Manuale GpsSpoofReveal

Autore

Giovanni Carollo

Email

giovanni.carollo.3@studenti.unipd.it

Data

10/9/2019

Versione Progetto

3.2.1

Indice

Capitolo 1

Informazioni generali

1.1 Descrizione del progetto

Gli obiettivi di GpsSpoofReveal sono la rilevazione di un attacco di spoofing del segnale GNSS e lo sviluppo di una solida base concettuale per eventuali progetti futuri. Lo spoofing del segnale GNSS permette la falsificazione della posizione del dispositivo della vittima. Negli ultimi anni questo argomento sta diventando di fondamentale importanza in diversi settori industriali e di ricerca.

1.2 Utenza di riferimento

Il manuale è un utile riferimento a tutti coloro che intendono approfondire o migliorare il funzionamento di GpsSpoofReveal.

1.3 Organizzazione del manuale

Il manuale è suddiviso in cinque capitoli.

- *Informazioni generali*: fornisce informazioni riguardo gli obiettivi di GpsSpoofReveal, l'utenza di riferimento e l'organizzazione del manuale.
- *Panoramica sul progetto*: fornisce una descrizione generale del funzionamento di GpsSpoofReveal.
- *Interfaccia utente*: fornisce una descrizione dettagliata dell'interfaccia utente dell'applicazione.

- *Infrastruttura server*: fornisce una descrizione dettagliata dell'infrastruttura server e della sua implementazione.
- *Installazione e analisi dell'applicazione*: fornisce una descrizione generale riguardo l'installazione e la programmazione dell'applicazione.

Le ultime due sezioni sono indicate principalmente a un'utenza esperta. In particolare verranno trattati questi argomenti: l'installazione di un web server e la programmazione di un'applicazione Android in Java.

Capitolo 2

Panoramica sul progetto

2.1 Struttura

GpsSpoofReveal è composto di due parti fondamentali: un'applicazione Android e un programma nel server. L'applicazione funziona solamente in smartphone con Android 7 o superiore e che permettono la lettura dei dati non elaborati del messaggio di navigazione (vedi `https://developer.android.com/guide/topics/sensors/gnss`). Nel caso in cui uno smartphone non permetta la lettura dei dati non elaborati del messaggio di navigazione l'affidabilità del controllo svolto dall'applicazione sarà limitato. Il server è configurato in modo che funzioni da web server, deve perciò rispondere a richieste HTTP. In assenza del web server o in caso in cui esso non sia raggiungibile l'applicazione non potrà funzionare correttamente.

2.2 Caratteristiche

GpsSpoofReveal permette di individuare un'azione di spoofing ai danni di uno smartphone Android. Inoltre l'applicazione, chiamata anch'essa GpsSpoofReveal, consente l'analisi e il salvataggio dei dati non elaborati e dei messaggi di navigazione ricevuti dai satelliti. Al fine di rilevare un attacco di spoofing vengono effettuate tre tipologie di controllo sul segnale GNSS ricevuto dallo smartphone:

- presenza del messaggio di navigazione nel segnale ricevuto,
- visibilità dei satelliti,
- correttezza delle effemeridi.

2.2.1 Presenza del messaggio di navigazione nel segnale ricevuto

L'applicazione controlla la presenza del messaggio di navigazione in ogni segnale GNSS ricevuto dallo smartphone. L'assenza del messaggio di navigazione costituisce un chiaro segnale di spoofing. In questo controllo non è richiesta la lettura dei dati non elaborati del messaggio di navigazione.

2.2.2 Visibilità dei satelliti

L'applicazione verifica che lo smartphone non rilevi satelliti in eccesso. Per svolgere questo controllo l'applicazione calcola longitudine e latitudine dello smartphone e crea una lista contenente ID, angolo di azimut e angolo di elevazione di ogni satellite visibile. L'applicazione invia poi una richiesta HTTP, contenente come parametro le coordinate dello smartphone, al web server. Il web server risponde con una lista contenente ID, angolo di azimut e angolo di elevazione dei satelliti che lo smartphone dovrebbe rilevare dalla posizione inviata (vedi Fig. 2.1). Ricevuta la risposta l'applicazione verifica che ogni ID presente nella lista dei satelliti visibili abbia un corrispettivo nella lista inviata dal server. L'applicazione cioè controlla che la lista dei satelliti visibili sia un sottoinsieme della lista inviata dal server. La presenza di un satellite in eccesso costituisce un segnale di spoofing. In questo controllo non è richiesta la lettura dei dati non elaborati del messaggio di navigazione.

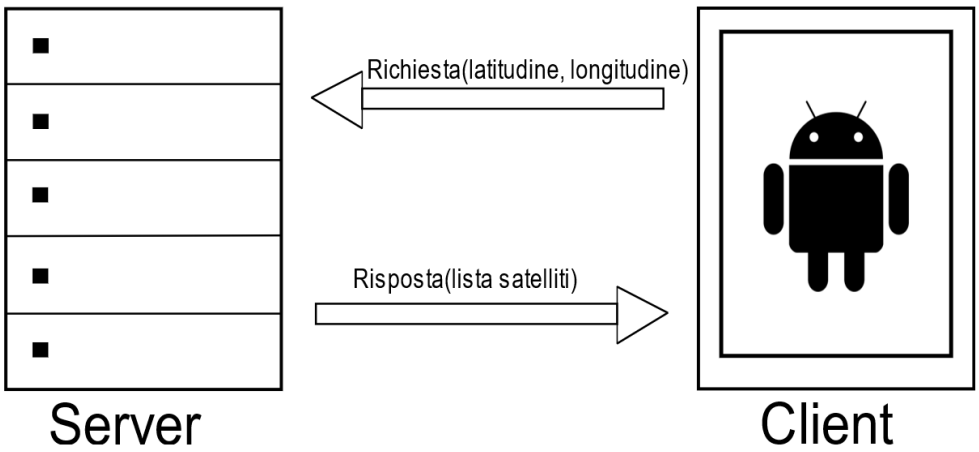

Figura 2.1: Modello Client-Server.

2.2.3 Correttezza delle effemeridi

L'applicazione verifica che l'angolo di azimut e l'angolo di elevazione di ogni satellite visibile siano uguali all'angolo di azimut e all'angolo di elevazione dei satelliti presenti nella lista inviata dal server, a meno di un margine d'errore. In questo controllo non è richiesta la lettura dei dati non elaborati del messaggio di navigazione. Viene poi eseguito un controllo di correttezza sulle effemeridi ricevute dai satelliti. In particolare vengono analizzati i seguenti parametri:

- *Week*: conta il numero di settimane dall'entrata in funzione della costellazione GPS. Questo parametro deve essere uguale per ogni satellite.
- *IODC (Issue of Data Clock)* e *IODE (Issue of Data Ephemeris)*: è il numero di emissione del messaggio di navigazione. In condizioni standard Iodc e Iode devono essere uguali all'interno di un unico messaggio di navigazione e sempre compresi tra 0 e 1023. Ogni messaggio di navigazione solitamente ha Iodc e Iode differenti rispetto agli altri.
- *TOC (Time of Clock)* e *TOE (Time of Ephemeris)*: è il tempo di emissione del messaggio di navigazione. In condizioni standard TOC e TOE devono essere uguali. Per acquisizioni inferiori a un'ora TOC e TOE devono essere uguali per ogni messaggio di navigazione.

Per determinare i dati relativi alle effemeridi di un satellite l'applicazione necessita dei primi tre subframe di uno dei qualsiasi frame inviati dallo stesso satellite. In generale l'applicazione non è in grado di ricavare le effemeridi per ogni satellite. L'analisi viene quindi effettuata unicamente per i satelliti dai quali è stato possibile ricavare le effemeridi. Per poter eseguire questo controllo lo smartphone deve permettere la lettura dei dati non elaborati del messaggio di navigazione.

Capitolo 3

Interfaccia utente

In questo paragrafo viene analizzata l'interfaccia utente dell'applicazione. Il layout dell'applicazione può cambiare in base al modello di smartphone nel quale l'applicazione è installata.

3.1 Primo avvio dell'applicazione

All'apertura dell'app verrà visualizzato un messaggio richiedente l'accettazione del permesso di accesso ai dati relativi alla posizione del dispositivo da parte dell'utente. Il layout di questo messaggio varia in base alla versione di Android utilizzata (vedi Fig. 3.1).

Figura 3.1: Autorizzazione posizione per Android 9.

- *Non chiedermelo più*: mettendo la spunta sarà poi possibile selezionare solamente Nega. Agli avvii successivi l'applicazione non mostrerà più questo avviso. L'applicazione non potrà funzionare.

- *Nega*: nega l'accesso ai dati relativi alla posizione all'applicazione durante questa esecuzione. L'applicazione non potrà funzionare.
- *Consenti*: consente all'applicazione di ricevere i dati riguardanti la posizione del dispositivo. Agli avvii successivi l'applicazione non mostrerà più questo avviso.

È possibile gestire le autorizzazioni dell'applicazione anche dal menù applicazioni presente nelle impostazioni di Android. Da tale menù è possibile togliere l'autorizzazione ad accedere alla posizione, nel caso in cui si abbia dato il consenso in precedenza, o dare l'autorizzazione all'applicazione nel caso in cui si abbia negato il consenso in precedenza. Questo menù risulta di fondamentale importanza nel caso in cui si neghi l'autorizzazione con il campo *Non chiedermelo più* spuntato.

3.2 Schermata iniziale

Il layout della schermata iniziale è presente alla pagina successiva in Fig. 3.2

- *START*: avvia l'acquisizione della posizione e la rilevazione dei satelliti GPS visibili dal dispositivo.
- *Timer*: consente, dopo l'avvio, di visualizzare il tempo totale di acquisizione dei dati.
- *INFO*: permette di visualizzare nella console le informazioni fondamentali riguardanti il progetto come la versione dell'applicazione, il creatore e i recapiti.
- *Coordinate*: consente, dopo l'avvio, di visualizzare la longitudine e la latitudine del dispositivo.
- *List of visible Sat. GPS Svid*: consente, dopo l'avvio, di visualizzare l'ID dei satelliti GPS visibili dal dispositivo.
- *Console*: consente, dopo l'avvio, di visualizzare informazioni di carattere generale durante l'acquisizione dei dati.

3.3 Acquisizione dei dati

Il layout della schermata di acquisizione dati è presente alla pagina successiva in Fig. 3.3.

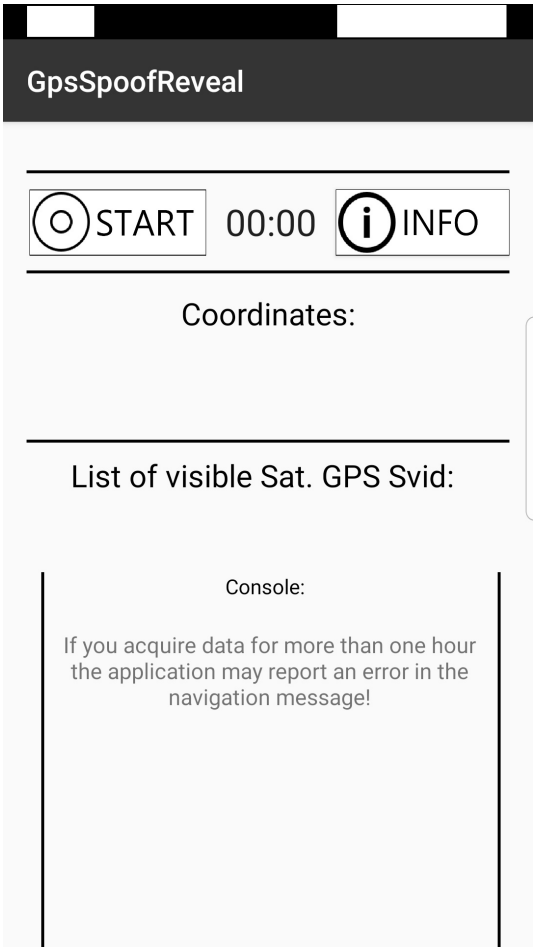

Figura 3.2: Schermata iniziale.

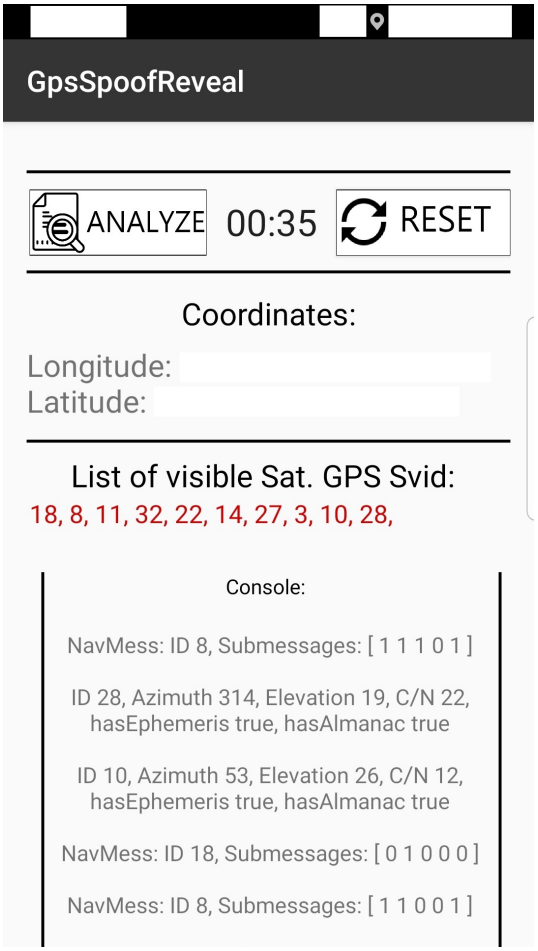

Figura 3.3: Acquisizione dati.

- *Simbolo di acquisizione della posizione*: una volta avviata l'acquisizione dei dati è possibile notare la presenza del simbolo che notifica l'utilizzo del ricevitore GNSS da parte dell'applicazione.

- *ANALYZE*: se premuto consente di analizzare i dati ricavati. Deve essere premuto dopo il tempo necessario da permettere al dispositivo di rilevare la propria posizione e di identificare un numero sufficiente di satelliti GPS. In caso contrario l'analisi dei dati sarà errata o non affidabile.

- *RESET*: se premuto consente di fermare e resettare l'acquisizione. Si tornerà quindi alla schermata iniziale.

All'interno della console è possibile visualizzare i dati relativi all'acquisizione che si sta effettuando. Alla rilevazione di un nuovo satellite il dispositivo fornisce i rispettivi ID, posizione e il C/N_0 *(Carrier-to-Noise Density,)* e informa se il segnale possiede il messaggio di navigazione e l'almanacco. Inoltre l'app notifica l'utente ogni qual volta riceve un subframe da un satellite.

3.4 Elaborazione dei dati

Il layout della schermata di elaborazione dati è presente alla pagina successiva in Fig. 3.4.

- *Simbolo rete Internet*: per permettere all'applicazione la comunicazione con il server e quindi l'elaborazione dei dati è fondamentale la presenza della connessione Internet. La tipologia di connessione Internet è indifferente.

- *Query*: permette la visualizzazione delle coordinate del dispositivo inviate al server.

- *GPS Satellites Orbital Parameters*: in questa sezione, sotto la voce *Supp. Vis.* sono presenti ID, angolo di azimut e angolo di elevazione relativi ai satelliti presenti nella lista inviata dal server. Alla voce *Visible* sono presenti ID, angolo di azimut e angolo di elevazione relativi ai satelliti presenti nella lista creata dall'applicazione. Sotto la voce *C/N0* è possibile visualizzare il rapporto tra la potenza del segnale ricevuto dai rispettivi satelliti e il rumore. Infine sotto la voce *result* l'applicazione fornisce l'esito della verifica per ogni satellite.

- *GPS Navigation Messages*: per ogni satellite visibile informa l'utente della ricezione di almeno un subframe. Nel caso in cui l'applicazione non abbia ricevuto nessun subframe da un satellite sarà presente la dicitura *MISS*.Se il satellite prevede l'invio del messaggio di navigazione, sotto la voce *Result* sarà presente la scritta *OK* in caso contrario sarà presente la scritta *NO*.

- *Conclusion*: visualizza il risultato dell'analisi. Tutti i possibili risultati saranno analizzati in seguito.

- *RETURN*: torna alla schermata di acquisizione.

- *TRY AGAIN*: permette, in caso di errore, di rinviare la richiesta al server.

- *VIEW NAVIGATION MESSAGE*: permette la visualizzazione delle effemeridi e dei subframe.

3.5 Visualizzazione del messaggio di navigazione

In questa sezione è possibile visualizzare l'ora di inizio dell'acquisizione, le coordinate dello smartphone, i dati non elaborati, le effemeridi e per acquisizioni non inferiori ai 12,5 minuti anche i dati relativi alla ionosfera. Nel caso in cui lo smartphone non permetta la lettura dei dati non elaborati del messaggio di navigazione questa sezione conterrà solamente l'ora di inizio dell'acquisizione e le coordinate. Il layout della schermata di visualizzazione dati è presente in Fig. 3.5.

Per salvare i dati all'interno del dispositivo è prima necessario accettare i permessi di accesso alla memoria. Il layout della schermata e le opzioni sono simili a quelle descritte nella sezione 3.1. I file verranno salvati all'interno di una cartella chiamata *GpsSpoofReveal*.

- *SAVE .TXT*: permette di generare un file di testo, nominato in base alla data di acquisizione dei dati, contenente le effemeridi e i subframe.

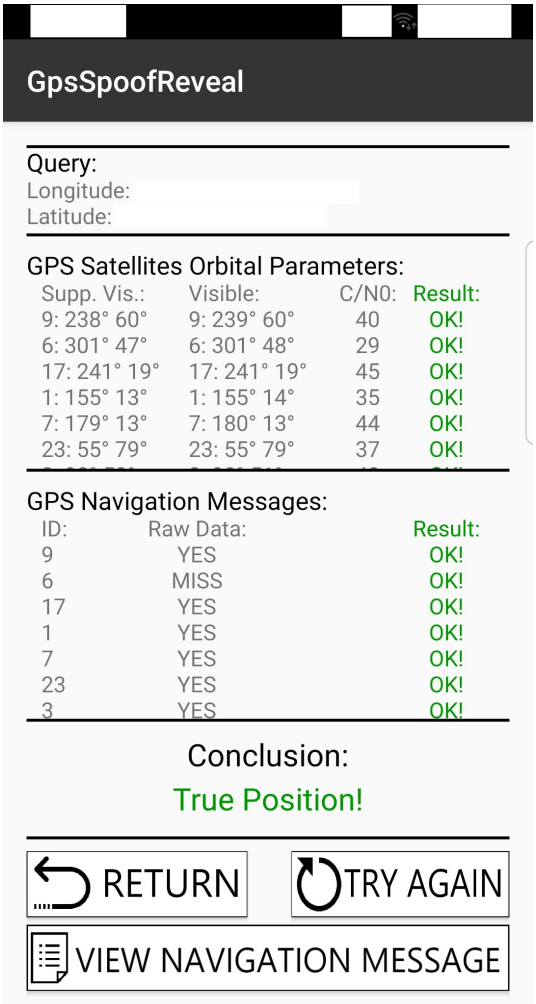

Figura 3.4: Elaborazione dati.

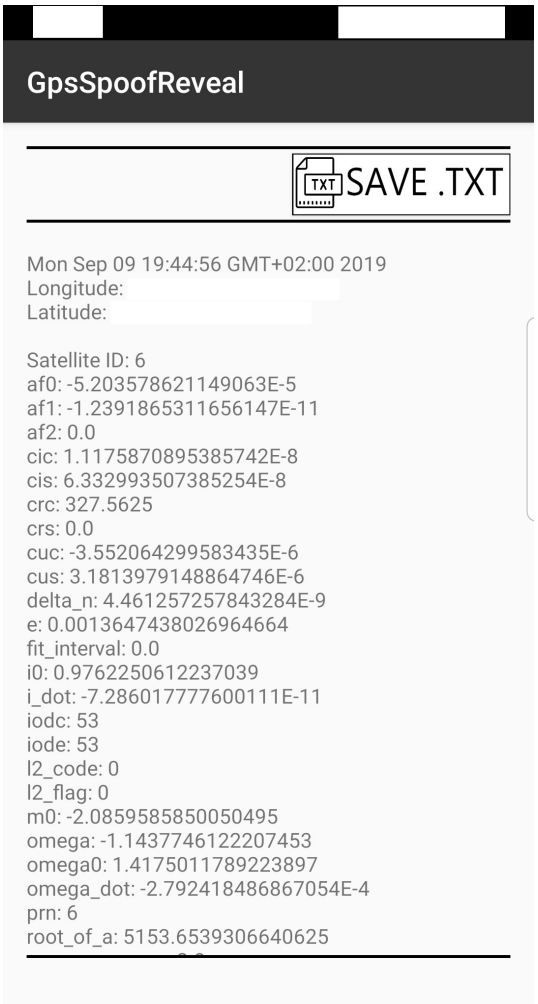

Figura 3.5: Visualizzazione dati.

3.6 Messaggi d'errore

Se l'acquisizione effettuata non presenta alcun errore, sotto la voce *Conclusion* della schermata di elaborazione dati sarà presente la scritta *True Position!*. Nel caso in cui almeno uno dei tre controlli descritti non viene superato sarà presente la scritta *False Position!* seguita da una breve descrizione dell'errore. L'applicazione invece si limiterà a restituire un semplice messaggio di avviso nei seguenti casi:

- l'applicazione ha rilevato meno di quattro satelliti,
- lo smartphone non è connesso a Internet,
- i dati della rilevazione sono corrotti o incompleti,
- il server non risponde.

In tutti i casi è comunque possibile visualizzare il messaggio di navigazione.

Capitolo 4

Interfaccia server

4.1 Premessa

Per l'implementazione del server è stato utilizzato Apache versione 2.4.39. Di seguito ne verrà descritta l'installazione per Windows 10, per Linux distribuzione Debian 9 e per Linux distribuzione Ubuntu.

È possibile suddividere il lavoro svolto dal server in due parti:

1. Gestione della comunicazione con il client.
2. Elaborazione delle coordinate e calcolo del risultato.

4.2 Gestione della comunicazione con il client

Il server deve gestire le richieste e le risposte HTTP da e verso i client. A tale scopo viene utilizzato un web server. In generale l'utente può utilizzare un qualsiasi web server.

Un generico messaggio di richiesta HTTP inviato dal client al server è così formato:

```
http://X.X.X.X/path/phpfunction.php?argument1=Latitude\&argument2=Longitude
```

Dove:

- *X.X.X.X*: è l'indirizzo IP del proprio server.
- *path*: è il percorso presso il quale si trovano le risorse utili al client.

- *phpfunction.php*: file PHP che permette l'esecuzione dell'applicativo JavaScript.
- *?argument1=Latitude&argument2=Longitude*: parametri forniti dal client.

Il messaggio di risposta fornito dal server è una lista contenente ID, angolo di azimut e angolo di elevazione di ogni satellite visibile. Il formato con cui questi elementi vengono inseriti nella lista è chiave-valore.

- PNR : ID
- Azimuth : angolo di azimut
- Elevation : angolo di elevazione

Il client per ricavare i valori dovrà effettuare il parsing del messaggio di risposta.

4.3 Elaborazione delle coordinate e generazione del risultato

Per fornire il messaggio di risposta al client il server utilizza due programmi:

1. *Applicativo JavaScript*: gestisce un file contenete le effemeridi aggiornate. Queste ultime vengono utilizzate, insieme alle coordinate fornite dal client, per creare il messaggio di risposta. A ogni nuova richiesta le effemeridi vengono aggiornate tramite Internet.
2. *Applicativo PHP*: ad ogni richiesta ricevuta apre un terminale nel quale esegue l'applicativo JavaScript. Inoltra poi il risultato ottenuto al client.

In Fig. 4.1 è presente uno schema riassuntivo del funzionamento del server.

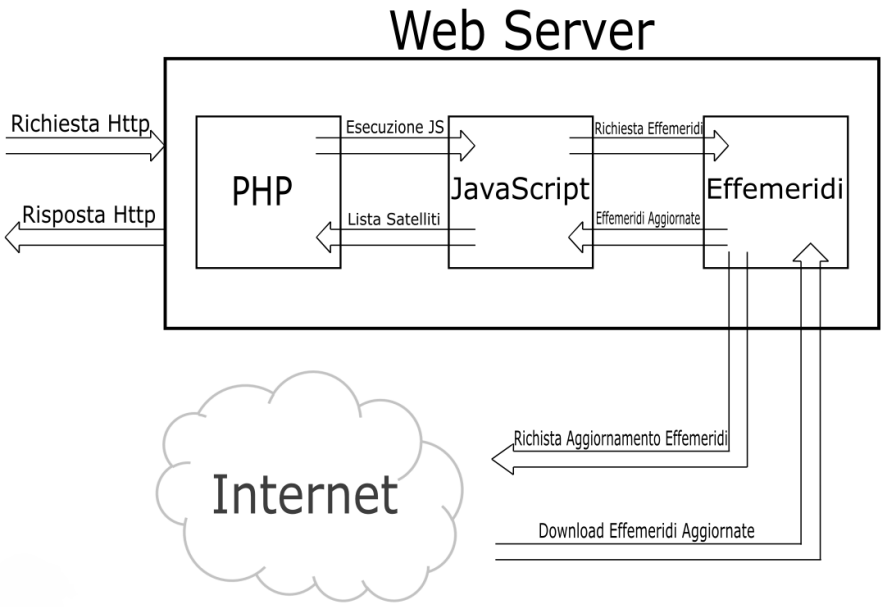

Figura 4.1: Schema logico del funzionamento del server.

4.4 Implementazione Apache su Windows 10

In questa sezione viene spiegato come installare e impostare correttamente il server per Windows 10.

4.4.1 Download e installazione di Apache

1. Scaricare Apache dal sito: `https://www.apachelounge.com/download/`

2. Per l'installazione seguire la procedura presente nel file ReadMe.

3. Per verificare la corretta installazione aprire il prompt dei comandi, andare nella cartella *C:\Apache24\bin* e digitare *"httpd.exe"* per avviare il server Apache. Successivamente digitare su un qualsiasi browser della propria macchina *"localhost/"* o in alternativa l'indirizzo IP privato della propria macchina. In entrambi i casi si deve visualizzare la scritta *It works!*.

4.4.2 Accesso al server da un client esterno alla rete locale

Terminata l'installazione è necessario permettere la comunicazione con il server anche ai client esterni alla rete locale.

1. Aprire la pagina di configurazione del proprio router da un browser qualsiasi. Di norma gli indirizzi IP privati standard dei router sono: 192.168.0.1, 192.168.1.1, 192.168.1.254.

2. La seguente procedura varia a seconda dell'azienda produttrice del router, verrà perciò esposta senza entrare nel dettaglio. Andare nella sezione in cui è possibile modificare la configurazione delle porte del proprio router.

3. Aggiungere un Virtual Server dotato di una connessione HTTP (quindi con porta interna ed esterna 80) che inoltri il traffico in ingresso verso l'indirizzo privato nel quale è attiva la propria macchina server.

4. Per testare il corretto funzionamento aprire un browser con un dispositivo collegato a Internet da una rete diversa da quella locale. Digitare sulla barra di ricerca degli URL: *"http://indirizzo_IP_pubblico/"* dove per *indirizzo_IP_pubblico* si intende l'indirizzo IP pubblico del proprio router visualizzabile presso la pagina di configurazione. Si deve visualizzare la scritta *It works!*.

È importante sottolineare che spesso gli indirizzi IP pubblici forniti dagli ISP ai normali utenti sono dinamici. Per questo motivo è necessario cambiare l'indirizzo IP della richiesta del client ogni volta che l'indirizzo IP del router al quale è collegato il server viene modificato.

A questo punto il server è in grado di ricevere anche le richieste provenienti dall'esterno della rete locale.

4.4.3 Download e installazione di PHP7.0

È necessario impostare il server in modo tale che possa gestire i file PHP.

1. Scaricare il file Zip di PHP7.0 versione *Thread Safe* dal sito ufficiale di PHP: `https://windows.php.net/download/`

2. Creare una cartella su *C:\PHP7* ed estrarre il file Zip al suo interno.

3. All'interno della cartella rinominare il file *php.ini-development* in *php.ini*.

4. Aprire il file rinominato con un editor di testo, cercare la riga *";extension_dir = "ext""* e rimuovere *";"*.

5. Cercare la lista delle estensioni e rimuovere il simbolo *";"* dalle seguenti estensioni:
   extension = curl, extension = fileinfo, extension = pgsql, extension = gettext, extension = pdo_sqlite, extension = gmp, extension=sockets, extension = shmop, extension = gd2, extension=sodium, extension=sqlite3, extension=tidy, extension=xmlrpc, extension=xsl

6. Aggiungere la voce *C:\PHP7* sulle variabili d'ambiente *Path* di Windows 10.

7. Per testare l'installazione di PHP7 aprire il prompt dei comandi e digitare *"php -v"*. Si deve visualizzare la versione di PHP installata sulla propria macchina. In caso contrario riavviare la macchina.

4.5 Implementazione Apache su Debian 9

In questa sezione viene spiegato come installare e impostare correttamente il server per Linux distribuzione Debian 9.

4.5.1 Download e installazione del web server

È necessario scaricare il metapacchetto contenente le funzionalità del web server. Nel caso in cui lo si fosse già fatto saltare questo passaggio.

1. Aprire il terminale ed eseguire l'accesso tramite il comando *"su"* come utente root.
2. Digitare *"tasksel"*.
3. Si presenterà una lista di metapacchetti. Spuntare quello con la dicitura *"server web"* e premere il pulsante *"Ok"*.
4. Per verificare la corretta installazione digitare su un qualsiasi browser della propria macchina *"localhost/"* o in alternativa l'indirizzo IP privato del server, in entrambi i casi deve comparire a schermo una pagina con titolo *"Apache2 Debian Default Page"*.

Per permettere l'accesso al server a un client esterno alla rete vedi sezione 4.4.3.

4.5.2 Download e installazione di PHP7.0

È necessario impostare il server in modo tale che possa gestire i file PHP.

1. Aprire il terminale ed eseguire l'accesso tramite il comando *"su"* come utente root.
2. Digitare *"apt install php7.0"*, dopodiché confermare l'installazione digitando *"S"*.
3. Riavviare il server Apache digitando *"service apache2 reload"*.

4.6 Implementazione Apache su Ubuntu

In questa sezione viene spiegato come installare e impostare correttamente il server per Linux distribuzione Ubuntu.

4.6.1 Download e installazione di Apache

1. Aprire il terminale e scaricare Apache digitando *"sudo apt -get install apache2"*.

2. Per verificare la corretta installazione digitare su un qualsiasi browser della propria macchina *"localhost/"* o in alternativa l'indirizzo IP privato del server, in entrambi i casi deve comparire a schermo una pagina con titolo *"Apache2 Ubuntu Default Page"*.

Per permettere l'accesso al server a un client esterno alla rete vedi sezione 4.4.3.

4.6.2 Download e installazione di PHP7.0

1. Aprire il terminale e scaricare PHP7.0 digitando *"sudo apt-get install apache2 php7.0 libapache2-mod-php7.0"*;
2. Verificare l'installazione digitando *"a2query -m php7.0"*;
3. Caricare PHP7.0 su Apache digitando *"sudo a2enmod php7.0"*;
4. Riavviare Apache digitando *"sudo service apache2 restart"*;

4.7 Elaborazione dei dati e calcolo del risultato

L'applicativo JavaScript che permette il calcolo del risultato è stato scaricato da: `https://github.com/rastapasta/satellites-above`. Per poter funzionare correttamente è richiesto anche un secondo programma: `https://github.com/nsat/jspredict`. L'installazione dell'applicativo JavaScript è differente per sistemi Windows 10, Debian 9 e Ubuntu.

4.7.1 Installazione applicativo per sistemi Windows 10

1. Scaricare e installare Nodejs in questo percorso *C:\Apache24\htdocs* seguendo la procedura guidata presente al seguente link: `https://nodejs.org/en/`.
2. Accedere alla cartella *C:\Apache24\htdocs\node_modules* tramite prompt dei comandi e successivamente digitare *"npm install jspredict"* per installare JsPredict.
3. Nella medesima cartella digitare *"npm install - -save satellites-above"* per installare satellites-above.

4.7.2 Installazione applicativo per sistemi Debian 9

1. Aprire il terminale ed eseguire l'accesso tramite il comando *"su"* come utente root.
2. Installare Nodejs digitando *"apt install nodejs"*.
3. Installare il comando npm digitando *"apt install npm"*.
4. Installare JsPredict digitando *"npm install jspredict"*.
5. Installare satellites-above digitando *"npm install - -save satellites-above"*.

4.7.3 Installazione applicativo per sistemi Ubuntu

1. Installare Nodejs digitando *"sudo apt install nodejs"*.
2. Installare il comando npm digitando *"sudo apt install npm"*.
3. Installare JsPredict digitando *"sudo npm install jspredict"*.
4. Installare satellites-above digitando *"sudo npm install - -save satellites-above"*.

4.8 Modifica del programma server

Da qui in poi il procedimento per la modifica del programma sarà identico per Debian 9 e per Ubuntu. Per usufruire di un servizio offerto dal server il client accede a un'apposita cartella presente all'interno del server stesso. La tipologia di servizio dipende dal formato del file selezionato dal client. Il percorso di tale cartella è diverso per Windows 10 e sistemi Linux.

Percorso cartella per Windows 10: C:\Apache24\htdocs

Percorso cartella per sistemi Linux: /var/www/html

All'interno di questa cartella è possibile creare sottocartelle per gestire in modo ottimale l'organizzazione dei file.

Per Windows 10 il programma sarà già presente nella cartella corretta, cioè sarà già raggiungibile dai client. Per sistemi Linux è necessario copiare le cartelle *jspredict* e *satellites-above* all'interno della cartella *node_modules*

e copiare quest'ultima all'interno del percorso */var/www/html*.

Il programma così scaricato non è ottimizzato per gli scopi di questo progetto. È possibile visualizzarne l'output aprendo il prompt dei comandi ed entrando nella cartella *C:\Apache24\htdocs\node_modules\satellites-above* e digitando *"node example.js"*. La procedura è simile per sistemi Linux. Si deve perciò ripulire l'output dai dati non utili. Al fine di questo progetto sono necessari solamente ID, angolo di azimut e angolo di elevazione dei satelliti.

1. Aprire il file *SatellitesAbove.coffee* presente al percorso *C:\Apache24\htdocs\node_modules\satellites-above\lib* Windows 10 e in */var/www/html/node_modules/satellites-above/lib* nei sistemi Linux con un editor di testo.

2. Eliminare le seguenti righe:
   riga 28: @log "[+] Loaded #count GPS satellites"
   riga 33: @log "[+] Downloading TLE file.."
   riga 56: @log "[+] Finding satellites above #lat, #lng"

3. Modificare la seguente riga 53:
   above: (lat, lng, altitude=0.1, minimalElevation=15) -> in
   above: (lat, lng, altitude=0.1, minimalElevation=0) ->

4. Salvare e chiudere il file senza rinominarlo.

L'eliminazione delle righe consente di ripulire l'output mentre la sostituzione della riga 53 permette di riconoscere come visibili tutti i satelliti che hanno un'elevazione superiore a zero gradi.

4.8.1 Lettura delle coordinate fornite dal client

Per permettere la lettura della posizione fornita come parametro esterno viene utilizzato il file di esempio fornito dal programma chiamato *example.js*.

1. Aprire il file *example.js* presente al percorso *C:\Apache24\htdocs\node_modules\satellites-above* in Windows 10 e */var/www/html/node_modules/satellites-above* nei sistemi Linux con un editor di testo.

2. Sostituire la riga 7:
   let sats = satellites.above(52.520645, 13.409779); in
   let sats = satellites.above(process.argv[2], process.argv[3]);

3. Salvare e chiudere il file. È possibile rinominare il file tenendo in considerazione che proseguendo in questo manuale quando verrà nominato il file *example.js* si intenderà in file rinominato a piacere. È necessario però mantenere l'estensione .js del file.

In questo modo è possibile fornire tramite prompt dei comandi o da terminale i parametri relativi alla posizione.

4.9 Creazione del file PHP

Si deve poter permettere ai client di eseguire il programma appena creato. Non è infatti possibile per un client eseguirlo direttamente. Per questo motivo si utilizza un file PHP appositamente creato che fa da tramite tra il client e il programma. L'utilizzo del file PHP ha due vantaggi:

1. La possibilità di passare i parametri della richiesta direttamente dall'URL.

2. La possibilità di invocare il prompt dei comandi o il terminale ed eseguire un comando. Il risultato potrà poi essere inoltrato al client.

La procedura da seguire per creare il file PHP è la seguente.

1. Aprire un editor di testo.

2. Scrivere all'interno il seguente codice:

```
<?php
//lettura primo parametro passato nella richiesta HTTP
$argument1 = $_GET['argument1'];

//lettura secondo parametro passato nella richiesta HTTP
$argument2 = $_GET['argument2'];

//invocazione promt dei comandi o terminale,
//esecuzione comando e invio dei risultati al client
echo shell_exec("node example.js $argument1 $argument2");
?>
```

3. Salvare il file al percorso *C:\Apache24\htdocs\node_modules\satellites-above* in Windows 10 oppure al percorso */var/www/html/node_modules/satellites-above* nei sistemi Linux con il nome *call.php*.

4.9.1 Verifica

Per testare il corretto funzionamento dell'interfaccia server:

1. Avviare il server Apache. Per Windows 10 aprire il prompt dei comandi, andare alla cartella *C:\Apache24\bin* e digitare *"httpd.exe"*. Per i sistemi Linux il server Apache dovrebbe essere già attivo, in caso contrario aprire il terminale e digitare *"service apache2 start"*.

2. Aprire il browser da un dispositivo qualsiasi e digitare nella barra di ricerca degli URL:

```
http://X.X.X.X/node_modules/satellites-above/
  call.php?argument1=52.520645\& argument2=13.409779
```

   In *X.X.X.X* inserire l'indirizzo IP pubblico del router al quale la macchina server è collegata.

3. Il risultato ottenuto deve essere una lista formata da ID, angolo di azimut e angolo di elevazione di satelliti GPS.

L'interfaccia server del progetto è conclusa e non necessita di ulteriori aggiunte o modifiche.

Capitolo 5

Installazione e analisi dell'applicazione

5.1 Premessa

Nella prima parte di questa sezione verrà esposto il procedimento per recuperare e installare su un dispositivo personale l'applicazione Android. Nella seconda parte verranno analizzate le sezioni principali del codice. Saranno perciò trascurate le parti di minor interesse come il layout, la dichiarazione delle variabili e la gestione degli elementi presenti a schermo come ad esempio bottoni, visualizzatori di testo e timer.

5.2 Recupero e installazione dell'applicazione

L'applicazione e il relativo codice di programmazione sono disponibili nella medesima cartella in cui è presente questo manuale. Inoltre è possibile recuperare il codice Android anche dalla repository GitLab dell'Università di Padova. Eventualmente è anche possibile scrivere alla mail presente nella prima pagina di questo manuale.

L'applicazione può essere installata su un dispositivo utilizzando direttamente il file APK. In questo caso è possibile che venga visualizzato un messaggio d'errore del tipo: *Bloccata da Play Protect*, cliccare quindi su *"Installa Comunque"*. Se risultasse tuttavia impossibile l'installazione dell'applicazione disattivare il Play Protect dal Play Store. In alternativa è possibile installare l'applicazione utilizzando Android Studio.

5.3 Android Studio

L'applicazione fornita può essere modificata liberamente. Per questo scopo è consigliato l'utilizzo di Android Studio poiché i seguenti procedimenti sono stati testati unicamente in questa suite.

5.3.1 Apertura e prima compilazione del progetto

1. Avviare Android Studio e aprire il progetto *GpsSpoofReveal*.
2. Attendere il caricamento del Gradle.
3. Terminata la fase di caricamento è possibile che nel terminale della Build siano presenti uno o più errori dovuti alla mancanza dei packages necessari per la compilazione dell'applicazione. Installare quindi tutti i packages mancanti utilizzando gli appositi link. Al termine del download, Android Studio ricaricherà il Gradle. Nel caso in cui il programma restituisse ancora degli errori provare a utilizzare il comando *invalidate Caches / Restart* presente nel menù a tendina *File*.
4. Terminata l'installazione dei packages mancanti l'applicazione può essere modificata oppure installata su un proprio dispositivo o su un dispositivo virtuale simulato da Android Studio.

5.4 Analisi del codice

Il codice presente in questa sezione è quello della versione 1.0 dell'applicazione quindi non rispetta fedelmente quello originale. Inoltre verranno analizzate unicamente le parti più importanti del codice. In questa sezione si vuole solo fornire una conoscenza di base delle principali strutture di programmazione implementate nell'applicazione.

5.4.1 Ricavare la posizione del dispositivo

1. Creare un oggetto che dia accesso al servizio di posizionamento del dispositivo.

```
private LocationManager mLocationManager;
```

2. Creare un oggetto che rilevi le variazioni di posizione.

```
private LocationListener listener;
```

3. Inizializzare l'oggetto LocationManager utilizzando il metodo getSystemService fornendo come parametro la tipologia di servizio che si sta richiedendo.

```
mLocationManager = (LocationManager)
  getSystemService(LOCATION_SERVICE);
```

4. Inizializzare l'oggetto LocationListener. Verrà richiesto di sovrascrivere quattro metodi.

```
listener = new LocationListener() {
    //this method is run when the device
      detects a change of position
    @Override
    public void onLocationChanged(Location location) {
    }
    @Override
    public void onStatusChanged
      (String s, int i, Bundle bundle) {
    }
    @Override
    public void onProviderEnabled(String s) {
    }
    @Override
    public void onProviderDisabled(String s) {
    }
};
```

5. Definire in base a quali parametri avviene una notifica di modifica della posizione.

```
mLocationManager.requestLocationUpdates
  (LocationManager.GPS_PROVIDER, 10000, 0, listener);
```

   Il primo parametro indica il fornitore della posizione, il secondo parametro il tempo minimo d'attesa tra un aggiornamento e l'altro, il terzo parametro la minima distanza di variazione e l'ultimo il LocationListener.

6. Definire le istruzioni da eseguire quando avviene una modifica della posizione.

```
@Override
public void onLocationChanged(Location location) {
    //the position (longitude and latitude) is updated
    Longitude = location.getLongitude();
    Latitude = location.getLatitude();
    tvGpsCooText = "Coordinate:\nLongitude: "
      + Longitude + "\nLatitude: " + Latitude;
    tvGpsCoo.setText(tvGpsCooText);
}
```

5.4.2 Rilevazione dei satelliti visibili

1. Creare un oggetto che dia accesso alle informazioni riguardanti al segnale GNSS ricevuto dal dispositivo.

```
private GnssStatus.Callback mGnssStatusCallback;
```

2. Sfruttare gli oggetti creati e istanziati in sezione 5.4.1.

3. Definire le operazioni da eseguire quando avviene un aggiornamento di stato del segnale GNSS sovrascrivendo il metodo onSatelliteStatusChange.

```
mGnssStatusCallback = new GnssStatus.Callback() {
    @Override
    public void onSatelliteStatusChanged(GnssStatus status) {
        satelliteCount = status.getSatelliteCount();
        v = " Scan: "+ (count)+ " ,Sat-count=" +(satelliteCount)+ "\n";
        for(int i = 0; i<satelliteCount; i++){
            sat_id = status.getSvid(i);
            constellationType = status.getConstellationType(i);

            //add only constellation type GPS
            if(constellationType ==
              GnssStatus.CONSTELLATION_GPS){
                v += "Sat n."+ i +", type= "+ constellationType +",
                  id= "+ sat_id + "\n";
                checkList(sat_id);
            }
        }
        v += "\n";
        v += tvGpsInfo.getText();
        tvGpsInfo.setText(v);
        count= count + 1;
    }
};
```

5.4.3 Gestione della comunicazione con il server

1. Import delle librerie utili per comunicazione HTTP con il server.

```
import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
```

2. Definizione di una stringa che costituirà la richiesta HTTP inviata al server.

```
//the content of this string depends on how the own web server is set
String url = "http://X.X.X.X/node_modules/satellites-above/call.php?
  argument1=" + Latitude + "&argument2=" + Longitude;
```

In *X.X.X.X* inserire l'indirizzo IP del proprio server.

3. Creare e istanziare gli oggetti che permettono la comunicazione con il server.

```
OkHttpClient client = new OkHttpClient();
Request request = new Request.Builder()
        .url(url)
        .build();
```

4. Inviare la richiesta al server e definire le istruzioni da eseguire in caso di risposta.

```
//sends http request
client.newCall(request).enqueue(new Callback() {
    @Override
    public void onFailure(Call call, IOException e) {
        e.printStackTrace();
    }

  //run when device receives a http response
    @Override
    public void onResponse(Call call, Response response)
      throws IOException {
        if(response.isSuccessful()){
            final String myResponse = response.body().string();
            Analyze.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                }
            });
        }
    }
});
```

Il contenuto della risposta del server viene salvato nella stringa myResponse. Il codice da eseguire all'interno del metodo run costituisce la parte di confronto dei risultati che è stata omessa per semplicità.

5.5 Conclusioni

Per eventuali dubbi riguardo all'implementazione del server o sulla programmazione dell'applicazione scrivere all'indirizzo email presente nella prima pagina del manuale.