angr

Information Security Inc.

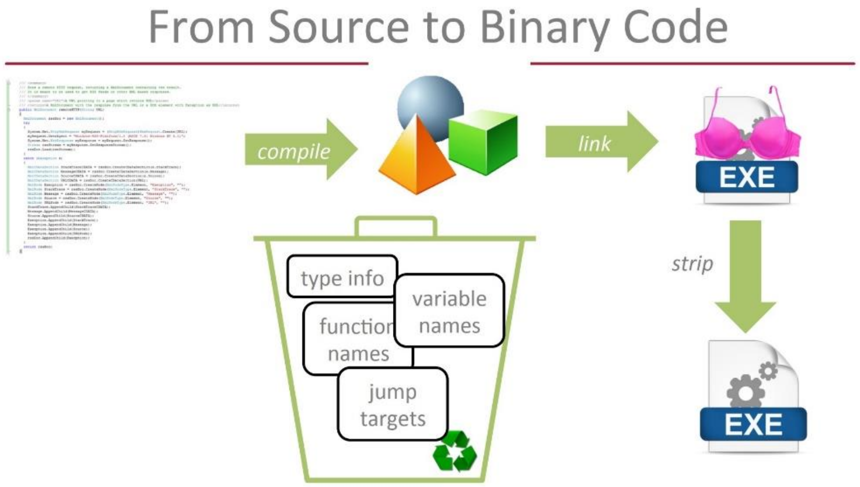
Contents

About angr

- angr is a python framework for analyzing binaries. It combines both static and dynamic symbolic analysis, making it applicable to a variety of tasks
- angr is a multi-architecture binary analysis toolkit, with the capability to perform dynamic symbolic execution (like Mayhem, KLEE, etc.) and various static analyses on binaries

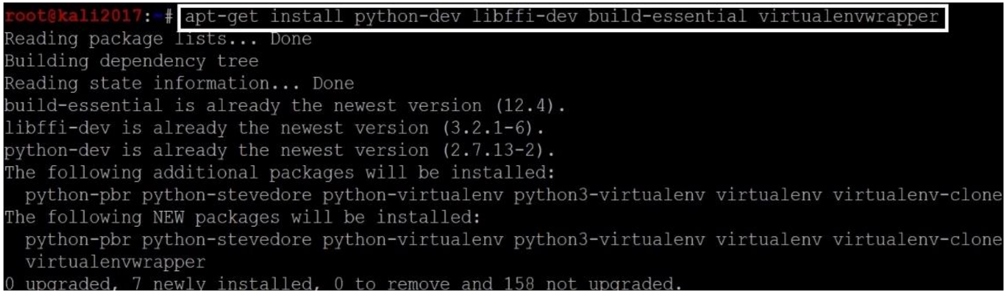
From Source to Binary Code

- Binaries lack significant information present in source



Dependencies

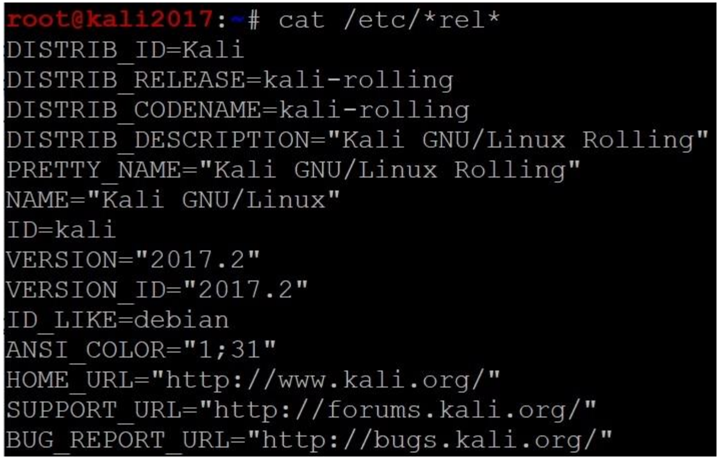
- All of the python dependencies should be handled by pip and/or the setup.py scripts
- libffi package
- apt-get install python-dev libffi-dev build-essential virtualenvwrapper



```
root@kali2017:~# apt-get install python-dev libffi-dev build-essential virtualenvwrapper
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.4).
libffi-dev is already the newest version (3.2.1-6).
python-dev is already the newest version (2.7.13-2).
The following additional packages will be installed:
  python-pbr python-stevedore python-virtualenv python3-virtualenv virtualenv virtualenv-clone
The following NEW packages will be installed:
  python-pbr python-stevedore python-virtualenv python3-virtualenv virtualenv virtualenv-clone
  virtualenvwrapper
0 upgraded, 7 newly installed, 0 to remove and 158 not upgraded.
```

Demo Setup

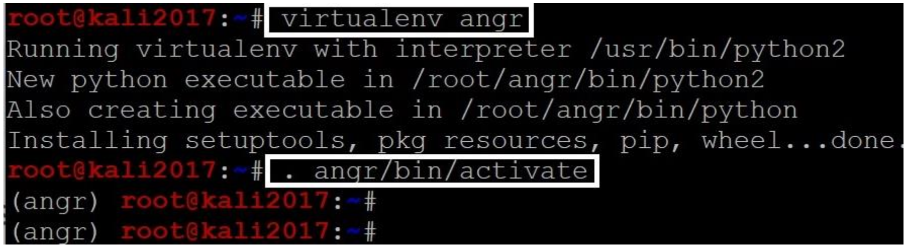
- Setup
- Kali Linux 2017



```
root@kali2017:~# cat /etc/*rel*
DISTRIB_ID=Kali
DISTRIB_RELEASE=kali-rolling
DISTRIB_CODENAME=kali-rolling
DISTRIB_DESCRIPTION="Kali GNU/Linux Rolling"
PRETTY_NAME="Kali GNU/Linux Rolling"
NAME="Kali GNU/Linux"
ID=kali
VERSION="2017.2"
VERSION_ID="2017.2"
ID_LIKE=debian
ANSI_COLOR="1;31"
HOME_URL="http://www.kali.org/"
SUPPORT_URL="http://forums.kali.org/"
BUG_REPORT_URL="http://bugs.kali.org/"
```

Installing angr

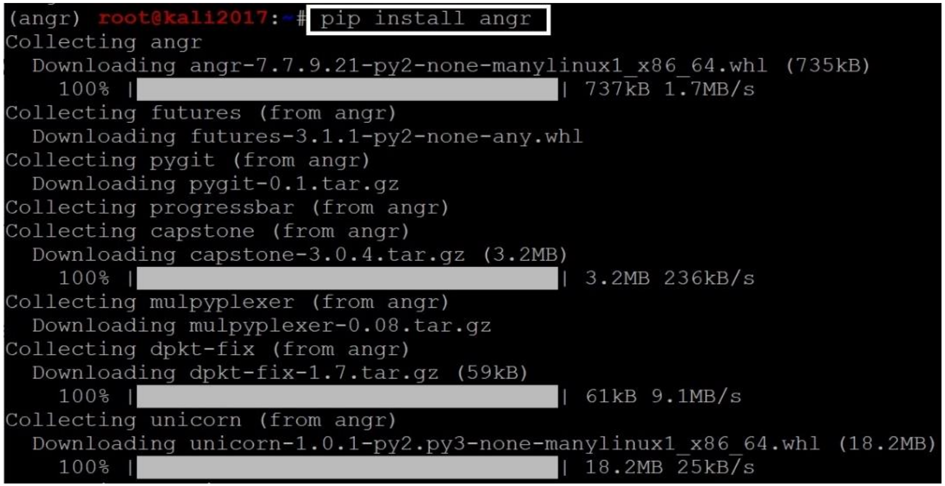
- angr is a python library, so it must be installed into the python environment before it can be used
- Make the python virtual environment



```
root@kali2017:~# virtualenv angr
Running virtualenv with interpreter /usr/bin/python2
New python executable in /root/angr/bin/python2
Also creating executable in /root/angr/bin/python
Installing setuptools, pkg resources, pip, wheel...done
root@kali2017:~# . angr/bin/activate
(angr) root@kali2017:~#
(angr) root@kali2017:~#
```

Installing angr

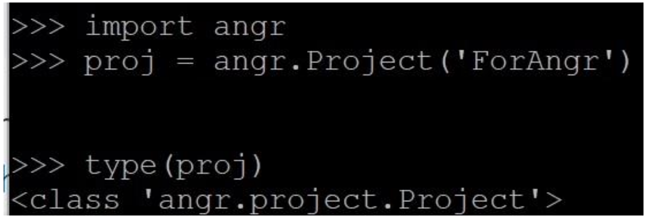
- angr is a python library, so it must be installed into the python environment before it can be used
- Installing using pip



```
(angr) root@kali2017:~# pip install angr
Collecting angr
  Downloading angr-7.7.9.21-py2-none-manylinux1_x86_64.whl (735kB)
    100% |████████████████████████████████| 737kB 1.7MB/s
Collecting futures (from angr)
  Downloading futures-3.1.1-py2-none-any.whl
Collecting pygit (from angr)
  Downloading pygit-0.1.tar.gz
Collecting progressbar (from angr)
Collecting capstone (from angr)
  Downloading capstone-3.0.4.tar.gz (3.2MB)
    100% |████████████████████████████████| 3.2MB 236kB/s
Collecting mulpyplexer (from angr)
  Downloading mulpyplexer-0.08.tar.gz
Collecting dpkt-fix (from angr)
  Downloading dpkt-fix-1.7.tar.gz (59kB)
    100% |████████████████████████████████| 61kB 9.1MB/s
Collecting unicorn (from angr)
  Downloading unicorn-1.0.1-py2.py3-none-manylinux1_x86_64.whl (18.2MB)
    100% |████████████████████████████████| 18.2MB 25kB/s
```

Using angr

- Core Concepts (https://github.com/angr/angr-doc/blob/master/docs/toplevel.md)
- First action with angr will always be to load a binary into a project. We'll use “ForAngr” binary for these examples.

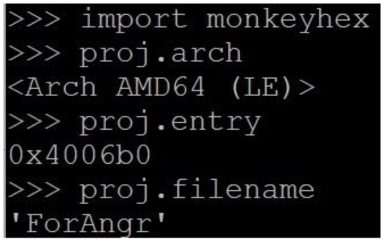


```
>>> import angr
>>> proj = angr.Project('ForAngr')


>>> type(proj)
<class 'angr.project.Project'>
```

Using angr

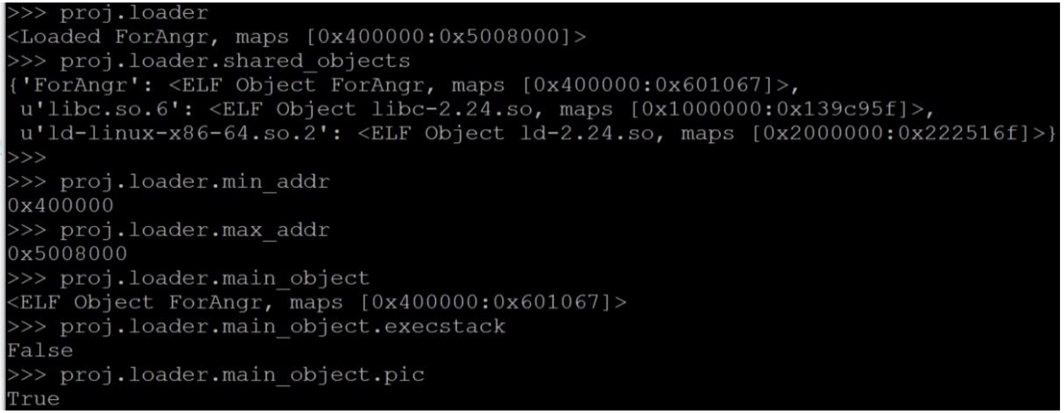
- Basic Properties about the project: its CPU architecture, its filename, and the address of its entry point
- import monkeyhex # this will format numerical results in hexadecimal



```
>>> import monkeyhex
>>> proj.arch
<Arch AMD64 (LE)>
>>> proj.entry
0x4006b0
>>> proj.filename
'ForAngr'
```

Using angr

- The Loader: Getting from a binary file to its representation in a virtual address space is pretty complicated!
- A module called CLE handles that
- Some basic queries about the loaded address space



```
>>> proj.loader
<Loaded ForAngr, maps [0x400000:0x5008000]>
>>> proj.loader.shared_objects
{'ForAngr': <ELF Object ForAngr, maps [0x400000:0x601067]>,
 u'libc.so.6': <ELF Object libc-2.24.so, maps [0x1000000:0x139c95f]>,
 u'ld-linux-x86-64.so.2': <ELF Object ld-2.24.so, maps [0x2000000:0x222516f]>}
>>>
>>> proj.loader.min_addr
0x400000
>>> proj.loader.max_addr
0x5008000
>>> proj.loader.main_object
<ELF Object ForAngr, maps [0x400000:0x601067]>
>>> proj.loader.main_object.execstack
False
>>> proj.loader.main_object.pic
True
```

References

- angr.io

http://angr.io/

- Kali Linux

https://www.kali.org/downloads/

- Wikipedia

https://en.wikipedia.org/wiki/Static_program_analysis
https://en.wikipedia.org/wiki/Symbolic_execution

- PIC (Position Independent Code)

https://en.wikipedia.org/wiki/Position-independent_code

- Executable stack protection

https://en.wikipedia.org/wiki/Executable_space_protection