

SECURITY IN ANDROID

GreenPS

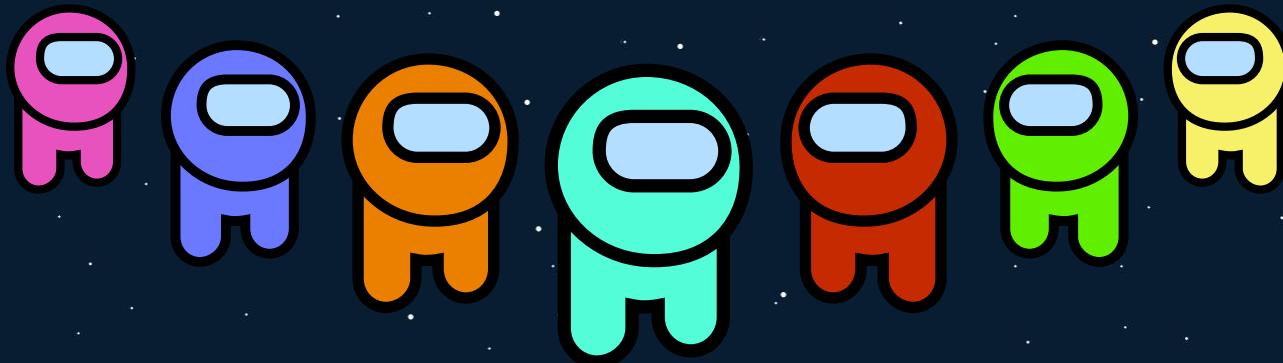
Ramírez Soto Alci Rene
Arias Muñoz Elsa Johanna
Vargas Romero Camilo Ernesto
Martinez Vanegas Marcel Julian

CONTENTS OF THIS TEMPLATE

1. Introducción y conceptos básicos
2. Tipos de ataques
3. Vulnerabilidades comunes
 - a. Tapjacking.
 - b. Incorrect activity configuration
 - c. Communication over clear text protocol
 - d. Weak lockout mechanism
 - e. Insecure data storage in external storage
 - f. Plaintext storage of credentials
 - g. Misuse of fingerprinter
 - h. Storing credentials with "REMEMBER ME" function
 - i. Insecure generation of encryption keys
4. Recopilatorio de Buenas prácticas
5. Bibliografia
6. Gracias

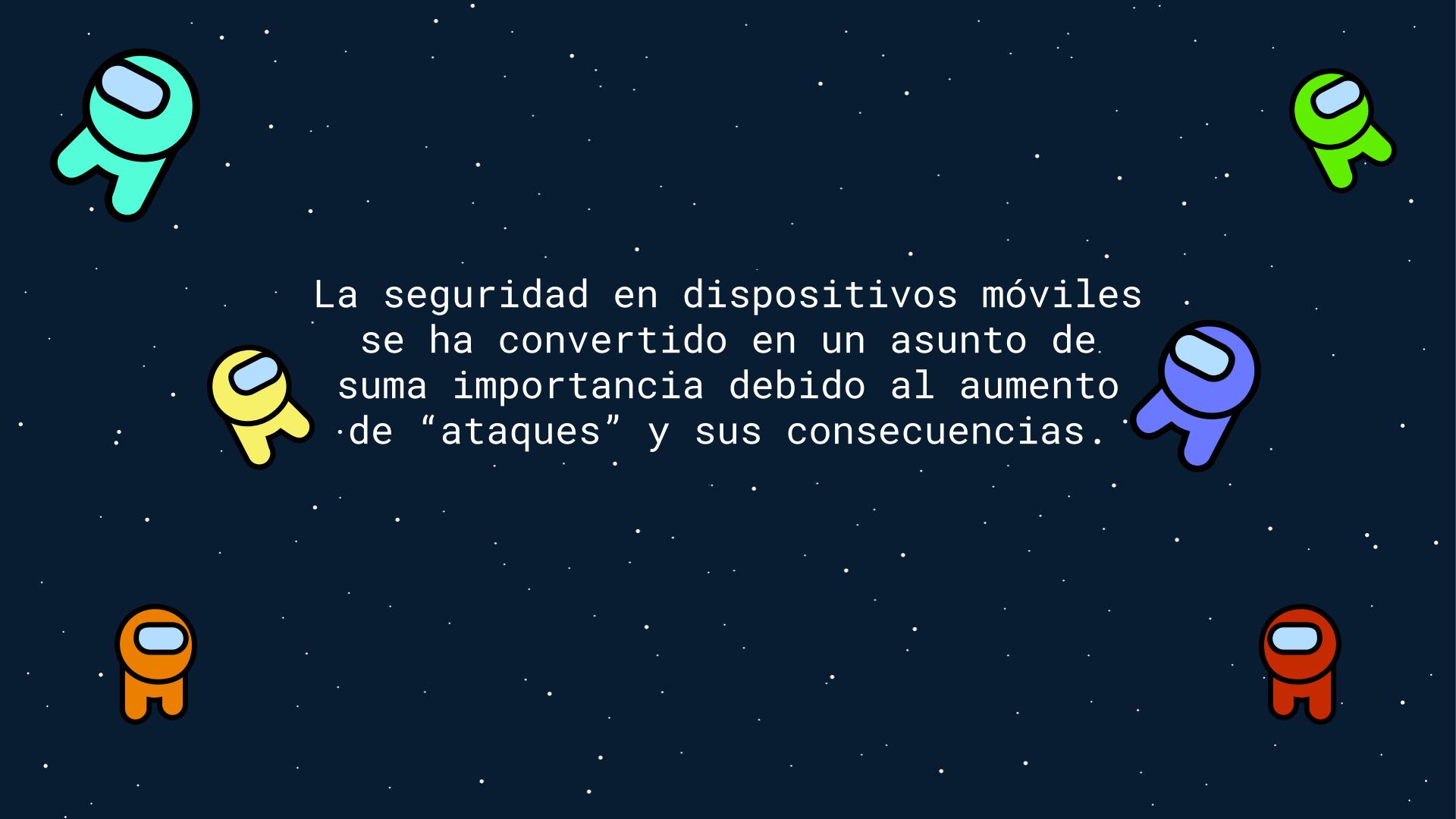
IMPOSTORS AMONG US

There is 1 Impostor among us 2k



INTRODUCCIÓN Y CONCEPTOS BÁSICOS





La seguridad en dispositivos móviles se ha convertido en un asunto de suma importancia debido al aumento de “ataques” y sus consecuencias.

CONCEPTOS BÁSICOS

Confidencialidad

propiedad que asegura que solo los que están autorizados tendrán acceso a la información.

Integridad

propiedad que asegura la no alteración de la información. Por alteración entendemos cualquier acción de inserción, borrado o sustitución de la información.

Autenticación

propiedad que hace referencia a la identificación. Es el nexo de unión entre la información y su emisor.

No repudio

propiedad que asegura que ninguna parte pueda negar ningún compromiso o acción realizados anteriormente.

REVOLUCIÓN TECNOLÓGICA

1. Hardware potente con mucho sensores.
2. Sistema operativo que facilita el uso de SDK . sencillos y potentes para los desarrolladores.
3. Un mercado de aplicaciones completamente integrado en el sistema y muy intuitivo, lo que facilita las transacciones tanto a los usuarios como a los desarrolladores.



CAPAS DE SEGURIDAD

01

El uso del espectro electromagnético como medio de comunicación implica que la información viaja por el aire sin que nada ni nadie le pueda poner límites.

02

Los sistemas operativos para móviles actuales son complejos, que contiene características que hasta hace unos años eran impensables, cómo ejecutar aplicaciones 3D, navegar en la Red o multitarea. Estas nuevas funcionalidades abren más vías de ataque y aumentan el interés de los atacantes.



CAPAS DE SEGURIDAD

03

En este nivel es muy importante revisar las medidas de seguridad que se han tomado para que las aplicaciones no puedan desestabilizar el sistema.

04

Se centra en intrusiones físicas, este tipo de intrusión es el más peligroso, ya que es muy vulnerable. Aun así, hay medidas que se pueden tomar con el fin de que los datos que almacenamos en nuestro dispositivo móvil permanezcan seguros.



TIPOS DE ATAQUES

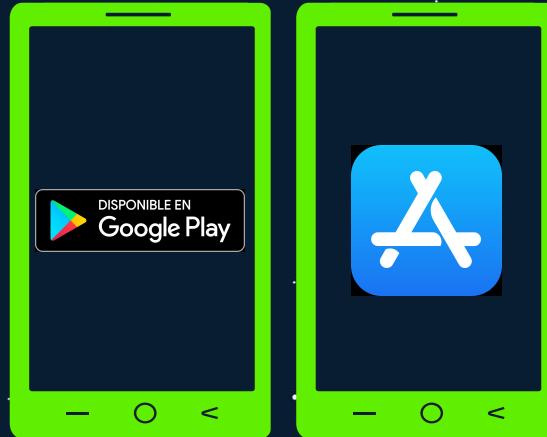


TIPOS DE ATAQUES

Posibles formas o medios que permiten a ciberatacantes violar la seguridad de un dispositivo móvil.

Vectores de ataque :

- Aplicaciones (mercados de aplicaciones)
- Redes
- Dispositivo
- Vulnerabilidades del sistema operativo

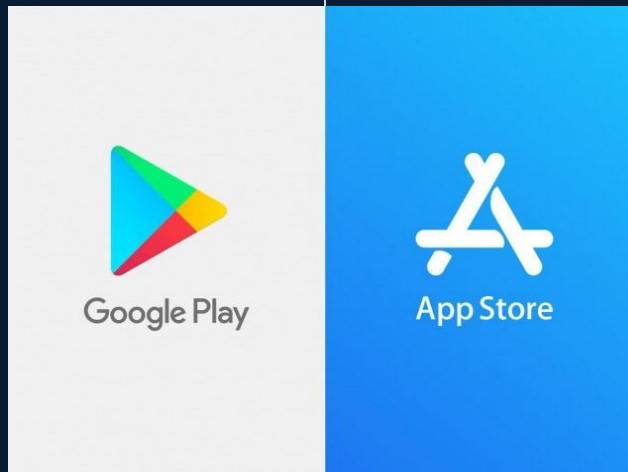


MERCADOS DE APLICACIONES

Plataforma abierta propiedad de Google que permite publicar aplicaciones de todo tipo.

Acciones :

- Antes de 2012 no analizan las aplicaciones.
- Desde 2012 mediante un software automático se analizan las aplicaciones ya publicadas.



Es una plataforma cerrada propiedad de Apple Inc que filtra las aplicaciones que allí están disponibles.

Acciones :

- Revisión minuciosa de las aplicaciones candidatas a estar disponibles.
- Revisión de seguridad.
- Revisión de atraktividad y funcionalidad.

A :.



WhatsApp Messenger

WhatsApp Inc. Communication

★★★★★ 60,186,133

3+

This app is compatible with all of your devices.

Installed

B :.



WhatsApp Messenger

WhatsApp Inc. Weather

★★★★★ 271

3+

Loading device compatibility...

Add to Wishlist

Install

C :.



Whatsapp Messenger

WhatsApp .Inc ** Communication

★★★★★ 484

Contains ads

Add to Wishlist

Install

Mercados de Aplicaciones



To



0

 **Poll locked.** Responses not accepted.

Mercados de Aplicaciones



Powered by  **Poll Everywhere**

Start the presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app

 **Poll locked.** Responses not accepted.

Mercados de Aplicaciones



Powered by  **Poll Everywhere**

Start the presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app



WhatsApp Messenger

WhatsApp Inc. Communication

3+

This app is compatible with all of your devices.



★★★★★ 60,186,133

Installed

B :



WhatsApp Messenger

WhatsApp Inc. Weather

3+

>Loading device compatibility...

★★★★★ 271

Add to Wishlist

Install

C :



WhatsApp Messenger

WhatsApp .Inc ** Communication

16+

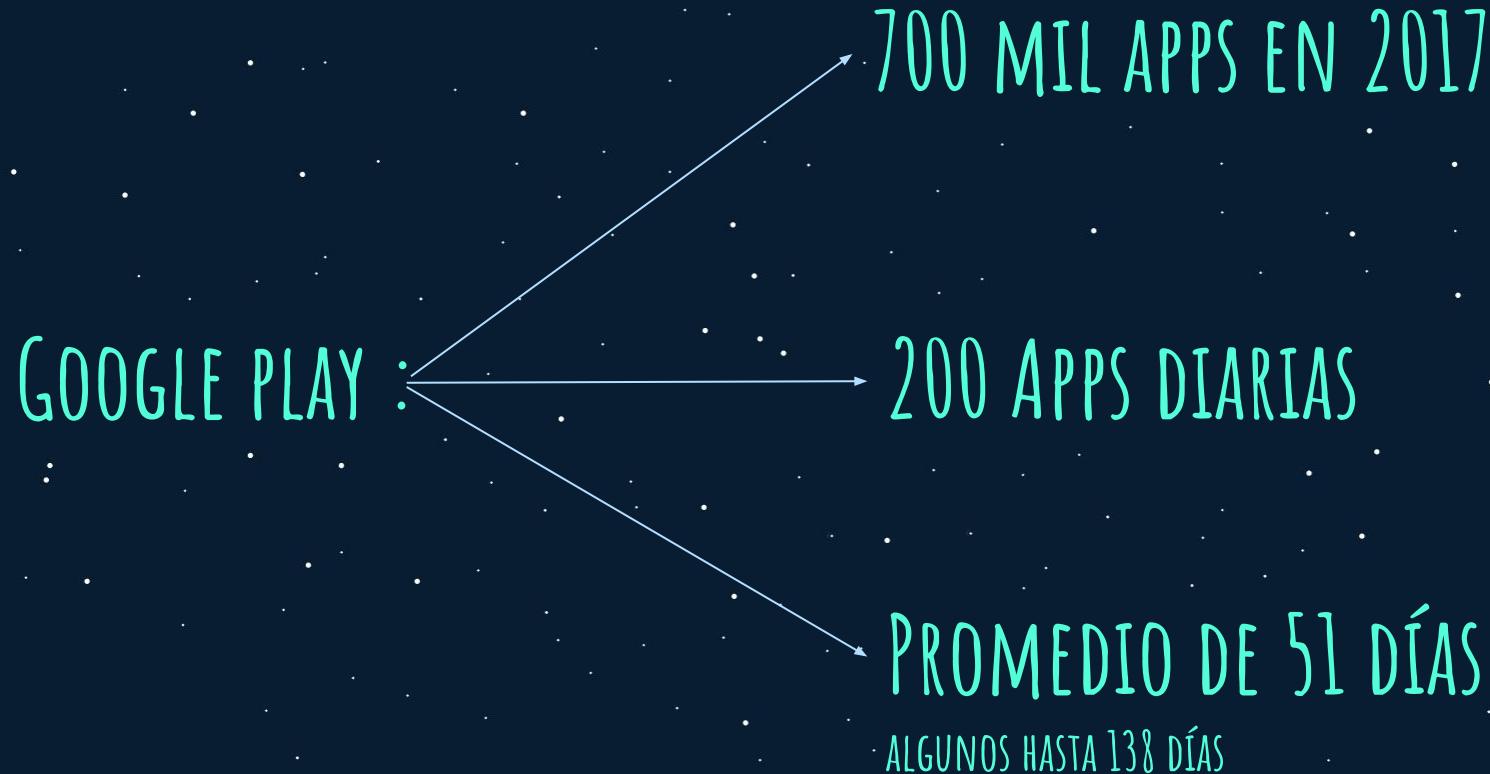
Contains ads

This app is compatible with all of your devices.

★★★★★ 484

Add to Wishlist

Install

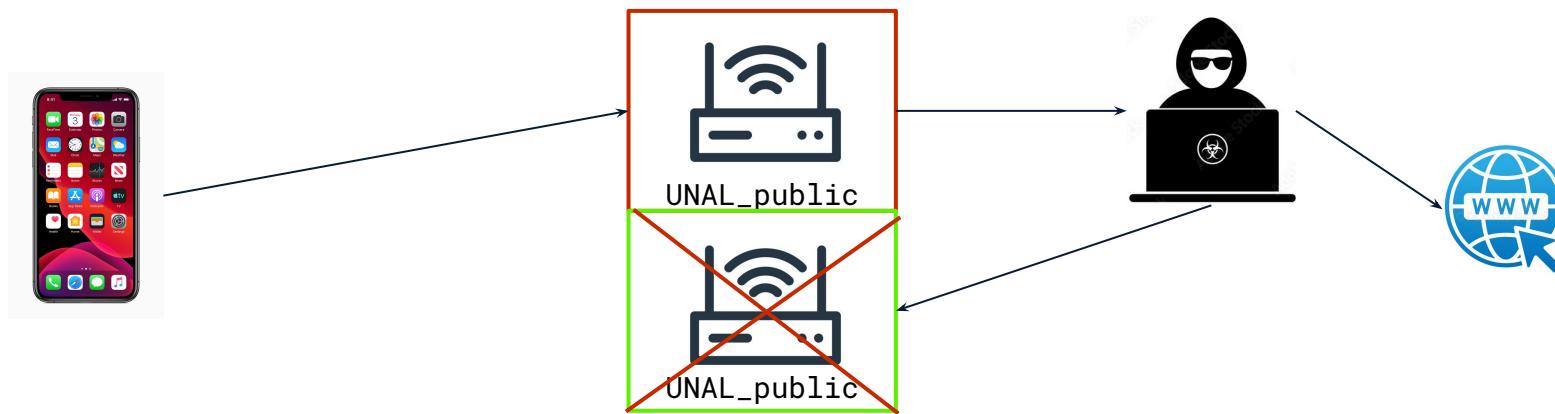
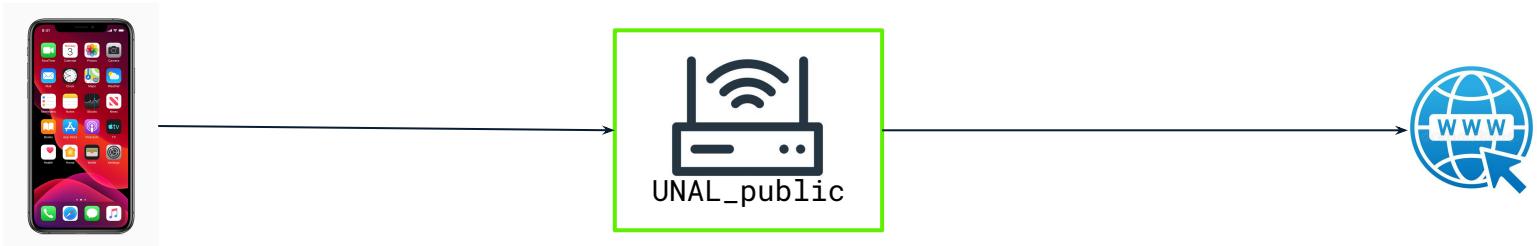


REDES

- Mensajes MMS maliciosos.
- Antenas y hotspots fraudulentos.
- Interceptación de mensajes.
- Espionaje.
- Redes públicas inseguras.



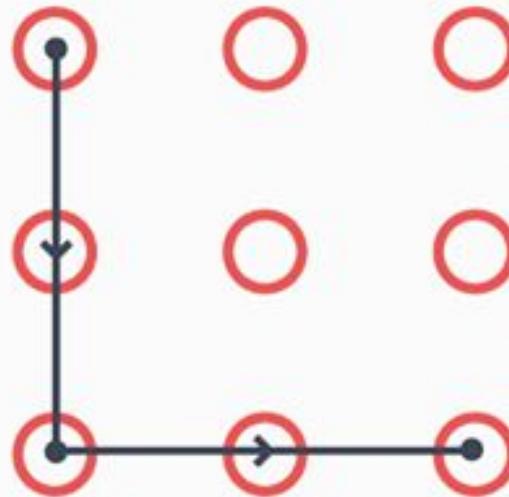
HOTSPOTS FRAUDULENTOS



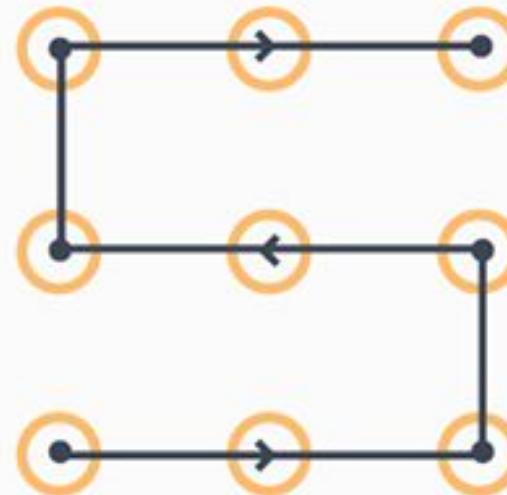
DISPOSITIVOS

- Correcto uso de contraseñas de acceso
- Gestión de información que contiene
- Robo o hurto
- Antivirus en celular

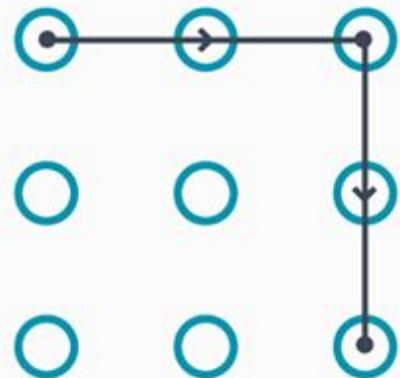




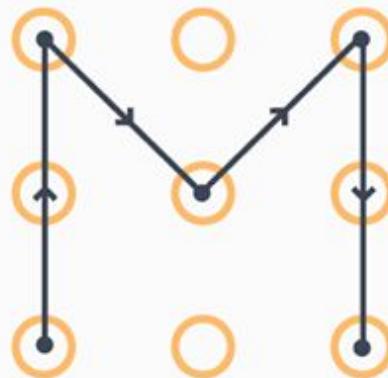
44% inicia arriba
a la izquierda.



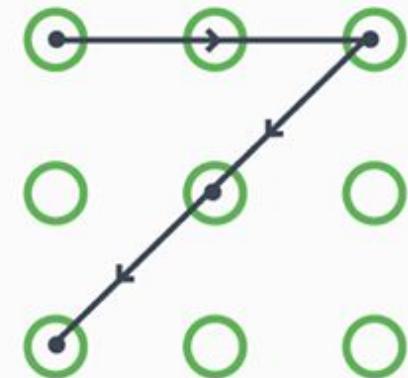
77% incia
de alguna esquina.



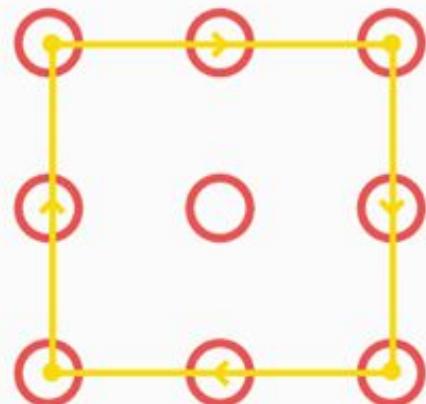
Más del 50% utiliza
una secuencia de 5 puntos.



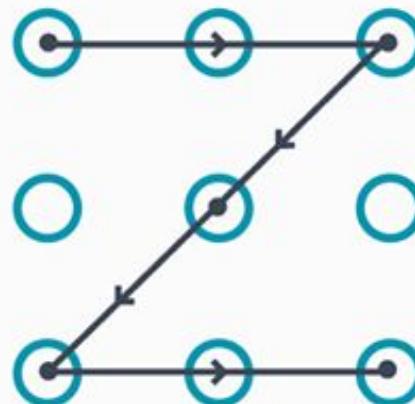
La mayoría
utiliza forma de letras...



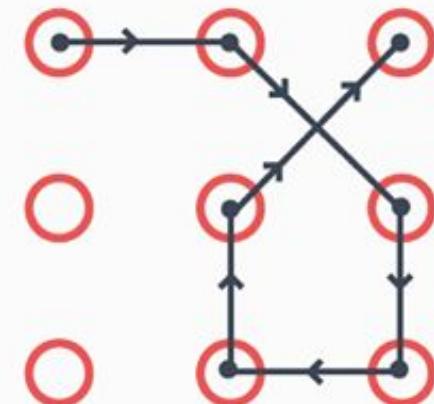
...o números.



La "O" es el unlock
más común...



...seguido por la "Z".



Usar 8 puntos y ser random
es lo más seguro :)

VULNERABILIDADES COMUNES



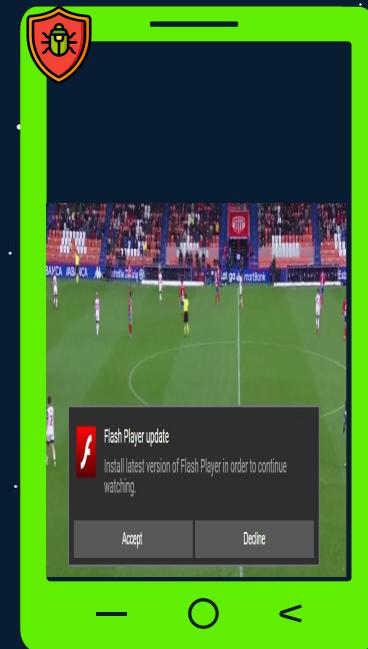
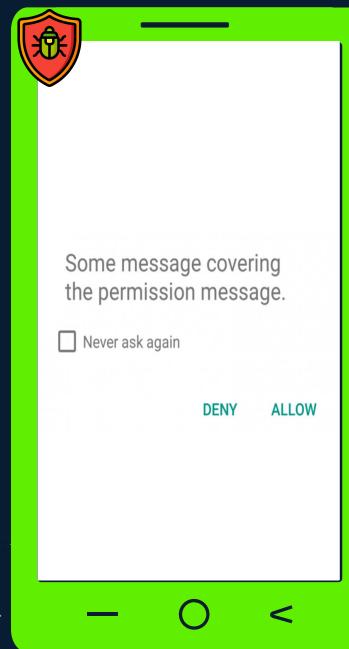
TAPJACKING

¿Qué es?

Es una vulnerabilidad de seguridad que se encuentra en los usos inadecuados de la plataforma.

Ejemplo:

Permisos sin saber el nombre de la aplicación que los pide, descarga de programa por defecto o ya instalados



COMO EVITARLO

Programadores:

Usar las lineas ⇒

Usuarios:

Identificar los permisos de aplicaciones, también las funcionalidades default de una app.

```
<Button  
    android:id="@+id/btnTrusted"  
    android:layout width="wrap content"  
    android:layout height="wrap content"  
    android:layout weight="1"  
    android:layout marginTop="3dp"  
    android:layout marginBottom="3dp"  
    android:layout marginLeft="3dp"  
    android:layout marginRight="3dp"  
    android:backgroundTint="#67AB9F"  
    android:text="Trust accept" />
```

android:filterTouchesWhenObscured = "true"
android:filterTouchesWhenObscured = "false"



VULNERABILIDAD :

A :
62 android:src="@drawable/ic_backspace_black_24dp"
63 android:filterTouchesWhenObscured="false"/> >
64

B :
62 android:src="@drawable/ic_backspace_black_24dp"
63 />
64

C :
62 android:src="@drawable/ic_backspace_black_24dp"
63 android:focusableInTouchMode="true"/> >
64

D :
61 android:gravity="center"
62 android:src="@drawable/ic_backspace_black_24dp"
63 android:filterTouchesWhenObscured="true"/> >
64

TAPJACKING

Three large, empty rectangular input fields stacked vertically, likely for tapjacking practice.

To



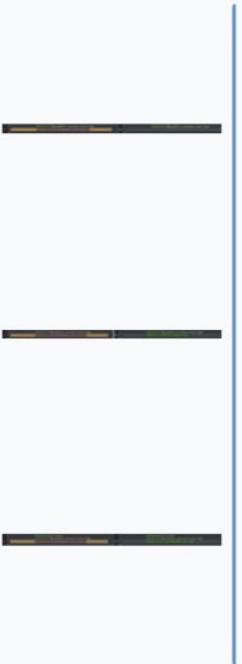
0

TAPJACKING



 **Poll locked.** Responses not accepted.

TAPJACKING



Powered by  **Poll Everywhere**

Start the presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app

VULNERABILIDAD :

62 android:src="@drawable/ic_backspace_black_24dp"
⚠ 63 android:filterTouchesWhenObscured="false"/> >
64

A .

62 android:src="@drawable/ic_backspace_black_24dp"
63 />
64

Default: android:filterTouchesWhenObscured="false"/> >

B .

62 android:src="@drawable/ic_backspace_black_24dp"
63 android:FocusableInTouchMode="true"/> >
64



61 android:gravity="center"
62 android:src="@drawable/ic_backspace_black_24dp"
63 android:filterTouchesWhenObscured="true"/> >
64

INCORRECT ACTIVITY CONFIGURATION

¿Qué es?

- Mala configuración de las actividades o diferentes módulos en el android manifest.

Ejemplo:

Malware, spyware, entre otras.



COMO EVITARLO

Programador:

- Usar las lineas ⇒ Accesibilidad menor.
- Agregar permisos para que solo la app reciba información.

Usuario:

Antivirus, programas de limpieza.

```
<activity  
    android:name=".FiltroClass"  
    android:exported="false" />  
<activity  
    android:name=".Filtros"  
    android:exported="false" />  
<activity  
    android:name=".MainActivity"  
    android:exported="true">  
        <intent-filter>  
            <action android:name="android.intent.action.MAIN"  
/>        />  
        <category  
            android:name="android.intent.category.LAUNCHER" />  
        </intent-filter>  
</activity>
```

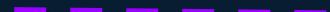
COMMUNICATION OVER CLEAR TEXT PROTOCOL

¿Qué es?

Es una vulnerabilidad que aparece cuando se envían datos a través de internet por canales no encriptados

Ejemplo:

- Ingresar a páginas que no tienen protocolo https



COMO EVITARLO

Programador:

Hacer uso de certificados propios o ofrecidos de confianza.

Usuario:

Ingresar solo a sitios de confianza que tengan el https, si se hará uso de datos sensibles.

The image shows a web browser window with two tabs open. The left tab is a BBVA website, and the right tab is a local host page for 'Verificación del personal'. A certificate dialog box is overlaid on the BBVA site. The dialog box contains the following text:

Información del certificado

Este certif. está destinado a los siguientes propósitos:

- Prueba su identidad ante un equipo remoto
- Asegura la identidad de un equipo remoto
- 2.23.140.1.2.2

* Para ver detalles, consulte la declaración de la entidad de c...

Emitido para: *.bbva.com.co

Emitido por: DigiCert SHA2 Secure Server CA

Válido desde: 27/08/2021 hasta: 30/08/2022

Aceptar

To the right of the dialog box, the local host tab shows a security warning from Google Chrome:

Your connection to this site is not secure
You should not enter any sensitive information on this site (for example, passwords or credit cards), because it could be stolen by attackers.
Learn more

You have chosen to turn off security warnings for this site. Turn on warnings

Cookies: 2 in use

Site settings

The local host page itself displays the following information:

Validación del personal de Infometrika
La siguiente persona está autorizada por Infometrika para la realización de encuestas

Datos de Personal

Nombre Completo: EILEEN DEL CARMEN HERNANDEZ MARTINEZ
Documento de Identificación: 45715962
Rol/Cargo: SEGMENTADOR RH: O+
Valido hasta: 2021-12-31

Infometrika
Consultores en información

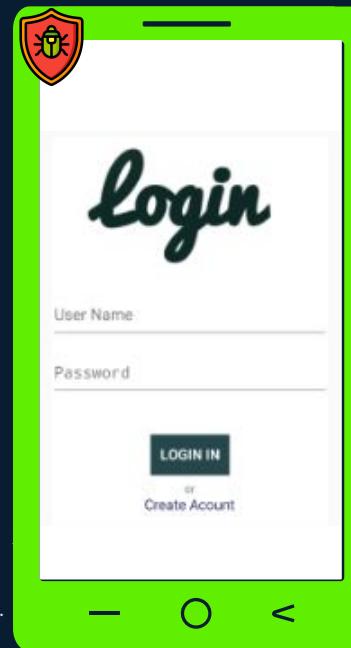
WEAK LOCKOUT MECHANISM

¿Qué es?

Es una vulnerabilidad de seguridad a la hora de autenticar al usuario.

Ejemplo:

Permitir al usuario un número ilimitado de intentos a la hora de ingresar sus credenciales de autenticación.



Fuerza bruta



COMO EVITARLO

Programador:

Implementar mecanismos de bloqueo más seguros como:

- Basados en tiempo
- Self-service



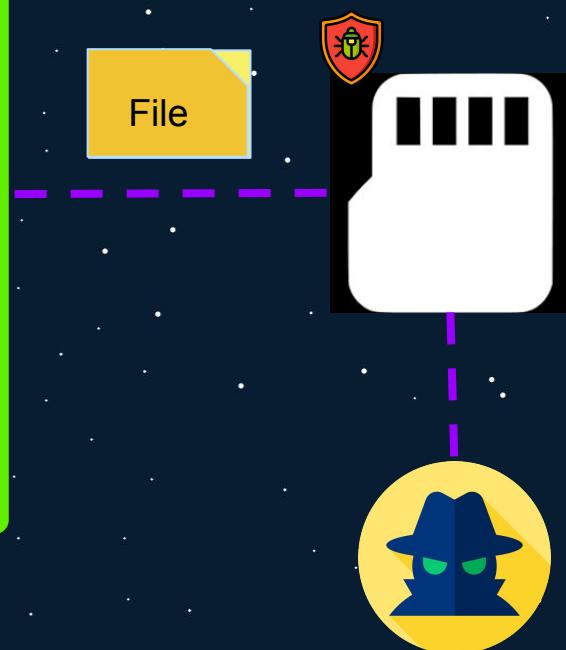
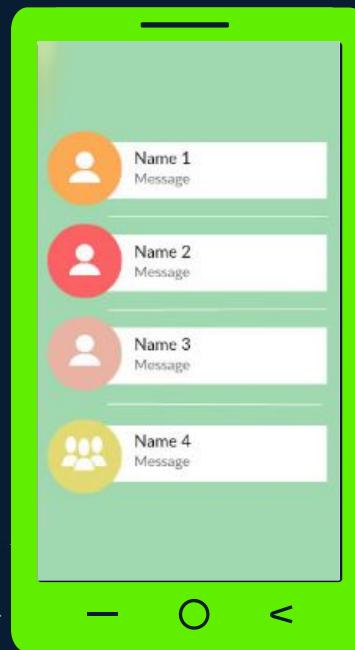
INSECURE DATA STORAGE IN EXTERNAL STORAGE

¿Qué es?

Las áreas de almacenamiento de datos siempre son propensas a vulnerabilidades, los archivos creados en un almacenamiento externo son accesibles por cualquier app en el dispositivo.

Ejemplo:

Al instalar una app que contenga algún malware, esta podría acceder al almacenamiento externo y ser capaz de manipular la información almacenada por otras apps.



COMO EVITARLO

Programador:

Siempre asumir que usuarios malintencionados y malware atacaran los almacenamientos externos, usar el almacenamiento interno del dispositivo en lo posible ya que es solamente accesible por la propia aplicación y hacer uso de librerías criptográficas para proteger la información sensible.



VULNERABILIDAD :

```
⚠ 123     try {
⚠ 124         String state = Environment.getExternalStorageState();
⚠ 125         if ( !Environment.MEDIA_MOUNTED.equals(state) ){
⚠ 126             if (BuildConfig.DEBUG) new Exception("External Storage Not
Available").printStackTrace();
⚠ 127         }
⚠ 128         File dir = new
    File(Environment.getExternalStorageDirectory().getAbsolutePath() +
    "/todosApp/");
⚠ 129         if (!dir.exists()) {
⚠ 130             dir.mkdirs();
⚠ 131         }
⚠ 132         File myFile = new File(dir, TODO_FILE_NAME);
⚠ 133         if (!myFile.exists()) {
```

¿ ES CORRECTA ESTA SOLUCIÓN ?

```
70             mPreview.mCamera.takePicture(null, null, new
71             Camera.PictureCallback() {
72                 @Override
73                 public void onPictureTaken(byte[] bytes, Camera camera)
74                 {
75                     try {
76                         File photoFileDir = new
77                             File(Environment.getExternalStorageDirectory().getAbsolutePath() + "/Cheques");
78                         if (!photoFileDir.exists()){
79                             photoFileDir.mkdirs();
80                         }
81                         photoFile = new
82                             File(photoFileDir.getAbsolutePath(), System.currentTimeMillis() + ".jpg");
83                         FileOutputStream outStream = new
84                             FileOutputStream(photoFile.getAbsolutePath());
85                         outStream.write(bytes);
86                         outStream.close();
87                         BitmapFactory.Options options = new
88                             BitmapFactory.Options();
89                         options.inJustDecodeBounds = true;
90                         BitmapFactory.decodeFile(photoFile.getAbsolutePath(), options);
91                         int imageWidth = options.outWidth;
92                         options.inJustDecodeBounds = false;
93                         options.inSampleSize = imageWidth / 500;
94                         imageCheque.setImageBitmap(BitmapFactory.decodeFile(photoFile.getAbsolutePath(),
95                             options));
96                         imageCheque.setVisibility(View.VISIBLE);
97                         hidePreview();
98                     } catch (Exception e) {
```

```
73             public void onPictureTaken(byte[] bytes, Camera camera)
74             {
75                 try {
76                     String fileName = System.currentTimeMillis() +
77                         ".jpg";
78                     FileOutputStream outStream =
79                         openFileOutput(fileName, MODE_PRIVATE);
80                     String encryptionKey = getKeyFromFile();
81                     if(TextUtils.isEmpty(encryptionKey)) {
82                         encryptionKey =
83                             AES256Cipher.generateKey(32);
84                         saveKeyToFile(encryptionKey);
85                     }
86                     outStream.write(AES256Cipher.encrypt(encryptionKey.getBytes(), bytes));
87                     outStream.close();
88                     photoFile = new File(getFilesDir(), fileName);
89                     BitmapFactory.Options options = new
90                         BitmapFactory.Options();
91                     options.inJustDecodeBounds = true;
92                     BitmapFactory.decodeFile(photoFile.getAbsolutePath(), options);
93                     int imageWidth = options.outWidth;
94                     options.inJustDecodeBounds = false;
95                     options.inSampleSize = imageWidth / 500;
96                     imageCheque.setImageBitmap(
97                         BitmapFactory.decodeByteArray(bytes, 0, bytes.length));
98                     imageCheque.setVisibility(View.VISIBLE);
99                     hidePreview();
100                 } catch (Exception e) {
101                     if (BuildConfig.DEBUG) {
```

Insecure data storage in external storage

```
    @Override
    public void onPictureTaken(byte[] bytes, Camera camera) {
        Camera.PictureCallback cb = new Camera.PictureCallback() {
            @Override
            public void onPictureTaken(byte[] bytes, Camera camera) {
                try {
                    File photoFileDir = new File(Environment.getExternalStorageDirectory(), "Dynamsoft");
                    if (!photoFileDir.exists()) {
                        photoFileDir.mkdirs();
                    }
                    photoFile = new File(photoFileDir, System.currentTimeMillis() + ".jpg");
                    FileOutputStream outStream = new FileOutputStream(photoFile);
                    outStream.write(bytes);
                    outStream.close();
                } catch (Exception e) {
                    BitmapFactory.Options options = new BitmapFactory.Options();
                    options.inJustDecodeBounds = true;
                    BitmapFactory.decodeFile(photoFile.getAbsolutePath(), options);
                    int imgWidth = options.outWidth;
                    options.inJustDecodeBounds = false;
                    options.inSampleSize = imgWidth / 500;
                    ImageChanger.setImgBitmap(BitmapFactory.decodeFile(photoFile.getAbsolutePath(), options));
                    ImageChanger.setVisibility(View.VISIBLE);
                    hidePreview();
                } catch (Exception e) {
                }
            }
        };
        camera.takePicture(cb, cb);
    }
}

public void onPictureTaken(byte[] bytes, Camera camera) {
    try {
        String fileName = System.currentTimeMillis() + ".jpg";
        FileOutputStream outStream = openFileOutput(fileName, MODE_PRIVATE);
        String encryptionKey = getKeyFromFile();
        if (TextUtil.isNotEmpty(encryptionKey)) {
            encryptionKey = AES256Cipher.generateKey();
            saveKeyToFile(encryptionKey);
        }
        outStream.write(AES256Cipher.encrypt(encryptionKey.getBytes(), bytes));
        outStream.close();
        photoFile = new File(getFileDir(), fileName);
        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inJustDecodeBounds = true;
        BitmapFactory.decodeFile(photoFile.getAbsolutePath(), options);
        int imgWidth = options.outWidth;
        options.inJustDecodeBounds = false;
        options.inSampleSize = imgWidth / 500;
        ImageChanger.setImgBitmap(BitmapFactory.decodeByteArray(bytes, 0, bytes.length));
        ImageChanger.setVisibility(View.VISIBLE);
        hidePreview();
    } catch (Exception e) {
        if (BuildConfig.DEBUG) {

```

SI

NO



Insecure data storage in external storage

```
70     mPreview.setCamera(camera);
71     camera.takePicture(null, null, new
72     {
73         @Override
74         public void onPictureTaken(byte[] bytes, Camera camera)
75         {
76             try {
77                 File photofileDir = new
78                 File(Environment.getExternalStorageDirectory(), "cheque");
79                 if (!photofileDir.exists())
80                     photofileDir.mkdirs();
81                 else
82                     photofileDir.delete();
83             }
84             catch (Exception e) {
85                 Log.e("Cheque", "Error creating directory", e);
86             }
87             File photofileDirFile = new
88                 File(photofileDir.getAbsoluteFile(), System.currentTimeMillis() + ".jpg");
89             FileOutputStream photostream = new
90                 FileOutputStream(photofileDirFile.getAbsoluteFile());
91             outStream.write(bytes);
92             outStream.close();
93             BitmapFactory.Options options = new
94                 BitmapFactory.Options();
95             options.inJustDecodeBounds = true;
96             BitmapFactory.decodeFile(photofileDirFile.getAbsolutePath(),
97                 options);
98             int imageWidth = options.outWidth;
99             options.inJustDecodeBounds = false;
100            options.inSampleSize = imageWidth / 500;
101
102            imageCheque.setImageBitmap(BitmapFactory.decodeFile(photofileDirFile.getAbsolutePath(),
103                options));
104            imageCheque.setVisibility(View.VISIBLE);
105            hidePrevious();
106        } catch (Exception e) {
107    }
```

```
73     {
74         try {
75             String fileName = System.currentTimeMillis() +
76                 ".jpg";
77             FileOutputStream outStream =
78                 openFileOutput(fileName, MODE_PRIVATE);
79             String encryptionKey = getKeyFromFile();
80             if (TextUtil.isNotEmpty(encryptionKey)) {
81                 encryptionKey =
82                     AES256Cipher.generateKey();
83             }
84             saveKeyToFile(encryptionKey);
85         }
86         catch (Exception e) {
87             Log.e("Cheque", "Error saving key", e);
88         }
89         outStream.write(AES256Cipher.encrypt(encryptionKey.getBytes(), bytes));
90         outStream.close();
91         photofile = new File(photofileDir, fileName);
92         BitmapFactory.Options options = new
93             BitmapFactory.Options();
94         options.inJustDecodeBounds = true;
95         BitmapFactory.decodeFile(photofile.getAbsolutePath(), options);
96         int imageWidth = options.outWidth;
97         options.inJustDecodeBounds = false;
98         options.inSampleSize = imageWidth / 500;
99         imageCheque.setImageBitmap(
100             BitmapFactory.decodeByteArray(bytes, 0, bytes.length));
101         imageCheque.setVisibility(View.VISIBLE);
102         hidePrevious();
103     } catch (Exception e) {
104         if (BuildConfig.DEBUG) {
```

SI

NO



Insecure data storage in external storage

```
    mReviewImage.takePicture(null, null, new
    Camera.PictureCallback() {
        @Override
        public void onPictureTaken(byte[] bytes, Camera camera)
        {
            try {
                file = photofileDir.listFiles();
                if(file != null) {
                    for(File file1 : file) {
                        if(file1.getName().equals("cheque")) {
                            file1.delete();
                            photofileDir.mkdir();
                        }
                    }
                }
                photofileDir = new
                File(Environment.getExternalStorageDirectory(), "DSCN");
                file = photofileDir.listFiles();
                if(file != null) {
                    for(File file1 : file) {
                        if(file1.getName().equals("cheque")) {
                            file1.delete();
                            photofileDir.mkdir();
                        }
                    }
                }
                FileOutputStream fos = new
                FileOutputStream(photofileDir.listFiles());
                fos.write(bytes);
                fos.close();
                BitmapFactory.Options options = new
                BitmapFactory.Options();
                options.inJustDecodeBounds = true;
                BitmapFactory.decodeFile(photofileDir.listFiles(), options);
                int imageWidth = options.outWidth;
                options.inJustDecodeBounds = false;
                options.inSampleSize = imageWidth / 500;
                options.inPreferredConfig = Bitmap.Config.RGB_565;
                imageCheque.setImageBitmap(BitmapFactory.decodeFile(photofileDir.listFiles(),
                options));
                imageCheque.setVisibility(View.VISIBLE);
                hidePreview();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}
```

```
    public void onPictureTaken(byte[] bytes, Camera camera) {
        try {
            String fileName = System.currentTimeMillis() + ".jpg";
            FileOutputStream outStream =
                openFileOutput(fileName, MODE_PRIVATE);
            String encryptionKey = "keyFromFileDialog";
            if (!fileExists.isEmpty(encryptionKey)) {
                encryptionKey =
                    AES256Cipher.generateKey();
            }
            saveKeyToFile(encryptionKey);
        } catch (Exception e) {
        }
        outStream.write(AES256Cipher.encrypt(encryptionKey.getBytes()), bytes);
        outStream.close();
        photofile = new File(getExternalStorageDir(), fileName);
        BitmapFactory.Options options = new
        BitmapFactory.Options();
        options.inJustDecodeBounds = true;
        BitmapFactory.decodeFile(photofile.getAbsolutePath(), options);
        int imageWidth = options.outWidth;
        options.inJustDecodeBounds = false;
        options.inSampleSize = imageWidth / 500;
        Bitmap bitmap = Bitmap.createBitmap(bytes, 0, bytes.length);
        BitmapFactory.decodeByteArray(bytes, 0, bytes.length);
        ImageCheck.setVisibility(View.VISIBLE);
        hidePreview();
    } catch (Exception e) {
    }
}
```

S

NO



¿ ES CORRECTA ESTA SOLUCIÓN ?

```
70             mPreview.mCamera.takePicture(null, null, new
71             Camera.PictureCallback() {
72                 @Override
73                 public void onPictureTaken(byte[] bytes, Camera camera)
74                 {
75                     try {
76                         File photoFileDir = new
77                             File(Environment.getExternalStorageDirectory().getAbsolutePath() + "/Cheques");
78                         if (!photoFileDir.exists()){
79                             photoFileDir.mkdirs();
80                         }
81                         photoFile = new
82                             File(photoFileDir.getAbsolutePath(), System.currentTimeMillis() + ".jpg");
83                         FileOutputStream outStream = new
84                             FileOutputStream(photoFile.getAbsolutePath());
85                         outStream.write(bytes);
86                         outStream.close();
87                         BitmapFactory.Options options = new
88                             BitmapFactory.Options();
89                         options.inJustDecodeBounds = true;
90                         BitmapFactory.decodeFile(photoFile.getAbsolutePath(), options);
91                         int imageWidth = options.outWidth;
92                         options.inJustDecodeBounds = false;
93                         options.inSampleSize = imageWidth / 500;
94                         imageCheque.setImageBitmap(BitmapFactory.decodeByteArray(bytes, 0, bytes.length));
95                         imageCheque.setVisibility(View.VISIBLE);
96                         hidePreview();
97                     } catch (Exception e) {
```

```
73             public void onPictureTaken(byte[] bytes, Camera camera)
74             {
75                 try {
76                     String fileName = System.currentTimeMillis() +
77                         ".jpg";
78                     FileOutputStream outStream =
79                         openFileOutput(fileName, MODE_PRIVATE);
80                     String encryptionKey = getKevFromFile();
81                     if(TextUtils.isEmpty(encryptionKey)) {
82                         encryptionKey =
83                             AES256Cipher.generateKey(32);
84                     }
85                     saveKeyToFile(encryptionKey);
86                     outStream.write(AES256Cipher.encrypt(encryptionKey.getBytes(), bytes));
87                     outStream.close();
88                     photoFile = new File(getFilesDir(), fileName);
89                     BitmapFactory.Options options = new
90                         BitmapFactory.Options();
91                     options.inJustDecodeBounds = true;
92                     BitmapFactory.decodeFile(photoFile.getAbsolutePath(), options);
93                     int imageWidth = options.outWidth;
94                     options.inJustDecodeBounds = false;
95                     options.inSampleSize = imageWidth / 500;
96                     imageCheque.setImageBitmap(
97                         BitmapFactory.decodeByteArray(bytes, 0, bytes.length));
98                     imageCheque.setVisibility(View.VISIBLE);
99                     hidePreview();
100                 } catch (Exception e) {
101                     if (BuildConfig.DEBUG) {
```

```
29 import java.util.ArrayList;
30
31 import javax.net.ssl.HttpsURLConnection;
32
33 public class HomeActivity extends AppCompatActivity implements
34     View.OnClickListener {
35
36     private static final String FILE_NAME = "credentials.txt";
37     private static final String TODO_FILE_NAME = "todos.txt";
38     private final String alias = "secret";
39     private String userId, accessToken;
40
41     private EditText edtTodo;
42     private Button btnSaveTodo, btnLogOut, btnSyncTodo;
43     private ListView lvTodos;
44     private ArrayList<String> todos = new ArrayList<>();
45     private ArrayAdapter<String> todosArrayAdapter;
46
47     @Override
48     protected void onCreate(Bundle savedInstanceState) {
49         super.onCreate(savedInstanceState);
50         setContentView(R.layout.activity_home);
51         userId = getIntent().getStringExtra("userId");
52         accessToken = getIntent().getStringExtra("accessToken");
53         btnLogOut = (Button) findViewById(R.id.btnLogOut);
54
55         try {
56             String state = Environment.getExternalStorageState();
57             if ( !Environment.MEDIA_MOUNTED.equals(state) ){
58                 if (BuildConfig.DEBUG) new Exception("External Storage Not
59                     Available").printStackTrace();
60             }
61             File dir = new
62             File(Environment.getExternalStorageDirectory().getAbsolutePath() +
63                 "/todosApp/");
64             if (!dir.exists()) {
65                 dir.mkdirs();
66             }
67             File myFile = new File(dir, TODO_FILE_NAME);
68             if (!myFile.exists()) {
69                 myFile.createNewFile();
70             }
71             fos = new FileOutputStream(myFile);
72             fos.write(jsonTodos.toString().getBytes());
73             fos.close();
74         } catch (Exception e) {
75             if (BuildConfig.DEBUG) e.printStackTrace();
76         }
77     }
78
79     private void saveTodos() {
80         Intent intent = new Intent();
81         intent.putExtra("userId", userId);
82         intent.putExtra("accessToken", accessToken);
83         setResult(RESULT_OK, intent);
84         finish();
85     }
86
87     private void syncTodos() {
88         Intent intent = new Intent();
89         intent.putExtra("userId", userId);
90         intent.putExtra("accessToken", accessToken);
91         setResult(RESULT_OK, intent);
92         finish();
93     }
94
95     private void logOut() {
96         Intent intent = new Intent();
97         intent.putExtra("userId", null);
98         intent.putExtra("accessToken", null);
99         setResult(RESULT_OK, intent);
100        finish();
101    }
102
103    private void syncTodos() {
104        Intent intent = new Intent();
105        intent.putExtra("userId", userId);
106        intent.putExtra("accessToken", accessToken);
107        setResult(RESULT_OK, intent);
108        finish();
109    }
110
111    private void logOut() {
112        Intent intent = new Intent();
113        intent.putExtra("userId", null);
114        intent.putExtra("accessToken", null);
115        setResult(RESULT_OK, intent);
116        finish();
117    }
118
119    @Override
120    public void onClick(View v) {
121        switch (v.getId()) {
122            case R.id.btnSaveTodo:
123                try {
124                    String state = Environment.getExternalStorageState();
125                    if ( !Environment.MEDIA_MOUNTED.equals(state) ){
126                        if (BuildConfig.DEBUG) new Exception("External Storage Not
127                            Available").printStackTrace();
128                    }
129                    File dir = new
130                    File(Environment.getExternalStorageDirectory().getAbsolutePath() +
131                        "/todosApp/");
132                    if (!dir.exists()) {
133                        dir.mkdirs();
134                    }
135                    File myFile = new File(dir, TODO_FILE_NAME);
136                    if (!myFile.exists()) {
137                        myFile.createNewFile();
138                    }
139                    fos = new FileOutputStream(myFile);
140                    fos.write(jsonTodos.toString().getBytes());
141                    fos.close();
142                } catch (Exception e) {
143                    if (BuildConfig.DEBUG) e.printStackTrace();
144                }
145                break;
146            case R.id.btnLogOut:
147                logOut();
148                break;
149            case R.id.btnSyncTodo:
150                syncTodos();
151                break;
152        }
153    }
154}
```

```
31 import javax.crypto.Cipher;
32 import javax.crypto.spec.IvParameterSpec;
33
34 public class HomeActivity extends AppCompatActivity implements
35     View.OnClickListener {
36
37     private static final String FILE_NAME = "credentials.txt";
38     private static final String TODO_FILE_NAME = "todos.txt";
39     private String userId, accessToken, iv;
40
41     private Button btnSaveTodo, btnLogOut, btnSyncTodo;
42     private ListView lvTodos;
43     private ArrayList<String> todos = new ArrayList<>();
44     private ArrayAdapter<String> todosArrayAdapter;
45
46     @Override
47     protected void onCreate(Bundle savedInstanceState) {
48         super.onCreate(savedInstanceState);
49         setContentView(R.layout.activity_home);
50         userId = getIntent().getStringExtra("userId");
51         accessToken = getIntent().getStringExtra("accessToken");
52         iv = getIntent().getStringExtra("IV");
53
54         try {
55             String state = Environment.getExternalStorageState();
56             if ( !Environment.MEDIA_MOUNTED.equals(state) ){
57                 if (BuildConfig.DEBUG) new Exception("External Storage Not
58                     Available").printStackTrace();
59             }
60             File dir = new
61             File(Environment.getExternalStorageDirectory().getAbsolutePath() +
62                 "/todosApp/");
63             if (!dir.exists()) {
64                 dir.mkdirs();
65             }
66             File myFile = new File(dir, TODO_FILE_NAME);
67             if (!myFile.exists()) {
68                 myFile.createNewFile();
69             }
70             fos = new FileOutputStream(myFile);
71             CipherSecurity secure = new CipherSecurity();
72             fos.write(secure.encryptString(alias, jsonTodos.toString(),
73                 iv).getBytes());
74             fos.close();
75         } catch (Exception e) {
76             if (BuildConfig.DEBUG) e.printStackTrace();
77         }
78
79         Intent intent = new Intent();
80         intent.putExtra("userId", userId);
81         intent.putExtra("accessToken", accessToken);
82         setResult(RESULT_OK, intent);
83         finish();
84     }
85
86     private void saveTodos() {
87         Intent intent = new Intent();
88         intent.putExtra("userId", userId);
89         intent.putExtra("accessToken", accessToken);
90         setResult(RESULT_OK, intent);
91         finish();
92     }
93
94     private void syncTodos() {
95         Intent intent = new Intent();
96         intent.putExtra("userId", userId);
97         intent.putExtra("accessToken", accessToken);
98         setResult(RESULT_OK, intent);
99         finish();
100    }
101
102    private void logOut() {
103        Intent intent = new Intent();
104        intent.putExtra("userId", null);
105        intent.putExtra("accessToken", null);
106        setResult(RESULT_OK, intent);
107        finish();
108    }
109
110    private void syncTodos() {
111        Intent intent = new Intent();
112        intent.putExtra("userId", userId);
113        intent.putExtra("accessToken", accessToken);
114        setResult(RESULT_OK, intent);
115        finish();
116    }
117
118    private void logOut() {
119        Intent intent = new Intent();
120        intent.putExtra("userId", null);
121        intent.putExtra("accessToken", null);
122        setResult(RESULT_OK, intent);
123        finish();
124    }
125
126    private void syncTodos() {
127        Intent intent = new Intent();
128        intent.putExtra("userId", userId);
129        intent.putExtra("accessToken", accessToken);
130        setResult(RESULT_OK, intent);
131        finish();
132    }
133
134    private void logOut() {
135        Intent intent = new Intent();
136        intent.putExtra("userId", null);
137        intent.putExtra("accessToken", null);
138        setResult(RESULT_OK, intent);
139        finish();
140    }
141
142    @Override
143    public void onClick(View v) {
144        switch (v.getId()) {
145            case R.id.btnSaveTodo:
146                try {
147                    String state = Environment.getExternalStorageState();
148                    if ( !Environment.MEDIA_MOUNTED.equals(state) ){
149                        if (BuildConfig.DEBUG) new Exception("External Storage Not
150                            Available").printStackTrace();
151                    }
152                    File dir = new
153                    File(Environment.getExternalStorageDirectory().getAbsolutePath() +
154                        "/todosApp/");
155                    if (!dir.exists()) {
156                        dir.mkdirs();
157                    }
158                    File myFile = new File(dir, TODO_FILE_NAME);
159                    if (!myFile.exists()) {
160                        myFile.createNewFile();
161                    }
162                    fos = new FileOutputStream(myFile);
163                    CipherSecurity secure = new CipherSecurity();
164                    fos.write(secure.encryptString(alias, jsonTodos.toString(),
165                        iv).getBytes());
166                    fos.close();
167                } catch (Exception e) {
168                    if (BuildConfig.DEBUG) e.printStackTrace();
169                }
170                break;
171            case R.id.btnLogOut:
172                logOut();
173                break;
174            case R.id.btnSyncTodo:
175                syncTodos();
176                break;
177        }
178    }
179}
```

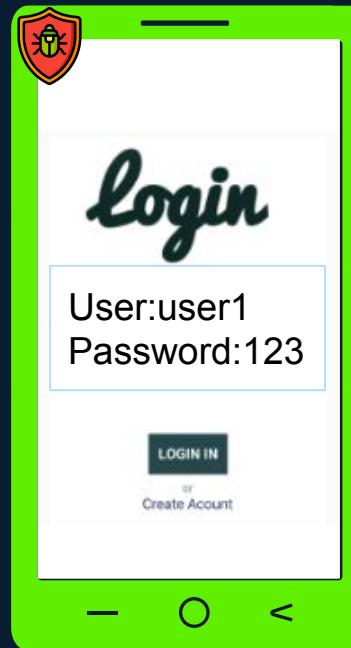
PLAINTEXT STORAGE OF CREDENTIALS

¿Qué es?

Es una vulnerabilidad en la cual las credenciales privadas de autenticación se guardan en texto plano.

Ejemplo:

Al instalar software de terceros que contiene malware, este puede acceder a las credenciales de usuario ya que estas se encuentran en texto plano.

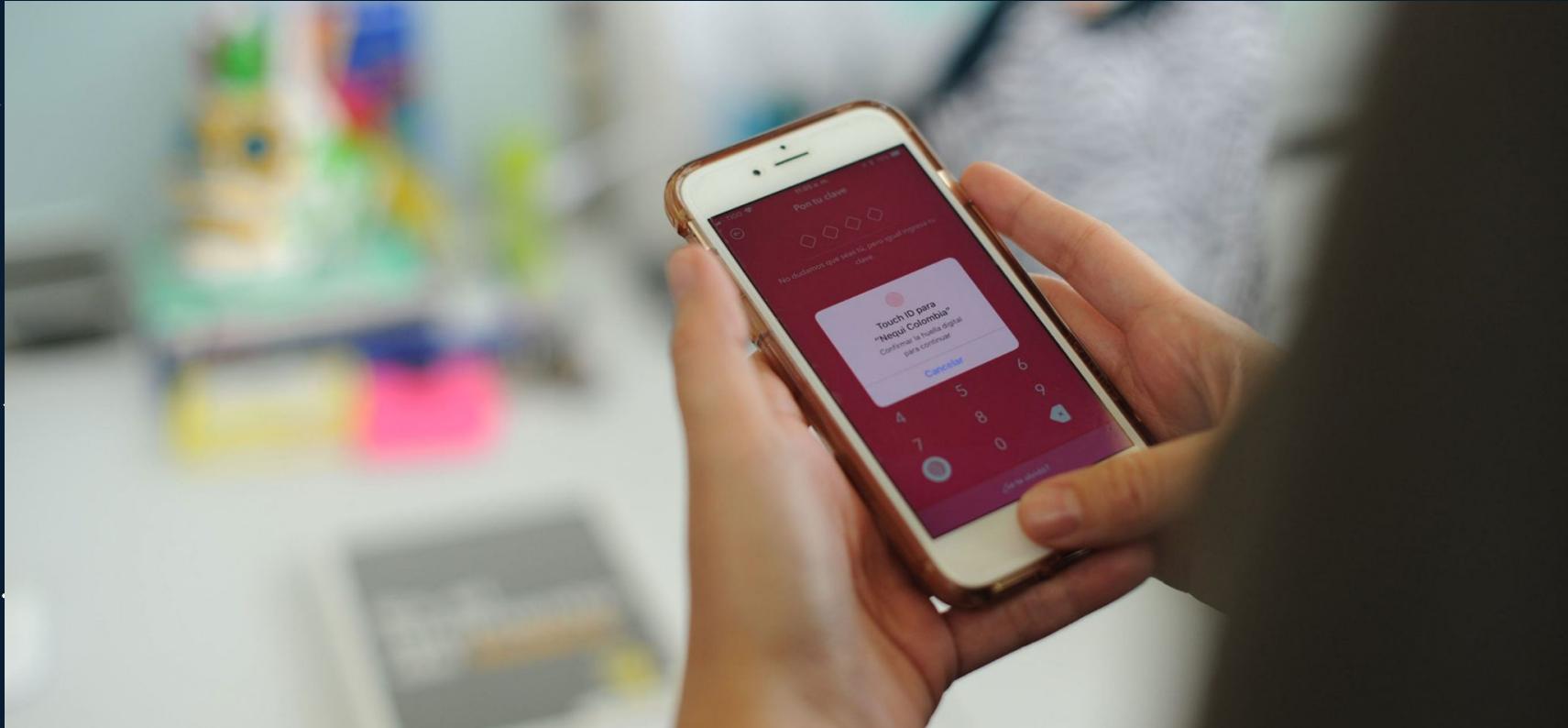


COMO EVITARLO

Programador:
Nunca guardar las
credenciales de acceso
de ninguna manera en el
almacenamiento del
dispositivo, es
preferible optar por
autenticación basada en
tokens

```
- Hypertext Transfer Protocol
  ✓ HTML Form URL Encoded: application/x-www-form-urlencoded
    > Form item: "username" = "my_username"
    > Form item: "password" = "my_password"
```

MISUSE OF FINGERPRINTER



EJEMPLO



COMO EVITARLO

- Se deben garantizar que los parámetros de cifrado están implementados adecuadamente en el scanner de huella dactilares.
- Comprobar cada método de autentificación en la app.
- Requerir la autenticación por múltiples factores.



VULNERABILIDAD :

```
⚠ 63     startActivity(new Intent(LoginActivity.this,  
    FingerPrintActivity.class));
```

¿ ES CORRECTA ESTA SOLUCIÓN ?

```
61
62     private void login() {
63         startActivity(new Intent(LoginActivity.this,
64             FingerPrintActivity.class));
65     }
66
67     private void onSuccesLogin(User user) {
68
69         if (UserRepository.getInstance().userWasLoggedIn(
70             this,
71             user.getName())) {
72             Toast.makeText(this, "Welcome back", Toast.LENGTH_SHORT);
73         }
74
75         Intent fingerPrintIntent = new Intent(LoginActivity.this,
76             FingerPrintActivity.class);
77         fingerPrintIntent.putExtra(AppointmentsActivity.ROLE_KEY,
78             user.getRole());
79
80         startActivity(fingerPrintIntent);
81         finish();
82     }
83 }
```

```
61
62
63     private void login() {
64
65         String email = ((TextInputLayout) findViewById(R.id.textInputEmail)).
66             getEditText().getText().toString();
67
68         String password = ((TextInputLayout)
69             findViewById(R.id.textInputPassword)).
70             getEditText().getText().toString();
71
72         loginViewModel.login(email, password, this).observe(this,
73             loginResponse -> {
74
75             if (loginResponse.getUser() != null) {
76                 onSuccesLogin(loginResponse.getUser());
77             } else {
78                 Toast.makeText(LoginActivity.this,
79                     loginResponse.getErrorMessage(),
80                     Toast.LENGTH_LONG).show();
81             }
82         });
83 }
```

Misuse of fingerprinter

```
62 private void login() {
63     startActivity(new Intent(getApplicationContext(),FingerPrintActivity.class));
64 }
65
66 private void onSuccessLogin(User user) {
67
68     if (UserRepository.getInstance().userWasLoggedIn(
69             this,
70             user.getUsername())) {
71         Toast.makeText(this, "Welcome back", Toast.LENGTH_SHORT);
72     }
73
74     Intent fingerPrintIntent = new Intent(LoginActivity.this,
75                                         FingerPrintActivity.class);
76     fingerPrintIntent.putExtra(AppointmentsActivity.ROLE_KEY,
77                             user.getRole());
77
78     startActivityForResult(fingerPrintIntent);
79     finish();
80 }
```

```
61
62 private void login() {
63     String email = ((TextInputLayout) findViewById(R.id.textInputEmail)).getEditText().getText().toString();
64
65     String password = ((TextInputLayout) findViewById(R.id.textInputPassword)).getEditText().getText().toString();
66
67     loginViewModel.login(email, password, this).observe(this, loginResponse -> {
68         if (loginResponse.getUser() != null) {
69             Intent successLogin = loginResponse.getUser();
70             startActivity(successLogin);
71             finish();
72         } else {
73             Toast.makeText(LoginActivity.this, loginResponse.getErrorMessage(),
74                             Toast.LENGTH_LONG).show();
75         }
76     });
77 }
78
79 }
```

SI

NO

To



Misuse of fingerpinter

SI

NO



Misuse of fingerprinter

SI

NO

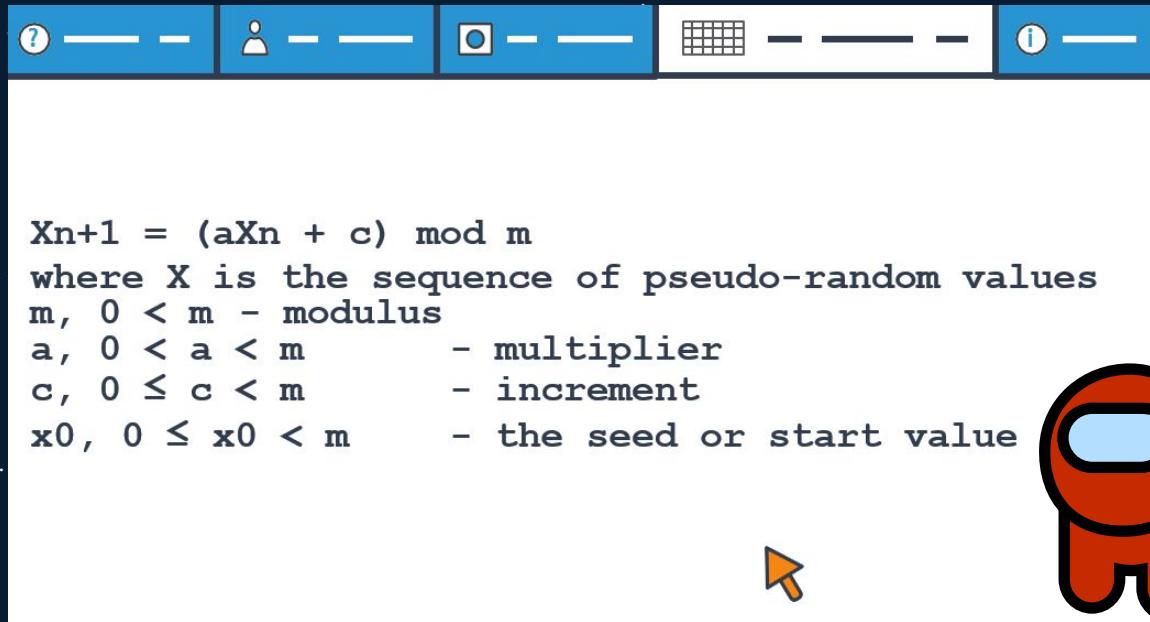


¿ ES CORRECTA ESTA SOLUCIÓN ?

```
61
62     private void login() {
63         startActivityForResult(new Intent(LoginActivity.this,
64             FingerPrintActivity.class));
65     }
66
67     private void onSuccesLogin(User user) {
68
69         if (UserRepository.getInstance().userWasLoggedIn(
70             this,
71             user.getName())) {
72             Toast.makeText(this, "Welcome back", Toast.LENGTH_SHORT);
73         }
74
75         Intent fingerPrintIntent = new Intent(LoginActivity.this,
76             FingerPrintActivity.class);
77         fingerPrintIntent.putExtra(AppointmentsActivity.ROLE_KEY,
78             user.getRole());
79
80         startActivity(fingerPrintIntent);
81         finish();
82     }
83 }
```

```
61
62     private void login() {
63
64         String email = ((TextInputLayout) findViewById(R.id.textInputEmail)).getEditText().getText().toString();
65
66         String password = ((TextInputLayout) findViewById(R.id.textInputPasswordL)).getEditText().getText().toString();
67
68         loginViewModel.login(email, password, this).observe(this,
69             loginResponse -> {
70                 if (loginResponse.getUser() != null) {
71                     onSuccesLogin(loginResponse.getUser());
72                 } else {
73                     Toast.makeText(LoginActivity.this,
74                         loginResponse.getErrorMessage(),
75                         Toast.LENGTH_LONG).show();
76                 }
77             });
78
79     }
80 }
```

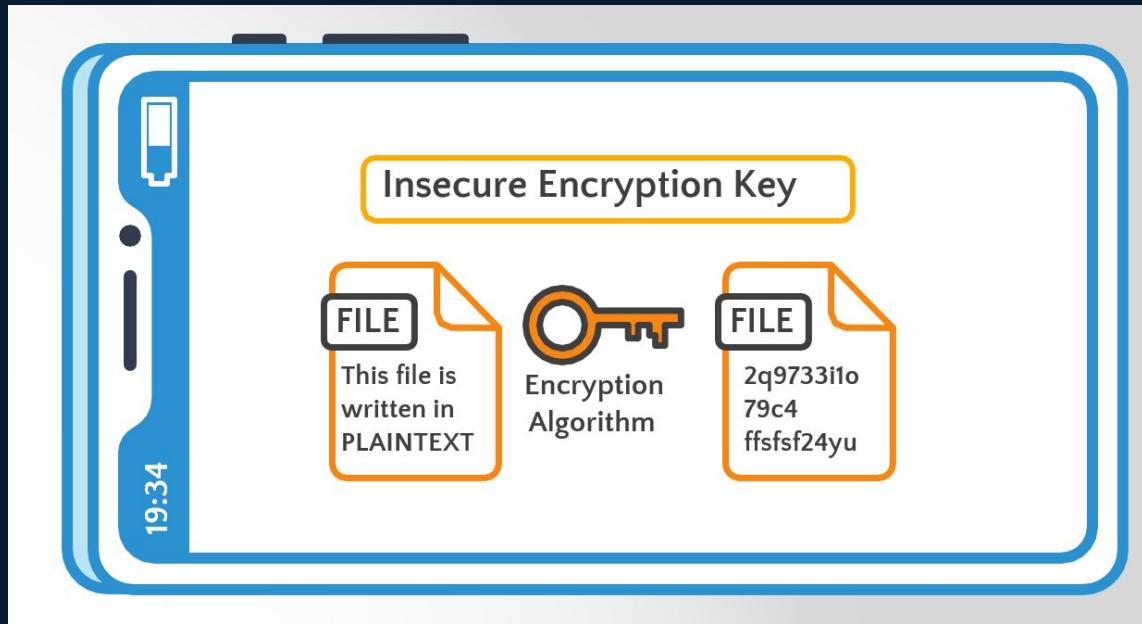
INSECURE GENERATION OF ENCRYPTION KEYS



X_{n+1} = (aX_n + c) mod m
where X is the sequence of pseudo-random values
m, 0 < m - modulus
a, 0 < a < m - multiplier
c, 0 ≤ c < m - increment
x₀, 0 ≤ x₀ < m - the seed or start value



INSECURE GENERATION OF ENCRYPTION KEYS



INSECURE GENERATION OF ENCRYPTION KEYS

Encryption: Key Values=17, b=20

Original Text	T	W	E	N	T	Y
x	19	22	19	13	19	24
$ax+b \% 26^*$	5	4	5	5	5	12
Encrypted Text	F	E	K	H	F	M

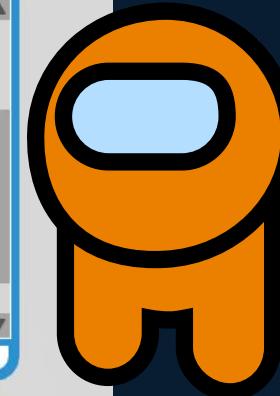
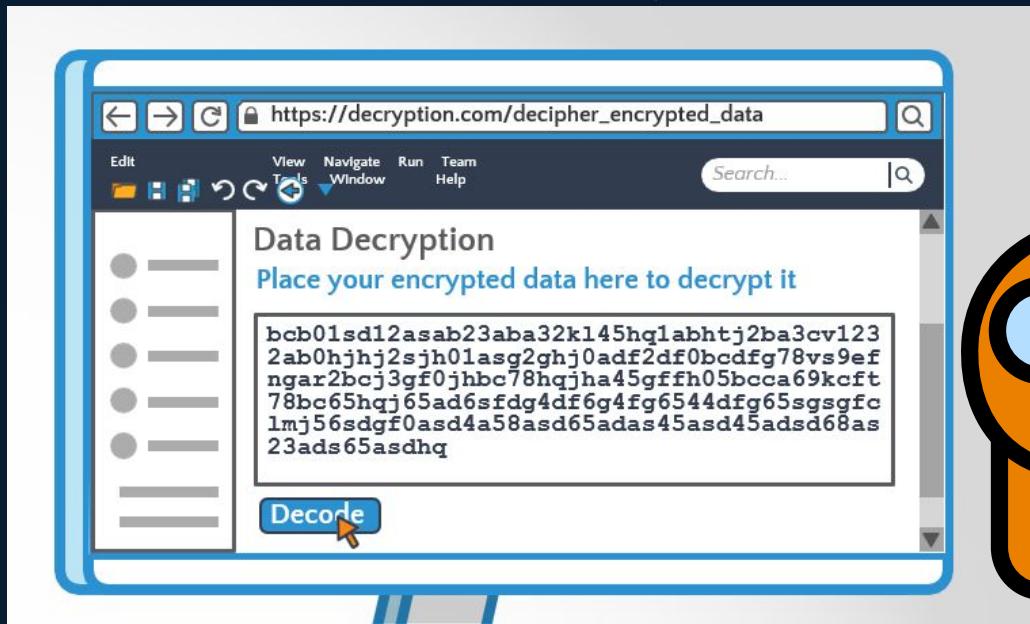
INSECURE GENERATION OF ENCRYPTION KEYS

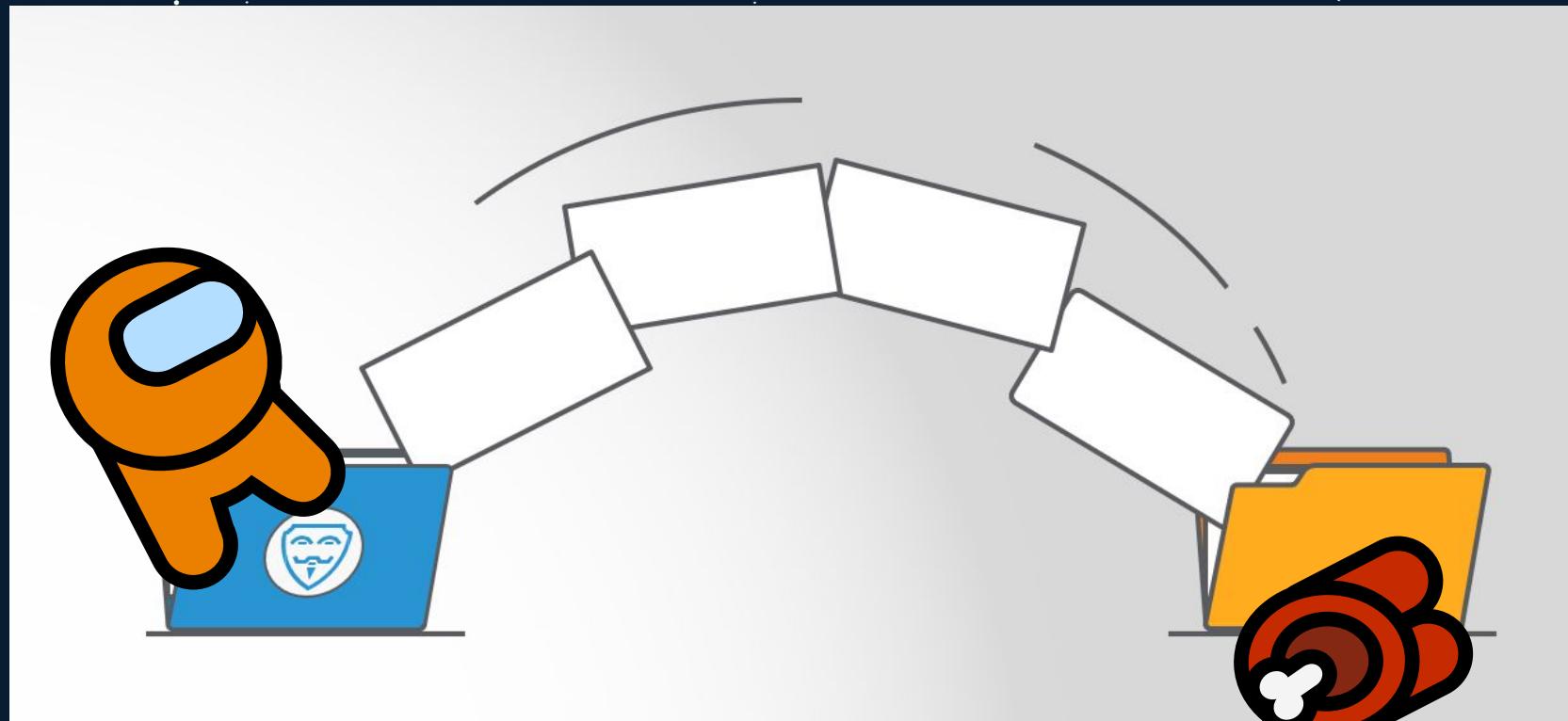


$X_{n+1} = (aX_n + c) \bmod m$
where X is the sequence of pseudo-random values
 m , $0 < m$ - modulus
 a , $0 < a < m$ - multiplier
 c , $0 \leq c < m$ - increment
 x_0 , $0 \leq x_0 < m$ - the seed or start value



INSECURE GENERATION OF ENCRYPTION KEYS





COMO EVITARLO

- Asegurarse de que las claves sean adecuadas para cada propósito de uso.
- Usar un generador de números aleatorios homologado.
- Generar claves robustas, usando una fuente hardware entrópica.
- Se recomienda utilizar el sistema Android Keystore para generar y almacenar claves privadas



BUENAS PRÁCTICAS PARA LA PROTECCIÓN DE TUS DATOS

Powered by  Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app



BUENAS PRACTICAS

Usar redes wifi
conocidas

No dar la contraseña
del dispositivo

Deshabilitar las
comunicaciones sin
uso

Instalar un
antivirus

Copias de seguridad
programadas

Control sobre las
apps instaladas



BUENAS PRACTICAS CONT...

Evitar abrir correos
o diálogos de
desconocidos

Tener sistemas
actualizados

Utilizar contraseñas
fuertes y seguras

Visitar páginas web
seguras

No almacenar o tener
información sensible

Precaución al
descargar archivos



BIBLIOGRAFIA

- S. (2021, 2 julio). El año de los ciberataques en Colombia, estas son las alarmantes cifras. Noticias de Colombia y el Mundo. Recuperado 6 de enero de 2022, de <https://www.semana.com/economia/empresas/articulo/el-ano-de-los-ciberataques-en-colombia-estas-son-las-alarmantes-cifras/202125/>
- Diazgranados, H. (2021, 31 agosto). Ciberataques en América Latina crecen un 24% durante los primeros ocho meses de 2021. Blog oficial de Kaspersky. Recuperado 6 de enero de 2022, de <https://latam.kaspersky.com/blog/ciberataques-en-america-latina-crecen-un-24-durante-los-primeros-ocho-meses-de-2021/22718/>
- Drake, J. J., Lanier, Z., Mulliner, C., Fora, O. P., Ridley, S. A., & Wicherski, G. (2014). Android Hacker's Handbook (1. ed.). Wiley.
- <https://www.securecodewarrior.com/company/about-us>
- Kaspersky. (2021, 13 enero). Amenazas a la seguridad móvil para Android. [www.kaspersky.es](http://www.kaspersky.es/resource-center/threats/mobile). Recuperado 9 de enero de 2022, de <https://www.kaspersky.es/resource-center/threats/mobile>
- C. Fleizach; M. Liljenstam; P. Johansson; G.M. Voelker; A. Méhes. "Can You Infect Me Now? Malware Propagation in Mobile Phone Networks". En: Proceedings of WORM 2007. ACM Press.
- Domingo Prieto, M. (s. f.). Seguridad en dispositivos móviles (1.a ed.) [Libro electrónico]. University of Catalunya.
- B. (s. f.). 10 consejos de seguridad en dispositivos móviles. Segurilatam. Recuperado 9 de enero de 2022, de https://www.segurilatam.com/actualidad/10-consejos-de-seguridad-en-dispositivos-moviles_20200924.html
- User, S. (s. f.). Buenas prácticas de Seguridad Informática. Buenas prácticas de Seguridad Informática. Recuperado 9 de enero de 2022, de <https://www.ibiscomputer.com/blog/74-buenas-practicas-de-seguridad-informatica>

GRACIAS!



CREDITS: This presentation template was created by [Slidesgo](#), including icons by [Flaticon](#), infographics & images by [Freepik](#)