Progetto Pianificazione e Valutazione della Sicurezza Informatica con Elementi di Informatica Forense: ScreenStealer

Jacopo Castellini

A. A. 2015/16

Scopo di questo progetto era quello di creare un'applicazione per smartphone Android che restasse nascosta sul telefono e che, ad una specifica richiesta da parte di un server, inviasse uno screenshot del dispositivo.

Per quanto riguarda il lato server abbiamo scelto di sviluppare il programma in linguaggio Java, per mantenere la massima compatibilità possibile con gli stream e le strutture dati usate dal socket sul dispositivo Android. Il server è composto da una sola classe, ScreenStealerServer.java, che crea un oggetto di tipo ServerSocket e si mette in ascolto di eventuali connessioni entranti sulla porta 7654 grazie al metodo accept() contenuto in un ciclo while(true). Se una connessione arriva (un telefono sta cercando di connettersi al server), viene creato un oggetto di tipo Socket che identifica la connessione e viene quindi avviato un nuovo Thread per gestirla. Questo thread, dopo la creazione degli opportuni stream di input e output con il Socket (incapsulati in un DataInputStream per riceve i dati e un PrintWriter per scriverli), riceve dal telefono il suo seriale con la chiamata DataInputStream.readUTF(), crea una cartella nuova per salvarvi gli eventuali screenshot ricevuti con il metodo File.mkdirs(path) e poi entra in un ciclo in cui aspetta l'input dell'utente per decidere cosa fare. L'utente può scegliere di chiudere la connessione con il telefono (inserendo 0, che corrisponde alla stringa "STOP") o richiedere uno screenshot (inserendo 1, che invece corrisponde alla stringa "GO"). Nel secondo caso il server invia la stringa al client sul telefono tramite il metodo PrintWriter.write("GO"), che catturerà la schermata e la manderà al server tramite un array di byte. Ricevuta la lunghezza della sequenza (metodo DataInputStream.readInt()) e poi la sequenza stessa (metodo DataInputStream.readFully()), il server la salverà tramite un apposito stream (precisamente un FileOutputStream) su di una nuova immagine PNG nella cartella precedentemente creata, quindi si rimetterà in attesa dell'input dell'utente (tutta la procedura sopra descritta è inserita in un while(true) che esegue anche dei controlli sulla persistenza della connessione).

Per quanto riguarda il client invece, l'activity principale (MainActivity.java) non fa altro che eseguire le restanti componenti, quindi richiedere i privilegi di root all'utente (questi ultimi necessari per poter fare uno screenshot) prima di cancellarsi dal menù del launcher e chiudersi. La classe ConnectionStatusBR.java implementa un BroadcastReceiver che controlla tramite il metodo onReceive() quando avviene un cambiamento nello stato della connettività del telefono (vanno abilitati i permessi ACCESS_NETWORK_STATE nel file AndroidManifest.xml). Se la connessione è disponibile viene avviato il Service ScreenStealerService.java, se la connessione viene chiusa questo viene fermato. Il Service non fa altro che avviare o chiudere un Thread per la comunicazione col server (rispettivamente nei metodi onStartCommand() e onDestroy()). La classe ClientThread.java implementa la comunicazione col server vera e propria. Questo si connette con un Socket all'indirizzo del server sulla porta 7654 e crea gli opportuni stream per la comunicazione (incapsulati in un DataOutputStream per l'invio dei dati e in un BufferedReader per la lettura), quindi gli invia il seriale del telefono (contenuto nella variabile android.os.Build.SERIAL) col metodo DataOutputStream.writeUTF(serial), poi esegue un while(true) in cui aspetta un comando dal server. Se il comando è "STOP" il client chiude le connessioni e termina il thread, mentre se invece il client riceve "GO" viene chiamata la funzione takeScreenshot(). Questa esegue tramite il terminale il programma di sistema screencap, che effettua uno screenshot e lo salva in un immagine

PNG, poi carica l’immagine in un array di byte e lo ritorna al client. Questo infine non fa altro che inviare la lunghezza dell’array (metodo DataOutputStream.writeInt(length)) e poi l’array stesso (metodo DataOutputStream.write(byte_array)) e cancellare l’immagine PNG dalla memoria del telefono col metodo File.delete(), prima di rimettersi in ascolto di nuove richieste dal server.