Universitat Oberta de Catalunya
uoc.edu

UNIVERSITAT OBERTA DE CATALUNYA
UNIVERSITAT AUTÓNOMA DE BARCELONA
UNIVERSITAT ROVIRA I VIRGILI

MASTER UNIVERSITARIO EN SEGURIDAD DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y DE LAS COMUNICACIONES

TRABAJO FIN DE MÁSTER

SEGURIDAD EN LOS ECOSISTEMAS IOT

Tutor Consultor
Carlos Hernández Gañán

Profesora de la Asignatura
Helena Rifà Pous

Alumno
Pablo A. Miñano Carmona

Fecha Entrega
Junio 2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Seguridad en los Sistemas IoT*
Nombre del autor:	*Pablo Antonio Miñano Carmona*
Nombre del consultor/a:	*Carlos Hernández Gañán*
Nombre del PRA:	*Helena Rifà Pous*
Fecha de entrega:	*06/2019*
Titulación:	*Máster en Seguridad de las TICs*
Área del Trabajo Final:	*Seguridad en la Internet de las Cosas*
Idioma del trabajo:	*Castellano*
Palabras clave:	*IoT, Seguridad, Pentesting*

Resumen del Trabajo:

En el siguiente Trabajo de fin de Máster (TFM), compuesto de dos partes bien diferenciadas, teórica y práctica, se aborda la salud de la seguridad en la Internet de las Cosas, conocido como IoT.

En la parte teórica se describirá la situación actual de la seguridad de la IoT, se presenta un modelo general del ecosistema que lo conforma, se tratan las vulnerabilidades y amenazas a las que se enfrenta y ahondaremos en los buscadores orientados en este mundillo, destacando a Shodan como referencia.

En la parte práctica, se ha desarrollado un código en Python para emular las opciones de búsqueda básicas del motor Shodan, así como una aplicación paralela para mostrar los hallazgos encontrados en formato web. Esta búsqueda se ha limitado geográficamente a direcciones IP localizadas en España.

Durante unos días, este *script* se ha puesto a prueba, encontrando resultados sorprendentes como cámaras web, FTPs, servidores Apache con versiones obsoletas e incluso la petición de credenciales para acceder a la configuración de un *router*. Es un fiel reflejo de la poca importancia que se le da a la seguridad informática, idea que lamentablemente es extrapolable a entornos públicos, privados y profesionales.

Por último, transmitir el deseo del autor de este trabajo, de que su lectura sea disfrutada tanto como él lo hizo en el desarrollo del mismo.

Abstract:

In the following Master's thesis, composed of two well-differentiated parts, theoretical and practical, the health of security on the Internet of Things, known as the IoT, is addressed.

In the theoretical part, the current security situation of the IoT will be described, a general model of the ecosystem is presented, the vulnerabilities and threats are dealt with, and we will delve into the search engines oriented in this world, highlighting Shodan as a reference.

In the practical part, a Python's code has been developed, to emulate the basic search options of the Shodan's engine, as well as a parallel web application in order to present the results. This search has been geographically limited to IP addresses located in Spain.

For a few days, this script has been put to the test, finding surprising results such as webcams, FTPs, Apache servers with obsolete versions and even the request for credentials to access the configuration of a router. It is a faithful reflection of the little importance given to computer security, an idea that unfortunately can be extrapolated to public, private and professional environments.

Finally, I hope you will enjoy reading this project as much as I did while developing it.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Actualmente, se está viviendo un momento muy relevante en lo que se ha denominado la Internet de las Cosas, también conocido como IoT (acrónimo de los términos *Internet of Things*). Los dispositivos con acceso a Internet para ampliar sus funcionalidades se contaban por 21.000 millones en 2018, con una previsión de 50.000 millones para el año 2023, según la compañía Juniper Research [1].

En aras de ganar cuota de mercado sacando productos antes que la competencia y que puedan ser manipulados y configurados por usuarios con conocimientos limitados (e incluso nulos), se está sacrificando un aspecto tan relevante como es el de la seguridad. Contraseñas inexistentes o débiles, que en determinados casos no pueden ser cambiadas, protocolos de seguridad no implementados y un sinfín de despropósitos en lo que a la seguridad se refiere, hacen que estos dispositivos contengan vulnerabilidades.

Estas, en ocasiones, se traducen en ataques que van desde violaciones de privacidad, como accesos a cámaras webs, hasta situaciones tan graves como posibles ataques de *Botnets*, produciendo denegaciones de servicios de ámbito mundial. Tenemos el ejemplo real de lo sucedido el 21 de Octubre de 2016 [2], afectando a servicios en línea de tanto renombre como Twitter, Netflix o Amazon, entre muchos otros.

Actualmente, existen ciertos sectores que muestra la creciente preocupación por esta tesitura. No en vano, han aparecido motores de búsquedas, centrados en los dispositivos conectados a Internet, como es el caso de Shodan. Mediante este servicio, se pueden descubrir toda clase de máquinas que tengan su propia IP pública, tales como cámaras, semáforos o estaciones de servicios, siempre que estén conectadas a Internet. En la mayoría de los casos, la seguridad es casi nula.

Esta adquisición de información se realiza a través de la captura de metadatos principalmente, por lo que las técnicas de recopilación de información se pueden encasillar desde un punto de vista de recopilación mediante fuentes OSINT (*Open Source Intelligence*), con toques de *footprinting* y *fingerprinting.* Aunque ciertos autores consideran que la actividad de estos motores de búsqueda está al límite de la legalidad, lo cierto es que, tanto en el método de adquisición, como la información revelada, no se produce ningún tipo de transgresión. Es una información que se encuentra disponible y que los responsables de los dispositivos no han tenido en cuenta, pudiendo camuflarla, modificarla o simplemente ocultarla.

Mediante este trabajo, se pretende abarcar las vulnerabilidades de los dispositivos IoT desde el punto de vista de la seguridad, enfocado al escaneo de los que estén conectados a Internet, con objeto de que puedan ser conocidas, para que posteriormente se tomen las medidas oportunas, si así fuese necesario.

Como resultado final, se desarrollará un código en Python que realizará escaneos a direcciones IP localizadas en España. De estas direcciones se obtendrá la información disponible, mediante conexiones *sockets* y lectura de banners y en los casos posibles, se realizará una captura de la página web que corresponda con dicha dirección.

Por último, se realizará una sencilla aplicación para mostrar los resultados de los hallazgos encontrados mediante formato web.

1.2 Objetivos del Trabajo

Mediante el siguiente trabajo se pretende abarcar los siguientes objetivos:

Descripción de la situación actual de la seguridad en la IoT.
Definición del ciclo de vida del ecosistema IoT.
Reconocimiento de las posibles amenazas.
Elaboración de código para escaneo de dispositivos IoT.
Programación de subtareas para la geolocalización.
Definición de un entorno amigable para la visualización de resultados.

Este trabajo se compone de dos partes fundamentalmente. Una primera de ámbito teórica, en la que se van a exponer los conceptos sobre los que se sustentan la IoT en el campo de la seguridad y una segunda, más práctica, en la que se elaborarán códigos para la realización de escaneos en un rango de direcciones IP.

Con este código, se automatizará la búsqueda de vulnerabilidades de los sistemas IoT, a través de las direcciones públicas IPv4, en el ámbito geográfico de España. Se elaborará una recolección de datos que serán almacenados para su posterior tratamiento, en el que se visualizarán los resultados obtenidos.

1.3 Enfoque y método seguido

La realización de este proyecto se implementará mediante código Python, haciendo uso de diversas librerías de apoyo *opensource.* También se usará algún tipo de bases de datos para tratar la información obtenida y una visualización de los resultados mediante Flask.

Inspirado en el funcionamiento de Shodan, se realizará *banner grabbing*, para la obtención de la información de los dispositivos escaneados, así como *sockets* para la obtención de datos que se consideren relevantes.

Se determinará cuáles serán los datos a almacenar así como el sistema gestor para tratarlos, que dependerá principalmente del tamaño de los datos acumulados, para su eficiente manipulación. También se elaborará un proceso de geolocalización para que los escaneos estén restringidos al ámbito geográfico que se ha determinado.

1.4 Planificación del Trabajo

En el diagrama siguiente que se expone tenemos una aproximación de la planificación temporal de las tareas y subtareas que se irán realizando a lo largo de este trabajo. Conforme se avance las tareas, se irán optimizando los tiempos para aquellas que se puedan realizar en plazos más reducidos.

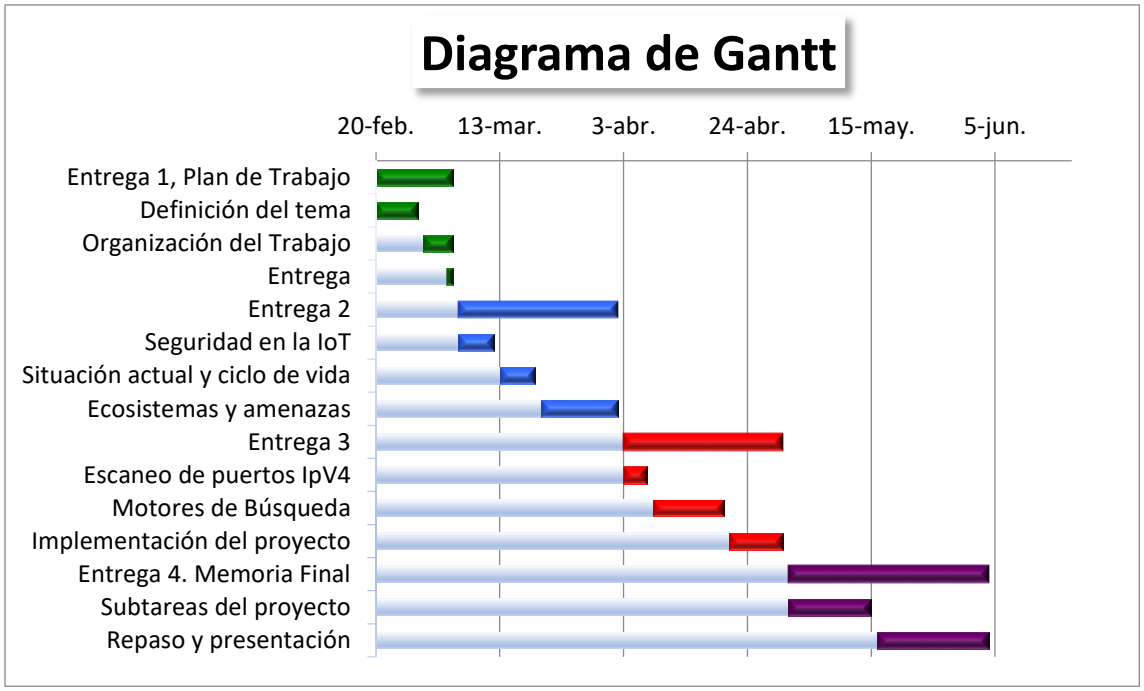

Figura 1 Diagrama de Gantt

1.5 Breve sumario de productos obtenidos

Como producto final obtendremos, en la parte teórica, un estudio completo de la situación de la seguridad de la IoT actualmente, las definiciones de ecosistemas y enumeraciones de posibles vulnerabilidades que afectan a las partes implicadas en la IoT, así como una aproximación a los buscadores

especializados en IoT. En el apartado práctico, tendremos una base de datos con los datos relevantes de los dispositivos encontrados, un programa funcional de escaneo para la realización de la búsqueda de vulnerabilidades, así como la interfaz para la reproducción visual de los datos.

1.6 Breve descripción de los otros capítulos de la memoria

En el capítulo 2 se comenzará por una exposición de los conceptos básicos de la IoT y la seguridad, haciendo referencia al estado actual en el que este campo se encuentra.

Seguidamente, en el capítulo 3 se abordarán las definiciones de los elementos comunes que componen el ecosistema en la IoT, detallando el ciclo de vida en la Internet de las Cosas. A continuación se ahondará en las posibles amenazas que podemos encontrar en los distintos elementos que componen los sistemas IoT.

El capítulo 4, se centra en detallar las funciones de los motores de búsqueda de la IoT, usando como ejemplo principal el de Shodan y se realizará la implementación de nuestro proyecto, con todas las tareas y subtareas que ello implica para que sea funcional. Se expondrán las partes que componen el proyecto, esto es, base de datos, geolocalización, *banner grabbing, sockets,* etc.

Por último, en el capítulo 5 se abordarán las conclusiones que se derivan de la realización de este proyecto.

2. Seguridad en la IoT

2.1 Descripción IoT y la seguridad en la misma.

Se atribuye a Kevin Ashton, en el año 2009, el primer uso de los términos “la internet de las cosas”, o “*The Internet of Things*”. Con este concepto, que se suele usar con su acrónimo IoT, se hace referencia a la conexión de todo tipo de dispositivos a Internet, para el intercambio de información entre estos dispositivos, pudiendo incluso, en determinados casos, actuar de modo automático ante la detección de ciertos eventos.

Como se puede imaginar, la aplicabilidad en este campo está limitada exclusivamente a la imaginación. Podemos encontrar ejemplos de dispositivos en domótica, en las cosas cotidianas, en las *Smart Cities*, en la salud, en la Industria 4.0, agricultura, ganadería, coches, y un sinfín de aplicaciones que van desde un espejo inteligente, neveras que advierten de la escasez de algún producto, *wearables* para mascotas que miden la calidad del sueño hasta sensores para vigilar el ganado. [3]

Si intentamos obtener una definición más formal del concepto IoT podemos encontrar que, según la RAE, en su diccionario del español jurídico, define la Internet de las Cosas como la interconexión digital de personas, animales y cosas (electrodomésticos, coches, etc.) con Internet. [4]

Una definición más interesante y completa es la que nos proporciona el RFID group [5] (*Radio Frequency IDentification*), catalogándola como una red mundial de objetos interconectados entre sí, basado exclusivamente en estándares de protocolos de comunicación”.

En la mayoría de los casos, estamos hablando de dispositivos que están conectados 24 horas al día. Es entonces cuando nos planteamos la pregunta

origen que le da sentido a este TFM. Y, ¿qué tal la seguridad de estos dispositivos? Respuesta corta: Mal.

La feroz competencia entre las empresas hace que el lanzamiento de los productos se realicen prematuramente, cometiendo errores de implementación de seguridad, en los casos en los que se implementa. El facilitar la configuración a los usuarios menos avanzados, tampoco ayuda demasiado en este campo, haciendo que en ocasiones no exista seguridad alguna. Como se verá en el apartado de posibles amenazas, apoyándose en estas debilidades, se producen ataques de denegación de servicio basándose en dispositivos IoT.

Contraseñas débiles por defecto, comunicaciones sin cifrar, *firmwares* no actualizados, interfaces de usuario web susceptibles de ser atacadas mediante SQL *Injection* o *Cross-site Scripting* (XSS), son algunas de las amenazas y vulnerabilidades reales con las que nos debemos enfrentar ante una gran mayoría de dispositivos IoT.

La empresa de seguridad Karpesky Lab, ya en 2015 comprobó que la seguridad de la mitad de los dispositivos analizados no eran seguros. A principios del 2018, se publicó un estudio parecido en el que constatan exactamente [6] los mismos resultados. Lo que más llama la atención, es que una de las vulnerabilidades más comunes encontradas ha sido una contraseña muy débil por defecto y la imposibilidad de cambiarla.

A esto debemos sumar la creencia de que el añadir la funcionalidad de conectividad a un dispositivo le da un cierto valor o prestigio. Es así que dispositivos con funcionalidades absurdas, sumado a seguridad mal implementada y en algunos casos, inexistente, aparece el concepto *de Internet of Shit.* Existe un twitter muy popular, https://twitter.com/internetofshit, donde se analizan los dispositivos IoT, poniendo en tela de juicio, sus calidades, funcionalidades y apartados de seguridad. El autor también está preparando una página, donde se indexarán los resultados obtenidos, en https://internetofshit.net.

2.2 Situación actual de la IoT

Actualmente, el crecimiento de los dispositivos conectados a internet es de orden exponencial. BI Intelligence, realizó un estudio en el que se estima que para el año 2020, habrá 34.000 millones de dispositivos conectados a Internet [7], lo que equivale a 4 dispositivos por cada habitante del planeta.

Si presuponemos que sólo un pequeño porcentaje tuviesen problemas de seguridad, hablaríamos de miles y miles de dispositivos. Pero, como ya hemos podido comprobar, el porcentaje no es para nada pequeño. Entonces, ¿sería una exageración hablar de una posibilidad real de colapso en Internet? Veamos los casos más recientes y llamativos de ataques a través de la IoT.

Ya en el año 2010, VirusBlokAda descubrió un gusano llamado Stuxnet. Su objetivo era el de espiar y reprogramar los sistemas industriales SCADA de control y monitorización de procesos de Siemens, WinCC/PCS 7. Todo apuntó a que las centrales nucleares de Irán eran el objetivo principal de este ataque. Karpesky declaró que se había creado un prototipo operativo y temible de un arma cibernética que llevará a la creación de una nueva carrera de armamentos en el mundo [8].

Uno de los ataques más sonados mediante IoT fue el de la *Botnet* Mirai. Se puede definir una *Botnet* [9] como un grupo de dispositivos que han sido infectados por un malware y `pueden ser controlados de forma remota. A los dispositivos que conforman esta red, se les llama *Bot*, o *zombie*, por el estado latente en el que se quedan esperando a ser controlados.

Mediante una *Botnet* se pueden llegar a realizar diferentes actuaciones, con fines maliciosos, como es el caso de la minería de *bitcoins*, ahorrando el coste energético tan elevado que se necesita, el envío de correos basura y por supuesto los archiconocidos ataques de denegación de servicios distribuidos, o DDOS, en los que un enorme número de dispositivos atacan a la vez a un determinado servicio, realizando peticiones constantes, hasta que este no sólo

no puede atender todas las peticiones sino que además suele provocarse la caída del sistema.

Y este es el tipo de ataque en el que la *Botnet* Mirai está especializada. Infecta dispositivos como cámaras IP, impresoras o *routers*. Fue el 21 de Octubre del 2016[10], cuando millones de dispositivos infectados y pertenecientes a esta botnet, atacaron al proveedor de servicios DNS Dyn, afectando a empresas de tanto renombre como la BBC, Amazon, The New York Times, Spotify, Visa, Twitter, Xbox Live, Paypal, entre otros.

Este ataque simultaneo con siguió la increíble cifra de 1,2 Terabits [11] de datos por segundos contra los servidores de Dyn. El colectivo de hackers New World Hackers, se atribuyeron la responsabilidad, El código se hizo público a poco tiempo del ataque, y es relativamente fácil de encontrar para ser descargado, como por ejemplo en la dirección: https://github.com/jgamblin/Mirai-Source-Code/blob/6a5941be681b839eeff8ece1de8b245bcd5ffb02/mirai/bot/scanner.c

Desde que el código se hiciese público, han surgido al menos 7 variantes conocidas de Mirai [12], aunque las diferencias principales están en la lista de credenciales, ya que Mirai utilizaba una reducida lista de 72 combinaciones de usuario y contraseñas, puertos nuevos así como otras arquitecturas tales como ARC y RCE.

Y es que sólo en el año 2018, los ataque contra dispositivos IoT alcanzaron más de 120.000 alteraciones de *malwares* [13], el triple que en todo 2017. La fuerza bruta sigue siendo con diferencia el método más empleado, con un 93% de los ataques detectados.

Uno de los motivos principales por el que la geolocalización de este proyecto se haya limitado a España, es que fue en esa primera mitad del 2018 cuando se produjo que el 80% de los ataques a nivel mundial estaban dirigidos a este país, según un informe de F5 Labs [14], muy por delante de otros países que tradicionalmente estaban siendo más atacados, como eran los casos de Rusia y

Estados Unidos. Brasil y China con un 18 y un 15% respectivamente, conforman el top de países cuyo origen son estos ataques.

Los dispositivos IoT más afectados siguen siendo cámaras IP, DVRs, cámaras de seguridad *routers* en pymes. El método más común empleado para la detección de estos dispositivos se realiza mediante rastreos en Internet para detectar servicios abiertos que permitan la administración remota de los mismos. Los protocolos más comunes detectados son el Telnet y el SSH, además de HNAP, UPnP y SOAP.

Además de las debilidades implementadas en los dispositivos IoT, los servidores y bases de datos a los que se conectan estos dispositivos, es decir, las infraestructuras sobre las que se apoyan también adolecen de la seguridad deseable.

Pese a que claramente, la seguridad en la IoT sigue sin ser una preferencia, el gasto mundial en seguridad de la IoT ha aumentado crecientemente, lo que indica que los fabricantes están tomando en consideración, no sólo las posibles consecuencias de tan nefasta política de implementación de seguridad, sino que también les puede afectar a su propia reputación. De hecho, el gasto según Gartner[15] del 2018 estaría en 1500 millones de dólares, con lo que se produce un incremento de un 28% con respecto al gasto de 1200 millones de dólares del 2017.

Y Mirai es sólo una de las *Botnets* existentes. Sólo en la primera mitad del 2018, se detectaron 13 nuevas *Botnets* del IoT. Con nombres como VPN Filter, Wicked, Omni, SORA, Pure Masuta y OMG, estas *Botnets* recopilan credenciales, monitorean el tráfico, producen ataques DDOS e incluso se usan para crypto-jacking, es decir, usar el dispositivo para minar o generar criptomonedas, sin el permiso del usuario al que pertenece el dispositivo. Es evidente que, una vez llegados a este punto, está más que justificado el interés que tiene la seguridad en los dispositivos IoT.

3. Ecosistemas en la IoT.

3.1 Ecosistemas IoT

La Internet de las Cosas usa la conectividad de Internet con el fin de recopilar los datos, centralizarlos, con el objetivo de que estos sean analizados y/o presentados al usuario final.

Para M. Tim Jones [16], desarrollador de IBM, el ecosistema IoT se compone de tres niveles principalmente:

- El Centro de Datos
- Las Puertas de Enlace
- Los Dispositivos Finales

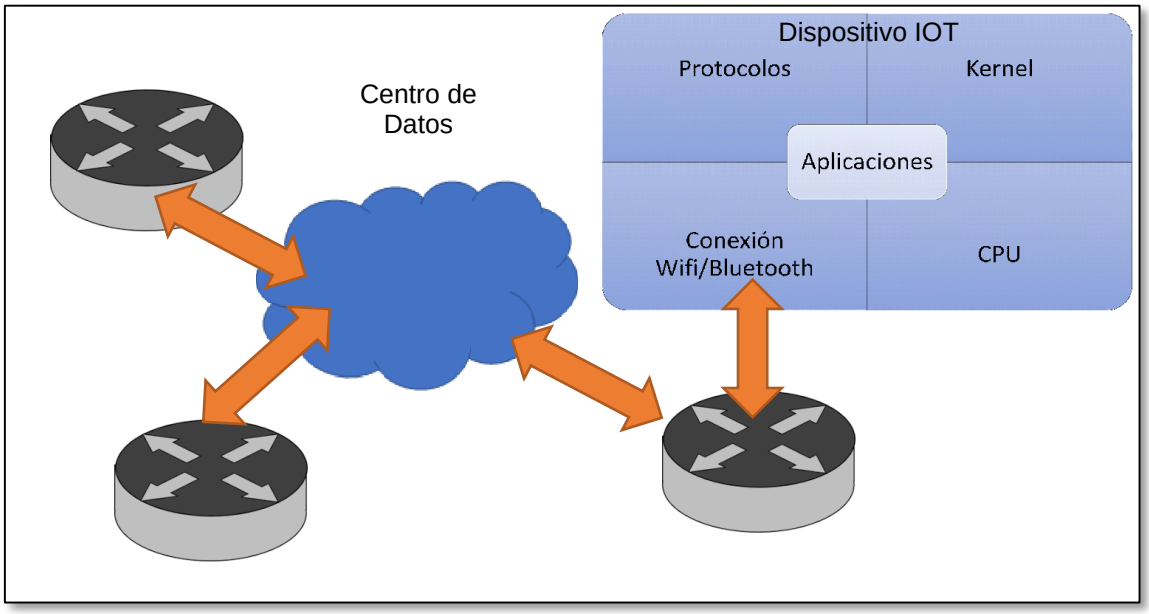

Figura 2 Ecosistemas IoT

El Centro de Datos:

El centro de datos se dispone en el nivel superior. Como se puede observar, representa a una nube pública, aunque podría ser privada (podría ser una híbrida que se constituye con las dos). El centro de datos puede ser usado para:

- Ser el origen del control de los dispositivos IoT
- También puede constituir el punto final al que irán los datos que envían los dispositivos IoT.

Las Puertas de Enlace:

Las puertas de enlace serán los puntos de acceso que hacen posible las comunicaciones de los dispositivos IoT y el centro de datos. También son conocidos como dispositivos de la red *Edge*. Estas conexiones suelen utilizar la tecnología WiFi o Bluetooth para el intercambio de los datos con el centro de datos y los dispositivos.

Dispositivos IoT:

Como se ha comentado, encontramos cámaras de vigilancia inalámbricas, sensores, estaciones meteorológicas, bombillas, ordenadores de viaje, dispositivos portables y muchos más que se encuentran y encontrarán en este nuevo universo del IoT. Cuanto más sencillos son los dispositivos, más dependen de las puertas de enlace para poder ejecutar funciones como las de seguridad o la de compresión de datos.

Centrándonos en la función de autenticación, observamos que tiene una gran importancia en el ecosistema IoT. La autenticación tradicional cuando se inicia sesión en un sistema, consiste en introducir una combinación usuario/clave que deben ser introducidos correctamente para poder autenticarse con éxito. Hablamos de datos estáticos. Esto no está recomendado en un sistema de producción y es lo que predomina en estos sistemas, como es el caso de las webcams. Tanto por ser contraseñas débiles, como porque por lo general se dejan las contraseñas predeterminadas son una fuente de vulnerabilidad importante.

Además, una vez que dos entidades se han autenticado entre ellas correctamente, al intercambiar la información, se suele hacer en texto sin ningún tipo de cifrado a través de la red. Otra importante vulnerabilidad, ya que es muy posible que esa información pueda ser capturada por algún tipo de *sniffer*, lo que podría causar que los datos de los dispositivos quedasen comprometidos.

Como se puede concluir, la seguridad de los datos debe existir no solo en los dispositivos cuando están almacenados, sino también cuando se producen las comunicaciones entre ellos. Al transformar los datos, suele ser necesario alguna clave para poder descifrarlos. Como sucede en la seguridad de autenticación vista anteriormente, la clave que se usará para cifrar los datos es otra forma de secreto compartido, con las mismas carencias de seguridad que el anterior, es decir, almacenando estas contraseñas en los dispositivos que se encargan del cifrado.

Será mediante infraestructuras de claves públicas o PKI, con las que podemos solucionar estos problemas de seguridad. Así, mediante la PKI se realizará la criptografía asimétrica o de claves públicas. Con un par de claves, una de ellas pública, los datos pueden ser cifrados. La clave pública es conocida por cualquier dispositivo y como se observa en el siguiente ejemplo, cifra los datos para la puerta de enlace. La clave privada sí es secreta, es decir, sólo la debe conocer el receptor y será usada para que éste descifre los datos.

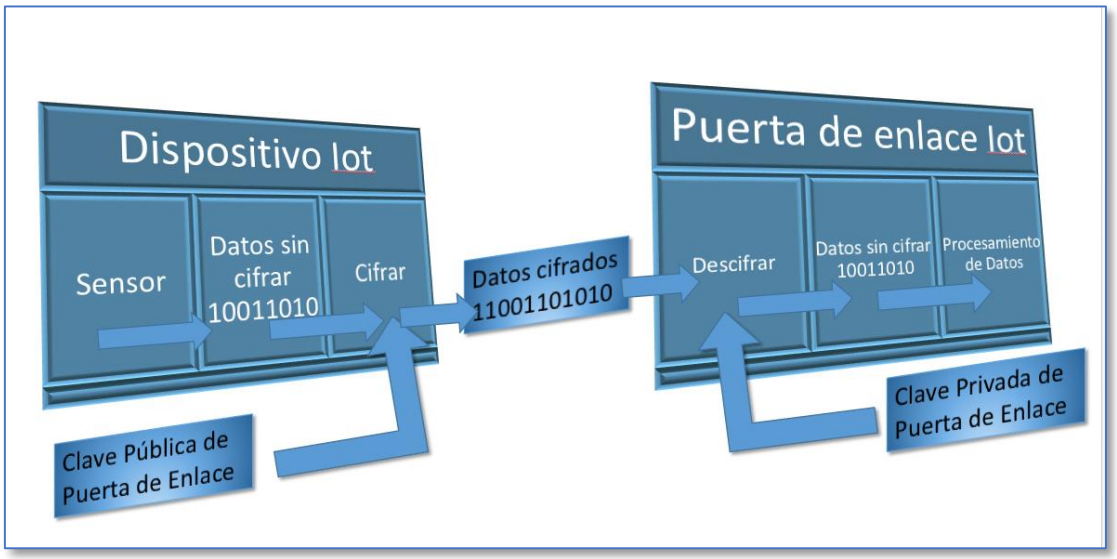

Figura 3 Encriptación de datos

Si los datos están cifrados, aunque fuesen capturados no se podría usar si no se dispone de la clave privada para descifrarlos. Imaginemos el típico ataque *Man in The Middle*.

Si pensamos en la cantidad de dispositivos IoT que se deben gestionar, imaginemos un sistema que tenga millones de ellos, es evidente que no resulta una gestión sencilla y además podría generar problemas de seguridad. Es necesario que la creación y revocación de claves, esté automatizada. Para esto tendremos la PKI.

En términos generales, las PKI, mediante una autoridad certificadora, unen las claves públicas a los dispositivos. Con esto se consigue que un dispositivo IoT pueda cifrar los datos para que estos estén protegidos mientras son enviados por Internet. Además, mediante este proceso de unión, el dispositivo se puede autenticar en la puerta de enlace de la IoT, gracias a la clave pública de la autoridad certificadora. Gracias al uso de PKI, se puede incrementar la seguridad de los sistemas IoT, haciendo que la gestión de la seguridad sea más liviana.

Sin embargo, no olvidemos que la criptografía es, en términos de complejidad computacional, bastante costosa. Es por esto que empiezan a surgir estándares criptográficos [17] más ligeros, que ayudan a que los dispositivos que disponen de poco ancho de banda y consumo de energía bajo, puedan disponer de la posibilidad de encriptar y descifrar los datos.

Sistemas y Redes IoT

Las necesidades y requisitos de los dispositivos, las aplicaciones y los servicios correspondientes del IoT serán únicas en cada caso, en lo referente a ancho de banda y la latencia aceptable, para que sea funcional. Además, dependerá de las posibilidades de conexión que la situación geográfica permita. Imaginemos conectar un dispositivo en una oficina y otro en una planta petrolífera.

Podemos distinguir 4 grandes bloques en la conectividad para la IoT.

1) **Conexión inalámbrica tradicional**: Aquí podemos distinguir las conexiones WiFi y celular, esta última que va desde el GPRS al 4G (ya casi 5G)
2) **Tecnologías de corto alcance**: Claves para el éxito de la actual expansión de la IoT, aunque ineficaces para despliegues amplios. En estas tecnologías se incluyen ZigBee, Z-Wave, 6LowPAN, etc.
3) **Corto alcance entre objetos y dispositivos móviles**: Entran dentro de la conectividad de corto alcance, como las conexiones Bluetooth y su versión de consumo reducido BLE(*Bluetooth Low Energy*), así como la tecnología NFC
4) **Nuevas tecnologías nativas**: Existe un surgimiento de nuevas tecnologías, cuyas características principales son: largo alcance, bajo consumo y no menos importante, bajo coste. Se pueden destacar tres principalmente:
    a) Sigfox: Operador francés que en España es operado por Cellnex Telecom.
    b) Lora: Tecnología alternativa a Sigfox, mediante el despliegue de redes privadas (actualmente no está disponible en España)
    c) Evoluciones LTE/4G adaptadas a la IoT: Versiones reducidas del 4G cuyas características se basan en la optimización de su bajo consumo de datos, así como de una baja necesidad energética. El ancho de banda es superior con respecto a Sigfox y Lora.

3.2 Ciclo de vida de la IoT

Desde un punto de vista técnico, un sistema IoT no es más que un conjunto de Servicios distribuidos, colaborando entre sí para que las aplicaciones IoT consigan culminar algún tipo de objetivo. Un diseño correcto del sistema es esencial para poder elaborar un sistema IoT eficaz y seguro. Sin embargo, es en este punto dónde podemos ver que la tecnología IoT, además de compleja, está bastante inmadura, ya que no existe un modelo totalmente definido y no hay demasiada información al respecto.

Como sucede con la mayoría de las arquitecturas de sistemas, el ciclo de vida puede ayudar a detectar problemas por una mala definición e implementación. Debido a la gran cantidad de elementos que toman parte dentro de la arquitectura de sistemas IoT, algunos de ellos con sus propios ciclos de vida, una visión global del ciclo de vida facilitará la identificación de algunos problemas que no son tan obvios a simple vista.

El ciclo de vida de un producto o sistema es la serie de etapas que atraviesa desde el inicio hasta que es retirado. Sería el equivalente al ciclo de vida típico para un sistema de *software*, que consta de seis etapas: análisis, diseño, implementación, prueba, actualización y mantenimiento. Gracias al ciclo de vida de un sistema, se puede obtener información sobre las actividades que están involucradas en cada una de las etapas del ciclo de vida, así como las partes responsables y afectadas por estas actividades, pudiendo detectar los problemas que puedan surgir.

A continuación se muestra el ciclo de vida que proponen Leila Fatmasari Rahmana, Tanir Ozcelebia y Johan Lukkien[18]

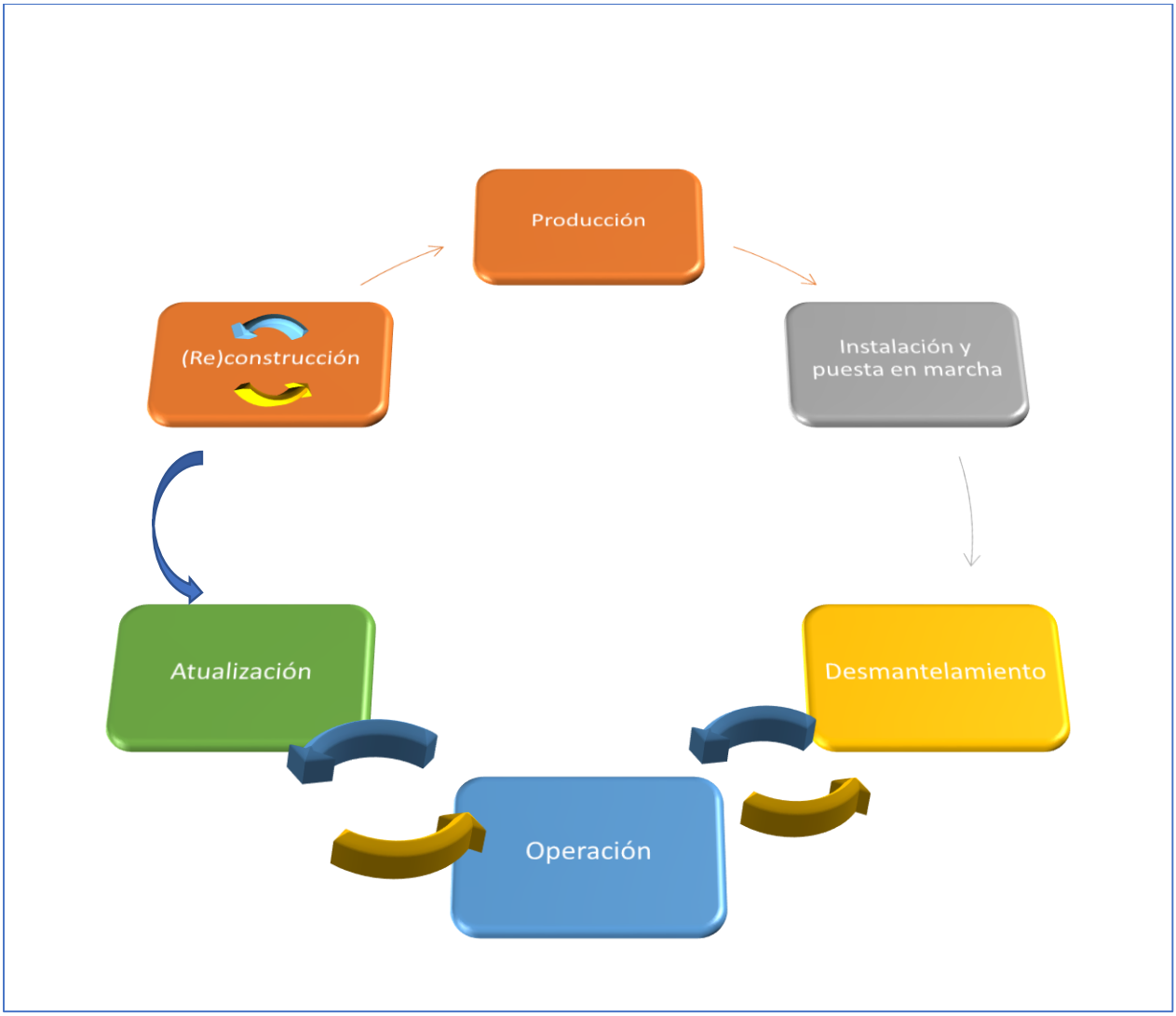

Figura 4 Ciclo de vida para la IoT

Etapa de (Re)Construcción

Incluye las siguientes sub-etapas:

- Construcción: Desarrollo del *software* inicial y del *hardware* del dispositivo
- Reconstrucción: Desarrollo de actualizaciones de *software* y actualizaciones de *firmwares*
- Desarrollo del ciclo de vida: Especificaciones, diseños, implementaciones y pruebas

Etapa de Producción

- Producción en masa del *hardware* del dispositivo
- Despliegue masivo del *software*
- Despliegue masivo de los componentes criptográficos

Etapa de Instalación y puesta en marcha

- Instalación:
    - Montaje y encendido
    - Conexión a la red
- Puesta en marcha:
    - Iniciar los componentes del sistema
    - Configuración de la ubicación e información de grupo
    - Configurar los parámetros de operación
    - Configurar los componentes criptográficos
    - Optimizar el arranque
    - Unirse a una red securizada

Etapa de Operación:

- Ejecutar los componentes del sistema que son usados por los servicios IoT y las aplicaciones mientras están ejecutándose
- Ejecutar los servicios y aplicaciones predeterminados de fábrica
- Monitorear el sistema

Etapa de actualización

- Ejecutar funciones de gestión para:
- Despliegue de las aplicaciones y servicios
- Terminación de las aplicaciones y servicios
- Reconfiguración de las aplicaciones y servicios
- Componentes del sistema y actualización del *firmware*

Etapa de desmantelamiento

- Restablecimiento de los valores de fábrica
- Desconectar el dispositivo de los servidores
- Apagado
- Eliminación física del dispositivo
- *End of life* del dispositivo

Aunque estas etapas pueden diferir de un dispositivo a otro, en términos generales, la etapa de Construcción hace referencia al desarrollo del dispositivo, tanto los componentes *hardware* como *software* iniciales, o como se suele decir, por defecto de fábrica, así como la reconstrucción del *firmware* y *software* inicial.

Ambas actividades tendrán su propio ciclo de vida, incluyendo las especificaciones, diseño, implementaciones y pruebas.

Tras la etapa anterior, el dispositivo se lleva a producción, es decir, fabricarlo a gran escala. En esta etapa, los componentes criptográficos también se pueden implementar.

En la etapa de instalación y puesta en marcha, se prepara al dispositivo para que las comunicaciones se realicen de forma segura dentro de la red. Como vemos en el esquema anterior, son varias las operaciones a realizar. Aquí vendrán apartados como la configuración de claves, certificados, configuración de parámetros, etc. Es una etapa esencial para que el dispositivo sea funcional y esté securizado. Hay que tener en cuenta que es importante para el usuario final, que estas acciones se realicen de forma sencilla. No por ello, se debe descuidar la seguridad. Encontrar el término medio es la clave para encontrar un equilibrio entre una seguridad óptima y la facilidad de uso.

De la etapa de operación dependerá el rendimiento del sistema. La elección correcta de la arquitectura y tecnología de los componentes afectará en gran medida a que este rendimiento sea óptimo.

La etapa de actualización se realizará cuando el equipo tenga que actualizar alguno de sus componentes, siendo la red el medio para ser actualizados en la mayoría de las ocasiones.

La última etapa es la de desmantelamiento o puesta en fuera de servicio. Esto puede suceder cuando haya un cambio de red o cuando se tenga que restablecer el dispositivo a los valores iniciales de fábrica. Resaltar la importancia de comunicar al servidor esta desconexión, para que éste sepa que ya no va a estar disponible y así el sistema podrá adaptarse a este cambio.

IoT según sectores

La IoT es muy dependiente del sector al que está orientado. Así es que el ciclo de vida que se acaba de plantear, tendrá diferencias sutiles según los dispositivos, los servicios y las aplicaciones que estemos tratando. Veamos a continuación una serie de ejemplos de uso en distintos sectores:

- **Energía**: La IoT se está empleando con el objetivo de una gestión más eficiente en los consumos energéticos (*Smart Cities*). Las operaciones más comunes son las de monitorización, mantenimiento y operaciones en remoto.
- **Transporte**: Previsión de rutas óptimas, tráfico más fluido, reducción del gasto energético, así como del impacto ambiental, optimización de flotas de transporte, son unos claros ejemplos del buen uso de la IoT en este sector.
- **Agricultura**: aprovechamiento más optimizado de los recursos naturales, gracias a la información aportada por los dispositivos y la automatización de las tareas.
- **Domótica**: El ahorro energético, la implicación en la seguridad y el aumento del confort son claros ejemplos de cómo el IoT está entrando en los hogares.
- **Industria**: Gracias a la monitorización continua, se consiguen ahorros en los costes y un mejor mantenimiento.
- **Logística**: El control que aporta en tiempo real de las localizaciones optimizan los servicios de este campo.
- **Deportes**: Monitorizaciones en tiempo real con la gran cantidad de dispositivos *wearables* que existen actualmente
- **Salud**: Incremento en la calidad de vida y disponibilidad de ser atendidos mediante tele-asistencia y monitorizaciones remotas

Estándares en la IoT

La falta de una estandarización general, es la tónica en la IoT. La Agencia de la Unión Europea para la Seguridad de las Redes y la Información (ENISA), ha realizado recientemente un estudio sobre el análisis de brechas en los estándares de seguridad de la IoT (https://www.enisa.europa.eu/publications/iot-security-standards-gap-analysis), que proporciona unas directrices que faciliten la adopción y desarrollo de dichos estándares, así como enumeración de brechas (*gaps*) en lo referente a la seguridad.

Como se ha comentado en este trabajo, no puede existir una solución específica que incluya a toda la IoT. Pero sí es posible abordar las normas existentes de seguridad de las que se disponen actualmente.

Y como tal, desde agosto de 2018, existe la norma de referencia ISO/IEC 30141, Internet of Things (IoT)—Reference Architecture. Mediante esta norma, se proporciona una serie de directrices con la intención de una serie de puntos comunes para el diseño y desarrollo de aplicaciones en la Internet de las Cosas, reforzando la seguridad y la protección, mediante la creación de sistemas fiables en cuanto a la privacidad y vulnerabilidades posibles.

Así por ejemplo, se dispone en el apartado 11.3.3 de la norma, una serie de medidas que pueden apoyar en el ciclo de vida de la IoT, en todos los niveles, que abarcan el diseño, el desarrollo, la fabricación e incluso la distribución.

Como se comenta en el documento de ENISA, aunque las normas no son esenciales, sí dan una garantía en cuanto a la interoperabilidad de los dispositivos así como la seguridad de los mismos. Destacar el Anexo B del documento de ENISA, en el que se realiza una propuesta para la evolución de los estándares de seguridad en el ámbito de la IoT.

3.3 Posibles amenazas en los ecosistemas

Las funciones principales [19] que realicen los dispositivos IoT son las siguientes:

- Adquisición y control de los datos.
- Procesar y almacenar los datos.

Para realizar estas funciones, debemos entender el *hardware* de los que se componen estos dispositivos. Esencialmente contienen sensores, accionadores o una combinación de ambos. Mediante los sensores se adquieren los datos y con los accionadores se controlarán los datos o se actuarán en base a los mismos.

Es así que mediante los sensores, se supervisará lo que quiera que sea la función del dispositivo, brindando datos como la intensidad de la luz, la temperatura o por ejemplo, el nivel de la batería.

Mediante los accionadores, se controlará un interruptor, la temperatura de un termostato, o un motor de un aspirador.

Sea cual fuere la función del dispositivo, la coordinación de los datos que mandan los sensores, el almacenamiento de los mismos, todas las funciones serán realizadas por el *firmware*.

Hablamos del *software* que está integrado y hace que el *hardware* y el mundo exterior interaccione. Existen dos categorías de *firmware*, el integrado y el basado en el sistema operativo.

Firmware integrado:

Para suplir las limitaciones de los dispositivos IoT, es frecuente encontrar que tiene un *firmware* incorporado y personalizado. Los ingenieros que deben programar estos *firmwares* para poder ejecutar el *hardware* del dispositivo, también tiene que hacerlo con el *software* para que interactúe con el *hardware* y el *software* de aplicación para que el usuario se comunique con el dispositivo.

Firmware basado en Sistemas Operativos:

Las funcionalidades de los dispositivos IoT cada vez son más complejas. Tienen más sensores, procesan más datos y una mayor capacidad de

almacenamiento. Esto hace que el *software* necesario para gestionarlos sea más complicado. En cierto modo, está sucediendo lo que pasó con los primeros ordenadores. En la época del Sinclair ZX Spectrum, muchos traían todo el *software* integrado en *firmware*. Al ser esto claramente insuficiente, los dispositivos se están implementando con una capa de abstracción, entre el sistema operativo y el *hardware*, y otro *software* que ejecuta las funciones del dispositivo.

Con esta abstracción del *hardware* del dispositivo respecto al *software* de aplicaciones, se pueden usar versiones reducidas de Linux, que tienen muchas de las funciones comunes de este sistema operativo y ocupa muy poco espacio.

Las comunicaciones de los dispositivos IoT casi siempre se realizan de forma inalámbrica, pudiendo disponer de ellos en cualquier ubicación sin necesidad de cableado. Algunos están preparados para realizar conexiones WiFi 802.11 directas con el *router*. Desde ahí se conectará a Internet, como es el caso de las cámaras de seguridad, que necesitan un ancho de banda considerable. Otros funcionan en conjunto con una serie de dispositivos IoT. Es el caso de Alexa de Amazon, que además de interaccionar en Internet, es capaz de controlar dispositivos que estén preparados para ello.

Para gestionar un dispositivo IoT, se realiza en dos pasos que son: conectándolo a la red, que se denomina suministrarlo y una vez conectado se supervisa y controla.

Existen muchos dispositivos IoT, en especial los que realizan pocas funciones y son pequeños, como un sensor de temperatura, que no tiene integrado la parte de *hardware* necesaria para que interactúen con los usuarios. Por lo tanto, carecen de una pantalla táctil. Estos dispositivos reciben la nomenclatura de “descabezados”.

Un modo muy común para conectarlos es a través de la habilitación en el *router* del WPS (WiFi Protected Setup). Así, pulsando un botón en el enrutador, y el correspondiente del dispositivo, quedan conectados.

Otros sin embargo, crean puntos de acceso WiFi, mediante los que se podrán conectar a través de un Smartphone y así acceder a los programas de configuración, pudiendo así configurar los parámetros necesarios para acceder al WiFi.

Y los dispositivos *gateways*, buscarán y añadirán los dispositivos que van detectando en modo configuración o emparejamiento. Así cuando el *gateway* está configurado para que tenga acceso a Internet, se dispone a los dispositivos en modo emparejamiento para que se puedan conectar al *gateway*.

La supervisión y control del dispositivo es posible una vez que está conectado a la red. La forma más común es realizarlo mediante un Smartphone. En el caso de las cámaras IP, se conectan directamente a Internet. De hecho, es posible conectarse directamente a ellos sin tener que usar un proveedor de servicios en la nube. Cuando se "*natea*" en el enrutador las direcciones de los dispositivos, estos se pueden acceder desde Internet para supervisarlo y controlarlo.

Y por último, entraremos en el tema de la seguridad. No es casualidad que se contemple como el último punto dentro de este apartado. La seguridad suele ser el último paso dentro de la configuración de los dispositivos. No vamos a negar que unos dispositivos con recursos tan limitados, que sean asequibles económicamente, que además consuman poca energía y que se puedan conectar de forma inalámbrica, el que lleguen a implementar una seguridad que lo convierta en confiable, sea complicado. Al fin y al cabo, si se implementa una buena seguridad, no los convierte en tan rentables ni asequibles al público en general y es entonces, con un nivel de seguridad bajo, cuando se producen los ataques de *malware* a estos dispositivos.

En cualquier tipo de ataque, el atacante tiene que golpear en lo que se denomina la superficie de ataque [20], que hace referencia al conjunto de todas las vulnerabilidades de un dispositivo. Dentro de esa superficie, se identificará un vector de ataque, que no es más que la ruta mediante la cual se puede utilizar el dispositivo con un propósito malicioso. Los vectores de ataque más comunes son los siguientes:

Contraseñas débiles

No se trata únicamente de que dentro del ciclo de la vida del desarrollo de los dispositivos, la seguridad apenas ocupa importancia. Además, los fabricantes quieren que sus dispositivos se puedan configurar de una manera sencilla y sean fáciles de usar y configurar. Al fin y al cabo, son conscientes de que los usuarios finales no tendrán demasiados conocimientos técnicos. Así que la decisión final es la de que los usuarios se identifiquen ante el dispositivo mediante la combinación de usuario y contraseña.

El problema principal que encontramos con este tipo de simplicidad en la seguridad, es que los usuarios no suelen cambiar las credenciales iniciales de autenticación del dispositivo. Suelen ser combinaciones del estilo admin/admin, admin/1234, etc. Así tendremos que esta combinación de usuario y contraseña, se añade en la lista de vulnerabilidades conocidas del dispositivo.

Además tenemos que el cifrado en las comunicaciones no son consideradas ni siquiera en la fase de desarrollo de los dispositivos. Es por esto que es necesario reconsiderar esta política, y que la criptografía se incorpore para manejar los cifrados de los datos y las autenticaciones de los dispositivos.

Puertas traseras

Algunos dispositivos incluyen algún mecanismo de acceso, que en principio no es conocido por los usuarios. Esto se suele hacer con vistas a mantener un soporte del dispositivo. El problema está en que cuando se descubre esa puerta trasera, es muy sencillo de que sea *hackeado*. Y esta suele ser un identificador de usuario/contraseña o un puerto abierto del dispositivo. Las soluciones suelen pasar por cerrar la puerta con una actualización de *firmware*.

Conexión a Internet

Cualquier dispositivo conectado a Internet, queda expuesto a cualquier tipo de ataque. Esto es extensible no solo a la IoT. Sin embargo, como estamos viendo, la seguridad no es el punto fuerte de estos dispositivos. Así que para el caso de los IoT, el peligro es mayor.

Dentro del Open Web Application Security Project (OWASP) [21], existe un proyecto aún sin acabar a día de hoy de lo más interesante en este campo, con una lista de las vulnerabilidades potenciales a las que son susceptibles los dispositivos IoT.

Todo ataque se compone de una exploración inicial del dispositivo y de una toma de control del mismo. Esto se realiza mediante un programa de Comando y Control o CNC [22]. Este programa explora las direcciones IP buscando puertos abiertos e intentando iniciar sesión con un conjunto de identificadores predeterminados conocidos [23] (los famosos admin/admin, root/admin, root/123456, etc)

Una vez logados con éxito, un *script* informa de la dirección IP del dispositivo, el puerto localizado y las credenciales exitosas. Es entonces cuando el CNC introduce en el dispositivo un malware y este queda *pwned* [24]. Se queda en lo que se denomina un estado zombi, esperando a las instrucciones del CNC para comenzar el ataque. Cada dispositivo localizado e infectado se denomina Bot. El conjunto de estos Bots conforman la red de Bots o Botnet.

Debido a la poca potencia de ataque que puede llegar a generar un solo dispositivo IoT para conseguir una denegación de servicio, será mediante una Botnet formada por miles de ellos, con la que se pueda generar un ataque significativo. Esto también es válido para conseguir mandar *spam*.

4. Reconocimiento de dispositivos IoT

4.1 Motores de búsqueda de la IoT. Un ejemplo práctico. Shodan

Un buscador orientado a la IoT no es más que otro buscador, en la que la información que ofrece está directamente relacionada a la Internet de la Cosas. El para qué se emplee esta información, dependerá exclusivamente de las intenciones del usuario que disponga de la misma.

Mediante estos motores de búsqueda, se puede llegar a conocer todo tipo de vulnerabilidades existentes en los dispositivos que conforman la IoT, como cámaras IP, *routers* sin protección, incluso cajeros automáticos y cualquier "cosa" que esté conectado a Internet.

Actualmente, existen una serie de motores de búsqueda [25] relacionados con la IoT, que por su uso extendido se han convertido en los más potentes y populares. Estos son los siguientes:

- **Shodan**: Es el más conocido de todos. Su extensa base de datos y sus continuas actualizaciones lo convierten en una herramienta imprescindible para la seguridad de la IoT. Se puede consultar en https://www.shodan.io/
- **FOFA**: Motor de búsqueda de activos en Internet, lanzado por WhiteHat Su dirección es https://fofa.so
- **Censys**: Otro motor de búsqueda en https://censys.io/. Desde la página principal podemos hacer una simple consulta por IP para ver los dispositivos expuestos a Internet.
- **Oshadan**: En https://www.oshadan.com/ encontraremos una herramienta con funcionalidades muy similares a Shodan.
- **Zoomeye**: Otro buscador que encontrará *hosts* y *webs* que corresponderán con el patrón dado a buscar. Se puede consultar en htttps://www.zoomeye.org

Comparativa entre los motores de búsqueda

En este punto del proyecto, resulta interesante plantear una comparativa entre los dos principales motores de búsqueda Shodan y Censys, centrándonos en sus aspectos positivos y negativos en cuanto al tipo de búsqueda que nos interesa.

Shodan:

A favor:

- Permite búsquedas personalizadas gracias a las opciones de filtrado que tiene implementados.
- Shodan es soportado por una comunidad muy numerosa, la más grande y activa de todos estos motores de búsqueda, por lo que constantemente recibe inyecciones de información nueva y es sencillo obtener apoyo en dudas y consultas.
- Su API favorece la posibilidad de integración en los proyectos IoT que se plantean y resulta muy sencillo integrarla.
- La generación de informes de manera automatizada facilita a los auditores su trabajo, mediante muestras gráficas de sus resultados, siendo esta opción, de las más completas comparado con otros buscadores.
- Dispone de una gran cantidad de documentación, no sólo de forma oficial si no mantenida por los usuarios del motor de búsqueda.
- Recibe una constante actualización de funcionalidades.

En contra:

- Las opciones para los usuarios que se registren de manera gratuita son muy limitadas.
- Es lento cargando datos en sus servidores
- Si se compara con otros motores, tiene pocos filtros de búsqueda.

Censys

A favor:

- Dispone de una API bastante completa.
- No pone limitaciones a los tipos de servicios que se pueden consultar.
- Se pueden realizar búsquedas no sólo de dispositivos en IPv4, sino de sitios web y de certificados.

- El sistema de filtrado es muy completo.
- Las búsquedas se realizar de manera rápida, aunque esto podría deberse a que tiene menos usuarios que otras plataformas como Shodan.
- Las búsquedas y el uso de la API son gratuitas.

En contra:

- La comunidad que lo conforma es menor en comparación con Shodan. No hay tanto soporte como en Shodan.
- El sistema de generación de informes es muy limitado.
- Su API no es tan sencilla de usar e integrar.

Pese a la extensa documentación, no es fácil encontrar información detallada sobre cómo funcionan estos motores. A continuación se muestra la comparativa de funcionamiento de los dos motores de búsqueda[26].

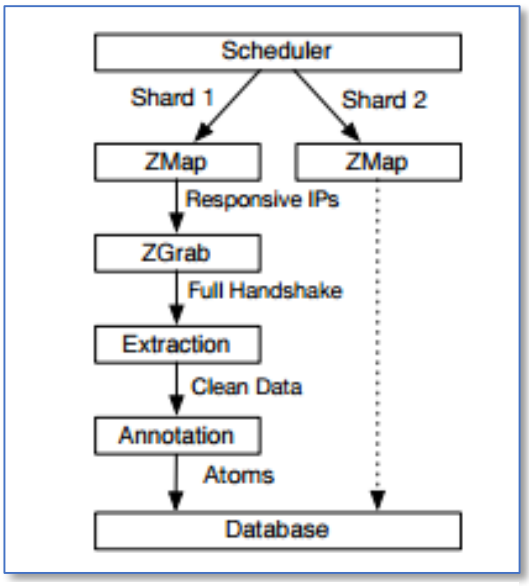

Figura 5 Búsqueda en Censys

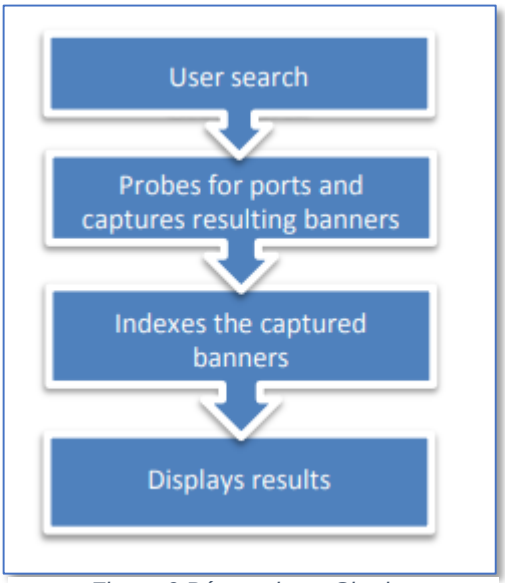

Figura 6 Búsqueda en Shodan

Buscador Shodan

Es el más conocido y usado de todos. Para comenzar, se debe aclarar que Shodan no es gratuito. Tiene tres tipos de facturaciones posibles, dependiendo de las necesidades de uso, es decir, el número de consultas de IP al mes, de resultados, más filtros para la realización de consultas, etc.

Además disponen de una API [26] que permite una integración más sencilla para los desarrolladores. La información que proporciona esta API es mucho mayor que la visualizada en los resultados web y mediante el uso de la misma, se puede automatizar de modo que se generen informes periódicos e incluso que muestre notificaciones en tiempo real.

El creador de Shodan, John Matherly, destaca en su libro [27] los modos en los que éste recopila información, que son:
Banner: Es el modo principal que usa Shodan. A través de los banners, se obtiene información textual que describe un servicio en un dispositivo.

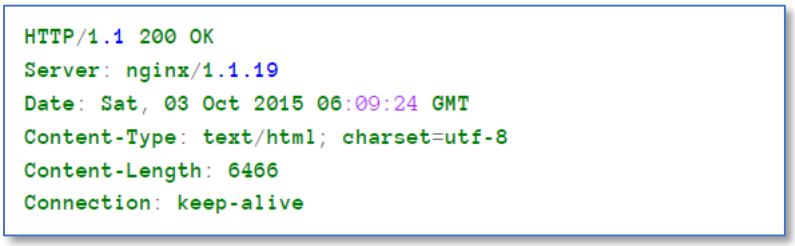

Figura 7 Ejemplo de Banner

Por ejemplo, en la imagen anterior podemos ver que el dispositivo del que se ha obtenido la información, está ejecutando la versión 1.1.19 de un servidor web nginx. La información será totalmente distinta de un dispositivo a otro, como se puede apreciar en la siguiente captura, en la que se muestra un dispositivo Siemens.

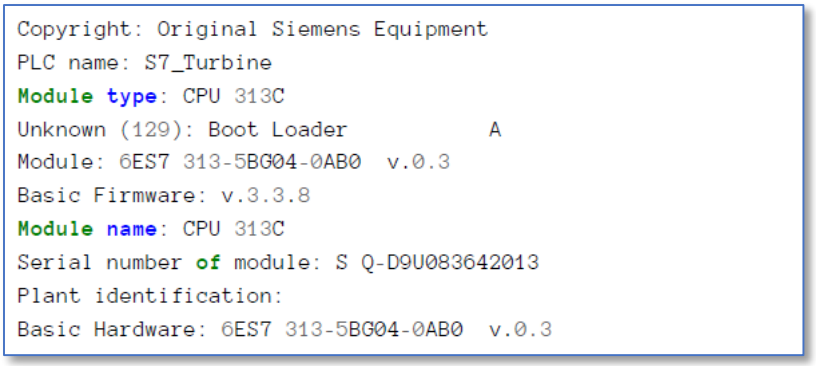

Figura 8 Banner de un dispositivo Siemens

En este caso podemos ver como el protocolo S7 de Siemens devuelve otro tipo de información, como el *firmware* que se está usando, el número de serie y el modelo de dispositivo, entre otros datos.

Metadatos de los dispositivos

Con los metadatos se puede obtener información tal como la ubicación, el sistema operativo, el nombre del *host*, etc.

IPv6

Desde Octubre de 2006, Shodan recopila información de *banners* de dispositivos que son accesibles en IPv6. Evidentemente, la información recolectada actualmente es mucho menor que IPv4.

La recolección de datos se produce de forma continua en todo momento. Para evitar el bloqueo de IPs por parte de ciertos países, los rastreadores de Shodan están distribuidos en diferentes localizaciones de todo el mundo, como son Estados Unidos, China, Francia y Vietnam, entre muchos otros.

El algoritmo de Shodan consiste en tres sencillos pasos como se describe a continuación:

- Generación de un número IP de forma aleatoria
- Generación de un puerto aleatorio (de entre un número predeterminado)
- Comprueba esa dirección IP con ese puerto y obtiene el *banner*
- Repetir desde el primer punto

Y ¿qué dice Shodan sobre la seguridad en España?

Podemos encontrar un enlace directo en https://exposure.shodan.io/#/ES, donde Shodan facilita una tabla de exposición a internet con un resumen de la información sobre la Seguridad en España. La consulta corresponde al 12 de abril de 2019. Se ha podido comprobar que los datos se actualizan constantemente de un día para otro.

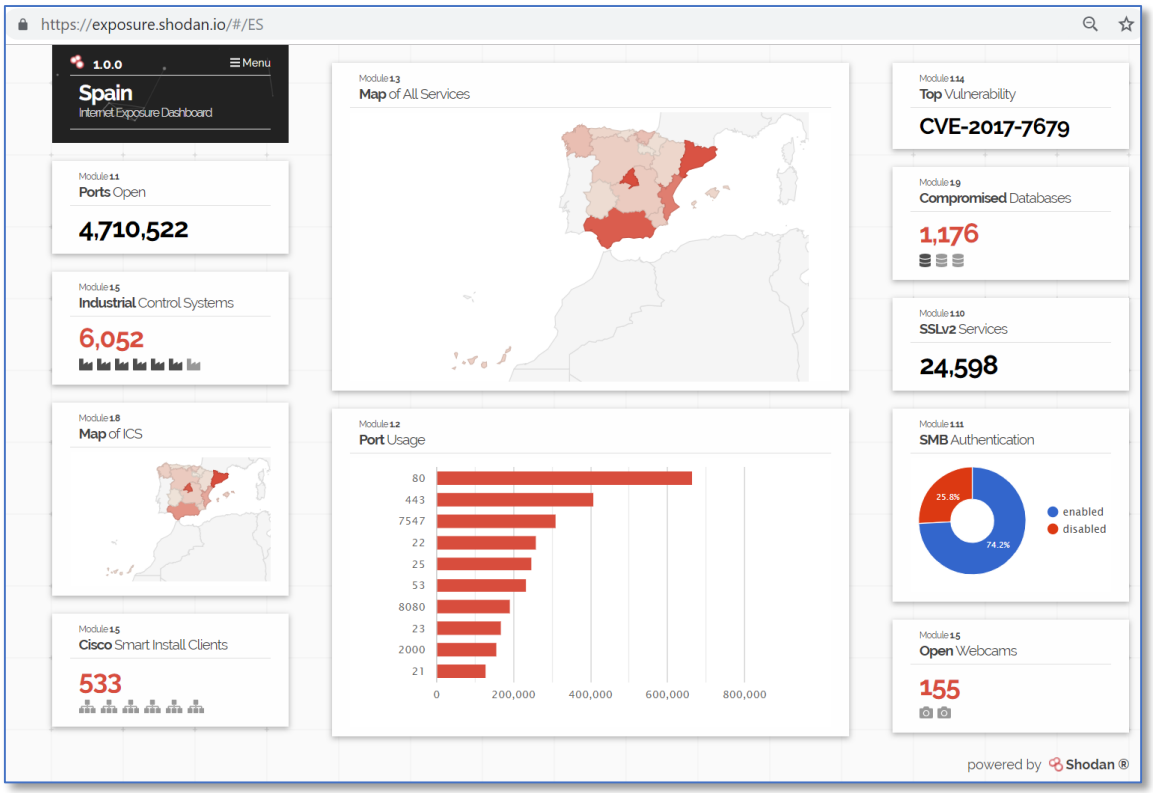

Figura 9 Internet Exposure Dashboard en España según Shodan

Se puede observar que Madrid, Cataluña y Andalucía, por este orden, lideran el ranking donde se encuentran más dispositivos con incidencias, tales como puertos abiertos, Bases de Datos Comprometidas, Webcams abiertas, Autenticaciones SMB, etc.

El puerto 80 es el que más se ha encontrado abierto en los dispositivos consultados, seguidos por el 443. También podemos ver como el 7547 es el tercero más encontrado. Este puerto suele estar relacionado con ciertos modelos de *routers*. En noviembre de 2016, mediante una vulnerabilidad de ciertos *routers* de la marca Zyxel [28] que disponían de este puerto abierto, fueron

atacados por la Botnet Mirai, dejando a los usuarios de Deutshe Telecom durante dos días sin conexión a Internet.

Usando Shodan

Lo primero que debemos hacer es registrarnos en Shodan para poder usar filtros en las búsquedas. El uso de filtros es nombre_de_filtro:valor_de_búsqueda

Es así que por ejemplo, si se quisiera buscar las IPs de la ciudad de Barcelona, con los puertos 23 y 1023, para realizar la búsqueda sería mediante city:"Barcelona" port:23,1023, como se observa en la siguiente captura.

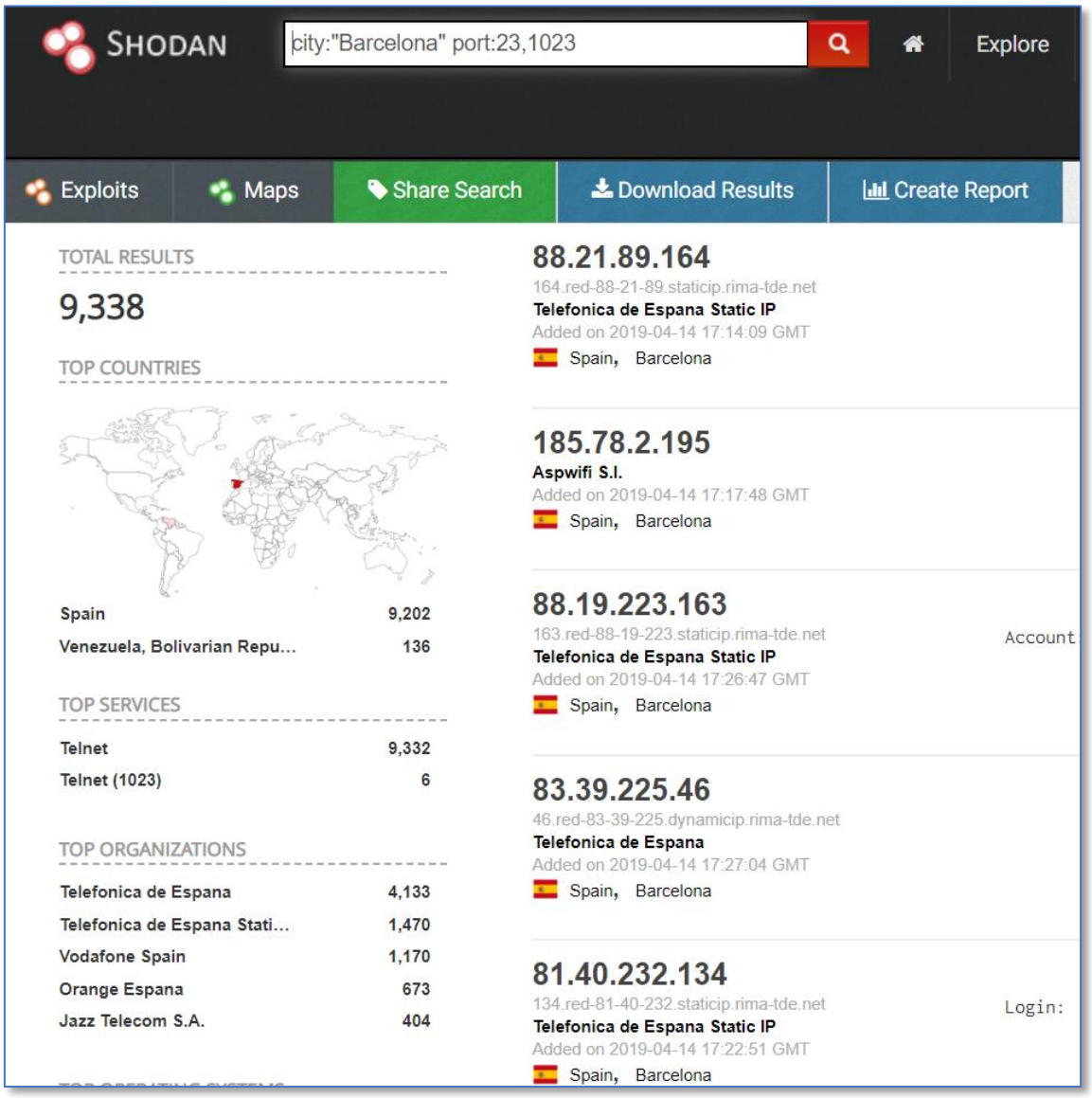

Figura 10 Búsqueda con filtros en Shodan

Una vez encontrados los resultados, se puede geolocalizar eligiendo la opción Maps. En la siguiente captura, vemos los resultados que se obtienen buscando el término UOC y la ciudad de Barcelona.

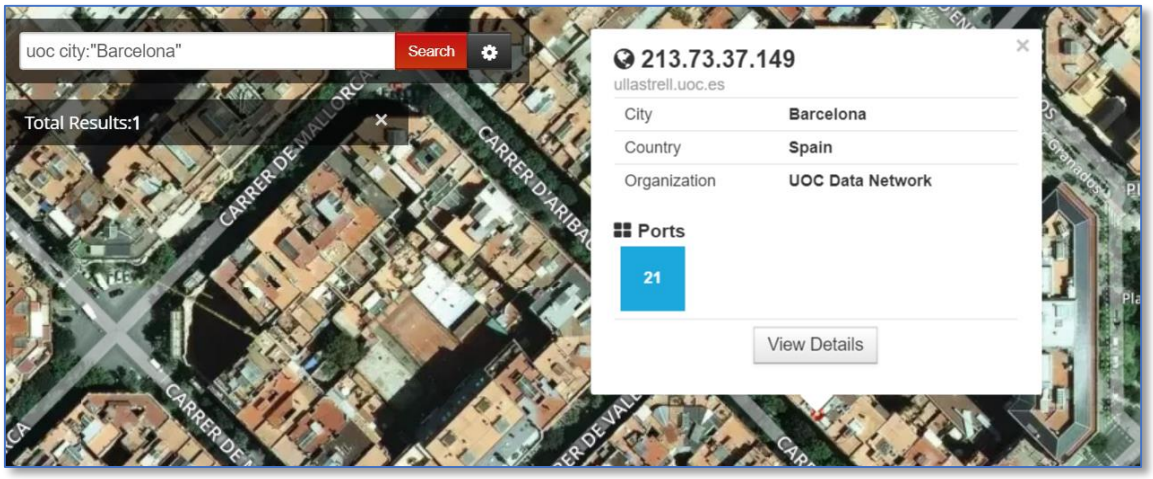

Figura 11 Búsqueda con filtros en Shodan

Si intentamos acceder a la dirección IP por ftp, efectivamente encontraremos una petición de autenticación de FTP.

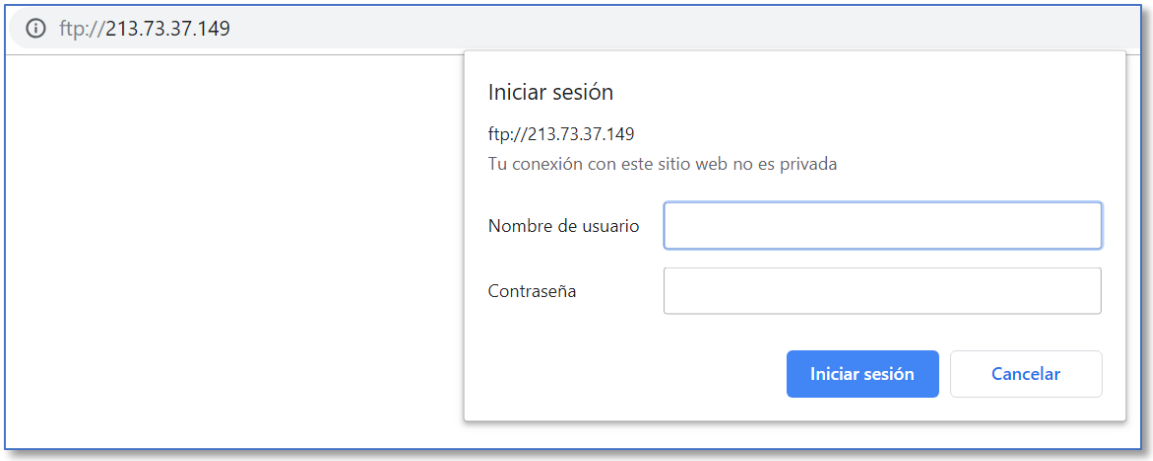

Figura 12 Inicio de sesión FTP

Las búsquedas con Shodan pueden ser más comprometedoras. Por ejemplo, imaginemos que se desea buscar los sistemas vnc que no requieren autenticación en Barcelona. Para ello, se usaría la siguiente búsqueda: city:"Barcelona" rfb authentication+disabled. Como podemos ver, los resultados incluyen capturas de pantalla en algunos casos.

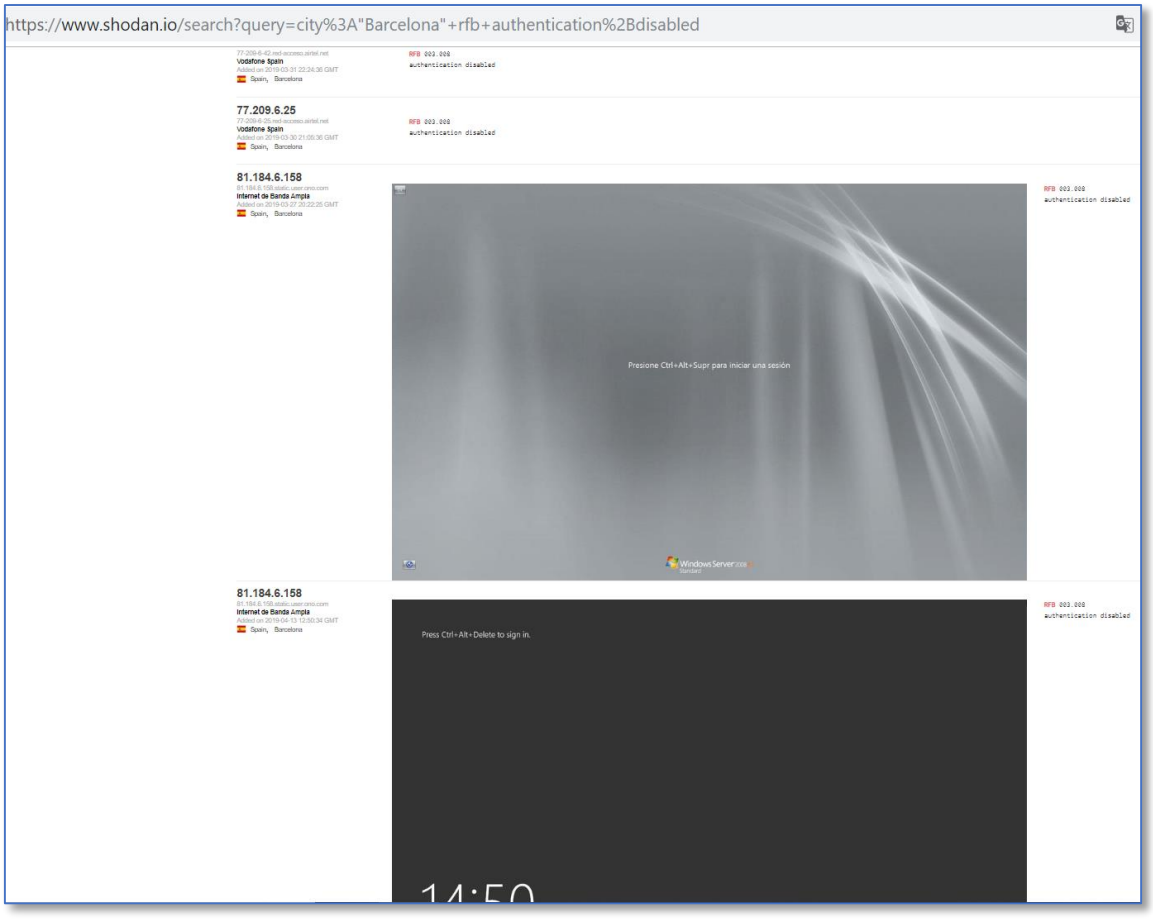

Figura 13 Búsqueda con filtros en Shodan

Por último, veremos un claro ejemplo de descuido en la seguridad. Como se puede observar, con una simple búsqueda, se ha encontrado una webcam, que no sólo no tiene usuario y contraseña, sino que permite el control de movimiento de la misma.

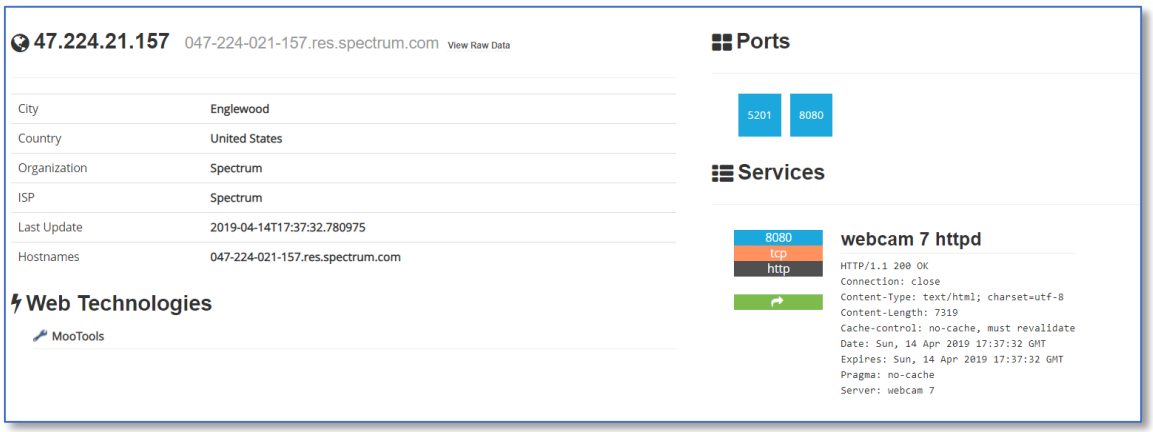

Figura 14 Datos de la webcam encontrada

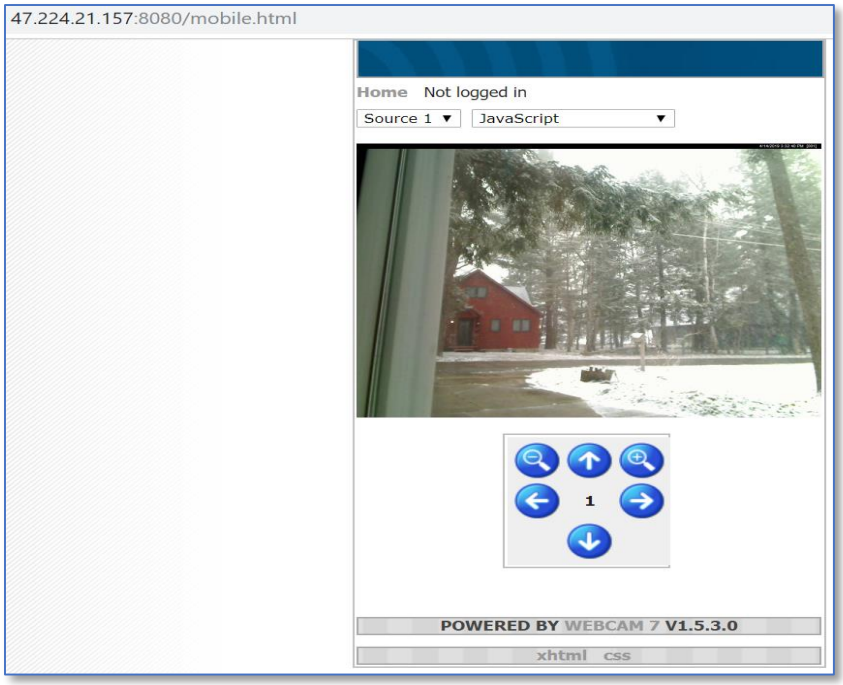

Figura 15 Accediendo a la webcam

Incluso podemos encontrar cámaras de seguridad como se observa en la siguiente captura.

Figura 16 Sistemas de seguridad

Se ha podido encontrar multivistas cuando son varias cámaras en la misma IP.

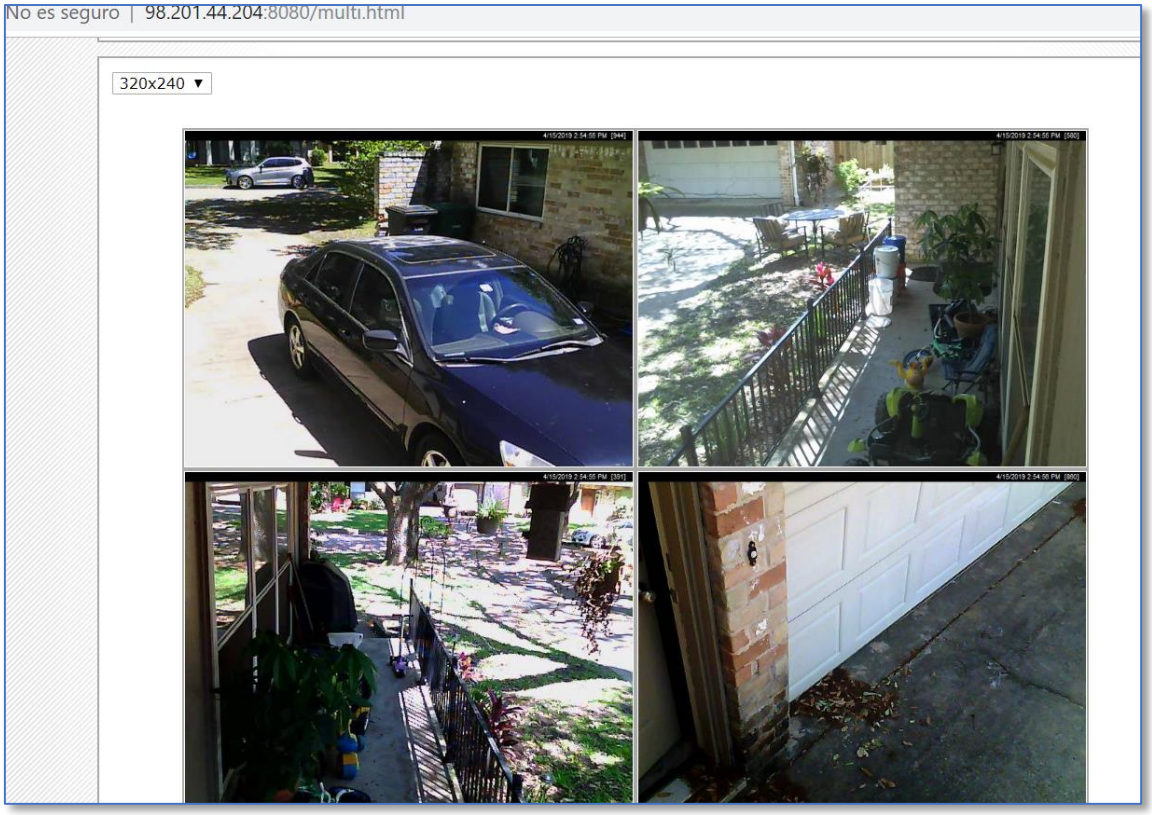

Figura 17 Múltiples cámaras

4.2 Obteniendo información.

El objetivo de este TFM, está claramente enmarcado en la recolección de información de una forma automatizada, para que posteriormente sean mostrados los hallazgos encontrados, inspirado en los buscadores más conocidos y como no podía ser de otra manera, de una manera legal y no intrusiva.

OSINT

Término que proviene del acrónimo formado por Open Source Intelligence, cuya traducción libre diremos que implica algo así como Fuente de conocimiento abierto. La principal característica de las fuentes de información clasificadas como OSINT, es que deben ser de acceso libre, es decir, se deben ofrecer de manera gratuita y además, la información que aporta sea no clasificada.

El uso de fuentes OSINT no tiene por qué estar relacionado con acciones malintencionadas. Por ejemplo [29], se podría obtener información sobre subvenciones en determinadas áreas, cartografiar un mapa tecnológico de un sector determinado, acciones que realizan las empresas en que compiten entre ellas. No deja de ser información muy valiosa para mejorar algunos aspectos competitivos.

Dentro de un proceso general de OSINT [30], tendríamos una primera fase de requerimientos, en la que se identificarán los objetivos que se persiguen; una segunda fase en la que se identifiquen las fuentes relevantes y por último una fase de proceso y tratamiento de la información para que puedan ser tratados.

A los buscadores IoT como Shodan, se les puede catalogar de fuentes OSINT, pero lo cierto es que casi todos tienen opciones de pago, aunque muchas funciones son gratuitas. Google, Bing, las redes sociales, son perfectos ejemplos de fuentes OSINT.

Escaneo de puertos IPv4

Antes de entrar en detalle sobre el escaneo de puertos, es necesario aclarar porqué sobre IPv4. Mediante IPv4 se pueden disponer de unos 4300 millones de direcciones IP, frente a los 340 sextillones de direcciones que ofrece IPv6 (32 bits vs 128 bits). Lo cierto es que pese a que las últimas direcciones IPv4 ya fueron asignadas en 2016, en España la implantación y uso de IPv6 roza unas cifras irrisorias, comparadas con otros países europeos, como podemos comprobar según los datos ofrecidos por Google, donde se analiza los accesos a su buscador según el tipo de IP, en la página siguiente:

https://www.google.com/intl/en/ipv6/statistics.html#tab=per-country-ipv6-adoption

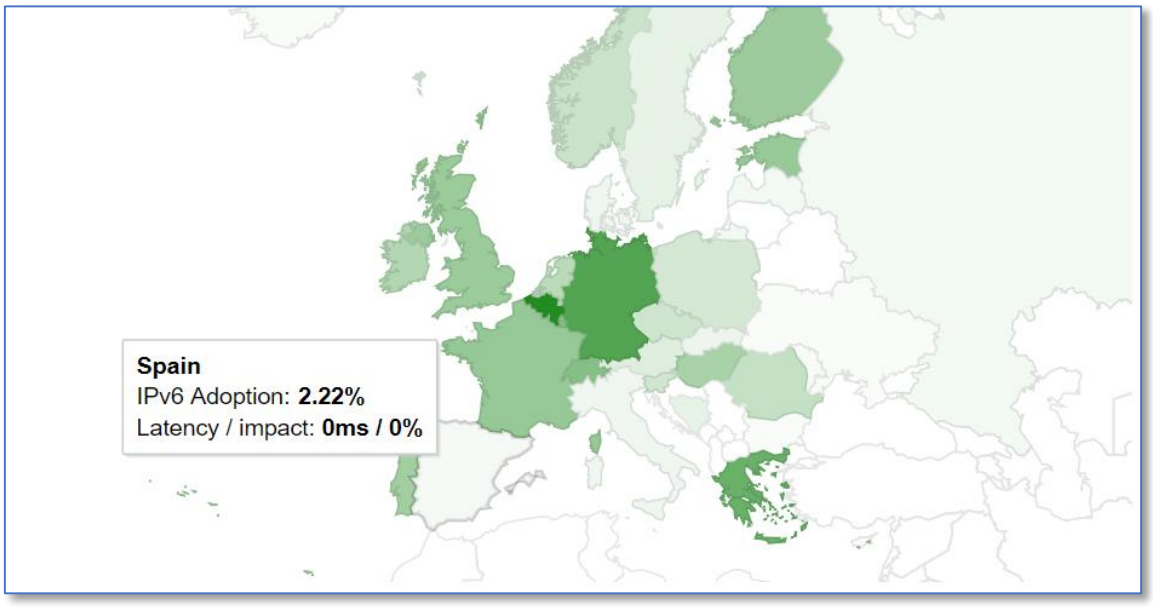

Figura 18 Acceso IPv6 en España según Google.

Esto se debe a diversas causas, entre las que destacar que el cambio de protocolo implicaría un cambio que obligaría habilitar el *routing* a través de IPv6. Es el caso de Movistar y las demás compañías, que están a la espera de que esta habilitación se produzca.

Además, se necesitan equipos que sean compatibles con los dos estándares, ya que durante un tiempo, tanto IPv4 como IPv6 deberán coexistir. Aunque los dispositivos de los últimos años soportan IPv6, aún existen una enorme cantidad que no lo tienen implementado.

Las ventajas de adopción de IPv6 son considerables, destacando la posible desaparición del NAT, la necesidad de traducir direcciones internas a públicas, la desaparición de direcciones dinámicas, la implementación directa de seguridad por IPsec, etc. De todos modos, parece que aún quedan unos años para que el número de dispositivos que se conecten por IPv6 sea superior al de IPv4. Es por esto, que este trabajo se centra principalmente en IPv4.

El escaneo de puertos IPv4 consiste en analizar qué puertos, tanto de tipo TCP como UDP, están activos dada una dirección IP. Mediante los puertos se envían los paquetes de información, usando los protocolos de la capa de transporte TCP y UDP. Un puerto de por sí ya constituye una inseguridad, es decir, todos los puertos son inseguros de alguna u otra manera. No existe ningún puerto que sea totalmente seguro de modo innato. Hablamos de 65.535 puertos de tipo TCP y otros tantos UDP.

Hay una serie de número de puertos bien conocidos que son asignados por IANA(Internet Assigned Numbers Authority), que van del 0 al 1023, como son los puertos 21(FTP), 22(SSH), 23(Telnet), 25(SMTP), 53(Protocolo DNS), 80(HTTP), 110(POP3), 139(NetBIOS Session Service) y 1023(Telnet).

El rango de puertos desde el 1024 al 49151 se denomina puertos registrados y los que pertenecen al rango entre 49152 y 65535, dinámicos o privados. En el Anexo 1 se muestra un listado de puertos más comunes relacionados con IoT.

Lectura de *banners*

Mediante la información que se obtiene a través de los *banners*, se puede obtener información variada sobre el sistema y los servicios que ofrece el sistema que se está consultando. Esta información puede incluir la tecnología *backend* usada y la versión, el firmware del dispositivo,

Este tipo de técnica de adquisición de información se denomina *banner grabbing*. Una de las formas más sencillas de realizarlo sería mediante un telnet hacia la IP deseada. Así, en un ejemplo real, sabemos que la IP 88.67.132.112 tiene el puerto 23 abierto. Si realizamos un telnet obtenemos la siguiente captura.

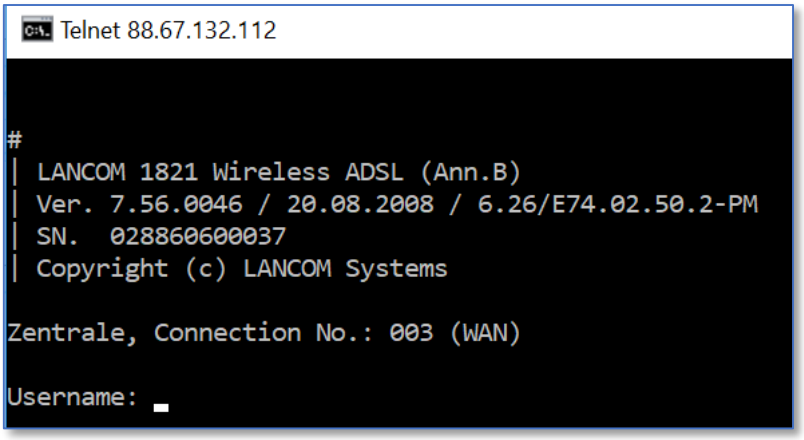

Figura 19 Telnet a un router ADSL LANCOM 1821

Como podemos observar, se trata de un *router* de ADSL LANCOM modelo 1821. También vemos que la versión y el número de serie aparecen disponible. La fecha de 20/08/2008 hace indicar que se trata de un dispositivo con una cierta antigüedad, lo que presupone que tendrá distintas fallas de seguridad. Si buscamos información sobre vulnerabilidades de esa versión, se podría saber cuáles tiene este dispositivo. Si vamos a la página del fabricante, observamos que la última versión del *firmware* fue la 8.82.

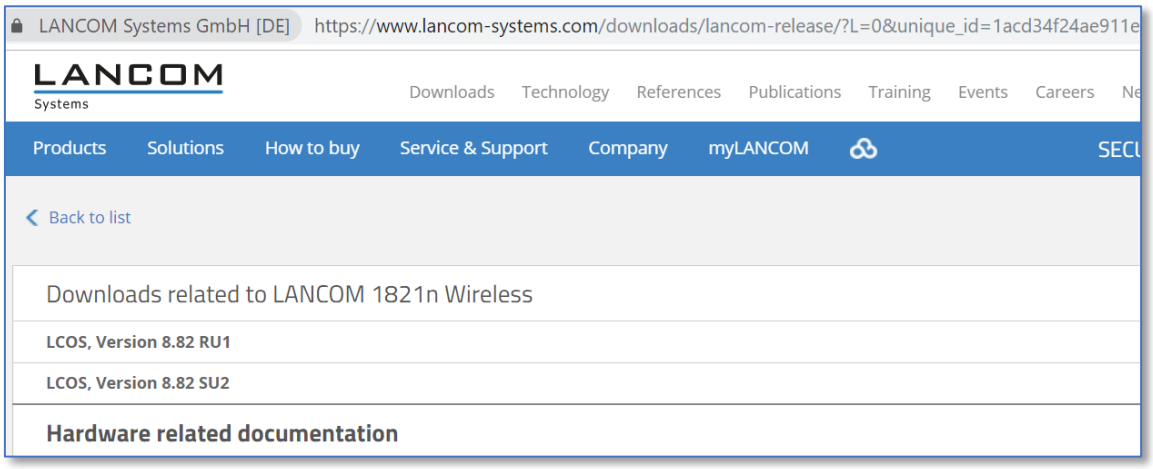

Figura 20 Último firmware disponible del router ADSL LANCOM 1821

Sin embargo, esta información es posible camuflarla e incluso ofrecer una información falsa. Así por ejemplo, en el caso de un servidor Apache [31], basta con que en el fichero de configuración httpd.conf se modifiquen las directivas ServerSignature Off y ServerTokens Prod. Con esto se consigue que sólo aparezca la palabra Apache en la información del *banner*. Si se modificase en el archivo httpd.h la línea #define SERVER_BASEPRODUCT “Apache” por

cualquier otro valor, éste sería el que apareciese una vez se recompilara e iniciara el servicio de Apache.

4.3 Proyecto a realizar

El lenguaje de programación elegido para la realización de este proyecto es Python [32]. Se trata de un lenguaje creado por Guido van Rossum [33] entre finales de los 80 y principios de los 90. Su nombre se debe a que el creador era muy aficionado al grupo humorístico Monty Python

La justificación de la elección de este lenguaje se basa en una serie de características que lo hacen ideal para el proyecto [34]:

- Es multiplataforma.
- Es *opensource.*
- Tiene una sintaxis sencilla, pero a su vez es rápido y potente.
- Existen muchas librerías y proyectos orientados a la seguridad informática escritos en Python.
- Con pocas líneas de código se consiguen realizar programas robustos.
- Es ideal para la realización de Pruebas de Concepto (PoC) [35] y prototipos de forma rápida.
- Dispone de frameworks sencillos de programar para la parte web del proyecto (Flax y Django)
- Hay mucha documentación disponible.

Dicho esto, comenzaremos a realizar las rutinas necesarias para poder llevar a cabo el proyecto planteado.

Geolocalización

Para la geolocalización de una IP, Python dispone de una librería llamada pygeoip. Además, necesitaremos una base de datos, donde se podrá realizar la consulta de una IP determinada. Esta base de datos se puede localizar en http://dev.maxmind.com/geoip/legacy/geolite/

En el siguiente *script* se muestra un ejemplo de uso de un objeto tipo pygeoip. Para el ejemplo, usamos la dirección IP que devuelve un DNS tras resolver www.uoc.edu

Así, una vez creamos nuestro objeto pygeoip, llamado obj usando la BD GeoLiteCity.dat, mostramos todos los datos que contiene, apoyándonos con la clase pprint para una representación formateada.

El código es el siguiente:

```
#!/usr/bin/python

#Ejemplo de script de geolocalización

import sys
import pygeoip
import pprint

ip = "213.73.40.242"

obj = pygeoip.GeoIP('GeoLiteCity.dat')

print("Ejemplo de script para Geolocalización de IP\n")

print ("-------------------------------------------------------")
pprint.pprint(obj.record_by_addr(str(ip)))
print ("-------------------------------------------------------")
print(" Nombre del País:",  obj.country_name_by_addr(str(ip)))
print(" Código del País:",  obj.country_code_by_addr(str(ip)))
```

Figura 21 Script Geolocalización

Y el resultado para la IP dada sería el siguiente.

```
Ejemplo de script para Geolocalización de IP

-------------------------------------------------------
{'area_code': 0,
 'city': None,
 'continent': 'EU',
 'country_code': 'ES',
 'country_code3': 'ESP',
 'country_name': 'Spain',
 'dma_code': 0,
 'latitude': 40.0,
 'longitude': -4.0,
 'metro_code': None,
 'postal_code': None,
 'region_code': None,
 'time_zone': None}
-------------------------------------------------------
 Nombre del País: Spain
 Código del País: ES
```

Figura 22 Resultado del script de Geolocalización

Direcciones IP

A continuación realizaremos un algoritmo mediante el cual se genere una dirección IP al azar IPv4, es decir, en formato de 4 números separados por puntos, donde el rango de cada uno puede ir de 0 a 255.

Para ello nos apoyaremos en la clase IPc4Address de la librería ipaddress y la clase randint de la librería random

El código sería el siguiente:

```
#!/usr/bin/python

#Ejemplo de script que genera IPs aleatorias válidas

from ipaddress import IPv4Address
from random import randint
def creaIp():
   ip = IPv4Address('{0}.{1}.{2}.{3}'.format(randint(0,255),randint(0,255),randint(0,255),randint(0,255)))
   if(ip.is_unspecified or ip.is_loopback or ip.is_multicast  or ip.is_reserved or ip.is_private or ip.is_link_local):
       return creaIp()
   return ip
ip=creaIp()
print("La IP creada al azar es: ",ip)
```

Figura 23 Script que genera una IP al azar

Como podemos observar, si en el proceso de generación de IP se crea una no válida para el propósito del proyecto, es decir, si es *multicast*, local, privada, etc., se repite el proceso hasta que se genera una dirección.

Creando IPs geolocalizadas en España.

Si juntamos los dos procesos, garantizamos que las direcciones IP que se generan pertenecen al ámbito territorial en el que este proyecto está orientado. Para ver la eficiencia de este proceso se ha implementado y ejecutado en un ordenador de capacidades muy reducidas. Concretamente se trata de una Tablet con Windows 10 instalado, con tan solo 4 Gb de RAM y un procesador Intel Atom x5-Z8350, como se observa en la captura.

Figura 24 Ordenador en el que se ejecuta el script

El código implementado es el siguiente.

```
#!/usr/bin/python
#Ejemplo de script que genera 10 Ips geolocalizadas en España

import sys
import pygeoip
import pprint
from ipaddress import IPv4Address
from random import randint
import time

def creaIp():
    ip = IPv4Address('{0}.{1}.{2}.{3}'.format(randint(0,255),randint(0,255),randint(0,255),randint(0,255)))

    if(ip.is_unspecified or ip.is_loopback or ip.is_multicast  or ip.is_reserved or ip.is_private or ip.is_link_local):
        return creaIp()

    return ip

obj = pygeoip.GeoIP('GeoLiteCity.dat')

cont=0
listaIP=[]

print("Listado de 10 IPs generadas al azar geolocalizadas en España")
print("Hora de Inicio: ", time.strftime("%I:%M:%S"))
while cont < 10:
    ip=creaIp()
    if obj.country_code_by_addr(str(ip))=="ES":
        listaIP.append(str(ip))
        cont=cont+1
print ("------------------------------------------------------------")
for a in range(0,len(listaIP)):
    print(a+1, " ", str(listaIP[a]))
print ("------------------------------------------------------------")
print("Hora final: ", time.strftime("%I:%M:%S"))
```

Figura 25 Código para generar 10 IPs geolocalizadas en España

Como se puede observar, el resultado es bastante satisfactorio, y mejor será cuando se realice en un ordenador con más potencia.

```
Listado de 10 IPs generadas al azar geolocalizadas en España
Hora de Inicio:  12:04:23
------------------------------------------------------------
1    82.86.73.134
2    85.50.223.195
3    77.224.122.29
4    83.62.177.187
5    46.26.225.17
6    213.98.82.110
7    80.34.17.63
8    77.208.21.91
9    79.145.240.120
10    88.31.105.42
------------------------------------------------------------
Hora final:  12:04:25
```

Figura 26 Resultado de la ejecución del script

Así mismo, se puede comprobar que todas las IPs están correctamente geolocalizadas. Se ha realizado a través de la página https://www.cual-es-mi-ip.net/geolocalizar-ip-mapa.

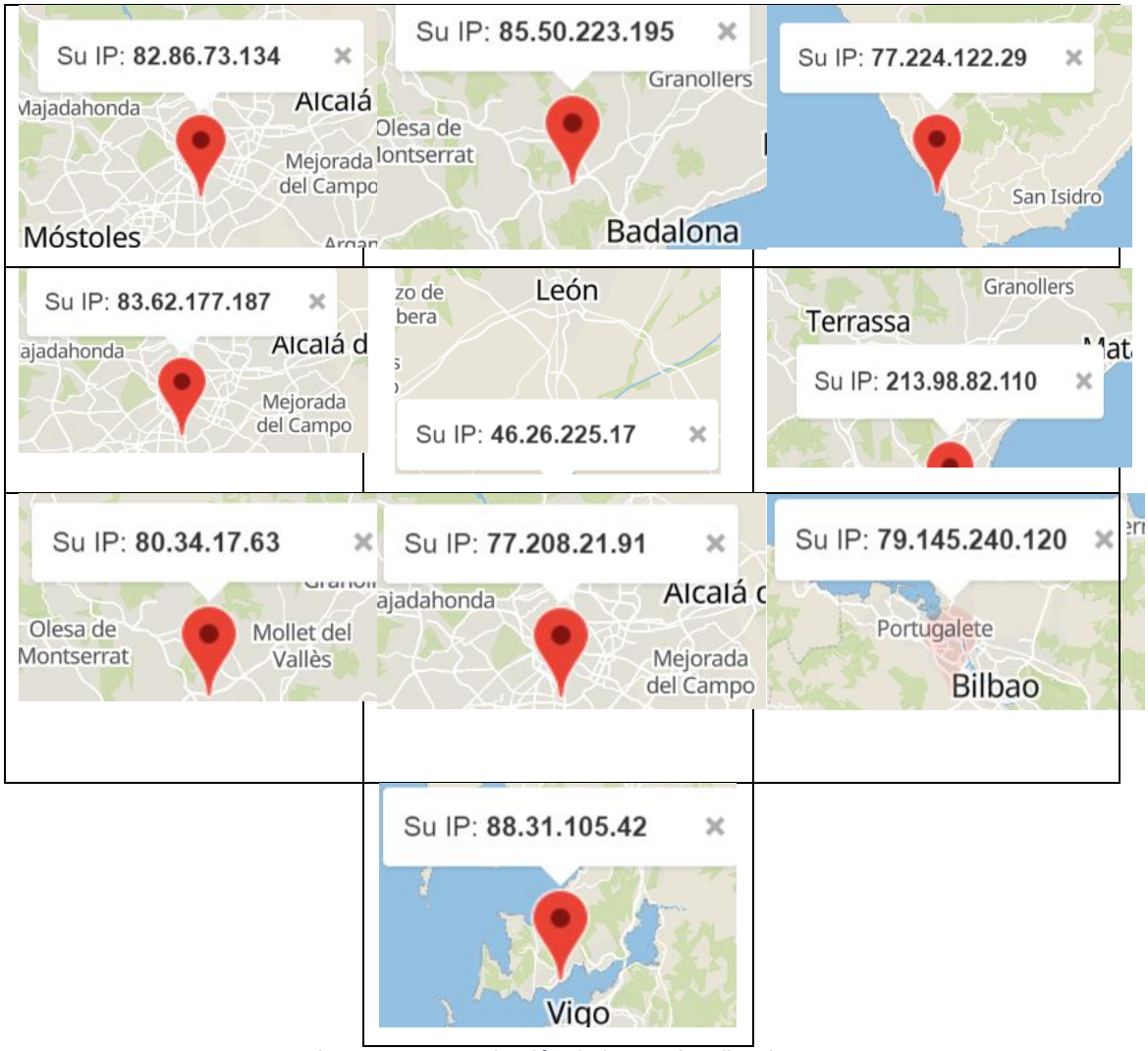

Figura 27 Comprobación de las geolocalizaciones

Lectura de *Banners* y uso de *Sockets*

Como se ha visto, mediante los *banners*, algunos servidores exponen la versión, el nombre del servidor e incluso la tecnología backend que usa. Mediante las librerías de Python, se puede capturar la información del *banner*. La forma más extendida de hacerlo es mediante el uso del módulo socket, como se muestra en el código siguiente:

```
#!/usr/bin/python3
# Ejemplo de lectura de banner de la página de la UOC mediante GET

import socket
import optparse
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(("213.73.40.210", 80))
sock.settimeout(2)
http_get = b"GET / HTTP/1.1\nHost: 213.73.40.210\n\n"
data = ''

try:
    sock.sendall(http_get)
    data = sock.recvfrom(1024)
    aux = str(data).replace('\\r\\n','<br/>')
    #formateamos los datos y quitamos los dos primeros caracteres (b y el último )
    data = aux[2:len(aux)-1]
    print ("Datos totales obtenidos")
    print (data)

except socket.error:
    print ("Error de Socket: ", socket.errno)
finally:
    print("Cerrando socket")
    sock.close()
```

Figura 28 Script lectura banner de la UOC

Con esta sencillo *script*, se ha realizado un socket a la IP de la UOC en el puerto 80 para obtener la información *banner* que ofrece. Como se puede apreciar en la siguiente lectura, hay una serie de datos que resultan interesantes, tales como el servidor o el autor.

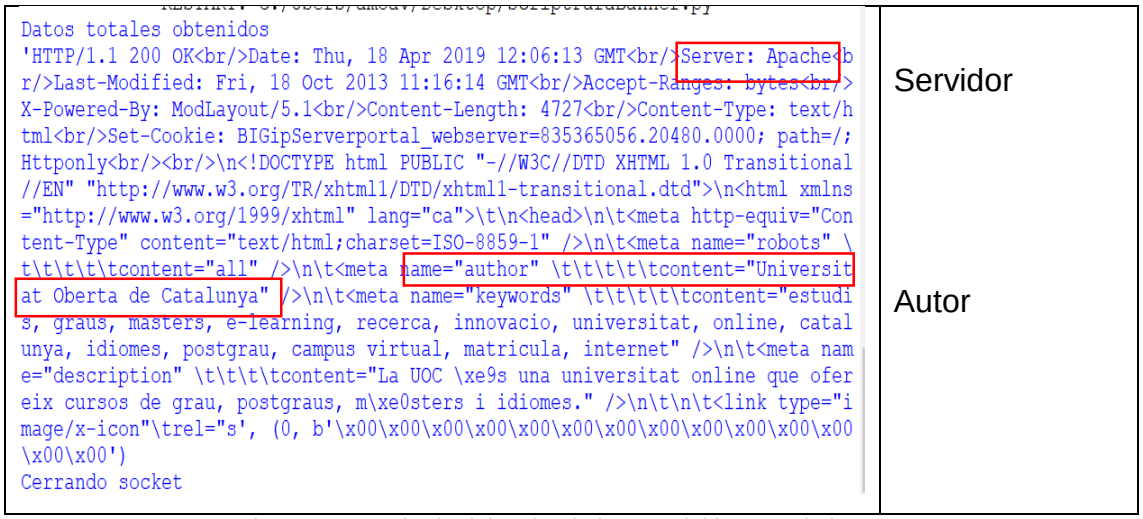

Figura 29 Resultado del script de lectura del banner de la UOC

También es posible el uso de http.client para obtener esta información. En el siguiente ejemplo vemos como se obtiene la información “Server” de la dirección de la UOC del *banner*.

```
import http.client

host="213.73.40.210"
http =http.client.HTTPConnection(host, 80, timeout = 2)
http.request("HEAD", "/")
server = http.getresponse().getheader('server')
print ("Resultado: ", server)
```

Figura 30 Obteniendo parámetros concretos del banner

```
========== RESTART:
Resultado:  Apache
```

Figura 31 Resultado de la búsqueda server del banner

Con una buena base de datos en la que se dispusiesen de las vulnerabilidades conocidas, se podría informar de la misma mediante una simple búsqueda, aunque eso se sitúa fuera del alcance de este proyecto.

Existen otros modos de obtener información, como puede ser mediante whois, BeautifulSoap y un larguísimo etcétera que nos brinda Python desde su generosa colección de librerías. El autor de este trabajo invita al lector a investigar sobre todos estos métodos y comprobar la gran cantidad de posibilidades que existen.

Datos a almacenar

A continuación, se determinarán qué datos serán almacenados y en qué formato. Debido al carácter teórico-práctico del proyecto y desde un punto de vista realista, no se va a manejar una gran cantidad de información. En un proyecto completo de buscador IoT, se podrían barajar opciones muy válidas para el tratamiento de cantidades de datos importantes, con bases de datos NoSQL como mongoDB. Python ofrece esta opción con el driver PyMongo[36].

Sin embargo, el método elegido será un sistema de bases de datos relacional llamado SQLite, creado por Dwayne Richard Hipp [37] y que ofrece un gestor de bases de datos liviano a la vez que robusto.
Los datos que se almacenarán serán los siguientes:

- IP
- *Banner*
- Captura de pantalla (si procede)
- Puertos abiertos
- Datos de DNS
- Datos de Whois

Presentación de datos en pantalla

Para la presentación de los datos, se usará un *framework* de python llamado Flasx. Este *framework* es minimalista y nos permite crear la presentación de datos que necesitamos en un tiempo record con unas mínimas líneas de código. Para mejorar el aspecto se usará BootStrap, consiguiendo resultados muy decentes, como se muestra en la siguiente captura. Éste se ejecutará en el puerto 8002.

Como se puede observar, el contenido Web comienza con una simple página de presentación, que hace referencia a las Universidades que conforman este Máster.

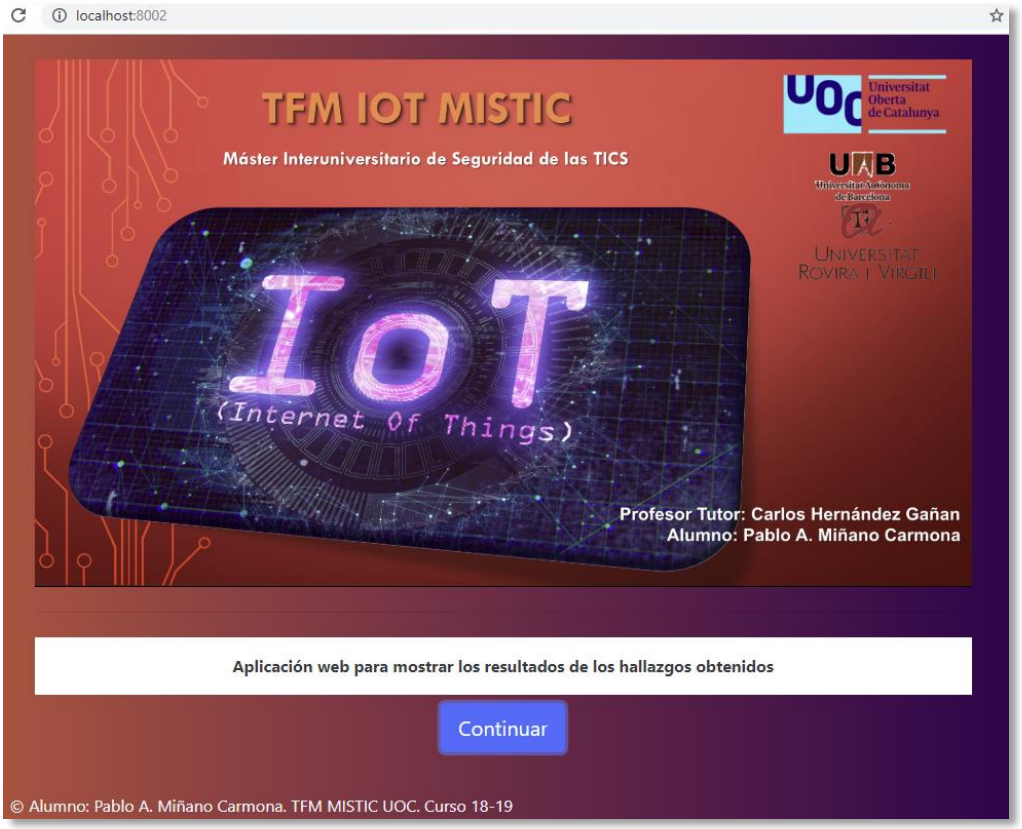

Figura 32 Página de presentación del proyecto

Esquema de nuestro proyecto

El proyecto se basará en dos *scripts* principales. El primero será el proceso automático con el que se descubrirán las direcciones IP. Este *script* se ejecuta de un modo muy sencillo. Se realiza una llamada al script con un posible valor numérico. Si el valor numérico es omitido o superior a 1000, el valor se fija en 1000. Este valor numérico será el número límite de direcciones IPv4 que se crearán en el entorno geográfico de España.

Cuando una dirección es creada, primero se comprobará que no exista en la base de datos. Si existiese, se comprobaría la fecha de inserción y sólo en caso de que se haya guardado con una antigüedad posterior a 30 días, se actualizarán los datos. En caso contrario, se desechará la dirección IP y se generará una nueva.

Una vez la IP ha sido generada, se obtendrá una serie de información sobre la dirección, y según el caso del puerto que esté abierto, se navegará con Mozilla a esa dirección y ese puerto, para posteriormente obtener una captura, cuyo

nombre será almacenado en la BBDD, y la imagen en el directorio capturas. Los hallazgos son mostrados por consola, para que se vaya viendo el progreso del script, con un listado de puertos abiertos al final del recorrido de cada IP, como se observa en la siguiente captura de ejemplo.

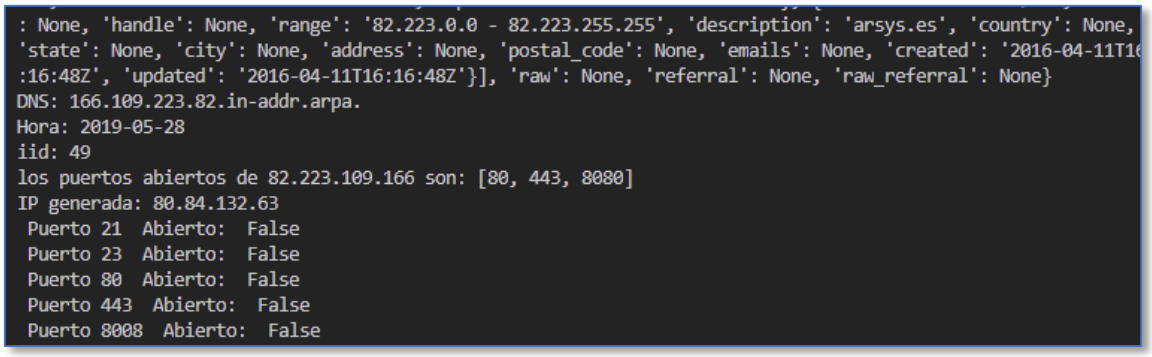

Figura 33 Script aplicacion.py en acción

Evidentemente, es un proceso lento, sobre todo si es ejecutado desde una única máquina. Es por ello, que para poder conseguir una serie de datos suficientemente significativa, se ha limitado mucho los puertos a escanear, siendo la lista usada la siguiente: 21, 23, 80, 443, 8008, 8080, 8081y 9443.

Para poder comprobar puertos adicionales, solo hay que añadirlos en la parte del código correspondiente, en la lista listado_puertos, como indica la captura.

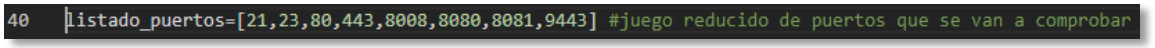

Figura 34 Lista de puertos para comprobar

A continuación se presenta el diagrama del algoritmo creado.

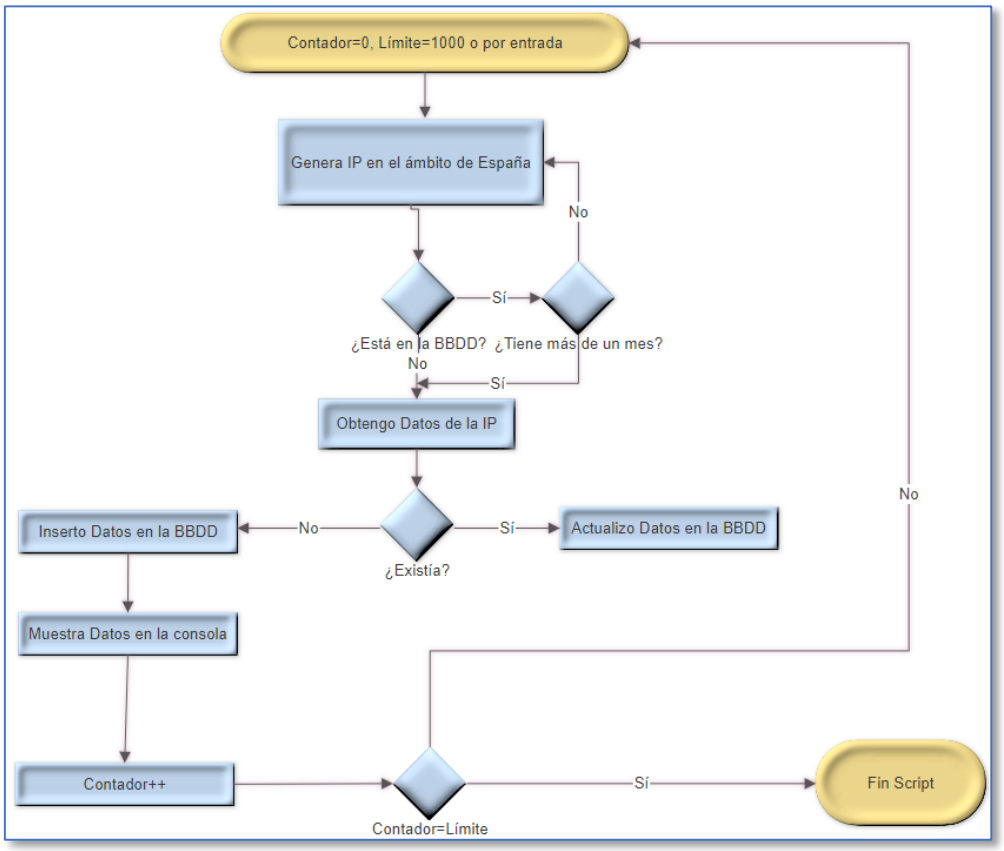

Figura 35 Esquema del script principal

El segundo *script* será el que active el *framework* de flask para que se puedan consultar los datos vía-web.

Figura 36 Opciones del Programa

Para realizar esta consulta se han habilitado dos posibilidades. La primera consiste en introducir una dirección IPv4 y si estuviese guardada en la BBDD se mostraría el contenido de los hallazgos encontrados referente a esa dirección. En la segunda opción, se muestra un catálogo completo de direcciones IPv4 encontradas, los puertos que se han descubierto abiertos y la fecha de la consulta.

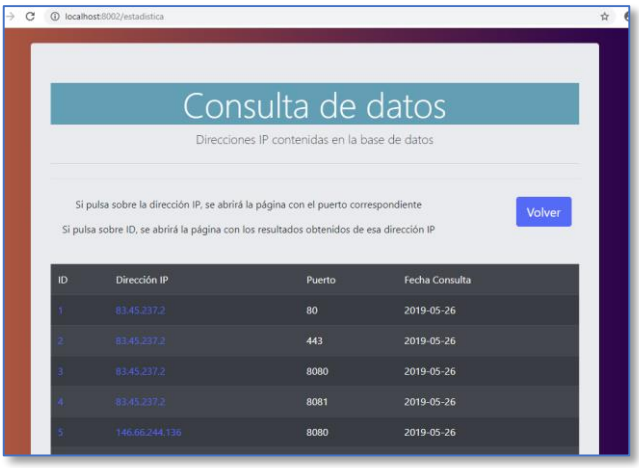

Figura 37 Listado Completo

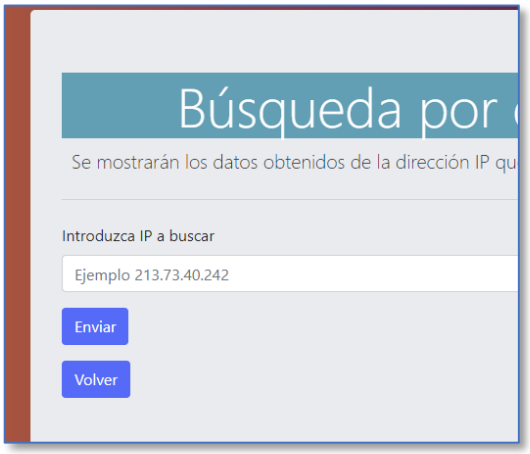

Figura 38 Búsqueda por IP

Por último se mostrará un par de capturas de hallazgos encontrados por el *script.*

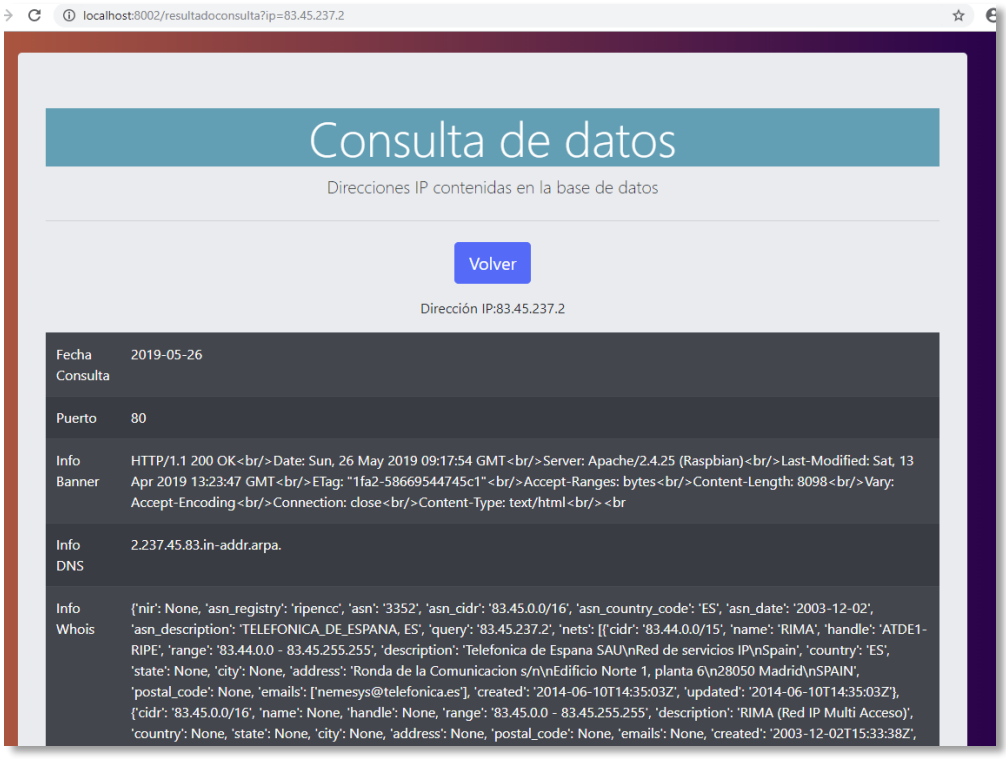

Figura 39 Ejemplo de hallazgos encontrados(1)

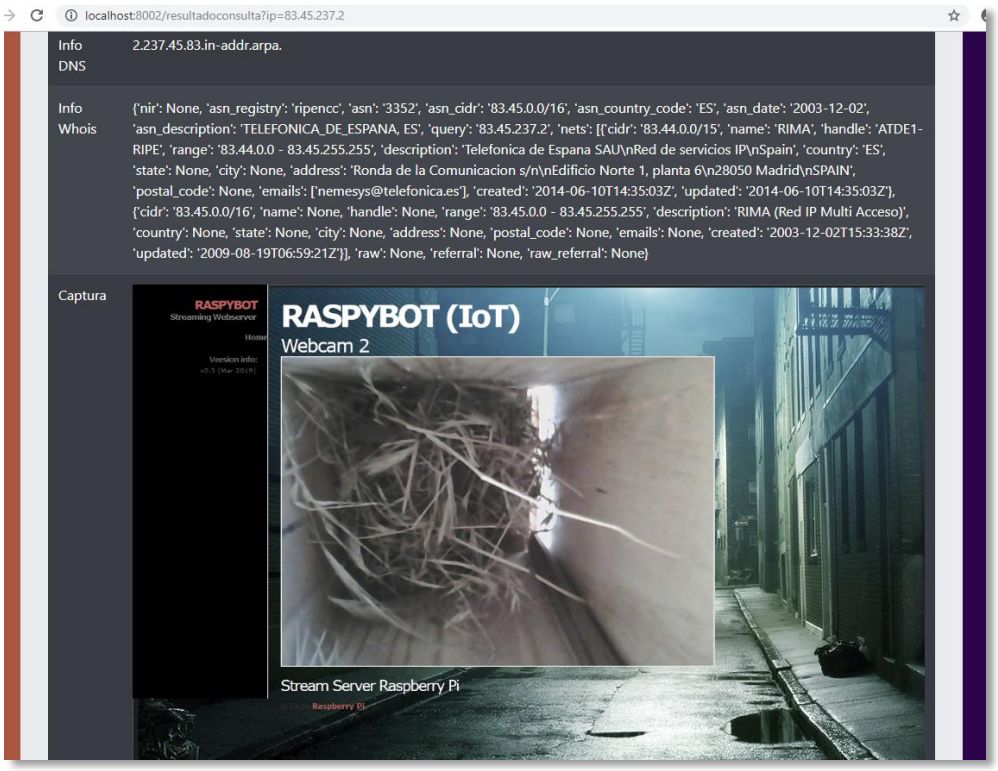

Figura 40 Ejemplo de hallazgos encontrados (2)

Se puede apreciar lo que viene a ser una webcam, donde se muestran unos nidos de aves.

5. Conclusiones

En mi primera aproximación “seria” al fascinante mundo de la IoT, mis impresiones finales son de una gran satisfacción. Desconocía por completo este impresionante “Nuevo Mundo” y aunque reconozco mis reticencias iniciales, no sólo por la falta de conocimientos en este campo, sino porque nunca había empleado el lenguaje Python a este nivel, el haber podido aprenderlo, son la mejor manera de recompensa que se puede obtener después del esfuerzo.

Ha sido un trabajo extenso, con muchas horas de investigación y aprendizaje. Como resultado del mismo, se han podido abarcar los objetivos marcados de manera satisfactoria. Con una parte teórica que incluye todos los aspectos iniciales planteados, tales como la seguridad de la IoT, los ecosistemas y sus amenazas, así como un estudio sobre los motores de búsqueda existentes más conocidos de este campo. Igualmente, se ha realizado una parte práctica que ha dado como resultado, un *software* que realiza unas funciones de búsqueda básicas, inspiradas en las que Shodan implementa.

La planificación inicial ha sido adecuada y muy orientativa para la realización de este proyecto. También destacar las aportaciones de mi tutor, Carlos Hernández Gañán, cuyas indicaciones y conocimiento en la materia han sido clave para poder concluirlo de manera satisfactoria.

Como futuras actuaciones sobre esta base, se podría, además de optimizar el código, realizar funcionalidades extras, como la de una automatización que contuviese una base de datos con las vulnerabilidades conocidas y cuando detectase un sistema en riesgo, crease algún tipo de alerta, como pudiera ser el envío de un correo electrónico, pudiendo contribuir de este modo a una Internet más segura.

Sea como fuere, no quisiera desperdiciar esta última oportunidad de agradecer a la gran familia UOC en general y a mi tutor en particular, por esta nueva oportunidad de la que he podido disfrutar, adquiriendo nuevos conocimientos y realizándome un poco más en este campo que tanto me satisface. Sin olvidar por supuesto a mi familia y amistades, de la que tantas horas me he privado de su tan grata compañía y han empleado tremendas dosis de paciencia y cariño conmigo.

Muchísimas gracias de corazón.

Recibid un muy cordial saludo.
Pablo Miñano

6. Glosario

Banner: En este contexto, hace referencia a la información proporcionada cuando realizamos una lectura de un puerto.

Botnet: Es un grupo de dispositivos conectados a una red, que han sido infectados por algún tipo de malware para ser controlados de forma remota.

Cross Site Scripting: Vulnerabilidad informática, conocida como XSS, mediante la cual, en ciertas páginas o aplicaciones web se puede inyectar y ejecutar código de ciertos lenguajes como Javascript, consiguiente efectos no programados por los creadores de estas páginas.

DOS: Responde al acrónimo *Denial of Service*. Consiste en un tipo de ataque, que como su nombre indica, se produce una denegación de servicio, consiguiendo que el servicio afectado no pueda responder a las peticiones lícitas que se produce (p.e. una caída de una página web)

DDOS: Acrónimo que responde a un ataque de denegación de servicio de forma distribuida. Es decir, se deniegan los servicios como en la definición anterior, pero mediante un ataque masivo desde distintos puntos, al mismo tiempo, para que el ataque sea más eficiente y no pueda ser evitado.

Fingerprinting: Mediante este proceso, se puede recopilar información de un sistema, de tal modo que se pueda identificarlo ante otros, por ejemplo, a través del navegador que estamos usando, que en este caso se suele denominar *browser fingerprinting*, lo que viene a ser algo así como la “huella digital” que deja el navegador tras el paso de los sitios webs.

Footprinting: Mediante esta técnica legal, se intenta obtener la máxima información posible sobre un objetivo (sistema, usuario, etc.). Esta información

deber ser pública, de tal modo que se pueda obtener, a través de los metadatos de los documentos, de las redes sociales, con buscadores, etc.

Framework: Dentro de la programación, se podría definir como una especie de marco de trabajo, donde están estandarizados ciertos conceptos y prácticas, en base a una serie de criterios, de tal modo que cuando se deba resolver el mismo problema, ya existe una referencia anterior creada.

Malware: Abreviatura de MALicious softWARE y engloba a cualquier programa cuyos objetivos y/o intenciones sean causar algún tipo de malfuncionamiento o daño a un sistema.

OSINT: Acrónimo de *Open Source Intelligence*, es decir, Inteligencia de Fuentes Abiertas. Consiste en una recopilación y análisis de información que debe ser obtenida de fuentes públicas y abiertas para que sean consideradas OSINT.

Sockets: Es una interfaz I/O(entrada y salida) que se crea entre dos sistemas en red, mediante una dirección IP y un número de puerto.

SQL Injection: Tipo de ataque a una base de datos, gracias al descuido por parte de los programadores de no comprobar que en los campos de la consulta van códigos junto a los datos.

7. Bibliografía

[1] Seguridad Global. ComputerWorld, 25 de Junio de 2018. http://seguridadti.cso.computerworld.es/tendencias/el-numero-de-dispositivos-iot-crecera-un-140-en-los-proximos-cuatro-anos. [Visitado el: 4 de Marzo de 2019.]

[2] Partarrieu, Catherine. Wide Defense. https://www.widefense.com/el-ataque-ddos-mas-grande-registrado-que-afecto-a-sitios-como-twitter-spotify-netflix-github-y-amazon/ [Visitado el 24 de Octubre de 2016]

[3] https://soniadurolimia.com/25-ejemplos-internet-de-las-cosas-te-dejaran-la-boca-abierta/ [Visitado el 12 de Marzo]

[4] http://dej.rae.es/lema/internet-de-las-cosas [Visitado el 12 de Marzo]

[5] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," Future Generation Computer Systems-the International Journal of Grid Computing and Escience, vol. 29, pp. 1645-1660, Sep. 2013, 2013. [Citado el 13 de Marzo]

[6] https://www.kaspersky.es/about/press-releases/2018_cuidado-con-los-regalos- envenenados [Visitado el 13 de Marzo]

[7] https://www.businessinsider.com/there-will-be-34-billion-iot-devices-installed-on-earth-by-2020-2016-5?IR=T [Visitado el 15 de Marzo]

[8] https://www.reuters.com/article/security-cyber-iran/update-2-cyber-attack-appears-to-target-iran-tech-firms-idUSLDE68N1OI20100924 [Visitado el 15 de Marzo]

[9] https://www.kaspersky.es/blog/que-es-un-botnet/755/ [Visitado el 16 de Marzo]

[10] https://es.wikipedia.org/wiki/Ciberataque_a_Dyn_de_octubre_de_2016 [Visitado el 16 de Marzo]

[11] https://www.infobae.com/america/mundo/2016/10/21/hackers-rusos-y-chinos-se-adjudicaron-el-mayor-ciberataque-de-la-ultima-decada/ [Visitado el 17 de Marzo]

[12] https://blog.ehcgroup.io/index.php/2018/12/10/7-nuevas-variantes-del-malware-mirai/ [Visitado el 17 de Marzo]

[13] https://www.ituser.es/seguridad/2018/10/confirmado-los-ataques-contra-dispositivos-iot-baten-un-nuevo-record-en-2018 [Visitado el 18 de Marzo]

[14] https://www.f5.com/labs/articles/threat-intelligence/the-hunt-for-iot--multi-purpose-attack-thingbots-threaten-intern [Visitado el 19 de Marzo]

[15] https://www.gartner.com/en/newsroom/press-releases/2018-03-21-gartner-says-worldwide-iot-security-spending-will-reach-1-point-5-billion-in-2018 [Visitado el 20 de Marzo]

[16] https://www.ibm.com/developerworks/ssa/library/seguridad_y_el_ecosistema_de_iot/ [Visitado el 21 de Marzo]

[17] https://eprint.iacr.org/2017/511.pdf [Visitado el 23 de Marzo]

[18] Leila Fatmasari Rahman et al. / Procedia Computer Science. [Visitado el 23 de Marzo de 2019] https://www.sciencedirect.com/science/article/pii/S1877050918305106

[19] https://www.ibm.com/developerworks/ssa/library/iot-anatomy-iot-malware-attack/index.html [Visitado el 23 de Marzo]

[20] https://internetofthingsagenda.techtarget.com/definition/IoT-attack-surface [Visitado el 24 de Marzo]

[21] https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project#IoT_Attack_Surface_Ar eas_Project [Visitado el 24 de Marzo]

[22] https://searchsecurity.techtarget.com/feature/Command-and-control-servers-The-puppet-masters-that-govern-malware [Visitado el 27 de Marzo]

[23] https://www.csoonline.com/article/3126924/here-are-the-61-passwords-that-powered-the-mirai-iot-botnet.html [Visitado el 31 de Marzo]

[24] https://es.wikipedia.org/wiki/Pwned [Visitado el 31 de Marzo]

[25] https://www.redeszone.net/2019/01/26/alternativas-shodan-hacking-etico-2019 [Visitado el 1 de Abril]

[26] https://developer.shodan.io/ [Visitado el 1 de Abril]

[27] Complete Guide to Shodan, John Matherly, 2016

[28] https://www.lasexta.com/tecnologia-tecnoxplora/internet/ciudad-con-ley/vulnerabilidad-deja-conexion-internet-casi-millon-routers-alemania_20161129583d7d3c0cf267e7fb95f5c3.html [Visitado el 4 de Abril]

[29] https://papelesdeinteligencia.com/que-son-fuentes-de-informacion-osint/ [Visitado el 5 de Abril]

[30] https://www.yolandacorral.com/que-es-osint-fases-fuentes-herramientas/ [Visitado el 6 de Abril]

[31] https://fluidattacks.com/web/es/defends/apache/restringir-banner/ [Visitado el 7 de Abril]

[32] https://es.wikipedia.org/wiki/Python [Visitado el 10 de Abril]

[33] https://es.wikipedia.org/wiki/Guido_van_Rossum [Visitado el 11 de Abril]

[34] Hacking ético con herramientas Python. Ed. Ra-ma. José Manuel Ortega Candel

[35] http://www.fundacionsadosky.org.ar/avt/glossary/prueba-de-concepto-poc-por-sus-siglas-en-ingles/ [Visitado el 14 de Abril]

[36] https://www.w3schools.com/python/python_mongodb_getstarted.asp [Visitado el 15 de Abril]

[37] https://en.wikipedia.org/wiki/D._Richard_Hipp [Visitado el 19 de Abril]

8. Anexos

8.1 Anexo 1

Listado de puertos más comunes según Shodan.

Puerto	Servicio		Puerto	Servicio		Puerto	Servicio
7	Echo		123	NTP		873	rsync
11	Systat		129	Password generator protocol		902	VMWare authentication
13	Daytime		137	NetBIOS		992	Telnet (secure)
15	Netstat		143	IMAP		993	IMAP with SSL
17	Quote of the day		161	SNMP		995	POP3 with SSL
19	Character generator		175	IBM Network Job Entry		1010	malware
21	FTP		179	BGP		1023	Telnet
22	SSH		195	TA14-353a		1025	Kamstrup
23	Telnet		311	OS X Server Manager		1099	Java RMI
25	SMTP		389	LDAP		1177	malware
26	SSH		443	HTTPS		1200	Codesys
37	rdate		444	TA14-353a, Dell SonicWALL		1234	udpxy
49	TACACS+		445	SMB		1434	MS-SQL monitor
53	DNS		465	SMTPS		1521	Oracle TNS
67	DHCP		500	IKE (VPN)		1604	Citrix, malware
69	TFTP, BitTorrent		502	Modbus		1723	PPTP
70	Gopher		503	Modbus		1741	CiscoWorks
79	Finger		515	Line Printer Daemon		1833	MQTT
80	HTTP, malware		520	RIP		1900	UPnP
81	HTTP, malware		523	IBM DB2		1911	Niagara Fox
82	HTTP, malware		554	RTSP		1962	PCworx
83	HTTP		587	SMTP mail submission		1991	malware
84	HTTP		623	IPMI		2000	iKettle, MikroTik bandwidth test
88	Kerberos		626	OS X serialnumbered		2082	cPanel
102	Siemens S7		636	LDAPS		2083	cPanel
110	POP3		666	Telnet		2086	WHM
111	Portmapper		771	Realport		2087	WHM
119	NNTP		789	Redlion Crimson3		2123	GTPv1

Figura 41 Listado de puertos comunes (1)

Puerto	Servicio		Puerto	Servicio		Puerto	Servicio
2152	GTPv1		4911	Niagara Fox with SSL		8090	Insteon HUB
2181	Apache Zookeeper		4949	Munin		8099	Yahoo SmartTV
2222	SSH, PLC5, EtherNet/IP		5006	MELSEC-Q		8112	Deluge (HTTP)
2323	Telnet		5007	MELSEC-Q		8139	Puppet agent
2332	Sierra wireless (Telnet)		5008	NetMobility		8140	Puppet master
2375	Docker		5009	Apple Airport Administration		8181	GlassFish Server (HTTPS)
2376	Docker		5060	SIP		8333	Bitcoin
2404	IEC-104		5094	HART-IP		8334	Bitcoin node dashboard (HTTP)
2455	CoDeSys		5222	XMPP		8443	HTTPS
2480	OrientDB			5269 XMPP Server-to-Server		8554	RTSP
2628	Dictionary		5353	mDNS		8880	Websphere SOAP
3000	ntop		5357	Microsoft-HTTPAPI/2.0		8888	HTTP, Andromouse
3306	MySQL		5432	PostgreSQL		8889	SmartThings Remote Access
3310	ClamAV		5577	Flux LED		9001	Tor OR
3386	GTPv1		5632	PCAnywhere		9002	Tor OR
3388	RDP		5672	RabbitMQ		9051	Tor Control
3389	RDP		5900	VNC		9100	Printer Job Language
3460	malware		5901	VNC		9151	Tor Control
3541	PBX GUI		5938	TeamViewer		9160	Apache Cassandra
3542	PBX GUI		5984	CouchDB		9191	Sierra wireless (HTTP)
3689	DACP		6000	X11		9418	Git
3780	Metasploit		6379	Redis		9443	Sierra wireless (HTTPS)
3787	Ventrilo		6666	Voldemort database, malware		9595	LANDesk Management Agent
4000	malware		6667	IRC		9600	OMRON
4022	udpxy		6881	BitTorrent DHT		9869	OpenNebula
4040	Deprecated Chef web interface		6969	TFTP, BitTorrent		9009	Julia
4063	ZeroC Glacier2		7218	Sierra wireless (Telnet)		10001	Automated Tank Gauge
4064	ZeroC Glacier2 with SSL		7474	Neo4j database		10243	Microsoft-HTTPAPI/2.0
4369	EPMD		7548	CWMP (HTTPS)		11211	Memcache
4443	Symantec Data Center Security		7777	Oracle		17000	Bose SoundTouch
4444	malware		7779	Dell Service Tag API		17185	VxWorks WDBRPC
4500	IKE NAT-T (VPN)		8010	Intelbras DVR		12345	Sierra wireless (Telnet)
4567	Modem web interface		8060	Roku web interface		13579	Media player classic web inter.
4070	HID VertX/ Edge door controller		8069	OpenERP		14147	Filezilla FTP
4800	Moxa Nport		8087	Riak		16010	Apache Hbase

Figura 42 Listado de puertos comunes (2)

Puerto	Servicio		Puerto	Servicio		Puerto	Servicio
18245	General Electric SRTP		27017	MongoDB		50070	HDFS Namenode
20000	DNP3		28017	MongoDB (HTTP)		51106	Deluge (HTTP)
20547	ProconOS		30313	Gardasoft Lighting		53413	Netis backdoor
21025	Starbound		30718	Lantronix Setup		54138	Toshiba PoS
21379	Matrikon OPC		32400	Plex		55553	Metasploit
23023	Telnet		37777	Dahuva DVR		55554	Metasploit
23424	Serviio		44818	EtherNet/IP		62078	Apple iDevice
25105	Insteon Hub		47808	Bacnet		64738	Mumble
25565	Minecraft		49152	Supermicro (HTTP)			
27015	Steam A2S server query, Steam RCon		49153	WeMo Link			

Figura 43 Listado de puertos comunes (3)

8.1 Anexo 2

Requerimientos de los *scripts*: Se ha usado la última versión de Python en el momento del desarrollo de los códigos, que ha sido la 3.7.3.

El *script* buscador de hallazgos en las direcciones IP, llamada aplicación.py, hace uso de las siguientes librerías: sys, pygeoip, pprint, ipaddress, random, time, socket, selenium, ipwhois, dns, sqlite3, os y datetime y para la versión de Windows, geckodriver.exe incluido en el directorio raíz, para las capturas del navegador.

El *script* appweb, que contiene el código para la presentación web de los resultados, hace uso de las siguientes librerías y *scripts* adicionales: flask, base64, os y sqlite3.

La estructura de directorios es la siguiente:

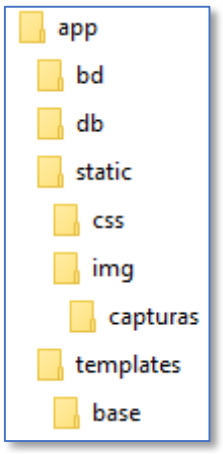

Figura 44 Estructura de directorios

- app es el directorio en el que podemos encontrar los *scripts* de aplicación.py y appweb.p
- bd contiene la base de datos, llamada datos.db
- db es el directorio que contiene la base de datos de la localización geográfica de direcciones IP, llamado GeoLiteCity.dat
- static es el directorio que contiene la hoja de estilo que se usa, además de una carpeta img con la imagen de presentación, y una carpeta capturas, con todas las capturas de los hallazgos encontrados.
- templates: contiene los html que conforman la web para mostrar los resultados.