"ANDROID SECURITY"

1. Εισαγωγή

Η κοινοπραξία Open Handset Alliance παρουσίασε το Google Android SDK στις 12 Νοεμβρίου 2007, το οποίο είχε ανακοινώσει μία εβδομάδα νωρίτερα. Ο αντίκτυπος ήταν απίστευτος, σχεδόν κάθε site του διαδικτύου που ασχολείται με τον προγραμματισμό έκανε μία αναφορά για την παρουσίαση του SDK. Τα Group της Google κατακλύστηκαν με περισσότερα από 2000 μηνύματα μέσα στις δύο πρώτες μέρες.

Η ιδέα της πλατφόρμας Android ήταν και είναι καταπληκτική και φυσικά προσελκύει όλο και περισσότερους προγραμματιστές κάθε μέρα. Ειδικά, η ανοιχτή αρχιτεκτονική που βασίζεται στα *Intents* (*Προθέσεις*) και η δυνατότητα αντικατάστασης ακόμα και της εφαρμογής της Αρχικής Οθόνης, εγγυάται πολύ μεγάλη ευελιξία σε όλη την πλατφόρμα.

Σε αυτό το έγγραφο θα αναλύσουμε τα βασικά στοιχεία του Android (*Κεφάλαια 2-4*) καθώς και το πως επιτυγχάνεται η ασφάλεια (*Κεφάλαιο 5*).

"ANDROID – IMAGINATION IS THE LIMIT"

Nicolas Gramlich – anddev.org Site-Admin

2. Τι είναι το Android?

Πριν να παρουσιάσει η Google το Android SDK υπήρχαν πολλές φήμες ότι θα πρόκειται για ένα GPhone (δηλ. Google Phone). Λεγόταν ότι θα είναι ένα κινητό τηλέφωνο κατασκευασμένο από την Google και θα παρέχει δωρεάν επικοινωνία, δείχνοντας διαφημίσεις στον χρήστη.

Το ενδεχόμενο GPhone

Όμως στις 5 Νοεμβρίου 2007 ανακοινώθηκε από τα Google Groups ότι:

"[Η Πλατφόρμα] Android – είναι πιο σπουδαία και φιλόδοξη από ένα μόνο τηλέφωνο."

Η Google με τη συνεισφορά της Open Handset Alliance (OHA) προσφέρει μία πλήρη συλλογή λογισμικού για κινητές συσκευές: *ένα λειτουργικό σύστημα, ένα middleware και κάποιες βασικές εφαρμογές κινητού*. Αυτό που παρουσιάστηκε μία εβδομάδα αργότερα δεν ήταν ένα τελικό προϊόν αλλά ένα "First Look SDK", πράγμα που πολλοί δεν είχαν συνειδητοποιήσει. Έτσι πολλά site άρπαξαν τα λεγόμενα κάποιων developer οι οποίοι είπαν ότι το Android είναι γεμάτο bugs και έχει έλλειψη τεκμηρίωσης. Όμως η πλειοψηφία παραδέχτηκε ότι το Android δεν είναι πιο προβληματικό από άλλα λογισμικά σε αντίστοιχη φάση της ζωής τους.

2.1. Οι προϋποθέσεις του Android

Ας ρίξουμε μια ματιά σε τι δίνει έμφαση η OHA στην πλατφόρμα Android:

- ***Openness***

> *"Το Android χτίστηκε από τη βάση του να επιτρέπει στους developers να δημιουργούν εφαρμογές που να εκμεταλλεύονται πλήρως αυτά που έχει να προσφέρει μία κινητή συσκευή. Χτίστηκε ώστε να είναι εντελώς ανοιχτό. Για παράδειγμα, μία εφαρμογή θα μπορούσε να πάρει τη θέση μίας κύριας λειτουργίας του Android, όπως η πραγματοποίηση κλήσεων, η αποστολή μηνυμάτων, ή η χρησιμοποίηση της κάμερας, επιτρέποντας στους developers να δημιουργήσουν πιο πλούσια εμπειρία για τους χρήστες."*

Αυτό είναι αλήθεια, ως developer μπορείς να κάνεις τα πάντα, από το να στέλνεις μηνύματα SMS με μόνο 2 γραμμές κώδικα, μέχρι να αντικαταστήσεις την Αρχική Οθόνη της συσκευής. Κάποιος θα μπορούσε εύκολα να δημιουργήσει ένα πλήρως προσαρμοσμένο λειτουργικό σύστημα σε μερικές εβδομάδες, το οποίο δεν θα παρέχει καμία από τις αρχικές εφαρμογές της Google στο χρήστη.

> *"Το Android χτίστηκε πάνω στο πυρήνα του Linux. Επιπλέον, χρησιμοποιεί μια προσαρμοσμένη εικονική μηχανή που έχει σχεδιαστεί για τη βελτιστοποίηση της μνήμης και των πόρων του υλικού σε ένα κινητό περιβάλλον. Το Android θα είναι open source. Μπορεί εύκολα να επεκταθεί έτσι ώστε να ενσωματώσει τις νέες τεχνολογίες αιχμής όταν αυτές προκύπτουν. Η πλατφόρμα θα συνεχίσει να εξελίσσεται όσο η κοινότητα των developer συνεργάζεται για την ανάπτυξη καινοτόμων εφαρμογών κινητών."*

Εδώ η Google φέρνει την επονομαζόμενη Dalvik Virtual Machine (*DalvikVM*), η οποία είναι μία εικονική μηχανή που βασίζεται σε καταχωρητές (*register based*). Σχεδιάστηκε και γράφτηκε από τον *Dan Bornstein* και μερικούς άλλους μηχανικούς

της Google, ώστε να είναι ένα σημαντικό μέρος της πλατφόρμας Android. Στις λέξεις "*register based*" βλέπουμε την πρώτη διαφορά με τις συνήθεις Java Virtual Machines (JVM) οι οποίες βασίζονται σε στοίβες (*stack based*).

- ***Όλες οι εφαρμογές είναι ισότιμες***

  "Το Android δεν διαφοροποιεί τις κύριες εφαρμογές του τηλεφώνου από εκείνες που έχουν δημιουργηθεί από τρίτους. Μπορούν όλες να κατασκευαστούν ώστε να έχουν την ίδια πρόσβαση στις δυνατότητες του τηλεφώνου παρέχοντας στους χρήστες ένα ευρύ φάσμα εφαρμογών και υπηρεσιών. Στις συσκευές που τρέχουν την πλατφόρμα Android, οι χρήστες θα μπορούν να προσαρμόσουν πλήρως το τηλέφωνο στις ανάγκες τους. Μπορούν να αλλάξουν την Αρχική Οθόνη, το στυλ του dialer, ή οποιαδήποτε εφαρμογή. Μπορούν επίσης να δώσουν εντολή στα κινητά τους να χρησιμοποιούν την αγαπημένη τους εφαρμογή επισκόπησης εικόνων για να ανοίγει όλες τις φωτογραφίες."

Το σύστημα επικοινωνίας του Android βασίζεται στα αποκαλούμενα *Intents* (*Προθέσεις*), τα οποία είναι κάτι περισσότερο ή λιγότερο από ένα String (με κάποια δεδομένα προσαρτημένα) και καθορίζουν μία ενέργεια που πρέπει να χειριστεί. Ένα παράδειγμα είναι το εξής:

```
"android.provider.Telephony.SMS_RECEIVED"
```

Κάποιος θα μπορούσε να «ακούσει» αυτό το Intent γράφοντας περίπου 5 γραμμές ορισμών. Το σύστημα έπειτα θα αναγνώριζε ότι υπάρχουν περισσότερες από μία εφαρμογές που θέλουν να χειριστούν αυτό το Intent και θα ρωτούσε το χρήστη να διαλέξει ποια θα ήθελε να το χειριστεί.

- ***Εξάλειψη των ορίων της εφαρμογής***

  "Το Android εξαλείφει τα όρια για την κατασκευή νέων και καινοτόμων εφαρμογών. Για παράδειγμα, ένας developer μπορεί να συνδυάσει πληροφορίες από το web με δεδομένα ενός μεμονωμένου κινητού τηλεφώνου, όπως οι επαφές, το ημερολόγιο ή η γεωγραφική θέση του χρήστη, για να παρέχει μία πιο σχετική-με-τον-χρήστη εμπειρία. Με το Android, ένας developer θα μπορούσε να φτιάξει μία εφαρμογή που θα επιτρέπει στους χρήστες να βλέπουν τη θέση των φίλων τους και να ειδοποιούνται όταν βρίσκονται κοντά, δίνοντάς τους τη δυνατότητα να συναντηθούν."

- ***Γρήγορη και εύκολη ανάπτυξη λογισμικού***

  "Το Android παρέχει πρόσβαση σε ένα μεγάλο πλήθος χρήσιμων βιβλιοθηκών και εργαλείων, τα οποία μπορούν να χρησιμοποιηθούν για την ανάπτυξη πλούσιων εφαρμογών. Για παράδειγμα, το Android καθιστά ικανούς τους developers να λάβουν τη θέση της συσκευής και επιτρέπει στις συσκευές να επικοινωνήσουν η

μία με την άλλη παρέχοντας έτσι τη δυνατότητα για πλούσιες κοινωνικές εφαρμογές peer-to-peer. Επιπρόσθετα, το Android περιλαμβάνει ένα πλήρες σύνολο εργαλείων που έχουν φτιαχτεί από τη βάση τους, παράλληλα με την πλατφόρμα, παρέχοντας στους developers μεγάλη παραγωγικότητα και βαθειά γνώση στις εφαρμογές τους."

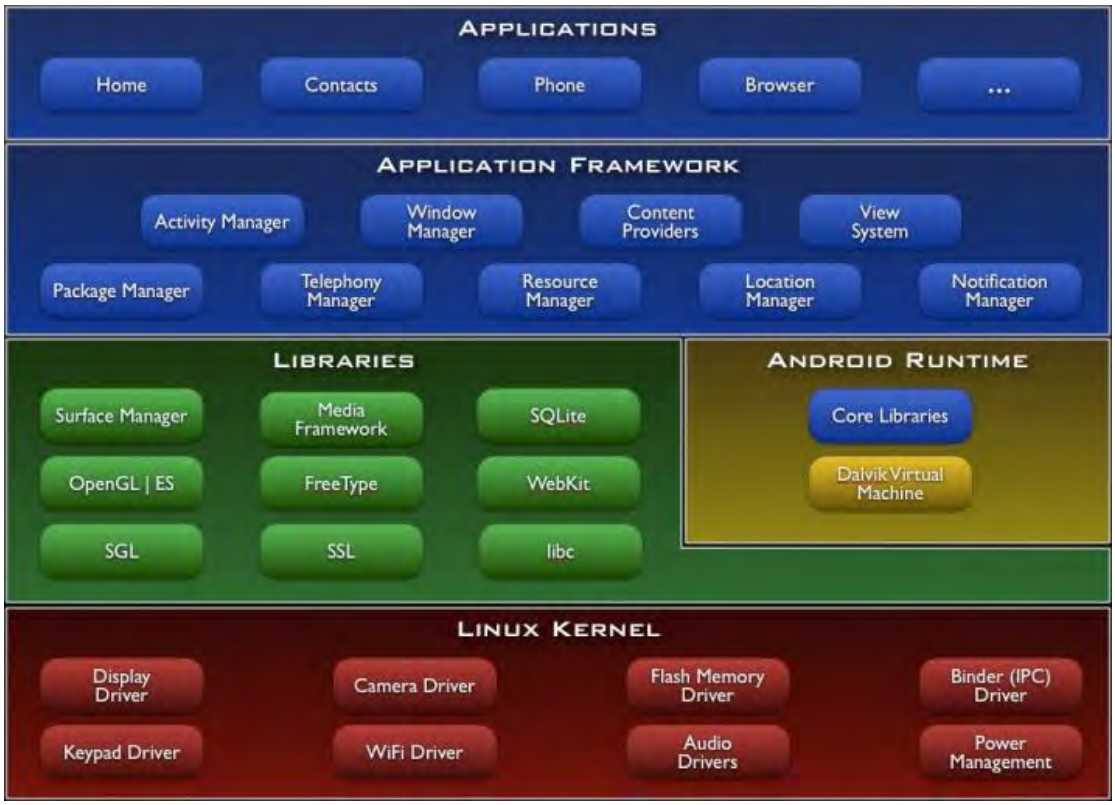
Με την επανάσταση του Web 2.0, η ανάπτυξη εφαρμογών πλούσιων σε περιεχόμενο μέσα σε μερικά λεπτά δεν ήταν πια ψευδαίσθηση. Το Android έφερε την ανάπτυξη σε πολύ μεγάλες ταχύτητες.

Η Google δίνει έμφαση στη δύναμη του Android να παρέχει υπηρεσίες βασισμένες στην τοποθεσία (*location-based-services*). Το *Google Maps* είναι τόσο πετυχημένο στο Android σαν να είχε αναπτυχθεί μόνο γι' αυτό. Κάποιος μπορεί να ενσωματώσει ένα χάρτη με ζουμ και δυνατότητα τραβήγματος, προσθέτοντας μόνο 3 (!) χαρακτήρες στον κώδικα Java και 3 γραμμές στον κώδικα XML.

Άλλα ωραία χαρακτηριστικά που είναι εύκολα στη χρήση με το Android είναι τα Animations και η αναπαραγωγή πολλών αρχείων media.

3. Η Αρχιτεκτονική του Android

Το ακόλουθο διάγραμμα παρουσιάζει τα κυριότερα συστατικά του λειτουργικού συστήματος Android. Κάθε τμήμα περιγράφεται αναλυτικότερα παρακάτω.



➢ *Applications*

Το Android κυκλοφορεί με ένα σύνολο βασικών εφαρμογών, περιλαμβάνοντας ένα email client, μία εφαρμογή SMS, ημερολόγιο, χάρτες, φυλλομετρητή, επαφές κ.α. Όλες οι εφαρμογές γράφονται χρησιμοποιώντας τη γλώσσα προγραμματισμού Java.

➢ *Application Framework*

Με την παροχή μίας ανοιχτής πλατφόρμας ανάπτυξης, το Android προσφέρει στους developers τη δυνατότητα να κατασκευάσουν εξαιρετικά πλούσιες και καινοτόμες εφαρμογές. Οι developers μπορούν να επωφεληθούν από το υλικό της συσκευής, να έχουν πληροφορίες για την τοποθεσία, να εκτελούν υπηρεσίες στο παρασκήνιο, να θέτουν συναγερμούς, να προσθέτουν ειδοποιήσεις στη μπάρα κατάστασης και πολλά άλλα.

Οι developers έχουν πλήρη πρόσβαση στα ίδια APIs που χρησιμοποιήθηκαν από τις βασικές εφαρμογές. Η αρχιτεκτονική της εφαρμογής έχει σχεδιαστεί για να απλοποιήσει την επαναχρησιμοποίηση των *components* (*συστατικών*). Κάθε εφαρμογή μπορεί να δημοσιεύσει τις δυνατότητές της και έτσι οποιαδήποτε άλλη εφαρμογή μπορεί να κάνει χρήση αυτών (με την επιφύλαξη περιορισμών ασφαλείας που επιβάλλονται από το framework). Ο ίδιος μηχανισμός επιτρέπει στο χρήστη να αντικαταστήσει components.

Στη βάση όλων των εφαρμογών βρίσκεται ένα σύνολο από υπηρεσίες και συστήματα, συμπεριλαμβανομένων των εξής:

- Ένα πλούσιο και επεκτάσιμο σύνολο από *Views* που μπορούν να χρησιμοποιηθούν για τη δημιουργία μιας εφαρμογής, όπως *λίστες (lists), πλέγματα (grids), πλαίσια κειμένου (text boxes), κουμπιά (buttons)*, ακόμα και έναν *embeddable web browser*.

- *Content Providers* (*Παρόχους Περιεχομένου*) που επιτρέπουν στις εφαρμογές να έχουν πρόσβαση σε δεδομένα από άλλες εφαρμογές (όπως οι *Επαφές*) ή να μοιράζονται τα δικά τους δεδομένα.

- Έναν *Resource Manager* (*Διαχειριστή Πόρων*) που παρέχει πρόσβαση σε πηγές χωρίς κώδικα (*non-code resources*) όπως strings με βάση την τοποθεσία, γραφικά και αρχεία διάταξης (*layout files*).

- Έναν *Notification Manager* (*Διαχειριστή Ειδοποιήσεων*) που επιτρέπει σε όλες τις εφαρμογές να απεικονίζουν προσαρμοσμένες ειδοποιήσεις στη μπάρα κατάστασης.

- Έναν *Activity Manager* (*Διαχειριστή Δραστηριοτήτων*) που διαχειρίζεται τον κύκλο ζωής (*lifecycle*) των εφαρμογών και παρέχει ένα συνηθισμένο ιστορικό μετάβασης (*navigation backstack*).

➢ *Libraries*

Το Android περιλαμβάνει ένα σύνολο από βιβλιοθήκες C/C++ που χρησιμοποιούνται από διάφορα components του συστήματος. Αυτές οι δυνατότητες εκτίθενται στους developers μέσω του Android application framework. Μερικές από τις βασικές βιβλιοθήκες αναφέρονται παρακάτω:

- **System C library** – μία υλοποίηση προερχόμενη από το BSD, της επίσημης βιβλιοθήκης συστήματος C (libc), βελτιστοποιημένη για ενσωματωμένες συσκευές που βασίζονται στο Linux.

- **Media Libraries** – βασισμένες στο OpenCORE της PacketVideo. Οι βιβλιοθήκες υποστηρίζουν την αναπαραγωγή και καταγραφή πολλών δημοφιλών μορφών ήχου και βίντεο, καθώς και στατικών αρχείων εικόνας. Συμπεριλαμβάνονται τα MPEG4, H.264, MP3, AAC, AMR, JPG, και PNG.

- **Surface Manager** – διαχειρίζεται την πρόσβαση στο υποσύστημα απεικόνισης και συνθέτει στρώματα 2D και 3D γραφικών από πολλαπλές εφαρμογές.

- **LibWebCore** – μία σύγχρονη μηχανή web browser η οποία χρησιμοποιείται και από τον Android browser και από τον embeddable web browser.

- **SGL** – η βασική μηχανή 2D γραφικών

- **3D libraries** – μια υλοποίηση βασισμένη στα APIs του OpenGL ES 1.0. Οι βιβλιοθήκες χρησιμοποιούν είτε την επιτάχυνση υλικού για 3D (όπου είναι διαθέσιμο) ή το πολύ καλά βελτιστοποιημένο λογισμικό απεικόνισης 3D (rasterizer).

- **FreeType** – bitmap και vector γραμματοσειρά φωτοσκίασης.

- **SQLite** – μία ισχυρή και ελαφριά σχεσιακή βάση δεδομένων, διαθέσιμη σε όλες τις εφαρμογές.

➢ *Android Runtime*

- ***Core Libraries***

Το Android περιλαμβάνει ένα σύνολο βασικών βιβλιοθηκών που παρέχουν τις περισσότερες από τις διαθέσιμες λειτουργίες των βασικών βιβλιοθηκών της Java.

- ***Dalvik Virtual Machine***

Γιατί "Dalvik"; Η Dalvik Virtual Machine πήρε το όνομά της από τον *Bornstein* ο οποίος ψάρευε στο χωριό Dalvík του Eyjafjörður (Ισλανδία), όπου ζούσαν κάποιοι πρόγονοί του.

Η Dalvik είναι μία εικονική μηχανή διερμηνέας, η οποία εκτελεί αρχεία της μορφής *.dex (Dalvik Executable), μια μορφή που είναι βελτιστοποιημένη για αποδοτική αποθήκευση και εκτέλεση με χαρτογραφημένη μνήμη (memory-mappable). Η εικονική μηχανή βασίζεται σε καταχωρητές και μπορεί να τρέξει κλάσεις που μεταγλωττίστηκαν από έναν Java compiler και έχουν μετασχηματιστεί στη δική της φυσική μορφή, χρησιμοποιώντας το παρεχόμενο εργαλείο *"dx".* Η VM τρέχει πάνω στον πυρήνα του Linux 2.6, στον οποίο βασίζεται για την υποκείμενη λειτουργικότητα (όπως η διαχείριση threads και η διαχείριση μνήμης σε χαμηλό επίπεδο). Κάθε εφαρμογή Android τρέχει στη δική της διαδικασία (*process*), με το δικό της στιγμιότυπο (*instance*) της Dalvik VM. Η DalvikVM βελτιστοποιήθηκε επίσης για να τρέχει σε πολλαπλά στιγμιότυπα με πολύ μικρή χρήση μνήμης. Μια σειρά από VM προστατεύουν μια εφαρμογή από το να υπολειτουργήσει εξαιτίας μιας άλλης εφαρμογής που «κόλλησε».

Διαφορές από μια κανονική JavaVM

Η JavaVM , που είναι πλέον εγκατεστημένη σχεδόν σε όλους τους προσωπικούς υπολογιστές, είναι βασισμένη σε «στοίβες» (stack based). Η DalvikVM από την άλλη είναι βασισμένη σε καταχωρητές (registered based), γιατί οι επεξεργαστές για συσκευές όπως τα κινητά είναι βελτιστοποιημένοι για εκτέλεση εφαρμογών με χρήση καταχωρητών. Επίσης οι VM που βασίζονται σε καταχωρητές επιτρέπουν γρηγορότερη εκτέλεση συχνά σε βάρος προγραμμάτων που είναι μεγαλύτερα σε μέγεθος μετά την μεταγλώττισή τους (*compilation*).

➢ *Linux Kernel*

Το Android βασίζεται στον πυρήνα του Linux 2.6 για τις βασικές υπηρεσίες του συστήματος όπως η ασφάλεια, η διαχείριση μνήμης, η διαχείριση διαδικασιών, η στοίβα δικτύου (network *stack*) και οι οδηγοί (*drivers*). Επίσης ο πυρήνας δρα ως ένα αφαιρετικό επίπεδο (*abstraction layer*) μεταξύ του υλικού και της υπόλοιπης στοίβας λογισμικού.

4. Ανατομία μιας εφαρμογής Android

Υπάρχουν 4 τμήματα κατασκευής (*building blocks*) σε μια εφαρμογή Android:

- *Activity (Δραστηριότητα)*
- *Intent Receiver (Δέκτης Πρόθεσης)*
- *Service (Υπηρεσία)*
- *Content Provider (Πάροχος Περιεχομένου)*

Δεν είναι απαραίτητο να υπάρχουν και τα 4 τμήματα σε μια εφαρμογή, αλλά κάθε εφαρμογή χρησιμοποιεί ένα συνδυασμό αυτών.

Από την στιγμή που θα αποφασιστεί ποια τμήματα χρειάζονται για την εφαρμογή, θα πρέπει να οριστούν σε ένα αρχείο που ονομάζεται `AndroidManifest.xml`. Αυτό είναι ένα XML αρχείο όπου δηλώνονται τα τμήματα της εφαρμογής και ποιές είναι οι δυνατότητες και οι απαιτήσεις τους.

➢ *Activity (Δραστηριότητα)*

Τα Activities είναι τα πιο κοινά από τα 4 τμήματα που προαναφέρθηκαν. Μια δραστηριότητα είναι συνήθως μια απλή οθόνη της εφαρμογής. Κάθε δραστηριότητα υλοποιείται σαν μια κλάση που επεκτείνει (*extends*) την βασική κλάση *Δραστηριότητα* (*Activity base class*). Η δική σας κλάση θα προβάλει μια *διεπαφή χρήστη* (*user interface*) αποτελούμενη από *Εικόνες* (*Views*) και θα απαντά σε *Συμβάντα* (*Events*). Οι περισσότερες εφαρμογές αποτελούνται από πολλαπλές οθόνες. Για παράδειγμα, μια εφαρμογή ανταλλαγής γραπτών μηνυμάτων, θα μπορούσε να έχει μια οθόνη που δείχνει μια λίστα από τις επαφές για να διαλέξετε σε ποιον θα στείλετε το μήνυμα, μια δεύτερη οθόνη για να γράφετε το μήνυμα στην επιλεγμένη επαφή και άλλες οθόνες για να βλέπετε παλιά μηνύματα ή να αλλάζετε τις ρυθμίσεις. Κάθε μια από αυτές τις οθόνες θα υλοποιούταν σαν μια δραστηριότητα. Η μετάβαση σε άλλη οθόνη επιτυγχάνεται με την έναρξη μιας νέας δραστηριότητας. Σε μερικές περιπτώσεις μια δραστηριότητα μπορεί να επιστρέφει μια τιμή σε μια προηγούμενη δραστηριότητα – για παράδειγμα μια δραστηριότητα που επιτρέπει στο χρήστη να επιλέξει μια φωτογραφία θα επέστρεφε την επιλεγμένη φωτογραφία στην δραστηριότητα που την κάλεσε.

Όταν μια νέα οθόνη ανοίγει, η προηγούμενη οθόνη μπαίνει σε μια στοίβα που κρατά το ιστορικό (*history stack*). Ο χρήστης μπορεί να πλοηγηθεί πίσω σε οθόνες που άνοιξε προηγουμένως μέσω του ιστορικού. Οι οθόνες μπορούν επίσης να αφαιρεθούν από το ιστορικό σε περιπτώσεις που δεν χρειάζεται να παραμείνουν προσβάσιμες. Το Android διατηρεί ιστορικό για κάθε εφαρμογή που εκκίνησε από την κεντρική οθόνη (*home screen*).

➢ *Intent and Intent Filters (Πρόθεση και Φίλτρα Πρόθεσης)*

Το Android χρησιμοποιεί μια ειδική κλάση που λέγεται *Πρόθεση* (*Intent*) για να κινείται από οθόνη σε οθόνη. Η *Πρόθεση* περιγράφει τι θέλει η εφαρμογή να γίνει στη συνέχεια. Τα δυο πιο σημαντικά μέρη της δομής δεδομένων της *Πρόθεσης* είναι η δράση (*action*) και τα δεδομένα βάσει των οποίων αυτή θα εκτελεστεί. Τυπικές τιμές για μια δράση είναι η MAIN (η κεντρική είσοδος της εφαρμογής), η VIEW, η PICK, η EDIT κλπ. Τα δεδομένα εκφράζονται ως URI (*Uniform Resource Indicator*). Για παράδειγμα, για να δείτε μια ιστοσελίδα στον browser, θα δημιουργούσατε ένα Intent με δράση VIEW και τα δεδομένα ως ένα website-URI.

```
new Intent(android.content.Intent.VIEW_ACTION,
     ContentURI.create("http://anddev.org"));
```

Υπάρχει μια σχετική κλάση που λέγεται *Φίλτρο Πρόθεσης* (*IntentFilter*). Ενώ μια *Πρόθεση* είναι στην ουσία ένα αίτημα για να γίνει κάτι, το *Φίλτρο Πρόθεσης* είναι μια περιγραφή του τι είναι δυνατόν να διαχειριστεί ένας *Δέκτης Πρόθεσης* (*intent receiver*). Μια δραστηριότητα που είναι σε θέση να προβάλει πληροφορίες επικοινωνίας για ένα άτομο θα ανακοίνωνε με ένα *Φίλτρο Πρόθεσης* (*IntentFilter*) ότι γνωρίζει πως να διαχειριστεί την VIEW_ACTION όταν τα δεδομένα αντιπροσωπεύουν ένα άτομο. Οι δραστηριότητες ανακοινώνουν τα *Φίλτρα Πρόθεσης* στο αρχείο `AndroidManifest.xml`.

Η πλοήγηση από οθόνη σε οθόνη πετυχαίνεται με *Προθέσεις*. Για να πλοηγηθείτε προς τα εμπρός, μια δραστηριότητα καλεί την `startActivity(myIntent)`. Το σύστημα τότε κοιτά στα *Φίλτρα Προθέσεων* (*intent filters*) για όλες τις εγκατεστημένες εφαρμογές και διαλέγει την δραστηριότητα που τα *Φίλτρα Πρόθεσης* ταιριάζουν καλύτερα με την `myIntent` παράμετρο της κλήσης. Η νέα *Δραστηριότητα* ενημερώνεται για την *Πρόθεση* και ξεκινά. Η διαδικασία της υλοποίησης των *Προθέσεων* συμβαίνει κατά τον χρόνο εκτέλεσης της εφαρμογής όταν καλείται η `startActivity`, πράγμα που προσφέρει 2 βασικά πλεονεκτήματα:

- Οι *Δραστηριότητες* μπορούν να επαναχρησιμοποιούν κάποια λειτουργικότητα από άλλα τμήματα του κώδικα απλά κάνοντας ένα αίτημα υπό την μορφή μιας *Πρόθεσης*.
- Οι *Δραστηριότητες* μπορούν να αντικατασταθούν οποιαδήποτε στιγμή από μια νέα *Δραστηριότητα* με ένα αντίστοιχο *Φίλτρο Πρόθεσης*.

➢ *Intent Receiver (Δέκτης Πρόθεσης)*

Μπορείτε να χρησιμοποιήσετε ένα `IntentReceiver` όταν θέλετε η εφαρμογή σας να εκτελεστεί σε απάντηση ενός εξωτερικού συμβάντος (*external event*), για

παράδειγμα, όταν το τηλέφωνο χτυπά, ή όταν το δίκτυο είναι διαθέσιμο, ή όταν είναι μεσάνυχτα. Οι *Δέκτες Πρόθεσης* δεν προβάλουν μια *διεπαφή χρήστη* (*UI*), ωστόσο μπορούν να προβάλουν *Ειδοποιήσεις* (`Notifications`) για να ειδοποιήσουν τον χρήστη για κάτι σημαντικό που συνέβη. Οι *Δέκτες Πρόθεσης* είναι επίσης καταχωρημένοι στο `AndroidManifest.xml`, αλλά μπορείτε επίσης να τους καταχωρήσετε από τον κώδικα χρησιμοποιώντας την `Context.registerReceiver()`. Η εφαρμογή σας δεν χρειάζεται να τρέχει για να κληθούν οι *Δέκτες Πρόθεσης* που έχει. Το σύστημα θα εκκινήσει την εφαρμογή σας, αν χρειαστεί, όταν ένας *Δέκτης Πρόθεσης* ενεργοποιηθεί. Οι εφαρμογές μπορούν επίσης να στέλνουν τις δικές τους *Ανακοινώσεις Πρόθεσης* (*intent broadcasts*) σε άλλους με την `Context.broadcastIntent()`.

➢ *Service (Υπηρεσία)*

Ένα `Service` είναι κώδικας που τρέχει για μεγάλο χρονικό διάστημα και χωρίς *διεπαφή χρήστη* (*UI*). Ένα καλό παράδειγμα είναι μια εφαρμογή που αναπαράγει μουσική από μια λίστα μουσικών κομματιών (*media player*). Σε μια τέτοια εφαρμογή, θα υπήρχαν κατά πάσα πιθανότητα μία ή και παραπάνω *Δραστηριότητες* που επιτρέπουν στον χρήστη να επιλέξει τραγούδια και να τα αναπαράξει. Ωστόσο, η αναπαραγωγή από μόνη της δεν θα έπρεπε να διαχειρίζεται από την *Δραστηριότητα* γιατί ο χρήστης θα περίμενε την μουσική να συνεχίζει να παίζει ακόμη και μετά την πλοήγησή του σε μια νέα οθόνη. Σε αυτή τη περίπτωση, η *Δραστηριότητα* της αναπαραγωγής μουσικής θα ξεκινούσε μια υπηρεσία χρησιμοποιώντας την `Context.startService()` για να τρέξει στο φόντο και να συνεχίσει η μουσική να παίζει. Το σύστημα τότε θα κρατά την υπηρεσία αναπαραγωγής ενεργή μέχρι να τελειώσει το κομμάτι. Σημειώστε ότι μπορείτε να συνδεθείτε σε μια υπηρεσία (και να την εκκινήσετε αν δεν έχει ξεκινήσει) με την μέθοδο `Context.bindService()`. Όταν συνδεθείτε σε μια υπηρεσία, μπορείτε να επικοινωνήσετε με αυτή μέσω μιας διεπαφής που προσφέρεται από την υπηρεσία. Για την υπηρεσία μουσικής, αυτό θα σας επέτρεπε να κάνετε παύση, να πάτε πίσω στο κομμάτι (rewind) κλπ.

➢ *Content Provider (Πάροχος Περιεχομένου)*

Οι εφαρμογές μπορούν να σώζουν τα δεδομένα τους σε αρχεία, σε μια SLQLite βάση δεδομένων, σε *προτιμήσεις* (*preferences*) ή σε οποιοδήποτε άλλο μηχανισμό μπορούν. Ένας *Πάροχος Περιεχομένου*, ωστόσο, είναι χρήσιμος αν θέλετε τα δεδομένα της εφαρμογής σας να είναι διαθέσιμα και σε άλλες εφαρμογές. Ένας *Πάροχος Περιεχομένου* είναι μια κλάση που υλοποιεί μια συγκεκριμένη ομάδα μεθόδων που επιτρέπουν σε άλλες εφαρμογές να αποθηκεύουν και να επανακτούν δεδομένα του τύπου που διαχειρίζεται ο *Πάροχος Περιεχομένου*.

5. Ασφάλεια και Άδειες

Το Android είναι ένα σύστημα πολλαπλών διαδικασιών (*multi-process*) στο οποίο κάθε εφαρμογή (και μέρη του συστήματος) τρέχουν στη δική τους διαδικασία. Η περισσότερη ασφάλεια μεταξύ των εφαρμογών και του συστήματος γίνεται σε επίπεδο διαδικασιών μέσω των διευκολύνσεων του Linux, όπως τα *user* και *group IDs* που εκχωρούνται στις εφαρμογές. Επιπρόσθετα χαρακτηριστικά ασφαλείας παρέχονται μέσω ενός μηχανισμού «*Αδειών*» (*Permission*), ο οποίος θέτει τους περιορισμούς για ειδικές λειτουργίες που μια συγκεκριμένη διαδικασία μπορεί να εκτελέσει και τα δικαιώματα που έχει καθένα *URI* για την επί τούτου χορήγηση πρόσβασης σε συγκεκριμένα τμήματα δεδομένων.

5.1. Η Αρχιτεκτονική της Ασφάλειας

Ένα κύριο σημείο στην αρχιτεκτονική της ασφάλειας του Android είναι ότι καμία εφαρμογή, από προεπιλογή, δεν έχει άδεια να κάνει οποιεσδήποτε λειτουργίες οι οποίες θα έχουν αρνητικές επιπτώσεις σε άλλες εφαρμογές, στο λειτουργικό σύστημα, ή στον χρήστη. Περιλαμβάνονται η ανάγνωση ή η εγγραφή των προσωπικών δεδομένων του χρήστη (όπως οι επαφές ή τα e-mail), η ανάγνωση ή η εγγραφή αρχείων άλλης εφαρμογής, η πρόσβαση στο δίκτυο, η διατήρηση αναμμένης της οθόνης της συσκευής, κλπ.

Η διαδικασία μιας εφαρμογής είναι ένα ασφαλές «*κουτί*» (*sandbox*). Δεν μπορεί να διαταράξει άλλες εφαρμογές, εκτός εάν δηλώσει ρητά τα δικαιώματα που χρειάζεται για πρόσθετες δυνατότητες που δεν παρέχονται από το βασικό «*κουτί*». Αυτά τα δικαιώματα που ζητάει μπορούν να διεκπεραιωθούν από το λειτουργικό με διάφορους τρόπους, συνήθως αυτόματα επιτρέποντας ή όχι βάση πιστοποιητικών, ή προτρέποντας τον χρήστη να αποφασίσει. Τα δικαιώματα που χρειάζονται από μία εφαρμογή δηλώνονται στατικά στην εφαρμογή, έτσι ώστε να είναι γνωστά κατά την εγκατάστασή της και δεν μπορούν να αλλάξουν μετέπειτα.

5.2. Υπογραφή της Εφαρμογής

Όλες οι Android εφαρμογές (αρχεία .apk) πρέπει να υπογράφονται με ένα πιστοποιητικό του οποίου το *ιδιωτικό κλειδί* κρατείται από τον developer τους. Το πιστοποιητικό δεν χρειάζεται να έχει υπογραφεί από μία αρχή πιστοποίησης και έτσι είναι απολύτως επιτρεπτό και σύνηθες για τις Android εφαρμογές, να χρησιμοποιούν αυτό-υπογραφόμενα πιστοποιητικά. Το πιστοποιητικό αυτό χρησιμοποιείται μόνο για την επίτευξη σχέσεων εμπιστοσύνης μεταξύ των εφαρμογών και όχι για το γενικό έλεγχο του κατά πόσο μία εφαρμογή μπορεί να εγκατασταθεί. Οι πιο σημαντικοί τρόποι που οι υπογραφές επιδρούν στην

ασφάλεια είναι με το να προσδιορίζεται ποιός μπορεί να έχει πρόσβαση στις άδειες βάση πιστοποιητικού και ποιός μπορεί να μοιράζει ID χρήστη.

5.3. ID Χρήστη και Πρόσβαση Αρχείων

Κάθε αρχείο Android package (.apk) που εγκαθίσταται στη συσκευή, λαμβάνει το δικό του μοναδικό *Linux ID χρήστη*, δημιουργείται γι' αυτό ένα «*κουτί*» και εμποδίζεται από το να έχει επαφή με άλλες εφαρμογές (ή οι άλλες εφαρμογές να έχουν επαφή μ' αυτό). Αυτό το *ID χρήστη* τού απονέμεται κατά την εγκατάσταση της εφαρμογής στη συσκευή και παραμένει σταθερό κατά τη διάρκεια της ζωής του σ' αυτή τη συσκευή.

Επειδή η επιβολή ασφάλειας συμβαίνει στο επίπεδο διαδικασίας, ο κώδικας οποιονδήποτε δύο packages δεν μπορεί να εκτελεστεί κανονικά στην ίδια διαδικασία, δεδομένου ότι πρέπει να τρέχουν ως διαφορετικοί χρήστες Linux. Μπορεί να χρησιμοποιηθεί το χαρακτηριστικό `sharedUserId` στην ετικέτα `manifest` του `AndroidManifest.xml` καθενός package, για να λάβουν έτσι το ίδιο ID χρήστη. Με τον τρόπο αυτό, για λόγους ασφαλείας τα δύο πακέτα αντιμετωπίζονται στη συνέχεια σαν να είναι η ίδια εφαρμογή, με το ίδιο ID χρήστη και τα ίδια δικαιώματα αρχείου. Αξίζει να σημειωθεί ότι για να διατηρηθεί η ασφάλεια, μόνο δύο εφαρμογές που υπογράφηκαν με το ίδιο πιστοποιητικό (και ζήτησαν το ίδιο `sharedUserId`) θα μπορέσουν να πάρουν το ίδιο ID χρήστη.

Στα δεδομένα που αποθηκεύονται από μία εφαρμογή, απονέμεται το ID χρήστη της εφαρμογής και δεν είναι προσβάσιμα από άλλα packages. Κατά τη δημιουργία ενός νέου αρχείου με τις `getSharedPreferences(String, int)`, `openFileOutput(String, int)`, ή την `openOrCreateDatabase(String, int, SQLiteDatabase.CursorFactory)`, μπορούν να χρησιμοποιηθούν τα flags `MODE_WORLD_READABLE` και/ή `MODE_WORLD_WRITEABLE` για να επιτρέψουν σε οποιοδήποτε package να διαβάσει/γράψει στο αρχείο. Όταν τίθενται αυτά τα flags, το αρχείο ανήκει ακόμα στην εφαρμογή αλλά τα δικαιώματα ανάγνωσης/εγγραφής έχουν τεθεί κατάλληλα έτσι ώστε οποιαδήποτε εφαρμογή θέλει μπορεί να έχει πρόσβαση σ' αυτό.

5.4. Χρησιμοποιώντας τις Άδειες

Μία βασική εφαρμογή Android δεν έχει άδειες-δικαιώματα που να συνδέονται μ' αυτή, που σημαίνει ότι δεν μπορεί να κάνει τίποτα που θα επηρεάσει αρνητικά την εμπειρία του χρήστη ή οποιαδήποτε δεδομένα στη συσκευή. Για να γίνει χρήση των προστατευόμενων λειτουργιών της συσκευής, θα πρέπει να συμπεριληφθούν στο αρχείο `AndroidManifest.xml` μία ή περισσότερες ετικέτες `<uses-permission>` δηλώνοντας τα δικαιώματα που χρειάζεται η εφαρμογή.

Για παράδειγμα, μία εφαρμογή που χρειάζεται να παρακολουθεί τα εισερχόμενα μηνύματα SMS θα ορίζει τα εξής:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.app.myapp" >

    <uses-permission android:name="android.permission.RECEIVE_SMS" />

</manifest>
```

Κατά την εγκατάσταση της εφαρμογής, οι άδειες που ζητήθηκαν απ' αυτή παραχωρήθηκαν από το πρόγραμμα εγκατάστασης πακέτων, με βάση τον έλεγχο του πιστοποιητικού της εφαρμογής και/ή προτρέποντας το χρήστη να αποφασίσει. Από τη στιγμή που τρέχει η εφαρμογή δεν γίνονται έλεγχοι αλληλεπίδρασης με το χρήστη και έτσι είτε χορηγήθηκε μια συγκεκριμένη άδεια όταν εγκαταστάθηκε και μπορεί να χρησιμοποιηθεί η αντίστοιχη λειτουργία, είτε η άδεια απορρίφθηκε και οποιαδήποτε προσπάθεια να χρησιμοποιηθεί εκείνη η λειτουργία θα αποτύχει αφού δεν εγκρίθηκε από το χρήστη.

Πολλές φορές μία αποτυχία άδειας καταλήγει σε *εξαίρεση ασφαλείας* (`SecurityException`) που επιστρέφεται στην εφαρμογή. Ωστόσο, δεν υπάρχει εγγύηση ότι θα συμβεί παντού. Για παράδειγμα, η μέθοδος `sendBroadcast(Intent)` ελέγχει τα δικαιώματα όσο τα δεδομένα παραλαμβάνονται από κάθε παραλήπτη, αφότου η κλήση της μεθόδου έχει επιστρέψει, έτσι ώστε να μην ληφθεί *εξαίρεση* εάν υπάρχουν αποτυχίες άδειας. Όμως, σε όλες σχεδόν τις περιπτώσεις, μία αποτυχία άδειας θα τυπωθεί στο αρχείο καταγραφής του συστήματος.

Οι άδειες που παρέχονται από το σύστημα του Android μπορούν να βρεθούν στο `Manifest.permission`. Κάθε εφαρμογή μπορεί επίσης να καθορίσει και να επιβάλλει τα δικά της δικαιώματα, έτσι ώστε να μην ζητείται μία πλήρης λίστα με όλες τις πιθανές άδειες.

Μία συγκεκριμένη άδεια μπορεί να επιβάλλεται σε διάφορα τμήματα της εφαρμογής κατά τη διάρκεια εκτέλεσής της, όπως:

- Κατά τη διάρκεια μιας κλήσης στο σύστημα, για να αποφευχθεί μία εφαρμογή από το να εκτελέσει κάποιες λειτουργίες.
- Κατά την εκκίνηση ενός *Activity*, για εμποδιστούν οι εφαρμογές από το να εκτελέσουν *Activities* άλλων εφαρμογών.
- Τόσο κατά την αποστολή όσο και κατά την λήψη *Broadcasts*, για να ελεγχθεί ποιος μπορεί να στείλει ή να λάβει *Broadcasts* σε/από ποιόν.
- Κατά την πρόσβαση και τον χειρισμό ενός *Content Provider*.
- Συνδέοντας (*Binding*) ή αρχίζοντας ένα *Service*.

5.5. Δήλωση και Επιβολή των Αδειών

Για να επιβάλλετε τις δικές σας άδειες, πρέπει πρώτα να τις δηλώσετε στο αρχείο `AndroidManifest.xml` χρησιμοποιώντας μία ή περισσότερες ετικέτες `<permission>`.

Για παράδειγμα, μία εφαρμογή που θέλει να ελέγχει ποιός μπορεί να εκκινήσει κάποιο από τα *Activities* της, θα μπορούσε να δηλώσει μία άδεια για αυτή την λειτουργία, όπως την ακόλουθη:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.me.app.myapp" >

    <permission
android:name="com.me.app.myapp.permission.DEADLY_ACTIVITY"
        android:label="@string/permlab_deadlyActivity"
        android:description="@string/permdesc_deadlyActivity"
        android:permissionGroup="android.permission-group.COST_MONEY"
        android:protectionLevel="dangerous" />
</manifest>
```

Το χαρακτηριστικό `<protectionLevel>` είναι απαραίτητο, και λέει στο σύστημα πώς ο χρήστης πρέπει να ενημερώνεται για τις εφαρμογές που χρειάζονται την άδεια, ή ποιός επιτρέπεται να έχει την άδεια.

Το χαρακτηριστικό `<permissionGroup>` είναι προαιρετικό, και χρησιμοποιείται μόνο για να βοηθήσει το σύστημα να απεικονίσει τις άδειες στο χρήστη. Συχνά τίθεται είτε ως ένα τυπικό group του συστήματος (βλέπε `android.Manifest.permission_group`) είτε σε πιο σπάνιες περιπτώσεις ως ένα group καθορισμένο από τον προγραμματιστή της εφαρμογής. Είναι προτιμότερο να χρησιμοποιείται ένα υπάρχον group επειδή απλοποιεί το UI (*User Interface*) που φαίνεται στον χρήστη.

Να σημειωθεί ότι η ετικέτα και η περιγραφή για την άδεια θα πρέπει να παρέχονται. Αυτά είναι strings που μπορούν να εμφανίζονται στο χρήστη όταν βλέπει μία λίστα από δικαιώματα (`android:label`) ή πληροφορίες μίας μόνο άδειας (`android:description`). Η ετικέτα θα πρέπει να είναι σύντομη· λίγες λέξεις που περιγράφουν το βασικό κομμάτι της λειτουργίας που προστατεύει η άδεια. Η περιγραφή θα πρέπει να είναι μία-δύο προτάσεις που περιγράφουν τι θα επιτρέψει η άδεια να κάνει αυτός που την ζητάει. Κατά σύμβαση, η περιγραφή είναι δύο προτάσεις, η πρώτη περιγράφει την άδεια και η δεύτερη προειδοποιεί τον χρήστη τι «στραβά» πράγματα μπορούν να συμβούν εάν χορηγηθεί το δικαίωμα σε μία εφαρμογή.

Ένα παράδειγμα της ετικέτας και της περιγραφής για την άδεια CALL_PHONE:

```
<string name="permlab_callPhone">directly call phone numbers</string>
    <string name="permdesc_callPhone">Allows the application to call
                    phone numbers without your intervention. Malicious
                    applications may cause unexpected calls on your
                    phone bill. Note that this does not allow the
                    application to call emergency numbers.</string>
```

Μπορείτε να δείτε τα δικαιώματα που είναι ορισμένα στο σύστημα με την εντολή `adb shell pm list permissions`. Ειδικότερα, η επιλογή "`-s`" εμφανίζει τις άδειες σε μορφή παρόμοια με εκείνη που τις βλέπει ο χρήστης:

```
$ adb shell pm list permissions -s
All Permissions:

Network communication: view Wi-Fi state, create Bluetooth
connections, full Internet access, view network state

Your location: access extra location provider commands, fine (GPS)
location, mock location sources for testing, coarse (network-based)
location

Services that cost you money: send SMS messages, directly call phone
numbers

...
```

5.5.1. Επιβολή Αδειών στο `AndroidManifest.xml`

Υψηλού επιπέδου άδειες που να περιορίζουν την πρόσβαση σε ολόκληρα τμήματα του συστήματος ή σε εφαρμογές, μπορούν να εφαρμοστούν μέσω του `AndroidManifest.xml`. Το μόνο που απαιτείται είναι να συμπεριληφθεί ένα χαρακτηριστικό `android:permission` στο επιθυμητό τμήμα του συστήματος και να καθοριστεί η άδεια που θα χρησιμοποιηθεί για να ελέγχει την πρόσβαση σ' αυτό.

Άδειες τύπου `Activity` (εφαρμόζονται με την ετικέτα `<activity>`) περιορίζουν ποιός μπορεί να αρχίσει το σχετικό *Activity*. Η άδεια ελέγχεται κατά την κλήση των `Context.startActivity()` και `Activity.startActivityForResult()`. Εάν αυτός που τις καλεί δεν έχει την απαιτούμενη άδεια τότε επιστρέφεται μία `SecurityException`.

Άδειες τύπου `Service` (εφαρμόζονται με την ετικέτα `<service>`) περιορίζουν ποιός μπορεί να αρχίσει ή να συνδεθεί στο σχετικό *Service*. Η άδεια ελέγχεται κατά την κλήση των `Context.startService()`, `Context.stopService()` και `Context.bindService()`. Εάν αυτός που τις καλεί δεν έχει την απαιτούμενη άδεια τότε επιστρέφεται μία `SecurityException`.

Άδειες τύπου `BroadcastReceiver` (εφαρμόζονται με την ετικέτα `<receiver>`) περιορίζουν ποιός μπορεί να στείλει *broadcasts* στο σχετικό *receiver*. Η άδεια ελέγχεται μετά την επιστροφή της `Context.sendBroadcast()`, καθώς το σύστημα προσπαθεί να παραδώσει το *broadcast* στο σχετικό *receiver*. Ως αποτέλεσμα, μία αποτυχία άδειας δεν θα επιστρέψει μία *εξαίρεση* (*exception*) σ' αυτόν που την κάλεσε, απλά δεν θα παραδώσει το *intent*. Με τον ίδιο τρόπο, μία άδεια μπορεί να τροφοδοτήσει την `Context.registerReceiver()` για να ελέγχεται ποιός μπορεί να στείλει *broadcasts* σ' έναν *καταχωρημένο* (*registered*) *receiver*. Ένας άλλος τρόπος, είναι μία άδεια να τροφοδοτήσει την `Context.sendBroadcast()` για να περιορίζει ποιά *αντικείμενα BroadcastReceiver* επιτρέπεται να λαμβάνουν το *broadcast* (δείτε παρακάτω).

Άδειες τύπου `ContentProvider` (εφαρμόζονται με την ετικέτα `<provider>`) περιορίζουν ποιός μπορεί να έχει πρόσβαση στα δεδομένα ενός `ContentProvider`. (Οι *Content Providers* έχουν μία πρόσθετη δυνατότητα ασφαλείας, τις *άδειες URI*, που περιγράφονται αναλυτικότερα παρακάτω.) Σε αντίθεση με τα προηγούμενα, εδώ υπάρχουν δύο ξεχωριστά χαρακτηριστικά άδειας που μπορούν να τεθούν: Το `android:readPermission` το οποίο περιορίζει ποιός μπορεί να διαβάσει από τον *provider* και το `android:writePermission` το οποίο περιορίζει ποιός μπορεί να γράψει σ' αυτόν. Να σημειωθεί ότι εάν ένας *provider* προστατεύεται τόσο με το δικαίωμα *ανάγνωσης* όσο και με της *εγγραφής*, το να θέσουμε μόνο την *άδεια εγγραφής* δεν σημαίνει ότι θα μπορούμε να διαβάσουμε κιόλας απ' αυτόν. Οι άδειες ελέγχονται όταν ανακτάται για πρώτη φορά ένας *provider* (εάν δεν έχουμε άδεια, τότε επιστρέφεται μία *SecurityException*) και όσο κάνουμε λειτουργίες στον *provider*. Χρησιμοποιώντας την `ContentResolver.query()` απαιτείται να έχουμε θέσει το *δικαίωμα ανάγνωσης* ενώ για τις `ContentResolver.insert()`, `ContentResolver.update()` και `ContentResolver.delete()` το *δικαίωμα εγγραφής*. Σε όλες τις περιπτώσεις, αν δεν έχει τεθεί η απαιτούμενη άδεια, επιστρέφεται μία `SecurityException`.

5.5.2. Επιβολή Αδειών κατά την αποστολή Broadcasts

Εκτός από την άδεια που τροφοδοτεί ποιός μπορεί να στείλει *Intends* σ' ένα *καταχωρημένο* `BroadcastReceiver` (όπως περιγράφηκε παραπάνω), μπορούμε επίσης να καθορίσουμε μία απαιτούμενη άδεια όταν στέλνουμε ένα *broadcast*. Καλώντας την `Context.sendBroadcast()` μ' ένα string άδειας, απαιτούμε από την εφαρμογή του *receiver* να έχει την κατάλληλη άδεια για να λάβει το *broadcast* μας.

Να σημειωθεί ότι και ο *receiver* και ο *broadcaster* μπορούν να απαιτήσουν μία άδεια. Όταν αυτό συμβαίνει, και οι δύο έλεγχοι άδειας πρέπει να είναι επιτυχείς για να παραδοθεί το *Intend* στον αντίστοιχο *receiver*.

5.5.3. Επιβολή άλλων Αδειών

Άλλες άδειες μπορούν να επιβληθούν σε οποιαδήποτε κλήση σε ένα *service*. Αυτό επιτυγχάνεται με τη μέθοδο `Context.checkCallingPermission()`. Καλώντας την με ένα επιθυμητό string άδειας, αυτή θα επιστρέψει έναν ακέραιο που δηλώνει εάν χορηγήθηκε άδεια για την τρέχουσα κλήση. Σημειώστε ότι αυτή μπορεί να χρησιμοποιηθεί μόνο όταν εκτελούμε μία κλήση προερχόμενη από μία άλλη διαδικασία, συνήθως μέσω μιας *διεπαφής IDL* (*Interface Definition Language*) δημοσιευμένης από ένα *service* ή με κάποιον άλλο τρόπο δοθέντα σε μία άλλη διαδικασία.

Υπάρχουν πολλοί άλλοι χρήσιμοι τρόποι για τον έλεγχο των αδειών. Αν έχουμε το *pid* (*process id*) μιας άλλης διαδικασίας, μπορούμε να χρησιμοποιήσουμε τη μέθοδο `Context.checkPermission(String, int, int)` για να ελέγξουμε την άδεια για εκείνο το *pid*. Εάν έχουμε το όνομα του *package* μιας άλλης εφαρμογής, μπορούμε να χρησιμοποιήσουμε τη μέθοδο `PackageManager.checkPermission(String, String)` για να διαπιστώσουμε αν στο συγκεκριμένο πακέτο έχει χορηγηθεί μία ειδική άδεια.

5.6. Άδειες URI *(Uniform Resource Indicator)*

Το τυπικό σύστημα αδειών που περιγράφηκε ως τώρα, συχνά δεν επαρκεί όταν χρησιμοποιείται με *Content Providers*. Ένας *content provider* μπορεί να θέλει να προστατέψει τον εαυτό του με *δικαιώματα ανάγνωσης* και *εγγραφής*, ενώ οι άμεσοί του *clients* πρέπει επίσης να θέτουν ειδικά *URIs* γι 'αυτούς, για να λειτουργήσουν οι άλλες εφαρμογές. Ένα χαρακτηριστικό παράδειγμα είναι τα συνημμένα σε μία εφαρμογή ηλεκτρονικού ταχυδρομείου. Η πρόσβαση στο μήνυμα θα πρέπει να προστατεύεται από άδειες, δεδομένου ότι πρόκειται για ευαίσθητα προσωπικά δεδομένα. Ωστόσο, αν ένα URI για μία επισυναπτόμενη φωτογραφία δοθεί σ' ένα πρόγραμμα προβολής εικόνων, αυτό δεν θα έχει το δικαίωμα να την ανοίξει επειδή δεν υπάρχει λόγος να διαθέτει μία άδεια για πρόσβαση σε ολόκληρο το email.

Η λύση γι' αυτό το πρόβλημα είναι οι άδειες *ανά-URI*: Όταν αρχίζει ένα *activity* ή επιστρέφεται ένα αποτέλεσμα σ' ένα *activity*, αυτός που καλεί μπορεί να ορίσει το `Intent.FLAG_GRANT_READ_URI_PERMISSION` και/ή το `Intent.FLAG_GRANT_WRITE_URI_PERMISSION`. Αυτό παραχωρεί στο λαμβάνον *activity* την άδεια για πρόσβαση στα συγκεκριμένα δεδομένα που το *URI* δείχνει στο *Intent*, ανεξάρτητα από το εάν έχει κάποια άδεια πρόσβασης στα δεδομένα του *content provider* που αντιστοιχεί στο *Intend*.

Αυτός ο μηχανισμός επιτρέπει σε ένα κοινό μοντέλο όπου η αλληλεπίδραση του χρήστη (άνοιγμα ενός συννημένου, επιλογή μιας επαφής από την λίστα, κλπ.) οδηγεί στην επί τούτου χορήγηση αδειών. Αυτό μπορεί να αποτελέσει μία βασική

διευκόλυνση για τη μείωση των αδειών που χρειάζονται οι εφαρμογές σε εκείνες που συνδέονται μόνο άμεσα με την συμπεριφορά τους.

Η χορήγηση αδειών *URI*, ωστόσο, απαιτεί κάποια συνεργασία με τον *content provider* για να ελέγχει αυτά τα *URIs*. Συνιστάται έντονα, οι *content providers* να εφαρμόζουν αυτό το μηχανισμό και να δηλώνουν ότι τον υποστηρίζουν μέσω του χαρακτηριστικού `android:grantUriPermissions` ή της ετικέτας `<grant-uri-permissions>`.

Βιβλιογραφία

1. Android Developers [http://developer.android.com]
2. Open Handset Alliance [http://www.openhandsetalliance.com]
3. Andbook [http://andbook.anddev.org]