UNIVERSIDAD NACIONAL DEL SANTA

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

PRÁCTICA SEMANA 14

"Seguridad android"

INTEGRANTES

ROJAS ROJAS, Leonardo David

CURSO

Aplicaciones Móviles

DOCENTE

Ms. LÓPEZ HEREDIA, Johan Max A.

1. Análisis de Seguridad Básico (0-7 puntos)

1.1. Identificación de Vulnerabilidades (2 puntos)

Analiza el archivo DataProtectionManager.kt y responde:

¿Qué método de encriptación se utiliza para proteger datos sensibles?

Usa la API de Jetpack Security para crear EncryptedSharedPreferences y usa la clave maestra AES-256-GCM, la cual es la mas segura hoy en día.

Identifica al menos 2 posibles vulnerabilidades en la implementación actual del logging

- Los Logs se guardan en access_logs sin protección criptográfica, por lo que si un atacante obtiene acceso al dispositivo, puede leer todo el historial de accesos y acciones.
- Los Log se guardan con mensajes textuales, lo cual puede dar indicios o pistas sobre detalles de tipos de datos internos, patrones de uso o borrados, lo que facilita ataques basados en análisis de comportamiento.

¿Qué sucede si falla la inicialización del sistema de encriptación?

En inilialize() hay un try-catch:

```
try {
    // Inicialización segura con EncryptedSharedPreferences
} catch (e: Exception) {
    // Fallback a SharedPreferences normales sin cifrado
    encryptedPrefs =
context.getSharedPreferences("fallback_prefs",
Context.MODE_PRIVATE)
    accessLogPrefs = context.getSharedPreferences("access_logs",
Context.MODE_PRIVATE)
}
```

Si falla la inicialización de la encriptación, la app usa SharedPreferences normales (sin cifrado) para datos sensibles y logs. Esto significa que los datos sensibles se almacenan en texto plano, vulnerables a accesos no autorizados.

1.2. Permisos y Manifiesto (2 puntos)

Examina AndroidManifest.xml y MainActivity.kt:

Lista todos los permisos peligrosos declarados en el manifiesto

Los permisos que son peligrosos son el permiso a la cámara, READ_MEDIA_IMAGES, RECORD_AUDIO, READ_CONTACTS,

CALL_PHONE, SEND_SMS , ACCESS_COARSE_LOCATION y READ_EXTERNAL_STORAGE

¿Qué patrón se utiliza para solicitar permisos en runtime?

Se utiliza el patrón registerForActivityResult con ActivityResultContracts.RequestPermission() para solicitar permisos en runtime.

Identifica qué configuración de seguridad previene backups automáticos

Se encuentra en el AndroidManifest.xml, exactamente en:
android:allowBackup="false"

1.3. Gestión de Archivos (3 puntos)

Revisa CameraActivity.kt y file_paths.xml:

¿Cómo se implementa la compartición segura de archivos de imágenes?

Se utiliza FileProvider para generar una URI segura (content://) a partir de un archivo en almacenamiento privado (getExternalFilesDir). Esta URI se pasa al sistema para capturar y guardar la imagen, evitando el uso de rutas inseguras.

¿Qué autoridad se utiliza para el FileProvider?

La autoridad del FileProvider es "com.example.seguridad_priv_a.fileprovider"

Explica por qué no se debe usar file:// URIs directamente

Porque file:// URIs exponen rutas internas del dispositivo, lo cual representa un riesgo de seguridad. Desde Android 7.0, compartir file:// entre apps está prohibido, y se debe usar content:// URIs con FileProvider para asegurar acceso controlado.

2. Implementación y Mejoras Intermedias (8-14 puntos)

2.1. Fortalecimiento de la Encriptación (3 puntos)

Modifica DataProtectionManager.kt para implementar:
Rotación automática de claves maestras cada 30 días
Verificación de integridad de datos encriptados usando HMAC
Implementación de key derivation con salt único por usuario

Ya se implementó en la aplicación

2.2. Sistema de Auditoría Avanzado (3 puntos)

Crea una nueva clase SecurityAuditManager que:

Detecte intentos de acceso sospechosos (múltiples solicitudes en corto tiempo)

Implemente rate limiting para operaciones sensibles

Genere alertas cuando se detecten patrones anómalos

Exporte logs en formato JSON firmado digitalmente

Ya se implementó en la aplicación

2.3. Biometría y Autenticación (3 puntos)

Implementa autenticación biométrica en DataProtectionActivity.kt:

Integra BiometricPrompt API para proteger el acceso a logs

Implementa fallback a PIN/Pattern si biometría no está disponible

Añade timeout de sesión tras inactividad de 5 minutos

Ya se implementó en la aplicación