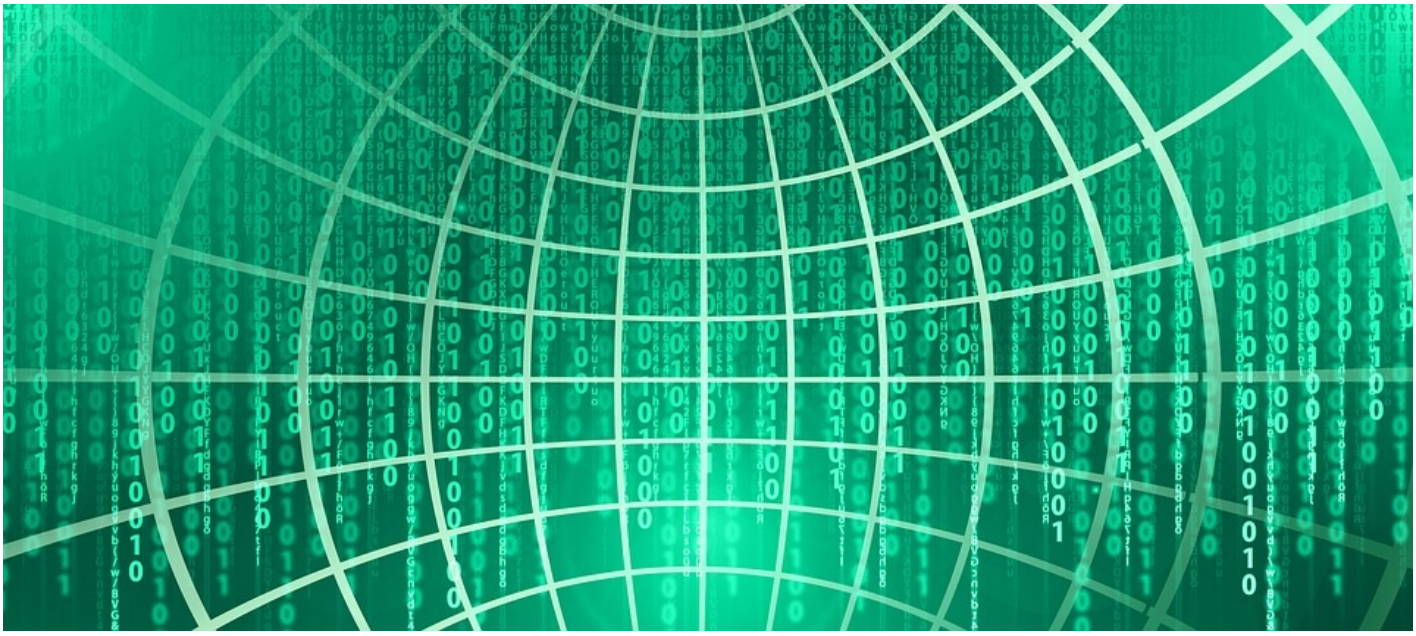
简易的IDAPython脚本

转载

CVGao 于 2020-07-06 21:09:57 发布

文末有干货 “Python高校”，马上关注

真爱，请置顶或星标



看雪论坛作者ID：GitRoy

0x0 写在前面

不知道你是否也和我遇到同样的问题，由于汇编指令不是很熟悉，通常一个样本会分析很久。白天工作，晚上分析样本，这时候遇到一个问题，前一天分析的东西，第二天要花好一段时间去复现昨日的现场。

.idb文件经过多次调试已经被玩坏了，怕分析流程没了，又不敢删除，硬着头皮看已经被搞的不像样的idb。

动态加载的so，分析的时候，自己截图去记录分析到哪里了。

每次定位一些函数（例如init_array、jni_onload）都要自己去算一下偏移，然后G,F2.....

样本中的花指令搞到头大，看100句花指令才能看到有一句有用的指令。

好不容易复现昨天的现场，忽然数据线掉了，或者忽然自己F9略过了关键函数，然后重头再来。

再或者很多拉低效率的问题。

记得曾经分析第一个样本的时候，我第一天分析的东西，都截图记录下来，第二天晚上回去继续分析的时候，我要花大概15-20分钟去复现昨天的现场，如果不小心错过了重要的函数，就得重头再来，后面前前后后浪费了很多时间才搞出来,最后分享会演示的时候，直接被老大鞭策了。

所以才开始了解并编写ida脚本，不过由于网上能看的资料实在是比较少，踩了不少坑，到今天为止终于可以愉快，并且顺畅的写脚本了。所以把流程分享给大家，希望能帮助像我一样的人提高分析效率。

0x1 简述流程

我使用的IDA是泄露版本的7.0然后python版本是2.7

1、首先创建代理脚本.py文件，我这边是roy_hook_proxy.py文件，内容如下：

```
# encoding: utf-8
import sys
sys.path.append('/Users/roy/Documents/PycharmProjects/roytool')
from royhook import RoyHook


def PLUGIN_ENTRY():
    return RoyHook()
```

这样好处显而易见，直接把代理文件丢进插件目录，然后引入我们插件的module就好了，这样就不会直接操作ida的插件目录，不容易出问题。我这边真正写插件的目录就是roytool

2、然后将roy_hook_proxy.py文件拷贝到ida插件目录

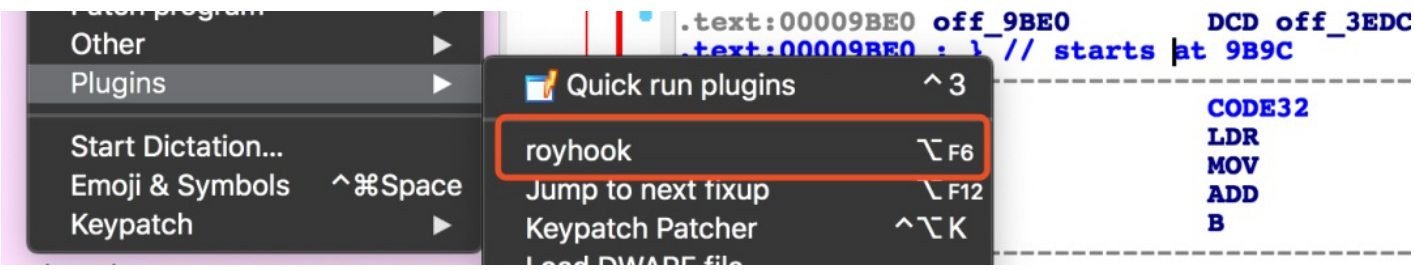
我这边目录/Applications/ida.app/Contents/MacOS/plugins。

3、编写插件主入口

下面我们就可以在我们之前append的module（roytool）下编写脚本了。

```
import sys
# 引入ida提供给我们的api
import idaapi
# 引入pyqt，编写交互界面
from PyQt5 import QtWidgets


# 这里一定要继承ida提供的插件的base类
class RoyHook(idaapi.plugin_t):
    flags = idaapi.PLUGIN_KEEP
    comment = "royhook a ida pro plugin"
    help = ""
    # ida插件的名字
    wanted_name = "royhook"
    # ida插件的快捷键
    wanted_hotkey = "Alt+F6"
    windows = None


    def __init__(self):
        super(RoyHook, self).__init__()
        flags = idaapi.PLUGIN_KEEP
        pass


    # 脚本初始化的时候调用
    def init(self):
        return idaapi.PLUGIN_OK


    # 初始化后开始运行的时候调用
    def run(self, arg):
        idaapi.require('view')
        idaapi.require('view.main_view')
        main_window = view.main_view.MainView()
        if self.windows is None or not self.windows.isVisible():
            self.windows = QtWidgets.QMainWindow()
            main_window.setupUi(self.windows)
            self.windows.showNormal()
        pass
    # 脚本结束的时候调用
    def term(self):
        return idaapi.PLUGIN_OK
```

打开ida直接验证效果，如果上面的代码没有问题，Edit-Plugins。



我们会看到上面图中的样子，点击我们的插件（或者用快捷键）就会运行init、run函数中的代码了。

到此为止，我们插件的基本流程就算完事了。下面继续介绍。

4、实时更新插件代码

在不使用idaapi.require(module)的情况下，每次更新插件代码，ida都要大退重启，才能生效，这时候就需要idaapi.require。

ex:

```
idaapi.require('view')
idaapi.require('view.logsaver_view')
```

注意：既然用了idaapi.require 就不要使用import功能了，不然代码依然不会实时更新。

0x2 提高效率

1、快速定位某个函数

这里举个例子，比如我们要快速定位到init_array函数。

```
def goInitarray(self):
    # _get_modules是idc提供的接口，如其名
    for module in idc._get_modules():
        # 遍历所有module，找到linker
        module_name = module.name
        if 'linker' in module_name:
            print 'linker address is ' + str(hex(module.base + 0x2464))
            # 0x2464是Android某个版本的init_array的偏移地址，
            # jumpto可以直接跳转到目标地址
            idc.jumpto(module.base + 0x2464)
            # 在init_array上下个断点
            idc.add_bpt(module.base + 0x2464, 1)
            # makecode更不用说了，相当于C
            idaapi.auto_make_code(module.base + 0x2464)
```

通过上面的代码就可以直接定位到init_array了，真滴是又简单，又方便。

我在这里可能至少能剩下1-2分钟的时间/每次调试，而且这里只是举例init_array, 我们也可以这样去调试我们要调试的函数。

2、保存日志、函数名字

我们分析的过程中，会在汇编指令做个中记录日志，比如某个寄存器的值了，或者某个函数是做什么的，再或者，我们会直接更改函数的名字，方便每次查看。

这时候，为了效率，或者说能快速的复现上次调试的现场，我们就可以用这个功能，代码也非常简单。

```
# 通过起始地址，终止地址，以及偏移地址去保存日志
def saveDebugMessage(self):
    # create file first
    # 用个轻量级的存储shelve
    f = shelve.open(self.id)
    # 保存日志的起始地址
    addr_start = int(self.address_start, 16)
    # 保存日志的终止地址
    addr_end = int(self.address_end, 16)
    log_dict = {}
    log_dict_list = []
    for num in range(addr_start, addr_end):
        # 获取我们当前地址的日志
        com = idc.GetCommentEx(num, True)
        if com != None:
            #获取函数名
            fun_name = idc.GetFunctionName(num)
            print fun_name
            if fun_name != None and not 'sub' in fun_name:
                log_dict = {'offset': str(num - addr_start), 'msg': str(com), 'funtion_name': str(fun_name)
            else:
                log_dict = {'offset': str(num - addr_start), 'msg': str(com)}
            log_dict_list.append(log_dict)
            pass
    print(log_dict_list)
    # 保存日志
    f['info'] = log_dict_list
    f.close()
 # 通过起始地址即可，会自动判断长度，并且获取偏移地址去设置日志
def loadDebugMessage(self):
    f = shelve.open(self.id)
    data = f['info']
    addr_start = int(self.address_start, 16)
    for num in range(0, len(data)):
        offset = data[num]['offset']
        msg = data[num]['msg']
        fun_name = data[num]['funtion_name']
        idc.MakeRptCmt(addr_start + int(offset), msg)
        if fun_name is not None and fun_name != '':
            idc.SetFunctionCmt(addr_start + int(offset), fun_name, False)
```

3、过一些简单的花指令

这个我在上一个脱壳流程中写到了，感兴趣的小伙伴可以去看一下。

篇幅问题，就简单的举几个常用的，就是想体现很简单的代码，就能节省我们大量的分析时间，我们可以找到自己效率低的地方，然后用idc接口去提高分析效率。

0x3 使用pyqt交互界面，让脚本更人性化

先看效果：

1、推荐工具Qt Designer工具进行画界面

最后保存为.ui文件,再使用pyuic5 -o xxx.py xxx.ui,就能直接输出.py文件了，我们保存到项目中就可以直接用（具体配置可以自行google）。

2、demo

```
class LogSaver_MainWindow(object):


    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(476, 410)
        self.windows = QtWidgets.QMainWindow()
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(110, 100, 91, 16))
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(110, 170, 91, 16))
        self.label_2.setObjectName("label_2")
        self.textEdit = QtWidgets.QTextEdit(self.centralwidget)
        self.textEdit.setGeometry(QtCore.QRect(210, 100, 131, 21))
        self.textEdit.setObjectName("textEdit")
        self.textEdit_2 = QtWidgets.QTextEdit(self.centralwidget)
        self.textEdit_2.setGeometry(QtCore.QRect(210, 170, 131, 21))
        self.textEdit_2.setObjectName("textEdit_2")
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(100, 240, 113, 51))
        self.pushButton.setObjectName("pushButton")
        self.pushButton_2 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_2.setGeometry(QtCore.QRect(260, 240, 113, 51))
        self.pushButton_2.setObjectName("pushButton_2")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(110, 50, 60, 16))
        self.label_3.setObjectName("label_3")
        self.textEdit_3 = QtWidgets.QTextEdit(self.centralwidget)
        self.textEdit_3.setGeometry(QtCore.QRect(210, 50, 131, 21))
        self.textEdit_3.setObjectName("textEdit_3")
        self.checkBox = QtWidgets.QCheckBox(MainWindow)
        self.checkBox.setGeometry(QtCore.QRect(90, 360, 141, 20))
        self.checkBox.setObjectName("checkBox")
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)


    def retranslateUi(self, MainWindow):
        print('create windows')
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "日志工具"))
        self.label.setText(_translate("MainWindow", "开始地址:"))
        self.label_2.setText(_translate("MainWindow", "结束地址:"))
        self.pushButton.setText(_translate("MainWindow", "存储"))
        self.pushButton_2.setText(_translate("MainWindow", "加载"))
        self.label_3.setText(_translate("MainWindow", "唯一id:"))
        self.checkBox.setText(_translate("Dialog", "是否自动保存函数名"))
        self.pushButton.clicked.connect(self.showDialog)
        self.pushButton_2.clicked.connect(self.showDialogLoad)
```

方便又简单。

OK，到这里就完成了，我相信我里面提到的问题，肯定还有更好的解决方案，不一定要写脚本，但是由于自己经验少，可能只能通过脚本来完成，我分享这个是希望告诉像之前的我一样的小伙伴，工欲善其事，必先利其器！最后谢谢大家的观看！

可以扫码加我微信，大量Python，和AI相关资源，欢迎加我交流

开源
我深夜用 Python 跑神经网络，只为关掉台灯！
阿里开源MNNKit：基于MNN的移动端深度学习SDK，支持安卓和iOS

Facebook开源算法代码库PySlowFast，轻松复现前沿视频理解模型

Facebook 开源的 Python 预测工具，用起来太方便了

基于TensorFlow 2.0的中文深度学习开源书来了！GitHub趋势日榜第一，斩获2K+星腾讯正式开源图计算框架 Plato，十亿级节点图计算
工具最靠谱的Pycharm 汉化安装+ 破解详细教程！
Python数据分析、挖掘常用工具Python 最强 IDE 详细使用指南！一款 Python 自动抢票神器，收藏起来回家不愁！
Python新工具：用三行代码提取PDF表格数据
实践和数据分析Python 开发植物大战僵尸游戏用 Python 来找合适的妹子一键分析你上网行为，看你是在认真工作还是摸鱼Python给照
10个经典的小技巧：快速用 Python 进行数据分析
使用 Python 进行微信好友分析爬虫我给曾经暗恋的初中女同学，用Python实现了她飞机上刷抖音
为了能早点买房，我用 Python 预测房价走势！
被女朋友三番五次拉黑后，我用 Python 写了个“舔狗”必备神器
谁偷偷删了你的微信？别慌！Python 揪出来为了给女友挑合适的内衣，我用 Python 爬了天猫内衣店的数据Python爬完数据后，我终于