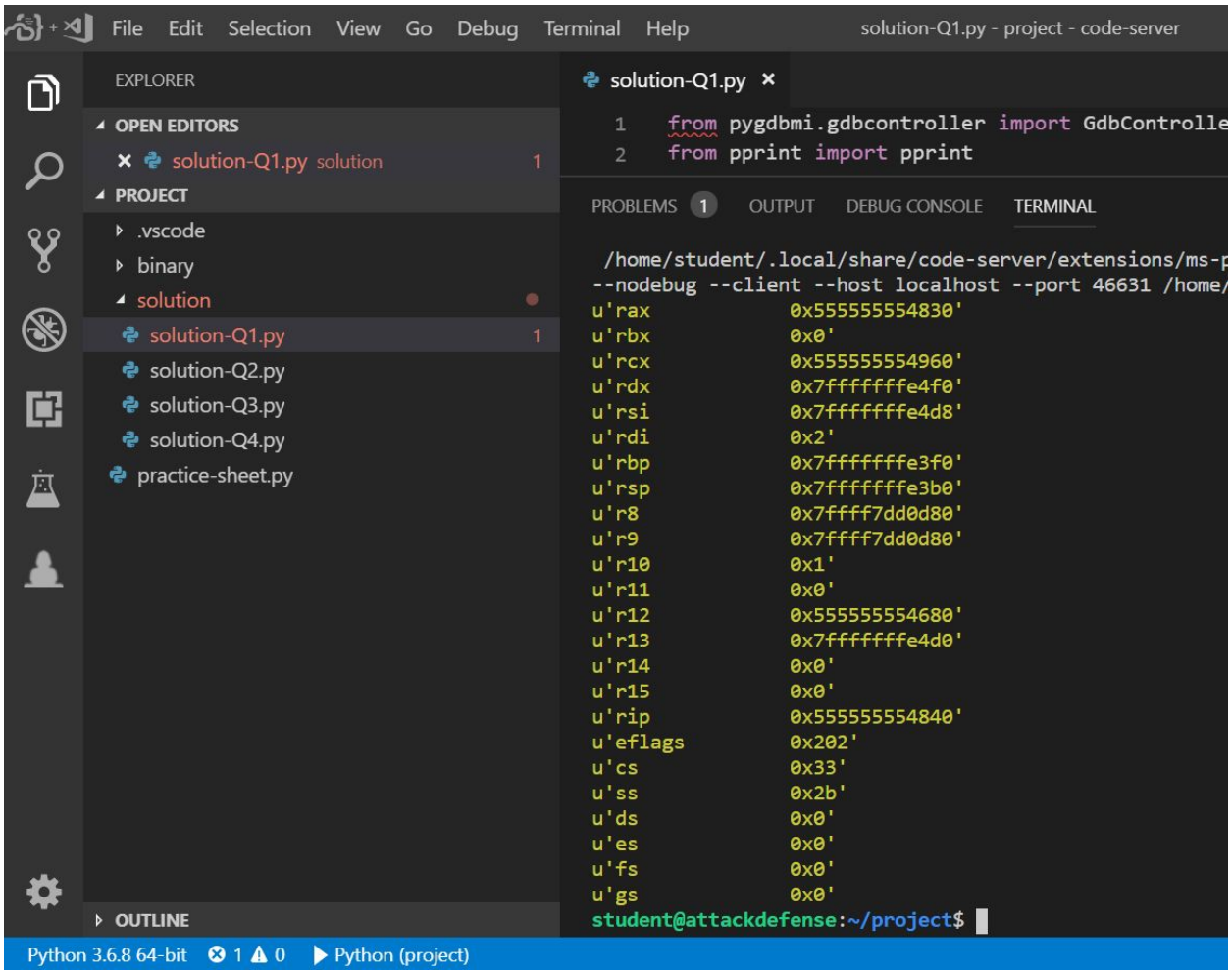
Step 1: Solutions code for all four tasks is given under the solution directory listed in Project Explorer.



Step 2: Open solution-Q1.py. Navigate to Debug Menu and click on “Start Without Debugging option”.



The python script will be executed and the output will be displayed on the integrated terminal. In this case, the output is the list of CPU registers with current values.



Step 3: SImilarly, open solution code for other tasks and run the code to see the output.

a. **solution-Q2.py:** Assembly code for main function

```
from pygdbmi.gdbcontroller import GdbController
from pprint import pprint

# Start gdb process
gdbmi = GdbController()
```

```
student@attackdefense:~/project$ cd /home/student/project ; env
ocal/share/code-server/extensions/ms-python.python-2019.6.24221/
t --port 33117 /home/student/project/solution/solution-Q2.py
'disass mai'
'Dump of assembler code for function main:'
'   0x0000000000000830 <+0>: push   %rbp'
'   0x0000000000000831 <+1>: mov    %rsp,%rbp'
'   0x0000000000000834 <+4>: push   %rbx'
'   0x0000000000000835 <+5>: sub    $0x38,%rsp'
'   0x0000000000000839 <+9>: mov    %edi,-0x34(%rbp)'
'   0x000000000000083c <+12>: mov    %rsi,-0x40(%rbp)'
'   0x0000000000000840 <+16>: mov    %fs:0x28,%rax'
'   0x0000000000000849 <+25>: mov    %rax,-0x18(%rbp)'
'   0x000000000000084d <+29>: xor    %eax,%eax'
'   0x000000000000084f <+31>: mov    %rsp,%rax'
'   0x0000000000000852 <+34>: mov    %rax,%rbx'
'   0x0000000000000855 <+37>: movl   $0x1e,-0x30(%rbp)'
'   0x000000000000085c <+44>: movl   $0x2,-0x2c(%rbp)'
'   0x0000000000000863 <+51>: mov    -0x30(%rbp),%eax'
'   0x0000000000000866 <+54>: add    $0x1,%eax'
'   0x0000000000000869 <+57>: movslq %eax,%rdx'
'   0x000000000000086c <+60>: sub    $0x1,%rdx'
'   0x0000000000000870 <+64>: mov    %rdx,-0x28(%rbp)'
'   0x0000000000000874 <+68>: movslq %eax,%rdx'
```

b. solution-Q3.py: List of all local variables

```
from pygdbmi.gdbcontroller import GdbController
from pprint import pprint

# Print passed responses
def print_console_resp(responses):
```

```
student@attackdefense:~/project$ cd /home/student/project ; env PYTHONIOENCO
ocal/share/code-server/extensions/ms-python.python-2019.6.24221/pythonFiles/
t --port 45001 /home/student/project/solution/solution-Q3.py
"All collected functions: ['get_rand_iteration', 'main', 'random_num_gen']"
'Adding breakpoints on all functions'
'Adding break at function : get_rand_iteration'
'Breakpoint 1 at 0x792: file /home/student/project/binary/code.c, line 5.'
'Adding break at function : main'
'Breakpoint 2 at 0x840: file /home/student/project/binary/code.c, line 22.'
'Adding break at function : random_num_gen'
'Breakpoint 3 at 0x7dc: file /home/student/project/binary/code.c, line 11.'
['max_len = -136423008',
 'args_num = 32767',
 'argument_string = \\"A\\"',
 'lower_num = 0',
 'high_num = 26',
 'iteration = 0',
 'random_num = 0',
 'itr = 0']
student@attackdefense:~/project$
```

c. **solution-Q4.py:** List of register state on each instruction

```
from pygdbmi.gdbcontroller import GdbController
from pprint import pprint

# Returns the number of console responses
def response_line_count(responses):
```

```
student@attackdefense:~/project$ cd /home/student/project ; env PYTHONIOENCODING=UTF-8 PYTHONUNBUFFERED=1 /usr/bin/python /home/student/.l
ocal/share/code-server/extensions/ms-python.python-2019.6.24221/pythonFiles/ptvsd_launcher.py --default --nodebug --client --host localhos
t --port 42517 /home/student/project/solution/solution-Q4.py
"All collected functions: ['get_rand_iteration', 'main', 'random_num_gen']"
'Number of assembly lines : 28'
'Number of assembly lines : 79'
'Number of assembly lines : 26'
'Total assembly line count : 133'
['"iteration":0,"register-state:"[\'rax          0x7fffffffe3b0\', \'rbx          0x7fffffffe3b0\', \'rcx          0x555555554960\',
 \'rdx          0x7fffffffe4f0\', \'rsi          0x7fffffffe4d8\', \'rdi          0x2\', \'rbp          0x7fffffffe3f0\', \'rsp
          0x7fffffffe3b0\', \'r8          0x7ffff7dd0d80\', \'r9          0x7ffff7dd0d80\', \'r10          0x1\', \'r11
0x0\', \'r12          0x555555554680\', \'r13          0x7fffffffe4d0\', \'r14          0x0\', \'r15          0x0\', \'rip
    0x555555554855\', \'eflags        0x246\', \'cs          0x33\', \'ss          0x2b\', \'ds          0x0\', \'es
 0x0\', \'fs          0x0\', \'gs          0x0\']',
 '"iteration":1,"register-state:"[\'rax          0x7fffffffe3b0\', \'rbx          0x7fffffffe3b0\', \'rcx          0x555555554960\',
 \'rdx          0x7fffffffe4f0\', \'rsi          0x7fffffffe4d8\', \'rdi          0x2\', \'rbp          0x7fffffffe3f0\', \'rsp
          0x7fffffffe3b0\', \'r8          0x7ffff7dd0d80\', \'r9          0x7ffff7dd0d80\', \'r10          0x1\', \'r11
```

References:

1. Visual Studio Code (https://code.visualstudio.com/)
2. VS Code Basic Editing (https://code.visualstudio.com/docs/editor/codebasics)