Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Τεχνολογίες Δικτύωσης στο Android** **Android Network Technologies**
Ονοματεπώνυμο Φοιτητή	**Θέμελης Καλόμοιρος**
Πατρώνυμο	**Σακελλάρης Καλόμοιρος**
Αριθμός Μητρώου	**ΜΠΠΛ/ 15018**
Επιβλέπων	**Ευθύμιος Αλέπης, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης Μάρτιος 2019

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ευθύμιος Αλέπης Επίκουρος Καθηγητής	Κωνσταντίνος Πατσάκης Επίκουρος Καθηγητής	Γεώργιος Τσιχριντζής Βαθμίδα

Περίληψη

Ελληνικά

Η μελέτη, ανάλυση και υλοποίηση των διαθέσιμων τεχνολογιών δικτύωσης που υπάρχουν στο λειτουργικό σύστημα Android είναι το θέμα αυτής της εργασίας. Για την επίδειξη των τεχνολογιών αυτών αναπτύχθηκε το DoSDroid.

Πρόκειται για μία εφαρμογή η οποία δίνει τη δυνατότητα δημιουργίας και εξαπόλυσης επιθέσεων τύπου Denial of Service (DoS), μέσω του πρωτοκόλλου HTTP.

Ένας χρήστης καθορίζει την επίθεση ενώ άλλοι δηλώνουν συμμετοχή σε αυτή, συνεισφέροντας έτσι στην αποτελεσματικότητα της. Κατά την ώρα της εκκίνησης το δίκτυο των συμμετεχόντων αρχίζει να επιτίθεται συγχρονισμένα.

Αγγλικά

The study, the analysis and the implementation of the relevant network connectivity solutions offered in Android is the main topic of this thesis. The DosDroid was developed for the sole purpose of demonstrating those network technologies.

DosDroid is an Android application which gives its users the potential to organize and launch Denial of Service (DoS) attacks towards HTTP Servers.

One device defines the attack whereas others join forces, thereby increasing the effectiveness of the attack. When it’s time for the launch, the network of those devices which take part will start attacking.

Περιεχόμενα

1. Εισαγωγή

Αν σε μία πανεπιστημιακή αίθουσα παρευρίσκονται περίπου εκατό άτομα τότε οι πιθανότητες ύπαρξης τουλάχιστον ισάριθμων έξυπνων κινητών συσκευών είναι πολύ μεγάλες. Πως όμως θα μπορούσαν να συνδυαστούν όλες αυτές οι συσκευές και για ποιον σκοπό;

Σε αυτά είναι τα δύο ερωτήματα καλείται να δώσει απάντηση η παρούσα εργασία, μέσω της εφαρνιγλης DoSDroid. Ένα “Android app” που οργανώνει και εξαπολύει επιθέσεις τύπου Denial of Service προς διάφορους HTTP εξυπηρετητές.

Η αποτελεσματικότητα των επιθέσεων αυτών είναι ανάλογη του πλήθος των συσκευών που έχουν συγκεντρωθεί για την εκτέλεση της επίθεσης. Σε αυτό το σημείο κρίνεται απαραίτητη η δικτύωση των συσκευών και ο συγχρονισμός τους.

Οι τεχνολογίες δικτύωσης που προσφέρονται από την δημοφιλή πλατφόρμα χρησιμοποιούνται στο DoSdroid για τη συγκέντρωση του δικτύου των επιτιθεμένων συσκευών.

Η εφαρμογή χρησιμοποιείται μόνον για εκπαιδευτικούς σκοπούς.

2. Ανασκόπηση Πεδίου

Με μία απλή αναζήτηση στο ηλεκτρονικό κατάστημα της Google μπορεί κανείς να βρει δεκάδες σχετικές εφαρμογές. Τόσο για “apps” που απλώς χρησιμοποιούν μία ή περισσότερες τεχνολογίες δικτύωσης, όσο και για επιθέσεις τύπου Denial of Service.

Ακολουθεί μία γρήγορη παρουσίαση σχετικών εφαρμογών:

DDos

Η εφαρμογή DDos έχει ως αντικειμενικό σκοπό τη δημιουργία και εξαπόλυση επιθέσεων τύπου Distributed Denial of Service (DDoS).

Χρήση

Ο χρήστης ορίζει τον στόχο μέσω της διεύθυνσης IP.

Στην συνέχεια καλείται να δώσει περαιτέρω λεπτομέρειες όσον αφορά την εκδήλωση της επίθεσης. Χρόνος αναμονής για επιτυχή σύνδεση με τον εξυπηρετητή (timeout), την πόρτα (port) που “ακούει” ο τελευταίος καθώς και αριθμό νημάτων (threads) που θα χρησιμοποιηθούν από τη συσκευή κατά την εκτέλεση της επίθεσης. Τέλος ο χρήστης έχει τη δυνατότητα να επιλέξει το πρωτόκολλο μέσα από το οποίο θα διεξαχθεί η επίθεση. TCP, UDP και HTTP είναι οι τρεις διαθέσιμες επιλογές.

Google Play

Η εφαρμογή είναι διαθέσιμη στο Google Play.

P2P Video Call

Μία εφαρμογή η οποία εκμεταλλεύεται την τεχνολογία Wifi Peer to Peer για λόγους επικοινωνίας.

Το P2P Video Call δίνει τη δυνατότητα στους χρήστες του να πραγματοποιήσουν “βιντεοκλήσεις” χωρίς τη μεσολάβηση κάποιου ασύρματου δικτύου Wifi ή δικτύου κινητής τηλεφωνίας, σε ακτίνα 100 μέτρων!

Προσφέρονται και ομαδικές βιντεοκλήσεις αλλά δεν είναι δωρεάν για τη συσκευή εξυπηρετητή.

Χρήση

Η κυρίως λειτουργία της εφαρμογής είναι η εξής:

1. Κατά την εκκίνηση η συσκευή αναζητά συσκευές που υποστηρίζουν την τεχνολογία Wifi Peer to Peer και βρίσκονται, προφανώς, εντός εμβέλειας.
2. Σε περίπτωση που βρεθεί η επιθυμητή συσκευή μπορεί να ξεκινήσει η επικοινωνία.

Από εκεί και πέρα ο κάθε χρήστης μπορεί να δηλώσει ένα ψευδώνυμο έτσι ώστε να είναι αναγνωρίσιμος από τους άλλους χρήστες κτλ.

Google Play

Η εφαρμογή είναι διαθέσιμη στο Google Play.

Spaceteam

Πρόκειται για ένα δημοφιλές (άνω του 1.000.000 εγκαταστάσεις) παιχνίδι, το οποίο παίζεται ομαδικά.

Αυτό καθίσταται δυνατό μέσω της τεχνολογίας Network Service Discovery (NSD) που χρησιμοποιείται. Ο περιορισμός όμως της συγκεκριμένης τεχνολογίας περιορίζει τους παίκτες να είναι συνδεμένοι στο ίδιο δίκτυο.

Χρήση

Ένας χρήστης θα αναλάβει ρόλο ομαδάρχη. Θα “φτιάξει παιχνίδι” και θα περιμένει τους υπόλοιπους παίκτες να συνδεθούν με τη συσκευή του ομαδάρχη. Ο ομαδάρχης δίνει το σήμα για εκκίνηση και κάπως έτσι ξεκινάει μία περιπέτεια στο διάστημα.

Google Play

Η εφαρμογή είναι διαθέσιμη στο Google Play.

Bluetooth Messenger

Κλείνοντας μία εφαρμογή που χρησιμοποιεί κατά κόρον την τεχνολογία Bluetooth. Και αυτή όπως και το P2P Video Call προηγουμένως, για λόγους επικοινωνίας μιας πρόκειται για “chat application”.

Η εφαρμογή προσφέρει άμεση επικοινωνία με μία συσκευή ωστόσο δύναται και η δημιουργία δωματίου επικοινωνίας όπου περισσότερες από μία συσκευές μπορούν να επικοινωνούν γραπτώς, να στείλουν εικόνες κτλ.

Χρήση

Ο χρήστης έχει τη δυνατότητα αναζήτησης συσκευών που υποστηρίζουν Bluetooth και να συνδεθεί με αυτές απευθείας. Μπορεί όμως και να δημιουργήσει δωμάτιο όπου υπόλοιπο χρήστες θα έχουν τη δυνατότητα σύνδεσης και επικοινωνίας μεταξύ τους.

Google Play

Η εφαρμογή είναι διαθέσιμη στο Google Play.

3. Παρουσίαση Εφαρμογής

Το DoSdroid είναι μία εφαρμογή για το λειτουργικό σύστημα Android. Μέσω αυτής οι χρήστες έχουν τη δυνατότητα δημιουργίας και εξαπόλυσης επιθέσεων τύπου DoS. Οι επιθέσεις αυτές γίνονται προς διάφορους στόχους, μέσω του πρωτοκόλλου HTTP.

Ο πηγαίος κώδικας είναι αναρτημένος στο github.

Χρήση

Δύο είναι οι κύριες περιπτώσεις χρήσης: η δημιουργία μίας επίθεσης και η δήλωση συμμετοχής.

Η εφαρμογή επιτρέπει σε μία συσκευή να έχει δημιουργήσει μία επίθεση ενώ ταυτόχρονα να παίρνει μέρος σε μία άλλη, ωστόσο αυτό το σενάριο δεν θεωρείται ξεχωριστή περίπτωση χρήσης.

Δημιουργία επίθεσης

Αρχικά ο δημιουργός ορίζει τον στόχο καθώς και διάφορες άλλες, απαραίτητες, λεπτομέρειες για την διεξαγωγή της επίθεσης. Όταν η δημιουργία επιτευχθεί, τότε η συσκευή του χρήστη λειτουργεί ως ένας εξυπηρετητής (server).

Λειτουργία εξυπηρετητή

Ο εξυπηρετητής παραμένει ενεργός αναμένοντας να ικανοποιήσει αιτήματα σύνδεσης από συσκευές πελάτες. Πρόκειται για τους ενδιαφερόμενους συμμετέχοντες στην επίθεση. Αυτοί θα απαρτίσουν το επιτιθέμενο δίκτυο (botnet).

Ο κάθε εξυπηρετητής δέχεται αιτήματα σύνδεσης μέσω μόνο μίας συγκεκριμένης τεχνολογίας δικτύωσης. Αυτή που επιλέχθηκε αρχικά από τον δημιουργό της επίθεσης. Θα μπορούσε να είναι το Bluetooth, η τεχνολογία Wifi Peer to Peer ή μία από τις άλλες που προσφέρονται στην εφαρμογή.

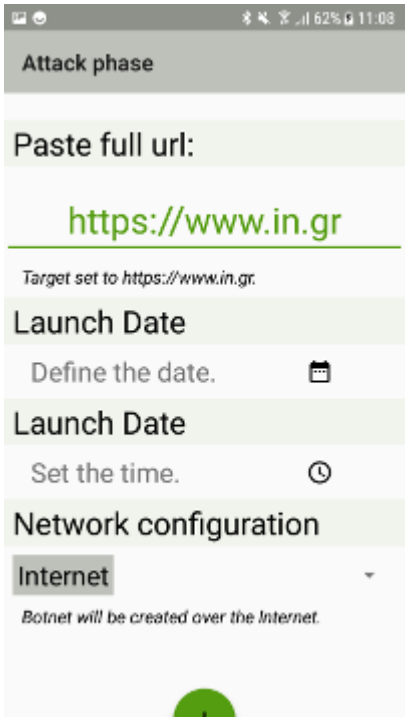
Ο χρήστης έχει το δικαίωμα να τερματίσει τη λειτουργία εξυπηρετητή όταν εκείνος το επιθυμήσει. Ως αποτέλεσμα όμως η επίθεση που είχε δημιουργηθεί αρχικά θα ακυρωθεί και το δίκτυο των επιτιθέμενων συσκευών δεν θα υπάρχει πλέον.

Διαδικασία

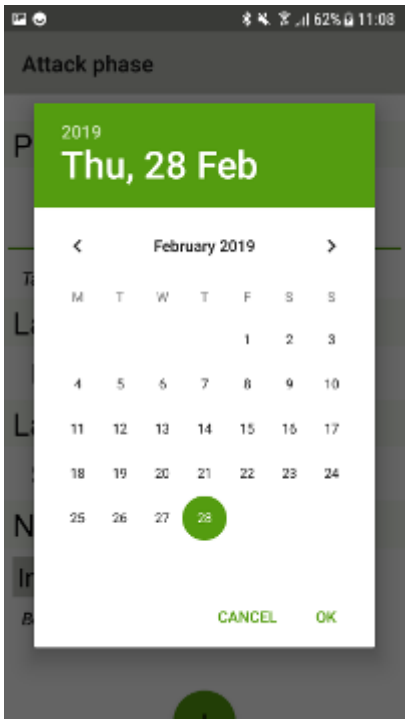
Ακολουθούν αναλυτικά τα βήματα για τη δημιουργία μίας επίθεσης στην εφαρμογή DoSdroid:

1. Εκκίνηση της εφαρμογής και επιλογής δημιουργίας επίθεσης από το μενού.
2. Ορισμός στόχου (διεύθυνση ip, URL).
3. Ορισμός ημερομηνίας και ώρας για την αφετηρία της επίθεσης.
4. Επιλογή τεχνολογίας δικτύωσης που θα “ακούει” ο εξυπηρετητής.
5. Επιβεβαίωση δημιουργίας.

1 Επιλογή ”Attack a Website



2 Επικόλληση της διεύθυνσης του στόχου



3 Ορισμός ημερομηνίας έναρξης

4 Ορισμός χρόνου έναρξης

Πολλαπλοί εξυπηρετητές

Το DoSdroid επιτρέπει τη δημιουργία πολλαπλών εξυπηρετητών, έναν για κάθε επίθεση.

Αυτό σημαίνει ότι ένας χρήστης μπορεί να επιλέξει τη δημιουργία επίθεσης και τη συλλογή του botnet μέσω του Bluetooth, αυτό όμως δεν τον εμποδίζει να ξεκινήσει νέα επίθεση (προς άλλο στόχο, δεν επιτρέπεται ο ίδιος) μέσω της τεχνολογίας Network Service Discovery.

Συμμετοχή σε επίθεση

Η δεύτερη περίπτωση χρήσης είναι η συμμετοχή σε μία επίθεση προς κάποιον απομακρυσμένο υπολογιστή.

Λειτουργία πελάτη

Η συσκευή-πελάτης οφείλει να συνδεθεί πρώτα με τον εξυπηρετητή και στη συνέχεια να παραλάβει κάποιες κρίσιμες πληροφορίες για την ενδεχόμενη επίθεση.

Οι πληροφορίες έχουν να κάνουν με το στόχο της επίθεσης, την ημερομηνία και ώρα εκκίνησης αλλά και άλλες σημαντικές πληροφορίες που αφορούν την τεχνολογία δικτύωσης που έχει στηθεί ο εξυπηρετητής.

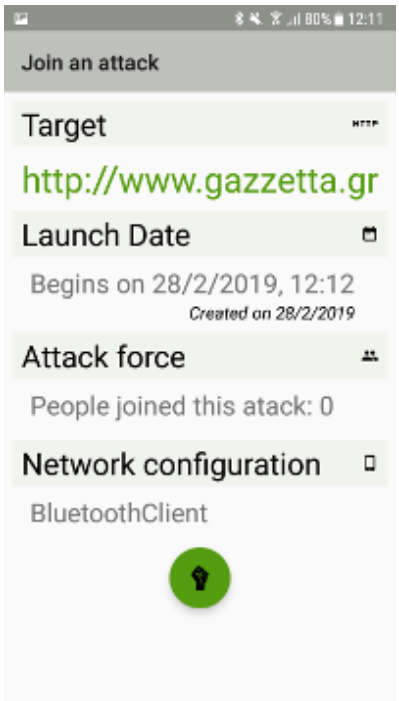
Τέλος, ο πελάτης μπορεί να δηλώσει παύση της συμμετοχής του σε μία συγκεκριμένη επίθεση, όποτε εκείνος το επιθυμήσει.

Διαδικασία

1. Εκκίνηση της εφαρμογής και επιλογή της συμμετοχής σε επίθεση.
2. Επιλογή ενός από τους δημοσιευμένους στόχους. Οι στόχοι είναι δημοσιοποιημένοι ανά κατηγορίες. Οι κατηγορίες είναι οι τεχνολογίες δικτύωσης (δημοσιευμένοι στόχοι μέσω Internet,, NSD κτλ).
3. Επιβεβαίωση συμμετοχής!

5 Επιλογή ”Join an attack” για προβολλή δημοσιευμένων επιθέσεων

6 Επιλογή επιθυμητής επίθεσης



7 Πάτημα πράσινου κουμπιού για δήλωση συμμετοχής

Σε αυτό το σημείο και αναλόγως με τη ρύθμιση δικτύωσης του εξυπηρετητή, υπάρχει περίπτωση να ζητηθεί από τον χρήστη-πελάτη να προβεί σε διάφορες ενέργειες που έχουν να κάνουν με τη σύνδεση.

Παραδείγματος χάρη, εάν ο εξυπηρετητής όρισε τρόπο δικτύωσης το Bluetooth τότε μπορεί να ζητηθεί από τον πελάτη η ενεργοποίηση του μηχανισμού Bluetooth της συσκευής κτλ.

Πολλαπλές συμμετοχές

Ένας χρήστης μπορεί να δηλώσει συμμετοχή σε όσες επιθέσεις θέλει. Μπορεί να επιβλέψει τις επιθέσεις που έχει συμμετάσχει με δύο διαφορετικούς τρόπους.

- Εκκίνηση εφαρμογής και επιλογής “Συνεισφορά” από το μενού
- Επιλογή ειδοποίησης πελάτη (notification)

Σύνδεση Πελάτη-Εξυπηρετητή

Παρακάτω περιγράφονται κάποια χαρακτηριστικά της σύνδεσης πελάτη - εξυπηρετητή.

Αυτόματη σύνδεση

Η σύνδεση του πελάτη με τον εξυπηρετητή, στο DoSdroid, γίνεται αυτόματα. Κάτι τέτοιο δεν ισχύει κατά τη διαδικασία σύνδεσης συσκευών μέσω σχετικών εφαρμογών που είναι προεγκατεστημένες στο λειτουργικό. Εκεί ο χρήστης καλείται να επιλέξει μία από τις διαθέσιμες συσκευές για σύνδεση.

Στην περίπτωση του DosDroid ο πελάτης λαμβάνει κατά τη δήλωση συμμετοχής του τα στοιχεία σύνδεσης του εξυπηρετητή που πρόκειται να συνδεθεί. Αποφεύγεται έτσι η προβολή άγνωστων συσκευών στον χρήστη και η διαδικασία να μαντέψει σωστά τον εξυπηρετητή.

Αποτυχία Σύνδεσης

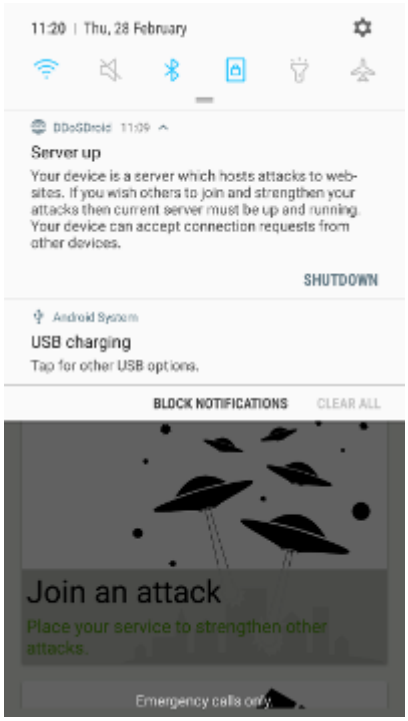
Η σύνδεση του πελάτη με τον εξυπηρετητή δεν είναι πάντα επιτυχής. Όλες οι επιθέσεις δημοσιεύονται στο διαδίκτυο, αυτό όμως δε σημαίνει ότι ένας εξυπηρετητής, Wifi Peer to Peer για παράδειγμα, βρίσκεται σε ακτίνα εμβέλειας με τον πελάτη (100 μέτρα ακτίνα στη συγκεκριμένη περίπτωση).

Σε τέτοιες περιπτώσεις, όπως και σε άλλες, η σύνδεση είναι καταδικασμένη να αποτύχει. Ο πελάτης ενημερώνεται με σχετικό μήνυμα για την αποτυχία της σύνδεσης.

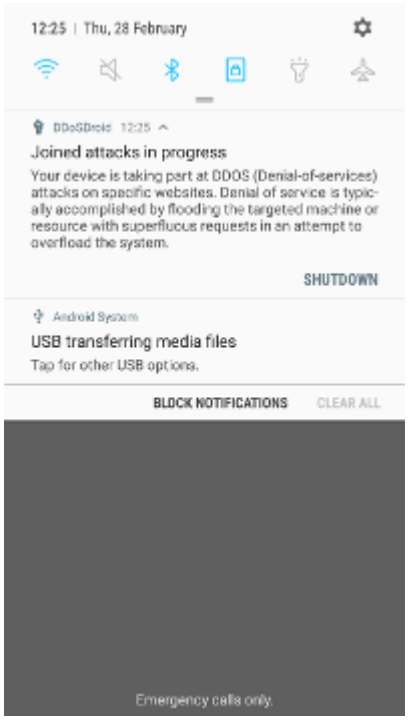
Ειδοποιήσεις (Notifications)

Η εφαρμογή ειδοποιεί τον χρήστη μέσω ειδοποιήσεων (Android Notifications) όταν

- ο χρήστης δημιούργησε με επιτυχία μία νέα επίθεση.
- ο πελάτης ολοκλήρωσε με επιτυχία τη συμμετοχή σε μία επίθεση.



8 Server notification



9 Client notification

Επιλέγοντας τις ειδοποιήσεις οι ενδιαφερόμενοι μπορούν να προβάλλουν τις επιθέσεις που έχουν δημιουργήσει ή που ακολουθούν, αναλόγως ποια ειδοποίηση επέλεξαν.

Τέλος, υπάρχει μία ειδική λειτουργία και για τις δύο ειδοποιήσεις, ο τερματισμός. Στην περίπτωση του εξυπηρετητή αυτή η λειτουργία ακυρώνει όλες τις επιθέσεις που έχουν δημιουργηθεί από τη συγκεκριμένη συσκευή. Στην άλλη περίπτωση ο πελάτης απλώς αποδεσμεύεται από όλες τις συμμετοχές του.

4. Αρχιτεκτονική Συστήματος

4.1 ΕΡΓΑΛΕΊΑ

Τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του DoSdroid είναι περίπου τα ίδια με αυτά που χρησιμοποιούνται για την ανάπτυξη οποιασδήποτε εφαρμογής για το λειτουργικό σύστημα Android.

- Πλατφόρμα ανάπτυξης: Linux (Ubuntu 18.04.2 LTS)
- IDE: Android Studio 3.3
- Γλώσσα: Java
- Version Control: Git
- Δύο κινητά τηλέφωνα: Samsung Galaxy A3 και Samsung Galaxy S5

4.2 ΑΡΧΙΤΕΚΤΟΝΙΚΉ ΣΧΕΔΊΑΣΗΣ ΛΟΓΙΣΜΙΚΟΎ

Από το επίπεδο δόμησης συστήματος μέχρι και το χαμηλότερο επίπεδο κώδικα το DoSdroid είναι σχεδιασμένο υπακούοντας σε ορισμένες αρχές. Οι λόγοι και τα πλεονεκτήματα που υπάρχουν ακολουθώντας αυτή την προσέγγιση δεν ανήκουν στην παρούσα εργασία.

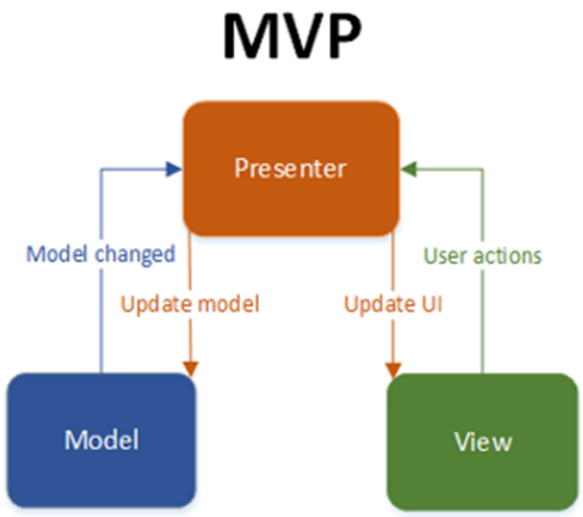
Δεν υπάρχει λόγος να γίνει κάποια ανάλυση για σχεδίαση στο χαμηλό επίπεδο του κώδικα, ωστόσο αξίζει να σημειωθεί ότι έχει καταβληθεί μεγάλη προσπάθεια για να διατηρηθεί όσο πιο “καθαρός” γίνεται, σύμφωνα πάντα με τις δυνατότητες του εξεταζομένου.

Είναι μείζονος σημασίας, ωστόσο, να γίνει αναφορά για την Αρχιτεκτονική Σχεδίασης Λογισμικού (Software Architectural Design Pattern) του DoSdroid, μιας και μιλάμε για επίπεδο εφαρμογής.

Model View Presenter

Το DoSDroid έχει σχεδιαστεί ακολουθώντας μία συγκεκριμένη Αρχιτεκτονική Σχεδίασης Λογισμικού (Software Architectural Design Pattern), την Model View Presenter (MVP). Για την αρχιτεκτονική MVP υπάρχει ένα σαφές μοντέλο που την περιγράφει, ωστόσο δεν υπάρχει ένας συγκεκριμένος, φορμαλιστικός, τρόπος υλοποίησης. Άλλωστε πρόκειται για design.

Το μοντέλο αυτό φαίνεται παρακάτω.



10 Το μοντέλο της MVP (TechYourChance, 2015)

Η Ανατομία της MVP

Από το μοντέλο διακρίνονται τρεις οντότητες. Οι Model, View και Presenter.

- Η οντότητα Model είναι υπεύθυνη για την μνημόνευση της τρέχουσας κατάστασης του συστήματος (system state), είτε αυτή αποθηκεύεται σε κάποιο σκληρό δίσκο, σε μία βάση δεδομένων ή στη μνήμη.
- Η View διαχειρίζεται την είσοδο και την έξοδο (input-output) από/προς τον χρήστη.
- Η οντότητα Presenter είναι αυτή που συγκεντρώνει τη λογικό μέρος του συστήματος.

Τα συστατικά του μοντέλου πρέπει να είναι ανεξάρτητα μεταξύ τους. Η οντότητα Presenter είναι η μόνη που γνωρίζει την ύπαρξη των άλλων δύο. Η View δεν γνωρίζει την ύπαρξη της Model και αντίστροφα.Από το παραπάνω συνεπάγεται το εξής: μπορούμε ανά πάσα στιγμή να ανταλλάξουμε ένα συστατικό, για παράδειγμα μία νέα View, χωρίς η ενέργεια αυτή να επηρεάσει τα Model και Presenter.

Πως λειτουργεί

Κατά τη λειτουργία οι αλληλεπιδράσεις μεταξύ των οντοτήτων είναι δύο. Αυτές μεταξύ View-Presenter και Presenter-Model. Η View αναφέρει είσοδο από τον χρήστη, για παράδειγμα το πάτημα ενός κουμπιού, και ο Presenter την διατάσσει να ενημερώσει το γραφικό περιβάλλον. Από την άλλη πλευρά ο Presenter ενημερώνει το Model και εκείνο αναφέρει την αλλαγή πίσω στον Presenter.

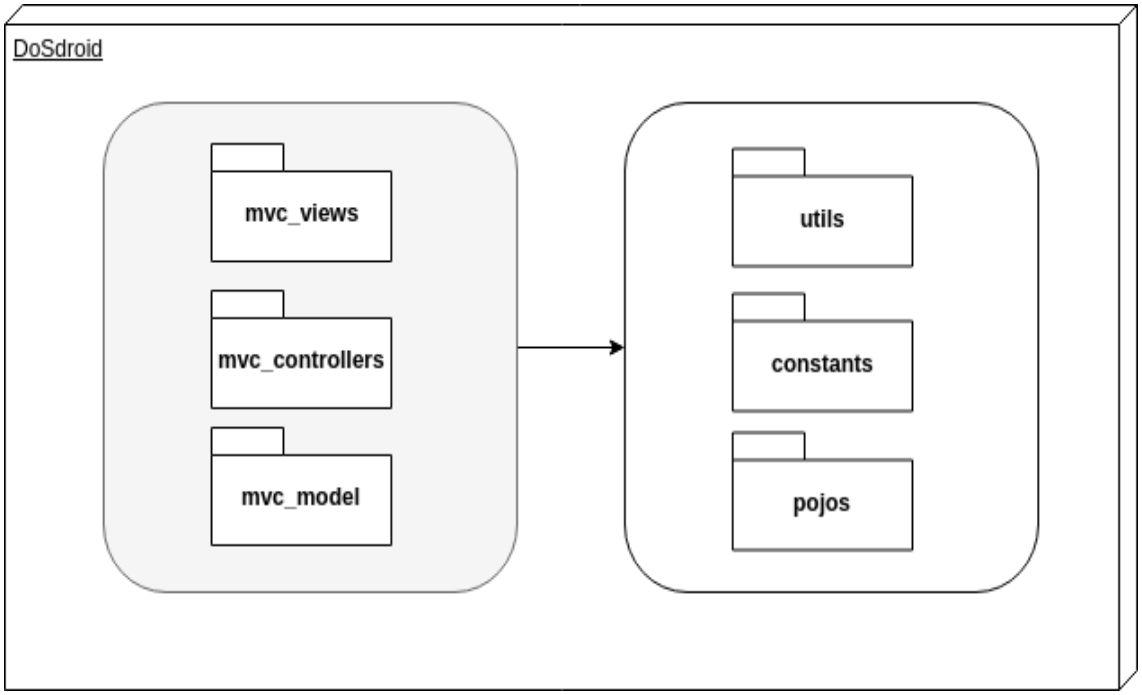
Ένα καθολικό παράδειγμα είναι το εξής: ο χρήστης έγραψε σε μία φόρμα το όνομα του και πάτησε save.

1. Η View αντιλαμβάνεται το πάτημα (είσοδο) του χρήστη και το αναφέρει στον Presenter.
2. Ο Presenter ζητάει από την οντότητα Model να αποθηκεύσει το όνομα του χρήστη.
3. Η Model ενημερώνει τον Presenter για την επιτυχία της αποθήκευσης.
4. Ο Presenter ζητάει από τη View να δείξει ένα σχετικό μήνυμα στον χρήστη (έξοδος).

Φυσικά υπάρχει και το σενάριο της αποτυχίας, άρα σε τούτη την περίπτωση ο Presenter θα ζητούσε από τη View να δείξει σφάλμα κτλ.

Υλοποίηση στο DosDroid

Ο κώδικας της εφαρμογής έχει χωριστεί σε δύο ομάδων πακέτα, όπως γίνεται αντιληπτό από το παρακάτω διάγραμμα.



Σημείωση: Η MVP είναι μία παραλλαγή της MVC (αντί Controller, Presenter). Εξού και ο λόγος του προθέματος “mvc_” αντί για “mvp_” αλλά και του “mvc_controllers” αντί για “mvp_presenters”.

Η μία ομάδα πακέτων είναι εκείνη που υλοποιεί την αρχιτεκτονική MVP ενώ η άλλη ομάδα αποτελείται από πακέτα τελείως ανεξάρτητα, φυσικά χρησιμοποιούνται από την εφαρμογή, όπως υποδηλώνει το βέλος.

- Το πακέτο *mvc_views* εμπεριέχει όλες τις κλάσεις για τη διεπαφή με το χρήστη (user interface-UI). Οι κλάσεις αυτές είναι τύπου ViewMvc.
- Το πακέτο mvc_model περιέχει το AttackRepository, ένα module που αποθηκεύει τις επιθέσεις.
- Το πακέτο mvc_controllers περιέχει όλες τις Activities/Fragments, μιας και αυτές αποτελούν τους Presenter στο Android, σύμφωνα με την παραλλαγή MVP που επιλέχθηκε. Επιπλέον εμπεριέχονται κάποια module που υλοποιούν τη σύνδεση συσκευών, τα οποία χρησιμοποιούν οι Presenter.

4.3 ΔΙΚΤΎΩΣΗ

Οι τρόποι δικτύωσης στο λειτουργικό σύστημα Android αποτελούν το κυρίως μέρος τούτης της εργασίας. Το DoSdroid αναπτύχθηκε με σκοπό την υλοποίηση αυτών των τεχνολογιών σε επίπεδο εφαρμογής.

Οι τεχνολογίες αυτές είναι το Internet, Bluetooth, Wifi Peer to Peer και Network Service Discovery (NSD). Η πλατφόρμα προσφέρει και άλλους τρόπους δικτύωσης, όπως Near Field Communication ή μέσω USB, αλλά δεν ταίριαζαν στην περίπτωση χρήσης του DoSdroid και επομένως δεν χρησιμοποιήθηκαν.

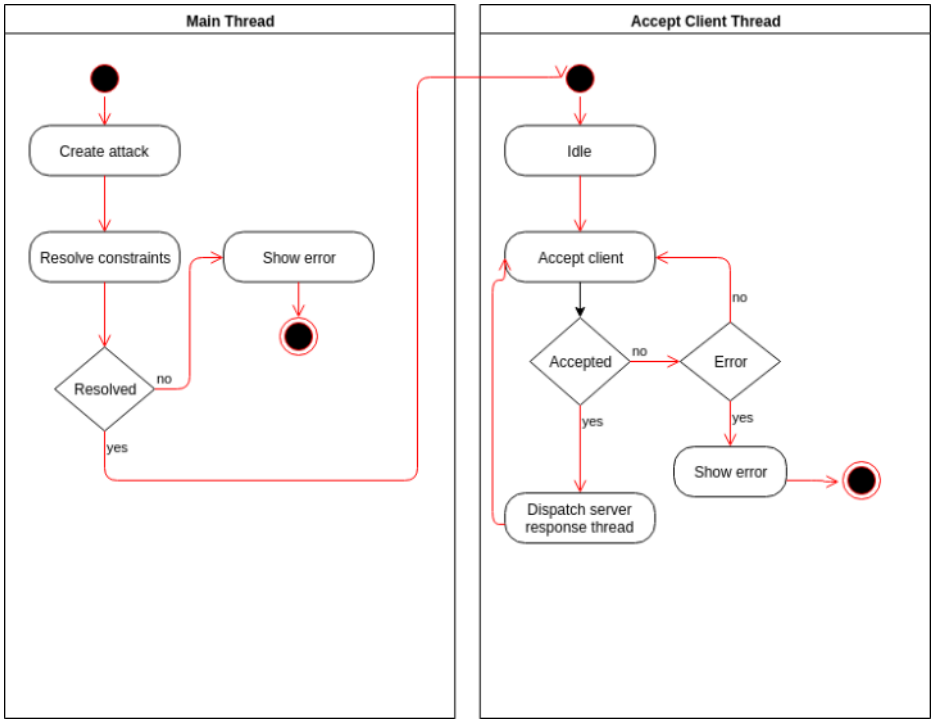
Σύνδεση συσκευών

Η σύνδεση μεταξύ συσκευών στο DoSdroid υλοποιείται στην περίπτωση που η μία θέλει να συμμετάσχει σε μία επίθεση που έχει δημιουργήσει η άλλη. Αυτό το στάδιο είναι υποχρεωτικό. Η συσκευή εξυπηρετητής διαχειρίζεται τον κάθε πελάτη ξεχωριστά. Αποδέχεται το αίτημα σύνδεσης και στέλνει ένα συνθηματικό. Στο τέλος κλείνει τη σύνδεση, αναμένοντας παράλληλα αιτήματα από άλλους πελάτες. Η επικύρωση του ληφθέντος συνθηματικού γίνεται στην πλευρά του πελάτη. Αν είναι απάντηση από τον εξυπηρετητή είναι έγκυρη τότε η συσκευή πελάτης προγραμματίζεται για την επίθεση.

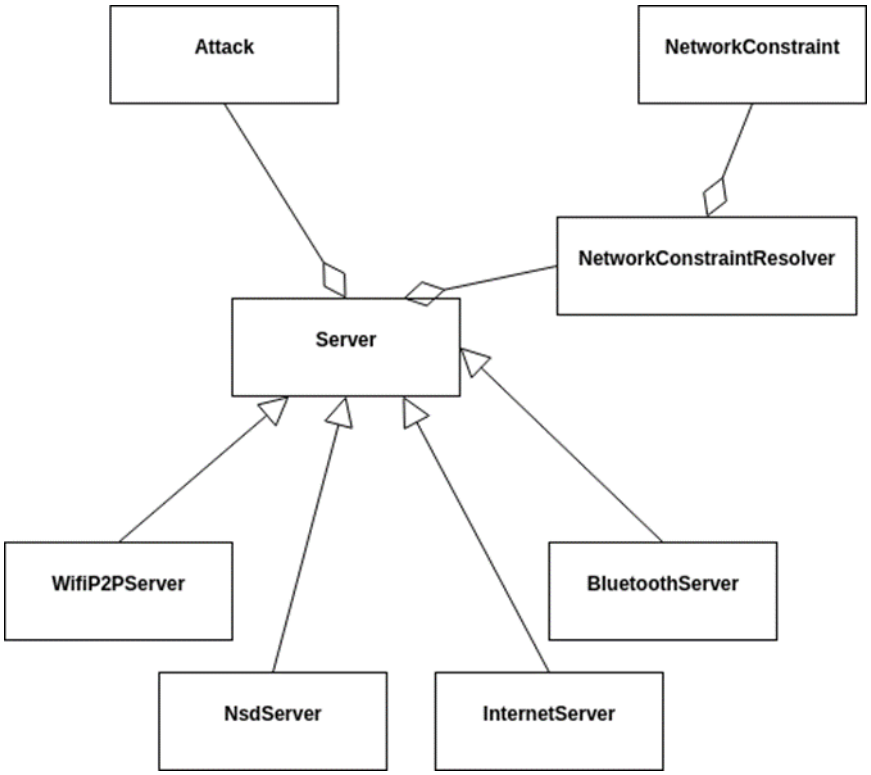
Λειτουργία Εξυπηρετητή

Η λειτουργία εξυπηρετητή (server mode) ενεργοποιείται όταν ένας χρήστης δημιουργήσει μία επίθεση. Αφού ο ικανοποιηθούν ορισμένα κριτήρια η συσκευή εξυπηρετητής στέλνει ένα συνθηματικό στον πελάτη που αιτήθηκε σύνδεση και τερματίζει τη σύνδεση. Παράλληλα αναμένει για νέα αιτήματα.

Η διαδικασία αυτή παρουσιάζεται στο παρακάτω διάγραμμα.



Η έννοια του εξυπηρετητή στην εφαρμογή αντιπροσωπεύεται από την κλάση Server. Το παρακάτω διάγραμμα τάξεων δείχνει τις βασικές κλάσεις που συσχετίζονται με την κλάση Server.



Κριτήρια δικτύωσης

Βασική προϋπόθεση είναι η ικανοποίηση μιας σειράς από ορισμένα κριτήρια έτσι ώστε να ολοκληρωθεί επιτυχώς η δημιουργία της επίθεσης και να εκκινήσει η λειτουργία εξυπηρετητή. Η ικανοποίηση της σειράς των κριτηρίων εγγυάται το γεγονός ότι η λειτουργία εξυπηρετητή είναι δυνατή εκείνη τη χρονική στιγμή. Ένα κριτήριο αν μείνει ανικανοποίητο η λειτουργία εξυπηρετητή δεν ξεκινάει και ο χρήστης ενημερώνεται για σφάλμα στη δημιουργία της επίθεσης. Η κάθε τεχνολογία δικτύου έχει τα δικά της κριτήρια. Ας πάρουμε ένα παράδειγμα από την τεχνολογία Wifi Peer to Peer. Η επίδοξη συσκευή εξυπηρετητής θα πρέπει να υποστηρίζει την συγκεκριμένη τεχνολογία. Αυτό είναι το πρώτο κριτήριο της σειράς. Το επόμενο θα μπορούσε να είναι η ενεργοποίηση του Wifi αντάπτορα (εφόσον είναι απενεργοποιημένος). Την έννοια του κριτηρίου δικτύωσης στο DoSdroid ενσαρκώνει η κλάση NetworkConstraint, ενώ η κλάση NetworkConstraintResolver είναι η κλάση που αναλαμβάνει την επίλυση των NetworkConstraint.

Αιτήσεις σύνδεσης

Αφού ικανοποιηθούν όλα τα κριτήρια δικτύωσης που περιγράφηκαν προηγουμένως ο εξυπηρετητής τίθεται σε λειτουργία και αναμένει συνεχώς νέα αιτήματα σύνδεσης από πελάτες.

Η εφαρμογή χρησιμοποιεί την κλασική αρχιτεκτονική πελάτη-εξυπηρετητή (client-server architecture), χρησιμοποιώντας διαφορετικά νήματα επεξεργασίας (thread). Ένα νήμα είναι υπεύθυνο για την αποδοχή νέων αιτημάτων σύνδεσης, ενώ για για κάθε πελάτη ξεχωριστά υπάρχει ένα νήμα που διαχειρίζεται την επικοινωνία με τον εξυπηρετητή.

Στη βάση της υλοποίησης της αρχιτεκτονικής πελάτη-εξυπηρετητή χρησιμοποιείται η λειτουργικότητα που παρέχουν οι Java Sockets από το πακέτο java.net. Πρόκειται για μία βιβλιοθήκη η οποία καθιστά δυνατή την επικοινωνία δικτύου σε χαμηλότερο επίπεδο. Ιδανική για αυτή την περίπτωση.

Στο παρακάτω τμήμα κώδικα (AcceptClientThread.java), μέσα στο επαναληπτικό βρόγχο, επιτυγχάνεται η αποδοχή αιτημάτων σύνδεσης (σειριακά) ενώ για κάθε νέο αίτημα ξεκινάει νέο νήμα (ServerResponseThread) το οποίο θα στείλει το συνθηματικό στον πελάτη.

```
public void run(){
      while (true) {
               Socket socket = serverSocket.accept();
               executor.execute(new ServerResponseThread(socket));
      }
}
```

Με τη σειρά του το ServerResponseThread στέλνει το συνθηματικό στον πελάτη και αμέσως κλείνει τη σύνδεση για να μη σπαταλάει άσκοπα πόρους. Τμήμα του κώδικα που υλοποιεί αυτή την λειτουργία φαίνεται παρακάτω.

```
public void run() {
        out.println(Server.RESPONSE);
        closeSocket();
}
```

Διαχείριση πολλαπλών εξυπηρετητών

Ο χρήστης του DoSdroid έχει τη δυνατότητα να επιτεθεί σε περισσότερους από έναν ιστοτόπους, επιλέγοντας όποια τεχνολογία δικτύου επιθυμεί. Ως αποτέλεσμα αυτού η συσκευή θα πρέπει να λειτουργεί ως εξυπηρετητής για όλες τις επιθέσεις που έχουν δημιουργηθεί.

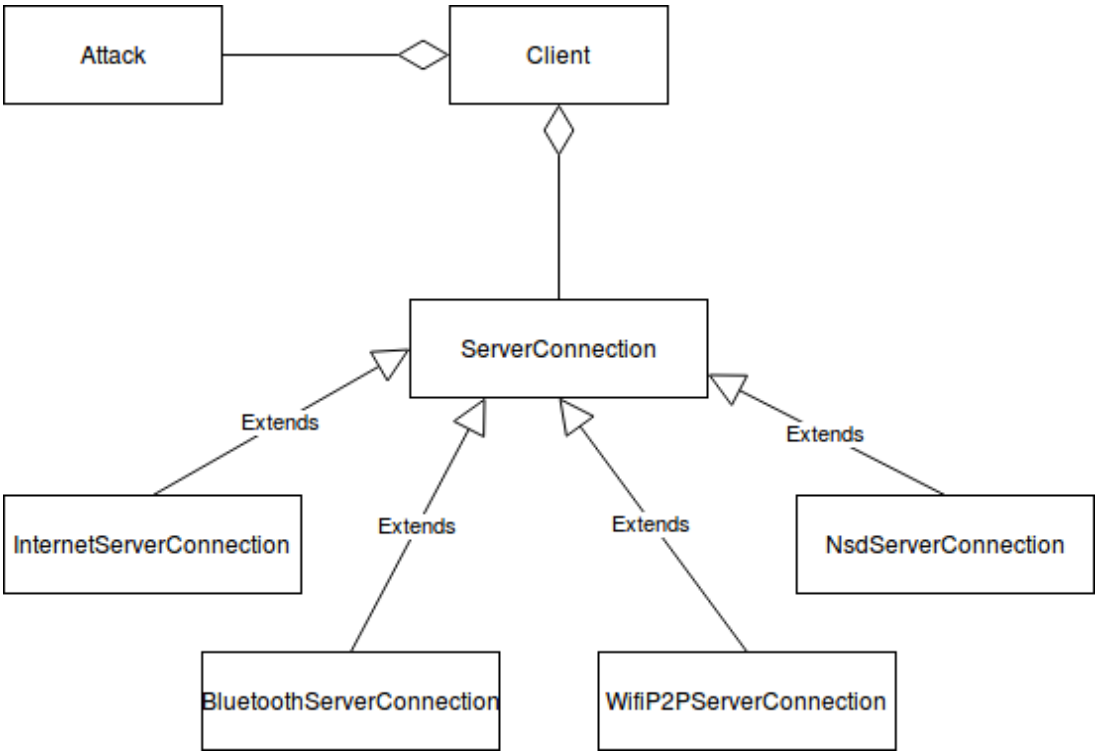
Αυτή την εργασία αναλαμβάνει η κλάση HostAttackService η οποία είναι ο διαχειριστής όλων των στιγμιοτύπων Server που έχει δημιουργηθεί. Η δημιουργία και ο τερματισμός στιγμιοτύπων τύπου Server γίνονται μέσω αυτής της υπηρεσίας.

Λειτουργία Πελάτη

Μία συσκευή μπαίνει σε ρόλο πελάτη από τη στιγμή που ο χρήστης της έχει δηλώσει συμμετοχή σε μία επίθεση που έχει δημιουργήσει κάποιος άλλος.

Ο πελάτης στέλνει ένα αίτημα σύνδεσης στον εξυπηρετητή της επίθεσης και περιμένει την αποδοχή του. Αφού το αίτημα γίνει αποδεκτό ο πελάτης θα πρέπει να λάβει απάντηση από τον εξυπηρετητή. Πρόκειται για το συνθηματικό το οποίο επαληθεύεται από τον πελάτη και προγραμματίζεται η διαδικασία της επίθεσης.

Η έννοια του πελάτη στο DoSdroid αντιπροσωπεύεται από την κλάση Client. Στο παρακάτω διάγραμμα διακρίνονται οι βασικές κλάσεις με τις οποίες συσχετίζεται η κλάση Client.



Σύνδεση με εξυπηρετητή

Τις λεπτομέρειες της υλοποίησης της σύνδεσης του πελάτη με τον εξυπηρετητή τις αναλαμβάνει ένα ξεχωριστό module στην εφαρμογή, το ServerConnection, για χάρη του πελάτη.

Η ServerConnection επιχειρεί τη σύνδεση με τον εξυπηρετητή και αναφέρει στον πελάτη το αποτέλεσμα για την επιτυχία της σύνδεσης. Όπως φαίνεται και το παραπάνω διάγραμμα τάξεων υπάρχει μία ξεχωριστή υλοποίηση της ServerConnection για κάθε τεχνολογία δικτύωσης που υλοποιείται στο DoSdroid.

Στο τεχνικό κομμάτι η διαδικασία της σύνδεσης πραγματοποιείται σε ξεχωριστό νήμα επεξεργασίας. Όπως και στην πλευρά του εξυπηρετητή, έτσι και εδώ στη βάση της υλοποίησης βρίσκονται οι Java Sockets.

Το παρακάτω τμήμα κώδικα από το αρχείο SocketConnectionThread.java δείχνει το επιχείρημα της σύνδεσης του πελάτη με τον εξυπηρετητή. Στη συνέχεια μία δομή ελέγχου αναφέρει το αποτέλεσμα της σύνδεσης.

```
public void run() {
        boolean connected = connectToServer();
        if (connected) {
            reportServerResponse();
        } else {
            serverResponseListener.onErrorServerResponse();
        }
    }
```

Σημειώνεται ότι μία σύνδεση με έναν εξυπηρετητή θεωρείται επιτυχής όταν και το συνθηματικό που λήφθηκε προηγουμένως είναι έγκυρο. Η σύνδεση από μόνη της δεν έχει καμία σημασία. Το παραπάνω διακρίνεται από την υλοποίηση της μεθόδου reportServerResponse():

```
private void reportServerResponse() {
        if (Server.isValid(getResponse())) {
            serverResponseListener.onValidServerResponse();
        } else {
            serverResponseListener.onErrorServerResponse();
        }
}
```

4.4 WIFI PEER TO PEER

Το Wi-Fi peer-to-peer (P2P) επιτρέπει την απευθείας σύνδεση συσκευών που έχουν το κατάλληλο υλικό και τρέχουν Android 4.0 και άνω. Η σύνδεση επιτυγχάνεται χωρίς κάποιο ενδιάμεσο σημείο πρόσβασης, όπως λόγου χάρη τη σύνδεση σε ένα δρομολογητή ενός οικιακού δικτύου.

Wi-Fi P2P Api

Το Android διαθέτει ένα Application Programming Interface (API) για τη χρήση της τεχνολογίας Wifi P2P στις γλώσσες Java και Kotlin.

Τα κύρια μέρη του API είναι τα εξής:

- Μέθοδοι που επιτρέπουν την ανακάλυψη και τη σύνδεση (κατόπιν αίτησης) συσκευών.
- Κλάσεις οι οποίες ειδοποιούν για την έκβαση του αποτελέσματος μίας σύνδεσης Wifi P2P.
- Ειδοποιήσεις για διάφορα γεγονότα που έχουν να κάνουν με τη συγκεκριμένη τεχνολογία δικτύωσης, όπως η ανακάλυψη μίας νέας συσκευής ή ο τερματισμός μίας σύνδεσης.

WifiP2pManager

Η κύρια κλάση του API είναι ο WifiP2pManager. Η τάξη WifiP2pManager παρέχει μεθόδους που επιτρέπουν την αλληλεπίδραση με το hardware του Wi-Fi στη συσκευή.

Διαθέσιμες ενέργειες:

- initialize() Εγγράφει την εφαρμογή στο Wi-Fi framework. Καλείται πριν από οποιαδήποτε άλλη μέθοδο.
- connect() Ξεκινά μία peer-to-peer σύνδεση με μία συσκευή.
- cancelConnect() Ακυρώνει την διαδικασία σύνδεσης.
- requestConnectInfo() Ζητάει πληροφορίες σύνδεσης μιας συσκευής.
- createGroup() Δημιουργεί μία peer-to-peer ομάδα με την τρέχουσα συσκευή σε ηγετικό ρόλο.
- removeGroup() Διαγράφει την ομάδα που περιγράφηκε προηγουμένως.
- requestGroupInfo()ει Ζητάει πληροφορίες της ομάδας peer-to-peer.
- discoverPeers() Εκκινεί τη διαδικασία ανακάλυψης διαθέσιμων συσκευών που υποστηρίζουν Wifi P2P.
- requestPeers() Ζητάει μία λίστα από τις συσκευές που έχουν ανακαλυφθεί.

Εκπομπές συμβάντων (event broadcasting)

Κατά τη διάρκεια μίας σύνδεσης Wifi P2P λαμβάνουν χώρα διάφορα γεγονότα όπως η απενεργοποίηση της αντένας του Wifi στη συσκευή. Οι εφαρμογές εγγράφονται για τα γεγονότα που ενδιαφέρονται να ειδοποιηθούν.

Το σύστημα ειδοποιεί για ένα συμβάν μέσω του μηχανισμού BroadcastReceiver. Οι εφαρμογές μπορούν να εγγραφούν στα εξής γεγονότα:

- WIFI_P2P_CONNECTION_CHANGED_ACTION Εκπέμπεται όταν η κατάσταση μίας σύνδεσης μέσω Wi-Fi έχει αλλάξει.
- WIFI_P2P_PEERS_CHANGED_ACTION Εκπέμπεται μετά από την εκτέλεση της μεθόδου discoverPeers(). Συνήθως εκτελείται η μέθοδος requestPeers() για τη λήψη λίστας με τις νέες ανακαλυφθείσες συσκευές.
- WIFI_P2P_STATE_CHANGED_ACTION Εκπέμπεται κατά την ενεργοποίηση/απενεργοποίηση του Wi-Fi.
- WIFI_P2P_THIS_DEVICE_CHANGED_ACTION Εκπέμπεται όταν κάποιες πληροφορίες της συσκευής, σχετικά με το Wi-Fi, έχουν αλλάξει. Για παράδειγμα το όνομα της συσκευής.

Η εφαρμογή DoSdroid εκμεταλλεύεται τέτοια γεγονότα στην περίπτωση του πελάτη (Client) και του εξυπηρετητή, πράττοντας αναλόγως. Το παρακάτω τμήμα κώδικα είναι από την κλάση WifiDirectReceiver, στην πλευρά του πελάτη, και φαίνονται οι ενέργειες που πραγματοποιούνται για κάθε γεγονός.

```
  public void onReceive(Context context, Intent intent) {
        switch (intent.getAction()) {
            case WIFI_P2P_STATE_CHANGED_ACTION:
                int state = intent.getIntExtra(EXTRA_WIFI_STATE, -1);
                handleWifiState(state);
                break;
            case WIFI_P2P_PEERS_CHANGED_ACTION:
                manager.requestPeers(channel, getPeerListListener());
                break;
            case WIFI_P2P_CONNECTION_CHANGED_ACTION:
                handleConnectionChange(intent);
                break;
            default:
                throw new IllegalArgumentException(TAG + ": Unknown
action");
        }
}
```

Για παράδειγμα αν ο WifiDirectReceiver λάβει ειδοποίηση για αλλαγή στη λίστα με τις διαθέσιμες συσκευές τότε ζητάει αυτή τη λίστα εκ νέου.

4.5 NETWORK SERVICE DISCOVERY

Η τεχνολογία Network Service Discovery (NSD) δίνει τη δυνατότητα σε μία εφαρμογή να βρει άλλες συσκευές σε ένα τοπικό δίκτυο, οι οποίες υποστηρίζουν τις ίδιες υπηρεσίες. Οι συσκευές αυτές μπορεί να είναι από κινητά τηλέφωνα μέχρι εκτυπωτές, κάμερες, HTTPS εξυπηρετητές κτλ. Το NSD υλοποιεί έναν μηχανισμό βασιζόμενο στο DNS (DNS-SD) ο οποίος επιτρέπει σε μία εφαρμογή Android να ζητήσει υπηρεσίες καθορίζοντας τον τύπο αυτής της υπηρεσίας αλλά και το όνομα της συσκευής που παρέχει την επιθυμητή υπηρεσία. Το DNS-SD υποστηρίζεται τόσο στο Android όσο και σε άλλες πλατφόρμες για έξυπνα κινητά τηλέφωνα. Μία συσκευή που υποστηρίζει NSD μπορεί να δημοσιεύσει μία υπηρεσία σε ένα τοπικό δίκτυο. Άλλες συσκευές στο ίδιο τοπικό δίκτυο μπορούν να αναγνωρίσουν αυτή τη συσκευή, εφόσον ενδιαφέρονται για αυτή την υπηρεσία, και να συνδεθούν.

Δημοσίευση υπηρεσίας στο δίκτυο

Η δημοσίευση μιας υπηρεσίας στο δίκτυο επιτυγχάνεται μέσω του σχετικού API. Στην περίπτωση του DoSdroid η συσκευή εξυπηρετητής είναι αυτή που θα δημοσιεύσει μία υπηρεσία στο τοπικό δίκτυο, ενώ οι ενδιαφερόμενοι (πελάτες) θα συνδεθούν στην υπηρεσία. Η διαδικασία πραγματοποιείται στην κλάση NsdServer. Ο αρχικοποιεί μία Socket και αποθηκεύει τον αριθμό της πόρτας στην οποία θα δέχεται αιτήσεις σύνδεσης από πελάτες.

```
private void initServerSocketAndPort() {
        try {
            serverSocket = new ServerSocket(0); // system chooses an
available port
            localPort = serverSocket.getLocalPort();
        }
}
```

Η δημοσίευση γίνεται στην μέθοδο registerService(). Οι πληροφορίες της υπηρεσίας προς δημοσίευση ενθυλακώνονται στο αντικείμενο NsdServiceInfo. Όνομα τύπος της υπηρεσίας καθώς και η πόρτα.

```
private void registerService() {
        NsdManager manager = (NsdManager)
context.getSystemService(NSD_SERVICE);
        manager.registerService(getNsdServiceInfo(),
NsdManager.PROTOCOL_DNS_SD, registrationListener);
    }

    private NsdServiceInfo getNsdServiceInfo() {
        NsdServiceInfo serviceInfo = new NsdServiceInfo();
        serviceInfo.setServiceName(INITIAL_SERVICE_NAME);
        serviceInfo.setServiceType(SERVICE_TYPE);
        serviceInfo.setPort(localPort);
        return serviceInfo;
}
```

Το αντικείμενο registrationListener είναι αυτό που θα ειδοποιήσει για τυχόν συμβάντα που έχουν να κάνουν με τη δημοσιοποίηση. Ο παρακάτω κώδικας δείχνει τις ενέργειες που πραγματοποιούνται σε περιπτώσεις επιτυχίας ή αποτυχίας δημοσίευσης της υπηρεσίας.

```
registrationListener = new NsdManager.RegistrationListener() {

            @Override
            public void onServiceRegistered(NsdServiceInfo
nsdServiceInfo) {
                nsdServiceName = nsdServiceInfo.getServiceName();
                uploadAttack();
                acceptClientThread.start();
                statusListener.onServerRunning(attack.getWebsite());
            }
            @Override
            public void onRegistrationFailed(NsdServiceInfo
nsdServiceInfo, int i) {
                Log.e(TAG, "Nsd registration failed");
                statusListener.onServerError(attack.getWebsite());
            }

            @Override
            public void onUnregistrationFailed(NsdServiceInfo
nsdServiceInfo, int i) {
                Log.e(TAG, "Nsd unregistration failed");
            }

            @Override
            public void onServiceUnregistered(NsdServiceInfo
nsdServiceInfo) {
                Log.d(TAG, "Nsd service unregistered successfully");
                statusListener.onServerStopped(attack.getWebsite());
                NsdServer.super.stop();
            }
};
```

Εντοπισμός υπηρεσιών στο δίκτυο

Για τον εντοπισμό υπηρεσιών, όπως και τη δημοσιοποίηση, χρειάζονται ένα αντικείμενο που θα αναφέρει την πρόοδο της διαδικασίας και η εκτέλεση της discoverServices(), η οποία εκκινεί την όλη διαδικασία.

Στο DoSdroid, όπως προαναφέρθηκε, μόνο οι εξυπηρετητές έχουν τη δυνατότητα δημοσίευσης υπηρεσιών. Από την άλλη πλευρά οι πελάτες γνωρίζουν ποια υπηρεσία τους ενδιαφέρει, αυτή του εξυπηρετητή, και προσπαθούν να εντοπίσουν τον τελευταίο. Την ευθύνη αυτή αναλαμβάνει η κλάση NsdServerConnection. Η εκκίνηση του εντοπισμού αποτελεί το πρώτο βήμα της διαδικασίας σύνδεσης με τον server. Στην προκειμένη περίπτωση το αντικείμενο που θα αναφέρει την πρόοδο της διαδικασίας είναι το ίδιο το στιγμιότυπο της NsdServerConnection (τρίτη παράμετρος στην discoverServices()).

```
public void connectToServer() {
        manager.discoverServices(Attacks.getNsdServiceType(attack),
NsdManager.PROTOCOL_DNS_SD, this);
}
```

Παρακάτω διακρίνονται οι ενέργειες που πραγματοποιεί η NsdServerConnection όταν λάβει ένα αποτέλεσμα για τη διαδικασία της ανακάλυψης.

```
@Override
    public void onServiceLost(NsdServiceInfo nsdServiceInfo) {
        Log.d(TAG, "Service lost");
    }

    @Override
    public void onStartDiscoveryFailed(String s, int errorCode) {
        Log.d(TAG, "Start discovery failed with error code: " +
errorCode);
        connectionListener.onServerConnectionError();
    }

    @Override
    public void onStopDiscoveryFailed(String s, int errorCode) {
        Log.d(TAG, "Stop discovery failed with error code: " +
errorCode);
    }

    @Override
    public void onValidServerResponse() {
        connectionListener.onServerConnection();
    }

    @Override
    public void onErrorServerResponse() {
        connectionListener.onServerConnectionError();
}
```

Σε περίπτωση που η συσκευή λάβει θετική απάντηση από τον εξυπηρετητή τότε η NsdServerConnection αναφέρει το αποτέλεσμα της σύνδεσης στον Client. Σε διαφορετική περίπτωση αναφέρει σφάλμα ή απλώς καταγράφει το συμβάν (log).

4.6 BLUETOOTH

Το Android παρέχει υποστήριξη για την τεχνολογία Bluetooth, η οποία επιτρέπει την ανταλλαγή δεδομένων μιας συσκευής με άλλες συσκευές (που υποστηρίζουν Bluetooth), ασύρματα. Η πρόσβαση στις λειτουργίες του Bluetooth επιτυγχάνεται μέσω των Android Bluetooth APIs.

Bluetooth APIs

Χρησιμοποιώντας τα συγκεκριμένα API μία εφαρμογή Android μπορεί να πραγματοποιήσει τα εξής:

- Αναζήτηση συσκευών Bluetooth
- Λήψη λίστας συσκευών που είχαν συνδεθεί μέσω Bluetooth στο παρελθόν
- Εγκαθίδρυση καναλιού τύπου RFCOMM
- Σύνδεση άλλες συσκευές
- Μεταφορά δεδομένων από και προς άλλες συσκευές
- Διαχείριση πολλαπλών συνδέσεων

Πρωταγωνιστικό ρόλο στα συγκεκριμένα APIs παίζει η κλάση BluetoothAdapter. Χρησιμοποιώντας τον BluetoothAdapter μία εφαρμογή μπορεί να βρει συσκευές Bluetooth έπειτα από νέα αναζήτηση ή από τη λίστα συσκευών που είχαν συνδεθεί στο παρελθόν.

Εύρεση συσκευών

Η εύρεση συσκευών είναι μια διαδικασία που σαρώνει την τοπική περιοχή για συσκευές Bluetooth, ενώ παράλληλα ζητάει διάφορες πληροφορίες για κάθε μία από αυτές.

Παρόλα αυτά, μια κοντινή συσκευή Bluetooth ανταποκρίνεται σε ένα αίτημα εντοπισμού μόνο εάν εκείνη τη στιγμή βρίσκεται σε κατάσταση “ανιχνεύσιμη” (discoverable). Εάν ισχύει το τελευταίο τότε η συσκευή ανταποκρίνεται στο αίτημα εντοπισμού, μοιράζοντας ορισμένες πληροφορίες, όπως το όνομα, την κλάση καθώς και τη μοναδική διεύθυνση MAC (MAC Address). Χρησιμοποιώντας αυτές τις πληροφορίες, η συσκευή που εκτελεί τη διαδικασία εντοπισμού μπορεί στη συνέχεια να επιλέξει να ξεκινήσει μια σύνδεση με τη συσκευή που εντοπίστηκε.

Μόλις γίνει σύνδεση για πρώτη φορά με μια απομακρυσμένη συσκευή, εμφανίζεται αυτόματα μια αίτηση ζευγαρώματος (pairing) στον χρήστη. Κατά τη διάρκεια του ζευγαρώματος οι βασικές πληροφορίες σχετικά με τη συγκεκριμένη συσκευή - όπως το όνομα, η τάξη και η διεύθυνση MAC της συσκευής - αποθηκεύονται και μπορούν να διαβαστούν χρησιμοποιώντας τα Bluetooth API.

Εάν μία συσκευή γνωρίζει τη διεύθυνση MAC μιας άλλης τότε μπορεί να ξεκινήσει μια σύνδεση μαζί της ανά πάσα στιγμή, χωρίς να γίνει εντοπισμός, υποθέτοντας φυσικά ότι η συσκευή εξακολουθεί να βρίσκεται εντός εμβέλειας.

Αξίζει να σημειωθεί η διαφορά της σύνδεσης-ζευγαρώματος μεταξύ συσκευών Bluetooth:

- Το ότι ζευγάρωσαν σημαίνει ότι δύο συσκευές γνωρίζουν την ύπαρξη η μια της άλλης. Μοιράζονται ένα κλειδί σύνδεσης που μπορεί να χρησιμοποιηθεί για έλεγχο ταυτότητας και είναι σε θέση να δημιουργήσουν μια κρυπτογραφημένη σύνδεση μεταξύ τους.
- Το ότι συνδέθηκαν σημαίνει ότι οι συσκευές μοιράζονται επί του παρόντος ένα κανάλι RFCOMM και είναι σε θέση να μεταδίδουν δεδομένα μεταξύ τους. Τα Bluetooth APIs απαιτούν το ζευγάρωμα των συσκευών πριν από τη δημιουργία μιας σύνδεσης RFCOMM. Αυτή πραγματοποιείται αυτόματα κατά την εκκίνηση μιας κρυπτογραφημένης σύνδεσης.

Λήψη διαζευγμένων συσκευών

Πριν καν μπει στη διαδικασία να εντοπίσει συσκευές Bluetooth, μία εφαρμογή μπορεί να ζητήσει τη λίστα με τις διαζευγμένες συσκευές. Ο παρακάτω κώδικας λαμβάνει τη λίστα από τον BluetoothAdapter. Το αντικείμενο BluetoothDevice δίνει πρόσβαση στο όνομα της συσκευής καθώς και στην διεύθυνση MAC.

```
Set<BluetoothDevice> pairedDevices =
bluetoothAdapter.getBondedDevices();
if (pairedDevices.size() > 0) {
```

```
    // There are paired devices. Get the name and address of each
paired device.
    for (BluetoothDevice device : pairedDevices) {
        String deviceName = device.getName();
        String deviceHardwareAddress = device.getAddress(); // MAC
address
    }
}
```

Η εκκίνηση της διαδικασίας αναζήτησης επιτυγχάνεται μέσω της εκτέλεσης της startDiscovery(). Η διαδικασία είναι ασύγχρονη και επιστρέφει μια τιμή boolean που καταδεικνύει αν η διαδικασία ξεκίνησε ή όχι. Η διαδικασία της σάρωσης κρατάει περίπου 12 δευτερόλεπτα.

Ειδοποιήσεις

Η ανακάλυψη μίας συσκευής είναι ένα συμβάν που εκπέμπεται από το σύστημα. Μία εφαρμογή μπορεί να λάβει πληροφορίες για την συσκευή εφόσον διαθέτει τον κατάλληλο BroadcastReceiver που πληροφορείται από το συγκεκριμένο γεγονός. Στο DoSdroid η συσκευή πελάτης είναι αυτή που αναζητά συσκευές, στην προσπάθεια της να εντοπίσει τον εξυπηρετητή. Παρακάτω φαίνεται ο BroadcastReceiver που χρησιμοποιεί η τάξη BluetoothServerConnection για την εύρεση συσκευών.

```
deviceDiscoveryReceiver = new BroadcastReceiver() {
            private boolean firstTimeDiscoveredServer = true;
            @Override
            public void onReceive(Context context, Intent intent) {
                boolean foundDevice =
BluetoothDevice.ACTION_FOUND.equals(intent.getAction());
                if (foundDevice) {
                    BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                    if (isServer(device) && firstTimeDiscoveredServer)
{

BluetoothConnectionThread.startInstance(device,
Attacks.getHostUUID(attack), BluetoothServerConnection.this);
                        firstTimeDiscoveredServer = false;
                    }
                }
            }

            private boolean isServer(BluetoothDevice device) {
                String deviceAddress = device.getAddress(); // MAC
address
                String serverAddress =
Attacks.getHostMacAddress(attack);
                return deviceAddress.equals(serverAddress);
            }
};
```

Όπως φαίνεται από τον κώδικα, όταν βρεθεί η συσκευή εξυπηρετητής τότε ξεκινάει ένα νέο νήμα που αναλαμβάνει τη σύνδεση.

4.7 ΕΠΙΘΈΣΕΙΣ

Οι επιθέσεις που εκτελούνται χρησιμοποιώντας την εφαρμογή DoSdroid είναι τύπου Denial of Service (DoS), όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο.

Η εκτέλεση γίνεται από τις συσκευές πελάτες και μέσω του πρωτοκόλλου HTTP. Στόχος η παρενόχληση εξυπηρετητών-στόχων που λειτουργούν μέσω του προαναφερθέντος πρωτοκόλλου, πάντα για εκπαιδευτικούς σκοπούς.

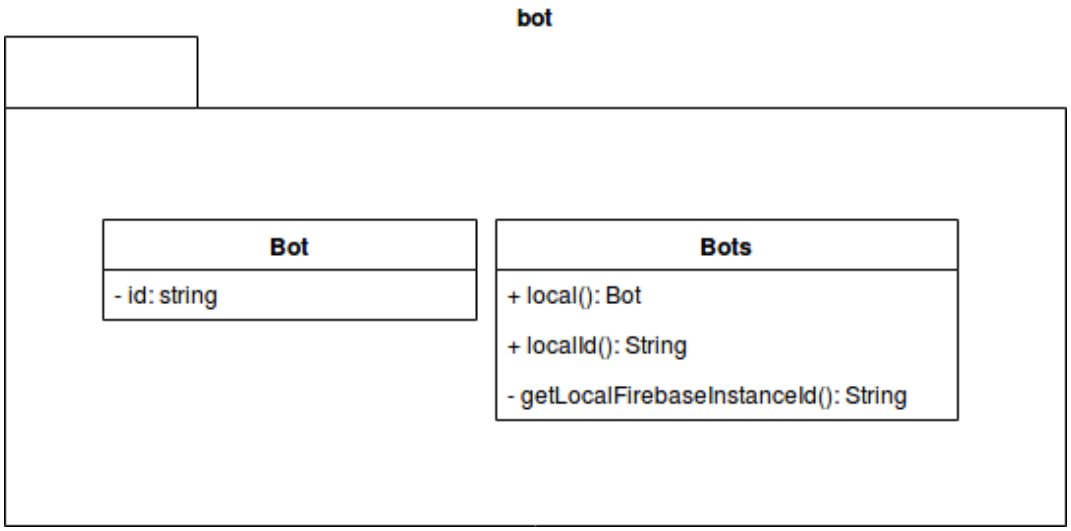
Attack

Την έννοια της επίθεσης στην εφαρμογή ενσαρκώνει η κλάση Attack. Η βοηθητική κλάση Attacks περιέχει λειτουργικότητα για τον μεταχείριση στιγμιοτύπων τύπου Attack. Ακολουθεί το σχετικό διάγραμμα:

Bot

Μία επίθεση περιλαμβάνει ένα δίκτυο από συσκευές που θα συμμετάσχουν στην επίθεση. Ο όρος που χρησιμοποιείται για αυτό το δίκτυο είναι ο “Botnet”, ενώ η κάθε μία συσκευή σε αυτό το δίκτυο ονομάζεται “Bot”.

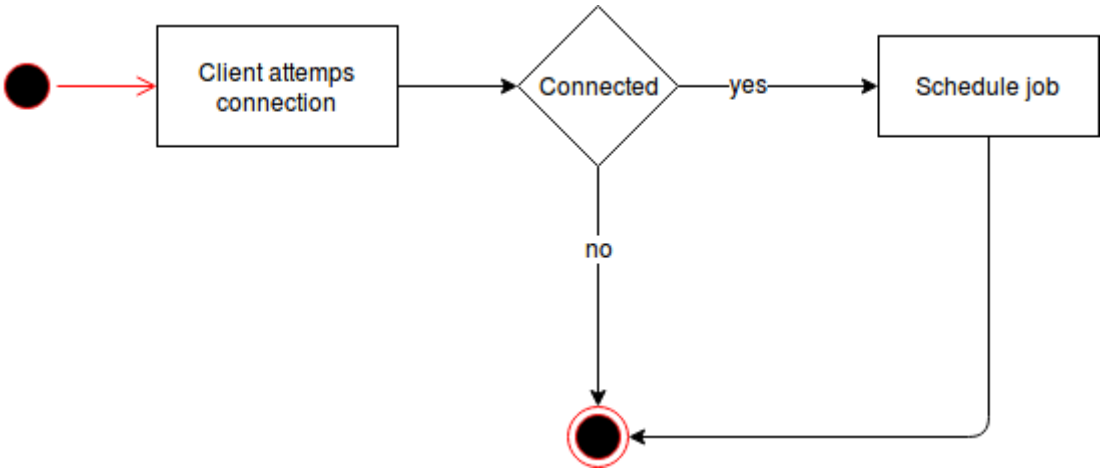
Η της ομώνυμη κλάση Bot ενσαρκώνει την ιδέα του ενός κόμβου μέσα στο σύστημα. Ακολούθως, στο ίδιο πακέτο, βρίσκεται και η βοηθητική κλάση Bots για τη μεταχείριση στιγμιοτύπων τάξης Bot.



Συγχρονισμός

Σε κάθε επίθεση έχει οριστεί ένα χρονοδιάγραμμα εκκίνησης από τον δημιουργό. Οι συσκευές που συμμετέχουν στην επίθεση έχουν πρόσβαση σε αυτή την πληροφορία και ως εκ τούτου μπορούν να συγχρονίσουν την εκκίνηση των επιθέσεων, για να πληγεί ο στόχος τα μέγιστα. Το παραπάνω σενάριο μπορεί να υλοποιηθεί στο Android με τη χρήση του JobScheduler, ωστόσο το API είναι διαθέσιμο για τις εκδόσεις 21 και άνω του λειτουργικού. Η εναλλακτική βρέθηκε στην βιβλιοθήκη ανοιχτού κώδικα Firebase JobDispatcher, μία βιβλιοθήκη η οποία προγραμματίζει εργασίες στο μπαγκγκράουντ.

Ο προγραμματισμός μίας επίθεσης στο μέλλον φαίνεται στο παρακάτω διάγραμμα:



Έναρξη επίθεσης

Όταν έρθει η ώρα της έναρξης το σύστημα εκκινεί μία υπηρεσία, συγκεκριμένα ένα Android Service το οποίο ενημερώνει με ειδοποίηση (Android Notification) το χρήστη για την έναρξη της επίθεσης και φυσικά εκτελεί τη ρουτίνα που προσβάλλει τον αντίπαλο στόχο.

Το στοιχείο του συστήματος που εκκινεί αυτή την υπηρεσία είναι το AttackJobService, ουσιαστικά η υλοποίηση του Firebase JobDispatcher που αναφέρθηκε προηγουμένως. Η υπηρεσία καθαυτή είναι η AttackLaunchService. Αυτό φαίνεται και στο παρακάτω τμήμα κώδικα:

```
public boolean onStartJob(@NonNull JobParameters job) {
        AttackLaunchService.Action.launch(job.getExtras(), this);
        jobFinished(job,false);
```

```
        return false; // Answers to the question: "Is there still work
going on?"
}
```

Όπως περιγράφηκε πριν, η AttackLaunchService προβάλλει την ειδοποίηση στον χρήστη και μετά ξεκινάει τη ρουτίνα που εκτελεί ουσιαστικά την επίθεση. Η ρουτίνα αυτή τρέχει φυσικά σε ξεχωριστό νήμα επεξεργασίας και περικλείεται στην κλάση AttackScript.

Κατά την εκτέλεση της επίθεσης η ρουτίνα αιτείται σύνδεση με τον εξυπηρετητή-στόχο. Ο server-στόχος είναι, στην καλύτερη περίπτωση, προγραμματισμένος να εξυπηρετεί όλους τους αιτούντες. Αφού η σύνδεση επιτευχθεί η ρουτίνα διαβάζει από τον server την ιστοσελίδας, θα μπορούσε να είναι ένα HTML αρχείο. Έπειτα κλείνει τη σύνδεση.

Η παραπάνω ρουτίνα θα επαναλαμβάνεται για πάντα μέχρι

- Ο χρήστης δημιουργός να ακυρώσει την επίθεση.
- Ο χρήστης πελάτης να εγκαταλείψει την επίθεση.
- Κατά το συμβάν ενός εσωτερικού σφάλματος, όπως ο τερματισμός της διεργασίας της εφαρμογής.

Το παραπάνω αποδίδεται στο τμήμα κώδικα παρακάτω:

```
public void run() {
       while (!stopped.get())
            readUrl();
}
```

Βάση δεδομένων

Όλες οι δημιουργηθείσες επιθέσεις αποθηκεύονται σε μία βάση δεδομένων στο διαδίκτυο. Με αυτό τον τρόπο δημοσιεύονται στους χρήστες που θέλουν να συμμετάσχουν σε μία ή περισσότερες από αυτές.

Η κλάση AttackRepository είναι υπεύθυνη για την διαχείριση της αποθήκευσης, ενημέρωσης και διαγραφής των επιθέσεων μέσα στο πλαίσιο του DoSdroid. Το είδος της βάσης δεδομένων που θα χρησιμοποιηθεί εξαρτάται πάντα από την υλοποίηση της κλάσης AttackRepository.

Στο DoSdroid αυτή η υλοποίηση παρέχεται από την τάξη FirebaseRepository, η οποία χρησιμοποιεί την Firebase Realtime Database για να αποθηκεύσει της επιθέσεις σε μία δομή δέντρου (JSON). Γονέας είναι ένας κόμβος που ονομάζεται “attacks” και παιδιά είναι οι επιθές.

5. Συμπεράσματα και Μελλοντικές Επεκτάσεις

Τα έξυπνα κινητά τηλέφωνα έχουν αναπτυχθεί σε τέτοιο βαθμό που τείνουν να αντικαταστήσουν τους παραδοσιακούς επιτραπέζιους (ή φορητούς) υπολογιστές. Αυτό είναι το οφθαλμοφανές συμπέρασμα των τελευταίων ετών.

Με την υπολογιστική τους ισχύ να αυξάνεται ραγδαία χρόνο με το χρόνο, μήνα με το μήνα, κάποιος μόνο να φανταστεί μπορεί τα αποτελέσματα που μπορεί να επιφέρει ο συνδυασμός αυτής της δύναμης. Οι τεχνολογίες δικτύωσης που προσφέρονται από το Android βοηθούν προς αυτή την κατεύθυνση.

Οι συσκευές που μπορούν να συνδυαστούν δεν χρειάζεται να είναι απαραιτήτως έξυπνα τηλέφωνα. Ούτε καν να “τρέχουν” το ίδιο λειτουργικό σύστημα. Η δικτύωση μπορεί να γίνει σε επίπεδο εφαρμογής και εκεί θα ορίζεται ο τρόπος διασύνδεσης των συσκευών, αρκεί βεβαίως να διαθέτουν το απαραίτητο υλικό για την επίτευξη αυτής.

Κλείνοντας, όσον αφορά το DosDroid θα πρέπει να γνωστοποιηθεί πως έχει ήδη γίνει μία επέκταση στην εφαρμογή, έπειτα από αίτημα του επιβλέποντα. Η αρχική έκδοση βρίσκεται εδώ. Η τελευταία έκδοση πρόσθεσε το χαρακτηριστικό του συγχρονισμού των επιθέσεων.

Σίγουρα υπάρχει χώρος για βελτίωση και επέκταση, Θα μπορούσε, λόγου χάρη, να βελτιωθεί ο τρόπος που διεξάγονται οι επιθέσεις, ώστε να είναι πιο αποτελεσματικές ή θα μπορούσαν να προστεθούν επιπλέον χαρακτηριστικά στην εφαρμογή.

Αυτή άλλωστε είναι η φύση του λογισμικού, η διαρκής αλλαγή!

6. Βιβλιογραφία

Connectivity. Android Developers. https://developer.android.com/guide/topics/connectivity. 15.10.2018.

MVP and MVC in Android. 2015. TechYourChance. https://www.techyourchance.com/mvp-mvc-android-1/. 12.11.2018.

Activities in Android are not UI Elements. 2015. TechYourChance. https://www.techyourchance.com/activities-android/. 2.10.2018.

Martin, Robert C.. 2009. Clean Code: A handbook of agile software craftsmanship. Upper Saddle River, NJ: Prentice Hall.

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα