gun

一道 android 题目，根据 libSecShell.so 和 assets 目录下的 classes0.jar 大致可以判断加了 18 年版本的梆梆免费版壳，这里选择 hook ClassLinker::DefineClass 来脱壳。
apk 在低于 android 9.0(api 28) 的设备下无法安装，即使修改 android:minSdkVersion 后也不能正常运行，看来是使用高版本 api 编译的。
但是我手里的 root 环境最高只有 android 6.0 的，所以这次我们必须免 root 脱壳。
我选择的是 frida-gadget （ frida 框架的免 root 加载方式，详细见 https://frida.re/docs/gadget/ ）
** 下面开始详细步骤： **

1. 在 github 上下载，与你的 frida 对应版本，且与 app 的位数一致（比如这道题是 32 位的就下载 arm32 版本的）的 frida-gadget ,把他改名为 libgadget.so 并放入 apk 的 lib 目录下。
2. 在 dex 中找到合适的时机，调用 loadLibrary 加载 libgadget.so ， 这里可以找到原 app 加载 so 的地方，然后修改他让我们的 so 也一起加载。
   在代码里搜索 loadLibrary 。

```
const-string v0, "SecShell"

invoke-static {v0}, Ljava/lang/System;->loadLibrary(Ljava/lang/String;)V
```

改成

```
const-string v0, "gadget"

invoke-static {v0}, Ljava/lang/System;->loadLibrary(Ljava/lang/String;)V

const-string v0, "SecShell"

invoke-static {v0}, Ljava/lang/System;->loadLibrary(Ljava/lang/String;)V
```

3. 这时候直接运行会以 frida 的 wait 模式启动，就是 app 会一直白屏等待我们电脑端连接，但是壳会开一个线程做校验，在我们连接之前就已经把应用退出了。所以我们让 libgadget.so 加载时，直接运行我们的脱壳脚本，在闪退之前把 dex 写出来。

```
{
  "interaction": {
    "type": "script",
    "path": "/sdcard/hook.js"
  }
}
```

这段代码保存到 lib 目录下的 libgadget.config.so 中（名字前半部分必须与 so 同名，后面加上 config ， 加 so 后缀的原因是这样安装的时候才会把它也复制进 /data/app/xxxx/lib 目录）
4. 脱壳脚本（修改自 https://github.com/lasting-yang/frida_dump/ ， 膜拜大佬）

```
function dump_dex() {
    var libart = Process.findModuleByName("libart.so");
    var addr_DefineClass = null;
    var symbols = libart.enumerateSymbols();
    for (var index = 0; index < symbols.length; index++) {
        var symbol = symbols[index];
        var symbol_name = symbol.name;
        if (symbol_name.indexOf("ClassLinker") >= 0 &&
            symbol_name.indexOf("DefineClass") >= 0 &&
            symbol_name.indexOf("Thread") >= 0 &&
            symbol_name.indexOf("DexFile") >= 0) {
            console.log(symbol_name, symbol.address);
            addr_DefineClass = symbol.address;
        }
    }//根据 DefineClass 函数的特征，在符号表中找到 DefineClass 函数的地址
    var dex_maps = {};
    var dex_count = 1;

    console.log("[DefineClass:]", addr_DefineClass);
    if (addr_DefineClass) {
        Interceptor.attach(addr_DefineClass, {//hook DefineClass
            onEnter: function(args) {
                var dex_file = args[5];
                var base = ptr(dex_file).add(Process.pointerSize).readPointer();
                var size = ptr(dex_file).add(Process.pointerSize +
Process.pointerSize).readUInt();

                if (dex_maps[base] == undefined) {
                    dex_maps[base] = size;
                    var magic = ptr(base).readCString();
                    if (magic.indexOf("dex") == 0) {//把 dex dump 进sd卡目录
                        var dex_dir_path = "/sdcard/dumped_dex";
                        var dex_path = dex_dir_path + "/class" + (dex_count == 1 ?
"" : dex_count) + ".dex";
                        console.log("[find dex]:", dex_path);
                        var fd = new File(dex_path, "wb");
                        if (fd && fd != null) {
                            dex_count++;
                            var dex_buffer = ptr(base).readByteArray(size);
                            fd.write(dex_buffer);
                            fd.flush();
                            fd.close();
                            console.log("[dump dex]:", dex_path);
                        }
                    }
                }
            },
            onLeave: function(retval) {}
        });
    }
}
```

```
dump_dex();
```

写（抄）完代码后，保存到 /sdcard/hook.js 。
5. 最后一步，为 app 增加 sd卡 权限，因为我们的脚本和 dump dex 的目录都在 sd卡 上。
打开 AndroidManifest.xml 。
uses-permission 加上这几项。

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
```

application 标签下加上（貌似是 android 10 的新要求）

```
android:requestLegacyExternalStorage="true"
```

打包签名安装运行，就可以在 /sdcard/dumped_dex 目录下找到 dex 啦。

把 dex 改名成 classes.dex~classes5.dex ，以 apk 模式压缩成 zip，用 jeb 进行分析。

```
public final class MainActivity extends j {
    public MainActivity() {
        super();
    }

    public void onCreate(Bundle arg2) {
        super.onCreate(arg2);
        ((j)this).setContentView(0x7F0B001C);
        new Thread(MainActivity$a.a).start();
    }
}
```

MainActivity 的 onCreate 中开启了一个线程。

```
public final void run() {
        try {
            new ol().run();
            new jm().run();
            new fn().run();
            new aj().run();
            new xl().run();
            new yn().run();
            new qm().run();
            new ao().run();
```

```
            new vn().run();
            new wm().run();
            new wj().run();
            new xn().run();
            new jk().run();
            new nl().run();
            new co().run();
            new um().run();
            new nn().run();
            new ll().run();
            new ul().run();
            new uj().run();
            new ln().run();
            new el().run();
            ......//省略几百行
        }
}
```

在这个线程中调用了几百个类的 run 方法。

```
    public void run() {
        ArrayList v0 = new ArrayList();
        ArrayList v14 = fd.h("bullet", "name", "q", "value");
        b v13 = jr.k;
        v0.add(b.a(v13, "bullet", 0, 0, " \"\':;<=>@[]^`{}|/\\?#&!$(),~", false,
false, true, false, null, 91));
        fr v0_1 = fd.j(v14, b.a(v13, "q", 0, 0, " \"\':;<=>@[]^`{}|/\\?#&!$(),~",
false, false, true, false, null, 91), v0, v14);
        a v1 = new a();
        v1.a("hgame.vidar.club", new String[]
{"sha256/ocfaPpOi8wBS01tMzoT6f+q+zF7ufbbxSe2wQUcpqXY="});
        v1.a("hgame.vidar.club", new String[]
{"sha256/GI75anSEdkuHj05mreE0Sd9jE6dVqUIzzXRHHlZBVbI="});
        v1.a("hgame.vidar.club", new String[]
{"sha256/GI75anSEdkuHj05mreE0Sd9jE6dVqUIzzXRHHlZBVbI="});
        rq v1_1 = v1.b();
        mr$a v2 = fd.c(v1_1, "certificatePinner");
        mp.a(v1_1, v2.q);
        v2.q = v1_1;
        fd.i("https://hgame.vidar.club", v0_1, 19530, new mr(v2)).d();
    }
}
```

```
    public void run() {
        ArrayList v0 = new ArrayList();
        ArrayList v14 = fd.h("bullet", "name", "e", "value");
        b v13 = jr.k;
        v0.add(b.a(v13, "bullet", 0, 0, " \"\':;<=>@[]^`{}|/\\?#&!$(),~", false,
```

```
false, true, false, null, 91));
        fr v0_1 = fd.j(v14, b.a(v13, "e", 0, 0, " \"\':;<=>@[]^`{}|/\\?#&!$(),~",
false, false, true, false, null, 91), v0, v14);
        a v1 = new a();
        v1.a("hgame.vidar.club", new String[]
{"sha256/ocfaPpOi8wBS01tMzoT6f+q+zF7ufbbxSe2wQUcpqXY="});
        v1.a("hgame.vidar.club", new String[]
{"sha256/GI75anSEdkuHj05mreE0Sd9jE6dVqUIzzXRHHlZBVbI="});
        v1.a("hgame.vidar.club", new String[]
{"sha256/GI75anSEdkuHj05mreE0Sd9jE6dVqUIzzXRHHlZBVbI="});
        rq v1_1 = v1.b();
        mr$a v2 = fd.c(v1_1, "certificatePinner");
        mp.a(v1_1, v2.q);
        v2.q = v1_1;
        fd.i("https://hgame.vidar.club", v0_1, 0x75F4, new mr(v2)).d();
    }
}
```

这些方法的代码几乎一模一样，只有两处不同，一处是第三行的"name"后面的字符串，第二处是倒数第三行的整数。
而虽然类名被混淆了，但是根据 "certificatePinner" 能判断出这是 okhttp3 的锁定证书操作，对应代码

```
CertificatePinner certificatePinner = new CertificatePinner.Builder()
        .add("hgame.vidar.club",
"sha256/GI75anSEdkuHj05mreE0Sd9jE6dVqUIzzXRHHlZBVbI=")
        .add("hgame.vidar.club",
"sha256/GI75anSEdkuHj05mreE0Sd9jE6dVqUIzzXRHHlZBVbI=")
        .add("hgame.vidar.club",
"sha256/GI75anSEdkuHj05mreE0Sd9jE6dVqUIzzXRHHlZBVbI=")
        .build();
```

目的是防抓包，所以这道题一定有关键的网络请求。
fd.i 方法中传入了 url ，可能含有请求动作，反编译一下看看。

```
    public static qq i(String arg1, fr arg2, long arg3, mr arg5) {
        or$a v0 = new or$a();
        v0.f(arg1);
        v0.d(((pr)arg2));
        or v1 = v0.a();
        Thread.sleep(arg3);
        return arg5.a(v1);
    }
```

看看 or$a 类

```
public final class or {
    public class a {
        public jr a;
        public String b;
        public ir$a c;
        public pr d;
        public Map e;

        public a() {
            super();
            this.e = new LinkedHashMap();
            this.b = "GET";
            this.c = new ir$a();
        }
        .......
```

果然没错，这就是 okhttp3 的 Request 类

```
public final class Request {
public static class Builder {
    HttpUrl url;
    String method;
    Headers.Builder headers;
    RequestBody body;


    public Builder() {
      this.method = "GET";
      this.headers = new Headers.Builder();
    }
    .......
```

所以 fd.i 的真正操作就是 Thread.sleep(arg3) 后发送请求，而请求的内容经过分析，大概是 name=bullet&value=xxx ，给线程延时的目的则是要创造出发送的先后顺序。所以，只要按照顺序得到发送的所有 value ，拼在一起，就能得到 flag 了。

知道逻辑了，求解方法就很多了，可以 frida 绕过证书验证后抓包，可以 frida 直接 hook okhttp3 的请求方法模拟抓包，可以改掉证书验证后重打包。但是在没 root 的环境和 app 的签名验证下，都不如直接静态脚本来的快。

本来想试试 jeb 的脚本的，奈何问题太多（不同版本 api 换来换去，太多类的时候枚举类会爆内存等等等等），还是有空的时候再学吧。反正我们需要操作的类反编译出来都差不多，那就直接导出 java 文件吧。

导出后，用 python 脚本提取 value 并排序。

```
import os

#遍历文件，查找字符串的位置，提取 value 和延迟发包的时长
```

```
pattern1 = 'fd.h("bullet", "name", "'
pattern2 = 'fd.i("https://hgame.vidar.club", v0_1, '
pattern3 = ', new mr('
keyVal = {}

def findPatt(fullPath):
    global keyVal
    file = open(fullPath,'r')
    try:
        content = file.read()
        file.close()
    except:
        file.close()
        return

    key=''
    orderNum=0
    pos = content.find(pattern1)
    if pos > 0:
        key = content[pos+len(pattern1):pos+len(pattern1)+1]
    else:
        return
    pos = content.find(pattern2)
    pos2 = content.find(pattern3)
    try:    #反编译出来的整数有十进制也有十六进制的
        orderNum = int(content[pos+len(pattern2):pos2])
    except:
        orderNum = int(content[pos+len(pattern2):pos2],16)
    keyVal[orderNum] = key
    return

def getDic(dir_):
    list_ = os.listdir(dir_)
    for i in range(0, len(list_)):
        path = os.path.join(dir_, list_[i])
        if os.path.isfile(path):
            findPatt(path)
    return

getDic('./smali')
print(keyVal)
for i in sorted(keyVal):#python排序真是太方便啦
    print(keyVal[i],end='')
```

运行脚本，得到一串文字： Oazsdgmpgmfuaz! ftq eqodqf ar ftq mbbe ue tqdq, ftue otmxxqzsq ue uzebudqp nk NkfqOFR arrxuzq omybmusz, ftq rxms ue tsmyq{dQh3x_y3_nk_z4F1h3_0d_zi7I0dw}

最后一步，凯撒密码解密： Congruaduation! the secret of the apps is here, this challenge is inspired by ByteCTF offline campaign, the flag is hgame{rEv3l_m3_by_n4T1v3_0r_nw7W0rk}

nice!

fake

程序在 init_array 里通过 TracerPid 的值检查了调试，如果是干净环境就对代码进行一下修改。
使用 ida python 修改汇编：

```
start_addr = 0x401216
save_arr = []

for i in range(0x43f):
    bt = Byte(start_addr+i)
    save_arr.append(bt)
    PatchByte(start_addr+i,bt^Byte(0x409080+i))
print save_arr
```

提取密文数组：

```
start_addr = 0x401243
end_addr = 0x4013ab
cipher = []
while start_addr!=end_addr:
    cipher.append(GetOperandValue(start_addr,1))
    start_addr = NextHead(start_addr)
print cipher
```

exp:

```
from z3 import *

arr = 'hgame{@_FAKE_flag!-do_Y0u_konw_SMC?}'
flag = []
for i in range(36):
    flag.append(BitVec('flag_%d' % i, 32))
ans = [BitVecVal(0,32)]*36
cipher = [55030, 61095, 60151, 57247, 56780, 55726, 46642, 52931, 53580, 50437,
50062, 44186, 44909, 46490, 46024, 44347, 43850, 44368, 54990, 61884, 61202,
58139, 57730, 54964, 48849, 51026, 49629, 48219, 47904, 50823, 46596, 50517,
48421, 46143, 46102, 46744]
solver = Solver()
for i in range(6):
    for j in range(6):
        for k in range(6):
            ans[6 * i + j] += BitVecVal(ord(arr[6 * k + j]),32) * flag[6 * i + k]
for l in range(6):
    for m in range(6):
        solver.add(ans[6 * l + m]==cipher[6 * l + m])
flg = ''
```

```
if solver.check()==sat:
    print(solver.model().sorts())
    for i in range(36):
        flg += chr(eval(str(solver.model().eval(flag[i]))))
    print(flg)
```

helloRe3

ce 搜索内存数据，找到存放输入的地址，ida 交叉引用，发现 exe 开了一个线程对输入进行验证。

```
__int64 sub_140095F70()
{
  *(__readgsqword(0x60u) + 2) = 1;
  hThread = CreateThread(0i64, 0i64, StartAddress, 0i64, 4u, 0i64);
  if ( !hThread )
  {
    MessageBoxW(0i64, L"ERROR", 0i64, 0);
    ExitProcess(0);
  }
  sub_14008D633();
  return 0i64;
}
```

验证流程并不复杂，首先判断 len(input)==20 ，再对输入按位取反，最后是一个 RC4 加密， hook 出密钥即可。
最后对解密出的结果，进行 按键的编号 到 按键上的字母 的一个映射变换。

```
outTable = '1234567890-+qwertyuiop{}|asdfghjkl;\'zxcvbnm,./'
inTable = ['%c'%i for i in range(21,33)]
inTable += ['%c'%i for i in range(37,50)]
inTable += ['%c'%i for i in range(54,65)]
inTable += ['%c'%i for i in range(66,76)]
inTable = ''.join(inTable)
transTable = str.maketrans(inTable, outTable)

flag = ";:6H'/\x1a\x1f=\x18=J\x18( \x17D\x18)0"
print(flag.translate(transTable).upper())
```