Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Air Profiling Network: Sistema para la realización de perfiles a través de capturas de red y Tacyt

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: ISMAEL MONJE BRAU

Tutor: ROMÁN GARCÍA GARCÍA

2016-2017

Resumen

Todo el tráfico que circula por una red deja un gran rastro e información de los hábitos y situaciones de las personas en su día a día. La incorporación del Smartphone en la vida diaria hace que el tráfico aumente y que datos privados de hábitos o usos del día a día queden expuestos a que alguien pueda procesarlos. Las redes WIFI inalámbricas no seguras instaladas en sitios abiertos y donde las personas se conectan de forma masiva son un punto crítico donde recopilar datos sobre los hábitos y datos privados de las personas.

El presente proyecto propone el diseño e implementación de un sistema capaz de recibir capturas de red y procesarlos generando un perfil de los usuarios que aparecen en la captura. El objetivo es poder crear un Timeline del usuario y la utilización de la red a nivel de Internet.

Palabras clave: wifi, perfil, Tacyt, red, análisis de datos.

Abstract

Online traffic leaves a track of information of people's habits and daily life situations. The incorporation of the Smartphone in people's current daily lives increases online traffic and brings to light private data about personal habits and everyday uses, resulting exposed to be processed. WiFi wireless unsecure networks, which are settled in open spaces and where people can massively be connected, are a critical point where people's habits and private data can be collected.

This project proposes the design and implementation of a system capable of receiving network captures that can be processed, generating a user's profile of the users who appear in the capture. The main objective is to create a user's timeline and to use this network at an Internet level.

Keywords: wifi, profile, Tacyt, network, analysis of data

Abstract

Tot el tràfic que circula per una xarxa deixa un gran rastre i informació dels hàbits i situacions de les persones en el seu dia a dia. La incorporació del Smartphone a la vida diària fa que el tràfic augmente i que dades privades de hàbits o usos del dia a dia queden exposats a que algú puga processar-los. Les xarxes WiFi sense fil no segures instal·lades en llocs oberts i on les persones es connecten de forma massiva son un punt crític on recopilar dades sobre els hàbits i dades privades de les persones.

El present projecte proposa el disseny i implementació d'un sistema capaç de rebre captures de xarxa i processar-los generant un perfil dels usuaris que apareixen en la captura. L'objectiu es poder crear un Timeline del usuari i la utilització de la xarxa en l'àmbit d'Internet.

Paraules clau: wifi, perfil, Tacyt, xarxa, anàlisi de dades.

Tabla de contenidos

Tabla de figuras

1. Introducción

1.1. Dedicatoria y agradecimientos

Dedico el presente trabajo a mis dos tutores Román García y Sergio de los Santos por darme su apoyo durante el transcurso del proyecto. Agradezco también a ElevenPaths por darme la oportunidad de trabajar con ellos y proporcionarme los recursos necesarios.

1.2. Motivación

Todo el tráfico que circula por una red deja un gran rastro e información de los hábitos y situaciones de las personas en su día a día. La incorporación del Smartphone en la vida diaria hace que el tráfico aumente y que datos privados de hábitos o usos del día a día queden expuestos a que alguien pueda procesarlos. Las redes WIFI inalámbricas no seguras instaladas en sitios abiertos y donde las personas se conectan de forma masiva son un punto crítico donde recopilar datos sobre los hábitos y datos privados de las personas.

El presente proyecto propone el diseño e implementación de un sistema capaz de recibir capturas de red y procesarlos generando un perfil de los usuarios que aparecen en la captura. El objetivo es poder crear un Timeline del usuario y la utilización de la red a nivel de Internet. Los datos de interés son:

Datos que pueden relacionar a un usuario con un dispositivo. Número de teléfono, direcciones MAC, nombres de personas obtenidos a través de diferentes servicios.

Datos que puedan provocar la inferencia de resultados sobre qué aplicaciones hay instaladas en un equipo o un dispositivo móvil. Estudio del protocolo DNS y creación de un pequeño motor de inferencia.

Datos que pueden provocar la inferencia sobre los hábitos o gustos de los usuarios, por ejemplo: las visitas a distintos sitios web, páginas de compras, de ocio, etcétera.

Datos geográficos de la captura y el instante en el que se llevó a cabo la comunicación.

Datos sobre la información del sistema operativo o dispositivo móvil utilizado.

1.3. Objetivos

Los objetivos para el presente proyecto son los siguientes:

- Estudio de aplicaciones y servicios que dejan rastro en las capturas de red. Estudio de qué información útil se puede obtener de ello.
- Diseño e implementación de una aplicación web que procese capturas de tráfico extrayendo la información sensible y valiosa de dicha captura.
- Diseño de módulos para procesar la información obtenida de la captura.
- Diseño y arquitectura de la base de datos donde se almacenará la información de los perfiles.
- Capacidad para presentar los perfiles de forma amistosa con el tiempo (Timeline).
- Definición de hechos o sucesos interesantes en navegaciones o aplicaciones instaladas inferidas a través del protocolo DNS.
- Integración con Tacyt para consulta de datos, a través de los resultados obtenidos con *Air Profiling Network*.

1.4. Usos

Algunos de los posibles usos que se han planteado para el proyecto son:

- **Publicidad personalizada.**

Debido a la gran cantidad de información que es capaz de obtener la aplicación se podrían realizar perfiles de gustos de los usuarios para poder mostrarles publicidad personalizada, por ejemplo, en centros comerciales.

Ya se usan técnicas como esta en otros servicios, como puede ser Google.

- **Análisis de malware.**

Debido a la integración con la plataforma Tacyt y a la posibilidad que proporciona esta para la creación de filtros, se podría detectar tráfico malicioso. Este tipo de uso podría ser muy beneficioso en redes privadas o empresariales.

- **Descubrir tráfico inadecuado en una red empresarial.**

En este tipo de uso al igual que en el anterior se podría detectar tráfico malicioso aparte de poder saber si alguno de los empleados está haciendo un mal uso de los recursos de la empresa.

- **Estudios estadísticos.**

En este tipo de uso se utilizaría la aplicación para generar estadísticas de gustos de los usuarios, horarios de mayor actividad, sistemas operativos y dispositivos más usados… Que podrían luego ser usados para otros propósitos.

2. Manual de usuario

En esta sección se van a describir las diferentes pantallas que posee la aplicación, para ver un ejemplo práctico ir directamente a el apartado 3:

2.1. Pantalla de inicio de sesión

Aunque no era un requisito del proyecto debido a la facilidad que proporciona el framework Django y al despliegue de la aplicación en internet se ha agregado una pantalla de autenticación de usuarios a través de usuario y contraseña.

Solo los administradores de la web pueden crear cuentas, no existe posibilidad de registro.

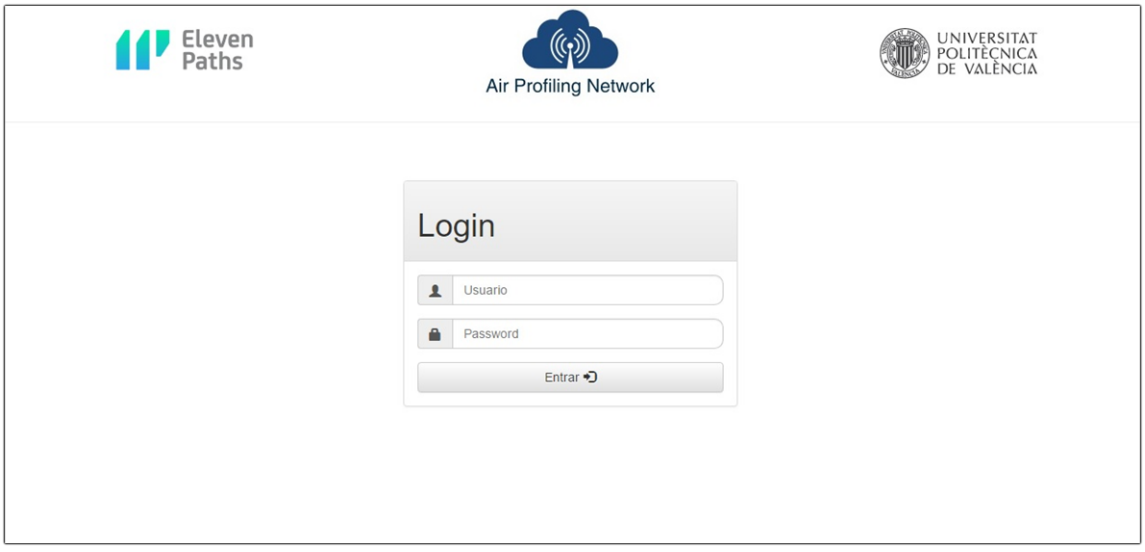

Ilustración 1. Pantalla de inicio de sesión

2.2. Pantalla del menú principal

En la pantalla de inicio de la aplicación aparece la posibilidad de subir PCAP para su procesamiento y obtener información. Debido a los pocos recursos del servidor donde se encuentra alojada la aplicación se limitó el tamaño máximo de archivo a solo 100 megabytes.

En caso de que el archivo no sea del formato correcto o exceda el tamaño máximo la aplicación devolverá un error indicándolo.

Ilustración 2. Pantalla de inicio

2.3. Pantalla de listado de capturas

En la pantalla de capturas se listan todos los archivos de capturas de red que han sido subidos a la aplicación, mostrando el nombre del archivo, el usuario que lo subió y la fecha. Mediante un pequeño buscador situado en la parte superior derecha se puede filtrar por cualquiera de estos campos.

Además, cada PCAP tiene un botón para poder ver en detalle la información obtenida.

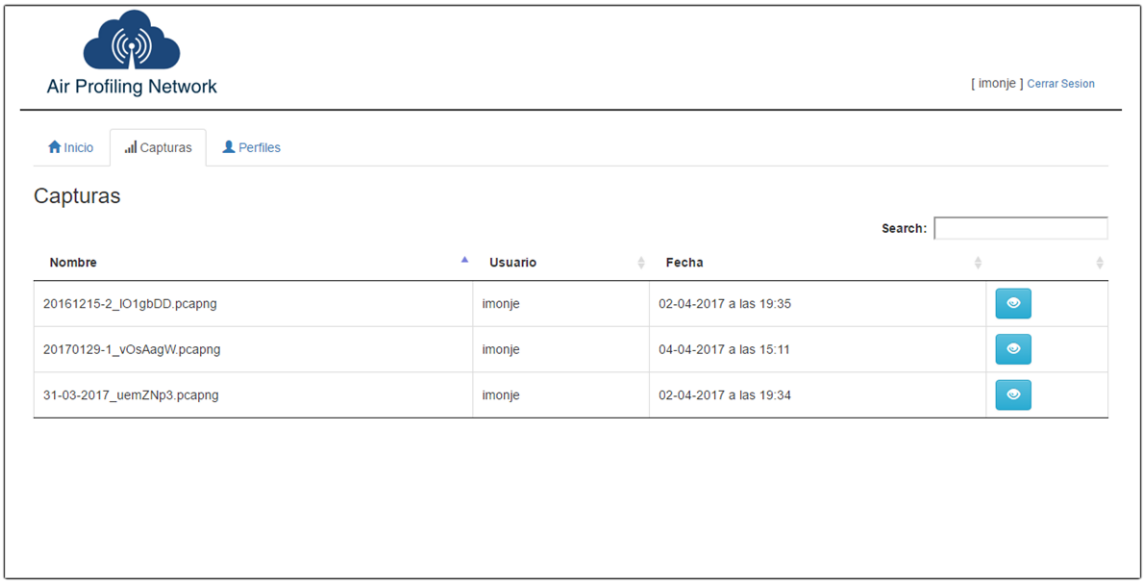

Ilustración 3. Pantalla de listado de capturas

2.4. Pantalla de captura individual

En la pantalla de captura se muestra información sobre un PCAP en concreto. Mostrando el nombre del archivo, el usuario que lo subió y la fecha.

Además, permite la opción de descargarse el archivo original y muestra un listado con los perfiles de usuario de los cuales se ha obtenido información en ese PCAP.

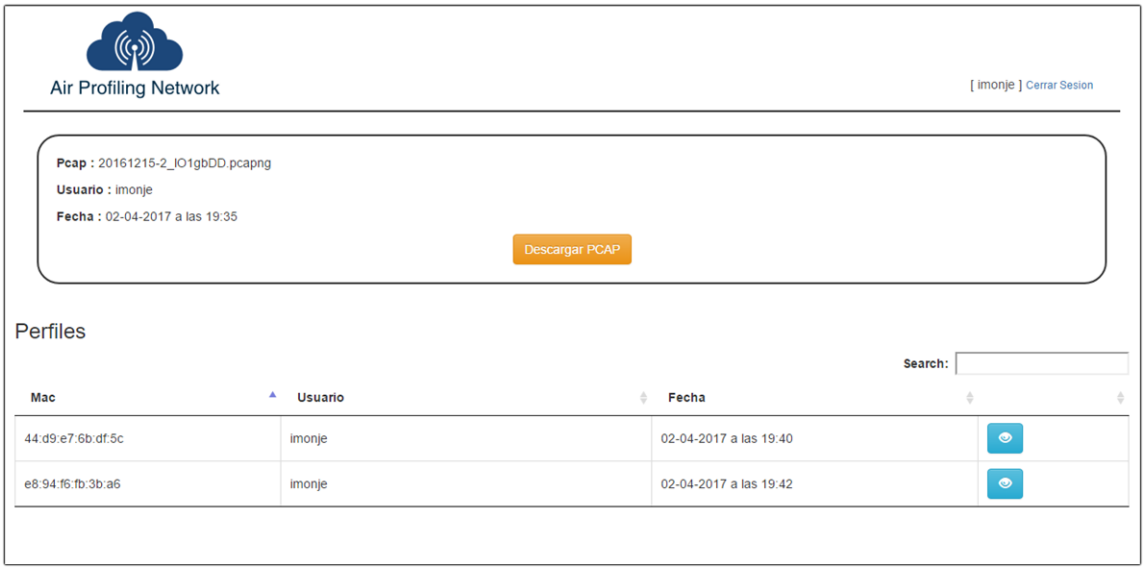

Ilustración 4. pantalla de captura individual

2.5. Pantalla de listado de perfiles

En la pantalla de perfiles se listan todos los perfiles que se han detectado en los ficheros de capturas de red que han sido procesados por la aplicación, mostrando el identificador único MAC, el usuario que subió el primer fichero donde se detectó y la fecha. Mediante un pequeño buscador situado en la parte superior derecha se puede filtrar por cualquiera de estos campos.

Además, cada perfil tiene un botón para poder ver en detalle la información obtenida.

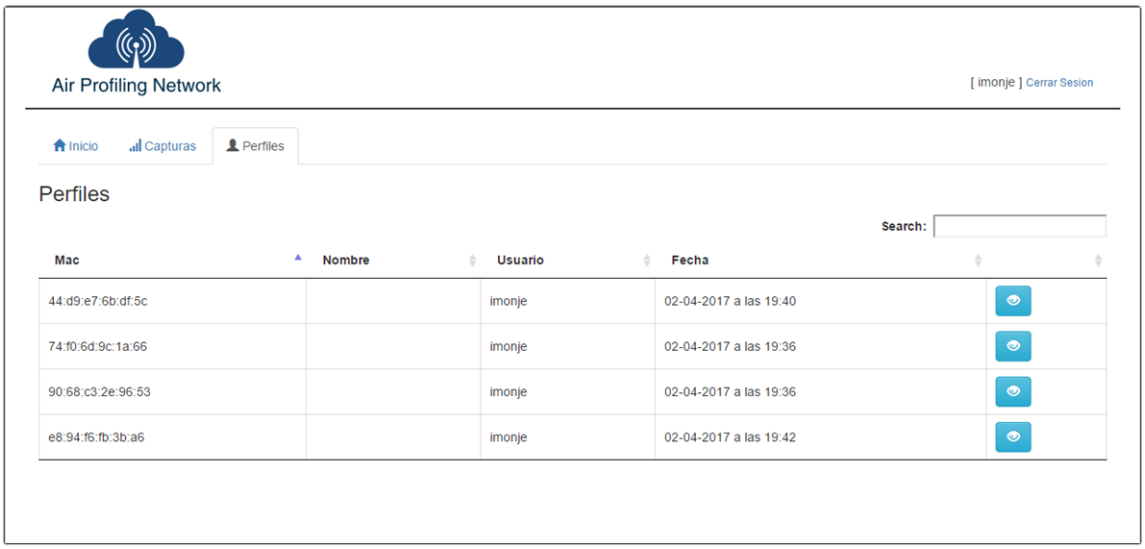

Ilustración 5. Pantalla de listado de perfiles

2.6. Pantalla de perfil individual

En la pantalla de perfil se muestra información sobre un perfil en concreto. Mostrando la MAC del dispositivo, el usuario que subió el primer fichero donde se detectó, la fecha, y en caso de disponer de los datos el fabricante asociado a la MAC y el sistema operativo del dispositivo.

Además, muestra un listado de las capturas de red en las que se ha detectado el perfil y un listado con los agentes de usuario usados en esas capturas.

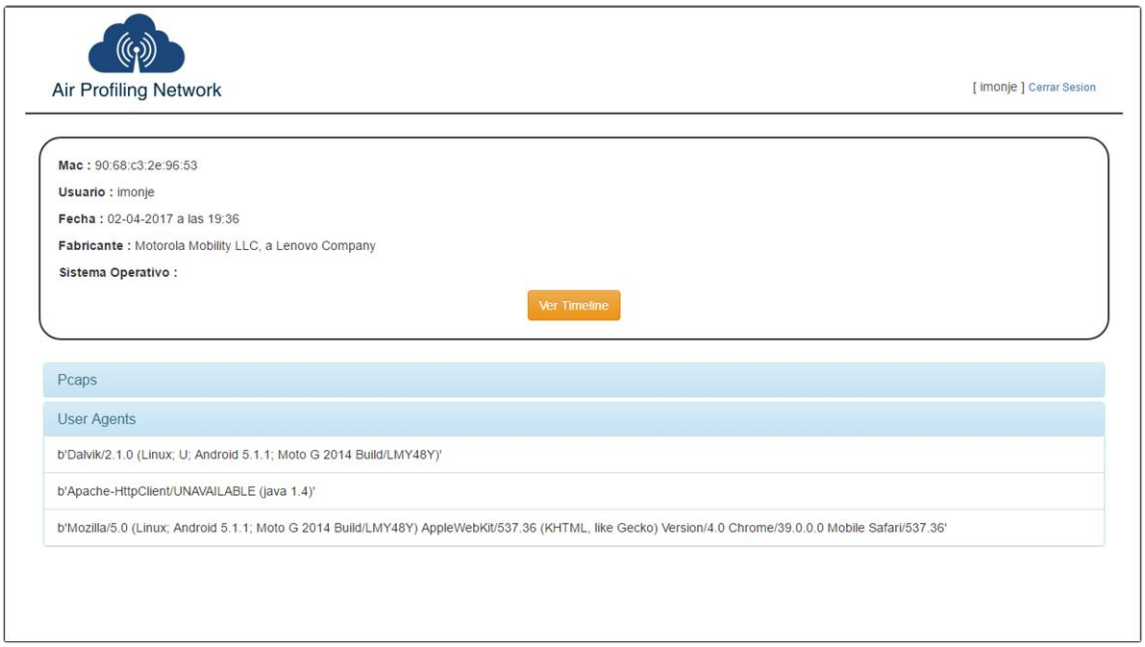

Ilustración 6. Pantalla perfil individual

2.7. Pantalla de la línea de tiempo

En la pantalla de *timeline* (línea de tiempo) se muestran en orden cronológico los paquetes procesados por la aplicación de un perfil en concreto.

A la parte izquierda y saliendo de una bola amarilla se muestran en forma de tarjeta los paquetes que ha mandado el perfil, a la parte derecha y saliendo de una bola azul se muestran en forma de tarjeta los paquetes que ha enviado el perfil.

La información que contienen las tarjetas es IP origen, IP destino, hora y fecha en la que se capturó el paquete, los perfiles origen y destino asociados, la información de reverse DNS y de localización (obtenida a través de servicios externos) y en caso de disponerlo el host en la cabecera HTTP.

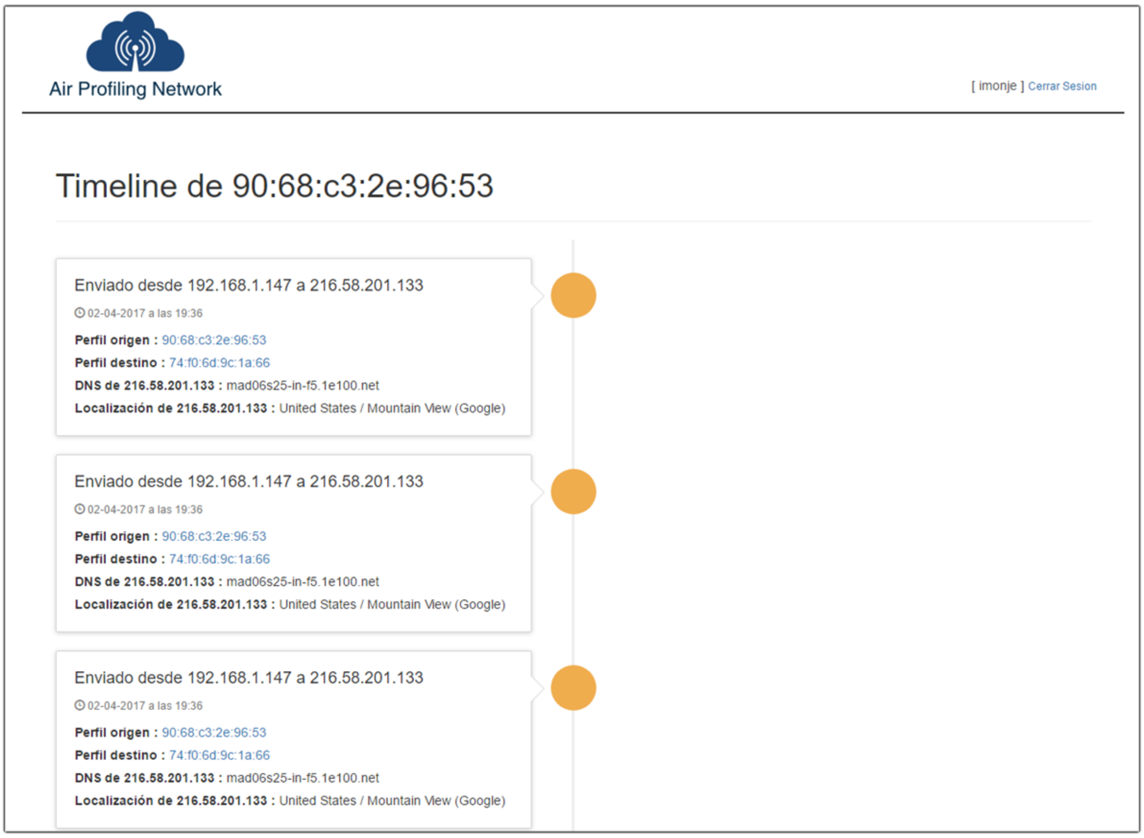

Ilustración 7. Pantalla línea de tiempo (1)

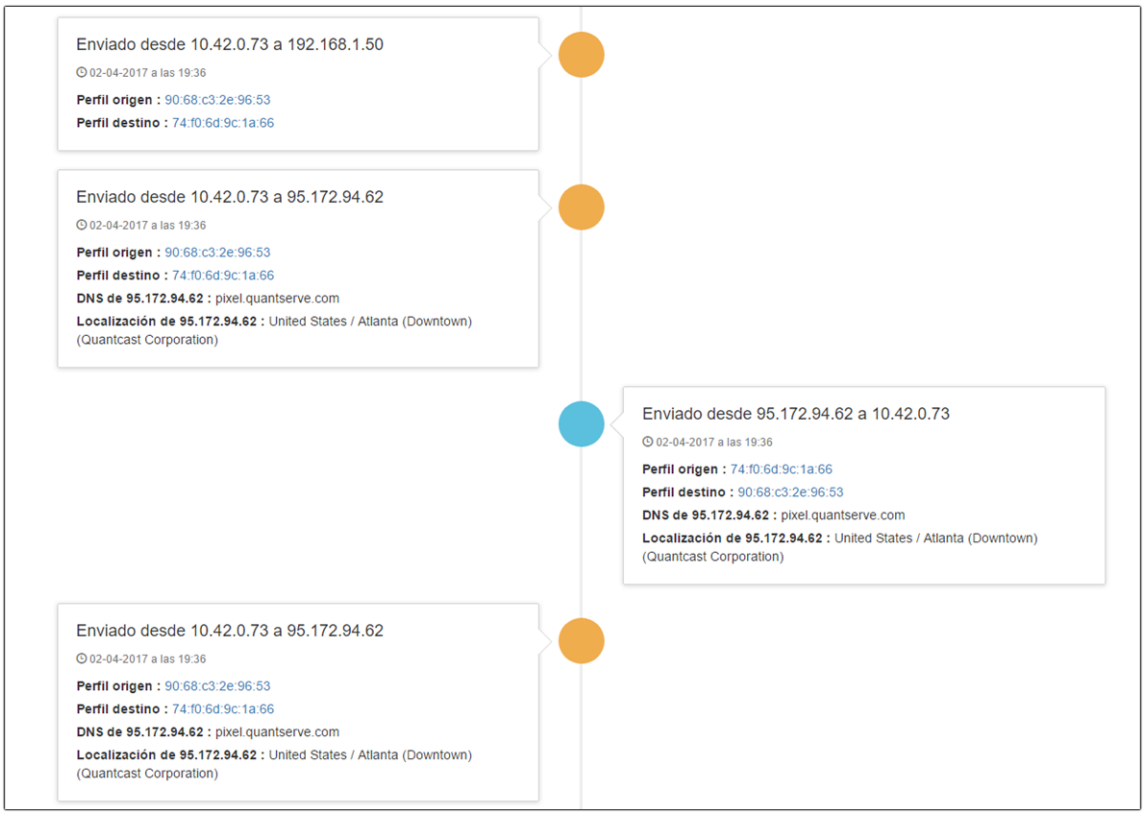

Ilustración 8. Pantalla línea de tiempo (2)

2.8. Pantalla de aplicaciones relacionadas

En esta pantalla se muestran las aplicaciones que pueden estar relacionadas con el link contenido dentro del paquete. Para ello se hace uso de la integración con Tacyt.

Ilustración 9. Pantalla aplicaciones relacionadas

3. Ejemplo práctico

En este punto vamos a analizar un caso práctico del uso de la aplicación:

Entramos a la plataforma de *Air Profiling Network*, subimos el fichero upv.pcapng y esperamos a que se procese.

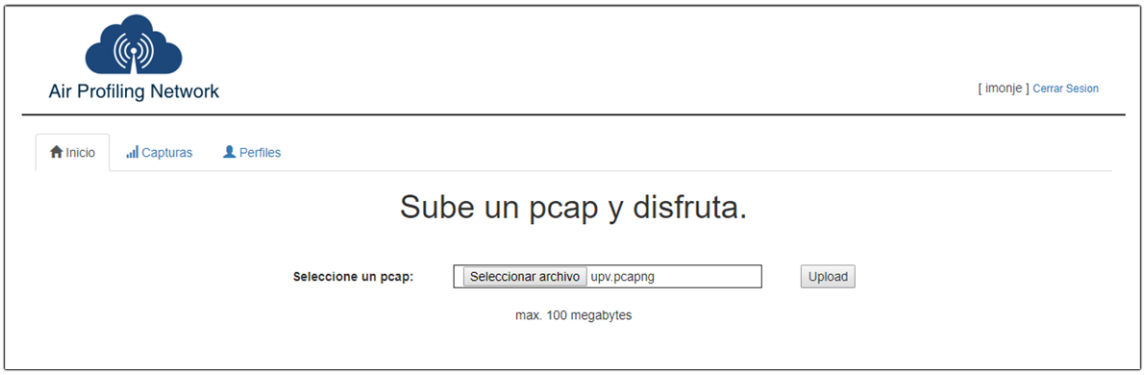

Ilustración 10. Subiendo una captura

Nos dirigimos al listado de capturas de red y buscamos nuestro PCAP nos dirigimos al botón de ver con más detalle.

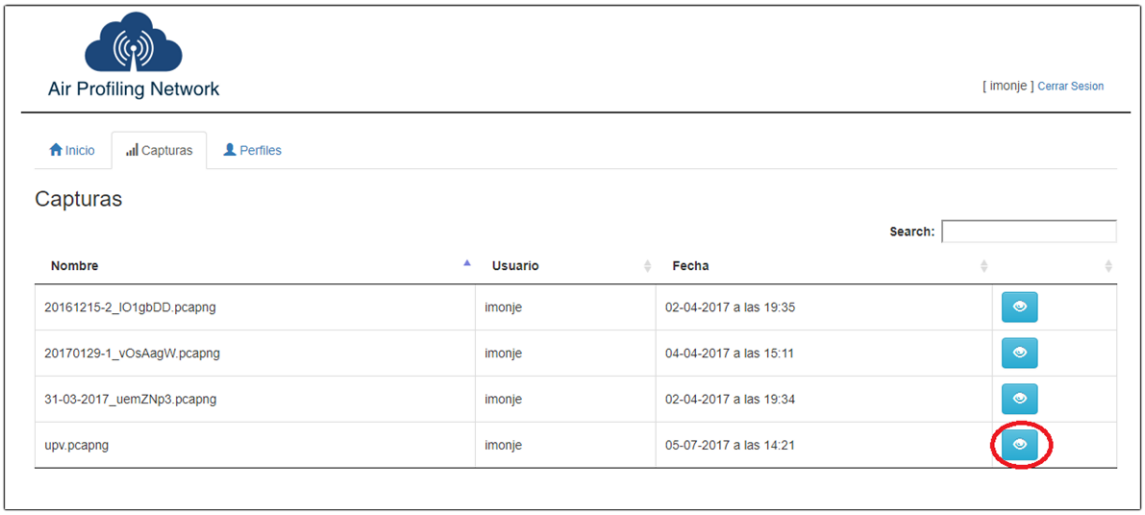

Ilustración 11. Accediendo a captura de red

Podemos observar que solo se ha detectado un perfil de usuario en esta captura identificado con la dirección MAC f0:d7:aa:d3:8d:f0 y nos dirigimos al botón de ver con más detalle.

Ilustración 12. Accediendo a perfil

Dentro de la pantalla de perfil observamos más detalles de este como por ejemplo el fabricante asociado a la dirección MAC en este caso Motorola, también podemos ver los agentes de usuario detectados en ese perfil con los que podemos deducir que este dispositivo dispone de un dispositivo Moto G5 con Android versión 7.0. Nos dirigimos al botón de ver línea de tiempo del perfil.

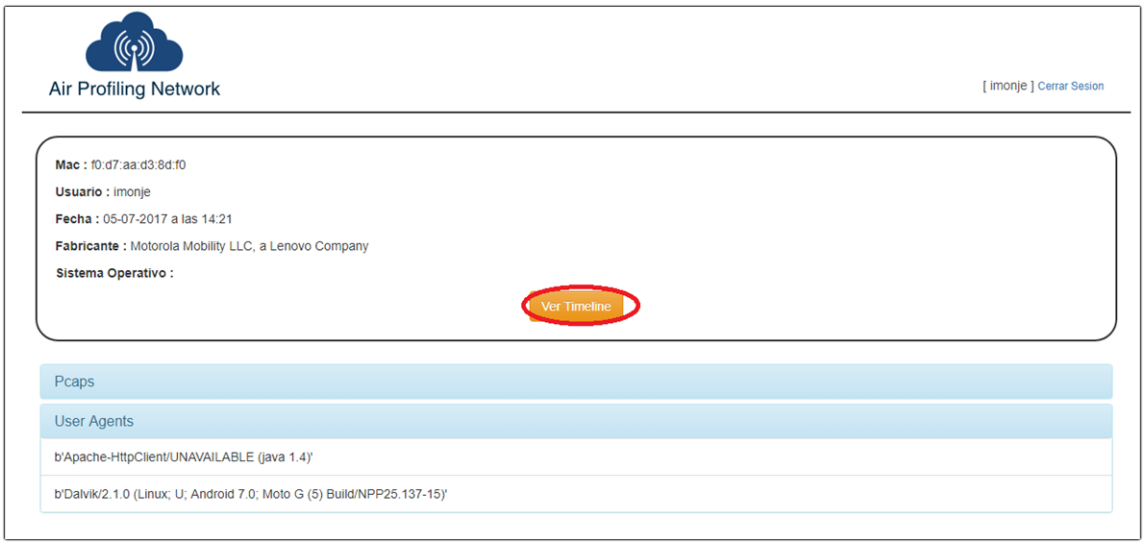

Ilustración 13. Accediendo al timeline de un perfil

En la línea de tiempo del usuario podemos observar numerosos paquetes, nosotros nos vamos a fijar en uno concreto.

Ilustración 14. Vista de un paquete de red

Analizando el paquete podemos ver la fecha y la hora a la que se envió, los perfiles de origen y destino y el host de la cabecera HTTP además se ha podido obtener información DNS de la IP 158.42.4.23 y su geolocalización donde podemos ver claramente que la IP está

relacionada con la universidad politécnica de valencia. Pulsamos el link de host.

Ilustración 15. Aplicaciones relacionadas con www.upv.es

En la pantalla de aplicaciones relacionadas podemos observar que existen tres aplicaciones relacionadas con el host www.upv.es, analizando con más detalle podemos deducir que la aplicación UPV - Politècnica de València esta repetida, esto es debido a que Tacyt obtiene información de distintas tiendas de aplicaciones (Googleplay, mobogenie …) por lo que se acota a solo dos aplicaciones posibles instaladas en el dispositivo del usuario.

4. Manual del programador

En esta sección se van a describir las partes más técnicas del proyecto hablando de qué se ha usado, cómo y con qué fin:

4.1. Tecnologías

Para el desarrollo del presente trabajo se han utilizado las siguientes tecnologías:

- **Python 3.4**

  Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

  Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

  Python. n.d https://es.wikipedia.org/wiki/Python (consultado el 22 junio 2017).

  Python ha sido usado en casi la totalidad del proyecto exceptuando las vistas de la aplicación desarrolladas en HTML, CSS, Javascript.

  La elección de esta tecnología se debió principalmente a dos motivos:

  1. Una de las primeras cosas que se plantearon para poder desarrollar correctamente la funcionalidad más básica del proyecto fue la funcionalidad de lectura de paquetes de red

para ello se buscaron librerías específicas con la elección final de Scapy (escrita en Python).

2. Aunque se encontraron librerías igualmente válidas en otros lenguajes de programación, debido a la poca experiencia en Python del autor en comparación a estas tecnologías y al claro concepto de un trabajo final de grado de aprender lo máximo posible se decidió el uso de Python como núcleo central del proyecto.

- **Django 1.8**

Django es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como Modelo–vista–controlador.

La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio No te repitas. Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos.

Django. n.d https://es.wikipedia.org/wiki/Django_(framework) (consultado el 22 junio 2017).

La elección de Django fue debida a que es el framework web más potente que posee Python y a su amplia documentación.

- **Scapy**

Scapy es una herramienta de manipulación de paquetes para redes informáticas escrita en Python por Philippe Biondi. Puede crear o decodificar paquetes, enviarlos y capturarlos.

Scapy. n.d https://en.wikipedia.org/wiki/Scapy (consultado el 22 junio 2017).

La elección de esta librería, como ya se ha comentado anteriormente, se debió al hecho de que era la que más se adaptaba a las necesidades del proyecto, aunque se tuvo que agregar un módulo extra para extraer más información de la capa HTTP.

- **MySQL**

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

Está desarrollado en su mayor parte en ANSI C y C++. Tradicionalmente se considera uno de los cuatro componentes de la pila de desarrollo LAMP y WAMP.

MySQL. n.d https://es.wikipedia.org/wiki/MySQL (consultado el 22 junio 2017).

El uso de MySQL en el desarrollo del proyecto se debió a que es gratuito y funciona perfectamente con el ORM Django.

- **User_agents**

User_agents es una librería de Python que proporciona una manera fácil de identificar y detectar dispositivos como teléfonos móviles, tabletas y sus capacidades al analizar cadenas de *user agents* (navegador / HTTP).

User_agents. n.d https://github.com/selwin/python-user-agents (consultado el 22 junio 2017).

El uso de esta librería se debió a que proporcionaba los datos de la misma forma en la que se querían guardar. Se contempló la posibilidad de utilizar un servicio externo para poder obtener los datos, pero finalmente se descartó por evitar depender tanto de otras aplicaciones.

- **Bootstrap 3.3.7**

Bootstrap es un conjunto de herramientas de Código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales.

Bootstrap. n.d https://es.wikipedia.org/wiki/Bootstrap_(framework) (consultado el 22 junio 2017).

El uso de este framework evitó complicaciones innecesarias a la hora de diseñar la aplicación. La elección se debió a que es un framework de los más populares y sencillos.

- **Jquery 3.1.1**

jQuery es una biblioteca multiplataforma de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones

propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

JQuery. n.d https://es.wikipedia.org/wiki/JQuery (consultado el 22 junio 2017).

El uso de esta librería al igual que el framework Bootstrap se debió a su sencillez y popularidad hoy en día.

4.2. Integraciones

Para el desarrollo del presente trabajo se han realizado las siguientes integraciones con servicios externos:

- **Fabricante por MAC**

El objetivo de esta integración es conseguir información del dispositivo a través de la MAC de los dispositivos que generan el tráfico que procesamos. Los primeros seis dígitos de una MAC están registrados a nombre de alguna empresa.

Para ello hacemos uso del servicio http://api.macvendors.com/ que posee un API HTTP para obtener fácilmente el fabricante. Cuando obtenemos esta información la guardamos en una tabla de la base de datos para evitar la necesidad de consultar a un servicio externo repetidamente.

- **Geolocalización**

El objetivo de esta integración es conseguir una localización aproximada de las IP que generan el tráfico que capturamos.

Para ello hacemos uso del servicio http://ip-api.com/ que posee un API para obtener información relacionada con la localización en

distintos formatos. Cuando obtenemos esta información la guardamos en una tabla de la base de datos para evitar la necesidad de consultar a un servicio externo repetidamente.

Ilustración 16. Ejemplo respuesta petición geolocalización

- **Reverse DNS**

El objetivo de esta integración es conseguir asociar un host a las IP que generan el tráfico que capturamos.

Para ello inicialmente se iban a utilizar servicios externos, pero al final se optó por usar la librería sockects de Python que permite obtener el host de una IP.

La información obtenida la almacenamos en una tabla de la base de datos para su posterior consulta.

- **Tacyt**

Tacyt es una innovadora herramienta de ciber inteligencia que facilita la investigación en entornos de aplicaciones móviles Android e iOS (apps) mediante su tecnología de big data.

Su utilización es necesaria debido a que las amenazas contra la seguridad relacionadas con el mundo de telefonía móvil crecen continuamente: ataques específicos, adware agresivo, imitaciones de apps que se comportan como legítimas pero que roban información o consumen servicios en segundo plano, etcétera. Estas apps siguen activas y disponibles en los Mercados de aplicaciones el tiempo suficiente como para afectar a miles de usuarios.

Tacyt monitoriza, almacena, analiza, correlaciona y clasifica millones de apps, añadiendo a su base de datos miles de nuevas aplicaciones cada día. Su potente motor permite responder con información en tiempo real sobre las amenazas, los atacantes, los perfiles, detectar nuevas metodologías y herramientas de ataque, protección temprana de la marca, observar el impacto de marketing, etcétera.

El objetivo de Tacyt es permitir la rápida detección, descubrimiento y análisis de estas amenazas para reducir su impacto potencial en las organizaciones.

Tacyt está dirigido a investigadores y analistas de seguridad, a proveedores de servicios gestionados, a empresas que prestan servicios de seguridad y/o ciber-inteligencia, a autoridades educativas locales, etcétera. Los profesionales que lo utilicen realizarán todo el proceso de investigación a través de un completo y potente portal web y una API que permite acceder a la funcionalidad del sistema mediante programación.

El objetivo de esta integración es conseguir enlazar las URL obtenidas en las capturas de red con la base de datos Tacyt y así poder inferir qué aplicaciones puede tener el usuario instaladas en su dispositivo.

5. Mejoras y ampliaciones

Durante el desarrollo del proyecto se ha detectado posibles mejoras a realizar:

- **Mejora de aplicaciones relacionadas.**

  Actualmente la aplicación solo hace una consulta a la base de datos Tacyt y muestra las aplicaciones en las que aparece el link indicado.

  Guardando esta información dentro de la aplicación se podría indicar un porcentaje de probabilidad basándose en el número de links que posee de una aplicación un perfil.

- **Agregar aviso por email al detectar tráfico malicioso.**

  Uno de los posibles usos comentados anteriormente era la detección de tráfico malicioso y aplicaciones malware.

  Tacyt posee la funcionalidad de RSS. Por ello suscribiéndose a los filtros públicos marcados como malware se podría en tiempo real avisar a los usuarios en cuanto Tacyt detectase una aplicación maliciosa nueva.

- **Analizar tráfico distinto a HTTP.**

  Actualmente la aplicación solo procesa los paquetes que contiene cabecera HTTP. Existen otros protocolos de los que se podría obtener información como por ejemplo DNS.

6. Conclusiones

El objetivo principal de este proyecto era obtener la mayor cantidad de información posible a partir de capturas de red y clasificar dicha información mostrándola de una forma amigable.

Debido al uso de las tecnologías y framework usados hemos podido centrarnos en las partes más técnicas del proyecto y obviar ciertos aspectos.

Durante el desarrollo del proyecto se ha podido observar que obtener información privada de una persona es muy simple con las herramientas adecuadas.

En síntesis, los usuarios deberían tener cuidado de a quién facilitan sus datos y tener muy claro que si se conectan a una red de forma gratuita, el que proporciona ese servicio puede obtener información sobre ellos.

7. Bibliografía

- Guía estructuración código Python: http://mundogeek.net/traducciones/python-idiomatico
- Documentación de scapy: http://www.secdev.org/projects/scapy/
- Guía de uso de scapy: https://charlesreid1.com/wiki/Scapy/Pcap_Reader
- Guía segmentación paquete de red con scapy: http://blog.sbarbeau.fr/2011/06/http-support-in-scapy.html
- Python reverse DNS: http://www.sfentona.net/?p=2184

- Uso de JSON en Python: https://stackoverflow.com/questions/12965203/how-to-get-json-from-webpage-into-python-script

- Querys ORM Django: https://docs.djangoproject.com/en/1.10/topics/db/sql/

- Guía de uso de librería user agents: https://github.com/selwin/python-user-agents

- Documentación Tacyt: https://github.com/ElevenPaths/tacyt-sdk-python

8. Anexos

8.1. Código procesamiento PCAP

```
# -*- coding: utf-8 -*-
from django.utils import timezone

from .models import Link
from .models import Location
from .models import Perfil
from updater.models import Pcap

from .integrations import getLocation
from .integrations import getVendor
from .integrations import getUA
from .integrations import getHost

from scapy.all import *
import scapy_http.http as scapyh

import datetime;

#Funcion que procesa los pcaps no procesados
def processPcaps():

    pcaps = Pcap.objects.filter(procesado=False)

    for i in range(len(pcaps)):
        pcap = pcaps[i]
        processPcap(pcap)
        pcap.procesado = True
        pcap.save()
```

```
    return len(pcaps)

#Función que procesa un pcap para optener perfiles y sus links
asociados.
def processPcap(pcap):

    #path = "/home/imonje/air_profiling" + pcap.docfile.url
    path = pcap.docfile.path
    print(path)

    packets = rdpcap(path)

    for i in range(len(packets)):

        p = packets[i]

        'Comprobamos si son MAC validas'
        if isValidMac(p.src) and isValidMac(p.dst):

            'Generamos los links'
            processLink(p, pcap)


#Funcion que obtiene un perfil de la BD o lo crea si no existe
def processPerfil(mac, pcap):

    try:
        perfil = Perfil.objects.get(mac=mac)
    except Perfil.DoesNotExist:
        perfil = None

    if perfil is None:
        perfil = Perfil()
        perfil.mac = mac
        perfil.user = pcap.user
        perfil.pcap = pcap
        perfil.save()
        getVendor(mac)


    return perfil

#Función que procesa un paquete para un link.
def processLink(packet, pcap):


    # si el paquete tiene capa ip lo procesamos
    if IP in packet:

        'Obtenemos los perfiles asociados'
        perfil_src = processPerfil(packet.src, pcap)
        perfil_dst = processPerfil(packet.dst, pcap)

        l =  Link();
        l.pcap = pcap
        l.perfil_src = perfil_src
        l.perfil_dst = perfil_dst
        l.ip_dst = packet[IP].dst
        l.ip_src = packet[IP].src
        l.time = getDate(packet.time)
```

```
        #si es un paquete HTTP
        if scapyh.HTTPRequest in packet:
            l.user_agent =
str(scapyh._get_field_value(packet[scapyh.HTTPRequest], "User-
Agent"))
            l.host =
str(scapyh._get_field_value(packet[scapyh.HTTPRequest], "Host"))
            #Parseamos el user agent
            getUA(l.user_agent)
        l.save()

        #Comprobamos si la ip tiene un objeto location asociado, si
no es asi generamos uno
        location_src =
Location.objects.filter(ip=l.ip_src).filter(pcap=l.pcap).first()
        location_dst =
Location.objects.filter(ip=l.ip_dst).filter(pcap=l.pcap).first()

        if location_src is None:
           getLocation(l.ip_src, l.pcap)
        if location_dst is None:
           getLocation(l.ip_dst, l.pcap)

        getHost(l.ip_src)
        getHost(l.ip_dst)

def isValidMac(mac):
    return mac != "ff:ff:ff:ff:ff:ff" and not mac.startswith("01")

def getFormatDate(time):
    return datetime.datetime.fromtimestamp(time).strftime('%d-%m-%Y
%H:%M:%S')

def getDate(time):
    time = datetime.datetime.fromtimestamp(time)
    time = timezone.make_aware(time,
timezone.get_current_timezone())
    return time
```

8.2. Código integraciones con otros servicios

```
# -*- coding: utf-8 -*-

import requests
import re
import socket

from user_agents import parse

from .models import Location
from .models import Vendor
from .models import UserAgent
from .models import Dns

import tacyt.TacytApp as tacytapp
```

```
def getLocation(ip, pcap):

    if not is_ip_private(ip):
        #Hacemos una petición HTTP
        url = "http://ip-api.com/json/" + ip
        print("Localizando ip " + ip)
        r = requests.get(url)
        aux = r.json()

        if aux.get("status") == "success":
            #Creamos un objeto Location con el resultado
            l = Location()
            l.pcap = pcap
            l.ip = ip
            l.timezone = aux.get("timezone", "")
            l.countryCode = aux.get("countryCode", "")
            l.org = aux.get("org", "")
            l.region = aux.get("region", "")
            l.latitud = aux.get("lat", "")
            l.longitud = aux.get("lon", "")
            l.country = aux.get("country", "")
            l.regionName = aux.get("regionName", "")
            l.isp = aux.get("isp", "")
            l.city = aux.get("city", "")
            l.save()

def getHost(ip):

    dns = Dns.objects.filter(ip=ip);
    if not is_ip_private(ip) and len(dns) < 1:
        try:

            dns = Dns()
            host = socket.gethostbyaddr(ip)
            dns.ip = ip
            print(host)
            dns.host = host[0]
            dns.save()
            print("Obtenido host " + host[0] + " desde la ip : " +ip)
        except socket.error:
            print("Error obteniendo la ip : " + ip)

def getTacytApps(host):
    result = []
    api = tacytapp.TacytApp("PpQbU3AWa773ghLdf2YE",
"9UWa3aqaJKrqTkYqtbyUUmyP8uT39NmUYH4HuQWJ")
    result_search = api.search_apps("links:\"http://" + host + "\"",1
, 15, '' ,True)
    list = result_search.data.get('result').get('applications')

    for i in range(len(list)):
        app = list[i]
        result.append(app)

    return result

def getUA(user_agent):
    u = UserAgent.objects.filter(value = user_agent).first()

    if u is None:
```

```
        s = parse(user_agent)
        agent = str(s).split("/")
        u = UserAgent()
        u.value = user_agent
        u.os = agent[1]
        u.browser = agent[2]
        u.device = agent[0]
        u.save()


def getVendor(mac):
    print("Procesando " + mac)
    vendor = Vendor.objects.filter(mac=mac[0:8]).first()

    if vendor is None:
        print("Obteniendo fabricante para la MAC " + mac)
        url = "http://api.macvendors.com/" + mac
        r = requests.get(url)
        vendor = Vendor()
        vendor.mac=mac[0:8]
        vendor.fabricante=r.text
        vendor.save()
    return vendor

def is_ip_private(ip):

    # https://en.wikipedia.org/wiki/Private_network
    priv_lo = re.compile("^127\.\d{1,3}\.\d{1,3}\.\d{1,3}$")
    priv_24 = re.compile("^10\.\d{1,3}\.\d{1,3}\.\d{1,3}$")
    priv_20 = re.compile("^192\.168\.\d{1,3}.\d{1,3}$")
    priv_16 = re.compile("^172.(1[6-9]|2[0-9]|3[0-1]).[0-9]{1,3}.[0-
9]{1,3}$")

    return priv_lo.match(ip) or priv_24.match(ip) or
priv_20.match(ip) or priv_16.match(ip)



def isValidIP(ip):
    return not ip.startswith("192.168")
```

8.3. Modelos Base de datos

```
from __future__ import unicode_literals
from django.utils import timezone

from django.db import models



class Link(models.Model):
    perfil_src = models.ForeignKey('core.Perfil',
related_name='perfil_src')
    ip_src = models.CharField(max_length=50)
    perfil_dst = models.ForeignKey('core.Perfil',
related_name='perfil_dst')
    ip_dst = models.CharField(max_length=50)
```

```
    host = models.CharField(max_length=50)
    user_agent = models.CharField(max_length=200)
    pcap = models.ForeignKey('updater.Pcap')
    time = models.DateTimeField()

class Location (models.Model):
    ip = models.CharField(max_length=50)
    latitud = models.CharField(max_length=50)
    longitud = models.CharField(max_length=50)
    pcap = models.ForeignKey('updater.Pcap')
    timezone = models.CharField(max_length=150)
    countryCode = models.CharField(max_length=50)
    org = models.CharField(max_length=250)
    region = models.CharField(max_length=50)
    country = models.CharField(max_length=250)
    regionName = models.CharField(max_length=250)
    isp = models.CharField(max_length=250)
    city = models.CharField(max_length=150)

class Perfil(models.Model):
    mac = models.CharField(max_length=50, primary_key=True)
    name = models.CharField(max_length=200)
    os = models.CharField(max_length=200)
    telefono = models.CharField(max_length=200)
    dispositivo = models.CharField(max_length=200)
    user = models.ForeignKey('auth.User')
    pcap = models.ForeignKey('updater.Pcap')
    created_date = models.DateTimeField(default=timezone.now)

class Vendor(models.Model):
    mac = models.CharField(max_length=10, primary_key=True)
    fabricante = models.CharField(max_length=200)

class Dns(models.Model):
    ip = models.CharField(max_length=50, primary_key=True)
    host = models.CharField(max_length=200)

class UserAgent(models.Model):
    value = models.CharField(max_length=200, primary_key=True)
    os = models.CharField(max_length=200)
    browser = models.CharField(max_length=200)
    device = models.CharField(max_length=200)
```

8.4. Glosario

- **Framework**: O entorno de trabajo, se trata de una infraestructura, en términos generales, conceptual y tecnológica de soporte definido que sirven de base para la organización y desarrollo de software, resultando como

herramienta que ayuda a desarrollar y unir diferentes componentes en un proyecto.

- **MAC:** En las redes de computadoras, la dirección MAC (siglas en inglés de Media Access Control) es un identificador de 48 bits que corresponde de forma única a una tarjeta o dispositivo de red.

- **DNS:** El sistema de nombres de dominio (DNS, por sus siglas en inglés, Domain Name System) es un sistema de nomenclatura jerárquico descentralizado para dispositivos conectados a redes IP como Internet o una red privada.

- **Sockect:** Socket designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.

  Malware: Es un tipo de software que tiene como objetivo infiltrarse o dañar una computadora o sistema de información sin el consentimiento de su propietario.

- **Adware:** Un programa de clase adware es cualquier programa que automáticamente muestra u ofrece publicidad, ya sea incrustada en una página web mediante gráficos, carteles, ventanas flotantes, o durante la instalación de algún programa al usuario, con el fin de generar lucro a sus autores.

8.5. Definición de abreviaturas

- **HTML**: *HyperText Markup Language*, lenguaje de marcas de hipertexto que se utiliza estructura base en el diseño de páginas web.
- **CSS**: *Cascading Style Sheet*, hojas de estilo en cascada, su uso es dar una apariencia agradable a la página web.

- **URL**: *Uniform Resource Locator*, es un identificador de recursos uniforme, cuyos recursos pueden cambiar, es decir, que dicha dirección apunte a recursos variables en el tiempo.
- **ORM**: *Object-Relational mapping,* es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.
- **PCAP**: El pcap es una interfaz de una aplicación de programación para captura de paquetes.

8.6. Enlaces de interés

- Web donde está alojada el Proyecto: http://imonje.pythonanywhere.com
- Repositorio de código: https://github.com/salime45/AirProfilingWeb
- Librería MindMap: http://www.jqueryscript.net/chart-graph/Simple-jQuery-Mind-Map-Diagram-Plugin-mindmap.html
- Librería Timeline: https://bootsnipp.com/snippets/featured/timeline-responsive
- Página de Tacyt: https://tacyt.elevenpaths.com/login