Security in Android

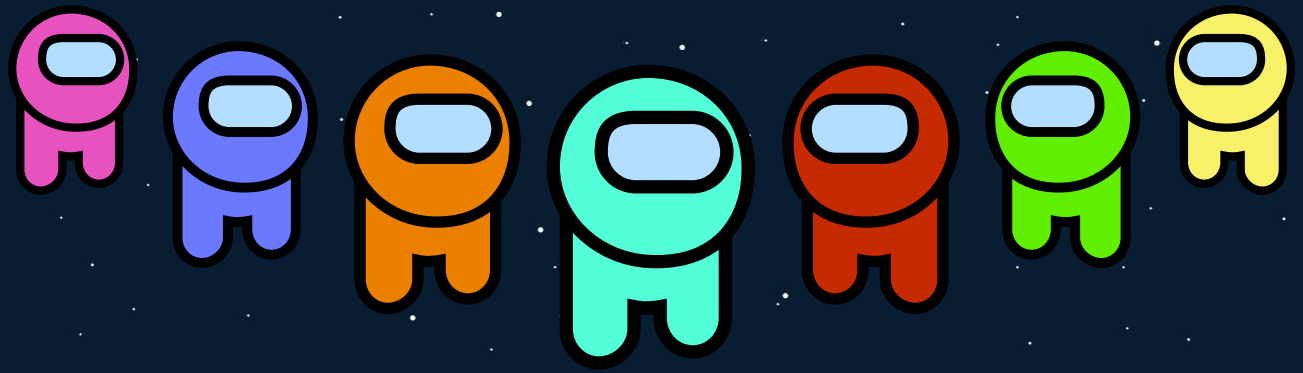
GreenPS

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

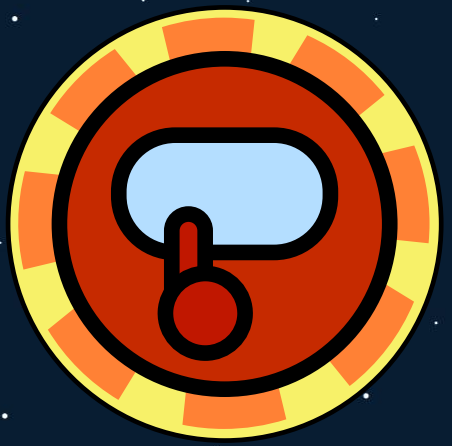
Contents of this Template

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.

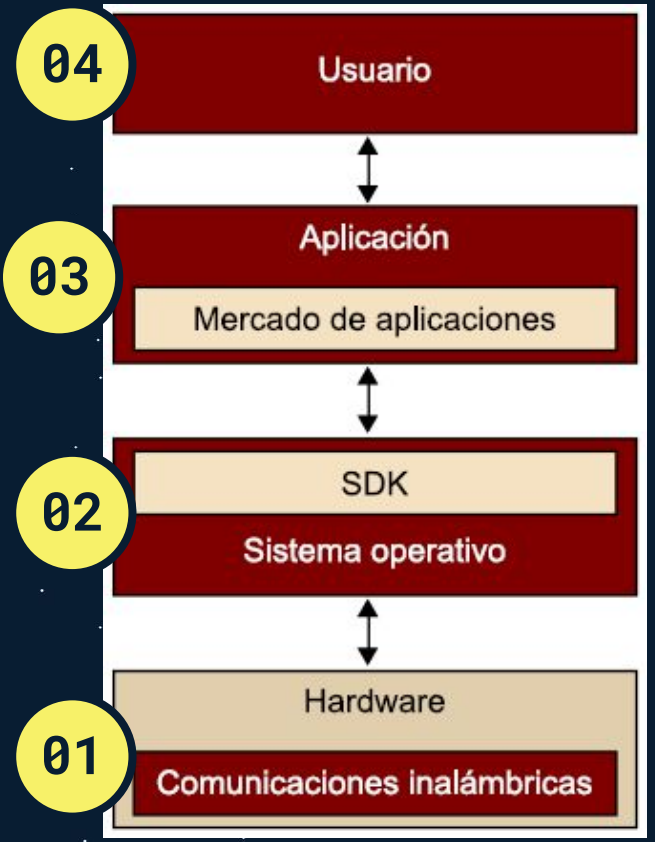
Capas de seguridad

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.



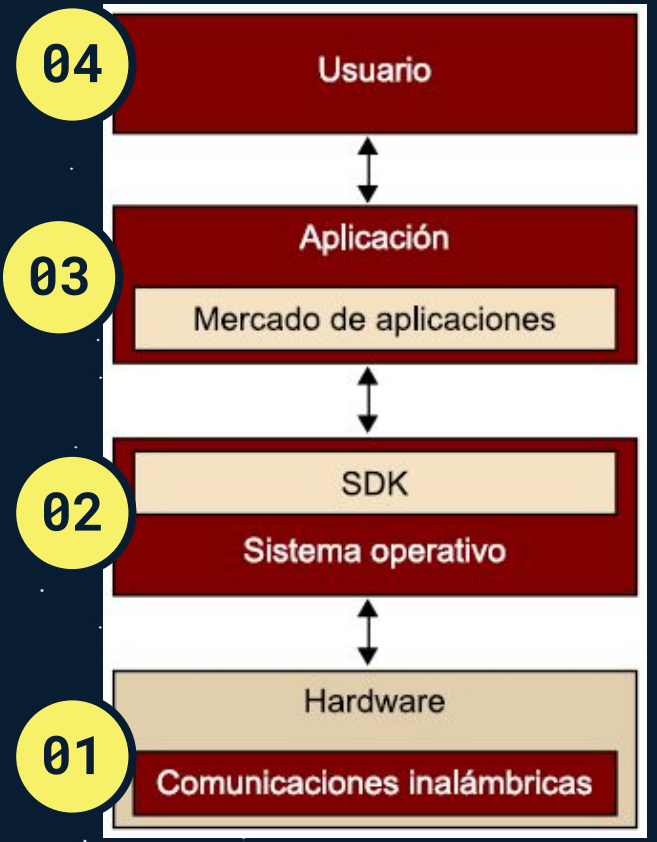
Capas de seguridad

03

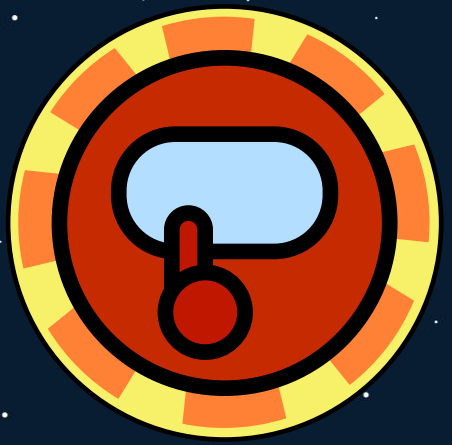
En este nivel es muy importante revisar las medidas de seguridad que se han tomado para que las aplicaciones no puedan desestabilizar el sistema.

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

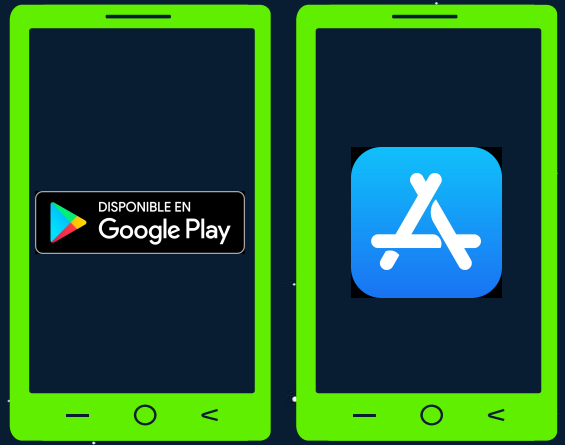


Tipos de ataques

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

Vectores de ataque :

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

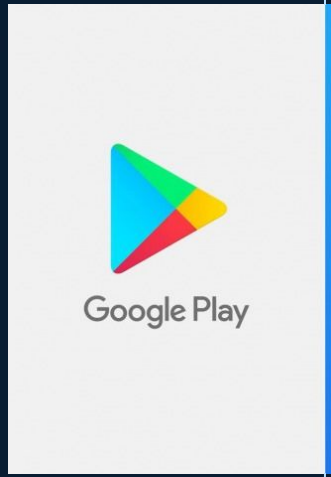


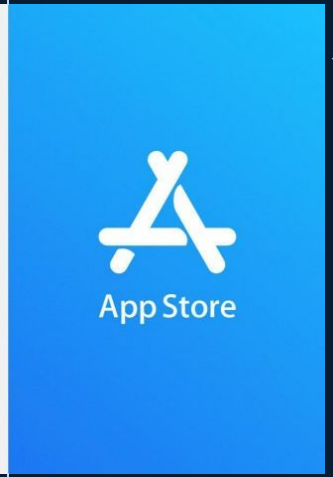
Mercados de Aplicaciones

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 atractividad 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

'hatsApp Inc.• Weather

271

3+

Loading device compatibility...

Add to Wishlist

Install

C :

Whatsapp Messenger

WhatsApp .Inc ** Communication

484

Contains ads

Add to Wishlist

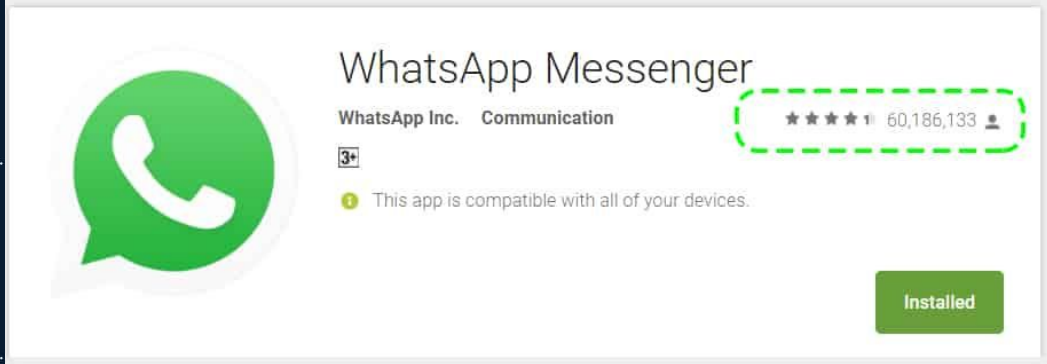
Install

Mercados de Aplicaciones

Mercados de Aplicaciones

Mercados de Aplicaciones

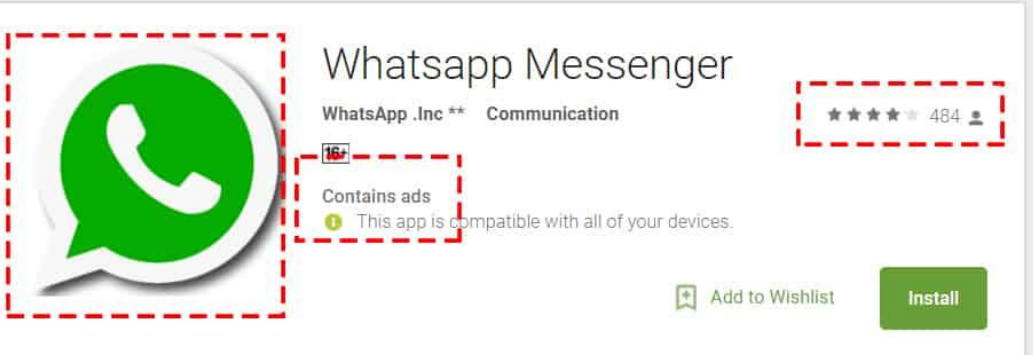
A :

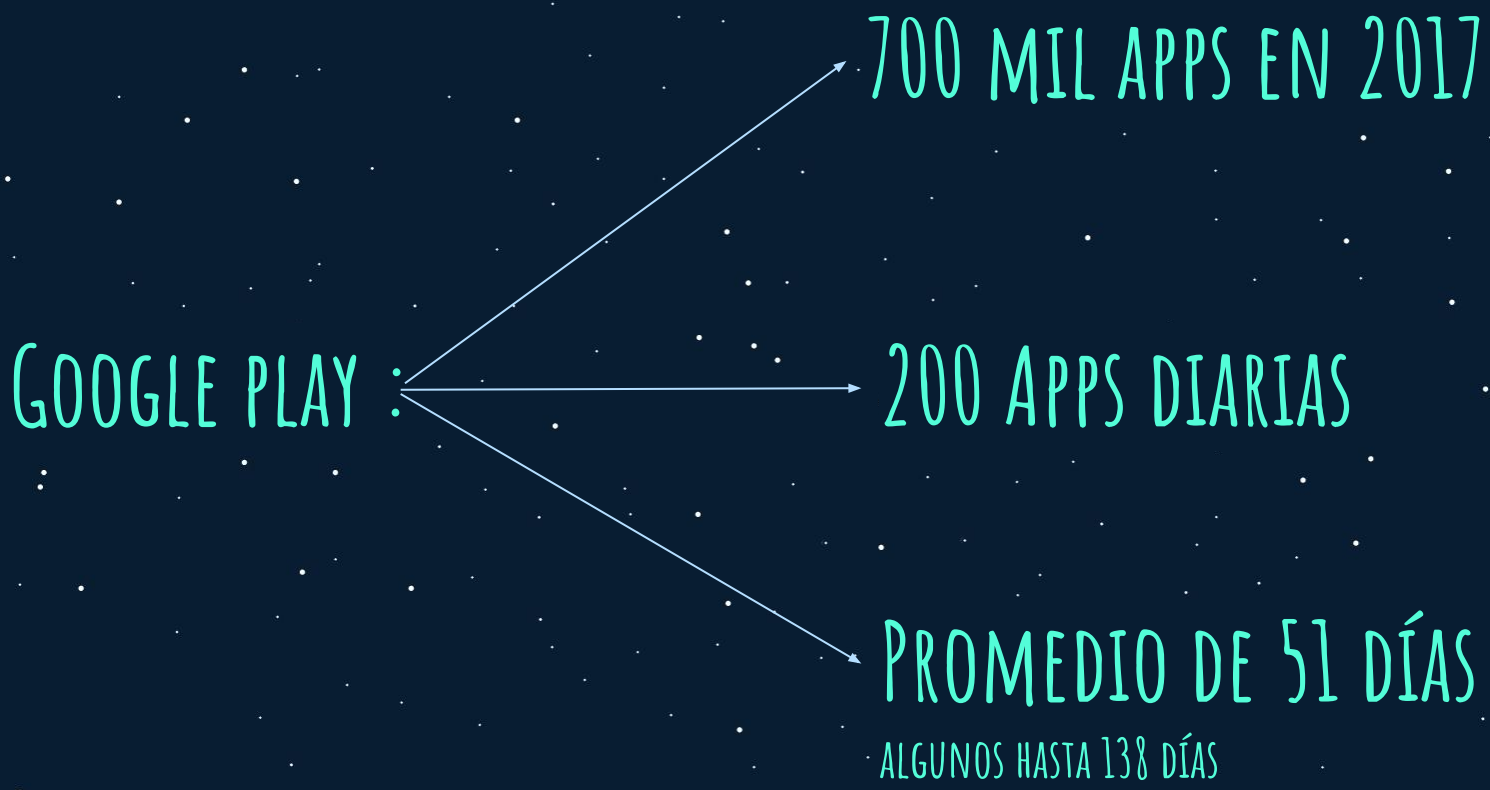

WhatsApp Messenger
WhatsApp Inc. Communication
60,186,133
3+
This app is compatible with all of your devices.
Installed

B :


WhatsApp Messenger
WhatsApp Inc.• Weather
271
3+
Loading device compatibility...
Add to Wishlist
Install

C :


Whatsapp Messenger
WhatsApp .Inc ** Communication
484
16+
Contains ads
This app is compatible with all of your devices.
Add to Wishlist
Install

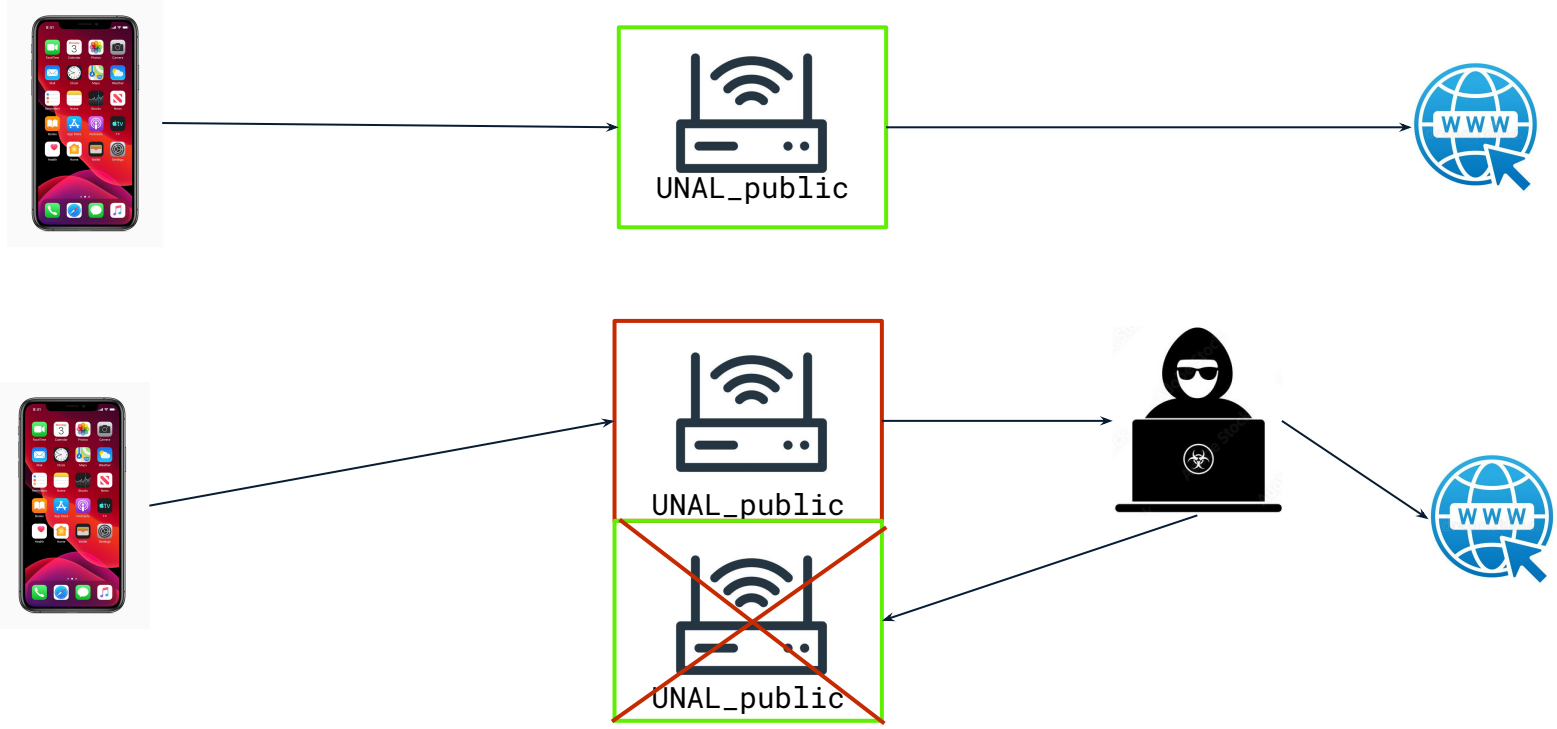


Fuente: Google play && ElevenPaths

Redes

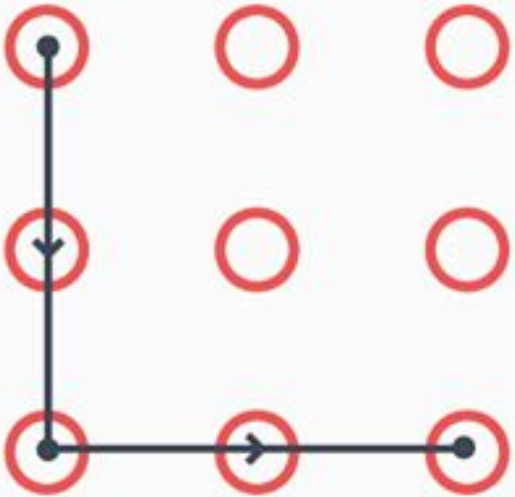
- Mensajes MMS maliciosos.
- Antenas y hotspots fraudulentos.
- Interceptación de mensajes.
- Espionaje.
- Redes públicas inseguras.

Hotspots Fraudulentos



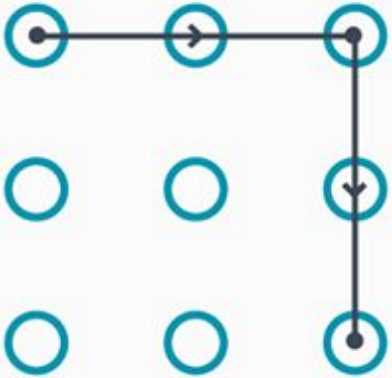
Dispositivos

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

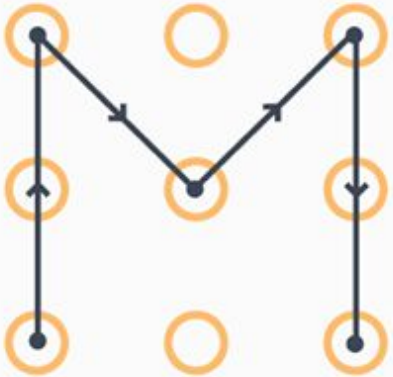


44% inicia arriba
a la izquierda.

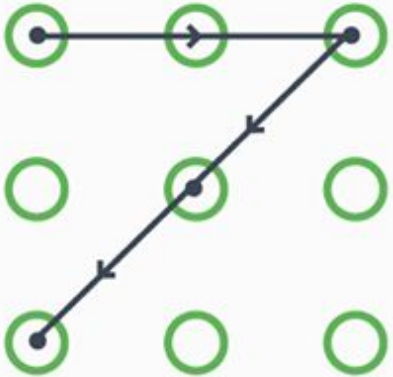
77% incia
de alguna esquina.



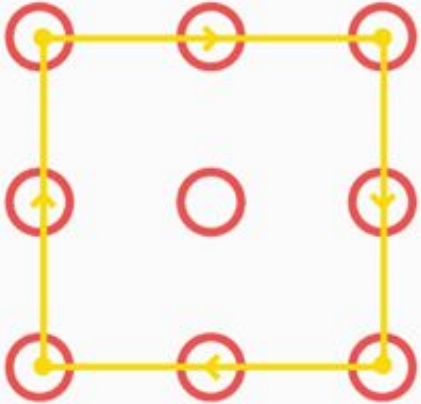
Más del 50% utiliza
una secuencia de 5 puntos.



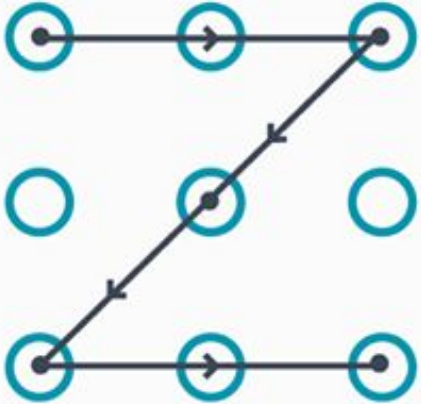
La mayoría
utiliza forma de letras...



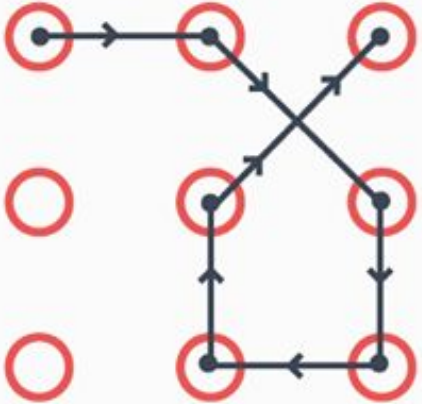
...o números.



La “O” es el unlock
más común...



...seguido por la “Z”.



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

Vulnerabilidades comunes

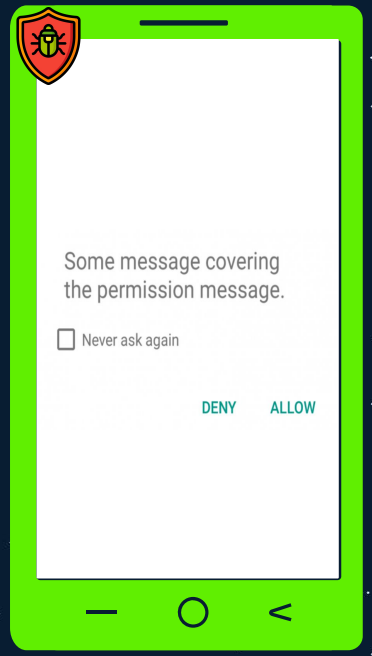
Tapjacking

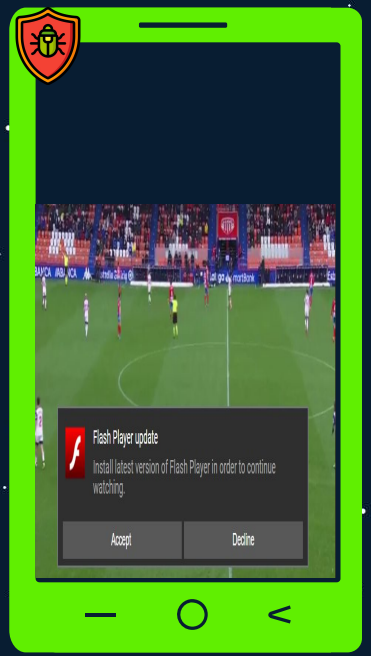
¿Qué es?

Es una vulnerabilidad de seguridad que se encuentra en los usos inadecuados de l a 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 :

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

A :

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

B :

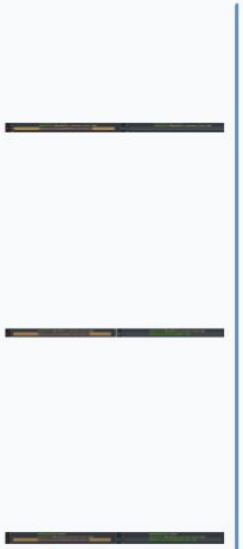
```
62          android:src="@drawable/ic_backspace_black_24dp"
63          android:focusableInTouchMode="true"/>
64
```

C :

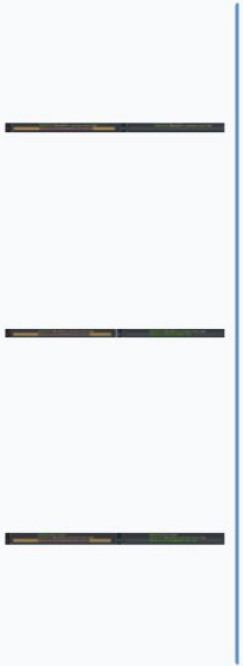
```
61          android:gravity="center"
62          android:src="@drawable/ic_backspace_black_24dp"
63          android:filterTouchesWhenObscured="true"/>
64
```

TAPJACKING

TAPJACKING



TAPJACKING



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.

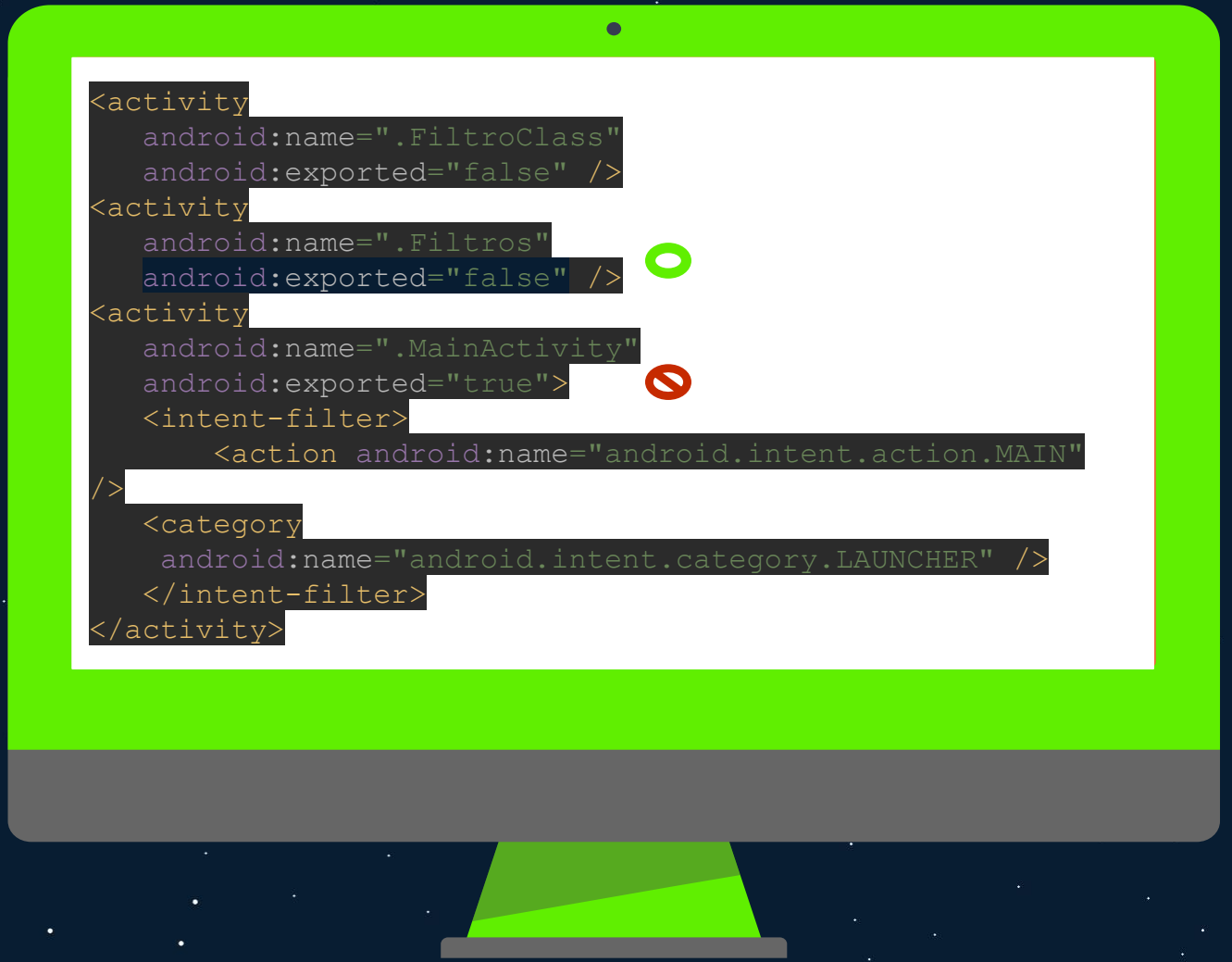
Como evitarlo

Programador:

Usar las lineas ⇒ Accesibilidad menor.

- Agregar permisos para que solo la app reciba información.

Usuario:
Antivirus, programas de limpieza.



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 paginas que no tienen protocolo https

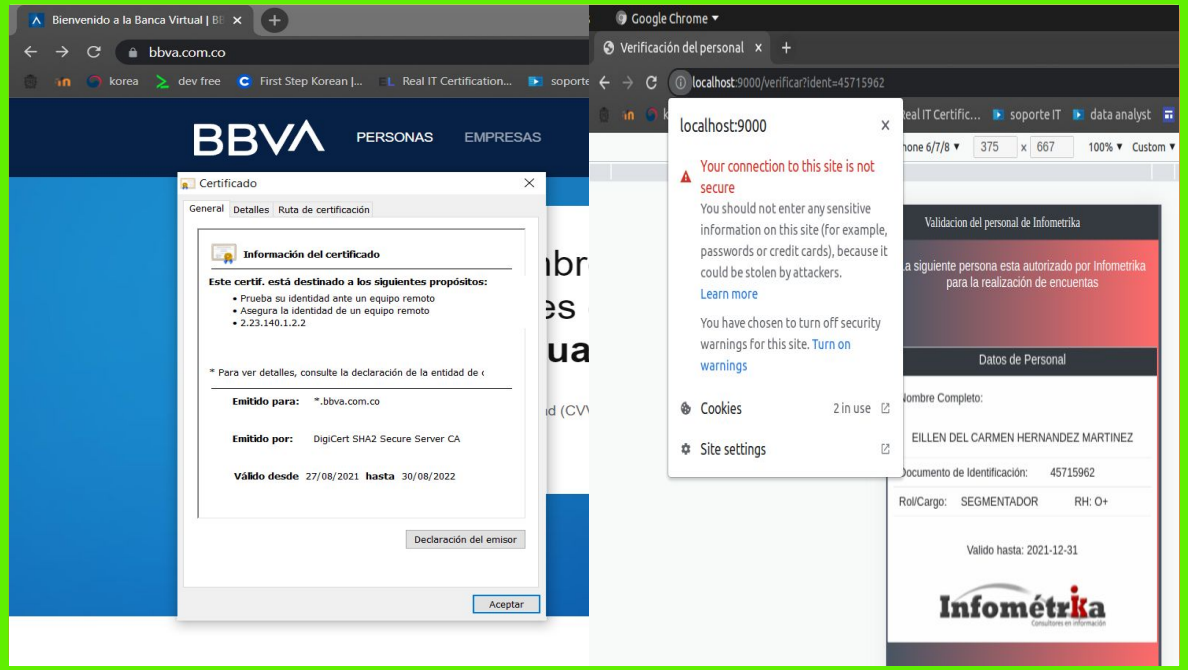
Como evitarlo

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.



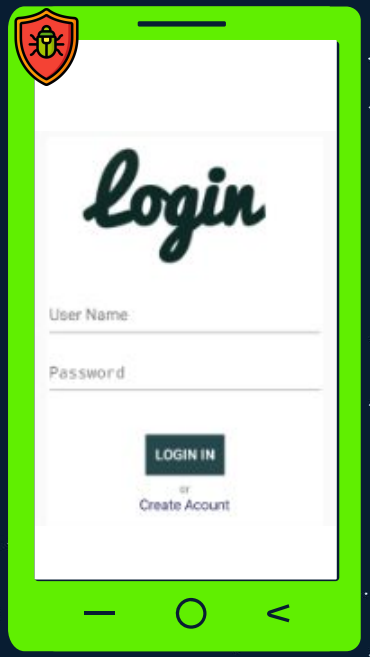
WEAK LOCKOUT MECHANISM

¿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.

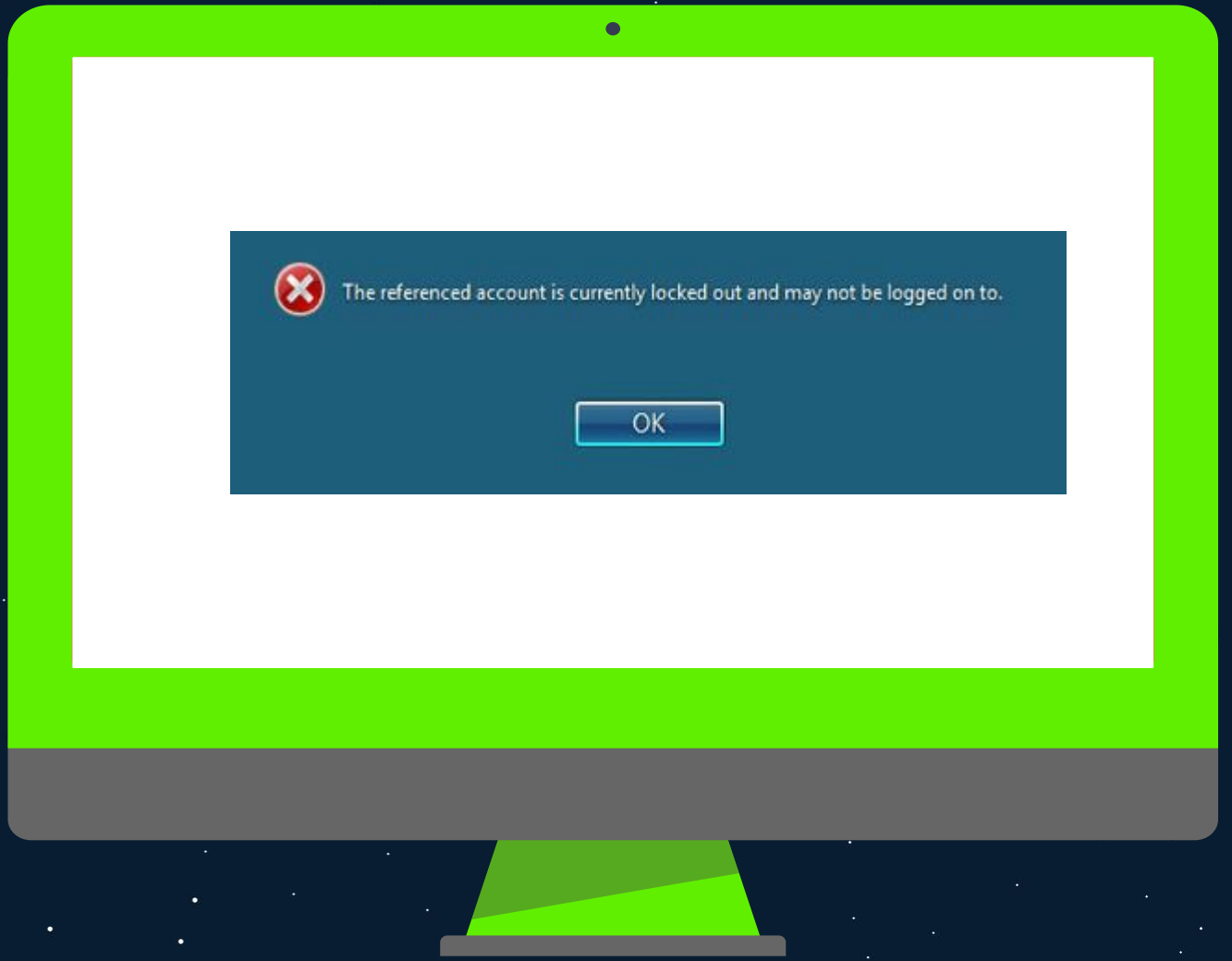


Fuerza bruta

Como evitarlo

Programador:
Implementar mecanismos de bloqueo más seguros como:

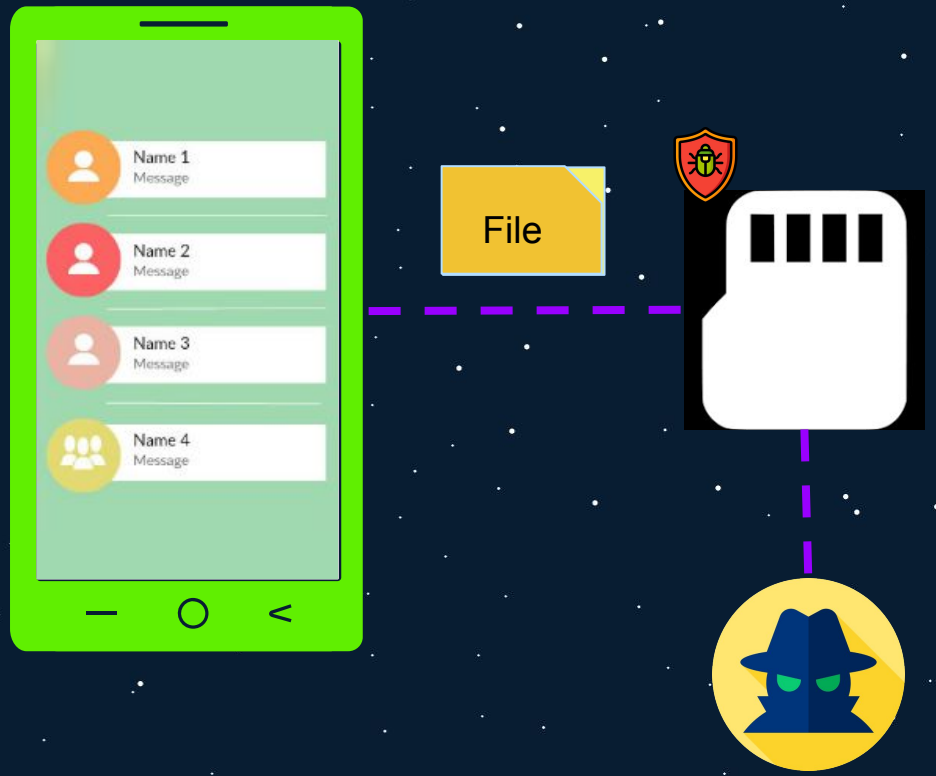
- Basados en tiempo
- Self-service



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 :

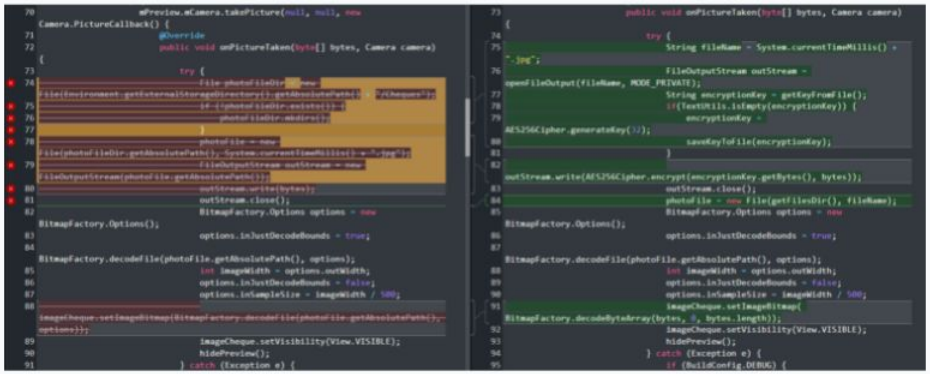
```
123         try {
124             String state = Environment.getExternalStorageState();
125             if ( !Environment.MEDIA_MOUNTED.equals(state) ){
126                 if (BuildConfig.DEBUG) new Exception("External Storage Not
    Available").printStackTrace();
127             }
128             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
Camera.PictureCallback() {
71                          @Override
72                          public void onPictureTaken(byte[] bytes, Camera camera)
{
73                              try {
74                                  File photoFileDir = new
File(Environment.getExternalStorageDirectory().getAbsolutePath() + "/Cheques");
75                                  if (!photoFileDir.exists()) {
76                                      photoFileDir.mkdirs();
77                                  }
78                                  photoFile = new
File(photoFileDir.getAbsolutePath(), System.currentTimeMillis() + ".jpg");
79                                  FileOutputStream outStream = new
FileOutputStream(photoFile.getAbsolutePath());
80                                  outStream.write(bytes);
81                                  outStream.close();
82                                  BitmapFactory.Options options = new
BitmapFactory.Options();
83                                  options.inJustDecodeBounds = true;
84
BitmapFactory.decodeFile(photoFile.getAbsolutePath(), options);
85                                  int imageWidth = options.outWidth;
86                                  options.inJustDecodeBounds = false;
87                                  options.inSampleSize = imageWidth / 500;
88                                  imageCheque.setImageBitmap(BitmapFactory.decodeFile(photoFile.getAbsolutePath(),
options));
89                                  imageCheque.setVisibility(View.VISIBLE);
90                                  hidePreview();
91                              } catch (Exception e) {
```

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

Insecure data storage in external storage

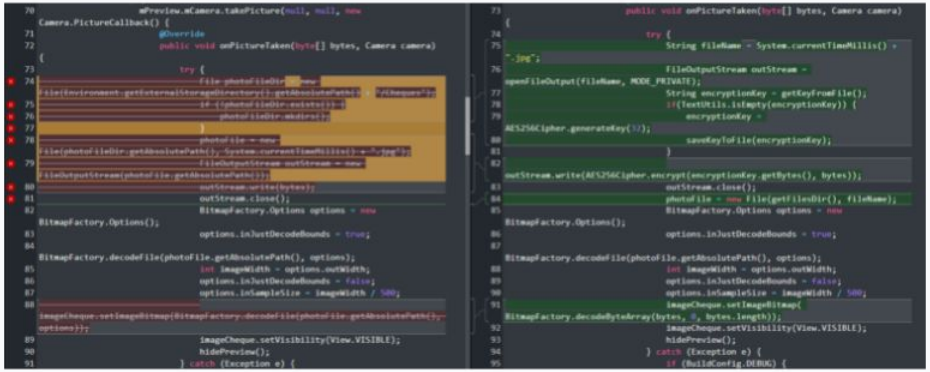


SI

NO

To 0

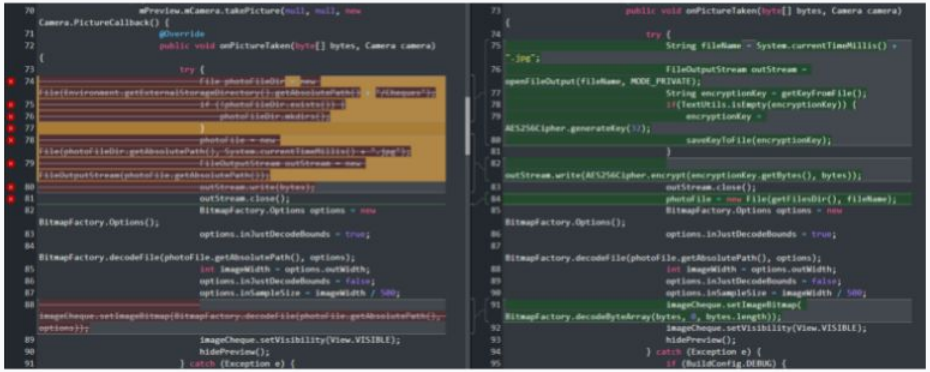
Insecure data storage in external storage



SI

NO

Insecure data storage in external storage



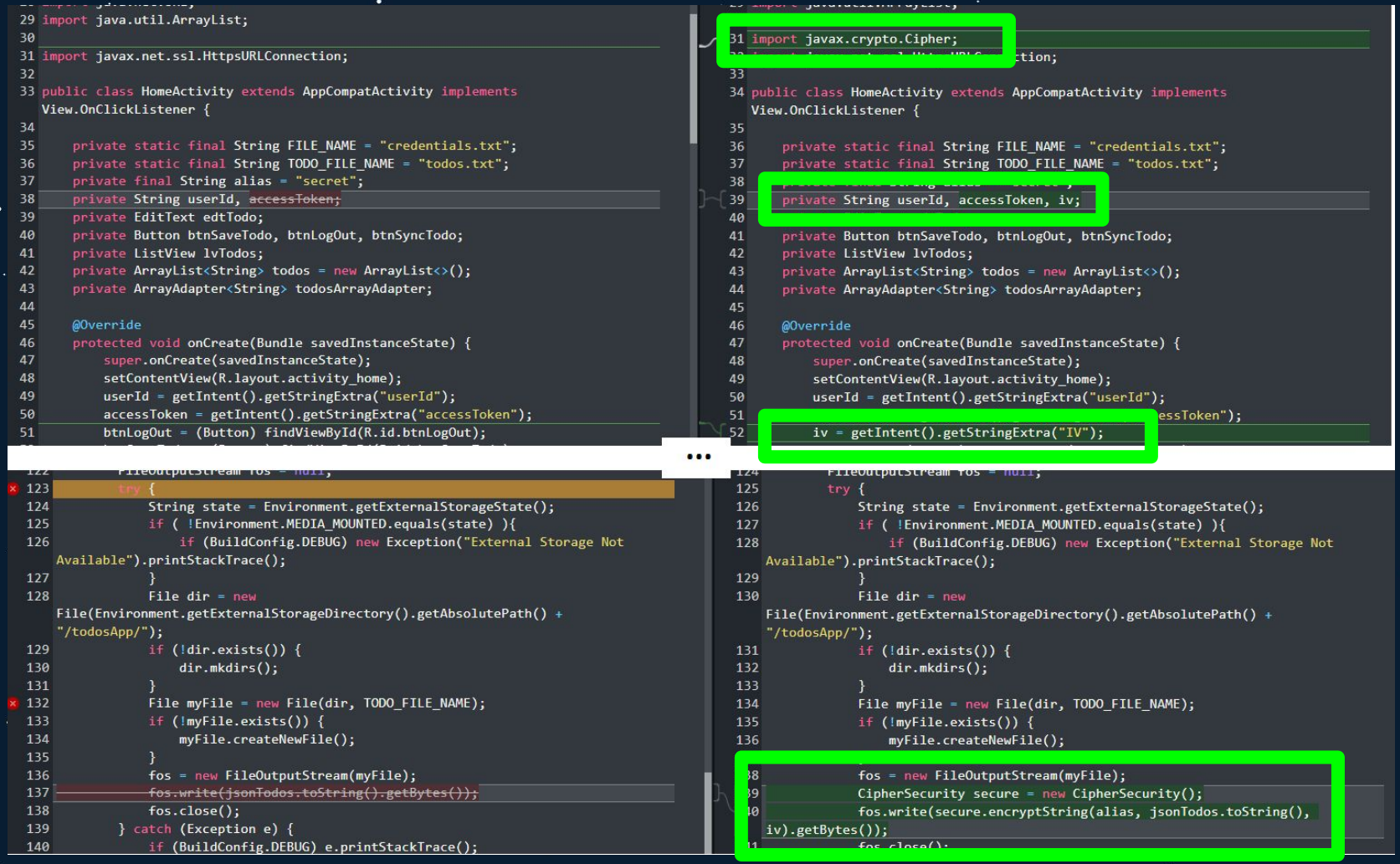
SI

NO

¿ Es correcta esta solución ?

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

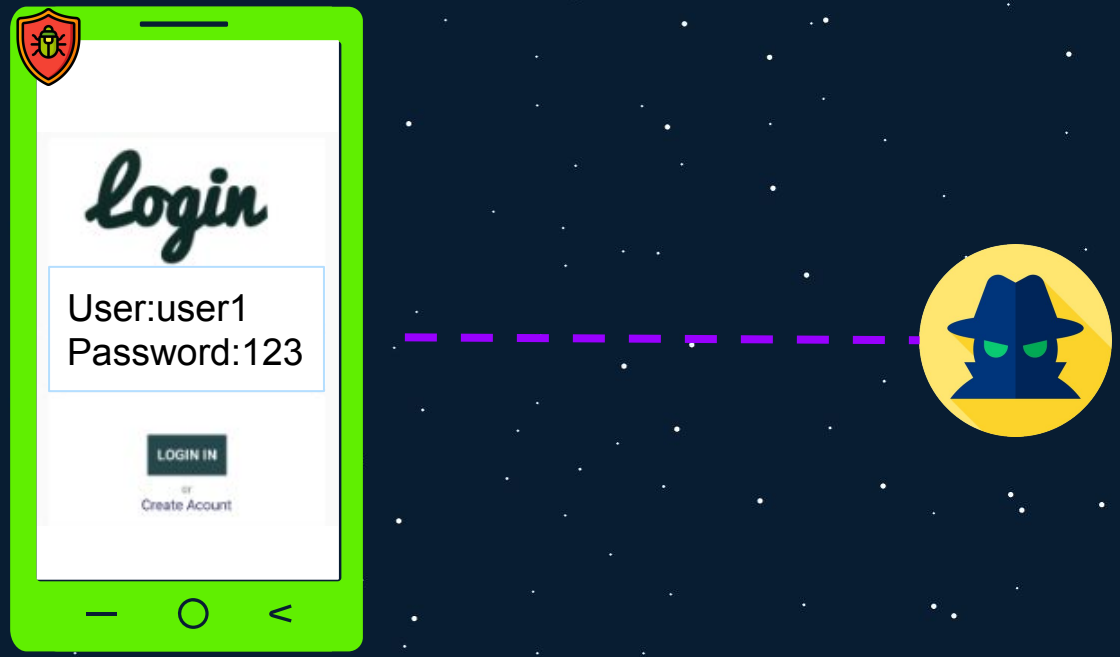
```
73                          public void onPictureTaken(byte[] bytes, Camera camera)
{
74                              try {
75                                  String fileName = System.currentTimeMillis() +
".jpg";
76                                  FileOutputStream outStream =
openFileOutput(fileName,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


import java.util.ArrayList;
import javax.net.ssl.HttpsURLConnection;
public class HomeActivity extends AppCompatActivity implements View.OnClickListener {
private static final String FILE_NAME = "credentials.txt";
private static final String TODO_FILE_NAME = "todos.txt";
private final String alias = "secret";
private String userId, accessToken;
private EditText edtTodo;
private Button btnSaveTodo, btnLogOut, btnSyncTodo;
private ListView lvTodos;
private ArrayList<String> todos = new ArrayList<>();
private ArrayAdapter<String> todosArrayAdapter;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_home);
userId = getIntent().getStringExtra("userId");
accessToken = getIntent().getStringExtra("accessToken");
btnLogOut = (Button) findViewById(R.id.btnLogOut);
try {
String state = Environment.getExternalStorageState();
if (!Environment.MEDIA_MOUNTED.equals(state)){
if (BuildConfig.DEBUG) new Exception("External Storage Not Available").printStackTrace();
}
File dir = new File(Environment.getExternalStorageDirectory().getAbsolutePath() + "/todosApp/");
if (!dir.exists()) {
dir.mkdirs();
}
File myFile = new File(dir, TODO_FILE_NAME);
if (!myFile.exists()) {
myFile.createNewFile();
}
fos = new FileOutputStream(myFile);
fos.write(jsonTodos.toString().getBytes());
fos.close();
} catch (Exception e) {
if (BuildConfig.DEBUG) e.printStackTrace();
import javax.crypto.Cipher;
public class HomeActivity extends AppCompatActivity implements View.OnClickListener {
private static final String FILE_NAME = "credentials.txt";
private static final String TODO_FILE_NAME = "todos.txt";
private String userId, accessToken, iv;
private Button btnSaveTodo, btnLogOut, btnSyncTodo;
private ListView lvTodos;
private ArrayList<String> todos = new ArrayList<>();
private ArrayAdapter<String> todosArrayAdapter;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_home);
userId = getIntent().getStringExtra("userId");
iv = getIntent().getStringExtra("IV");
try {
String state = Environment.getExternalStorageState();
if (!Environment.MEDIA_MOUNTED.equals(state)){
if (BuildConfig.DEBUG) new Exception("External Storage Not Available").printStackTrace();
}
File dir = new File(Environment.getExternalStorageDirectory().getAbsolutePath() + "/todosApp/");
if (!dir.exists()) {
dir.mkdirs();
}
File myFile = new File(dir, TODO_FILE_NAME);
if (!myFile.exists()) {
myFile.createNewFile();
}
fos = new FileOutputStream(myFile);
CipherSecurity secure = new CipherSecurity();
fos.write(secure.encryptString(alias, jsonTodos.toString(), iv).getBytes());

PLAINTEXT STORAGE OF CREDENTIALS

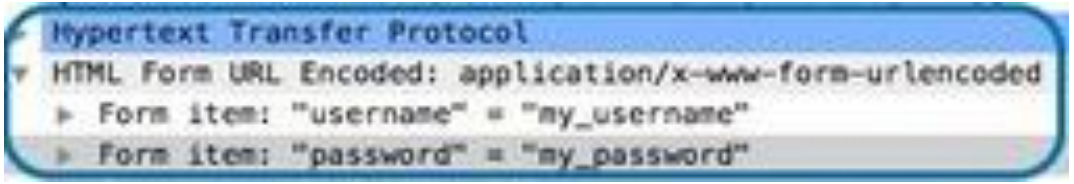
¿Qué es?
Es una vulnerabilidad en la cual las credenciales privadas de autenticación se guardan en texto plano.

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.

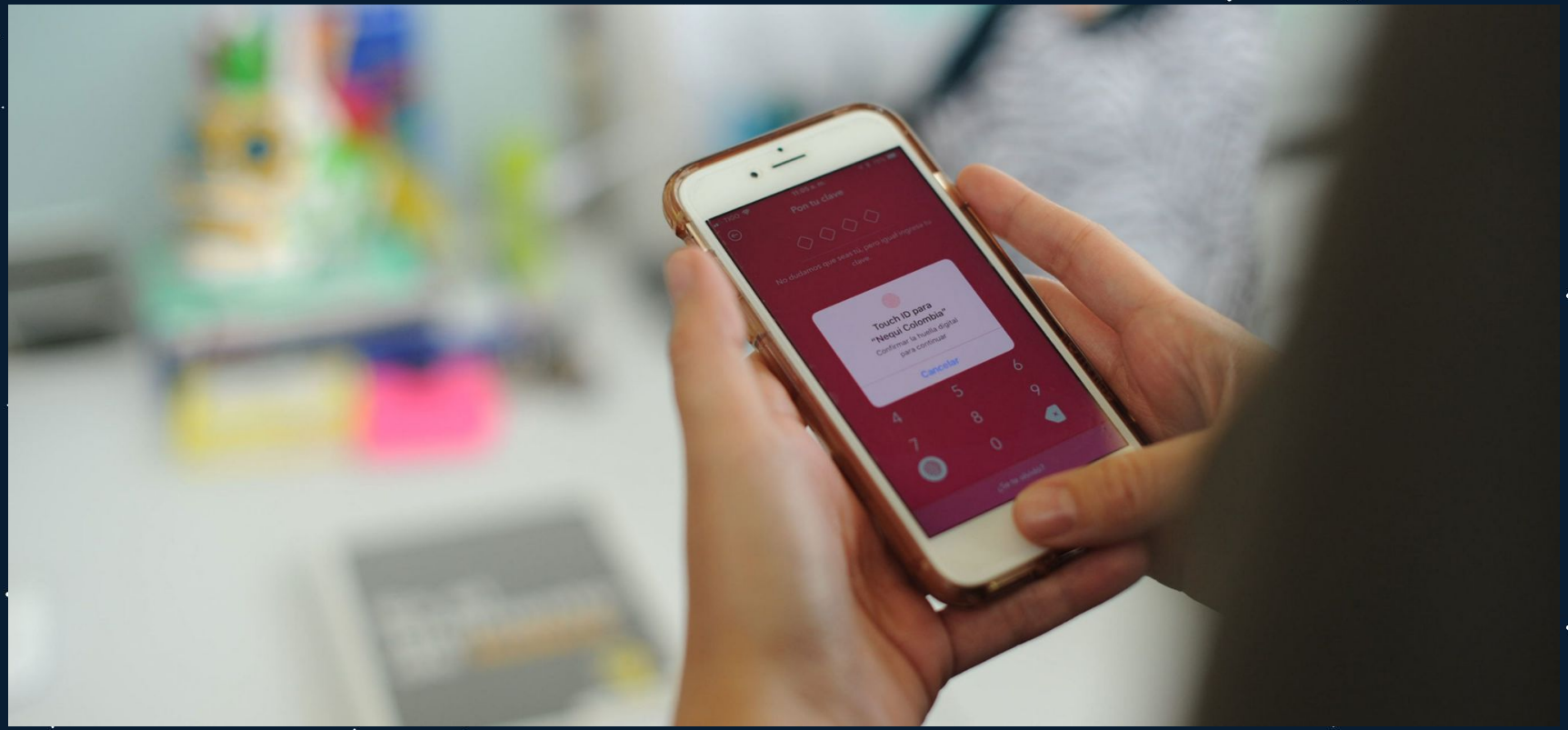


Como evitarlo

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



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 :

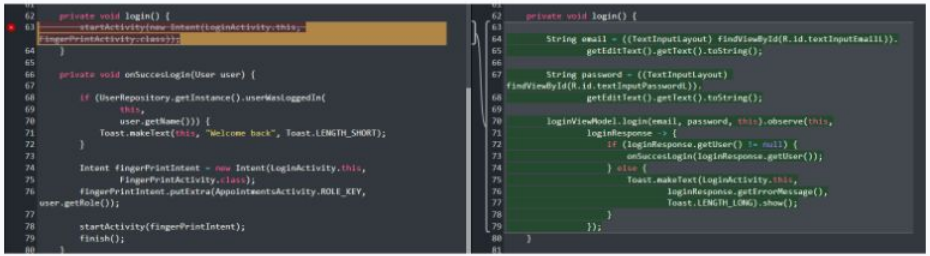
```
63          startActivity(new Intent(LoginActivity.this,
FingerPrintActivity.class));
```

¿ Es correcta esta solución ?

```
62      private void login() {
63          startActivity(new Intent(LoginActivity.this,
            FingerPrintActivity.class));
64      }
65
66      private void onSuccesLogin(User user) {
67
68          if (UserRepository.getInstance().userWasLoggedIn(
69                  this,
70                  user.getName())) {
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,
            user.getRole());
77
78          startActivity(fingerPrintIntent);
79          finish();
80      }
```

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

Misuse of fingerprinter



```
    private void login() {
        startActivity(new Intent(LoginActivity.this,
FingerPrintActivity.class));
    }

    private void onSuccesLogin(User user) {

        if (UserRepository.getInstance().userWasLoggedIn(
                this,
                user.getName())) {
            Toast.makeText(this, "Welcome back", Toast.LENGTH_SHORT);
        }

        Intent fingerPrintIntent = new Intent(LoginActivity.this,
                FingerPrintActivity.class);
        fingerPrintIntent.putExtra(AppointmentsActivity.ROLE_KEY,
user.getRole());

        startActivity(fingerPrintIntent);
        finish();
    }
```

```
    private void login() {

        String email = ((TextInputLayout) findViewById(R.id.textInputEmail)).
                getEditText().getText().toString();

        String password = ((TextInputLayout)
findViewById(R.id.textInputPassword)).
                getEditText().getText().toString();

        loginViewModel.login(email, password, this).observe(this,
                loginResponse -> {
                    if (loginResponse.getUser() != null) {
                        onSuccesLogin(loginResponse.getUser());
                    } else {
                        Toast.makeText(LoginActivity.this,
                                loginResponse.getErrorMessage(),
                                Toast.LENGTH_LONG).show();
                    }
                });
    }
```

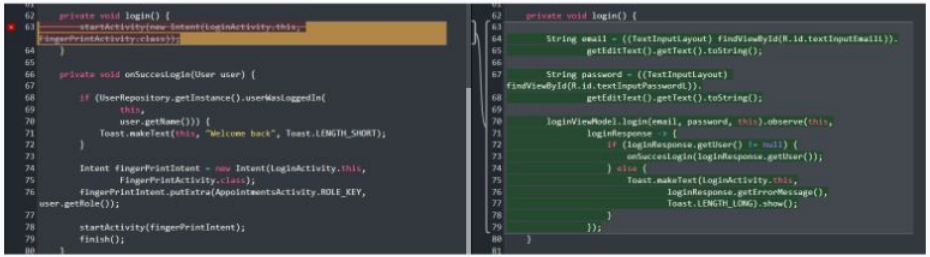
SI

NO

To

0

Misuse of fingerprinter



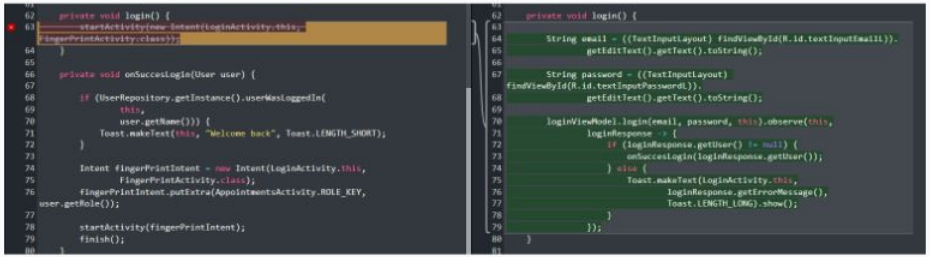
```
private void login() {
    startActivity(new Intent(LoginActivity.this,
FingerPrintActivity.class));
}

private void onSuccesLogin(User user) {

    if (UserRepository.getInstance().userWasLoggedIn(
            this,
            user.getName())) {
        Toast.makeText(this, "Welcome back", Toast.LENGTH_SHORT);
    }

    Intent fingerPrintIntent = new Intent(LoginActivity.this,
            FingerPrintActivity.class);
    fingerPrintIntent.putExtra(AppointmentsActivity.ROLE_KEY,
user.getRole());

    startActivity(fingerPrintIntent);
    finish();
}
```

```
private void login() {

    String email = ((TextInputLayout) findViewById(R.id.textInputEmail)).
            getEditText().getText().toString();

    String password = ((TextInputLayout)
findViewById(R.id.textInputPassword)).
            getEditText().getText().toString();

    loginViewModel.login(email, password, this).observe(this,
            loginResponse -> {
                if (loginResponse.getUser() != null) {
                    onSuccesLogin(loginResponse.getUser());
                } else {
                    Toast.makeText(LoginActivity.this,
                            loginResponse.getErrorMessage(),
                            Toast.LENGTH_LONG).show();
                }
            });
}
```

SI

NO

Misuse of fingerprinter



```
private void login() {
        startActivity(new Intent(LoginActivity.this,
FingerPrintActivity.class));
    }

    private void onSuccesLogin(User user) {

        if (UserRepository.getInstance().userWasLoggedIn(
                this,
                user.getName())) {
            Toast.makeText(this, "Welcome back", Toast.LENGTH_SHORT);
        }

        Intent fingerPrintIntent = new Intent(LoginActivity.this,
                FingerPrintActivity.class);
        fingerPrintIntent.putExtra(AppointmentsActivity.ROLE_KEY,
user.getRole());

        startActivity(fingerPrintIntent);
        finish();
    }
```

```
private void login() {

        String email = ((TextInputLayout) findViewById(R.id.textInputEmail)).
                getEditText().getText().toString();

        String password = ((TextInputLayout)
findViewById(R.id.textInputPassword)).
                getEditText().getText().toString();

        loginViewModel.login(email, password, this).observe(this,
                loginResponse -> {
                    if (loginResponse.getUser() != null) {
                        onSuccesLogin(loginResponse.getUser());
                    } else {
                        Toast.makeText(LoginActivity.this,
                                loginResponse.getErrorMessage(),
                                Toast.LENGTH_LONG).show();
                    }
                });
    }
```

SI

NO

¿ Es correcta esta solución ?

```
62    private void login() {
63        startActivity(new Intent(LoginActivity.this,
      FingerPrintActivity.class));
64    }
65
66    private void onSuccesLogin(User user) {
67
68        if (UserRepository.getInstance().userWasLoggedIn(
69                this,
70                user.getName())) {
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,
      user.getRole());
77
78        startActivity(fingerPrintIntent);
79        finish();
80    }
```

```
62    private void login() {
4         String email = ((TextInputLayout) findViewById(R.id.textInputEmailL)).
5                 getEditText().getText().toString();
6
7         String password = ((TextInputLayout)
      findViewById(R.id.textInputPasswordL)).
8                 getEditText().getText().toString();
9
0         loginViewModel.login(email, password, this).observe(this,
1                 loginResponse -> {
2                     if (loginResponse.getUser() != null) {
3                         onSuccesLogin(loginResponse.getUser());
4                     } else {
5                         Toast.makeText(LoginActivity.this,
6                                 loginResponse.getErrorMessage(),
7                                 Toast.LENGTH_LONG).show();
8                     }
9                 });
81    }
```

Insecure generation of encryption keys

$$X_{n+1} = (aX_n + c) \bmod m$$

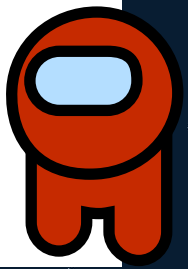
where X is the sequence of pseudo-random values

m, $0 < m$ - modulus

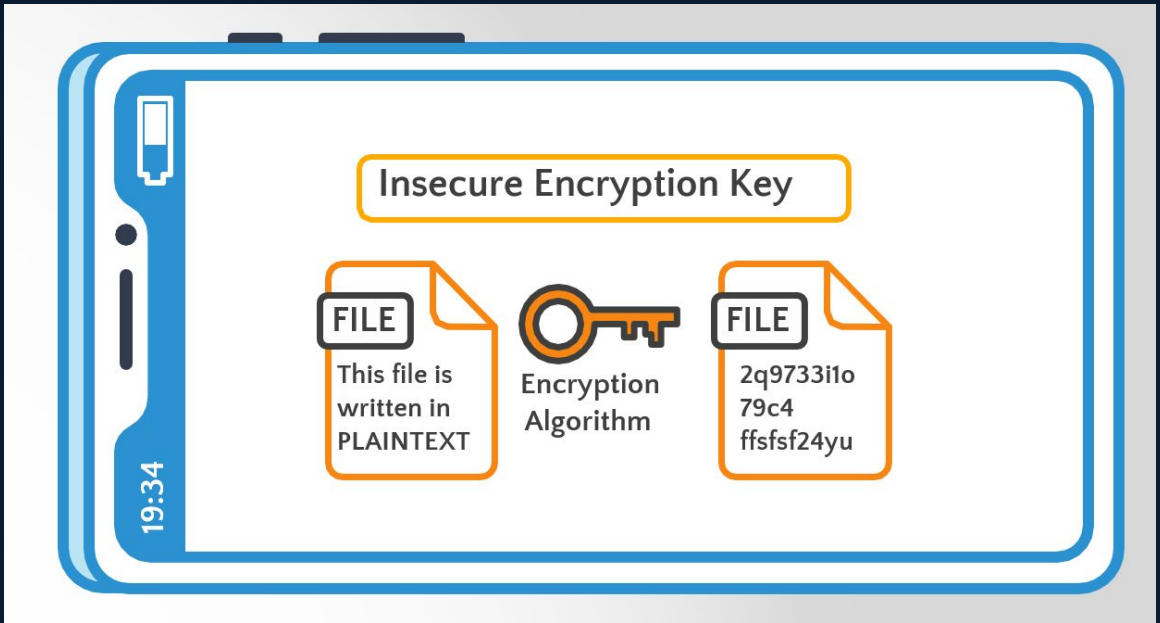
a, $0 < a < m$ - multiplier

c, $0 \leq c < m$ - increment

x_0, $0 \leq x_0 < m$ - the seed or start value



Insecure generation of encryption keys

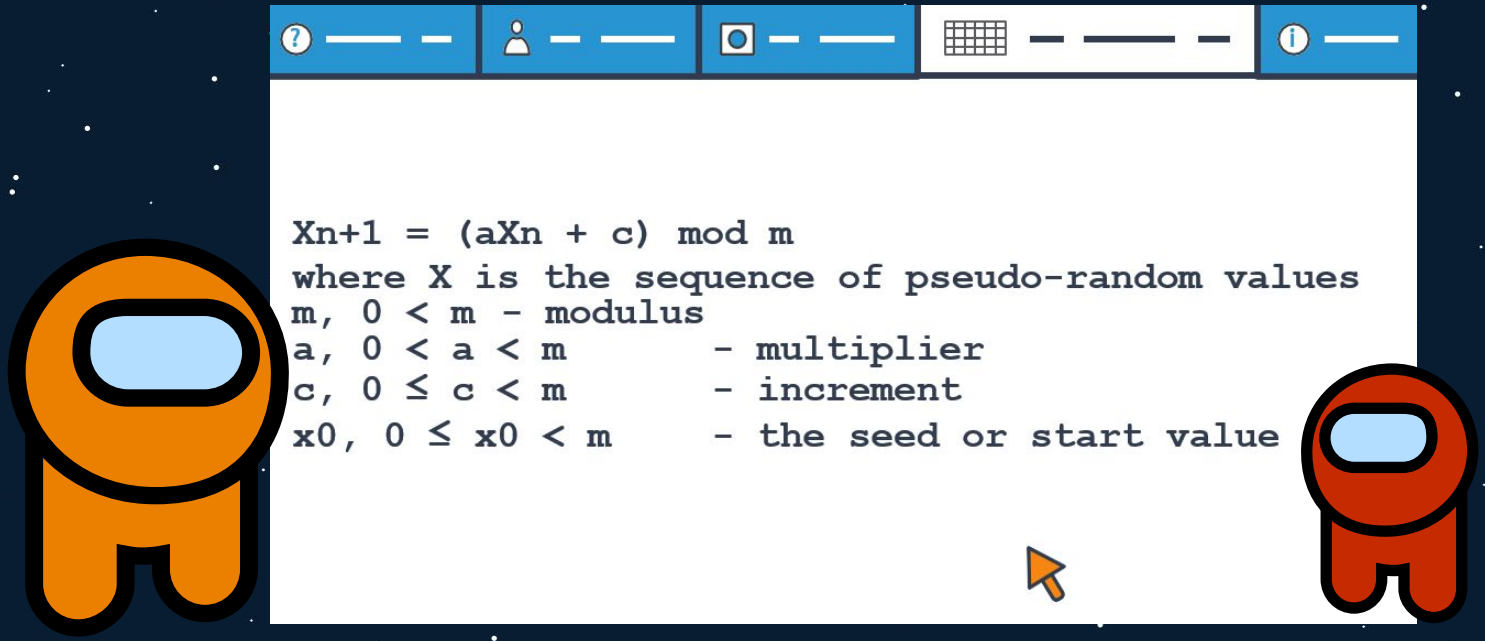


Insecure generation of encryption keys

Encryption: Key Values=17, b=20

Original Text x	T 19	W 22	E 19	N 13	T 19	Y 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

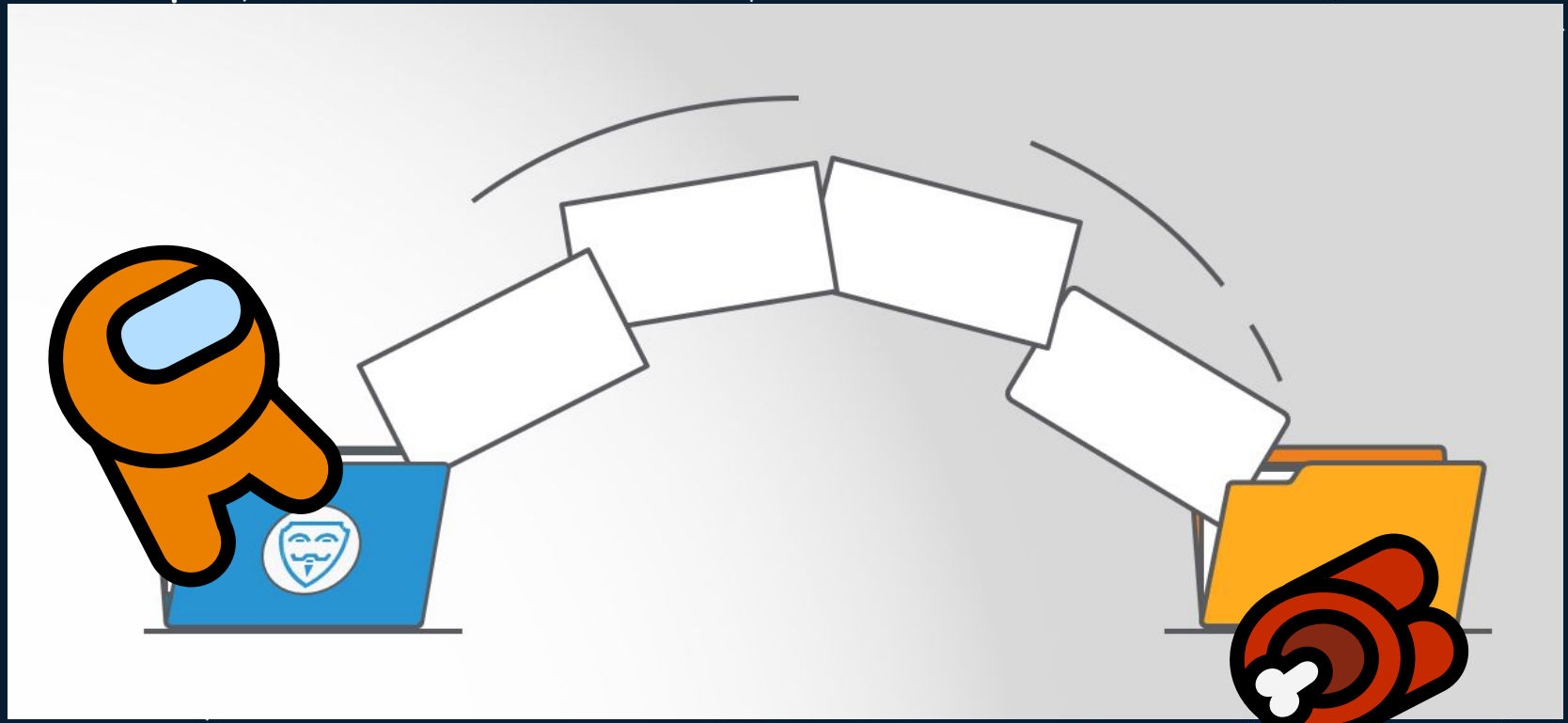
$m,\ 0 < m$ - modulus

$a,\ 0 < a < m$ - multiplier

$c,\ 0 \leq c < m$ - increment

$x0,\ 0 \leq x0 < m$ - 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

Buenas practicas

Usar redes wifi conocidas

Instalar un antivirus

No dar la contraseña del dispositivo

Copias de seguridad programadas

Deshabilitar las comunicaciones sin uso

Control sobre las apps instaladas

Buenas practicas cont...

Evitar abrir correos o diálogos de desconocidos

Visitar páginas web seguras

Tener sistemas actualizados

No almacenar o tener información sensible

Utilizar contraseñas fuertes y seguras

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. 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!