1 Chef installation

1.1 Building S2E

The core framework here is S2E. Hence, the first step is that we have to install S2E. Assume the root building repository is chef. Follow these steps to build S2E:

- First and foremost, we have to install fundamental required packages:
  - sudo apt-get install build-essential
  - sudo apt-get install subversion
  - sudo apt-get install git
  - sudo apt-get install gettext
  - sudo apt-get install liblua5.1-0-dev
  - sudo apt-get install libsdl1.2-dev
  - sudo apt-get install libsigc++-2.0-dev
  - sudo apt-get install binutils-dev
  - sudo apt-get install python-docutils
  - sudo apt-get install python-pygments
  - sudo apt-get install nasm
  - sudo apt-get install libiberty-dev
  - sudo apt-get install libc6-dev-i386
- Next, we have to install the dependencies for llvm-3.0 [11] and qemu [12]:
  - sudo apt-get install build-dep llvm-3.3
  - sudo apt-get install build-dep qemu
- The third step is to clone the S2E repository into chef/s2e:
  - cd chef/s2e
  - git clone https://github.com/dslab-epfl/s2e.git
- The for step is building S2E. This step is quite long that often requires 3-4 hours. Therefore, we have to be patient.

  - mkdir chef/s2e/build
  - cd chef/s2e/build
  - make -f ../s2e/Makefile

1.2 Creating Chef Virtual Machine

- Firstly, we have to create repository called vm inside chef directory. In the **chef/vm** directory, create a new S2E raw disk image with volume of 4 Gb called chef_disk.raw:
  - mkdir chef/vm
  - chef/s2e/build/qemu-release/qemu-img create -f raw chef_disk.raw 4G
- Debian installation image cannot be download by wget as suggestion by instructions in official github page. We have to download manually at `http://mirror.exetel.com.au/pub/debian/debian-cd/7.4.0/i386/iso-cd/`
- Run QEMU and install the OS:
  - chef/s2e/build/qemu-release/i386-softmmu/qemu-system-i386 chef_disk.raw -enable-kvm -m 1024 -cdrom debian-7.4.0-i386-CD-1.iso
  - We just follow the instruction of QEMU and complete the installation of Chef virtual machine.

1.3 Setting up the Chef VM

To facilitate interaction with S2E, Chef provides the **run_qemu.py** wrapper script. **run_qemu.py** operates in one of three possible modes:

- **kvm** - concrete mode, hardware accelerated. This mode is most suitable for expensive setup operations, such as building and installing new packages. In this mode, all changes to the VM disk are persistent. Since the KVM snapshots are incompatible with the regular snapshots, the KVM mode should boot from scratch and the machine be shut down cleanly at the end.

- **prep** - snapshot preparation mode. In this mode, the VM still boots in non-symbolic mode, but all changes to the disk are ephemeral and only stored in snapshots. The purpose of this mode is to boot up the system from scratch and prepare it up to the point it is ready for symbolic execution.
- **sym** - symbolic mode. In this mode, the snapshot created in prep mode is executed in full symbolic mode.

To set up Chef virtual machine, these following instructions need to be executed.

1. Clone chef tool
   - mkdir chef/tools
   - cd chef/tools
   - git clone https://github.com/dslab-epfl/chef-tools.git
2. Boot up the Chef VM:
   - cd chef/tools
   - ./run_qemu.py kvm
3. Install additional packages for checking Python:
   - sudo apt-get install build-essential
   - sudo apt-get install unzip
   - sudo apt-get install git-core
   - sudo apt-get install libssl-dev
   - sudo apt-get install libsqlite3-dev
   - sudo apt-get install libreadline-dev
   - sudo apt-get install libbz2-dev
4. Clone the chef-tools repository and set up the interpreters you want to use. The root repository inside virtual machine is call chef_guest. We set up for both Python and Lua languages. You should get the following directory structure inside the guest:

Code 1: Directory structure inside virtual machine

```
chef_guest
   tools
   python
      chef
         build
   lua
      chef
         build
```

5. Shut down the VM: sudo halt

1.4 Preparing the host replay environment

Essentially, we need to download and set up again the interpreters we use, but this time on our own machine. We set up for both Python and Lua that are introduced in Chef. At the end of this part, we should get the following directory structure:

Code 2: Direction structure of host machine

```
chef
   tools
   s2e
      build
      s2e
   python
      chef
         build
   lua
      chef
         build
```

1.5 Running a Chef VM

1. Boot up the VM in preparation mode:
   - cd chef/tools

- ./run_qemu.py prep

2. Bring the VM to a point ready for symbolic execution. Then run **savevm 1** in the Qemu console and then **quit**.
3. Run the VM snapshot in symbolic mode:
   - ./run_qemu.py sym

2 Running symbolic execution for Python

The previous part is for the process of running symbolic execution of interpreted language. The virtual machine will go through **kvm**, **prep**, **sym** mode. In this section, we will introduce how to perform on Python program in particular.

2.1 Setting up the Chef environment for Python

We assume the following project structure in the guest or virtual machine like we introduced in the previous part:

Code 3: Directory structure of virtual machine

```
chef_guest
   tools
   python
      chef
         build
```

2.2 Phase 1: Preparing the guest environment

- Run S2E in normal mode (**kvm**)
- In the virtual machine repository chef_guest/python/chef/build, run the following command:
  - make -f ../Makefile.interp
- Then activate the resulting Python environment using:

  - source $HOME/chef_guest/python/chef/build/python-env/bin/activate

- Next, we install the Chef native extension:
  - cd chef_guest/python/chef/pychef
  - pip install -e .

2.3 Phase 2: Preparing the symbolic environment

- Run S2E in **prep** mode.
- Activate the Python environment:
  - source $HOME/chef_guest/python/chef/build/python-env/bin/activate
- Enable symbolic execution mode:
  - export PYTHONSYMBEX=1

2.4 Phase 3: Symbolic Execution

- Run S2E in **sym** mode
- Run the target symbolic test case. For instance:
  - cd symtests
  - python asplos_tests.py ArgparseTest
- In asplos_test.py, it has 6 tests: ArgparseTest, ConfigParserTest, HTMLParserTest, SimpleJSONTest, XLRDTest, UnicodeCSVTest. We can change ArgparseTest by each test and consider the result.