Laboratorium nr 2 – adb, podpisywanie aplikacji i plik apk

1. Narzędzie adb

ADB (Android Debug Bridge) jest narzędziem, które pozwala komunikować się z telefonem przez kabel USB lub WiFi (lub emulatorem). Pozwala na wykonanie wielu operacji takich jak kopiowanie plików, instalacja aplikacji, lista procesów, lista zainstalowanych aplikacji itd., np.:

- `adb devices` – lista dostępnych urządzeń

- `adb install /sciezka-do-pliku.apk` – instalacja pliku apk

- `adb shell` – włączenie 'powłoki' (terminala) na urządzeniu. Pozwala uruchamiać komendy w bezpośrednio na urządzeniu, bez konieczności poprzednia ich '`adb`'. Możemy wykonywać niektóre polecenia znane z linuksa

- `adb push/pull` – wgrywanie/zgrywanie plików do i z urządzenia

adb shell:

- `pm list packages` – lista zainstalowanych aplikacji

- `pm path <nazwa_pakietu>` – ścieżka do danego pakietu

- `pm install/uninstall <nazwa_pakietu>` – instalacja/usunięcie aplikacji z podanego pakietu

- `pm disable <nazwa_pakietu>` - wyłączenie niechcianego pakietu, którego nie można usunąć

Zadanie 1:

Wykonaj polecenie `pm list packages` na swoim telefonie lub emulatorze. Wybierz jeden z pakietów i ściągnij jego plik apk na dysk. Będzie nam potrzebny później.

2. Inzynieria wsteczna pliku apk

Plik `AndroidManifest.xml` może dostarczyć dużo informacji o analizowanym pliku .apk.

Plik ten można analizowac nawet bez otwierania pakietu apk – narzędzie aapt dostarczane wraz z Android SDK pozwala wyświetlić dane z pliku AndroidManifest.xml:

```
aapt dump xmltree plik.apk AndroidManifest.xml
```

Dane te są jednak trochę nieczytelne w wyświetlanym formacie, służą bardziej jako wejście dla innych programów analizujących. Obecnie Android Studio dostarcza bardzo fajny tool do analizy pliku apk. W tym celu należy uruchomić Android Studio, kliknąć na ekranie powitalnym ‘Profile or Debug apk’ i wybrać plik apk do analizy.

Zadanie 2

Poświęć chwilę na analizę pliku apk. Co ciekawego możesz dowiedzieć się o analizowanej aplikacji? Czego możesz dowiedzieć się z pliku manifestu – jakie komponenty zawiera aplikacja, jakich wymaga pozwoleń? Opisz w kilku zdaniach z przykładami fragmentów kodu.

Plik .apk jest w istocie spakowanym archiwum – można to zobaczyć, zmieniając jego rozszerzenie na .zip i rozpakowując, lub wykonując komendę

```
unzip plik.apk
```

Komenda rozpakuje plik, ujawniając wszystkie pliki, które się tam znajdują (pliki będą takie same, jak widoczne w Android Studio).

Na tym etapie można np. podejrzeć wszystkie resources (m.in. obrazki) używane przez aplikację. Nas najbardziej będzie interesować odzyskanie kodu aplikacji. Można zobaczyć, że rozpakowany plik. Apk zawiera kilka plików .dex, które są skompilowanymi plikami – można je w prosty sposób przywrócić do postaci mniej-więcej czytelnej dla człowieka. W tym celu możemy posłużyć się narzędziem **dex2jar:** https://github.com/pxb1988/dex2jar/releases

```
./d2j-dex2jar.sh sciezka/do/pliku.apk -o /sciezka/do/pliku.jar
```

Wygenerowany plik jar można przeglądać w wielu dostępnych narzędziach, jednym z najpopularniejszych jest **JD-GUI**: http://java-decompiler.github.io/

Narzędzie czasami nie radzi sobie z niektórymi plikami – lepsze wyniki można uzyskać korzystając np. z serwisu http://www.javadecompilers.com/apk

Pliki wynikowe z podanego serwisu są najłatwiejsze w analizie – mamy podane klasy javy wraz z odkodowanymi zasobami.

Poświęć chwilę na analizę uzyskanego kodu aplikacji. Spróbuj znaleźć kilka klas komponentów wymenionych w manifeście, m.in. główną aktywność. Co robi? Czy uruchamia jakiś serwis, inną aktywność?

Do uzyskania kolejnej postaci zdekompilowanej aplikacji bliskiej kodowi źródłowego aplikacji można użyć również narzędzia apktool: https://ibotpeaches.github.io/Apktool/

```
apktool d sciezka/do/pliku.apk
```

Narzędzie to umożliwia zdekompilowanie pliku apk wraz z zasobami (resources), wprowadzenie zmian oraz ponownie skompilowanie go po wprowadzeniu zmian.

Apktool nie generuje klas .java w wynikowych plikach – zamiast tego mamy pliki w języku smali. Porównanie plików z poprzednio uzskanego pliku .apk z plikami .smali pozwala łatwiej znaleźc miesjca, które chcielibyśmy zmodyfikować.

Zmodyfikowany apk można zbudować komendą:

```
apktool b sciezka/do/folderu
```

W przypadku problemów ze zbudowanie aplikacji należy wywołać najpierw

```
apktool empty-framework-dir
```

Wynikowy apk znajduje się w folderze `dist/`

Spróbuj zainstalować wygenerowany .apk – co się dzieje?

Wygenerowany zostaje błąd `INSTALL_PARSE_FAILED_NO_CERTIFICATES` – oznacza to, że plik apk nie został podpisany. Takiego apk nie da się zainstalować na urządzeniu.

3. Podpisywanie aplikacji

Aplikacje na system Android podpisywane są za pomocą pliku z rozszerzeniem .keystore. Plik ten można łatwo wygenerować z uzyciem liniii poleceń lub z poziomu Android Studio. Każda aplikacja musi być podpisana jakimś certyfikatem – dotyczy to także aplikacji w trakcie developmentu. Android Studio domyślnie tworzy certyfikat debug, który jednak nie może zostać użyty do podpisania aplikacji wysłanej do sklepu GooglePlay. W tym celu musimy wgenerować własny

```
keytool -genkey -keystore test.keystore -validity 3650 -alias test
```

Jeden plik keystore może być użyty do podpisywania wielu aplikacji. Można w nim utworzyć kilka aliasów)w tym przyadku o nazwie `test`) i używać każdego do innej aplikacji

Musimy następnie odpowiedzieć na kilka pytań, m.in. o hasła dostąpu do pliku keystore.

Plik ten służy do udowodnienia, że aplikacja pochodzi od danego developera, nie należy go więc umieszczać w repozytorium ani ogólnodostępnym miejscu. Niestety, po zgubieniu tego pliku nie można więcej aktualizowac własnych aplikacji – trzeba wypuścić całkiem nowe wersje z nowymi pakietami (nazwami pakietów).

Po wygenerowaniu pliku keystore należy podpisać plik apk uzyskany w poprzednim punkcie:

```
jarsigner -keystore test.keystore -verbose plik.apk test
```

Spróbuj zainstalować tak podpisany plik apk. Co się dzieje?

Wyskakuje błąd:

```
adb: failed to install base/dist/base.apk: Failure
[INSTALL_FAILED_UPDATE_INCOMPATIBLE: Package pl.mpay.app
signatures do not match previously installed version; ignoring!]
```

Błąd spowodowany jest tym, że mamy już zainstalowaną aplikację z danym pakietem, jednak podpisy się nie zgadzają. Dzięki temu nie można tak łatwo podszyć się pod kogoś i podmienić aplikacji na swoją wersję. Trzeba najpierw odinstalować poprzednią wersję – wtedy będzie możliwe zainstalowanie nowej, zmienionej wersji aplikacji.

Zadanie 3

Użyj narzędzia apktool do zdekompilowania wybranego pliku apk. Wprowadź kilka zmian (wyświetlanie innej aktywności po kliknięciu w przycisk, zmiana layoutu). Opisz, jakie zmiany zastosowałeś/aś względem oryginalnego pliku.

Wygeneruj własny plik .keystore, zbuduj zmodyfikowaną aplikację i podpisz ją wygenerowanym plikiem, tak, żeby można ją było zainstalować na urządzeniu i zmiany były widoczne.

Do zadania należy przesłać:

- oryginalny plik .apk

- opis zmian

- zmieniony plik .apk