

Επίθεση Android Repackaging

Εισαγωγή

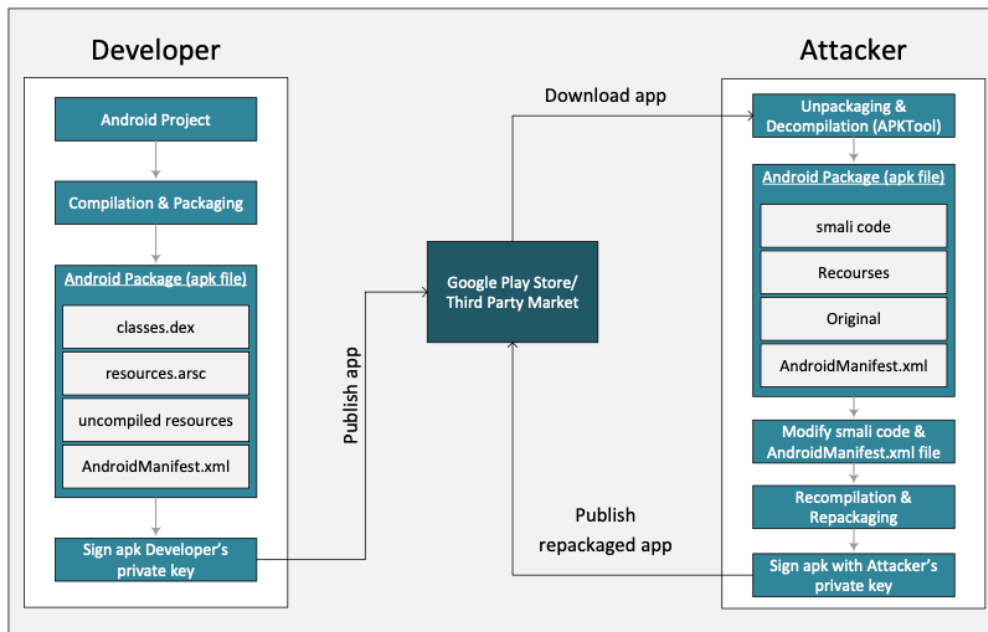
Η επίθεση **Repackaging** (επανασυσκευασία ή επανασυναρμολόγηση) αποτελεί έναν πολύ κοινό τύπο επιθέσεων σε συσκευές Android. Σε μια τέτοια επίθεση, οι επιτιθέμενοι τροποποιούν μια δημοφιλή εφαρμογή που βρίσκουν σε μια αγορά εφαρμογών (app store), προσθέτοντας κακόβουλο κώδικα και στη συνέχεια, ανεβάζουν την τροποποιημένη εφαρμογή πάλι σε αγορές εφαρμογών. Οι χρήστες μπορούν εύκολα να ξεγελαστούν, επειδή είναι δύσκολο να παρατηρηθεί η διαφορά μεταξύ της αρχικής και της τροποποιημένης εφαρμογής. Μόλις εγκατασταθεί μια τροποποιημένη εφαρμογή, ο κακόβουλος κώδικας που έχει ενσωματωθεί σε αυτή μπορεί να πραγματοποιήσει επιθέσεις, συνήθως στο παρασκήνιο. Για παράδειγμα, τον Μάρτιο του 2011, διαπιστώθηκε ότι το DroidDream Trojan¹ είχε ενσωματωθεί σε περισσότερες από 50 εφαρμογές σε επίσημο app store του Android και είχε μολύνει πολλούς χρήστες που τις εγκατέστησαν. Το DroidDream Trojan εκμεταλλεύεται ευπάθειες στο Android για να αποκτήσει πρόσβαση root στη συσκευή.

Ο μαθησιακός στόχος αυτού του εργαστηρίου είναι να αποκτήσετε άμεση εμπειρία στην επίθεση **Android Repackaging** ώστε να μπορείτε να κατανοήσετε καλύτερα τον συγκεκριμένο κίνδυνο που σχετίζεται με το Android και να είστε πιο προσεκτικοί κατά τη λήψη εφαρμογών στις συσκευές σας, ειδικά από μη αξιόπιστες αγορές εφαρμογών. Σε αυτό το εργαστήριο, θα κληθείτε να πραγματοποιήσετε μια απλή επίθεση **Android Repackaging** σε μια επιλεγμένη εφαρμογή σε περιβάλλον εικονικής μηχανής Android VM. Τονίζεται ότι, δεν θα πρέπει να αναρτήσετε παρόμοιο κώδικα σε καμία αγορά εφαρμογών, διαφορετικά θα αντιμετωπίσετε νομικές συνέπειες. Επίσης, δεν θα πρέπει να εκτελέσετε την επίθεση στις δικές σας συσκευές Android, καθώς αυτό μπορεί να προκαλέσει πραγματικές ζημιές.

Αυτό που καθιστά εύκολη τη συγκεκριμένη επίθεση είναι ότι ο δυαδικός κώδικας (binary code) των εφαρμογών Android μπορεί εύκολα να αντιστραφεί, λόγω της έλλειψης προστασίας στα δυαδικά αρχεία. Στο αριστερό μέρος της Εικόνας 1 απεικονίζεται η τυπική διαδικασία ανάπτυξης εφαρμογών Android, κατά την οποία παράγεται ένα αρχείο που ονομάζεται **αρχείο APK**. Αυτό είναι το αρχείο που μεταφορτώνεται στη συνέχεια στις αγορές εφαρμογών ώστε να είναι διαθέσιμο στους τελικούς χρήστες.

Στο δεξιό μέρος της Εικόνας 1 απεικονίζεται η διαδικασία του repackaging: ο επιτιθέμενος (κακόβουλος χρήστης), αφού λάβει το αρχείο APK, είναι σε θέση χρησιμοποιώντας εργαλεία αντίστροφης μηχανικής (reverse engineering) να το "αποσυσκευάσει", να προσθέσει τον κακόβουλο κώδικα και στη συνέχεια να το "επανασυσκευάσει". Στη συνέχεια, ο επιτιθέμενος ανεβάζει την επανασυσκευασμένη εφαρμογή σε αγορές εφαρμογών, οι περισσότερες από τις οποίες δεν διαθέτουν αντίμετρα για να ανιχνεύσουν εάν μια εφαρμογή είναι επανασυσκευασμένη ή όχι.

¹ https://www.f-secure.com/v-descs/trojan_android_droiddream.shtml



Εικόνα 1. Επισκόπηση της επίθεσης Repackaging

Περιβάλλον εργαστηρίου

Η άσκηση αυτή απαιτεί δύο εικονικές μηχανές, την **SEEDUbuntu 16.04** και την **SEEDAndroid 7.1**. Η πρώτη εικονική μηχανή (SEEDUbuntu) θα χρησιμοποιηθεί για την προετοιμασία της επίθεσης από την πλευρά του επιτιθέμενου, δηλαδή για τη λήψη της εφαρμογής, την από-συσκευασία, την προετοιμασία του κακόβουλου κώδικα, την επανασυσκευασία και τέλος την προώθηση της εφαρμογής στον στόχο της επίθεσης. Η μηχανή έχει ήδη προεγκατεστημένα τα εργαλεία που θα χρειαστούν (**adb**, **apktool**, **keytool**, **jarsigner**). Η δεύτερη εικονική μηχανή (SEEDAndroid²), όπως υποδηλώνει και το όνομα της, τρέχει λειτουργικό σύστημα Android 7.1 (Nougat, API Level 25) και θα χρησιμοποιηθεί σαν στόχος της επίθεσης, προσομοιώνοντας τη συσκευή όπου θα εγκατασταθεί η επανασυσκευασμένη εφαρμογή. Η μηχανή έχει ήδη προεγκατεστημένες κάποιες εφαρμογές που θα χρειαστούν στην πλευρά της συσκευής (**Terminal Emulator**, **MockLocation**).

Για τις ανάγκες της άσκησης, οι δυο εικονικές μηχανές θα πρέπει να συνδεθούν στο ίδιο τοπικό δίκτυο. Για το σκοπό αυτό, θα πρέπει και στις δυο εικονικές μηχανές να ρυθμίσουμε τον ίδιο προσαρμογέα "Δίκτυο Nat". Σχετικές οδηγίες για την αρχική εγκατάσταση του SEEDAndroid 7.1 VM αλλά και τη ρύθμιση του τοπικού δικτύου στις δυο μηχανές παρέχονται στο σχετικό manual που θα βρείτε στο eclass.

Το λογισμικό που θα απαιτηθεί για την άσκηση, βρίσκεται στο eclass:

- RepackagingLab.apk (εφαρμογή-στόχος και στα δυο σενάρια)
- MaliciousCode.smali (κακόβουλος κώδικας στο σενάριο 1)
- MaliciousCode_Location.zip (MaliciousCode.smali, SendData.smali, SendData\$1.smali) (κακόβουλος κώδικας στο σενάριο 2)

² Στο eclass θα βρείτε links για να κατεβάσετε τη μηχανή καθώς και τα σχετικά manuals

Δραστηριότητα 1: Λήψη και εγκατάσταση εφαρμογής-στόχου

Σκοπός αυτής της δραστηριότητας είναι η λήψη της αρχικής εφαρμογής-στόχου (στην οποία θα ενσωματωθεί ο κακόβουλος κώδικας) και η **εξοικείωση** με το εργαλείο adb το οποίο θα χρησιμοποιηθεί για την εγκατάσταση της εφαρμογής στη συσκευή-στόχο (για τις ανάγκες των σεναρίων της άσκησης).

Σε πραγματικές επιθέσεις, οι επιτιθέμενοι συνήθως επιλέγουν να τοποθετήσουν τον κακόβουλο κώδικα σε δημοφιλείς εφαρμογές, επειδή έτσι είναι πιθανότερο περισσότερα άτομα να κατεβάσουν τις "επανασυσκευασμένες" εφαρμογές. Για τη συγκεκριμένη άσκηση, μπορείτε να αναπτύξετε τη δική σας εφαρμογή android ή να χρησιμοποιήσετε μια υπάρχουσα εφαρμογή, κατεβάζοντας το αρχείο ark από σχετικούς ιστότοπους³. Εναλλακτικά, παρέχεται μια έτοιμη εφαρμογή που έχει αναπτυχθεί για το σκοπό αυτό, με το όνομα **RepackagingLab.apk**.

Σημείωση: Θα πρέπει να σημειωθεί ότι σε αυτό το εργαστήριο χρησιμοποιούμε ένα Android VM και όχι μια φυσική συσκευή Android, επομένως ορισμένες εφαρμογές που θα δοκιμάσετε ενδέχεται να μην εκτελούνται (θα κολλήσουν). Ένας από τους πιθανούς λόγους είναι ότι η εφαρμογή μπορεί να περιέχει native κώδικα για επεξεργαστές ARM, ενώ το Android VM τρέχει σε επεξεργαστές αρχιτεκτονικής x86. Για να εκτελούνται αυτές οι εφαρμογές στο Android VM, ο native κώδικας πρέπει να μεταγλωττιστεί εκ νέου για επεξεργαστές x86. Αυτό απαιτεί να έχουμε στη διάθεσή μας τον πηγαίο κώδικα, ο οποίος είναι δύσκολο να ληφθεί. Επομένως, εάν αντιμετωπίσατε αυτό το πρόβλημα, απλώς βρείτε μια άλλη εφαρμογή.

Βήμα 1: Λήψη εφαρμογής-στόχου από τον επιτιθέμενο

Αρχικά θα πρέπει να εκκινήσετε την εικονική μηχανή του επιτιθέμενου (SEEDUbuntu VM) και χρησιμοποιώντας τον Firefox να πλοηγηθείτε στο eclass του μαθήματος και να κατεβάσετε το αρχείο **RepackagingLab.apk**.

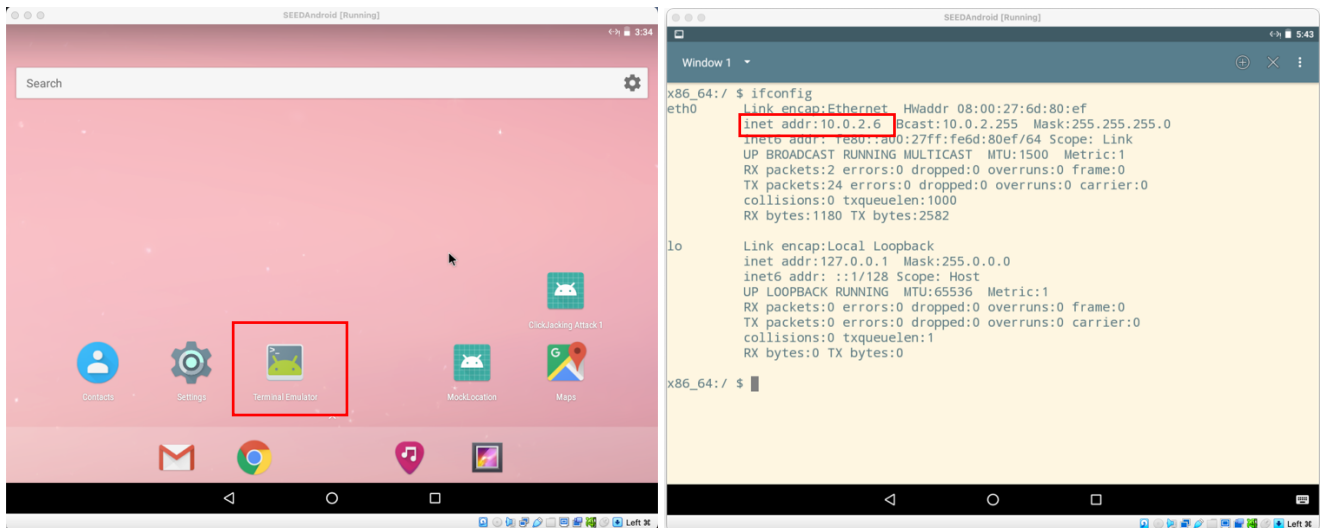
Βήμα 2: Εγκατάσταση της εφαρμογής-στόχου στη συσκευή Android του θύματος

Για τις ανάγκες του σεναρίου της άσκησης, θα χρησιμοποιήσουμε το εργαλείο adb⁴ για να εγκαταστήσουμε μια εφαρμογή-στόχο (RepackagingLab.apk ή άλλη της επιλογής μας) στη συσκευή του θύματος. Η συγκεκριμένη διαδικασία γίνεται μόνο για τις ανάγκες της άσκησης, καθώς σε πραγματικές συνθήκες ο χρήστης μιας συσκευής θα εγκαθιστούσε εφαρμογές κατεβάζοντας τις απευθείας από σχετικά app stores.

Θα πρέπει να εκκινήσουμε και τη δεύτερη εικονική μηχανή (SEEDAndroid VM) και να σιγουρευτούμε ότι είναι συνδεδεμένες στο ίδιο **Δίκτυο NAT**. Στη συνέχεια, πρέπει να βρούμε τη διεύθυνση IP του SEEDAndroid VM. Αυτό μπορεί να επιτευχθεί εκτελώντας την εντολή **ifconfig** στην εφαρμογή Android Terminal που είναι ήδη εγκατεστημένη στη συσκευή:

³ όπως π.χ. <https://apkpure.com>

⁴ <https://developer.android.com/studio/command-line/adb>



Εικόνα 2. Εύρεση της διεύθυνσης IP της μηχανής SEEDAndroid VM

Έπειτα, επιστρέφουμε στην **SEEDUbuntu VM**, και εκτελούμε τις παρακάτω εντολές στο terminal:

```
// Σύνδεση με το Android VM χρησιμοποιώντας το adb
$ adb connect <ip_address_of_android_vm>;

// Εγκατάσταση της εφαρμογής στο Android
$ adb install <application_name>.apk
```

Το αποτέλεσμα των παραπάνω βημάτων θα είναι η εγκατάσταση της επιλεγμένης εφαρμογής στη μηχανή SEEDAndroid VM. Μπορούμε να το ελέγξουμε πηγαίνοντας στις **Εφαρμογές** και αναζητώντας το όνομα της εφαρμογής που εγκαταστήσαμε.

Στην αναφορά σας, εκτός από την επεξήγηση των ενεργειών σας, θα πρέπει να συμπεριλάβετε στιγμιότυπα οθόνης και από τα δύο VMs, με τις εντολές που χρησιμοποιήσατε και τα αποτελέσματα αυτών.

Σημείωση: μετά το τέλος της Δραστηριότητας 1, θα πρέπει να απεγκαταστήσετε (χειροκίνητα) την εφαρμογή από τη μηχανή Android VM, καθώς θα επαναλάβουμε τη διαδικασία αφού γίνει το repackaging.

Δραστηριότητα 2: Αποσυναρμολόγηση της εφαρμογής

Για να ξεκινήσουμε την **επίθεση repackaging** πρέπει αρχικά να τροποποιήσουμε κατάλληλα την **εφαρμογή-στόχο**. Παρόλο που μπορούμε να τροποποιήσουμε απευθείας το αρχείο APK, κάτι τέτοιο δεν είναι εύκολο, καθώς ο κώδικας στο αρχείο APK βρίσκεται σε μορφή **Dalvik bytecode**⁵ (αρχεία **.dex**), που δεν προορίζεται για ανάγνωση από τον άνθρωπο. Θα πρέπει λοιπόν να μετατρέψουμε τον **bytecode** σε μορφή που να είναι αναγνώσιμη από τον άνθρωπο. Η πιο κοινή τέτοια μορφή για τον κώδικα **Dalvik** είναι γνωστή ως **Smali**⁶.

Για να αποσυναρμολογήσουμε το αρχείο APK, θα χρησιμοποιήσουμε ένα εργαλείο γραμμής εντολών που ονομάζεται **APKTool**⁷. Το **APKTool** είναι ένα πολύ ισχυρό εργαλείο αντίστροφης μηχανικής (reverse engineering) για εφαρμογές Android, το οποίο μπορεί να αποσυναρμολογήσει τον κώδικα **dex** απευθείας σε κώδικα **smali**. Μπορούμε να τροφοδοτήσουμε ολόκληρο το αρχείο APK στο εργαλείο ως εξής:

```
$ apktool d <appname>.apk
```

Ένα APK είναι ουσιαστικά ένα συμπιεσμένο αρχείο, το οποίο περιέχει τα διάφορα αρχεία που αποτελούν μια εφαρμογή, όπως:

- **classes.dex**: πρόκειται για αρχεία που περιέχουν μεταγλωττισμένο κώδικα java, σε μια μορφή που ονομάζεται **Dalvik bytecode**.
- **resources.arsc**: πρόκειται για αρχεία πόρων της εφαρμογής, όπως αρχεία xml, εικόνες, κλπ.
- **AndroidManifest.xml**: πρόκειται για το βασικό αρχείο ρυθμίσεων μιας εφαρμογής android.

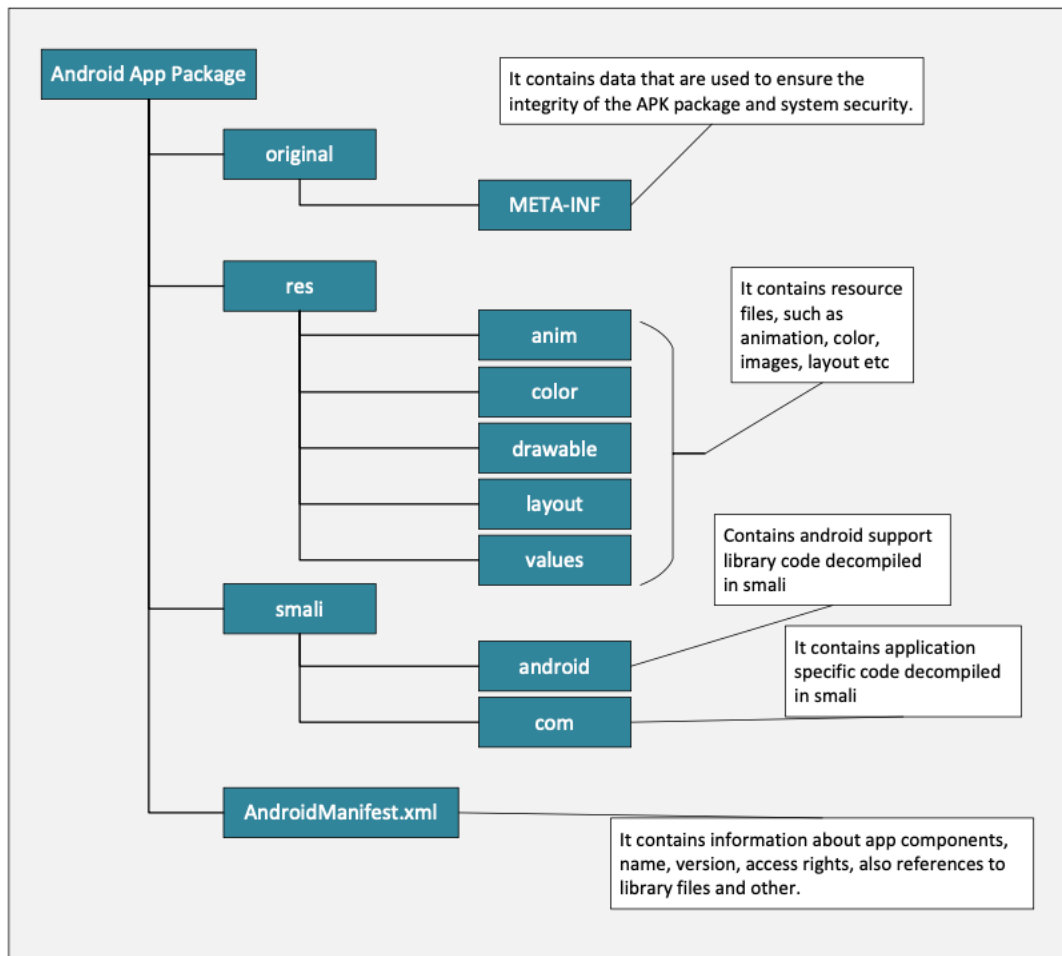
Το **APKTool** βασικά αποσυμπιέζει το αρχείο APK και αποκωδικοποιεί τα περιεχόμενα του. Για τα αρχεία πόρων, δεν χρειάζεται να γίνουν πολλά καθώς είναι ήδη σε αναγνώσιμη μορφή. Τα αρχεία τύπου **Dalvik bytecode** (**classes.dex**), αποσυναρμολογούνται σε αρχεία **smali**, τα οποία τοποθετούνται σε έναν νέο κατάλογο.

Η τυπική δομή καταλόγων του αρχείου APK μετά την αποσυναρμολόγηση φαίνεται στην Εικόνα 3. Τα αρχεία πόρων και το αρχείο **AndroidManifest** είναι άμεσα αναγνώσιμα καθώς βρίσκονται στην αρχική τους μορφή (π.χ. XML). Ο αποσυναρμολογημένος κώδικας **smali** τοποθετείται στο κατάλογο **/smali**. Συνήθως, κάθε αρχείο τύπου **smali** περιέχει τον κώδικα που αντιστοιχεί σε μία κλάση **Java**.

⁵ Τα αρχεία **.dex** παράγονται από τα αρχεία **.class**, με κωδικοποίηση από το εργαλείο **dx** του Android

⁶ Τα αρχεία αυτά μπορούν να παραχθούν με κωδικοποίηση των αρχείων **.dex** με το εργαλείο **backsmali** (<https://github.com/JesusFreke/smali>). Τα ονόματα "**Smali**" και "**Baksmali**" είναι τα ισλανδικά ισοδύναμα του "**assembler**" και "**disassembler**", αντίστοιχα.

⁷ <https://ibotpeaches.github.io/Apktool/>



Εικόνα 3. Δομή καταλόγων ενός APK μετά την αποσυναρμολόγηση με το **APKTool**

Στα πλαίσια της δραστηριότητας 2, θα πρέπει να αποσυναρμολογήσετε την εφαρμογή-στόχο με τη χρήση του **apktool** και να βρείτε τα αντίστοιχα αρχεία που παράγονται. Η διαδικασία αυτή πρέπει να λάβει χώρα στο μηχάνημα του επιτιθέμενου, δηλαδή στην εικονική μηχανή SEEDUbuntu VM.

Στην αναφορά σας θα πρέπει να συμπεριλάβετε στιγμιότυπα οθόνης με την εκτέλεση της αποσυναρμολόγησης και τη δομή των καταλόγων που προκύπτει.

Δραστηριότητα 3: Ενσωμάτωση κακόβουλου κώδικα

Στη δραστηριότητα αυτή, θα ενσωματώσουμε τον κακόβουλο κώδικα στην εφαρμογή-στόχο. Υπάρχουν πολλοί τρόποι για να γίνει αυτό. Μια προσέγγιση είναι να τροποποιήσουμε απευθείας κάποιο υπάρχον αρχείο **smali**, προσθέτοντας τμήματα κακόβουλου κώδικα σε αυτό. Μια άλλη προσέγγιση, η οποία είναι πολύ πιο εύκολη, είναι να προσθέσουμε ένα εντελώς νέο, ανεξάρτητο στοιχείο στην εφαρμογή, το οποίο δεν επηρεάζει την υπόλοιπη συμπεριφορά της. Δεδομένου ότι κάθε ανεξάρτητο στοιχείο μπορεί να τοποθετηθεί σε ξεχωριστό αρχείο **smali**, χρησιμοποιώντας αυτήν την προσέγγιση, χρειάζεται απλώς να δημιουργήσουμε ένα νέο αρχείο **smali** που να περιέχει τον κακόβουλο κώδικα.

Ανατομία μιας εφαρμογής Android

Υπάρχουν τέσσερις τύποι στοιχείων που χρησιμοποιούνται ως δομικά συστατικά μιας εφαρμογής Android:

- Activities,
- Services,
- Broadcast Receivers, και
- Content Providers.

Τα δυο πρώτα στοιχεία χρησιμοποιούνται πιο συχνά από εφαρμογές, ενώ τα άλλα δυο δεν είναι τόσο κοινά. Μπορούμε να προσθέσουμε οποιοδήποτε από αυτά τα στοιχεία στην εφαρμογή-στόχο, αλλά το πιο σημαντικό πρόβλημα για την επίθεση είναι να βρούμε έναν τρόπο να ενεργοποιήσουμε τον κακόβουλο κώδικα χωρίς να γίνει αντιληπτός από τους χρήστες.

Αν και υπάρχουν λύσεις στο πρόβλημα για όλα αυτά τα στοιχεία, η πιο εύκολη είναι χρησιμοποιώντας **BroadcastReceivers**. Ένας **BroadcastReceiver** είναι ουσιαστικά κώδικας που ενεργοποιείται από εκπομπές (broadcasts) που εκπέμπονται από το σύστημα Android κατά τη διάρκεια λειτουργίας μιας συσκευής, με σκοπό να ενημερώσουν (όσες εφαρμογές "ενδιαφέρονται") ότι η συγκεκριμένης ενέργεια έλαβε χώρα. Για παράδειγμα, όταν ρυθμίζεται η ώρα σε μια συσκευή, θα σταλεί η εκπομπή **TIME SET**, ενώ όταν γίνει επανεκκίνηση του συστήματος, θα σταλεί η εκπομπή **BOOT COMPLETED**.

Μια εφαρμογή που "ενδιαφέρεται" για τη λήψη κάποιας από αυτές τις εκπομπές, μπορεί να περιλάβει έναν "δέκτη εκπομπής" (**BroadcastReceiver**) ο οποίος θα τη λαμβάνει και θα δρομολογεί κατάλληλες ενέργειες.

Ανάλυση κακόβουλου κώδικα

Στη συγκεκριμένη περίπτωση, θα γράψουμε και θα ενσωματώσουμε στην εφαρμογή έναν **BroadcastReceiver**, ο οποίος θα ακούει μία από αυτές τις εκπομπές του συστήματος και στη συνέχεια θα ενεργοποιεί κάποιον κακόβουλο κώδικα. Με τον τρόπο αυτό διασφαλίζουμε ότι η εκτέλεση του κακόβουλου κώδικα θα γίνεται "μυστικά", χωρίς να απαιτείται κάποια οφθαλμοφανής ενέργεια από την πλευρά του χρήστη (θύματος) της συσκευής. Ο πηγαίος κώδικας Java του **BroadcastReceiver** είναι ο εξής:

```

public class MaliciousCode extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        ContentResolver contentResolver = context.getContentResolver();
        Cursor cursor = contentResolver.query
            (ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
        while (cursor.moveToNext()) {
            String lookupKey = cursor.getString
                (cursor.getColumnIndex(ContactsContract.Contacts.LOOKUP_KEY));
            Uri uri = Uri.withAppendedPath
                (ContactsContract.Contacts.CONTENT_LOOKUP_URI, lookupKey);
            contentResolver.delete(uri, null, null);
        }
    }
}

```

Ο παραπάνω κώδικας, όταν εκτελεστεί, θα διαγράψει όλες τις εγγραφές επαφών από τη συσκευή. Ο κώδικας υλοποιεί έναν **BroadcastReceiver**, ο οποίος θα ενεργοποιηθεί από **ένα συμβάν** εκπομπής. Μόλις ενεργοποιηθεί, εκτελείται η μέθοδος **onReceive()** όπου υλοποιείται η κακόβουλη δράση του. Ο κώδικας αλληλεπιδρά βασικά με το στοιχείο **ContentProvider** της εφαρμογής **Contacts** (που βρίσκεται προεγκατεστημένη στο Android) και του ζητά να καταργήσει όλες τις καταχωρήσεις του, ουσιαστικά διαγράφοντας όλες τις επαφές.

Στον κώδικα, χρησιμοποιείται ένα αντικείμενο τύπου **ContentResolver** για πρόσβαση στις επαφές που είναι αποθηκευμένες στο τηλέφωνο. Για να λάβουμε τη λίστα επαφών, αποστέλλεται ένα ερώτημα (query) στον **ContentProvider** της εφαρμογής **Contacts**. Αυτό το ερώτημα δεν περιέχει κανένα κριτήριο, επομένως επιστρέφονται όλες οι εγγραφές επιστρέφονται σε ένα αντικείμενο τύπου **Cursor**. Στη συνέχεια, ο κώδικας διατρέχει όλες τις εγγραφές και τις διαγράφει μία προς μία.

Πριν την επανασυναρμολόγηση, η εφαρμογή θα πρέπει να ρυθμιστεί κατάλληλα ώστε να καλείται ("ενεργοποιείται") ο συγκεκριμένος **BroadcastReceiver**. Για το σκοπό αυτό θα πρέπει να "καταχωρηθεί" (register) η συγκεκριμένη κλάση Java στο αρχείο **AndroidManifest.xml** της εφαρμογής-στόχου. Το αρχείο **AndroidManifest** περιλαμβάνει αρχικές ρυθμίσεις και πληροφορίες για τα δομικά στοιχεία μιας εφαρμογής android (activities, services, κλπ.). Επίσης, σε αυτό δηλώνονται και οι σχετικές άδειες (permissions) τις οποίες ζητάει η εφαρμογή από το σύστημα. Στη συγκεκριμένη περίπτωση, οι άδειες που απαιτούνται είναι η δυνατότητα **ανάγνωσης** και **εγγραφής** των **Επαφών**.

Βήμα 1: Κατέβασμα του κακόβουλου κώδικα (σε μορφή smali)

Ο παραπάνω κακόβουλος κώδικας είναι διαθέσιμος απευθείας σε μορφή **smali**. Θα πρέπει αρχικά να κατεβάσετε το αρχείο **MaliciousCode.smali** από το eclass. Η διαδικασία αυτή πρέπει να λάβει χώρα στο μηχάνημα του επιτιθέμενου, δηλαδή στην εικονική μηχανή SEEDUbuntu VM.

Βήμα 2: Ενσωμάτωση κακόβουλου κώδικα στην εφαρμογή-στόχο

Στη συνέχεια θα πρέπει να τοποθετήσετε το αρχείο **MaliciousCode.smali** στον κατάλογο **/smali/com/** που δημιουργήθηκε όταν αποσυναρμολογήσατε την εφαρμογή με το **APKTool** (κατά την προηγούμενη δραστηριότητα).

Βήμα 3: Ρύθμιση της επανασυναρμολογημένης εφαρμογής

Αρχικά, θα πρέπει να προσθέσετε τις δυο απαιτούμενες άδειες (permissions) για να επιτραπεί στην εφαρμογή να διαβάζει και να γράφει στον **ContentReceiver** της εφαρμογής **Contacts**. Θα πρέπει να σημειωθεί ότι αυτά τα δικαιώματα προστίθενται έξω από το μπλοκ **<application>**, αλλά εντός του μπλοκ **<manifest>**. Έπειτα, θα πρέπει να καταχωρήσετε τον **BroadcastReceiver** και να ρυθμίσετε το **<intent-filter>** ώστε να λαμβάνει το συμβάν εκπομπής **TIME SET**. Με αυτό τον τρόπο, η κλάση θα μπορεί να ενεργοποιείται κάθε φορά που ο χρήστης αλλάζει την ώρα στη συσκευή του. Η εγγραφή αυτή πρέπει να προστεθεί μέσα στο μπλοκ <application>, και όχι μέσα στο μπλοκ **<activity>**.

Ο παρακάτω κώδικας xml παρουσιάζει ακριβώς τα στοιχεία που πρέπει να προσθέσετε στο αρχείο **AndroidManifest.xml** (το οποίο μπορείτε επίσης να βρείτε στο φάκελο που δημιουργήθηκε από το **APKTool**):

```
<manifest...>
...
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
....
<application>
.....
.....
<receiver android:name="com.MaliciousCode" >
  <intent-filter>
    <action android:name="android.intent.action.TIME_SET" />
  </intent-filter>
</receiver>
</application>
</manifest>
```

Στην αναφορά σας θα πρέπει να περιγράψτε την εκτέλεση των παραπάνω ενέργειων και να δείξετε με σχετικά στιγμιότυπα οθόνης τις αλλαγές που κάνατε και τα τελικά αρχεία που έχουν προκύψει.

Σημείωση: Στην περίπτωση που χρησιμοποιείτε σαν εφαρμογή-στόχο μια πραγματική εφαρμογή που έχετε κατεβάσει από κάποιο app store, ενδέχεται να συναντήσετε ένα αρκετά πιο πολύπλοκο αρχείο **AndroidManifest.xml**. Σε αυτή τη περίπτωση θα πρέπει να τροποποιήσετε προσεκτικά το αρχείο και να τοποθετήσετε το περιεχόμενο στη σωστή θέση.

Δραστηριότητα 4: Επανασυναρμολόγηση της εφαρμογής

Μετά την ενσωμάτωση του κακόβουλου κώδικα, μπορούμε να προχωρήσουμε στην επανασυναρμολόγηση της εφαρμογής (repackaging) και να δημιουργήσουμε εκ νέου το αρχείο APK με χρήση του εργαλείου **APKTool**.

Το Android απαιτεί επίσης την **ψηφιακή υπογραφή** όλων των εφαρμογών για να μπορέσουν να εγκατασταθούν. Αυτό σημαίνει ότι κάθε αρχείο APK θα πρέπει να έχει *ψηφιακή υπογραφή* και αντίστοιχο *πιστοποιητικό δημόσιου κλειδιού*. Το πιστοποιητικό και η υπογραφή χρησιμοποιούνται από το Android ώστε να αναγνωρίσει τον κατασκευαστή μιας εφαρμογής.

Από την άποψη της ασφάλειας, το πιστοποιητικό πρέπει να υπογραφεί από μια **Αρχή Έκδοσης Πιστοποιητικών**, η οποία, πριν υπογράψει, πρέπει να επαληθεύσει ότι η ταυτότητα που είναι αποθηκευμένη στο πιστοποιητικό είναι όντως αυθεντική. Η λήψη πιστοποιητικού από μια αποδεκτή αρχή έκδοσης πιστοποιητικών συνήθως δεν είναι δωρεάν, επομένως το Android επιτρέπει στους προγραμματιστές να υπογράψουν οι ίδιοι τα πιστοποιητικά τους, χρησιμοποιώντας το δικό τους ιδιωτικό κλειδί, δηλαδή το πιστοποιητικό είναι **αυτο-υπογεγραμμένο**. Ο σκοπός τέτοιων αυτο-υπογεγραμμένων πιστοποιητικών προορίζεται για τη δυνατότητα εκτέλεσης εφαρμογών σε συσκευές Android και όχι για λόγους ασφάλειας. Οι προγραμματιστές μπορούν να βάλουν οποιοδήποτε όνομα θέλουν στο πιστοποιητικό, ανεξάρτητα από το αν το όνομα ανήκει νόμιμα σε άλλους ή όχι, επειδή δεν εμπλέκεται καμία αρχή έκδοσης πιστοποιητικών για να το ελέγξει.

Προφανώς, αυτό ακυρώνει εντελώς τον σκοπό του πιστοποιητικού και της υπογραφής. Το **Google Play Store** πραγματοποιεί κάποια επαλήθευση ονόματος πριν αποδεχτεί μια εφαρμογή, αλλά άλλες αγορές εφαρμογών τρίτων δεν πραγματοποιούν πάντα μια τέτοια επαλήθευση. Σε αυτό το εργαστήριο, θα χρησιμοποιήσουμε απλώς ένα αυτο-υπογεγραμμένο πιστοποιητικό.

Τα παρακάτω βήματα θα πρέπει να λάβουν χώρα στο μηχάνημα του επιτιθέμενου, δηλαδή στην εικονική μηχανή SEEDUbuntu VM. Τα σχετικά εργαλεία γραμμής εντολών (**apktool**, **keytool**, **jarsigner**) είναι ήδη εγκατεστημένα.

Βήμα 1: Ανακατασκευή του αρχείου APK

Χρησιμοποιήστε το **APKTool** για να δημιουργήσετε το νέο αρχείο APK:

```
$ apktool b <application_folder>
```

Το νέο αρχείο APK θα αποθηκευτεί στον κατάλογο **RepackagingLab/dist**.

Βήμα 2: Δημιουργία ζεύγους κλειδιών

Για τη δημιουργία δημόσιου και ιδιωτικού κλειδιού χρησιμοποιήστε το εργαλείο **keytool**. Χρησιμοποιήστε την παρακάτω εντολή για να δημιουργήσετε την κλειδοθήκη με όνομα π.χ. **mykey.keystore** και μέσα σε αυτή να δημιουργήσετε ένα ζεύγος κλειδιών με το ψευδώνυμο **<alias>**:

```
$ keytool -alias <alias> -genkey -v -keystore mykey.keystore
```

Το **keytool** θα σας ζητήσει έναν κωδικό πρόσβασης, ο οποίος χρησιμοποιείται για την προστασία της κλειδοθήκης. Επίσης, θα σας ζητήσει κάποιες πρόσθετες πληροφορίες (όνομα, επώνυμο, διεύθυνση κατόχου) οι οποίες θα ενσωματωθούν στο ψηφιακό πιστοποιητικό.

Βήμα 3: Ψηφιακή υπογραφή του αρχείου APK

Μετά την παραγωγή των κλειδιών, θα πρέπει να χρησιμοποιήσετε το πρόγραμμα **jarsigner** για να υπογράψετε το αρχείο APK. Θα πρέπει να ορίσετε την κλειδοθήκη, το όνομα του αρχείου που θέλετε να υπογράψετε και το ψευδώνυμο του κλειδιού που θα χρησιμοποιήσετε:

```
$ jarsigner -keystore mykey.keystore <app_name>.apk <alias>
```

Το **jarsigner** θα σας ζητήσει να εισάγετε τον κωδικό πρόσβασης της συγκεκριμένης κλειδοθήκης και στη συνέχεια, χρησιμοποιώντας το κλειδί (που αναγνωρίζεται από το ψευδώνυμο που δώσατε) θα υπογράψει το αρχείο APK.

Στην αναφορά σας θα πρέπει να περιγράψτε την εκτέλεση των παραπάνω ενεργειών και να δείξετε, με σχετικά στιγμιότυπα οθόνης από το terminal, τα αποτελέσματα της εκτέλεσης.

Δραστηριότητα 5: Εκτέλεση της επίθεσης

Σε αυτή τη δραστηριότητα, θα εγκαταστήσουμε την τροποποιημένη εφαρμογή στο SEEDAndroid VM και θα ελέγξουμε εάν η επίθεση είναι επιτυχής ή όχι. Σκοπός της επίθεσης είναι να σβηστούν οι επαφές του χρήστη-θύματος, όταν αυτός επιχειρήσει να αλλάξει την ώρα στη συσκευή του.

Βήμα 1: Εγκατάσταση εφαρμογής στη συσκευή Android

Η επανασυναρμολογημένη και ψηφιακά υπογεγραμμένη εφαρμογή βρίσκεται στον κατάλογο **RepackagingLab/dist** (βλέπε Δραστηριότητα 4). Οι οδηγίες εγκατάστασης της τροποποιημένης εφαρμογής είναι οι ίδιες με αυτές της **Δραστηριότητας 1**.

Σημείωση: Εάν είχατε εγκαταστήσει την εφαρμογή **RepackagingLab** κατά τη Δραστηριότητα 1, θα πρέπει να την απεγκαταστήσετε (χειροκίνητα) από το Android VM. Διαφορετικά, δεν θα μπορέσουμε να εγκαταστήσουμε την τροποποιημένη εφαρμογή, λόγω της αναντιστοιχίας της υπογραφής.

Βήμα 2: Ενεργοποίηση των αδειών (permissions) στο Android VM

Πριν εκτελέσετε την επίθεση, θα πρέπει να παραχωρηθούν στην εφαρμογή-στόχο οι σχετικές άδειες πρόσβασης στις **Επαφές**. Σε ένα πραγματικό σενάριο, οι ίδιες οι εφαρμογές κατά τη χρήση τους συνήθως ζητούν άδεια από τους χρήστες (π.χ. πολύ συχνά ζητούνται άδειες για πρόσβαση σε φωτογραφίες, επαφές, τοποθεσία κ.λπ.). Για τις ανάγκες της άσκησης, η παραχώρηση αδειών θα γίνει χειροκίνητα από τα Settings του Android. Ακολουθήστε την παρακάτω διαδρομή ώστε να παραχωρήσετε στην εφαρμογή-στόχο τις σχετικές άδειες:

Settings -> Apps -> Repackaging Lab -> Permissions -> toggle contacts on

Βήμα 3: Εκτέλεση της επίθεσης

Αρχικά, θα πρέπει να εκτελέσετε την επανασυναρμολογημένη εφαρμογή **RepackagingLab** τουλάχιστον μία φορά (για να καταχωρηθεί ο **BroadcastReceiver**). Διαφορετικά, ο κακόβουλος κώδικας που εισήχθη δεν θα εκτελεστεί. Στη συνέχεια, θα πρέπει να προσθέσετε μερικές επαφές στην εφαρμογή **Contacts**. Τέλος, θα πρέπει να αλλάξετε (χειροκίνητα) την ώρα από τα Settings του Android, ακολουθώντας την παρακάτω διαδρομή:

Settings -> Date and Time -> Set time

Εάν η επίθεσή είναι επιτυχής, θα παρατηρήσετε ότι **έχουν διαγραφεί** όλες οι εγγραφές επαφών που μόλις εισαγάγατε.

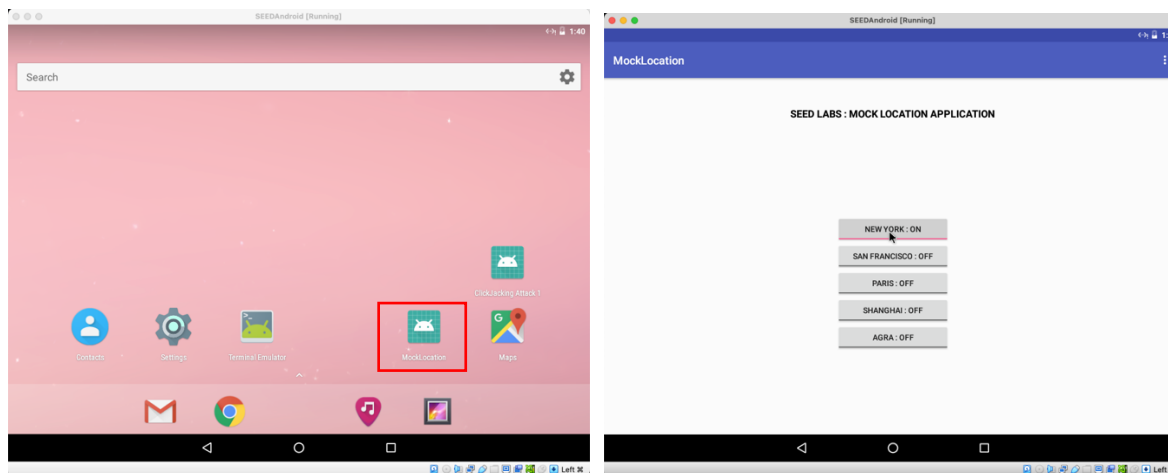
Στην αναφορά σας θα πρέπει να περιγράψτε την εκτέλεση των παραπάνω ενέργειων και να δείξετε με σχετικά στιγμιότυπα οθόνης (και από τις δύο εικονικές μηχανές) τα αποτελέσματα της εκτέλεσης.

Δραστηριότητα 6: Παρακολούθηση της τοποθεσίας του θύματος

Στη δραστηριότητα αυτή θα χρησιμοποιήσουμε την τεχνική του repackaging για να εκτελέσουμε μια διαφορετική επίθεση. Αυτή τη φορά, σκοπός της επίθεσης είναι να παρακολουθούμε **την τοποθεσία** του χρήστη-θύματος.

Ρύθμιση εικονικών τοποθεσιών

Σε μια συσκευή Android, μπορούμε να λάβουμε τις πληροφορίες τοποθεσίας από το ενσωματωμένο GPS. Το Android VM που χρησιμοποιούμε για τις ανάγκες της άσκησης δεν διαθέτει τέτοιο υλικό, ωστόσο το Android OS μας επιτρέπει να παρέχουμε **εικονικές τοποθεσίες** σε εφαρμογές. Το μόνο που χρειάζεται να κάνουμε είναι να εγκαταστήσουμε μια εικονική εφαρμογή τοποθεσίας και, στη συνέχεια, να διαμορφώσουμε το λειτουργικό σύστημα Android ώστε να λαμβάνει τοποθεσίες από αυτή την εφαρμογή, αντί για ένα πραγματικό GPS. Στο Android VM που χρησιμοποιούμε, έχει ήδη εγκατασταθεί μια τέτοια εφαρμογή (**MockLocation**), η οποία μπορεί να προσομοιώνει ίχνη τοποθεσίας σε έξι διαφορετικές πόλεις (Εικόνα 4).



Εικόνα 4. Εφαρμογή προσομοίωσης τοποθεσίας

Βήμα 1: Διαμόρφωση DNS στο Android VM

Σύμφωνα με το σενάριο της άσκησης, ο κακόβουλος κώδικας στην επανασυσκευασμένη εφαρμογή θα πρέπει να στείλει τις συντεταγμένες του θύματος στον διακομιστή του επιτιθέμενου στην (εικονική) διεύθυνση **www.repackagingattacklab.com**. Θα χρησιμοποιήσουμε τη εικονική μηχανή του επιτιθέμενου (**SEEDUbuntu VM**) για να φιλοξενήσουμε αυτόν τον διακομιστή. Επομένως, θα πρέπει να αντιστοιχίσουμε τη συγκεκριμένη **web διεύθυνση** στην **IP διεύθυνση** του **SEEDUbuntu VM**. Ο ευκολότερος τρόπος για να γίνει αυτό είναι να ρυθμίσουμε κατάλληλα το αρχείο **/system/etc/hosts** στο **Android VM**.

Για να κάνετε την παραπάνω ρύθμιση, μπορείτε να χρησιμοποιήσετε ένα εργαλείο επεξεργασίας κειμένου (όπως το **vi** ή κάποιο παρόμοιο) και να προσθέσετε απευθείας τη συγκεκριμένη γραμμή στο αρχείο **hosts** του Android. Χρησιμοποιώντας την εφαρμογή **Terminal Emulator** που υπάρχει στο Android VM ανοίξτε το αρχείο ως εξής:

```
$ su root
# vi /system/etc/hosts
```

Έπειτα προσθέστε την παρακάτω γραμμή στο τέλος του αρχείου:

```
<Ubuntu IP Address> www.repackagingattacklab.com
```

όπου <Ubuntu IP Address> είναι η IP διεύθυνση της εικονικής μηχανής **SEEDUbuntu VM**.

Εναλλακτικά, εάν δεν θέλετε να κάνετε την προσθήκη απευθείας στο Android, μπορείτε να αντιγράψετε το αρχείο στη μηχανή του επιτιθέμενου (**SEEDUbuntu VM**), να κάνετε τις αλλαγές και, στη συνέχεια, να αντιγράψετε το αρχείο ξανά στο **Android VM**. Οι ακόλουθες εντολές θα πρέπει να εκτελεστούν στο terminal της μηχανής **SEEDUbuntu VM**:

```
// Εκκίνηση του adb daemon με δικαιώματα root
$ adb root
// Σύνδεση με την εικονική μηχανή Android VM
$ adb connect <Android VM IP>
// Λήψη του αρχείου hosts από το Android VM
$ adb pull /system/etc/hosts
// Ρύθμιση του αρχείου hosts όπως προηγουμένως με χρήση του gedit
$ gedit ./hosts
// Φόρτωση του νέου αρχείου hosts στο Android VM
$ adb push ./hosts /system/etc/hosts
```

Βήμα 2: Επανασυσκευασία (repackaging) και εγκατάσταση της εφαρμογής-στόχου

Για να επανασυσκευάσετε την εφαρμογή μπορείτε να ακολουθήσετε τις οδηγίες των προηγούμενων δραστηριοτήτων. Αρχικά, θα πρέπει να απεγκαταστήσετε από το Android VM την εφαρμογή-στόχο που χρησιμοποιήσατε στο προηγούμενο σενάριο. Θα χρησιμοποιήσουμε την ίδια εφαρμογή-στόχο για να υλοποιήσουμε και το νέο σενάριο επίθεσης. Για ευκολία μπορείτε να χρησιμοποιήσετε μια εντελώς καθαρή έκδοση της εφαρμογής-στόχου, όπως την κατεβάσατε αρχικά. Επίσης, θα πρέπει να κατεβάσετε από το `eclass` τρία νέα αρχεία **smali**, τα οποία περιέχουν τον κακόβουλο κώδικα:

- **MaliciousCode.smali**,
- **SendData\$1.smali**, και
- **SendData.smali**

Αφού αποσυναρμολογήσετε την εφαρμογή-στόχο (με το **apktool**), θα πρέπει να τοποθετήσετε τα αρχεία **smali** στο φάκελο **smali/com/mobiseed/repackaging**. Έπειτα, θα πρέπει να τροποποιήσετε κατάλληλα το αρχείο **AndroidManifest.xml**, μιας και ο κακόβουλος κώδικας απαιτεί τρεις άδειες που σχετίζονται με την **τοποθεσία** και μια για **πρόσβαση στο Διαδίκτυο**. Επίσης, θα πρέπει να καταχωρηθεί ο **BroadcastReceiver**. Όλες οι προσθήκες φαίνονται στο παρακάτω τμήμα κώδικα:

```

<manifest...>
    ...
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_mock_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET"/>
    ....
    <application>
        .....
        .....
        <receiver android:name="com.mobiseed.repackaging.MaliciousCode" >
            <intent-filter>
                <action android:name="android.intent.action.TIME_SET" />
            </intent-filter>
        </receiver>

    </application>
</manifest>

```

Τέλος, ακολουθώντας τα βήματα του προηγούμενου σεναρίου, θα πρέπει να επανασυναρμολογήσετε την εφαρμογή **RepackagingLab**, να την υπογράψετε ψηφιακά και να την εγκαταστήσετε στο **Android VM** με χρήση του **adb**.

Βήμα 3: Ενεργοποίηση των αδειών (permissions) στο Android VM

Η προσθήκη των απαιτούμενων αδειών τοποθεσίας στο αρχείο **AndroidManifest** προφανώς δεν αρκεί για την ορθή λειτουργία της εφαρμογής. Όταν μια εφαρμογή εγκαθίσταται μέσω ενός app store (ή κατά την πρώτη χρήση της εφαρμογής), οι χρήστες συνήθως ερωτώνται εάν θα παραχωρήσουν τις άδειες τοποθεσίας στην εφαρμογή. Εφόσον οι χρήστες εγκρίνουν, το Android θα ενημερωθεί ότι έχουν παραχωρηθεί τα δικαιώματα και η εφαρμογή θα τρέχει κανονικά. Ωστόσο, όταν εγκαθιστούμε την εφαρμογή μέσω του εργαλείου **adb**, δεν ζητείται η έγκριση του χρήστη σχετικά με τις άδειες, ούτε ενεργοποιούνται αυτόματα τα σχετικά δικαιώματα.

Γι' αυτό, για τις ανάγκες του σεναρίου της άσκησης, θα πρέπει να ρυθμίσετε χειροκίνητα την άδεια τοποθεσίας της εφαρμογής-στόχου στο Android VM, ακολουθώντας την παρακάτω διαδρομή:

```
Settings -> Apps -> <Repackaging Lab> -> Permissions -> toggle location on
```

Βήμα 4: Ενεργοποίηση της επίθεσης

Αρχικά θα πρέπει να εκτελέσετε τουλάχιστον μία φορά την **εφαρμογή-στόχο (RepackagingLab)**, ώστε να ενεργοποιηθεί ο **BroadcastReceiver**. Έπειτα, θα πρέπει να εκτελέσετε την εφαρμογή εικονικής τοποθεσίας (**MockLocation** - βλέπε Εικόνα 4) και

επιλέξτε μια από τις διαθέσιμες τοποθεσίες. Τέλος, θα πρέπει να αλλάξετε την ώρα (χειροκίνητα) στο Android VM, ακολουθώντας τη διαδρομή:

Settings -> date and time -> set time

Βήμα 5: Παρακολούθηση του θύματος

Μόλις ενεργοποιηθεί ο κακόβουλος κώδικας, θα πρέπει να επιστρέψετε στη μηχανή του επιτιθέμενου (**SEEDUbuntu VM**), και να επισκεφτείτε τη διεύθυνση <http://www.repackagingattacklab.com> στο πρόγραμμα περιήγησης Firefox. Εάν η επίθεσή είναι επιτυχής, θα πρέπει να μπορείτε να παρακολουθείτε την τοποθεσία του θύματος.

Για να ελέγξετε εάν ο κακόβουλος κώδικας μπορεί να παρακολουθεί σωστά την κίνηση του θύματος, μπορείτε να επιστρέψετε στο Android VM, να αλλάξετε την τοποθεσία στην εφαρμογή **MockLocation**, και έπειτα να αλλάξετε ξανά την ώρα (χειροκίνητα). Επιστρέφοντας στον browser του επιτιθέμενου και κάνοντας refresh στη διεύθυνση www.repackagingattacklab.com θα πρέπει να δείτε τη νέα τοποθεσία.

Στην αναφορά σας θα πρέπει να περιγράψετε την εκτέλεση των παραπάνω ενέργειων και να δείξετε με σχετικά στιγμιότυπα οθόνης (από το Android VM και το SEEDUbuntu VM) τα αποτελέσματα της εκτέλεσης.

Υποβολή

Θα πρέπει να υποβάλετε μια λεπτομερή **εργαστηριακή αναφορά** (σε PDF) που θα περιγράφει όλες τις ενέργειες που κάνατε σε κάθε δραστηριότητα, τα αποτελέσματα και τις παρατηρήσεις σας. Στην περιγραφή σας χρησιμοποιείτε στιγμιότυπα οθόνης και αποσπάσματα κώδικα, σύμφωνα με τις οδηγίες κάθε δραστηριότητας.