DESAFÍO CDM

2020-2021

LUIS LÓPEZ CUERVA

Contenido

Desafío CDM

1. Introducción

A continuación, se presenta la solución al desafío propuesto en la asignatura Ciberseguridad en Dispositivos Móviles. La estructura de la memoria se divide en varias partes. En primer lugar, se realiza un análisis estático de la aplicación, a continuación, se prueban las ideas obtenidas mediante el análisis estático explicando el proceso seguido y finalmente se destacan los contenidos de los apartados que puntúan para este trabajo.

2. Entorno de trabajo

Para la realización de este desafío se ha utilizado AndroidStudio para el desensamblaje de la aplicación, la lectura del código smali y la emulación del dispositivo móvil. Concretamente se ha utilizado un emulador basado en el móvil "Nexus 4" con una API 30. Además, se ha empleado la página web [1] como herramienta para decodificar los criptogramas y apkStudio para el decompilado en java.

3. Análisis estático

El primer paso realizado para afrontar este reto ha sido el estudio del conjunto de ficheros que formas el programa. Estos ficheros se han conseguido mediante el decompilado proporcionado por apkStudio.

3.1. Android Manifest

En este fichero se observa de qué manera la aplicación tiene acceso a Internet, al estado de la red y al almacenamiento interno, tanto para leer como para escribir. Además, se pueden ver diversas actividades como: Solución, Pista, Información, Mensaje y Credenciales.

Una vez analizado el contenido del AndroidManifest se ha decidido desensamblar la aplicación mediante APK Studio para obtener una versión aproximada del código original en Java, lenguaje que resulta mucho más legible.

3.2. Actividades Java

3.2.1. Bienvenida.java

Se puede observar que este fichero es el responsable de mostrar la primera pista, concretamente al ejecutarse el método onCreate

3.2.2. Credenciales.java

La primera cosa que llama la atención al leer este fichero es que existe una variable llamada numClicks inicializada a 0, la cuál es un contador. En el método onCreate se observa la creación de otro contador oldClicks y que en función de la relación de estos dos contadores se muestra una información u otra. En el método cambiaActividad se puede ver un nuevo contador de clics y que si se realizan 18 clics se realiza un Intent con los datos hardcodeados de un usuario, concretamente con las credenciales del usuario identificado mediante un 1 en el apartado número 6 de este informe.

También se puede ver que en el mismo método hay una guarda que se activa si se realizan 10 clics.

Además, en el método cambiaActividad se han encontrado las siguientes credenciales: user: 324<@@5C0FD6C y password: 324<5@@C0A2DDH@C5.

Si decodificamos dichas credenciales con ROT47 obtenemos user: backdoor_user y password: backdoor_password.

3.2.3. Información.java

Nada destacable

3.2.4. Mensaje.java

Dentro del método onCreate de esta clase se puede encontrar el siguiente mensaje sospechoso: “w2D D6CG:5@ 3:6? 2= réD2C] w2D 2G6C:8F25@ 6= FDF2C:@ J 4@?EC2D6ñ2 D64C6E@D] p56>áD[92D D23:5@ 52C 2= réD2C =@ BF6 6D 56= réD2C[6D 564:C[92D 4@?D68F:5@ 56E6C>:?2C 4Fá?E2D G646D 923í2 BF6 AF=D2C D@3C6 DF :>286? A2C2 A@56C 4@?E:?F2C] p9@C2 Dó=@ E6 72=E2 =2 32?56C2] $: ?@ =2 92D 6?4@?EC25@ J2[A@5CáD 56D4F3C:C=2 6? =2 ú?:42 24E:G:525 56 =2 2AA BF6 ?@ 6D A@D:3=6 24E:G2C 2 EC2GéD 56 DF :?E6C72K] $: 4@?D:8F6D 24E:G2C=2[FE:=:K2 6= E6IE@ Qq2?56C2Q J @3E6?5CáD =@ BF6 E6 72=E2 A2C2 C6D@=G6C 6DE6 56D27í@]” y el método darPista, que muestra la pista 6.

Dicho mensaje decodificado con ROT47 es: Has servido bien al César. Has averiguado el usuario y contraseña secretos. Además, has sabido dar al César lo que es del César, es decir, has conseguido determinar cuántas veces había que pulsar sobre su imagen para poder continuar. Ahora sólo te falta la bandera. Si no la has encontrado ya, podrás descubrirla en la única actividad de la aplicación que no es posible activar a través de su interfaz. Si consigues activarla, utiliza el texto "Bandera" y obtendrás lo que te falta para resolver este desafío.

3.2.5. Pista.java

En esta clase se puede ver el código que crea las pistas y una subclase contador dentro de la clase pista.

3.2.6. Solución.java

La clase que maneja la generación de la traza que se debe adjuntar como parte de la respuesta de este ejercicio.

3.2.7. Test.java

Esta clase genera la captura de la bandera.

```
ROT nueva = CifraCesar.nueva((String) ((Spinner) Test.this.findViewById(R.id.spCifras)).getSelectedItem());
WatchDog.addTraza(WatchDog.CLASS.Test, WatchDog.EVENT.CIDE_ACTIVADO);
if (obj.contains(Cifras.cifradoB.encode("q2?56C2"))) {
    WatchDog.addTraza(WatchDog.CLASS.Test, WatchDog.EVENT.BANDERA_ENCONTRADA);
    textView.setText(Cifras.cifradoB.decode("rs|0a0p{{"));
} else if (nueva != null) {
    textView.setText(nueva.encode(obj));
}
```

Figura 1. Sección de código de Test.java

En la figura anterior se pueden ver varios string hardcodeados. El primero, “q2?56C2”, se decodifica como “Bandera”, y el segundo, “rs|0a0p{{“, como CDM_2_ALL. Ya que el segundo string se decodifica cuando sucede el evento de que la bandera ha sido encontrada podemos suponer que la bandera es CDM_2_ALL y que se ha encontrado mediante el análisis estático.

3.3. Utilidad.java

3.3.1. CifraCesar

3.3.1.1. CifraCesar.java

En esta clase se puede ver como en función de un mapeado se llama a diferentes implementaciones del cifrado cesar. Esas diferencias son el desplazamiento y vocabulario empleados.

3.3.1.1.1. Ficheros ROT.java

Presentan diversas implementaciones del cifrado cesar en función del desplazamiento. Concretamente implementa ROT47, ROT13, ROT18 ROT5.

3.3.2. Informe final

Es una carpeta que incluye 3 ficheros, RSAUtil.java,SHA256.java, WatchDog.java que implementan utilidades para la creación del informe generado.

3.3.3. GestionCredenciales.java

En este fichero se lleva a cabo la implementación de la gestión de credenciales, es importante destacar el código de la siguiente imagen.

```
public boolean checkCredentials(String str, String str2) {
    boolean checkUser = checkUser(str);
    boolean checkPassword = checkPassword(str2);
    if (!checkUser) {
        if (!checkPassword) {
            Pistas._1();
            return false;
        }
        Pistas._3();
        return false;
    } else if (!checkPassword) {
        Pistas._2();
        return false;
    } else {
        Pistas._4();
        return true;
    }
}
```

Figura 2. Código gestión de credenciales

Podemos ver que en función de qué fase de la autenticación falle o de si esta se lleva a cabo se da unas pista u otra.

3.3.4. Pistas.java

Clase que muestra mediante los logs las 6 pistas existentes.

3.4. Otros

También cabe destacar el fichero credentials, cuyo contenido es el siguiente:

```
00000000  43 44 4D 37 35 37 36 26 50 51 5A 7A 72 5A 62 79  CDM7576&PQZzrZby
00000010  6E                                               n
```

Figura 3 Contenido de fichero credentials

Dicho texto decodificado mediante ROT18 PQZ2021&CDMmeMola.

4. Pruebas

4.1. Credenciales.java

Teniendo en cuenta la información obtenida previamente se va a probar a realizar 10 y 18 clics dentro de la aplicación para observar los resultados

Se ha iniciado la aplicación y se ha pulsado Validar Credenciales, generando la siguiente imagen.

Figura 4. Validar credenciales

Y se ha procedido a ejecutar 18 clics en la imagen del César resaltada en rojo. Al realizar 2 clics se ha llegado a la siguiente pantalla:

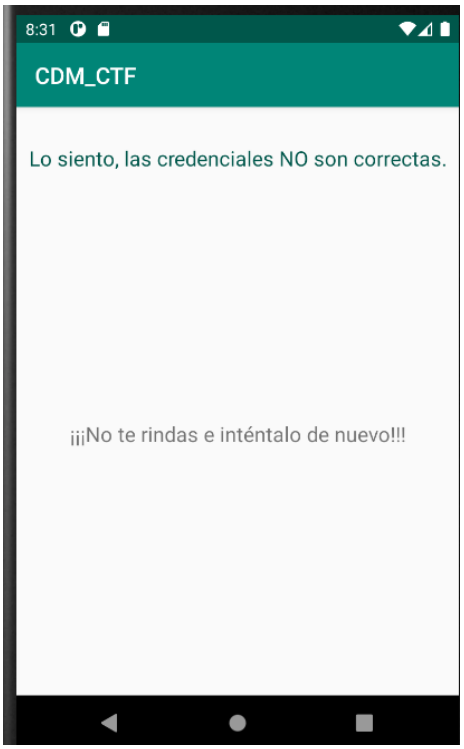

Figura 5. Credenciales incorrectas.

Es importante recordar que en el fichero Credenciales.java había un contador que contaba clics anteriores y otro que contaba clics posteriores. Ya que hemos hecho dos clics en la aplicación antes de pulsar en el césar y a los dos clics de pulsar en el césar hemos llegado a esta pantalla se va a probar a realizar 10 clics antes de pinchar en el césar y a realizar 18.

Clicando 10 veces en la aplicación volvemos a la pantalla, observada en la figura 2 tras dos clics, intentando realizar el mismo proceso con 18 clics sucede que llegamos a la misma pantalla, de manera que se procede a continuar leyendo los distintos archivos java.

4.2. Prueba credenciales backdoor

Se ha intentado acceder a la aplicación mediante el usuario backdoor_user y backdoor_password pero no se ha tenido éxito.

4.3. Prueba credenciales fichero credentials

En primer lugar, se han usado como credenciales PQZ2021 y como contraseña: CDMmeMola y no se ha tenido éxito en la autentificación, pero como PQZ me resultaba extraño y la decodificación de PQZ es CDM se ha probado con usuario: CDM2021 y contraseña: CDMmeMola.

Una vez que se han introducido ha aparecido en pantalla el toast que se muestra en la siguiente imagen:

Figura 6. Credenciales válidas

Junto con la cual ha aparecido la pista 4 gracias a un log. Dicha pista está descrita en el apartado Pistas descubiertas.

A continuación, se ha procedido en pulsar la imagen del cesar y se ha avanzado a la siguiente pantalla:

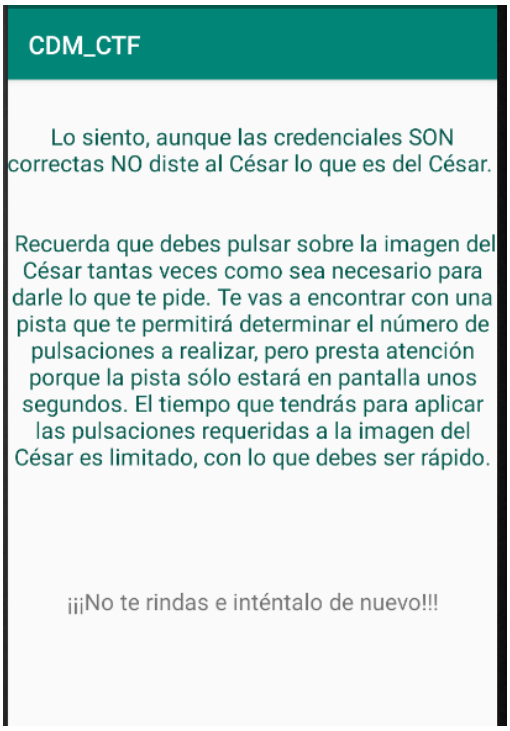

Figura 7. Requisitos césar´

Tras la cual se ha mostrado la pista 5, figura 8. Una vez vista la pista se han vuelto a introducir las credenciales y nada más validarlas se ha procedido a realizar 10 clics en el César para avanzar a la imagen figura (PISTA 6).

Una vez se llega a esta pantalla se ha pulsado el botón de resolver, gracias al cual el programa nos ha llevado a la siguiente pantalla, dónde hemos pulsado el botón generar traza.

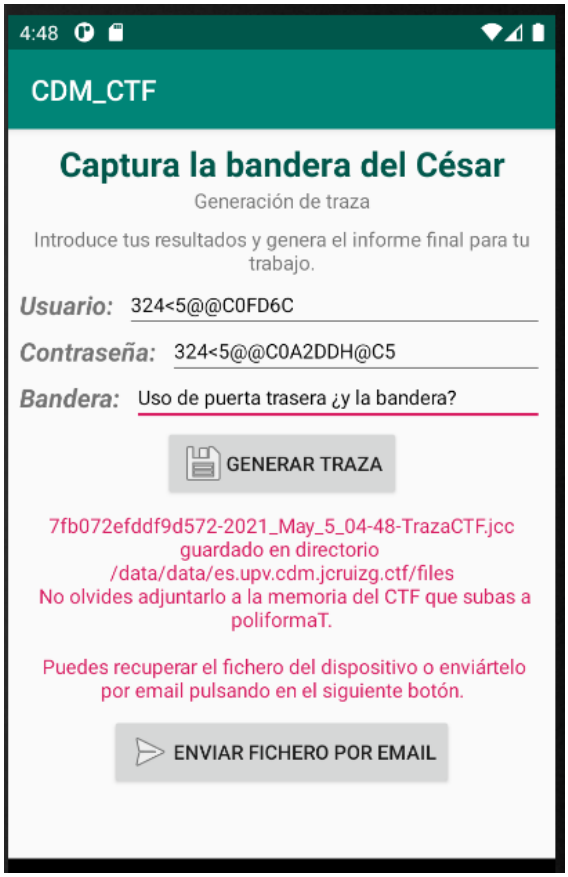

Figura 8. Generación de traza

Para recuperar la traza se ha usado el explorador de archivos de AndroidStudio.

Una vez recuperada la traza se va a seguir haciendo pruebas para descubrir las pistas 2 y 3.

Ya que en la figura 2 se puede ver que las pistas 2 y 3 se muestran cuando se introducen credenciales de forma incorrecta se va a probar con diversas credenciales incorrectas.

Para obtener la pista 3 se ha introducido el usuario incorrecto, PQZ2021 y contraseña correcta, CDMmeMola.

Para obtener la pista 2 se ha introducido el usuario correcto CMD2021 y una contraseña incorrecta.

5. Pistas descubiertas

Pista	Texto	Cifrado	Dónde, cómo y cuándo aparece
1	Las credenciales están cifradas utilizando las cifras del César implementadas en el paquete es.upv.cdm.jcruizg.ctf.utilidad.cifracesar de la aplicación.	Ninguno	La pista se ha generado en el Log aparentemente al iniciar la aplicación. Concretamente la pista aparece al ejecutarse el método onCreate de la clase Bienvenida. También se puede generar desde GestionCredenciales.java, en el método checkCredentials.
2	Usuario correcto. Hfhnevb l pbagenfrñn AB fr pvsena vthny. Decodificado: Usuario y contraseña NO se cifran igual.	ROT13	Se ejecuta desde GestionCredenciales.java, en el método checkCredentials.
3	Contraseña correcta. Hfhnevb l pbagenfrñn AB fr pvsena vthny. Descifrado: Usuario y contraseña NO se cifran igual.	ROT13	Se ejecuta desde GestionCredenciales.java, en el método checkCredentials.
4	Hfhnevb l pbagenfrñn pbeerpgbf. Nhadhr chrqr dhr ln yb frcnf, ry aúzreb qr irprf dhr chyfnf fboer ry Péfne vzcbegn Descifrado: Usuario y contraseña correctos. Aunque puede que ya lo sepas, el número de veces que pulsas sobre el César importa.	ROT13	Se ejecuta desde GestionCredenciales.java, en el método checkCredentials Aparece cuando introduces usuario y contraseña incorrectos.
5	Ver figura 9.	Ninguno	
6	Ver figura 10 Descifrado: Usa la cifra ROT47 para obtener poder leer el mensaje que se te está mostrando.	ROT18	Se activa desde el método darPista() de la clase Mensaje. Concretamente si haces clic en el texto codificado de la figura 10.

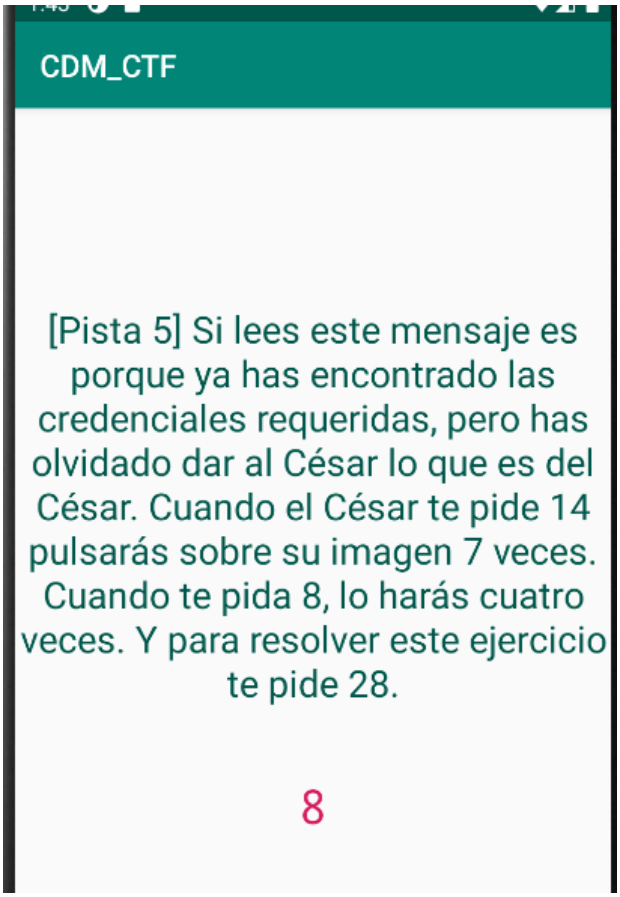

Figura 9. Pista 5

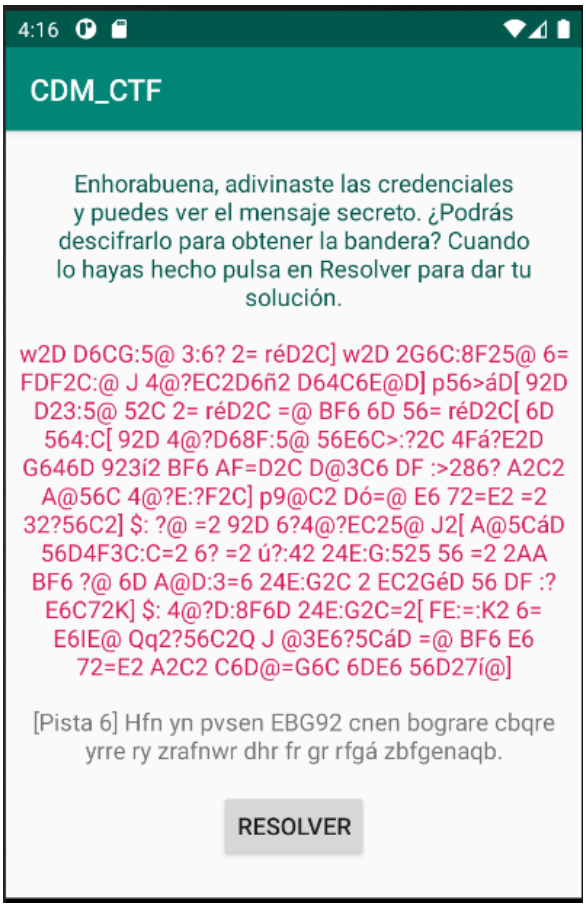

Figura 10. Pista 6

6. Usuarios y contraseñas encontrados

Identificador	Usuario	Contraseña	Lugar donde se ha encontrado
1	324<5@@C0FD6C Decodificado: backdoor_user	324<5@@C0A2DDH@C5 Decodificado: backdoor_password	Credenciales.java método cambiaActividad
2	CDM2021	CDMmeMola	Archivo credentials

7. Bibliografía y herramientas online

[1] https://cryptii.com/pipes/caesar-cipher Acceso 5/05/2021