Task 2 for angr Tutorial

—

Common Strategy to Mitigate Path Explosion

Cheat Sheet of angr

```
import angr
proj = angr.Project('a.out', auto_load_libs=False, load_debug_info=True)
state = proj.factory.entry_state()
simgr = proj.factory.simgr(state)
simgr.explore(find=[List of Find Address], avoid=[List of Avoid Address])
s = simgr.found[0]
print(s.posix.dumps(1))
```

Find More there

https://docs.angr.io/en/latest/core-concepts/pathgroups.html

https://docs.angr.io/en/latest/core-concepts/simulation.html

Type of Avoid Path

- Remove the failed operation path.
  - Print Invalid argument / command and Try again.
  - Return Error in a function
- Remove unnecessary command branch
  - Help command: (Just print some information and return to event dispatch again)
  - The command not used in Task
- Remove the duplicate switch-case branch
- Shrink the size of Array before compile

```
1.    link_node* find_user_node (const char *name)
2.    {
3.      for (link_node* node = username_list_head;
      node; node = node->next)
4.          if (strcmp (name, node->name) == 0)
5.            return node;
6.      return NULL;
7.    }
```

Return NULL in line 7 seems like a failed path and should be avoid, but sometime the program needs to ensure the same name node do not exist before adding.

Practice Task 1

- It’s a subset of the FTP protocol that only permits the following login methods:
  - >USER admin
  - 331 User admin OK.
  - >PASS ftp
  - 230 OK. Current user is admin.
  - You successfully logged.
- It uses a structure `session` to save the state of login process.
  - Session.state = 0 init state
  - Session.state = 1 username recorded.
  - Session.state >=2 logged in
- However, this program has a problem that allows users to log in without a password or even the PASS command.

Tip:

When strcpy is performed inside a structure, a buffer overflow occurs.

Compile the main.c by:

1. gcc main.c -g -gdwarf-4 -o main
2. Don’t forget the `-g gdwarf-4` flag to add debug info.

Practice Task 1 Solve Strategy

How to set avoid to restrict search space in order to locate the state that triggered the bug.

- Since we don’t use PASS command, set avoids at any command we don’t need.
  - command_PASS function
  - command_QUIT function
- We want to find the shortest path to the target state. Therefore, we don’t want to waste time on any failed attempts. Avoid failed attempts and ensure that every step in the path is valuable.
  - Line 81 printf("530 You aren't logged in.\n");
  - Line 66 printf("550 Invalid argument (no newline)\n");
  - Line 28 printf("530 Please tell me who you are\n");
  - Line 37 printf("530 Login authentication failed as User %s.\n", session->user);
- Set Find at
  - Line 85 printf("You successfully logged in.\n");

Practice Task 2

- This program reads a string from standard input and outputs the escaped version on standard output.
- However, input may cause the software to crash.

Tip: when a character is pushed into a buffer on the stack, a buffer overflow occurs.

Compile the main.c by:

1. gcc main.c -g -gdwarf-4 -o main
2. Don’t forget the `-g gdwarf-4` flag to add debug info.

Practice Task 2 Solve Strategy

The number of states in the escape function loop grows rapidly.

- 5 ways loop indicates 5 times the number of states following one loop.
- We need 64 loop to complete the string.
- 5^64 is an unsolvable large number.

We could reduce the amount of the buffer to accelerate program termination.

Four of the five branches transform one illegal character to two escaped characters. We need only one of them.

- Line 5 #define BUFF_LEN 64
  - Shrink the size to 8
  - Don't forget to recompile the source code.
- Avoid Line 11 ~ 13:
  - case '\n': *dest++ = '\\'; *dest = 'n'; break;
  - Only one branch remained to keep program's functional.
- The total number of state will be
  - 2^8 much better!

Practice Task 3

- Similar to the Practice Task 1 but
  - You are logged in at the first time
  - We need the QUIT command now.
  - Struct session are moved to heap area so we need free it after
- Do we manage the memory currently?

Tip: use-after-free

Compile the main.c by:

1. gcc main.c -g -gdwarf-4 -o main
2. Don’t forget the `-g gdwarf-4` flag to add debug info.

Practice Task 3 Solve Strategy

- Avoid failed attempts and ensure that each step along the path is significant.
    - Line 53 printf("550 Invalid argument (no newline)\n");
    - Line 64 printf("530 ???.\n");
    - Line 15 HELP command function