Android 动态逆向分析工具（三）

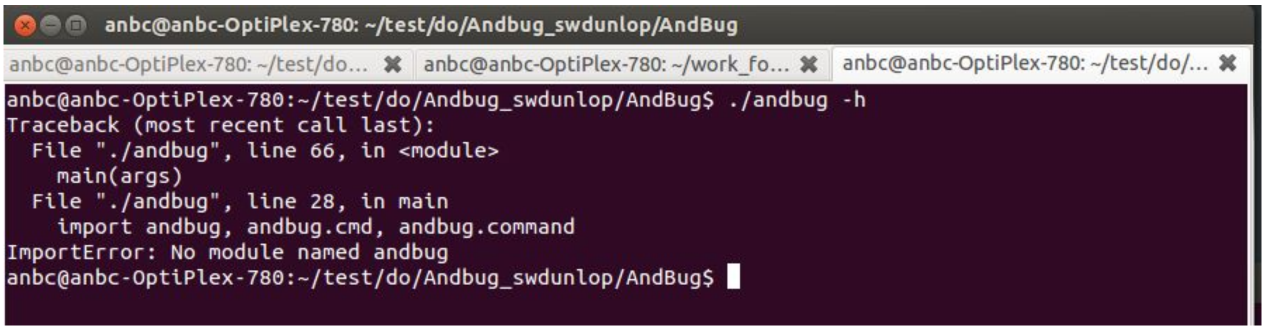
——Andbug 常见问题汇总

anbingchun@163.com

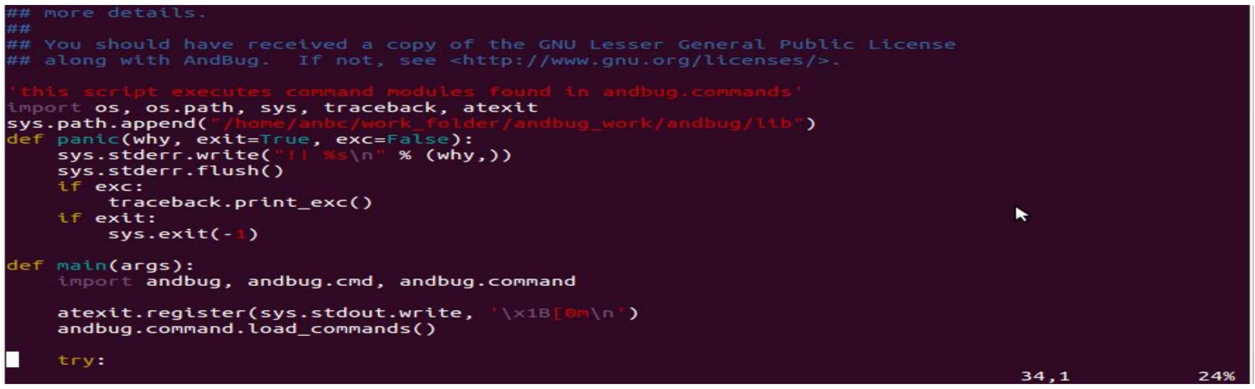
1、py 文件 import 找不到路径

git clone 后直接运行，会提示如下错误，原因是 andbug 工程中的某些 py 文件的路径没有正确指定导致，找不到这些 py 文件。

解决办法是（1）、通过“ PYTHONPATH=lib”指定路径。（2）、在 andbug 文件中添加 sys.path.append("/home/anbc/work_folder/andbug_work/andbug/lib")，来指定路径。


anbc@anbc-OptiPlex-780: ~/test/do/Andbug_swdunlop/AndBug
anbc@anbc-OptiPlex-780: ~/test/do...
anbc@anbc-OptiPlex-780: ~/work_fo...
anbc@anbc-OptiPlex-780: ~/test/do/...

```
anbc@anbc-OptiPlex-780:~/test/do/Andbug_swdunlop/AndBug$ ./andbug -h
Traceback (most recent call last):
  File "./andbug", line 66, in <module>
    main(args)
  File "./andbug", line 28, in main
    import andbug, andbug.cmd, andbug.command
ImportError: No module named andbug
anbc@anbc-OptiPlex-780:~/test/do/Andbug_swdunlop/AndBug$
```

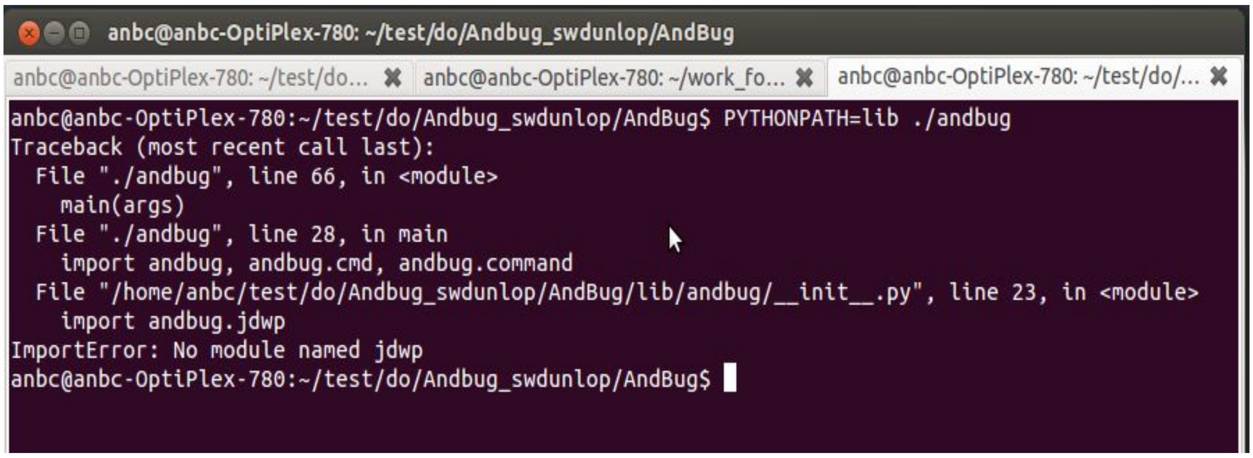


```
## more details.
##
## You should have received a copy of the GNU Lesser General Public License
## along with AndBug.  If not, see <http://www.gnu.org/licenses/>.

'this script executes command modules found in andbug.commands'
import os, os.path, sys, traceback, atexit
sys.path.append("/home/anbc/work_folder/andbug_work/andbug/lib")
def panic(why, exit=True, exc=False):
    sys.stderr.write("!! %s\n" % (why,))
    sys.stderr.flush()
    if exc:
        traceback.print_exc()
    if exit:
        sys.exit(-1)

def main(args):
    import andbug, andbug.cmd, andbug.command

    atexit.register(sys.stdout.write, '\x1B[0m\n')
    andbug.command.load_commands()

    try:
                                                              34,1          24%
```

2、未编译 so 模块导致的错误

为了提高数据的处理速度，andbug 在数据处理部分没有使用 python 编写，而是使用了 C 语言编写，以 so 的方式调用。所以正式使用 andbug 前，需要对 C 代码进行编译。用来编译 C 代码的 makefile 已经在 Andbug 文件夹里。可以直接使用 make 命令进行编译。

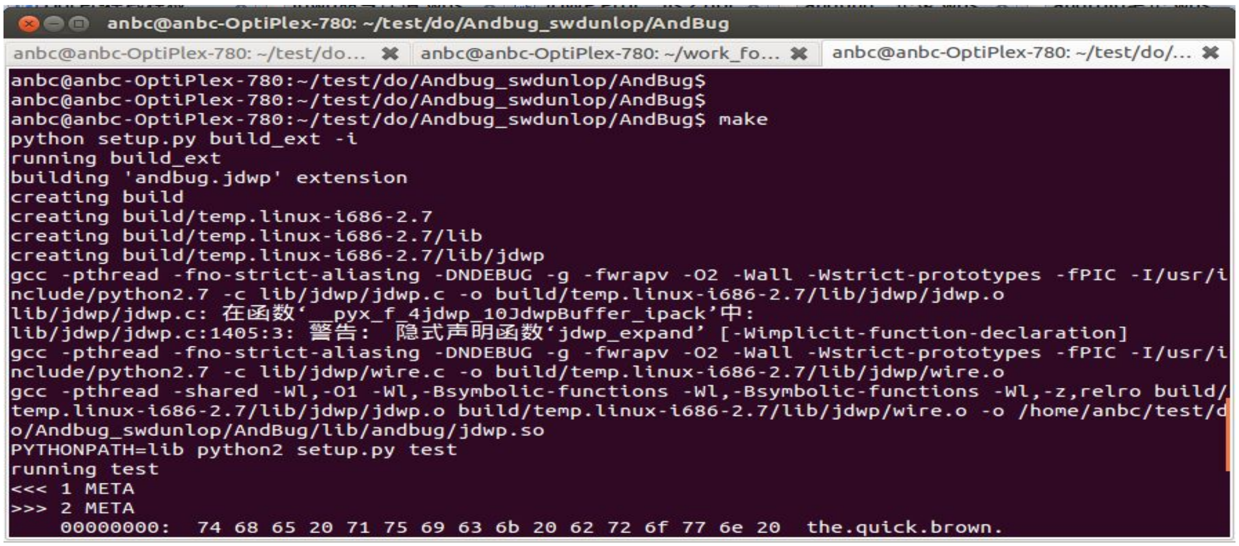
直接运行会提示如下错误：



anbc@anbc-OptiPlex-780: ~/test/do/Andbug_swdunlop/AndBug

```
anbc@anbc-OptiPlex-780:~/test/do/Andbug_swdunlop/AndBug$ PYTHONPATH=lib ./andbug
Traceback (most recent call last):
  File "./andbug", line 66, in <module>
    main(args)
  File "./andbug", line 28, in main
    import andbug, andbug.cmd, andbug.command
  File "/home/anbc/test/do/Andbug_swdunlop/AndBug/lib/andbug/__init__.py", line 23, in <module>
    import andbug.jdwp
ImportError: No module named jdwp
anbc@anbc-OptiPlex-780:~/test/do/Andbug_swdunlop/AndBug$ 
```

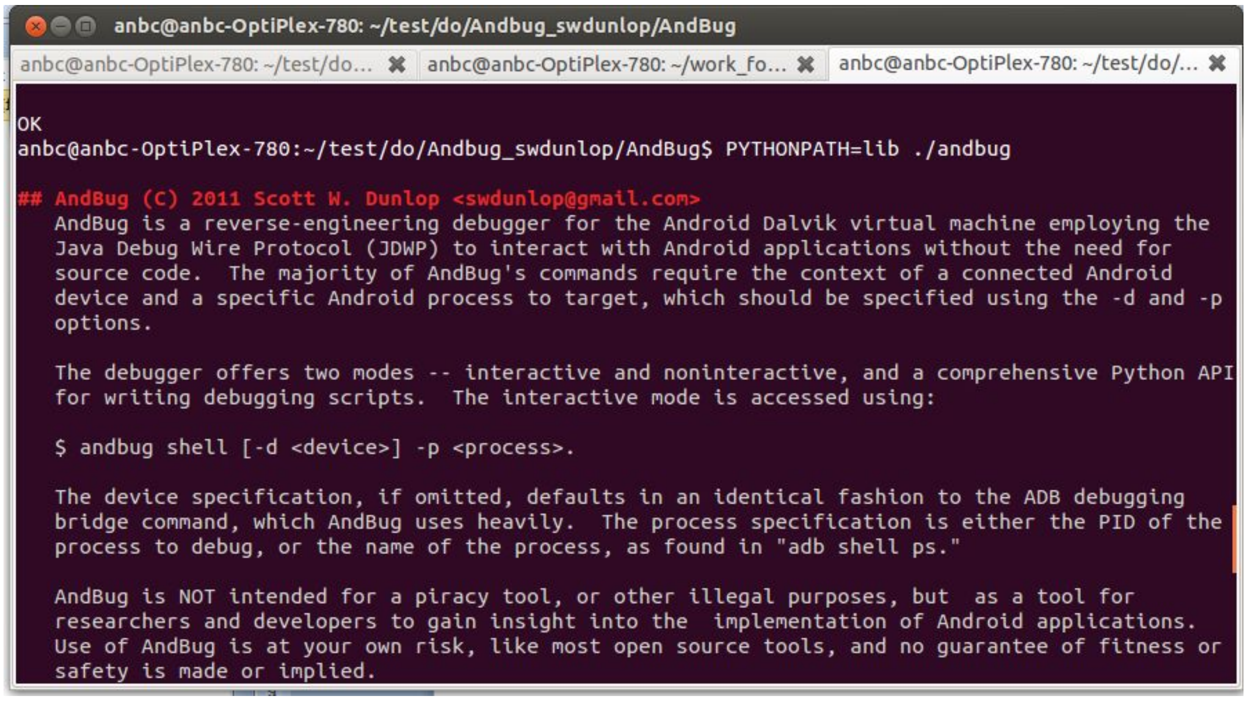
具体编译过程如下：



anbc@anbc-OptiPlex-780: ~/test/do/Andbug_swdunlop/AndBug

```
anbc@anbc-OptiPlex-780:~/test/do/Andbug_swdunlop/AndBug$
anbc@anbc-OptiPlex-780:~/test/do/Andbug_swdunlop/AndBug$
anbc@anbc-OptiPlex-780:~/test/do/Andbug_swdunlop/AndBug$ make
python setup.py build_ext -i
running build_ext
building 'andbug.jdwp' extension
creating build
creating build/temp.linux-i686-2.7
creating build/temp.linux-i686-2.7/lib
creating build/temp.linux-i686-2.7/lib/jdwp
gcc -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-prototypes -fPIC -I/usr/i
nclude/python2.7 -c lib/jdwp/jdwp.c -o build/temp.linux-i686-2.7/lib/jdwp/jdwp.o
lib/jdwp/jdwp.c: 在函数‘__pyx_f_4jdwp_10JdwpBuffer_ipack’中:
lib/jdwp/jdwp.c:1405:3: 警告:  隐式声明函数‘jdwp_expand’ [-Wimplicit-function-declaration]
gcc -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-prototypes -fPIC -I/usr/i
nclude/python2.7 -c lib/jdwp/wire.c -o build/temp.linux-i686-2.7/lib/jdwp/wire.o
gcc -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions -Wl,-Bsymbolic-functions -Wl,-z,relro build/
temp.linux-i686-2.7/lib/jdwp/jdwp.o build/temp.linux-i686-2.7/lib/jdwp/wire.o -o /home/anbc/test/d
o/Andbug_swdunlop/AndBug/lib/andbug/jdwp.so
PYTHONPATH=lib python2 setup.py test
running test
<<< 1 META
>>> 2 META
    00000000:  74 68 65 20 71 75 69 63 6b 20 62 72 6f 77 6e 20  the.quick.brown.
```

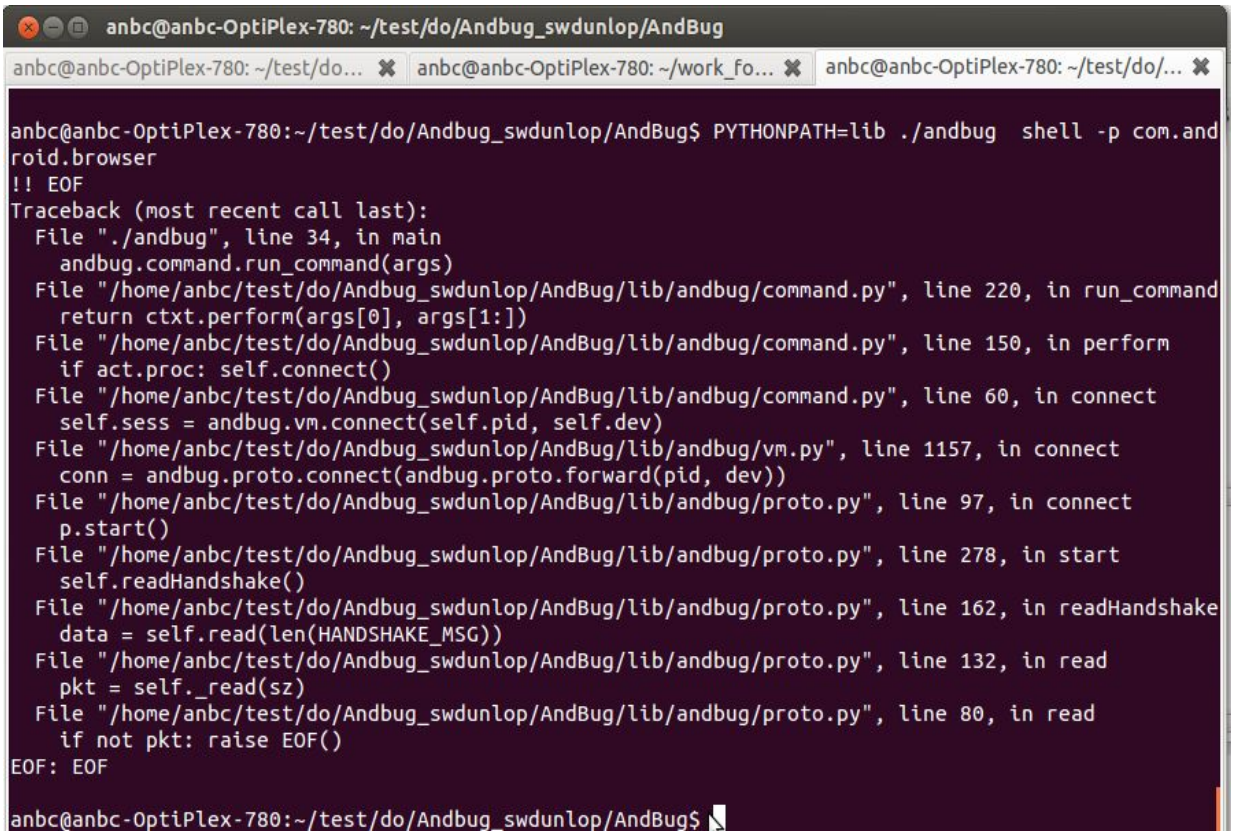
再次运行 Andbug，成功：


anbc@anbc-OptiPlex-780: ~/test/do/Andbug_swdunlop/AndBug

```
OK
anbc@anbc-OptiPlex-780:~/test/do/Andbug_swdunlop/AndBug$ PYTHONPATH=lib ./andbug

## AndBug (C) 2011 Scott W. Dunlop <swdunlop@gmail.com>
   AndBug is a reverse-engineering debugger for the Android Dalvik virtual machine employing the
   Java Debug Wire Protocol (JDWP) to interact with Android applications without the need for
   source code.  The majority of AndBug's commands require the context of a connected Android
   device and a specific Android process to target, which should be specified using the -d and -p
   options.

   The debugger offers two modes -- interactive and noninteractive, and a comprehensive Python API
   for writing debugging scripts.  The interactive mode is accessed using:

   $ andbug shell [-d <device>] -p <process>.

   The device specification, if omitted, defaults in an identical fashion to the ADB debugging
   bridge command, which AndBug uses heavily.  The process specification is either the PID of the
   process to debug, or the name of the process, as found in "adb shell ps."

   AndBug is NOT intended for a piracy tool, or other illegal purposes, but  as a tool for
   researchers and developers to gain insight into the  implementation of Android applications.
   Use of AndBug is at your own risk, like most open source tools, and no guarantee of fitness or
   safety is made or implied.
```

3、多调试端链接模拟器导致的错误

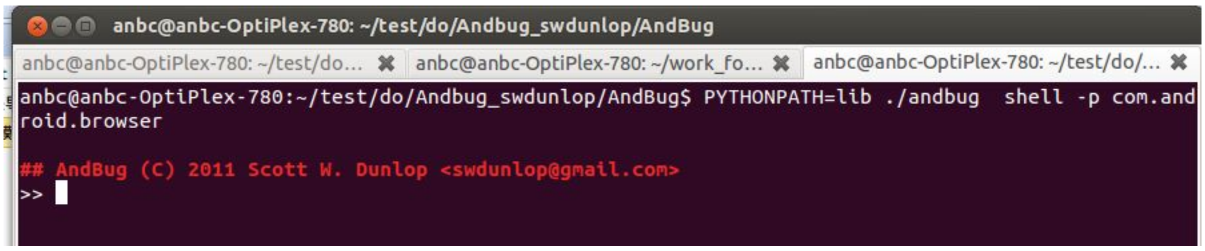
在启动 andbug 时，可能遇到由于多个调试端同时链接模拟器导致的运行错误。


anbc@anbc-OptiPlex-780: ~/test/do/Andbug_swdunlop/AndBug

```
anbc@anbc-OptiPlex-780:~/test/do/Andbug_swdunlop/AndBug$ PYTHONPATH=lib ./andbug  shell -p com.and
roid.browser
!! EOF
Traceback (most recent call last):
  File "./andbug", line 34, in main
    andbug.command.run_command(args)
  File "/home/anbc/test/do/Andbug_swdunlop/AndBug/lib/andbug/command.py", line 220, in run_command
    return ctxt.perform(args[0], args[1:])
  File "/home/anbc/test/do/Andbug_swdunlop/AndBug/lib/andbug/command.py", line 150, in perform
    if act.proc: self.connect()
  File "/home/anbc/test/do/Andbug_swdunlop/AndBug/lib/andbug/command.py", line 60, in connect
    self.sess = andbug.vm.connect(self.pid, self.dev)
  File "/home/anbc/test/do/Andbug_swdunlop/AndBug/lib/andbug/vm.py", line 1157, in connect
    conn = andbug.proto.connect(andbug.proto.forward(pid, dev))
  File "/home/anbc/test/do/Andbug_swdunlop/AndBug/lib/andbug/proto.py", line 97, in connect
    p.start()
  File "/home/anbc/test/do/Andbug_swdunlop/AndBug/lib/andbug/proto.py", line 278, in start
    self.readHandshake()
  File "/home/anbc/test/do/Andbug_swdunlop/AndBug/lib/andbug/proto.py", line 162, in readHandshake
    data = self.read(len(HANDSHAKE_MSG))
  File "/home/anbc/test/do/Andbug_swdunlop/AndBug/lib/andbug/proto.py", line 132, in read
    pkt = self._read(sz)
  File "/home/anbc/test/do/Andbug_swdunlop/AndBug/lib/andbug/proto.py", line 80, in read
    if not pkt: raise EOF()
EOF: EOF

anbc@anbc-OptiPlex-780:~/test/do/Andbug_swdunlop/AndBug$
```

开启多个调试链接的情况会有很多，如：运行了两个 andbug 程序；在用 eclipse 等工具调试 apk 程序等情况。只要将其他调试程序关掉，andbug 就可以正常使用了。

关掉其他调试链接后，可以运行成功了。



```
anbc@anbc-OptiPlex-780:~/test/do/Andbug_swdunlop/AndBug$ PYTHONPATH=lib ./andbug  shell -p com.and
roid.browser

## AndBug (C) 2011 Scott W. Dunlop <swdunlop@gmail.com>
>> 
```