

UNIVERSIDAD DIEGO PORTALES
FACULTAD DE INGENIERÍA
ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES
INGENIERÍA DE SOFTWARE

Análisis y Auditoría de Aplicación Android Waze

Autores:

- Matías Álvarez Sabaté.
- Maximiliano Vega Curaqueo.

Índice

| | |
|--|-----------|
| 1. Introducción | 1 |
| 2. Impacto de auditar Waze | 2 |
| 3. Defacing | 2 |
| 4. Decompilación de la aplicación | 3 |
| 5. Técnicas de seguridad dentro del código | 7 |
| 6. Ataques anteriores a Waze | 8 |
| 6.1. Estudiantes Israelíes | 8 |
| 6.2. ¿Está resuelta esta vulnerabilidad? | 8 |
| 7. Función Indent (Direcciones waze://) | 14 |
| 8. Modificación y eliminación de paquetes | 15 |
| 9. Funcionalidad en maquina virtual | 16 |
| 10. Conclusiones | 25 |
| 11. Bibliografía | 26 |

Índice de figuras

| | | |
|-----|---|----|
| 1. | Imagen de prueba para cambian en iconografía | 2 |
| 2. | Error al intentar cambiar imagen | 3 |
| 3. | Decompilación del APK Studio | 4 |
| 4. | Archivos Smali convertidos a Java | 5 |
| 5. | Busqueda de referencia de una base de datos | 5 |
| 6. | Sentencia sql | 6 |
| 7. | Busqueda de direcciones que comienzan con “waze://” | 6 |
| 8. | Cotenido de los ficheros de IntentManager | 7 |
| 9. | Notificación de Waze de fallo de enviar alerta | 9 |
| 10. | Archivo build.prop | 9 |
| 11. | Envío de alertas ubicación 1, dispositivo 1 | 10 |
| 12. | Vista del punto de envío de las alertas por el dispositivo 1, en el dispositivo 2 | 11 |
| 13. | Envíos de alerta en el dispositivo 2 | 12 |
| 14. | Vista del punto de envío de las alertas por el dispositivo 2, en el dispositivo 1 | 13 |
| 15. | Sitio HTML que invoca url waze:// | 14 |
| 16. | Ejecución de Waze al abrir la url Waze:// en la imagen anterior | 14 |
| 17. | Error de conexión a internet | 15 |
| 18. | Error generado al ser modificado previamente un servidor DNS en el dispositivo | 16 |
| 19. | Aplicaciones instaladas para fakegps | 16 |
| 20. | Waze no muestra mapa en maquina virtual (VirtualBox) | 17 |
| 21. | En la maquina virtual no se muestra el mensaje de alerta enviada | 17 |
| 22. | Mensaje de alerta enviada mostrada en un celular | 18 |
| 23. | Waze en el emulador de Android Bluestacks | 18 |
| 24. | En Bluestack, Waze si muestra el mapa como en un celular | 19 |
| 25. | Alerta enviada usando Bluestacks aparece se muestra en un celular | 20 |
| 26. | Archivos de Waze dentro de la maquina virtual | 21 |
| 27. | Más archivos de Waze dentro de la máquina virtual | 22 |
| 28. | Contenido archivo alerts.menu | 22 |
| 29. | Archivo alerts.menu modificado | 23 |
| 30. | Menu de alertas no es afectado por la modificación del archivo alerts.menu | 24 |
| 31. | Modificación del archivo menu usando SSHelper | 25 |

1. Introducción

Waze es una aplicación social para conductores donde los usuarios pueden buscar direcciones y Waze los guiará calculando una ruta de acuerdo a las preferencias del usuario y las condiciones de los posibles caminos. Las condiciones del camino son determinada por lo que los mismos usuarios retroalimentan al usar la aplicación mientras conducen y la aplicación en compensación por ese feedback, otorga puntaje e íconos personalizados. Los usuarios más experimentados incluso pueden editar el mapa para contribuir en posibles problemas que existan como por ejemplo vías en sentido contrario o los mismos cambios contemporáneos de construcciones de nuevos caminos. También los usuarios pueden alertar la presencia de policías, problemas en el camino, accidentes, problemas de tráfico y por último la instancia de poder conversar (chat) o enviar mensajes a otros “wazers”.

En este informe, auditaremos Waze analizando el tráfico que genera en cada una de sus acciones con el objetivo de encontrar vulnerabilidades por alguna mala práctica de los programadores de la aplicación o bien de los sistemas.

Fue elegida Waze porque es una aplicación con muchos usuarios, con funciones importantes que ayudan a una ciudad más inteligente, además de un chat entre los usuarios con lo que si hubiera una posibilidad de ataque, se podrían producir cambios drásticos en las vías escogidas, generando problemas de tráfico en alguna avenida o carretera importante en algún país del mundo provocando retrasos en las llegadas de muchas personas a sus destinos que podría incidir económicamente en dicho país. Además de ser una de las pocas aplicaciones del ámbito, ya que de por sí, es un elemento distractor.

Un caso práctico en Chile podría ser en el área del cobre, al ser un pilar importante en la economía nacional. Si este ataque se hiciera en la Región de Antofagasta, muchos trabajadores verán retrasada su llegada a la mina de Chuquicamata. Lo mismo que en el caso de la Región de O’Higgins para la mina El teniente, entre otras. Que baje la producción de cobre en Chile afecta negativamente la economía nacional, incluso en millones de dólares en pérdidas por falta de mano de obra.

2. Impacto de auditar Waze

Waze comenzó en el año 2008 como una empresa independiente, Waze Ltd., como un GPS social el cual se fue masificando debido a la calidad de información que entrega a los conductores y, luego de ser adquirida por Google en 2013, aumentó aún más su popularidad en Play Store y hoy en día es usada por millones de usuarios en el mundo.

Conjunto a la popularidad cae una gran responsabilidad en Waze. Como millones de usuarios se fían de la información entregada por la aplicación, los sistemas deben ser capaces de combatir falsos avisos, avisos masivos o caídas por denegación de servicio. Además de contar con información sensible como la localización del usuario, el teléfono del usuario (solicitado al instalar la aplicación y además con la necesidad de verificarlo) y sus rutas favoritas, no se puede tomar a la ligera la comunicación con el backend que tenga Waze.

3. Defacing

El defacing consiste en cambiar o alterar las imágenes o iconografías de la aplicación, en este caso con Waze se intentó cambiar el ícono de Waze por una lata de SPAM.



Figura 1: Imagen de prueba para cambiar en iconografía

Dentro del directorio de Assets, donde se encuentran las gráficas.

No fue posible realizarlo ya que Waze cuenta con ProGuard. Esto no nos permitió volver a compilar la aplicación dada la ofuscación presente en el código [Error en Figura 2].

```

Warning: okio.Okio: can't find referenced class java.nio.file.OpenOption
Warning: okio.Okio: can't find referenced class java.nio.file.Path
Warning: okio.Okio: can't find referenced class java.nio.file.OpenOption
Warning: okio.Okio: can't find referenced class com.waze.IgnoreJRERequirement
Warning: there were 234 unresolved references to classes or interfaces.
      You may need to add missing library jars or update their versions.
      If your code fails to run due to missing classes, you can suppress
      the warnings with '-dontwarn' options.
      (http://proguard.sourceforge.net/manual/troubleshooting.html#unresolvedclass)
Warning: there were 4 unresolved references to program class members.
      Your input classes appear to be inconsistent.
      You may need to recompile the code.
      (http://proguard.sourceforge.net/manual/troubleshooting.html#unresolvedprogramclassmember)
:app:proguardRelease FAILED
Warning: Google ID not recognized, are you the real developer of this app?

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':app:proguardRelease'.
> java.io.IOException: Please correct the above warnings first.

* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log
BUILD FAILED

Total time: 14.674 secs
Please correct the above warnings first.
13:40:35: External task execution finished 'assembleRelease'.

```

Figura 2: Error al intentar cambiar imagen

ProGuard es una herramienta de compilación para Android. Ofusca, minimiza, optimiza y licencia el código de fuente para dificultar la ingeniería inversa en una aplicación. [1]

4. Decompileación de la aplicación

Los archivos APK, que son archivos instalables en Android que viene de la sigla Application Package es una variante de los ficheros JAR, de Java Archive, que es un paquete que contiene los archivos necesarios para instalar o hacer correr una aplicación Java.

A su vez, un JAR es comparable a un ZIP pero la mayoría de las veces viene sin compresión.

Además, Java, el lenguaje de programación de las aplicaciones Android, es un lenguaje que se compila a bytecode, que es un intermediario entre código de máquina y binario. Este bytecode se hace correr sobre una máquina virtual Java.

Finalmente, esto quiere decir que Java es precompilado o preparado para correr más rápido sobre una máquina virtual de Java y gracias a esto es posible lograr programas multiplataforma. Basta con generar una máquina virtual para otro sistema y el programa ya debería correr.

El problema de esto es que la capacidad de poder realizar ingeniería inversa es mayor, como el código es intermedio, es más cercano a lo que se programó originalmente.

El software APK Studio permite realizar esta ingeniería inversa y generar el árbol de directorios y los instructivos de Java en Smali. Smali es otra forma de decir Assembly (Assembler)

| Name | Date modified | Type | Size |
|---------------------|----------------|--------------|-------|
| assets | 21/04/15 12:42 | File folder | |
| lib | 21/04/15 12:42 | File folder | |
| original | 21/04/15 12:43 | File folder | |
| res | 21/04/15 12:39 | File folder | |
| smali | 21/04/15 12:42 | File folder | |
| unknown | 21/04/15 12:43 | File folder | |
| AndroidManifest.xml | 21/04/15 12:38 | XML Document | 29 KB |
| apktool.yml | 21/04/15 12:43 | YML File | 37 KB |

Figura 3: Decompilación del APK Studio

En la figura 3 podemos ver la descompilación que realizó APK Studio. En Assets y RES se encuentran las imágenes de la aplicación, en Original encontramos los manifiestos (archivos de configuración de muestra de la aplicación en android).

Dentro de Smali/com/waze encontramos los assembly de los diferentes trozos de código.

Como queremos un código más entendible, transformamos Smali a Java con smali2java ([2]).

| Nombre | | Fecha de modificación | Tamaño | Clase |
|---------------------------------------|---|-----------------------|-----------|------------------|
| ActionIntent.java | | 12-04-2015 17:56 | 2 KB | Código...te Java |
| ActivityNativeManagerBase.java | | 12-04-2015 17:56 | 1 KB | Código...te Java |
| ActivityRecognitionConnection.java | | 12-04-2015 17:56 | 2 KB | Código...te Java |
| ActivityRecognitionReceiver.java | | 12-04-2015 17:56 | 1 KB | Código...te Java |
| ActivityRecognitionService.java | | 12-04-2015 17:56 | 2 KB | Código...te Java |
| AdsTracking.java | | 12-04-2015 17:56 | 3 KB | Código...te Java |
| AdsTracking\$1.java | | 12-04-2015 17:56 | 2 KB | Código...te Java |
| AdsTracking\$AdsData.java | | 12-04-2015 17:56 | 476 bytes | Código...te Java |
| analytics | ▶ | 12-04-2015 17:56 | -- | Carpetas |
| animation | ▶ | 12-04-2015 17:55 | -- | Carpetas |
| AppService.java | | 12-04-2015 17:56 | 22 KB | Código...te Java |
| AppService\$1.java | | 12-04-2015 17:56 | 835 bytes | Código...te Java |
| AppService\$2.java | | 12-04-2015 17:56 | 1 KB | Código...te Java |
| AppService\$3.java | | 12-04-2015 17:56 | 538 bytes | Código...te Java |
| AppService\$EndCallListener.java | | 12-04-2015 17:56 | 2 KB | Código...te Java |
| AppService\$EndCallListener\$1.java | | 12-04-2015 17:56 | 805 bytes | Código...te Java |
| AppService\$ServiceMsgDispatcher.java | | 12-04-2015 17:56 | 1 KB | Código...te Java |
| AppUUID.java | | 12-04-2015 17:56 | 1 KB | Código...te Java |
| authenticator | ▶ | 12-04-2015 17:56 | -- | Carpetas |
| autocomplete | ▶ | 12-04-2015 17:56 | -- | Carpetas |
| BuildConfig.java | | 12-04-2015 17:56 | 315 bytes | Código...te Java |
| CloseIntent.java | | 12-04-2015 17:56 | 1 KB | Código...te Java |
| config | ▶ | 12-04-2015 17:56 | -- | Carpetas |
| ConfigItem.java | | 12-04-2015 17:56 | 1 KB | Código...te Java |
| ConfigManager.java | | 12-04-2015 17:56 | 8 KB | Código...te Java |
| ConfigManager\$1.java | | 12-04-2015 17:56 | 1 KB | Código...te Java |
| ConfigManager\$2.java | | 12-04-2015 17:56 | 506 bytes | Código...te Java |
| ConfigManager\$3.java | | 12-04-2015 17:56 | 506 bytes | Código...te Java |
| ConfigManager\$4.java | | 12-04-2015 17:56 | 506 bytes | Código...te Java |
| ConfigManager\$5.java | | 12-04-2015 17:56 | 980 bytes | Código...te Java |
| ConfigManager\$6.java | | 12-04-2015 17:56 | 1 KB | Código...te Java |
| ConfigManager\$ConfigUpdater.java | | 12-04-2015 17:56 | 375 bytes | Código...te Java |
| ConnEventReceiver.java | | 12-04-2015 17:56 | 3 KB | Código...te Java |
| CPUProfiler.java | | 12-04-2015 17:56 | 5 KB | Código...te Java |

Figura 4: Archivos Smali convertidos a Java

En la figura 4 ya tenemos el código convertido en Java. Aprovechando esto, primero buscamos alguna referencia de alguna base de datos:

```
→ waze grep -rl '.sqlite' ./
  ./db/CustomPathSQLiteOpenHelper.java
  ./db/WazeFavoritePlaceSQLiteHelper.java
→ al waze
```

Figura 5: Busqueda de referencia de una base de datos

En el primer archivo (Figura 5) solo hay configuración de la aplicación para utilizar una ruta específica para el fichero SQLite y en el segundo, encontramos una sentencia SQL:

```

Cursor cursor = sqitedatabase.rawQuery("SELECT type, longitude, 1
atitute FROM FAVORITES join PLACES on FAVORITES.place_id = PLACES.id where
(FAVORITES.type = 1 or FAVORITES.type = 2)", null);

```

Figura 6: Sentencia sql

En la figura 26 podemos ver que las ubicaciones favoritas (función de Waze) se guarda en el archivo SQLite.

Comenzamos la búsqueda de direcciones pero notamos que las direcciones comienzan con “waze://”, esto en una aplicación quiere decir que se definió un esquema único para la aplicación (Esto es, que desde cualquier otra aplicación o navegador, se puede ejecutar Waze mediante una URL ([3]).

```

source grep -rn "waze://" ./100% Texto normal Verdana 11 B I Z U A - 561 items, 17.94 GB disk
//smali/com/waze/analytics/AnalyticsEvents.smali:13: field public static final ANALYTICS_ADS_PHONE_PREFIX:Ljava/lang/String; = "waze://open_url"
//smali/com/waze/analytics/AnalyticsEvents.smali:15: field public static final ANALYTICS_ADS_URL_PREFIX:Ljava/lang/String; = "waze://?open_url"
//smali/com/waze/IntentManager.smali:68: const-string v5, "waze://"
//smali/com/waze/IntentManager.smali:91: const-string v13, "waze://?l1="
//smali/com/waze/IntentManager.smali:17: const-string v13, "waze://?q"
//smali/com/waze/IntentManager.smali:79: const-string v13, "waze://?l1=" Figura[25]
//smali/com/waze/IntentManager.smali:66: const-string v13, "waze://?p"
//smali/com/waze/IntentManager.smali:68: const-string v13, "waze://?p"
//smali/com/waze/IntentManager.smali:96: const-string v13, "waze://?l1=" solo hay configuración de la aplicación para
//smali/com/waze/IntentManager.smali:98: const-string v13, "waze://?q" para el fichero SQLite y en el segundo, encontramos
//smali/com/waze/IntentManager.smali:96: Uri const-string v13, "waze://?l1="
//smali/com/waze/IntentManager.smali:97: Uri const-string v13, "waze://?q"
//smali/com/waze/IntentManager.smali:99: const-string v13, "waze://?l1="
//smali/com/waze/IntentManager.smali:108: const-string v13, "waze://?q"
//smali/com/waze/IntentManager.smali:108: Uri const-string v13, "waze://?l1=" la const-string v4, "waze://s (función de Waze) se
//smali/com/waze/IntentManager.smali:108: (Fc const-string v13, "waze://?l1=" "FAVITES.type = 2)", null1;
//smali/com/waze/mwaze/ScoreboardActivity.smali:4: const-string v4, "waze://"
//smali/com/waze/NavigationManager.smali:79: field public static final WAZE_URL_PATTERN:Ljava/lang/String; = "waze://"
//smali/com/waze/navigate/AddressOptionsActivity$PreviewTabManager$2.smali:144: const-string v5, "waze://?open_url"
//smali/com/waze/navigate/AddressOptionsActivity$PreviewTabManager$2.smali:173: const-string v5, "waze://"
//smali/com/waze/navigate/AddressOptionsActivity$PreviewTabManager$3.smali:130: la const-string v4, "waze://s
//smali/com/waze/navigate/AddressPreviewActivity$3.smali:133: const-string v1, "waze://open_url"
//smali/com/waze/navigate/AddressPreviewActivity$3.smali:159: const-string v1, "waze://"
//smali/com/waze/navigate/AddressPreviewActivity$3.smali:159: const-string v1, "waze://"
//smali/com/waze/profile/FacebookConnectActivity.smali:127: const-string v2, "waze://dialog_hide_current"
//smali/com/waze/profile/FacebookConnectActivity.smali:136: const-string v2, "waze://browser_close"
//smali/com/waze/profile/FoursquareConnectActivity.smali:92: const-string v1, "waze://dialog_hide_current"
//smali/com/waze/profile/FoursquareConnectActivity.smali:181: const-string v1, "waze://browser_close"
//smali/com/waze/profile/TwitterConnectActivity.smali:155: const-string v1, "waze://dialog_hide_current"
//smali/com/waze/profile/TwitterConnectActivity.smali:164: const-string v1, "waze://browser_close"
//smali/com/waze/push/Alert.smali:153: const-string v1, "waze://?pickup"
//smali/com/waze/push/Alert.smali:158: const-string v0, "waze://"
//smali/com/waze/push/Alert.smali:152: const-string v6, "waze://"
//smali/com/waze/push/Constants.smali:63: field protected static final URL ACTION_PREFIX:Ljava/lang/String; = "waze://"
//smali/com/waze/share/FacebookLikeActivity.smali:02: const-string v1, "waze://dialog_hide_current"
//smali/com/waze/share/FacebookLikeActivity.smali:01: const-string v1, "waze://browser_close"
//smali/com/waze/share/FacebookShareActivity.smali:87: const-string v1, "waze://dialog_hide_current"
//smali/com/waze/share/FacebookShareActivity.smali:96: const-string v1, "waze://browser_close"
//smali/com/waze/share/TwitterFollowActivity.smali:87: const-string v1, "waze://dialog_hide_current"
//smali/com/waze/share/TwitterFollowActivity.smali:96: const-string v1, "waze://browser_close"
//smali/com/waze/view/web/SimpleWebActivity.smali:38: const-string v0, "waze://dialog_hide_current"
//smali/com/waze/view/web/SimpleWebActivity.smali:43: const-string v0, "waze://browser_close"
//smali/com/waze/widget/WazeAppWidgetService.smali:201: const-string v3, "Starting waze waze://?favorite="
```

Figura 7: Busqueda de direcciones que comienzan con “waze://”

Investigando, las aplicaciones deben realizar una instanciación de la clase Intent para modificar el comportamiento con una URL específica de una aplicación.

Los ficheros de IntentManager encontramos lo siguiente:

```
-keep class android.support.** { *; }

-keep interface android.support.** { *; }

-keep class com.waze.** { *; }

-keep interface com.waze.** { *; }
```

Figura 8: Contenido de los ficheros de IntentManager

Lo que quiere decir que utilizaron técnicas de ofuscación de código para ocultar la URL original.

La ofuscación de código se utiliza para justamente ocultar partes del código sensibles, forzando a que sean compiladas en el momento de crear el APK.

En toda solicitud URL hay ofuscación de código. ([88]).

5. Técnicas de seguridad dentro del código

- Waze utiliza archivos Smali que contienen funciones de confidencialidad de la información de cada usuario como son los mensajes por chat, mantener privacidad ubicación del usuario y sus datos personales, y que estos viajen encriptados y que sea difícil obtener ésto en texto plano.
- Hay código ofuscado con proguard.

6. Ataques anteriores a Waze

6.1. Estudiantes Israelíes

En abril del 2014, 2 estudiantes de ingeniería de software llamados Shir Yadid y Meital Ben-Sinaí del Instituto de Tecnología de Israel, como proyecto de su carrera, crearon su propio software. La función de este software consistía en simular un gran taco en una carretera importante de Israel, para desviar el tráfico por carreteras alternativas, pero teniendo en cuenta que esto podría haber provocado un gran caos fue hecho experimentalmente en una de las calles secundarias de su campus. Este experimento se compuso de las siguientes etapas según cuentan los estudiantes ([5],[6]):

- Primera etapa: Crear dispositivos móviles con android falsos con un emulador de android.
- Segunda etapa: un sistema de control que simulara la intervención humana en el dispositivo que incluía la creación de una cuenta, iniciar sesión y hacer uso de las funciones de Waze. Esto provocó muchos usuarios “wazers” falsos operando.
- Tercera etapa: Analizar la densidad del tráfico.

Luego de éste trabajo, los estudiantes lo notificaron a los desarrolladores de Waze sobre la vulnerabilidad encontrada.

6.2. ¿Está resuelta esta vulnerabilidad?

Para comprobarlo, realizamos el mismo ataque. Creando 3 máquinas virtuales con Android 4.4 (aprovechando el proyecto Android X86 [8]) instalamos Waze en cada máquina y realizamos un seguimiento de lo que informa Waze en una misma calle.

La instalación de Waze en la máquina virtual fue exitosa y además pudimos verificar la cuenta, pero al momento de cargar el mapa no se pudo, dado al problema de GPS con el cual no se cuenta en la máquina virtual y además alertar en cualquier ubicación fue imposible:

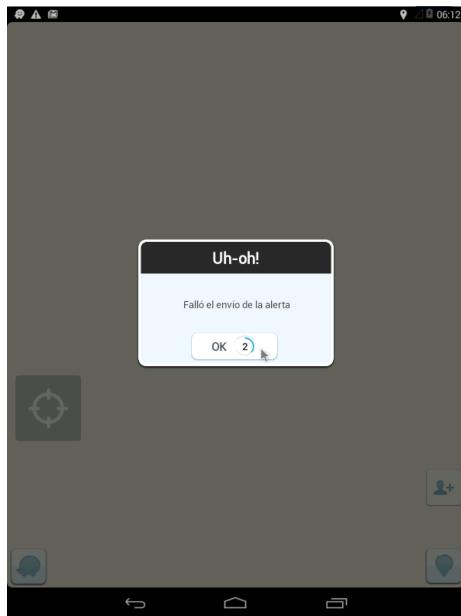


Figura 9: Notificación de Waze de fallo de enviar alerta

Para comprobar si Waze discrimina por información del OS (Android posee un archivo donde guarda la información de la versión del Sistema Operativo), modificamos el archivo /system/build.prop con la misma configuración del teléfono móvil:

```
< build.prop
# begin build properties
# autogenerated by buildinfo.sh
ro.build.id=MindCr Rom V1.0.12
ro.build.display.id=Jellybean 4.2.2 by Maddy
ro.build.version.incremental=XXLA2
ro.build.version.sdk=10
ro.build.version.codename=REL
ro.build.version.release=4.2.2
```

Figura 10: Archivo build.prop

De todas maneras no fue posible enviar una alerta desde una máquina virtual.

De manera alterna, se intentó alertar con más de un dispositivo móvil una misma ubicación. Al parecer, si el usuario Waze no ha pasado por la ubicación o no se encuentra cerca, su alerta se recibe pero no se hace válida. En nuestro caso, uno de nosotros desde el sector oriente de Santiago realizó una alerta en el sector sur, La Florida, de tráfico denso. Esta alerta no apareció en la aplicación, pero al alertar cerca de la ubicación apareció inmediatamente en ambos dispositivos.

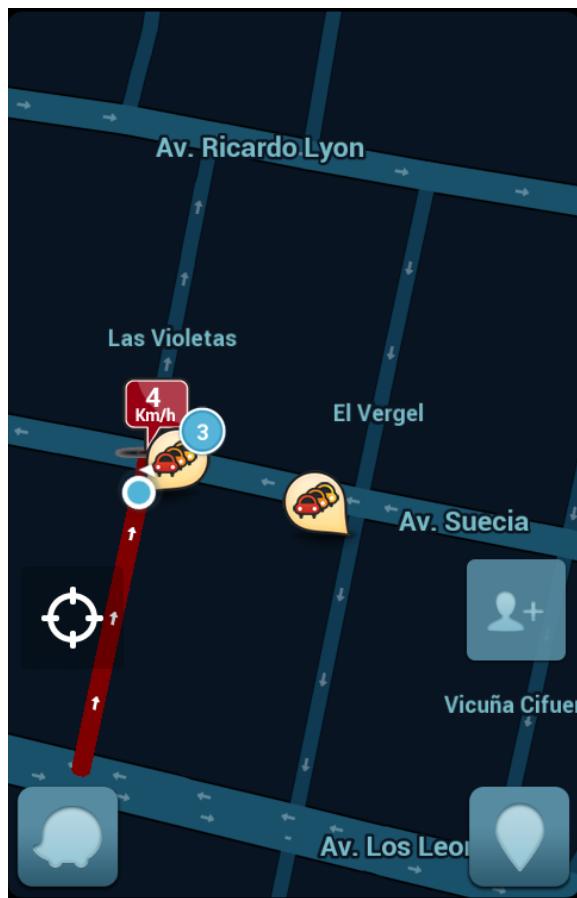


Figura 11: Envío de alertas ubicación 1, dispositivo 1

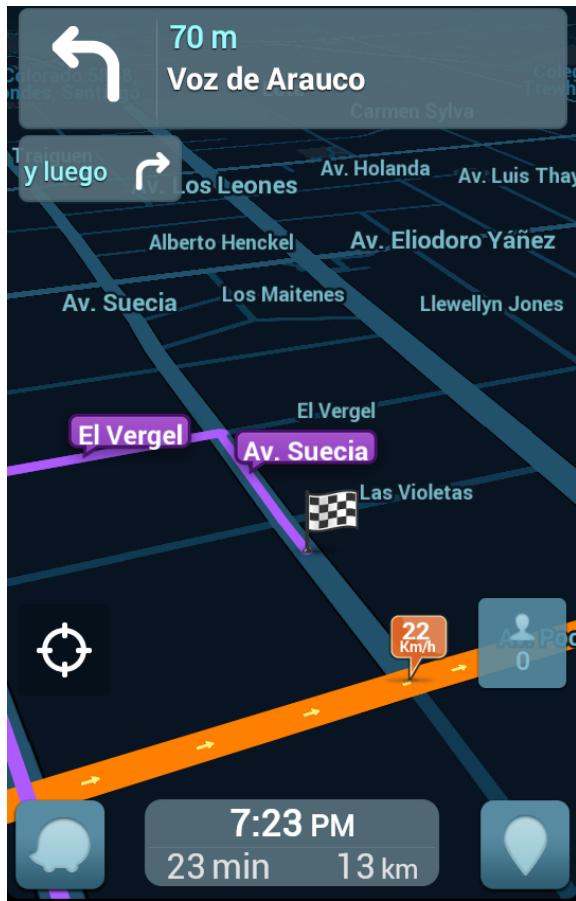


Figura 12: Vista del punto de envío de las alertas por el dispositivo 1, en el dispositivo 2

En la figura 11, se muestran 3 envíos seguidos, realizados de alerta de tráfico detenido, desde un dispositivo móvil. Estas alertas fueron posibles enviarlas por la ubicación del dispositivo, que era cercana a la calle.

En la figura 20, desde otro dispositivo no vemos que haya tráfico en la misma dirección.

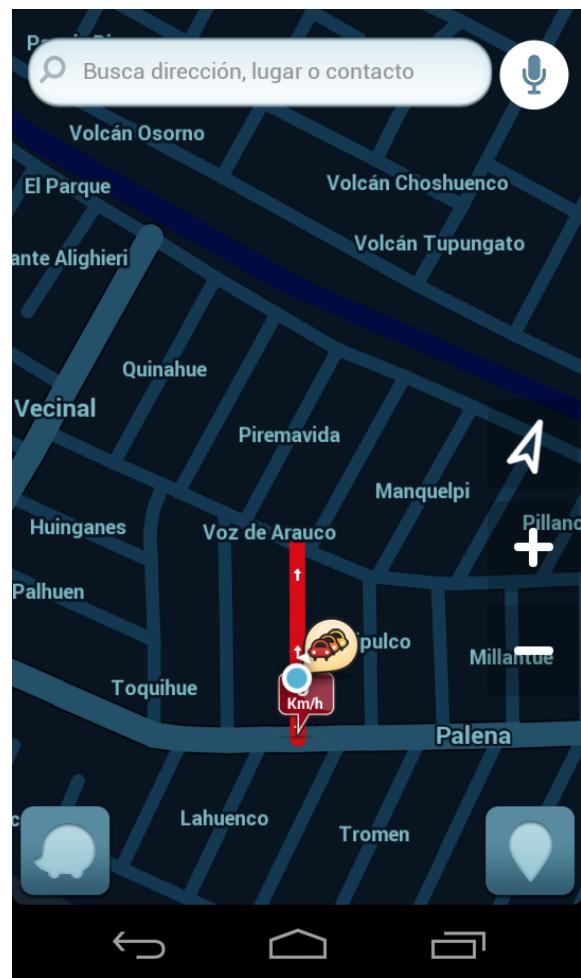


Figura 13: Envíos de alerta en el dispositivo 2



Figura 14: Vista del punto de envío de las alertas por el dispositivo 2, en el dispositivo 1

En caso contrario, usando un usuario frecuente en Waze se hizo la alerta en una calle cercana a la ubicación del dispositivo con ese usuario y con el otro dispositivo si se pudo ver la alerta.

En Waze, los usuarios ganan puntos a medida que utilizan la aplicación, a medida que utilizan el mapa, conducen, generan acciones con la aplicación se gana puntaje. [1]

Para este ejemplo, el usuario frecuente tiene el ranking de Waze Loyalty, los cuales son los usuarios pertenecientes al 1

Con estas pruebas se comprueba que los desarrolladores de Waze corrigieron la vulnerabilidad encontrada por los estudiantes israelíes.

7. Función Indent (Direcciones waze://)

Como se dijo en la etapa de descompilado, Waze utiliza direcciones únicas para sus consultas y esta dirección puede ser utilizada por otras aplicaciones para ejecutar Waze con algún comando.

Analizando los ejemplos para desarrolladores, una forma de invocar Waze es con la dirección “waze://q=” y concatenando la búsqueda y esto genera una búsqueda dentro de Waze tal como si el usuario la buscara.

Por tanto, se probó en un sitio HTML que invocara la url “waze://q=Facultad de Ingeniería UDP, Chile Santiago” (<http://output.jsbin.com/telecozazi>) para que ejecutara Waze con esa búsqueda.



Figura 15: Sitio HTML que invoca url waze://

Al abrir el enlace en el navegador de Android se ejecutó Waze generando la búsqueda dicha.

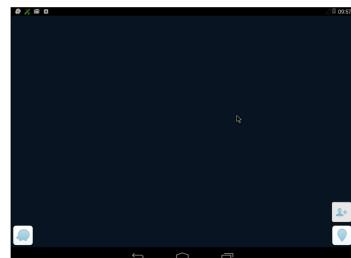


Figura 16: Ejecución de Waze al abrir la url Waze:// en la imagen anterior

El mapa, no se muestra debido a que Waze de alguna manera detecta los programas que establecen ubicaciones que no corresponden, pero si realizó la acción solicitada.

8. Modificación y eliminación de paquetes

No fue posible modificar o alterar la interacción con el servidor de Waze dado que el protocolo está ofuscado y no podemos realizar auditoría al respecto.

De todas formas, se alteró el comportamiento de la conectividad de Waze utilizando IPTables para ver su comportamiento.

Con IPTables, al momento de realizar una petición de rutas en Waze (colocando una dirección a la cual queremos llegar), añadimos una regla que votara todo tráfico proveniente que tuviera la IP con la cual nuestra APP se había estado comunicando:

```
iptables -A INPUT -s 65.55.44.100 -j DROP
```

Estos argumentos en IPTables indican que debe ser añadida una regla para los paquetes de entrada (INPUT) desde el origen (-s) 65.55.33.100 con la acción DROP (votar el paquete).

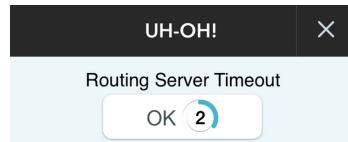


Figura 17: Error de conexión a internet

Waze al no recibir los paquetes de respuesta de la solicitud de ruteo nos advierte que esperó al servidor de rutas pero este no le respondió [Figura 17].

De manera similar, se hizo para cuando se hizo una alerta de tráfico en Waze, al seleccionar que había mucho tráfico en nuestra ubicación, se realizó el mismo comando con IPTables.



Figura 18: Error generado al ser modificado previamente un servidor DNS en el dispositivo

Waze nos advierte que hay un error, pero esta vez no nos indica de qué se trata. Podría ser una excepción no rescatada dentro del código (Figura[18]).

9. Funcionalidad en maquina virtual

En VirtualBox, aunque se instalamos más de una aplicación como fake gps (figura 17):

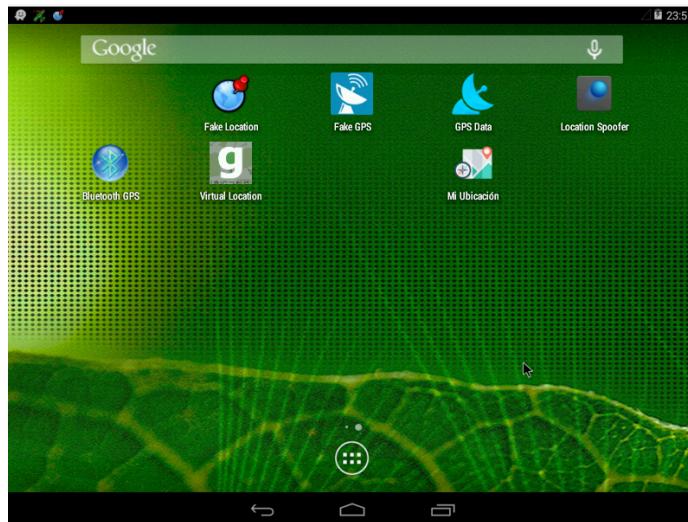


Figura 19: Aplicaciones instaladas para fakegps

Waze no mostró el mapa, como se ve en la figura 20:

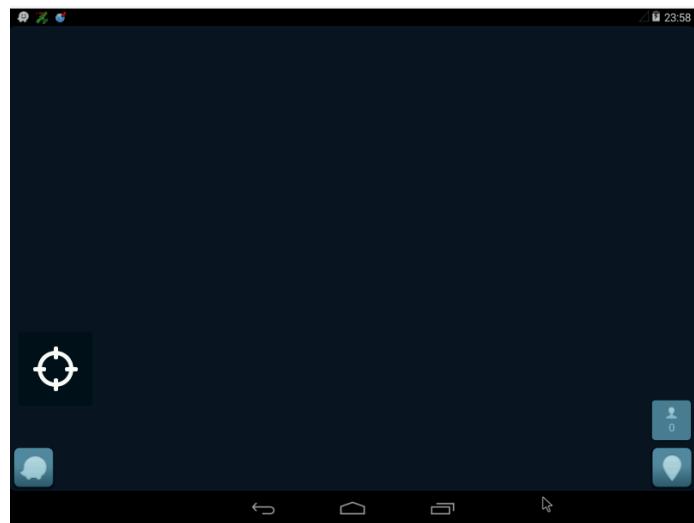


Figura 20: Waze no muestra mapa en maquina virtual (VirtualBox)

En la máquina virtual al enviar una alerta, no aparece el mensaje de alerta enviada, figura 19:

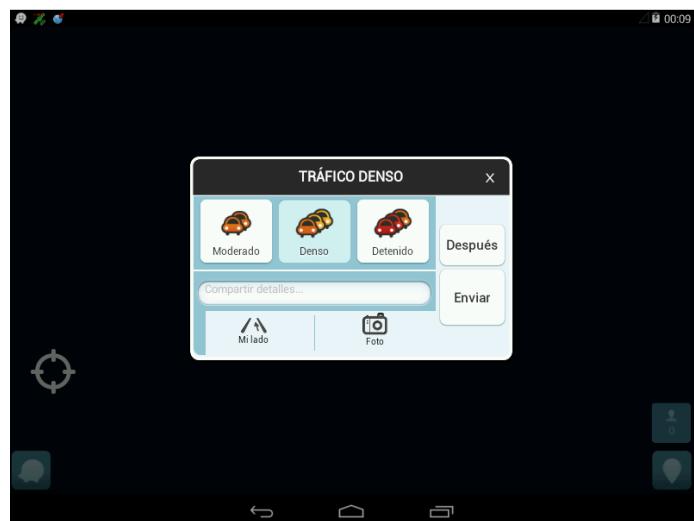


Figura 21: En la maquina virtual no se muestra el mensaje de alerta enviada

Como es el caso de enviar una alerta desde un celular:

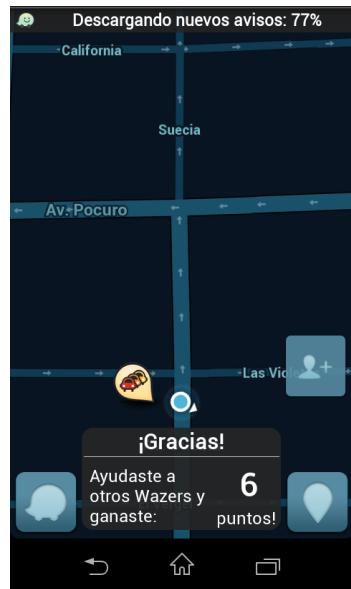


Figura 22: Mensaje de alerta enviada mostrada en un celular

Otra opción descartando VirtualBox, fue probar con el emulador de android BlueStacks:

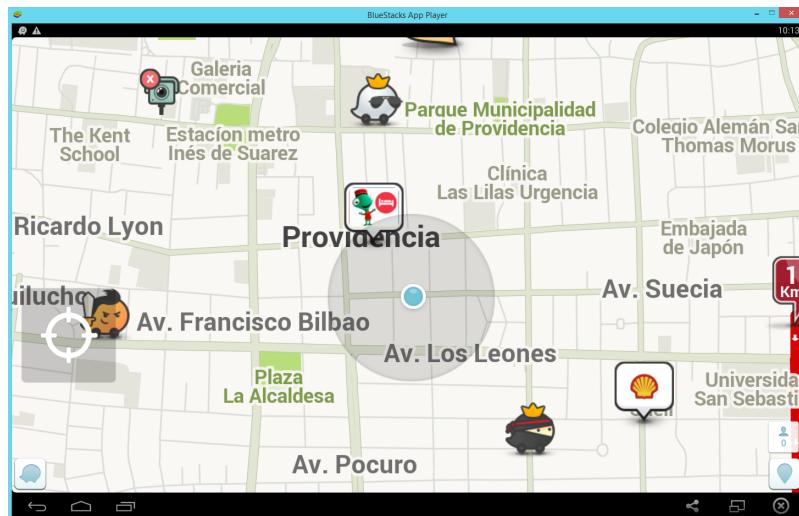


Figura 23: Waze en el emulador de Android Bluestacks

En este caso si se muestra el mapa, y también es posible enviar una alerta:



Figura 24: En Bluestack, Waze si muestra el mapa como en un celular

y luego de enviarla aparece en el celular:

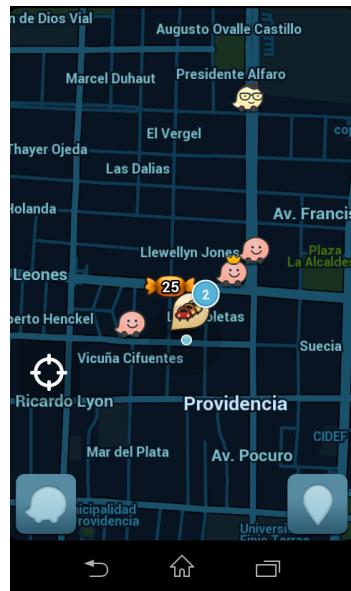


Figura 25: Alerta enviada usando Bluestacks aparece se muestra en un celular

El problema que encontramos con BlueStacks fue que no es posible acceder a los archivos de la aplicación desde afuera. Pero si en la máquina virtual de Android:

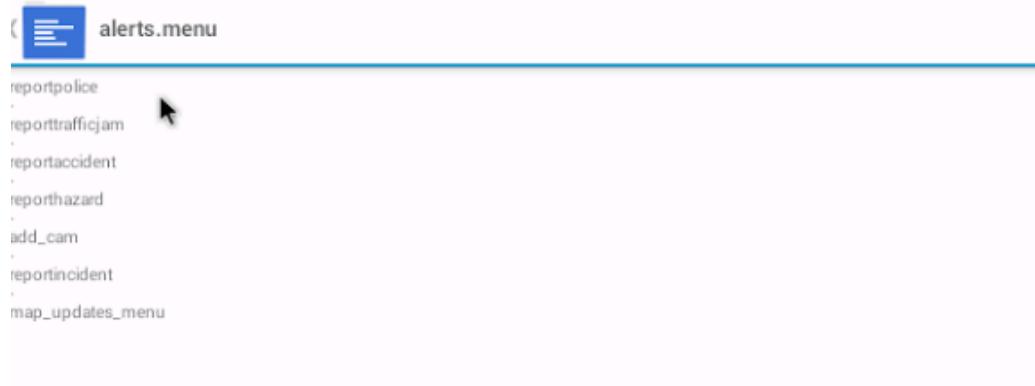
| | | | | | | | | | |
|--|--|-----------|-----|----------|---|--|-----------|-----|----------|
|  lang.chinese | | 100.57 KB | -rw | 6/7/2015 |  lang.conf | | 42.00 B | -rw | 6/7/2015 |
|  lang.eng | | 6.74 KB | -rw | 6/7/2015 |  lang.heb | | 130.28 KB | -rw | 6/7/2015 |
|  map_updates.menu | | 42.00 B | -rw | 6/7/2015 |  more.menu | | 45.00 B | -rw | 6/7/2015 |
|  prompt_list.txt | | 526.00 B | -rw | 6/7/2015 |  prompts_conf.buf | | 2.33 KB | -rw | 6/7/2015 |
|  quick.menu | | 501.00 B | -rw | 6/7/2015 |  recommend_50.menu | | 90.00 B | -rw | 6/7/2015 |
|  recommend.menu | | 77.00 B | -rw | 6/7/2015 |  report_list_nopolice.menu | | 106.00 B | -rw | 6/7/2015 |
|  report_list.menu | | 127.00 B | -rw | 6/7/2015 |  search_conf | | 6.98 KB | -rw | 6/7/2015 |
|  search.menu | | 71.00 B | -rw | 6/7/2015 |  settings.menu | | 49.00 B | -rw | 6/7/2015 |
|  sprites | | | | | | | | | |

Figura 26: Archivos de Waze dentro de la maquina virtual

| | | | | | | | |
|--|-----------|-----|----------|---|-----------|-----|----------|
|  maps | 3 item | drw | 6/7/2015 |  skins | 1 item | drw | 6/7/2015 |
|  sound | 3 item | drw | 6/7/2015 |  tts | 2 item | drw | 6/7/2015 |
|  about.html | 21.23 KB | -rw | 6/7/2015 |  adios.html | 11.51 KB | -rw | 6/7/2015 |
|  alerts_nopolice.menu | 97.00 B | -rw | 6/7/2015 |  alerts_offline.menu | 40.00 B | -rw | 6/7/2015 |
|  alerts.menu | 112.00 B | -rw | 6/7/2015 |  asr_cmd | 4.30 KB | -rw | 6/7/2015 |
|  asr_cmd_alerter | 395.00 B | -rw | 6/7/2015 |  cacert.pem | 248.54 KB | -rw | 6/7/2015 |
|  crash_log.txt | 0.00 B | -rw | 6/7/2015 |  help.menu | 6.00 B | -rw | 6/7/2015 |
|  lang.chinese | 100.57 KB | -rw | 6/7/2015 |  lang.conf | 42.00 B | -rw | 6/7/2015 |

Figura 27: Más archivos de Waze dentro de la máquina virtual

Dentro de los archivos no se encontró nada configurable. El archivo lang.conf solo mantenía los nombres de los lenguajes disponibles del software. [Figura 26 y 27] En los archivos menús al parecer mantiene llamadas a funciones de Waze, ya que solo mantiene líneas con nombres de acciones aparentemente. [Figura 26]



```

alerts.menu
reportpolice
reporttrafficjam
reportaccident
reporthazard
add_cam
reportincident
map_updates_menu

```

Figura 28: Contenido archivo alerts.menu

De todas formas, intentamos modificar para ver si esto tiene incidencias en la aplicación, modificando el mismo archivo de alertas con uno similar:

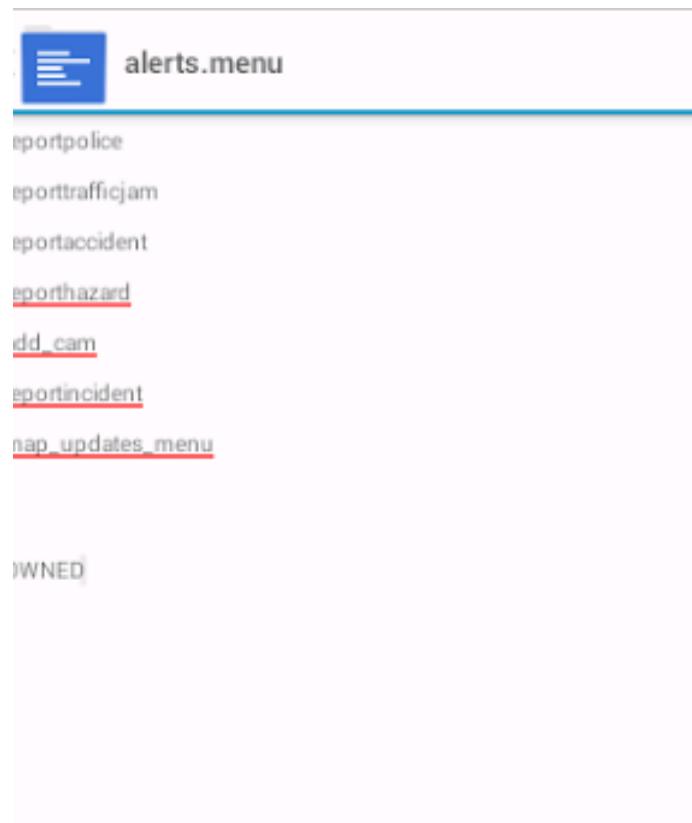


Figura 29: Archivo alerts.menu modificado

Pero esto no tuvo repercusiones en la aplicación:

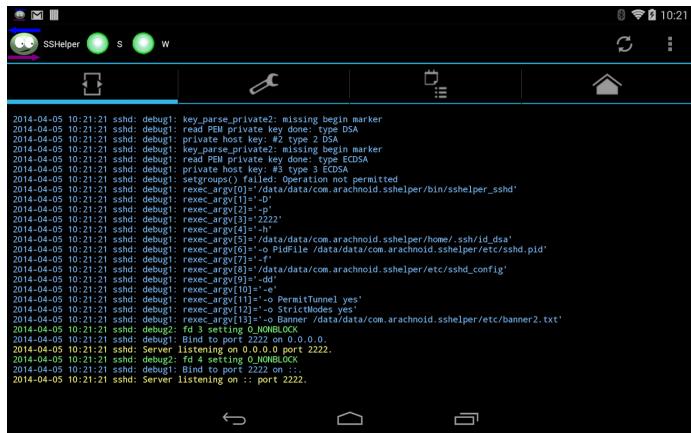


Figura 30: Menu de alertas no es afectado por la modificación del archivo alerts.menu

Se volvió a verificar la aplicación y esta había reemplazado el archivo alerts.menu con el archivo original, por tanto, podemos deducir que la aplicación genera estos ficheros al iniciar.

Se volvió a verificar la aplicación y esta había reemplazado el archivo alerts.menu con el archivo original, por tanto, podemos deducir que la aplicación genera estos ficheros al iniciar.

Para verificar si la aplicación se ve alterada al modificar estos mismos archivos mientras corre, se instaló SSHHelper[21] para poder entrar mediante SSH al dispositivo y modificar los archivos mientras corría la aplicación.



```

2014-04-05 10:21:21 sshd: debug1: key_parse_private2: missing begin marker
2014-04-05 10:21:21 sshd: debug1: read PEM private key done: Type DSA
2014-04-05 10:21:21 sshd: debug1: private host key #2 type 2 DSA
2014-04-05 10:21:21 sshd: debug1: key_parse_private2: missing begin marker
2014-04-05 10:21:21 sshd: debug1: read PEM private key done: Type ECDSA
2014-04-05 10:21:21 sshd: debug1: private host key #3 type 3 ECDSA
2014-04-05 10:21:21 sshd: debug1: setgroups() failed: Operation not permitted
2014-04-05 10:21:21 sshd: debug1: rexec_argv[0]='/usr/local/etc/ssh/sshd'
2014-04-05 10:21:21 sshd: debug1: reexec_argv[1]='-d'
2014-04-05 10:21:21 sshd: debug1: reexec_argv[2]='-p'
2014-04-05 10:21:21 sshd: debug1: reexec_argv[3]='-h'
2014-04-05 10:21:21 sshd: debug1: reexec_argv[4]='-h'
2014-04-05 10:21:21 sshd: debug1: reexec_argv[5]='/data/data/com.arachnoid.sshelper/home/.ssh/id_dsa'
2014-04-05 10:21:21 sshd: debug1: reexec_argv[6]='-o PidFile /data/data/com.arachnoid.sshelper/etc/sshd.pid'
2014-04-05 10:21:21 sshd: debug1: reexec_argv[7]='/data/data/com.arachnoid.sshelper/etc/sshd_config'
2014-04-05 10:21:21 sshd: debug1: reexec_argv[8]='-d'
2014-04-05 10:21:21 sshd: debug1: reexec_argv[9]='-d'
2014-04-05 10:21:21 sshd: debug1: reexec_argv[10]='-o PermitTunnel yes'
2014-04-05 10:21:21 sshd: debug1: reexec_argv[11]='-o StrictModes yes'
2014-04-05 10:21:21 sshd: debug1: reexec_argv[12]='-o Banner /data/data/com.arachnoid.sshelper/etc/banner2.txt'
2014-04-05 10:21:21 sshd: debug1: Bind to port 2222 on 0.0.0.0.
2014-04-05 10:21:21 sshd: Server listening on 0.0.0.0 port 2222.
2014-04-05 10:21:21 sshd: debug1: reexec_argv[13]='-o UserKnownHostsFile /data/data/com.arachnoid.sshelper/etc/known_hosts'
2014-04-05 10:21:21 sshd: debug1: Bind to port 2222 on ::.
2014-04-05 10:21:21 sshd: Server listening on :: port 2222.

```

Figura 31: Modificación del archivo menu usando SSHelper

Al modificar el archivo de alertas mientras la aplicación corría, esta mostró igualmente el menú de alertas sin problemas y el fichero se reescribió.

10. Conclusiones

Waze resulta ser una aplicación muy robusta y con buenas prácticas en su codificación.

Las buenas prácticas de seguridad deben ser un punto fuerte en el desarrollo de cualquier tipo de software y más aún cuando deben ser programados en lenguajes descompilables como Java, donde no se tiene otra opción. Si no se toman medidas en cuanto a la comunicación de la aplicación y técnicas de ofuscación es muy probable que usuarios con malas intenciones generen aplicativos que roben información o alteren el flujo normal de la misma.

11. Bibliografía

- ProGuard <http://developer.android.com/tools/help/proguard.html>
- Smali2Java <https://github.com/darkguy2008/smali2java>
- Custom Ident for Launch Activity from other app <http://developer.android.com/training/basics/intents/filters.html>
- Ataque de estudiantes Israelíes <http://goo.gl/L0y0B3>
- ProGuard <http://developer.android.com/tools/help/proguard.html>
- Proyecto Android x86 <http://www.android-x86.org/>