Umetrip 逆向记录

环境

1. 主机：win10
2. 手机：Pixel 4 ， Android 10
3. APP版本：V7.1.6

工具

IDA、JADX、Frida、Wireshark

逆向思路

流量抓包分析

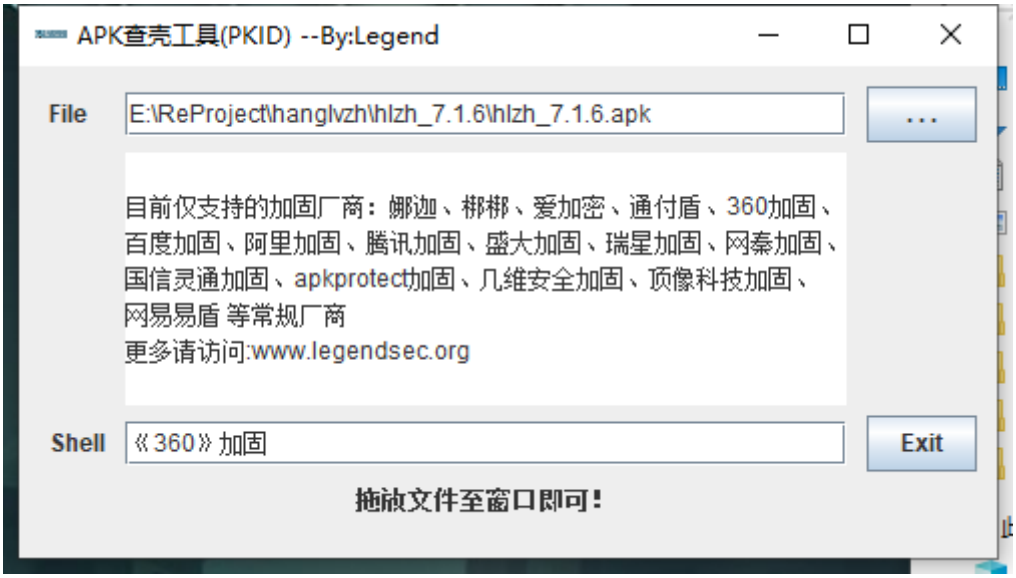
流量为HTTP明文形式，可直接抓取，如下图：

```
POST /gateway/api/umetrip/native?encrypt=1 HTTP/1.1
Accept-Language: zh-CN,zh;q=0.8
User-Agent: okhttp-okgo/jeasonlzy
rcver: AND_a01_06.62.0113
rpid: 1000000
rpver: 5.0
rcuuid: 2e325a87a9b5c4bd4803200eb9c5f1b92
Content-Serialize: pb
transactionID: 2e325100000001615432104717
Content-Type: application/octet-stream
Content-Length: 684
Host: 223.70.194.3
Connection: Keep-Alive
Accept-Encoding: gzip

OCuk23z3JIK0dbN2N7XjW41v8w59NcdEjPapNQZwjfhCMJq4On5pbgOlGenvtuYBk6ds0jhx4ALkDgZGd5gRoLapk3byk3X2lO42FnZ1Qn4qRNwO
bzS2DF1ewqc2SqNfCWoTu0OYv/c52xLpPcQKPiMp6bh2+sZsVNExTcvsL4Jz3XcQoSOTcwcaMRFP+A2k/IPWkBfPPuNfQx+UP/03zFk4AZsKK5l9
+5YjiJYkPtsk4M5EeSwlNSwFD+JHnQ4md7joRQbw4e2SUE3UYDyoGhMKY79GWUd8cwd82sMfVA1qhurAO+rk7bePcsxGTTrFyJtETy3mHJ7GthjB
Ia3AUvqhnpe5nsMe3SxLqb9pdd9r3NSO8eAaVgfqLLyaaDmr2i
+mfGwT8toqVsgIDAcPlF8IVUuQSjUrtQbTrd8vLlKPB1O4E9V0jk55f6IujLsc22eABi15zJGUjGf89IilkP31DrT10VxKQFuJqDxj2PyTKIJS+F
+yCZIG1IciO9NERMQ83+Kkkef2CkRCSLqo6p6pUk49ya5WeX9raKZZ16aOMcBJnVuPI4Qniu4EpIg+/qADlOvWyi1ZD3zwxq2F/KcWgSxEcvsnk=
Date: Thu, 11 Mar 2021 03:08:24 GMT
Server: Apache-Coyote/1.1
X-Application-Context: gateway:8060
Content-Serialize: pb
Content-Encoding: gzip
Content-Length: 220
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Set-Cookie: X-LB=2.1c.1d.443301ed.50; Expires=Thu, 11 Mar 2021 04:08:23 GMT; Path=/

..........b.`.bP.0202.50.54T04.2..22Q.000.b74.....\..9.,..QP.a..].....R..L.-..-.L.M.RL,.....R.,.M...,..r2JJ
...3s..JsSK.2....s.......CsS=s..S}..3....2..2sR.+.KRs..SK..<.J...}#....dc..4s3 al...R..j[.Q[\.X..,......I.b....P
```

其中上行数据的Body部分加密，需要对其解密

使用OKHTTP框架

查壳、脱壳



360加固，使用普通脱壳脚本即可

代码分析

JAVA层

通过明文http流量里的字段定位到以下类：

> com.umetrip.android.umehttp.ParamBuilder

通过buildRequestParam方法追溯到加解密的函数

> com.umetrip.android.umehttp.RequestBodyBuilder.a(com.lzy.okgo.model.HttpHeaders, java.lang.Object, java.lang.String, java.lang.String, boolean)

```
public byte[] a(HttpHeaders httpHeaders, Object obj, String str, String str2, String str3) {
    C2sBodyWrap c2sBodyWrap;
    String sub_0515;
    boolean a2 = HttpConstants.a(str2);
    if (a2) {
        c2sBodyWrap = new C2sBodyWrapPB();
    } else {
        c2sBodyWrap = new C2sBodyWrap();
    }
    c2sBodyWrap.init(str2);
    try {
        c2sBodyWrap.rpid = str;
        c2sBodyWrap.rkey = a();
        c2sBodyWrap.rsid = a(c2sBodyWrap.rsid, str3);
        c2sBodyWrap.netType = a(this.f);
        if (this.h.n() != null) {
            c2sBodyWrap.latitude = this.h.n().a();
            c2sBodyWrap.longitude = this.h.n().b();
        }
        long currentTimeMillis = System.currentTimeMillis();
        String a3 = a(c2sBodyWrap.rcuuid, str, currentTimeMillis);
        c2sBodyWrap.lastTransactionID = this.g.c();
        c2sBodyWrap.transactionID = a3;
        httpHeaders.put("transactionID", a3);
        this.g.a(a3);
        String d2 = this.g.d();
        if (!TextUtils.isEmpty(d2)) {
            c2sBodyWrap.lastReqTime = d2;
        }
        this.g.c(String.valueOf(currentTimeMillis));
        if (a2) {
            if (obj != null) {
                ((C2sBodyWrapPB) c2sBodyWrap).setRequestBody(ProtostuffSerialization.a(obj));
                XlogUtil.a("OkHttp_Request_Str data:", 11, Convert.a(obj), new Object[0]);
            }
            XlogUtil.a("OkHttp_Request_Str:", 11, Convert.a(c2sBodyWrap), new Object[0]);
            sub_0515 = UmeJni.sub_0515(this.f, Base64.a(ProtostuffSerialization.a(c2sBodyWrap)).replace(IOUtils.LINE_SEPARATOR_WINDOWS, ""));
        } else {
            c2sBodyWrap.rparams = obj;
            String a4 = Convert.a(c2sBodyWrap);
            sub_0515 = UmeJni.sub_0515(this.f, a4);
            XlogUtil.a("OkHttp_Request_Str before:", 11, a4, new Object[0]);
        }
        return sub_0515.getBytes("UTF-8");
    } catch (Exception e2) {
        XlogUtil.a("OkHttp_error", 11, "setRequestParameter", e2);
        return null;
    }
}
```

sub_0515为native函数

```
public class UmeJni {
    public static native String sub_0515(Object obj, String str);

    public static native String sub_0516(Object obj, String str);

    public static native String sub_0517(Object obj, String str);
}
```

第一个参数当前的context，第二个参数为需要加密的参数

Native层

native 层sub_0515函数被ollvm，无法直接分析，于是采用unidbg模拟运行方式，查看执行流程。unidbg代码如下：

```
package com.umetrip.android.msky.app;

import com.github.unidbg.AndroidEmulator;
import com.github.unidbg.Module;
```

```
import com.github.unidbg.linux.android.AndroidEmulatorBuilder;
import com.github.unidbg.linux.android.AndroidResolver;
import com.github.unidbg.linux.android.SystemPropertyHook;
import com.github.unidbg.linux.android.SystemPropertyProvider;
import com.github.unidbg.linux.android.dvm.*;
import com.github.unidbg.linux.android.dvm.array.ByteArray;
import com.github.unidbg.memory.Memory;

import org.apache.commons.codec.binary.Base64;

import java.awt.geom.RectangularShape;
import java.io.File;
import java.io.IOException;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.List;
//import king.trace.GlobalData;
//import king.trace.KingTrace;

public class umetrip extends AbstractJni{
    private final AndroidEmulator emulator;
    private final VM vm;
    private final Module module;

    umetrip() {
        emulator =
AndroidEmulatorBuilder.for32Bit().setProcessName("com.umetrip.android.msky.app").build();
 // 创建模拟器实例，要模拟32位或者64位，在这里区分
        final Memory memory = emulator.getMemory(); // 模拟器的内存操作接口
        memory.setLibraryResolver(new AndroidResolver(23)); // 设置系统类库解析
        vm = emulator.createDalvikVM(new File("E:\\unidbg-0.9.5\\unidbg-
android\\src\\test\\resources\\example_binaries\\umetrip\\hlzh_7.1.6.apk"));
        vm.setVerbose(true);

        DalvikModule dm = vm.loadLibrary(new File("E:\\unidbg-0.9.5\\unidbg-
android\\src\\test\\resources\\example_binaries\\umetrip\\libumejni.so"), true);

        vm.setJni(this);
        module = dm.getModule();
//        emulator.traceCode(module.base, module.base + module.size);

        System.out.println("call JNIOnLoad");
        dm.callJNI_OnLoad(emulator);
    }
    public void callSub_0515() {
        List<Object> list = new ArrayList<>(10);
        String data = "
{\"lastReqTime\":\"8151\",\"lastTransactionID\":\"1267d11000331602210574619\",\"latitude\"
 +
                "
"\"rchannel\":\"10000025\",\"rcuuid\":\"1267df7f194d74c5f922c5295499b92ee\"," +
                    "\"rcver\":\"AND_a01_06.51.0914\",\"rkey\":\"2020-10-09 10:30:46
8000\"," +
                    "\"rparams\":
{\"mobile\":\"13188886666\",\"passWord\":\"123456789abcdef\",\"rttimestamp\":0,\"validateC
```

```
 +
"\"rpid\":\"1100033\",\"rpver\":\"3.0\",\"rsid\":\"\",\"transactionID\":\"1267d11000331602

        DvmObject context = vm.resolveClass("android/content/Context").newObject(null);
        list.add(vm.getJNIEnv()); // 第一个参数是env
        list.add(0); // 第二个参数，实例方法是jobject，静态方法是jclazz，直接填0，一般用不
到。
        list.add(vm.addLocalObject(context));
        list.add(vm.addLocalObject(new StringObject(vm, data)));
        //trace code
//        GlobalData.ignoreModuleList.add("libc.so");
//        GlobalData.ignoreModuleList.add("libhookzz.so");
//        GlobalData.watch_address.put(0x4001259b,"");
//        GlobalData.is_dump_ldr=true;
//        GlobalData.is_dump_str=true;
//        KingTrace trace=new KingTrace(emulator);
//        trace.initialize(1,0,null);
//        emulator.getBackend().hook_add_new(trace,1,0,emulator);
//console debugger
        emulator.attach().addBreakPoint(module.base + 0x0000B805);
        module.callFunction(emulator, 0x0000B805, list.toArray());

    }

    public static void main(String[] args) {
        umetrip test = new umetrip();
        test.callSub_0515();
    }

    @Override
    public boolean callStaticBooleanMethod(BaseVM vm, DvmClass dvmClass, DvmMethod
dvmMethod, VarArg varArg) {
        switch (dvmMethod.getSignature()) {
            case "com/ume/android/lib/common/storage/PreferenceData->getPFlag()Z":
                return true;
        }
        return super.callStaticBooleanMethod(vm, dvmClass, dvmMethod, varArg);

    }

    @Override
    public DvmObject<?> getStaticObjectField(BaseVM vm, DvmClass dvmClass, String
signature) {
        switch (signature) {
            case "com/umetrip/android/msky/app/BuildConfig-
>uWyMrFzw:Ljava/lang/String;":
                return new StringObject(vm, "L_2QCh>");

        }
        return super.getStaticObjectField(vm, dvmClass, signature);
    }
}
```

首先定位到AES和RC4算法key的生成函数：

> sub_7BCC sub_8154 sub_84C8 sub_8974

这些函数经过ollvm处理，第一个参数为固定字符串，第二个参数为返回值

可以通过AndroidEmu或者unidbg还原算法

AndroidEmu 脚本如下：

```
from unicorn import *
from androidemu.emulator import Emulator
from UnicornTraceDebugger import udbg
from unicorn.arm_const import *
from androidemu.java.helpers.native_method import native_method
from androidemu.utils import memory_helpers
import binascii, re

import logging
import sys
import zipfile, os, re, shutil

# Configure logging
logging.basicConfig(stream=sys.stdout,
                    level=logging.DEBUG,
                    format="%(asctime)s %(levelname)7s %(name)34s | %(message)s")
logger = logging.getLogger(__name__)


@native_method
def __aeabi_memclr(mu, addr, size):
    print('__aeabi_memclr(%x,%d)' % (addr, size))
    mu.mem_write(addr, bytes(size))


emulator = Emulator()
emulator.modules.add_symbol_hook('__aeabi_memclr4',
emulator.hooker.write_function(__aeabi_memclr) + 1)
emulator.modules.add_symbol_hook('__aeabi_memclr',
emulator.hooker.write_function(__aeabi_memclr) + 1)
libmod = emulator.load_library('lib/libc.so', do_init=False)
libmod = emulator.load_library('lib/libdl.so', do_init=False)
libmod = emulator.load_library('lib/libumejni.so', do_init=False)
print(libmod.base)

image1 = 0xf200000
image_size1 = 0x10000 * 3
emulator.mu.mem_map(image1, image_size1)
emulator.mu.mem_write(image1, 'h[+_{qDGiXoYjiRHfjo_lU'.encode('utf-8'))

image2 = 0xf300000
image_size2 = 0x10000 * 3
emulator.mu.mem_map(image2, image_size2)
```

```
try:

    dbg = udbg.UnicornDebugger(emulator.mu, mode=1)
    # dbg.add_bpt(0xcbc73f24)
    emulator.call_native(0xCBC6C000 + 0x9328 + 1, image1, image2)

    print(emulator.mu.mem_read(image2, 32))
    # emulator.call_symbol('sub_7f24',image1, image2)
except UcError as e:
    list_tracks = dbg.get_tracks()
    for addr in list_tracks[-100:-1]:
        print(hex(addr - 0xcbc6c000))
    print(e)
```

然后根据打印出来的寄存器值一步一步还原就好了