How to access Coverity Connect

The server is running at http://coverity.cs.ru.nl/ (http://coverity.cs.ru.nl/)

Your username is the same as your C&CZ login name; your password has been mailed to you (or you can try resetting it).

Coverity Connect does not perform any analysis by itself; its purpose is only to allow you to manage defect reports and allow you to triage them.

Note that you have a lot of access rights on this server. Feel free to play around, but behave responsibly.

How to run Coverity Analysis

Coverity is a proprietary software project, so we only have a restricted number of licenses. Therefore, we have installed it for you on the faculty login server `lilo5.science.ru.nl`. If you want to run Coverity Analysis on your own machine, please read the instructions at the end.

There are two ways to run a Coverity Analysis; either from the command line or by using the graphical `cov-wizard` tool. We recommend using the command line, so you know what is going on.

This document is based on this cheatsheet (https://community.synopsys.com/s/question/0D53400003tHtG6CAK/quick-start-guide-covconfigure-covbuild-and-covcommit) from the Synopsis website.

Step 0: Add Coverity Analysis to your path

At your bash prompt, type:

```
export PATH=$PATH:/vol/ds/coverity/cov-analysis-linux64-2019.03/bin
```

Step 1: Configuring a compiler

We have already configured Coverity for you, so you don't need to do anything.

> But if you want, you can also create a custom configuration. E.g.
>
> ```
> cov-configure --config myConfig.xml --compiler /usr/bin/gcc --comptype gcc
> ```
>
> This will tell coverity that `/usr/bin/cc` is GCC. You will then need to pass the `--config myConfix.xml` argument to all subsequent commands, however.

Step 2: Capturing a build

You now will have to show Coverity what happens when you build your project. This is called *capturing*. You do this by simply passing whatever command you use to build a project to `cov-build`, and telling it where to store its intermediate files.

E.g., if you normally compile using `gcc foo.c`, you now use:

```
cov-build --dir MY_EMIT_DIR gcc foo.c
```

In case of a serious project that uses makefiles, make sure that your build directory is clean before capturing:

```
make clean
cov-build --dir MY_EMIT_DIR make all
```

If your want to capture a project needs some preparation before building (e.g. `./configure` or `cmake`), you do that *before* capturing.

> (Of course, you can replace `MY_EMIT_DIR` with whatever you want.)

Step 3: Analyze

Now, you are ready to run the analysis:

```
cov-analyze --dir MY_EMIT_DIR
```

Note that `cov-analyze` supports many command line arguments, which lets you control how thoroughly the code gets analyzed. For instance, to get more errors, you could run:

```
cov-analyze --dir MY_EMIT_DIR --all --aggressiveness-level high
```

You can at this moment also inspect the results in a rudimentary fashion by using `cov-format-errors`:

```
cov-format-errors --dir MY_EMIT_DIR --html-output HTML_OUTPUT_DIR
```

And then look at the `HTML_OUTPUT_DIR/index.html` file generated. However, to perform *defect management*, you need to use Coverity Connect.

Step 4: Administration

Before you can commit your defects to Coverity Connect, you will have to create a stream and associate this with a project. In a business setting an administator will have done this and tell you what stream to commit to.

First, you will have to create a *stream* to which you will commit your report. You can do this with the command:

```
cov-manage-im --host coverity.cs.ru.nl --mode streams
  --add --set name:YOUR_STREAM_NAME
```

This will ask you for your Coverity Connect password.

Then, create a project and associate your stream with it:

```
cov-manage-im --host coverity.cs.ru.nl --mode projects
  --add --set name:YOUR_PROJECT_NAME --insert stream:YOUR_STREAM_NAME
```

Note that this will create a stream using the *default triage store*, which is shared by everybody. This means you can influence what your colleagues are doing (and vice versa). If you don't want that, you can create a seperate *triage store*:

```
cov-manage-im --host coverity.cs.ru.nl --mode triage
  --add --set name:YOUR_TRIAGE_NAME
```

And then specify (during creation) that your stream should use this store:

```
cov-manage-im --host coverity.cs.ru.nl --mode streams
  --add --set name:YOUR_STREAM_NAME --set triage:YOUR_TRIAGE_NAME
```

> (Obviously, substitute something original for `YOUR_XXX_NAME` here.)

Step 5: Committing your report

You can now upload your results to the Coverity Connect server.

```
cov-commit-defects --host coverity.cs.ru.nl --stream YOUR_STREAM_NAME --dir MY_EMIT_DIR
```

Which will again ask for your Coverity Connect password.

You can now log in to Coverity Connect using your browser and triage the results.

Step 6: (Optional) Generating an authentication key

If you don't want to type your password, you might be tempted to add is using the `--password` option to `cov-commit-defects`. However, `lilo5` is a public server, so this is not the best idea. An alternative is to generate an authentication key. You can do this with the command:

```
cov-manage-im --host coverity.cs.ru.nl --mode auth-key --create --output-file MY_KEY
```

And then you can run `cov-commit-defects` with the added switch `--auth-key-file MY_KEY`.

Running Coverity on your system

If you want to run Coverity Analysis on your own machine, this is possible. There are some requirements for this:

1. You have to have a working compiler installed.

2. We have to provide you with a license key, which will bind Coverity Analysis to run on a single machine.

If you want to obtain a key, email me (Marc Schoolderman), and tell me what operating system and machine you want to run Coverity Analysis on.