Godfrey Nolan

Bulletproof Android™

Practical Advice for Building Secure Apps

Bulletproof Android™

Listing 2-11 **Disassembled `initializeLocalProfile()` protected with DexGuard**

```
.method private ?()V
     .registers 7
     .line 0
     iget-object v0, p0, Lcom/example/android/sip/WalkieTalkieActivity;->?:Landroid/
     net/sip/SipManager;
     if-nez v0, :cond _ 5
     .line 116
     return-void
     .line 119
     :cond _ 5
     iget-object v0, p0, Lcom/example/android/sip/WalkieTalkieActivity;->?:Landroid/
     net/sip/SipProfile;
     if-eqz v0, :cond _ c
     .line 120
     invoke-direct {p0}, Lcom/example/android/sip/WalkieTalkieActivity;->?()V
     .line 123
     :cond _ c
     invoke-virtual {p0}, Lcom/example/android/sip/WalkieTalkieActivity;->getBase
     Context()Landroid/content/Context;
     move-result-object v0
     invoke-static {v0}, Landroid/preference/PreferenceManager;->getDefaultShared
     Preferences(Landroid/content/Context;)Landroid/content/SharedPreferences;
     move-result-object v0
     .line 124
     move-object v3, v0
     const-string v1, "namePref"
     const-string v2, ""
     invoke-interface {v0, v1, v2}, Landroid/content/SharedPreferences;->getString
     (Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
     move-result-object v4
     .line 125
     const-string v0, "domainPref"
     const-string v1, ""
     invoke-interface {v3, v0, v1}, Landroid/content/SharedPreferences;->getString(
     Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
     move-result-object v5
     .line 126
     const-string v0, "passPref"
     const-string v1, ""
     invoke-interface {v3, v0, v1}, Landroid/content/SharedPreferences;->getString(
     Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
     move-result-object v3
     .line 128
     invoke-virtual {v4}, Ljava/lang/String;->length()I
     move-result v0
     if-eqz v0, :cond _ 3f
     invoke-virtual {v5}, Ljava/lang/String;->length()I
     move-result v0
     if-eqz v0, :cond _ 3f
     invoke-virtual {v3}, Ljava/lang/String;->length()I
     move-result v0
     if-nez v0, :cond _ 44
     .line 129
     :cond _ 3f
     const/4 v0, 0x3
```

```
invoke-virtual {p0, v0}, Lcom/example/android/sip/WalkieTalkieActivity;->show
Dialog(I)V
.line 130
return-void
.line 134
:cond _ 44
:try _ start _ 44
new-instance v0, Landroid/net/sip/SipProfile$Builder;
invoke-direct {v0, v4, v5}, Landroid/net/sip/SipProfile$Builder;-><init>(Ljava
/lang/String;Ljava/lang/String;)V
.line 135
move-object v4, v0
invoke-virtual {v0, v3}, Landroid/net/sip/SipProfile$Builder;->setPassword
(Ljava/lang/String;)Landroid/net/sip/SipProfile$Builder;
.line 136
invoke-virtual {v4}, Landroid/net/sip/SipProfile$Builder;->build()Landroid/net/
sip/SipProfile;
move-result-object v0
iput-object v0, p0, Lcom/example/android/sip/WalkieTalkieActivity;->?:Landroid/
net/sip/SipProfile;
.line 138
new-instance v0, Landroid/content/Intent;
invoke-direct {v0}, Landroid/content/Intent;-><init>()V
.line 139
move-object v3, v0
const-string v1, "android.SipDemo.INCOMING _ CALL"
invoke-virtual {v0, v1}, Landroid/content/Intent;->setAction(Ljava/lang/String;)
Landroid/content/Intent;
.line 140
const/4 v0, 0x0
const/4 v1, 0x2
invoke-static {p0, v0, v3, v1}, Landroid/app/PendingIntent;->getBroadcast
(Landroid/content/Context;ILandroid/content/Intent;I)Landroid/app/Pending
Intent;
move-result-object v3
.line 141
iget-object v0, p0, Lcom/example/android/sip/WalkieTalkieActivity;->?:Landroid/
net/sip/SipManager;
iget-object v1, p0, Lcom/example/android/sip/WalkieTalkieActivity;->?:Landroid/
net/sip/SipProfile;
const/4 v2, 0x0
invoke-virtual {v0, v1, v3, v2}, Landroid/net/sip/SipManager;->open(Landroid/
net/sip/SipProfile;Landroid/app/PendingIntent;Landroid/net/sip/SipRegistration
Listener;)V
.line 147
iget-object v0, p0, Lcom/example/android/sip/WalkieTalkieActivity;->?:Landroid/
net/sip/SipManager;
iget-object v1, p0, Lcom/example/android/sip/WalkieTalkieActivity;->?:Landroid/
net/sip/SipProfile;
invoke-virtual {v1}, Landroid/net/sip/SipProfile;->getUriString()Ljava/lang/
String;
move-result-object v1
new-instance v2, Lo/?;
invoke-direct {v2, p0}, Lo/?;-><init>(Lcom/example/android/sip/WalkieTalkie
Activity;)V
invoke-virtual {v0, v1, v2}, Landroid/net/sip/SipManager;->setRegistration
Listener(Ljava/lang/String;Landroid/net/sip/SipRegistrationListener;)V
```

```
    :try_end_7c
    .catch Ljava/text/ParseException; {:try_start_44 .. :try_end_7c} :catch
    _7d
    .catch Landroid/net/sip/SipException; {:try_start_44 .. :try_end_7c}
    :catch_89
    .line 161
    return-void
    .line 162
    :catch_7d
    const-string v4, "Connection Error."
    move-object v3, p0
    new-instance v0, Lo/?;
    invoke-direct {v0, v3, v4}, Lo/?;-><init>(Lcom/example/android/sip/WalkieTalkie
    Activity;Ljava/lang/String;)V
    invoke-virtual {p0, v0}, Lcom/example/android/sip/WalkieTalkieActivity;->runOn
    UiThread(Ljava/lang/Runnable;)V
    return-void
    .line 163
    .line 164
    :catch_89
    const-string v4, "Connection error."
    move-object v3, p0
    new-instance v0, Lo/?;
    invoke-direct {v0, v3, v4}, Lo/?;-><init>(Lcom/example/android/sip/Walkie
    TalkieActivity;Ljava/lang/String;)V
    invoke-virtual {p0, v0}, Lcom/example/android/sip/WalkieTalkieActivity;->runOn
    UiThread(Ljava/lang/Runnable;)V
    .line 166
    return-void
.end method.
```

So there really isn't much we can do here to protect our APK. Obfuscation will only protect you from decompiling the APK, not from disassembling it. The good news is that it's a lot harder to understand Smali, so it's going to be orders of magnitude harder to hack than decompiled Java code. It will limit, but not eliminate, any security issues that might be exposed.

Hiding Business Rules in the NDK

Many Android apps contain business logic that may have taken many years to develop, and not understanding the repercussions of others being able to decompile your Android app means essentially giving away all that work for free.

Some examples that I've come across recently are connecting to a Bluetooth device in a car that had a proprietary data stream and connecting to a VoIP server API to make phone calls. The value of both of these apps is based on the business logic to allow the user to easily connect to external systems. The apps are not as attractive anymore if there are other cheaper apps on the marketplace that have copied this code.

The Native Developer Kit (NDK) enables developers to write code as a C++ library. This can be useful if you want to try to hide any business rules code in binary.

Unlike Java code, C++ cannot be decompiled, only disassembled. It's not enough to stop someone from reading the binary, but it does put the Android app's code on a par with Objective-C in iOS devices. Yes, you can still open the file using a hexadecimal editor or a tool like IDA Pro, but no one will be able to decompile back into code as easy to read as the original C++.

We'll cover the NDK in more detail in Chapter 5.

Conclusion

We've seen how easy it is to remove an APK from a phone. We've made several attempts at making it more and more difficult to decompile the code by using increasingly stronger versions of obfuscation. We've seen how someone can manipulate your APK by disassembling it and editing the Smali and then reassembling it to change its behavior. However, none of these completely protect the code. They raise the bar, quite high in some cases, but they all come with risks such that determined hackers with some time on their hands can debug your app and get at your code.

But what if you put it elsewhere? It's perfectly acceptable—with one caveat—to keep your most important code on a backend server. We will look at this in more detail in Chapter 6.

This page intentionally left blank

3

Authentication

In this chapter we look at how some of the authentication mechanisms have failed and what developers have been using to log in to mobile apps that has been more effective.

Secure Logins

Providing a secure login mechanism for your users is harder than on the Web. The trend on mobile devices is to make things as easy as possible for the user. Mobile keyboards are also small, so it's unlikely that someone is going to enter more than six characters to log in to an app.

But if you make it too easy to log in to your app, you run the risk of unauthorized users gaining access to sensitive data by going around this authentication.

The following tokens are common on Android devices as part of the login process:

- Username and password
- Device information, such as DeviceID and AndroidID
- Network information, such as IP address

The classic login of username and password is still the most common authentication on an Android phone.

Many apps ask to remember your username for the next login. This approach isn't recommended because it means the username is typically stored somewhere on the device.

Much worse, however, is when the app asks to remember the password when you log in because there is effectively no login the second time around (see Figure 3-1).

The app is caching the username and password, and it's probably being stored in the shared preferences or a client-side database. The app may also be using device-specific information, such as the AndroidID or an IP address, to log in each subsequent time.