Ciberseguridad en Dispositivos Móviles

Práctica 1 – Emulación y pruebas con Android

Contenido

Luis López Cuerva

Pablo Alcarria Lozano

Ejercicio 1

Como puede apreciarse en las capturas de código, los métodos *onCreate()*, *onStart()*, *onResume()*, *onPause()*, *onStop()* y *onDestroy()* deben reescribirse en la clase de actividad principal, llamada MainActivity.java. En el caso del *onCreate()*, es donde escribiremos los *listeners* o la redirección a nuevas actividades cuando sea deseado. En el caso, de las otras, añadiendo *logs* podemos comprobar en qué momento se ejecutan cuando estemos utilizando el emulador. Respecto a las situaciones, hemos encontrado 2 que se distinguen firmemente:

1. Cuando salimos de la aplicación volviendo al menú principal o moviéndonos a otra aplicación, vemos como la aplicación pasa por el método *onPause()* y *onStop()*, y una vez volvemos a ella, pasa por *onStart()* y *onResume()*.

```
D/holamundo.MainActivity: onPause
    onStop
D/EGL_emulation: eglMakeCurrent: 0xaf034c40: ver 2 0 (tinfo 0xaf039410)
W/IInputConnectionWrapper: showStatusIcon on inactive InputConnection
D/holamundo.MainActivity: onStart
    onResume
```

2. Si una vez dentro de la aplicación le damos al botón de salir, aparte de pasar por *onPause*() y *onStop*(), también se hace la llamada a *onDestroy*(), dejando espacio en la pila para otras aplicaciones. Tanto la primera vez que entramos una vez se inicie el terminal como en este caso, que hemos salido del todo, cuando se abre la aplicación pasa por *onCreate*(), *onStart*() y *onResume*().

```
D/holamundo.MainActivity: onPause
D/holamundo.MainActivity: onStop
    onDestroy
D/EGL_emulation: eglMakeCurrent: 0xaf034c40: ver 2 0 (tinfo 0xaf039410)
W/IInputConnectionWrapper: showStatusIcon on inactive InputConnection
D/holamundo.MainActivity: Mensaje cambiado en onCreate
D/holamundo.MainActivity: onStart
D/holamundo.MainActivity: onResume
```

Ejercicio 2

Para realizar esta actividad, hemos creado un *ImageView* en activity_main.xml que contiene una imagen de un icono de información. La idea es que cuando se haga click en ella, se abra una nueva actividad enseñando los datos de los integrantes del grupo y sus emails. Para ello, dentro del método *onCreate()* de MainActivity.java hemos creado un *listener* asociado a la imagen, de forma que cuando se haga *click* en ella, en base al contexto de la aplicación se cree una nueva actividad, llamada créditos. Dentro de ***activity_creditos.xml*** nos encontramos con dos *TextView*, que de forma similar a MainActivity.java, dentro del *onCreate*() se añade el texto correspondiente. Para salir de esta actividad, con el botón de retroceder volvemos a la actividad principal.

```
public class creditos extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_creditos);
        TextView s = findViewById(R.id.creditosNombres);
        TextView c = findViewById(R.id.creditosCorreos);
        s.setText("Luis López Cuerva y Pablo Alcarria Lozano");
        c.setText("luilocu2@inf.upv.es y pabloal1@inf.upv.es");
        Log.d( tag: "holamundo.creditos",  msg: "créditos mostrados");
    }
}
```

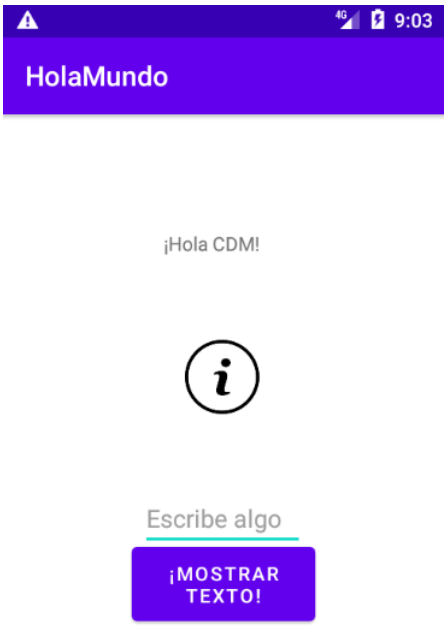
Ejercicio 3

Por último, hemos añadido un *TextView* en el que los usuarios pueden escribir, debajo de los elementos ya mencionados. Este *TextView* está configurado para que no muestre lo que escribas y lo genere en un *Toast* cuando se haga click en el botón “¡Mostrar texto!”. Para esta funcionalidad es necesario también añadir un *onClickListener()* al botón.

```
((Button)findViewById(R.id.button)).setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //generar toast que muestre lo escrito en el textedit
                //if (nada escrito) print("No has escrito nada!")
                EditText text = (EditText) findViewById(R.id.et_Contrasenya);
                String value = text.getText().toString();
                Log.d( tag: "holamundo.MainActivity", value);
                if (value.equals("")) { Toast.makeText(getApplicationContext(), text: "No has escrito nada!", Toast.LENGTH_SHORT).show(); }
                else { Toast.makeText(getApplicationContext(),value, Toast.LENGTH_SHORT).show(); }
            }
        }
);
```

Como puede apreciarse en el código, el *listener* se hace sobre el botón en sí y haciendo casting a la clase *Button*, asegurándonos de que estamos apretando el botón correctamente. Una vez creado el *listener*, se recupera el texto con la clase *getText*() sobre un *EditText*, recuperado desde su id. Para la generación de *Toasts* podemos conectar que al igual que en la creación de una nueva actividad, se utiliza el método *getApplicationContext*() para recuperar el contexto de la aplicación.

Aquí podemos apreciar la aplicación nada más abrirse con los 3 ejercicios implementados:



Ejercicio 4

Se ha estudiado el malware proporcionado y hemos llegado a la conclusión de que el código maligno se activa solo pero es necesario dar permiso a los contactos para que sea efectivo, no es necesario que el usuario comparta su lista de la compra o ni tan solo abra la aplicación. Esto se debe a que la rutina que activa el malware para que este programe un ciclo de periodo inexacto que borre los contactos se activa al reiniciar el dispositivo móvil una vez que se ha instalado el software. Esto se puede observar a nivel de código en el fichero *AndroidManifest.xml* donde se puede ver que el programa solicita accesos a la información de arranque del sistema y prepara un receiver que se activa cuando se enciende el dispositivo. Este receiver al activarse hace que se ejecute el código malicioso que permite a la aplicación borrar los contactos.

Este problema tiene una solución muy sencilla, no conceder acceso a los contactos a la aplicación. Esta acción no impide que se ejecute la rutina maligna que pretende borrar los contactos pero hace que esta no tenga éxito, es decir, la rutina se ejecuta pero es incapaz de borrar ningún contacto, de manera que resuelve el problema.

Ejercicio 5

El diseño de este malware es sencillo y es extraordinariamente fácil de replicar en otras aplicaciones. Para reproducir este comportamiento hemos seguido los siguientes pasos:

1. Copiar a la nueva aplicación la carpeta *Maldad* y actualizar los nombres de los paquetes.
2. Actualizar el fichero *AndroidManifest.xml* con las actividades, permisos y receivers necesarios. Este paso es muy sencillo ya que simplemente hay que observar las diferencias entre la aplicación maligna de la que se coge el código y la aplicación a modificar.
3. Hacer que la nueva aplicación solicite acceso a los contactos.

En nuestra implementación del ejercicio 5 hemos optado por solicitar los permisos nada más se inicia la aplicación, este hecho puede causar sospechas en el usuario, de manera que se podría usar ingeniería social para buscar una excusa mejor para solicitar el acceso a los contactos como podría ser añadir la opción de compartir alguna característica de la aplicación, como noticias dentro de la aplicación o algunas conversiones realizadas. No obstante hemos decido no seguir esta última vía ya que no consideramos que sea de interés compartir ninguna característica de la aplicación y hemos pensado que esa solicitud de contactos nada más iniciar la aplicación puede ser más efectiva.