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 due classi: la classe ScreenStealerServer.java che, dopo aver caricato l'opportuno driver ed aver stabilito la connessione con un database PostgreSQL usato per salvare informazioni di sicurezza usabili per un analisi forense sugli screenshot ricevuti, crea un oggetto di tipo ServerSocket e si mette in ascolto di eventuali connessioni entranti sulla porta 7654. Se una connessione arriva, viene creato un oggetto di tipo Socket che identifica la connessione e viene quindi avviato un nuovo Thread per gestirla, implementato nella classe ClientThread.java. Questo thread, dopo la creazione degli opportuni stream di input e output con il socket, riceve dal telefono il suo seriale e crea una cartella nuova per salvarvi gli eventuali screenshot ricevuti da esso, 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, che catturerà la schermata e la manderà al server tramite un array di byte. Ricevuti il timestamp di acquisizione, la lunghezza della sequenza e poi la sequenza stessa ed infine l'hash SHA-1 del file, il server salverà la sequenza tramite un apposito stream su di una nuova immagine PNG nella cartella precedentemente creata, quindi farà una INSERT nel database di una tripla del tipo (seriale, timestamp, hash del file) e si rimetterà in attesa dell'input dell'utente (tutta la procedura sopra descritta è inserita in un loop infinito 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 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, quindi gli invia il seriale del telefono (contenuto nella variabile android.os.Build.SERIAL), poi entra in un loop infinito 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 prima salvato ed inviato l'attuale timestamp (opportunamente formattato come stringa) e poi 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 e poi l’array stesso, oltre che all’hash SHA-1 calcolato sull’array di byte che rappresenta l’immagine. Infine viene cancellata l’immagine PNG dalla memoria del telefono per non lasciare tracce di utilizzo, prima di rimettersi in ascolto di nuove richieste dal server.