6.7 Debugging automated assessment in labs

Developing automated assessment for a new lab typically requires some amount of debugging. This section is intended to guide new developers through the process.

When the `gradelab` script is run from labtainers-instructor, the configuration files in labs/[lab name]/instr_config are validated. If syntax errors are found, error messages are displayed at the terminal and processing halts. The error messages identify the offending results.config or goals.config entry. Refer to sections 6.2 and 6.3 for the expected syntax of these files.

Once initial syntax checking is passed, the lab is graded for each student. If the grading table does not display, or it displays incorrects values, then find run the gradelab command with the `-d` option. At the resulting terminal, enter the `instructor.py` command. That may display diagnostics at the terminal. It will also generate a /tmp/instructor.log file of debugging messages.

At this point, the workflow is easiest if you edit/test from that container – just remember to transfer your revised .config files from the container before doing a `gradelab [lab] -r`! A copy of your config files are in `~/.local/instr\_config`. You can edit those and try running instructor.py again. The [lab].grades.json contains the results of the goals assessment. You can find the results assessment for each student beneath the directory whose name is prefaced with the student email. From there, look in .local/result to find json files reflecting intermediate results of assessing the student results. The actual student result arifacts can be found in `~/[student dir]/[lab].[container].student/.local/result`.

The mechanics of performing the lab (so that you can test grading for different outcomes) can be automated using the SimLab tool described in Appendix A.

7 Quizzes

Labs may include simple quizzes intended to re-enforce a student's understanding of concepts necessary to perform the lab. The quizzes are not intended to be a primary source of student assessment, rather, they are intended to help the student understand if they understand. No attempt is made to protect quiz answers, or to randomize or parameterize quizzes. An example application of quizzes is to allow the student to confirm his or her understanding of a security policy prior to trying to implement enforcement of that policy.

Quizzes are performed on the Labtainer host from within the `labtainer-student` directory using the `quiz` command. Use the `-h` option to see its usage.

A lab may have multiple quizzes. Each is defined in a file in the lab `config` directory within a file having an extension of `.quiz`. Each quiz includes a set of questions. Each question is defined by a comma separated list. If a line terminates without a comma or a backslash, it is treated as the end of the question, and the next line is treated as the beginning of the next question. Question types include the following:

7.1 True or False

A question whose answer is either true or false.

```
ID, TrueFalse, question, answer, right_response, wrong_response, prerequisit
```

Where:

- ID – any string identifier, must be unique. Currently only used to identify prerequisites as described below.
- TrueFalse – identifies this as a True or False question.