# Framework multiplataforma para automatizar la intercepción de tráfico en aplicaciones móviles Android

## Cross-platform framework to automate traffic interception in Android mobile applications

Presentación: 21/08/2024

**Fabián GIBELLINI**
Universidad Tecnológica Nacional – Facultad Regional Córdoba
fabiangibellini@gmail.com

**Leonardo CICERI**
Universidad Tecnológica Nacional – Facultad Regional Córdoba
leonardociceri@gmail.com

**Ileana BARRIONUEVO**
Universidad Tecnológica Nacional – Facultad Regional Córdoba
ilebarrionuevo@gmail.com

**Germán PARISI**
Universidad Tecnológica Nacional – Facultad Regional Córdoba
germannparisi@gmail.com

**Milagros ZEA CARDENAS**
Universidad Tecnológica Nacional – Facultad Regional Córdoba
milyzc@gmail.com

**Analía RUHL**
Universidad Tecnológica Nacional – Facultad Regional Córdoba
analialorenaruhl@gmail.com

**Marcelo AUQUER**
Universidad Tecnológica Nacional – Facultad Regional Córdoba
marcelo.auquer@gmail.com

**Federico BERTOLA**
Universidad Tecnológica Nacional – Facultad Regional Córdoba
fedebertola@gmail.com

**Juliana NOTRENI**
Universidad Tecnológica Nacional – Facultad Regional Córdoba
julinotreni@gmail.com

**Sergio QUINTEROS**
Universidad Tecnológica Nacional – Facultad Regional Córdoba
ser.quinteros@gmail.com

**Ignacio SÁNCHEZ BALZARETTI**
Universidad Tecnológica Nacional – Facultad Regional Córdoba
ignaciojsb@gmail.com

## Resumen

El análisis dinámico de aplicaciones móviles requiere de destinar esfuerzo a la inspección de tráfico web entre aplicación y servidor para encontrar vulnerabilidades y comprender el funcionamiento de los flujos de la aplicación. Para realizar este proceso, se debe tener en cuenta que las aplicaciones están desarrolladas en distintos lenguajes de programación, con distintos frameworks, para distintas plataformas, y además poseen ciertas protecciones para evitar que un intermediario intercepte esa comunicación entre cliente y servidor. Es en este punto donde los pentesters se encuentran con algunas dificultades que deben sortear manualmente y con algunas herramientas automatizadas, pero éstas no cubren todas las expectativas ya que comúnmente requieren de pasos adicionales y, algunas veces, extensos, para lograr que la intercepción sea exitosa. Es por ello que el proyecto pretende reunir tanto técnicas manuales como automatizadas que sirvan como abanico de posibilidades para el pentester en una sola herramienta.

**Palabras clave**: Seguridad, Test de penetración, Intercepción, Burp Suite, Android .

## Abstract

Dynamic analysis of mobile applications requires investing effort to web traffic inspection between the application and the server to find vulnerabilities and understand how the application flows work. To carry out this process, it must be taken into account that the applications are developed in different programming languages, with different frameworks, to work in several platforms, and also have certain protections to prevent an intermediary from intercepting that communication between the client and the server. It is at this point where pentesters face some obstacles that they must overcome manually and with automated tools, but these do not meet all expectations since they commonly require additional and sometimes extensive steps to achieve successful interception. That is why the project aims to bring together both manual and automated techniques that serve as a range of possibilities for the pentester in a single tool.

**Keywords**: Security, Pentesting, Interception, Burp Suite, Android.

## Introducción

Entre los distintos ataques a las aplicaciones móviles, el proyecto se ha enfocado en uno en particular: es el ataque de tipo man-in-the-middle, que explota el hecho de que el servidor HTTPS envía un certificado con su clave pública al navegador web. Si este certificado no es confiable, toda la comunicación es vulnerable (Callegati et ál, 2009) ya que reemplaza el certificado original que autentica al servidor HTTPS servidor con un certificado modificado. El ataque tiene éxito si el usuario se niega a verificar el certificado cuando el navegador envía una notificación de advertencia. Todo esto corresponde al tipo de análisis activo, en donde el investigador intercepta activamente toda la comunicación de red, pudiendo analizar, interactuar, modificar los datos en ese preciso instante. Aquí se utiliza generalmente un proxy web y todas las llamadas hechas y recibidas por la aplicación y el servidor pasan a través del mismo (Gupta, 2023).

El presente proyecto hace hincapié en facilitar mediante una sola herramienta, las técnicas de análisis activo de red, que bien pueden servir para un análisis pasivo.

## Desarrollo

Después de una categorización profunda de las técnicas y herramientas actuales (Gibellini et ál, 2023), se ha decidido la incorporación de las siguientes técnicas y herramientas que presentan las siguientes particularidades:

- BurpSuite requiere de la descarga del certificado desde su propia interfaz, la modificación de la extensión de dicho certificado, y la posterior instalación manual en el dispositivo (PortSwigger, s.f.).

- Frida: implica la instalación manual y ejecución de un agente en dispositivo rooteado, y luego correr el script para saltar la protección de SSL pinning. Esta herramienta posee cobertura respecto al agente servidor, en dispositivos rooteados únicamente. Para dispositivos no rooteados, posee cobertura únicamente a través de la inyección de un gadget en la aplicación objetivo. Funciona tanto en iOS como en Android (Frida, s.f.).

- Objection: requiere modificar la aplicación, generar otra a la que se le inyecta un gadget de Frida, e instalarla en el dispositivo. Funciona con dispositivos rooteados o no rooteados. Posee cobertura tanto en Android como iOS (SensePost, s.f.).

- reFlutter: también implica modificar la aplicación original para generar otra, y además el puerto de proxy ya viene determinado, por lo cual deberá configurar el proxy en escucha en el mismo puerto que indica reFlutter, y en el dispositivo móvil también. Su cobertura abarca dispositivos rooteados y no rooteados (PtSwarm, s.f.).

- patch-apk: script de unificación de splits de apk, requiere de Linux (NickstaDB, s.f.).

La ejecución de estas herramientas posee algunas carencias o fallas en Windows. Cabe destacar que las mismas son compatibles tanto con dispositivos móviles reales como con emuladores.

En el proyecto se ha desarrollado un menú integrado de herramientas como adb, apktool y objection, automatizando: obtención de apk desde el dispositivo; obtención e instalación del certificado de proxy en el dispositivo; decompilado de la aplicación; modificación de las configuraciones de seguridad en el archivo network_security_config.xml; compilación y firma de la aplicación modificada, instalación de la aplicación modificada en el dispositivo; interacción con objection para conectar con la sesión en el dispositivo.

La arquitectura del proyecto fue planteada como observamos en la Fig. 1:

```
project_root/
│
├── resources/               # Carpeta para almacenar archivos intermedios
│   └── (archivos .cer, .apk, etc.)
│
├── src/                     # Carpeta que contiene todo el código fuente
│   ├── __init__.py          # Archivo para declarar que esta carpeta es un paquete
│   ├── colores.py           # Archivo para códigos de color ANSI
│   ├── consola.py           # Funciones relacionadas con la consola
│   ├── adb_utils.py         # Funciones relacionadas con ADB
│   ├── objection.py         # Funciones relacionadas con Objection
│   ├── validaciones.py      # Funciones relacionadas con las validaciones en el proyecto
│   └── menu.py              # Funciones relacionadas con el menú
│
├── tools/                   # Carpeta para scripts y archivos relacionados con la instalación y configuración de herramientas
│   ├── setup.py             # Script de configuración para instalar las dependencias del proyecto y las herramientas necesarias
│   ├── install_tools.sh     # Script de bash para instalar herramientas externas como adb, OpenSSL, y Objection
│
├── main.py                  # Archivo principal que ejecuta el programa
│
├── requirements.txt         # Archivo que lista las dependencias del proyecto para ser instaladas con pip
├── README.md                # Archivo de documentación que explica el propósito del proyecto y cómo usarlo
└── .gitignore               # Archivo para especificar qué archivos o carpetas deben ser ignorados por Git
```

Figura 1. Arquitectura del proyecto.

Se avanzó en la detección del dispositivo Android conectado mediante USB debugging, determinando si es root o no root. Se ofrece el siguiente menú, que agrupa en cada opción otro abanico de posibilidades para poder interactuar con los paquetes de apk y las integraciones (Fig. 2):


Figura 2. Menú de la herramienta.

Respecto a la gestión de certificados, las opciones en las que se avanzó mayormente se refieren a:

- Opción 1: almacenar un certificado (.der) directamente en el dispositivo.
- Opción 2: verificar y/o eliminar certificados instalados en el dispositivo móvil.
- Opción 3: facilita la configuración del proxy, en este caso BurpSuite, de manera fácil y rápida, ya sea un solo dispositivo o en múltiples dispositivos conectados.

Se dispone de la opción de “Gestión de APK”, la cual permite realizar diversas acciones con las APK:

- Opción 1: Permite instalar una APK en uno de los dispositivos seleccionados.
- Opción 2: A partir de esta opción se listan todas las APK instaladas en el dispositivo, y luego se permite seleccionar aquella que se quiera descargar.
- Opción 3: Permite eliminar una APK que se encuentre instalada en el dispositivo.

Se incorpora también la posibilidad de detectar si existen splits en el dispositivo, para luego poder unificarlos mediante la herramienta *patch-apk* (Fig. 3):


Figura 3. Detección de splits y unificación.

Finalmente, se obtiene un solo archivo apk con el cual se facilita el trabajo de intercepción de tráfico posterior.

Además, se ha incorporado la herramienta *objection* para inyección del gadget de *Frida* que permite la interacción entre comandos Linux y la propia aplicación, pudiendo no sólo romper el mecanismo de SSL pinning, sino también interceptar las funciones del código de la APK en tiempo de ejecución.

Frida ha sido una de las mejores incorporaciones para este proyecto, ya que su compatibilidad con las plataformas y dispositivos permite al investigador desarrollar diversos scripts que luego se ejecutarán para manipular las aplicaciones. El script más utilizado por su efectividad es llamado "*frida_multiple_unpinning.js*". Cubre diferentes funciones posiblemente utilizadas por los desarrolladores para proteger el tráfico de la aplicación, pero con este script se busca deshabilitarlas. Al incorporar Frida y objection como herramientas base de nuestro proyecto, el investigador solamente debe decidir cuál desea utilizar, y se ejecutará dicho requerimiento sin necesidad de realizar configuraciones extras.

Ejemplo del script "*frida_multiple_unpinning.js*" (Fig. 4):

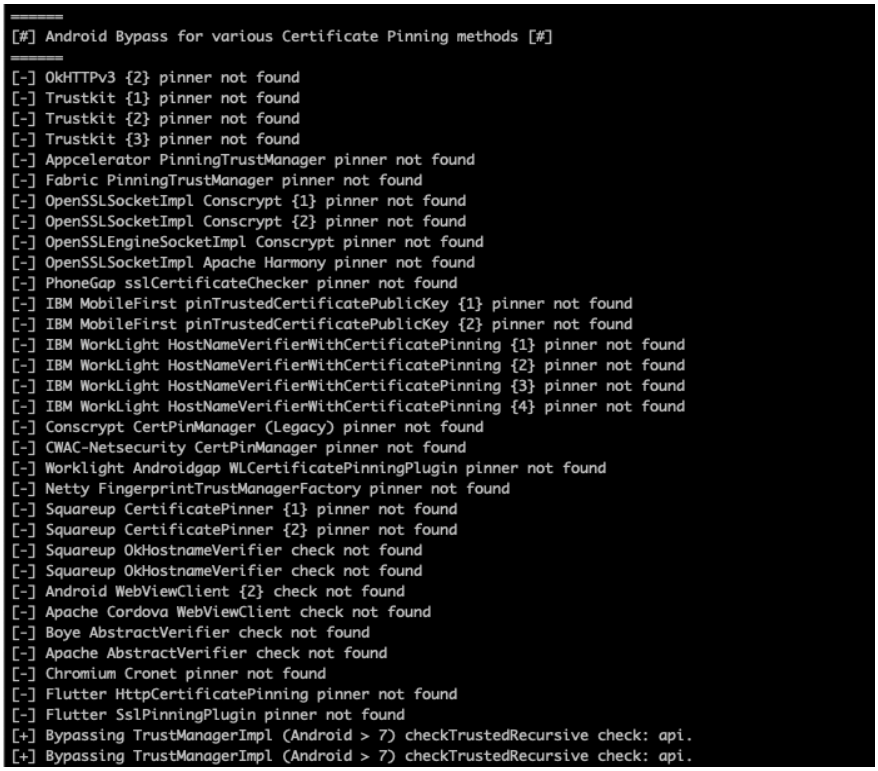


Figura 4. Chequeo de presencia de métodos de SSL pinning.

Notar que las dos últimas líneas hacen referencia al método encontrado y que luego es deshabilitado para intercepción del dominio en la aplicación. El investigador, posterior a este paso, únicamente debe seguir interactuando con la aplicación e interceptando el tráfico en un web proxy (BurpSuite) sin inconvenientes, logrando así el objetivo del proyecto.

## Conclusiones

Como se ha mencionado se pretende crear una herramienta multiplataforma, es decir que funcione tanto en sistemas operativos Windows (WSL) y Linux/GNU, por lo que seguir con Python ha sido la mejor opción. Por otro lado, se habían realizado búsquedas y estudios sobre aplicaciones iOS, pero se decidió realizar un enfoque sobre aplicaciones en Android, ya que se requiere la utilización de recursos de los cuales no se dispone en el proyecto, como la adquisición de un dispositivo iOS. Sin embargo, varias de las herramientas incorporadas en el proyecto ya brindan soporte para intercepción de tráfico en iOS, por lo que, de alguna manera, las aplicaciones para dicha plataforma quedan cubiertas.

La primera versión de este framework es capaz de automatizar las tareas que permitan interceptar una comunicación entre una aplicación móvil y su servidor, mediante Frida y objection. Posteriormente se

avanzará sobre reFlutter. El foco actual del equipo de desarrollo está puesto en mejoras de la detección de dispositivos, condiciones de la aplicación instalada, y mejoras en el script de instalación de componentes de acuerdo al kernel de Linux, de manera tal que también el proyecto funcione en WSL (Windows).

En resumen, la herramienta refleja un abanico de posibilidades para el investigador, facilitando su tarea y evitando inversiones de tiempo extra de instalación de herramientas por separado, configuraciones y corrección manual de errores en cada técnica que suele aplicar.

## Referencias


Callegati, F., Cerroni, W., & Ramilli, M. (2009). Man-in-the-middle attack to the HTTPS protocol. IEEE Security & Privacy, 7(1), 78-81. https://doi.org/10.1109/MSP.2009.12

Frida. (s.f.). FRIDA - Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers. Recuperado el 19 de abril de 2024, de https://frida.re/

Gibellini, F., Ciceri, L., Notreni, J., Parisi, G., Ruhl, A., Zea Cardenas, N., Auquer, M., Barrionuevo, I., Bertola, F., Quinteros, S., & Sanchez Balzaretti, I. (2023). Intercepción de tráfico en aplicaciones móviles. Memorias de las JAIIO, 9(8), 52-52. Recuperado de https://publicaciones.sadio.org.ar/index.php/JAIIO/article/view/756

Gupta, A. (2023). Learning pentesting for Android devices. Packt Publishing.

NickstaDB. (s.f.). patch-apk - App Bundle/Split APK Aware Patcher for Objection. Recuperado el 19 de abril de 2024, de https://github.com/NickstaDB/patch-apk

PortSwigger. (s.f.). Burp Suite Community Edition. Recuperado el 19 de abril de 2024, de https://portswigger.net/burp/communitydownload

PtSwarm. (s.f.). ReFlutter. Recuperado el 19 de abril de 2024, de https://github.com/ptswarm/reFlutter

SensePost. (s.f.). Objection - Runtime Mobile Exploration. Recuperado el 19 de abril de 2024, de https://github.com/sensepost/objection