Android_CTF: kgb_messenger

原创

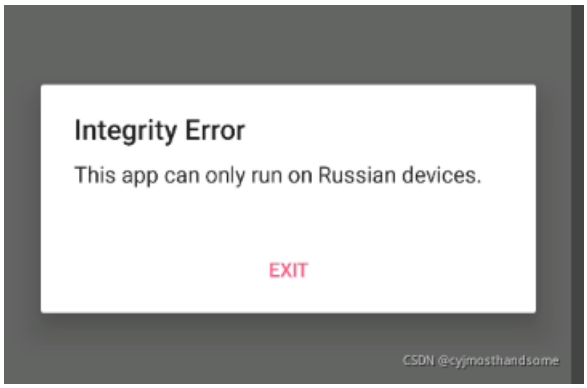
cyjmosthandsome 于 2021-11-10 10:39:23 发布 1569 收藏
分类专栏： 移动安全 文章标签： android CTF 逆向
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/cyjmosthandsome/article/details/121241131
版权


移动安全 专栏收录该内容
4 篇文章 1 订阅
订阅专栏

环境：kgb_messenger.apk，测试机：OnePlus Andorid 9

1. Alerts

安装该apk后，使用发现如下的界面



用 jadx 反编译该 apk，搜索字符串 "Russian", 在MainActivity中发现了以上字符串，且代码逻辑如下

```
    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView((int) R.layout.activity_main);
        String property = System.getProperty("user.home");
        String str = System.getenv("USER");
        if (property == null || property.isEmpty() || !property.equals("Russia")) {
            a("Integrity Error", "This app can only run on Russian devices.");
        } else if (str == null || str.isEmpty() || !str.equals(getResources().getString(R.string.User))) {
            a("Integrity Error", "Must be on the user whitelist.");
        } else {
            a.a(this);
            startActivity(new Intent(this, LoginActivity.class));
        }
    }
```

CSDN @cyjmosthandsome

其中 System.getProperty() 函数用于获取当前系统属性，包括 java版本、操作系统版本等信息，其中可以获得的属性值，参见：JAVA 命令参数详解：-D_枝叶飞扬_新浪博客。

其中 System.getenv() 函数用于获取指定的环境变量的值。

审计代码，发现，会判断 user.home 对应的系统属性的值是否为 Russia, 然后判断 USER 对应的环境变量是否为 R.string.User 对应的字符串的值，如果两个判断都通过，则会启动一个 LoginActivity。

首先想去看一下 R.string.User 对应的字符串是啥，R.string.User 表示在res文件夹中，定义了一个string类型的名为 User 的字符串。于是用apktool 如下命令对该apk解压

```
java -jar apktool.jar d xxx.apk
```

并在res文件夹下使用如下命令搜索 User 字符串

```
F:\Mobile_Security\Android_CTF\kgb_messenger\kgb-messenger\res
$ grep -r "User"
layout/activity_login.xml:    <EditText n1:id="@id/login_username" n1:layout_width="200.0dip" n1:layout_
height="wrap_content" n1:layout_marginTop="15.0dip" n1:hint="Username" n1:layout_below="@id/kgb_logo" n1
:layout_centerHorizontal="true" />
values/public.xml:    <public type="string" name="User" id="0x7f0d0000" />
values/strings.xml:    <string name="User">RkxBR3s1N0VSTDFOR180UkNIM1J9Cg==</string>
```

结合搜索结果来看，果然在strings.xml文件中，定义了一个名为 User 的字符串，其对应的值明显是一个base64编码的字符串，不管三七二十一，先拿去base64解码再说，解码后得到FLAG：FLAG{57ERL1NG_4RCH3R}

2. Login

在1中的MainActivity中，想要进入

```
startActivity(new Intent(this, LoginActivity.class));
```

就必须得进入else分支，于是想到直接去修改Smali文件，删除前两个if-else if的分支check，直接进入第三个else分支，启动LoginActivity。

查看对应MainActivity.java的smali代码(该文件存在于之前使用apktool解压后的文件夹中，存在于对应的smali文件夹中)，对smali代码进行修改，然后重打包成apk。将onCreate()函数中的两个if-else if分支中的内容注释掉，尤其注意要把 :goto_0 return-void 移到函数末尾，否则会直接执行 return-void

```
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 3

    invoke-super {p0, p1}, Landroid/support/v7/app/c;->onCreate(Landroid/os/Bundle;)V

    const v0, 0x7f09001c

    invoke-virtual {p0, v0}, Lcom/tlamb96/kgbmessenger/MainActivity;->setContentView(I)V

    const-string v0, "user.home"

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

    move-result-object v0

    const-string v1, "USER"

    invoke-static {v1}, Ljava/lang/System;->getenv(Ljava/lang/String;)Ljava/lang/String;

    move-result-object v1

    # if-eqz v0, :cond_0

    # invoke-virtual {v0}, Ljava/lang/String;->isEmpty()Z

    # move-result v2

    # if-nez v2, :cond_0
```

```
# const-string v2, "Russia"

# invoke-virtual {v0, v2}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z

# move-result v0

# if-nez v0, :cond_1

# :cond_0
# const-string v0, "Integrity Error"

# const-string v1, "This app can only run on Russian devices."

# invoke-direct {p0, v0, v1}, Lcom/tlamb96/kgbmessenger/MainActivity;->a(Ljava/lang/String;Ljava/lang/S

# :goto_0
# return-void

# :cond_1
# if-eqz v1, :cond_2

# invoke-virtual {v1}, Ljava/lang/String;->isEmpty()Z

# move-result v0

# if-nez v0, :cond_2

# invoke-virtual {p0}, Lcom/tlamb96/kgbmessenger/MainActivity;->getResources()Landroid/content/res/Reso

# move-result-object v0

# const/high16 v2, 0x7f0d0000

# invoke-virtual {v0, v2}, Landroid/content/res/Resources;->getString(I)Ljava/lang/String;

# move-result-object v0

# invoke-virtual {v1, v0}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z

# move-result v0

# if-nez v0, :cond_3

# :cond_2
# const-string v0, "Integrity Error"

# const-string v1, "Must be on the user whitelist."

# invoke-direct {p0, v0, v1}, Lcom/tlamb96/kgbmessenger/MainActivity;->a(Ljava/lang/String;Ljava/lang/S

# goto :goto_0

:cond_3
invoke-static {p0}, La/a/a/a/a;->a(Landroid/content/Context;)V

new-instance v0, Landroid/content/Intent;

const-class v1, Lcom/tlamb96/kgbmessenger/LoginActivity;
```

```
    invoke-direct {v0, p0, v1}, Landroid/content/Intent;-><init>(Landroid/content/Context;Ljava/lang/Class;

    invoke-virtual {p0, v0}, Lcom/tlamb96/kgbmessenger/MainActivity;->startActivity(Landroid/content/Intent

    goto :goto_0

    :goto_0
    return-void
.end method
```

修改完smali文件后，对apk进行重打包和签名，完成后在手机上安装签名后的apk。打开对应APP后，直接进入了LoginActivity对应的登录界面。

3. Social Engineering

这一步是需要输入正确的用户名和密码，完成登录。通过审计代码发现用户名是定义在res文件夹中的一个字符串 "codenameduchess"，密码是一个md5()后的哈希值，直接丢到网站上无法破解成功，官方的writeup中写的是

```
用户名codenameduchess其实是一个动漫里Archer中特工的代号，代号为duchess
```

然后google对应的内容就可以拿到duchess的账号密码：guest。输入对应的账号和密码就可以成功登录，拿到flag。

(这脑洞就离谱？？谁猜得到啊？！)