ATTACK DEFENSE

by PentesterAcademy

Name	Recording/Replaying Inferior's Execution
URL	https://www.attackdefense.com/challengedetails?cid=2167
Type	Reverse Engineering : GDB Basics

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.

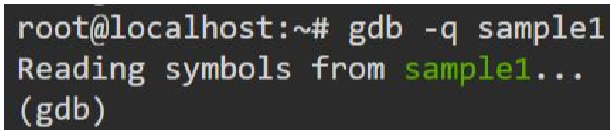
Objective: Learn how to run programs backward in GDB and check out different commands/options/methods.

Solution:

GDB allows the user to record a log of the process execution and replay it later with forward/reverse execution commands.

Step 1: Open sample1 binary using gdb.

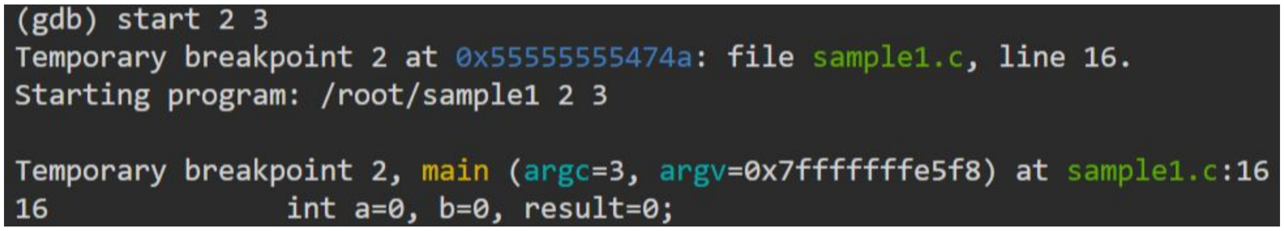
Command: gdb -q sample1



```
root@localhost:~# gdb -q sample1
Reading symbols from sample1...
(gdb)
```

Step 2: Run the program again.

Command: start 2 3

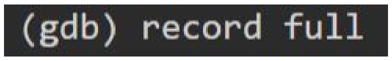

```
(gdb) start 2 3
Temporary breakpoint 2 at 0x55555555474a: file sample1.c, line 16.
Starting program: /root/sample1 2 3

Temporary breakpoint 2, main (argc=3, argv=0x7fffffffe5f8) at sample1.c:16
16              int a=0, b=0, result=0;
```

The program execution stopped at first line of the main() function.

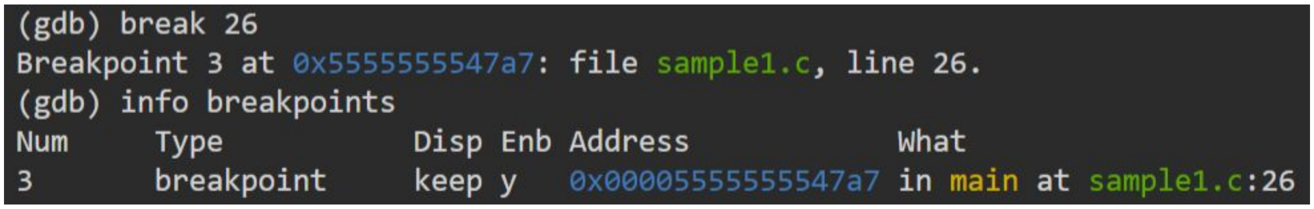
Step 3: Start the recording.

Command: record full


```
(gdb) record full
```

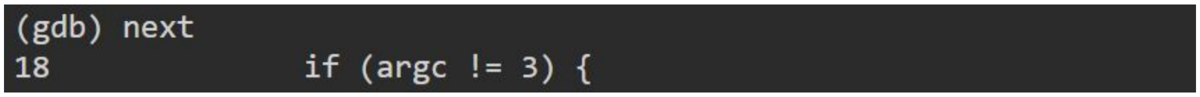
Step 4: Set a breakpoint at source code line 26 and list the breakpoints.

Commands:
break 26
info breakpoints


```
(gdb) break 26
Breakpoint 3 at 0x5555555547a7: file sample1.c, line 26.
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
3       breakpoint     keep y   0x00005555555547a7 in main at sample1.c:26
```

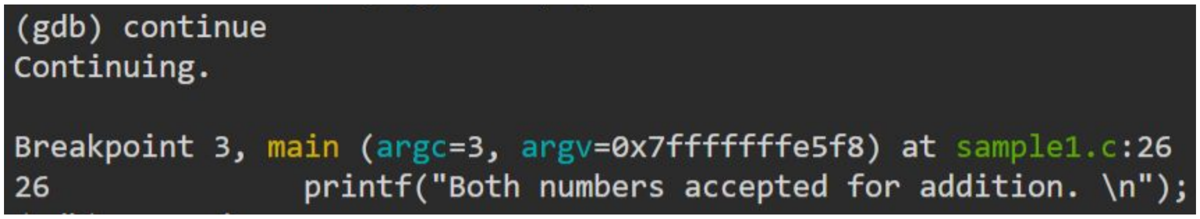
Step 5: Step to the next source line

Command: next


```
(gdb) next
18              if (argc != 3) {
```

Step 6: Step to the next checkpoint

Command: continue

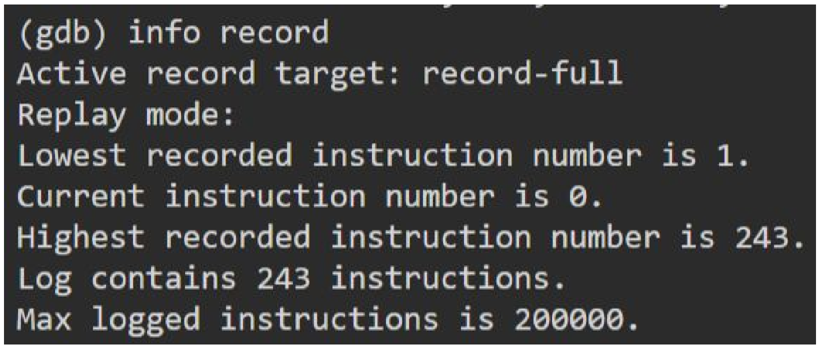


```
(gdb) continue
Continuing.

Breakpoint 3, main (argc=3, argv=0x7fffffffe5f8) at sample1.c:26
26              printf("Both numbers accepted for addition. \n");
```

Step 7: Check the details of the record.

Command: info record

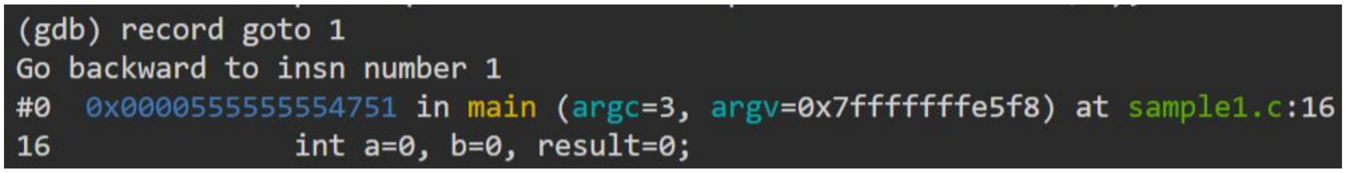


```
(gdb) info record
Active record target: record-full
Replay mode:
Lowest recorded instruction number is 1.
Current instruction number is 0.
Highest recorded instruction number is 243.
Log contains 243 instructions.
Max logged instructions is 200000.
```

The steps executed until now are recorded. The details of the record show that 243 instructions were recorded that can be referred to using instruction number.

Step 8: The user can jump to any location in the record. Jump to instruction number 1.

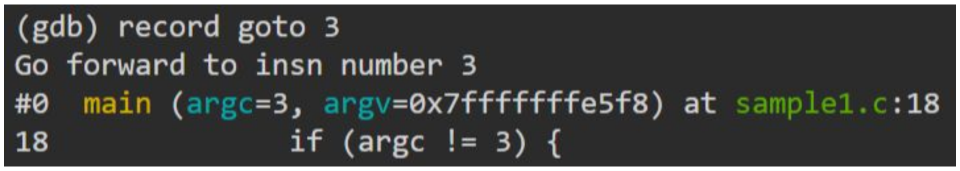
Command: record goto 1



```
(gdb) record goto 1
Go backward to insn number 1
#0  0x0000555555554751 in main (argc=3, argv=0x7fffffffe5f8) at sample1.c:16
16              int a=0, b=0, result=0;
```

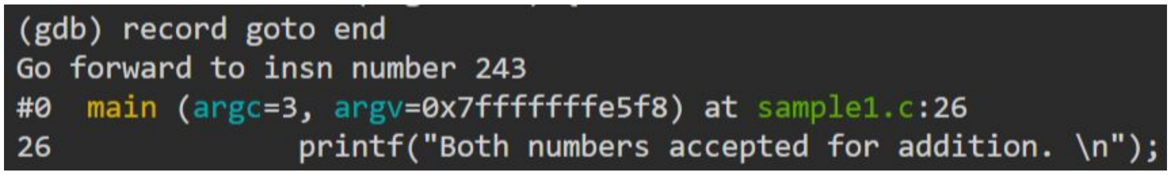
Step 9: Jump to instruction number 3.

Command: record goto 3


```
(gdb) record goto 3
Go forward to insn number 3
#0  main (argc=3, argv=0x7fffffffe5f8) at sample1.c:18
18              if (argc != 3) {
```

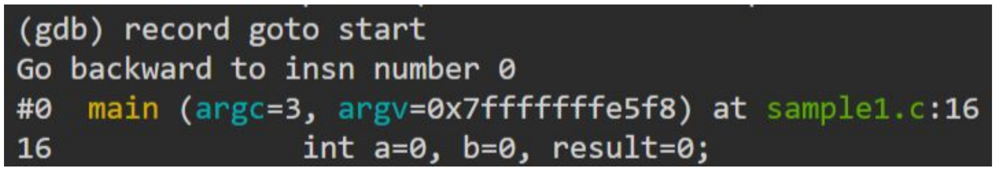
Step 10: Jump to the end of the record.

Command: record goto end


```
(gdb) record goto end
Go forward to insn number 243
#0  main (argc=3, argv=0x7fffffffe5f8) at sample1.c:26
26              printf("Both numbers accepted for addition. \n");
```

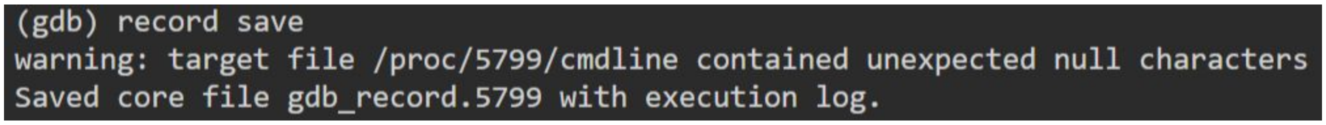
Step 11: Jump to the start of the record.

Command: record goto start


```
(gdb) record goto start
Go backward to insn number 0
#0  main (argc=3, argv=0x7fffffffe5f8) at sample1.c:16
16              int a=0, b=0, result=0;
```

Step 12: The record can also be saved to a file so it can be used later.

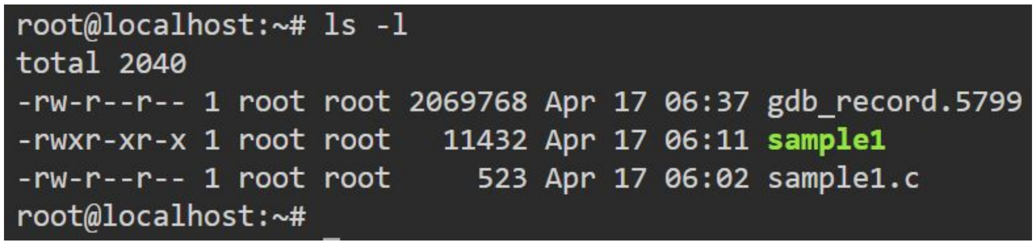
Command: record save


```
(gdb) record save
warning: target file /proc/5799/cmdline contained unexpected null characters
Saved core file gdb_record.5799 with execution log.
```

record save command takes file_name as the argument. If no argument is provided to then the default filename is gdb_record.process_id, where process_id is the process ID of the inferior.

This file can be found in the present working directory (outside of GDB).

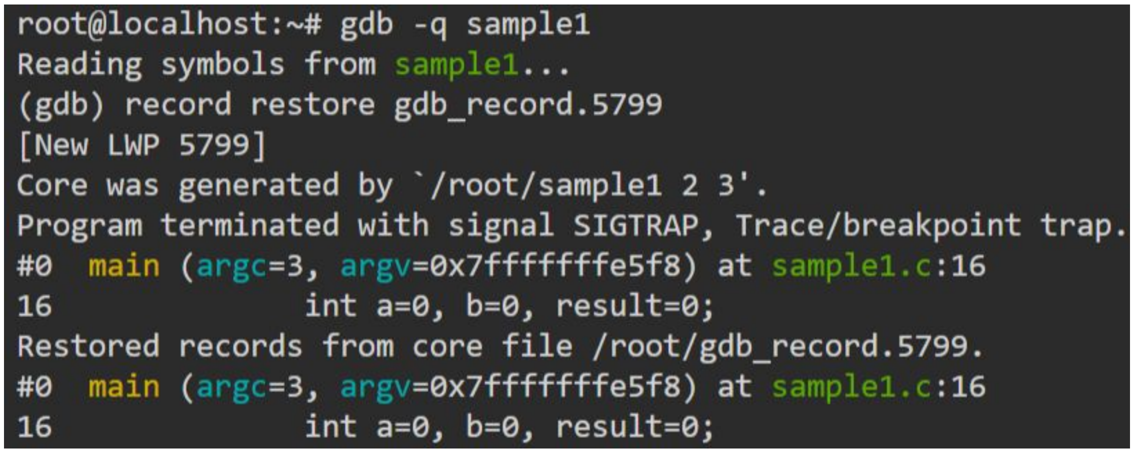
Command: ls -l



```
root@localhost:~# ls -l
total 2040
-rw-r--r-- 1 root root 2069768 Apr 17 06:37 gdb_record.5799
-rwxr-xr-x 1 root root   11432 Apr 17 06:11 sample1
-rw-r--r-- 1 root root     523 Apr 17 06:02 sample1.c
root@localhost:~#
```

Step 13: Restart the GDB with the sample1 binary again and load the saved record from the file gdb_record.5799.

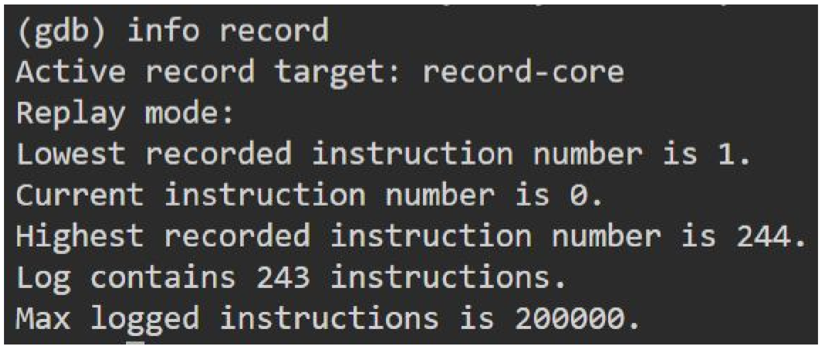
Commands:
gdb -q sample1
record restore gdb_record.5799



```
root@localhost:~# gdb -q sample1
Reading symbols from sample1...
(gdb) record restore gdb_record.5799
[New LWP 5799]
Core was generated by `/root/sample1 2 3'.
Program terminated with signal SIGTRAP, Trace/breakpoint trap.
#0  main (argc=3, argv=0x7fffffffe5f8) at sample1.c:16
16              int a=0, b=0, result=0;
Restored records from core file /root/gdb_record.5799.
#0  main (argc=3, argv=0x7fffffffe5f8) at sample1.c:16
16              int a=0, b=0, result=0;
```

Step 14: Check the details of the imported record.

Command: info record

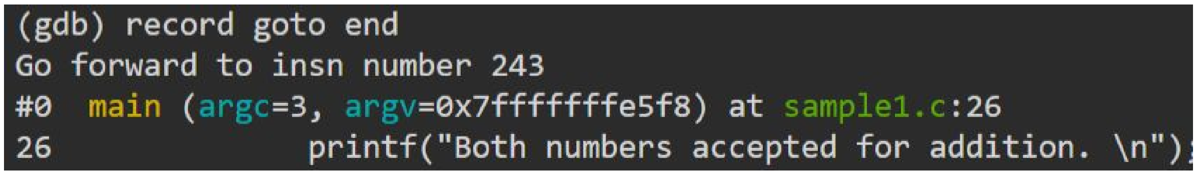


```
(gdb) info record
Active record target: record-core
Replay mode:
Lowest recorded instruction number is 1.
Current instruction number is 0.
Highest recorded instruction number is 244.
Log contains 243 instructions.
Max logged instructions is 200000.
```

The record is loaded successfully.

Step 15: Jump to the end of the record.

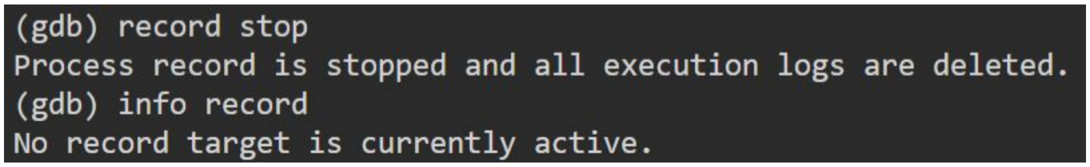
Command: record goto end



```
(gdb) record goto end
Go forward to insn number 243
#0  main (argc=3, argv=0x7fffffffe5f8) at sample1.c:26
26              printf("Both numbers accepted for addition. \n")
```

Step 16: Stop the record. Stopping the record deletes the logs. Also, check the record list again to verify that the record is deleted.

Commands:
record stop
info record



```
(gdb) record stop
Process record is stopped and all execution logs are deleted.
(gdb) info record
No record target is currently active.
```

In this manner, recording and replaying of inferior/program execution are possible in GDB.

References:

1. GDB Documentation (https://sourceware.org/gdb/current/onlinedocs/gdb)