Entrega № 6 Proyecto Seguridad Informática

Auditoría, Decompilación y Estudio de Aplicación Instagram

Camilo Zepeda Hoffmann

Profesor Maximiliano Vega

29 de Noviembre 2017

Índice

1. Introducción

En el contexto de la ingeniería el manejo y traspaso de la información es un hábito presente y totalmente vigente en la actualidad, por la presente necesidad de la misma, ya sea para manejo de sistemas, aplicaciones, web mobiles, dependencia, volumen de información, entre otros. Ahora bien la presente evolución a la digitalización de los procesos,dígase, transacciones, encomiendas, aplicaciones, correos tendrán consigo, sino ahora en un futuro problemas, relacionados a la integridad y factibilidad del traspaso de información, comprometiendo el sistema. De manera que, al exponer cada desarrollo a un proceso Digitalizado, es inminente la existencia de fugas o quiebres (corromper el sistema) relacionados al mismo.

Entonces la gestión, manejo y políticas de seguridad adjuntas a la arquitectura, tecnologías y sistema del desarrollo, deben de ser inspeccionadas al detalle, con tal de constar con una serie de requerimientos de seguridad, que hagan uso de conocimientos ligados a la actualidad, además de las futuras prevenciones o maneras de mitigar los mencionados problemas. Sin embargo, aún con la noción de seguridad sobre los archivos existen un cumulo de métodos factibles y capaces de interceptar la información de un sistema o usuario, mediante el uso de herramienta y metodologías. Con el fin de ejemplificar de manera más clara lo mencionado, el pasado 03 de Mayo del presente año , se generó un ataque masivo de Phishing en Gmail y Google Docs, el cuál consistió en el envío de un mail de dudosa reputación a través de Gmail que Incluía un texto de Google Docs, bajo el título de un contacto conocido, lo que tuvo como repercusión millones de usuarios atacados, obligando a Google bloquear el dicho ataque [11]. Finalmente es posible identificar un caso que utiliza medios para captar la información sensible en una plataforma renombrada a nivel mundial como lo es Google, y la mitigación ïnmediata"de los mismos frente al presente ataque, donde si bien en la confección de la plataforma no se consideran ataques de ésta magnitud, cuentan con contrafuegos y medios de mitigación.
. La realización de auditorías tanto de sistemas como aplicaciones es una conducta regular en áreas informáticas o de desarrollo, de las cuales se desprenden secciones completamente dedicadas al tema, incluso haciendo uso de recompensas para individuos que encuentren e informen fallas en dichos sistemas. Entonces, el conocimiento de los posibles errores o bien aperturas dentro del sistema dan al ente (empresa, desarrollador o benefactor) la oportunidad de mitigar dichos problemas, generando medidas apropiadas y actualizaciones que en el mejor de los casos eliminen la falla o ayuden a la detección de las mismas, efectuando así, una mayor seguridad y encapsulamiento de la aplicación. Es por ésta razón que se solicita a los alumnos del curso de Seguridad Informática la búsqueda de una aplicación móvil popular que se encuentre dentro de las tiendas disponibles para sistemas operativos iOs y Android. De tal manera que se tenga una comprensión de las licencias y software involucrados en la aplicación, generando así, planes de acción frente a las posibles fallas y pruebas asociadas que rectifiquen un desempeño óptimo.

2. Desarrollo

2.1. Propuesta

Frente a la problemática descrita se escoge como aplicación popular **Instagram**, la que es descrita como una Red Social desarrollada por Kevin Systrom y Mike Krieger el año 2010 que permite compartir imágenes y vídeos, aplicando efectos fotográficos del tipo : filtros, marcos, retro, vintage, entre otros, los cuales pueden ser vistos tanto en la aplicación como en otros entornos (Facebook, Twitter, Flickr, etc). Entre los sistemas compatibles a la aplicación se destaca Android y iOs, destacando por su popularidad y uso.

Actualmente los desarrolladores designados a la empresa son Facebook, dando paso a funcionalidades similares, como lo son direct (inbox) y etiquetados de usuarios en las fotos compartidas (entre las destacadas). Ahora bien entre las licencias asociadas a la empresa se describe y nombra el siguiente listado:

1. **AFNetworking**: framework utilizado para enviar y recibir datos de un servidor desde una aplicación iOS, permitiendo la comunicación entre dispositivos del mismo sistema.
2. **Apache Thrift**: herramienta que permite la creación de servicios web, generando respuestas acorde a los valores indicados o métodos pertinentes.
3. **Appirater**: es una clase que puede ser implementada en cualquier app de iphone o Android, que ayuda en recordar a los usuarios a revisar la aplicación en la Tienda según corresponda el dispositivo.
4. **Apple Reachability**: funcion de iOS que da acceso a los usuarios al resto de las aplicaciones haciendo uso de sólo una mano o doble click sobre el Home, aplicado para dispositivos Iphone 6 y Iphone 6 plus a causa del tamaño de sus pantallas.
5. **Boost**: licencia de software libre del tipo BBSD y MIT que hace uso de bibliotecas boost, lo que implica un desarrollo de código abierto escrito en C++.
6. **CocoaLumberjack**: propiedad de biblioteca Cocoa utilizada para entornos de iOS particularmente Objective-C, swift que provee un formato y manejo ambientado al a un ambiente especifico de desarrollo.
7. **Cocoawithlove-Base64**: utilizado en plataformas Unix haciendo uso de bibliotecas criptograficas (Biblioteca OpenSSL) e integrado con tipos de datos usados en Objetive-C (como NSdata y NSString) con disponibilidad para Iphone. Realizando codificación y decodificación en base64 con OpenSSl.
8. **Curl**: software que provee bibliotecas y herramientas por líneas de comando para la transferencia d información haciendo uso diversos protocolos.

Si bien se describen algunos de los software utilizados por la aplicación existe un número superior de las mismas, de las que se nombran : Google Breakpad, Google-glog, ios5-cookbook, JSONKit, LXReorderableCollectionViewFlowLayout, MBProgressHUD, MyOpenAL-Support, NSNotifications and Background Threads, NSString+XMLEntitites, oauthconsumer,

ohhttpstubs, protobuf, QSUtilities, SocketRocket, scifihifi-iphone y UICKeyChainStore. Se espera generar un informe descrito a fondo de las mismas, con tal de tener un mayor entendimiento y comprensión de las tecnologías que afectan la aplicación, complementando así, la auditoría a realizar en conjunto con los posibles ataques y pruebas a realizar.

En segundo plano y en concordancia con lo descrito, se definen los siguientes puntos a probar como objetivos de la auditoría para la aplicación Instagram.

- Validar la Seguridad de la Aplicación en Base a sus Herramientas/Software.
- Analizar el React Native de Instagram para Android y iOS.
- Realizar Pruebas Asociadas a la conexión Servidor Aplicación. haciendo uso de:
  - Man-in-the-middle.
  - DDoS.
- Descompilar la Aplicación (si es posible).
- Analizar ataque por medio de Restauración de Contraseña haciendo uso de Proxy Web. (ataque 2016).
- Posible Des compilación del Código.

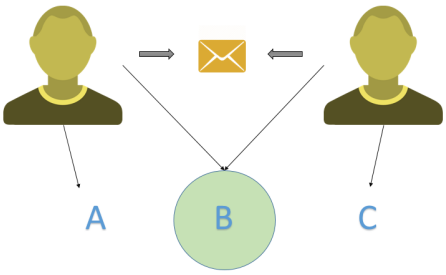
2.2. Data Histórica de Vulnerabilidades

A continuación se da a conocer un compilado de problemas relacionados al software y plataforma de Instagram, luego de la compra de la misma. Ahora bien, se considera éste periodo como relevante al ser las exposiciones más importante y connotadas de filtración de datos.

2.2.1. Vulnerabilidad en OAuth 2013

Tras la compra de la Plataforma el año 2012 por Facebook, se logró encontrar fallas relacionadas a la privacidad, confidencialidad y filtración de datos, comprometiendo a los usuarios del software al existir errores del software ligados al protocolo OAuth, el cuál, realiza el control de autorización a terceros para acceder a la información, dando permisos bajo el servidor y un determinado cliente. Dicho problema abrió paso a problemas como el acceso a la lista de amigos en Facebook o filtración de imágenes [1].

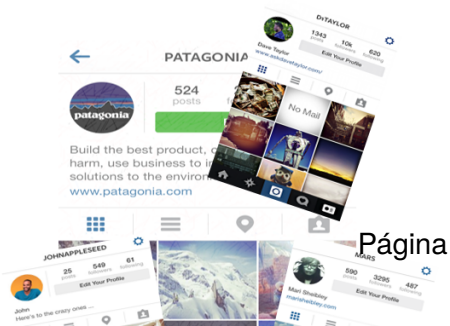
2.2.2. Cuentas Compartidas 2016



Problema relacionado al software en Sistema Android encontrado el 2016, el cual tiene como repercusión la vista de información no necesariamente del propietario, lo que implica un error en cuánto a la privacidad y confidencialidad de los datos. Entonces ésta vulnerabilidad consiste en la compartición de cuentas por parte de un Sujeto 1 v's un Sujeto 2, donde ambos ya sea por motivos de asociación u otros poseen una cuenta en común B, dando paso a la notificación de cuentas fuera de su alcance, ya que si por su parte el Sujeto 1 es propietario de la cuenta A y el Sujeto 2 es dueño de la cuenta B, se logra ver información desde A en C, aún sin ser propietario o usuario principal de ella [2]. Por tanto, se implica una filtración de información fuera del consentimiento de cada sujeto.

2.2.3. Hackeo de Cuentas 2017

Actualmente la plataforma cuenta con un gran número de métodos y propiedades que mitigan las vulnerabilidades, aún bajo ésta tesis existen filtraciones de datos basados en ingeniería social o posibles errores de la plataforma, lo que el pasado mes de



agosto del presente año se hizo de gran popularidad, logrando un hackeo masivo de cuentas, comprometiendo la seguridad de los usuarios en redes sociales. Llegando a circular datos de acceso a los perfiles de hasta $10 dolares cada cuenta. La información fue almacenada en la base de datos **Doxagram** [3] en donde se realiza la venta y comercialización de la información.

2.3. Aplicación

Previo a la obtención y análisis de la aplicación dado los datos a obtener se propone definir los requerimientos base que conforman la actividad realizada. El listado de Herramientas utilizadas es:

- **Página de Descarga**: se utiliza una página que disponga de la app.
- **Lionsec**: dado que es necesario decompilar la aplicación, se utiliza el nombrado sistema operativo, debido a sus multiples funciones y herramientas que faciliten el trabajo.
- **Apktool**: herramienta que realiza la lectura y depuración de la aplicación.
- **Dex2jar**: herramienta utilizada para convertir los diferentes .dex propios de android a .jar.
- **Java Decompiler**: utilizado para lograr depurar los archivos .jar.

2.3.1. Obtención de la Aplicación

En primera instancia es necesario conocer que la aplicación se encuentra disponible tanto para sistemas operativos iOs como Android, donde si bien es posible descargar ambos formatos no es posible decompilar los dos por igual. Cabe aclarar que se refiere a decompilar, como la obtención de toda la data proveniente de la aplicación. Entonces si bien, iOs dispone de los layouts o vistas, las cuales pueden ser editadas no se logra editar y ver los códigos base de la app.

Entonces en concordancia con lo anterior, se realiza un análisis de la aplicación **.apk** (Sistema Operativo Android) al ser más amigable al momento de depurar, entregando una serie de archivos y datos correspondientes a la plataforma. Para lograr la obtención de la aplicación se hace uso de **ApkPure** página que dispone de la última versión de Instagram descargable para el computador. Ya en obtención de la aplicación y bajo el uso de las herramientas mencionadas anteriormente se procede a la depuración de la app.

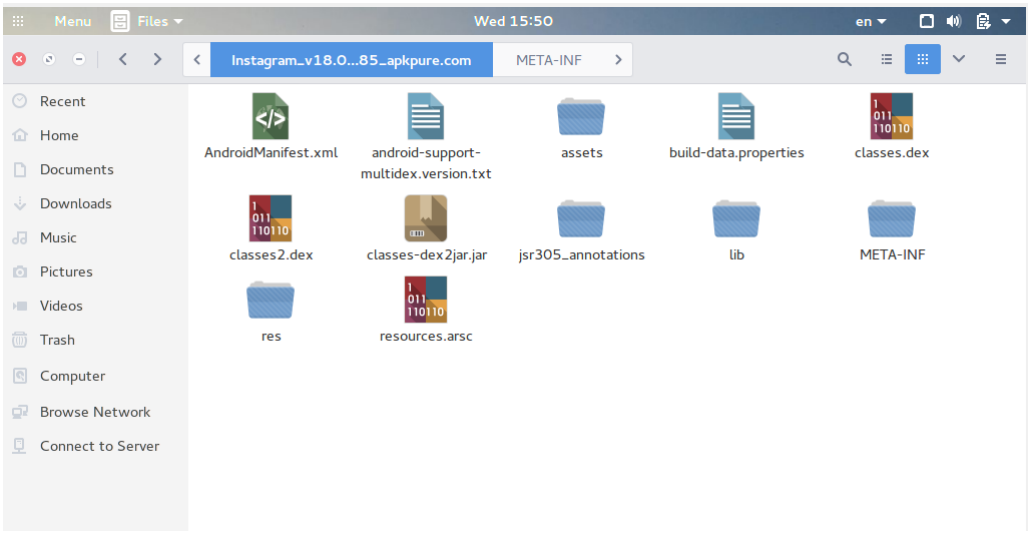
2.3.2. Decompilación de la Aplicación

Con tal de lograr un mejor entendimiento de lo realizado se describe una serie de pasos de la mano con el material obtenido, en formato de imagen o similar.

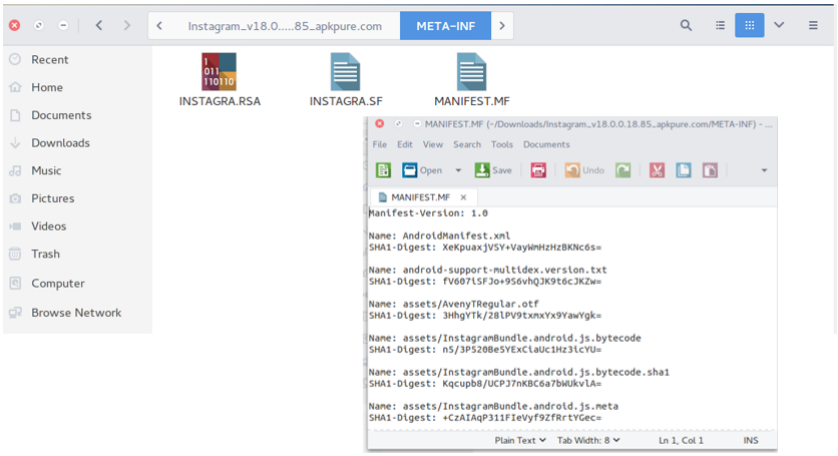
- **Paso 1**: ya en posesión de la .apk correspondiente a la aplicación la primera actividad fue el transformar el archivo de:

  ```
  Intagram.apk => Instagram.zip
  ```

  De la conversión anterior se obtiene acceso a los archivos en bruto de la aplicación, así no, no es posible observar los xml, java o react de la misma, ya que aún no se ha depurado a cabalidad. Igualmente, se obtienen los **.dex** equivalente a las clases utilizadas por la app.



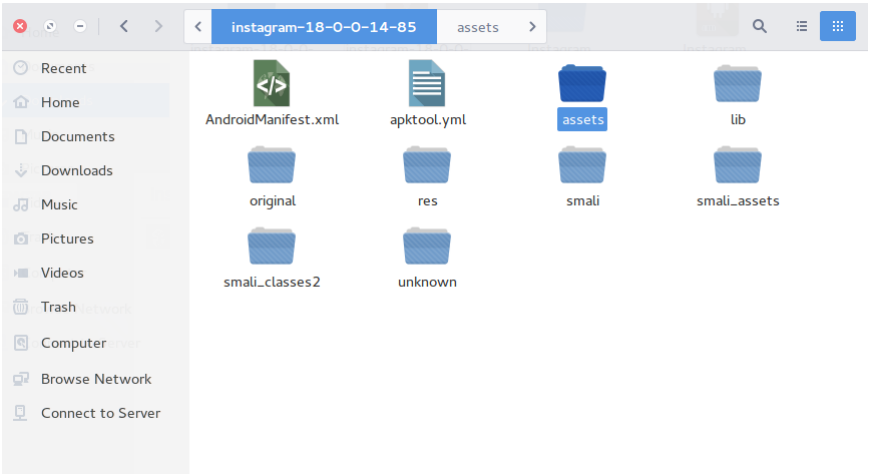
- **Paso 2**: ya extraídos los archivos anteriormente mencionados, se puede observar la carpeta **META-INF** equivalente al certificado de los desarrolladores de la aplicación, la cual al revisar los datos internos se determina que se usa Hashing Sha1 para el certificado de desarrollador y cifrado RSA, para el certificado entregado por Android. Lo dicho puede verse en la siguiente figura:



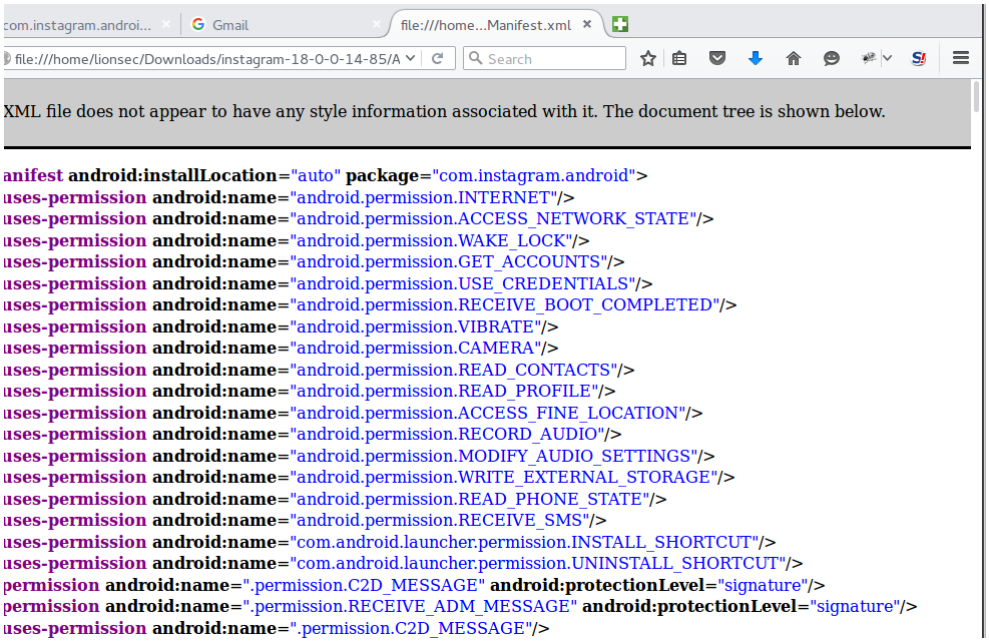
- **Paso 3**: Una vez realizado los pasos anteriores fue posible determinar que no se pueden ver tanto los xml, como los java provenientes de la aplicación. Es por esta razón que se hace uso de software **ApkTool** ya interno en Lionsec. Ya familiarizado con la herramienta se realiza el siguiente comando:

```
apktool d Intagram.apk
```

Donde se define **d** como la lectura del archivo, obteniendo entonces los siguientes archivos:



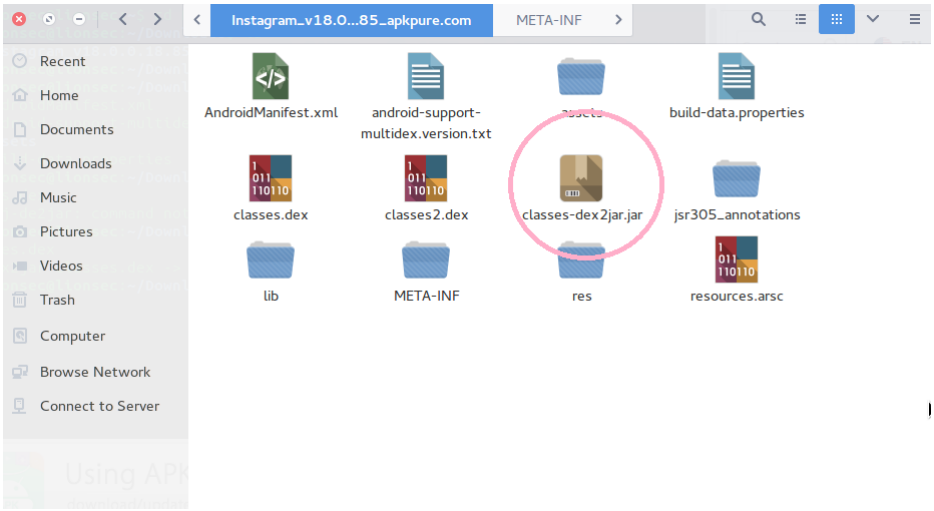
- **Paso 4**: Finalmente tras disponer de la información y datos pertinentes a la depuración de la aplicación. Es necesario revisar los datos obtenidos e información relevante. Los primeros datos revisados fueron los XmlAndroid, los cuales definen parte de los requerimientos y facultades que asocia la aplicación para su desempeño. La siguiente figura da una perspectiva de lo visto.



Si bien se puede encontrar una serie de disposiciones y permisos relacionados a la aplicación, no se considera un punto relevante para la depuración, ya que son puntos relativamente comunes para la aplicación, dígase, no se encuentra anomalías que puedan ser explotables en futuros incrementales.

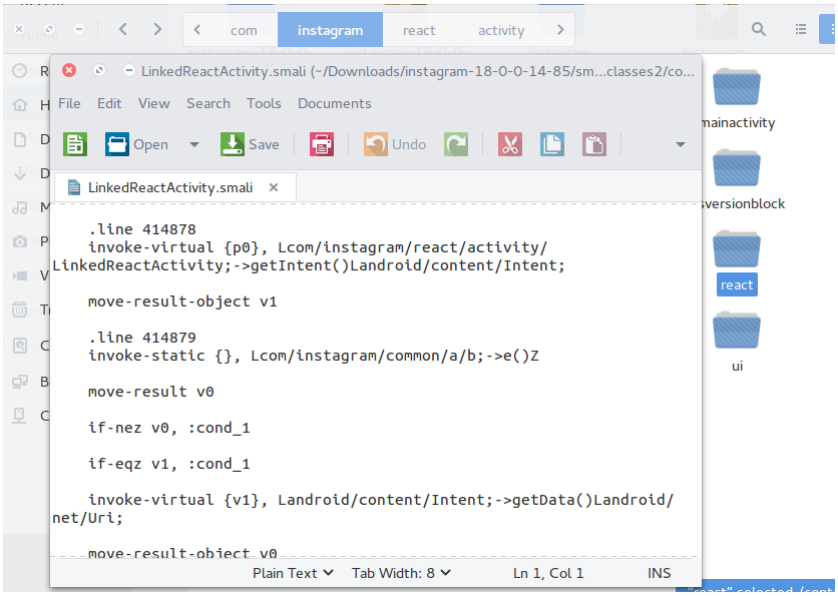
Por otra parte, mediante el uso de la herramienta dex2jar, se realiza una conversión de las clases intrinsecas en la aplicación, mediante el comando:

```
d2j-dex2jar classes.dex
```



Lo que convierte los archivos a un formato .jar, el que por medio de una Java Decompiler y jd-gui interno del mismo, puede lograrse un análisis de los archivos java desarrollados en la aplicación.

Por último se realiza una revisión de los archivos React, los cuales se encuentran en formato **.smali** propio de Java. Obteniendo entonces lo siguiente:



Cabe destacar que los archivos vistos en la figura anterior tienen una real validez en cuanto a vulnerabilidades explotables para la auditoría a realizar, ya que dichas opciones son las actividades, acciones y funciones propias de la aplicación, dando apertura a protocolos, urls, traspaso de información y otras actividades que comprometen la aplicación.

2.4. Análisis de la Aplicación

Previo a la obtención de la aplicación dado los datos a obtener se propone definir los requerimientos base que conforman la actividad realizada. El listado de Herramientas utilizadas es:

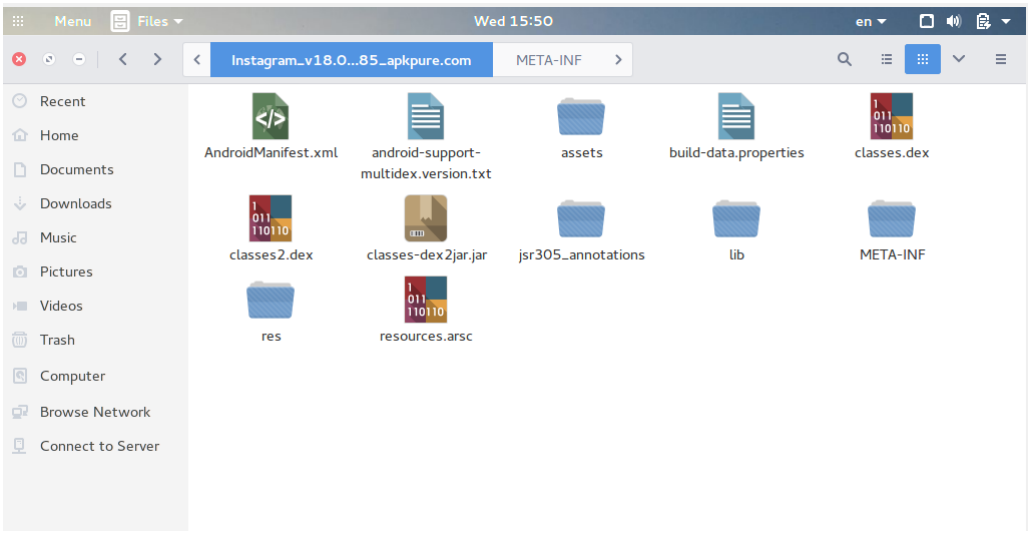
- **Lionsec**: dado que es necesario decompilar la aplicación, se utiliza el nombrado sistema operativo, debido a sus multiples funciones y herramientas que faciliten el trabajo.
- **Apktool**: herramienta que realiza la lectura y depuración de la aplicación.
- **Dex2jar**: herramienta utilizada para convertir los diferentes .dex propios de android a .jar.
- **Java Decompiler**: utilizado para lograr depurar los archivos .jar.
- **Kali Linux**: utilizado para explotar ciertas bajas de la aplicación, con tal de utilizar ciertas herramientas insertas en el sistema operativo.
- **WireShark:** se hace uso de la aplicación como captador de paquetes, según sea pertinente.
- **Metasploit:** herramienta inserta en kali, utilizada para generar pruebas.
- **SQLmap:** herramienta utilizada para pruebas de inyección sql en la app.

2.4.1. Pruebas de Aplicación Decompilada

Con tal de lograr un mejor entendimiento de lo realizado se describe una serie de pasos de la mano con el material obtenido, en formato de imagen o similar.

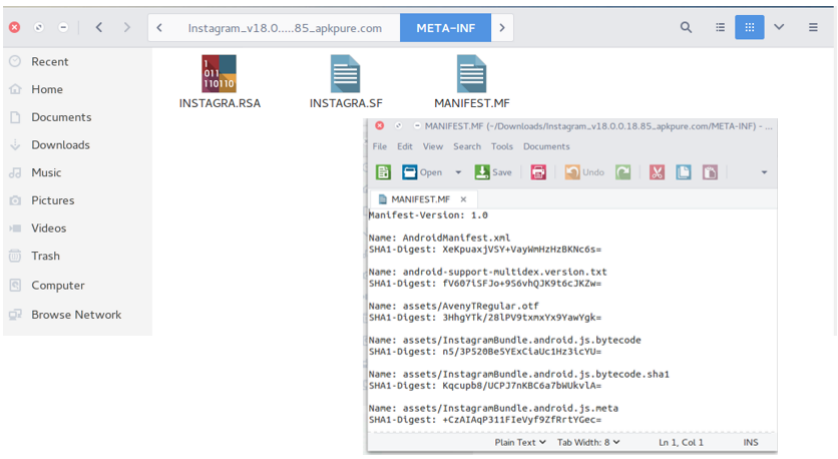
- **Paso 1**: ya en posesión de la .apk correspondiente a la aplicación la primera actividad fue el transformar el archivo de:

```
Intagram.apk => Instagram.zip
```



De la conversión anterior se obtiene acceso a los archivos en bruto de la aplicación, así no, no es posible observar los xml, java o react de la misma, ya que aún no se ha depurado a cabalidad. Igualmente, se obtienen los **.dex** equivalente a las clases utilizadas por la app.

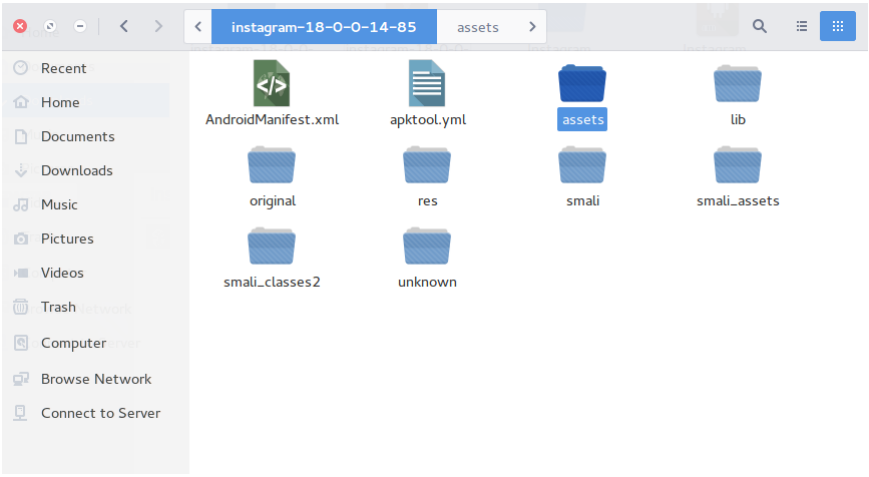
- **Paso 2**: ya extraídos los archivos anteriormente mencionados, se puede observar la carpeta **META-INF** equivalente al certificado de los desarrolladores de la aplicación, la cual al revisar los datos internos se determina que se usa Hashing Sha1 para el certificado de desarrollador y cifrado RSA, para el certificado entregado por Android. Lo dicho puede verse en la siguiente figura:



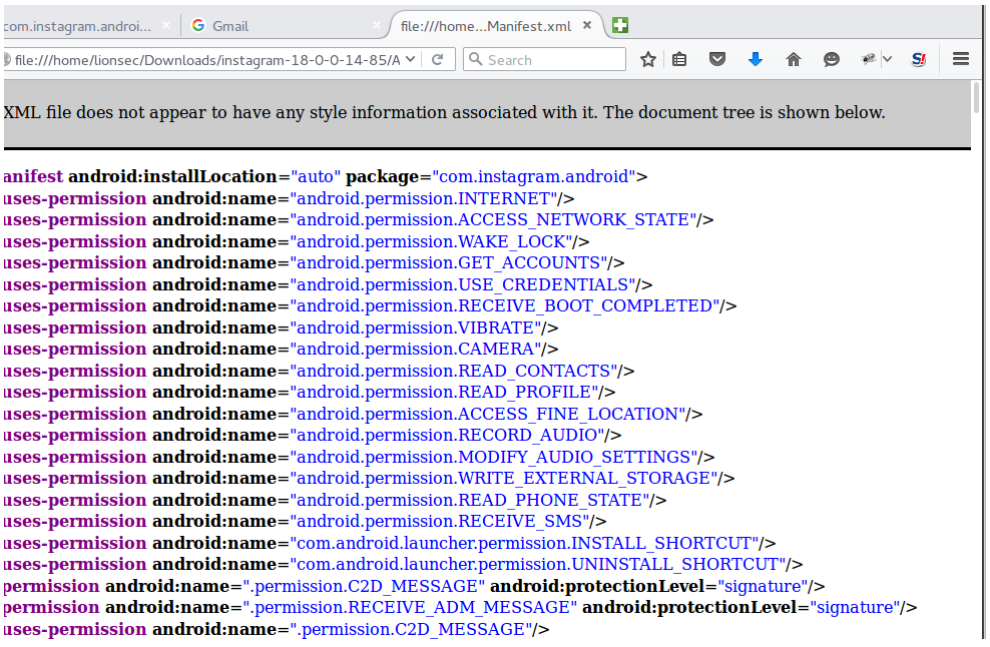
- **Paso 3**: Una vez realizado los pasos anteriores fue posible determinar que no se pueden ver tanto los xml, como los java provenientes de la aplicación. Es por esta razón que se hace uso de software **ApkTool** ya interno en Lionsec. Ya familiarizado con la herramienta se realiza el siguiente comando:

```
apktool d Intagram.apk
```

Donde se define **d** como la lectura del archivo, obteniendo entonces los siguientes archivos:



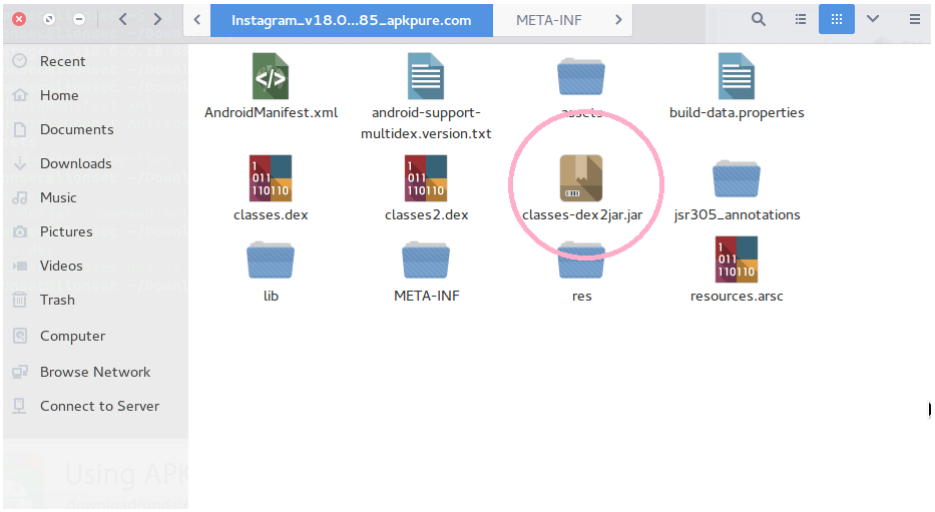
- **Paso 4**: Finalmente tras disponer de la información y datos pertinentes a la depuración de la aplicación. Es necesario revisar los datos obtenidos e información relevante. Los primeros datos revisados fueron los XmlAndroid, los cuales definen parte de los requerimientos y facultades que asocia la aplicación para su desempeño. La siguiente figura da una perspectiva de lo visto.



Si bien se puede encontrar una serie de disposiciones y permisos relacionados a la aplicación, no se considera un punto relevante para la depuración, ya que son puntos relativamente comunes para la aplicación, dígase, no se encuentra anomalías que puedan ser explotables en futuros incrementales.

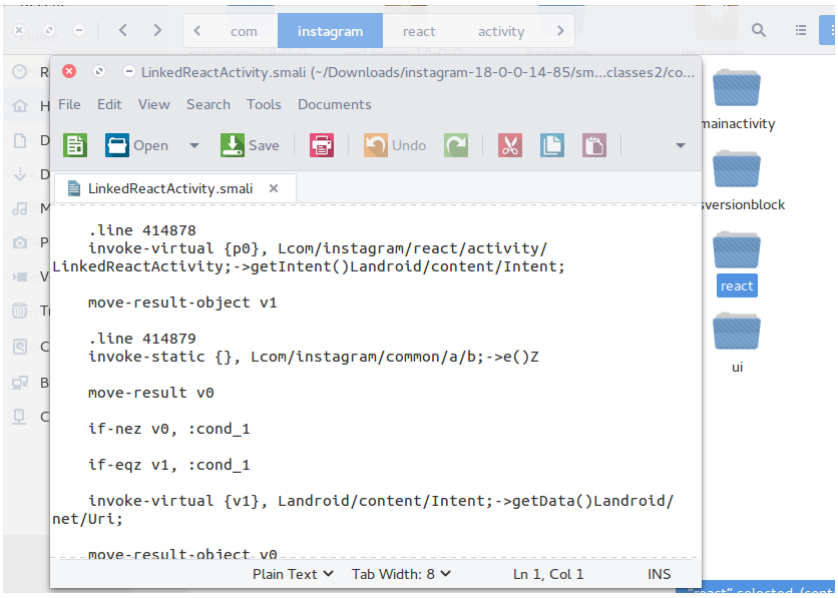
Por otra parte, mediante el uso de la herramienta dex2jar, se realiza una conversión de las clases intrinsecas en la aplicación, mediante el comando:

```
d2j-dex2jar classes.dex
```



Lo que convierte los archivos a un formato .jar, el que por medio de una Java Decompiler y jd-gui interno del mismo, puede lograrse un análisis de los archivos java desarrollados en la aplicación.

Por último se realiza una revisión de los archivos React, los cuales se encuentran en formato **.smali** propio de Java. Obteniendo entonces lo siguiente:



Cabe destacar que los archivos vistos en la figura anterior tienen una real validez en cuanto a vulnerabilidades explotables para la auditoría a realizar, ya que dichas opciones son las actividades, acciones y funciones propias de la aplicación, dando apertura a protocolos, urls, traspaso de información y otras actividades que comprometen la aplicación.

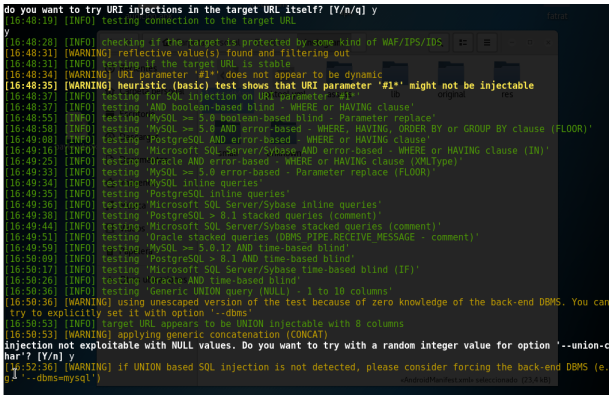
- **Paso 5**: dar a conocer los host encontrados dentro del archivo **Manifest**, donde se encuentra el siguiente host:

www.instagram.com

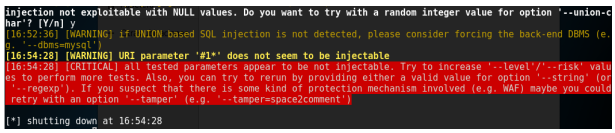
De las cuales se obtiene un listado perteneciente a los métodos navegables del mismo Host, lo que implica un número acotado de url's a analizar. El listado mencionado consta de: Android, explore, mainfeed, news, promote, profile, user, fb-friends y insights.

Por tanto al utilizar la herramienta **SQLMAP**, se generará una serie de inyecciones sql en las direcciones mencionadas, lo que implica una búsqueda de elementos o posibles aperturas dentro del sistema navegable. Al utilizar el siguiente comando:

sqlmap -u www.instagram.com/**Hosts**



En la figura anterior se da una vista de lo anteriormente mencionado, de lo que se obtiene finalmente:



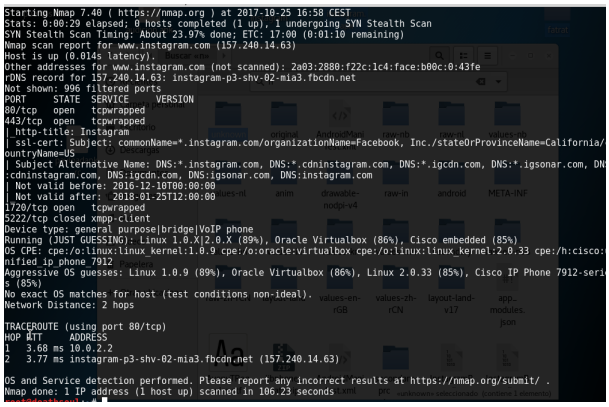
Es entonces que se infiere que no existe ningún resultado concluyente frente a las direcciones probadas, ya que si bien existen ciertas respuestas, sólo implican la respuesta de

vista de usuario o imágenes obtenidas, pero no se encuentra ningún tipo de apertura a explotar dentro de la app.

- **Paso 6:** analizar los posibles puertos expuestos dentro de la aplicación. En la realización de dicha tarea se utilizará el mismo host encontrado en el paso anterior (www.instagram.com), la cuál por medio de la herramienta **NMAP**, se utilizará el siguiente comando:

  nmap -O www.instagram.com

  De lo que obtiene el siguiente resultado:

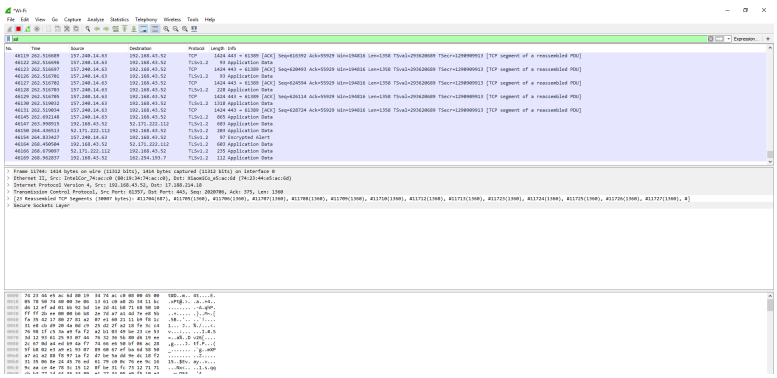


  Donde se encuentra sólo la apertura del puerto 80, perteneciente a flujos HTTP y 443 correspondiente a HTTPS, lo que significa que no existen puntos a explotar o bien puertos expuestos dentro de la aplicación.

- **Paso 7:** se realiza una prueba de man-in-the-middle, donde, con el fin de ver los paquetes captados se utiliza la herramienta **Wireshark** y el celular en versión móvil. Para lograr entender lo anterior mencionado se describe lo siguiente:

  - Utilizar el computador personal, como fuente de internet para analizar los paquetes.
  - Conectar el celular a la señal del computador.
  - Cerrar toda aplicación, página u similar dentro del computador, con tal de captar sólo los paquetes del celular.
  - Utilizar wireshark como captador de las querys realizadas dentro del instagram.

  Ya realizados los pasos anteriores se obtiene el siguiente resultado:

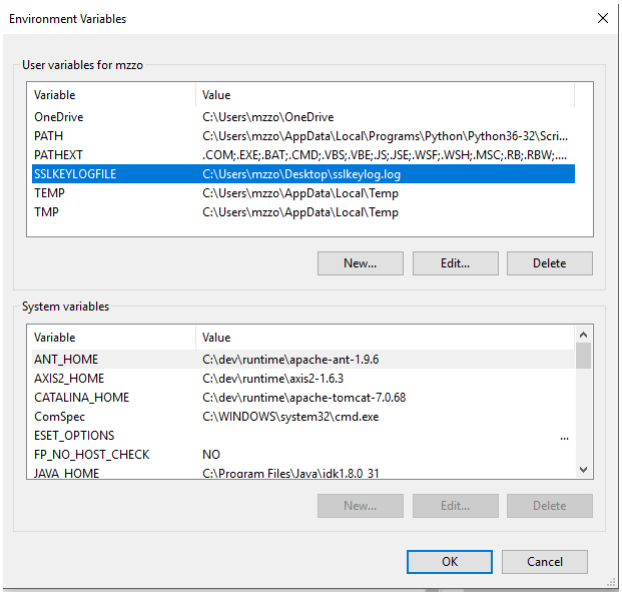


Donde se encuentran todos los paquetes cifrados bajo el certificado ssl, afectando así, en la visualización de la información. Por tanto, se da a entender que no es posible encontrar data **AL MENOS CON WIRESHARK** proveniente de la aplicación.

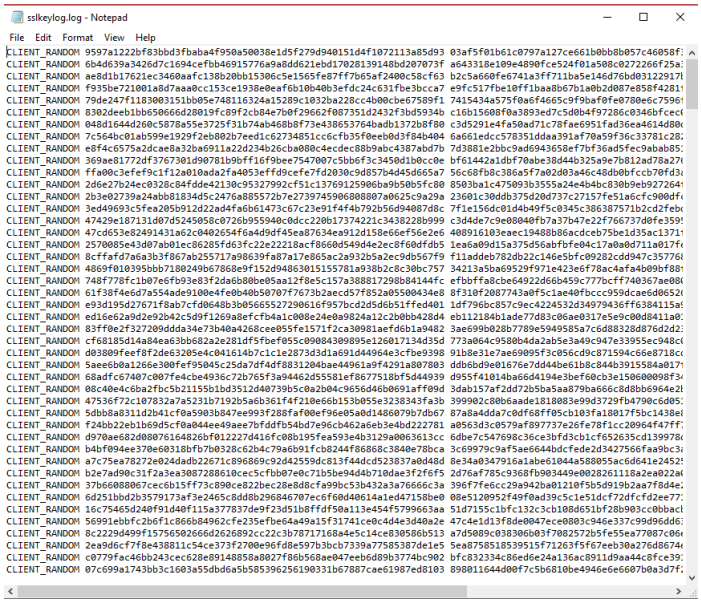
- **Paso 8**: es necesario generar una variable ambiente del tipo **PATH** que contenga de nombre SSL y la ruta del archivo a efectuar:

```
C\\ruta\\ruta\\ sslkeylog.log
```

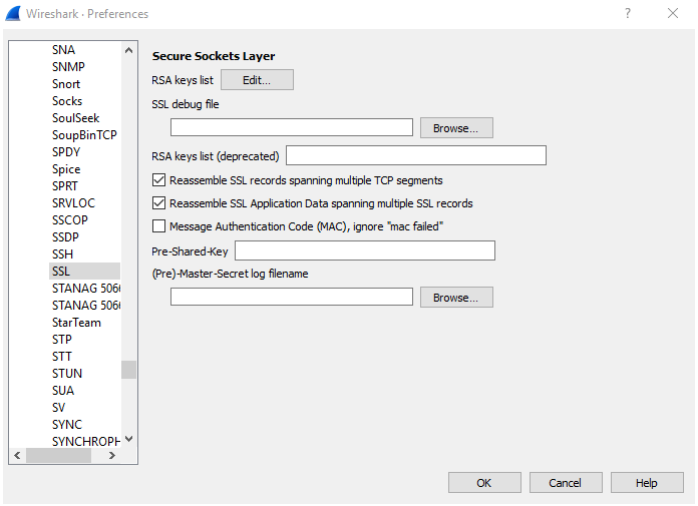
donde **sslkeylog.log** corresponde al archivo contenedor de los certificados ssl falsificados. Lo anterior se describe en la siguiente imagen:



Al generar la variable de ambiente dentro del entorno del sistema operativo, se genera de manera automáticamente el archivo nombrado, el cual contiene:

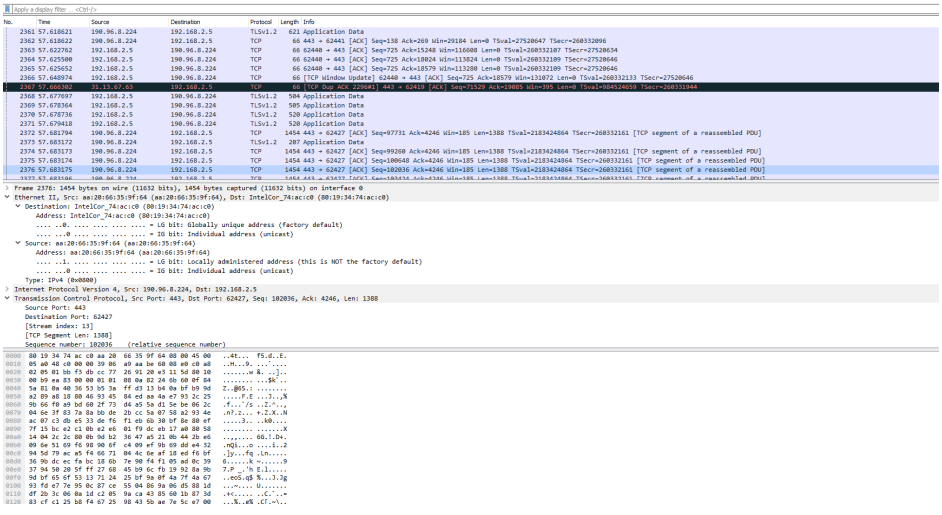


- **Paso 9**: ya extraído el archivo anteriormente mencionado, se procede a adjuntar los certificados a herramienta **WireShark**, con tal de generar una captación y evaluación real del desempeño de los certificados. A continuación se presenta lo descrito:

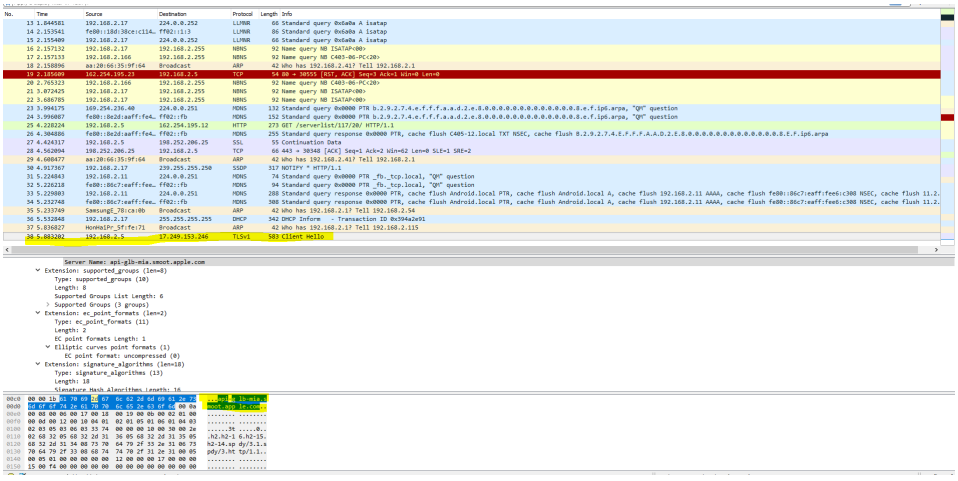


Es necesario proveer la ruta especifica en donde se encuentra el archivo generado por el PATH, de tal manera que trabajen como decifradores de las request generadas al hacer man-in-the-middle, trabajando entonces sobre los paquetes captados al probar la aplicación.

- **Paso 10**: Una vez realizado los pasos anteriores se procede generar la prueba de captación de paquetes, para lo cual fue necesario implementar el computador como emisor de conexión WIfi recepcionado por el dispositivo móvil contenedor de la aplicación. Entonces se obtiene:



La figura anterior describe la prueba asociada, dando a conocer los paquetes e intentando descifrar los archivos y request solicitadas, por medio de los certificados generados. Otro ejemplo concluyente de los resultados fue:



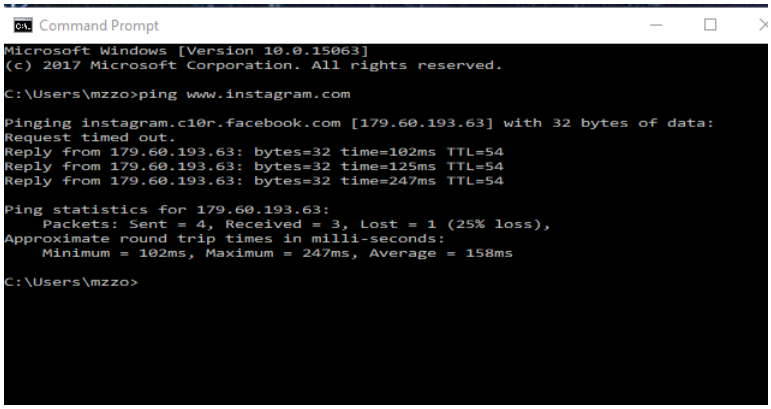
Es posible apreciar lectura de los paquetes, entregando paquetes del tipo TCP y CLIENT HELLO, de la cuál se adjunta una recepción del paquete mayormente ilegible, pero aún así, no se encuentra información consistente y tangible de las respuestas. Lo que implica que no es posible captar la información de los usuarios o bien sentencias explotables, dando a entender que los certificados generados no proveen de información y por tanto de utilidad.

- **Paso 11**: se hace uso de la Herramienta Metasploit utilizada para desarrollar y ejecutar exploits contra una máquina remota, generando scripts de utilidad, declarando ejercicios como DoS u otros, de tal manera de generar pruebas dentro de la IP obtenida.

- **Paso 12**: es necesario realizar un ping por CMD de la página atacar, de tal manera de obtener su Ip y datos pertinentes, utilizando el siguiente comando:

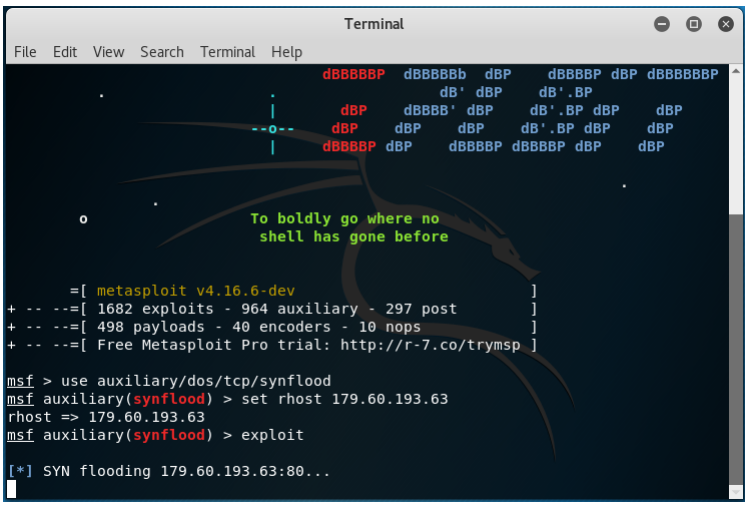
  ping www.instagram.com

  de lo cual se obtiene:



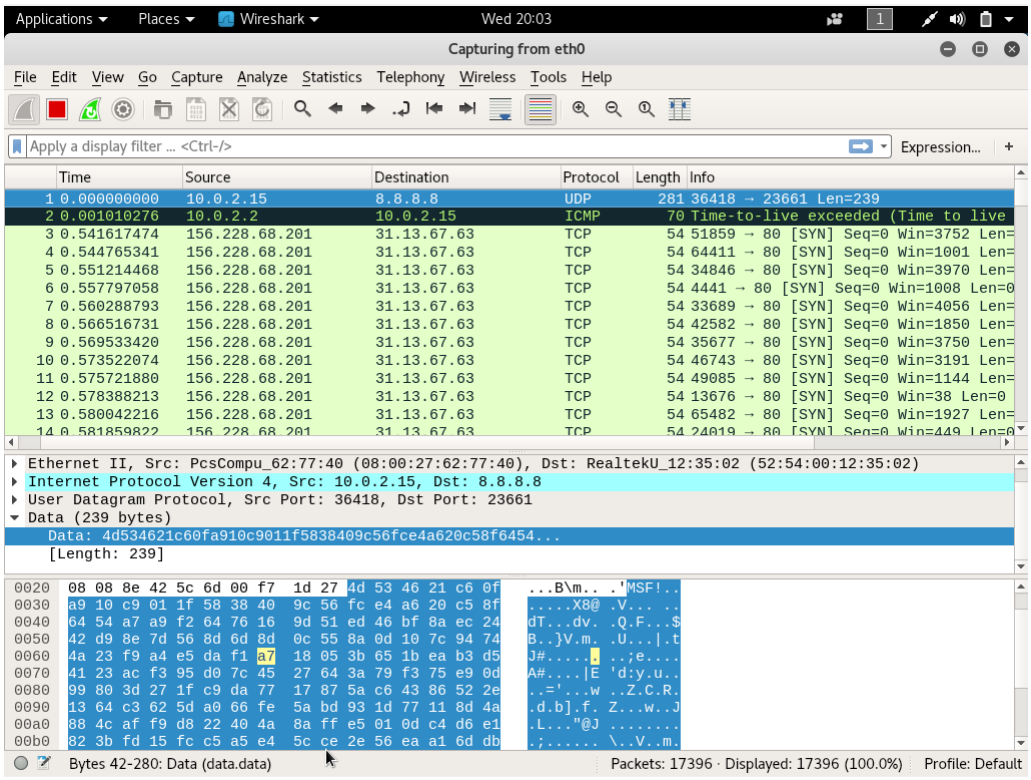
  Si bien al realizar el ping se obtiene una ip consistente, tras realizar diversos ping al mismo sitio se generan distintas IP's lo que implica una distribución realizada por parte de los desarrolladores. Aún con la resolución del asunto es posible destacar respuestas por partes del sitio, dando paso a vistas del tipo ICMP. Entonces se logra destacar que existe uso del servidor **Gunicorn** para manejo de peticiones, el cual a diferencia de Apache es más fácil de implementar y menos intensivo con la CPU, de igual manera para la ejecución de comando se utiliza **Fabric** como despliegue de ejecución (consultas)

- **Paso 13**: Se dispone a utilizar el siguiente comando **use auxiliary/dos/tcp/synflood**, donde se provee **auxiliary**, el cual permite la obtención de información sobr el objetivo, con tal de determinar las posibles vulnerabilidades. Además de proveer el protocolo **tcp** y el tipo de ataque a realizar, el cual para este caso corresponde a **dos** y **synflood**



Entonces tras proveer los comandos necesarios para realizar el ataque es necesario describir el rhost al cual atacar, donde la IP obtenida corresponde al ping generado en el paso anterior, siendo entonces **179.60.193.63**

- **Paso 14**: ya inicializado el ataque se utiliza la herramienta wireshark con tal de ver el envio de paquetes en línea, siendo la respuesta:



En la figura anterior es posible apreciar los diferentes paquetes enviados, asociados a sus respectivas respuestas. Cabe destacar, que si bien el ataque se realiza exitosamente no es posible generar un ataque consistente debido a los escasos recursos y lo amplio de la red de la página asociada.

2.5. Análisis de Riesgos

Frente a las pruebas realizadas, las condiciones y el total de herramientas utilizadas se puede desplegar un listado de aptitudes descriptivas y correspondientes al riesgo real que posee la aplicación, considerando las distintas fuentes y vulnerabilidades probadas, de entre los cuales se nombra:

- **Inyección SQL (Página Web):** aún con el uso de herramientas generadoras de inyecciones, no se encuentra nada crucial o explotables al hablar de la aplicación, lo que implica la inexistencia de vulnerabilidades que dejen en expuesta la aplicación y su data.
- **Apertura de Puertos (Página Web):** haciendo uso de herramientas para escanear los puertos expuestos no se logra encontrar nada concluyente, dando apertura a simples puertos como HTTP o HTTPS, de los cuales no se puede sacar más que el protocolo que se utiliza en la app.
- **Falsificación SSL:** aún con la falsificación o creación exitosa de los certificados, no es posible acceder a información vulnerable, de tal manera que no es posible explotar la aplicación desde ese apartado.
- **Denegación de Servicio (Página Web):** se hace uso de aplicaciones del tipo sploit, con tal de atacar la página, generando scripts basura que manifiesten request o bien entradas en la página generando un posible caída de la app. Dicha actividad no fue existías, debido al alto tráfico capaz de soportar la aplicación como tal, además de poseer ip's distribuidas, dificultando la actividad como tal.
- **Man in the Middle:** aún con el éxito de la actividad, no es posible conseguir elementos explotables dentro de la aplicación, debido a su cifrado incrustado en su desarrollo. Por tanto, no es concluyente la actividad.
- **Decompilación y Compilación:** proceso defectuoso, ya que la aplicación cuenta con seguridad relacionada al asunto, lo que implica que si bien es posible acceder a los datos y en cierta manera cambiarlos al momento de realizar el proceso inverso (compilación) se genera un error permanente relacionado a la confección de la aplicación, dejando claro la imposibilidad de la actividad.

Es posible apreciar un cúmulo de actividades generadas a lo largo de la experiencia, que si bien se logran no cumplen con el cometido, el que es la explotación de vulnerabilidades relacionadas a la aplicación, dejando en claro las claras aptitudes del desarrollo, además de su perceptible seguridad en cada materia y arista de la aplicación. Entonces, es claro mencionar que no existen riesgos relacionados a la integridad de la información propuesta por la aplicación, al menos desde el punto de vista del sistema; así no, los usuarios que utilizan el sistema por medio de ingeniería social son capaces de filtrar información sensible, pero dicho problema recae exclusiva y únicamente en los usuarios de la app.

3. Conclusión

Se logra en su totalidad el objetivo del proyecto, integrando experiencia y conocimientos relacionados al área de seguridad, complementando con factores técnicos de mitigación de errores, herramientas de ataque, fragilidad de redes, tecnologías y búsqueda de ataques. También, se produce la interiorización de software de análisis, escaneo de redes, sistemas operativos, herramientas, sploit y la fragilidad o rigurosidad de los diferentes sistemas, de tal manera de conocer métodos y conceptos vitales en el rubro que determinarán la capacidad en ámbitos de seguridad de un específico sistema.

Se consigue comprender los mínimos requerimientos necesarios para un óptimo y robusto sistema, integrando conceptos, tales como el uso de Servidores, certificado SSL/TLS, deshabilitación de recursos, protección de puertos (ftp,mysql,domain, entre otros), uso de https, cifrado de paquetes, uso de proxy, siendo los mencionados un estándar de recursos esperados en cualquier sistema relativamente seguro.

Por otra parte, frente a la aplicación propuesta se dan a conocer los conceptos previamente mencionados, dando a entender una práctica de desarrollo esperada, tal y como lo es efectuada por los desarrolladores como Facebook o Instagram, siendo para éste caso los mismos, donde se promueve un sistema capaz de solventar un número consistente de información sin poseer grandes bugs que afecten su desempeño o validación, siendo aprobado tanto por la comunidad como los usuarios de la misma, dejando en claro su apta capacidad frente a diferentes pruebas, manteniéndose así, en la vanguardia de los sistemas más seguros.

Referencias

[1] *"AFNetworking: un gran framework de comunicaciones para iPhone/iPad, Blog de Miguel Gutiérrez"*, 2017. [Online]. Available: https://miguelgutierrezmoreno.wordpress.com/2012/11/26/afnetworking-un-extraordinario-framework-para-iphoneipad/. [Accessed: 27- Sep- 2017]

[2] *"Guía de referencia de Nmap (Página de manual)"*, Nmap.org, 2017. [Online]. Available: https://nmap.org/man/es/index.html. [Accessed: 24- Aug- 2017].

[3] *"Peticiones a un servidor desde iOS con AFNetworking"*, AxiaCore, 2017. [Online]. Available: https://axiacore.com/blog/2012/08/peticiones-a-un-servidor-desde-ios-con-afnetworking/. [Accessed: 27- Sep- 2017]

[4] 2017. [Online]. Available: http://www.elconspirador.com/2015/01/20/que-es-apache-thrift-server/. [Accessed: 27- Sep- 2017]

[5] *"Appirater - Remember your users to Rate your App (iOS)"*, Buzztouch.com, 2017. [Online]. Available: https://www.buzztouch.com/forum/thread.php?tid=B60C6EE41CD7D289048B0F4. [Accessed: 27- Sep- 2017].

[6] *"arashpayan/appirater"*, GitHub, 2017. [Online]. Available: https://github.com/arashpayan/appirater. [Accessed: 27- Sep- 2017].

[7] *"Así es el modo Reachability para poder utilizar el iPhone 6 Plus con una sola mano [video]"*, iPhoneros, 2017. [Online]. Available: https://iphoneros.com/44273/asi-es-el-modo-reachability-para-poder-utilizar-el-iphone-6-plus-con-una-sola-mano-video. [Accessed: 10- Oct- 2017].

[8] *"curl - History"*, Curl.haxx.se, 2017. [Online]. Available: https://curl.haxx.se/docs/history.html. [Accessed: 11- Oct- 2017].

[9] La posibilidad de usar varias cuentas en Instagram presenta un fallo de seguridad - TreceBits, TreceBits, 2017. [Online]. Available: http://www.trecebits.com/2016/02/16/la-posibilidad-de-usar-varias-cuentas-en-instagram-presenta-un-fallo-de-seguridad/. [Accessed: 11- Oct- 2017].

[10] El hackeo a Instagram, más grave de lo esperado: 6 millones de cuentas, ADSLZone, 2017. [Online]. Available: https://www.adslzone.net/2017/09/02/el-hackeo-instagram-mas-grave-de-lo-esperado-6-millones-de-cuentas/. [Accessed: 11- Oct- 2017].

[11] R. Álvarez, El sofisticado ataque masivo de phishing que se propagó como pólvora en Gmail y Google Docs, Xataka.com, 2017. [Online]. Available: https://www.xataka.com/seguridad/el-sofisticado-ataque-masivo-de-phishing-que-se-propago-como-polvora-en-gmail-y-google-docs. [Accessed: 30- Aug- 2017].

[12] *"Instagram"*, Instagram, 2017. [Online]. Available: http://www.instagram.com. [Accessed: 19- Nov- 2017].

[13] *"Si6networks.com"*, 2017. [Online]. Available: https://www.si6networks.com/presentations/ANTEL07/fgont-antel07-icmp-attacks.pdf. [Accessed: 19- Nov- 2017].

[14] *"Kali.org"*, 2017. [Online]. Available: https://www.kali.org/. [Accessed: 20- Nov- 2017].

[15] *"Metasploit | Penetration Testing Software, Pen Testing Security | Metasploit"*, Metasploit, 2017. [Online]. Available: https://www.metasploit.com/. [Accessed: 20- Nov- 2017].