Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**AndroPatchApp:** : **Αντιμετώπιση των κακόβουλων διαφημίσεων**
Ονοματεπώνυμο Φοιτητή	**Τσιάκος Βασίλειος**
Πατρώνυμο	**Σωτήριος**
Αριθμός Μητρώου	**ΜΠΠΛ/ 13115**
Επιβλέπων	**Κ. Πατσάκης , Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Ιούλιος 2016**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Κ.Πατσάκης Επίκουρος Καθηγητής	Γ. Τσιχριντζής Καθηγητής	Ε. Αλέπης Επίκουρος Καθηγητής

Πίνακας περιεχομένων

Πίνακας εικόνων

Abstract

The proliferation of smart phones has created a new market in the field of portable devices, such as advertisements. By extension, the revenues arising therefrom are an important incentive for any form of attack. Despite the protection afforded by the Android operating system, many malicious exploit operating system vulnerabilities to gain access to personal user data. To address these problems have been proposed several solutions based on either software or hardware with the corresponding advantages and disadvantages. The AdnroPatchApp is an effective solution in malicious webView methods used by adnetworks for sensitive information such as user coordinates, camera etc. The conclusions of the thesis presented at the conference International Conference on Mobile, Secure and Programmable Networking (MPSN '2016) in Paris. The code of the application is under MIT license and can be downloaded from : *https://github.com/excecutor/AndroPatchApp*.

Περίληψη

Η διάδοση των έξυπνων τηλεφώνων δημιούργησε μια νέα αγορά στο χώρο των φορητών συσκευών , αυτή των διαφημίσεων. Κατ' επέκταση, τα έσοδα που απορρέουν από αυτές αποτελούν ένα σημαντικό κίνητρο για κάθε μορφή επίθεσης. Παρά την προστασία που παρέχει το λειτουργικό σύστημα Android, πολλοί κακόβουλοι εκμεταλλεύονται ευπάθειες του λειτουργικού συστήματος, ώστε να αποκτήσουν πρόσβαση σε προσωπικά δεδομένα του χρήστη. Για την αντιμετώπιση αυτών των προβλημάτων προτάθηκαν αρκετές λύσεις που βασίζονται είτε στο λογισμικό ή στο υλικό με τα αντίστοιχα πλεονεκτήματα και μειονεκτήματά τους. Το AdnroPatchApp αποτελεί μια αποτελεσματική λύση στην κακόβουλη χρήση μεθόδων *webView* από τα δίκτυα διαφημίσεων με στόχο τις ευαίσθητες πληροφορίες του χρήστης όπως συντεταγμένες, κάμερα κ.α. Τα συμπεράσματα της διπλωματικής εργασίας παρουσιάστηκαν στο συνέδριο *International Conference on Mobile, Secure and Programmable Networking (MPSN' 2016)* που διεξήχθη στο Παρίσι. Ο κώδικας της εφαρμογής υπόκειται στη άδεια του MIT και είναι διαθέσιμος *: https://github.com/excecutor/AndroPatchApp.*

Εισαγωγή

Η διάθεση εφαρμογών για κινητές συσκευές αποτελεί σημαντική επιτυχία στο χώρο των smartphones, εφόσον επιτρέπει την ανάπτυξη και πώλησή τους σε χρήστες. Το App Store της εταιρείας Apple και το Google Play της Google αποτελούν τις 2 βασικές πλατφόρμες όπου οι developers είτε πωλούν είτε διανέμουν δωρεάν τις εφαρμογές τους. Οι δυο εταιρείες έχουν επιτύχει μια όσο το δυνατό "δίκαιη" κατανομή των εσόδων από τις εφαρμογές για κινητές συσκευές. Συγκεκριμένα το 73% των εφαρμογών στο Google Play διατίθεται δωρεάν, γεγονός το οποίο αυξάνει τον αριθμό χρηστών που τις χρησιμοποιούν. Το μοντέλο εσόδων που υιοθετήθηκε από τις δωρεάν εφαρμογές περιλαμβάνει ωστόσο, και διαφημίσεις που είναι ενσωματωμένες στην εφαρμογή και εμφανίζονται κατά τη διάρκεια της χρήσης τους. Συνεπώς , η δωρεάν διάθεση εξυπηρετεί όχι μόνο τον χρήστη αλλά και τους φορείς του διαφημιστικού περιβάλλοντος (brands,ad networks, ad agencies , publisher).

Ωστόσο , το περιβάλλον των διαφημίσεων εγείρει προβληματισμό σχετικά με την ασφάλεια των χρηστών, καθώς οι developers προσπαθούν να αυξήσουν τα έσοδα τους, ενώ παράλληλα οι διαφημιστικές εταιρείες πραγματοποιούν στοχευμένες καμπάνιες σε κάθε χρήστη ξεχωριστά. Κατά συνέπεια, εκθέτουν σε τρίτους προσωπικά στοιχεία, όπως η γεωγραφική τοποθεσία ή οι επαφές του χρήστη. Παρά την πολιτική ασφαλείας που παρέχει σε διάφορα επίπεδα το λειτουργικό σύστημα Android, έχει διαπιστωθεί, ότι ο τρόπος με τον οποίο υλοποιούνται οι διαφημίσεις και το δίκτυο διανομής τους στις εφαρμογές καθιστούν τον χρήστη ευάλωτο σε επιθέσεις από τρίτους ενώ παράλληλα διευκολύνουν τη διαρροή προσωπικών δεδομένων.

Το περιβάλλον των διαφημίσεων περιλαμβάνει πολλούς εμπλεκόμενους : οι εταιρείες (brands) θέλουν να προσελκύσουν πελάτες, οι διαφημιστικοί πράκτορες (ad agencies) που σχεδιάζουν διαφημιστικές καμπάνιες για τις εταιρείες, τα δίκτυα διαφημίσεων (ad networks) που διανέμουν το περιεχόμενο των διαφημίσεων, οι εκδότες (publishers) που δημιουργούν τις εφαρμογές για τις κινητές συσκευές και οι χρήστες στους οποίους προβάλλονται οι διαφημίσεις. Ένας επιπλέον παράγοντας είναι οι υπηρεσίες Mediation, οι οποίες υποστηρίζουν πολλά δίκτυα διαφημίσεων , επιτρέποντας στους εκδότες να συνδυάζουν διαφορετικά δίκτυα. Το βασικό πλεονέκτημα αυτής της δυνατότητας είναι ότι μπορεί να αυξήσει τα έσοδα του εκδότη καθώς εάν αποτύχει κάποιο ad network να επιστρέψει μια διαφήμιση τότε φορτώνεται από άλλο δίκτυο.

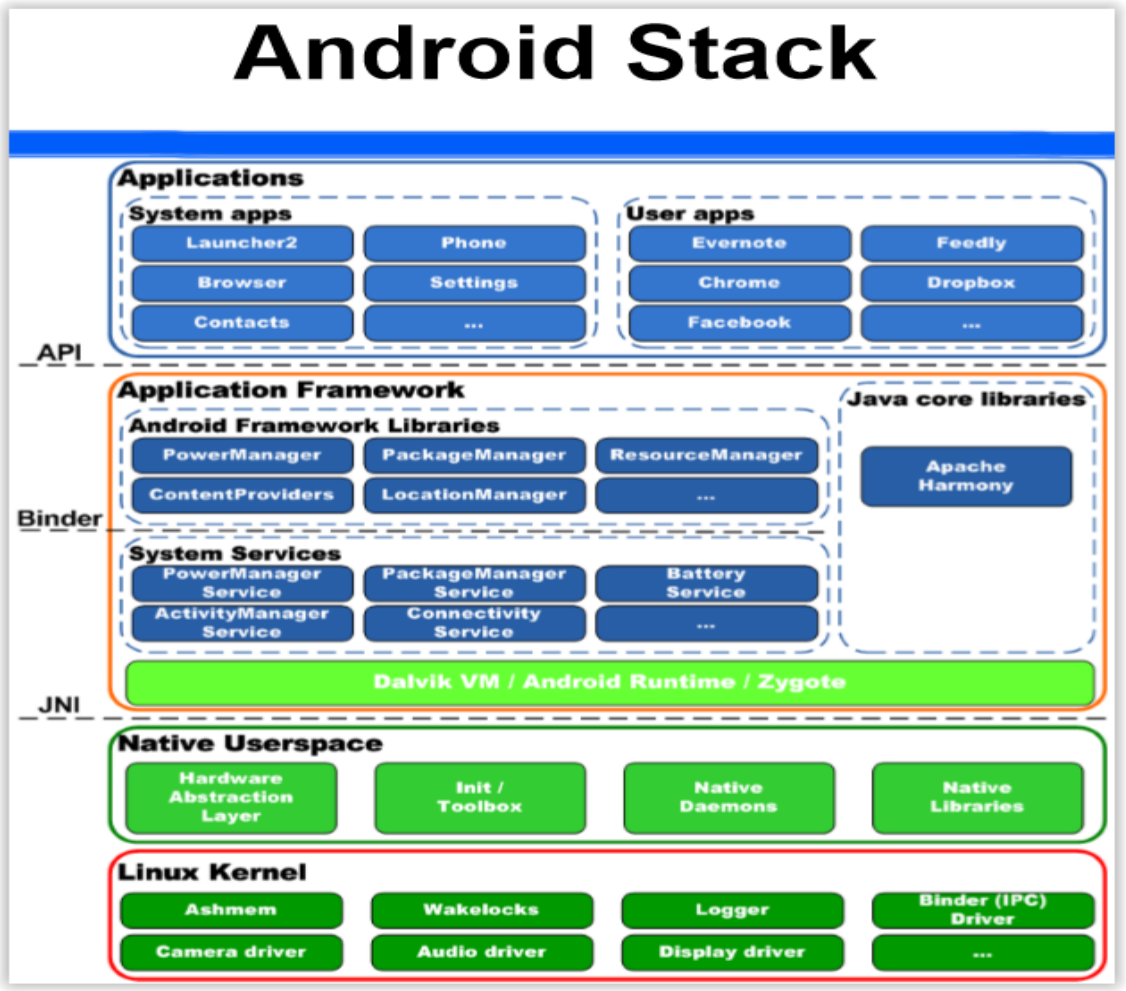
Το μοντέλο διαφημίσεων βασίζεται σε βιβλιοθήκες JAVA, οι οποίες παρέχονται από τους διαφημιστικούς πράκτορες και ενσωματώνονται στον κώδικα της εφαρμογής. Έτσι, υπάρχουν 3 κύριες διεπαφές για τις ad libraries: με την υπόλοιπη εφαρμογή, με τους πόρους του internet (επικοινωνία με το δίκτυο διαφημίσεων) και με το λειτουργικό σύστημα Android. Το λειτουργικό σύστημα Android περιορίζει τα δικαιώματα της εφαρμογής μέσω του συστήματος αδειοδότησης (permission system). Τα δικαιώματα που απαιτεί μια εφαρμογή δηλώνονται στο αρχείο manifest το οποίο έχει ενσωματωθεί στην εφαρμογή. Επειδή τα δικαιώματα που απαιτούνται είναι σε επίπεδο εφαρμογής, οι διαφημιστικές βιβλιοθήκες που χρησιμοποιούνται μοιράζονται τα ίδια δικαιώματα με την εφαρμογή που τη φιλοξενεί. Αυτό έχει ως αποτέλεσμα ο χρήστης να μη γνωρίζει εάν τα δικαιώματα για τα οποία συμφωνεί να κάνει χρήση η εφαρμογή τα χρησιμοποιούν και οι ad libraries. Μετά τον έλεγχο 114,000 δωρεάν [56] εφαρμογών από το Google Store διαπίστώθηκε ότι οι διαφημιστικές βιβλιοθήκες προσπαθούν να εκμεταλλευτούν τα δικαιώματα που έχει ζητήσει η host εφαρμογή. Ιδιαίτερα ανησυχητική είναι η αύξηση των διάφορων επικίνδυνων αδειών (dangerous permission) που θέτουν σε κίνδυνο τα ιδιωτικά δεδομένα.

1 Η πλατφόρμα Android

Το Android είναι ένα λειτουργικό σύστημα , που αναπτύσσεται από την Google, βασίζεται στο πυρήνα του Linux και σχεδιάστηκε αρχικά για φορητές συσκευές. Όπως όλα τα λειτουργικά συστήματα, το Android επιτρέπει στις εφαρμογές να κάνουν χρήση του υλικού που διαθέτουν μέσω του abstraction layer και να παρέχουν ένα καθορισμένο περιβάλλον για τις εφαρμογές αυτές.

Το λειτουργικό σύστημα Android έχει γραφτεί σε C και τρέχει σε εικονικό μηχάνημα VM. Για το λόγο αυτό το Android διαθέτει την εικονική μηχανή Dalvik VM που εκτελεί το δικό της byte code. Το Dalvik είναι βασικό στοιχείο το πυρήνα, όπως όλες οι εφαρμογές του χρήστη και τα πλαίσια εργασίας των εφαρμογών είναι γραμμένα σε Java και εκτελούνται από το Dalvik.

Η αρχιτεκτονική δομή του λειτουργικού συστήματος Android μπορεί να διαιρεθεί σε 5 στρώματα (layers): Πυρήνα και τα εργαλεία χαμηλού επιπέδου, φυσικές βιβλιοθήκες (native libraries), Android Runtime , framework layer και τις εφαρμογές, οι οποίες αποτελούν το τελευταίο στρώμα του λειτουργικού συστήματος.



Εικόνα 1: Αρχιτεκτονική δομή λειτουργικού συστήματος Android[1]

1.1 Πυρήνας Linux

Ο πυρήνας του Linux αποτελεί τη καρδιά της αρχιτεκτονικής Android. Είναι υπεύθυνος για τους οδηγούς συσκευών, τη διαχείριση των διεργασιών, της μνήμης και των συσκευών, αλλά και για την πρόσβαση στους πόρους του συστήματος. Ο πυρήνας του Linux που χρησιμοποιείται στο Android παρέχει επιπλέον λειτουργίες όπως τη διαχείριση της ενέργειας, τους Binder IPC οι οποίοι αποτελούν έναν μηχανισμό για την επικοινωνία των εσωτερικών διεργασιών, το υποσύστημα ashmem αποτελεί έναν νέο allocator διαμοιραζόμενης μνήμης.

1.2 Native Userspace

Το native userspace αναφέρεται σε όλο τον κώδικα που εκτελείται έξω από τον πυρήνα του λειτουργικού συστήματος, όπως είναι βιβλιοθήκες ή προγράμματα, τα οποία αλληλεπιδρούν με τον πυρήνα. Οι φυσικές(native) βιβλιοθήκες μεταφέρουν σύνολο εντολών, ώστε να οδηγούν τις συσκευές και να διαχειρίζονται διαφορετικούς τύπους δεδομένων.

1.3 Χρόνος λειτουργίας (Android runtime)

Το επόμενο στρώμα περιλαμβάνει τις βιβλιοθήκες του πυρήνα (Java core libraries), οι οποίες παρέχουν την πρόσβαση σε αρχεία, δικτυακή πρόσβαση, δομές δεδομένων, γραφικά κ.α. Η εικονική μηχανή Dalvik είναι υπεύθυνη για την εκτέλεση των εφαρμογών, παρέχει φορητότητα (portability) στις εφαρμογές, υποστηρίζει πολλαπλά στιγμιότυπα (instances) κ.α. Η εικονική μηχανή Dalvik μοιάζει με την εικονική μηχανή της Java με τη διαφορά ότι έχει σχεδιαστεί για κινητές συσκευές. Καταναλώνει λίγη μνήμη και παρέχει γρήγορες επιδόσεις.

1.4 Πλαίσιο εργασίας (framework) Android

Στη κορυφή των φυσικών βιβλιοθηκών και του Android χρόνου εκτέλεσης βρίσκεται το πλαίσιο εργασίας(framework) του Android. Το πλαίσιο εργασίας του Android περιλαμβάνει τις υπηρεσίες συστήματος και τις βιβλιοθήκες του πλαισίου.

Οι υπηρεσίες συστήματος παρέχουν δυνατότητες πρόσβασης στο υλικό της συσκευής όπως την υπηρεσία τηλεφωνίας (telephony service) , την υπηρεσία τοποθεσίας (location service) κ.α. Επιπλέον παρέχουν υπηρεσίες της πλατφόρμας Android, όπως η ActivityManager Service, η PackageManager service, η PowerManager Service κ.α

Οι βιβλιοθήκες πλαισίου παρέχουν τις διεπαφές προγραμματισμού εφαρμογών (APIs). Συγκεκριμένα, υποστηρίζουν ένα πλήθος από κλάσεις και διεπαφές για προγραμματιστές των Android εφαρμογών, όπως την Activity Manager, Package Manager, Power Manager, Location Manager, Sensor Manager, Telephony Manager.κ.α.

1.5 Εφαρμογές

Το τελευταίο στρώμα της αρχιτεκτονικής του Android είναι οι εφαρμογές. Οι εφαρμογές χωρίζονται σε αυτές του συστήματος, όπως Email application, SMS application, Contacts, Phone, Browser και άλλες, οι οποίες παρέχονται από την πλατφόρμα και στις εφαρμογές τρίτων μελών (third-party), οι οποίες δημιουργούνται από ανεξάρτητους προγραμματιστές. πχ. Facebook, Evernote,Chrome. [1]

2 Μοντέλο ασφάλειας του Android

Το μοντέλο ασφάλειας του Android αξιοποιεί τα χαρακτηριστικά που του παρέχονται από το πυρήνα του Linux. Το Linux είναι ένα λειτουργικό σύστημα για πολλαπλούς χρήστες (multiuser) , ενώ ο πυρήνα μπορεί να απομονώσει τον ένα χρηστή από τον άλλο ,όπως απομονώνει τις διεργασίες. Σε ένα σύστημα Linux ένας χρήστης δε μπορεί να έχει πρόσβαση στα αρχεία ενός άλλου χρήστη, εκτός και αν έχει μεγαλύτερα δικαιώματα. Κάθε διεργασία εκτελείται με τα συγκεκριμένα δικαιώματα (user id (UID) και group id (GID)) του χρήστη που την εκκίνησε.

Το Android λαμβάνει τα πλεονεκτήματα από την απομόνωση του χρήστη. Ωστόσο, διαχειρίζεται τους χρήστες με διαφορετικό τρόπο από ένα παραδοσιακό σύστημα Linux. Σε ένα παραδοσιακό σύστημα, ένα UID καταχωρείται είτε σε ένα φυσικό χρήστη ο οποίος μπορεί να συνδεθεί στο σύστημα και να εκτελέσει εντολές μέσω του κελύφους, ή σε μια υπηρεσία

συστήματος (daemon) που εκτελείται στο παρασκήνιο (background). Αρχικά, το Android σχεδιάστηκε για έξυπνες συσκευές. Επειδή, όμως τα κινητά τηλέφωνα αποτελούν προσωπική συσκευή, δεν υπήρχε η ανάγκη να καταχωρηθούν διαφορετικοί χρήστες. Τα UID χρησιμοποιούνται για τη διάκριση των διεργασιών. Αυτό αποτελεί τη βάση για την εφαρμογή sandboxing.

2.1 Εφαρμογή sandboxing

Το Android αυτόματα καταχωρεί ένα μοναδικό UID - συχνά καλείτε *app ID* - σε κάθε εφαρμογή κατά την εγκατάσταση και εκτελεί την εφαρμογή σε μια μοναδική διεργασία που τρέχει ως UID. Επιπλέον σε κάθε εφαρμογή δίνεται ένας μοναδικός κατάλογος για την αποθήκευση των δεδομένων, στα οποία έχει δικαιώματα εγγραφής/ανάγνωσης. Έτσι οι εφαρμογές είναι απομονωμένης ή sandboxed σε επίπεδο διεργασιών και σε επίπεδο αρχείων. Αυτό δημιουργεί μια εφαρμογή sandboxing σε επίπεδο πυρήνα η οποία εφαρμόζεται σε όλες τις εφαρμογές, ανεξάρτητα με το που έχουν εκτελεστεί, σε φυσική ή εικονική διεργασία.

Οι δαίμονες του συστήματος και οι εφαρμογές εκτελούνται υπό καθορισμένα και σταθερά UIDs, ενώ πολύ λίγοι δαίμονες εκτελούνται σαν χρήστης root. Στο Android τα UIDs του συστήματος ορίζονται στατικά στο αρχείο header *android_filesystem_config.h*

Ο κατάλογος των δεδομένων μιας εφαρμογής ηλεκτρονικού ταχυδρομείου παίρνει το όνομα του πακέτου της εφαρμογής και δημιουργείται κάτω από τον κατάλογο */data/data* σε συσκευές ενός χρήστη (οι συσκευές πολλαπλών χρηστών ακολουθούν διαφορετικό σχήμα ονοματοδοσίας). Όλα τα αρχεία που βρίσκονται στον κατάλογο δεδομένων ανήκουν σε ένα μόνο Linux χρήστη. Οι εφαρμογές μπορούν να δημιουργήσουν αρχεία χρησιμοποιώντας τις σημαίες *MODE_WORLD_READABLE* και *MODE_WORLD_WRITABLE*, οι οποίες επιτρέπουν την πρόσβαση στα αρχεία από άλλες εφαρμογές. Ωστόσο, η απευθείας κοινή χρήση των αρχείων δεν αποτελεί καλή πρακτική και αυτές οι σημαίες θεωρούνται ξεπερασμένες από την έκδοση 4.2

Τα UIDs της εφαρμογής χρησιμοποιούνται μαζί με άλλα πακέτα στο αρχείο /data/system/packages.xml και επιπλέον γράφονται στο αρχείο */data/system/packages.list*. Στην παρακάτω εικόνα φαίνεται η καταχώρηση του UID στο πακέτο com.google.android.email όπως εμφανίζεται στο αρχείο packages.list.

```
# grep 'com.google.android.email' /data/system/packages.list
com.google.android.email 10037 0 /data/data/com.google.android.email default 3003,1028,1015
```

Εικόνα 2: Αντιστοίχιση UID σε εφαρμογή, αρχείο /data/packages.list[2]

Στη παραπάνω εικόνα το πρώτο πεδίο είναι το όνομα το πακέτου, το δεύτερο πεδίο είναι το UID που καταχωρήθηκε στην εφαρμογή, το τρίτο πεδίο είναι η σημαία για την αποσφαλμάτωση (debug), το τέταρτο πεδίο είναι η διαδρομή (path) του καταλόγου των αρχείων και το πέμπτο πεδίο η *seinfo* ετικέτα που χρησιμοποιείται από το SELinux. Το τελευταίο πεδίο είναι μια λίστα από επιπρόσθετα GIDs, τα οποία εκκινεί η εφαρμογή. Κάθε GID συνεργάζεται με ένα δικαίωμα (permission) στο Android και η λίστα GID δημιουργείται με βάση τα δικαιώματα που παρέχονται στην εφαρμογή.

Οι εφαρμογές μπορούν να εγκατασταθούν με τη χρήση του ίδιου UID (*shared user UID*). Στην περίπτωση αυτή μπορούν να διαμοιραστούν αρχεία και ακόμα να τρέξουν μέσα στην ίδια διεργασία. Τα shared user IDs χρησιμοποιούνται ευρέως από τις εφαρμογές συστήματος, οι οποίες συχνά χρειάζονται να χρησιμοποιήσουν τους ίδιους πόρους για διαφορετικά πακέτα.

Παρότι η ευκολία που παρέχει το shared user ID δεν προτείνεται για εφαρμογές που δεν είναι του συστήματος, ωστόσο παρέχεται σε εφαρμογές τρίτων μελών. Προκειμένου να μοιραστούν το ίδιο UID, οι εφαρμογές πρέπει να υπογραφούν από το ίδιο κλειδί.

2.2 Δικαιώματα

Επειδή ακριβώς οι εφαρμογές του Android είναι sandboxed, μπορούν να έχουν πρόσβαση μόνο στα δικά τους αρχεία και σε πόρους του συστήματος που δεν απαιτείται άδεια. Μια τέτοια περιορισμένη εφαρμογή δεν θα ήταν πολύ ενδιαφέρουσα, αν και το Android θα μπορούσε να χορηγήσει επιπλέον δικαιώματα πρόσβασης στην εφαρμογή, προκειμένου να καταστεί πιο λειτουργική. Αυτές οι άδεις πρόσβασης ονομάζονται δικαιώματα (permissions) και μπορούν να ελέγχουν την πρόσβαση σε συσκευές υλικού, στη σύνδεση στο διαδίκτυο, σε δεδομένα, ή σε υπηρεσίες του λειτουργικού συστήματος.

Τα δικαιώματα που ζητούν οι εφαρμογές ορίζονται στο αρχείο *Manifest.xml*. Κατά την εγκατάσταση της εφαρμογής το Android ελέγχει τη λίστα με τα δικαιώματα που αιτήθηκε η εφαρμογή και αποφασίζει, εάν θα τα χορηγήσει ή όχι. Μόλις τα δικαιώματα αυτά χορηγηθούν δεν μπορούν να ανακληθούν και είναι διαθέσιμα στην εφαρμογή χωρίς καμία πρόσθετη επιβεβαίωση. Ωστόσο, για χαρακτηριστικά όπως το ιδιωτικό κλειδί ή η πρόσβαση στο λογαριασμό του χρήστη, απαιτείται επιπλέον επιβεβαίωση για κάθε πρόσβαση αντικειμένου, ακόμα και αν έχουν χορηγηθεί τα δικαιώματα στην εφαρμογή. Μερικά δικαιώματα μπορούν να χορηγηθούν μόνο σε εφαρμογές οι οποίες είναι μέρος του λειτουργικού συστήματος Android, είτε επειδή είναι προ-εγκατεστημένες είτε έχουν υπογραφεί με το ίδιο κλειδί του λειτουργικού συστήματος. Οι εφαρμογές τρίτων μπορούν να ορίσουν τα δικά τους δικαιώματα και παρόμοιους περιορισμούς γνωστούς ως “επίπεδα προστασίας δικαιωμάτων” (permission protection levels), περιορίζοντας έτσι την πρόσβαση της εφαρμογής στις υπηρεσίες και τους πόρους που δημιουργούνται από τον ίδιο προγραμματιστή.

Τα δικαιώματα μπορούν να εκτελεστούν σε διαφορετικά επίπεδα. Τα αιτήματα για πρόσβαση σε πόρους χαμηλού επιπέδου του συστήματος, όπως τα αρχεία της συσκευής, εκτελούνται από το πυρήνα του Linux, ελέγχοντας το UID ή GID της διεργασίας που το κάλεσε σε αντιπαράθεση με τον ιδιοκτήτη των πόρων του συστήματος. Κατά την πρόσβαση σε πόρους σε υψηλότερου επιπέδου του Android, η εκτέλεση πραγματοποιείται είτε από το λειτουργικό σύστημα είτε από κάθε μέρος του (ή και από τα δύο).

2.3 Επικοινωνία εσωτερικών διεργασιών (IPC)

Το Android χρησιμοποιεί έναν συνδυασμό οδηγού πυρήνα και userspace βιβλιοθηκών για να υλοποιήσει την επικοινωνία IPC. Ο οδηγός του πυρήνα Binder εγγυάται, ότι το UID και PID των καλούντων δεν μπορεί να είναι πλαστά. Πολλές υπηρεσίες του συστήματος εμπιστεύονται το UID και PID που παρέχεται από το Binder για δυναμικό έλεγχο της πρόσβασης στα ευαίσθητα APIs που εκτίθενται μέσω της επικοινωνίας IPC. Για παράδειγμα, η υπηρεσία του συστήματος που διαχειρίζεται το Bluetooth επιτρέπει μόνο στις εφαρμογές του συστήματος να την ενεργοποιήσουν.

Επιπλέον, άδειες που επηρεάζουν όλες τις μεθόδους της εφαρμογής εκτίθενται μέσω της επικοινωνίας IPC και μπορούν να εκτελεστούν αυτόματα από το σύστημα καθορίζοντας μια άδεια στη δήλωση της υπηρεσίας. Τα δικαιώματα που αιτείται μια εφαρμογή δηλώνονται στο αρχείο *AndroidManifest.xml*. Όπως ο δυναμικός έλεγχος των δικαιωμάτων στο παραπάνω παράδειγμα, τα δικαιώματα που αιτούνται τα ξεχωριστά στοιχεία της εφαρμογής υλοποιούνται σύμφωνα με το UID της καλούμενης διεργασίας, το οποίο λαμβάνεται μέσω του Binder. Το σύστημα χρησιμοποιεί μια βάση δεδομένων για να καθορίσει τα δικαιώματα που απαιτούνται από το στοιχείο που το κάλεσε, και στη συνέχεια αντιστοιχεί το UID σε ένα όνομα πακέτου και

ανακαλεί το σύνολο των δικαιωμάτων που χορηγούνται στην καλούμενη υπηρεσία. Εάν τα απαιτούμενα δικαιώματα βρίσκονται σε αυτό το σύνολο, η κλήση είναι επιτυχής. Εάν όχι, αποτυγχάνει και το σύστημα παράγει εξαίρεση (*SecurityException*).

2.4 Binders

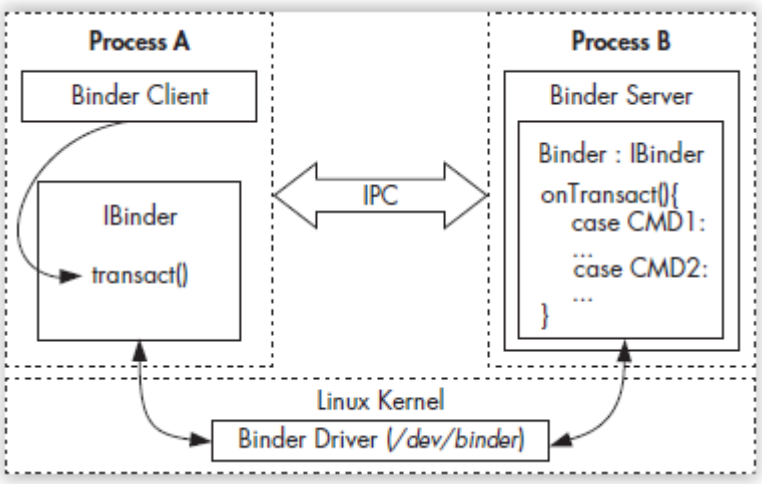
Επειδή ο βασικός μηχανισμός IPC δεν ήταν αρκετά εύχρηστος και αξιόπιστος, ένας νέος μηχανισμός αναπτύχθηκε για το Android και ονομάζεται *Binder* .Ο binder για τα Android αποτελεί μια νέα υλοποίηση, είναι βασισμένο στην αρχιτεκτονική και τις ιδέες του OpenBinder.6

Ο Binder υλοποιείται ως ένα στοιχείο με κατανεμημένη αρχιτεκτονική που βασίζεται στην αφηρημένη διεπαφή (abstract interface). Είναι παρόμοιο με το Windows Common Object Model (COM) και το Common Object Broker Request Architectures (CORBA) στο Unix, αλλά σε αντίθεση με αυτά τα πλαίσια εργασίας, λειτουργεί σε μια συσκευή και δεν υποστηρίζει απομακρυσμένες κλήσεις (remote procedure calls RPC) μέσω δικτύου. Τα βασικά στοιχεία του Binder παρουσιάζονται στις παρακάτω ενότητες.

2.4.1 Υλοποίηση Binder

Όπως αναφέρθηκε νωρίτερα , σε ένα σύστημα Unix, μια διεργασία δε μπορεί να έχει πρόσβαση στη μνήμη μιας άλλης διεργασίας. Παρόλα αυτά, ο πυρήνας έχει τον έλεγχο όλων των διεργασιών και ως εκ τούτου μπορεί να εκθέσει μια διεπαφή που ενεργοποιεί την IPC επικοινωνία. Η διεπαφή του Binder είναι η συσκευή */dev/binder*, η οποία υλοποιείται από τον οδηγό πυρήνα Binder. Ο οδηγός Binder είναι ένα κεντρικό αντικείμενο του πλαισίου εργασίας και όλες οι κλήσεις IPC περνούν μέσα από αυτόν. Η επικοινωνία *inter-process* υλοποιείται με μια μόνο κλήση *ioctl()* η οποία στέλνει και λαμβάνει δεδομένα μέσω της δομής *binder_write_read*. Η τελευταία αποτελείται από ένα *write_buffer* που περιέχει εντολές για τον οδηγό, και έναν *read_buffer* που περιέχει εντολές που χρειάζεται να εκτελεστούν στο *userspace*.

Ο οδηγός Binder διαχειρίζεται μέρος του χώρου της μνήμης για κάθε διεργασία προοριζόμενο μόνο για ανάγνωση της διεργασίας αυτής, ενώ όλες οι εγγραφές εκτελούνται από τον πυρήνα. Όταν μια διεργασία στέλνει ένα μήνυμα σε μια άλλη διεργασία, ο πυρήνας διαθέτει κάποιο χώρο μνήμης για τη διεργασία στόχος και αντιγράφει τα δεδομένα του μηνύματος απευθείας από τη διεργασία που αποστέλλει το μήνυμα. Στη συνέχεια ένα σύντομο μήνυμα αποστέλλετε στη διεργασία παραλήπτη όπου την ενημερώνει για το που βρίσκεται το μήνυμα που πρέπει να παραληφθεί. Ο παραλήπτης τότε μπορεί να έχει απευθείας πρόσβαση (διότι είναι στον ίδιο χώρο μνήμης). Όταν έχει ολοκληρωθεί η διαδικασία με το μήνυμα, η διεργασία ενημερώνει τον οδηγό Binder να ελευθερώσει το χώρο μνήμης που είχε δεσμεύσει. Στην εικόνα 3 φαίνεται η αρχιτεκτονική του Binder



Εικόνα 3: Binder IPC.[2]

Οι IPC επικοινωνίες σε υψηλότερο επίπεδο στο Android όπως *Intents* (εντολές με τα αντίστοιχα δεδομένα που παραδίδονται σε στοιχεία μέσω των διεργασιών), *Messengers* (αντικείμενα που επιτρέπουν την ανταλλαγή μηνυμάτων μεταξύ των διεργασιών) και *ContentProviders* (στοιχεία που εκθέτουν τη διαχείριση δεδομένων μεταξύ των διεργασιών) υλοποιούνται μέσω του Binder. Επιπλέον , υπηρεσίες διεπαφής που πρέπει να εκτεθούν σε άλλες διεργασίες μπορούν να ορισθούν με τη χρήση του *Android Interface Definition Language (AIDL)*, επιτρέποντας έτσι στους πελάτες να καλούν απομακρυσμένες υπηρεσίες σαν να ήταν τοπικά αντικείμενα της Java. Το εργαλείο *aidl* δημιουργεί αυτόματα *stubs (*αναπαράσταση του αντικειμένου από τη πλευρά του πελάτη) και *proxies* που αντιστοιχεί τις μεθόδους της διεπαφής σε χαμηλότερου επιπέδου μεθόδους του Binder και φροντίζει για τη μετατροπή των παραμέτρων σε μια μορφή που μπορεί να μεταδοθεί από τον Binder (αυτή η παράμετρος ονομάζεται *marshalling/unmarshalling)*. Τα stubs και proxies που δημιουργούνται από το AIDL παρέχουν επίσης ασφάλεια τύπου περιλαμβάνοντας το όνομα της διεπαφής στόχος σε κάθε συναλλαγή Binder (στο proxy) και επικυρώνοντας το στο stub.

2.4.2 Ασφάλεια Binder

Σε ένα υψηλότερο επίπεδο ασφαλείας, κάθε αντικείμενο που μπορεί να έχει πρόσβαση μέσω του Binder υλοποιεί τη διεπαφή *IBinder* και καλείται αντικείμενο Binder. Οι κλήσεις σε ένα αντικείμενο Binder πραγματοποιούνται μέσα στη συναλλαγή Binder, η οποία περιέχει μια αναφορά στο αντικείμενο στόχο, το αναγνωριστικό ID της μεθόδου που εκτελείται και τα δεδομένα. Ο οδηγός Binder προσθέτει αυτόματα το αναγνωριστικό ID της διεργασίας (PID) και του χρήστη (EUID) της διεργασίας καλώντας στα δεδομένα της συναλλαγής. Η αποκαλούμενη διεργασία (*callee*) μπορεί να επιθεωρήσει το PID , EUID και να αποφασίσει αν θα πρέπει να εκτελέσει την αιτούμενη μέθοδο που βασίζεται στην εσωτερική λογική του.

Δεδομένου ότι τι PID και EUID συμπληρώνονται από τον πυρήνα, οι διεργασίες του καλούμενου δεν μπορούν να παραλλάξουν την ταυτότητάς τους για να αποκτήσουν περισσότερα προνόμια από αυτά που τους επιτρέπονται από το σύστημα (ο Binder εμποδίζει την κλιμάκωση προνομίων) . αυτό είναι ένα από τα κεντρικά κομμάτια στην ασφάλεια του Android, όπως τα δικαιώματα βασίζονται στη συγκεκριμένη λειτουργία. Το EUID και PID του

καλούμενου είναι προσβάσιμα μέσω των μεθόδων *getCallingPid() και getCallingUid()* της κλάσσης A*ndroid.os.Binder,* το οποίο αποτελεί μέρος του Android API.

2.4.3 Ταυτότητα Binder

Μια από τις πιο σημαντικές ιδιότητες των αντικειμένων Binder είναι ότι διατηρούν μια μοναδική ταυτότητα σε όλες τις διεργασίες. Έτσι, αν η διαδικασία A δημιουργεί ένα αντικείμενο Binder και περνάει για επεξεργασία στη διεργασία B, η οποία με τη σειρά της τη μεταδίδει στη διεργασία C, οι κλήσεις και από τις τρεις διεργασίες θα αναλυθούν από το ίδιο αντικείμενο Binder. Στη πραγματικότηταμ η διεργασία A θα αναφέρεται απευθείας στο αντικείμενο Binder από τη διεύθυνση μνήμης, ενώ οι διεργασίες B και C θα λάβουν μόνο έναν handler για το αντικείμενο Binder.

Ο πυρήνας διατηρεί την αντιστοίχηση μεταξύ του “ζωντανού” αντικειμένου Binder και των handlers των άλλων διεργασιών. Επειδή η ταυτότητα ενός αντικειμένου Binder είναι μοναδική και συντηρείται από τον πυρήνα, είναι αδύνατο για τις διεργασίες του userspace να δημιουργήσουν ένα αντίγραφο του αντικειμένου Binder . έτσι ένα αντικείμενο Binder είναι μοναδικό που μπορεί να λειτουργήσει ως ασφάλεια *token*. Αυτό επιτρέπει τη χρήση του *capability-based security* στο Android.

2.4.4 Capability-Based Security

Στο μοντέλο ασφάλειας *capability-based* , τα προγράμματα έχουν πρόσβαση σε έναν συγκεκριμένο πόρο του συστήματος , δίνοντας τους μια ικανότητα (*unforgeable capability)*, γνωστή ως «κλειδί». Συγκεκριμένα, Αναφέρεται σε μια τιμή που δείχνει ένα αντικείμενο μαζί με ένα σύνολο δικαιωμάτων πρόσβασης. Μόνο το γεγονός, ότι ένα πρόγραμμα διαθέτει ένα κλειδί αρκεί για να δώσει πρόσβαση στους πόρους του συστήματος. Συνεπώς, δεν υπάρχει η ανάγκη για τη διατήρηση της λίστας ελέγχου πρόσβασης ACL ή παρόμοιες δομές που σχετίζονται με τους πόρους του συστήματος.

2.4.5 Android Token

Στο Android, τα αντικείμενα Binder μπορούν να λειτουργήσουν ως δυνατότητες (*abilities)* και τα οποία ονομάζονται *Binder Tokens* όταν χρησιμοποιούνται με αυτόν τον τρόπο. Ένα Binder token μπορεί να είναι και μια ικανότητα και ένας πόρος στόχου. Η κατοχή ενός Binder token παρέχει στην ίδια τη διεργασία πλήρη πρόσβαση στο αντικείμενο Binder, και της επιτρέπει να πραγματοποιεί συναλλαγές με τον αντικείμενο στόχο. Αν το αντικείμενο Binder υλοποιεί πολλαπλές ενέργειες (επιλέγοντας την ενέργεια που πρέπει να εκτελεστεί με βάση τις παραμέτρους κώδικα της Binder συναλλαγής), ο καλούμενος τότε μπορεί να εκτελεί οποιαδήποτε ενέργεια όταν υπάρχει μια αναφορά στο αντικείμενο Binder. Εάν απαιτείται πιο λεπτομερής έλεγχος πρόσβασης, η υλοποίηση κάθε ενέργειας πρέπει να εφαρμόσει τους απαραίτητους ελέγχους δικαιωμάτων, συνήθως με τη χρήση του PID και EUID της διεργασίας του καλούντος.

Μια κοινή πρακτική στα Android είναι να επιτρέπονται όλες οι ενέργειες όταν ο καλούμενος χρήστης είναι το σύστημα (UID 1000) ή ο διαχειριστής (UID 0), αλλά εκτελούνται επιπλέον έλεγχοι δικαιωμάτων για όλες τις άλλες διεργασίες. Έτσι η πρόσβαση σε σημαντικά αντικείμενα Binder όπως οι υπηρεσίες συστήματος ελέγχονται με δυο τρόπους : με τον περιορισμό αυτών που μπορούν να λάβουν αναφορά στο αντικείμενο Binder και με τον έλεγχο της ταυτότητας πριν από την εκτέλεση μιας ενέργειας για το αντικείμενο Binder.

Εναλλακτικά , ένα αντικείμενο Binder μπορεί να χρησιμοποιηθεί μόνο ως ικανότητα χωρίς την εφαρμογή καμίας άλλης λειτουργίας. Με αυτή τη πρακτική το ίδιο το αντικείμενο

Binder έχει στη κατοχή του δυο συνεργαζόμενες διεργασίες, και ο ένας μόνο λειτουργεί ως διακομιστής χρησιμοποιώντας το Binder token για να πιστοποιήσει τον πελάτη.

Αυτή η μέθοδος χρησιμοποιείται εσωτερικά από το πλαίσιο εργασίας του Android και είναι ως επί το πλείστο αόρατη για τις εφαρμογές. Μια αξιοσημείωτη περίπτωση χρήσης του Binder token που είναι ορατό στο API, είναι τα *windows tokens*. Το παράθυρο κάθε δραστηριότητας συνδέεται με ένα Binder token , το οποίο παρακολουθεί ο διαχειριστής παραθύρων του Android. Οι εφαρμογές μπορούν να αποκτήσουν τα δικά τους window tokens αλλά δεν μπορούν να έχουν πρόσβαση στα windows tokens άλλων εφαρμογών.

2.4.6 Πρόσβαση σε αντικείμενα Binder

Παρά το γεγονός ότι το Android ελέγχει τη πρόσβαση στα αντικείμενα Binder για λόγους ασφαλείας, ο μόνο τρόπος για να επικοινωνήσει με το αντικείμενο Binder είναι να του δοθεί μια αναφορά σε αυτό, όμως κάποια αντικείμενα Binders πρέπει να προσβάσιμα σε όλους. Είναι όμως πρακτικό να μοιράζονται οι αναφορές σε όλες τις υπηρεσίες του συστήματος και σε κάθε διεργασία; Έτσι, απαιτείται ένας μηχανισμός που θα επιτρέπει στις διεργασίες να ανακαλύπτουν και να αποκτήσουν αναφορές σε υπηρεσίες του συστήματος.

Προκειμένου να ενεργοποιηθεί η υπηρεσία ανακάλυψης, το πλαίσιο εργασίας του Binder έχει ένα μόνο διαχειριστή περιεχομένου (*context manager*), ο οποίος διατηρεί τις αναφορές για τα αντικείμενα Binder. Η υλοποίηση του διαχειριστή περιεχομένου για Android είναι ο δαίμονας *servicemanager*. Εκκινείται πολύ νωρίς κατά τη διαδικασία boot έτσι ώστε οι υπηρεσίες συστήματος να μπορούν να εγγραφούν καθώς ξεκινάνε. Οι υπηρεσίες εγγράφονται περνώντας ένα όνομα υπηρεσίας και μια αναφορά Binder στο διαχειριστή υπηρεσίας. Μόλις μια υπηρεσία έχει καταχωρηθεί , κάθε πελάτης μπορεί να αποκτήσει τη δική του αναφορά Binder χρησιμοποιώντας το όνομά του. Ωστόσο, οι περισσότερες υπηρεσίες συστήματος εφαρμόζουν πρόσθετους ελέγχους δικαιωμάτων, λαμβάνοντας έτσι μια αναφορά δεν εγγυάται αυτομάτως την πρόσβαση σε όλες τις λειτουργίες του . επειδή ο καθένας μπορεί να έχει πρόσβαση σε μια αναφορά Binder. Όταν έχει η αναφορά αυτή έχει καταχωρηθεί από τον διαχειριστή υπηρεσιών, μόνο ένα μικρό πλήθος πιστοποιημένων διεργασιών μπορεί να καταχωρήσει υπηρεσίες συστήματος.

Παρακάτω φαίνεται η λίστα των καταχωρημένων υπηρεσιών με χρήση της εντολής *service list* , η οποία επιστρέφει το όνομα κάθε καταχωρημένης υπηρεσίας και της υλοποίηση της διεπαφής IBinder. Παράδειγμα της εντολής φαίνεται στη παρακάτω εικόνα

```
$ service list
service list
Found 79 services:
0       sip: [android.net.sip.ISipService]
1       phone: [com.android.internal.telephony.ITelephony]
2       iphonesubinfo: [com.android.internal.telephony.IPhoneSubInfo]
3       simphonebook: [com.android.internal.telephony.IIccPhoneBook]
4       isms: [com.android.internal.telephony.ISms]
5       nfc: [android.nfc.INfcAdapter]
6       media_router: [android.media.IMediaRouterService]
7       print: [android.print.IPrintManager]
8       assetatlas: [android.view.IAssetAtlas]
9       dreams: [android.service.dreams.IdreamManager]
--snip--
```

Εικόνα 4: Χρήση service list[2]

2.5 Υπογραφή κώδικα και κλειδιά πλατφόρμας

Όλες οι εφαρμογές Android πρέπει να υπογραφούν από τους προγραμματιστές τους, συμπεριλαμβανομένων και των εφαρμογών συστήματος. Επειδή τα αρχεία APK του Android είμαι μια επέκταση της μορφής των αρχείων JAR της Java, η μέθοδος υπογραφής του κώδικα που χρησιμοποιείται βασίζεται στην υπογραφή των αρχείων JAR. Το Android χρησιμοποιεί την υπογραφή του APK για να εξασφαλίσει ότι οι αναβαθμίσεις της εφαρμογής γίνονται από τον ίδιο προγραμματιστή (αυτό ονομάζεται *same origin policy*) και για να εδραιώσει σχέση εμπιστοσύνης μεταξύ των εφαρμογών. Και τα δυο χαρακτηριστικά ασφάλειας υλοποιούνται με τη σύγκριση του υπογεγραμμένου πιστοποιητικού της τρέχουσας εγκατεστημένης εφαρμογής με το πιστοποιητικό της εφαρμογής που θα αναβαθμιστεί.

Οι εφαρμογές συστήματος υπογράφονται από έναν αριθμό κλειδιών πλατφόρμας (*Platform Keys)*. Διαφορετικά στοιχεία του συστήματος μπορούν να μοιραστούς τους πόρους και να εκτελεστούν μέσα στην ίδια διεργασία όταν έχουν υπογραφεί από το ίδιο κλειδί πλατφόρμας. Τα κλειδιά πλατφόρμας δημιουργούνται και ελέγχονται από όποιον συντηρεί την έκδοση του Android που έχει εγκατασταθεί στη συγκεκριμένη συσκευή (κατασκευαστές κινητών τηλεφώνων, πάροχοι υπηρεσιών).

2.6 Υποστήριξη πολλαπλών χρηστών

Το Android αρχικά σχεδιάστηκε για έξυπνα τηλέφωνα, συσκευές δηλαδή οι οποίες έχουν ένα φυσικό χρήστη και εκχωρούν ένα μοναδικό UID σε κάθε εγκατεστημένη εφαρμογή. Το Android ξεκίνησε να υποστηρίζει τη δυνατότητα για πολλαπλούς χρήστες από την έκδοση 4.2, ωστόσο η υποστήριξη πολλαπλών χρηστών είναι ενεργοποιημένη μόνο στις συσκευές tablet. Η υποστήριξη πολλαπλών χρηστών στα κινητά τηλέφωνα είναι απενεργοποιημένη, θέτοντας το μέγιστο αριθμό χρηστών σε 1.

Σε κάθε χρήστη εκχωρείτε ένα μοναδικό αναγνωριστικό χρήστη ID, με αρχικό 0 ενώ οι χρήστες έχουν τον δικό τους κατάλογο δεδομένων στο μονοπάτι */data/system/users/<user ID>*, το οποίο καλείτε “κατάλογο συστήματος του χρήστη”. Στον κατάλογο αυτό αποθηκεύονται πληροφορίες του χρήστη όπως τα δεδομένα του λογαριασμού του, παράμετροι για τη homescreen και μια λίστα με τις εφαρμογές που έχουν εγκατασταθεί. Ενώ τα αρχεία της εφαρμογής μοιράζονται μεταξύ των χρηστών, κάθε χρήστης παίρνει ένα αντίγραφο του καταλόγου των δεδομένων των εφαρμογών.

Το Android για να διακρίνει τις εφαρμογές που έχουν εγκατασταθεί για ένα χρήστη, εκχωρεί ένα νέο ενεργό UID για κάθε εφαρμογή όταν εγκαθίσταται για ένα συγκεκριμένο χρήστη. Αυτό το ενεργό UID βασίζεται στην ταυτότητα του φυσικού χρήστη μέσω της χρήσης του *user ID* και το UID της εφαρμογής (*appID*). Η δομή με την οποία συντίθεται η χορήγηση UID εγγυάται, ότι ακόμα και αν η ίδια εφαρμογή εγκατασταθεί από δύο διαφορετικούς χρήστες, και τα δύο στιγμιότυπα της εφαρμογής θα έχουν το δικό τους sandbox. Επιπρόσθετα, το Android εγγυάται τη διανομή του αποθηκευτικού χώρου σε κάθε φυσικό χρήστη. Ο πρώτος χρήστης που ενεργοποιεί τη συσκευή καλείται ιδιοκτήτης (*owner*), και μπορεί μόνο να διαχειριστεί άλλους χρήστες ή να πραγματοποιήσει εργασίες διαχειριστή οι οποίες θα επηρεάσουν όλη τη συσκευή.

2.7 SELinux

Το παραδοσιακό μοντέλο ασφάλειας στο Android εξαρτάται σε μεγάλο βαθμό από τα UIDs και GIDs που εκχωρούνται στις εφαρμογές. Παρότι αυτά εξαρτώνται από τον πυρήνα, και τα αρχεία κάθε εφαρμογής είναι ιδιωτικά από προεπιλογή, τίποτε δεν εμποδίζει μια εφαρμογή από το να έχει πρόσβαση στα αρχεία της.

Ομοίως, τίποτα δεν εμποδίζει μια κακόβουλη εφαρμογή να επωφεληθεί από τα δικαιώματα που θα τις εκχωρηθούν και να αποκτήσει πρόσβαση στο σύστημα αρχείων και τις δικτυακές επικοινωνίες. Στην πραγματικότητα, η μη σωστή εκχώρηση δικαιωμάτων σε μια εφαρμογή ή σύστημα αρχείων ήταν η πηγή μεγάλου αριθμού τρωτών σημείων του λειτουργικού συστήματος Android. Αυτές οι ευπάθειες είναι αναπόφευκτες στο προεπιλεγμένο μοντέλο ελέγχου πρόσβασης που χρησιμοποιείτε από το Linux, γνωστό ως «διακριτός έλεγχος πρόσβασης *DAC»* (discretionary access control). Διακριτός σημαίνει, ότι, μόλις ένας χρήστης αποκτήσει πρόσβαση στον συγκεκριμένο πόρο, μπορεί να τον δώσει σε κάποιον άλλο χρήστη, κατά τη κρίση του.Αντίθετα, ο υποχρεωτικός έλεγχος πρόσβασης *MAC* (mandatory access control) εξασφαλίζει, ότι η πρόσβαση στους πόρους είναι σύμφωνη με το σύνολο των κανόνων αδειοδότησης, η οποία ονομάζεται πολιτική. Η πολίτική μπορεί να αλλάξει μόνο από τον διαχειριστή, ενώ οι χρήστες δεν μπορούν να το παρακάμψουν ή το παραβιάσουν προκειμένου, για παράδειγμα, να χορηγήσουν στον οποιοδήποτε την πρόσβαση στα δικά τους αρχεία.

Security Enhanced Linux (SELinux) είναι η μια υλοποίηση του MAC για τον πυρήνα του Linux και έχει ενσωματωθεί στα βασικά χαρακτηριστικά του πυρήνα για πάνω από 10 χρόνια. Από την έκδοση 4.3, το Android έχει ενσωματώσει μια τροποποιημένη έκδοση, το SELinux, η οποία υποστηρίζει ειδικά χαρακτηριστικά του Android, όπως τους Binder. Στο Android , το SELinux χρησιμοποιείται για να απομονώσει τις διεργασίες συστήματος του πυρήνα και τις εφαρμογές του χρήστη σε διαφορετικές περιοχές ασφάλειας και να καθορίζει διαφορετικές πολιτκές για κάθε χώρο. Από την έκδοση 4.4, το SELinux εξελίχθηκε σε λειτουργία επιβολής (παραβιάσεις του συστήματος δημιουργούν λάθη στο χρόνο εκτέλεσης του συστήματος), αλλά αυτή η πολιτική εφαρμόζεται μόνο στις διεργασίες συστήματος του πυρήνα.

2.8 Ενημέρωση Συστήματος

Οι συσκευές Android μπορούν να ενημερωθούν *over-the-air* (OTA) είτε συνδέοντας τη συσκευή σε έναν υπολογιστή χρησιμοποιώντας τη βασική εφαρμογή *Android debug bridge* (ADB) είτε από κάποιον άλλο πάροχο, εφόσον η εφαρμογή αυτή έχει παρόμοια λειτουργικότητα. Η συνάφεια της λειτουργικότητας πρέπει να υφίσταται , εφόσον, εκτός από τα αρχεία συστήματος, μια ενημέρωση του Android μπορεί να χρειαστεί να τροποποιήσει το firmware , bootloader και άλλα μέρη της συσκευής, τα οποία δεν είναι άμεσα προσβάσιμα από το λειτουργικό σύστημα. Η διαδικασία της ενημέρωσης, τυπικά, χρησιμοποιεί ένα ειδικού σκοπού λειτουργικό σύστημα OS

με αποκλειστική πρόσβαση σε όλο το υλικό της συσκευής. Αυτό ονομάζεται λειτουργικό σύστημα ανάκτησης *recover OS*.

Οι OTA ενημερώσεις εκτελούνται με τη λήψη ενός αρχείου πακέτου OTA (τυπικά είναι ένα συμπιεσμένο αρχείο ZIP, στο οποίο έχει προστεθεί η υπογραφή του κώδικα), το οποίο περιέχει ένα αρχείο script για να παρέμβει στο λειτουργικό σύστημα ανάκτησης και να εκκινηθεί η συσκευή σε λειτουργία ανάκτησης. Ενναλακτικά , ο χρήστης μπορεί να εισέλθει σε λειτουργία ανάκτησης με τη χρήση ειδικού συνδυασμού κλειδιών, όταν εκκινείται η συσκευή και να εφαρμόσει την ενημέρωση χειροκίνητα επιλέγοντάς την από το μενού επιλογών, στο οποίο συνήθως η πλοήγηση γίνεται με τη χρήση των κουμπιών της συσκευής.

Στις συσκευές παραγωγής η διαδικασία της ανάκτησης δέχεται ενημερώσεις που έχουν υπογραφεί μόνο από τον κατασκευαστή της συσκευής. Τα αρχεία ενημερώσεων υπογράφονται με την επέκταση της μορφής του αρχείου ZIP ,ώστε η υπογραφή να έχει ισχύ σε όλο το αρχείο. Τα αρχεία ενημερώσεων εξάγονται και πιστοποιούνται, πριν από την εγκατάσταση της ενημέρωσης. Σε μερικές συσκευές (συμπεριλαμβανομένων όλων των συσκευών Nexus και μερικών κατασκευαστών τηλεφώνων), μπορεί να αντικατασταθεί το λειτουργικό σύστημα ανάκτησης και να απενεργοποιηθεί το σύστημα ενημερώσεων και πιστοποίησης αδειών, επιτρέποντας με αυτόν τον τρόπο την εγκατάσταση ενημερώσεων από τρίτα μέρη. Θέτοντας στη συσκευή το bootloader σε λειτουργία , επιτρέπεται η αλλαγή του λειτουργικού συστήματος ανάκτησης. Η λειτουργία ονομάζεται *bootloader unlocking* και απαιτεί καθαρισμό των δεδομένων του χρήστη για να βεβαιωθεί, ότι καμία κακόβουλη εικόνα του συστήματος δε θα φορτωθεί για να αποκτήσει πρόσβαση στα δεδομένα του χρήστη. Στις περισσότερες εμπορικές συσκευές το ξεκλείδωμα του bootloader έχει ως αποτέλεσμα να ακυρωθεί η εγγύηση της συσκευής.

2.9 Επαλήθευση εκκίνησης

Το Android από την έκδοση 4.4 υποστηρίζει την επαλήθευση εκκίνησης (*verified boot*) χρησιμοποιώντας το *verity target* ενός Linux's Device-Mapper. Η διαδικασία verity παρέχει έλεγχο της ακεραιότητας των συσκευών με χρήση κρυπτογραφικών hash δένδρων. Κάθε κόμβος του δέντρου είναι ένας κρυπτογραφικός hash, με φύλλα κόμβους που περιέχουν τις τιμές hash ενός φυσικού μπλοκ δεδομένων και των ενδιάμεσων κόμβων που περιέχουν τιμές hash των παιδιών του κόμβου. Επειδή το hash στον κόμβο του διαχειριστή βασίζεται στις τιμές όλων των άλλων κόμβων, μόνο το hash του διαχειριστή πρέπει να είναι αξιόπιστο, προκειμένου να ελέγξει το υπόλοιπο δέντρο.

Η επαλήθευση επιτυγχάνεται με ένα δημόσιο κλειδί RSA που περιλαμβάνεται στο boot partition. Η συσκευή ελέγχεται στο χρόνο εκτέλεσης υπολογίζοντας την τιμή hash, όπως διαβάζεται, και συγκρίνοντάς την με την καταγεγραμμένη τιμή hash του δένδρου. Εάν οι τιμές δεν ταιριάζουν, το αποτέλεσμα της λειτουργίας ανάγνωσης είναι ένα σφάλμα εισόδου/εξόδου που υποδεικνύει, ότι το αρχείο συστήματος είναι κατεστραμμένο. Επειδή όλοι οι έλεγχοι εκτελούνται από τον πυρήνα, η διαδικασία εκκίνησης πρέπει να επαληθεύσει την ακεραιότητα του πυρήνα, προκειμένου να πιστοποιηθεί η διαδικασία εκκίνησης. Αυτή η διαδικασία είναι συγκεκριμένη για κάθε συσκευή και τυπικά υλοποιείται με τη χρήση ενός αναλλοίωτου και συγκεκριμένου κλειδιού το οποίο "καίγεται" (εγγράφεται στη περιοχή μνήμης write-only) μέσα στη συσκευή. Αυτό το κλειδί χρησιμοποιείται για να επαληθεύσει την ακεραιότητα κάθε επιπέδου του bootloader και κατά συνέπεια τον πυρήνα.

3 Ζητήματα ασφάλειας στο Android

Η ασφάλεια του Android είναι χτισμένη πάνω σε έναν μηχανισμό αδειών, ο οποίος ρυθμίζει την πρόσβαση εφαρμογών τρίτων μελών σε κρίσιμους πόρους της συσκευής. Τέτοιου είδους μηχανισμός έχει επικριθεί για τον έλεγχο αδειών των εφαρμογών και για την αναποτελεσματική διαχείριση τους. Για παράδειγμα, οι χρήστες είτε μπορεί να δεχθούν όλες τις άδειες που απαιτεί η εφαρμογή για την εγκατάστασή της είτε να μην εγκαταστήσουν την εφαρμογή.[4] Αυτό το είδος διαχείρισης της άδειας αποδείχθηκε, ότι είναι ανεπιθύμητο για την ασφάλεια των συσκευών. Στην ενότητα αυτή θα συζητήσουμε τα κύρια θέματα της ασφάλειας στο Android.[3]

3.1 Διαρροή πληροφοριών (Information leakage)

Η αρχιτεκτονική του Android έχει σχεδιαστεί με τέτοιον τρόπο που περιορίζει τις εφαρμογές να έχουν πρόσβαση σε πόρους ή άλλες εφαρμογές, εκτός αν έχει εγκριθεί από τον χρήστη. Οι χρήστες πρέπει να χορηγούν όλα τα αιτήματα για πρόσβαση στους πόρους, πριν από την εγκατάσταση και χρήση της εφαρμογής. Η διαρροή πληροφοριών εμφανίζεται, όταν οι χρήστες χορηγούν την άδεια για πρόσβαση στους πόρους του συστήματος χωρίς κανέναν περιορισμό από το λειτουργικό σύστημα. Ωστόσο, ο μηχανισμός ελέγχου των αδειών του Android έχει αποδειχθεί αναποτελεσματικός στην προστασία των προσωπικών δεδομένων του χρήστη. Μελέτες έχουν δείξει, ότι το 70% των εφαρμογών για έξυπνα τηλέφωνα ζητούν να συλλέγουν δεδομένα άσχετα με τη βασική λειτουργία της εφαρμογής[5][6]. Με περισσότερες από 1,4 εκατομμύρια εφαρμογές στο Google Play και έναν μεγάλο αριθμό εφαρμογών από τρίτους κατασκευαστές , ένας σημαντικός αριθμός κακόβουλων εφαρμογών έχουν διανεμηθεί σε χρήστες για την εγκατάσταση τους. Ωστόσο, κατά την εγκατάσταση μόνο ένα μικρό μέρος των χρηστών δίνουν προσοχή στους πόρους που θα έχει πρόσβαση η εφαρμογή που εγκαθιστούν, δεδομένου, ότι βιάζονται να την χρησιμοποιήσουν. Μόνο ένα μικρό ποσοστό (3%) των χρηστών είναι προσεκτικό και δίνει τις σωστές απαντήσεις στη χορήγηση των αδειών που αιτείται η εφαρμογή.

Οι λόγοι για την αναποτελεσματικότητα του υφιστάμενου συστήματος ελέγχου αδειών περιλαμβάνει : (1) οι άπειροι χρήστες δεν αντιλαμβάνονται, ότι τα αιτήματα για πρόσβαση στους πόρους δεν είναι σχετικά με την εφαρμογή που θέλουν να εγκαταστήσουν και έτσι θέτουν σε κίνδυνο τα προσωπικά τους δεδομένα.(2) οι χρήστες έχουν την ανάγκη να χρησιμοποιήσουν την εφαρμογή και μπορεί να υποχρεωθούν να ανταλλάσσουν την ιδιωτικότητά τους για τη χρήση της εφαρμογής.

3.2 Κλιμάκωση προνομίων (Privilege escalation)

Η κλιμάκωση προνομίων (privilege escalation) και οι επιθέσεις αυτού του είδους οφείλονται στην αξιοποίηση των ευπαθειών του πυρήνα του Android για να αποκτήσουν αυξημένη πρόσβαση σε πόρους του συστήματος που κανονικά προστατεύονται από μια εφαρμογή ή έναν χρήστη. Αυτού του είδους η επίθεση μπορεί να οδηγήσει σε μη εξουσιοδοτημένες ενέργειες από την εφαρμογή με περισσότερα προνόμια από αυτά που προβλέπονται, τα οποία μπορούν να προκαλέσουν διαρροή εμπιστευτικής πληροφορίας. Τα μέρη (component) του Android που χρησιμοποιούνται σε μια εφαρμογή μπορεί να τα εκμεταλλευτεί ο επιτιθέμενος για να αποκτήσει πρόσβαση σε κρίσιμα δικαιώματα. [7]

3.3 Επανασυσκευασία Εφαρμογής (Repackaging App)

Η επανασυσκευασία (repackaging) της εφαρμογής είναι ένα από τα πιο σημαντικά και κοινά θέματα ασφαλείας του Android. Η επανασυσκευασία είναι μια διαδικασία αποσυναρμολόγησης/μεταγλώττισης (disassembling/decompiling) ενός αρχείου .apk με τη χρήση τεχνικών αντίστροφης μηχανικής (reverse engineering) και την προσθήκη (injecting) κακόβουλου κώδικα μέσα στον κύριο πηγαίο κώδικα. Η τεχνική της επανασυσκευασίας που μπορεί να χρησιμοποιηθεί στην πλατφόρμα Android επιτρέπει στον κακόβουλο κώδικα να μοιάζει με κανονική εφαρμογή. Είναι δύσκολο να γίνει διάκριση μεταξύ ενός επανασυσκευασμένου κακόβουλου κώδικα και της κανονικής εφαρμογής, διότι η επανασυσκευασμένη εφαρμογή φαίνεται να λειτουργεί με τον ίδιο τρόπο με την κανονική. Τα βήματα που ακολουθούνται για την επανασυσκευασία της εφαρμογής είναι τα ακόλουθα:

Απόσυσκευασία (Unpacking): η διαδικασία αποσυσκευασίας πραγματοποιείται με τη χρήση εργαλείων που βασίζονται στην αντίστροφη μηχανική, όπως το *apktool*.

Μεταγλώττιση (decompiling): η μεταγλώττιση του πηγαίου κώδικα της Java με τη χρήση JAD και εξαγωγή των Java κλάσεων του πηγαίου κώδικα.

Προσθήκη κώδικα (code injection): προσθήκη κώδικα και πόρων στον κυρίως πηγαίο κώδικα χρησιμοποιώντας JDE (Java Development Environment).

Επανασυσκευασία (repacking): τα αρχεία επαναδομούνται (rebuilding) με τη χρήση του apktool και τα νέα αρχεία που παράγονται υπογράφονται με τη χρήση του j*arsigner*. Geimini και KungFu είναι παραδείγματα δούρειων ίππων (Trojan) που βασίζονται στην επανασυσκευασία του APK. [8,9]

3.4 Η επίθεση «άρνηση υπηρεσίας» (Denial of Service attack)

Ο αυξανόμενος αριθμός χρηστών έξυπνων τηλεφώνων και η επικράτηση των κινητών συσκευών που συνδέονται στο διαδίκτυο μπορεί να αποτελεί μια πλατφόρμα για την ανάπτυξη επιθέσεων DoS. Καθώς η πλειοψηφία αυτών των συσκευών δεν είναι εξοπλισμένη με τις ίδιες προστασίες, όπως ένας προσωπικός υπολογιστής (π.χ. αντϊικά προγράμματα), η υπερβολική χρήση της CPU, μνήμης, δικτυακού εύρους και η κατανάλωση της μπαταρίας είναι από τους βασικούς στόχους των επιθέσεων DoS.[10]

3.5 Συμπαιγνία (Colluding)

Η απειλή «συμπαιγνίας» είναι μια επίθεση πελάτη (client side attack). Σε αυτή την επίθεση ο χρήστης εγκαθιστά ένα σύνολο εφαρμογών που υλοποιήθηκαν από τον ίδιο προγραμματιστή και έχουν υπογραφεί από το ίδιο πιστοποιητικό. Μετά την εγκατάσταση, οι εφαρμογές εκμεταλλεύονται το κοινό UID και παίρνουν πρόσβαση σε όλες τις άδειες και τους πόρους του συστήματος.[11]

4 Φορείς επιθέσεων

Αν και τα έξυπνα τηλέφωνα είναι προστατευμένα από επιθέσεις μέσω των χαρακτηριστικών που παρέχει το λειτουργικό σύστημα, ένα κακόβουλο πρόγραμμα μπορεί να εισαχθεί σε μια συσκευή μέσω οποιουδήποτε φορέα επίθεσης που συνδέεται, συνήθως, με τις τελικές συσκευές. Το πιο σύνηθες σημείο εισόδου ενός κακόβουλου προγράμματος είναι μέσω της εγκατάστασης μίας εφαρμογής.Άλλες επιθέσεις μπορούν να πραγματοποιηθούν μέσω επισκέψεων σε κακόβουλες ιστοσελίδες, spam , κακόβουλα μηνύματα και κακόβουλες διαφημίσεις.

4.1 Δούρειοι ίπποι

Οι περισσότερες επιθέσεις πραγματοποιούνται μέσω των δούρειων ίππων. Είναι προγράμματα ή κώδικας, ο οποίος μεταμφιέζεται σε κάτι χρήσιμο. Μόλις εγκατασταθεί στο λειτουργικό σύστημα, προσπαθεί να εκτελέσει σκόπιμα επιβλαβείς ενέργειες. Δεν εκμεταλλεύονται κάποια τεχνική αδυναμία του λειτουργικού συστήματος ή της συσκευής για να πραγματοποιήσουν την επίθεση, αλλά κάνουν χρήση της ανθρώπινης ευπιστίας και άγνοιας του χρήστη. Ως εκ τούτου , αυτού του είδους η επίθεση είναι δύσκολο να αντιμετωπιστεί ή να δημιουργηθούν κατάλληλα εργαλεία, εφόσον αυτή δεν έγκειται σε τεχνική αδυναμία. Τα ανθρώπινα λάθη είναι δύσκολο να εξαλειφθούν, ενώ οι τεχνικές αδυναμίες επιλύονται από τους κατασκευαστές ή προγραμματιστές. Αυτά τα στοιχεία έχουν συμβάλει στη διάδοση των δούρειων ίππων.

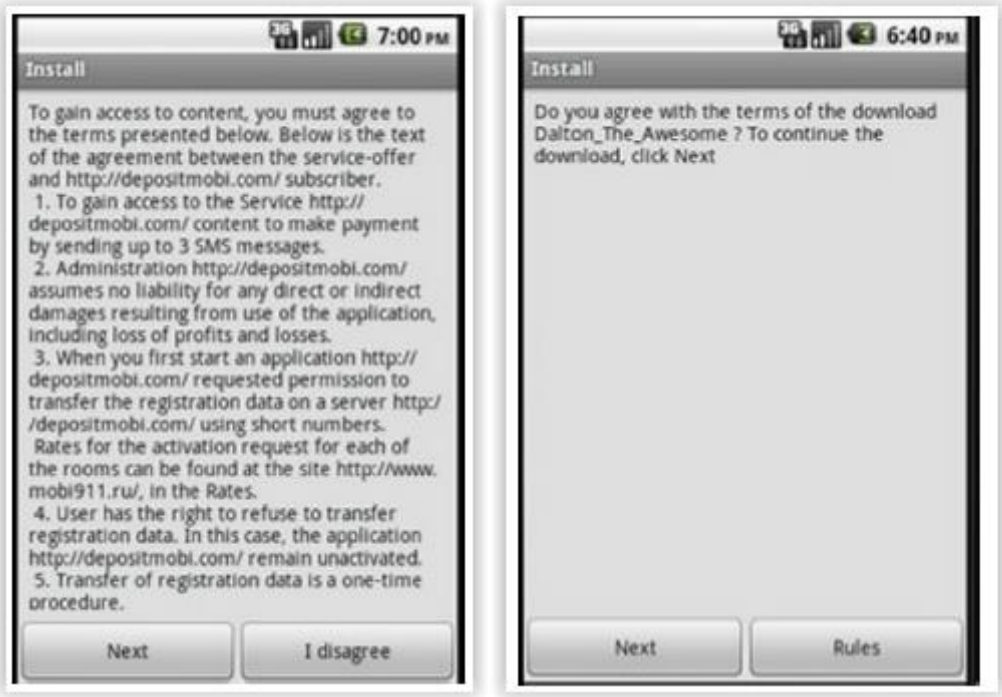
4.1.1 Πως λειτουργούν

Υπάρχουν διαφορετικοί τρόποι για να πραγματοποιηθούν οι επιθέσεις μέσω δούρειων ίππων και μπορούν να ταξινομηθούν σε τρεις κατηγορίες.

- Η μεταμφίεση σε μια νόμιμη εφαρμογή είναι η πιο κοινός τύπος. Ένας χρήστης, ο οποίος ψάχνει για μια εφαρμογή μέσω του δικτύου ή από τρίτα καταστήματα, είναι ένα τυπικό θύμα. Εάν η αναζήτηση οδηγήσει σε μια ιστοσελίδα που μοιάζει με την επίσημη , και περιέχει την περιγραφή της εφαρμογής, φωτογραφίες της , σχόλια χρηστών, και αν το εικονίδιο της εφαρμογής μοιάζει με το αυθεντικό, τότε ο χρήστης θα εγκαταστήσει την εφαρμογή.
- Η ενσωμάτωση του κακόβουλου κώδικα στη νόμιμη εφαρμογή είναι μια άλλη μορφή επίθεσης αλλά είναι πιο σπάνια. Εκμεταλλεύεται το γεγονός ότι οι χρήστες συχνά προτιμούν να αναζητούν τρίτους παρόχους που φιλοξενούν δωρεάν εφαρμογές από το να πληρώσουν τις νόμιμες εφαρμογές. Ο επιτιθέμενος ενσωματώνει τον κακόβουλο κώδικα στην εφαρμογή και την παρέχει δωρεάν.
- Ισχυρισμός, ότι πρόκειται για μια ενημέρωση κάποιας κρίσιμης εφαρμογής. Κακόβουλες ιστοσελίδες μπορεί να εμφανίσουν μια προτροπή που μιμείται μια ειδοποίηση για αναβάθμιση κάποιας εφαρμογής. Εάν ο χρήστης αποδεχτεί αυτή την ειδοποίηση, τότε η εφαρμογή κατεβαίνει στη συσκευή και εγκαθίσταται.
- Driven-by downloading. Είναι μια τεχνική, στην οποία το κατέβασμα της εφαρμογής εκκινείται αυτόματα, μόλις ο χρήστης επισκεφτεί μια κακόβουλη ιστοσελίδα.[13][14] Η μόνη ενέργεια που απαιτείται, είναι το φόρτωμα της ιστοσελίδας, η οποία φαίνεται ότι είναι μια αθώα ή νόμιμη ενέργεια. Στην πραγματικότητα η τεχνική αυτή δεν είναι κακόβουλη από μόνη της, καθώς τη χρησιμοποιούν και νόμιμες εφαρμογές για να πραγματοποιούν αναβαθμίσεις.

Μια τέτοιου είδους επίθεση επιτυγχάνεται σε συνδυασμό με τις διαφημίσεις. Κανονικά, αυτή η μέθοδος στοχεύει να μολύνει παραδοσιακά υπολογιστικά συστήματα, όμως υπάρχουν περιστατικά, στα οποία ο επιτιθέμενος στόχευε τις κινητές συσκευές. Ένα παράδειγμα επίθεσης, είναι το κακόβουλο λογισμικού GGTracker που εκμεταλλεύεται μια ενσωματωμένη διαφήμιση για να οδηγήσει τους χρήστες σε μια κακόβουλη ιστοσελίδα, ώστε να κατεβάσουν τον κακόβουλο κώδικα.

Μετά το κατέβασμα του κακόβουλου κώδικα με μια από τις παραπάνω μεθόδους, το κρίσιμο σημείο της επίθεσης δεν έχει ολοκληρωθεί. Δεδομένου, ότι οι περισσότερες επιθέσεις γίνονται μέσω λήψεων εκτός του Google Store, η εγκατάσταση δεν πραγματοποιείται αυτόματα αλλά πρέπει να εκτελεστεί χειροκίνητα από τον χρήστη. Κατά την εγκατάσταση, εμφανίζεται μήνυμα στον χρήστη, το οποίο του ζητάει να αποδεχθεί τα απαιτούμενα δικαιώματα. Όπως φαίνεται στην εικόνα 5., με την αποδοχή από την πλευρά του χρήστη, εγκαθίσταται η εφαρμογή και ο επιτιθέμενος έχει επιτύχει τον σκοπό του.[15]



Εικόνα 5 :Αποδοχή όρων και προϋποθέσεων[12]

Εξέλιξη της πολυπλοκότητας

Καθώς οι δούρειοι ίπποι έχουν αποδειχθεί, ότι είναι ένας επιτυχής τρόπος διάδοσης κακόβουλων προγραμμάτων, είναι σημαντικό να διευκρινιστεί, ότι τα περισσότερα περιστατικά συμβαίνουν μέσων τρίτων καταστημάτων ή από ανεπίσημους φορείς. Τα επίσημα κανάλια πωλήσεων έχουν μηχανισμούς ασφάλειας για να εντοπίσουν ή να μετριάσουν τις απειλές. Λόγω των παραπάνω, οι προγραμματιστές κακόβουλου κώδικα ενσωματώνουν στις επιθέσεις τους εξυπνάδα και πολυπλοκότητα, θέλοντας να προκαλέσουν ή να αμφισβητήσουν τις πιο προηγμένες μεθόδους ασφαλείας. Επίσης, μερικοί δούρειοι ίπποι, όπως το DroidDream και BadNews, έχουν καταφέρει να διαδοθούν μέσω του Google Play.

Ένα παράδειγμα βελτιωμένης επίθεσης, που δημιουργήθηκε για να παρακάμπτει την ανίχνευση ιών, είναι οι πολυμορφικοί διακομιστές. Οι πολυμορφικοί αυτοί διακομιστές έχουν την ικανότητα να ανταποκρίνονται διαφορετικά σε αίτημα της ίδιας διεύθυνση URL, όταν φτάνουν σε αυτούς από διαφορετικές πηγές. Κάνοντας χρήση της διεύθυνση IP του θύματος, ο διακομιστής μπορείς να δημιουργήσει ένα μοναδικό αρχείο APK με μικρές αλλαγές στην υλοποίηση του κώδικα, χωρίς να επηρεαστεί η τελική εφαρμογή. Αυτό αποτελεί ξεκάθαρα πλεονέκτημα για τον επιτιθέμενο, επειδή το μοναδικό αρχείο APK θα δημιουργήσει μια μοναδική τιμή HASH και θα θεωρηθεί ως μια καινούργια εφαρμογή. Έτσι, ο κακόβουλος κώδικας μπορεί να χρησιμοποιηθεί πολλές φορές, ακόμα και αν έχει ανιχνευθεί σε προηγούμενες περιπτώσεις.

Obfuscation και recompilation είναι μια άλλη τεχνική. Σημαίνει την αλλαγή της ονομασίας όλων των μεταβλητών και μεθόδων με αυθαίρετους χαρακτήρες, κάνοντας την αντίστροφη μηχανική αδύνατη. Σε τελική ανάλυση, έχει το ίδιο αποτέλεσμα. Πολλοί δούρειοι ίπποι περιλαμβάνουν τεχνικές αντί-αντίστροφης μηχανικής για να αποφεύγεται η δυναμική ανάλυση και η εκτέλεση του κακόβουλου κώδικα σε εξομοιωτή. [12]

4.1.2 Διαφορετικές επιπτώσεις

Οι δούρειοι ίπποι μπορούν να διαιρεθούν περαιτέρω σε υποομάδες , ανάλογα με το είδος των επιπτώσεων που προκαλούν. Οι πιο συνηθισμένες είναι η κλοπή πληροφοριών, η αποστολή μηνυμάτων σε προορισμούς με υψηλή χρέωση και η υποκλοπή των πόρων της συσκευής. Ωστόσο, αυτοί οι τρόποι που επηρεάζουν τον χρήστη δεν προκύπτουν μόνο από τις επιθέσεις των δούρειων ίππων.

Ψευδείς εγκατάσταση (Fake installer)

Η εταιρεία F-Secure στην αναφορά που συνέταξε για τις απειλές στις κινητές συσκευές διαπίστωσε, ότι στις τέσσερις από τις πέντε απειλές υπάρχει οικονομικό κίνητρο. [16] Μεταξύ αυτών , η αποστολή γραπτών μηνυμάτων σε προορισμούς υψηλής χρέωσης φαίνεται να είναι η επιλογή τους για τη δημιουργία εσόδων από το κακόβουλο λογισμικό τους, τα οποία είναι γνωστά ως fake installers. Το όνομα τους είναι παρόμοιο με την έννοια των δούρειων ίππων, αλλά αποτελούν μια υποκατηγορία λόγω του ειδικού σκοπού τους. Το 2012 διαπιστώθηκε ότι το 80% των δούρειων ίππων είναι fake installer, ενώ το 2013 αυτό το ποσοστό ήταν 73% και αποτελεί τη πιο διαδεδομένη μορφή κακόβουλου κώδικα.

Ο FakePlayer είναι ο πρώτος SMS δούρειος ίππος, ο οποίος στόχευε σε συσκευές Android και ανιχνεύθηκε τον Αύγουστο του 2010. Ο χρήστης έπρεπε χειροκίνητα να κατεβάσει και να εγκαταστήσει την εφαρμογή. Σε αυτό το σημείο, ο FakePlayer ζητούσε από τον χρήστη την άδεια να στέλνει μηνύματα SMS. Καθώς έτρεχε για πρώτη φορά, η εφαρμογή προσπαθούσε να στείλει μηνύματα σε κάποιον πάροχο της Ρωσίας, ενώ απέκρυπτε από τον χρήστη αυτή τη λειτουργία.

Οι περιοχές στις οποίες δραστηριοποιούνται τέτοιοι κακόβουλοι είναι η Ρωσία και χώρες της Ανατολική Ευρώπη. Οι χώρες αυτές δεν είναι τυχαίες, καθώς έχουν λιγότερο αυστηρούς νομικούς κανονισμούς σχετικά με τα premium SMS. Επιπλέον, η Google έχει πιο αδύναμη θέση στην αγορά. Επίσης, η κατάσταση είναι ίδια και στη Κίνα. Όλες αυτές οι χώρες έχουν αγορές τρίτων για τις εφαρμογές, οι οποίες κυριαρχούν στον τομέα της κατανάλωσης από τον χρήστη. Είναι σαφές, ότι χωρίς τα χαρακτηριστικά ασφάλειας που συνοδεύουν τα επίσημα κανάλια πωλήσεων, η δυνατότητα να μολυνθούν πολλοί χρήστες είναι υψηλότερη.[17]

Spyware

Τα Spyware είναι λογισμικό που συλλέγει πληροφορίες από τη συσκευή του χρήστη, χωρίς την συγκατάθεσή του. Ένα έξυπνο τηλέφωνο περιέχει πολλούς αισθητήρες που το καθιστούν ένα χρήσιμο εργαλείο παρακολούθησης, όπως η κάμερα, το GPS, μικρόφωνο, καθώς και τα προσωπικά δεδομένα που είναι αποθηκευμένα, όπως επαφές, ηλεκτρονικές διευθύνσεις, ημερολόγια και άλλες εφαρμογές. Τα spyware μπορούν να ταξινομηθούν σε δυο κατηγορίες , τα μη στοχευμένα (untargeted) και τα στοχευμένα (targeted) spywares.[13]

Τα μη στοχευμένα spyware είναι συχνά ένα φαινομενικά ασφαλές και αξιόπιστο λογισμικό, το οποίο, μόλις εγκατασταθεί, εκμεταλλεύεται τα προνόμια που παρέχονται από τον χρήστη και συλλέγει πληροφορίες. Ένα παράδειγμα αυτού, μπορεί να είναι μια εφαρμογή πρόγνωσης καιρού, η οποία ζητά από τον χρήστη την τοποθεσία του και πρόσβαση στο διαδίκτυο. Επίσης, μπορεί να χρησιμοποιηθεί για να μοιραστεί την τοποθεσία του χρήστη με ένα διακομιστή διαφήμισης, ώστε να μπορούν να στέλνουν στον χρήστη στοχευμένες διαφημίσεις. Τα μη στοχευμένα spyware δεν έχουν σχεδιαστεί για να κατασκοπεύουν τον χρήστη αλλά, αντίθετα για να συλλέγουν πληροφορίες. Είναι πιθανό να συλλέγει πληροφορίες, όπως δεδομένα τοποθεσίας, πληροφορίες των επαφών, δεδομένα περιήγησης και καταγραφής πληκτρολογίου (key logging).[18]

Τα στοχευμένα spyware, από την άλλη πλευρά, επικεντρώνονται σε συγκεκριμένα άτομα, όπως υποδηλώνει το όνομά τους. Ένα τυπικό παράδειγμα, είναι οι γονείς που κατασκοπεύουν τα παιδιά τους ή τους συζύγους. Τα στοχευμένα spyware τείνουν να συγκεντρώνουν περισσότερα δεδομένα από τα μη στοχευμένα spyware και μπορεί να έχουν τη δυνατότητα του απομακρυσμένου ελέγχου της συσκευής. Αυτού του είδους τα spyware είναι πιο δύσκολο να εγκατασταθούν από ότι τα untargeted, διότι απαιτείται φυσική πρόσβαση στη συσκευή για να εγκατασταθεί το κακόβουλο λογισμικό. Το TapSnake είναι ένα παράδειγμα ενός στοχευμένου spyware, το οποίο ανιχνεύθηκε το 2010. Η εφαρμογή ήταν διαθέσιμη μέσω του Android Market, και είχε μεταμφιεστεί σε ένα ακίνδυνο παιχνίδι “φιδάκι”,ωστόσο, συνέλεγε την τοποθεσία του θύματος. Αυτού του είδους η εφαρμογή εμπίπτει στην κατηγορία των κακόβουλων προγραμμάτων που ενσωματώνονται στον κώδικα της νόμιμης εφαρμογής. Σε αυτή τη περίπτωση , το παιχνίδι παίζεται πραγματικά με τον τρόπο που αναμένεται. Κατά τη διαδικασία της εγκατάστασης, ο επιτιθέμενος πρέπει να έχει φυσική πρόσβαση στη συσκευή, καθώς απαιτείται η εισαγωγή PIN για να αναγνωριστεί το θύμα, όταν αρχίσει ο ιός να μεταδίδει τα δεδομένα του.[18]

Κλοπή πόρων συσκευής (Hijacking device resources)

Η G Data στην έκθεσή της , που βασίζεται σε στοιχεία δεδομένων το 2013[19] για κακόβουλες εφαρμογές σε κινητές συσκευές, ισχυρίζεται ότι το 10 τοις εκατό των επιθέσεων σε έξυπνες συσκευές πραγματοποιείται με χρήση “κερκόπορτας” backdoor. Η χρήση κερκόπορτας έχει διάφορους σκοπούς για έναν εισβολέα, ο οποίος μπορεί να τη χρησιμοποιήσει για να αποκτήσει απομακρυσμένη πρόσβαση στη συσκευή. Αυτό επιτρέπει στον εισβολέα είτε να κλέψει πληροφορίες που είναι αποθηκευμένες στο τηλέφωνο, ή να πάρει τον έλεγχο των πόρων της συσκευής. Μια μολυσμένη συσκευή από μόνη της μπορεί να μην είναι χρήσιμη για τον εισβολέα, αλλά όταν αυτός καταφέρει να μολύνει επιτυχώς ένα πλήθος συσκευών, τότε οι διαθέσιμοι πόροι που θα έχει πρόσβαση ο επιτιθέμενος είναι σημαντικοί. Η λέξη για ένα τέτοιο παραβιασμένο δίκτυο υπολογιστών , είναι botnet, όπου κάθε μολυσμένη συσκευή είναι ένα ρομπότ λογισμικού.[20]

Τα botnets είναι ικανά να εκτελούν υπολογιστικές εργασίες κατά απαίτηση, αλλά μπορούν επίσης να χρησιμοποιηθούν σε «κατανεμημένη επίθεση άρνησης υπηρεσίας» (DDoS).

Τέτοιου είδους botnet και δούρειοι ίπποι έχουν διαφορετικό βαθμό λειτουργικότητας και πολυπλοκότητας. Χαρακτηριστικό παράδειγμα είναι το DroidDream, το οποίο κατάφερε το 2011 να εξαπλωθεί μέσω της επίσημης αγοράς , Google Play. Ο κακόβουλος κώδικας ενσωματώθηκε σε περισσότερες από 50 εφαρμογές. Μετά την πρώτη μόλυνση της συσκευής, ο κακόβουλος κώδικας ήταν ικανός να κλέψει προσωπικές πληροφορίες και να κατεβάσει άλλες εφαρμογές από το διαδίκτυο. Έτσι, το πρώτο κύμα της επίθεσης δεν επέφερε ζημία στο θύμα, ωστόσο, η δεύτερη εφαρμογή που κατέβαζε, έκανε τη συσκευή μέρος του δικτύου botnet.[21][22]

Η εμφάνιση των ψηφιακών νομισμάτων έχει προσφέρει νέους τρόπους για να έχουν κέρδος οι κακόβουλες εφαρμογές. BitCoin, LiteCoin και άλλα τέτοια νομίσματα είναι “mined” , εξορύσσονται, δηλαδή, από τη χρήση υπολογιστικής ισχύος. Όσο περισσότερη ενέργεια καταναλώνεις, τόσο υψηλότερη ανταμοιβή κερδίζεις. Είναι εύκολο να δούμε, πώς ένα botnet μπορεί να αξιοποιηθεί για αυξηθεί η ικανότητα εξόρυξης ψηφιακού νομίσματος . Ένα τέτοιο παράδειγμα είναι η εφαρμογή TuneInRadio Pro, η οποία χρησιμοποιήθηκε για να μολύνει έξυπνα τηλέφωνα και στη συνέχεια να εκμεταλλευτεί τους πόρους των μολυσμένων συσκευών για να πραγματοποιήσει εξόρυξη. Για να αποφευχθεί η καχυποψία από τον χρήστη, η εφαρμογή μεταμφιεζόταν σε μια άρτια εφαρμογή ραδιοφώνου και μόνο, όταν η συσκευή ήταν ανενεργή, ο κακόβουλος κώδικας χρησιμοποιούσε τους πόρους της CPU, για να εξορύξει Dogecoins. Ο χρήστης τότε θα παρατηρούσε μικρότερη διάρκεια ζωής της μπαταρίας ή αυξημένη θερμοκρασία της συσκευής λόγω χρήσης της CPU, αλλά, εκτός από αυτά, η συσκευή θα λειτουργεί φυσιολογικά, έτσι ώστε η παρατήρηση της μόλυνσης να είναι πιο δύσκολη. Επιπλέον, η διεργασία του ονομάζεται «Google Service» με σκοπό να παραπλανήσει τον χρήστη.

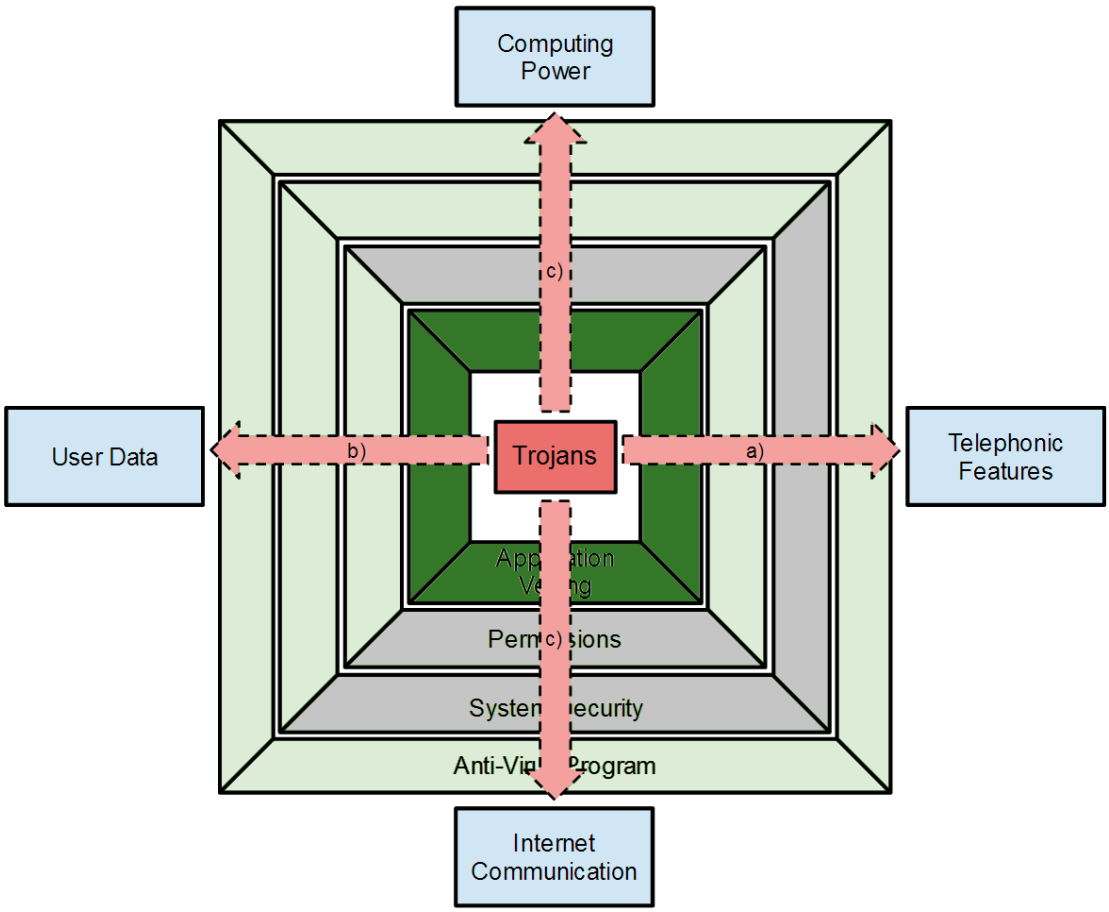
4.1.3 Επιθέσεις δούρειων ίππων και μοντέλο απειλής

Όπως αναφέραμε παραπάνω, οι επιθέσεις Trojan συμβαίνουν εκτός της επίσημης αγοράς. Αυτό δε σημαίνει, ωστόσο , ότι οι επιτιθέμενοι σταματούν να προσπαθούν. Σημαίνει, ότι αποτυγχάνουν κατά τη διεργασία application vetting. Αυτό αντικατοπτρίζεται στο μοντέλο απειλής, όπως φαίνεται στο παρακάτω σχήμα, όπου το application vetting έχει χαρακτηριστεί ως ιδιαίτερα αποτελεσματικό ενάντια σε όλες τις μορφές Trojan επιθέσεων.

Τα δικαιώματα έχουν χρώμα πράσινο για τις επιθέσεις fakeInstaller και spyware. Ο λόγος είναι, ότι ορισμένοι χρήστες μπορεί να απορρίψουν την εγκατάσταση της εφαρμογής που ζητάει δικαιώματα.

Η ασφάλεια συστήματος, ωστόσο, διασφαλίζει, ότι καμία εφαρμογή δε χρησιμοποιεί ολόκληρο το υλικό της συσκευής. Επίσης, αποτρέπεται η πρόσβαση μιας κακόβουλης εφαρμογής στα εσωτερικά δεδομένα άλλης εφαρμογής.

Η αποτελεσματικότητα των αντιικών προγραμμάτων εξαρτάται σε μεγάλο βαθμό από την εκδότρια εταιρεία και την βάση δεδομένων της. Εάν ένα κακόβουλο λογισμικό είναι γνωστό, το πρόγραμμα AV μπορεί να προειδοποιήσει τον χρήστη, πριν γίνει οποιαδήποτε επιβλαβής ενέργεια.



Εικόνα 6: Μοντέλο απειλής επιθέσεων Trojan[55]

4.2 Απειλές διαφημίσεων

Οι επιθέσεις μέσω διαφημίσεων είναι διαφορετικές από αυτές των δούρειων ίππων, εφόσον εκμεταλλεύονται αδυναμίες του συστήματος και όχι την άγνοια του χρήστη. Οι ευπάθειες στις οποίες στοχεύουν οι επιθέσεις διαφημίσεων σχετίζονται με το τον τρόπο που υλοποιούνται μέσα στην εφαρμογή (Βιβλιοθήκες διαφημίσεων) και στην επικοινωνία τους με το διακομιστή που παρέχει τις διαφημίσεις (Ad Network).

4.2.1 Παραβίαση δικαιωμάτων

Το πλαίσιο εργασίας του Android περιέχει ένα προκαθορισμένο σύνολο αδειών που χρησιμοποιείται ως ένα επίπεδο προστασίας για να περιορίσει την πρόσβαση των εφαρμογών στους πόρους του συστήματος. Οι άδειες αυτές χρησιμοποιούνται, μεταξύ άλλων, για την πρόσβαση στο διαδίκτυο, την κάμερα, το GPS, την ανάγνωση των SMS κ.α. Οι βιβλιοθήκες διαφημίσεων υλοποιούνται έτσι, ώστε να εκτελούνται συγχρόνως με την εφαρμογή που τις φιλοξενεί. Αυτό σημαίνει, ότι λειτουργούν μέσα στο ίδιο περιβάλλον μιας εικονικής μηχανής Dalvik, και η βιβλιοθήκη της διαφήμισης θα έχει το ίδιο UID με την εφαρμογή. Όταν μια εφαρμογή θέλει να χρησιμοποιήσει ένα συγκεκριμένο πόρο του συστήματος, το λειτουργικό

σύστημα ελέγχει, εάν η εφαρμογή έχει τα απαραίτητα δικαιώματα. Ο έλεγχος αυτός βασίζεται στο UID της εφαρμογής. Οι δραστηριότητες των διαφημιστικών βιβλιοθηκών και οι διαφημίσεις που φορτώνονται έχουν τα ίδια δικαιώματα με την εφαρμογή που τις φιλοξενεί. Η διαδικασία αυτή λειτουργεί και αμφίδρομα, καθώς η εφαρμογή που φιλοξενεί τη διαφήμιση θα έχει πρόσβαση στα δικαιώματα που παρέχονται από τις βιβλιοθήκες διαφημίσεων.

Ως αποτέλεσμα των κοινών δικαιωμάτων, πολλές βιβλιοθήκες διαφημίσεων, θα παρακολουθούν την εφαρμογή και θα προσπαθούν να ανακαλύψουν, ποιοι πόροι του συστήματος έχουν χορηγηθεί σε αυτές [24]. Κακόβουλες διαφημίσεις που περιλαμβάνονται σε εφαρμογές με πολλά δικαιώματα μπορεί να έχουν άσχημες συνέπειες για τον χρήστη. Τα πιθανά σενάριο είναι δύο:

Απόκτηση προσωπικών δεδομένων. Ένα δίκτυο διαφημίσεων μπορεί να συλλέξει την τοποθεσία του χρήστη (κάνοντας χρήση των αδειών ACCESS_FINE_LOCATION ή ACCESS_COARSE_LOCATION), ενώ στη συνέχεια αιτούνται (με χρήση της άδειας INTERNET) και αποθηκεύουν αυτή την πληροφορία στα συστήματά τους. Αυτή η πληροφορία μπορεί να χρησιμοποιηθεί για σταλεί στον χρήστη (στοχευμένη διαφήμιση).

Αποστολή spam. Ένα πιο επιθετικό σενάριο είναι, όταν τα δίκτυα διαφημίσεων αποκτούν πρόσβαση στις επαφές του χρήστη (κάνοντας χρήση της άδειας READ_CONTACTS) και στέλνουν μηνύματα spam. Αυτό μπορεί να γίνει είτε από τα συστήματα των διαφημιστικών δικτύων είτε μέσω της διεύθυνσης ηλεκτρονικού ταχυδρομείου ή αποστολής μηνυμάτων από τη συσκευή του χρήστη (πραγματοποιούνται με τη χρήση των αδειών INTERNET ή SEND_SMS)

Το πρώτο σενάριο αποτελεί εισβολή στην ιδιωτικότητα του χρήστη, ενώ χρησιμοποιούνται τεχνικές διαφημίσεων, ώστε να αυξηθούν τα έσοδα από αυτές. Το δεύτερο σενάριο, αποτελεί απειλή , επειδή η διαφήμιση αναζητά τα πιθανά δικαιώματα που έχουν εκχωρηθεί στην εφαρμογή, ενώ ,στη συνέχεια, γίνεται κατάχρηση των δικαιωμάτων αυτών για να πραγματοποιηθεί «επιθετικό» μαρκετινγκ.

Επίσης, αξίζει να σημειωθεί, ότι, όταν ο χρήστης κοιτάζει τις άδειες που απαιτούνται κατά την εγκατάσταση της εφαρμογής, δεν είναι σε θέση να προσδιορίσει ποια δικαιώματα είναι μόνο για την εφαρμογή και ποια χρησιμοποιούνται για τις διαφημίσεις [23]. Το γεγονός αυτό καθιστά αυτές τις απειλές δύσκολα ανιχνεύσιμες, ώστε να αποφευχθούν από τον χρήστη.

4.2.2 Ανάκτηση και φόρτωση δυναμικού κώδικα

Ορισμένα δίκτυα διαφήμισης είναι ικανά να ανακτούν και να κατεβάζουν κώδικα (συνήθως από το internet). Αυτή η πρακτική αποτελεί μεγάλη απειλή για δύο λόγους. [23] Αρχικά, η φόρτωση του κώδικα δυναμικά μπορεί να μην αναλυθεί , παρακάμπτοντας με αυτό τον τρόπο την στατική ανάλυση. Επιπλέον, ο κώδικας ο οποίος κατεβαίνει μπορεί να αλλάξει οποιαδήποτε στιγμή, γεγονός που καθιστά δύσκολο να προβλεφθεί η συμπεριφορά της εφαρμογής.

Η διαφημιστική βιβλιοθήκη Plankton κάνει χρήση της παραπάνω τεχνικής και φορτώνει δυναμικά τον κώδικα της. [24] Στην περίπτωση του Plankton , μόνο ένα μικρό μέρος της διαφημιστικής βιβλιοθήκης που περιλαμβάνεται στην κύρια εφαρμογή είναι κακόβουλο. Αυτό το κομμάτι κώδικα είναι μια υπηρεσία, η οποία επικοινωνεί απομακρυσμένα με τον διακομιστή στον οποίο στέλνει μια λίστα με τα διαθέσιμα δικαιώματα της εφαρμογής και το IMEI του χρήστη.[23] Ο απομακρυσμένος διακομιστής παρέχει στη διαφημιστική βιβλιοθήκη μια διαδικτυακή τοποθεσία (URL) για να κατεβάζει ένα αρχείο Java (.jar) που περιέχει κώδικα ειδικά για αυτή τη συσκευή. Αυτό το αρχείο περιέχει τον κακόβουλο κώδικα και φορτώνεται δυναμικά μέσα στην εφαρμογή χρησιμοποιώντας το αντικείμενο dalvik system DexClass Loader. Αυτό το αντικείμενο φορτώνει κλάσεις από αρχεία .jar και APK και μπορεί να χρησιμοποιηθεί για την εκτέλεση κώδικα που δεν είναι μέρος της εφαρμογής. Μόλις το αρχείο κατέβει, φορτώνεται στην

εφαρμογή και την μετατρέπει σε bot που ανταποκρίνεται σε απομακρυσμένες εντολές. Ένα μεγάλο πλήθος εφαρμογών που μολύνθηκαν από τη διαφημιστική βιβλιοθήκη αφαιρέθηκαν από το επίσημο κατάστημα της Google.[23]

4.2.3 Κακόβουλες βιβλιοθήκες διαφημίσεων

Μια άλλη μορφή επίθεσης που συνδέεται με την ενσωμάτωση της διαφήμισης στην εφαρμογή είναι ο τύπος της κακόβουλης βιβλιοθήκης. Αυτού του είδους οι διαφημιστικές βιβλιοθήκες, παρότι εμφανίζονται ως ένα νόμιμο δίκτυο, περιέχουν κακόβουλο κώδικα, τον οποίο ο προγραμματιστής περιλαμβάνει στην εφαρμογή του χωρίς να το γνωρίζει.

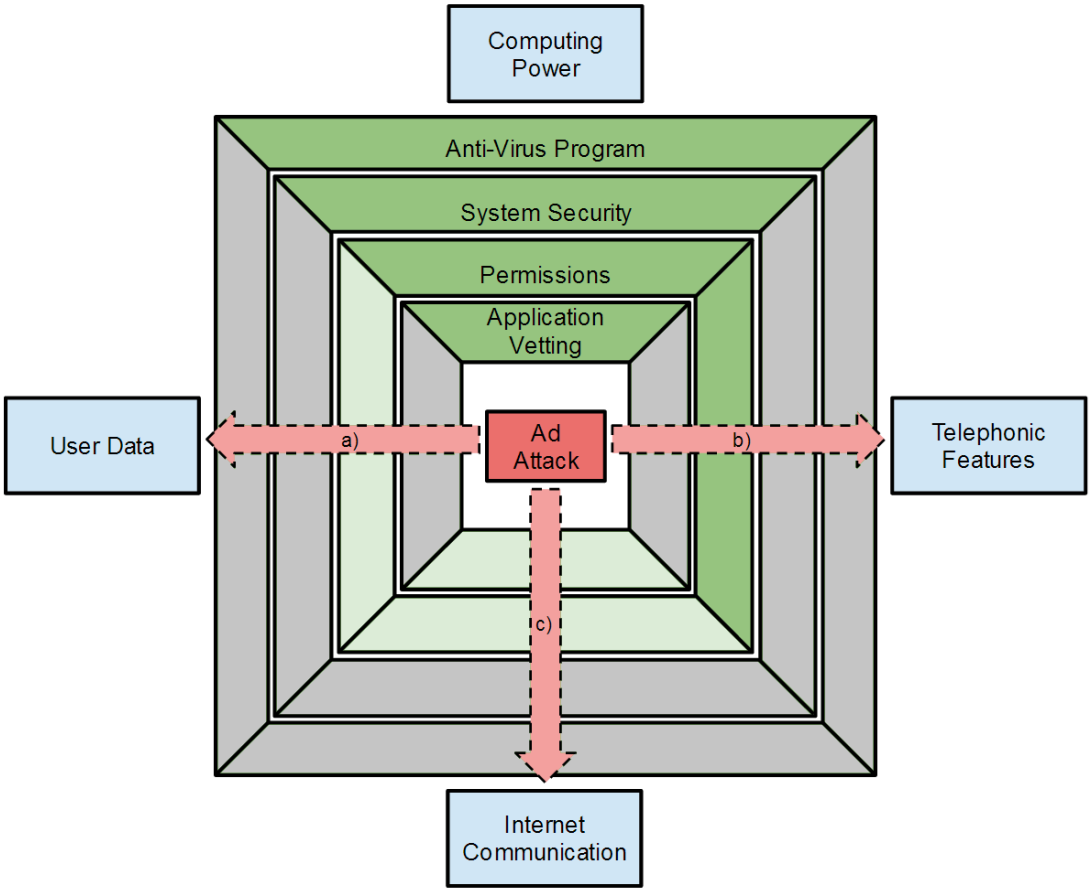
Το πρώτο δίκτυο που ανιχνεύθηκε με αυτή τη συμπεριφορά ήταν το BadNews τον Απρίλιο του 2013. Οι προγραμματιστές που επέλεξαν να συμπεριλάβουν το BadNews στις εφαρμογές ,είχαν άγνοια του κινδύνου στον οποίο έθεταν τους χρήστες. Μόλις μια μολυσμένη εφαρμογή εγκαθίστατο, τότε το BadNews επικοινωνούσε με διακομιστές C&C και τους ενημέρωνε για τις προσωπικές πληροφορίες του χρήστη (IMSI , IMEI). Η απάντηση που λάμβανε από τον διακομιστή ήταν να προβάλει στον χρήστη ψεύτικες ειδοποιήσεις. Μια τυπική ειδοποίηση θα μπορούσε να είναι, ότι η εφαρμογή χρειάζεται αναβάθμιση, και στη συνέχεια να παρέχει μια διαδικτυακή διεύθυνση, από την οποία η συσκευή θα λάμβανε κακόβουλο κώδικα. Η εφαρμογή BadNews κατάφερε να παρακάμψει την προστασία της Google (Google Bouncer), επειδή ο κακόβουλος κώδικας μόλυνε τη συσκευή μετά την εγκατάσταση της εφαρμογής. Το BadNews εμφανίστηκε σε 32 εφαρμογές σε τέσσερις διαφορετικούς λογαριασμούς προγραμματιστών και μόλυνε δύο με εννέα εκατομμύρια συσκευές.[25]

4.2.4 Διαφήμιση και μοντέλο απειλής

Όπως φαίνεται στη εικόνα 4.3, οι επιθέσεις διαφημίσεων στοχεύουν σε τρία διαφορετικά στοιχεία. Τα δεδομένα του χρήστη, τις λειτουργίες της συσκευής και στη διαδικτυακή επικοινωνία. Για να επιτευχθούν αυτοί οι στόχοι, ο επιτιθέμενος πρέπει να παρακάμψει διαφορετικά επίπεδα ασφάλειας:

Application Vetting. Οι επιθέσεις στις διαφημίσεις των εφαρμογών είναι δύσκολο να ανιχνευθούν χρησιμοποιώντας application vetting, διότι ο κακόβουλος κώδικας δεν είναι μέρος της εφαρμογής που φιλοξενεί τη διαφήμιση. Ωστόσο, οι κακόβουλες διαφημιστικές βιβλιοθήκες και οι απειλές από τη φόρτωση δυναμικού κώδικα μπορούν να εντοπιστούν μέσω του Application vetting.

Δικαιώματα. Τα δικαιώματα που ζητούν οι κακόβουλες διαφημίσεις εμφανίζονται ως νόμιμες εφαρμογές και στη συνέχεια κληρονομούν τα δικαιώματα της εφαρμογής που τους φιλοξενεί. Αυτό περιορίζει τις ενέργειες που μπορεί να εκτελέσουν. Οι λειτουργίες τηλεφώνου απαιτούν περισσότερα δικαιώματα (και πιο ασυνήθιστα) από τα δεδομένα του χρήστη και την πρόσβαση στο διαδίκτυο.



Εικόνα 7: Το μοντέλο απειλών διαφημίσεων που στοχεύουν στα δεδομένα του χρήστη, λειτουργίες τηλεφώνου και στην επικοινωνία με το διαδίκτυο. [55]

4.3 Επιθέσεις με χρήση WebView

Το webview είναι χαρακτηριστικό των λειτουργικών συστημάτων Android και iOS, τα οποία επιτρέπουν στις εφαρμογές να φορτώνουν και να εμφανίζουν το περιεχόμενο τους από το διαδίκτυο χρησιμοποιώντας το πρότυπο HTTP.[26] Επιτρέπουν στον χρήστη να αλληλεπιδρά με το περιεχόμενο των ιστοσελίδων, ενώ παράλληλα περιλαμβάνουν βασικές λειτουργίες του προγράμματος περιήγησης, όπως η πλοήγηση σελίδας, η σελιδοποίηση (page rendering) και η εκτέλεση Javascript.[26]. Με λίγα λόγια, ενεργούν με τον ίδιο τρόπο, όπως και οι πραγματικοί φυλλομετρητές (Firefox, Safari και Internet Explorer). Το WebView ενσωματώνεται μέσα στην εφαρμογή. Ωστόσο, υπάρχουν μερικές σημαντικές διαφορές , εφόσον οι φυλλομετρητές έχουν σχεδιαστεί να είναι ανεξάρτητοι από τις διαδικτυακές εφαρμογές, ενώ το webview βασίζεται στις διαδικτυακές εφαρμογές.

Ένα παράδειγμα μιας εφαρμογής που βασίζεται στο WebView, είναι το “Facebook Mobile” ,το οποίο έχει σχεδιαστεί με τέτοιο τρόπο, ώστε να αλληλεπιδρούν ευκολότερα οι χρήστες των κινητών συσκευών. Αυτός είναι ο λόγος που πολλοί προτιμούν αυτό σε σχέση με την κλασική διαδικτυακή διεπαφή του FaceBook.[27]

Η αλληλεπίδραση του χρήστη και η παραμετροποίηση γίνεται μέσω της χρήσης APIs που παρέχεται από το WebView. Ένα άλλο χαρακτηριστικό των APIs, το οποίο είναι πιο σημαντικό, είναι, ότι επιτρέπει την αμφίδρομη αλληλεπίδραση μεταξύ της εφαρμογής και της ισχοσελίδας. Η εφαρμογή μπορεί να ανακαλέσει ή να εισαγάγει κώδικα Javascript στην ιστοσελίδα, ενώ μπορεί να την παρακολουθεί και να ανταποκρίνεται στα γεγονότα (events) της. Από την άλλη πλευρά, αυτού του είδους η επικοινωνία επιτρέπει στην εφαρμογή να καταχωρεί διαφορετικές διεπαφές, επιτρέποντας στην ιστοσελίδα τη χρήση Javascript έτσι, ώστε να αλληλεπιδρά με την εφαρμογή και τη συσκευή. Αυτοί οι δυο τρόποι αλληλεπίδρασης καθιστούν το WebView έναν ισχυρό φυλλομετρήτή σε σχέση με τους παραδοσιακούς.

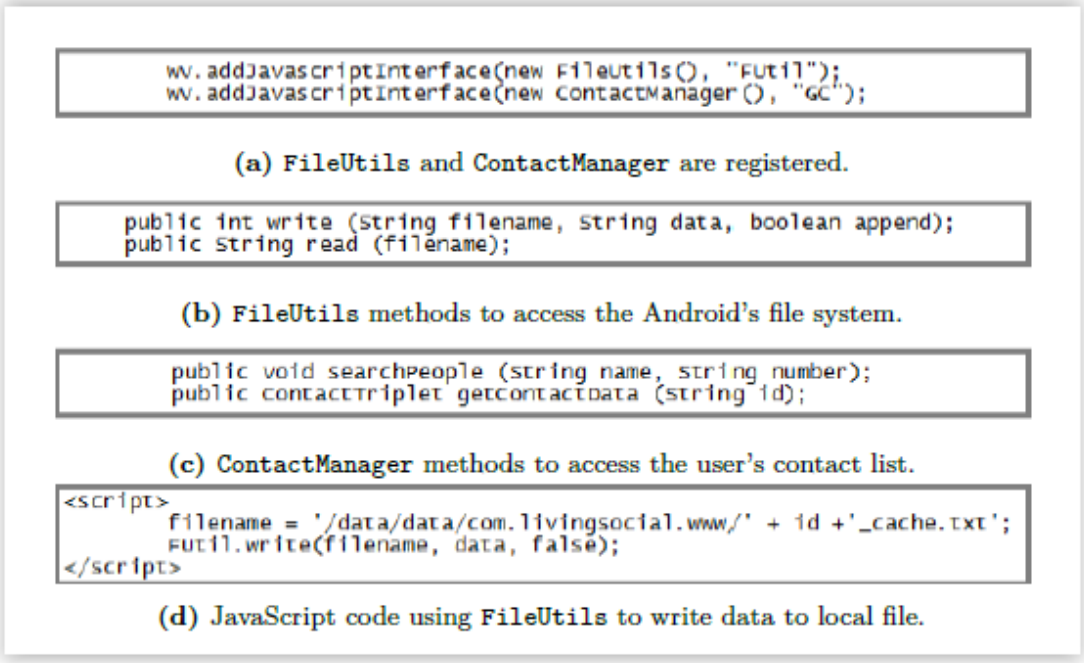
Με την εισαγωγή του WebView, το τοπίο ασφάλειας στο διαδίκτυο έχει αλλάξει. Ένας παραδοσιακός φυλλομετρητής έχει αναπτυχθεί από γνωστές εταιρείες (π.χ. Google Chrome, Internet Explorer) που είναι αξιόπιστες. Ωστόσο, με το WebView αυτό δεν ισχύει. Πολλοί άγνωστοι φυλλομετρητές που έχουν παρουσιαστεί είναι αμφιβόλου αξιοπιστίας. Παραδοσιακές διαδικτυακές εφαρμογές βασίζονται στον φυλλομετρητή, για να εξασφαλίσουν την ασφάλεια των “cookies”, τον κώδικα Javascript και τα αιτήματα HTTP , τα οποία δεν αποτελούν πρόβλημα στους γνωστούς φυλλομετρητές. Οι εταιρείες που αναπτύσσουν τους φυλλλομετρητές, ξοδεύουν πολύ χρόνο και πόρους, για να τους κρατήσουν αξιόπιστους και ασφαλείς. Η εισαγωγή των αναξιόπιστων WebView δημιουργεί προβληματισμό για την ασφάλεια των χρηστών της.[27]

Ένας άλλος μηχανισμός ασφάλειας που είναι ενσωματωμένος στους παραδοσιακούς φυλλομετρητές, είναι το sandboxing, ο οποίος δεν επιτρέπει την επικοινωνία εκτός του προγράμματος περιηγητή. Για να βελτιωθεί η αλληλεπίδραση μεταξύ εφαρμογής και ιστοσελίδας, το WebView έχει δημιουργήσει τρύπες στην τεχνική sandboxing, με αποτέλεσμα ένας εν δυνάμει επιτιθέμενος να είναι σε θέση να τις εκμεταλλευτεί.[27]

4.3.1 Αλληλεπίδραση Java και Javascript στο Android

Javascript σε Java

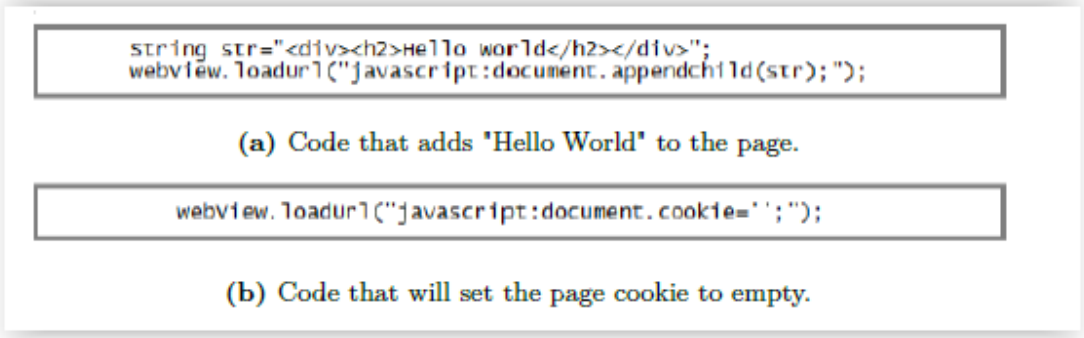
Το Webview παρέχει λειτουργίες που επιτρέπουν στον Javascript κώδικα να καλεί εφαρμογές του Android που είναι γραμμένες σε Java κώδικα. Με τη χρήση ενός API που ονομάζεται *addJavascriptInterface* , οι εφαρμογές Android μπορούν να καταχωρήσουν αντικείμενα Java στο WebView. Στη συνέχεια, όλες οι δημόσιες μέθοδοι (σε αυτή την περίπτωση τα Java αντικείμενα) μπορούν να χρησιμοποιηθούν από τον Javascript κώδικα μέσα από το WebView. Ο απαραίτητος κώδικας φαίνεται στην παρακάτω εικόνα. Η δυνατότητα να καταχωρείται μια διεπαφή στο WebView έχει ως αποτέλεσμα να παραβιάζεται το μοντέλο sandboxing .[27]



Εικόνα 8: Χρήση αντικειμένων Java για πρόσβαση σε πόρους του λειτουργικού έξω από το WebView. [55]

Java σε Javascript

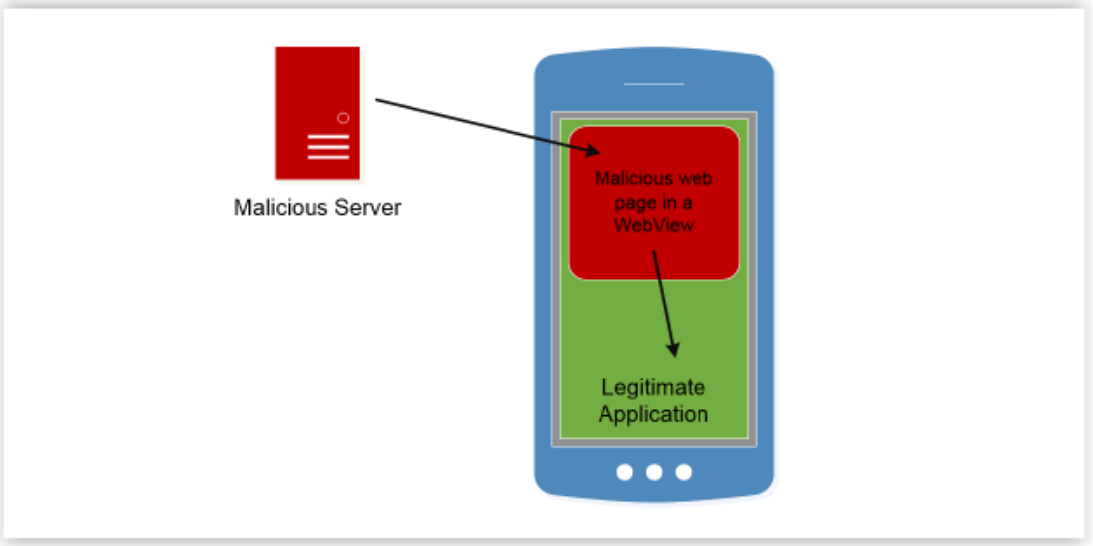
Με τη χρήση του WebView επιτυγχάνεται η μέθοδος *loadURL* (από την εφαρμογή του Android), η οποία φορτώνει κώδικα javscript (εκτελείται από την ιστοσελίδα στο εσωτερικό του WebvView).[27] Εάν η διεύθυνση URL από την εφαρμογή του Android ξεκινά με «javascript : » ακολουθούμενη από τον κώδικα javascript, το API θα εκτελέσει αυτό τον κώδικα της ιστοσελίδας μέσα στο webview[27]. Παρακάτω φαίνεται ο κώδικας που δίνει αυτή τη δυνατότητα.[27]



Εικόνα 9: Κώδικας Javascript που θα εκτελεστεί μέσα στο WebView. [55]

4.3.2 Επιθέσεις από ιστοσελίδες

Κακόβουλες ιστοσελίδες μπορούν να πραγματοποιούν επιθέσεις μέσω της χρήσης του WebView. Για να επιτευχθεί αυτό, ο επιτιθέμενος πρέπει να ξεγελάσει το θύμα, ωστε να φορτώσει τη δική του ιστοσελίδα μέσα την εφαρμογή του Webview. [27] Η επίθεση μπορεί να επιτευχθεί με διάφορα μέσα, όπως διαφημίσεις και ηλεκτρονική αλληλογραφία.Η σχέση μεταξύ του διακομιστή, του webView και της εφαρμογής φαίνεται παρακάτω:



Εικόνα 10: Επίθεση από διακομιστή με χρήση webview για να αποκτήσει πρόσβαση στη συσκευή. [55]

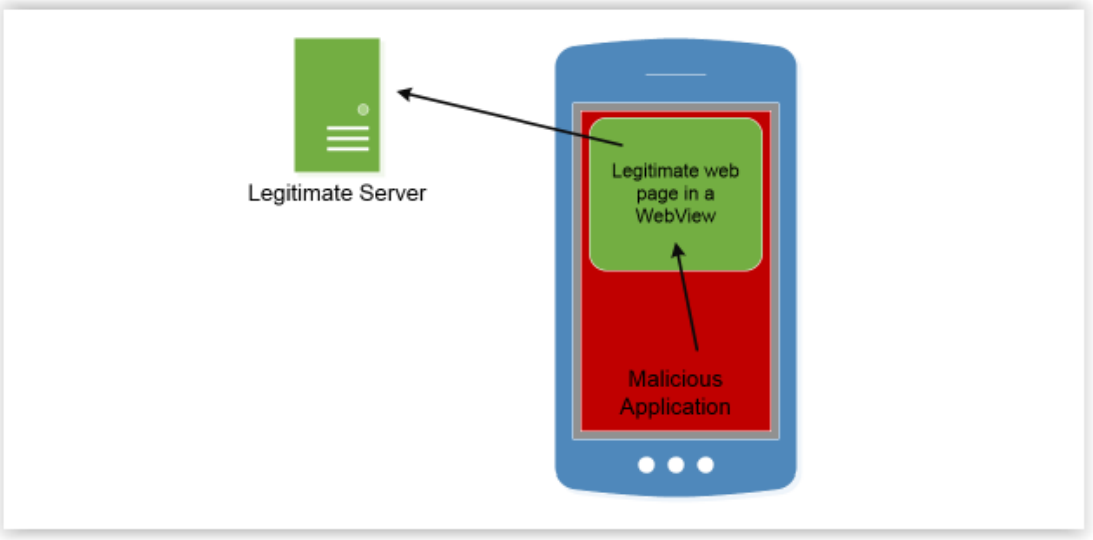
Επιθέσεις μέσω παραβίασης του sandboxing

Για την προστασία έναντι του κινδύνου λειτουργίας μη αξιόπιστων Javascript προγραμμάτων, οι περιηγητές υλοποιούν ένα χαρακτηριστικό ασφάλειας, το sandboxing. Το sandboxing είναι σε θέση να επιτύχει δυο στόχους. Πρώτον, το διαχωρισμό μεταξύ ιστοσελίδων και συστήματος, και δεύτερον, τον διαχωρισμό μεταξύ των ιστοσελίδων. Η παραβίαση του sandboxing επιτρέπει στον επιτιθέμενο να εκμεταλλεύεται , όπως αναφέρθηκε προηγουμένως, μέσω της διεπαφής addJavaInterface τον κώδικα της Java εφαρμογής. Ένα παράδειγμα χρήσης της διεπαφής addJavaInterface φαίνεται στην εικόνα 11. Το αποτέλεσμα είναι, ότι ο επιτιθέμενος μπορεί να αποκτήσει πρόσβαση στους πόρους του συστήματος, όπως κάμερα , τοποθεσία GPS, επαφές και άλλα αρχεία. Επίσης, παραβιάζεται ο διαχωρισμός του sandbox μεταξύ των ιστοσελίδων και του λειτουργικού συστήματος. Επιπλέον , όταν μια διεπαφή έχει καταχωρηθεί στο WebView (μέσω της χρήσης addJavaInterface), τότε το αποτέλεσμα είναι, ότι όλες οι ιστοσελίδες που φορτώνονται μέσω του συγκεκριμένου WebView έχουν πρόσβαση στου πόρους και τα δεδομένα του συστήματος. Αυτό δίνει τη δυνατότητα σε δύο ιστοσελίδες διαφορετικής προέλευσης (same origin policy) να επηρεάζει η μια την άλλη. [27]

4.3.3 Επιθέσεις από την εφαρμογή

Ένας επιτιθέμενος μπορεί επίσης να χρησιμοποιήσει το WebView για να επιτεθεί σε ιστοσελίδες. Μετά την εγκατάσταση της κακόβουλης εφαρμογής από τον χρήστη, είναι δυνατό να φορτωθεί η ιστοσελίδα μέσα στο WebView. Μέχρι αυτό το σημείο, καμία ζημιά δε γίνεται ,καθώς πολλές νόμιμες εφαρμογές χρησιμοποιούν αυτή τη δυνατότητα. Η σχέση μεταξύ

διακομιστή, WebView και εφαρμογής φαίνεται παρακάτω. Υπάρχουν πολλοί τρόποι για να εκτελεστεί αυτού του είδους επίθεση. Οι βασικότερες κατηγορίες είναι δύο : *Javascript Injection* και *Event sniffing and hijacking* [27]



Εικόνα 11: Επίθεση μέσω της Android Εφαρμογής. [55]

Javascript Injection

Το injection του Javascript κώδικα μπορεί να γίνει με τη χρήση του Webview έτσι, ώστε να εκτελεστεί ο κώδικας javascript στην ιστοσελίδα - στόχος. Μια εφαρμογή μπορεί να κάνει χρήση της μεθόδου loadURL() του WebView για να επιτύχει το injection του Javascript κώδικα στην ιστοσελίδα. Ο κώδικας javascript εισάγεται από το webview και η μέθοδος loadURL() τον εκτελεί στην ιστοσελίδα.Ο κώδικας αυτός έχει τα ίδια δικαιώματα με τον υπόλοιπο κώδικα javascript. Αυτό σημαίνει, ότι ο κώδικας που έχει εισαχθεί μπορεί και διαχειρίζεται τα cookies και την ίδια τη σελίδα. Στην παρακάτω εικόνα φαίνονται οι γραμμές του κώδικα που απαιτείται για να πραγματοποιηθεί αυτού του είδους η επίθεση.[27]

```
String js = "javascript: var newscript
     = document.createElement(\"script\");";
js += "newscript.src=\"http://www.attack.com/malicious.js\";";
js += "document.body.appendChild(newscript);";
mWebView.loadUrl(js);
```

Εικόνα 12: Javascript Injection. [55]

Event sniffing και hijacking

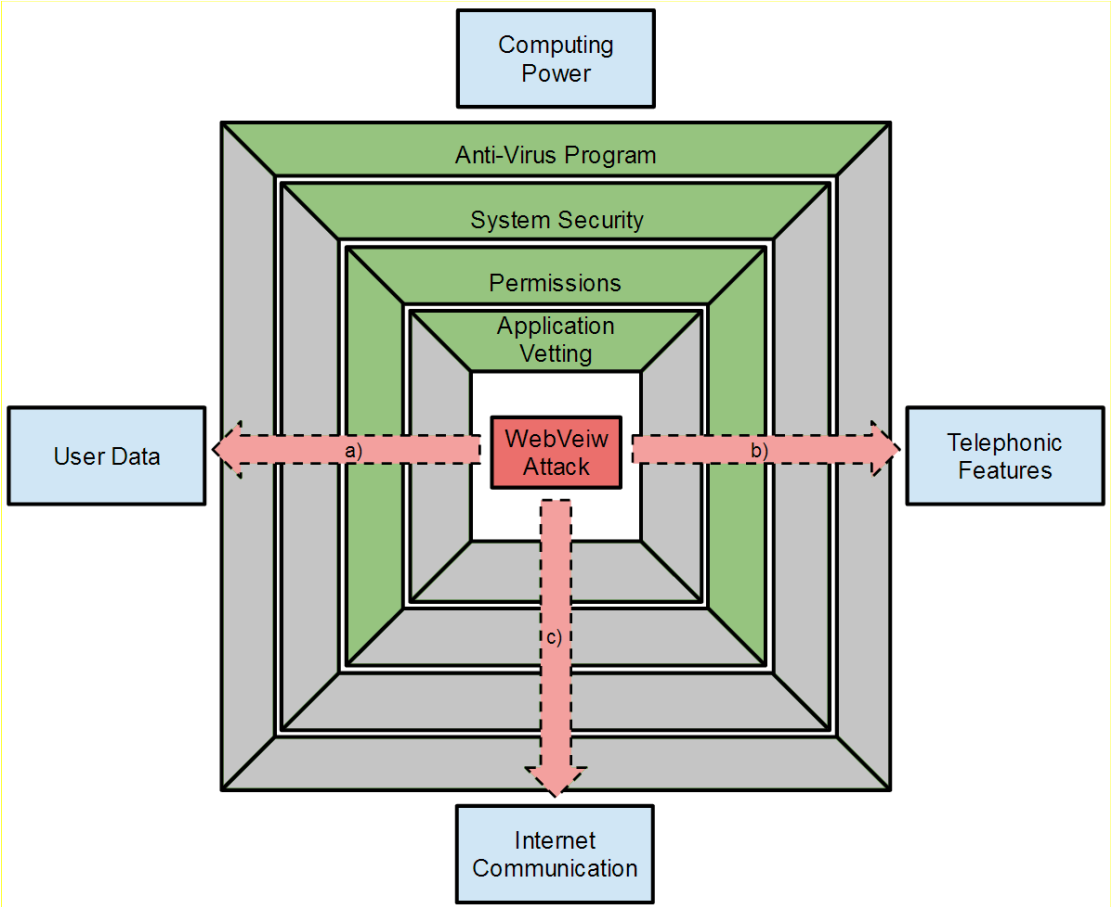
Η επίθεση αυτού του είδους εκμεταλλεύεται τα APIs που χρησιμοποιούνται για την αλληλεπίδραση μεταξύ Webview και ιστοσελίδας. Ο εισβολέας παρακολουθεί αυτά τα APIs και παρεμβαίνει έτσι, ώστε να πραγματοποιήσει την επίθεση εξωτερικά από το WebView. Η κλάση WebViewClient έχει ένα πλήθος μεθόδων, τις οποίες μπορεί να χρησιμοποιήσει μια εφαρμογή για να καταχωρήσει έναν event handler στο WebView. Εάν ο χρήστης πραγματοποιήσει μια ενέργεια, τότε δημιουργείται ένα συμβάν και ενημερώνεται η εφαρμογή.[27]

Event sniffing. Όταν φορτώνεται μια σελίδα, εφαρμόζεται η μέθοδος *doUpdateVisitedHistory*. Η μέθοδος αυτή χρησιμοποιείται, για να ενημερώσει την εφαρμογή, ώστε να ανανεώσει τη βάση δεδομένων της με τους διαδικτυακούς συνδέσμους που είχε επισκεφθεί. Αυτή η ενεργοποίηση παρέχει στην εφαρμογή τη λίστα με τους διαδικτυακούς συνδέσμους που έχει επισκεφθεί. Μια πιο σοβαρή περίπτωση είναι, όταν η εφαρμογή είναι σε θέση να παρακολουθήσει και να αλλάξει το συμβάν που ενεργοποιείται.

Event hijacking.Στην περίπτωση αυτή χρησιμοποιείται η μέθοδος *shouldOverrideUrlLoading* και ενεργοποιείται, όταν μια νέα διεύθυνση πρόκειται να φορτωθεί. Δίνει στην εφαρμογή τη δυνατότητα να παρέμβει και να τροποποιήσει το URL. Η εφαρμογή τότε μπορεί “στείλει” τον χρήστη σε άλλη ιστοσελίδα από αυτή που περιμένει.

4.3.4 Επίθεση WebView και μοντέλο απειλής

Οι επιθέσεις μέσω webview είναι πολύ αποτελεσματικές, ενώ υπάρχουν πολύ λίγα μέτρα προστασίας. Τα δικαιώματα είναι λιγότερο αποτελεσματικά. Σε μια επίθεση από ιστοσελίδα, η εφαρμογή καθορίζει τα δικαιώματα και η επίθεση περιορίζεται από τα διαθέσιμα δικαιώματα.



Εικόνα 13: Μοντέλο απειλής WebView. [55]

5 Μελλοντικές λύσεις.

Με τον συνεχώς αυξανόμενο αριθμό περιπτώσεων κακόβουλων εφαρμογών κρίνεται απαραίτητη η βελτίωση της ασφάλειας στις έξυπνες συσκευές. Πιθανές λύσεις παρουσιάζονται στις επόμενες ενότητες του κεφαλαίου.

Οι δυο πρώτες λύσεις που προτείνονται βασίζονται στο υλικό. Ωστόσο , λόγω των διαφορετικών κατασκευαστών κινητών τηλεφώνων, απαιτείται μια εύκολη λύση λογισμικού που να μπορεί να ενσωματωθεί στο λειτουργικό σύστημα. Οι διαφημίσεις και ο τρόπος με τον οποίο η αρχιτεκτονική του λειτουργικού συστήματος τις διαχειρίζεται, αποτελούν σημεία που μπορεί να εκμεταλλευθεί ο κακόβουλος για να πραγματοποιήσει επιθέσεις. Παρακάτω παρουσιάζονται τρεις ξεχωριστοί τρόποι για να διευθετηθεί αυτό το πρόβλημα. Τέλος, προτείνεται μια διαφορετική προσέγγιση της τεχνολογίας WebView, που χρησιμοποιείται σε πολλές δημοφιλείς εφαρμογές.[13]

5.1 Λύσεις Hardware

Όποιος επιθυμεί, μπορεί να κατασκευάσει έξυπνο τηλέφωνο με λειτουργικό σύστημα Android. Αξίζει να σημειωθεί, ότι η κατασκευή των μερών του τηλεφώνου από διαφορετικούς κατασκευαστές επηρεάζει την τελική ασφάλεια του τηλεφώνου. Ενώ η Apple ελέγχει όλη την αλυσίδα παραγωγής , την αξία του τηλεφώνου και των εφαρμογών που θα εγκατασταθούν σε αυτό, δε μπορεί να εφαρμοστεί το ίδιο στα τηλέφωνα Android. Αυτό σημαίνει, ότι το λειτουργικό σύστημα Android δε μπορεί να προσαρμοστεί και να βελτιστοποιηθεί για κάποιο συγκεκριμένο υλικό. Το γεγονός αυτό καθιστά την ασφάλεια της συσκευής μέσω του υλικού δύσκολη , καθώς αυτό ποικίλλει από συσκευή σε συσκευή.

Μια λύση είναι η προσθήκη υλικού στο τελικό προϊόν και παρουσιάζεται ως η πρώτη από τις δύο λύσεις. Ενώ, η δεύτερη λύση εκμεταλλεύεται το γεγονός, ότι ένα συγκεκριμένο κομμάτι υλικού από έναν συγκεκριμένο κατασκευαστή χρησιμοποιείται σχεδόν σε όλες τις έξυπνες συσκευές.

Παρακάτω αναφέρονται μερικά πλεονεκτήματα και μειονεκτήματα της παρούσας κατάστασης, όσον αφορά στην ασφάλεια των έξυπνων τηλεφώνων με βάση το υλικό.

Πλεονεκτήματα

- Κάθε κατασκευαστής μπορεί να χρησιμοποιήσει ελεύθερα οποιοδήποτε υλικό τον διευκολύνει μέσα από μια μεγάλη ποικιλία προϊόντων που διαφέρουν σε τιμή και ποιότητα.

Μειονεκτήματα

- Δεν υπάρχει περιοχή του υλικού να εκτελούνται λειτουργίες αξιόπιστα και με ασφάλεια.
- Οι λύσεις υλικού που υπάρχουν δεν διευθετούν τον διαχωρισμό των λειτουργιών της συσκευής σε ξεχωριστά επίπεδα.
- Το υλικό και το λογισμικό μιας έξυπνης συσκευής σχεδιάζεται και παράγεται από διαφορετικούς ανθρώπους, γεγονός που καθιστά δύσκολη την αξιοποίηση του υλικού από το λειτουργικό σύστημα.
- Το λογισμικό βασίζεται σε λύσεις αντιϊκών AV (anti virus) που μπορούν να εξαντλήσουν τους πόρους του συστήματος, ακόμα και αν οι εταιρείες AV

προσπαθούν να ελαχιστοποιήσουν τον φόρτο του συστήματος που προκαλείται από τις λύσεις τους[28].

5.1.1 Προσθήκη Hardware

Η εταιρεία Cupp Computing έχει αναπτύξει μια λύση, η οποία βασίζεται σε Micro SD κάρτα[29]. Σκοπός αυτής της λύσης είναι η παροχή λειτουργιών anti-malware, τείχους προστασίας, φίλτρου DNS και ανίχνευση εισβολών (IDS/IPS). Επιπλέον, η νέα αυτή τεχνολογία εισάγει το διαχωρισμό της συσκευής σε δυο περιοχές, μια ιδιωτική περιοχή και μια επαγγελματική[29].

Η λύση της Cupp βασίζεται στο πακέτο της κάρτα Micro SD με 8 Gigabyte αποθηκευτικού χώρου flash, 512 Megabyte μνήμης τυχαίας προσπέλασης , και έναν επεξεργαστή του ARM Cortex.[29]

Χαρακτηριστικά

Βελτίωση επίδοσης της συσκευής. Τα παραδοσιακά λειτουργικά συστήματα χρησιμοποιούν για την ασφάλειά τους αντιικά προγράμματα, τα οποία λειτουργούν μέσα στο OS με αποτέλεσμα να δεσμεύουν πόρους από το σύστημα. Αντιθέτως, με τη χρήση αυτής της τεχνολογίας παρέχεται ξεχωριστός επεξεργαστής και μνήμη για την ασφάλεια της συσκευής.

Non-invasive Bring your own device (BYOD). Υψηλού επιπέδου ασφάλεια παρέχεται με την εισαγωγή της κάρτας Micro SD της Cupp. Με την αφαίρεση της κάρτας από τη συσκευή , θα επιστρέφει στην αρχική κατάσταση. Αυτό έχει ως αποτέλεσμα τη δημιουργία δυο διαφορετικών σχημάτων ασφάλειας, χωρίς να επηρεάζεται η συσκευή του χρήστη. Συν τοις άλλοις, πρόκειται για ένα χρήσιμο χαρακτηριστικό για επαγγελματικά και εταιρικά σενάρια.[29]

Διαχείριση των κλειδιών με βάση το Hardware. Τα πρωτόκολλα κωδικοποίησης μπορούν να αποθηκευτούν στη Micro SD κάρτα, παρέχοντας μια ξεχωριστή και πιο ασφαλή τοποθεσία. Αυτό θα εμποδίσει το exploit της εφαρμογής και θα περιορίσει την πρόσβαση στα κλειδιά κωδικοποίησης του Android συστήματος.

Παρακολούθηση Δικτύου. Η Cupp για να παρέχει IDS/IPS χρησιμοποιεί μια έκδοση του Snort για την πλατφόρμα ARM. Το Snort είναι ένα δικτυακό εργαλείο ικανό να αναλύει τη δικτυακή κίνηση σε πραγματικό χρόνο και μπορεί να εκτελείται σε τρεις διαφορετικές λειτουργίες:[4.19]

- Sniffer mode , διαβάζει τα πακέτα του δικτύου.
- Packet Logger mode, καταγράφει τα πακέτα δικτύου.
- Network Intrusion Detection System (NIDS) mode, πραγματοποιεί ανίχνευση και ανάλυση της δικτυακής κίνησης.

Το NIDS λειτουργεί μαζί με ένα σύνολο καθορισμένων κανόνων. Οι κανόνες αυτοί εφαρμόζονται στα πακέτα που ρέουν μέσω του προσαρμογέα δικτύου, ενώ στη συνέχεια επιλέγονται οι ενέργειες που πρέπει να γίνουν. Οι κανόνες του Snort είναι ένας τρόπος για

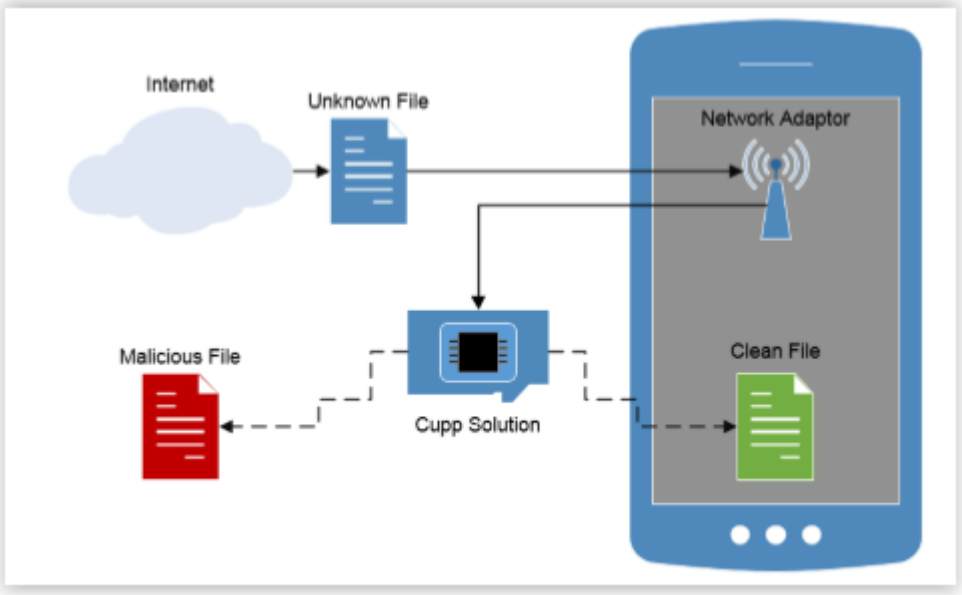
την εκτέλεση ανίχνευσης, ενώ ανιχνεύονται ευπάθειες, χωρίς τη χρήση υπογραφών (τις χρησιμοποιούν συνήθως τα αντιικά). Η ανίχνευση που βασίζεται στις υπογραφές ελέγχει γνωστά exploits. Με την εκτέλεση ανίχνευσης ευπάθειας, το Snort είναι ικανό να ανιχνεύει άγνωστα exploits. Αυτό είναι ένα τεράστιο πλεονέκτημα σε σχέση με το σχήμα *signature-based*, επειδή μπορεί να χρησιμοποιηθεί για την ανίχνευση επιθέσεων *zero-day* [31]. Ένα σύνολο κανόνων παρέχονται ελεύθερα από την κοινότητα του Snort και μπορούν να βρεθούν στην ιστοσελίδα τους. Ένας βασικός κανόνας φαίνεται στην εικόνα 14.

Σάρωση εφαρμογής. Πολλές από τις πιο ισχυρές τεχνικές για την ανίχνευση malware δεν είναι δυνατό να πραγματοποιηθούν σε συσκευή Android, λόγω των περιορισμών που τίθενται από το λειτουργικό σύστημα. Το μοντέλο ασφαλείας που βασίζονται τα δικαιώματα απαγορεύει την αλληλεπίδραση μεταξύ των εφαρμογών, γεγονός που περιορίζει τα προγράμματα που παρέχουν ασφάλεια να επιτύχουν τον στόχο τους. Στην πραγματικότητα, πρόκειται για άλλη μια εφαρμογή, η οποία πρέπει να λειτουργεί με τους ίδιους περιορισμούς που ισχύουν για όλες τις εφαρμογές. Το αποτέλεσμα είναι, ότι τα προγράμματα ασφαλείας που βασίζονται στο software δε μπορούν να παρέχουν sandboxing ή να κάνουν ανάλυση τον πηγαίο κώδικα της εφαρμογής.[29]

```
alert tcp !192.168.1.0/24 any -> 192.168.1.0/24 111 \
    (content:"|00 01 86 a5|"; msg:"external mountd access";)
```

Εικόνα 14: Μοντέλο απειλής WebView. [55]

Έχοντας μια λύση ασφαλείας ανεξάρτητη από τη συσκευή, η Cupp έχει τη δυνατότητα να θέσει σε καραντίνα μια εφαρμογή ή να αρνηθεί την πρόσβασή της στη συσκευή.[29] Επιπλέον, είναι δυνατές πιο πολύπλοκες αναλύσεις για την ασφάλεια των εφαρμογών, όπως η σάρωση του APK και ο έλεγχος του κώδικα. Η εικόνα 15 παρέχει μια βασική εικόνα παρακολούθησης της εφαρμογής και εξέτασης των αρχείων.



Εικόνα 15: Ένα άγνωστο εισερχόμενο αρχείο παρακολουθείται και ελέγχεται. [55]

Πλεονεκτήματα και μειονεκτήματα

Η λύση που παρουσιάζει η Cupp ανταποκρίνεται καλά στα μειονεκτήματα του υφιστάμενου συστήματος:

- Μια ξεχωριστή λύση ασφάλειας θα εισαγάγει μια νέα περιοχή που θα αφορά λειτουργίες της ασφάλειας.
- Με τη λύση που παρέχει η Cupp, είναι δυνατό να έχουμε μια λύση χωρίς επέμβαση στο υφιστάμενο λειτουργικό σύστημα.
- Αυτή λύση που παρέχεται είναι ανεξάρτητη από τον κατασκευαστή συσκευών.
- Με την παροχή ανεξάρτητης κάρτας Micro SD, χρησιμοποιείται ξεχωριστός επεξεργαστής και μνήμη , με αποτέλεσμα ο φόρτος του συστήματος να μειώνεται.

Επιπλέον, εισάγονται πολλά νέα χαρακτηριστικά. Τα πιο σημαντικά από αυτά είναι η ανάλυση της εφαρμογής που πραγματοποιείται εκτός της συσκευής που τη φιλοξενεί, η παρακολούθηση του δικτύου και το μειωμένο φορτίο του συστήματος. Επιπλέον, αυτή η λύση δεν έχει καμία προϋπόθεση και μπορεί να εφαρμοστεί με οποιαδήποτε συσκευή.

Με τον διαχωρισμό της ασφάλειας από το λειτουργικό σύστημα της συσκευής , οι προγραμματιστές θα μπορούν να χρησιμοποιούν μια αξιόπιστη λύση για την ασφάλεια της εφαρμογής τους. Με την προσθήκη επιπλέον hardware θα μπορούν να υποστηρίζονται πολλαπλές πλατφόρμες και κατασκευαστές συσκευών, εφόσον η λύση που προτείνεται είναι ανεξάρτητη από την ίδια τη συσκευή.

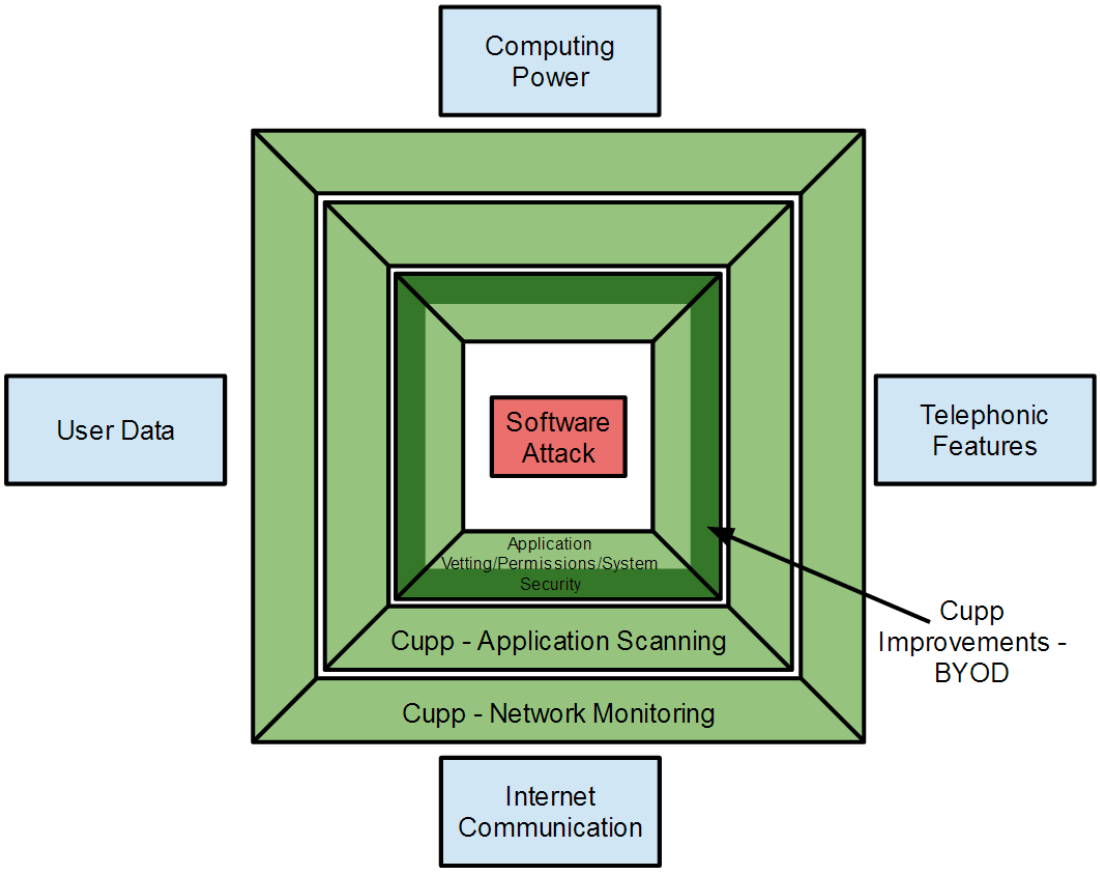
Το πιο σημαντικό μειονέκτημα της λύσης που προτείνεται είναι η πολυπλοκότητα και οι προκλήσεις που θα παρουσιαστούν, όσον αφορά στην εμπειρία του χρήστη. Επιπλέον , προβάλλουν μερικοί προβληματισμοί για τη λύση αυτή, όπως τι θα συμβεί, εάν η κάρτα Micro SD που περιέχει τον μηχανισμό ασφάλειας κλαπεί ή χαθεί. Σε αυτή την περίπτωση μπορεί ο επιτιθέμενος να αποκτήσει πρόσβαση στα ευαίσθητα δεδομένα της κάρτας. Για τους λόγους αυτούς, ένα νέο σενάριο για τον έλεγχο της πρόσβασης στην κάρτα πρέπει να εισαχθεί, το οποίο, ωστόσο, θα επιφέρει πολυπλοκότητα για τον τελικό χρήστη.

5.1.2 Νέος μηχανισμός στο υφιστάμενο Hardware

ARM's TrustZone technology

Η ARM είναι μια Βρετανική εταιρεία περισσότερο γνωστή για τον σχεδιασμό επεξεργαστών. Οι επεξεργαστές τους έχουν γίνει η επιλογή πολλών κατασκευαστών έξυπνων συσκευών, με μερίδιο της αγοράς άνω του 95% [32]. Η ARM αναπτύσσει την αρχιτεκτονική και το σετ εντολών,

αλλά στην πραγματικότητα δεν κατασκευάζει επεξεργαστές. Αντ' αυτού παραχωρεί τις άδειες του σχεδιασμού και το σετ εντολών σε τρίτες εταιρείες. Εταιρείες όπως η Apple, Samsung,



Εικόνα 16: Μοντέλο απειλής Cupp. [55]

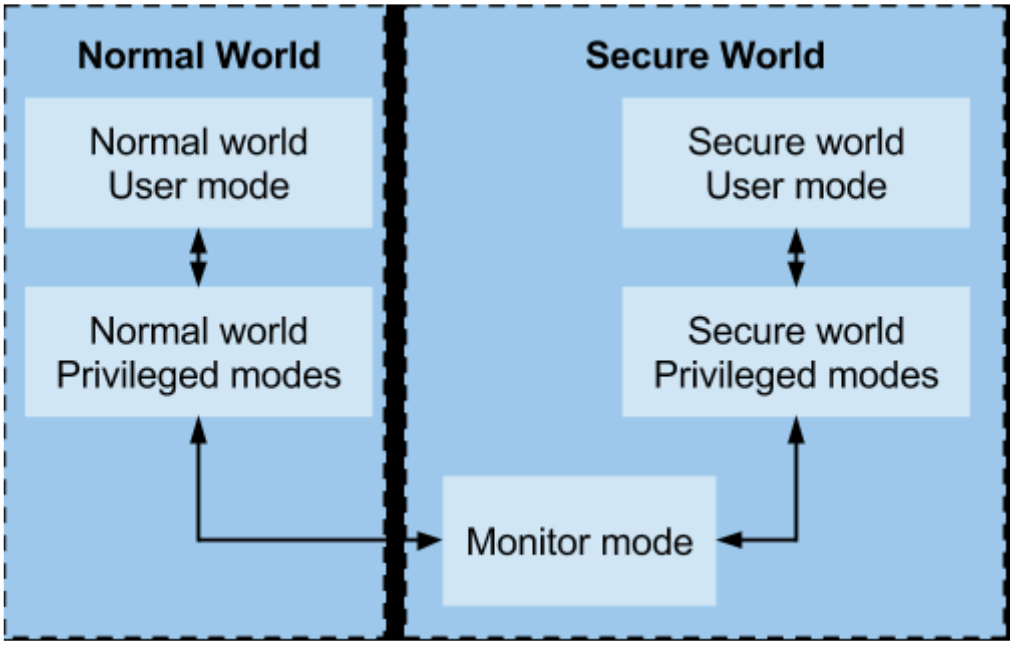
Nintendo, Nvidia και Texas Instrument είναι όλοι δικαιούχοι των σχεδίων της ARM και χρησιμοποιούν τη λογική της στα δικά τους SoCs.

Η τεχνολογία ARM TrustZone είναι άλλο ένα από τα σχέδια της ARM, αλλά δεν είναι αυτόνομο προιόν. Είναι μια επέκταση της ασφάλειας, η οποία περιλαμβάνεται στις νέες αρχιτεκτονικές της. Τα τελευταία μοντέλα των κατασκευαστών έξυπνων τηλεφώνων περιέχουν ένα SoC που βασίζεται στη λογική, ότι τα χαρακτηριστικά της τεχνολογίας TrustZone μπορούν να ενεργοποιηθούν από τον κατασκευαστή.[33]

Η λύση TrustZone

Ο πρωταρχικός στόχος της αρχιτεκτονικής ασφάλειας είναι να καταστεί δυνατή η δημιουργία ενός προγραμματιζόμενου περιβάλλοντος που θα επιτρέπει την προστασία της εμπιστευτικότητας και της ακεραιότητα των αγαθών του συστήματος, ώστε να προστατεύονται από συγκεκριμένες επιθέσεις. Το ARM TrustZone έχει σχεδιαστεί για να παρέχει αυτή την προστασία χωρίς την προσθήκη επιπλέον SoC. Η ασφάλεια του συστήματος επιτυγχάνεται μέσω της κατανομής όλων των πόρων του υλικού και λογισμικού SoC, έτσι, ώστε να υπάρχουν σε μια από τις δυο περιοχές (την ασφαλή περιοχή για το υποσύστημα ασφάλειας και την

κανονική περιοχή για οτιδήποτε άλλο). Η τεχνολογία του υλικού που παρουσιάζεται υπάρχει μέσα στο TrustZone. Ενεργοποιώντας τη λειτουργία αυτή διασφαλίζεται, ότι όλα τα στοιχεία της κανονικής περιοχής δεν έχουν πρόσβαση στους πόρους της ασφαλούς περιοχής, επιτρέποντας έτσι τη δημιουργία μιας ισχυρής περιμέτρου μεταξύ των δυο περιοχών (όπως φαίνεται στην εικόνα 17).



Εικόνα 17: Αρχιτεκτονική TrustZone της ARM. [55]

Μπαίνοντας σε λειτουργία παρακολούθησης από την κανονική περιοχή, απαιτείται από το λογισμικό να εκτελεστεί ένα σύνολο εντολών που μπορούν να προκαλέσουν την είσοδο στην περιοχή. Το λογισμικό λειτουργίας οθόνης βλέπει όλες τις εντολές ως εξαιρέσεις, αξιολογώντας προσεκτικά κάθε μια πριν της χορηγήσει πρόσβαση στη περιοχή ασφάλειας. Όταν αλλάζει η κατάσταση από τη μια περιοχή στην άλλη, το λογισμικό παρακολούθησης αποθηκεύει την κατάσταση της τρέχουσας περιοχής και επαναφέρει την κατάσταση της περιοχής στη θέση, στην οποία μεταβαίνει. Στη συνέχεια, μια return-from-exception διεργασία επανεκκινεί την επεξεργασία στην ανακτημένη περιοχή. Το γεγονός αυτό καθιστά τις αξιόπιστες και μη καταστάσεις όχι μόνο διακριτές στην εικονική μνήμη αλλά και στον χρόνο. Ένας τέτοιος διαχωρισμός επιτρέπει στον πυρήνα της εφαρμογής την εναλλαγή μεταξύ των δύο περιοχών με ένα τέτοιο τρόπο, ώστε να αποτρέπεται η είσοδος των ευαίσθητων πληροφοριών της εφαρμογής σε μια λιγότερο αξιόπιστη περιοχή.

Χαρακτηριστικά TrustZone

Αρχικά, η τεχνολογία TrustZone δεν παρέχει μια σταθερή λύση ασφάλειας, η οποία επιλύει όλα τα προβλήματα, ούτε είναι μια ανεξάρτητη εξωτερική συσκευή. Είναι ένας «ενεργοποιός», που παρέχει την υποδομή και επιτρέπει στον σχεδιαστή SoC να επιλέξει από μια σειρά στοιχείων που ικανοποιούν τις συγκεκριμένες λειτουργίες μέσα σε ένα περιβάλλον ασφαλείας. Η τεχνολογία TrustZone διασφαλίζει, ότι τα ευαίσθητα δεδομένα αποθηκεύονται, αναλύονται και

προστατεύονται σε ένα αξιόπιστο περιβάλλον. Ένα παράδειγμα, είναι ο τρόπος με τον οποίο η Apple διαχειρίζεται το λεγόμενο *TouchID* , ένα χαρακτηριστικό με το οποίο οι χρήστες χρησιμοποιούν το δακτυλικό τους αποτύπωμα για να ξεκλειδώσουν την οθόνη της συσκευής. Η Apple αποθηκεύει το αποτύπωμα στον ασφαλή χώρο και ο σαρωτής δακτυλικών αποτυπωμάτων λειτουργεί μόνο όταν ο ασφαλής χώρος είναι ενεργοποιημένος.[60] Ομοίως, άλλα περιφερειακά όπως το πληκτρολόγιο, η κάμερα, το Near Field Communication (NFC) και το Bluetooth μπορούν να ασφαλιστούν με τον ίδιο τρόπο. Κάθε εφαρμογή μπορεί να εκτελεστεί σε ένα αξιόπιστο περιβάλλον, εφόσον μια αξιόπιστη διεπαφή χρήστη έχει δημιουργηθεί. Είναι θέμα επιλογής, ποιο στοιχείο θα προστατευτεί.[33]

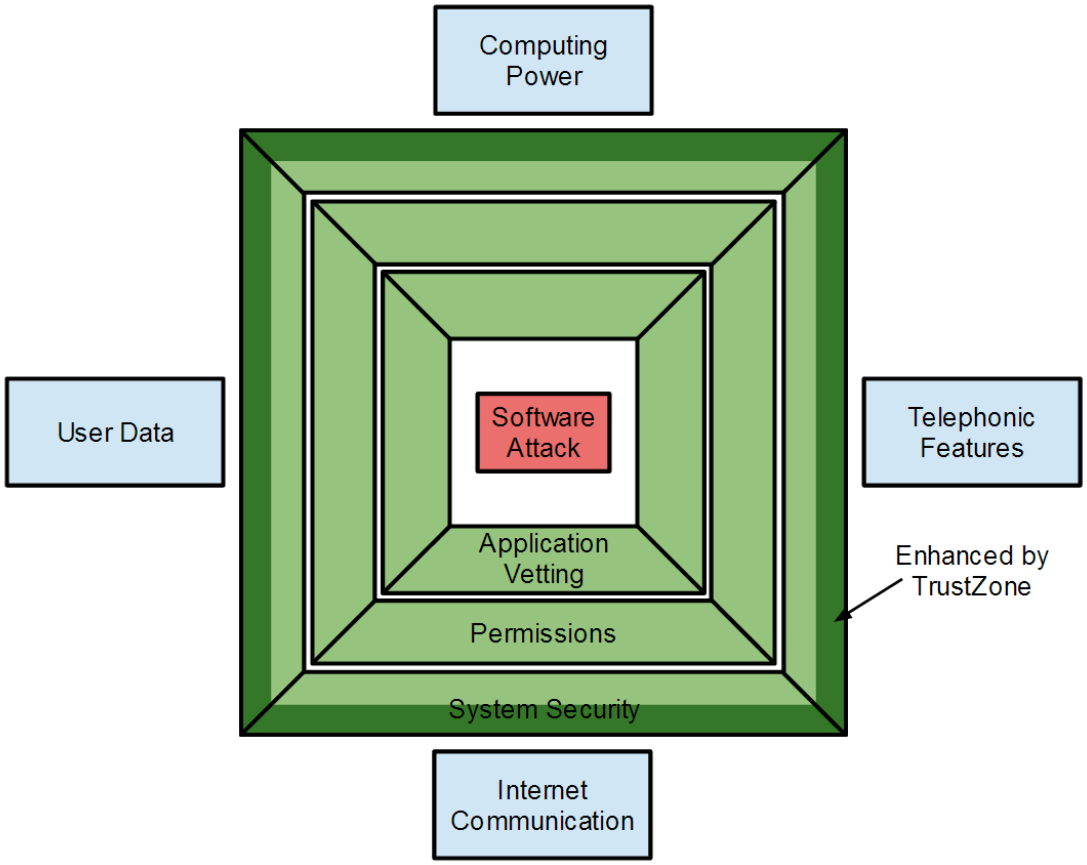
Μια εφαρμογή κινητής συσκευής για πληρωμές είναι ένα άλλο παράδειγμα. Για να διασφαλιστεί η ασφαλής πληρωμή, η εφαρμογή μπορεί να εμφανίζει τις πληροφορίες της σε ένα αξιόπιστο παράθυρο με την είσοδο ενός PIN από το πληκτρολόγιο του χρήστη, ώστε να γίνεται αποδεκτή η συναλλαγή. Έχοντας ένα ασφαλισμένο κανάλι, μέσω του οποίου οι χρήστες θα καταχωρούν τις πληροφορίες τους στη συσκευή , η παρακολούθηση και η υποκλοπή πληροφοριών γίνεται πολύ πιο δύσκολη. Παρακάτω είναι μια λίστα με παραδείγματα εφαρμογών.

- Ασφαλής είσοδος PIN για τον έλεγχο ταυτότητας χρήστη σε πληρωμές μέσω κινητού τηλεφώνου
- Προστασία DRM στα μέσα υψηλής αξίας
- Διαχωρισμός ιδιωτικής και επαγγελματικής λειτουργίας (BYOD)
- Διαχείριση αδειών χρήσης λογισμικού
- Ασφαλές boot
- Διαχείριση πιστοποιητικών.

Τυπικά, ένα λειτουργικό σύστημα εκτελείται στον κανονικό χώρο , ενώ κάποιες λειτουργίες, όπως αυτές που αναφέρθηκαν παραπάνω, εκτελούνται στον ασφαλή χώρο.

Μοντέλο απειλής και TrustZone

Η επίδραση της τεχνολογίας στο μοντέλο απειλής φαίνεται στην παρακάτω εικόνα. Το αποτέλεσμα είναι η δημιουργία ενός σταθερού στρώματος ασφαλείας, το οποίο θα αποτρέψει τη διαρροή των δεδομένων του χρήστη.



Εικόνα 18: Μοντέλο απειλής και η επίδραση του TrustZone. Με σκούρο πράσινο φαίνεται η επίδραση της τεχνολογίας TrustZone. [55]

5.2 Λύσεις Διαφημίσεων

Τα περισσότερα προβλήματα που σχετίζονται σήμερα με τις διαφημίσεις προέρχονται από το γεγονός, ότι οι βιβλιοθήκες διαφημίσεων είναι ενσωματωμένες στην εφαρμογή που φιλοξενούνται. Οι βιβλιοθήκες διαφημίσεων απαιτούν επιπλέον δικαιώματα και μια νέα δραστηριότητα πρέπει να προστεθεί. Αυτή η προσέγγιση έχει θετικά και αρνητικά αποτελέσματα:

Πλεονεκτήματα

- Οι βιβλιοθήκες διαφημίσεων είναι ενσωματωμένες στην εφαρμογή που τις φιλοξενούν με τέτοιο τρόπο, ώστε είναι δύσκολο να διαχωριστούν και να εμποδιστούν. Αυτό καθιστά δύσκολο το ενδεχόμενο να εξαπατηθεί ο πάροχος δικτύου.
- Η μέθοδος υλοποίησης είναι εύκολη και δεν απαιτεί εξειδικευμένες τεχνικές γνώσεις από τον προγραμματιστή ή τον χρήστη.

Μειονεκτήματα

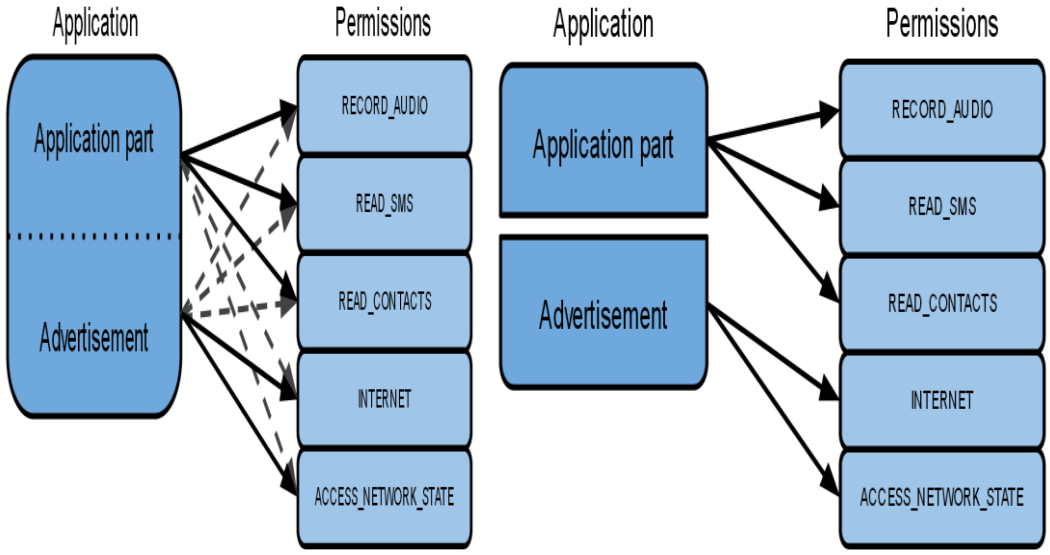
- Η εφαρμογή και οι βιβλιοθήκες διαφημίσεων μοιράζονται τα ίδια δικαιώματα και έχουν πρόσβαση στους ίδιους πόρους του συστήματος. Αυτό, επίσης, περιέχει περιττή πληροφορία, όπως προσωπικές πληροφορίες και δεδομένα χρηστών ,τα οποία ενδεχομένως να είναι απρόθυμοι ή ανυποψίαστοι ότι θα μοιραστούν με τον πάροχο του δικτύου. Επιπλέον, αυτές οι πληροφορίες μπορούν να διαρρεύσουν σε κάποιον κακόβουλο.
- Επειδή το API των διαφημίσεων εκτελείται σε μία δραστηριότητα ,έχει την ικανότητα να εκτελεί κώδικα, και αυτό οδηγεί επικίνδυνα στη δυνατότητα λήψης κακόβουλου κώδικα κατά τον χρόνο εκτέλεσης.
- Οι τελικοί χρήστες δεν είναι σε θέση να γνωρίζουν, αν η εφαρμογή που θέλουν να εγκαταστήσουν περιέχει διαφημίσεις.
- Οι τελικοί χρήστες δεν γνωρίζουν ποια δικαιώματα χορηγούνται στην εφαρμογή και ποια στις βιβλιοθήκες διαφημίσεων.

Οι νέες προτάσεις που προτείνονται προσπαθούν να επιλύσουν τις αρνητικές συνέπειες του υφιστάμενου συστήματος, όσο το δυνατόν, και παράλληλα προσπαθούν να διατηρήσουν τα θετικά του. Επιπλέον, θα πρέπει να διατηρηθούν οι βασικές λειτουργίες των διαφημίσεων, όπως η εμφάνιση των banner , η υποστήριξη διαφορετικών παρόχων διαφημίσεων αλλά και η προσφορά εσόδων στους προγραμματιστές.

Στο υφιστάμενο περιβάλλον των διαφημίσεων δεν υπάρχει κανένας πραγματικός τρόπος για προστατευτούν οι χρήστες από τις κακόβουλες διαφημίσεις. Μια πιθανή λύση είναι η προσπάθεια αποφυγής χρήσης δωρεάν εφαρμογών, ωστόσο, αυτό μπορεί να είναι περίπλοκο και δαπανηρό. Αυτό συμβαίνει, επειδή ο χρήστης δεν μπορεί να γνωρίζει, εάν μια εφαρμογή περιέχει διαφημίσεις και συνήθως οι εφαρμογές που δεν περιέχουν διαφημίσεις κοστίζουν. Πιθανές βελτιώσεις για την αύξηση της ασφάλειας του χρήστη παρουσιάζονται στη συνέχεια.

5.2.1 Διαχωρισμός των δικαιωμάτων μέσα στην εφαρμογή

Μια λύση είναι η δημιουργία περιοχών μέσα στην εφαρμογή με διαφορετικά δικαιώματα[34]. Με αυτόν τον τρόπο μπορεί να έχουμε μια περιοχή δικαιωμάτων για την εφαρμογή, η οποία διατηρεί όλα τα αρχικά δικαιώματα και δε θα επηρεάζεται από τις άλλες περιοχές δικαιωμάτων (σχήμα 5.6). Σε μια άλλη περιοχή μπορούμε να έχουμε τα δικαιώματα των διαφημίσεων, και έτσι να περιορίσουμε τα δικαιώματα για αυτήν την περιοχή. Οι μέθοδοι που εφαρμόζονται μεταξύ της εφαρμογής και της βιβλιοθήκης διαφημίσεων θα διατηρηθούν και θα εκτελούνται στην ίδια εικονική μηχανή, αλλά θα πρέπει να έχουν ξεχωριστά δικαιώματα.



α) Υφιστάμενο σύστημα β) Διαχωρισμός περιοχών

Εικόνα 19: Σχέση μεταξύ εφαρμογής και δικαιωμάτων στο υφιστάμενο σύστημα και στο σύστημα με διαχωρισμό περιοχών. [55]

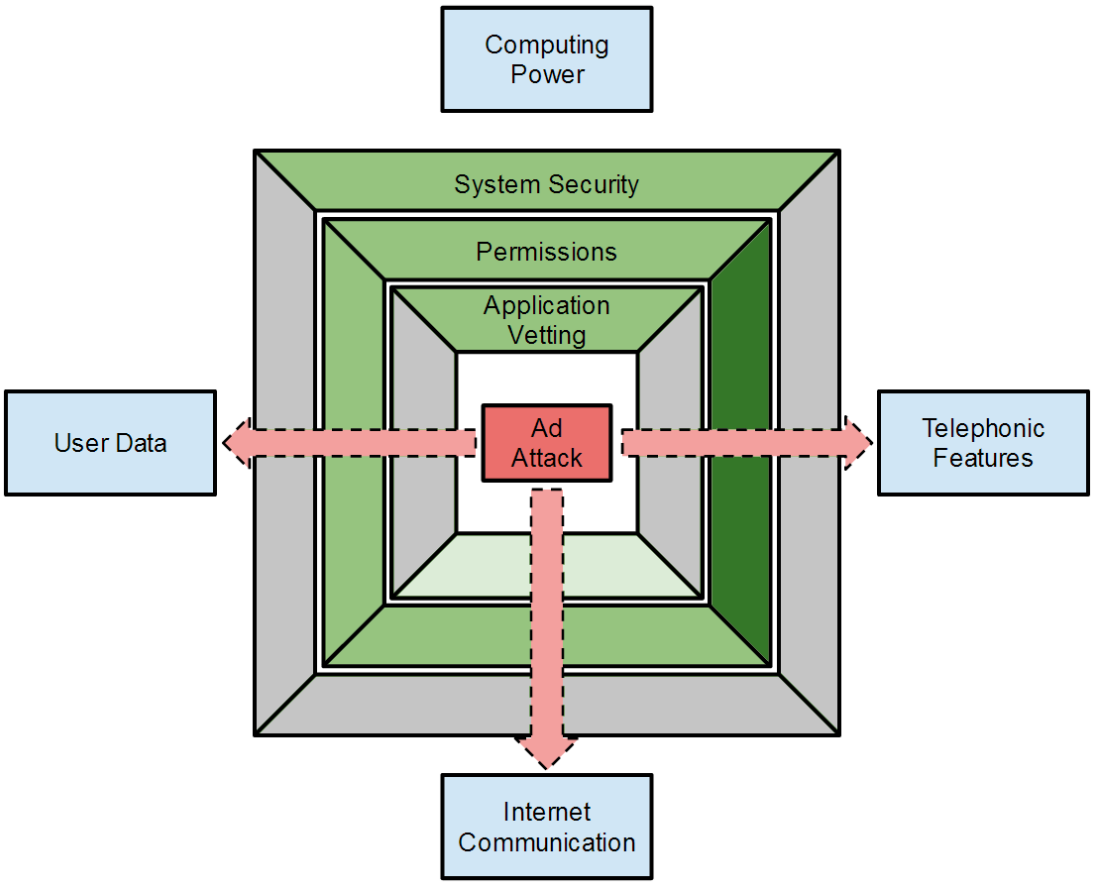
Η λύση αυτή διατηρεί όλα τα θετικά χαρακτηριστικά του ισχύοντος συστήματος, διότι υπάρχουν ελάχιστες αλλαγές στον τρόπο με τον οποίο η διαφήμιση ενσωματώνεται μέσα στην εφαρμογή. Η μόνη διαφορά είναι, ότι η βιβλιοθήκη διαφημίσεων ορίζεται ως μια υπό-περιοχή μέσα στην εφαρμογή και θα έχει το δικό της σύνολο δικαιωμάτων.

Η λύση αυτή διευθετεί πολλά από τα μειονεκτήματα που αναφέρθηκαν παραπάνω. Ο διαχωρισμός των δικαιωμάτων σε περιοχές θα αφαιρέσει τα “τρωτά” σημεία που δημιουργούνται από την κοινή χρήση δικαιωμάτων, και η ίδια η περιοχή θα ενημερώνει τον χρήστη για τα δικαιώματα που έχουν χορηγηθεί στην εφαρμογή. Η ικανότητα λήψης και εκτέλεσης κακόβουλου κώδικα θα υπάρχει, διότι η βιβλιοθήκη διαφημίσεων θα είναι ακόμα ενσωματωμένη στην εφαρμογή και θα μοιράζονται το ίδιο UID, αλλά οι πιθανότητες μόλυνσης θα είναι περιορισμένες, καθώς θα έχει μειωθεί το σύνολο των διαθέσιμων δικαιωμάτων. Οι χρήστες θα εξακολουθούν να μην μπορούν να αναγνωρίσουν τις διαφημίσεις που είναι σε μια εφαρμογή, εφόσον αυτές ενυπάρχουν σε μια υπό-περιοχή της.

Το πρόβλημα με αυτή τη λύση είναι, ότι απαιτείται μετατροπή του λειτουργικού συστήματος. Το λειτουργικό σύστημα πρέπει να τροποποιηθεί έτσι, ώστε να υποστηρίζει δυο περιοχές δικαιωμάτων. Αυτή είναι μια αλλαγή, η οποία πρέπει να γίνει από την επίσημη ομάδα προγραμματιστών του Android. Ένα πιο σοβαρό πρόβλημα είναι, ότι η εικονική μηχανή Dalvik δεν υποστηρίζει τον διαχωρισμό του κώδικα μέσα στην ίδια εικονική μηχανή [33]].Η παρεμπόδιση εκτέλεσης μεταξύ των διαφορετικών μερών της εφαρμογής πρέπει να αντιμετωπιστεί.

Επίδραση στο μοντέλο απειλής

Αυτή η λύση βελτιώνει τα δικαιώματα σχετικά με τις επιθέσεις διαφημίσεων. Παρακάτω φαίνεται το μοντέλο απειλής:

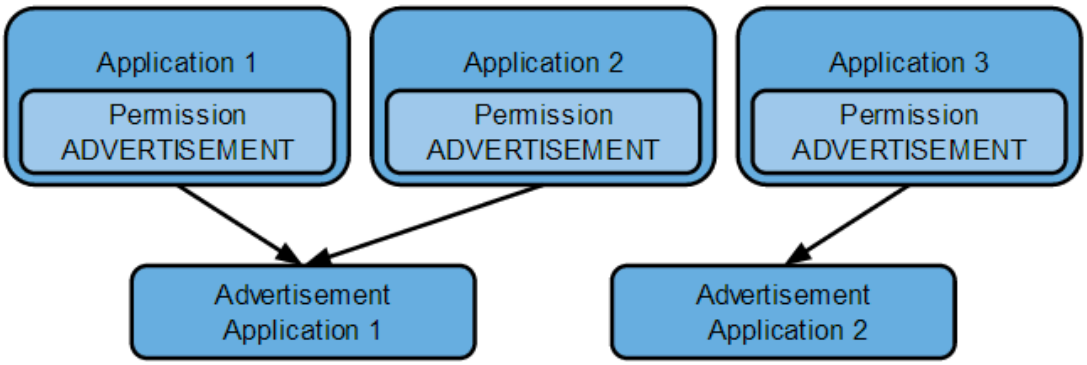


Εικόνα 20: Μοντέλο απειλής για τον διαχωρισμό των δικαιωμάτων μέσα στην εφαρμογή. [55]

5.2.2 Διαχωρισμός Εφαρμογής

Μια διαφορετική προσέγγιση για να επιτευχθεί ο διαχωρισμός των δικαιωμάτων είναι να έχουμε τις διαφημιστικές βιβλιοθήκες σε διαφορετική εφαρμογή [33] μιας και οι εφαρμογές δεν μοιράζονται τα δικαιώματα. Όταν ένας χρήστης εγκαθιστά μια κανονική εφαρμογή, αυτή θα τον ενημερώνει ποια διαφημιστική βιβλιοθήκη απαιτείται και θα ζητά την εγκατάσταση της διαφημιστικής εφαρμογής (εάν δεν είναι ήδη εγκατεστημένη). Κάθε διαφορετική βιβλιοθήκη διαφημίσεων θα χρειάζεται μια εφαρμογή και όλες οι κανονικές εφαρμογές θα συνδέονται με μια (ή περισσότερες) από τις εφαρμογές διαφημίσεων (εικόνα 21). Οι δύο αυτές εφαρμογές θα έχουν το δικό τους σύνολο δικαιωμάτων και οι κανονικές εφαρμογές θα περιλαμβάνουν μια νέα άδεια, *ADVERTISEMENT*. Αυτή η άδεια χρησιμοποιείται για να επιτρέψει στις εφαρμογές να εμφανίζουν διαφημίσεις, για να ενημερώνει τον χρήστη, ότι μια εφαρμογή περιέχει διαφήμιση και ότι εξαρτάται από άλλη εφαρμογή η εύρυθμη λειτουργία της. Για να προβληθούν οι διαφημίσεις, το σύστημα πρέπει δείξει δύο εφαρμογές στην οθόνη ταυτόχρονα. Για να επιτευχθεί αυτό, το λειτουργικό σύστημα Android πρέπει να τροποποιηθεί και θα πρέπει να περιλαμβάνει ορισμένα πρωτότυπα χαρακτηριστικά που θα επιτρέπουν σε μια δραστηριότητα που εκτελείται στο παρασκήνιο να γίνει ανεξάρτητη[35]. Επιπλέον, για την εμφάνιση των δυο εφαρμογών, οι

εφαρμογές των διαφημίσεων πρέπει να έχουν τη δυνατότητα να καταχωρούν «click» γεγονότα , τα οποία προσθέτουν επιπλέον εμπόδια [35].



Εικόνα 21: Ξεχωριστές διαφημιστικές εφαρμογές βιβλιοθηκών συνδεδεμένες σε διαφορετικές κανονικές εφαρμογές. [55]

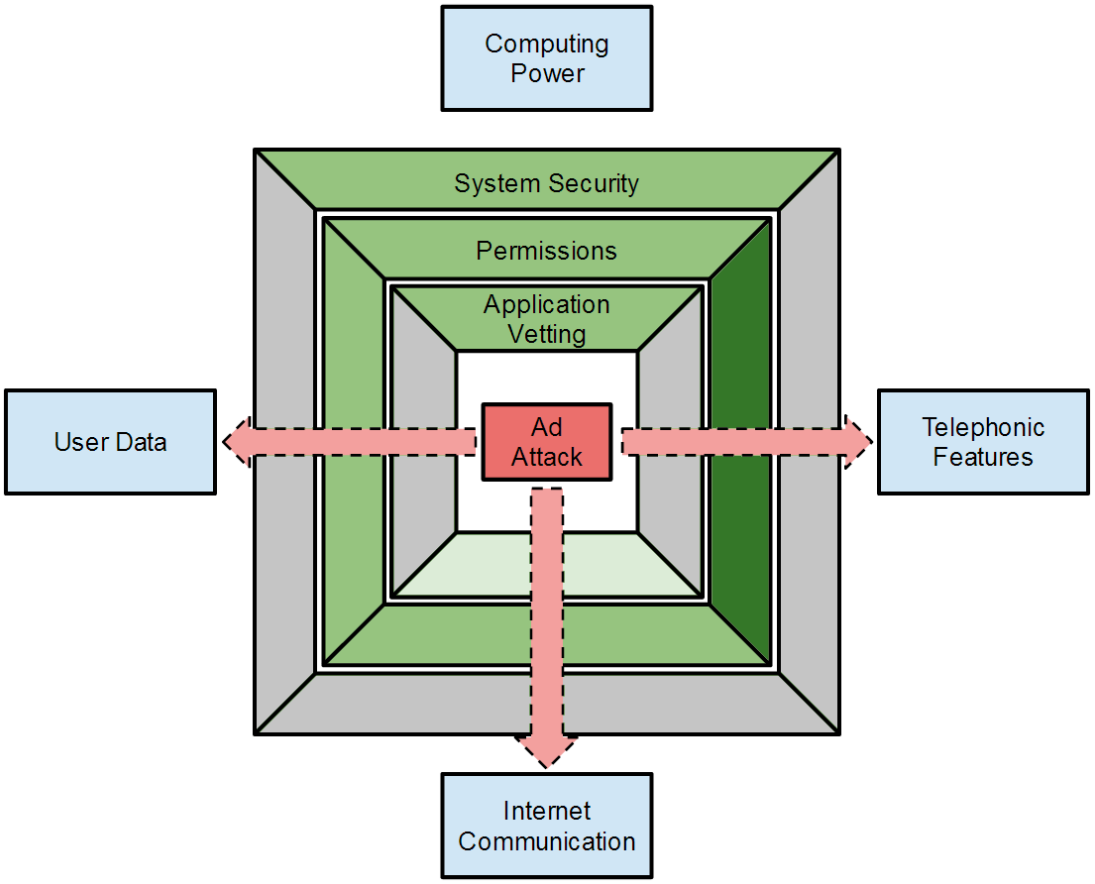
Ένα πρόβλημα με αυτή τη λύση είναι, ότι και τα δυο χαρακτηριστικά του υφιστάμενου συστήματος χάνονται. Με τη λογική της τοποθέτησης των διαφημίσεων σε διαφορετική εφαρμογή θα είναι πολύ πιο εύκολη η απεγκατάστασή της ή ο αποκλεισμός της, με αποτέλεσμα να επηρεάσει τα έσοδα των προγραμματιστών που βασίζονται στις διαφημίσεις. Επιπλέον, το νέο σύστημα έχει πιο περίπλοκη σχέση μεταξύ της κανονικής εφαρμογής και της βιβλιοθήκης διαφημίσεων. Το αποτέλεσμα είναι μια πιο περίπλοκη χρήση τόσο για τους προγραμματιστές, όσο και για τους χρήστες.

Η απώλεια των θετικών χαρακτηριστικών αντισταθμίζεται λαμβάνοντας υπ' όψιν τα αρνητικά χαρακτηριστικά που επιλύει. Κατ' αρχάς, με το διαχωρισμό της βιβλιοθήκης διαφημίσεων και της εφαρμογής που τη φιλοξενεί σε δυο διαφορετικές εφαρμογές. Το ζήτημα με τα κοινά δικαιώματα επιλύεται και έτσι , βιβλιοθήκη και εφαρμογή δεν θα έχουν το ίδιο UID. Θα εξακολουθεί να είναι δυνατή η λήψη και εκτέλεση κακόβουλου κώδικα, αλλά η έλλειψη πρόσθετων δικαιωμάτων περιορίζει το πρόβλημα. Δεύτερον, παρέχει πρόσθετη προστασία για την απειλή, η οποία μπορεί να επιτευχθεί πιο εύκολα , διότι η λογική της διαφήμισης έχει απομονωθεί σε ξεχωριστή εφαρμογή. Οι τελικοί χρήστες θα έχουν κάποιο τρόπο να αναγνωρίσουν τις διαφημίσεις σε μια εφαρμογή (μέσω της άδειας *ADVERTISEMENT)* αλλά και τα δικαιώματα που αυτές απαιτούν.

Τα βασικά μειονεκτήματα με αυτή τη λύση είναι η απώλεια των θετικών χαρακτηριστικών του τρέχοντος συστήματος. Επίσης , απαιτεί είτε τροποποίηση του λειτουργικού συστήματος είτε υποστήριξη των νέων χαρακτηριστικών από τις εφαρμογές, έτσι ώστε να εμφανίζονται δυο εφαρμογές στη οθόνη ταυτόχρονα.

Επίδραση στο μοντέλο απειλής

Παρακάτω στο σχήμα 5.9 φαίνεται η επίδραση της λύσης που προτάθηκε στο μοντέλο απειλής.

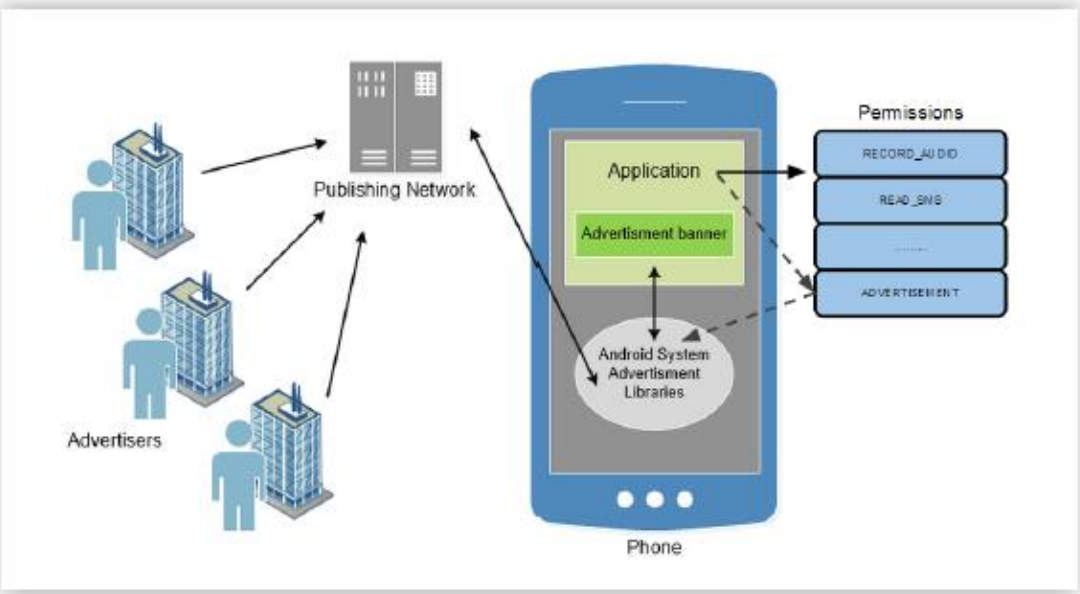


Εικόνα 22: Μοντέλο απειλής της λύσης για τον διαχωρισμό εφαρμογών. [55]

5.2.3 AdDroid

Η Τρίτη λύση είναι μια πρόταση που παρουσιάζεται στην εργασία “AdDroid: Privilege separation for applications and advertisers in Android” από τους Paul Pearce, Adrienne Prter Felt, Gabriel Nunez, και David Wagner . Προτείνεται να συμπεριληφθεί η βιβλιοθήκη διαφημίσεων μέσα στο Android σύστημα και όχι το διαφημιστικό δίκτυο. Αυτό το σύστημα θα απαιτεί μια νέα άδεια *ADVERTISEMENT*, έτσι ώστε όταν μια εφαρμογή δηλώνει αυτή την άδεια, τότε θα αποκτάει πρόσβαση σε ένα API που θα παρέχει λειτουργίες διαφημίσεων[34].

Το AdDroid αποτελείται από δύο μέρη. Το API διαφημίσεων είναι το πρώτο από αυτά και υπάρχει στον χώρο της εφαρμογής (μέσα στην εφαρμογή που φιλοξενεί τη διαφήμιση). Οι κλάσεις και μέθοδοι που χρησιμοποιούνται για τις λειτουργίες διαφημίσεων παρέχονται στον προγραμματιστή μέσω API [34]. Αυτό περιλαμβάνει το δίκτυο διαφημίσεων που θα συνδεθεί και μια νέα διεπαφή χρήστη για την εμφάνιση των διαφημιστικών banner. Το API θα είναι το ίδιο για όλες τις εφαρμογές, ανεξάρτητα από το διαφημιστικό δίκτυο που θα χρησιμοποιεί ο προγραμματιστής. Επειδή το API βρίσκεται στον χώρο της εφαρμογής, μοιράζεται, επίσης, τα δικαιώματα με την εφαρμογή που τη φιλοξενεί.



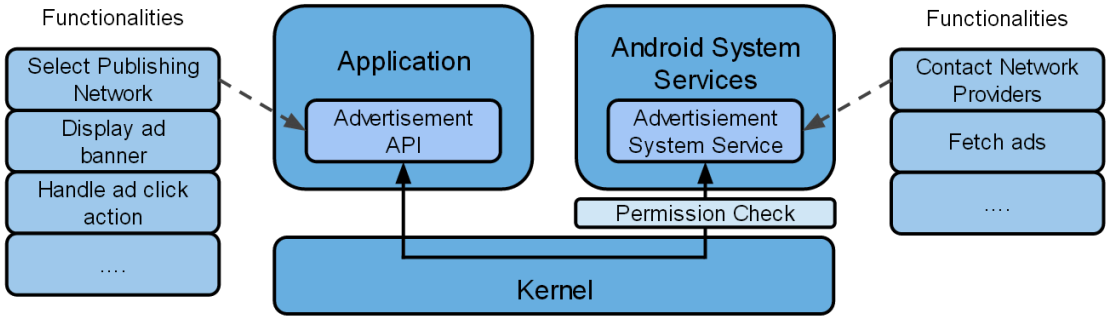
Εικόνα 23: Σχέση μεταξύ Διαφημιστών, Διαφημιστικών δικτύων, Διαφημιστικών βιβλιοθηκών και εφαρμογής μέσα στο AdDroid. [55]

Για να αποφευχθεί η ανταλλαγή των δικαιωμάτων μεταξύ της εφαρμογής που φιλοξενεί τη διαφήμιση και της βιβλιοθήκης διαφημίσεων, οι ευαίσθητες λειτουργίες τοποθετούνται σε μια υπηρεσία συστήματος αντί του API [34]. Η υπηρεσία συστήματος του AdDroid βρίσκεται έξω από τον εσωτερικό χώρο της εφαρμογής. Η πρόσβαση στην υπηρεσία συστήματος προστατεύεται με τη χρήση της άδειας *ADVERTISEMENT*. Όταν μια εφαρμογή αιτείται μια νέα διαφήμιση να εμφανιστεί, θα κάνει μια κλήση IPC στην υπηρεσία συστήματος που στη συνέχεια θα εκτελεί τον έλεγχο δικαιωμάτων. Στην εικόνα 24 φαίνεται η σχέση μεταξύ του API και του συστήματος υπηρεσίας. Το μεγαλύτερο μέρος της λογικής των διαφημίσεων τοποθετείται στο API και οι μόνες λειτουργίες που βρίσκονται στις υπηρεσίες συστήματος είναι:

- Σύνδεση διαφημίσεων με τον πάροχο διαφημίσεων.
- Λήψη διαφημίσεων που αποστέλλονται μέσω του παρόχου διαφημίσεων.
- Παροχή της εφαρμογής με τις διαφημίσεις που έχουν ληφθεί.

Η λύση αυτή διατηρεί τα πλεονεκτήματα του υφιστάμενου συστήματος και ακόμη βελτιώνει τις μεθόδους της εφαρμογής. Η λύση κατά την οποία το λειτουργικό σύστημα Android χειρίζεται το μεγαλύτερο μέρος της λογικής διαφημίσεων θα είναι μια πολύ πιο εύκολη διεργασία για τους προγραμματιστές. Επειδή η διαφημιστική βιβλιοθήκη περιλαμβάνεται ήδη στο σύστημα,

πολλά από τα βήματα υλοποίησης της βιβλιοθήκης δεν θα είναι αναγκαία. Επιπλέον, η μόνη αλλαγή που απαιτείται στο αρχείο manifest.xml είναι η δήλωση της άδειας *ADVERTISEMENT*.



Εικόνα 24: Σύνδεση μεταξύ API διαφημίσεων και υπηρεσίας συστήματος. [55]

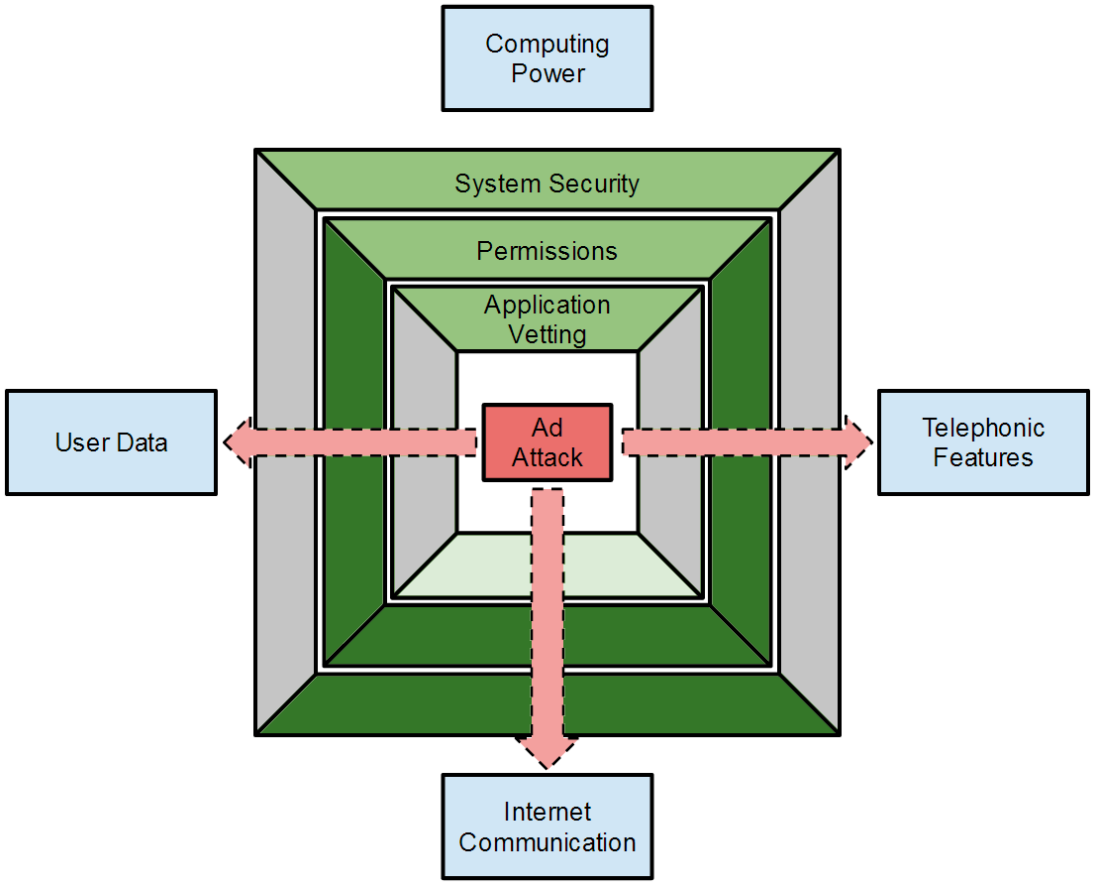
Το AdDroid λύνει όλα τα μειονεκτήματα που αναφέρθηκαν στο τρέχον σύστημα διαφημίσεων:

- Με το διαχωρισμό των δικαιωμάτων στον χώρο των διαφημιστικών βιβλιοθηκών, τα δίκτυα διαφημίσεων δεν μπορούν να ζητήσουν ευαίσθητες πληροφορίες από τη συσκευή ενός χρήστη.
- Επειδή το σύστημα Android παρέχει τον κώδικα για τις βιβλιοθήκες διαφημίσεων, τα κακόβουλα δίκτυα διαφημίσεων δεν μπορούν να εισάγουν δικές τους λειτουργίες και ευπάθειες.
- Τα δικαιώματα της *ADVERTISEMENT* θα ενημερώνουν τον χρήστη, ότι η εφαρμογή περιέχει διαφημίσεις.
- Με αυτή τη λύση όλα τα δικαιώματα εκχωρούνται στην εφαρμογή που φιλοξενεί τη διαφήμιση.

Μια σημαντική πολυπλοκότητα που σχετίζεται με τη προτεινόμενη λύση είναι, ότι η επίσημη ομάδα προγραμματιστών του Android πρέπει να τροποποιήσει το λειτουργικό σύστημα. Επιπλέον, στην υφιστάμενη λύση τα δίκτυα διαφημίσεων αναβαθμίζουν και εισάγουν νέες λειτουργίες στις δικές τους βιβλιοθήκες διαφημίσεων και αυτό συμβαίνει ανεξάρτητα από την έκδοση του λειτουργικού συστήματος Android. Με το AdDroid, οι αναβαθμίσεις στις βιβλιοθήκες των διαφημίσεων θα γίνονται μέσω νέας επίσημης διανομής του Android συστήματος.

Επίδραση στο μοντέλο απειλής

Όπως και οι δυο προηγούμενες λύσεις, έτσι και αυτή, βελτιώνει τον τρόπο με τον οποίο διευθετούνται τα δικαιώματα των διαφημίσεων. Σε αντίθεση με τις προηγούμενες λύσεις, το AdDroid βελτιώνει την ασφάλεια του συστήματος και αυτό βοηθά στην αποτροπή κακόβουλων βιβλιοθηκών και λήψης δυναμικού κώδικα που στοχεύει στην επικοινωνία της συσκευής.



Εικόνα 25: Μοντέλο απειλής για τη λύση AdDroid. [55]

5.3 Λύσεις WebView

Οι λύσεις που παρουσιάζονται παρακάτω έχουν σχεδιαστεί για να εμποδίσουν τις επιθέσεις WebView από κακόβουλες ιστοσελίδες.

5.3.1 Λύση στο σύστημα Android

Με το API 17 του λειτουργικού συστήματος Android , εισήχθη προστασία έναντι των WebView επιθέσεων.[36] Οι εφαρμογές που στοχεύουν σε αυτό το API λαμβάνουν την παρακάτω ενημέρωση : «WebView.addJavascriptInterface requires explicit annotations on method for them to ba accessible from Javascript» [36]. Αυτό σημαίνει, ότι οποιοσδήποτε προγραμματιστής οφείλει να σχολιάσει κάθε δημόσια μέθοδο που θέλει να έχει πρόσβαση στη Javascript χρησιμοποιώντας το 'JavascriptInterface'. Δείτε το παράδειγμα 1. Για τις προηγούμενες εκδόσεις Android όλες οι δημόσιες μέθοδοι είναι διαθέσιμες μέσω του *webView.addJavascriptInterface.*

```
class JsObject {
        @JavascriptInterface
        public String toString() { return "injectedObject"; }
        }
    webview.addJavascriptInterface(new JsObject(), "injectedObject");
    webview.loadData("", "text/html", null);
    webview.loadUrl("javascript:aler(injectedObject.toString())");
```

Παράδειγμα 1: Μέθοδος η οποία επιτρέπει την εκτέλεση Javascript

Οι συσκευές που χρησιμοποιούν Android API μικρότερο του 4.1 είναι ευάλωτες στις επιθέσεις WebView μέσω της χρήσης *addJavascriptInterface*.

5.3.2 Εναλλακτικά μέτρα ασφάλειας

Απενεργοποίηση υποστήριξης.

Όταν χρησιμοποιείται ένα WebView, οι προγραμματιστές δε θα πρέπει να ενεργοποιούν επιπλέον λειτουργίες του από αυτές που χρειάζονται. Παρακάτω παρουσιάζονται τρία παραδείγματα.

Απενεργοποίηση Javascript.

Εάν το WebView δε χρησιμοποιεί Javascript, δεν υπάρχει λόγος να είναι αυτή η επιλογή ενεργοποιημένη. Η Android κλάση *WebSetting* μπορεί να χρησιμοποιηθεί για την απενεργοποίηση του Javascript μέσα στο Webview . Η απαραίτητη μέθοδος παρουσιάζεται στο παράδειγμα 2 [37]. Αυτή η μέθοδος είναι απενεργοποιημένη από το σύστημα ,αλλά καλό είναι να δηλώνεται εκ νέου.

```
Webview.getSettings().setJavascriptEnabled(false);
```

Παράδειγμα 2: μέθοδος η οποία απενεργοποιεί τη Javascript μέσα στο WebView

Απενεργοποίηση πρόσβασης στο σύστημα αρχείων.

Η μέθοδος *setAllowFileAccess(false).* Από την κλάση WebSettings μπορεί να απενεργοποιηθεί η πρόσβαση στο σύστημα αρχείων του Android μέσω του WebView, όπως φαίνεται στο παράδειγμα παρακάτω (παράδειγμα 3) :

```
Webview.getSettings().setAllowFileAccess(false);
```

Παράδειγμα 3: Κώδικας για την απενεργοποίηση πρόσβασης στο σύστημα αρχείων μέσω του WebView.

Απενεργοποίηση γέφυρας Javascript.

Η χρήση του addJavascriptInterface μπορεί να κάνει προσβάσιμο ένα αντικείμενο από τη Javascript. Αυτή η λειτουργία παραβιάζει την πολιτική *same origin policy* και πρέπει να χρησιμοποιείται μόνο, όταν είναι απαραίτητη, καθώς αποτελεί την αιτία για πολλά προβλήματα στο WebView.[37].

Αποτροπή φόρτωσης περιεχομένου από τρίτους

Εάν ένας χρήστης αιτηθεί έναν πόρο του συστήματος μέσα από ένα WebView, η μέθοδος *shouldOverrideUrlLoading* μπορεί να χρησιμοποιηθεί για να παρέμβει στο URL, το οποίο θα φορτωθεί και να καθορίζει, εάν είναι ασφαλές να φορτωθεί ή όχι.[37] Ένα παράδειγμα παρουσιάζεται στο σχήμα 5.16. που εξετάζει το νέο URL. Αν αυτό δεν είναι ασφαλές ο χρήστης θα αναδρομολογηθεί και το URL θα ανοιχτεί στον περιηγητή του συστήματος αντί του WebView,

Ένα πρόβλημα με αυτή τη λύση είναι, ότι η *shouldOverrideUrlLoading* δεν μπορεί να επέμβει, όταν οι πόροι του συστήματος φορτώνονται από ένα IFRAME ή από 'src'.

Επιθεώρηση πόρων συστήματος

Όταν οι πόροι του συστήματος φορτώνονται μέσω ενός IFRAM ή της ιδιότητας 'src', η μέθοδος shouldInterceptRequest από την κλάση WebviewClient δεν μπορεί να χρησιμοποιηθεί.

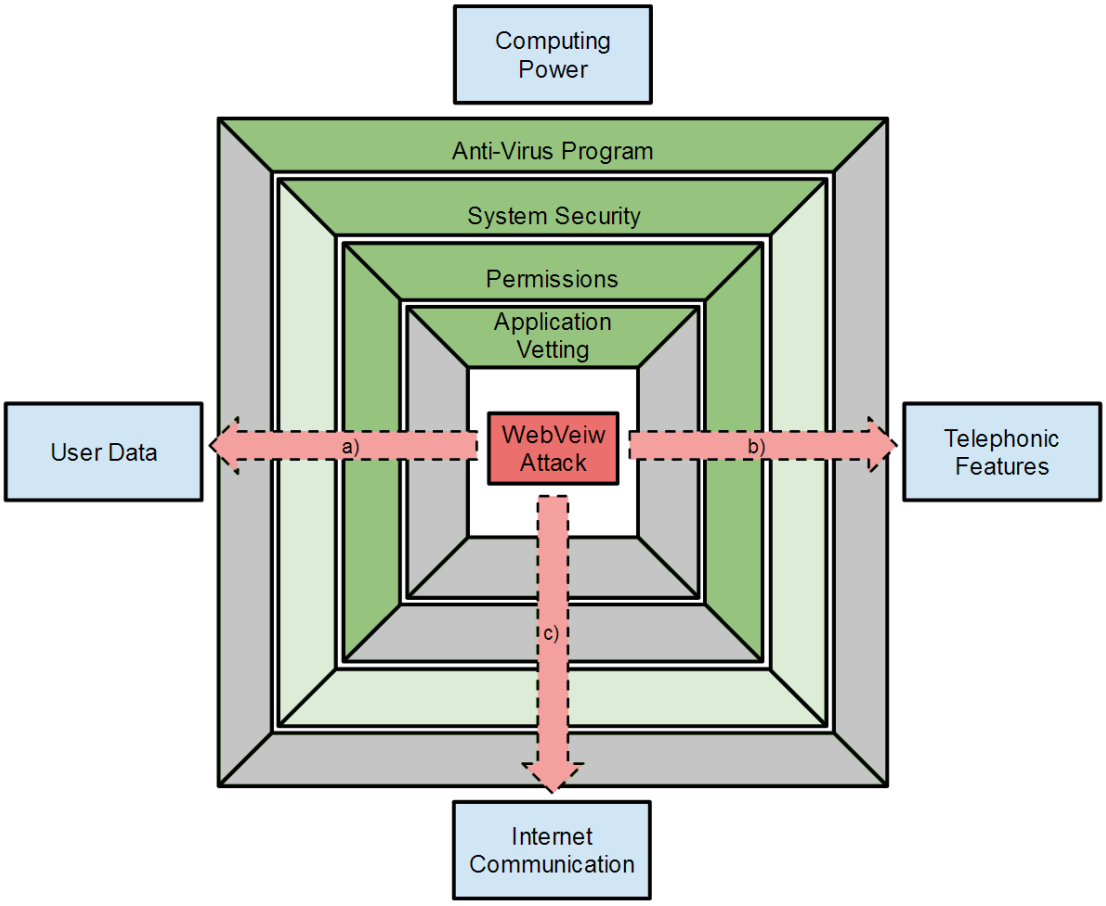
```
@Override
public boolean shouldOverrideUrlLoading(WebView webview, String url)
{
        if(Uri.parse(url).getHost() != "www.safeURL.com")
        {
                Intent defaultBrowser = new Intent(Intent.ACTION_VIEW, url);
                startActivity(defaultBrowser);
                return true;
        }
        else
        {
                return false;
        }
}
```

Εικόνα 26: Κώδικας για την επιθεώρηση ενός νέου URL που φορτώνεται μέσα από το WebView. [55]

Η μέθοδος ενημερώνει την εφαρμογή για ένα αίτημα πόρου του συστήματος, η οποία περιέχει ένα μοναδικό αναγνωριστικό *Uniform Resource Identifier (URI)* του πόρου του συστήματος. Για την παρακολούθηση των αιτημάτων μπορεί να χρησιμοποιηθεί ως μοτίβο μια αντιστοίχιση σχήματος. Μερικές χρήσιμες μέθοδοι για το σχήμα είναι η *getHost()*,*getScheme()* και η *getPath()*. Η μέθοδος μπορεί για παράδειγμα να χρησιμοποιηθεί για την ανίχνευση αιτημάτων για πόρους Javascript [37].

5.3.3 Λύσεις WebView και μοντέλο απειλής

Όλες οι λύσεις που παρουσιάστηκαν χρησιμοποιούνται για τη βελτίωση του συστήματος Android περιορίζοντας τις προσβάσεις και λειτουργίες ενός WebView που δεν είναι περιττές. Αυτές οι βελτιώσεις τοποθετούνται στο χώρο της ασφάλειας του συστήματος, όπως φαίνεται στην εικόνα 27 :

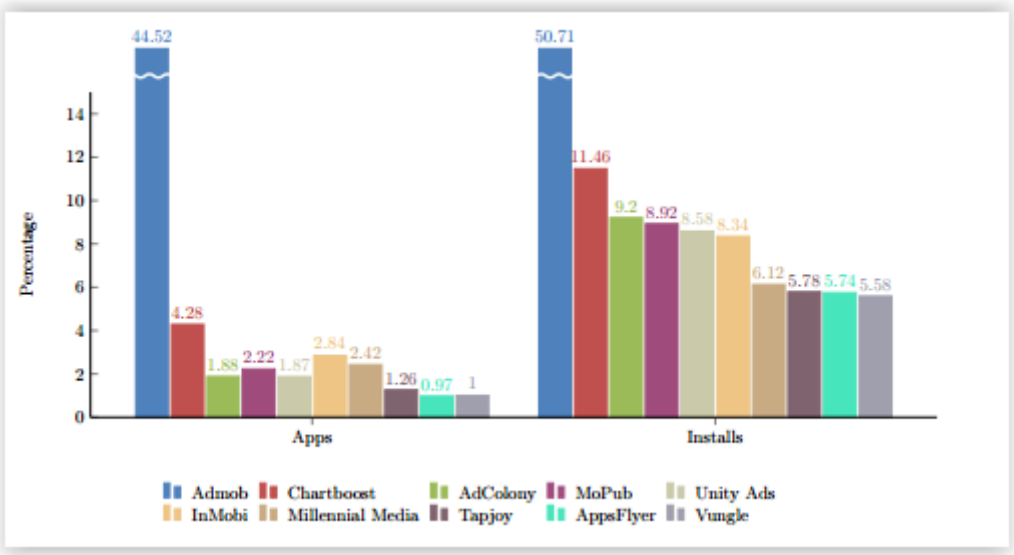


Εικόνα 27: Μοντέλο απειλής του WebView. [55]

6 AndroPatchApp

6.1 Τοποθέτηση του προβλήματος

Όπως ήδη αναφέρθηκε, μία από τις κύριες πηγές εσόδων στην αγορά των έξυπνων τηλεφώνων είναι το “freemium” μοντέλο διαφημίσεων, το οποίο μπορεί να ενσωματωθεί στις εφαρμογές με διάφορους τρόπους. Στο Android, η πιο κοινή βιβλιοθήκη διαφήμισης είναι το AdMob, σύμφωνα με την εικόνα 28, το οποίο επιτρέπει στις εφαρμογές να προβάλλουν διαφημίσεις στην κεφαλίδα ή το υποσέλιδο της οθόνης. Αυτό σημαίνει, ότι ο προγραμματιστής προσθέτει ένα πρόγραμμα περιήγησης, που μοιάζει με αντικείμενο, όπως αναφέραμε σε προηγούμενο κεφάλαιο. Αυτό το αντικείμενο ονομάζεται webview (προβολή ιστού) στην εφαρμογή του, η οποία λαμβάνει HTML και JavaScript κώδικα από τον διακομιστή διαφημίσεων και εμφανίζεται ανάλογα στην οθόνη του χρήστη.



Εικόνα 28: Στατιστικά στοιχεία των Adroid Ads. [38]

Πρέπει να σημειωθεί, ότι το σημερινό μοντέλο των διαφημίσεων λειτουργεί μέσω HTTP. Η χρήση του HTTP διευκολύνει τους διακομιστές διαφημίσεων, καθώς μπορούν να αποθηκεύουν προσωρινά το περιεχόμενό τους και να το συμπιέζει αυξάνοντας σημαντικά τη διακίνησή του. Ωστόσο, δεδομένου, ότι το περιεχόμενο παραδίδεται σε απλό κείμενο, ένας επιτιθέμενος θα μπορούσε εύκολα να εκτελέσει επίθεση “άνθρωπος στη μέση” Man In the Middle και να παρεμβάλει τον δικό του κώδικα. Επομένως, η χρήση του HTTP εκθέτει τον χρήστη σε επιθέσεις, οι οποίες μπορεί να προέλθουν από τον διαχειριστή του δικτύου ή τον φορέα παροχής (π.χ. η περίπτωση της Verizon "super-cookies" [39]) , όπου ο επιτιθέμενος άλλαζε το περιεχόμενο της ληφθείσας διαφήμισης, Τα παραπάνω σενάρια απεικονίζονται στο

εικόνα 29. Επίσης, ο χρήστης είναι εκτεθειμένος σε επιθέσεις από τον διαφημιζόμενο, καθώς οι διαφημίσεις που παραδίδονται στον διακομιστή διαφημίσεων για τη διανομή μπορεί να είναι κακόβουλος κώδικας [40].

Η έννοια της ad injection (διαφήμιση - ένεση) είναι πολύ απλή. Ο επιτιθέμενος παίρνει τον έλεγχο ό,τι εμφανίζεται στο πρόγραμμα περιήγησης ή την εφαρμογή των θυμάτων και διοχετεύει εκεί τις διαφημίσεις του. Με αυτό τον τρόπο, ο διαφημιζόμενος λαμβάνει νόμιμη κυκλοφορία και μπορεί να αυξήσει τους χρήστες που παρακολουθούν τη διαφήμιση. Συνεπώς, οι διαφημιζόμενοι είναι πρόθυμοι να πληρώσουν τον επιτιθέμενο για να τους παρέχει αυτού του είδους την υπηρεσία. Ενώ για τον επιτιθέμενο και τον διαφημιζόμενο το σενάριο είναι «win-win», η ποιότητα της υπηρεσίας που λαμβάνει ο χρήστης είναι υποβαθμισμένη. Οι ιστοσελίδες και οι εφαρμογές γεμίζουν με ενοχλητικές διαφημίσεις που κρύβουν το πραγματικό περιεχόμενο από τον χρήστη. Αυτού του είδους η πρακτική αυτή έχει γίνει κοινή στο οικοσύστημα των διαφημίσεων. Χαρακτηριστικό είναι το παράδειγμα ενός κακόβουλου λογισμικού, που ονομάζεται adware, του οποίου μοναδικός στόχος είναι να παρεμβάλλει διαφημίσεις στη συσκευή του θύματος.

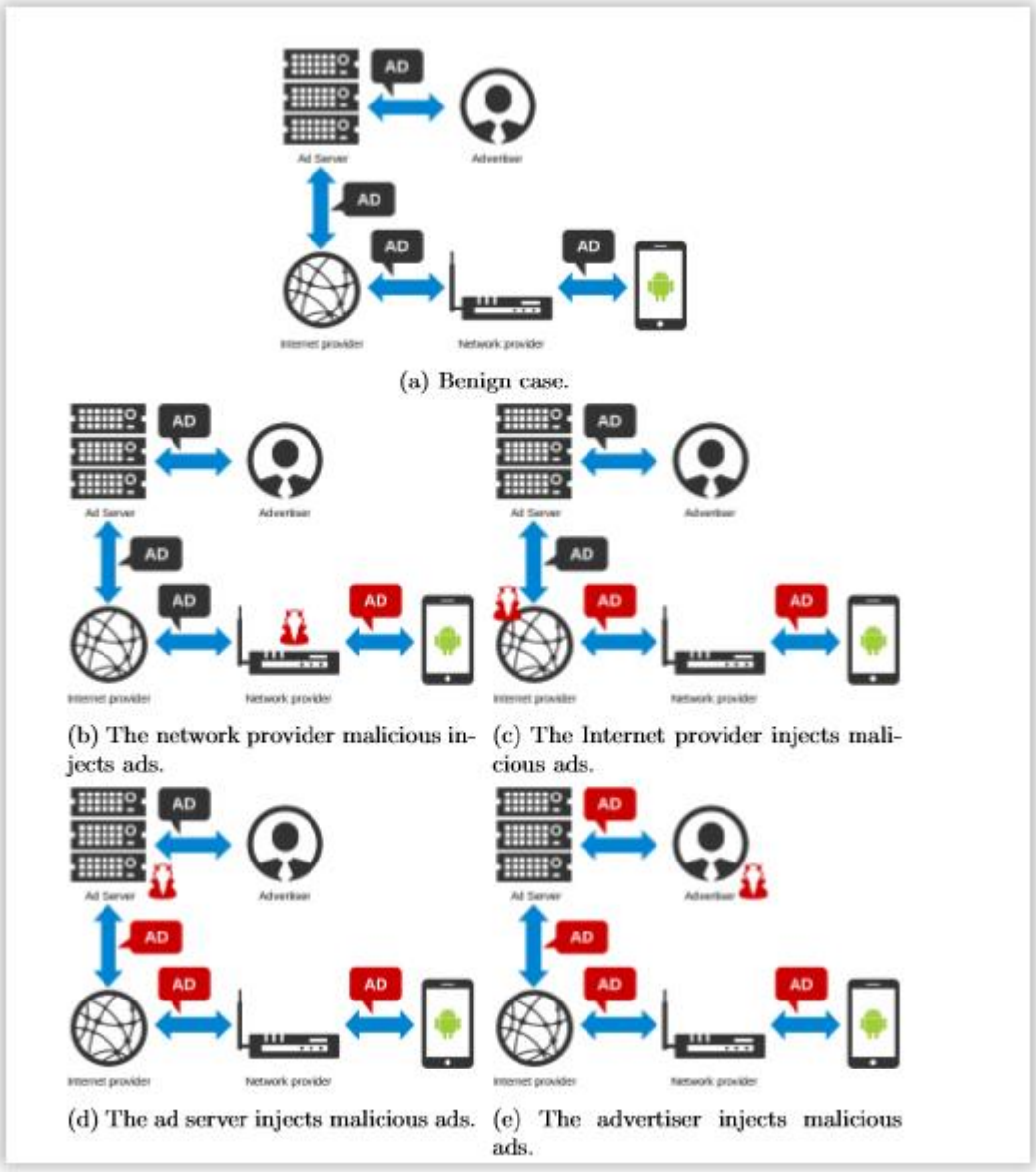
Η τάση αυτή έχει φτάσει σε τέτοιο βαθμό που μπορεί να γίνει κατανοητή από το γεγονός ότι 5,5% της κίνησης των μοναδικών IPs που μελετήθηκαν από την Google έγιναν inject με διαφημίσεις [41]. Ένα άλλο παράδειγμα αποτελεί ένας κατασκευαστής έξυπνων τηλεφώνων,ο οποίος πωλούσε τις συσκευές του με την adware εφαρμογή Superfish προεγκατεστημένη [42]. Επιπλέον, εκατοντάδες εφαρμογές στο Google Play έχουν βρεθεί να συνδέονται με τον εντοπισμό, τοποθεσίες που σχετίζονται με τις διαφημίσεις, και ιστοσελίδες που σχετίζονται με malware δραστηριότητα , ενώ άλλα έχουν βρεθεί να περιέχουν ευπάθειες[43].

Ας υποθέσουμε, ότι ένας χρήστης θέλει να εγκαταστήσει μια εφαρμογή στη συσκευή του. Κάθε εφαρμογή στο Android είναι συσκευασμένη σε ένα αρχείο APK, το οποίο είναι ένα συμπιεσμένο αρχείο που περιέχει όλα όσα πρέπει να εκτελέσει μια εφαρμογή, όπως ο κώδικας, εικόνες και τα πολυμέσα. Για να εγκατασταθεί μια εφαρμογή χρησιμοποιείται η υπηρεσία Package Manager . Το Google Play χρησιμοποιείται για να παρέχει έναν εύκολο τρόπο στον χρήστη να διαχειριστεί τις εφαρμογές του. Η άλλη επιλογή είναι να καλέσει τον Package Installer κατά τη λήψη ή την αντιγραφή μιας εφαρμογής στη συσκευή και να ζητήσει από το λειτουργικό σύστημα να εγκατασταθεί ή με τη χρήση ADB. Και στις δύο πρώτες περιπτώσεις, θα εμφανιστούν στην οθόνη του χρήστη τα δικαιώματα που πρέπει να χορηγήσει στην εφαρμογή, προκειμένου να εγκατασταθεί στη συσκευή του. Πρέπει να παρατηρήσουμε, ότι σε όλες εκδόσεις μέχρι τη Marshmallow , τα δικαιώματα των εφαρμογών δίνονταν μία φορά για την εφαρμογή και δεν μπορούσαν να ανακληθούν. Ωστόσο, στην έκδοση Marshmallow ο χρήστης μπορεί να ανακαλέσει τις άδειες ή να τις χορηγήσει, κατόπιν αιτήματος. Ενώ σε ορισμένες περιπτώσεις, τα δικαιώματα των εφαρμογών θα μπορούσαν να παρακαμφθούν [44][45], σε γενικές γραμμές θεωρείται, ότι αυτή η μέθοδος λειτουργεί αποτελεσματικά. Ιδιαίτερα, η συνεχής αύξηση των αιτημάτων της εφαρμογής για πρόσβαση σε ευαίσθητα δεδομένα ή δικαιώματα που δε σχετίζονται με τη λειτουργικότητα της εφαρμογής, έχουν κάνει τους χρήστες να αγνοούν τις προειδοποιήσεις [46] του λειτουργικού συστήματος. Ο κύκλος ζωής εγκατάστασης μιας εφαρμογής απεικονίζεται στην εικόνα 30.

Επιπλέον, η αρχιτεκτονική του Android επιτρέπει την ανταλλαγή δεδομένων μεταξύ άλλων υπηρεσιών. Εφαρμογές και πάροχοι περιεχομένου χρησιμοποιούν την επικοινωνία inter-component (ICC) [47][48]. Έτσι, ο προγραμματιστής μπορεί να ορίσει την πρόσβαση κάθε στοιχείου της συσκευής σε οποιοδήποτε επίπεδο της εφαρμογής δηλώνοντάς τη στο αρχείο AndroidManifest.xml. Ο προγραμματιστής θα ορίσει τα δικαιώματα που χρειάζεται η εφαρμογή στο αρχείο.

Ενώ η παραπάνω εφαρμογή φαίνεται συμπαγής, υπάρχουν αρκετά προβλήματα. Για παράδειγμα, η Grace et al[49] διαπίστωσε, ότι πολλές εφαρμογές χρησιμοποιούν το πρωτόκολλο HTTP για να κατεβάσουν και να εκτελέσουν τον κακόβουλο κώδικα στη συσκευή του χρήστη. Παρότι η χρήση του δεν είναι ασφαλής, έχει παρατηρηθεί σε πολλές Android εφαρμογές[50]. Παρ' όλα αυτά υπάρχει άλλη μια εγγενής απειλή. Η πρόσβαση στους πόρους

του συστήματος χορηγείται σε όλη την εφαρμογή και όχι ξεχωριστά σε κάθε στοιχείο της. Θεωρητικά, αυτό δεν εγείρει σημαντικό ζήτημα, δεδομένου, ότι όλα τα στοιχεία της συσκευής ορίζονται από την ίδια οντότητα, τη οποία ο προγραμματιστής έχει δημιουργήσει στην εφαρμογή του.



Εικόνα 29: Σενάρια επίθεσης στο δίκτυο διαφημίσεων. [38]

Ο προγραμματιστής για να ενσωματώσει τις διαφημίσεις στην εφαρμογή του, παρέχει μια περιοχή, στην οποία θα προβάλλονται οι διαφημίσεις και συνδέει τα στοιχεία της εφαρμογής των διαφημίσεων με τον λογαριασμό του έτσι, ώστε να λαμβάνει τη πληρωμή του. Η τελευταία εξαρτάται από το πλήθος των χρηστών που είδαν τη διαφήμιση και την επέλεξαν,

ακολουθώντας το μοντέλο «click per ad». Στο λειτουργικό σύστημα αυτό το χαρακτηριστικό παρέχεται μέσω του αντικειμένου WebView. Όπως αναφέραμε σε προηγούμενο κεφάλαιο, είναι ένας φυλλομετρητής που ενσωματώνεται στις εφαρμογές και μπορεί να προβάλλει τις διαφημίσεις με χρήση του πρωτοκόλλου HTTP.

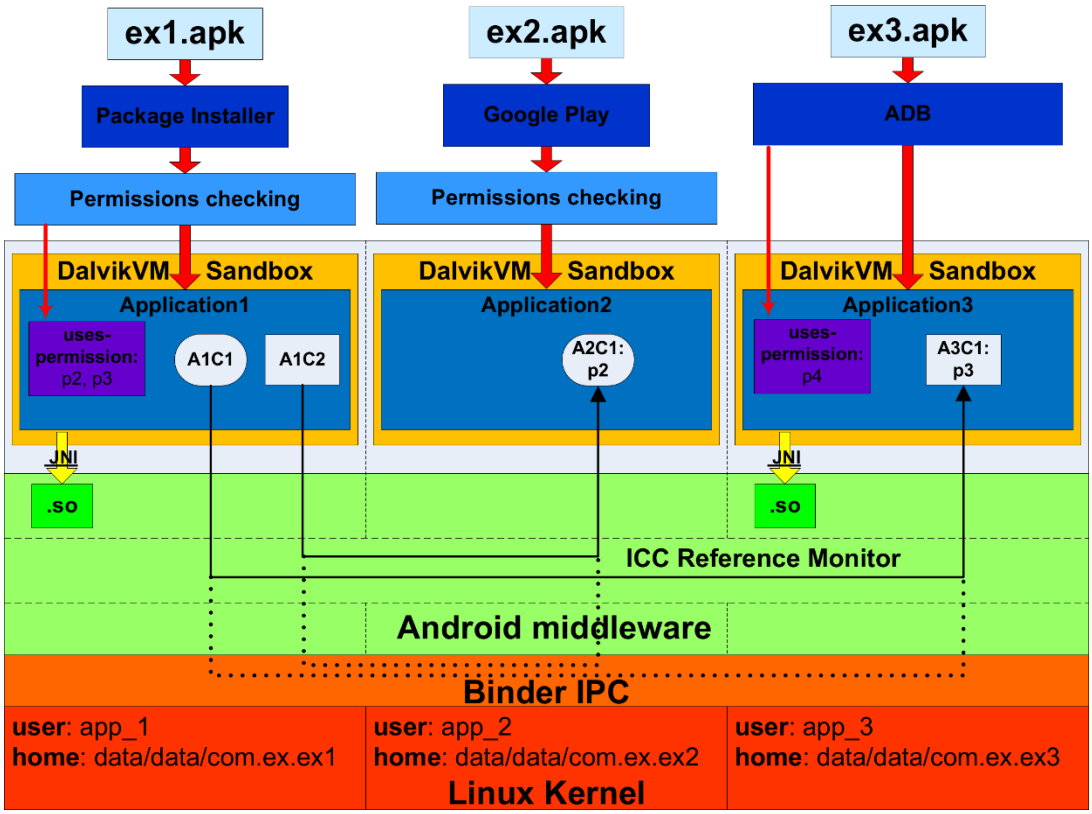
Όταν ένας χρήστης αποφασίζει να εγκαταστήσει μια εφαρμογή, ο πρώτος έλεγχος που πραγματοποιείται είναι στα δικαιώματα της. Σε περίπτωση που ο χρήστης αποδεχτεί τα δικαιώματα που αιτείται η εφαρμογή, τότε πραγματοποιείται η εγκατάσταση. Αξίζει να σημειωθεί, ότι στην τελευταία έκδοση του Android, ο χρήστης μπορεί να επιλέξει την ανάκληση των δικαιωμάτων της εφαρμογής μετά την εγκατάστασή της στη συσκευή. Παρ' όλα αυτά, ο χρήστης δεν μπορεί να προσδιορίσει, ποιο στοιχείο της συσκευής αιτείται την άδεια. Ωστόσο, ο κώδικας που εκτελείται από το παρεχόμενο WebView έχει τα ίδια δικαιώματα με την εφαρμογή.

Επίσης ο Stevens et al. [51] βρήκε, ότι πολλές διαφημίσεις χρησιμοποιούν άδειες, οι οποίες δεν περιγράφονται στη λειτουργία τους, ενώ ο Grace et al[52] παρατήρησε, ότι μισές από τις βιβλιοθήκες διαφημίσεων ελέγχουν τα δικαιώματα που έχει η εφαρμογή για να τα εκμεταλλευτούν και να υποκλέψουν προσωπικές πληροφορίες.

6.2 Προτεινόμενη λύση

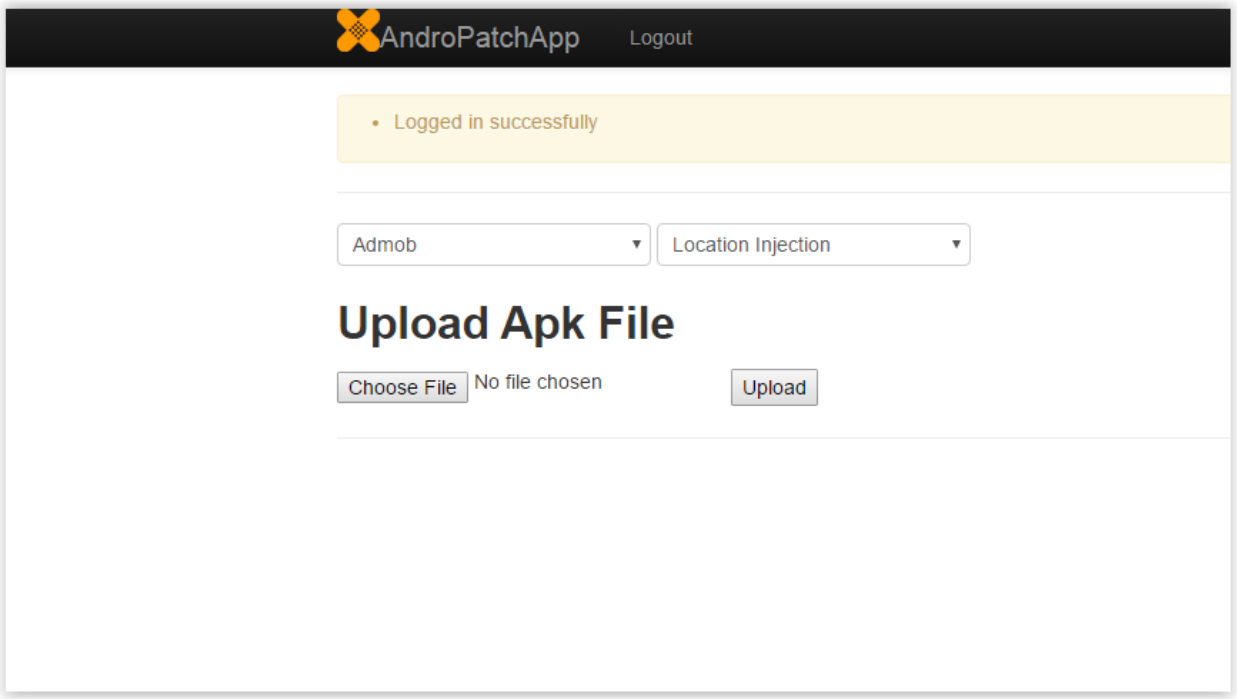
Το πεδίο εφαρμογής του AndroPatchApp είναι να παρέχει μια εναλλακτική λύση στο πρόβλημα χωρίς την ανάγκη παρέμβασης στο λειτουργικό σύστημα ή αλλαγής της αρχιτεκτονικής του. Πολλά από τα προβλήματα και τις απειλές που αναφέρθηκαν παραπάνω, κυρίως για τις περιπτώσεις των διαφημιστικών βιβλιοθηκών, μπορούν να επιλυθούν από τους προγραμματιστές, όπως για παράδειγμα να γίνει χρήση του πρωτοκόλλου στην επικοινωνία με τους διακομιστές του δικτύου. Επιπλέον, μια παρέμβαση στο λειτουργικό σύστημα απαιτεί εξειδικευμένη γνώση από τον χρήστη και μπορεί να εκθέσει τη συσκευή με μεγαλύτερο κίνδυνο.

Το AndroPatch, πριν την εγκατάσταση μιας εφαρμογής, παρεμβαίνει στον κώδικα του Dalvik και ενσωματώνει μια βιβλιοθήκη, η οποία μπορεί να αποτρέψει κακόβουλα αιτήματα. Το διάγραμμα ροής φαίνεται στην παρακάτω εικόνα :



Εικόνα 30: Μέθοδοι εγκατάστασης Android εφαρμογής. [38]

Ο χρήστης αρχικά, ανεβάζει την εφαρμογή του σε μορφή αρχείου apk. Το AndroPatchApp επεξεργάζεται το αρχείο apk και στη συνέχεια το παρέχει στον χρήστη, για να το κατεβάσει. Κατά τη διάρκεια της επεξεργασίας, το αρχείο αποσυμπιέζεται με το εργαλείο ApkTool[52], το οποίο επιστρέφει τον κώδικα της εφαρμογής σε αρχεία smali. Στη συνέχεια, το AndroPatchApp ελέγχει τον smali κώδικα για αναφορές στο αντικείμενο WebView. Όταν εντοπίσει το WebView ό,τι αποτελεί μέρος της διαφημιστικής βιβλιοθήκης, τότε τροποποιεί το smali κώδικα με τέτοιον τρόπο, ώστε να φορτωθεί η βιβλιοθήκη του AndroPatch, σύμφωνα με τις επιλογές του χρήστη.(εικόνα 31)



Εικόνα 31: AndroPatchApp

Αυτό που επιτυγχάνει το AndroPatchApp με την παρεμβολή του στον κώδικα της εφαρμογής στόχου είναι:

Απόκρυψη των συντεταγμένων του χρήστη : όπως έχει ήδη αναφερθεί, οι διαφημίσεις από όπου και αν προέρχονται, έχουν πρόσβαση στις συντεταγμένες του χρήστη, εφόσον στην εφαρμογή που τη φιλοξενεί έχουν εκχωρηθεί τα αντίστοιχα δικαιώματα. Για να αντιμετωπιστεί το πρόβλημα, ο κώδικας του AndroPatchApp που παρεμβαίνει, επιστρέφει μηδενική τιμή (0,0) όπως φαίνεται στην παρακάτω εικόνα:

```
function overrideLocation() {
                    navigator.geolocation.getCurrentPosition = function(success, failure){
                        success({ coords: {latitude: 0, longitude: 0,}, timestamp: Date.now() });
                    }
                    }
```

Εικόνα 32: Javascript , Μηδενισμός συντεταγμένων

Απόκρυψη διαθέσιμων πόρων: το AndoPatchApp μπορεί να αποκρύψει τους διαθέσιμους αισθητήρες της συσκευής, για παράδειγμα τη χρήση μικροφώνου και κάμερας. Αυτό επιτυγχάνεται με την παρεμβολή ενός script, το οποίο απενεργοποιεί αυτούς τους πόρους. Αυτή η προσέγγιση απενεργοποιεί πρακτικές που χρησιμοποιήθηκαν σε παρόμοιες περιπτώσεις, όπως η διαφημιστική βιβλιοθήκη “Silver Push”.

```
function overrideMedia() {
                    navigator.webkitGetUserMedia = function(constraints,successCallback, errorCallback){
                           constraints({video: false, audio:false });}
                    }
```

Εικόνα 33: Javascript , απενεργοποίηση λειτουργίας μικροφώνου και κάμερας

Απενεργοποίηση Javascript: το AndropatchApp μπορεί να επιτρέψει μόνο σε στατικό περιεχόμενο να προβληθεί με την απενεργοποίηση της Javascript σε όλα τα WebView αντικείμενα που χρησιμοποιούνται στις βιβλιοθήκες διαφημίσεων. Αυτό επιτυγχάνεται θέτοντας την τιμή *false* στη μέθοδο *setJavascriptEnabled*.

Αφαίρεση περιεχομένου διαφημίσεων: Τελικά, το AndroPatchApp μπορεί να αφαιρέσει το σύνολο του περιεχομένου των διαφημίσεων, τροποποιώντας τον κώδικα HTML που φορτώνεται στο αντικείμενο WebView.

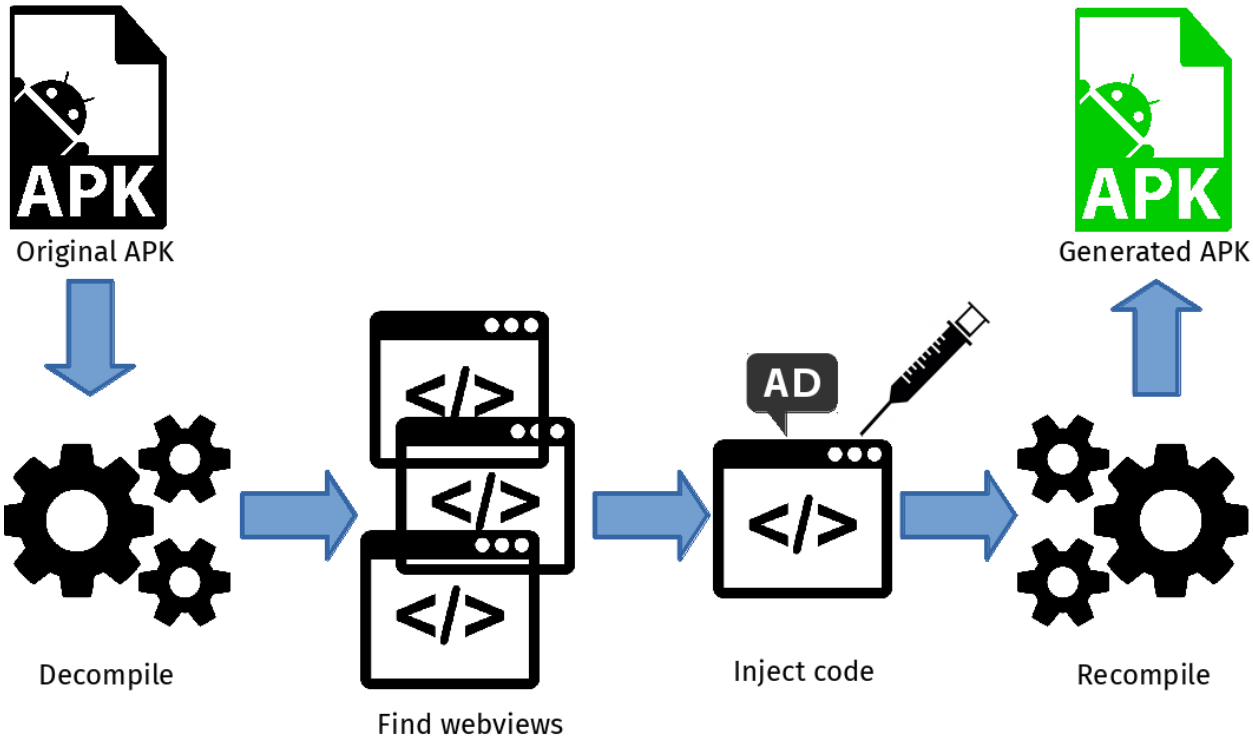
```
function overrideAd() {
                           document.body.parentNode.innerHTML = "";


}
```

Εικόνα 34: Javascript , απενεργοποίηση HTML περιεχομένου

Καθώς το AndropatchApp παρεμβαίνει μόνο στο WebView της διαφημιστικής βιβλιοθήκης, δεν επηρεάζει τη ροή των δεδομένων ή άλλα μέρη του κώδικα της εφαρμογής. Γι' αυτό τον λόγο, η λειτουργικότητα της εφαρμογής παραμένει ίδια. Με αυτό τον τρόπο δε θα παρουσιαστεί σφάλμα κατά τη διαδικασία της επαναπακετοποίησης της εφαρμογής. Συν τοις άλλοις, η όλη διαδικασία πραγματοποιείται αυτόματα και δεν απαιτείται η επίβλεψη από τον χρήστη.

Το AndroPatchApp χρησιμοποιεί το *apktool* για την πακετοποίηση του τροποποιημένου apk και το παραδίδει στον χρήστη υπογεγραμμένο και έτοιμο για εγκατάσταση.



Εικόνα 35: Ροή AndroPatchApp. [38]

Το AndroPatchApp για να παρεμβάλει τον κώδικά του στο webview, εκμεταλλεύεται τη μέθοδο *onPageFinished*. Η μέθοδος αυτή ενημερώνει την εφαρμογή, ότι ολοκληρώθηκε η φόρτωση του κώδικα, ότι το αντικείμενο HTML έχει δημιουργηθεί και κατά συνέπεια, μπορεί να φορτωθεί ο κώδικας Javascript του AndroPatchApp.

Επιπλέον, μπορεί να αλλάξει τη συμπεριφορά του webview, θέτοντας τις τιμές των μεθόδων *setAllowUniversIAccessFromFileUrls, setAllowFileAccessFromFileURLs* και *setJavaScriptEnabled* από *true* σε *false*. Αξίζει να σημειωθεί, ότι όλες οι παραπάνω μέθοδοι δίνουν πρόσβαση σε πόρους του συστήματος, στους οποίους οι διαφημιστικές βιβλιοθήκες δεν πρέπει να έχουν πρόσβαση. Για παράδειγμα, το AndroPatchApp εντοπίζει στα αρχεία smali του *decompile* apk την υπο εξέταση μέθοδο:

```
invoke-virtual {v2, v6}, Landroid/webkit/WebSettings;->setJavaScriptEnabled(Z)V
```

Η τιμή της παραπάνω μεθόδου ορίζεται από τη μεταβλητή *v6,* στην οποία εκχωρείται η τιμή 0x0, πριν γίνει η κλήση της μεθόδου από τη βιβλιοθήκη.

Ένα μέρος του κώδικα φαίνεται στην παρακάτω εικόνα:

```
static_code_injection ='    const/4 '
control_webview_value_false=' 0x0'
```

```
1   myWebView.setWebViewClient(new WebViewClient() {
2
3   @Override
4   public void onPageFinished(WebView view, String url)
5   String js_loc = "var newscript = document.createElement(\"
    script\");";
6   js_loc = js_loc + "newscript.onload=function(){
    overrideLocation();};";
```

Listing 1.1: Overriding onPageFinished.

```
1   function overrideLocation() {
2       navigator.geolocation.getCurrentPosition = function(
    success, failure) {
3           success({
4               coords: {
5                   latitude: 0,
6                   longitude: 0,
7               },
8               timestamp: Date.now()
9           });
10      }
11  }
```

Listing 1.2: Overriding location.

Εικόνα 36: Overriding Location Injection. [38]

Συγκρίνοντάς το με το Adsplit, το AndroPatchApp έχει πολλά επιπλέον χαρακτηριστικά. Αρχικά, μπορεί να εφαρμοστεί σε οποιαδήποτε διαφημιστική βιβλιοθήκη με ελάχιστο κόστος. Ενώ το Adsplit προσπαθεί να εντοπίσει τις API κλήσεις και να τις αντικαταστήσει, το AndroPatch ενσωματώνει τον κώδικά του στην πηγαία εφαρμογή. Επιπλέον, το Adsplit για να παράσχει τη λειτουργικότητά του, προσθέτει επιπλέον φόρτο στην εφαρμογή με αποτέλεσμα να αυξάνεται η απαιτούμενη υπολογιστική ισχύς και να μειώνεται ο χρόνος λειτουργίας της συσκευής. Αντιθέτως, με το AndroPatch δεν επηρεάζει τη λειτουργία της εφαρμογής σε σχέση με τη συσκευή. Τέλος και πιο σημαντικό είναι το γεγονός, ότι το AndroPatchApp αποτελεί μια *standalone* λύση, χωρίς να υπάρχει κάποια εξάρτηση ή τροποποίηση στον τρόπο με τον οποίο εγκαθίστανται οι εφαρμογές στο Android. Σε αντίθεση, το Adsplit εξαρτάται από το Quire[53] , το οποίο απαιτεί την τροποποίηση της πλατφόρμας Android. Γι' αυτό τον λόγο αποτελεί μια ευέλικτη προσέγγιση δεδομένου, ότι απαιτεί ελάχιστη αλληλεπίδραση με τον χρήστη.

Πίνακας 1: AndroPatchApp Ads library injection. [38]

AdMob	Flurry	InMobi
TapJoy	MobClix	ChartBoost
AdWhirl	MoPub	GreyStripe

7 Συμπεράσματα

Με την ολοένα και πιο συχνή χρήση των κινητών συσκευών , οι κατασκευαστές εξοπλίζουν τα έξυπνα τηλέφωνα με ένα σύνολο αισθητήρων. Αυτό έχει ως αποτέλεσμα τα δίκτυα διαφημίσεων να γίνονται περισσότερο «επιθετικά» απέναντι στον χρήστη. Αξιοποιώντας την ικανότητα τους να συλλέγουν, όσον το δυνατόν, περισσότερα δεδομένα από τον χρήστη, έχουν αυξηθεί δραματικά τα αιτήματα χορήγησης δικαιωμάτων σε εφαρμογές που ενσωματώνουν βιβλιοθήκες διαφημίσεων στο κώδικά τους. Η τάση αυτή φαίνεται να έχει μεγάλη απήχηση στους χρήστες, οι οποίοι, τις περισσότερες φορές, δεν αναλογίζονται τους πιθανούς κινδύνους που ενέχει η χορήγηση τέτοιων αδειών σε τρίτους για τα προσωπικά τους δεδομένα .

Το AndroPatchApp, είναι ένα νέο εργαλείο, το οποίο ασχολείται με αυτό το πρόβλημα ενεργώντας στο αρχείο APK της εφαρμογής, χωρίς να θέτει σε κίνδυνο τη λειτουργικότητά της. Πιο συγκεκριμένα, το AndroPatchApp επεξεργάζεται το αρχείο APK για να το μετατρέψει σε κώδικα smali της εικονικής μηχανής Dalvik και ενθυλακώνει τον κώδικά του στο WebView αντικείμενο που ελέγχεται από τις διαφημιστικές βιβλιοθήκες. Αυτό επιτρέπει στο χρήστη να επιλέξει τον τρόπο με τον οποίο θα λειτουργήσει το AndroPatchApp, από το να αφαιρέσει τις συντεταγμένες που εμφανίζονται στο αντικείμενο Webview μέχρι την πλήρη αφαίρεση του περιεχομένου. Όλα τα παραπάνω επιτυγχάνονται χωρίς να επηρεάζεται η εμπειρία του χρήστη καθώς δεν απαιτεί επιπλέον υπολογιστική ισχύ ή τεχνικές γνώσεις από αυτόν.

Παρ’ όλα αυτά, το μεγαλύτερο μειονέκτημα σε αυτή τη λύση είναι οι νομικές πτυχές. Από τη μια πλευρά, οι προγραμματιστές δεν επιθυμούν κάποιος τρίτος να επεμβαίνει στον κώδικα της εφαρμογής τους, καθώς αυτό θα έχει ως αποτέλεσμα να μειωθούν τα έσοδά του από τις διαφημίσεις. Από την άλλη πλευρά, ο χρήστης δεν επιθυμεί να μοιραστεί τις ευαίσθητες πληροφορίες του με άλλους. Στην πραγματικότητα, τέτοιοι όροι είναι τεκμηριωμένοι και αναφέρονται στους όρους και προϋποθέσεις κατά την εγκατάσταση της εφαρμογής. Εάν ο χρήστης γνώριζε, ότι με την αποδοχή των όρων χρήσης τα δεδομένα του θα εκτεθούν σε τρίτους, είναι πολύ πιθανό, ότι δε θα επέλεγε την εγκατάσταση της εφαρμογής. Ως εκ τούτου, το AndroPatchApp μπορεί να θεωρηθεί ένα μέσο για την επιλογή των πληροφοριών που θα μοιράζεται με τα δίκτυα διαφημίσεων.[38]

Βιβλιογραφία

1. Android Open Source Projec.Available from: https://source.android.com/
2. Nikolay Elenkov. Android Security Internals: an In-Depth Guide to android's security architecture, 2015 San Francisco , no starch press
3. Bahman Rashidiand Carol Fung Virginia Commonwealth University, Richmond, Virginia, USA. A Survey of Android Security Threats and Defenses. 2015.
4. P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. Gaur, M. Conti, and R. Muttukrishnan, "Android security: A survey of issues, malware penetration and defenses," 2015. [Online]. Available: http://dx.doi.org/10.1109/COMST.2014.2386139
5. C. Efstratiou and I. Leontiadis, "What is the price of free," Online; accessed at April 17, 2012, http://www.cam.ac.uk/research/news/what-is-the-price-of-free
6. S. Gunasekera, Android Apps Security, 1st ed. Berkely, CA, USA: Apress, 2012
7. L.Davi,A.Dmitrienko,A.-R.Sadeghi,and M.Winandy,"Privilegeescalationattacksonandroid,"inProc. of the 13th Information Security Conferenec (ISC'11), Boca Raton, Florida, USA, LNCS, M. Burmester, G. Tsudik, S. Magliveras, and I. Ilic, Eds., vol. 6531. Springer Berlin Heidelberg, October 2011, pp. 346–360. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-18178-8
8. H. Huang, S. Zhu, P. Liu, and D. Wu, "A framework for evaluating mobile app repackaging detection algorithms," in Proc. of the 6th International Conference on Trust and Trustworthy Computing (TRUST'13), London, UK, LNCS, M. Huth, N. Asokan, S. ˇCapkun, I. Flechais, and L. ColesKemp, Eds., vol. 7904. Springer Berlin Heidelberg, June 2013, pp. 169–186. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38908-5
9. W. Zhou, Y. Zhou, X. Jiang, and P. Ning, "Detecting repackaged smartphone applications in third-party android marketplaces," in Proc. of the 2nd ACM Conference on Data and Application Security and Privacy (CODASPY'12), San Antonio, Texas, USA. ACM,March2012,pp.317–326.[Online].Available: http://doi.acm.org/10.1145/2133601.2133640
10. E. Kovacs, "Wi-fi direct flaw exposes android devices to dos attacks," Online; accessed at July 8, 2015, http://www.securityweek.com/wi-fi-direct-flaw-exposes-android-devices-dos-attacks.
11. C. Marforio, A. Francillon, and S. Capkun, Application Collusion Attack on the Permission-Based Security Model and Its Implications for Modern Smartphone Systems. Department of Computer Science, ETH Zurich, 2010. [Online]. Available: https://books.google.com/books?id=nvszMwEACAAJ
12. Fernando Ruiz. 'fakeinstaller' leads the attack on android phones. Available from: http://blogs.mcafee.com/mcafee-labs/ fakeinstaller-leads-the-attack-on-android-phones.AccessedNovember12, 2013
13. Lookout Mobile Security. State of Mobile Security, 2012. Available from: https:// www.lookout.com/static/ee_images/lookout-state-of-mobile-security-2012.pdf.
14. McAfee. What is a "drive-by" download? Available from: http://blogs.mcafee. com/consumer/drive-by-download.AccessedNovember29,2013.
15. Lookout. Security alert: Hacked websites serve suspicious android apps (notcompatible). Available from: https://blog.lookout.com/blog/2012/05/02/ security-alert-hacked-websites-serve-suspicious-android-apps-noncompatible/. Accessed November 29, 2013.

16. F-Secure Labs. Mobile threat report q3 2013. Technical report, 2013. Available from: http://www.f-secure.com/static/doc/labs_global/Research/Mobile_ Threat_Report_Q3_2013.pdf.

17. Carlos A. Castillo. Android malware past, present, and future. Technical report, McAfee Mobile Security Working Group, 2011. Available from: http://www.mcafee.com/us/resources/white-papers/ wp-android-malware-past-present-future.pdf.

18. Dr Giles Hogben, Dr Marnix Dekker. Smartphone Security. Technical report, ENISA, 2010. Available from: http://www. enisa.europa.eu/activities/identity-and-trust/risks-and-data-breaches/ smartphones-information-security-risks-opportunities-and-recommendations-for-users.

19. GDataSecurityLabs.Mobilemalwarereport-half-yearreportjuly-december2013. Technical report, 2014. Available from: https://public.gdatasoftware.com/Presse/ Publikationen/Malware_Reports/GData_MobileMWR_H2_2013_EN.pdf.

20. DSLReports.com. What is a botnet trojan? Available from: http://www. dslreports.com/faq/14158,April2009.AccessedApril9,2014.

21. Lookout. Security alert: Droiddream malware found in official android market. Available from: https://blog.lookout.com/blog/2011/03/ 01/security-alert-malware-found-in-official-android-market-droiddream/,March 2011. Accessed October 13, 2013.

22. Lookout. Technical analysis droiddream malware. Available from: https://blog. lookout.com/droiddream/,March2011.AccessedOctober13,2013

23. Xuxian Jiang Michael Grace, Wu Zhou and Ahmad-Reza Sadeghi. Unsafe Exposure Analysis of Mobile In-App Advertisements. In WISEC ’12 Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks, 2012

24. AVG. Android/dgen plankton a. Available from: http://www.avgthreatlabs.com/ virus-and-malware-information/info/android-dgen-plankton-a/.AccessedApril 8, 2014.

25. Lookout. The bearer of badnews. Available from: https://blog.lookout.com/blog/ 2013/04/19/the-bearer-of-badnews-malware-google-play/,April2013.Accessed October 14, 2013.

26. Android Developers. Webview. Available from: http://developer.android.com/ reference/android/webkit/WebView.html.AccessedDecember1,2013.

27. Wenliang Du Yifei Wang Tongbo Luo, Hao Hao and Heng Yin. Attacks on WebView in the Android System. In ACSAC ’11 Proceedings of the 27th Annual Computer Security Applications Conference, 2011

28. AV-Comparatives. Mobile security review. Technical report, August 2013. Available from: http://www.av-comparatives.org/wp-content/uploads/2013/08/avc_ mob_201308_en.pdf.

29. CUPP Computing. The micro sd format hardware security form cupp computing. Technical report, May 2014. Version 0.9

30. Snort. Snort users manual. Technical report, April 2014.

31. Snort. What is a snort-rule. Available from: https://github.com/vrtadmin/ snort-faq/blob/master/Rules/What-is-a-Snort-rule.md.AccessedMay13,2014.

32. Timothy Prickett Morgan v/The Register. Arm holdings eager for pc and server expansion. Available from: http://www.theregister.co.uk/2011/02/01/ arm_holdings_q4_2010_numbers/,February2011.AccessedMay12,2014

33. Apple. ios security. Technical report, February 2014. Available from: http: //images.apple.com/ipad/business/docs/iOS_Security_Feb14.pdf

34. Paul Pearce, Adrienne Porter Felt, Gabriel Nunez, and David Wagner. Addroid: Privilege separation for applications and advertisers in android. Technical report, EECS Department, University of California, Berkeley, May 2013. Available from: http://www.cs.berkeley.edu/~pearce/papers/addroid_asiaccs_2012.pdf.

35. Yuliy Pisetsky Anhei Shu Dan S. Wallach Michael Dietz, Shashi Shekhar. Quire: Lightweight provenance for smart phone operating systems. Technical report, February2011. Availablefrom: https://www.usenix.org/legacy/event/sec11/tech/ full_papers/Dietz7-26-11.pdf.

36. Android Developers. Build.version_codes. Available from: http: //developer.android.com/reference/android/os/Build.VERSION_CODES. html#JELLY_BEAN.AccessedMay22,2014.

37. MWR Labs. Adventures with android webviews. Available from: https://labs. mwrinfosecurity.com/blog/2012/04/23/adventures-with-android-webviews/.Accessed May 31, 2014

38. V. Tsiakos and C. Patsakis: “AndroPatchApp: Taming Rogue Ads in Android”, International Conference on Mobile, Secure and Programmable Networking (MSPN'2016), Paris, France, June 1-3, 2016.

39. J. Hoffman-Andrews, Verizon injecting perma-cookies to track mobile customers, bypassing privacy controls, https://www.eff.org/deeplinks/2014/11/verizonx-uidh (2014)

40. E. Mills, Malware delivered by yahoo, fox, google ads, http://www.cnet.com/ news/malware-delivered-by-yahoo-fox-google-ads/ (2010).

41. K. Thomas, E. Bursztein, C. Grier, G. Ho, N. Jagpal, A. Kapravelos, D. McCoy, A. Nappa, V. Paxson, P. Pearce, et al., Ad injection at scale: Assessing deceptive advertisement modifications, in: Security and Privacy (SP), 2015 IEEE Symposium on, IEEE, 2015, pp. 151–167

42. R. Graham, Extracting the superfish certificate, http://blog.erratasec.com/ 2015/02/extracting-superfish-certificate.html (2015).

43. W. Enck, D. Octeau, P. McDaniel, S. Chaudhuri, A study of android application security, in: Proceedings of the 20th USENIX conference on Security, USENIX Association, 2011, pp. 21–21

44. L. Davi, A. Dmitrienko, A.-R. Sadeghi, M. Winandy, Privilege escalation attacks on android, in: Information Security, Springer, 2011, pp. 346–360

45. C. Orthacker, P. Teufl, S. Kraxberger, G. Lackner, M. Gissing, A. Marsalek, J. Leibetseder, O. Prevenhueber, Android security permissions–can we trust them?, in: Security and Privacy in Mobile Information and Communication Systems, Springer, 2012, pp. 40–51

46. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, D. Wagner, Android permissions: User attention, comprehension, and behavior, in: Proceedings of the Eighth Symposium on Usable Privacy and Security, ACM, 2012, p. 3.

47. W. Enck, M. Ongtang, P. McDaniel, Understanding android security, IEEE security & privacy 7 (1) (2009) 50–57.

48. P. Felt, E. Chin, S. Hanna, D. Song, D. Wagner, Android permissions demystified, in: Proceedings of the 18th ACM conference on Computer and communications security, ACM, 2011, pp. 627–638

49. M. C. Grace, Y. Zhou, Z. Wang, X. Jiang, Systematic detection of capability leaks in stock android smartphones., in: NDSS, 2012.

50. S. Poeplau, Y. Fratantonio, A. Bianchi, C. Kruegel, G. Vigna, Execute this! analyzing unsafe and malicious dynamic code loading in android applications, in: 21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014, The Internet Society, 2014.

51. R. Stevens, C. Gibler, J. Crussell, J. Erickson, H. Chen, Investigating user privacy in android ad libraries, in: Proceedings of the 2012 Workshop on Mobile Security Technologies (MoST), 2012.

52. M. C. Grace, W. Zhou, X. Jiang, A.-R. Sadeghi, Unsafe exposure analysis of mobile in-app advertisements, in: Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks, WISEC '12, ACM, 2012, pp. 101– 112

53. R. Winsniewski, Android–apktool: A tool for reverse engineering android apk files, http://ibotpeaches.github.io/Apktool/ (2012).

54. M. Dietz, S. Shekhar, Y. Pisetsky, A. Shu, D. S. Wallach, Quire: Lightweight provenance for smart phone operating systems, in: Proceedings of the 20th USENIX Conference on Security, SEC'11, USENIX Association, Berkeley, CA, USA, 2011, pp. 23–23

55. Fredrik Bugge Lyche and Jørgen Haukedal Lytskjold, Improving security solutions in modern smartphones. Norwegian University of Science and Technology, 2014

56. T.Book, Dan S. Wallach, A Case of Collusion: A Study of the Interface Between Ad Libraries and their Apps, Rice University, 23 Jul 2013

ΠΑΡΑΡΤΗΜΑ Α – AndroP@tchAppCode Main Body

```
# -*- coding: utf-8 -*-
import argparse
from argparse import RawTextHelpFormatter
import json
import logging
import os
import re
import subprocess
import sys
from time import sleep
from shutil import copyfile
from termcolor import colored


# Initialise the logger:
logger = logging.getLogger("Droid")
logger.setLevel(logging.DEBUG)
ch = logging.StreamHandler()  # console handler for the logger
ch.setLevel(logging.INFO)
ch.setFormatter( logging.Formatter("  >> %(name)s: [%(levelname)s] %(message)s") )  # log message format
logger.addHandler(ch)  # add the handler to the logger

APKTOOL = "../Apktool/apktool"
SIGNLIB = "../lib/signapk.jar"
CAKEY = '../keys/certificate.pem'
USKEY = '../keys/key.pk8'
SOURCE_PATCH = "../source"
SOURCE_FILE_1 = "JSinject.smali"
SOURCE_FILE_2 = "BuildConfig.smali"
SOURCE_FILE_3 = "JSinject$1.smali"
SOURCE_FILE_4 = 'LocJSinject.smali'
SOURCE_FILE_5 = 'AdJSinject.smali'
SOURCE_FILE_6 = 'htmlReplace.smali'
```

```
SOURCE_FILE_7 = 'injectRepRemov.smali'


###MAP 1-1 between VUL_COD & PYTHON Script

#Vulnurable code methods
VUL_CODE_1 = '\'.*iget-object.*android/webkit/WebView;\''
VUL_CODE_2 = '\'.*setAllowFileAccessFromFileURLs*\''
VUL_CODE_3 = '\'.*setAllowUniversalAccessFromFileURLs*\''
VUL_CODE_4 = '\'.*setJavaScriptEnabled*\''
#VUL_CODE = [" 'setJavaScriptEnabled' "," 'setAllowFileAccess' "," 'setAllowFileAccessFromFileURLs' "," 'shouldOverrideUrlLoading' "]
#Script Suppport
PYTHON_SCRIPT_1 ='../libinject.py'
PYTHON_SCRIPT_2 ='../inject_FileAccessURL.py'
PYTHON_SCRIPT_3 ='../inject_setUniversalAccessFromurl.py'
PYTHON_SCRIPT_4 ='../inject_setJVen.py'

ADNETWORK_LIST =" 'com/google/ads' "
adnet1  = " 'com/google/ads' "
adnet2 =  " 'com/flurry'     "
adnet3 =  " 'com/inmobi'     "
adnet4 =  " 'com/tapjoy'     "
adnet5 =  " 'com/mobclix'    "
adnet6 =  " 'com/chartboost' "
adnet7 =  " 'com/adwhirl'    "
adnet8 =  " 'com/mopub'      "
adnet9 =  " 'com/greystripe' "
adnet10= " 'com/google/ads|com/greystripe|com/flurry|com/inmobi|com/tapjoy|com/mobclix|com/chartboost|com/adwhirl|com/mopub|com/greystripe|com/google/android/gms' "

def main(argv=None):

    # Retrieve command-line parameters:
    parser = argparse.ArgumentParser(description="Droid description: \n"
                                     "  >> %(prog)s /path/to/file.apk\n"
                                     "  >> %(prog)s --no-string-processing /path/to/file.apk\n"
                                     "  >> %(prog)s /path/to/file.apk --emextract\n"
                                     "  >> %(prog)s --no-string-processing /path/to/file.apk --extract\n"
                                     "  >> %(prog)s /path/to/file.apk --extract /dir/where/to/save/APK/info/\n"
                                     "  >> %(prog)s --help",
                          formatter_class=RawTextHelpFormatter)
```

```
parser.add_argument("target", metavar="TARGET_FILE", type=str, help="The targeted APK package to analyse.")
parser.add_argument("-e", "--extract", type=str, nargs="?", const="./", action="store", dest="extract_to_directory", help="Extract and store all the APK entries as well as it info in a given folder (default: ./APK).")
parser.add_argument("-m", "--mode", type=str, nargs="?", const="./", action="store", dest="mode_injection", help="Choose mode of Injection.'locInject|adInject|adRequest' ")
parser.add_argument("-n", "--netad", type=str, nargs="?", const="./", action="store", dest="adnet_injection", help="Choose Ad Network of Injection.'admob|flurry|inmobi|tapjoy|chartboost|adwhirl|flury|greystripe' ")

args = parser.parse_args()
print colored ('Procedure Started.....','green')
print colored('Filename :' +  args.target,'green')
# Check mode Injection
modeInject = args.mode_injection
modeInject = modeInject.strip(' ')
adnet = args.adnet_injection
adnet = adnet.strip(' ')

if   (adnet == "admob"):
      ADNETWORK_LIST = adnet1
elif (adnet  == "flurry"):
      ADNETWORK_LIST = adnet2
elif (adnet == "inmobi"):
      ADNETWORK_LIST = adnet3
elif (adnet == "tapjoy"):
      ADNETWORK_LIST = adnet4
elif (adnet == "mobcli"):
      ADNETWORK_LIST = adnet5
elif (adnet == "chartboost"):
      ADNETWORK_LIST = adnet6
elif (adnet == "adwhirl"):
      ADNETWORK_LIST = adnet7
elif (adnet == "mopub"):
      ADNETWORK_LIST = adnet8
elif (adnet == "greystripe"):
      ADNETWORK_LIST = adnet9
elif (adnet == "allAdNet"):
      ADNETWORK_LIST = adnet10
else:
      pass
if (modeInject == "locInject"):
```

```
            print colored('Mode Injection: ' + modeInject,'green')
            SOURCE_INJCT = SOURCE_FILE_4
        elif (modeInject == "adInject"):
            print colored('Mode Injection: ' + modeInject,'green')
            SOURCE_INJCT = SOURCE_FILE_5
        elif (modeInject == "adRequest"):
            print colored('Mode Injection: ' + modeInject,'green')
            SOURCE_INJCT = SOURCE_FILE_6
        elif (modeInject == "adReplRem"):
            print colored('Mode Injection: ' + modeInject,'green')
            SOURCE_INJCT = SOURCE_FILE_7
        else:
            print colored ('Wrong mode selection', 'red')
            exit()

        # Extract the APK file name (without extension):
        filename = os.path.basename(args.target)
        if re.search("\.apk", args.target, re.IGNORECASE):
            filename = str(filename[0:-4])

        if args.extract_to_directory is not None:
        # Extract all the APK entries and info to a given directory:
            dir = args.extract_to_directory

            # Check whether no output directory was given (i.e. the default one):
            if args.extract_to_directory == "./":
                dir += filename
        decompileApk(dir, args.target, filename)
        NUM=0

injection_code0(dir,VUL_CODE_1,PYTHON_SCRIPT_1,ADNETWORK_LIST,NUM,adnet)

injection_code0(dir,VUL_CODE_2,PYTHON_SCRIPT_2,ADNETWORK_LIST,NUM,adnet)

injection_code0(dir,VUL_CODE_3,PYTHON_SCRIPT_3,ADNETWORK_LIST,NUM,adnet)

injection_code0(dir,VUL_CODE_4,PYTHON_SCRIPT_4,ADNETWORK_LIST,NUM,adnet)
        injectLib(dir, SOURCE_PATCH, SOURCE_FILE_1, SOURCE_FILE_2,
SOURCE_FILE_3,SOURCE_INJCT)
        patchname = compileApk(dir, args.target, filename)
        signApk   = signapk(patchname, dir, SIGNLIB,CAKEY,USKEY)


    def injectLib(output_dir, path, file1, file2, file3,fileInjct):
        command =  output_dir + '/' + 'smali' + '/' + 'com/vasts'
```

```
        command1 = output_dir + '/' + 'smali' + '/' + 'com'
        if (os.path.isdir(output_dir + '/' + 'smali' + '/' + 'com')):
            os.makedirs(command)
        else:
            os.makedirs(command1)
            os.makedirs(command)

        src0 = path + '/' + fileInjct
        dst0 = path + '/' + file3
        copyfile(src0, dst0)


        src1 = path + '/' + file1
        dst1 = command + '/' + file1
        copyfile(src1, dst1)

        src2 = path + '/' + file2
        dst2 = command + '/' + file2
        copyfile(src2, dst2)

        src3 = path + '/' + file3
        dst3 = command + '/' + file3
        copyfile(src3, dst3)

    def injection_code0(output_dir,vul_code,supp_script,ad_list,num,adnet):
        path_file = "/tmp/inject%s.log" % (num)
        #print "Dont forget to delete manually file %s" %path_file
        open(path_file, 'a').close()
        code = vul_code
        inscript = supp_script
        find_grep = subprocess.Popen('which grep' , stdout=subprocess.PIPE, stderr=None,
shell=True)
        grep, err = find_grep.communicate()
        grep = grep.strip(' \t\n\r')

        find_egrep = subprocess.Popen('which egrep' , stdout=subprocess.PIPE,
stderr=None, shell=True)
        egrep, err = find_egrep.communicate()
        egrep = egrep.strip(' \t\n\r')

        find_python = subprocess.Popen('which python' , stdout=subprocess.PIPE,
stderr=None, shell=True)
```

```
        python, err = find_python.communicate()
        python      = python.strip(' \t\n\r')

        def run(cmd, logfile):
            with open(logfile, 'w' ) as fout:
                p = subprocess.Popen(cmd, stdout=fout, stderr=None, shell=True)
                ret_code = p.wait()
                return ret_code

        # define our method
        def replace_all(text, dic):
            for i, j in dic.iteritems():
                text = text.replace(i, j)
            return text

        def replace_grep(adlist,grep,egrep,adnet):
          if ( adnet == "allAdNet"):
            return egrep
          else :
            return grep


        reps = { '$' :'\$'}
        command = grep + ' -ir ' + ' ' + code + ' '+   output_dir + '/' + 'smali' +   ' | ' +
replace_grep(ad_list,grep,egrep,adnet) + ' -i ' + ad_list  +   " | awk -F ':' '{print$1}'  | sort -u "
        print colored('Injection Procedure Started.... ','green')
        print colored("Inject lib %s " %ad_list,'blue')
        print colored('Command: %s ' %command ,'red')
        status = run(command,path_file)


        with open(path_file, 'r+' ) as input_path:
            for path in input_path.readlines():
                path = replace_all(path, reps)
                p  =  subprocess.Popen(python    +  '  '  +  inscript  +  '  '  +  path          ,
stdout=subprocess.PIPE, stderr=None, shell=True)
                inf_file, err = p.communicate()
                path = path.strip(' \t\n\r')
        input_path.close()
        print colored ('Injection Successfully  finished', 'green')


    def decompileApk(output_dir, filepath, filename ):
        print colored ("Decompile apk ...",'green')
```

```
        os.makedirs(output_dir)
        p = subprocess.Popen(APKTOOL + ' ' + 'd '+ '-f '+ '-o '+ output_dir + ' ' + filepath    , stdout=subprocess.PIPE, stderr=None, shell=True)
        out, err = p.communicate()
        #print out
        print colored(  "APK Successful Decompiled  in directory:  %s "  %(output_dir) , 'red')

    def compileApk(output_dir, filepath, filename ):
        print colored ("Recompile apk ...",'green')
        p = subprocess.Popen(APKTOOL + ' ' + 'b '+ '-f '+ '-o '+ 'patched.' + filename + '.apk' + ' ' + output_dir   , stdout=subprocess.PIPE, stderr=None, shell=True)
        out, err = p.communicate()
        patched_name = "patched.%s.apk" %(filename)
        #print "APK Successful Compiled \n" , "OUTput Directory: " + output_dir +  '\n' + 'Patched APK Filename :' + patched_name + '\n'
        print colored("APK Successful Compiled \n" ,  'green')
        patched_name = patched_name.strip(' \t\n\r')
        return patched_name

    def signapk( apkfilename , output_dir, SIGNLIB,CAKEY,USKEY):
        print colored ("Sign Apk", 'green')
        print colored (apkfilename, 'red')

        signapkfilename = "sign.%s" %(apkfilename)
        signapk = "java -jar %s %s %s %s %s " %(SIGNLIB,CAKEY,USKEY,apkfilename,signapkfilename)
        p = subprocess.Popen( signapk  , stdout=subprocess.PIPE, stderr=None, shell=True)
        out, err = p.communicate()

        if os.path.isfile(signapkfilename):
            print colored ("APK Successfully Signed: %s" %(signapkfilename) , 'red')
            p = subprocess.Popen( 'mv ' + signapkfilename + ' ../upload/'    , stdout=subprocess.PIPE, stderr=None, shell=True)
            out, err = p.communicate()
            delete_all(apkfilename, 'decompile')
            return signapkfilename

    def delete_all(del_patch, del_deco ):
    del_all = [ del_patch, del_deco]

    for x in del_all:
      pipe = subprocess.Popen("rm  -rf "  + x , stdout=subprocess.PIPE, stderr=None, shell=True)
```

```
if __name__ == "__main__":
    sys.exit(main())
```

ΠΑΡΑΡΤΗΜΑ Β – AndroP@tchAppCode Scripts

inject_setJVen.py

```
# Inject code to smali file for method setJavaScriptEnabled(true)
import sys
import re
import subprocess



smalifile = sys.argv[1]
outfile = smalifile + '.patched'
p = re.compile('^\.method')
regex = re.compile('.*setJavaScriptEnabled*')

text_array = []
with open(smalifile, 'r' ) as input_data, open(outfile,'w')  as outputfile:

    def varManipulaton( string ):
        static_code_injection ='    const/4 '
        control_webview_value_false=' 0x0\n\n'
        control_webview_value_true =' 0x1\n'
        m = re.search('{.*[0-9]}', string)
        both_variables = m.group(0)
        a , b = both_variables.split(',')
        variable1 = re.search('[a-w][0-9]',a)   #Contrunction of string
        variable2 = re.search('[a-w][0-9]',b)
        final_string_injection_false = static_code_injection + variable2.group(0) + ',' +
control_webview_value_false
        final_string_injection_true = static_code_injection + variable2.group(0) + ',' +
control_webview_value_true
        return       final_string_injection_false

    # Skips text before the beginning of the interesting block:
    for line in input_data:
        text_array.append(line)

    idexList = []
```

```
        for idx, val in enumerate(text_array):
            if (  regex.match(val)):
                a  = varManipulaton(val)
                print idx, val ,a
                idexList.append(int(idx))
                #text_array.insert(idx + 1,'\n')
                #text_array.insert(idx + 1,a)
                #text_array.insert(idx + 1,'\n')
        for num in idexList:
                text_array.insert(num -1,a)
                print "test %s" ,num

        for item in text_array:
            outputfile.write(item)
        print 'File  successfully created!!!'

    # define our method
    def replace_all(text, dic):
        for i, j in dic.iteritems():
            text = text.replace(i, j)
            return text
    print 'Smalif file : ' ,smalifile
    print 'OutFile : ' , outfile

    reps = { '$' :'\$'}
    new_outfile = outfile
    new_outfile = replace_all(new_outfile, reps)
    new_smalifile = smalifile
    new_smalifile = replace_all(new_smalifile, reps)

    print 'New Smalif file : ' ,new_smalifile
    print 'NEW OutFile : ' , new_outfile

    p = subprocess.Popen('mv ' + new_outfile + ' ' + new_smalifile   , stdout=subprocess.PIPE,
stderr=subprocess.PIPE, shell=True)
    out, err = p.communicate()
```

libinject.py

```
# Inject code to smali file for method setJavaScriptEnabled(true)
import sys
import re
import subprocess

smalifile = sys.argv[1]
outfile = smalifile + '.patched'
lookup = 'setJavascript'
p = re.compile('^\.method')
regex = re.compile('.*iget-object.*android/webkit/WebView;')

text_array = []
with open(smalifile, 'r' ) as input_data, open(outfile,'w')  as outputfile:

    def varManipulaton( string ):
        static_code_injection_1 ='    invoke-static {'
        static_code_injection_2                    ='},                    Lcom/vasts/JSinject;-
>setInject(Landroid/webkit/WebView;)V'
        control_webview_value_false=' 0x0\n\n'
        control_webview_value_true =' 0x1\n'
        m = re.search('[a-z][0-9]', string)
        both_variables = m.group(0)
        Final_inject = static_code_injection_1 + both_variables + static_code_injection_2
        return  Final_inject

    # Skips text before the beginning of the interesting block:
    for line in input_data:
        text_array.append(line)
    idexList = []
    for idx, val in enumerate(text_array):
        if (  regex.match(val)):
            a  = varManipulaton(val)
            print idx, val ,a
            text_array.insert(idx + 1,'\n')
            text_array.insert(idx + 1,a)
            text_array.insert(idx + 1,'\n')
    for item in text_array:
        outputfile.write(item)
```

```
        print 'File  successfully created!!!'

    # define our method
    def replace_all(text, dic):
        for i, j in dic.iteritems():
            text = text.replace(i, j)
            return text

    print 'Smalif file : ' ,smalifile
    print 'OutFile : ' , outfile

    reps = { '$' :'\$'}
    new_outfile = outfile
    new_outfile = replace_all(new_outfile, reps)
    new_smalifile = smalifile
    new_smalifile = replace_all(new_smalifile, reps)

    print 'New Smalif file : ' ,new_smalifile
    print 'NEW OutFile : ' , new_outfile

    p = subprocess.Popen('mv ' + new_outfile + ' ' + new_smalifile   , stdout=subprocess.PIPE,
stderr=subprocess.PIPE, shell=True)
    out, err = p.communicate()
```

inject_setUniversalAccessFromurl.py

```
    # Inject code to smali file for method setJavaScriptEnabled(true)
    import sys
    import re
    import subprocess



    smalifile = sys.argv[1]
    outfile = smalifile + '.patched'
    p = re.compile('^\.method')
    regex = re.compile('.*setAllowUniversalAccessFromFileURLs*')
    text_array = []

    with open(smalifile, 'r' ) as input_data, open(outfile,'w')  as outputfile:
        def varManipulaton( string ):
```

```
                static_code_injection ='    const/4 '
                control_webview_value_false=' 0x0\n\n'
                control_webview_value_true =' 0x1\n'

                m = re.search('{.*[0-9]}', string)
                both_variables = m.group(0)
                a , b = both_variables.split(',')
                variable1 = re.search('[a-w][0-9]',a)   #Contrunction of string
                variable2 = re.search('[a-w][0-9]',b)
                final_string_injection_false = static_code_injection + variable2.group(0) + ',' +
control_webview_value_false
                final_string_injection_true = static_code_injection + variable2.group(0) + ',' +
control_webview_value_true
                return        final_string_injection_false

        # Skips text before the beginning of the interesting block:
        for line in input_data:
                text_array.append(line)
        idexList = []
        for idx, val in enumerate(text_array):
                if (  regex.match(val)):
                        a  = varManipulaton(val)
                        print idx, val ,a
                        idexList.append(int(idx))
                        #text_array.insert(idx + 1,'\n')
                        #text_array.insert(idx + 1,a)
                        #text_array.insert(idx + 1,'\n')
        for num in idexList:
                        text_array.insert(num -1,a)
                        print "test %s" ,num

        for item in text_array:
                outputfile.write(item)
        print 'File  successfully created!!!'

    # define our method
    def replace_all(text, dic):
        for i, j in dic.iteritems():
                text = text.replace(i, j)
                return text

    print 'Smalif file : ' ,smalifile
    print 'OutFile : ' , outfile
```

```
reps = { '$' :'\$'}
new_outfile = outfile
new_outfile = replace_all(new_outfile, reps)
new_smalifile = smalifile
new_smalifile = replace_all(new_smalifile, reps)

print 'New Smalif file : ' ,new_smalifile
print 'NEW OutFile : ' , new_outfile

p = subprocess.Popen('mv ' + new_outfile + ' ' + new_smalifile  , stdout=subprocess.PIPE,
stderr=subprocess.PIPE, shell=True)
out, err = p.communicate()
```

inject_FileAccessURL.py

```
# Inject code to smali file for method setJavaScriptEnabled(true)
import sys
import re
import subprocess



smalifile = sys.argv[1]
outfile = smalifile + '.patched'
p = re.compile('^\.method')
regex = re.compile('.*setAllowFileAccessFromFileURLs*')

text_array = []



with open(smalifile, 'r' ) as input_data, open(outfile,'w')  as outputfile:

        def varManipulaton( string ):
                static_code_injection ='    const/4 '
                control_webview_value_false=' 0x0\n\n'
                control_webview_value_true =' 0x1\n'



                m = re.search('{.*[0-9]}', string)
```

```
                both_variables = m.group(0)
                a , b = both_variables.split(',')
                variable1 = re.search('[a-w][0-9]',a)   #Contrunction of string
                variable2 = re.search('[a-w][0-9]',b)
                final_string_injection_false = static_code_injection + variable2.group(0) + ',' +
control_webview_value_false
                final_string_injection_true = static_code_injection + variable2.group(0) + ',' +
control_webview_value_true
                return        final_string_injection_false

        # Skips text before the beginning of the interesting block:

        for line in input_data:
                text_array.append(line)

        idexList = []
        for idx, val in enumerate(text_array):
                if (  regex.match(val)):
                        a  = varManipulaton(val)
                        print idx, val ,a
                        idexList.append(int(idx))
        for num in idexList:
                        text_array.insert(num -1,a)
                        print "test %s" ,num

        for item in text_array:
                outputfile.write(item)
        print 'File  successfully created!!!'

    # define our method
    def replace_all(text, dic):
        for i, j in dic.iteritems():
                text = text.replace(i, j)
                return text
    print 'Smalif file : ' ,smalifile
    print 'OutFile : ' , outfile

    reps = { '$' :'\$'}
    new_outfile = outfile
    new_outfile = replace_all(new_outfile, reps)
    new_smalifile = smalifile
    new_smalifile = replace_all(new_smalifile, reps)

    print 'New Smalif file : ' ,new_smalifile
```

```
    print 'NEW OutFile : ' , new_outfile

    p = subprocess.Popen('mv ' + new_outfile + ' ' + new_smalifile   , stdout=subprocess.PIPE,
stderr=subprocess.PIPE, shell=True)
    out, err = p.communicate()
```