UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Universidad Internacional de La Rioja
Máster universitario en Seguridad Informática

Análisis semiautomático de vulnerabilidades de redes en remoto

Trabajo Fin de Máster

presentado por: Muñoz Lloret, Adrián

Director/a: Lucas Simarro, José Luís

Ciudad: Málaga
Fecha: 18 de septiembre de 2018

Resumen

Este trabajo pretende ser un primer paso hacia la automatización de las auditorías de seguridad a gran escala. En él se pretende desarrollar una aplicación capaz de ser ejecutada por un dispositivo del tamaño de la palma de la mano, que guarde los resultados en un servidor externo, y se sirva del mismo para delegar las acciones para las que se requiera más capacidad de cómputo y no sea necesario estar online. Ésta aplicación auditará todas las redes WiFi, guardando sus principales datos en una pequeña base de datos con el objetivo de acceder a las mismas sin conocimiento previo de su clave. Para finalizar, se expondrán algunas ideas futuras de mejora que podrían hacer de ésta, una aplicación de interés en el mundo de la seguridad.

Palabras: Auditoría automática, seguridad informática, sistemas remotos, análisis de redes, Aircrack-ng.

Abstract

In this research we pretend to find and achieve a solution for the thick fog of missinformation that actually exist when we need to localizate someting indoors. We had a look into the possibles uses of the Bluetooth Low Energy (BLE) technology for this purpose. This will allow us to realize actions in order to localizate things in closed spaces, which GPS technology is not able to do. Nowadays there is not enough information about this type of use for Bluetooth technology. However it seems to be a very interesting topic to be studied, because it can provide solutions to real needs in this days. That's why we're going to realize a study in many enviroments using handy and cheap hardware like Raspberry Pi, PC and wearable devices such as the Xiaomi Mi Band 2 bracelet or smartphones. To sum up, we expose some ideas of improvement and innovation that may set an inflection point in the history of this technology, which is still uncharted.

Keywords: Bluetooth Low Energy, BLE, Bluetooth 4.0, Positioning, Geolocalization, Raspberry Pi, Location, Trilateration.

Índice general

Índice de figuras

Capítulo 1

Introducción

En la actualidad, la seguridad informática avanza a pasos agigantados, lo que supone un problema para quien intenta proteger su negocio: cada nueva vulnerabilidad y cada nuevo fallo es un nuevo vector de ataque para quienes quieran sacarle partido, pero también es un factor más a tener en cuenta por quienes se encargan de que la información siga estando segura. Esto hace que sean necesarias constantemente nuevas herramientas que permitan a un auditor enfocarse en las nuevas vulnerabilidades mientras que las antiguas se analizan de forma automática. No contentos con la cantidad de vulnerabilidades que hay que arreglar, también es necesario asegurarse de que los sistemas cercanos al nuestro no sean un vector de ataque que pueda usarse como pivote: es posible tener el sistema más seguro del mundo en una empresa, pero basta conque uno de sus empleados se conecte a la red privada virtual(Seid and Lespagnol, 1998) de la empresa desde una red WiFi (ValorTop, 2017) pública para que todo el capital invertido en seguridad halla sido insuficiente. Para ello se tienen en la red varios ejemplos de noticias en las que el acceso a la red por parte de un atacante ha significado una gran pérdida para una empresa, como el ataque que recibió un casino por culpa de una mala configuración de red en el termostato que usaba la pecera[1] o la noticia de Reino Unido que comenta que mediante la red WiFi, los atacantes eran capaces de controlar los trenes[2]. Ciertamente, esto no es más que un esbozo de la realidad; ya que estas cosas suelen ocurrir bastante más a menudo, pero a las empresas no les interesa hacerlo público para no dañar su imagen pública.

Es por esto por lo que se ha definido e implementado la aplicación sobre la que se desarrolla este proyecto: una aplicación que permite realizar pruebas de caja negra(Ostrand, 2002) en una empresa atacando el vector *red inalámbrica*, pero que sea fácilmente ampliable mediante nuevos módulos que añadan funcionalidad al sistema.

Lo dicho anteriormente no debería resultar una novedad para el lector: ya hay programas que permiten romper la seguridad de las redes inalámbricas y obtener la clave de las mismas[3]. Sin embargo a veces no es factible hacerlo debido al equipo que es necesario llevar a cuestas ya que, en la tecnología, el tamaño y la potencia suelen ser inversamente proporcionales.

En este proyecto se va un paso más allá, permitiendo llevar un pequeño dispositivo portátil, capaz de guardarse en un bolso, y con la autonomía suficiente para mantenerse atacando varias

[1] http://www.elmundo.es/tecnologia/2018/04/17/5ad4a14c46163f1f658b4630.html

[2] https://actualidad.rt.com/actualidad/271637-wifi-hackers-trenes-datos-espiar

[3] Entre otros, se encuentran wifite2 (https://github.com/derv82/wifite2) o la conocida suite Aircrack-ng (https://www.aircrack-ng.org).

horas; pero que se conectará a otro de mayor potencia que podría estar situado en cualquier parte del mundo. Será este segundo dispositivo el que le permitirá delegar toda la capacidad de cómputo.

1.1. Motivación

La motivación de este trabajo es el diseño de un dispositivo portátil, asequible y con escalabilidad suficiente para no tener limitaciones de potencia.

Hasta hace unos cinco años, la mayoría de redes utilizaban el sistema de cifrado WEP (más adelante se hablará más detenidamente sobre estos cifrados). Poco tiempo duró este cifrado sin que fuera fácilmente rompible por cualquier persona con un poco de curiosidad. Es por esto que aparecieron las redes con claves WPA, que usaban un cifrado seguro, lo cual representaba un problema para todo el que quisiera romperlas. Esto hizo que cambiara el vector de ataque: se buscaron contraseñas comunes con las que crear diccionarios, se estudiaban los algoritmos que usaban los proveedores para configurar sus routers. En el 2017 se descubrió una vulnerabilidad que permitía ver los datos transmitidos por una red con WPA2 (Shapiro, 2017) y en Abril de este mismo año se descubrió un nuevo ataque para las redes WPA protegidas con PSK (J. Steube, 2018), por lo que hace escasos meses, la Wifi Alliance[4] presentó el nuevo protocolo de cifrado WPA3 (Wifi Alliance, 2018).

Con este resumen de la historia de la seguridad WiFi no es difícil darse cuenta de que, con los años, seguirán apareciendo nuevos cifrados y nuevas formas de romperlos. Todo esto no es más que una motivación a seguir investigando en los vectores de ataque modernos, y a automatizar los antiguos.

1.2. Tecnologías a utilizar

La aplicación se desarrollará en Python[5] con el framework Django[6]. Esto permite definir una aplicación multiplataforma (aunque dependiente de paquetes que veremos más adelante). Además, Django brinda la comodidad de desarrollar la aplicación con paquetes, por lo que adquiere una alta flexibilidad a la hora de introducir mejoras o integrar la aplicación en otra más grande.

Se usarán también paquetes de programas como Aircrack-ng[7], nmap[8], etc. Esto decanta que la aplicación se ejecute principalmente en plataformas Unix, aunque Microsoft ya está implementando herramientas como estas en sus sistemas Windows.

La ejecución automática de tareas correrá bajo el framework Celery[9], el cual estará integrado con el propio Django.

La aplicación correrá sobre una Raspberry Pi[10] 3, un dispositivo portátil que ejecutará el

[4] https://www.wi-fi.org
[5] https://www.python.org
[6] https://www.djangoproject.com
[7] https://www.aircrack-ng.org
[8] https://nmap.org
[9] http://www.celeryproject.org
[10] https://www.raspberrypi.org

sistema operativo Kali[11] y tendrá conectados una batería externa y una antena usb Alfa. Esto nos permitirá tener la antena Alfa en modo promiscuo (rm-rf.es, 2009) mientras la Raspberry Pi está conectada a cualquier fuente de internet (como un móvil que ofrece internet vía WiFi) transmitiendo todo lo que obtiene.

Además, la potencia se la dará una máquina de sobremesa, conectada en red local con el cliente, que será la encargada de recibir los datos, procesarlos y mostrarlos usando una interfaz REST.

Para las pruebas se usará la herramienta *Postman* para lanzar las peticiones HTTP, mientras que la memoria se redactará en lenguaje LaTeX, usando el compilador online OverLeaf[12].

1.3. Estructura del documento

La memoria está estructurada de la siguiente manera:

- En el capítulo 2 se describe el *Estado del Arte* y la actualidad de los estudios relacionados con este trabajo. Se pretende mostrar una visión general acerca de como se encuentra en estos momentos la seguridad de redes WiFi y el software usado para analizarlas.
- En el capítulo 3 se comentarán los objetivos generales del proyecto, así como las metodologías de trabajo usadas.
- En el capítulo 4 se describirán todas las partes del proyecto, así como los problemas acontecidos antes de obtener un resultado final concluyente.
- El capítulo 5 se desarrollará sobre la ejecución y los resultados finales del trabajo.
- El capítulo 6 tratará sobre las conclusiones finales del proyecto y las posibles mejoras a realizar del mismo, de cara a futuros trabajos de ampliación sobre el mismo.
- Para finalizar, en la bibliografía se podrá encontrar el listado con todo el material consultado, tanto en formato digital como físico.

[11] https://www.kali.org
[12] https://v2.overleaf.com/

Capítulo 2

Estado del arte

Actualmente el mercado está lleno de herramientas (gratuitas y de pago) para análisis y auditorías de seguridad: Detectify[1], Alienvault[2], Zabbix[3], Snort[4], etc, pero cada una de ellas está enfocado en un tipo de auditoría distinto: Análisis de código seguro, análisis de redes y sistemas desde dentro, detectores de intrusos en red (Rowland, 2002)... Sin embargo no se encuentran en la web herramientas de análisis de caja negra para redes inalámbricas que actúen de forma automática.

Desde hace varios años, la herramienta por excelencia para obtener las claves de las redes WiFi se llama *Aircrack-ng*. Esta suite de herramientas nació en Febrero de 2006 como un fork del proyecto original *Aircrack*, y lo mejoró tanto que ya apenas se encuentra información en la web de su predecesor. Actualmente es un conjunto de herramientas que permiten monitorizar y atacar redes WiFi, testear interfaces de red (InformaticaESP, 2012) y romper las contraseñas de los cifrados WEP, WPA y WPA2.

Existe un módulo de *Aircrack-ng* llamado *Airoscript-ng* (Aircrack-ng.org, 2018b) que servía de interfaz de usuario para ejecutar ataques de una forma determinista (ConocimientosWeb, 2013) siguiendo siempre los mismos pasos, aunque no se acerca al automatismo que se pretende llegar con este estudio.

Debido a la constante evolución de la seguridad, la documentación científica queda obsoleta rápidamente, sin embargo encontramos aportes recientes interesantes que intentan encontrar nuevas vías de ataque a redes con cifrado WPA2 como (Acosta-López et al., 2018), que usa el software Linset (vk496, 2014) para emular la red que usa la víctima y hacer que se conecte al AP (Todo-Redes, 2018) falso.

2.1. Protocolos de cifrado WiFi

A continuación se hará un breve repaso de la historia de los distintos protocolos de cifrado utilizados (Wong, 2003).

[1] https://detectify.com
[2] https://www.alienvault.com
[3] https://www.zabbix.com
[4] https://www.snort.org

2.1.1. OSA

En el inicio de las redes WiFi, al igual que en el del propio Internet y el protocolo TCP/IP (Oracle, 2018), la funcionalidad primaba por encima de la seguridad.

Por esto, el primer sistema de conexión de redes WiFi se basaba en OSA (TechTarget, 2008). Este simple sistema aceptaba todas las peticiones de autenticación que le llegaban. Esto limitaba las medidas de seguridad que podía aplicar un usuario a un simple filtrado MAC (P.G. Bejerano, 2015), algo vulnerable ante un sencillo ataque de MAC Spoofing (1&1, 2017).

Esta falta de privacidad hizo que los usuarios con más conocimiento y/o preocupación por sus datos se negaran a usar WiFi y siguieran usando internet bajo cable.

2.1.2. WEP

Debido a las debilidades de OSA, en 1999 se presentó como estándar el protocolo de seguridad WEP (EcuRed, 2018b) (Wired Equivalent Privacy) que, como su nombre indica, pretendía ser el equivalente en seguridad a usar una red cableada. Sin embargo, dos años después ya existían exploits (Segu.Info, 2009) en internet (Wikipedia, 2018) que mostraban que este cifrado era fácil de romper desde un gran número de vectores de ataque (Lehembre, 2006), en 2003 la *Wifi Alliance* anunció que el protocolo WEP sería sustituido por el protocolo WPA (Alliance, 2003a).

La forma de romper este cifrado es capturando un gran número de IVs y usando métodos estadísticos como cuenta (Stubblefield et al., 2002). Lo único importante para un ataque satisfactorio es tener suficientes paquetes IVs.

En la imagen 2.1 se puede ver una ejecución de Aircrack-ng infructuosa, mientras que en la imagen 2.2 se ve cómo sí ha sido capaz de hallar la clave WEP definida en 2.5.4.

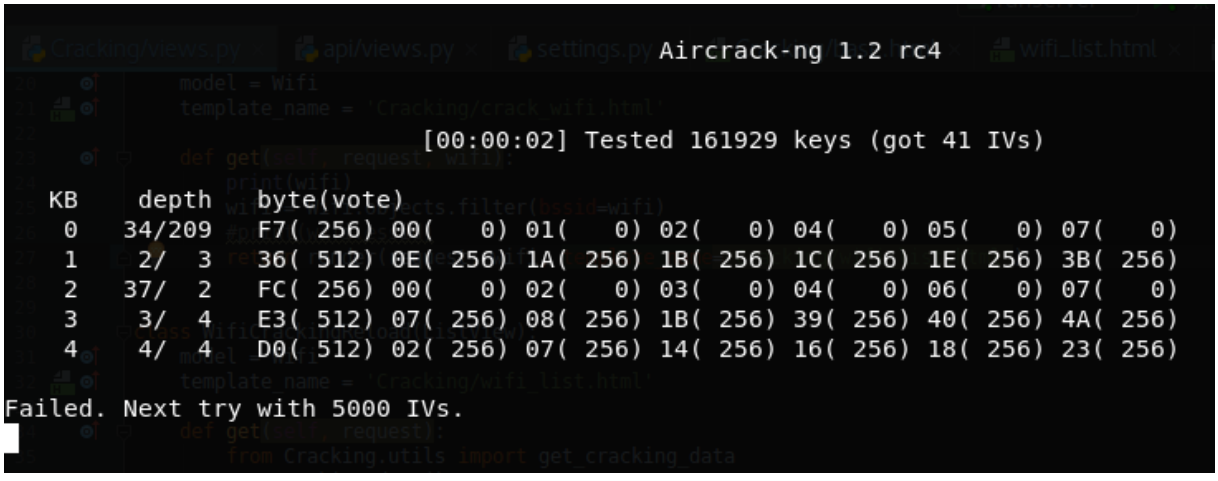


Figura 2.1: Ataque fallido de Aircrack-ng contra un cifrado WEP.

2.1.3. WPA

Debido a la demanda de mayor seguridad por parte de los usuarios, la WiFi Alliance adelantó la publicación del protocolo WPA (WiFi Protected Access) (Alliance, 2003b), el cual integraría el

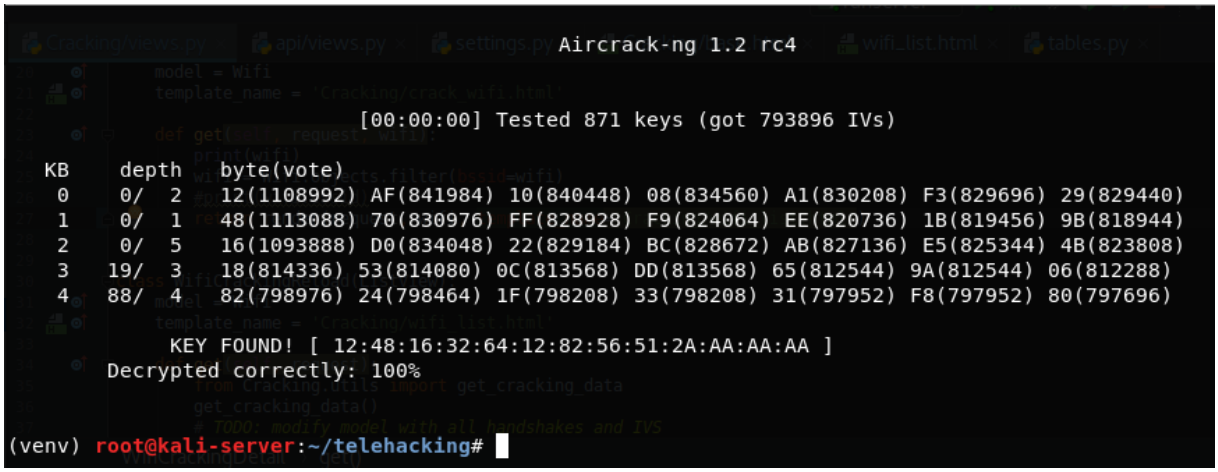


Figura 2.2: Ataque satisfactorio de Aircrack-ng contra un cifrado WEP.

Protocolo de Integridad de Clave Temporal (J. Snyder, R. Thayer, 2004) (TKIP), solucionando las debilidades del cifrado WEP.

Dentro del cifrado WPA coexisten varios modos de cifrado(Vila Ríos, 2017):

- WPA PSK (TKIP): (Temporal Key Integrity Protocol) Es el modo menos seguro de las redes que usan PSK (techopedia, 2018) (Pre-Shared Keys). Está diseñado para pequeñas redes domésticas o de oficina, donde los usuarios comparten una misma clave. Sin embargo, a día de hoy, TKIP no se considera seguro.

- WPA2 PSK (AES): Esta nueva versión con cifrado AES (E. Vinda, 2013) (Advanced Encryption Standard) salió solo un año después que la WPA original para dar una robustez mayor al cifrado.

- WPA2 PSK (TKIP+AES): Similar al anterior pero permite usar TKIP con dispositivos que no soportan AES.

- EAP: Extensible Authentication Protocol (Intel, 2018) es un protocolo de autenticación que deja la potestad de autenticar y autorizar a una entidad autenticadora (S. Noriega, 2015) (normalmente un servidor). Suele ser usado en empresas grandes y es compatible tanto con WPA como con WPA2

- RADIUS: (RemoteAuthentication Dial-In UserService (A. Crespo, 2017)) Se basa en la figura de un servidor centralizado de autenticación, encargado de autenticar las conexiones remotas de forma segura (Ulloa Santana and Fonseca Sánchez, 2012). Implementado tanto para WPA como para WPA2, en 2003 se hizo compatible con EAP

- WPS: Propuesto en 2007, Wifi Protected Setup (J. Lopez Arredondo, 2015) permitía a los usuarios conectarse a las redes sin tener que introducir la clave. Para ello se usaban diferentes técnicas como introducir un pin, pulsar un botón o usar NFC (J. Penalba, 2017).

  4 años más tarde se descubrió que un atacante podía recuperar el PIN en pocas horas usando ataques de fuerza bruta (J. García, 2018).

La forma de atacar WPA es muy diferente a la de WEP. En este caso no existe un método fijo que asegure la obtención de la clave, sino que es necesario obtener el *handshake* para poder atacar por fuerza bruta. Kali lleva incorporados varios diccionarios de fuerza bruta, como se puede ver en 2.3. Sin embargo, y ya que esto solo es una prueba de funcionamiento, se usará un diccionario propio, incluído en la aplicación, con el contenido mostrado en 2.4[5].

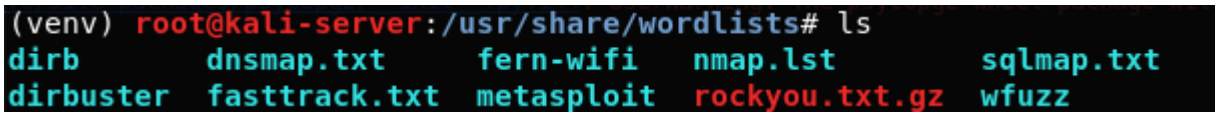


Figura 2.3: Ubicación de los diccionarios de Kali Linux.

```
00000000
1111
1111111
1234
12345
123456
54321
666666
7ujMko0admin
support
system
tech
ubnt
user
vizxv
xc3511
xmhdipc
zlxx.
Zte521
MasterSeguridadUNIR18
12481632641282565l2aaaaaaa
```

Figura 2.4: Diccionario usado por la aplicación.

2.2. Herramientas para atacar redes

En este apartado se profundizará en las herramientas actuales para atacar redes.

2.2.1. Aircrack-ng

Aircrack-ng es una suite de herramientas de auditoría de red basada en el antiguo paquete Aircrack. Este paquete está pensado para trabajar bajo Linux (aunque ya hay versiones para Windows, no se recomienda su uso por conflictos con los controladores (Aircrack-ng.org, 2007)) y solo con ciertos tipos de tarjetas de red, las cuales poseen modelos concretos de chipsets Atheros

[5]En realidad esta lista de palabras no es más que un head y un tail a la lista de contraseñas del archivo */usr/share/wordlists/metasploit/mirai_pass.txt*, seguido de las dos contraseñas usadas para la prueba.

o Ralink, los cuales permiten habilitar el *modo promiscuo*. Entre otras, Aircrack-ng cuenta con las siguientes herramientas:

- aircrack-ng: Le da nombre a la suite. Se encarga de descifrar la clave dados los IVs o el handshake.
- airodump-ng: El escáner de redes. Permite capturar los IVs y handshakes, guardándolos en ficheros *.cap*. Además, permite filtrar por essid, bssid, canal, etc. Es por todo esto que será la herramienta principal del proyecto.
- aireplay-ng: Herramienta que permite inyectar paquetes a los APs, con la intención de generar más tráfico para obtener un handshake o mejorar la velocidad de captura de IVs (Aircrack-ng.org, 2018a). Los ataques implementados son:
  - Deautenticación.
  - Falsa autenticación.
  - Selección del paquete a enviar.
  - Reinyección de ARP
  - Chopchop de Korek.
  - Ataque de fragmentación.
  - Ataque de caffe-latte.
  - Ataque de fragmentación orientado al cliente.
  - Modo de migración WPA.
  - Prueba de inyección.
- airmon-ng: Script que permite activar, desactivar y ver el estado del modo monitor en las tarjetas de red.

Además de estas, existen otras muchas herramientas como airbase, airdecap, airdecloak o airdriver (entre otras) con funcionalidades puntuales para mejorar la probabilidad de obtener claves. Es posible ver el total de las mismas en (Aircrack-ng, 2018).

2.2.2. Mergecap

Mergecap[6] es una herramienta poco conocida, pero ha sido de mucha utilidad en el proyecto. Su función consiste en unir archivos *.cap* en uno solo. Esto ha permitido que se puedan crear pequeños paquetes en el cliente y que estos sean transmitidos al servidor donde éste los unirá en un solo archivo. De esta manera no se tiene un gran archivo en la Raspberry Pi, la cual no tiene demasiados recursos para manejar este tipo de archivos.

El uso del comando es sencillo. Se usa el atributo *-F pcap* para especificar el tipo de archivo[7] y *-w final.cap* para especificar el archivo que contendrá la unión de los demás. En el proyecto, el comando usado es
`mergecap -F pcap -w /tmp/final.cap /tmp/final.cap /tmp/scan-01.cap`.

[6] https://www.wireshark.org/docs/man-pages/mergecap.html
[7] Aircrack-ng trabaja con archivos .cap y .pcap.

2.2.3. Reaver

Esta herramienta no será usada en el trabajo actual, sin embargo merece ser comentada por ser un vector de ataque alternativo al usado por la suite Aircrack-ng.

Reaver (Aked et al., 2012) explota el uso de la tecnología WPS cuando permite el acceso por pin, atacando por fuerza bruta a este código de 6 dígitos. Esto permite conseguir la contraseña de la red en pocas horas. En 2.5.

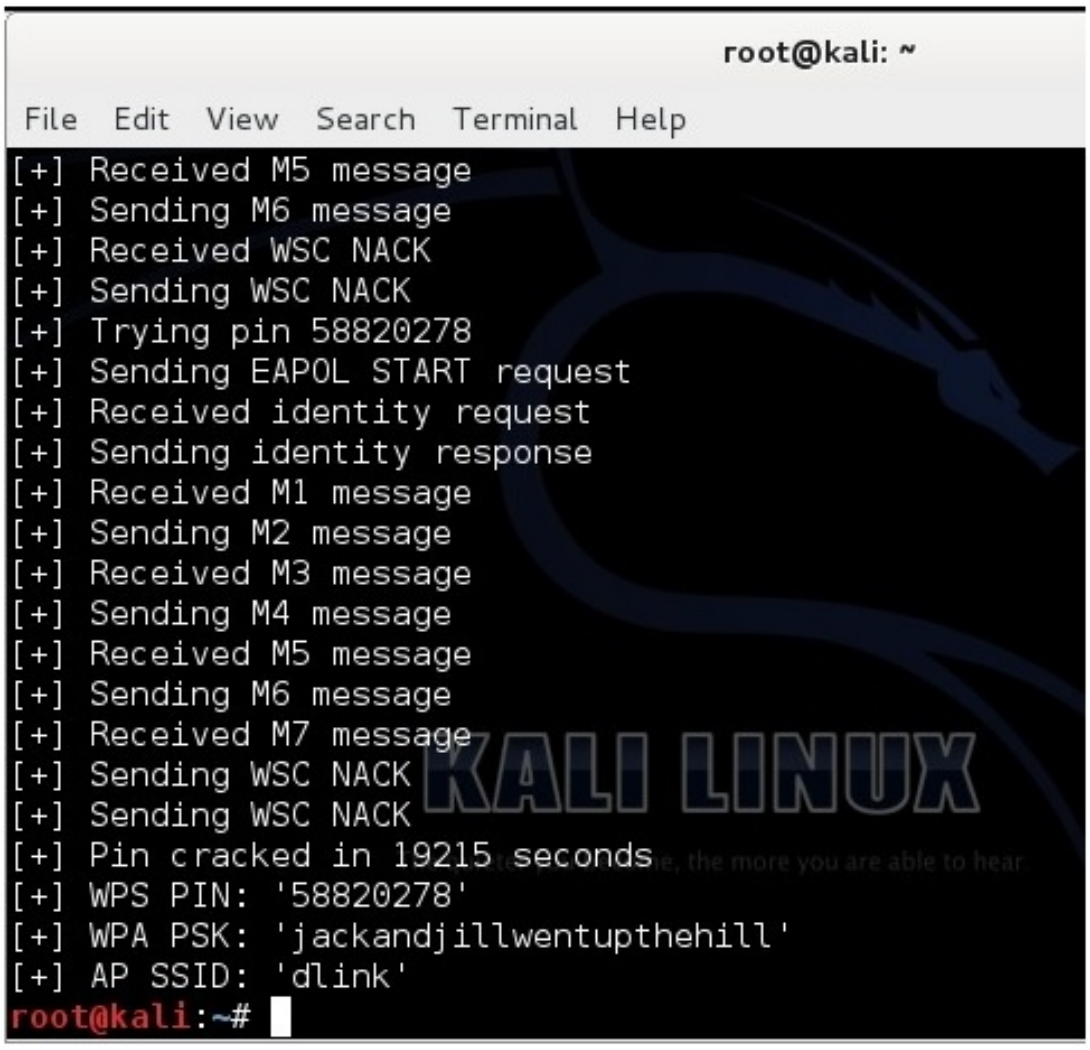


Figura 2.5: Captura de la obtención de una clave usando Reaver
https://www.wirelesshack.org/step-by-step-kali-linux-and-wireless-hacking-basics-reaver-part-4.html

2.2.4. Kali Linux

Kali Linux es un sistema operativo basado en Debian y orientado al pentesting. Anteriormente ya existieron otras distros similares como Wifiway, Wifislax, ParrotOS o Backtrack con finalidades similares (de hecho, Kali es una versión reescrita y actualizada de este último).

Kali fue desarrollado (y sigue siendo mantenido a día de hoy) por el equipo de Offensive

Security[8] el 13 de marzo del año 2013.

Actualmente, el sistema Kali cuenta con más de 600 herramientas, las cuales se instalan por defecto, entre las que se encuentran la suite Aircrack-ng, Metasploit[9], Wireshark[10] o nmap. Esto la convierte en una de las mejores herramientas para pentesters a día de hoy.

En la imagen 2.6 es posible ver una captura del menú, organizado en las diferentes fases del pentesting.

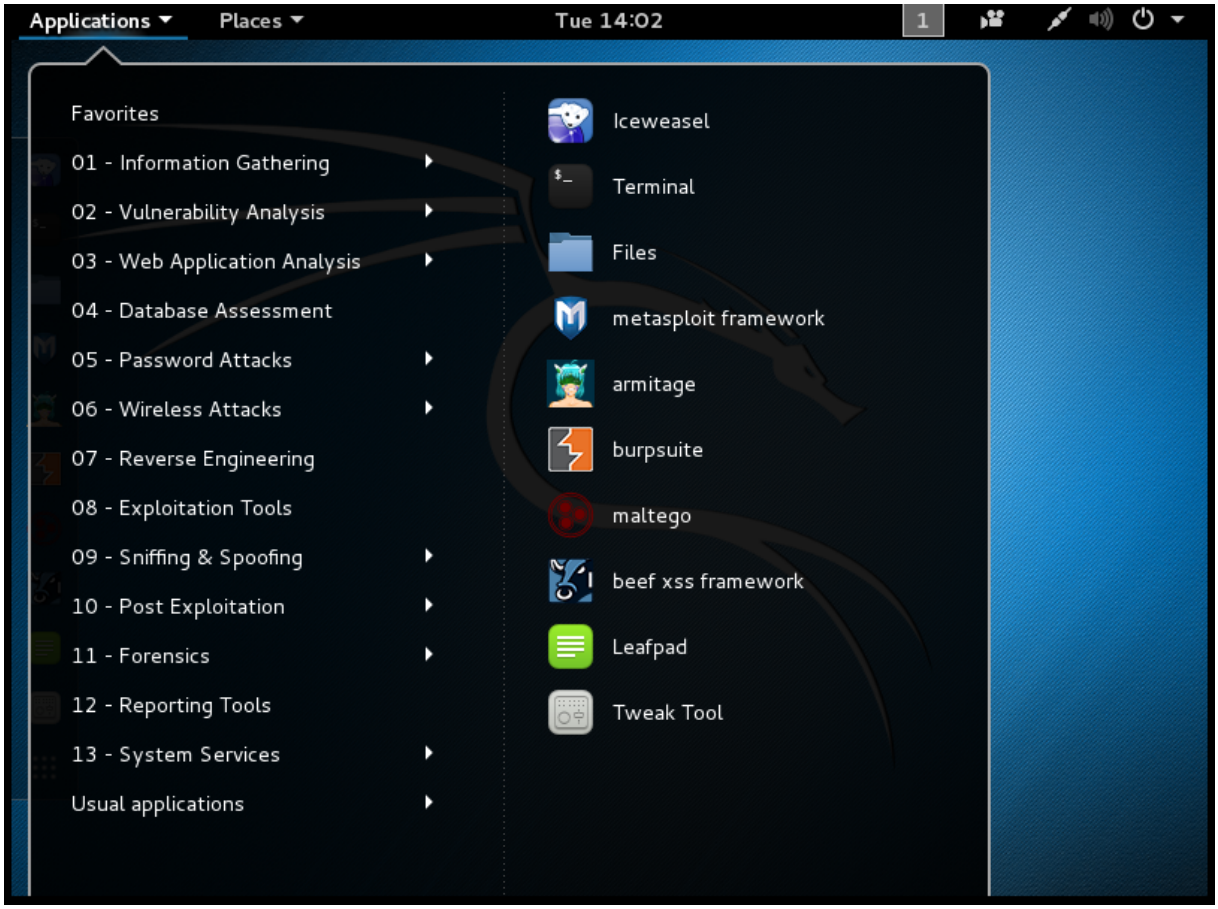


Figura 2.6: Menú principal de la herramienta Kali Linux.
Fuente: https://cyberarms.wordpress.com/2015/10/01/kali-linux-2-0-new-desktop-overview/

2.3. Información sobre las tecnologías usadas

En este apartado se comentarán los distintos frameworks y tecnologías usados.

2.3.1. Python y Django

El lenguaje de programación elegido ha sido Python[11] 3. Esto ha sido así principalmente por los siguientes motivos:

[8] https://www.offensive-security.com
[9] https://www.metasploit.com
[10] https://www.wireshark.org
[11] https://www.python.org

- Es un lenguaje de scripting que permite polimorfismo (Libros Web, 2018), orientación a objetos (Stroustrup, 1988), etc.
- Es de código abierto (M. Rouse, 2009).
- Es multiplataforma (EcuRed, 2018a).
- La instalación es simple. No requiere de *virtual machines* (Lindholm et al., 2014) o compiladores (Qiu, 2016) para ser ejecutado.
- Es un lenguaje dominado por el equipo de desarrollo del proyecto (Muñoz, 2017).
- Permite una fácil integración con scripts en bash (Newham and Rosenblatt, 2005).
- Django.

El último de estos motivos ha sido el más decisivo para la elección del lenguaje. Django es un framework web de alto nivel para Python que permite un desarrollo rápido y limpio de una aplicación web, permitiendo al desarrollador olvidarse de los detalles del desarrollo web, control de la entrada de datos, conexión con la base de datos, etc; y centrarse en el backend (J. Long, 2018).

2.3.2. Celery

Celery es un conjunto de herramientas que permiten trabajar de forma sencilla con múltiples servicios, llamados tareas, de forma simultánea. Existen algunas alternativas a Celery como Python-RQ[12], pero ninguna con una comunidad tan grande y tanta documentación.

Una de las virtudes de Celery es que permite una facilidad para programar tareas que de otra forma serían bastante más complicadas. Para instalarlo es necesario ejecutar sólamente el comando `pip install celery`, y una vez instalado, crear una tarea es tan facil como lo muestra el ejemplo 2.7.

```
1 from celery import Celery
2
3 app = Celery('tasks', broker='pyamqp://guest@localhost//')
4
5 @app.task
6 def remind(msg="Hello, World!"):
7     return msg
```

Figura 2.7: Código ejemplo de tarea usando Celery

Ahora se pondrá de ejemplo una tarea real del proyecto. Esta, además, muestra el pequeño cambio necesario para hacer que dicha tarea sea periódica (cada 5 minutos). Es posible ver esta tarea (sin el código no relacionado con Celery) en la figura 2.8

Una vez creada la tarea, basta con lanzar los procesos de Celery. El primero de ellos será el *Beat* (que significa latido) y será el que se encargue de lanzar la tarea a la cola de ejecución. Se puede ver una imagen de este proceso lanzando la tarea anterior en la figura 2.9.

[12] http://python-rq.org/

```
@app.task
def sniff_each_5_min():
    [...]

app.conf.beat_schedule = {
    "sniff-each-5-min-task": {
        "task": "Sniffing.tasks.sniff_each_5_min",
        "schedule": 300.0
    }
}
```

Figura 2.8: Tarea periódica con Celery

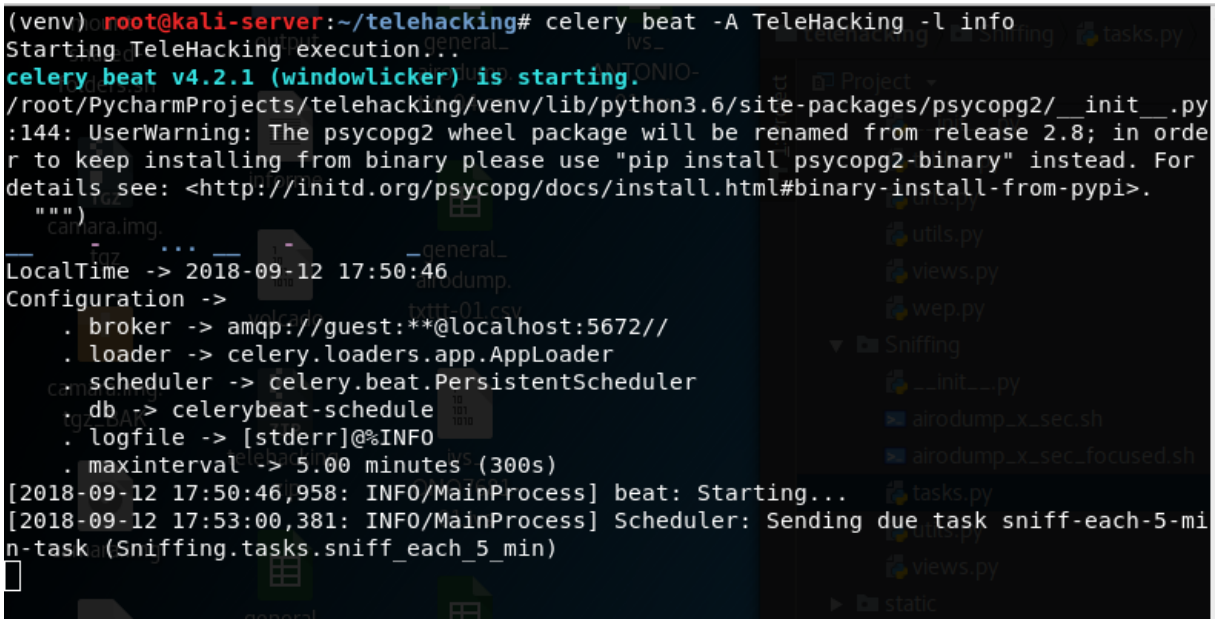


Figura 2.9: Ejecución de Celery Beat.

El otro proceso de Celery es el *Worker*. Este se encarga de coger la primera tarea de la cola de ejecución y lanzarla. También es posible ver la ejecución de este en la figura 2.10.

En ambas imágenes (2.9 y 2.10) se puede ver la sincronización de ambos procesos.

Es posible encontrar más información sobre Celery y sus tareas en (García, 2015). Sin embargo, Celery tiene algunos problemas como que no garantiza la entrega ni el orden de las tareas, no es capaz de mantener las tareas si el worker está ocupado o desconectado, etc. Es por esto que se hablará a continuación de *RabbitMQ*.

2.3.3. RabbitMQ

Debido a los problemas comentados en el punto anterior con Celery, es necesario buscar una *cola de mensajes* (F. Escribano, 2016). En este ámbito, existen muchas bases de datos para colas, como pueden ser Redis[13], ActiveMQ[14] o OpenAMQ[15]. Sin embargo, la que más se encuentra en

[13] https://redis.io/
[14] http://activemq.apache.org/
[15] http://freshmeat.sourceforge.net/projects/openamq

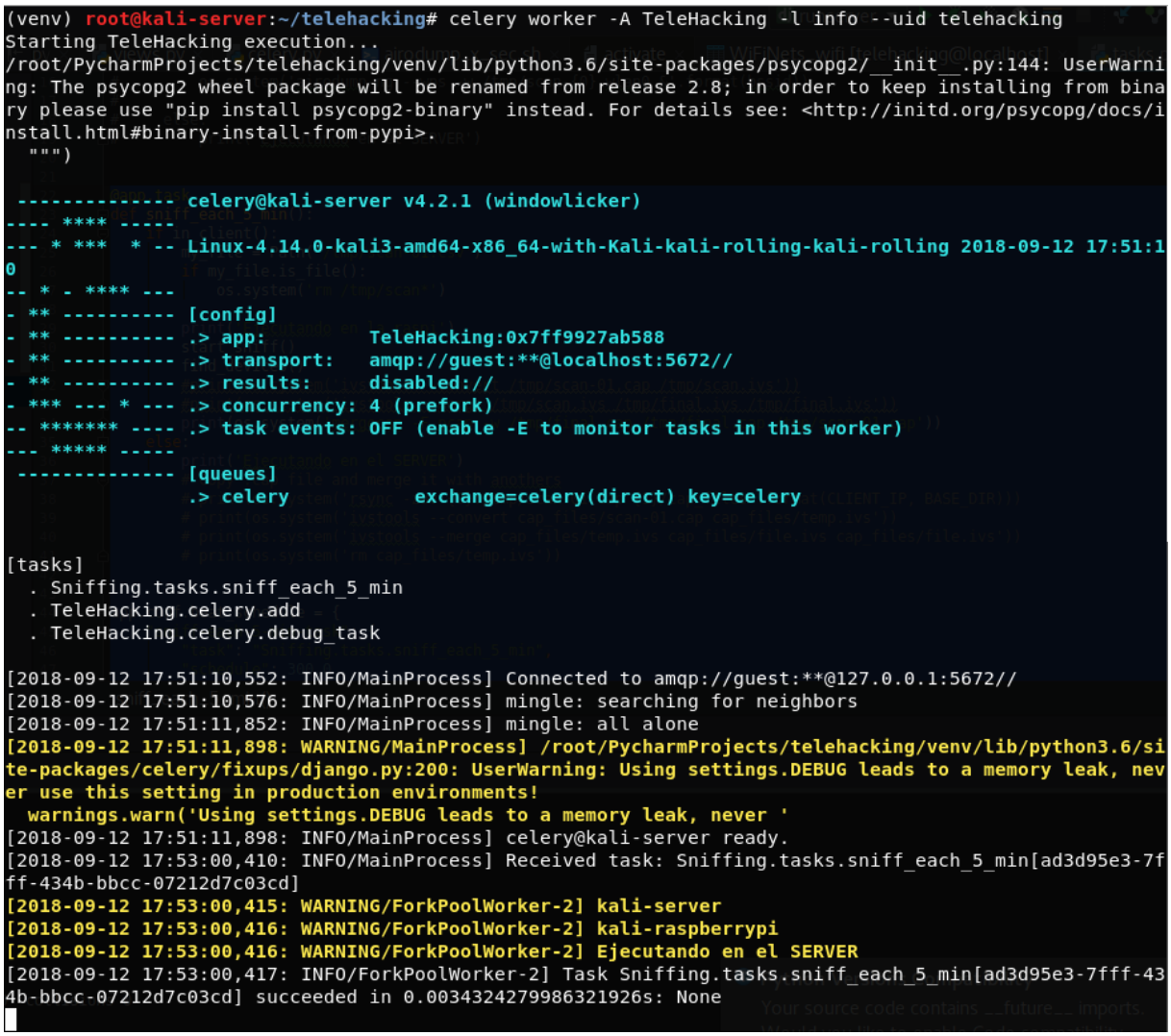


Figura 2.10: Ejecución de Celery Worker.

la red junto a Celery es RabbitMQ, y no es para menos. La configuración es mínima como se podrá ver a continuación.

RabbitMQ se instala desde los paquetes del sistema operativo con la ejecución del comando `apt-get install rabbitmq-server`. Una vez instalado, el servicio se inicia con el comando `/etc/init.d/rabbitmq-server start`.

Para configurarlo, solo hay que ir a la configuración de Celery (por ejemplo, en la figura 2.7) y cambiar la configuración del broker por `broker='amqp://guest:guest@localhost:5672//'`. A partir de este momento, RabbitMQ se convierte en el gestor de colas de Celery, sin ningún tipo más de configuración[16].

[16]Para asegurar que está correctamente instalado basta con probar a parar el proceso de RabbitMQ e intentar lanzar Celery como en el punto anterior. Si todo está correctamente configurado, Celery debería dar error de conexión hasta que restauremos el servicio de RabbitMQ.

2.3.4. HTTP, HTTPS y SSL/TSL

HTTP (*Hypertext Transfer Protocol*) es un protocolo de comunicación desarrollado en 1996, con el RFC 1945 (Berners-Lee et al., 2005), siendo actualmente la versión en uso HTTP2, con el RFC 7540 (Belshe et al., 2015). Este protocolo define la sintaxis y la semántica a utilizar por los distintos elementos de software web para comunicarse, pero sin guardar estado ni mantener ningún tipo de seguridad. HTTP usa el puerto 80 para aceptar conexiones.

Además, define una serie de *métodos de petición*, entre los que se encuentran GET, POST, PUT, DELETE y OPTIONS, entre otros.

SSL/TLS es un conjunto de protocolos de seguridad que permiten crear comunicaciones seguras por una red, proporcionando herramientas para el intercambio de claves (Canetti and Krawczyk, 2001), la autenticación (Krawczyk, 2001) y la confidencialidad (Bisel, 2007) de las comunicaciones.

La falta de seguridad en el protocolo HTTP ha hecho de HTTPS una necesidad. HTTPS añade SSL en la capa de transporte. HTTPS, a diferencia de HTTP, usa el puerto 443 para realizar las conexiones.

Sin embargo, el uso de SSL/TLS y de las comunicaciones seguras no entra dentro del ámbito de este proyecto, por lo que se usará HTTP.

2.3.5. UFW

En la actualidad existe un gran mercado de firewalls (Tecnología&Informática, 2018) para Linux, entre los que se encuentran Iptables[17], IPCop[18], Shorewall[19], etc. En este proyecto, el firewall usado para asegurar el acceso a la aplicación será UFW[20] (Uncomplicated FireWall).

Se ha elegido UFW por ser extremadamente sencillo de usar, así como por la cantidad de documentación que se puede encontrar en internet sobre él.

2.3.6. API REST

Se ha comentado anteriormente que este proyecto va a basarse en una API REST (BBVA, 2016), pero no se ha definido qué es.

Una API es un conjunto de normas y procedimientos que ofrece una aplicación o biblioteca para ser utilizado como una capa de abstracción. Esto permite a los usuarios poder crear aplicaciones propias para interactuar con aplicaciones de terceros, usando capas personalizadas.

REST es el acrónimo de "REpresentational State Transfer". La clave de un servicio REST es que no tiene estado, a diferencia de los servicios actuales de internet en los cuales, tras hacer login, el sistema te "recuerda". Para hacer que un servicio REST reconozca al remitente, este debe demostrarle quién es en cada llamada, ya sea con credenciales, tokens o cualquier otro tipo de autenticación.

[17] https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/6/html/security_guide/sect-security_guide-iptables

[18] http://www.ipcop.org

[19] http://shorewall.org

[20] https://help.ubuntu.com/community/UFW

En este proyecto, para evitar una conexión permanente que esté gastando recursos, se ha optado por esta tecnología y, para ello, se va a trabajar con *Django REST Framework*[21].

Esta librería permite definir URLs en el servidor, y actuar según el tipo de petición HTTP recibida (GET, POST, PUT o DELETE) y el mensaje que traiga. Por temas de organización se ha definido que el formato de mensaje sea siempre JSON.

Por tanto, se han definido varias URLs en la API del proyecto:

- /wifi/: Es la URL principal de la API. Cuando se envía una petición PUT con un JSON bien formado (imagen 2.11, es posible guardar una nueva red en la base de datos. Si se ha ejecutado correctamente, devolverá como respuesta el JSON guardado, lo que permite asegurarse de los datos enviados. La salida puede verse en la imagen 2.12.

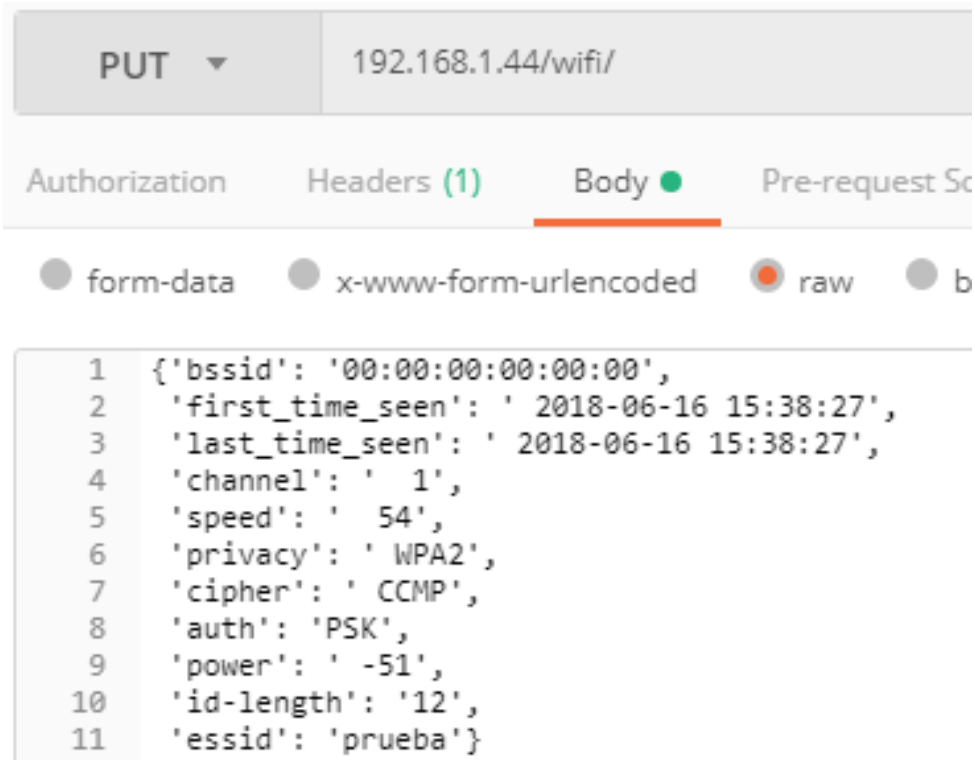


Figura 2.11: Petición PUT a la URL /wifi/ con un JSON

- Si por el contrario, lo que se lanza a la url /wifi/ es una petición POST, se obtiene una lista con cada una de las redes Wifi guardadas en la base de datos, como se puede ver en la imagen 2.13.

[21] http://www.django-rest-framework.org/

```
{
    "alias": null,
    "essid": "prueba",
    "bssid": "00:00:00:00:00:00",
    "latitude": null,
    "longitude": null,
    "password": "",
    "gateway": null,
    "description": "",
    "auth": "PSK",
    "cypher": null,
    "privacy": "WPA2",
    "channel": 1,
    "power": -51,
    "handshake": null
}
```


Figura 2.12: Respuesta del servidor tras una petición PUT a /wifi/

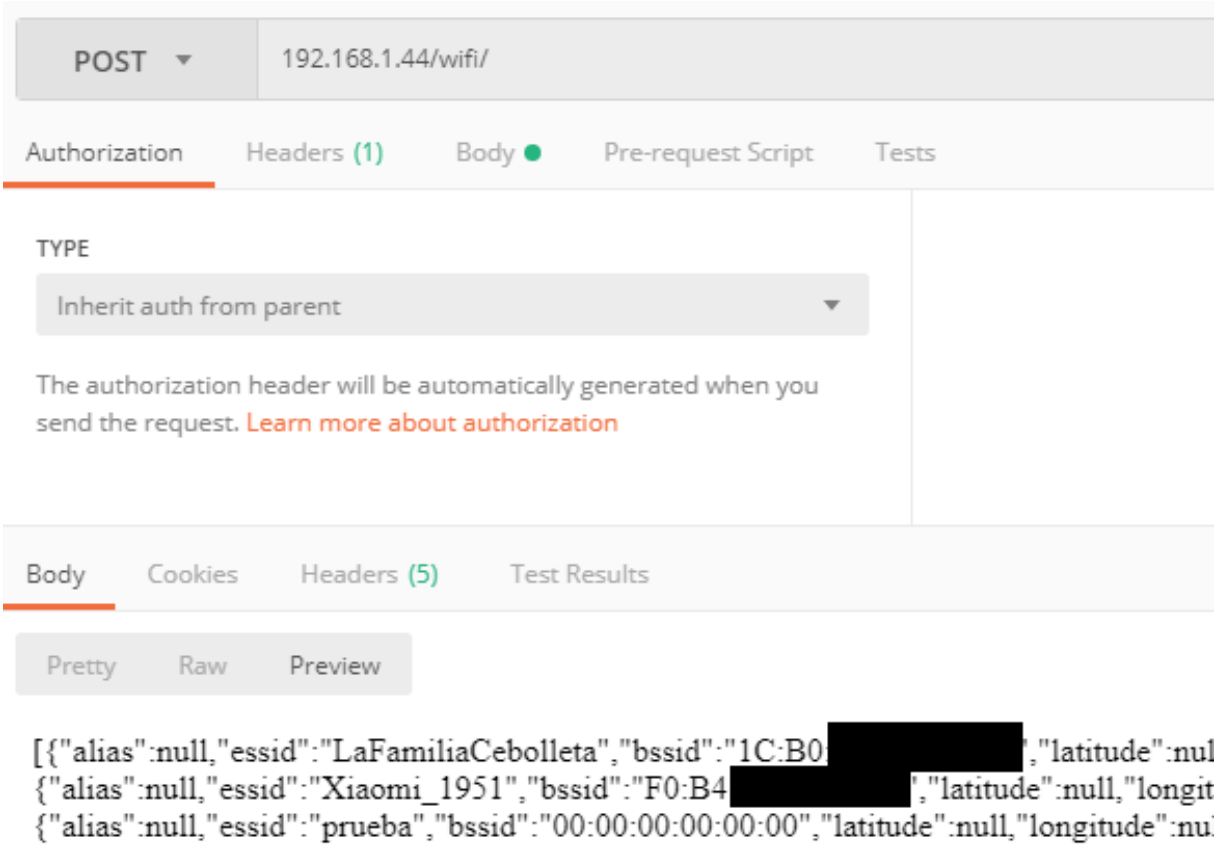


Figura 2.13: Petición POST a /wifi/

2.4. Diseño de Software

Aquí se hablará sobre la teoría de la metodología usada, así como los diagramas creados para el proyecto.

2.4.1. Metodologías ágiles

Para este proyecto se ha dudado entre usar dos metodologías ágiles(evaluandosoftware, 2018) de software: Scrum (ProyectosAgiles.com, 2018) y eXtreme Programming (XP) (J.L. Avalos, 2018). Estas dos metodologías tienen varios puntos en común, ya que pertenecen ambas al grupo de las metodologías ágiles (basadas en los valores del *agile manifiesto*, se realizan entregas periódicas al cliente, etc). Sin embargo, entre las diferencias encontramos:

- XP es más flexible a cambios en los requerimientos durante las iteraciones.
- Los equipos de XP trabajan con iteraciones de 1 a 2 semanas, bastante más cortas que los sprints del Scrum (de 2 semanas a un mes).
- Las tareas son susceptibles a modificaciones, incluso después de que funcionen correctamente

Debido a la mayor flexibilidad de XP, y dado que el equipo estaba formado por una única persona, se decidió seguir el camino del eXtreme Programming.

2.4.2. Diagramas de secuencia

Durante la ejecución del proyecto se ha hecho uso de diferentes diagramas de secuencia, para aclarar el comportamiento que debiera tener el producto cuando trabajara de manera autónoma.

Un diagrama de secuencia (Lucidchart, 2018) se encarga de modelar la interacción de un conjunto de objetos de un sistema a través del tiempo, permitiendo definir de forma ordenada los pasos ejecutados por cada uno, englobando el algoritmo de la aplicación.

2.5. Hardware utilizado

En esta sección se comentará el hardware utilizado para el desarrollo del proyecto, así como las configuraciones realizadas sobre el mismo.

2.5.1. Raspberry Pi

La ejecución del cliente se hará en una Raspberry Pi 3 (imagen 2.14) con el sistema operativo Kali Linux (versión ARM) instalado.

La Raspberry Pi es un mini ordenador de placa única, desarrollada en 2012 en Reino Unido para promover la informática en las escuelas. La versión más actual es la Raspberry Pi 3B+, que salió en marzo de este mismo año. Sin embargo, en el proyecto se usará la versión 3B.

Este modelo (el 3B) dispone de una CPU ARMv8, de 64 bits y 4 núcleos a 1.2GHZ, 1GB de memoria RAM, 4 puertos USB, una entrada Ethernet 10/100 por RJ-45, WiFi, conectividad

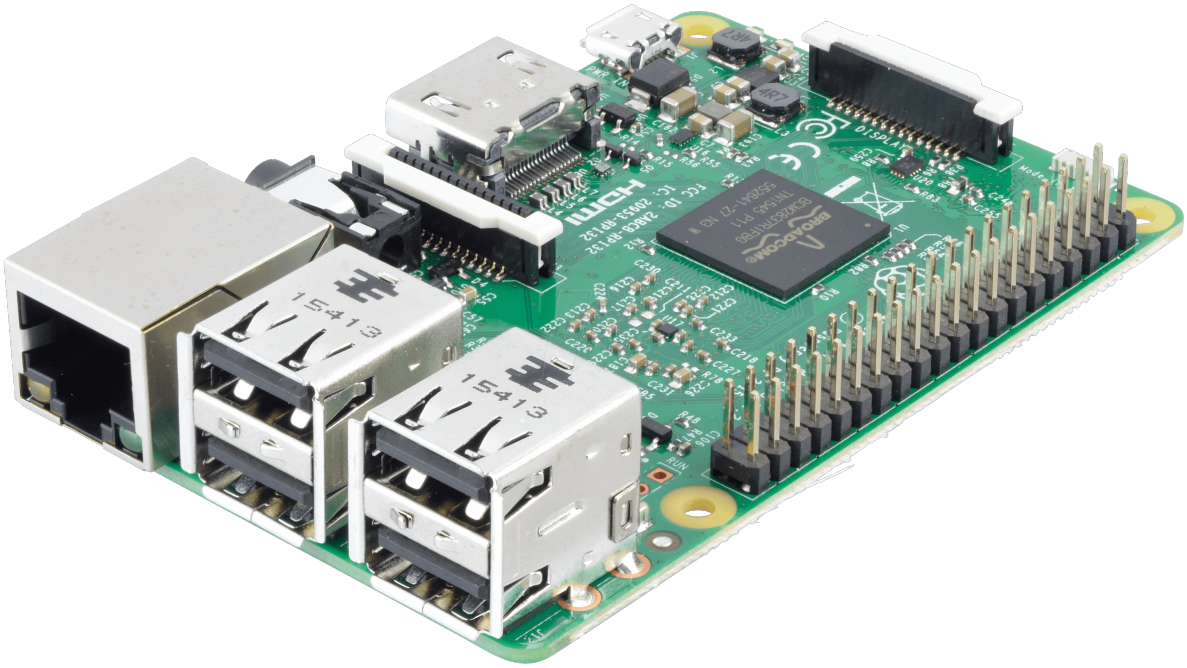


Figura 2.14: Dispositivo Raspberry Pi 3B+

BLE, salida de audio por jack de 3.5mm, salida de video por HDMI y un lector de tarjetas microSD que, de hecho, hará las veces de disco duro del sistema. Además, su consumo es de solo 0.8A, por lo que con una batería de 10A, la autonomía supera las 12 horas.

Otra ventaja añadida es la existencia de los modelos *Raspberry Pi Zero*, con una potencia, un consumo, un precio y un tamaño menores. Estas placas también son recomendables para el proyecto, ya que la falta de potencia no es un inconveniente en este caso.

El hecho de que el dispositivo sea ante todo de pequeño tamaño posibilita, como se ha dicho anteriormente, la aplicación en ámbitos de caja negra, donde los empleados del cliente no pueden sospechar de que están siendo auditados. Además, es un dispositivo que, junto a una batería, el auditor puede dejar escondido en cualquier rincón, recogiéndolo días después o conectándose al mismo mediante SSH.

2.5.2. Alfa AWUS036H

La Raspberry Pi estará conectada una tarjeta de red inalámbrica USB *AWUS036H* (A. Crespo, 2016) (comúnmente conocida como *Alfa 1W*) como la que podemos ver en la imagen 2.15.

Entre sus ventajas cabe destacar que esta tarjeta tiene una potencia de 1W, mientras que las tarjetas convencionales no suelen pasar de los 500mW. Además su precio no es excesivo, pues se puede encontrar por menos de 30€. Otro punto interesante a tener en cuenta es que tiene salida RP-SMA (Data-Alliance, 2018) macho, con lo que es posible conectarle antenas externas de mayor potencia que la que trae de fábrica.

Figura 2.15: Dispositivo AWUS036H

2.5.3. El servidor

El servidor se ejecutará en una máquina de sobremesa en la misma red local que el cliente. Se han hecho pruebas con una instancia gratuita de Amazon[22], pero por cuestiones de potencia y que hacerlo "en la nube" no aporta nada al experimento, se ha decidido realizar la prueba final en un equipo propio.

El equipo en cuestión es un Intel i5 4670k con 24G de memoria RAM Kingston, un SSD Samsung EVO 850 y una ATI Radeon 7870HD de 2G DDR4, y correrá la aplicación sobre una máquina virtual VMware[23], también con el sistema operativo Kali Linux.

2.5.4. Router Sweex LW050V2

Similar al de la imagen 2.16, y con fecha de compra en 2007. Este dispositivo servirá para montar una red WiFi con cifrado WEP, ya que los routers modernos ya no permiten montar este tipo de redes inseguras.

[22]https://aws.amazon.com/es/ec2/
[23]https://www.vmware.com/es/products/workstation-pro.html

Figura 2.16: Router Sweex LW050V2 usado para las pruebas de seguridad WEP

La configuración del router ha sido simple (como se puede ver en la imagen 2.17), definiendo una password hexadecimal de 128 bits, pero sin filtros MAC ni técnicas de ocultación, ya que no dan seguridad real. La contraseña es:

$$1248163264128256512aaaaaaa$$

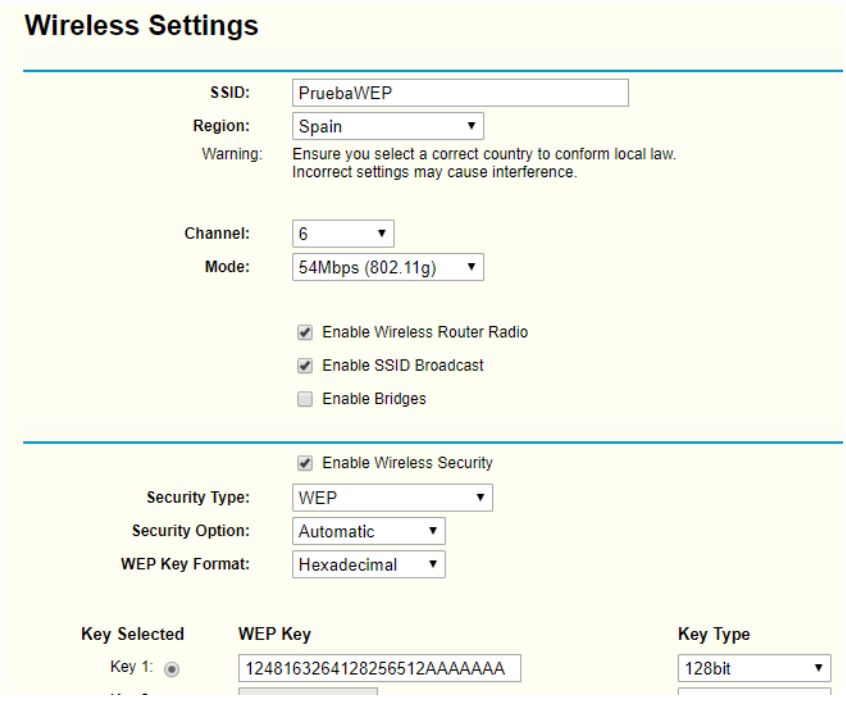


Figura 2.17: Configuración WEP para el router Sweex

2.5.5. Router Xiaomi Mi Wifi 3

Este será el dispositivo encargado de montar una red con cifrado WPA para las pruebas.

Figura 2.18: Router Xiaomi Mi Wifi 3 usado para las pruebas de seguridad WPA

En este caso, para la configuración se ha optado por un cifrado WPA 2. El nombre de la red es "Xiaomi_1951", y la contraseña es "MasterSeguridadUNIR18", como se puede ver en la imagen 2.19.

Figura 2.19: Configuración WPA para el router Xiaomi

Capítulo 3

Objetivos y Metodología de Trabajo

A continuación se introducen los objetivos del proyecto y las metodologías de trabajo usadas.

3.1. Objetivos generales del proyecto

El objetivo principal del proyecto es crear una aplicación 100 % funcional y robusta, capaz de ejecutar de forma automática, y con ayuda de un sistema remoto de mayor potencia, un análisis de las redes WiFi cercanas, obteniendo la contraseña de las mismas cuando sea posible. Se define entonces que la aplicación funciona cuando, en un espacio controlado, es capaz de obtener al menos la clave de una red con seguridad WEP y de otra con seguridad WPA.

Además, se tienen en cuenta otros objetivos secundarios como mostrar la versatilidad que ofrece el framework Django, junto a Python, para poder automatizar tareas de seguridad y dar paso a que la comunidad contribuya en el proyecto, creando nuevos módulos para mejorarlo. Otros objetivos son la comprensión del funcionamiento de los cifrados usados por las redes WiFi y el desarrollo de un proyecto software completo desde cero.

3.2. Metodología de trabajo

Debido a que el desarrollo de la aplicación corre a cargo de una única persona, se ha optado por la metodología ágil *eXtreme Programming* (XP) basada en el desarrollo de los distintos paquetes del proyecto de uno en uno, con una fase de integración y otra de testing tras acabar cada paquete.

El proyecto se ha diseñado inicialmente con un esbozo general de la funcionalidad, el cual se separó poco después en paquetes. Entonces comenzó el desarrollo de cada paquete, ordenados por prioridad y por la capacidad de poder ser probado tras ser acabado. Inicialmente se comenzó por el paquete principal, llamado *TeleHacking*, el cual incluía el servicio de API REST. A continuación le siguió el paquete *api*, que permitía ser probado con la herramienta Postman. En tercer lugar, se implementó el paquete *sniffing*, el cual se encargaba de obtener la información de las redes cercanas y transmitirlas a la API. El cuarto paquete fue *cracking*, que poseía las funciones necesarias para obtener las claves con los datos de la base de datos.

Cada una de estas tareas se ha dividido en diferentes subtareas, que se iban guardando y ordenando por relevancia.

Para lo dicho anteriormente, se han utilizado distintas herramientas para controlar dicha metodología, entre las que se encuentran:

- Bitbucket[1] para el control de versiones (hipertextual, 2014), ya que la cuenta gratuita permite tener proyectos de forma privada, y no se pretende hacer público el proyecto hasta haber acabado con el mismo.
- Para mantener el orden en las tareas se ha trabajado con la plataforma Trello[2], la cual tiene plena compatibilidad con Bitbucket.
- Se ha usado Lucidchart[3] para crear los diagramas usados en el proyecto.

[1] https://bitbucket.org
[2] https://trello.com
[3] https://www.lucidchart.com/

Capítulo 4

Desarrollo del estudio

En este capítulo será descrito el camino recorrido durante el desarrollo del proyecto, haciendo hincapié en las partes importantes y/o más delicadas del mismo.

4.1. Diseño del prototipo

Analizando las necesidades comentadas anteriormente, se propone un esquema que sirva de prototipo en el que basar el proyecto y que sirva de guía para la elección de las tecnologías que se usen.

Se pueden observar en la imagen 4.1 dos partes bien diferenciadas en la aplicación. La primera de ellas se encapsula en un pequeño dispositivo, y se encarga de recopilar datos del entorno en el que se encuentra. La segunda, que se encuentra en el servidor, recibe dichos datos, los procesa, obtiene los resultados y los guarda en la base de datos.

Para realizar esta sincronización se usará una conexión a internet externa, que bien puede ser la de un teléfono móvil compartiendo red o la de una tarjeta SIM conectada al dispositivo. En este estudio, por simplicidad, se usará una conexión en red local.

4.2. Diagramas de secuencia

En los diagramas de secuencia que se presentan a continuación se podrá observar que el usuario final no es necesariamente un actor relevante: en el primero de ellos, en la figura 4.2, ni siquiera interviene. Esto es porque el objetivo de la aplicación pretende ser un análisis automático. Sin embargo, sí que podemos apreciar una actuación del técnico con la finalidad de poder seleccionar objetivos, supliendo el trabajo de la heurística. 4.3. Como se puede ver, automáticamente cada hora, el sistema recoge todos los datos de importancia del cliente, entre los que se encuentran los paquetes IVs (Reddy et al., 2010) para redes con cifrado WEP, y handshakes (Liu et al., 2008) para las redes con cifrado WPA. Tras esta recogida, se encarga de buscar la red con más probabilidades de poder ser crackeada y lo intenta. Sin embargo, la aplicación permite, desde una interfaz visual, que el usuario recoja los datos e intente crackear la red que desee, como se puede ver en el diagrama de la figura 4.3. Además, en la imagen 4.4 se puede ver una captura de la interfaz visual de la aplicación.

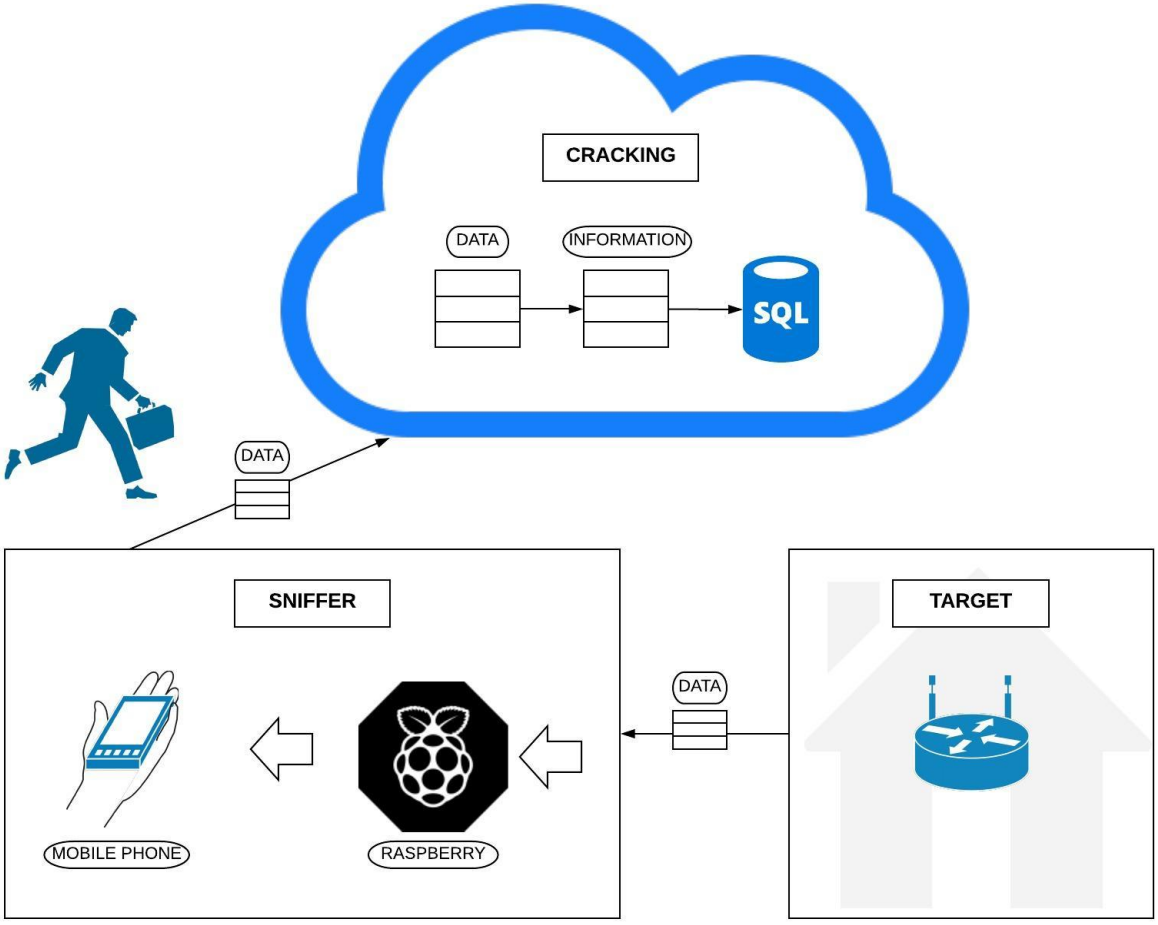


Figura 4.1: Esquema visual que representa la idea original de la aplicación

4.3. Modelo del proyecto

A continuación se comentará la clase[1] principal del proyecto. Además se mostrarán, en detalle, los atributos de la misma.

La clase *WiFi* engloba todo lo necesario para que el proyecto funcione. Esto puede cambiar con futuras ampliaciones del proyecto, pero en este caso es suficiente así.

En la imagen 4.5 es posible ver los atributos de la clase WiFi, entre los que destacan:

- bssid: Es la clave primaria (1keydata.com, 2018) de la clase. Diferencia unívocamente cada objeto, lo cual hace que si de pronto llega a la aplicación un nuevo objeto con una *bssid* conocida, se actualizará el objeto existente.

[1]En Django los modelos de la base de datos, aunque en última instancia sean tablas de la misma, en el propio proyecto se definen como clases en archivos llamados *model.py*

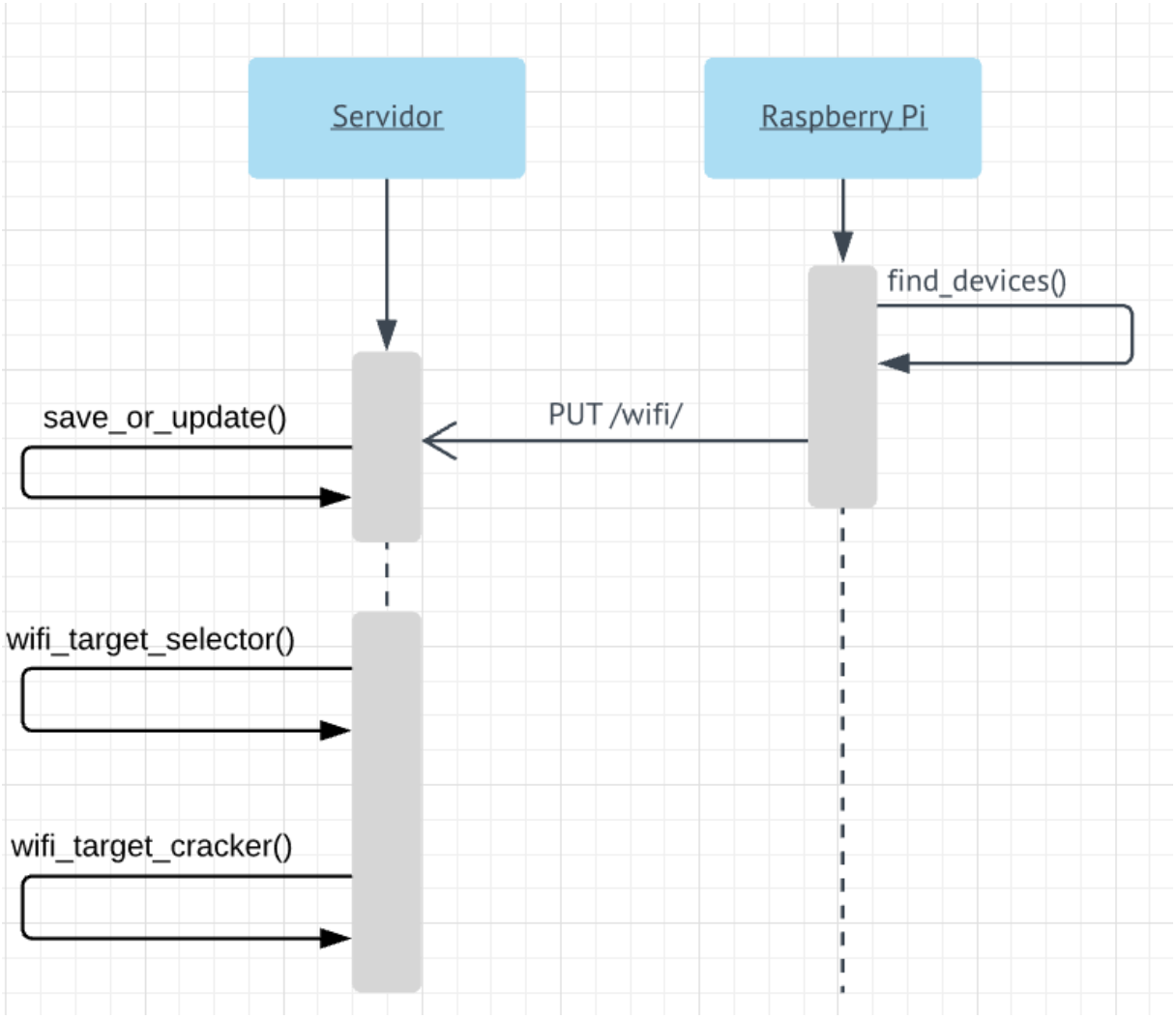


Figura 4.2: Diagrama de secuencia de la automatización de la aplicación.

- password: Por defecto aparece en blanco, hasta que el programa sea capaz de obtenerla.
- ivs: Es el número de paquetes de Vectores de Inicialización obtenidos para esa red. Esta información es útil en el caso de las redes WEP. Sin embargo es necesario tener varios cientos de miles para que sean útiles, siendo necesario en algunos casos tener más de un millón. No hay ningún método determinista que permita calcular cuántos son necesarios para una red WiFi concreta.
- handshake: Para las redes con cifrado WPA. Por defecto aparece en blanco, hasta que se captura uno, en cuyo caso aparece la palabra *"Sí"*.

4.4. Fases de Implementación del proyecto

Se pretende realizar una aplicación homogénea que, dependiendo del nombre del sistema que la ejecute, se comporte como cliente o como servidor. Sin embargo, se empezará desarrollando la parte del servidor por simplicidad: esta parte puede probarse de forma individual, mientras

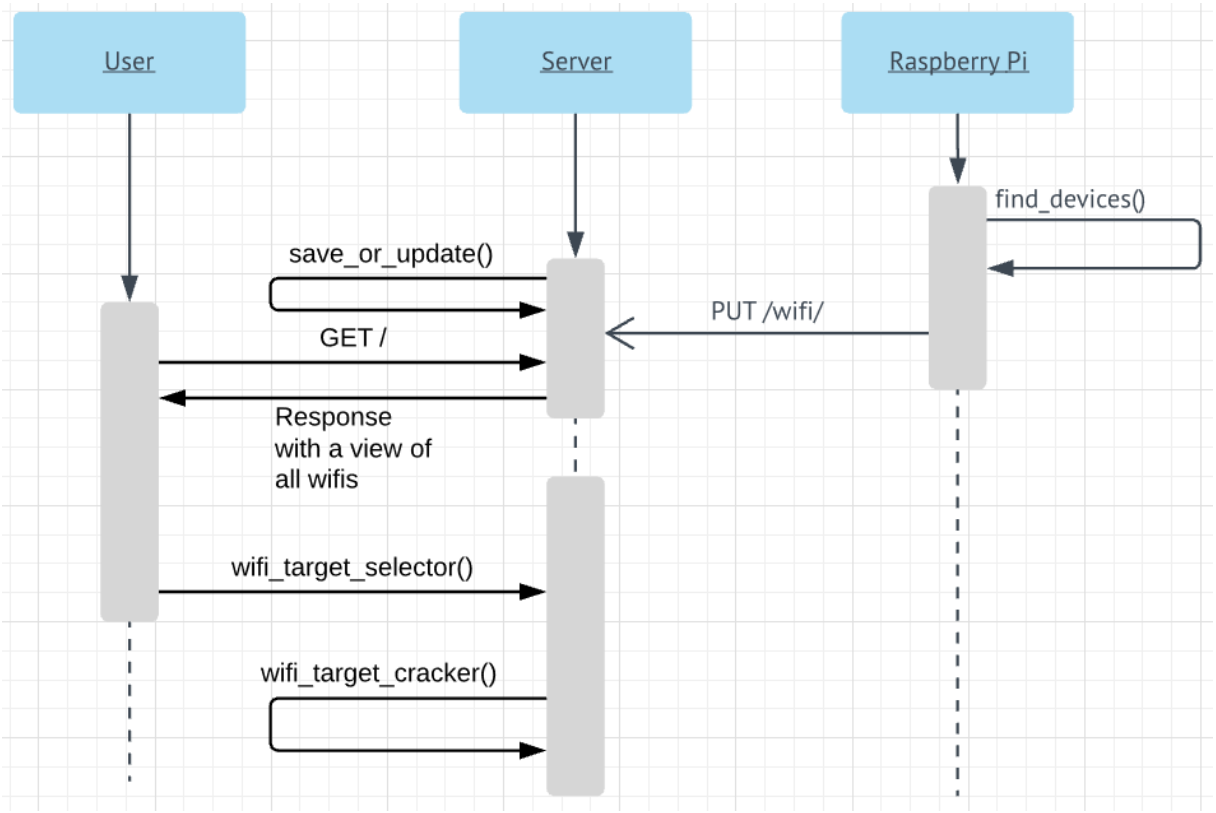


Figura 4.3: Diagrama de secuencia de la aplicación con la interacción de un usuario.

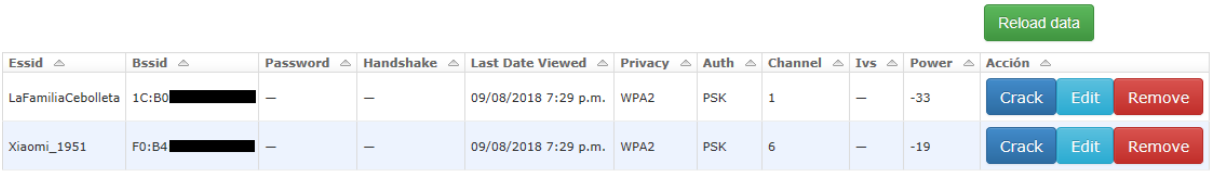


Figura 4.4: Interfaz visual para la interacción con el usuario.

que la parte del cliente necesita un servidor con el que interactuar. A continuación se comentará cómo va a funcionar esta integración:

- Los datos se enviarán directamente del cliente al host. En futuras versiones se podría estudiar el crear un cluster o un *Master-Slave*[2]. Sin embargo, esto sobrepasa los objetivos del trabajo.
- Los datos se crearán mediante el envío de datos en formato JSON a URLs, mediante el método PUT, a la url:

$$http://IP\_SERVIDOR/api/wifi/$$

[2]Una conexión Maestro-Esclavo (*Master-Slave*) entre dos bases de datos es un modo de comunicación en el que uno de los dispositivos tiene control unidireccional sobre el resto. Esto permitiría tener dos bases de datos idénticas que permitirían recopilar datos estando desconectadas, y sincronizarse automáticamente al reconectarse.

```
class Wifi(models.Model):
    alias = models.CharField(blank=True, null=True, max_length=25, default=None)
    essid = models.CharField(max_length=25, blank=True, null=True)
    bssid = models.CharField(primary_key=True, max_length=17)
    latitude = models.FloatField(null=True, blank=True)
    longitude = models.FloatField(null=True, blank=True)
    password = models.CharField(blank=True, null=True, max_length=100)
    handshake = models.CharField(blank=True, null=True, max_length=100)
    gateway = models.CharField(blank=True, null=True, max_length=15)
    date_readed = models.DateTimeField(auto_now_add=True)
    last_date_viewed = models.DateTimeField(auto_now=True)
    description = models.TextField(blank=True, max_length=2500, default="")
    privacy = models.CharField(blank=True, null=True, max_length=25, default=None)
    auth = models.CharField(blank=True, null=True, max_length=25, default=None)
    cypher = models.CharField(blank=True, null=True, max_length=25, default=None)
    channel = models.IntegerField(blank=True, null=True)
    ivs = models.FileField(blank=True, null=True, db_index=True, upload_to='./cap_files/')
    power = models.IntegerField(blank=True, null=True)
    wps = models.CharField(blank=True, null=True, max_length=250, default=None)
```

Figura 4.5: Código de la clase WiFi, del modelo de la aplicación.

Y con un formato de datos con una estructura similar a:

```
{
        'bssid': '58:8F:36:11:22:33',
        'first_time_seen': '2018-06-16 15:38:27',
        'last_time_seen': '2018-06-16 15:38:27',
        'channel': '1',
        'speed': '54',
        'privacy': 'WPA2',
        'cipher': 'CCMP',
        'auth': 'PSK',
        'power': '-51',
        'beacons': '1',
        'iv': '0',
        'id-length': '12',
        'essid': 'AP_de_Prueba',
        'password': ''
}
```

Figura 4.6: Entrada de la API en formato JSON

- Como se dijo anteriormente, en el caso de que ya exista una red con el bssid de los datos de entrada, esta únicamente se actualizará.
- El procedimiento de elección de dispositivos a atacar dependerá de varios factores, los cuales se pueden observar en la imagen 4.7
- Un usuario puede lanzar un ataque a un dispositivo concreto mediante la interfaz web que sirve la aplicación en el servidor

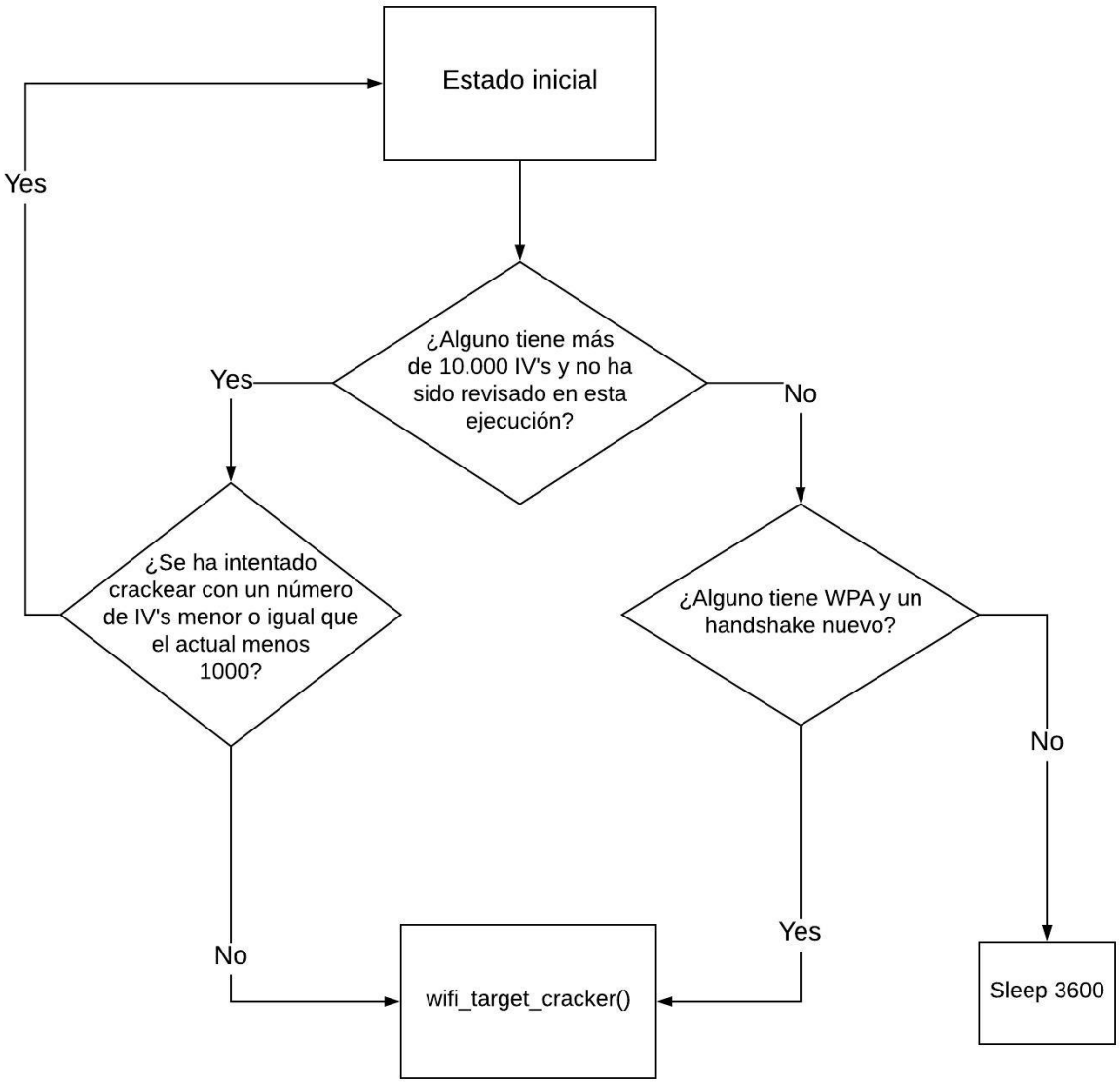


Figura 4.7: Algoritmo de selección de nuevas víctimas.

4.4.1. Desarrollo del servidor

La tarea principal del servidor se concentra en el paquete *cracking*, que será la encargada de ver las entradas de WiFis en la base de datos y elegir cuál de ellas intentar crackear. Además de esto, el servidor ofrece una serie de herramientas de entrada/salida, las cuales se alojan en el paquete *api*.

api

Formada por una *ListView*[3], nos permite definir la URL principal de la API: 127.0.0.1/wifi/. En esta URL es posible usar peticiones POST para obtener la lista de AP's de la base de datos, y PUT para añadir un nuevo AP a la base de datos, como ya se vió en el punto 2.3.6.

[3] *ListView* es un tipo de vista de Django que permite, de una manera simple y elegante, definir las acciones de los métodos POST y PUT

cracking

El paquete *cracking* alberga todo lo necesario para lanzar ataques contra los datos de las distintas redes. Las funciones más importantes de este paquete son las siguientes:

- get_data() se encarga de convertir los archivos *.cap* y *.pcap* de capturas de paquetes en información, obteniendo el número de IVs y de handshakes que contiene dicho archivo, así como a quienes pertenecen. El código de esta función se puede ver en la imagen 4.8.

```
1  def get_data(file, timeout=30):
2      """
3      Executes aircrack-ng and obtains encryption data from access points
4
5      :param file: absolute path of the `.cap` file
6      :param timeout: time to wait for the aircrack-ng process to
           terminate
7      :return: `array` of `dict` with keys: `bssid` and `encryption`
8      """
9      # run the process and capture output
10     aircrack = subprocess.Popen(['aircrack-ng', file], stdout=subprocess
           .PIPE)
11     try:
12         out, err = aircrack.communicate(timeout=timeout)
13     except subprocess.TimeoutExpired:
14         aircrack.kill()
15         out, err = aircrack.communicate()
16
17     # split lines
18     data = str(out).split('\\n')
19     data = data[5:-2]
20
21     # extract data
22     for i in range(len(data)):
23         data[i] = data[i].strip()
24         data[i] = re.split(' {2,}', data[i])
25         data[i] = {
26             'bssid': data[i][1],
27             'encryption': data[i][-1],
28         }
29
30         if 'handshake' in data[i]['encryption']:  # wpa handshake
31             handshake = int(re.split(' +', data[i]['encryption'])
                   [1][1:])
32             data[i]['encryption'] = -1 if handshake > 0 else None
33         elif 'IV' in data[i]['encryption']:  # wep iv
34             data[i]['encryption'] = re.split(' +', data[i]['encryption'
                   ])[1][1:]
35         else:  # no data...
36             data[i]['encryption'] = None
37
38     # filter unnecessary
39     return [entry for entry in data if entry['encryption']]
```

Figura 4.8: Código de la función *get_data*.

- target_selector() es una función heurística que determina cuál de los AP's sin contraseñas es más débil ante un ataque. El algoritmo usado se comentó en 4.7, y devolverá el primero que encuentre. En caso de que no hayan dispositivos disponibles a elegir, devolverá None. En la figura 4.9 es posible ver el código usado.

```
1  def target_selector():
2      """
3      algorithm to select next target
4      :return: a wifi
5      """
6      # if there are any new WEP target
7      for w in Wifi.objects.filter(password='', ivs__gt=10000,
           data_updated=True):
8          return w
9
10     # or there are any new WPA target
11     for w in Wifi.objects.filter(password='', handshake=True, data\
           _updated=True):
12         return w
13
14     # otherwise
15     return None
```

Figura 4.9: Código del algoritmo selector de objetivos.

- crack_AP() recibe el id de un AP y lanza un ataque contra él, teniendo en cuenta el tipo de cifrado que utiliza. Uno de sus parámetros es un timeout (por defecto puesto en 5) que determina el número de horas máximas durante el que puede estar ejecutando este ataque. Es posible ver el código en la figura 4.10. Esta función la lanza una tarea automática cada hora, usando el target devuelto por la función anterior. Sin embargo, también se puede forzar un ataque usando la interfaz visual.

Además, este paquete tiene integradas las vistas que permiten al usuario listar y ver, mediante interfaz web, los APs de los que se tiene información (y que están añadidos en la lista blanca de la configuración), así como ejecutar un ataque sobre ellos, eliminarlos o ver su estado actual.

4.4.2. Desarrollo del cliente

Complementando al servidor, el cliente será quien se encargue de obtener los datos y depositarlos en la base de datos. Para ello, va a basar su comportamiento exclusivamente en una tarea: la búsqueda de los APs cercanos. Cada 5 minutos, el cliente lanzará una búsqueda de 30 segundos de los APs que se encuentran alrededor, enviando todos los datos que encuentre al servidor, mediante la API REST.

Para ello se ha definido el paquete *sniffing*. Este paquete guarda las tareas periódicas y las funciones necesarias para poder auditar el entorno del cliente y guardar las trazas. En él destaca la función *sniff_each_5_min* que, como su nombre indica, se ejecuta cada 5 minutos. Su código se puede ver en 4.11

```
def crack_AP(wifi, timeout=5*60*60):
    """
    :param wifi: Wifi object to be cracked
    :return: the key, and it saves it in the database, or None if it
       could not.
    """
    # sets if it needs a dictionary attack
    dict_attack = False,
    if wifi.handshake:
        dict_attack = True

    # attack and retrieve the key
    key = get_key(wifi.bssid, dict_attack=dict_attack, timeout=timeout)
    wifi.data_updated = False

    # save the key at database
    if key:
        wifi.password = key
    wifi.save()

    return key
```

Figura 4.10: Código del algoritmo de cracking

```
def sniff_each_5_min():
    if in_client():
        my_file = Path('/tmp/scan-01.csv')
        if my_file.is_file():
            os.system('rm /tmp/scan*')

        start_sniff()
        find_devices()
        print(os.system('mergecap -F pcap -w /tmp/final.cap /tmp/final.
           cap /tmp/scan-01.cap'))
```

Figura 4.11: Código Python de la tarea periódica para auditar las redes.

En la función se puede ver que, tras comprobar que se encuentra en el cliente, elimina los temporales anteriores y ejecuta la función *start_sniff()*. Esta función mantiene durante 30 segundos activo el comando `airodump-ng -w /tmp/scan --wps wlan1`, que guarda la traza en el archivo */tmp/scan-01.csv* y los paquetes en */tmp/scan-01.cap*. Acto seguido ejecuta la función *find_devices()*, que se encarga de leer el archivo */tmp/scan-01.csv*, generar los JSONs de los APs y mandarlos mediante petición HTTP a la API, para que el servidor los almacene en la base de datos.

Finalmente, con el comando `mergecap`, mezcla el archivo */tmp/scan-01.cap* con las trazas anteriores, a fin de unificar las salidas. Como se dijo al principio, el proyecto iba a ser homogéneo: no iba a diseñarse una aplicación para el cliente y otra para el servidor, sino que la aplicación será la misma. Para controlar esto, se ha definido dos variable en el archivo *settings.py*. La primera se llama *IS_SERVER* y estará a *True* en el servidor y a *False* en el cliente. La segunda de ellas

se llama *SERVER_IP* y guardará la dirección del servidor[4].

4.5. Problemas acontecidos

En esta sección se hablará sobre los problemas que han ido apareciendo a lo largo del desarrollo del proyecto.

4.5.1. Actualización de AP's

Uno de los primeros problemas aparecidos ha sido la actualización vía API de las entradas de la tabla Wifi. Esto es así se ha decidido que el cliente tenga la menor capacidad de decisión posible (para simplificar el proyecto) por lo que, cada vez que el cliente hacía una petición PUT sobre un AP que ya existía, el servidor devolvía un error debido a que la clave utilizada ya existía. A continuación se muestra un extracto del código utilizado para la obtención y el guardado de los AP's:

```
1  class WifiListView(ListView):
2      model = Wifi
3
4      def get_queryset(self):
5          wifis = Wifi.objects.all()
6          serializer = WifiSerializer(wifis, many=True)
7          return JSONResponse(serializer.data)
8
9      def post(self, request):
10         wifis = Wifi.objects.all()
11         serializer = WifiSerializer(wifis, many=True)
12         return JSONResponse(serializer.data)
13
14     def put(self, request):
15         data = JSONParser().parse(request)
16         serializer = WifiSerializer(data=data)
17         if serializer.is_valid():
18             serializer.save()
19             return JSONResponse(serializer.data, status=201)
20         return JSONResponse(serializer.errors, status=400)
```

Figura 4.12: Código de captura de llamadas a la API.

Es necesario pararse a entender la última función (*put*) de la clase, pues es la que se encarga de actuar ante llamadas *PUT*.

A grandes rasgos se puede entender que esta función obtiene los datos, los formatea para que los entienda el serializador[5] y si es un serializador válido, los guarda. Esto es un problema porque el serializador, en su función *is_valid()* se asegura por defecto de que no haya colisiones en la clave primaria, además de crear los atributos *data* y *error* en el propio serializador (para ver qué datos se van a guardar y si se han encontrado errores al validar los datos).

[4]Es recomendable usar las direcciones IP, ya que si se usan las direcciones DNS y se cae el servidor DNS fallaría la disponibilidad del servidor.

[5]Un serializador es una clase definida para convertir los datos de un JSON en un objeto de la clase para la que se haya creado. En este caso, convierte el JSON recibido en un objeto Wifi, parseando todos sus parámetros.

Entre las diferentes alternativas para solucionar este error se pensó en modificar la función *is_valid()* (lo cual generaría un nuevo problema, porque al guardar con la función *save()* volvería a dar un problema de colisión de claves primarias), o en intentar crear siempre un objeto y luego actualizarlo con el resto de valores. Finalmente se decidió por una tercera opción más simple, en la que primero se validaría y, si esta devolviera *False*, se comprobaría que el error ha sido porque ya existe el elemento, en cuyo caso se obtendría y se actualizaría, como se puede ver en el siguiente extracto de código:

```
    def put(self, request):
        data = JSONParser().parse(request)
        serializer = WifiSerializer(data=data)
        query = Wifi.objects.filter(bssid=data['bssid'])
        if serializer.is_valid():
            serializer.save()
            return JSONResponse(serializer.data, status=201)
        elif len(query) > 0:
            serializer.update(query[0], data)
            return JSONResponse(serializer.data, status=201)
        return JSONResponse(serializer.errors, status=400)
```

Figura 4.13: Código de captura de peticiones PUT arreglado.

4.5.2. Mezclado de los paquetes

Inicialmente, el proceso Airodump-ng se lanzaba en segundo plano, se esperaba 30 segundos y, finalmente, se mataba con el comando *kill*.

Cuando se mandaba el archivo al servidor, este usaba el comando *mergecap* para unirlo con los anteriores. Sin embargo, a veces aparecía el error que se puede ver en la imagen 4.14.

```
(venv) root@kali-server:/tmp# mergecap -F pcap -w /tmp/final.cap /tmp/final.cap /tmp/scan-01.cap
mergecap: The file "/tmp/final.cap" appears to have been cut short in the middle of a packet.
```

Figura 4.14: Error obtenido tras lanzar mergecap.

Este error se debe a que, al matar el proceso Airodump-ng, este deja a medio escribir el último paquete en el fichero, lo cual lo deja corrupto.

Para arreglar el error se ha probado con el comando *pcapfix*[6], pero no se ha obtenido un resultado satisfactorio, pues el error seguía apareciendo, como se puede ver en la imagen 4.15.

Otro intento infructuoso fue probar a matar el proceso con diferentes señales de Unix (A. Alanjawi , 2018).

Finalmente, la solución pasó por usar el atributo *–write-interval 29* al lanzar el comando *airodump-ng*. Este atributo obliga a que cada 29 segundos se deje que escribir en el archivo actual y se pase a uno nuevo, lo cual permite que en el segundo 30 se pueda matar el proceso sin afectar a la escritura del archivo principal.

[6] https://f00l.de/pcapfix/

```
(venv) root@kali-server:/tmp# pcapfix scan-01.cap
pcapfix 1.1.1 (c) 2012-2014 Robert Krause

[*] Reading from file: scan-01.cap
[*] Writing to file: fixed_scan-01.cap
[*] File size: 57124 bytes.
[+] This is a PCAP file.
[*] Analyzing Global Header...
[-] The global pcap header seems to be corrupt! ==> CORRECTED
[*] Analyzing packets...
[*] Progress:  20.21 %
[*] Progress:  40.00 %
[*] Progress:  60.45 %
[*] Progress:  80.05 %
[*] Progress: 100.00 %
[*] Wrote 509 packets to file.
[+] SUCCESS: 1 Corruption(s) fixed!

(venv) root@kali-server:/tmp# mergecap -F pcap -w /tmp/final.cap /tmp/final.cap /tmp/scan-01.cap
mergecap: The file "/tmp/final.cap" appears to have been cut short in the middle of a packet.
(venv) root@kali-server:/tmp#
```

Figura 4.15: Error obtenido tras lanzar mergecap.

Capítulo 5

Resultados finales

A continuación, y tras todo el desarrollo realizado, se va a ejecutar la aplicación, con la pretensión de alcanzar el objetivo del que se habló en el apartado 3.1. Para ello se comenzará habilitando todo el laboratorio de pruebas, que describiremos a continuación.

5.1. Instalación de los equipos

A continuación se comentará lo necesario para poder instalar los equipos de una manera similar a la que se ha usado durante la creación del proyecto.

5.1.1. Instalación de Kali Linux en la Raspberry Pi

El sistema operativo Kali Linux se ha instalado en la Raspberry Pi siguiendo los pasos del libro (Orcero, 2018), los cuales se comentan a continuación:

- Descargar la ISO de Kali Linux para procesadores ARM. Para ello se ejecutará el comando `wget https://images.offensive-security.com/arm-images/kali-linux-2018.2-rpi3-nexmon.img.xz`

Figura 5.1: Descarga de la imagen de Kali Linux para ARM.

- Se comprueba la firma SHA256 (K. Kao, 2014) con el comando `sha256sum kali-linux-2018.2-rpi3-nexmon.img.xz`.

  La firma debería ser idéntica a `69b96f5dd2bd1c23f8878e5ef117e6c2ecb6e5f9a75d66bdea81687fa8433f24` como se puede ver en la imagen 5.2

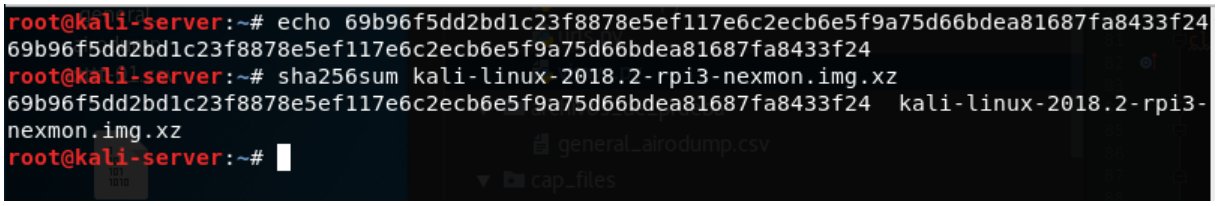


Figura 5.2: Comprobación de firmas para Imagen de Kali Linux para ARM.

- Suponiendo las firmas coincidentes, el siguiente paso es descomprimirlo. Para ello se usará el comando unxz, de la siguiente manera:
  `unxz kali-linux-2018.2-rpi3-nexmon.img.xz`

- A continuación, se grabará la imagen en la tarjeta microSD. Suponiendo que la misma se encuentre en /dev/mmcblk0, el comando sería:
  `cat kali-linux-2018.2-rpi3-nexmon.img > /dev/mmcblk0`

- Ya está Kali instalado en la microSD en una partición, ocupando el 100 % del espacio de la misma. Con la herramienta *gparted*[1] se edita la partición para que use el mayor espacio posible, como en la imagen 5.3. Una vez hecho esto, es posible ver el espacio final con el comando `df -h`.

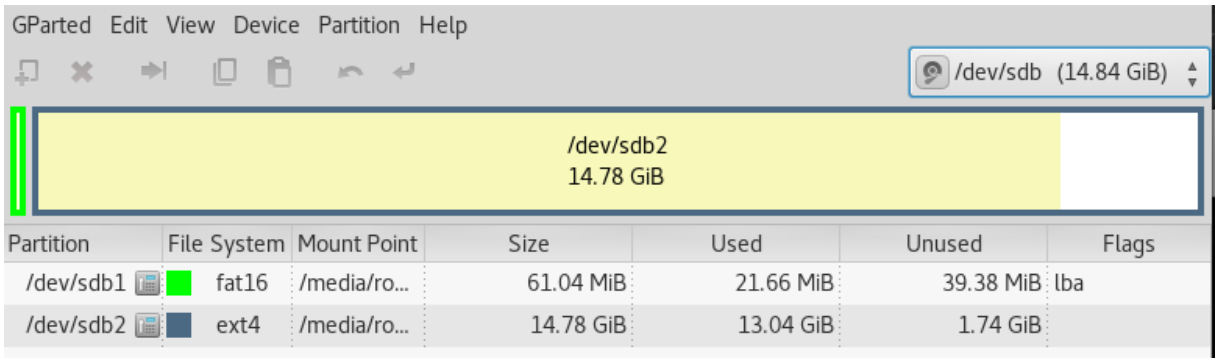


Figura 5.3: Captura de la herramienta *gparted* tras ampliar la microSD de Kali Linux.

- Ya es momento de introducir la microSD en la Raspberry Pi y encenderla. Tras unos instantes aparece la pantalla de login. El nombre de usuario es *root* y la contraseña es *toor* (Offensive Security, 2018). Navegando un poco por los menús se ve que le faltan casi todas las herramientas. A continuación se verá cómo instalarlas.

[1] https://gparted.org/

- Suponiendo que está conectada a la red, el siguiente paso será actualizarla. Para ello se abrirá una consola y se ejecutará el comando `apt-get update`. Esto tardará bastante, ya que son muchos paquetes y la Raspberry Pi no destaca por su potencia.

- Una vez acabado el paso anterior, toca instalar la versión completa de Kali. Para ello se ejecutará el siguiente comando:
  `apt-get install kali-linux-full`.

  Este comando instalará una suite de paquetes de más de 3 gigas de tamaño. Tardará bastante tiempo, pero no requiere de muchas intervenciones por parte del usuario.

- Una vez acabado, se podrá ver en el menú principal el conjunto de paquetes, ordenados de forma similar a los del Kali Linux de AMD y i386, como se vió en la imagen 2.6.

El siguiente paso será configurar una IP fija para no tener problemas de conexión si se apaga, configurando la IP 192.168.1.44 para el servidor y la IP 192.168.1.40 para la Raspberry Pi.

Para ello se ejecutará el comando `ifconfig`, el cual muestra todos los dispositivos de red conectados al sistema y activos, como se puede ver en la imagen 5.4.

```
root@kali-server:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.44  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::20c:29ff:fe71:6a9b  prefixlen 64  scopeid 0x20<link>
        ether 00:0c                 txqueuelen 1000  (Ethernet)
        RX packets 1233598  bytes 1721936532 (1.6 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 554314  bytes 33970720 (32.3 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 5058  bytes 1831524 (1.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5058  bytes 1831524 (1.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura 5.4: Captura de la salida del comando ifconfig.

En este caso se pretende anclar la interfaz *eth0* a la IP *192.168.1.44*[2]. Para ello se procede a editar el archivo */etc/network/interfaces* y añadiremos al final las líneas de la figura 5.5.

El siguiente paso consiste en reiniciar el sistema para que surta efecto la configuración. A continuación se deberán realizarán los mismos pasos para la Raspberry Pi.

5.1.2. Instalación de Kali en una máquina virtual

A continuación se mostrará el proceso de instalación de Kali Linux sobre una máquina virtual soportada por VMware. Para empezar, se descargará e instalará el programa *VMware*

[2]Como el DHCP ha elegido esa IP, podemos estar seguros de que no está siendo usada por otro dispositivo.

```
autho eth0
iface eth0 inet static
        address 192.168.1.44
        netmask 255.255.255.0
        network 192.168.1.0
        broadcast 192.168.1.255
        gateway 192.168.1.1
```

Figura 5.5: Sentencias para fijar una IP a una interfaz.

Workstation Player desde la URL `https://www.vmware.com/products/workstation-player.html`[3].

Además, se hará la descarga de la versión para VMware de Kali Linux, lo cual ahorrará tiempo de configuración de la máquina virtual (montar carpetas compartidas como dice (VMware, 2018b), instalar las *VMware-tools* como explica (VMware, 2018a), etc). Para ello se realizan las descargas desde la URL `https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/`, eligiendo la que tenga extensión *.ova* y que mejor convenga según la arquitectura del sistema sobre el que se trabaje.

Una vez descargada, se abrirá VMware y se hará click en una frase que dice "Open a Virtual Machine" como la de la imagen 5.6. Esto abrirá un navegador de carpetas donde se deberá buscar la imagen de Kali descargada y seleccionarla. Se selecciona donde se guardará la máquina virtual, su nombre y clic en aceptar. Unos instantes después acabará la importación y aparecerá una nueva entrada en el listado de máquinas virtuales. Esto se puede ver en la imagen 5.7.

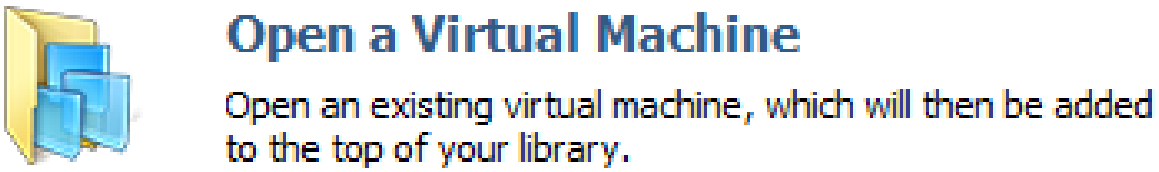


Figura 5.6: Botón para añadir una nueva máquina a VMware Player.

Antes de iniciar la máquina, es preferible configurarla. Para ello se hará clic en el botón que se puede ver en la imagen 5.7 y que dice "Edit virtual machine settings". En la ventana que aparece habrá que ajustar la memoria RAM dedicada, número de núcleos y demás, pero la configuración importante se encuentra en la tarjeta de red, que por defecto viene en modo *NAT*. El modo más interesante para herramientas de red suele ser el modo *bridge*, pues permite que la máquina actúe como un dispositivo más en la red local, por lo que se debe configurar el adaptador como se ve en la imagen 5.8. Podemos encontrar más información en (Scott D. Lowe, 2011).

Hecho esto, solo queda aceptar y dar click en el botón de "Play virtual machine" de la figura

[3]Gratuito siempre que no se use con fines comerciales.

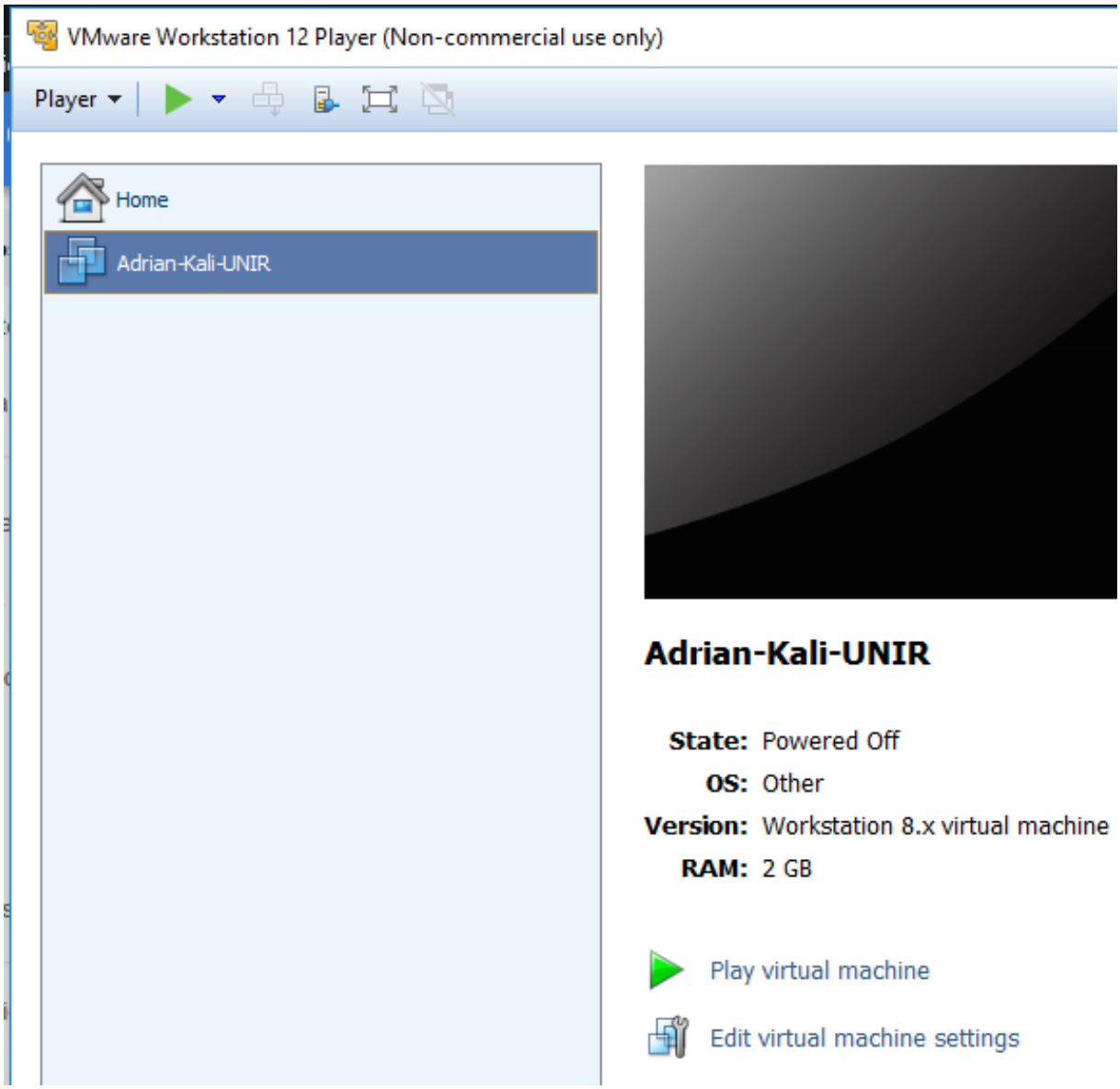


Figura 5.7: Captura de la interfaz de VMware con Kali añadido.

5.7, lo que iniciará el sistema operativo Kali Linux tras unos instantes, mostrando una pantalla gris similar a la de la figura 5.9 pidiendo las credenciales.

Al igual que en el Kali Linux de ARM, el nombre de usuario para este es *root* y la contraseña, *toor*. De todos modos, aunque esta versión ya viene con todas las herramientas instaladas, no viene mal actualizar los paquetes con los comandos `apt-get update` y `apt-get upgrade`.

5.1.3. Firewall

El último paso a realizar en ambos dispositivos será la apertura de los puertos del firewall. Estos pasos deberán hacerse el servidor para que puedan conectarse entre sí. Como la Raspberry Pi no necesitará aceptar conexiones entrantes, no será necesario activar ninguna regla. Se seguirán los pasos marcados en (P. Arianto, 2015).

Se comenzará instalando *ufw* con el comando `apt-get install ufw`. Una vez instalado, con

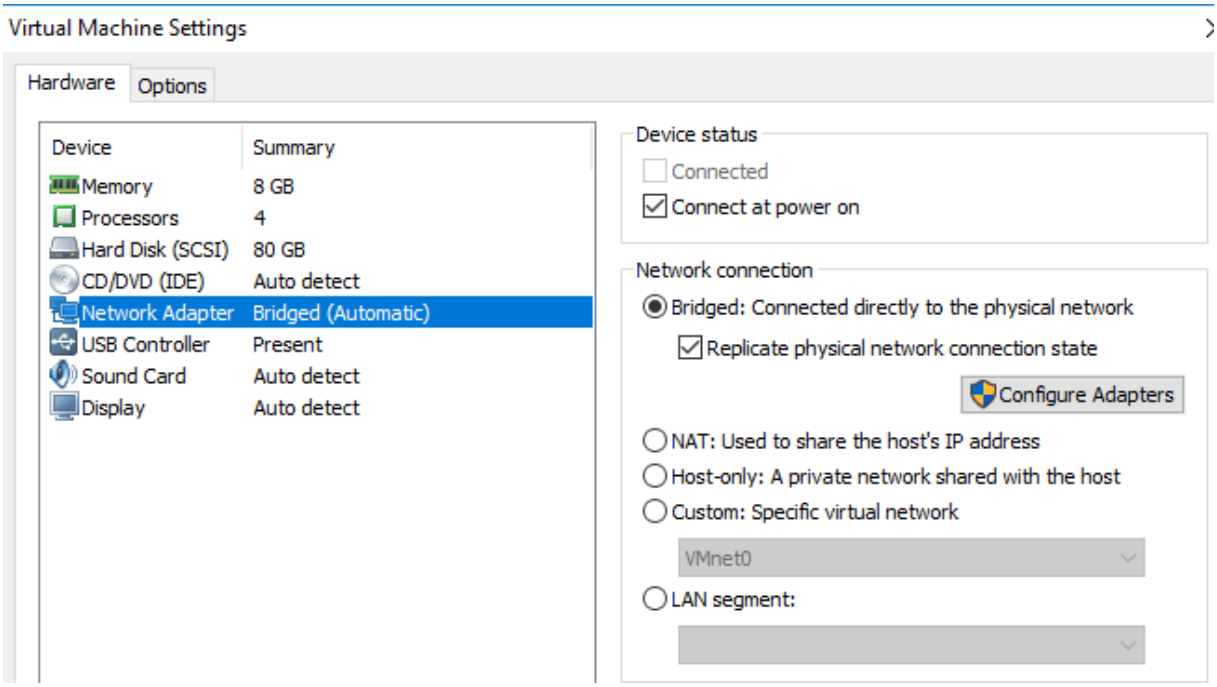


Figura 5.8: Configuración de red para Kali en VMware Player.

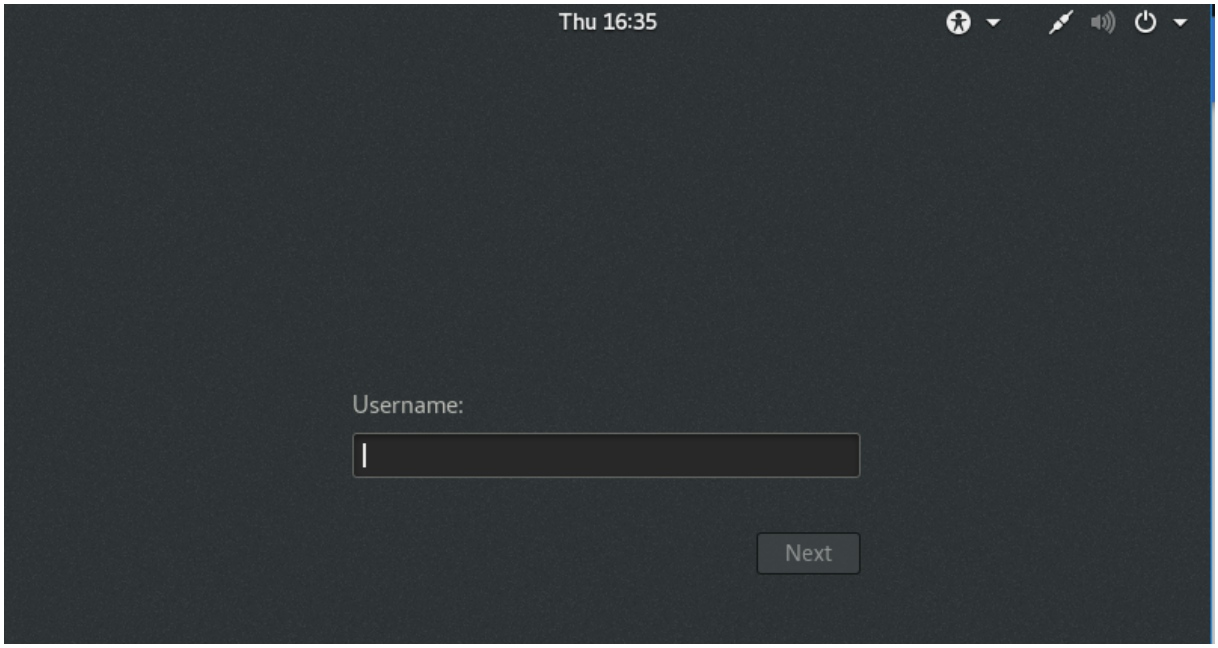


Figura 5.9: Pantalla de login de Kali Linux.

el comando `ufw status` se podrá ver el estado del mismo.

A continuación será necesario ejecutar la siguiente sentencia en la consola:

```
ufw allow from 192.168.1.40 to any port 80
```

Si se ejecuta ahora el comando `ufw status` se deberá ver algo similar a lo que muestra la imagen 5.10

Esta sentencia permite la conexión al puerto HTTP (el puerto 80) a la Raspberry Pi. Si se

```
(venv) root@kali-server:/tmp# ufw status
Status: inactive
(venv) root@kali-server:/tmp# ufw allow from 192.168.1.40 to any port 80
Rules updated
(venv) root@kali-server:/tmp# ufw enable
Firewall is active and enabled on system startup
(venv) root@kali-server:/tmp# ufw status
Status: active

To                         Action      From
--                         ------      ----
80                         ALLOW       192.168.1.40

(venv) root@kali-server:/tmp#
```

Figura 5.10: Salida del comando *ufw status* tras ser configurado.

quisiera acceder a la interfaz visual desde otro dispositivo, sería necesario repetir la sentencia con la IP del mismo.

Una vez hecho esto, es necesario activar el firewall con el comando `ufw enable` el cual pedirá confirmación[4].

5.2. Requisitos previos

En esta sección se comentarán todos los pasos previos a la ejecución de la aplicación, desde la descarga hasta la configuración de las tecnologías usadas.

5.2.1. Descarga e instalación de la aplicación.

La aplicación se encuentra subida en la herramienta *Bitbucket* en un repositorio privado. Esto es así por seguridad: Esta es una herramienta que automatiza la obtención de claves red, por lo que podría ser una insensatez que la misma estuviera depositada en un repositorio público y permitir que sea usada por personas sin el conocimiento y/o madurez suficiente.

A continuación se describirán los pasos necesarios para la descarga e instalación de la misma.

- Lo primero será instalar la clave RSA privada que autenticará contra Bitbucket, permitiendo el acceso a la descarga del repositorio. Para ello habrá que copiar la clave, en la figura 5.11. A continuación se creará un archivo llamado *telehacking-ro-amunoz* donde se copiará la clave completa (incluyendo las cabeceras que comienzan con "—").
- Una vez creado el archivo, se deberá guardar en la lista de claves SSH para que Git la encuentre. Para esto es necesario tan solo ejecutar el comando

[4]En el caso de que se esté accediendo al servidor de forma telemática (SSH, X11 o similares) será necesario también abrir dichos puertos para no perder el acceso.

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAoNx7cEG5moNkmjnZwUwhh/hcSwSDv/Ey1/tq9itvF+7mXa4L
nAQHk3+ytF52Zrv4lR+4UJtBWlp45sSDIQBVT8mDPhR9RNC+P9SXeVtEuMFEEKXv
80B5iWw+EgmcMkJzzx7G4s3uLu2tq/uZkqVPfsM/TPvCscjz7AnrHE8ZjL5iuD7U
v6IAHT4FU+FDF42DEjwYJTqfn7CduGWi8ckzReP8KgjNw/rCWBuz0Y5LQ4i7jgJ+
cPHe5BWwl0V1cELeWJzfI6A/adNhCIHqM7Dnt+tdMwP95rauagRRo2reIESU+82G
Vd1eaDryI+mBurqYGwYjjlxRAq/IQqpvm1H1sQIDAQABAoIBACGXqZd5wj7ILZwf
dJ/vPBle/jcUTEcrIYkFEgMyctK2qxjpHbeJB+P/iNwt1eACZRkPNLhwk6dAuXrH
kvKzVR2CYYv3T2mthe1igfdv3dwPUB2B6agM0NSn7BZsN1gvlGuua641fwYB0IDX
2+xJ5V39fByhfrkN1BJ4x8ZzpkxDqSxbtyH8dk0hKQUvZ//GtZsFJYvWHP2plv5g
ZEtRUKMcolyjtn61KJ2qDf9dZT+ocVQZBHwk3j7PlrsSG037vwBu4Yj02yelZGLQ
mqKW4o6LMRdJZsUFcaFTTtO0IDaLKhom3jDuTKLmwnDfPFxQyWLCkJeEqYoqv3qS
Xt7Kg6ECgYEAyvGAthyrcHZjc+lKTIMY7Onch7NsbD9uW/QFLOv68VU0LQoZB+tV
D//ywbD3Fh3aOMHFUsEWiP/S+M5+Fg5ZQhVAud/fUbvJnQdFK5/yjGDfvmF/yeOG
mD1cMhhVOTZswJQ4RgI2ULpqK4r6cEDPg9FeDBaBJK+OPXU8l0LP200CgYEAyuqH
IsHfLJivblGhASUbyR2o6mX9eNn7UHhVMBufkfQHqKSXwxuBm7IVDLytt52DZUVU
ipNcsZc8SrhqLRBM9ElinkDUCwmlbhtBp88pEUcmkBHabu0TAai601tLJIuntS39
eHI+wtXkulVp+eDs4Wr9gw8oexiA3un54gz86fUCgYBHQNeyGQrQZ6Cvd79Qcc70
j3+QXZy5vOOggWK3zPrEdOfm0F7NewDYp0KyYtG+ACk4OEBxt72TyE5ocQev8sW/
//pDyKh95/L3oS/WC0h7pR+plk3psmr1VMXkCMsTVVqFmwTO/8PWQD06oIchln/F
hmMA133sRX8pcNofOaQFGQKBgHjLoOLaEudThavgwD4UPzQ8id/wEzKsYWWIHP5k
k1upsB1+zjGB4Gg1XT6V27xg8mWJHkjYbOfzphq9Kcfl2knanfCjmxdvCKJBa5Zg
RoxYlOZTGnvKefQU5BMKV/NLh+zdkfTv3djSyDT8v/fXJG0U6hHpJioRbjZFh1JA
dRc5AoGBAKqkvXsdzoRLHd/nAsqRgf4XHHURwi/y9AvMWgwrQWKyiKls+3h9iuf2
fyPDKEku/6VJisP3BcaxV+70R8FPAf/f+Gp0G9qcS9PglyMkQXYK71HMRPjHK0Pf
aSUjgg2MUzh1o/oAcRHk5stHLAKNavQOB/P0kgSqbKBTUiVlYV0H
-----END RSA PRIVATE KEY-----
```

Figura 5.11: Clave RSA para la descarga del proyecto.

`ssh-add telehacking-ro-amunoz`, como explica (Bitbucket Support, 2018). Un ejemplo de salida satisfactoria se puede ver en la imagen 5.12.

```
root@kali-server:~# ssh-add telehacking-ro-amunoz
Identity added: telehacking-ro-amunoz (telehacking-ro-amunoz)
```

Figura 5.12: Salida satisfactoria tras añadir la nueva clave RSA.

- Activada la clave, ya es posible descargar el proyecto. Para ello habrá que ejecutar el comando `git clone git@bitbucket.org:NikNitroX/telehacking.git`[5]. En la imagen 5.13 es posible ver la ejecución del comando y el interior de la carpeta clonada.

Hecho todo lo anterior, es momento de ir al archivo telehacking/TeleHacking/settings.py para modificar las variables globales:

- SERVER_NAME y CLIENT_NAME deben tener como valores los nombres de las máquinas servidor y cliente. Esto servirá para distinguir en qué máquina se encuentran las tareas de Celery durante la ejecución de las mismas.

[5]Es necesario asegurarse de que no exista en la carpeta actual ningún elemento llamado "telehacking" (sin las comillas) para evitar conflictos durante la descarga.

```
root@kali-server:~# git clone git@bitbucket.org:NikNitroX/telehacking.git
Cloning into 'telehacking'...
remote: Counting objects: 586, done.
remote: Compressing objects: 100% (163/163), done.
remote: Total 586 (delta 99), reused 0 (delta 0)
Receiving objects: 100% (586/586), 1.16 MiB | 1.95 MiB/s, done.
Resolving deltas: 100% (255/255), done.
root@kali-server:~# ls telehacking/
api                Cracking    manage.py  requirements.txt  static       users
archivos_de_prueba db.sqlite3  README.md  Sniffing          TeleHacking  WiFiNets
root@kali-server:~#
```

Figura 5.13: Salida satisfactoria tras la clonación del repositorio.

- SERVER_IP y CLIENT_IP tendrán como valor las IPs de ambas máquinas.
- AP_LIST contendrá una lista con los nombres de los APs a auditar. Esto impide incurrir en temas legales por romper claves a las que no se esté autorizado.

Se puede ver un ejemplo de estas variables en la imagen 5.14.

```
16  # BEGIN Global variables
17  SERVER_NAME = 'kali-server'
18  CLIENT_NAME = 'kali-raspberrypi'
19
20  SERVER_IP = '192.168.1.44'
21  CLIENT_IP = '192.168.1.40'
22
23  AP_LIST = ['LaFamiliaCebolleta',
24             'nitrowifi',
25             'Xiaomi_1951',
26             'Router_Pruebas_UNIR']
27  # END Global variables
```

Figura 5.14: Lista de las variables globales de la aplicación.

Antes de poder iniciar la aplicación, será necesario instalar y ejecutar una serie de procesos, los cuales permitirán un correcto funcionamiento de la misma. La primera de ellas será Celery, que se encargará del lanzamiento y ejecución de tareas asíncronas de forma periódica.

5.2.2. Celery

En el caso de Celery, solo es necesario asegurarse de que se está ejecutando correctamente. Para ello se lanzarán los comandos

`/etc/init.d/celerybeat status`

`/etc/init.d/celeryd status`

esperando una salida similar a la de la imagen **??**.

En el caso de que no se estén ejecutando correctamente, debiera bastar con hacer un restart de ambos servicios.

5.2.3. RabbitMQ

Además, será necesario tener un proceso que se encargue de la gestión de la cola de las tareas, para que no se pierdan si el Worker está ocupado con cualquier otra. El proceso encargado de esto será RabbitMQ[6], el cual sí será necesario instalarlo como un paquete más del sistema operativo. Con el comando `apt-get install rabbitmq-server` podrá ser instalado en Kali, y para lanzarlo como proceso se podrá usar el comando `/etc/init.d/rabbitmq-server start`.

Durante la ejecución del proyecto se podrán ver las tareas encoladas en RabbitMQ mediante la ejecución del comando`rabbitmqctl list_queues`. Esto permitirá ver si el Worker de Celery está consumiendo correctamente las tareas o se le van acumulando.

Es posible ver una captura de la ejecución de este comando en la imagen 5.15.

```
16  # BEGIN Global variables
17  SERVER_NAME = 'kali-server'
18  CLIENT_NAME = 'kali-raspberrypi'
19
20  SERVER_IP = '192.168.1.44'
21  CLIENT_IP = '192.168.1.40'
22
23  AP_LIST = ['LaFamiliaCebolleta',
24             'nitrowifi',
25             'Xiaomi_1951',
26             'Router_Pruebas_UNIR']
27  # END Global variables
```

Figura 5.15: Ejecución del comando de RabbitMQ para listar las tareas pendientes de ejecutar.

5.2.4. PostgreSQL

Django, por defecto, monta una base de datos SQLite[7]. Esta base de datos es muy cómoda para entornos de prueba, ya que está formada por un único archivo, de poco peso, y no necesita de instalación ninguna. Sin embargo, esta base de datos tampoco permite conexiones en paralelo, y la aplicación está diseñada para trabajar de forma asíncrona, por lo que es necesario instalar otra base de datos con mejores prestaciones. La base de datos elegida ha sido PostgreSQL[8], y la instalación se ha hecho siguiendo el manual (M. Mele, 2016). Para instalar la base de datos bastará con ejecutar `apt-get install postgresql`, mientras que para configurarla se lanzará

Además, será necesario permitir la conexión del cliente a la base de datos. Para ello será necesario acceder al archivo de configuración de PostgreSQL (ubicado por defecto en la ruta /et-

[6] https://www.rabbitmq.com
[7] https://sqlite.org/index.html
[8] https://www.postgresql.org

```
su -u postgres -c "psql  << EOF
CREATE ROLE adrian WITH LOGIN PASSWORD password;
CREATE DATABASE telehacking ENCODING UTF-8;
GRANT ALL PRIVILEGES ON DATABASE telehacking TO adrian;
ALTER USER adrian CREATEDB;
ALTER DATABASE telehacking OWNER TO adrian
EOF"
```

Figura 5.16: Código de configuración de PostgreSQL

c/postgresql/10/main/postgresql.conf) y cambiar la línea *listen_addresses = '127.0.0.1'* por *listen_addresses = '*'*. A continuación añadiremos al archivo /etc/postgresql/10/main/pg_hba.conf la línea:

`host telehacking adrian 192.168.1.44 trust`

siendo *192.168.1.44* la IP del dispositivo cliente.

A continuación, se ejecutará un reinicio del servicio con `/etc/init.d/postgresql restart`, asegurando el correcto reinicio del servicio, como se puede ver en la imagen **??**.

5.2.5. Python

El paso siguiente será configurar correctamente los paquetes de Python y el entorno virtual de ejecución. Para comenzar, se instalarán los paquetes de sistema referentes a Python3 y virtualenv usando el comando `apt-get install -y python3 virtualenv`.

Hecho esto, se procederá a crear un entorno virtual con la ejecución del comando `virtualenv .venv --python=/usr/bin/python3` como se ve en la imagen 5.17. Una vez creado, se activará con el comando `source venv/bin/activate`. Si todo funciona correctamente, se podrá ver, a la izquierda del prompt y entre paréntesis, el nombre del entorno virtual. Para salir del entorno virtual basta con ejecutar el comando `deactivate`

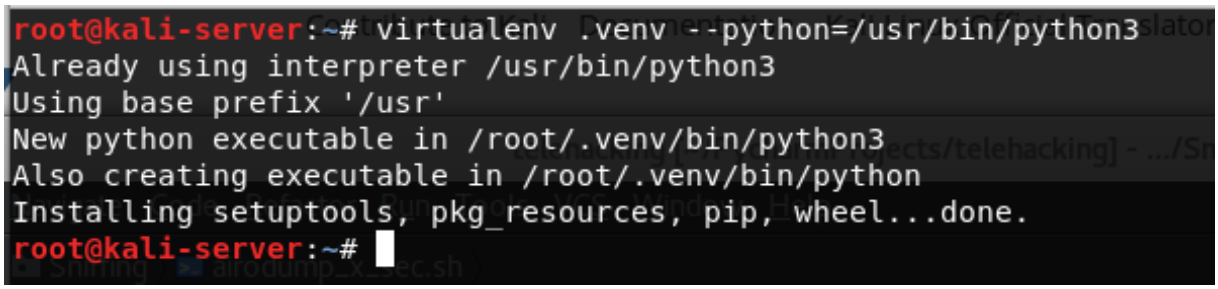


Figura 5.17: Creación de un entorno virtual para Python 3.

A continuación, se instalarán todos los paquetes necesarios, los cuales están listados en el archivo *requirements.txt*. Para instalarlos todos de una vez, se usará el comando pip,con el argumento -r, que permite usar un fichero: `pip install -r requirements`. Es posible ver un ejemplo de esto en las imágenes 5.18 y 5.19[9]

[9]La traza de la instalación era demasiado larga para mostrarla entera, por lo que solo se muestra el inicio y el final de la misma.

```
root@kali-server:~# source .venv/bin/activate
(.venv) root@kali-server:~# pip install -r telehacking/requirements.txt
Collecting amqp==2.3.2 (from -r telehacking/requirements.txt (line 1))
  Using cached https://files.pythonhosted.org/packages/7f/cf/12d4611fc67babd4ae250c9e8249
2.3.2-py2.py3-none-any.whl
Collecting anyjson==0.3.3 (from -r telehacking/requirements.txt (line 2))
Collecting Babel==2.6.0 (from -r telehacking/requirements.txt (line 3))
  Using cached https://files.pythonhosted.org/packages/b8/ad/c6f60602d3ee3d92fbed87675b6f
-2.6.0-py2.py3-none-any.whl
Collecting billiard==3.5.0.4 (from -r telehacking/requirements.txt (line 4))
Collecting celery==4.2.1 (from -r telehacking/requirements.txt (line 5))
  Using cached https://files.pythonhosted.org/packages/e8/58/2a0b1067ab2c12131b5c089dfc57
y-4.2.1-py2.py3-none-any.whl
```

Figura 5.18: Ejecución del comando de instalación de paquetes de Python.

```
Installing collected packages: vine, amqp, anyjson, pytz, Babel, billiard, kombu, celer
y, certifi, chardet, Django, django-bootstrap3, django-celery, django-forms-bootstrap,
djangorestframework, django-rest-framework, django-tables2, tornado, flower, idna, psyc
opg2, psycopg2-binary, urllib3, requests
Successfully installed Babel-2.6.0 Django-2.1.1 amqp-2.3.2 anyjson-0.3.3 billiard-3.5.0
.4 celery-4.2.1 certifi-2018.4.16 chardet-3.0.4 django-bootstrap3-9.1.0 django-celery-3
.2.2 django-forms-bootstrap-3.1.0 django-rest-framework-0.1.0 django-tables2-1.21.2 dja
ngorestframework-3.8.2 flower-0.9.2 idna-2.7 kombu-4.2.1 psycopg2-2.7.5 psycopg2-binary
-2.7.5 pytz-2017.3 requests-2.19.1 tornado-5.1 urllib3-1.23 vine-1.1.4
(.venv) root@kali-server:~#
```

Figura 5.19: Salida final del comando de instalación de paquetes en Python.

Además, por seguridad, las credenciales de la base de datos no están guardadas en el proyecto, sino que este las toma de las variables de sistema. Para hacer esto más automático, se escribirán en el comando *activate* del entorno virtual. Para hacer esto se abrirá el archivo *bin/activate*[10] y añadir al final la exportación de las variables, como es posible ver en la imagen 5.20. Para que se borren tras lanzar el comando *deactivate*, se editará (en el mismo fichero) la función deactivate, añadiendo al final las líneas necesarias, tal y como se muestra en la imagen 5.21.

```
# This should detect bash and zsh, which have a has
# be called to get it to forget past commands.  Wit
# past commands the $PATH changes we made may not b
if [ -n "${BASH:-}" -o -n "${ZSH_VERSION:-}" ] ; th
    hash -r
fi
export DATABASE_NAME='telehacking'
export DATABASE_USER='adrian'
export DATABASE_PASSWORD='password'
```

Figura 5.20: Adición de líneas al comando *activate* para usar las credenciales de la base de datos.

[10]Es necesario recordar que el archivo a editar es el del entorno virtual. No confundir con la carpeta *bin* de la raíz del directorio.

```
    unset VIRTUAL_ENV
    if [ ! "$1" = "nondestructive" ] ; then
    # Self destruct!
        unset -f deactivate
    fi
    unset DATABASE_NAME
    unset DATABASE_USER
    unset DATABASE_PASSWORD
}
```

Figura 5.21: Adición de líneas al comando *activate* para eliminar las credenciales de la base de datos.

Finalmente, también es necesario definir en el archivo TeleHacking/settings.py los nombres de los equipos, así como sus IP's. También hay que tener en cuenta que los equipos tengan conexión entre sí[11].

Ya se encuentran cubiertas todas las dependencias del sistema. A continuación se comentarán los resultados de la ejecución del proyecto.

5.3. Ejecución del proyecto

Teniendo la certeza de que están corriendo correctamente los procesos de Celery Beat, Celery Worker y RabbitMQ en el cliente, y PostgreSQL en el servidor, es hora de lanzar la aplicación.

Para ello, es necesario entrar en la carpeta principal del proyecto y ejecutar el comando `python -W ignore manage.py runserver 192.168.1.44:80`[12]. La salida será similar a la que se puede ver en la imagen 5.22.

```
(venv) root@kali-server:~/telehacking# python -W ignore manage.py runserver 192.168.1.44:80
Starting TeleHacking execution...
Starting TeleHacking execution...
Performing system checks...

System check identified no issues (0 silenced).
September 14, 2018 - 18:51:38
Django version 2.1.1, using settings 'TeleHacking.settings'
Starting development server at http://192.168.1.44:80/
Quit the server with CONTROL-C.
```

Figura 5.22: Salida del comando *python manage.py runserver*, que inicia el servidor.

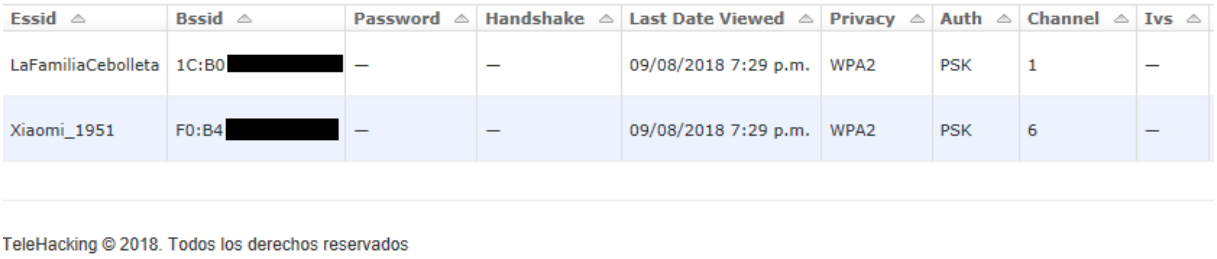
[11] Es necesario asegurarse de que puede haber firewalls, routers y demás dispositivos interceptando la conexión, por lo que se deben configurar todos estos dispositivos de forma adecuada para permitir la comunicación cliente-servidor

[12] La IP usada debe ser la que tiene la propia máquina. En otro caso será inaccesible desde fuera.

Este comando no es necesario que sea lanzado en la Raspberry Pi, pues esta se basta con Celery Beat y Celery Worker en ejecución.

Ahora se listarán los hechos verificados durante la ejecución del proyecto:

- Tras lanzar el comando anterior, será posible acceder a la interfaz visual accediendo a la IP *192.168.1.44* desde cualquier navegador, obteniendo una vista similar a la de la imagen 5.23.

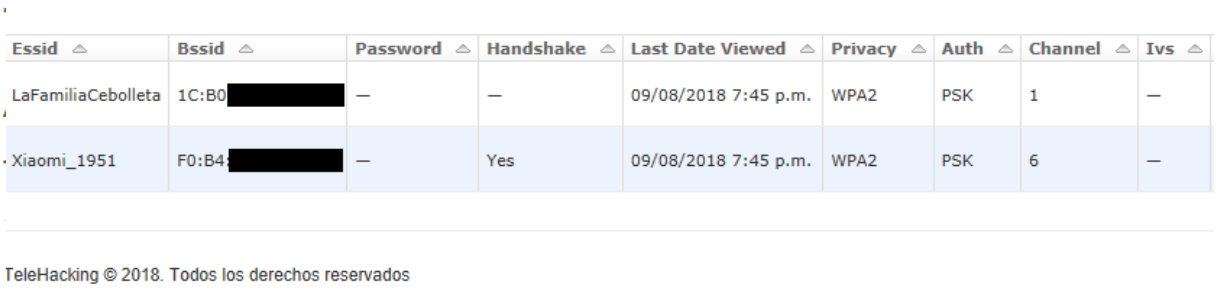


Essid	Bssid	Password	Handshake	Last Date Viewed	Privacy	Auth	Channel	Ivs
LaFamiliaCebolleta	1C:B0[redacted]	—	—	09/08/2018 7:29 p.m.	WPA2	PSK	1	—
Xiaomi_1951	F0:B4[redacted]	—	—	09/08/2018 7:29 p.m.	WPA2	PSK	6	—

TeleHacking © 2018. Todos los derechos reservados

Figura 5.23: Vista HTML que devuelve la ejecución de la aplicación.

- Como se puede ver, ya ha encontrado dos posibles redes vulnerables[13]. Será necesario esperar a que algún cliente se conecte para poder capturar el handshake.
- Pasado un tiempo, es posible observar en la imagen 5.24 cómo la aplicación ha conseguido obtener un handshake para la red *Xiaomi_1951*. Seguramente en estos momentos el servidor esté ejecutando un ataque de diccionario contra dicha red.



Essid	Bssid	Password	Handshake	Last Date Viewed	Privacy	Auth	Channel	Ivs
LaFamiliaCebolleta	1C:B0[redacted]	—	—	09/08/2018 7:45 p.m.	WPA2	PSK	1	—
Xiaomi_1951	F0:B4[redacted]	—	Yes	09/08/2018 7:45 p.m.	WPA2	PSK	6	—

TeleHacking © 2018. Todos los derechos reservados

Figura 5.24: Vista HTML cuando se obtiene un handshake.

- Efectivamente, poco rato después el servidor ha sido capaz de descifrar el handshake anterior, obteniendo la clave de la red *Xiaomi_1951*. Además, ha encontrado una nueva red: *PruebaWEP*, para la que ha conseguido varios IVs.

[13]Hay que recordar que en la imagen 5.14 se definió una lista blanca de redes. Esto es lo que permite que filtre los resultados totales y solo muestre estas dos.

Essid	Bssid	Password	Handshake	Last Date Viewed	Privacy	Auth	Channel	Ivs
PruebaWEP	00:16[redacted]	—	—	09/08/2018 8:05 p.m.	WEP	—	9	42
LaFamiliaCebolleta	1C:B0[redacted]	—	—	09/08/2018 8:05 p.m.	WPA2	PSK	1	—
Xiaomi_1951	F0:B4[redacted]	MasterSeguridadUNIR18	Yes	09/08/2018 8:05 p.m.	WPA2	PSK	6	—

Figura 5.25: Vista HTML tras crackear el handshake, y con una red WEP encontrada.

- Dos horas más tarde se revisa de nuevo la interfaz, descubriendo que se han conseguido 853.678 IVs para la red *PruebaWEP*, y que han sido suficientes para poder obtener la password. Se puede ver una captura de la misma en la imagen 5.26

Essid	Bssid	Password	Handshake	Last Date Viewed	Privacy	Auth	Channel	Ivs
PruebaWEP	00:16[redacted]	1248163264128256512aaaaaaa	—	09/08/2018 9:55 p.m.	WEP	—	9	853678
LaFamiliaCebolleta	1C:B0[redacted]	—	—	09/08/2018 9:55 p.m.	WPA2	PSK	1	—
Xiaomi_1951	F0:B4[redacted]	MasterSeguridadUNIR18	Yes	09/08/2018 9:55 p.m.	WPA2	PSK	6	—

Figura 5.26: Vista HTML tras crackear la red WEP

Es posible concluir el experimento, habiendo llegado ya al objetivo principal de este trabajo.

Capítulo 6

Conclusiones y trabajo futuro

El objetivo principal del trabajo se ha centrado en crear una aplicación capaz de automatizar, con ayuda de un sistema remoto de mayor potencia, el análisis de redes inalámbricas con el fin de facilitar y agilizar el trabajo de auditorías, ya sea tanto para empresas como particulares. Esto hace posible que el auditor de seguridad pueda dedicar más tiempo a analizar otros vectores de ataque mientras tanto, economizando el trabajo de la auditoría.

La tecnología utilizada permite, en última instancia, obtener la clave de una red inalámbrica o, al menos, comprobar su seguridad con los principales diccionarios que se utilizan en este tipo de ataques. Para romper una clave de red (ya sea WEP o WPA) de una forma tan rápida como se ha hecho en este estudio, es necesario que se den ciertas circunstancias, como por ejemplo que la distancia al AP (en el caso de las claves WEP) sea la mínima posible, o que un nuevo cliente se conecte a éste (en el caso de las claves WPA).

En el caso de redes con cifrado WEP, es solo cuestión de tiempo obtener la clave, mientras que para cifrados WPA, al menos se probará la robustez de la contraseña. Suponiendo un sistema similar al de (P. Kennedy, 2018), se podrían probar $1{,}724x10^{10}$ contraseñas por hora. Sabiendo que, para una contraseña numérica de 10 caracteres hay 10^{10} posibilidades, se tardaría menos de media hora en probar todas las posibles combinaciones; todo depende de la potencia del servidor utilizado.

Bien es cierto que las técnicas actuales para crackear las redes inalámbricas son más avanzadas, pues tienen un gran abanico de ataques (vistos en 2.2.1) que no se usan en este trabajo, lo cual deja una puerta abierta al desarrollo de nuevos módulos de selección y ejecución automática de ataques según las variables del entorno, además de dejar abierta la implementación del ataque a las redes WPA usando el vector de ataque del WPS, como se comentó en 2.2.3.

Otra posible vía de trabajo futuro podría ser la implementación de autenticación y SSL en las comunicaciones, como ya se habló en el apartado 2.3.4.

Asímismo quedan también abiertas varias vías de actualización del trabajo mediante el desarrollo de nuevos módulos y funcionalidades como pueden ser el análisis de dispositivos dentro de la red local, de sus servicios, de dispositivos bluetooth, etc. Una de las ventajas de haber usado Django en este proyecto es que la adición de nuevos módulos no implica apenas cambios en el proyecto actual, sino que únicamente es necesario desarrollar un nuevo módulo, con cambios mínimos para la integración.

Bibliografía

1&1 (2017). MAC spoofing: qué es y cuándo se utiliza. https://www.1and1.es/digitalguide/servidores/know-how/que-es-el-mac-spoofing/. [Online].

1keydata.com (2018). SQL Clave primaria. https://www.1keydata.com/es/sql/sql-clave-primaria.php. [Online].

A. Alanjawi (2018). Unix Signals. https://people.cs.pitt.edu/~alanjawi/cs449/code/shell/UnixSignals.htm. [Online].

A. Crespo (2016). Alfa Network AWUS036H. https://www.redeszone.net/alfa-network/awus036h/. [Online].

A. Crespo (2017). Qué es un servidor RADIUS y cómo funciona. https://www.redeszone.net/2017/06/02/servidor-radius-funciona/. [Online].

Acosta-López, A., Melo-Monroy, E. Y., and Linares-Murcia, P. A. (2018). Evaluation of the wpa2-psk wireless network security protocol using the linset and aircrack-ng tools. *Revista Facultad de Ingenieria*, 27(47):1–15.

Aircrack-ng (2018). Aircrack-ng suite. https://www.aircrack-ng.org/doku.php?id=Main#aircrack-ng_suite. [Online].

Aircrack-ng.org (2007). Tutorial: Aircrack-ng Suite under Windows for Dummies. https://www.aircrack-ng.org/doku.php?id=es:aircrack-ng_suite-under-windows_for_dummies. [Online].

Aircrack-ng.org (2018a). Aireplay-ng. https://www.aircrack-ng.org/doku.php?id=aireplay-ng. [Online].

Aircrack-ng.org (2018b). Airoscript-ng. https://www.aircrack-ng.org/doku.php?id=airoscript-ng. [Online].

Aked, S., Bolan, C., and Brand, M. (2012). A proposed method for examining wireless device vulnerability to brute force attacks via wps external registrar pin authentication design vulnerability.

Alliance, W. (2003a). Wi-fi alliance announces first products certified for wi-fi protected access security. [Internet; accedido 8-junio-2018].

Alliance, W.-F. (2003b). Wi-fi protected access: Strong, standards-based, interoperable security for today's wi-fi networks. *White paper, University of Cape Town*, pages 492–495.

BBVA (2016). API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos. `https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos`. [Online].

Belshe, M., Thomson, M., and Peon, R. (2015). Rfc 7540. *Hypertext transfer protocol version*, 2.

Berners-Lee, T., Fielding, R., and Frystyk, H. (2005). Rfc 1945: Hypertext transfer protocol—http/1.0, may 1996. *Status: INFORMATIONAL*, 61.

Bisel, L. D. (2007). The role of ssl in cybersecutiry. *IT Professional*, 9(2).

Bitbucket Support (2018). Set up an SSH key. `https://confluence.atlassian.com/bitbucket/set-up-an-ssh-key-728138079.html`. [Online].

Canetti, R. and Krawczyk, H. (2001). Analysis of key-exchange protocols and their use for building secure channels. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 453–474. Springer.

ConocimientosWeb (2013). El Algoritmo determinista. `https://www.conocimientosweb.net/dcmt/ficha17.html`. [Online].

Data-Alliance (2018). RP-SMA y SMA: Conectores y Cables para Antenas. `http://es.data-alliance.net/conectores-sma-y-rp-sma/`. [Online].

E. Vinda (2013). Sencilla explicación sobre AES. `https://es.slideshare.net/elvisvinda/sencilla-explicacin-sobre-aes`. [Online].

EcuRed (2018a). Multiplataforma. `https://www.ecured.cu/Multiplataforma`. [Online].

EcuRed (2018b). WEP. `https://www.ecured.cu/WEP`. [Online].

evaluandosoftware (2018). ¿Qué es desarrollo de software ágil? `http://www.evaluandosoftware.com/desarrollo-software-agil/`. [Online].

F. Escribano (2016). Descubriendo RabbitMQ: una solución para colas de mensajería . `https://www.beeva.com/beeva-view/tecnologia/descubriendo-rabbitmq-una-solucion-para-colas-de-mensajeria/`. [Online].

García, M. (2015). Python distribuido: Celery. `https://magmax.org/blog/python-distribuido-celery/`. [Online].

hipertextual (2014). Qué es un sistema de control de versiones y por qué es tan importante. `https://hipertextual.com/archivo/2014/04/sistema-control-versiones/`. [Online].

InformaticaESP (2012). ¿Qué es una tarjeta de red y para qué sirve? https://informaticaesp.wordpress.com/2012/02/11/que-es-una-tarjeta-de-red-y-para-que-sirve/. [Online].

Intel (2018). Descripción general y los tipos de EAP 802.1X. https://www.intel.la/content/www/xl/es/support/articles/000006999/network-and-i-o/wireless-networking.html. [Online].

J. García (2018). Qué es un ataque por fuerza bruta. https://faqoff.es/que-es-un-ataque-por-fuerza-bruta/. [Online].

J. Long (2018). Frontend vs. Backend. http://blog.teamtreehouse.com/i-dont-speak-your-language-frontend-vs-backend. [Online].

J. Lopez Arredondo (2015). ¿SABES PARA QUÉ SIRVE EL BOTÓN WPS DE TU ROUTER Y LO ÚTIL QUE ES? https://cincodias.elpais.com/cincodias/2015/06/12/lifestyle/1434110441_418389.html. [Online].

J. Penalba (2017). NFC: qué es y para qué sirve. https://www.xataka.com/moviles/nfc-que-es-y-para-que-sirve. [Online].

J. Snyder, R. Thayer (2004). Explaining TKIP. https://www.networkworld.com/article/2325772/network-security/explaining-tkip.html. [Online].

J. Steube (2018). New attack on WPA/WPA2 using PMKID. https://hashcat.net/forum/thread-7717.html. [Online].

J.L. Avalos (2018). Metodología XP. https://es.scribd.com/doc/57257203/Metodologia-XP. [Online].

K. Kao (2014). MD5 y SHA256 Checksum ¿qué es, para qué sirve y como saberlo con Hash Checker? . http://quepuedohacerconlinux.blogspot.com/2014/01/md5-y-sha256-checksum-que-es-para-que.html. [Online].

Krawczyk, H. (2001). The order of encryption and authentication for protecting communications (or: How secure is ssl?). In *Annual International Cryptology Conference*, pages 310–331. Springer.

Lehembre, G. (2006). Seguridad wi-fi–wep, wpa y wpa2. *Recuperado el*, 9(10).

Libros Web (2018). Polimorfismo. https://librosweb.es/libro/algoritmos_python/capitulo_15/polimorfismo.html. [Online].

Lindholm, T., Yellin, F., Bracha, G., and Buckley, A. (2014). *The Java virtual machine specification*. Pearson Education.

Liu, J., Ye, X., Zhang, J., and Li, J. (2008). Security verification of 802.11 i 4-way handshake protocol. In *Communications, 2008. ICC'08. IEEE International Conference on*, pages 1642–1647. IEEE.

Lucidchart (2018). Tutorial de diagrama de secuencia UML . https://www.lucidchart.com/pages/es/diagrama-de-secuencia. [Online].

M. Mele (2016). Install and Configure PostgreSQL. http://www.marinamele.com/taskbuster-django-tutorial/install-and-configure-posgresql-for-django. [Online].

M. Rouse (2009). What is open source? https://whatis.techtarget.com/definition/open-source. [Online].

Muñoz, A. (2017). *Localización en interiores usando BLE*. PhD thesis, Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga.

Newham, C. and Rosenblatt, B. (2005). *Learning the bash shell: Unix shell programming*. .ºReilly Media, Inc.".

Offensive Security (2018). What is the Default Kali root Password ? https://docs.kali.org/faq/what-is-the-default-kali-root-password. [Online].

Oracle (2018). Modelo de arquitectura del protocolo TCP/IP. https://docs.oracle.com/cd/E19957-01/820-2981/ipov-10/. [Online].

Orcero, D. (2018). *Pentesting Con Kali*. Lulu.com.

Ostrand, T. (2002). Black-box testing. *Encyclopedia of Software Engineering*.

P. Arianto (2015). How to Install and Configure UFW. https://www.tecmint.com/how-to-install-and-configure-ufw-firewall/. [Online].

P. Kennedy (2018). Password Cracking with 8x NVIDIA GTX 1080 Ti GPUs. https://www.servethehome.com/password-cracking-with-8x-nvidia-gtx-1080-ti-gpus/. [Online].

P.G. Bejerano (2015). ¿Qué es el filtrado MAC de tu router? https://blogthinkbig.com/filtrado-mac-router. [Online].

ProyectosAgiles.com (2018). Qué es SCRUM. https://proyectosagiles.org/que-es-scrum/. [Online].

Qiu, X. (2016). What is a compiler?

Reddy, S. V., Ramani, K. S., Rijutha, K., Ali, S. M., and Reddy, C. P. (2010). Wireless hacking-a wifi hack by cracking wep. In *Education Technology and Computer (ICETC), 2010 2nd International Conference on*, volume 1, pages V1–189. IEEE.

rm-rf.es (2009). Configurar tarjeta de red en modo promiscuo. http://rm-rf.es/configurar-tarjeta-de-red-en-modo-promiscuo/. [Online].

Rowland, C. H. (2002). Intrusion detection system. US Patent 6,405,318.

S. Noriega (2015). Autoridad Certificadora en Internet. https://www.certsuperior.com/Blog/autoridad-certificadora-en-internet. [Online].

Scott D. Lowe (2011). NAT vs. bridged network: A simple diagram. http://techgenix.com/nat-vs-bridged-network-a-simple-diagram-178/. [Online].

Segu.Info (2009). Exploit. https://www.segu-info.com.ar/malware/exploit.htm. [Online].

Seid, H. A. and Lespagnol, A. (1998). Virtual private network. US Patent 5,768,271.

Shapiro, A. (2017). Krack and the ethics of disclosure.

Stroustrup, B. (1988). What is object-oriented programming? *IEEE software*, 5(3):10–20.

Stubblefield, A., Ioannidis, J., Rubin, A. D., et al. (2002). Using the fluhrer, mantin, and shamir attack to break wep. In *NDSS*.

techopedia (2018). Wi-Fi Protected Access Pre-Shared Key (WPA-PSK). https://www.techopedia.com/definition/22921/wi-fi-protected-access-pre-shared-key-wpa-psk. [Online].

TechTarget (2008). What is Open System Authentication (OSA)? https://searchsecurity.techtarget.com/definition/Open-System-Authentication-OSA. [Online].

Tecnología&Informática (2018). Que es un Firewall y como funciona. https://tecnologia-informatica.com/que-es-firewall-como-funciona-tipos-firewall/. [Online].

Todo-Redes (2018). Access point. https://todo-redes.com/equipos-de-redes/access-point-punto-de-acceso. [Online].

Ulloa Santana, J. A. and Fonseca Sánchez, V. H. (2012). *Análisis de la gestión de seguridad de los protocolos WPA y WPA2 en el tráfico de datos de una red LAN con tecnología WiFi*. PhD thesis, Universidad Nacional Autónoma de Nicaragua, Managua.

ValorTop (2017). ¿Qué es WiFi? ¿Qué significa y para qué sirve? http://www.valortop.com/blog/que-es-wifi-que-significa-y-para-que-sirve. [Online].

Vila Ríos, J. R. (2017). Redes wifi:¿ realmente se pueden proteger?

vk496 (2014). Herramienta Linset. https://github.com/vk496/linset. [Online].

VMware (2018a). Installing VMware Tools. https://www.vmware.com/support/ws55/doc/new_guest_tools_ws.html. [Online].

VMware (2018b). Mounting Shared Folders in a Linux Guest. https://pubs.vmware.com/workstation-9/index.jsp?topic=%2Fcom.vmware.ws.using.doc%2FGUID-AB5C80FE-9B8A-4899-8186-3DB8201B1758.html. [Online].

Wifi Alliance (2018). Wi-Fi Alliance® introduces Wi-Fi CERTIFIED WPA3™ security. https://www.wi-fi.org/news-events/newsroom/wi-fi-alliance-introduces-wi-fi-certified-wpa3-security. [Online].

Wikipedia (2018). Wired equivalent privacy — wikipedia, la enciclopedia libre. [Internet; descargado 8-junio-2018].

Wong, S. (2003). The evolution of wireless security in 802.11 networks: Wep, wpa and 802.11 standards.