项目一 file program

介绍

file program是Unix系统的文件命令库和魔数库。能够处理数千种不同的文件类型。代码被重构成库，能够被第三方库使用。

Angora fuzz流程

- 下载解压

```
wget https://github.com/file/file/archive/FILE5_32.tar.gz
tar -xvzf FILE5_32.tar.gz
```

- 使用Angora编译

```
cp -r file-FILE5_32 track
cd track
autoreconf -i
CC=~/angora/bin/angora-clang ./configure --prefix=`pwd`/install --disable-shared

USE_TRACK=1 make
```

- USE_TRACK=1 make出现error

```
/usr/bin/ld: ./.libs/libmagic.a(compress.o): in function `uncompresszlib':
/home/xx/example/track/src/compress.c:507: undefined reference to `dfs$inflateInit_'
/usr/bin/ld: /home/xx/example/track/src/compress.c:507: undefined reference to `dfs$inflateInit2_'
/usr/bin/ld: /home/xx/example/track/src/compress.c:511: undefined reference to `dfs$inflate'
/usr/bin/ld: /home/xx/example/track/src/compress.c:516: undefined reference to `dfs$inflateEnd'
/usr/bin/ld: /home/xx/example/track/src/compress.c:525: undefined reference to `dfs$zError'
```

- 将所有缺失库的taints丢弃(驱动2可以自己链接libz.so，不需要丢弃缺失库)

```
./angora/tools/gen_library_abilist.sh /usr/lib/x86_64-linux-gnu/libz.so  discard > zlib_abilist.txt
```

- 设置环境变量并重新编译

```
export ANGORA_TAINT_RULE_LIST=~/path-to/zlib_abilist.txt
make clean
USE_TRACK=1 make
make install
```

- 编译fast版本

```
cd ..
cp -r file-FILE5_32 fast
cd fast
```

```
autoreconf -i
CC=~/angora/bin/angora-clang ./configure --prefix=`pwd`/install --disable-shared
make
make install
```

- 加入随机测试内容

```
cd ..
mkdir seeds
echo "Hello World" > seeds/seed.txt
```

- 使用Angora开始模糊测试

```
~/angora/angora_fuzzer -i seeds -o output -t ./track/install/bin/file -- ./fast/install/bin/file -m
./fast/install/share/misc/magic.mgc @@
```

- 结果

- 驱动介绍

  i. 驱动1:/path-to-file/src/file （源代码自带的，将整个项目make完可以直接得到可执行文件），输入文件是seed.txt

  ii. 驱动2:/path-to-file/tests/test.c 测试代码转化成驱动，将test.o和make install生成的install/lib目录下的libmagic.a,libz.so链接成可执行文件，输入文件也是seed.txt

项目二 libconfig

介绍

Libconfig是一个用于处理结构化配置文件的简单库。相较于XML，这种结构化的配置文件内容更加紧凑，且可读性更强。和XML不一样的是，它能够感知类型，所以不需要在应用代码中做字符串解析。 libconfig本身设计得很紧凑，占用空间只是XML

解析库的一小部分。这让它在存储大小非常有限的小型设备中应用广泛。这个库在C和C++语句上都有对应的实现。可以工作在类UNIX操作系统如Linux, Mac OS X, FreeBSD, Android 和 Windows等等。

Angora fuzz流程

- git clone

```
https://github.com/hyperrealm/libconfig.git
```

- 生成Makefile

```
autoscan
aclocal
autoconf
automake --add-missing
./configure
CC=~/angora/bin/angora-clang ./configure --prefix=`pwd`/install --disable-shared
```

- 修改Makefile

  修改主目录、tests、tinytest、lib、examples等文件夹下Makefile文件中的CC，使CC=~/angora/bin/angora-clang

- 编译两个二进制文件

```
USE_FAST=1 make -j
make install
cp -r target/* ./build_fast

make clean
USE_TRACK=1 make -j
make install
cp -r target/* ./build_track
```

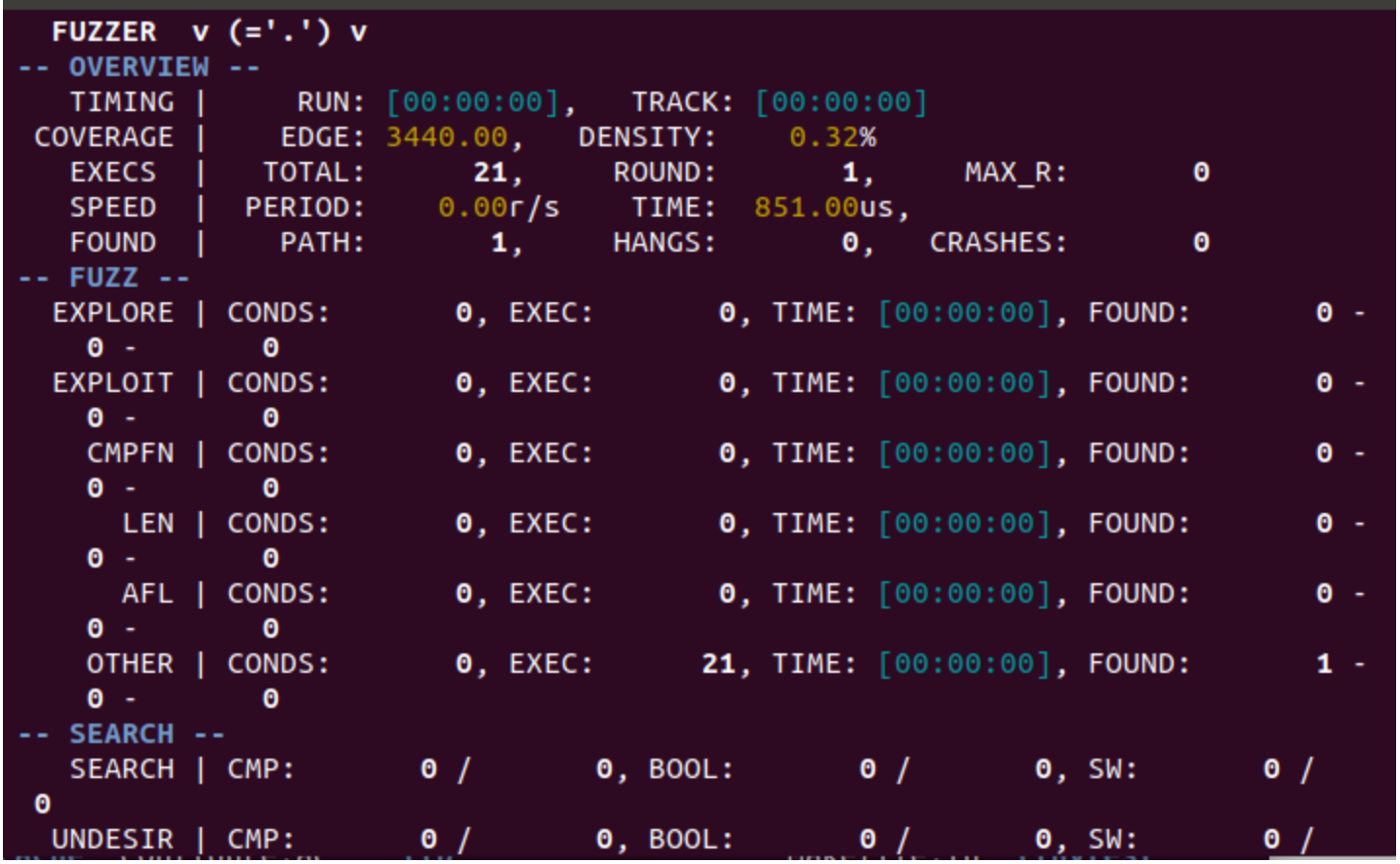
- 集中测试文件

```
mkdir IN
cp tests/testdata/* ./IN
```

- 使用Angora测试testdriver1生成的可执行文件

```
./angora_fuzzer -i ../libconfig/IN -o ../libconfig/output -t ../libconfig/build_track/lib/app.track --
../libconfig/build_fast/lib/app.fast @@
```

- 结果



- 驱动介绍

  i. 驱动一:testdriver1.c,将testdriver1.o和make install生成的lib/libconfig.a链接

  ii. 驱动二:testdriver2.c,将testdriver2.o和make install生成的lib/libconfig.a链接

项目三 libcypher-parser

介绍

libcypher-parser是一个专门为图查询语言Cypher设计的一种解析库以及验证工具。这个解析器由C语言编写，目的是为了在不同的语言中构建工具和库存。libcypher-parser使用的是和Neo4j相同的Cypher实现PEG，PEG是描述语言语法的一个相对新的方法，在解析软件编程语言方面很多优势。

Angora fuzz流程

- git clone

```
git clone https://github.com/cleishm/libcypher-parser
```

- 生成configure文件，Makefile文件

```
./autogen.sh   # 出现error直接搜一般都有解决方案
CC=~/angora/bin/angora-clang ./configure --prefix=`pwd`/install --disable-shared # Angora的生成方法，其他工具需要做适当修改
```

- 安装make时需要的leg包

```
apt-get install leg
```

- make install

  生成的install文件夹下的lib/libcypher-parser.a是库文件

- 驱动介绍

  i. 驱动一:cypher-lint.c,将cypherlint.o和libcypher-parser.a链接起来
  ii. 驱动二:cypher-lint3.c,将cypherlint3.o和libcypher-parser.a链接起来