Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Seguridad Informática

Aplicación para la comparación de Shodan y ZoomEye en la búsqueda de dispositivos IoT

Trabajo fin de estudio presentado por:	Jon Legarda González Gonzalo Ochoa de Eguileor López de Maturana
Tipo de trabajo:	*Piloto experimental*
Director/a:	Rafael Alejandro Rodríguez Gómez
Fecha:	9 de febrero de 2022

Resumen

La cantidad de dispositivos conectados a Internet aumenta de forma vertiginosa. Este hecho aporta muchos beneficios para las personas, pero también conlleva riesgos, dado que nos hace cada vez más dependientes de servicios y aplicaciones que pueden estar expuestos a muy variadas amenazas. Existen plataformas destinadas a inventariar los dispositivos conectados y poner esta información a disposición de cualquier usuario. En este trabajo se ha realizado una aplicación en forma de prototipo que permite consultar dos de las más utilizadas a nivel mundial, Shodan y ZoomEye. El objetivo principal de esta herramienta es facilitar la investigación sobre el uso de estas plataformas, ya que permite obtener datos, comparar los resultados que devuelven, sacar información histórica e incluso validar de forma pasiva la eficacia de los resultados obtenidos con la herramienta de *fingerprinting* Nmap. Por último, se presentan algunos escenarios para el uso práctico de la aplicación y algunas líneas de trabajo futuro.

Palabras clave: Shodan, ZoomEye, IoT, *footprinting*, ciberseguridad

Abstract

The number of devices connected to the Internet is increasing rapidly. This may bring many benefits for individuals, but it also entails risks, since it makes us increasingly dependent on services and applications that can be exposed to a wide variety of threats. There are platforms dedicated to inventorying connected devices and making this information available to any user. Throughout this work, we have developed a software application that makes it easy to query two of the most worldwide used IoT search engines: Shodan and ZoomEye. The main objective of this tool is to facilitate research on the use of these platforms, since it allows obtaining data, comparing the results returned, extracting historical information, and even validating it with the fingerprinting tool Nmap. Finally, some scenarios for the practical use of the application and some lines of future work are presented.

Keywords: Shodan, ZoomEye, IoT, footprinting, cybersecurity

Índice de contenidos

Índice de figuras

Índice de tablas

1. Organización del trabajo en grupo

El trabajo expuesto en el presente documento ha sido elaborado por parte de Jon Legarda González y Gonzalo Ochoa de Eguileor López de Maturana.

Para la consecución del trabajo ambos autores han participado en todas las fases del mismo, haciendo uso de una metodología *Scrum* con el objetivo de estar coordinados en durante todo el tiempo del proyecto. Para ello, se han utilizado mecanismos de videollamada y reuniones virtuales de forma semanal.

Con el fin de alcanzar los hitos principales del trabajo, las tareas a realizar se han dividido en subtareas más simples que se han desarrollado alternativamente por los coautores, de tal forma que ambos han sido conscientes en todo momento de la situación del proyecto y han realizado aportes de forma equilibrada en cada fase.

En el apartado *4 Objetivos concretos y metodología de trabajo* se explican los objetivos principales, específicos y adicionales, además del detalle de la planificación y metodología utilizadas.

2. Introducción

2.1.INTERNET OF THINGS

La "Internet de las Cosas" o "*Internet of Things*" es uno de los conceptos tecnológicos que está revolucionando el mundo. En el ámbito de las tecnologías de la información es un aspecto que está al alza y que lo seguirá estando a futuro (Hassija et al., 2019).

2.1.1. Definición

Antes de intentar explicar un concepto como el *Internet of Things* es preciso aclarar que no hay establecida una definición estándar en la industria. Hasta la fecha muchos académicos, investigadores, desarrolladores e intelectuales han definido con sus palabras el término en función de su propia perspectiva.

Todas las definiciones del concepto parecen converger en la idea de que la primera versión de Internet se basa en datos creados por personas, mientras que la siguiente versión de Internet se basará en datos creados por las cosas (Madakam et al., 2015).

A fin de cuentas, la Internet de las Cosas se puede definir como la interconexión entre diferentes dispositivos, sensores y elementos cotidianos de Internet que proveen al objeto y su entorno de mayor capacidad y habilidad de computación.

2.1.2. Historia y evolución

El término IoT fue acuñado en primera instancia por Kevin Ashton, Director Ejecutivo de Auto-ID Center en el Instituto de Tecnología de Massachussets (Madakam et al., 2015). Antes de ello, ya en la década de los 80, se produjo la primera aplicación del uso de Internet en un elemento de uso habitual: una máquina de refrescos de cola. Un equipo de programadores de la Universidad de Carnegie Melon logró conectar la máquina de refrescos a Internet y así verificar su estado y determinar si había disponibilidad de bebidas.

Años más tarde, tras el bautizo de la palabra por parte de Kevin Ashton, en 2003, se popularizó el término "IoT" gracias a la propia empresa Auto-ID Center y a diferentes publicaciones de análisis de mercado que ponían este concepto encima de la mesa.

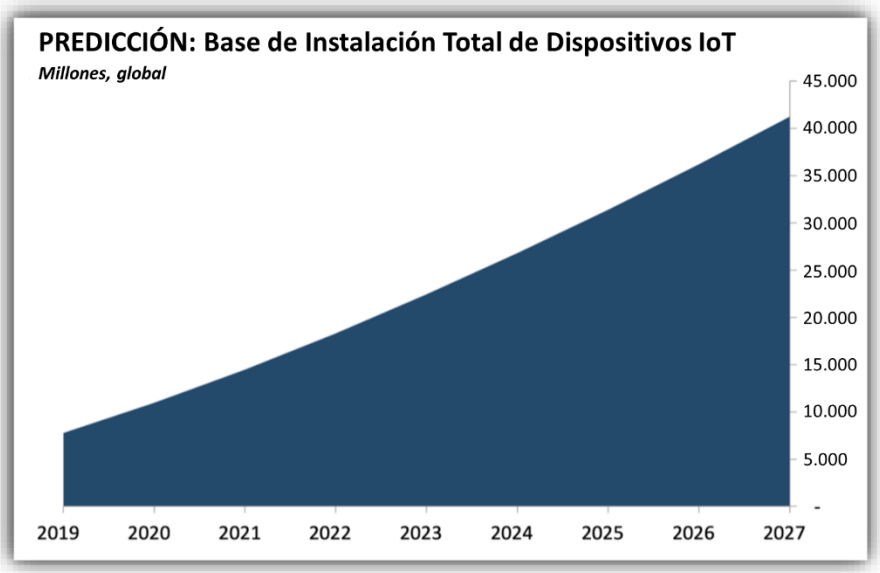
En 2005 la Unión Internacional de Telecomunicaciones de las Naciones Unidas (*International Telecommunications Union*, ITU) publicó su primer informe relacionado con el IoT llamado

"*ITU Internet Reports 2005: The Internet of Things*". En él, la ITU abordaba el tema desde una visión muy amplia y holística describiendo el concepto, analizando las oportunidades de mercado, listando posibles desafíos a los que se expondría en un futuro e, incluso, se atrevía a describir cómo sería un día normal de un habitante europeo en el año 2020 gracias a este tipo de dispositivos.

Tres años después del informe de la ITU, en el año 2008, el Internet de las Cosas logra reconocimiento por parte de la Unión Europea y se lleva a cabo la primera conferencia europea sobre este aspecto. Ese mismo año, un grupo de empresas lanza la IPSO Alliance para promover el uso de IP (*Internet Protocol*) en redes de objetos inteligentes y así facilitar el uso del Internet de las Cosas (Madakam et al., 2015).

A partir de esos años el uso de Internet por parte de diferentes tipos de dispositivos solo ha ido en aumento. Según un informe de la revista *Business Insider* publicado a mediados de 2020, para el año 2027 se estiman más de 41 billones de dispositivos conectados a Internet (Newman, s. f.). El incremento es relevante teniendo en cuenta que el mismo informe estima que actualmente existen alrededor de 15 miles de millones de dispositivos conectados a Internet (ver Figura 1).

Figura 1: Estimación dispositivos IoT conectados 2019 - 2027



Fuente: Elaboración propia basada en el informe de *Business Insider Intelligence*

2.1.3. Desafíos actuales y futuro

Los aspectos positivos que trae consigo este tipo de tecnologías son muchos. Sin embargo, la revolución que se está produciendo gracias a los dispositivos IoT también acarrea una serie de cuestiones a resolver. Más allá del desafío que supone interconectar millones de dispositivos entre sí y lograr unos niveles de interoperabilidad y operatividad adecuados, esto supone un desafío de complejidad muy alta para un tema en alza en los últimos tiempos: la ciberseguridad de los dispositivos.

Los desafíos relativos a la ciberseguridad del mundo IoT son muchos: existen numerosas amenazas, riesgos y vulnerabilidades. De acuerdo a un estudio de *Hewlett Packard Entreprise Research* (Čolaković & Hadžialić, 2018), la mayoría de los problemas de seguridad en este tipo de dispositivos vienen provocado por: insuficiente autenticación y autorización, falta de cifrado en tránsito, interfaces web inseguras, software y firmware inseguros, etcétera.

Se podrían poner muchos ejemplos de la necesidad de lograr unos niveles de seguridad adecuados en el uso de estos dispositivos.

Por ejemplo, si escogemos uno de los ámbitos de aplicación del que se está hablando cada vez más, las *Smart Cities*, preguntémonos qué podría ocurrir si un atacante malicioso pudiera llegar a controlar los semáforos (conectados a Internet) de una ciudad. Otro ejemplo claro son los vehículos autónomos y conectados, que cada vez están más conectados a Internet y que pueden llegar a poner en peligro vidas humanas si un atacante es capaz de controlar remotamente el vehículo explotando sus vulnerabilidades. Un caso práctico que mostró en su momento la necesidad de securizar este tipo de dispositivos fue el llevado a cabo por la Mirai Botnet. Este ha sido uno de los mayores ataques por Denegación de Servicio Distribuida (DDoS) de la historia y se originó a partir de un tipo de malware contra dispositivos IoT (Margolis et al., 2017).

2.2. Planteamiento del trabajo del problema elegido

El trabajo que se expone en la presente memoria nace de la necesidad de disponer de herramientas que permitan caminar hacia un uso de dispositivos IoT de forma segura.

Dos de las bases de datos más temibles y completas en internet son Shodan y ZoomEye (R. Li et al., 2020). Estas herramientas son una base de datos de dispositivos IoT conectados a Internet de las cuales podemos recoger una amplia variedad de datos que se listarán en

capítulos posteriores. En el próximo apartado se tratará de describir adecuadamente las características tanto de Shodan como de ZoomEye.

Sin embargo, el uso de estas herramientas también tiene una parte negativa ya que pueden dar visibilidad a vulnerabilidades y/o fallos de seguridad que pueden ser explotados por atacantes maliciosos.

Por una parte, este trabajo debe facilitar el uso de las herramientas Shodan y ZoomEye. De esta manera, el usuario podrá realizar búsquedas en ambas herramientas mediante una única aplicación y con criterios estandarizados, de forma que se mejore la gestión de los resultados que devuelvan en ambas herramientas.

Por otra parte, el trabajo también supone una prueba de concepto para la comparación entre herramientas similares (en este caso Shodan y ZoomEye) de descubrimiento de componentes públicos. Las interfaces se han de diseñar de forma que permitan la comparación entre los resultados de las dos herramientas.

En definitiva, este proyecto, con su consiguiente aplicación, pretende ser una prueba de concepto para futuras investigaciones que se realicen haciendo uso de herramientas de descubrimiento de dispositivos públicos o privados conectados a Internet como Shodan o ZoomEye. Además, la aplicación también servirá como una versión beta dirigida a empresas a las que les puedan interesar conocer la exposición de sus redes y sistemas de información de forma pública.

En el presente documento se utilizarán términos como “trabajo”, “proyecto”, “desarrollo” o “documento”, dependiendo el caso para referirse a todo el aporte realizado por parte de sus autores en todo el proceso del Trabajo de Fin de Máster.

2.3. Estructura del trabajo

El trabajo, independientemente de los apartados de la presente memoria, se ha estructurado en dos fases diferenciales:

1. Búsqueda de información y herramientas que traten de cubrir los objetivos planteados en el apartado *4. Objetivos concretos y metodología de trabajo*. En esta fase se ha realizado la búsqueda de información para poder después realizar el diseño de la herramienta.

2. Desarrollo de software a modo "prueba de concepto" para cubrir todos los objetivos planteados. A partir del apartado *5. Diseño de DÉDALO* .

El trabajo que se presenta en este documento ha tenido como resultado final una aplicación que se ha llamado "DÉDALO". Para facilitar la comprensión lectora, en el documento se utilizarán frases como "la aplicación", "desarrollo software" o similares para referirse a DÉDALO. Por supuesto, también se utilizará el nombre oficial de la aplicación para hacer referencia a ella.

Se ha añadido en el Anexo del presente documento una referencia al repositorio de GitHub en el que se aloja el código de la aplicación desarrollada como resultado de este trabajo. En dicho apartado se puede encontrar la ruta y algunas notas importantes a tener en cuenta por parte de usuarios que quieran utilizar su código.

El presente documento tiene los apartados que se describen a continuación:

1. **Introducción**: se realiza una breve introducción acerca de los dispositivos IoT, así como una primera definición del planteamiento del trabajo y problema a resolver.
2. **Estado del arte**: se realiza el estado del arte sobre IoT, fuentes abiertas (OSINT), la aplicación Shodan y los estudios realizados hasta el momento con dicha herramienta. Por último, se añaden una serie de conclusiones sobre el propio estado del arte.
3. **Objetivos concretos y metodología de trabajo**: se listan los objetivos concretos del trabajo y se define la metodología llevada a cabo para su desarrollo.
4. **Diseño de DÉDALO**: en este capítulo se define la lógica de la aplicación DÉDALO implementada. Para ello, se describen los pasos dados desde el diseño de casos de uso hasta los diagramas de secuencia. En primer lugar, se desarrollan los casos de uso que se han definido para el uso de la herramienta. Tras ello, se listan los requisitos funcionales y no-funcionales planteados al inicio del proyecto. Se define la arquitectura utilizada, empezando por el entorno de desarrollo utilizado en el trabajo, diagramas de componentes, diagramas de clases y una descomposición del código. Por último, se presentan los diagramas de secuencia más relevantes.
5. **Análisis de DÉDALO**: se muestra el resultado de la aplicación. Para ello, se describen las interfaces de usuario de la aplicación, mostrando imágenes de estas. Por otra parte, se valora el resultado completo de la herramienta, mostrando la consecución de requisitos funcionales y no-funcionales. Por último, se describen las principales

desviaciones que han ocurrido a lo largo del desarrollo del proyecto y el impacto que han tenido para la consecución del proyecto.

6. **Escenarios de aplicación prácticos de DÉDALO:** se proponen cuatro casos de análisis realizados a partir del uso de la herramienta. Es decir, se proponen algunos usos que puede tener la herramienta de cara al usuario y la forma que tiene de explotarla. Esos cuatro casos son: el estudio de los datos encontrados de una red conocida; la comparativa de número de resultados y el tiempo empleado por parte de cada una de las herramientas, la evaluación de las estadísticas; y, por último, un análisis para verificar si la API de ZoomEye bloquea resultados localizados en China.
7. **Conclusiones y trabajo futuro**: se analizan las conclusiones obtenidas del proyecto, así como una propuesta de posibles evolutivos a realizar a futuro.
8. **Anexo**: se describe el contenido de la página del repositorio GitHub en la que se ha subido la aplicación DÉDALO, así como la manera de descargarla, configurarla y ejecutarla localmente.

3. Estado del arte

En el presente apartado se analiza el estado del arte de todo el contexto relacionado con Shodan, ZoomEye y sus posibles aplicaciones prácticas.

En primer lugar, se define el concepto “OSINT” o inteligencia de fuentes abiertas, dado que es un término muy importante para entender la tipología de herramientas que abarca.

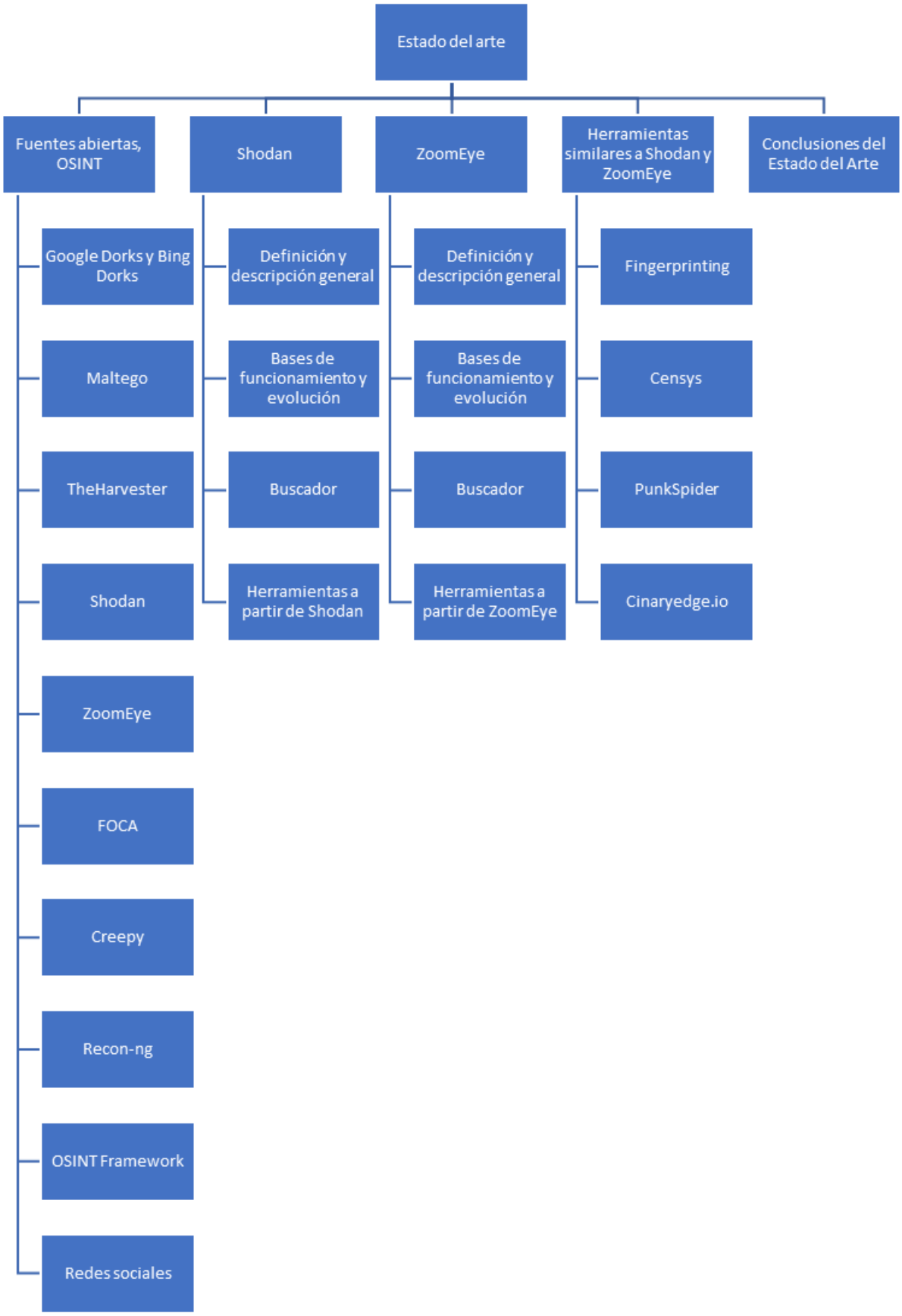
En segundo lugar, se desarrollan cuatro apartados relacionados con Shodan. Para ello, se comienza definiendo la herramienta, su base de funcionamiento y evolución, el uso del buscador del que dispone, así como la API de la que dispone. Se realiza el mismo estudio con la plataforma ZoomEye.

Más tarde, se listan y describen algunas de las herramientas similares a Shodan y ZoomEye más destacadas del mercado y algunas herramientas desarrolladas utilizando Shodan que dan cierta visión en cuanto al trabajo realizado hasta la fecha.

Por último, se dan las conclusiones derivadas del estudio del estado del arte realizado en todo el apartado.

En la siguiente Figura 2 se listan de forma gráfica los diferentes conceptos que se describen en el presente apartado.

Figura 2: Mapa conceptual del estado del arte



3.1. FUENTES ABIERTAS, OSINT

Los dispositivos IoT tienen, por su propia definición, un elemento importante: están conectados a Internet. Esto incluye, de la misma manera, una característica intrínseca: están expuestos. Este hecho da lugar a que puedan convertirse en vectores de ataque explotados por parte de actores maliciosos, que suelen utilizar la llamada "Inteligencia de fuentes abiertas" para dar el primer paso de sus posteriores ataques: la recopilación de información.

Lo que denominamos "*Open Source Intelligence*" (conocido por sus siglas en inglés "OSINT") o "Inteligencia de fuentes abiertas" es la inteligencia que se produce a partir de información pública y disponible que es recolectada, explotada y diseminada de manera oportuna a un público apropiado con el fin de abordar un requisito de inteligencia específica (Vidal & Augusto, s. f.).

Según Alberto Fonte, colaborador habitual de *derechodelared.com*, OSINT hace referencia al conjunto de técnicas y herramientas para recopilar información pública, analizar los datos y correlacionarlos convirtiéndolo en conocimiento útil (*OSINT, ¿Qué es? ¿Para qué sirve? | Derecho de la Red*, s. f.). Es decir, aplicando OSINT se puede conseguir información de fuentes públicas para después convertirse en pura inteligencia.

OSINT tiene definidas una serie de fases que permiten agilizar las tareas de todo el proceso: toma de requisitos, identificación de fuentes de información, adquisición, procesamiento de la información, análisis de los datos obtenidos y presentación de resultados.

Para elaborar esto, se disponen de muchos tipos de herramientas que, en una fase u otra, ayudan a la interceptación de información, a su tratamiento y/o al análisis de esta.

3.1.1. Herramientas OSINT

A continuación, se listan algunas de las herramientas OSINT más relevantes. Sin embargo, existen muchas más herramientas de las listadas.

3.1.1.1. Google Dorks y Bing Dorks

El uso de "*dorks*" en buscadores como Google o Bing permite afinar las búsquedas para que el resultado que devuelva el buscador esté filtrado con información precisa. Estos "*dorks*" son etiquetas y operadores avanzados que permiten localizar cadenas específicas de texto.

3.1.1.2. Maltego

Maltego es una aplicación software que proporciona información y datos de muchos tipos, además de visualizarlos de una forma esquemática y ordenada. Maltego extrae información pública de diversas fuentes, por ejemplo: motores de búsqueda, redes sociales, APIs en línea, metadatos, registros DNS, etc.

3.1.1.3. The Harvester

The Harvester es una herramienta que permite recolectar correos electrónicos, subdominios, hosts, nombres, puertos abiertos y banners.

3.1.1.4. FOCA

Herramienta de la empresa ElevenPaths para buscar metadatos e información oculta en los documentos que procesa. Tiene capacidad de tratar muchos tipos de archivos y de recoger los documentos desde la web. Además, tiene un amplio abanico de funcionalidades.

3.1.1.5. Shodan

Shodan es un buscador de direcciones HTTP que están conectadas a Internet. Es el llamado buscador del "IoT", ya que tiene capacidad de encontrar dispositivos que no encuentra buscadores como Google, por ejemplo, webcams, cámaras de seguridad, etc.

3.1.1.6. ZoomEye

Herramienta homóloga de Shodan de origen chino. Es una herramienta de similar funcionamiento a Shodan ya que tiene la capacidad de listar dispositivos IoT.

3.1.1.7. Creepy

Herramienta OSINT que recopila información relacionada con la geolocalización desde fuentes en línea y tiene capacidad de visualizar los datos de varias maneras: uso de mapa, formato CSV, etc.

3.1.1.8. Recon-ng

Se trata de un *framework* completo de reconocimiento diseñado para ejecutar reconocimiento basado en webs de fuentes abiertas de manera ágil.

3.1.1.9. OSINT Framework

Se trata de un repositorio alojado en la página web www.osintframework.com que recopila información acerca de las herramientas OSINT disponibles en la red. El *framework* tiene un gran abanico de herramientas y opciones según sus categorías.

3.1.1.10. Redes sociales

Las redes sociales también son fuentes de información desde las cuales se pueden obtener todo tipo de datos personales de personas, quienes en muchas ocasiones publican información sin conocer los numerosos riesgos que eso entraña.

Un ejemplo de ello es la noticia publicada en medios de comunicación en noviembre de 2021 relacionada con el uso por parte de la Policía de Sevilla (España) de técnicas OSINT en redes sociales para localizar fiestas de *Halloween* ilegales (20minutos, 2021).

3.2. SHODAN

En este apartado se describe Shodan mediante diferentes subapartados. En primer lugar, se da una visión general de la herramienta, sus bases de funcionamiento y la evolución que ha seguido. Después se describe el funcionamiento y forma de uso del buscador del que dispone. En última instancia, se analiza de forma breve la API de la que dispone y mediante la cual se puede utilizar Shodan desde entornos diferentes a la versión web.

3.2.1. Definición y descripción general

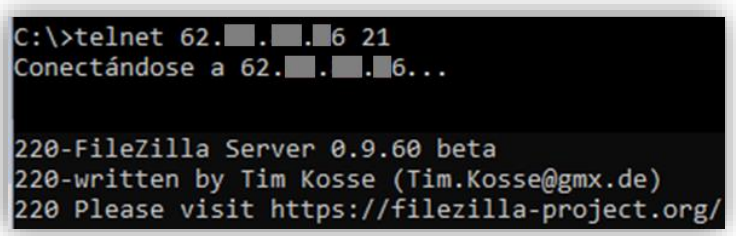
El proyecto Shodan consta de un motor de búsqueda y otras utilidades enfocadas a encontrar información detallada sobre todos los dispositivos y servicios accesibles en Internet. En vez de indexar páginas web, tal y como lo hacen los buscadores tradicionales, Shodan contiene una base de datos con la información que ha recogido en los continuos rastreos que hace sobre Internet. Según la propia compañía (*On-Demand Scanning - Shodan Help Center*, s. f.), Shodan rastrea la Web completamente al menos una vez por semana, lo cual le permite tener datos históricos de la evolución de los dispositivos conectados a nivel mundial.

3.2.2. Bases de funcionamiento y evolución

Shodan fue creado en 2009 por John Matherly. El proyecto inicial consistía en un buscador que interrogaba a una base de datos alimentada mediante un rastreador que aplica técnicas de *fingerprinting* de forma sistemática en Internet. En su primera forma, el rastreador buscaba los *banners* de los servicios. Habitualmente se conoce como *banner* la respuesta que devuelve

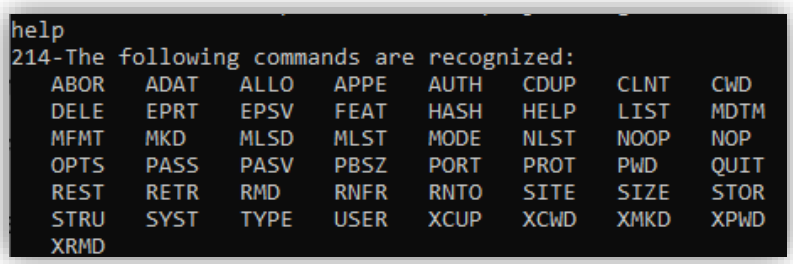
un servicio a un intento de conexión. Si, por ejemplo, se abre una conexión telnet al puerto 21 de una dirección IP que aloja un servidor FTP, se establecerá la comunicación con el servicio de FTP. Dependiendo de su configuración, el servicio puede devolver un *banner* con información acerca de del servidor FTP que está corriendo en la máquina, tal y como se observa en la Figura 3.

Figura 3: Ejemplo de *banner* en un servidor FTP



El propio creador de Shodan, John Matherly (Matherly, 2020) define *banner* como los metadatos de los servicios e indica que una línea de trabajo muy importante del proyecto Shodan ha consistido en enriquecer la información que se recoge de los dispositivos. El rastreador no se conforma con escuchar la primera respuesta que da el dispositivo, sino que lanza pequeños comandos que pueden dar información más detallada. De esta forma, todos los resultados llevarán una serie de datos básicos como dirección IP, puerto o *hostname*, que en muchos casos se complementan con información extra como nombre de producto, versión, etc. Siguiendo con el ejemplo, tras la conexión a un servidor FTP, la ejecución del comando "*help"* devolverá la lista de comandos disponibles (Figura 4). Aunque el *banner* inicial no hubiera devuelto texto alguno, la respuesta al comando podría dar pistas sobre el software que da el servicio FTP.

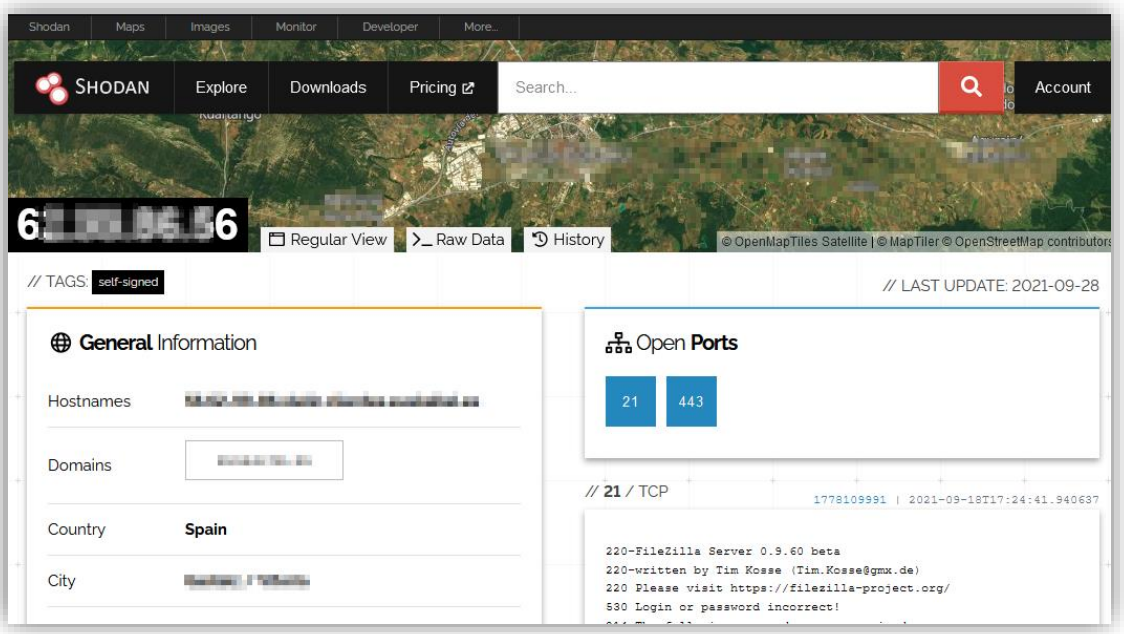
Figura 4: Información extendida mediante envío de comandos



La gran potencia de Shodan ha sido incorporar paulatinamente nuevas herramientas que amplían las posibilidades de un simple motor de búsqueda. Las búsquedas que lanza Shodan por defecto son asíncronas, pero algunas licencias de usuario permiten hacer rastreos de redes en tiempo real. Aunque no sean todos de acceso gratuito, los productos que ofrece Shodan actualmente son:

- **Motor de búsqueda:** Es la herramienta más utilizada, un buscador, visualmente similar a Google o Bing, pero que encuentra servicios y dispositivos en vez de páginas web, como se observa en la Figura 5**.**

Figura 5: Vista del buscador de shodan.io



Fuente: www.shodan.io

- **Monitor:** posibilita la monitorización de una red en tiempo real. Además de descubrir todo lo que es visible en Internet para esa red, lo analiza y ayuda a detectar los dispositivos o servicios más vulnerables.
- **API para desarrolladores:** Interfaz de programación que permite que aplicaciones software externas se comuniquen y hagan uso de las herramientas de Shodan mediante lenguajes de programación como Python. La funcionalidad de la API está limitada por el tipo de licencia de que disponga el desarrollador. La más básica permite hacer una cantidad limitada de búsquedas, mientras que el licenciamiento más elevado elimina esas limitaciones y añade muchas más funciones como, por ejemplo, la de hacer rastreos en tiempo real. Junto con la API se distribuye la herramienta "Shodan CLI", que es una versión del buscador en modo comando.
- **Mapas:** Permite ver los dispositivos encontrados distribuidos en un mapa según su localización.
- **Datos masivos:** Permite la descarga de la información recogida por Shodan para que el usuario la trate y haga uso de ella libremente.
- **Imágenes:** El usuario tiene acceso a las imágenes que obtiene Shodan de los servicios y dispositivos y permite buscar incluso por su contenido, ya que está almacenado mediante un OCR.

3.2.3. Shodan: Uso del buscador

3.2.3.1. Búsquedas avanzadas mediante filtros

En la barra de búsquedas de Shodan se pueden hacer consultas simples introduciendo una dirección IP o un nombre cualquiera. El motor intentará encontrar los datos introducidos entre todos los campos de su base de datos o siguiendo el algoritmo que hayan implantado los desarrolladores. Sin embargo, es posible filtrar los campos por los que se quiere lanzar la consulta mediante filtros, palabras clave que delimitan los criterios de búsqueda y ayudan a eliminar resultados no deseados.

3.2.3.2. Formato de los filtros

El formato general de los filtros es el siguiente:

```
filtro:valor
```

No debe haber espacios en blanco alrededor del símbolo de dos puntos ":". Dependiendo del filtro el valor irá entre comillas dobles "valor" o sin ellas. Si al nombre del filtro viene precedido por un guion, se excluirán los valores indicados:

```
-filtro:valor
```

Algunos filtros permiten que se puedan introducir varios valores separados por comas:

```
filtro:valor1,valor2
```

Para hacer búsquedas más complejas se pueden anidar filtros que serán evaluados como condiciones AND:

```
Filtro1:valor1 filtro2:valor2
```

3.2.3.3. Clasificación y ejemplos de filtros

Hay una gran cantidad de filtros que Shodan agrupa por categorías. La lista y clasificación completa de los filtros se puede encontrar en la propia página web de Shodan (*Filter Reference*, s. f.). A continuación, se ofrecen algunas de las categorías de filtros que destacan Shodan, aunque existen más:

- **General:** filtros que no están asociados a ningún servicio en concreto.
- **Http:** filtros asociados al protocolo http.
- **SSL:** filtros asociados a SSL.
- **Bitcoin:** filtros asociados a servicios relacionados con esta criptomoneda.
- **NTP:** filtros relacionados con el protocolo de sincronización horaria.
- **SNMP:** filtros relacionados con el protocolo SNMP, de intercambio de registros de información de los sistemas.
- **Screenshots:** filtros para buscar por las etiquetas asociadas a las capturas de pantalla rescatadas por Shodan.
- **Telnet:** filtros asociados al protocolo telnet.
- **Cloud:** filtros para interrogar por servicios en la nube.
- **SSH:** filtros que tienen que ver con el protocolo SSH.
- **Restricted:** filtros restringidos para usuarios de los planes de contratación de la API más costosos.

A modo de pequeña muestra de los filtros de Shodan, a continuación, se describen algunos de los más utilizados y se ofrecen ejemplos de su uso.

- **Filtro "city".** Para encontrar dispositivos que se encuentran en una ciudad en concreto: `city:"Logroño"`.
- **Filtro "country".** Para encontrar dispositivos de un país (por ejemplo, España): `country:"Es"`.
- **Filtro "geo".** Para buscar dispositivos alrededor de una coordenada concreta: `geo:42.46667,-2.45`
- **Filtro "hostname".** Para buscar dispositivos con nombres concretos: `hostname:"my_host"`
- **Filtro "IP".** Para buscar por IP. `ip:8.8.8.8`.
- **Filtro "net".** Para buscar dispositivos en una red concreta: `net:8.8.8.0/24`.
- **Filtro "os".** Para descubrir dispositivos con un determinado sistema operativo: `os:"Windows 10"`.
- **Filtro "port".** Para buscar puertos concretos: `port:21,443`.

3.2.4. La API de Shodan

La API es la herramienta que permite que los desarrolladores conecten su propio software a Shodan para explotar todas las funcionalidades disponibles. Todo lo que se puede hacer desde la web se puede hacer con la API, siempre que se disponga de las licencias necesarias. Como Shodan describe en su portal www.developer.shodan.io/api/introduction existen dos tipos de API:

- API REST: sirve para hacer búsquedas asíncronas como se haría en la web, permitiendo utilizar los mismos filtros y utilidades.
- API Streaming: se utiliza para alimentar bases de datos personales. Mediante ella se van recopilando todos los datos que van recogiendo en tiempo real los rastreadores de Shodan. No se puede hacer búsquedas directamente sobre el flujo de datos,

El uso de la API está limitado por el tipo de licencia que se disponga. Shodan regula mediante un sistema de créditos los usos fundamentales: la búsqueda y el escaneo (*Shodan Credits Explained - Shodan Help Center*, s. f.). Por una parte, un crédito de búsqueda se consume cuando se lanza una consulta con filtros o al recuperar páginas de resultados recuperados posteriores a la primera, que es gratuita. Por otra parte, un crédito de escaneo permite lanzar un escaneo de una IP. Las licencias más avanzadas no solo disponen de una mayor cantidad de créditos, sino que incorporan funcionalidades extras como: mayor número de IPs a

monitorizar, uso de todos los filtros incluyendo 'vuln' y 'tag', búsquedas de IPs en lote, descarga de flujo de datos, etc.

3.2.5. Herramientas a partir de Shodan

En este apartado se enumeran herramientas que han sido desarrolladas a partir de Shodan, tanto por ser utilizada en algún punto de su operativa, como por hacer uso de la API en la herramienta.

3.2.5.1. Argo

Argo (www.github.com/M0tHs3C/Argo) es una herramienta que utiliza las APIs de Shodan y de Censys para descubrir el estado de cámaras IP. Solo funciona para los siguientes tipos de dispositivos, de cámaras: cámaras Hikvision, Viola dvr, AVTECH, Geovision, goAhead, Atlantis y ANPR; servidor Bticino, dispositivos RSP y el servidor de telecomunicaciones SAMIP Selta.

En la Figura 6 se puede ver la apariencia de la aplicación, que básicamente tiene 4 opciones principales:

- Recuperar información de los dispositivos mediante Shodan o Censys.
- Verificar si los dispositivos están activos.
- Testear vulnerabilidades en los dispositivos.
- Explotar vulnerabilidades en los dispositivos.

Figura 6: Menú principal de Argo

Fuente: www.github.com/M0tHs3C/Argo

3.2.5.2. Shogun

Shogun (www.github.com/NullArray/Shogun) es una interfaz de línea de comandos alternativa para Shodan, ya que necesita hacer uso de su API. No aporta características adicionales, ya que sus funcionalidades solo incluyen una versión limitada de todo lo que ofrece Shodan.

3.2.5.3. Shodanier

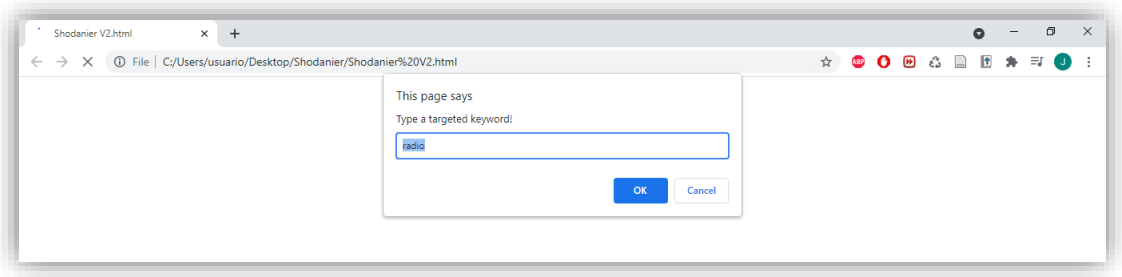
La herramienta Shodanier proporciona una versión extendida del propio Shodan. La fecha más antigua en la que hay información sobre la herramienta es abril de 2018. En el post de "hackxcrack.net" se presenta la herramienta describiéndola con las siguientes tres características:

- Shodanier devuelve más resultados que buscando directamente en su buscador nativo sin necesidad de iniciar sesión.
- La herramienta evita honeypots o puntos de acceso falsos conectándose únicamente al puerto 80.
- Muy fácil de utilizar.

La realidad es que no existe en Internet más información sobre ella y tampoco un manual o guía de uso de la herramienta. Se puede descargar de forma gratuita desde MediaFire.

Se ha descargado la versión 2.0 de la herramienta y simplemente contiene un archivo CSS, dos archivos de formato WOFF2 (Web Open Font Format) y un archivo HTML que al abrirlo presenta la siguiente interfaz:

Figura 7: Interfaz principal de Shodanier



Ejecución de la herramienta en local.

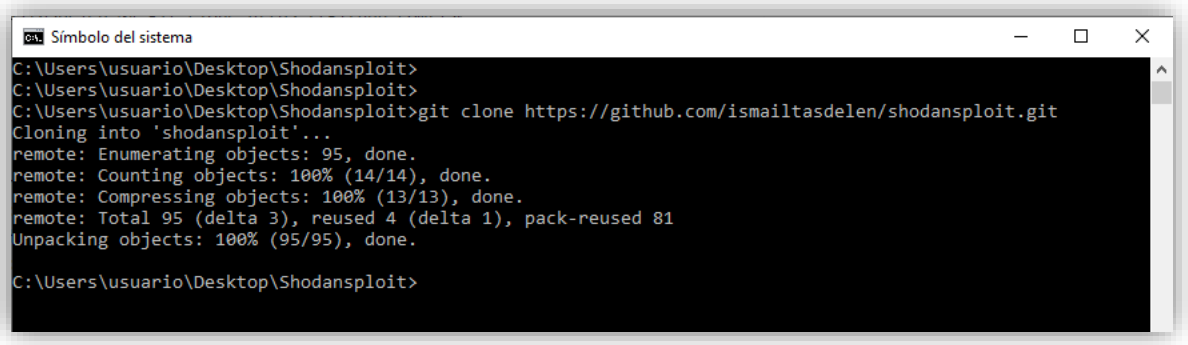
Sin embargo, en la actualidad no devuelve ningún tipo de resultado. Este último hecho coincide con el comentario posteado en el foro de "sinister.ly" donde un usuario indica que la

herramienta ha dejado de funcionar, a fecha 18 de diciembre de 2020 (*Shodanier - A expanded version of Shodan search engine*, s. f.).

3.2.5.4. Shodansploit

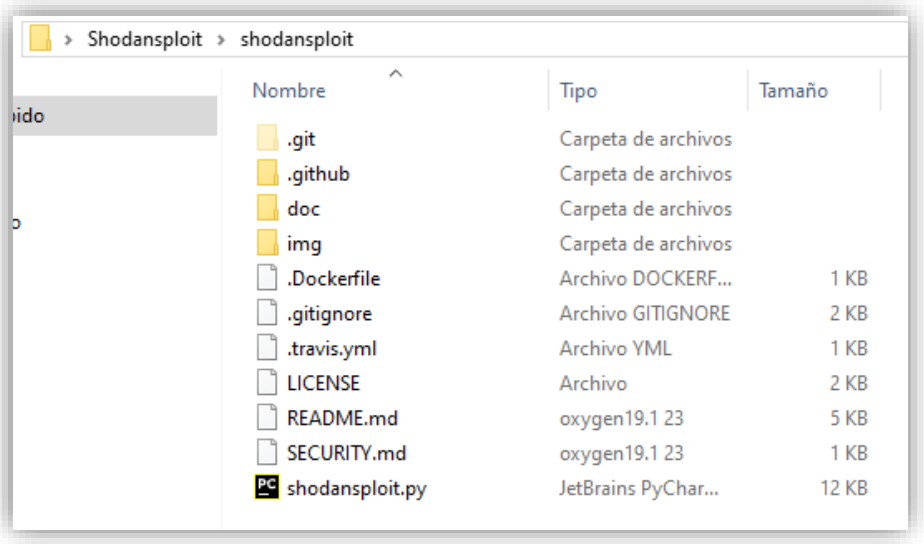
Shodansploit es una herramienta *open source* disponible en GitHub que permite disponer de todas las llamadas en el terminal y permite realizar búsquedas más detalladas.

Figura 8: Instalación de Shodansploit



Para ello, el usuario debe instalar la API de Shodan y modificar ligeramente el código de la aplicación para añadir la API Key del propio usuario, y así poder compilar y ejecutar el archivo sin problema.

Figura 9: Código fuente de Shodansploit descargado de GitHub



Carpeta con ficheros contenidos en el proyecto Shodansploit

Una vez instalado, Shodansploit permite tener las llamadas a la API en la terminal de una manera sencilla y ver el resultado en formato JSON devuelto en la propia terminal. La capacidad interesante de este programa es que, de alguna manera, ordena de cara al usuario las peticiones o búsquedas que se le realizan al buscador de Shodan. A nivel práctico es una herramienta sencilla con un código en Python de no más de 370 líneas.

3.2.5.5. ShoVAT

La herramienta ShoVAT es acrónimo de "*Shodan-based vulnerability Assesment Tools*" y fue presentada en el artículo "*ShoVAT: Shodan-based vulnerability assessment tool for Internet-facing services*" por parte de B. Genge y C. Enaschescy en 2015 (Genge & Enăchescu, 2016).

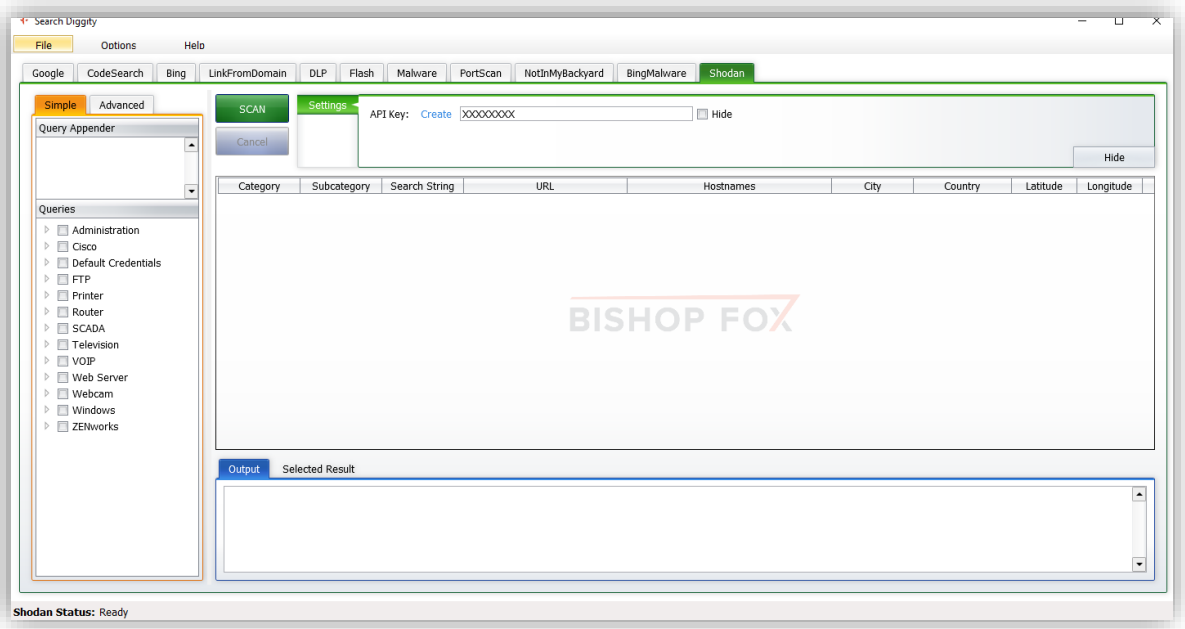
ShoVAT trata de añadir más funcionalidades a las ofrecidas por parte de Shodan con capacidades de evaluación de vulnerabilidades avanzadas. La herramienta principalmente utiliza peticiones tradicionales de Shodan y realiza un análisis en profundidad teniendo en cuenta la información de los servicios. Es decir, ShoVAT ayuda a la identificación de vulnerabilidades, mediante el uso de la *National Vulnerability Database*, haciendo uso de diferentes técnicas (Genge & Enăchescu, 2016).

Desde la publicación del artículo en diferentes revistas científicas y especializadas, la realidad es que no se ha observado continuación en la línea de investigación iniciada por los profesores B. Genge y C. Enachescu. Además, no se ha podido probar la aplicación, ya que su código no se encuentra libre y tampoco está a la venta.

3.2.5.6. Search Diggity

Search Diggity es una herramienta gratuita desarrollada por la empresa Bishop Fox que contiene herramientas que permiten hacer consultas de fuentes abiertas con diferentes buscadores como Google, Bing o Shodan. Dispone de varias pestañas en la que se ofrecen consultas predefinidas para cada tecnología. Como se observa en la Figura 10, las búsquedas de Shodan necesitan la introducción de una API. Su mayor aporte consiste en el cuadro *Queries*, mediante el que se pueden lanzar las consultas predefinidas sin tener que teclear, aunque esto también limita las posibilidades de la aplicación, ya que para que ser útil, las necesidades del usuario tendrían que coincidir necesariamente con alguna de las consultas predefinidas. Se pueden ampliar las búsquedas mediante cuadros de texto en los que se puede utilizar la sintaxis de las búsquedas de Shodan.

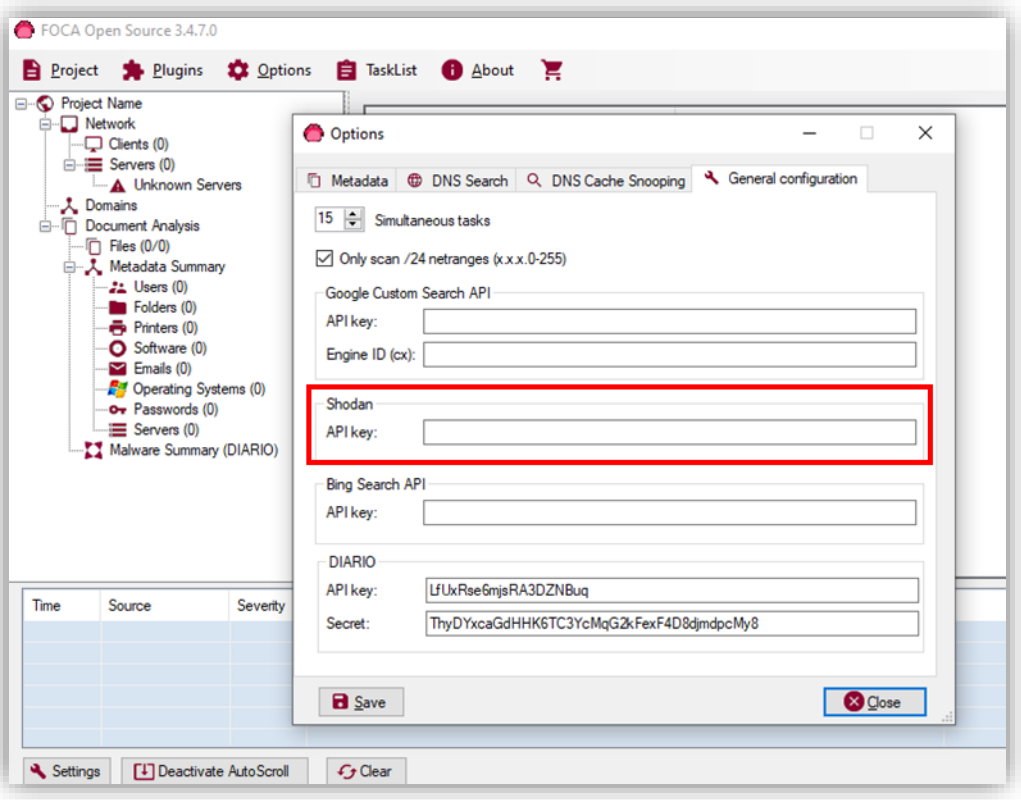
Figura 10: Vista general de Seach Diggity



3.2.5.7. Foca

FOCA es una herramienta de Eleven Paths que es capaz de recopilar información asociada a un dominio mediante las características avanzadas de Google y Bing. A partir de la versión 2 y, especialmente en la versión 3, ha integrado la API de Shodan a su aplicación. La filosofía de funcionamiento es diferente a la vista en herramientas anteriores. FOCA aprovecha su integración con Shodan para mejorar la búsqueda de información de los dominios. Para potenciar FOCA con la API de Shodan, es necesario introducir el valor de la clave de la API, como se observa en la Figura 11.

Figura 11: Ventana de opciones de FOCA, con posibilidad de añadir la API de Shodan



3.2.5.8. Otras herramientas en GitHub

A parte de las herramientas mencionadas, algunas de las cuales han sido usadas como referentes en artículos publicados (Bada & Pete, 2020), existen en la web más herramientas que realizan funciones concretas.

Tras realizar una búsqueda adicional en el repositorio GitHub se ha evidenciado un número alto de repositorios de código relacionados con Shodan. Los usuarios principalmente disponen de bibliotecas de consultas para ejecutar en Shodan.

Sin embargo, no se han encontrado herramientas de desarrolladores alternativos que ofrezcan una interfaz gráfica amigable para el usuario, búsquedas tanto en Shodan como en ZoomEye, comparativas de las búsquedas realizadas en ambas herramientas y/o mapeo de *exploits* y CVEs con los resultados encontrados.

3.3.ZOOMEYE

En este apartado se describe ZoomEye, de manera similar al apartado anterior de Shodan. Primeramente, se da una visión general de la herramienta, sus bases de funcionamiento y la

evolución seguida. Más tarde, se describe el funcionamiento de la herramienta y forma de uso del buscador principal. Por último, se analiza la API de la que dispone y que ha sido utilizada en el presente proyecto.

3.3.1. Definición y descripción general

El proyecto ZoomEye (www.zoomeye.org) es una alternativa a Shodan en el mercado chino. El buscador de dispositivos permite hacer búsquedas simples y otras más complejas utilizando filtros. Por ejemplo, para encontrar un modelo de firewall determinado, tendríamos que escribir `app:"Meraki firewall"` en la caja de búsqueda. ZoomEye también cuenta con una API Rest y de librerías para su explotación en Python (www.github.com/knownsec/zoomeye-python).

La web de ZoomEye.org se presenta en idioma chino, al igual que la documentación, por lo que su implantación fuera del continente asiático no está muy extendida, pese a contar probablemente con la mayor base de datos de dispositivos (R. Li et al., 2020).

ZoomEye dispone de un producto orientado a organismos regulatorios, grandes empresas y gobiernos, que proporciona un mapa detallado de activos y vulnerabilidades: www.knownsec.com/#/product/zoomeye.

3.3.2. Bases de funcionamiento y evolución

ZoomEye es una herramienta cuyo *core* está basado en dos motores de detección llamados Xmap y Wmap para recoger información de dispositivos expuestos de forma pública y servicios web, además de hacer análisis de *fingerprinting*.

En julio de 2013 se lanzó la primera versión de la herramienta. Menos de un año después, en marzo de 2014, el equipo de ZoomEye profundizó en el impacto a nivel global de las vulnerabilidades de cámaras web, llegando a mostrar sus resultados en la Televisión Central de China (CCTV). En 2017, esa misma cadena de televisión expuso el hackeo de un gran número de cámaras con contraseñas débiles, tras lo cual los laboratorios de seguridad de red pudieron utilizar ZoomEye para reproducir dichas vulnerabilidades y poder mitigar las deficiencias.

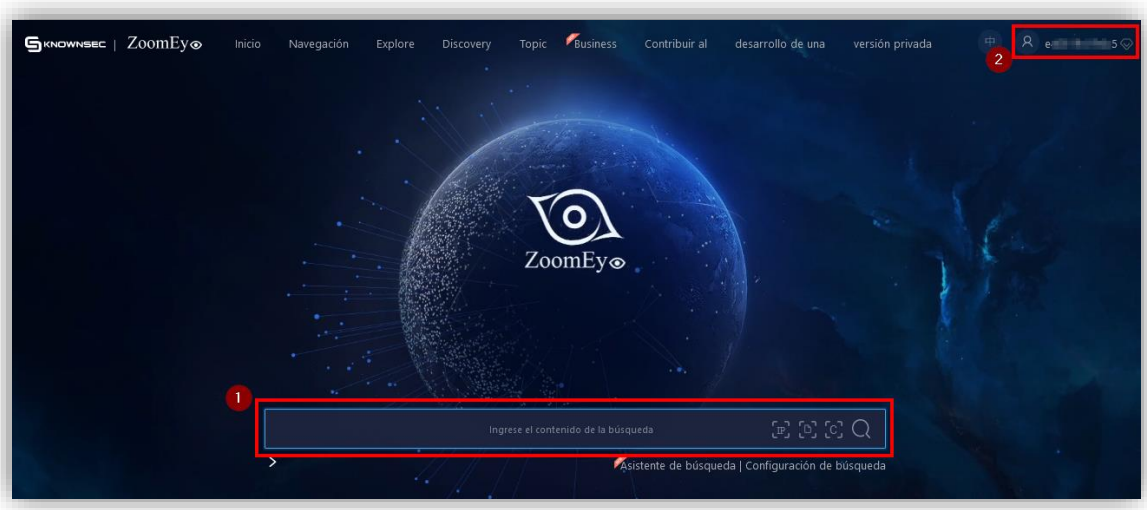
En 2020, los investigadores Li, R., Shen, M., Yu, H., Li, C., Duan, P., y Zhu, L publican un artículo titulado *"A Survey on Cyberspace Search Engine"* donde analizan ZoomEye y otras

herramientas similares y concluyen que es ZoomEye la que más capacidad de búsqueda tiene (R. Li et al., 2020).

ZoomEye limita a un máximo de 20 resultados las búsquedas sin registro en la web. El simple registro en la página web "www.zoomeye.org" otorga un crédito superior y existen planes de licenciamiento de pago que permiten aumentar las posibilidades de búsqueda. A cada consulta que se realice, ZoomEye le asigna una cantidad de créditos determinada. Cada tipo de usuario tiene asignados un total de créditos por mes: el usuario básico cuenta con 10.000, el usuario "privilegiado" 30.000 y el "vip" 40.000.

Para el registro en www.zoomeye.org hay que proporcionar una dirección de correo electrónico y un número de teléfono válidos. Si se desconoce el idioma chino, se recomienda utilizar alguna herramienta en el navegador que traduzca la página. En la Figura 12 se observa la página principal de ZoomEye para un usuario registrado, que haya iniciado sesión. La parte más importante de la pantalla es el cuadro marcado con el número 1, ya que es la que permite realizar las búsquedas. En la parte superior derecha, destacado con el 2, se observa la zona que permite acceder a la información del usuario.

Figura 12: Vista de la página de inicio de www.zoomeye.org

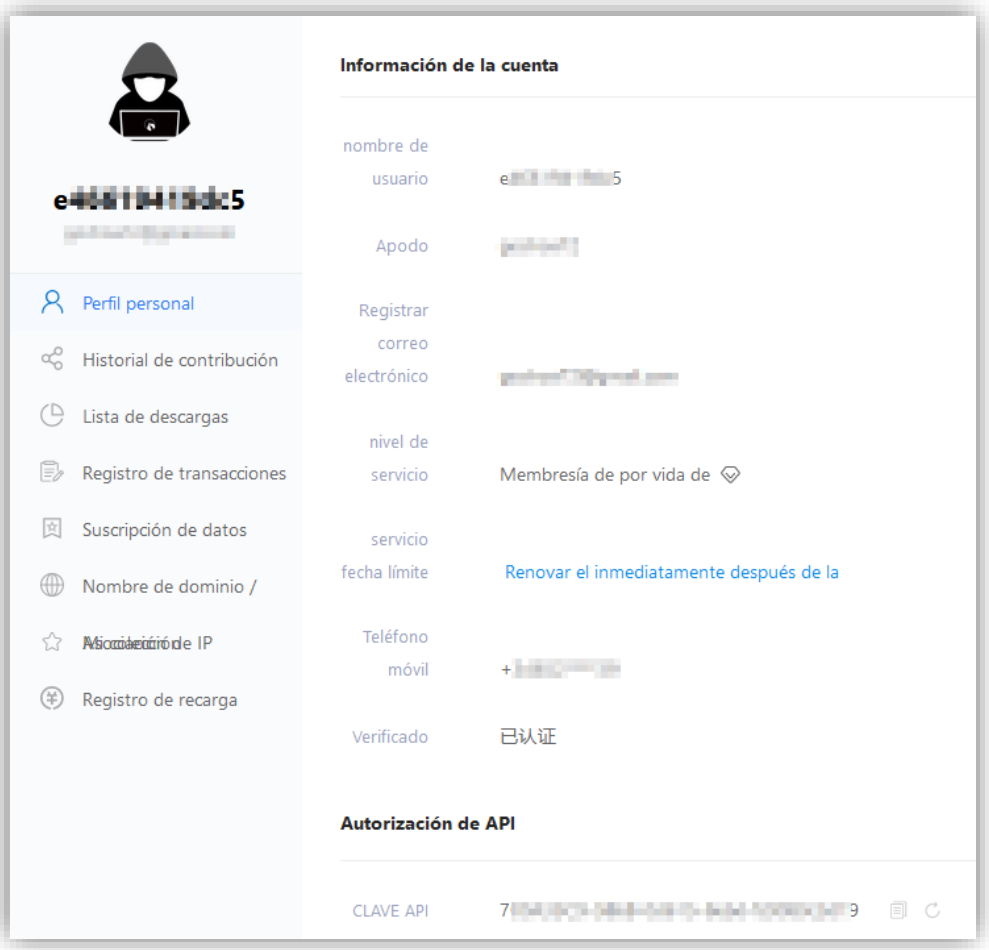


Fuente: www.zoomeye.org (página original traducida con el traductor de Google)

Al seleccionar sobre el código de usuario que aparece en la zona superior derecha, se accede a www.zoomeye.org/profile que proporciona la información de la cuenta del usuario. Como se observa en la Figura 13, además de los datos básicos de código de usuario, correo

electrónico y número de teléfono, en la parte inferior podemos encontrar la *API-key* en la zona inferior.

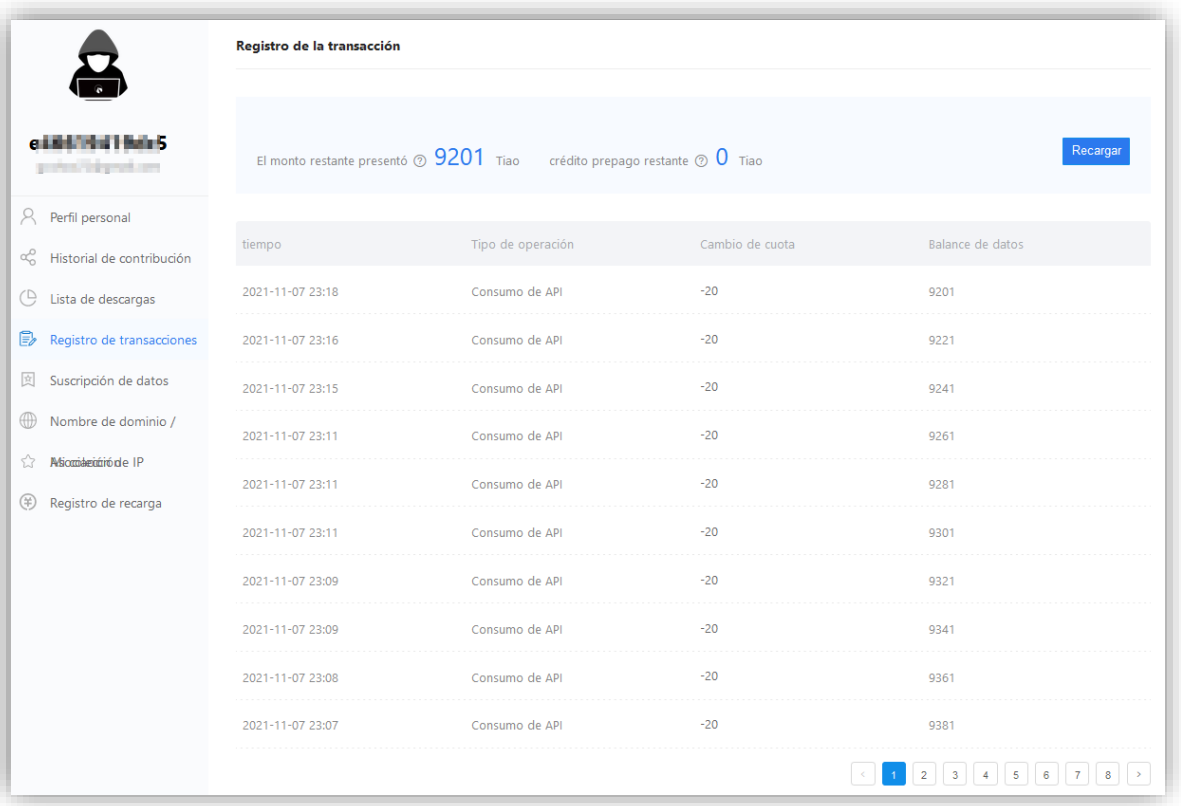
Figura 13: Detalles de la cuenta de usuario registrado



Fuente: www.zoomeye.org/profile (página original traducida con el traductor de Google)

Es posible acceder al histórico de todas las transacciones (consultas) realizadas por un usuario, así como el crédito disponible que le resta, clicando sobre “Registro de transacciones”, lo que lleva a la página mostrada en la Figura 14. Esta página muestra el total de créditos que el usuario tiene disponibles y la fecha y hora de cada consulta que supuso un consumo de créditos.

Figura 14: Registro de transacciones de una cuenta de usuario

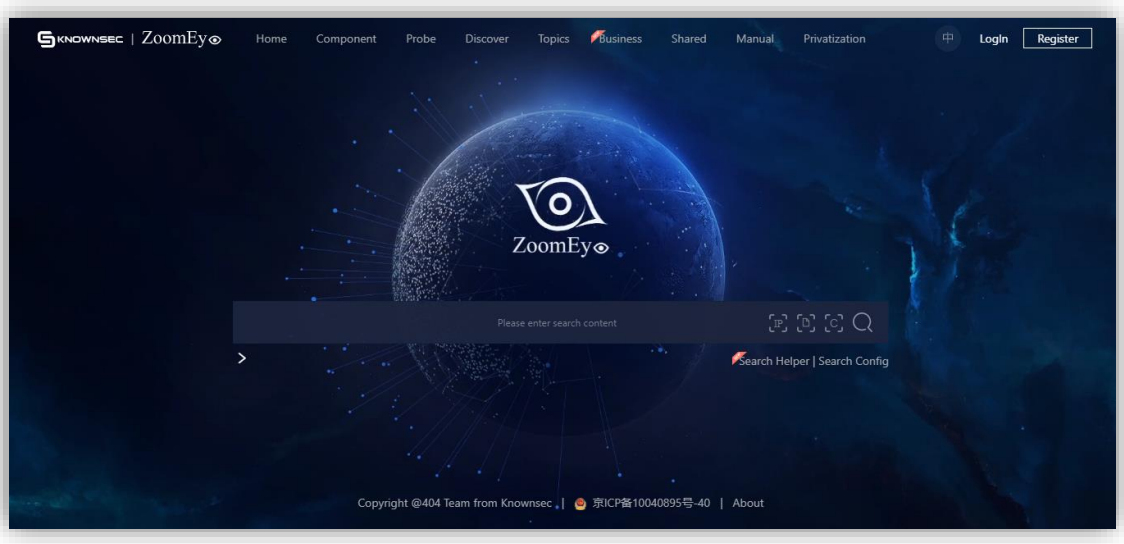


Fuente: www.zoomeye.org/profile/record (página original traducida con el traductor de Google)

3.3.3. ZoomEye: Uso del buscador

Para realizar búsquedas en la herramienta, se debe utilizar el buscador de la página principal que se muestra en la Figura 15 y lanzar búsquedas en función del objetivo que queramos conseguir:

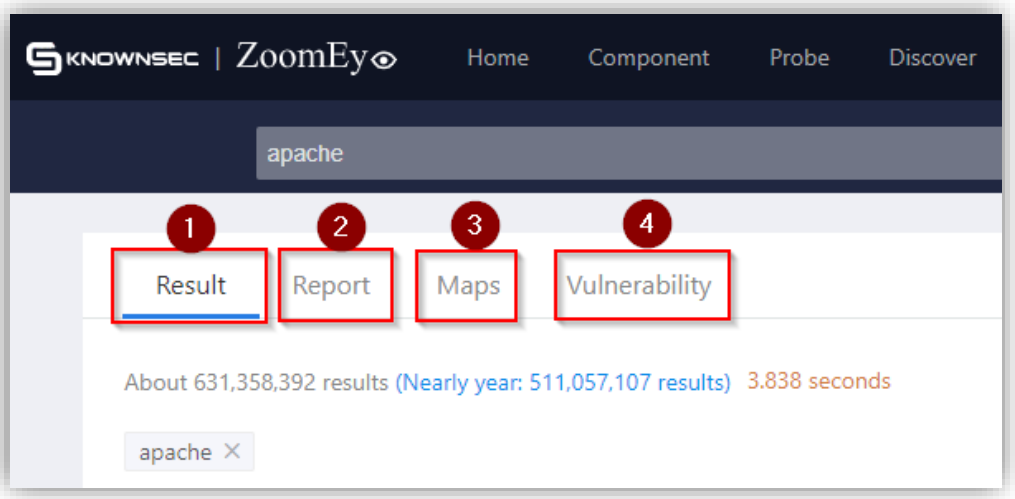
Figura 15: Página principal de ZoomEye



Fuente: www.zoomeye.org

En dicha pantalla se pueden buscar, por ejemplo, dispositivos “Apache”, simplemente escribiendo en el buscador dicha palabra. Tras encontrar datos relevantes, ZoomEye desplegará una pantalla con cuatro tipos de datos a enseñar: resultados generales, reportes relacionados, mapas y vulnerabilidades asociadas.

Figura 16: Ejemplo de búsqueda simple de dispositivos "apache" en ZoomEye



Fuente: www.zoomeye.org

3.3.3.1. Búsquedas avanzadas mediante filtros

Para poder afinar más las búsquedas, se deben incluir una serie de filtros o etiquetas que permiten acotar los resultados. De esta manera, en función de los filtros que establezcamos y de los operadores lógicos que se utilicen, el buscador se limitará a encontrar dispositivos y datos que coincidan con dicha búsqueda.

3.3.3.2. Formato de los filtros

El formato general de los filtros es el siguiente:

```
filtro:valor
```

Como en el caso de Shodan, no deben existir espacios en blanco alrededor de los dos puntos (":"). Dependiendo el filtro, el valor deberá ir entre comillas o no las necesitará.

Por otra parte, se disponen de cuatro operadores lógicos para relacionar los diferentes filtros establecidos:

- **Espacio**: se utiliza como operador lógico "OR". Por ejemplo, si se quieren buscar dispositivos con servicio ssh o http, se debe escribir:
  - `service:"ssh" service:"http"`.
- **Signo "+"**: se utiliza como operador lógico "AND". Por ejemplo, si se quieren buscar dispositivos que sean de tipo router y que se encuentre en China, se debe escribir: `device:"router"+country:"CN"`.
- **Signo "-"**: se utiliza como operador lógico "NOT". Por ejemplo, si se quieren buscar dispositivos que sean de tipo router pero no estén en China, se debe escribir: `device:"router"-country:"CN"`.
- **Signo "()"**: se utiliza para dar prioridad en el procesamiento a unos filtros sobre otros. Por ejemplo, si se quieren buscar dispositivos router en China y después aquellos encontrados después del 2020, se debe escribir: `(device:"router"+country:"CN")+after:"2020-01-01"`.

3.3.3.3. Clasificación y ejemplos de filtros

ZoomEye dispone de un gran número de filtros que tiene listados en su página web (www.zoomeye.org/doc). A continuación, se listan las categorías de filtros que tiene ZoomEye:

- **Host**: filtros asociados a hosts, de los que se encuentran algunos como aplicación, dispositivo, sistema operativo, IP, ciudad, etcétera.

- **Servicios Web**: filtros relacionados con servicios web, como por ejemplo, aplicación, encabezado, título, página web, etcétera.

A continuación, se listan filtros y ejemplos relacionados con los hosts:

- **Filtro "app"**. Para encontrar dispositivos de un tipo de aplicación concreta: `app:"ProFTPD"`.
- **Filtro "device"**. Para encontrar dispositivos de un tipo concreto: `device:"router"`.
- **Filtro "os"**. Para encontrar dispositivos con un tipo de sistema operativo concreto: `os:"Ubuntu"`.
- **Filtro "IP"**. Para encontrar dispositivos con una IP concreta: `ip:192.168.1.1`.
- **Filtro "port"**. Para encontrar dispositivos con un puerto concreto: `port:80`.
- **Filtro "city"**. Para encontrar dispositivos en una ciudad concreta: `city:Madrid`.
- **Filtro "country"**. Para encontrar dispositivos en un país concreto: `country:"CN"`.

Por su parte, a continuación se listan filtros y ejemplos relacionados con las webs:

- **Filtro "webapp"**. Para encontrar webs de un tipo de aplicación concreta: `webpp:"wordpress"`.
- **Filtro "header"**. Para encontrar webs con un encabezado concreto: `header:server`.
- **Filtro "title"**. Para encontrar webs con un título concreto: `title:Google`.
- **Filtro "IP"**. Para encontrar webs con una IP concreta: `ip:192.168.1.1`.
- **Filtro "site"**. Para encontrar webs con de una página concreta: `site:helloworld.com`.
- **Filtro "city"**. Para encontrar webs en una ciudad concreta: `city:Madrid`.
- **Filtro "country"**. Para encontrar servicios web en un país concreto: `country:"CN"`.

3.3.4. La API de ZoomEye

ZoomEye, al igual que Shodan, dispone de una API que permite a los desarrolladores hacer uso del motor de búsqueda ZoomEye para buscar mediante diferentes métodos y funciones. En el caso de esta herramienta, tiene una API compatible con Python disponible en su propia web y en un repositorio de GitHub: www.github.com/knownsec/ZoomEye-python.

La autenticación haciendo uso de la API se puede realizar de dos formas: mediante API-Key y mediante usuario y contraseña. Dispone de una colección de métodos que emulan las búsquedas que se pueden realizar en la web.

3.3.5. Herramientas a partir de ZoomEye

Como se ha comentado previamente, ZoomEye es una aplicación desarrollada en China y la realidad es que la información que se tiene de la misma es limitada y se ajusta a la página web oficial y a unos *papers* en inglés. Estos pocos artículos escritos en inglés no llegan a desarrollar aplicaciones basándose en la API de ZoomEye. Es verdad que existen algunos artículos científicos en chino, pero no ha sido posible obtener la traducción de los mismos.

Por otra parte, se han buscado desarrollos alternativos en repositorios reconocidos como GitHub. Sin embargo, los desarrollos basados en la API de ZoomEye son mínimos y no disponen de funcionalidades que puedan suponer un avance relevante en la materia.

En conclusión, no se han encontrado herramientas que utilicen la API de ZoomEye relevantes en el mercado ni tampoco en repositorios conocidos como, por ejemplo, GitHub.

3.4. HERRAMIENTAS SIMILARES A SHODAN Y ZOOMEYE

Como hemos comentado previamente, las fuentes de inteligencia OSINT están compuestas por una gran variedad de herramientas que buscan recopilar información pública. Atendiendo a la información que recogen tanto Shodan como ZoomEye (por ejemplo, IPs, *hostnames*, servidores, geolocalización, etc.), existen otras herramientas que también tienen como objetivo la recolección de ese tipo de información.

Por ello, en este subapartado se listan y describen algunas otras herramientas reconocidas que tienen funciones similares y buscan objetivos similares a Shodan y ZoomEye.

3.4.1. Herramientas de *fingerprinting* de código abierto: NMAP y ZMAP

El rastreo de la red de Shodan se realiza utilizando prácticas de *fingerprinting*. El *fingerprinting* es la técnica mediante la que se recoge información de sistemas informáticos interactuando con ellos. Estas técnicas son detectables por el sistema interrogado, que podría interponer barreras y medidas que eviten que el sondeo sea exitoso. Entre los proyectos pioneros en este ámbito se encuentran Nmap y Zmap, que han servido de base y modelo para la fase de sondeo tanto de Shodan como de algunas de las herramientas que se explican a continuación.

La herramienta por excelencia para el escaneo de puertos es Nmap, un proyecto de código abierto iniciado en 1997 por Gordon Lyon. Funciona en línea de comandos y está disponible

para múltiples sistemas operativos y distribuciones e incluso cuenta con una utilidad gráfica llamada Zenmap.

El funcionamiento de Nmap es similar al descrito anteriormente para explicar la manera en la que Shodan recoge los *banners* de los servicios. Para obtener la información, Nmap envía paquetes a direcciones IP o a redes y a partir de sus respuestas descubre los hosts activos, los puertos abiertos y los servicios que mantienen disponible.

ZMap Proyect es un proyecto que dispone de varias herramientas *open source* para realizar estudios a gran escala de hosts y servicios, en general, de Internet. Es un proyecto que comenzó la Universidad de Michigan con el objetivo de mejorar la velocidad de los sondeos de Nmap. El primer paquete de escaneos ZMap data de 2003. A partir de ese momento, se han ido desarrollando diferentes módulos al proyecto que en el año 2015 tomó el nombre de "Zmap Project" para cobijar bajo su paraguas todas estas herramientas.

La primera aplicación del proyecto y la que le da el nombre es ZMap. Esta herramienta está diseñada para realizar inspecciones en todo Internet de manera integral y rápida. Según sus autores es capaz de escanear el espacio de dirección pública IPv4 para un único puerto en menos de 45 minutos al teóricamente 97% de rapidez máxima de 1 gigabit Ethernet y con una cobertura estimada del 98% de los hosts disponibles de forma pública (Zakir Durumeric, Eric Wustrow, y J. Alex Halderman, 2013) y según indica la propia web del proyecto (*The ZMap Project*, s. f.-a), con una conexión de 10 gigabits Ethernet es capaz de escanear en cinco minutos el espacio completo de direcciones IP.

Zmap realiza los escaneos mediante los recursos del usuario, lo cual tiene algunos inconvenientes:

- La rapidez de los rastreos dependerá de la velocidad del hardware y la conexión a Internet del usuario.
- En los logs de todos los dispositivos escaneados quedará registrada la dirección IP desde la que se lanza el rastreo, con el riesgo de entrar en listas negras de usuarios.

En la actualidad tiene activas 12 herramientas: ZMap, ZGrab, ZDNS, ZCrypto, ZLint, ZCertificate, ZIterate, ZBlacklist, ZTee, ZAnnotate, ZSchema y mrt2json.

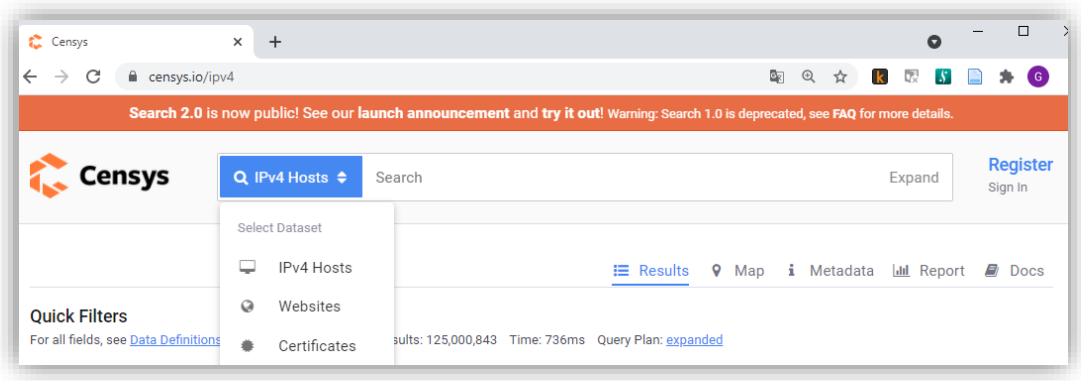
3.4.2. Censys

El proyecto censys.io se originó en la Universidad de Michigan con parte del grupo que participaba en el proyecto ZMap y fue presentado en 2015 en la *Conference on Computer and Communication Security* (Durumeric et al., 2015) como un motor de búsqueda público y servicio de tratamiento de datos obtenidos a través de rastreos continuos de Internet. Posteriormente se añadieron funciones de pago, aunque sigue siendo accesible de forma gratuita para la comunidad investigadora.

Como muestra la Figura 17 el buscador permite encontrar información a través de tres bases de datos (*Research Access to Censys Data*, s. f.):

- **Hosts de Internet:** Tabla con información sobre 2032 puertos de los hosts IPv4 activos y públicos.
- **Sitios Web:** Datos de todos los dominios activos.
- **Certificados:** Con datos de los certificados X.509 encontrados por el rastreador de Censys.

Figura 17: Vista de la página de inicio de www.censys.io



Fuente: www.censys.io/ipv4

3.4.3. PunkSpider

PunkSpider es una herramienta libre que permite realizar escaneos de vulnerabilidades en páginas web en tiempo real con el objetivo de detectar debilidades en la misma. La herramienta consulta una base de datos que contiene páginas web con vulnerabilidades y cuando el usuario navega en ella avisa con el número de vulnerabilidades conocidas. Esta base de datos contiene información acerca de webs escaneadas previamente, pero permite

solicitar el escaneo de una página web no registrada y mostrar los resultados un tiempo después.

Esta herramienta estuvo activa entre 2013 y 2015, periodo en el que incluso llegó a recibir fondos de la Agencia de Proyectos de Investigación Avanzados de Defensa estadounidense. Tras 5 años de inactividad, volvió a estar operativa en 2021.

3.4.4. Binaryedge.io

BinaryEdge es un portal cuyo objetivo es adquirir información sobre dispositivos y ponerlo a disposición de los usuarios, de forma similar a como lo hacen Shodan o ZoomEye. El rastreo de la red se hace utilizando dos métodos:

- Con sondas que recogen información.
- Mediante *honeypots* accesibles en la red, que almacenan los datos de los sistemas que se conectan a ellos.

El objetivo primordial de www.binaryedge.io es mostrar a los usuarios la superficie de ataque que tienen expuesta y destacar aquellas vulnerabilidades que deberían proteger. Además de listar los puertos y servicios mostrados en Internet, muestra avisos sobre posibles vulnerabilidades y hace énfasis en algunos de los puntos que típicamente presentan mayores debilidades: escritorios remotos accesibles, carpetas compartidas de forma incorrecta, seguridad de bases de datos mal configurada, etc.

BinaryEdge cuenta con una API para acceder a los datos y cuatro planes de licenciamiento, siendo gratuito el más limitado de todos ellos.

3.5. CONCLUSIONES DEL ESTADO DEL ARTE

Tras haber realizado la búsqueda de información referente a Shodan, aplicaciones similares y aplicaciones que utilizan Shodan se manifiesta que es una herramienta considerada una de las referentes en búsqueda de fuentes abiertas.

Según el artículo "*A Survey on Cyberspace Search Engines*" (R. Li et al., 2020), las dos herramientas de búsqueda de dispositivos en internet son Shodan y ZoomEye (ver Tabla 1). En primer lugar, ZoomEye es la herramienta que, según este estudio, más dispositivos es capaz de encontrar. La diferencia es notoria entre ZoomEye (1.190.860.679 dispositivos) y Shodan (436.489.751 dispositivos), ya que el valor de este último es más de tres veces inferior al

primero. En cualquier caso, entre Shodan y la siguiente herramienta también hay una diferencia notoria: Shodan encuentra casi cuatro veces más dispositivos que Censys (11.368.143 dispositivos), que ocupa la tercera posición.

El hecho de que Shodan y ZoomEye estén entre las herramientas de descubrimiento de dispositivos IoT más importantes a nivel mundial es el motivo para haberlas convertido en piedras angulares del proyecto.

Tabla 1: Comparación de dispositivos detectados en diversas plataformas

	Shodan	Censys	**ZoomEye**	Fofa	BinaryEdge
dispositivos	**436.489.751**	111.368.143	**1.190.860.679**	270.363	89.871.839

Fuente: (R. Li et al., 2020)

Por otra parte, se ha evidenciado que existen varias aplicaciones que utilizan Shodan, pero no aplicaciones que hagan uso de ZoomEye. Aparentemente uno de los motivos que puede explicar este hecho, es que ZoomEye sea una herramienta de origen chino, lo que dificulta su uso al existir menos documentación al respecto.

Además, no existen herramientas que dispongan de una interfaz amigable que facilite al usuario, aunque sea de forma mínima, el uso y aprovechamiento de ambas herramientas. Sí que existen algunos módulos o pequeños desarrollos que muestran gráficas de resultados obtenidos de Shodan, pero generalmente no permiten realizar las búsquedas mediante una interfaz más amigable para el usuario. Este hecho podría incluso aumentar, más si cabe, el uso que la comunidad da a las herramientas.

Respecto a las herramientas que usan Shodan, se ha evidenciado una cierta carencia de madurez en las mismas. En el subapartado *3.2.5 Herramientas a partir de Shodan* se han listado algunas aplicaciones que estaban fuera de soporte o mantenimiento y que ya no están operativas. Incluso tras la búsqueda en GitHub se ha puesto de manifiesto que muchas de las aplicaciones son simplemente listados de consultas sin ninguna aportación reseñable. Sin embargo, no se ha evidenciado aplicaciones bien soportadas con algún elemento diferencial. Este hecho se suma a que no se han podido encontrar herramientas que hagan uso de la API de ZoomEye para aprovechar sus capacidades y explotar las funcionalidades que ofrece.

En conclusión, el presente trabajo trata de continuar la investigación que rodea al entorno de las herramientas ZoomEye y Shodan y crear una prueba de concepto, en formato de

aplicación. Esta prueba de concepto pretende ser un primer acercamiento a la comparación de este tipo de herramientas, pudiéndose analizar tiempos de cómputo o cantidad de resultados. Adicionalmente, servirá para que un particular o una organización pueda ser capaz de rastrear la red que requiera y analizar los dispositivos expuestos que atesora.

4. OBJETIVOS CONCRETOS Y METODOLOGÍA DE TRABAJO

4.1. Objetivo principal

El objetivo principal de este proyecto es diseñar e implementar una aplicación, a modo prueba de concepto, que permita explotar las herramientas Shodan y ZoomEye a través de una interfaz gráfica y utilizando las APIs para desarrolladores, con el objetivo de poder comparar ambas herramientas y profundizar en las mismas.

Desde una perspectiva de investigación, el objetivo es realizar un aporte a la comunidad de fuentes abiertas enfocándonos en las capacidades de Shodan y ZoomEye, de tal manera que en el futuro otros compañeros y compañeras de la comunidad académica o de la industria puedan continuar comparando ambas herramientas, añadiendo capacidades y funcionalidades a la prueba de concepto existente y otras posibles ideas que se listan en el apartado *8.2 Líneas de trabajo futuro*. El propósito es que esta aplicación a modo prueba de concepto pueda utilizarse para:

- Comparar herramientas o motores de búsqueda de dispositivos IoT, concretamente Shodan y ZoomEye.
- Investigar el motivo de las diferencias de resultados que se obtienen de las herramientas.
- Lograr información más completa de la red de la que una organización disponga.
- Ampliar capacidades y funcionalidades de la aplicación en diferentes líneas.
- Concienciar a personal de la industria IoT, Industria 4.0 y similares sobre la importancia de la ciberseguridad en este tipo de dispositivos.

4.2. Objetivos específicos

1. Desarrollar una prueba de concepto, en formato de aplicación, que sea capaz de realizar consultas simples y complejas haciendo uso del motor de búsquedas Shodan.
2. Desarrollar una prueba de concepto, en formato aplicación, que sea capaz de realizar consultas simples y complejas haciendo uso del motor de búsquedas ZoomEye.
3. Utilizar la API de Shodan para Python para realizar el desarrollo de la aplicación software.

4. Utilizar la API de ZoomEye para Python para realizar el desarrollo de la aplicación software.
5. Desarrollar una interfaz gráfica que simplifique al usuario la realización de consultas compuestas en Shodan. Actualmente la versión web dispone de un buscador que requiere del uso de etiquetas para realizar búsquedas complejas (por ejemplo, "`country:ES city:Madrid host:192.178.90.3`". Mediante la consecución de este objetivo el usuario dispondrá de varios campos en los que añadir los parámetros de búsqueda, con un formato similar al de un formulario.
6. Desarrollar una interfaz gráfica que simplifique al usuario la realización de consultas compuestas en ZoomEye. Actualmente la versión web dispone de un buscador que requiere del uso de etiquetas para realizar búsquedas complejas (por ejemplo, "`country:ES+city:Madrid+host:192.178.90.3`"). Mediante la consecución de este objetivo el usuario dispondrá de varios campos en los que añadir los campos de búsqueda, con un formato similar al de un formulario.
7. Unificar los campos de búsqueda o filtros a utilizar de Shodan y ZoomEye enfocándose en un grupo limitado y reducido de ellos, que puedan buscar en ambas aplicaciones.
8. Relacionar los resultados encontrados por Shodan en las búsquedas con sus posibles vulnerabilidades haciendo uso también de la API de Shodan.
9. Relacionar los resultados encontrados por ZoomEye en las búsquedas con sus posibles vulnerabilidades haciendo uso también de la API de ZoomEye.
10. Visualizar los resultados de tal manera que puedan realizarse comparativas simples entre Shodan y ZoomEye.

4.3. OBJETIVOS ADICIONALES

1. Realizar el desarrollo de la aplicación con herramientas de código abierto.
2. Desarrollar la prueba de concepto de forma modular, para que pueda ser más sencillo incluir otras APIs de otros motores de búsqueda.
3. Unificar el lenguaje, de forma genérica, de Shodan y ZoomEye.
4. Mostrar datos estadísticos de Shodan y ZoomEye.
5. Añadir una funcionalidad de configuración de las *API-Key* de las herramientas Shodan y ZoomEye de forma que facilite su utilización y administración.

6. Añadir alguna herramienta que verifique y confirme los resultados obtenidos por parte de Shodan y ZoomEye, de forma simple.

4.4. Planificación y metodología

El trabajo documentado en la presente memoria será desarrollado de inicio a fin en un total de 22 semanas, comenzando el 6 de septiembre de 2021 y acabando el 7 de febrero de 2022.

4.4.1. Uso de metodología Scrum

Con el objetivo de plantear un trabajo adecuadamente dimensionado con recursos suficientes y objetivos generales y específicos coherentes, se ha utilizado una metodología Scrum desplegada en varias iteraciones que se detallarán a continuación.

La metodología ágil conocida como "SCRUM" es una reconocida metodología de trabajo que asume la dificultad del proceso de desarrollo de sistemas en general. SCRUM consta de las siguientes fases: planificación, diseño de arquitectura a alto-nivel, desarrollo y cierre. En relación con las fases se dispone del concepto de "*Sprint*" donde se definen plazos temporales cortos con objetivos marcados, pero flexibles (Schwaber, 1997).

Mediante la metodología escogida por los autores, se pretende disminuir el riesgo de fracaso del proyecto e identificar de manera ágil bloqueos y riesgos relevantes del proyecto.

A continuación, se listan las iteraciones que se desplegarán durante el proyecto:

- **Iteración 1**: recopilación de información e investigación del estado del arte. Se han recopilado los proyectos más importantes que utilizan Shodan y ZoomEye del mercado y de repositorios de código y/o foros conocidos.
- **Iteración 2**: diseño y comienzo de desarrollo de la prueba de concepto. Se han establecido las especificaciones funcionales y no-funcionales de la prueba de concepto a realizar que disponga de interfaz gráfica amigable y las funcionalidades básicas.
- **Iteración 3**: finalización de desarrollo de la prueba de concepto para cumplir con los objetivos del proyecto, las especificaciones funcionales y no-funcionales, así como el inicio de la redacción de conclusiones del trabajo. Se ha generado un entregable en formato de librería de Python.
- **Iteración 4**: finalización de las conclusiones, documentación de posibles líneas de trabajo futuras, finalización de la memoria y repaso del documento final.

4.4.2. Plazos de las iteraciones

Los plazos marcados al inicio del proyecto son las siguientes:

- Iteración 1: desde el 6 de septiembre de 2021 hasta el 7 de octubre de 2021.
- Iteración 2: desde el 8 de octubre de 2021 hasta el 10 de noviembre de 2021.
- Iteración 3: desde el 11 de noviembre de 2021 hasta el 15 de diciembre de 2021.
- Iteración 4: desde el 16 de diciembre de 2021 hasta el 1 de febrero de 2022.

Al inicio de las iteraciones 2, 3 y 4 se convocarán reuniones de seguimiento con el director del proyecto para comentar las más relevantes desviaciones y corregir aquellos aspectos que se consideren mejorables.

Figura 18: Diagrama de Gantt con la planificación del proyecto

Por otra parte, el equipo del proyecto, formado por Gonzalo Ochoa de Eguileor y Jon Legarda, han mantenido sesiones semanales donde se han tratado los temas avanzados desde la última reunión, para tomar decisiones y plantear los objetivos a acometer hasta la siguiente reunión.

5. DISEÑO DE DÉDALO

En el presente apartado, se define la lógica de DÉDALO, siguiendo los pasos más habituales de un desarrollo software: definición de casos de uso, listados de requisitos funcionales y no-funcionales, definición de herramientas de desarrollo a utilizar y justificación del uso de los mismos, diagrama de componentes, diagrama de clases, composición modular de la aplicación y diagramas de secuencia.

Nota 1: es importante recalcar el carácter de "prueba de concepto" del trabajo realizado. Es decir, el foco principal de este proyecto no ha sido el desarrollo de una aplicación como fin último, si no como medio para conseguir explorar las posibilidades que ofrecen Shodan y ZoomEye, comparar ambas herramientas y poder abrir líneas de investigación a partir del trabajo realizado.

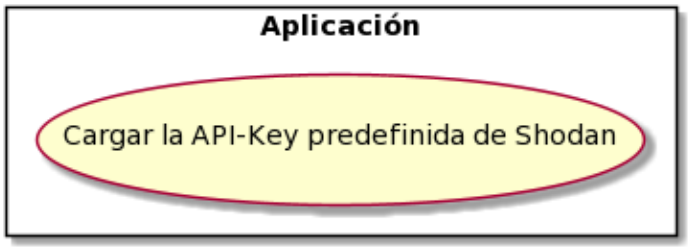
Nota 2: se muestran los diagramas en castellano porque el proyecto se ha llevado a cabo en castellano. Sin embargo, el código de la aplicación ha sido desarrollado en inglés con el objetivo de que pueda llegar a más audiencia y pueda reutilizarse en un futuro por parte de investigadores y expertos en el ámbito.

5.1.CASOS DE USO DEFINIDOS

A continuación, se listan los diagramas UML de los casos de uso definidos para el desarrollo realizado. Se ha utilizado la herramienta PlantUML (ver 5.3.6 Otras herramientas utilizadas).

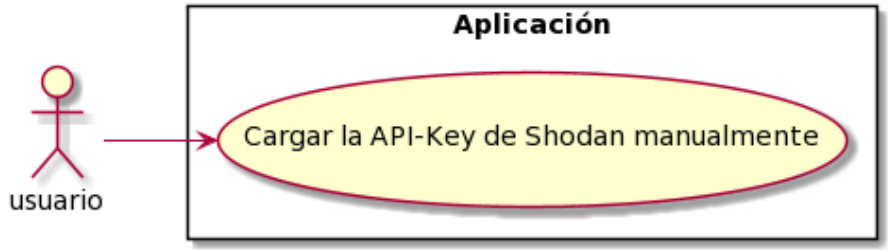
5.1.1. Carga predeterminada de la API-Key de Shodan

Figura 19: CU: Carga predeterminada de la API-Key de Shodan



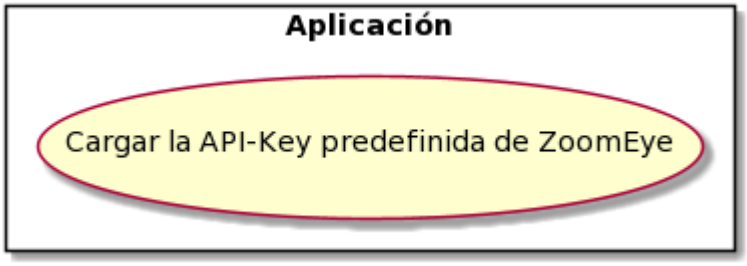
5.1.2. Carga manual de la API-Key de Shodan

Figura 20: CU: Carga manual de la API-Key de Shodan



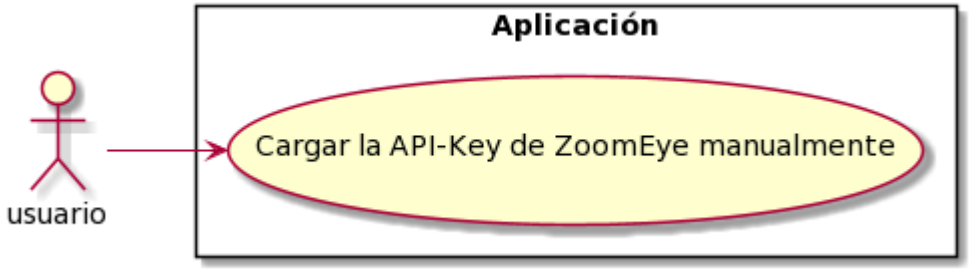
5.1.3. Carga predeterminada de la API-Key de ZoomEye

Figura 21: CU: Carga predeterminada de la API-Key de ZoomEye



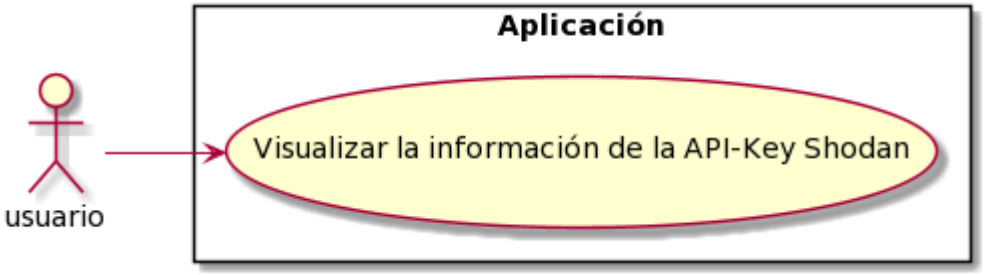
5.1.4. Carga manual de la API-Key de ZoomEye

Figura 22: CU: Carga manual de la API-Key de ZoomeEye



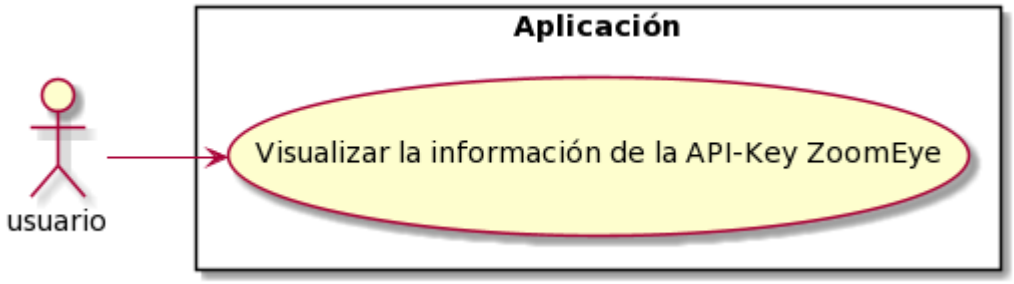
5.1.5. Visualizar información del usuario mediante API-Key de Shodan

Figura 23: CU: Visualizar información del usuario mediante API-Key de Shodan



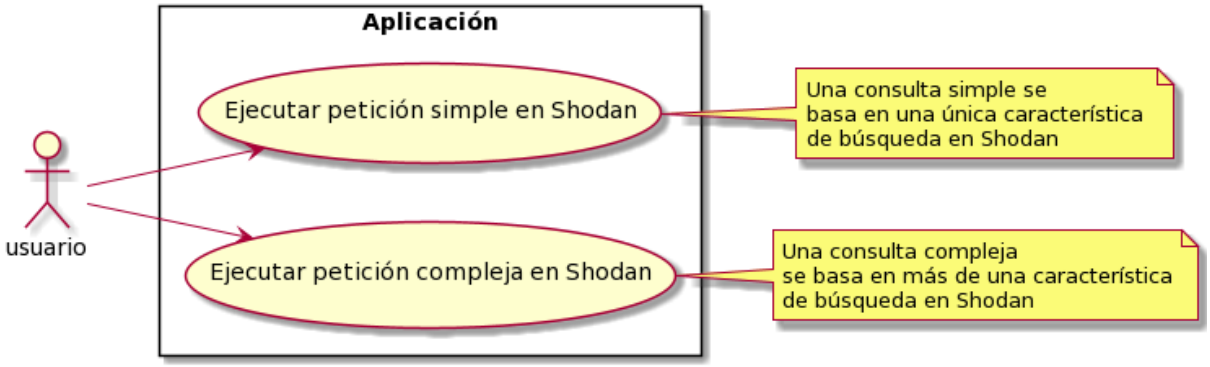
5.1.6. Visualizar información del usuario mediante API-Key de ZoomEye

Figura 24: CU: Visualizar información del usuario mediante API-Key de Shodan



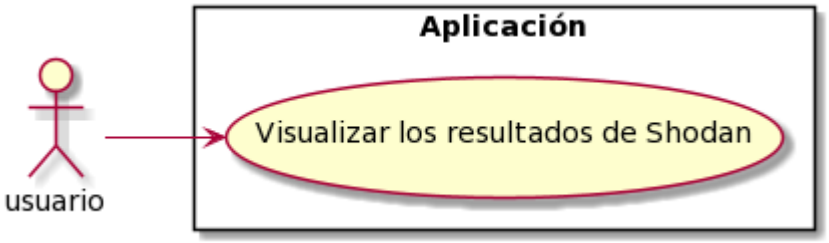
5.1.7. Realizar búsquedas simples y complejas en Shodan

Figura 25: CU: Realizar búsquedas simples y complejas en Shodan



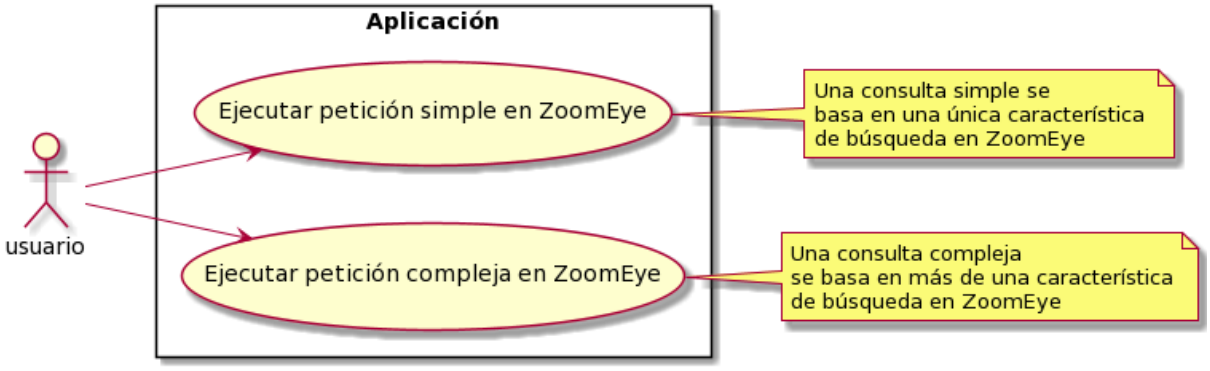
5.1.8. Visualizar resultados de Shodan

Figura 26: CU: Visualizar resultados de Shodan



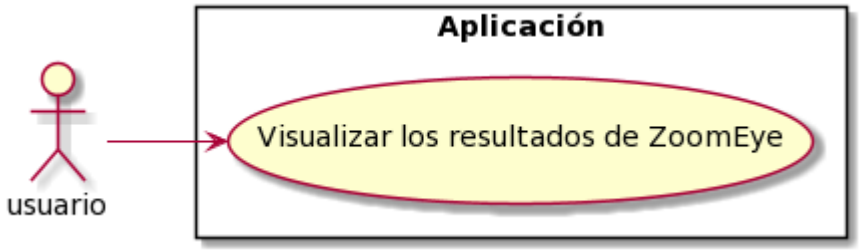
5.1.9. Realizar búsquedas simples y complejas en ZoomEye

Figura 27: CU: Realizar búsquedas en ZoomEye



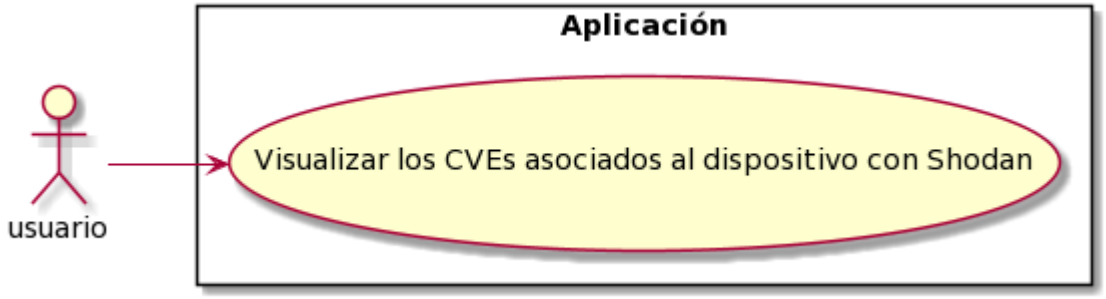
5.1.10. Visualizar resultados de ZoomEye

Figura 28: CU: Visualizar resultados de ZoomEye



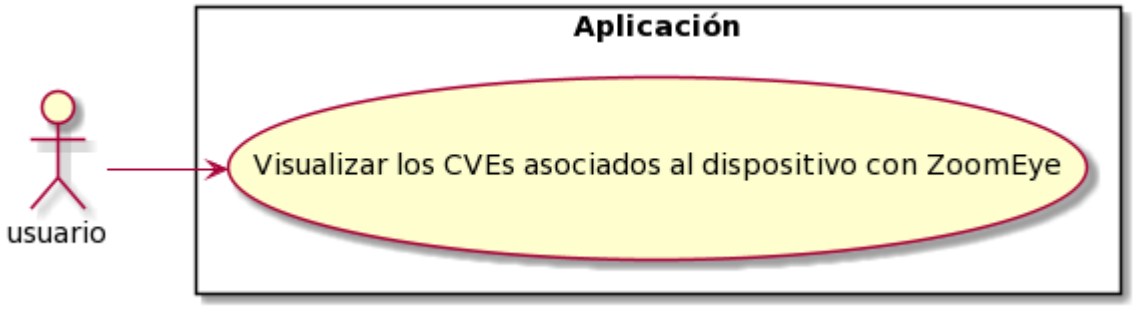
5.1.11. Visualizar CVEs aplicables al dispositivo con Shodan

Figura 29: CU: Visualizar CVEs aplicables al dispositivo con Shodan



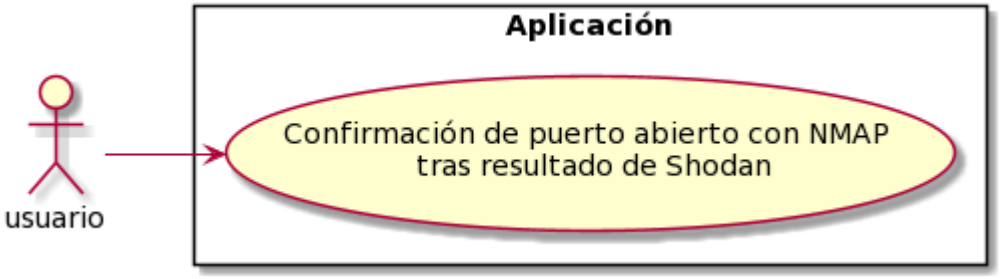
5.1.12. Visualizar CVEs aplicables al dispositivo con ZoomEye

Figura 30: CU: Visualizar CVEs aplicables al dispositivo con ZoomEye



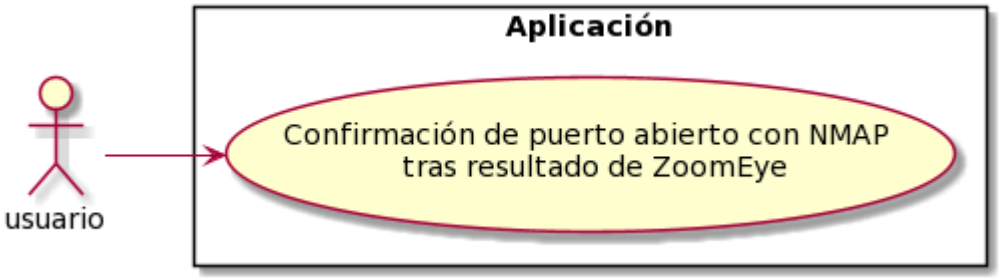
5.1.13. Confirmación de puerto abierto con NMAP tras resultado de Shodan

Figura 31: CU: Confirmación de puerto abierto con NMAP tras resultado de Shodan



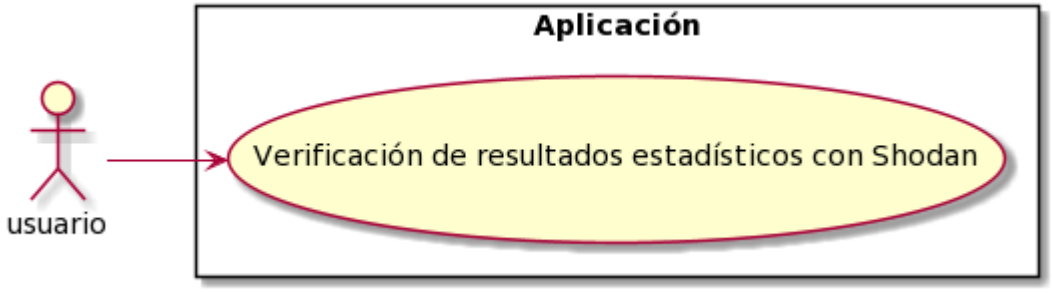
5.1.14. Confirmación de puerto abierto con NMAP tras resultado de ZoomEye

Figura 32: CU: Confirmación de puerto abierto con NMAP tras resultado en ZoomEye



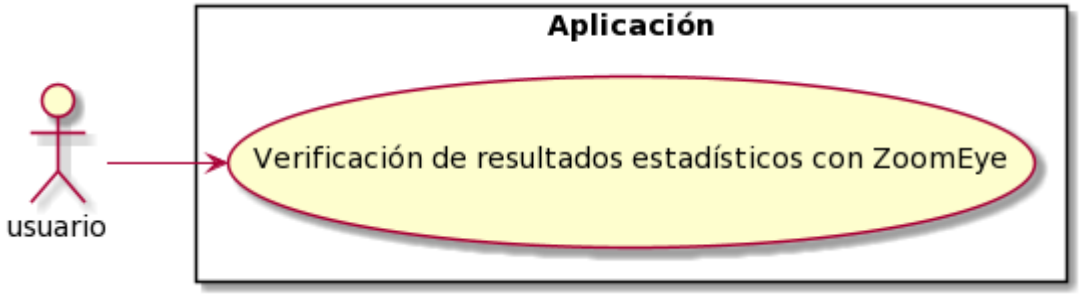
5.1.15. Verificación de resultados estadísticos de Shodan

Figura 33: CU: Verificación de resultados estadísticos de Shodan



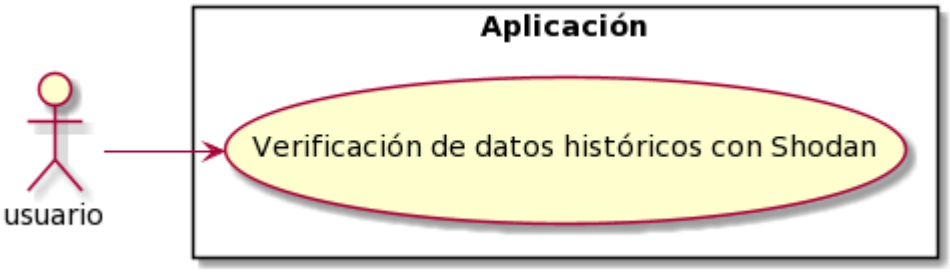
5.1.16. Verificación de resultados estadísticos de ZoomEye

Figura 34: CU: Verificación de resultado estadísticos con ZoomEye



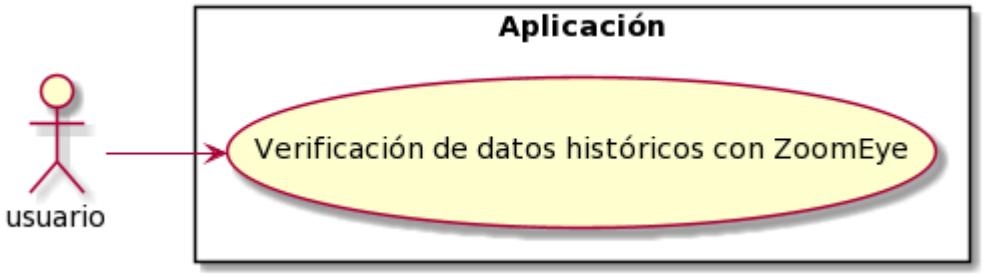
5.1.17. Verificación de datos históricos de Shodan

Figura 35: CU: Verificación de resultados estadísticos de Shodan



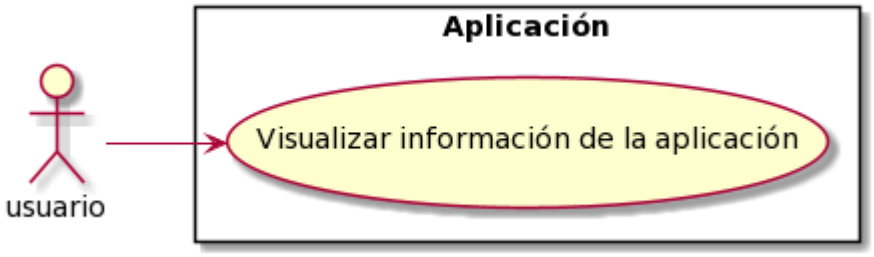
5.1.18. Verificación de datos históricos de ZoomEye

Figura 36: CU: Verificación de resultados estadísticos con ZoomEye



5.1.19. Visualizar información de la aplicación

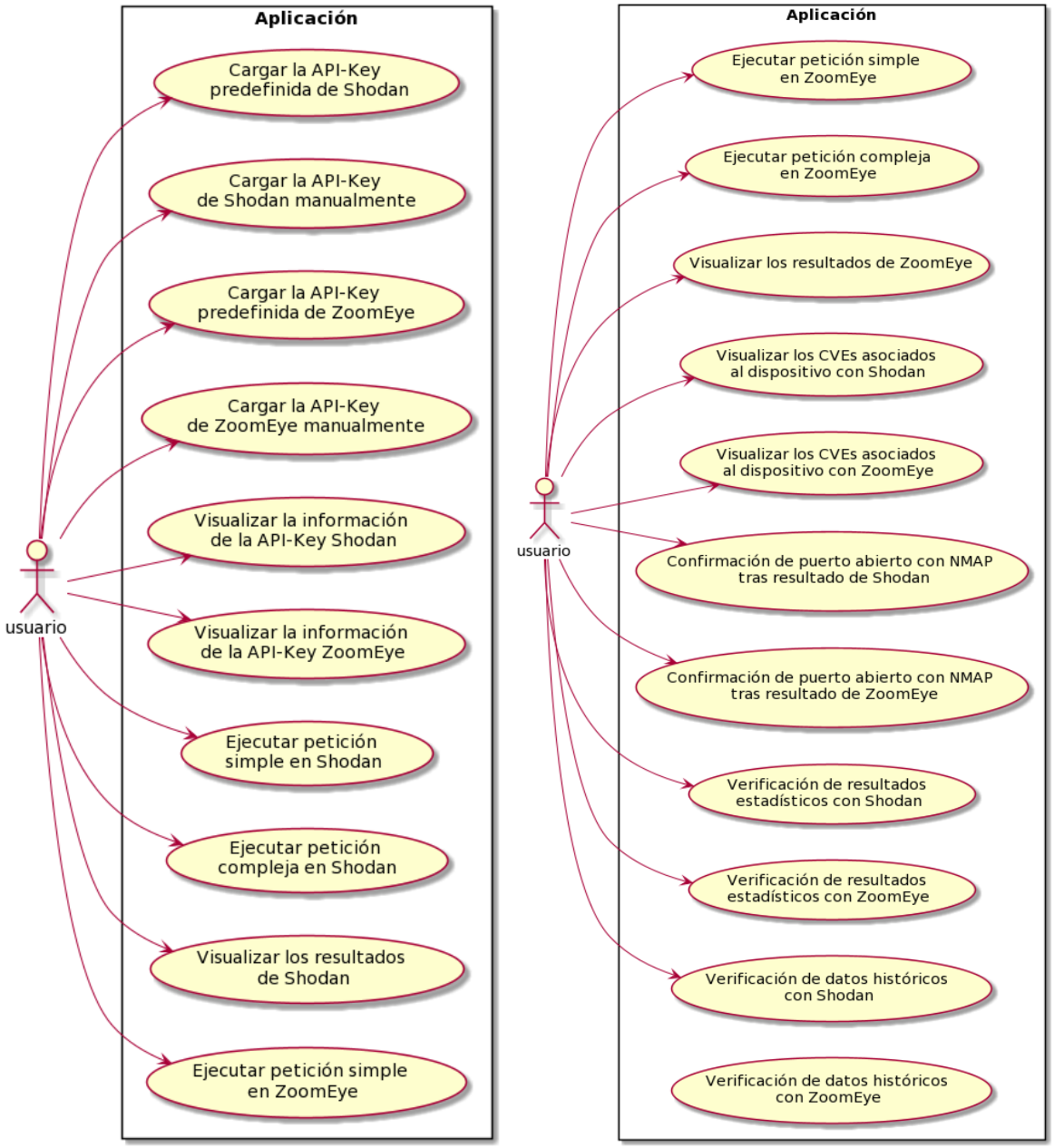
Figura 37: CU: Visualizar información de la aplicación



5.1.20. Diagrama completo de Casos de Uso

El siguiente diagrama muestra todos los casos de uso previamente presentados.

Figura 38: Diagrama de casos de uso completo



5.2. REQUISITOS FUNCIONALES Y NO FUNCIONALES

En este apartado se listan tanto los requisitos funcionales como los no-funcionales que se plantean para el desarrollo de DÉDALO.

En las tablas de los siguientes subapartados se dispone de estos campos:

- Código: un identificador del requisito.
- Nombre: nombre simple del requisito.
- Descripción: descripción más completa del requisito a cumplir.

- Prioridad: importancia o criticidad del requisito. Se tiene tres valores en función de la importancia: Alta, Media y Baja.

5.2.1. Tabla de requisitos Funcionales

La Tabla 2 hace referencia a los requisitos funcionales planteados. En el apartado *6.7 Consecución de requisitos funcionales de DÉDALO*, se presenta la Tabla 7 con la relación de los requisitos logrados tras la implementación y cierre de DÉDALO.

Tabla 2: Tabla de requisitos funcionales de la aplicación

Código	Nombre	Descripción	Prioridad
REQF01	Carga de la API-Key de Shodan.	La aplicación tendrá posibilidad de cargar la API-Key de Shodan.	Media
REQF02	Carga de la API-Key de ZoomEye.	La aplicación tendrá posibilidad de cargar la API-Key de ZoomEye.	Media
REQF03	Búsquedas en Shodan.	La aplicación deberá realizar búsquedas sobre la base de datos de Shodan.	Alta
REQF04	Búsquedas en ZoomEye.	La aplicación deberá realizar búsquedas sobre la base de datos de ZoomEye.	Alta
REQF05	Uso de la API de Shodan para Python.	La aplicación hará uso de la API de Shodan para el lenguaje de programación Python.	Media
REQF06	Uso de la API de ZoomEye para Python.	La aplicación hará uso de la API de ZoomEye para el lenguaje de programación Python.	Media
REQF07	Muestra de resultados de Shodan.	La aplicación mostrará los resultados de Shodan en la interfaz.	Alta
REQF08	Muestra de resultados de ZoomEye.	La aplicación mostrará los resultados de ZoomEye en la interfaz.	Alta
REQF09	Comparación de resultados.	La aplicación dispondrá de indicadores de comparación entre los resultados de Shodan y ZoomEye.	Media
REQF10	Búsqueda de exploits o CVEs asociados a los resultados de Shodan.	La aplicación será capaz de buscar exploits o CVEs asociados a los resultados de las búsquedas realizadas de Shodan.	Media
REQF11	Búsqueda de exploits o CVEs asociados a los resultados de ZoomEye.	La aplicación será capaz de buscar exploits o CVEs asociados a los resultados de las búsquedas realizadas de ZoomEye.	Media

5.2.2. Tabla de requisitos no-funcionales

La Tabla 3 hace referencia a los requisitos no-funcionales planteados. En el apartado *6.8 Consecución de requisitos no-funcionales de DÉDALO* se ha incluido la Tabla 8 con la relación de los requisitos logrados tras la implementación y cierre de DÉDALO.

Tabla 3: Tabla de requisitos no-funcionales de la aplicación.

Código	Nombre	Descripción	Prioridad
REQNF01	Aplicación de fácil ejecución.	La aplicación deberá ser de fácil ejecución. Es decir, el usuario tiene que ser capaz de ejecutar la aplicación sin un conocimiento alto de la materia.	Alta
REQNF02	Búsquedas con interfaz amigable.	La aplicación permitirá realizar búsquedas mediante una interfaz fácil de utilizar e intuitiva para el usuario.	Alta
REQNF03	Verificación de datos.	La aplicación será capaz de verificar los datos de entrada y su formato.	Media
REQNF04	Baja necesidad de cómputo.	La aplicación deberá ser capaz de ser ejecutadas en PCs estándares sin necesidad de un cómputo alto y sin que interfiera su ejecución sobre otros procesos ejecutados en el ordenador.	Baja
REQNF05	Facilidad de entender comparativas	La aplicación deberá ser suficientemente entendible y clara a la hora de mostrar comparativas entre las distintas herramientas.	Media

5.3. ENTORNO DE DESARROLLO UTILIZADO

5.3.1. Python

Se utiliza el lenguaje de programación Python para el desarrollo de DÉDALO. Fue creado por Guido van Rossum a finales de la década de los 80 («Python», 2021).

Se utiliza este lenguaje de programación por la legibilidad del código, el licenciamiento de código abierto y porque tanto Shodan como ZoomEye disponen de *APIs* para ser utilizados en este lenguaje de programación.

5.3.2. TKinter

Se utiliza el paquete *TKinter* (o "*tkinter*") para aportar la interfaz de usuario a la herramienta. Este paquete es la interfaz estándar de Python del *kit* de herramientas "Tcl/Tk". Está disponible en la mayoría de las plataformas de Unix, macOS y sistemas Windows (*tkinter — Interface de Python para Tcl/Tk — documentación de Python - 3.10.0*, s. f.). Dispone de varios tipos de elementos (etiquetas, campos de texto, botones, pestañas, paquete TreeView, etcétera).

5.3.3. API Shodan

La API de Shodan es uno de los dos componentes clave de DÉDALO, ya que es el interfaz mediante el cual Python se comunica con el repositorio de Shodan. No solamente permite acceder a la misma información que la web, ya que incluso se puede llegar a repositorios inaccesibles desde ella.

La librería oficial de Shodan es libremente accesible en GitHub, en la página www.github.com/achillean/shodan-python. Además de permitir las búsquedas sobre Shodan, permite sacar toda la información referente a un host completo, puede proporcionar información en tiempo real de los rastreos de Shodan, posee una API de búsqueda de vulnerabilidades e incluso un cliente en línea de comando. Para su utilización en Python, esta librería debe ser instalada mediante el comando:

```
pip3 install shodan
```

Para conectarse a la librería en un programa Python hay que importarla y después crear un objeto invocando el método "*shodan.Shodan()*", al que se le pasará una *API-Key* (clave de la API) válida como parámetro.

```
import shodan
my_shodan = shodan.Shodan("SHODAN_API_KEY")
```

Sobre ese objeto conectado a Shodan se pueden llamar a varios métodos:

- *search*: ejecuta una búsqueda que devuelve un diccionario de datos cuya primera entrada es el total de resultados y las siguientes son la información de los 100 primeros resultados encontrados. Un ejemplo de llamada a este método sería `my_shodan.search('webcam')`
- *host*: saca información completa de un host. A este método se le pasa la dirección IP. La llamada se realizaría como sigue: `my_shodan.host('xxx.xxx.xxx.xxx')`

La clase principal de la API es llamada "*Shodan*", que se inicializa con el parámetro "*key*" de tipo *str* (cadena de caracteres) y que hace referencia a una clave válida para hacer uso de la API. A partir de la página www.shodan.readthedocs.io/en/latest/api.html podemos obtener una clasificación de los atributos y los métodos que se pueden utilizar. En la Tabla 4 se pueden observar los métodos disponibles para la clase principal `Shodan.Shodan(KEY)`.

Tabla 4: Funciones de la clase principal "*Shodan*"

Método	Descripción
`alerts(aid, include_expired)`	Lista todas las alertas activas que el usuario ha creado.
`count(query, facets)`	Devuelve el número total de resultados devueltos por la petición realizada.
`create_alert(name, ip, expires)`	Crea una alerta para una IP concreta y aportando el nombre de la misma.
`delete_alert(aid)`	Elimina la alerta a partir del ID dado.
`host(ips, history, minify)`	Devuelve toda la información disponible para una IP.
`info()`	Devuelve información acerca de la API Key, así como otras características del tipo de cuenta.
`ports()`	Devuelve el listado de puertos que Shodan es capaz de buscar.
`protocols()`	Devuelve el listado de protocolos que Shodan es capaz de buscar.
`queries(page, sort, order)`	Devuelve un listado de peticiones (*queries*) compartidas por otros usuarios.
`queries_search(query, page)`	Busca las peticiones de búsquedas guardadas en el directorio de Shodan.
`scan(ips, force)`	Escanea una red.
`scan_internet(port, protocol)`	Escanea una red.

Método	Descripción
`scan_status(scan_id)`	Devuelve la información del estado de un escaneo previamente lanzado.
`search(query, page, limit, offset, facets, minify)`	Busca en la base de datos de Shodan según los parámetros dados.
`search_cursor(query, minify)`	Busca en la base de datos Shodan según una *query* y devuelve un objeto iterador que puede ser utilizado directamente en una función recurrente.
`search_tokens(query)`	Devuelve información acerca de los parámetros utilizados sobre la petición (*query*).
`services()`	Devuelve el listado de servicios que Shodan es capaz de buscar.

www.shodan.readthedocs.io/en/latest/api.html

Además, se cuenta con una clase llamada "*Exploits*" que permite buscar diferentes vulnerabilidades que tienen asociadas los dispositivos encontrados:

```
Class Exploits(parent)
```

Tabla 5: Funciones de la clase "*Exploits*" de la API de Shodan.

Método	Descripción
`count(query, facets)`	Devuelve el número total de *exploits* encontrados para una petición dada.
`search (query, facets, page)`	Busca los *exploits* a partir de unos parámetros dados.

www.shodan.readthedocs.io/en/latest/api.html

5.3.3.1. *Facets* en Shodan

Otra interesante característica de la que dispone Shodan es el uso de *facets*. Los *facets* son resúmenes de información de propiedades. Para ello, se debe utilizar un tipo de objeto concreto y se deben insertar en los métodos que permiten el uso de *facets*.

En primer lugar, se crea el listado de propiedades para las que se quiere el resumen:

```
FACETS = [ 'org', 'domain', 'port', 'asn', 'country']
```

En segundo lugar, se debe insertar este listado en los métodos en los que se permite utilizar *facets* (ver Tabla 4):

Por ejemplo, `count("os:linux", FACETS)`, en esta llamada, se van a listar los Top 5 valores más recurrentes en cuanto a organizaciones, dominios, puertos, sistemas autónomos y países que tengan Linux como sistema operativo. Se puede modificar la cantidad de valores "Top" a devolver pasando el siguiente listado:

```
FACETS = [ ('org', 3), 'domain', 'port', 'asn', 'country']
```

En ese caso, Shodan devolverá los 5 valores "Top" para todos los parámetros (por defecto), exceptuando el parámetro "*org*" (organizaciones) para el cual devolverá solamente los 3 valores más recurrentes.

Por último, es importante recalcar que no todos los atributos o filtros (sistema operativo, organización, ciudad, etc.) se pueden utilizar como *facets* para que Shodan devuelva estadísticas de todos los atributos. No se ha evidenciado en la documentación oficial qué filtros tienen asociados la posibilidad de tener este tipo de datos estadísticos.

5.3.4. API ZoomEye

La API de ZoomEye es el segundo componente clave que se utiliza en DÉDALO. A diferencia del caso de Shodan, la documentación que se ha encontrado sobre ella es escasa y principalmente está suministrada en la página oficial de la herramienta: www.zoomeye.org.

ZoomEye proporciona una API que puede ser utilizada desde Python. Hay dos formas de instalar la API en un sistema:

- Utilizando *pip*: `pip3 install zoomeye-python`
- Desde *github*: `pip3 install git+https://github.com/knownsec/zoomeye-python.git`

Existen dos formas de utilizar la API: la primera de ellas, mediante un cliente de consola o *cli* y la segunda forma, el *sdk zoomeye.sdk*, que es la opción que se ha utilizado en el presente trabajo debido a que permite integrar perfectamente la herramienta de búsqueda en DÉDALO.

Para utilizar el sdk en un programa Python hay que realizar la importación de la librería:

```
import zoomeye.sdk as ZoomEye
```

Previamente a la realización de consultas, debe definirse un objeto de tipo ZoomEye que necesita inicializarse mediante la operación de *login* para que posteriormente pueda ser utilizado en consultas. ZoomEye permite el uso del nombre de usuario y contraseña para el *login*, aunque no sea esta una opción recomendable porque se corre el riesgo de que las credenciales queden expuestas.

```
zm = zoomeye.ZoomEye()
zm.username = '****@*****.com'
zm.password = '********'
zm.login()
```

La forma más aconsejable de realizar el *login* es mediante el uso de la *API-Key* que tiene asignado el usuario. Este método ofrece mayor seguridad debido a que la *API-Key* no permite abrir sesión en z*oomeye.org* ni, por tanto, a acceder a la información fundamental de la cuenta del usuario.

Además, la *API-Key* se puede regenerar muy fácilmente desde el área de usuario en caso de cualquier actividad sospechosa. Con este método, la inicialización del objeto ZoomEye se realiza en una única línea:

```
zm = zoomeye.ZoomEye(api_key=" XXXXXXXX-XXXX-XXXXX-XXXX-XXXXXXXXXXX")
```

La Tabla 6 muestra los métodos que ofrece el sdk para un objeto de tipo ZoomEye, como muestra la propia documentación de ZoomEye en (*ZoomEye-python*, 2016/2021).:

Tabla 6: Métodos del objeto principal "zoomeye".

Método	Descripción
`login()`	Para autenticar una sesión utilizando usuario y contraseña

Método	Descripción
`dork_search(dork, page=0, resource="host", facets=None)`	Búsqueda en ZoomEye utilizando *dorks (campos de búsqueda). Devuelve los primeros 20 registros.*
`multi_page_search(dork, page=1, resource="host", facets=None)`	Búsqueda en ZoomEye que permite recuperar las diferentes páginas en las que se agrupan los resultados.
`resources_info()`	Devuelve la información del usuario que ha iniciado la sesión.
`show_count()`	Devuelve el número total de resultados que se obtuvieron en la última búsqueda.
`dork_filter(keys)`	Para extraer los datos del campo concreto que se pase como parámetro.
`get_facet()`	Para obtener resultados agregados de una búsqueda.
`history_ip(ip)`	Consulta de información histórica de una IP.
`show_site_ip(data)`	Devuelve el dominio y la dirección IP del conjunto de datos obtenido en la última consulta.
`show_ip_port(data)`	Devuelve la dirección IP y el puerto de los datos obtenidos.

Fuente: www.github.com/knownsec/zoomeye-python

5.3.4.1. *Facets* en ZoomEye

ZoomEye también tiene la opción de utilizar *facets* para obtener información estadística y resumida de las búsquedas que se lanzan. Para ello, se utiliza un mecanismo similar al de Shodan:

En primer lugar, se inicializa una variable de tipo lista, por ejemplo:

```
FACETS_ZOOMEYE = ['product', 'device', 'webapp', 'service', 'os',
'port', 'country', 'city']
```

Ese tipo de objeto se puede utilizar en varias funciones y en él se pueden incluir diferentes atributos en función del propio método al que se le llama. Es decir, no todas las funciones que permiten *facets* permiten los mismos o todos los atributos. La relación completa de funciones que permiten *facets* y los atributos permitidos para cada método están en la documentación de ZoomEye: www.zoomeye.org/doc#zoomeyepy.

5.3.5. NMAP

En el proyecto también se utiliza la herramienta *Network Mapper*, también conocida como NMAP que se puede utilizar para realizar escaneo de puertos sobre una IP o red completa.

Existe una cierta discusión acerca de la legalidad del uso de NMAP y más concretamente acerca de si es legal o no realizar escaneos de puertos. Sin embargo, el uso de esta herramienta es extenso entre la comunidad TIC, y más en concreto la comunidad de ciberseguridad y especialmente en *hacking*.

En el caso concreto del trabajo realizado, se utiliza NMAP a modo de verificación o chequeo del estado de un puerto concreto para una IP dada en el prototipo desarrollado. Para ello, se utiliza la API en un punto concreto de la aplicación (*6.1.6 Botón de chequeo de Nmap)*.

5.3.6. Otras herramientas utilizadas

A parte de las herramientas que ya se han listado, a continuación, se listan algunas otras herramientas para completar el trabajo:

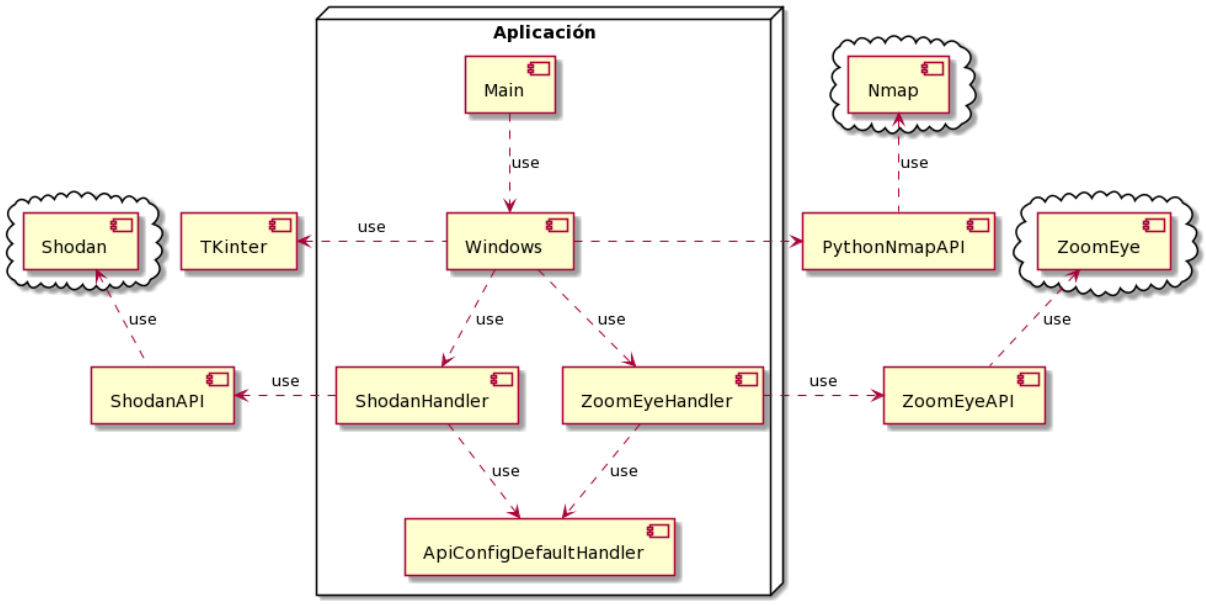
- **PyCharm Community Edition 2021.2**: entorno de desarrollo realizado por la empresa Jetbrains, específico para el lenguaje Python. Contiene muchas utilidades que facilitan el desarrollo de código en Python y se integra con GitHub, simplificando la actualización y consulta del repositorio de programas. Se ha utilizado la *Community Edition*, que es gratuita, existiendo una versión de pago que tiene características adicionales, la *Professional Edition*.
- **GitHub**: repositorio reconocido en la industria que se utilizará para almacenar el código y llevar un control de versiones adecuado.
- **PlantUML**: herramienta de código abierto que permite realizar diagramas de varios tipos de forma ágil y haciendo uso de un lenguaje de etiquetas. Se puede integrar con

varias herramientas, aunque también dispone de un servidor online para crear los diagramas. Mediante PlantUML se realizarán los diagramas para la documentación del programa: diagrama de casos de uso, diagramas de clase, diagramas de secuencia, diagrama de Gantt, etcétera. Página web: www.plantuml.com.

5.4. DIAGRAMA DE COMPONENTES

En el siguiente diagrama se muestra la relación de componentes utilizados y la arquitectura alto nivel desplegada.

Figura 39: Diagrama de componentes de DÉDALO



El diagrama de la figura superior muestra, a grandes rasgos, los componentes más importantes de la aplicación.

DÉDALO de forma interna tiene cinco componentes: el componente *Main* que lanza la aplicación y utiliza el componente que gestiona las ventanas de la aplicación, llamado *Window*. Esta última utiliza los componentes diseñados para administrar las conexiones con las APIs de Shodan y ZoomEye, respectivamente *ShodanHandler* y *ZoomEyeHandler*. Estos componentes de gestión de las herramientas de seguridad utilizan los recursos externos de cada una de las herramientas, *Shodan API* y *ZoomEye API*, que a su vez hacen uso de la nube para devolver los datos. Eso mismo pasa con la aplicación NMAP, que es accedida desde DÉDALO haciendo uso de la API que esta ofrece.

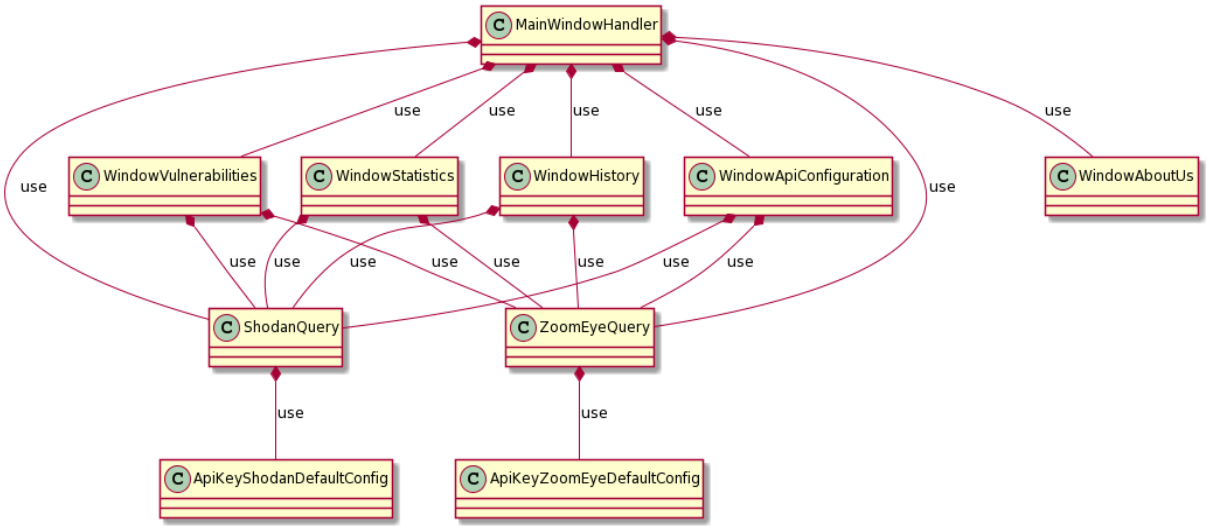
Además, el componente interno de la aplicación *ApiConfigDefaultHandler* es utilizado por parte de los componentes de gestión de Shodan y ZoomEye (*ShodanHandler* y *ZoomEyeHandler*) para administrar las claves de las APIs en cada caso.

El diagrama mostrado describe la arquitectura a alto nivel, por lo tanto, los componentes que aparecen en él se descomponen en diferentes clases o módulos en el momento de la codificación. En los siguientes subapartados se sigue desgranando DÉDALO a más bajo nivel.

5.5. DIAGRAMA DE CLASES

A continuación, se muestra el diagrama de clases del código de la aplicación. En el apartado *5.6 Composición de DÉDALO: módulos y librerías externas* se desglosan y profundizan en mayor grado cada una de las clases que componen el código de la aplicación.

Figura 40: Diagrama de clase de DÉDALO



El diagrama de clases de la figura superior muestra las clases más importantes de DÉDALO y la interacción entre sí. Hay algunos extractos de código en los que no se ha utilizado el concepto de "Clase" para realizar alguna función (por ejemplo, el uso de NMAP). Por ello, no se ha mostrado en la figura superior.

Por otra parte, estas clases han sido codificadas en diferentes archivos Python (formato "*.py*") y en ocasiones incluso se ha utilizado el mismo archivo para albergar más de una clase (por ejemplo, el archivo "*api_config_default_handler.py*" que alberga las clases *ApiKeyShodanDefaultHandler* y *ApiKeyZoomEyeDefaultHandler*. Es decir, no hay una relación uno a uno (1…1) entre las clases y los módulos o ficheros que se desglosan en el siguiente apartado. Se ha implementado de esta manera para tratar de lograr un código legible y

reutilizable, dentro de la complejidad que puede llegar a suponer la lectura de un código software ajeno.

5.6.COMPOSICIÓN DE DÉDALO: MÓDULOS Y LIBRERÍAS EXTERNAS

El código de DÉDALO se ha repartido en trece módulos, que agrupan las clases y funciones según su utilidad. Hay un módulo principal simple, llamado *main*, que se encarga de lanzar el interfaz del programa y los otros doce se encargan de la gestión de las diferentes ventanas, las clases para el acceso a las APIs, la definición de las constantes y otras utilidades.

En los siguientes subapartados se han añadido algunas representaciones de las clases diseñadas usando lenguaje UML. Los diagramas de clase contienen todos los métodos y funciones, pero solamente se explican aquellas funciones y métodos que se han considerado relevantes para el entendimiento del funcionamiento de DÉDALO.

5.6.1. Módulo "*main.py*"

Es el módulo que lanza la aplicación. Como se observa en la Figura 41, este módulo simplemente importa la clase *MainWindowHandler*, usada por la función *main()* para montar el interfaz gráfico de DÉDALO.

Figura 41: Código del método *main()*

```
from window import MainWindowHandler

def main():
    window = MainWindowHandler()
    window.window.mainloop()

main()
```

5.6.2. Módulo "*constants.py*"

Contiene la definición de una clase que tiene como atributos un conjunto de constantes con todos los valores por defecto que se utilizan para armar la interfaz gráfica. Entre esas constantes se han metido los siguientes tipos de datos:

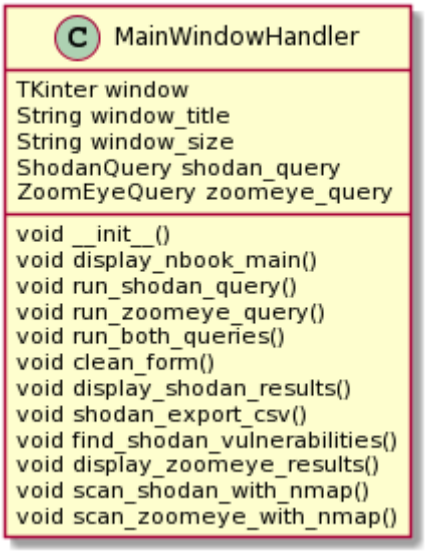
- Título de la pantalla principal.
- Tamaño de la pantalla principal.
- Tamaño de los tipos de campos.
- Texto de las etiquetas de los campos.

- Texto de la pestaña "*About Us*".

Para acceder a las variables, en DÉDALO se importa el módulo con el alias "*var*" (*import constants as var*). El objeto "*var*" tiene como atributo todas las constantes definidas en el módulo.

5.6.3. Módulo "*window.py*"

Figura 42: Diagrama de clase de *MainWindowHandler* con atributos y métodos



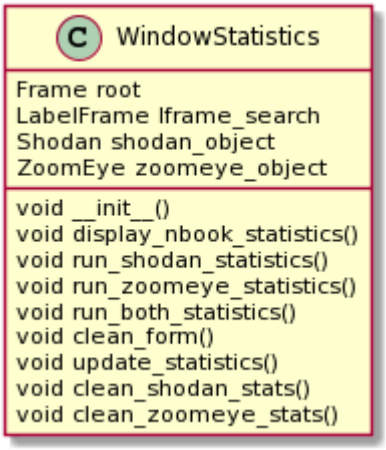
Este módulo contiene la clase *MainWindowHandler*. Su constructor genera la ventana con todas sus pestañas, inicializa con los valores de las claves-API los objetos Shodan y ZoomEye y carga el contenido de la pestaña principal, llamada "*Queries & Findings"*. Como métodos más importantes podemos destacar las siguientes:

- *run_shodan_query()*: lanza una consulta en la plataforma Shodan conforme a los valores que se han introducido en los campos de los filtros de búsqueda.
- *run_zoomeye_query()*: lanza la consulta en ZoomEye a partir de los valores que se hayan introducido en los campos de los filtros de búsqueda.
- *display_shodan_results()*: recorre los resultados devueltos por la función *run_shodan_query()* y muestra los valores de algunos campos en un objeto de tipo *Treeview*.
- *display_zoomeye_results()*: recorre los resultados devueltos por la función *run_zoomeye_query()* y muestra los valores de algunos campos en un objeto de tipo *Treeview*.

- *find_shodan_vulnerabilities()*: realiza una búsqueda de vulnerabilidades del registro seleccionado en la tabla de resultados de la búsqueda de Shodan. Muestra los resultados mediante la clase *WindowVulnerabilities*, definida en el módulo *window_vulnerabilities*.
- *scan_shodan_with_nmap()*: Realiza una consulta Nmap del registro seleccionado de la tabla de resultados de Shodan. Para ello llama a la función *scan_ip_port()* del módulo *nmap_handler*. Actualiza el registro de resultados con el estado del puerto devuelto por Nmap.
- *scan_zoomeye_with_nmap()*: Realiza una consulta Nmap del registro seleccionado de la tabla de resultados de ZoomEye. Para ello llama a la función *scan_ip_port()* del módulo *nmap_handler*. Actualiza el registro de resultados con el estado del puerto devuelto por Nmap.

5.6.4. Módulo "*window_statistics.py*"

Figura 43: Diagrama de clase de *WindowStatistics* con atributos y métodos



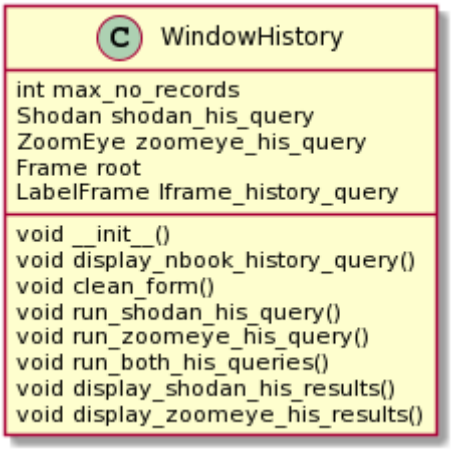
Es el módulo que gestiona la pestaña con la información de las búsquedas estadísticas, mediante la clase *WindowStatistics*. Su constructor genera los objetos de tipo Shodan y ZoomEye y presenta en pantalla los campos de filtro de búsqueda, los botones para lanzar las consultas y el marco con los resultados, incluyendo el *checkbox* donde el usuario puede elegir el tipo de estadística que desea visualizar. Sus métodos principales son:

- *run_shodan_statistics()*: lanza la consulta estadística en Shodan a partir de los valores utilizados en los filtros.

- *run_zoomeye_statistics()*: lanza la consulta estadística en ZoomEye a partir de los valores utilizados en los filtros.
- *update_statistics()*: agrupa los resultados de las consultas realizadas en Shodan y ZoomEye según el campo indicado por el usuario en el *checkbox*.

5.6.5. Módulo "*window_history.py*"

Figura 44: Diagrama de clase de *WindowHistory* con atributos y métodos

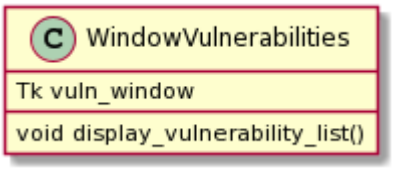


El módulo contiene la definición de la clase *WindowHistory* encargada de la pestaña de búsquedas históricas. Sus métodos principales son:

- *run_shodan_his_query()*: ejecuta sobre Shodan la consulta que busca todos los puertos de un *host* que en algún momento hayan sido detectados.
- *run_zoomeye_his_query()*: ejecuta sobre ZoomEye la consulta que busca todos los cambios en la información sobre un host que ha recogido la plataforma a lo largo de la historia. Desafortunadamente, no ha sido posible hacerla funcionar debido a que la clave-API utilizada no disponía de los suficientes privilegios.
- *display_shodan_his_results()*: presenta en un objeto de tipo Treeview la información devuelta por el método *run_shodan_his_query().*
- *display_zoomeye_his_results()*: presenta en un objeto de tipo Treeview la información devuelta por el método *run_zoomeye_his_query().*

5.6.6. Módulo "*window_vulnerabilities.py*"

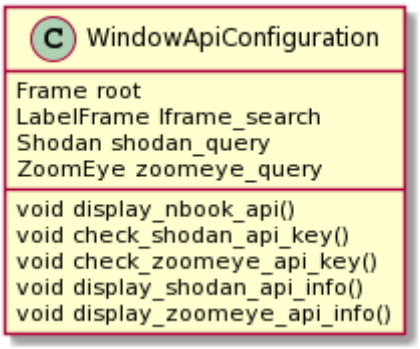
Figura 45: Diagrama de clase de *WindowVulnerabilities* con atributos y métodos



El módulo contiene la definición de la clase *WindowVulnerabilities*. A su constructor se le pasan los datos de uno de los resultados de la consulta Shodan. La clase muestra los resultados en una ventana independiente mediante el método *display_vulnerability_list()*.

5.6.7. Módulo "*window_api_configuration.py*"

Figura 46: Diagrama de clase de *WindowApiConfiguration* con atributos y métodos

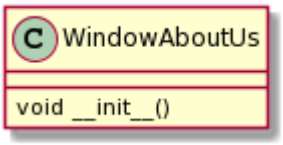


En este módulo se define la clase *WindowApiConfiguration*, utilizada para la ventana en la que se muestran las claves de las APIs guardadas en el archivo config.cfg. Cuenta con métodos para comprobar las claves y mostrarlas en pantalla:

- *check_shodan_api_key()*: lanza la consulta de la validez de la clave de la API de Shodan.
- *check_zoomeye_api_key()*: lanza la consulta de la validez de la clave de la API de ZoomEye.
- *display_shodan_api_info()*: muestra información en pantalla de la cuenta asociada a la clave de la API de Shodan.
- *display_zoomeye_api_info()*: muestra información en pantalla de la cuenta asociada a la clave de la API de ZoomEye.

5.6.8. Módulo "*window_about_us.py*"

Figura 47: Diagrama de clase de *WindowAboutUs* con atributos y métodos



Este módulo contiene la clase *WindowAboutUs*, que se encarga de la pestaña en la que se muestra información sobre el presente proyecto y sus autores.

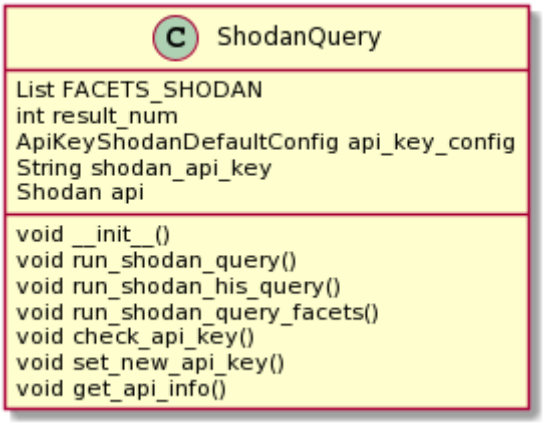
5.6.9. Módulo "*shodan_handler.py*"

Este módulo contiene el código con el que se hace la conexión con la API de Shodan. Tiene una función independiente, *shodan_query_creator()*, que monta el comando de búsqueda con la sintaxis adecuada a partir de los valores introducidos en los campos de filtro.

El componente principal es la clase *ShodanQuery* en cuya construcción genera la conexión a Shodan mediante la clave de la API. Los principales métodos de que consta son:

- *run_shodan_query()*: ejecuta la consulta de Shodan a partir de la consulta que genera la función *shodan_query_creator().*
- *run_shodan_his_query()*: ejecuta la consulta que devuelve los datos históricos que contiene Shodan de un host.
- *run_shodan_query_facets()*: ejecuta la consulta con estadísticas.
- *check_api_key()*: método que verifica la validez de una clave de API.

Figura 48: Diagrama de clase de *ShodanQuery* con atributos y métodos



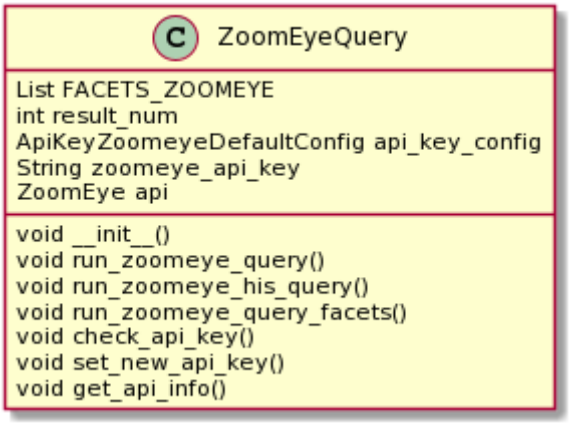
5.6.10. Módulo "*zoomeye_handler.py*"

Este módulo contiene el código con el que se hace la conexión con la API de ZoomEye. Tiene una función independiente, *zoomeye_query_creator()*, que monta el comando de búsqueda con la sintaxis adecuada a partir de los valores introducidos en los campos de filtro.

El componente principal es la clase *ZoomeyeQuery* en cuya construcción genera la conexión a ZoomEye mediante la clave de la API. Los principales métodos de que consta son:

- *run_zoomeye_query()*: ejecuta la consulta de ZoomEye a partir de la consulta que genera la función *zoomeye_query_creator().*
- *run_zoomeye _his_query()*: ejecuta la consulta que devuelve los datos históricos que contiene ZoomEye de un host.
- *run_zoomeye _query_facets()*: ejecuta la consulta con estadísticas.
- *check_api_key()*: método que verifica la validez de una clave de API.

Figura 49: Diagrama de clase de *ZoomEyeQuery* con atributos y métodos



5.6.11. Módulo "*nmap_handler.py*"

Contiene la definición de la función *scan_ip_port()* a la que se le pasan la IP y el puerto a interrogar y devuelve el estado del puerto obtenido por un comando Nmap.

5.6.12. Módulo "*api_config_default_handler.py*"

Contiene las clases *ApiKeyShodanDefaultConfig* y *ApiKeyZoomeyeDefaultConfig* que se utilizan para comprobar la validez de las claves de las APIs de Shodan y ZoomEye, respectivamente.

Figura 50: Diagrama de clase de *ApiKeyShodanDefaultConfig* con atributos y métodos

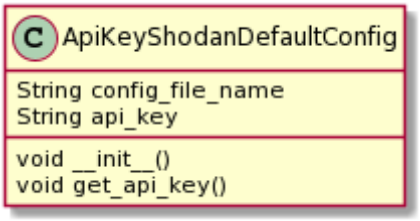
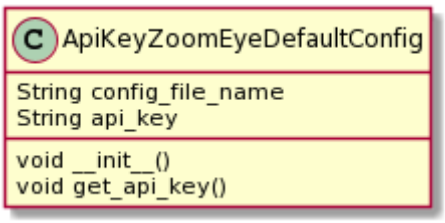


Figura 51: Diagrama de clase de *ApiKeyZoomEyeDefaultConfig* con atributos y métodos



5.6.13. Módulo "*utils.py*"

Este módulo está pensado para contener la definición de funciones con utilidades diversas que pueden ser utilizados en diversos puntos de DÉDALO. Contiene la función *format_number()* que añade el separador de miles en valores numéricos.

5.6.14. Librerías externas más relevantes

Se han utilizado las siguientes librerías externas:

- *Tkinter*: es el interfaz gráfico por defecto que ofrece Python.
- Shodan: es la librería que permite acceder a la API de Shodan y lanzar consultas sobre esta plataforma. Para su utilización es necesario contar con un usuario registrado, mediante el que se consigue la clave API necesaria para realizar la conexión. Algunas funciones están limitadas según el tipo de usuario que se utilice. En este trabajo se ha utilizado un usuario gratuito de tipo educativo, que tiene más privilegios que uno básico. Para obtener este usuario, hay que solicitarlo mediante el envío un email a la dirección academic@shodan.io, en el que quede acreditada la condición de estudiante del remitente.
- *Zoomeye.sdk*: es la librería que contiene las funciones para acceder a ZoomEye. Al igual que en el caso de Shodan requiere una clave API que se obtiene mediante registro. En este trabajo se ha utilizado un usuario básico y gratuito, que permite la utilización únicamente de las funcionalidades más simples.

- *Nmap*: librería que permite utilizar funciones para hacer escaneos con Nmap utilizando Python. Se ha utilizado para hacer escaneos adicionales sobre los resultados devueltos por Shodan y ZoomEye.
- *Configparser*: librería para la gestión de archivos de configuración. Facilita la creación, lectura y manipulación de archivos que guardan los datos globales de la aplicación.
- *IPy*: ofrece una clase y herramientas para realizar operaciones con direcciones IP y direccionamientos de red. En DÉDALO se ha utilizado para descomponer un direccionamiento de red en las direcciones IP individuales que lo conforman.
- *Json*: librería para el manejo de archivos con formato JSON. En este caso se ha utilizado para guardar fácilmente los resultados de las consultas a Shodan y ZoomEye, ya que devuelven los datos en este formato.
- *Time*: librería con funciones de manejo de tiempo. En este proyecto ha servido para calcular la duración de las consultas a Shodan y ZoomEye.

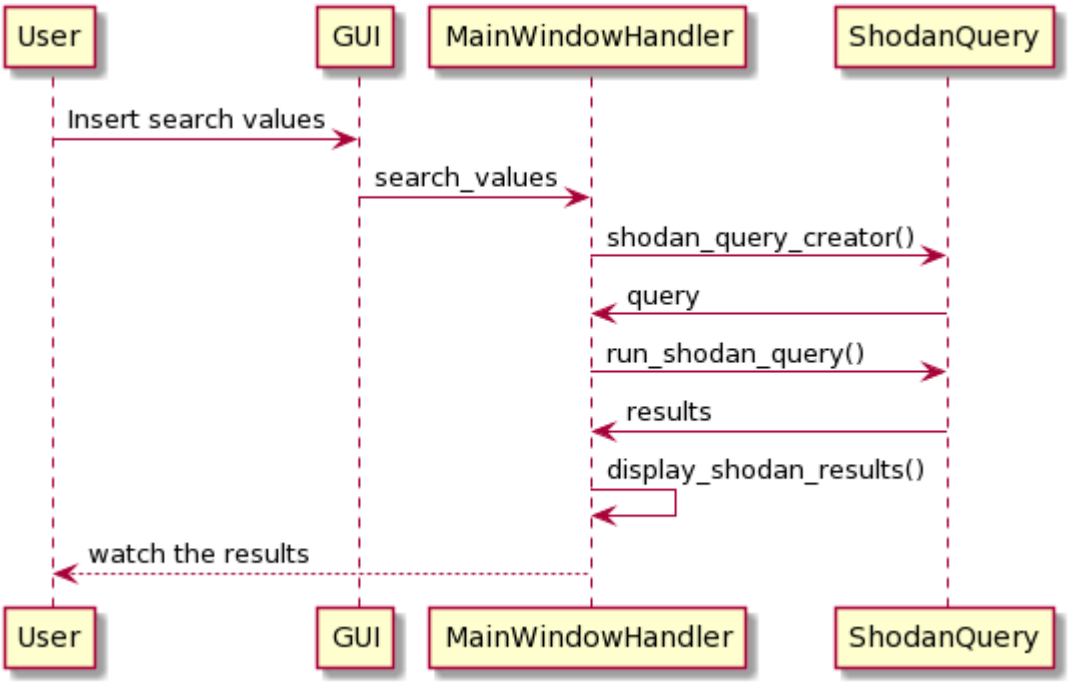
5.7. DIAGRAMAS DE SECUENCIA

En este apartado se presentan tres diagramas de secuencia para cuatro casos de uso de DÉDALO. Únicamente se han desarrollado tres diagramas de secuencia para mostrar de forma simple el funcionamiento a modo de ejemplo, sin entrar a desarrollar los otros muchos diagramas ya que no aportarían demasiado valor a la presente documentación.

Se utilizan en todos ellos la misma plantilla base, formada por todas las entidades y componentes de DÉDALO para dar una visión más clara de cómo está formada la misma.

5.7.1. Secuencia: Realizar búsquedas simples o complejas en Shodan

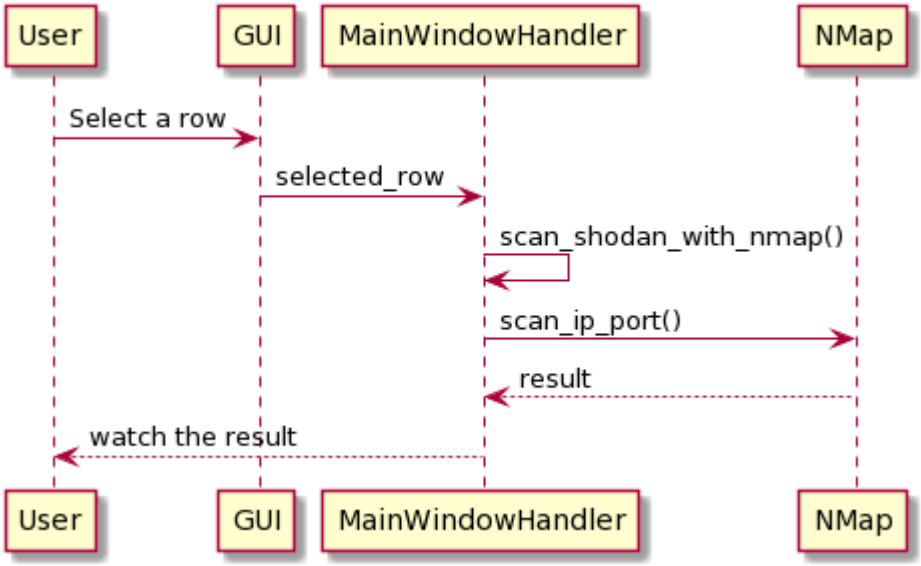
Figura 52: DS: Realizar búsquedas simples o complejas en Shodan



Esta misma secuencia sería aplicable a las búsquedas, tanto simples como complejas, de ZoomEye, ya que siguen el mismo patrón secuencial.

5.7.2. Secuencia: Confirmación de puerto abierto con NMAP

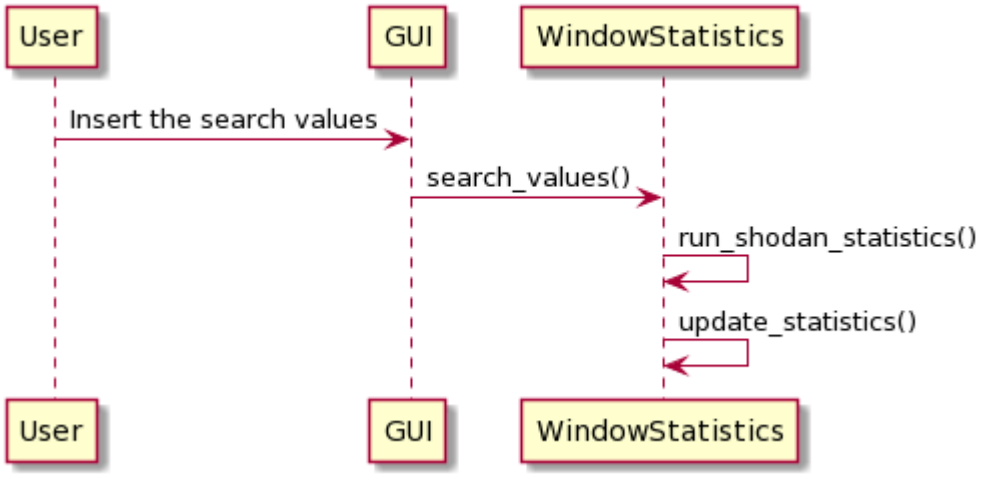
Figura 53: DS: Confirmación de puerto abierto con NMAP tras resultado de Shodan



Esta misma secuencia sería aplicable a la confirmación de puerto abierto con NMAP tras el resultado de ZoomEye, ya que siguen el mismo patrón de secuencia.

5.7.3. Secuencia: Verificación de resultados estadísticos de Shodan

Figura 54: DS: Verificación de resultados estadísticos con Shodan

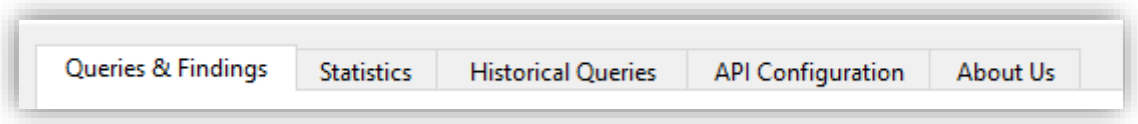


Esta misma secuencia sería aplicable a la verificación de resultados estadísticos de ZoomEye, incluso en el lanzamiento de resultados para ambas herramientas.

6. ANÁLISIS DE DÉDALO

La aplicación está dividida en cinco secciones, a las que se puede acceder mediante las pestañas que aparecen en el programa principal, como se ve en la Figura 55. En este capítulo se explica cada una de esas pestañas, además de explicar el contenido del fichero de configuración "*config.cfg*".

Figura 55: Pestañas de la pantalla principal



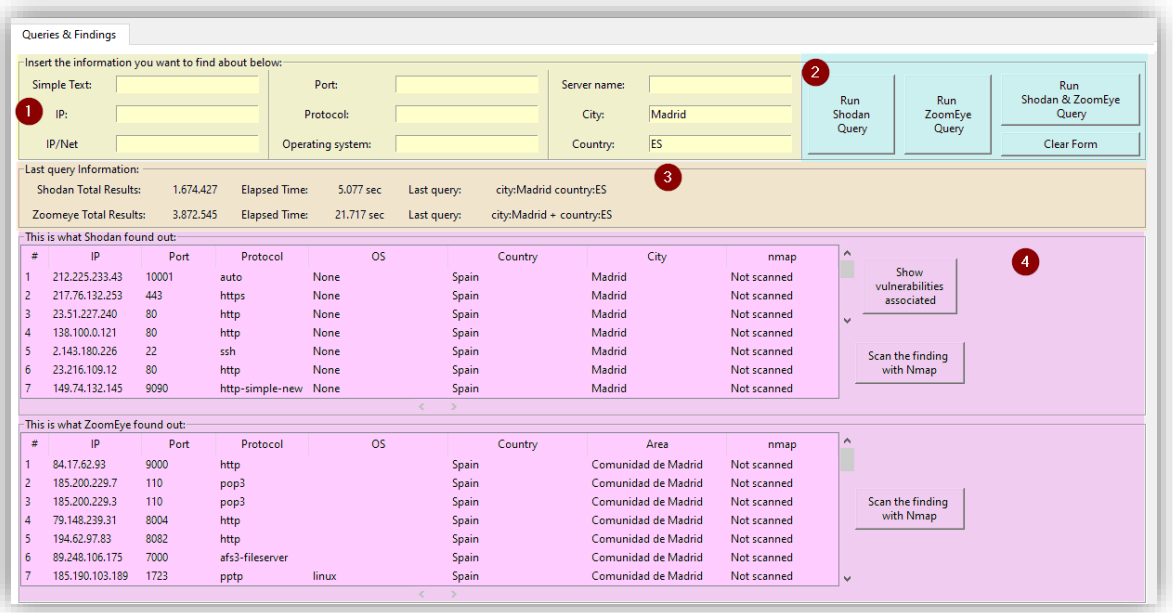
El idioma utilizado en DÉDALO, tanto en el desarrollo como en las interfaces de usuario, es el inglés, con el objetivo de que sea accesible a una comunidad más grande y pueda ser reutilizado y permita desarrollos evolutivos por parte de un número mayor de desarrolladores.

6.1. PANTALLA DE BÚSQUEDAS

Como se observa en la Figura 56, la pantalla de búsquedas tiene cuatro partes diferenciadas:

- Zona de introducción de criterios de búsqueda (1).
- Botones de búsqueda (2).
- Resumen de la última búsqueda realizada (3).
- Zona de resultados (4).

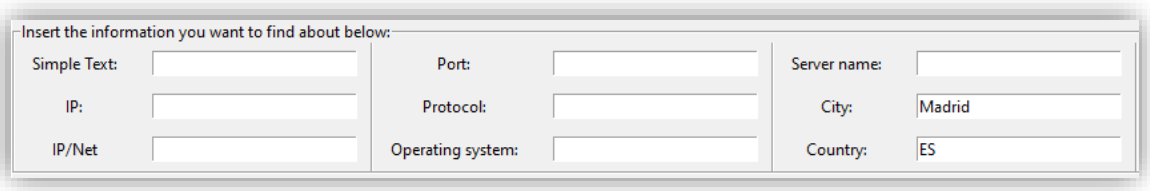
Figura 56: Vista de la pestaña de búsquedas y sus partes



6.1.1. Zona de introducción de criterios de búsqueda

En esta zona se pueden introducir criterios para delimitar la búsqueda de dispositivos. Se han seleccionado ocho criterios diferentes y uno más de uso general (Figura 57).

Figura 57: Zona de criterios de búsqueda



El primer campo, denominado "*Simple Text",* permite utilizar condiciones de búsqueda simples o con filtros. En el resto de los campos hay que introducir los valores de los filtros seleccionados. En *"IP"* una dirección concreta, *"IP/Net"* una red con su máscara, *"Port"* para buscar puertos, *"Protocol"* para búsquedas de protocolos, *"Operating system"* para buscar por sistema operativo, "*Server name*" para filtrar por nombre de servidor y *"City"* y *"Country"* para encontrar resultados de ciudades y países concretos, respectivamente.

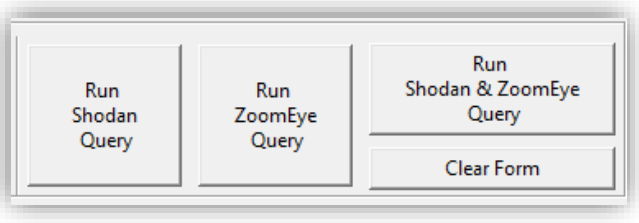
Los países se deben introducir mediante su código ISO 3166-1 alfa-2 (www.iso.org/iso-3166-country-codes.html). La ejecución de la búsqueda se realiza uniendo todos los filtros con el operador "Y". En Shodan se hace separando los filtros por un espacio en blanco y en ZoomEye

mediante el símbolo "+". De esta forma, para buscar los registros de la ciudad "Madrid" y país "España", en el primer caso pasaríamos "city:Madrid country:ES" y en el segundo "city:Madrid+country:ES"

6.1.2. Botones de búsqueda.

Junto a la zona de filtrado, hay cuatro botones (Figura 58). Además de "*Clear Form*", que limpia todos los campos de filtrado, el resto de los campos permiten ejecutar consultas en Shodan y ZoomEye a la vez *"Run Shodan & ZoomEye Query"* o en cada plataforma de forma independiente *"Run Shodan Query"* y "*Run ZoomEye Query"*

Figura 58: Botones de la pantalla de búsquedas

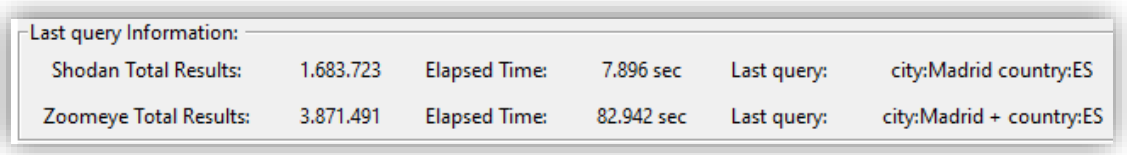


6.1.3. Resumen de la última búsqueda realizada

La Figura 59 muestra la zona de resumen de las búsquedas. Por cada plataforma se muestra el número de resultados de la última búsqueda realizada, el tiempo que ha tardado la búsqueda y el filtro tal y como se le ha pasado a Shodan o a ZoomEye.

El tiempo transcurrido ("*Elapsed Time"*) lo calcula la propia aplicación, comparando el *timestamp* del lanzamiento de la consulta y el del final.

Figura 59: Zona resumen de la última búsqueda



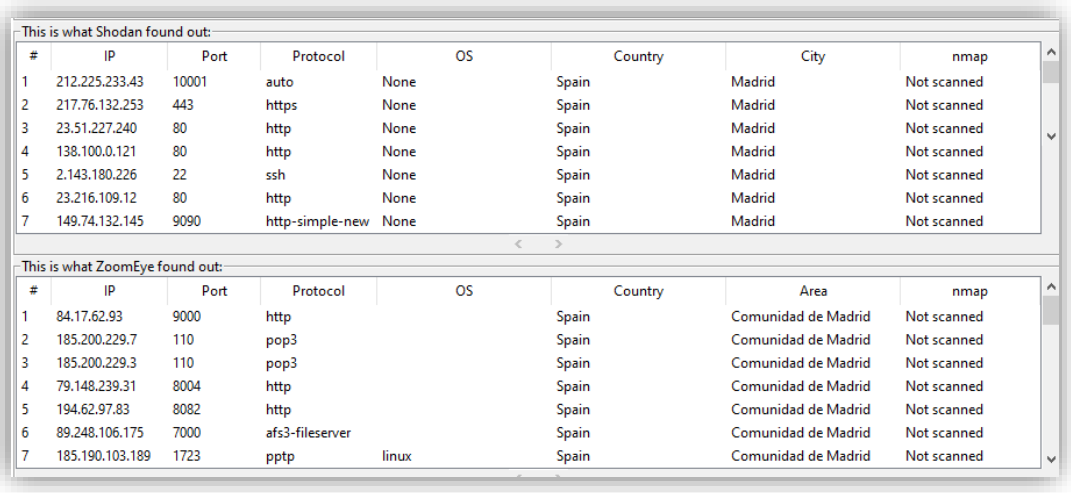
6.1.4. Zona de resultados

Los resultados de Shodan y ZoomEye se muestran en sendas tablas, como puede observarse en la Figura 60. Cada una puede tener un número máximo de 100 resultados. Los resultados se muestran en el mismo orden en el que cada plataforma devuelve los resultados. Las

columnas de información mostradas son IP, puerto, protocolo, sistema operativo, país y ciudad o área. ZoomEye no devuelve en un campo fijo la ciudad donde ubica el dispositivo, así que se ha optado por mostrar el área.

El último campo de cada tabla es el resultado de la verificación del estado del puerto utilizando Nmap. Por defecto no se hace la comprobación con Nmap, ya que por cada puerto abierto se incrementaría el tiempo de búsqueda en casi 20 segundos y, además, porque Nmap lanza comandos que generan tráfico directo contra los servidores analizados.

Figura 60: Zona de resultados

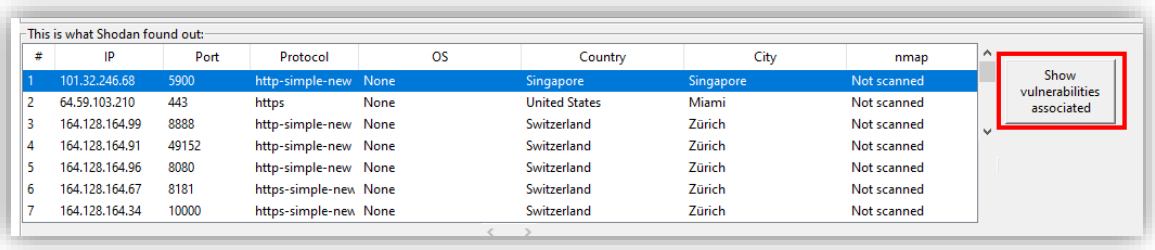


Además, DÉDALO crea un archivo de formato JSON cada vez que se realiza una búsqueda con cualquiera de las dos herramientas, Shodan y ZoomEye. Estos ficheros se crean en dos carpetas del código fuente llamadas "*results_shodan*" y "*results_zoomeye*". Esto se ha implementado con el objetivo de poder analizar en bruto los resultados obtenidos en caso de ser necesario.

6.1.5. Botón de vulnerabilidades

Shodan dispone de una base de datos de vulnerabilidades ligada a algunos dispositivos. Se ha añadido a DÉDALO una función para realizar la búsqueda de vulnerabilidades a partir de los resultados de la última consulta. Como se puede observar en la Figura 61, para lanzar la búsqueda de vulnerabilidades hay que seleccionar un registro y pulsar el botón "*Show vulnerabilities*".

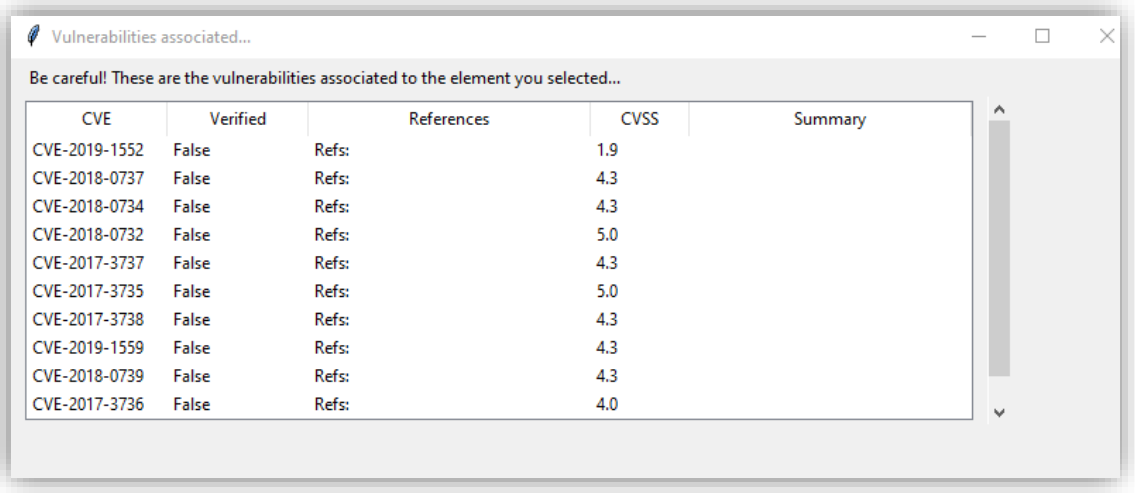
Figura 61: Botón de búsqueda de vulnerabilidades a partir de los resultados



#	IP	Port	Protocol	OS	Country	City	nmap
1	101.32.246.68	5900	http-simple-new	None	Singapore	Singapore	Not scanned
2	64.59.103.210	443	https	None	United States	Miami	Not scanned
3	164.128.164.99	8888	http-simple-new	None	Switzerland	Zürich	Not scanned
4	164.128.164.91	49152	http-simple-new	None	Switzerland	Zürich	Not scanned
5	164.128.164.96	8080	http-simple-new	None	Switzerland	Zürich	Not scanned
6	164.128.164.67	8181	https-simple-new	None	Switzerland	Zürich	Not scanned
7	164.128.164.34	10000	https-simple-new	None	Switzerland	Zürich	Not scanned

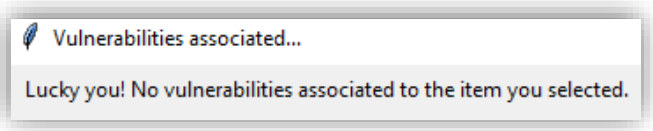
Si la búsqueda encuentra resultados, se obtendrá una ventana como la mostrada en la Figura 62, en la que encontraremos el código CVE de la vulnerabilidad y la puntuación asignada por CVSS. También se informa si la vulnerabilidad está verificada o no. Por el contrario, si Shodan no encuentra vulnerabilidades asociadas al resultado, se abrirá una ventana como la mostrada en la Figura 63.

Figura 62: Búsqueda de vulnerabilidades con resultados



CVE	Verified	References	CVSS	Summary
CVE-2019-1552	False	Refs:	1.9	
CVE-2018-0737	False	Refs:	4.3	
CVE-2018-0734	False	Refs:	4.3	
CVE-2018-0732	False	Refs:	5.0	
CVE-2017-3737	False	Refs:	4.3	
CVE-2017-3735	False	Refs:	5.0	
CVE-2017-3738	False	Refs:	4.3	
CVE-2019-1559	False	Refs:	4.3	
CVE-2018-0739	False	Refs:	4.3	
CVE-2017-3736	False	Refs:	4.0	

Figura 63: Búsqueda de resultados sin resultados



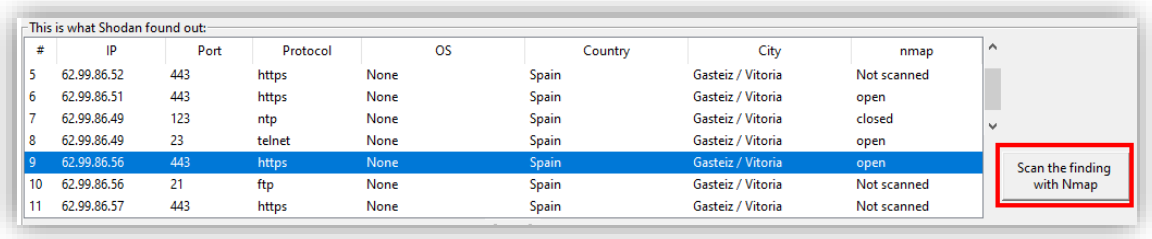
6.1.6. Botón de chequeo de Nmap

Para complementar la información que devuelven Shodan y ZoomEye, en la aplicación se ha añadido la posibilidad de consultar con Nmap, utilizando la librería Nmap desarrollada para Python. Cada consulta de IP y puerto tarda alrededor de 20 segundos, así que lanzarla para

los 100 posibles resultados de Shodan o ZoomEye tardaría en exceso. Por esa razón, se ha optado por crear un botón que lanza el comando Nmap para la IP y el puerto seleccionado, como muestra la Figura 64. El campo "*nmap*" de la tabla de resultados se actualiza según lo que devuelva el comando, pudiendo resultar en uno de los cuatro estados siguientes:

- **Open**: Nmap obtiene respuesta.
- **Filtered**: un *firewall* se interpone en el camino de la consulta realizada, por lo que Nmap no puede saber si el puerto realmente está abierto o cerrado.
- **Closed**: Nmap no encuentra ningún servicio escuchando en el puerto, ni la consulta ha sido filtrada por un *firewall*.

Figura 64: Botón de chequeo Nmap del registro seleccionado



6.2. PANTALLA DE ESTADÍSTICAS

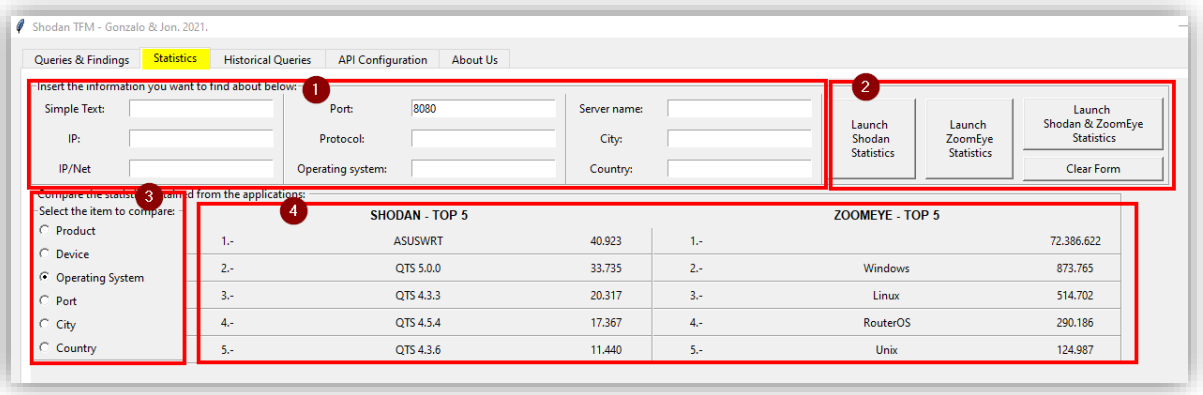
La segunda pestaña de DÉDALO, "*Statistics*", se utiliza para realizar búsquedas con resultados clasificados. Se ha utilizado lo que Shodan y ZoomEye denominan búsquedas con "*facets*". Son búsquedas que devuelven los datos agrupados según ciertos criterios (o *facets*) y permiten presentarlos ordenados de mayor a menor número de resultados.

La pantalla de estadísticas se ha organizado en cuatro partes, según se aprecia en la Figura 65:

1. Filtros de búsqueda: mite introducir los criterios de búsqueda, al igual que en la pantalla principal de DÉDALO. Los campos por los que se puede buscar son: texto libre, IP, red, puerto, protocolo, sistema operativo, servidor, ciudad y país.
2. Ejecución de búsqueda: cuenta con botones para lanzar la búsqueda independiente en Shodan o ZoomEye o lanzar ambas a la vez.
3. Selección de criterio de ordenación: el usuario debe indicar el criterio según el cual desea ver ordenados los datos de la consulta realizada. Se puede elegir entre producto, dispositivo, sistema operativo, puerto, ciudad y país.

4. Cuadro de resultados: muestra los datos agrupados según el criterio escogido y ordenados por el número de resultados de mayor a menor.

Figura 65: Disposición de la pantalla de estadísticas



A continuación, se presenta un ejemplo de búsqueda de dispositivos con sistema operativo Windows. En la Figura 66 observamos los resultados agrupados por país y en la Figura 67 por sistema operativo. Es significativo la diferencia de resultados, tanto en los totales por categoría como en los grupos con más resultados.

Figura 66: Resultados de búsqueda "*Sistema operativo= Windows*" ordenados por país

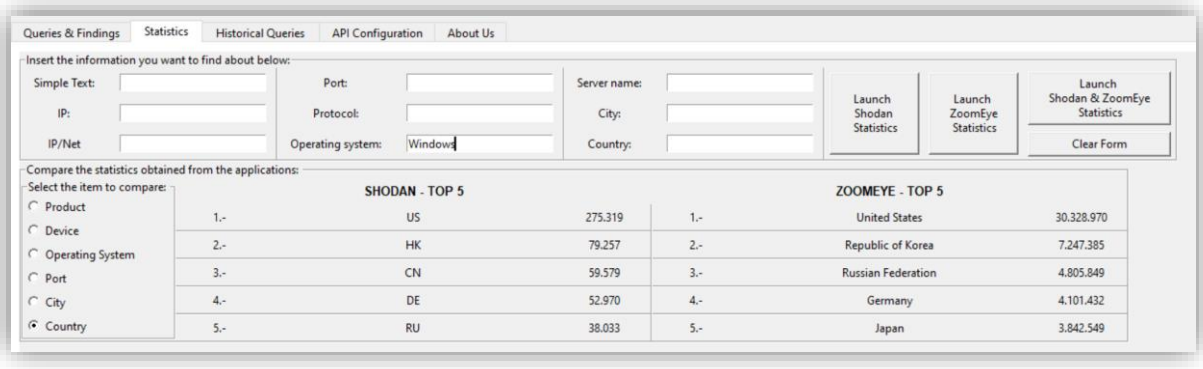
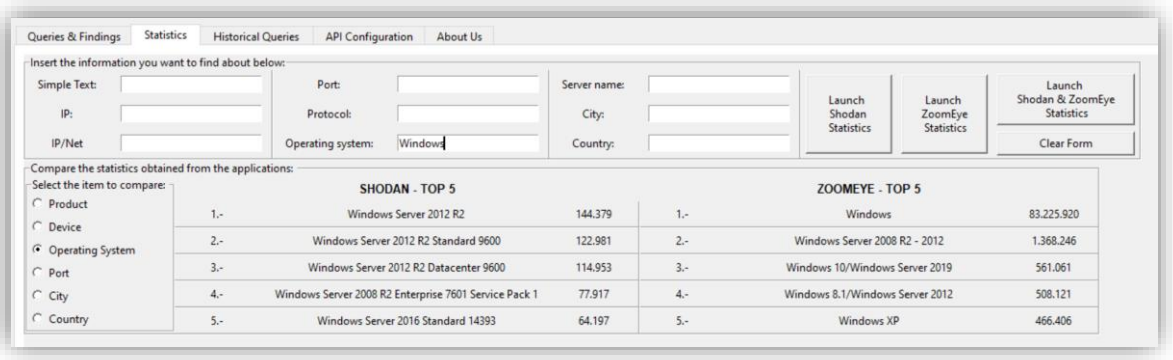


Figura 67: Resultados de búsqueda "*Sistema operativo=Windows*" ordenados por SO



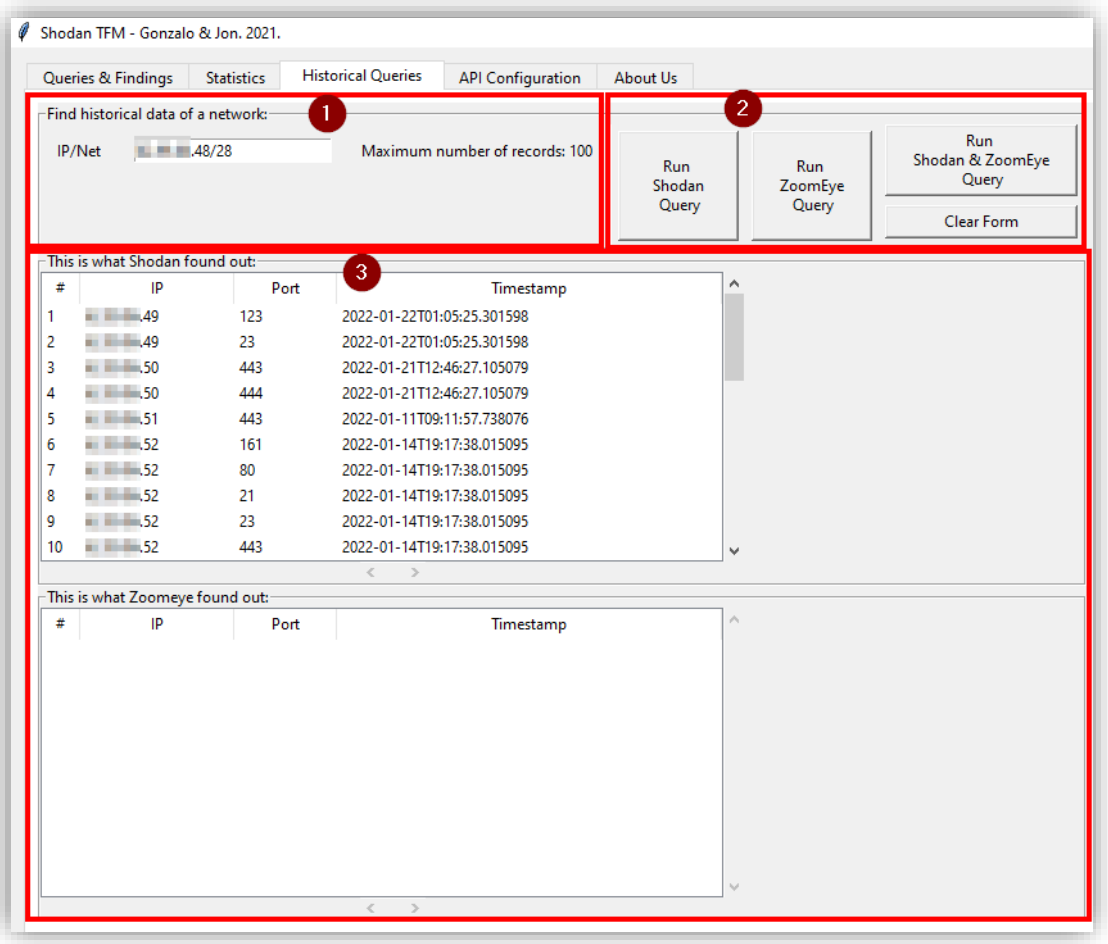
6.3. PANTALLA DE HISTÓRICO DE BÚSQUEDAS

La pestaña de consultas históricas (“*Historical Queries*”) utiliza un tipo de búsquedas de Shodan y ZoomEye que presenta todos los conjuntos de IP y puerto que han sido detectados en algún momento en la plataforma. Desgraciadamente, la licencia de ZoomEye de la que se disponía para este trabajo no da acceso a esta funcionalidad y solamente Shodan devuelve resultados, como se explica con más detalle en el apartado 7.1.

La pantalla está organizada según se observa en la Figura 72:

1. Red sobre la que se hará la búsqueda: se debe introducir una subred IPv4 en formato CIDR (a.b.c.d/n). Utilizando el módulo de Python IPy, la subred se descompone en la lista de IPs que contiene y se lanza la búsqueda para cada una de ellas.
2. Botones para ejecutar la búsqueda: contiene botones para lanzar la búsqueda en la plataforma Shodan y en ZoomEye independientemente, o en ambas a la vez.
3. Zona de resultados: hay dos cuadros de resultados, uno para Shodan y otro para ZoomEye. Como se ha comentado, en este trabajo no se muestran resultados en ZoomEye debido a que la clave API gratuita no incluye esta funcionalidad. Como se puede ver en la lista de resultados de Shodan, se presentan tres valores por resultado, IP, puerto y *timestamp*, que almacena la última fecha y hora en la que fue detectada la dupla IP y puerto o cambiaron los datos asociados a ella.

Figura 68: Disposición de la pantalla de búsquedas históricas



No está controlado el tamaño de la subred, por lo que, a fin de evitar que se bloquee la aplicación, se ha limitado el número de resultados que esta puede devolver. El máximo aceptado está definido en parámetro de configuración *MAX_NO_RECORDS* y tiene un valor por defecto de 100 (como se detallará más adelante).

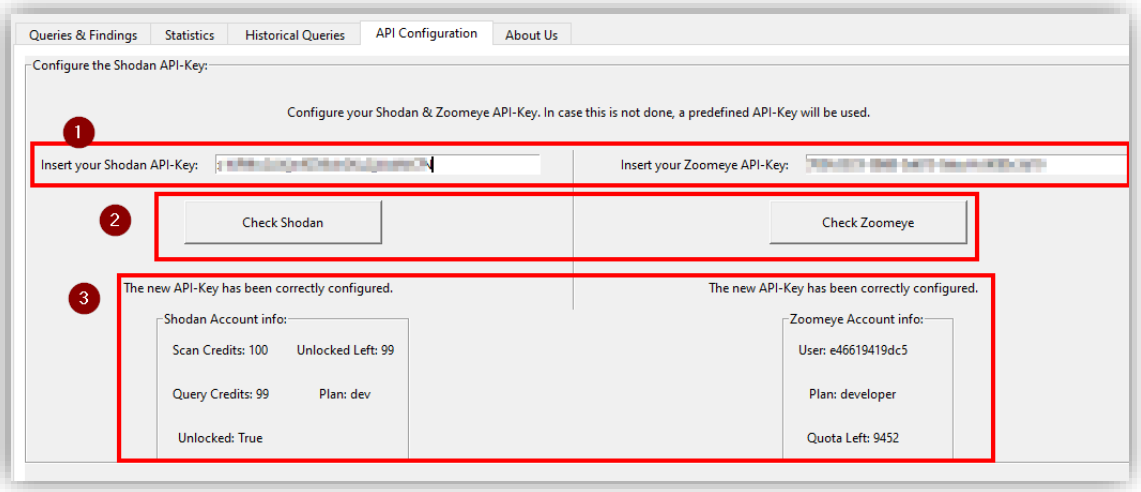
6.4. PANTALLA DE CONFIGURACIÓN DE API

La pestaña "*API Configuration*" permite comprobar el estado de las claves API que estamos utilizando. En la Figura 69 se observan las diversas secciones en las que se divide la pantalla:

1. En los campos editables se cargan automáticamente las claves de Shodan y ZoomEye con los valores del archivo de configuración "*config.cfg*" (explicado más adelante). Pueden ser modificadas para comprobar cualquier otra clave.
2. Los botones "*Check Shodan*" y *"Check Zoomeye"* sirven para lanzar la consulta de la clave en ambas plataformas.

3. Se muestra el resultado de la comprobación. En primer lugar, se indica si la clave es válida y, en caso de serlo, se muestran unos datos de la clave. Además, permite averiguar el tipo de plan licenciado y los créditos o cuota que tiene disponible la clave para el mes en curso.

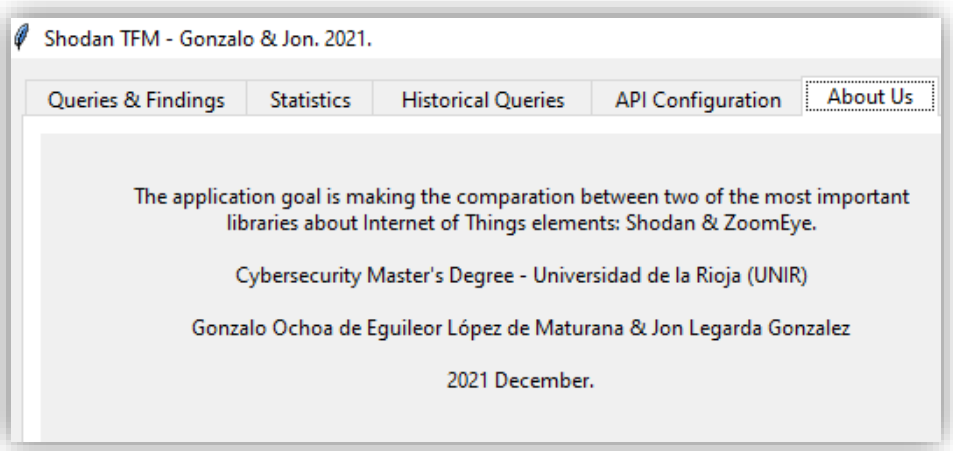
Figura 69: Pantalla de configuración de la clave de la API



6.5. PANTALLA DE ABOUT US

Esta pantalla (Figura 70) no tiene ninguna funcionalidad, simplemente muestra datos relativos a DÉDALO y sus autores.

Figura 70: Contenido de la pantalla "*About Us*"



6.6. ARCHIVO DE CONFIGURACIÓN "*CONFIG.CFG*"

Colgando del directorio base de la aplicación existe un archivo llamado "*/config/config.cfg*".

En este fichero se han creado tres secciones donde se guardan datos importantes para DÉDALO, como se observa en la Figura 71:

- Sección [SHODAN] – API_SHODAN: clave que se utilizará para acceder a la API de Shodan.
- Sección [ZOOMEYE] – API_ZOOMEYE: clave que se utilizará para acceder a la API de ZoomEye.
- Sección [HISTORICAL] – MAX_NO_RECORDS: número máximo de registros que se mostrarán en las consultas históricas.

Figura 71: Contenido de ejemplo de "*config.cfg*"

```
[SHODAN]
API_SHODAN = x1t6oh2Ku2jiR9ZX9kv2AVjm6oh2Ku2jIxbkNX7N


[ZOOMEYE]
API_ZOOMEYE = 700FD575-5C7B-0d845-8ebd-fc9032c3d19

[HISTORICAL]
MAX_NO_RECODS = 100
```

6.7. CONSECUCIÓN DE REQUISITOS FUNCIONALES DE DÉDALO

En el apartado 5.2.1 se definían los requisitos funcionales para DÉDALO. La Tabla 7 resume el grado de éxito en la implementación de la aplicación con respecto a los requisitos planteados en la Tabla 2.

Tabla 7: Tabla de requisitos funcionales y su consecución final

Código	Nombre	Consecución	Comentario
REQF01	Carga de la API-Key de Shodan.	Sí	La API-Key de Shodan se utiliza en todas las búsquedas sobre esta plataforma y en la pestaña de comprobación de APIs.
REQF02	Carga de la API-Key de ZoomEye.	Sí	La API-Key de ZoomEye se utiliza en todas las búsquedas sobre esta plataforma y en la pestaña de comprobación de APIs.
REQF03	Búsquedas en Shodan.	Sí	Se han implementado búsquedas estándar e históricas.
REQF04	Búsquedas en ZoomEye.	Sí	Se han implementado las búsquedas estándar de la plataforma

Código	Nombre	Consecución	Comentario
REQF05	Uso de la API de Shodan para Python.	Sí	Sí
REQF06	Uso de la API de ZoomEye para Python.	Sí	Sí
REQF07	Muestra de resultados de Shodan.	Sí	Se muestran 100 resultados por búsqueda.
REQF08	Muestra de resultados de ZoomEye.	Sí	Se muestran 100 resultados por muestra.
REQF09	Comparación de resultados.	Sí	Se comparan los tiempos de lanzamiento de la búsqueda y el número de resultados.
REQF10	Búsqueda de *exploits* o CVEs asociados a los resultados de Shodan.	Sí	Se pueden buscar los CVEs que guarda Shodan de cada uno de los resultados obtenidos.
REQF11	Búsqueda de *exploits* o CVEs asociados a los resultados de ZoomEye.	No	No se ha conseguido completar por completo. Aún y todo, el requisito REQF10 cubre esta funcionalidad de forma parcial, por lo que se ha añadido esta funcionalidad a las líneas de trabajo futuro.

6.8. CONSECUCIÓN DE REQUISITOS NO-FUNCIONALES DE DÉDALO

En el punto *5.2.2* se definían los requisitos no-funcionales para la aplicación. La Tabla 8 resume el grado de éxito en la implementación de DÉDALO con respecto a los requisitos planteados en la Tabla 3.

Tabla 8: Tabla de requisitos no-funcionales y su consecución final

Código	Nombre	Consecución	Comentario
REQNF01	Aplicación de fácil ejecución.	Sí	Es sencillo ejecutar la aplicación para un usuario que tiene un mínimo de conocimiento en cuanto a este tipo de herramientas.
REQNF02	Búsquedas con interfaz amigable.	Sí	Se han puesto los filtros más interesantes en campos para facilitar la sintaxis de la consulta y evitar tener que aprenderse cómo se filtran las búsquedas en cada plataforma.
REQNF03	Verificación de datos.	No	No se ha conseguido implementar la lógica de verificación de datos de entrada. Es un requisito no-funcional que se consideró de prioridad menor y se ha incluido dentro de las líneas de trabajo futuro.

Código	Nombre	Consecución	Comentario
REQNF04	Baja necesidad de cómputo.	Sí	La implementación y las pruebas se han llevado a cabo en ordenadores sin ninguna característica de cómputo especial. Las herramientas, Shodan y ZoomEye, pueden tardar más o menos en realizar las búsquedas, pero no se ha demostrado que ese hecho haya sido derivado por una necesidad de cómputo importante. En ocasiones, las búsquedas en dichas herramientas realizadas en la propia web también tienen diferentes tiempos de respuesta.
REQNF05	Facilidad de entender comparativas	Sí	La aplicación ofrece una interfaz muy simple con texto y datos simples y ordenados para entender las comparativas estadísticas.

6.9. PRINCIPALES DESVIACIONES DURANTE EL TRABAJO

Durante el desarrollo del proyecto, han ocurrido varias desviaciones positivas y negativas que han hecho modificar el resultado final de DÉDALO y la presente memoria.

Las desviaciones positivas son aquellos desarrollos evolutivos no contemplados en primera instancia (y que previsiblemente no se tenían en cuenta en los requisitos funcionales o no-funcionales de la aplicación) que se han realizado con éxito. Estas desviaciones han sido implementadas por considerarse un aporte relevante al trabajo.

Por otra parte, las desviaciones negativas son aquellos objetivos planteados en primera instancia que no se han conseguido resolver por cualquier motivo, así como cualquier suceso que haya impedido completar el diseño.

6.9.1. Principales desviaciones positivas

Las principales desviaciones positivas ocurridas durante el desarrollo del trabajo han sido las siguientes:

- **Uso de Nmap para la verificación de puertos abiertos**. En primera instancia, no se ha contemplado este requisito. Sin embargo, al observar ciertas diferencias en los resultados obtenidos por las dos herramientas, se ha decidido incluir una verificación de una tercera aplicación: Nmap. De esta manera, se dispone de una verificación

semiautomática que prueba si un puerto concreto para una IP específica está abierto o no.

- **Implementación de un fichero de configuración**. Durante la fase de implementación de DÉDALO se ha tenido que diseñar una lógica que fuera capaz de guardar claves de API para las herramientas (Shodan y ZoomEye). Por ello, se decidió establecer un archivo de configuración que permitiera realizar esto de forma adecuada.
- **Implementación de las consultas históricas**. A medida que el desarrollo de DÉDALO ha avanzado y tras analizar los resultados obtenidos, se ha considerado que otro aporte interesante a la aplicación podía ser la ejecución de consultas históricas. De esta manera, se ha incluido la pestaña o pantalla "*Historical Queries*" que permitiera aportar esa visión. En este caso, al considerarse un desarrollo evolutivo con impacto sobre el resultado final visual, se ha desarrollado también los casos de uso pertinentes en el presente documento.

6.9.2. Principales desviaciones negativas

Por su parte, a continuación, se listan las principales desviaciones negativas ocurridas durante el transcurso del proyecto:

- **Cantidad de resultados limitado**. La limitación de las APIs y su licenciamiento ha impactado directamente en la cantidad de resultados que la aplicación muestra. En este caso, se ha limitado a 100 los resultados obtenidos de las dos herramientas (Shodan y ZoomEye). Se puede evolucionar a futuro para utilizar licenciamientos de mayor coste e implementar la lógica necesaria para mostrar un número mayor de resultados.
- **CVEs asociados a ZoomEye**. No se ha cubierto el requisito no-funcional "REQF11", por falta de tiempo y porque tras encontrar posibles desarrollos evolutivos con otras funcionalidades, como las mostradas en el apartado *6.9.1 Principales desviaciones positivas*, se ha decidido implementar dichas funcionalidades.
- **Verificación de datos de entrada**. No se ha completado el requisito no-funcional "REQNF03" (ver apartado *6.8 Consecución de requisitos no-funcionales de DÉDALO*). En este caso, se trata de un requisito con menor prioridad que no se ha completado al considerar que otros evolutivos aportaban mayor funcionalidad a la aplicación.

7. ESCENARIOS DE APLICACIÓN PRÁCTICOS DE DÉDALO

En el presente apartado se presentan diferentes escenarios en los que se ha hecho uso de la aplicación desarrollada como resultado expuesto en este documento. Se han descrito cuatro escenarios de diferente tipo en los que DÉDALO puede ayudar a un usuario a conseguir los objetivos.

Estas cuatro aplicaciones prácticas planteadas son resultado de la explotación de DÉDALO y el análisis posterior de los datos obtenidos.

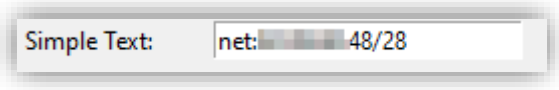
7.1. ESTUDIO DE LOS DATOS DE UNA RED CONOCIDA

Se analizan los datos devueltos por la API de Shodan y la de ZoomEye en una red pública conocida. La red cuenta con 14 IPs (/28) y nos referiremos a ella como xx.yy.zz.48/28.

La extracción de los datos se realizó en un estadio intermedio de la aplicación, por lo que se introdujo la consulta en el campo "*Simple Text*" que permite introducir consultas complejas con la sintaxis de cada aplicación.

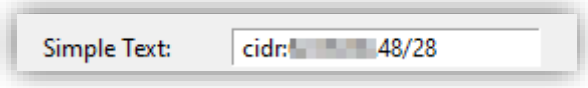
Para Shodan la búsqueda se lanzó como: net:xx.yy.zz.48/28

Figura 72: Búsqueda de subred en Shodan



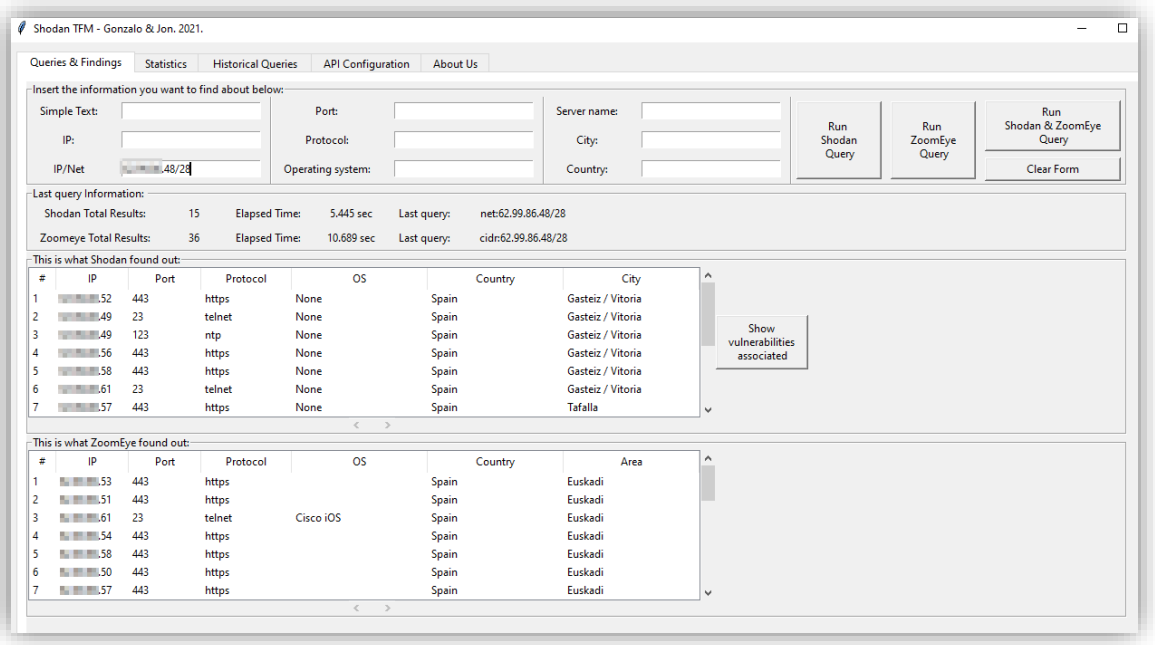
En ZoomEye la búsqueda de una red se realiza con el *dork* "cidr": cidr:xx.yy.zz.48/28

Figura 73: Búsqueda de subred en ZoomEye



Como se puede observar en la Figura 74, con fecha posterior a la de la actualización de los datos se ha actualizado la aplicación para que devuelva dirección de la red utilizando el campo "*IP/Net*" del panel de búsquedas. El botón "*Run Shodan & ZoomEye Query*" ejecuta la búsqueda en Shodan y ZoomEye.

Figura 74: Búsqueda de la red xx.yy.zz.48/28 en la aplicación



Para explotar los datos, se recogieron los archivos JSON con los resultados de cada consulta. Es llamativa la diferencia en la cantidad de resultados encontrados. Como se observa en la Tabla 9, ZoomEye ha encontrado 36 resultados totales, mientras que Shodan solo ha devuelto 16. Además, ZoomEye ha encontrado algún puerto abierto en cada una de las 14 direcciones IP del rango estudiado y Shodan los ha encontrado únicamente en 10.

Tabla 9: Resumen de puertos abiertos encontrados

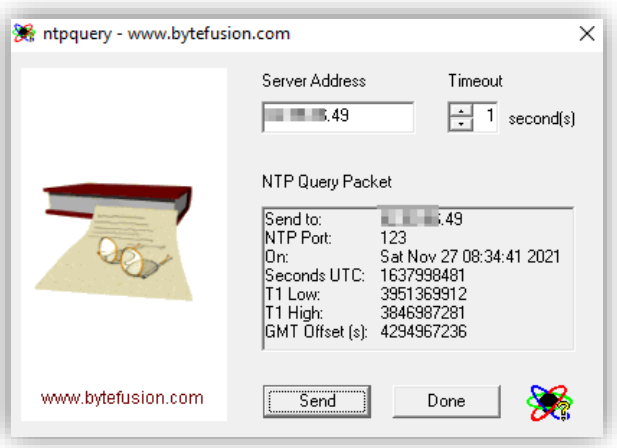
	ZoomEye	**Shodan**
Puertos abiertos encontrados	36	16
IPs con puertos abiertos encontrados	14	10

Habida cuenta de la diferencia de resultados, es necesario comprobar qué puertos son accesibles. Para ello utilizamos dos herramientas sencillas:

- Ejecutamos el comando '*telnet IP puerto*' sobre todos los puertos abiertos encontrados.

- Para interrogar el puerto 123, que es utilizado por el servicio ntp, utilizamos la herramienta NTP Query Tool, como se muestra en la Figura 75. Esta herramienta se puede descargar en www.bytefusion.com/download/index.htm.

Figura 75: Ejemplo de uso de la herramienta ntpquery



Tras verificar si manualmente se podía obtener respuesta en los puertos examinados, se obtuvieron los datos expuestos en la Tabla 10, en la que se han incluido todas las combinaciones de IP y puerto abiertos que hayan sido encontrados por ZoomEye o por Shodan. En la columna "*Respuesta manual*" se indica si se ha podido verificar que el puerto está realmente abierto y en el campo "*Test realizado*" se indica la herramienta que devolvió el resultado positivo, un telnet desde la línea de comandos o la herramienta Ntpquery. Con fondo verde se destacan aquellos registros en los que se ha podido verificar que el puerto está abierto, con fondo naranja se destaca el único puerto que aparece como abierto para ZoomEye y Shodan pero que, en realidad, está cerrado. Las conclusiones de esta verificación son:

- Hay una IP y puerto (IP xx.yy.zz.58 – puerto 80) marcado como abierto tanto por ZoomEye como por Shodan que, en realidad, parece estar cerrado.
- Los otros 15 registros devueltos por Shodan como positivos, están efectivamente abiertos.
- Hay 21 registros marcados como abiertos por Shodan que están cerrados.

Tabla 10: Comparación de resultados con verificación manual sobre red

IP	Puerto	Servicio	ZoomEye	Shodan	Respuesta manual	Test realizado
xx.yy.zz.49	22	ssh	SÍ	NO	NO	
xx.yy.zz.49	23	telnet	SÍ	SÍ	SÍ	telnet xx.yy.zz.49 23
xx.yy.zz.49	123	ntp	SÍ	SÍ	SÍ	ntpquery
xx.yy.zz.50	443	https	SÍ	SÍ	SÍ	telnet xx.yy.zz.50 443
xx.yy.zz.50	9000	http	SÍ	NO	NO	
xx.yy.zz.51	25	smtp	SÍ	NO	NO	
xx.yy.zz.51	443	https	SÍ	SÍ	SÍ	telnet xx.yy.zz.51 443
xx.yy.zz.52	21	ftp	SÍ	NO	NO	
xx.yy.zz.52	23	telnet	SÍ	NO	NO	
xx.yy.zz.52	80	https	SÍ	NO	NO	
xx.yy.zz.52	161	snmp	SÍ	NO	NO	
xx.yy.zz.52	443	https	SÍ	SÍ	SÍ	telnet xx.yy.zz.52 443
xx.yy.zz.53	443	https	SÍ	SÍ	SÍ	telnet xx.yy.zz.53 443
xx.yy.zz.54	21	ftp	SÍ	NO	NO	
xx.yy.zz.54	80	http	SÍ	NO	NO	
xx.yy.zz.54	443	https	SÍ	SÍ	SÍ	telnet xx.yy.zz.54 443
xx.yy.zz.54	3389	ms-wbt-server	SÍ	NO	NO	
xx.yy.zz.55	443	https	SÍ	NO	NO	
xx.yy.zz.56	21	ftp	SÍ	SÍ	SÍ	telnet xx.yy.zz.56 21
xx.yy.zz.56	443	https	SÍ	SÍ	SÍ	telnet xx.yy.zz.56 443
xx.yy.zz.57	25	smtp	SÍ	NO	NO	
xx.yy.zz.57	443	https	SÍ	SÍ	SÍ	telnet xx.yy.zz.57 443
xx.yy.zz.58	21	ftp	SÍ	NO	NO	
xx.yy.zz.58	23	telnet	SÍ	NO	NO	
xx.yy.zz.58	80	https	SÍ	SÍ	NO	
xx.yy.zz.58	443	https	SÍ	SÍ	SÍ	telnet xx.yy.zz.58 443
xx.yy.zz.59	21	ftp	SÍ	NO	NO	
xx.yy.zz.59	80	https	SÍ	NO	NO	
xx.yy.zz.59	443	https	SÍ	NO	NO	
xx.yy.zz.59	9000	http	SÍ	NO	NO	
xx.yy.zz.60	9000	http	SÍ	NO	NO	
xx.yy.zz.61	22	ssh	SÍ	NO	NO	
xx.yy.zz.61	23	telnet	SÍ	SÍ	SÍ	telnet xx.yy.zz.61 23
xx.yy.zz.61	123	ntp	SÍ	SÍ	SÍ	ntpquery
xx.yy.zz.62	23	telnet	SÍ	SÍ	SÍ	telnet xx.yy.zz.62 23
xx.yy.zz.62	123	ntp	SÍ	SÍ	SÍ	ntpquery

La conclusión más evidente que se puede sacar es que la tasa de falsos positivos de Shodan es muy inferior a ZoomEye:

- El 94% (15 de 16) de los resultados de Shodan han podido ser verificados.
- El 42% (15 de 36) de los resultados de ZoomEye han podido ser confirmados.

Resulta necesario profundizar más en los datos de ZoomEye para averiguar por qué la tasa de falsos positivos ha sido tan alta. Por el conocimiento de los autores de la red escaneada, se sabe que la mayoría de los puertos que aparecen abiertos por ZoomEye estuvieron en servicio en algún momento hasta el año 2017 aproximadamente y dejaron de estarlo según se renovaron los equipos y las tecnologías de esa red.

La explicación se encuentra en la propia documentación de ZoomEye (www.zoomeye.org/doc?channel=user), que explica que la plataforma utiliza el modo *overlay* (superposición) al escanear los datos. La primera vez que ZoomEye encuentra un puerto abierto, sus datos quedan registrados en la base de datos. Si en rastreos consecutivos el puerto aparece como cerrado, ya no actualizará los datos registrados y aparecerá como abierto. Para verificarlo, se puede consultar el campo *timestamp* que ZoomEye devuelve como parte de los datos, que guarda en la base de datos la última ocasión en la que se obtuvo respuesta para esa IP y puerto.

La Tabla 11 recoge todos los registros verificados como abiertos ordenados por la fecha de actualización en ZoomEye. Hay que tener en cuenta que la extracción de los datos se realizó el 24 de noviembre de 2021, por lo que podemos comprobar que la actualización es reciente en todos los registros.

Tabla 11: Fecha de actualización en ZoomEye de los puertos confirmados como abiertos

IP	Puerto	Timestamp
xx.yy.zz.49	123	2021-05-04T10:08:09
xx.yy.zz.62	123	2021-05-07T04:51:07
xx.yy.zz.61	123	2021-06-18T22:28:19
xx.yy.zz.53	443	2021-10-23T21:36:58
xx.yy.zz.56	21	2021-10-26T16:59:05
xx.yy.zz.49	23	2021-11-06T20:22:00
xx.yy.zz.56	443	2021-11-14T03:48:41
xx.yy.zz.61	23	2021-11-16T10:26:29
xx.yy.zz.51	443	2021-11-17T18:31:22
xx.yy.zz.54	443	2021-11-17T18:31:36

IP	Puerto	Timestamp
xx.yy.zz.52	443	2021-11-17T18:32:02
xx.yy.zz.62	23	2021-11-23T11:16:46
xx.yy.zz.57	443	2021-11-23T14:47:25
xx.yy.zz.50	443	2021-11-24T09:07:43
xx.yy.zz.58	443	2021-11-25T04:50:07

La Tabla 12 muestra los datos de los registros considerados como falsos positivos ordenados por fecha de actualización descendente. Aunque la fecha de actualización de la mayor parte de los registros tiene una antigüedad superior al año, hay 5 registros actualizados como abiertos en los 40 días previos a la obtención de los datos, lo cual indica que o bien esos puertos estuvieron temporalmente abiertos coincidiendo con un rastreo de ZoomEye o que la herramienta los ha detectado como abiertos incorrectamente.

Tabla 12: Fecha de actualización de los puertos cerrados mostrados por ZoomEye

IP	Puerto	Timestamp
xx.yy.zz.59	443	2021-11-17T18:31:31
xx.yy.zz.55	443	2021-11-17T18:31:21
xx.yy.zz.52	80	2021-10-20T22:57:57
xx.yy.zz.58	80	2021-10-20T22:57:57
xx.yy.zz.59	80	2021-10-20T22:57:57
xx.yy.zz.57	25	2019-09-24T00:20:58
xx.yy.zz.52	21	2019-09-10T17:47:50
xx.yy.zz.59	21	2019-09-10T17:47:41
xx.yy.zz.58	23	2019-07-09T22:36:04
xx.yy.zz.52	23	2018-11-19T17:43:53
xx.yy.zz.54	80	2018-08-09T16:44:45
xx.yy.zz.52	161	2018-07-23T17:23:45
xx.yy.zz.58	21	2017-06-12T19:11:56
xx.yy.zz.61	22	2017-05-23T20:16:41
xx.yy.zz.49	22	2017-05-23T15:47:44
xx.yy.zz.60	9000	2017-03-26T22:58:58
xx.yy.zz.59	9000	2017-03-26T22:58:49
xx.yy.zz.50	9000	2017-03-26T22:58:25
xx.yy.zz.54	3389	2016-09-30T00:52:26
xx.yy.zz.54	21	2014-11-10T03:18:59
xx.yy.zz.51	25	2014-10-09T15:04:30

Shodan ofrece información histórica de una IP, a través de la función *host* con el parámetro "*history=true*". Mediante esta búsqueda vamos a ver todos los puertos que han sido visibles para Shodan y cuál fue la última actualización de la IP. ZoomEye nos ofrece información algo más detallada, ya que nos permite averiguar la última vez en que un puerto fue descubierto abierto, mientras que Shodan lo ofrece a nivel de IP.

La Figura 76 muestra el código Python que lanza la búsqueda histórica en Shodan para la red estudiada xx.yy.zz/48. En la Figura 77 se puede observar el resultado devuelto.

Figura 76: Código en Python que lanza consulta de datos históricos de IP en Shodan

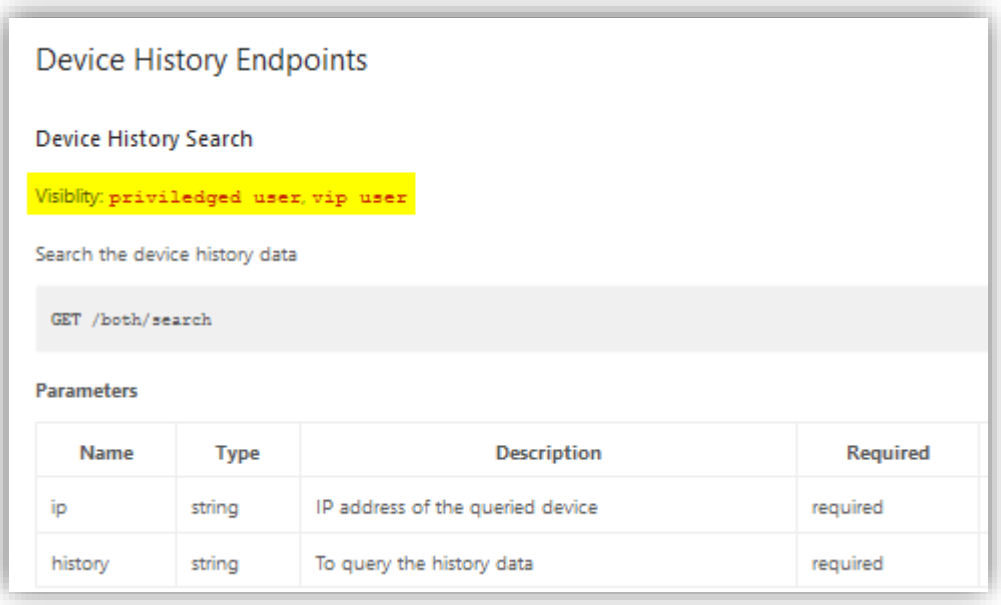
```python
from shodan import Shodan
import configparser
from IPy import IP
from window import
configuracion = configparser.ConfigParser()
configuracion.read('config.cfg')

resultado = []
api_shodan= configuracion['SHODAN']['API_SHODAN']
api=Shodan(api_shodan)
ip = IP('xx.yy.zz.48/28')
for contador in ip:
    try:
        currentip=str(contador)
        host=api.host(currentip, history=True)
        resultado.append(host)
    except Exception as e:
        pass
```

Figura 77: Resultado de la búsqueda histórica sobre la red xx.yy.zz.48/28

```
C:\Users\mcygo\OneDrive - UNIR\TFM\TFMPython>py 006_shodan_history.py
  Direccion: xx.yy.zz.49   Last Update: 2021-11-17T08:02:33.336389     Puertos: [123, 23]
  Direccion: xx.yy.zz.50   Last Update: 2021-11-27T21:55:40.632834     Puertos: [443, 444]
  Direccion: xx.yy.zz.51   Last Update: 2021-10-25T01:23:21.521217     Puertos: [443]
  Direccion: xx.yy.zz.52   Last Update: 2021-11-03T20:08:23.748021     Puertos: [161, 80, 21, 23, 443, 444]
  Direccion: xx.yy.zz.53   Last Update: 2021-11-15T14:01:00.009053     Puertos: [443]
  Direccion: xx.yy.zz.54   Last Update: 2021-11-26T19:00:59.149455     Puertos: [80, 443, 3389]
  Direccion: xx.yy.zz.55   Last Update: 2021-09-04T10:03:34.270811     Puertos: [443]
  Direccion: xx.yy.zz.56   Last Update: 2021-11-22T14:43:24.327911     Puertos: [443, 444, 21]
  Direccion: xx.yy.zz.57   Last Update: 2021-11-26T03:56:06.377290     Puertos: [443]
  Direccion: xx.yy.zz.58   Last Update: 2021-11-30T12:28:03.659149     Puertos: [161, 80, 21, 23, 443, 444]
  Direccion: xx.yy.zz.59   Last Update: 2021-10-16T14:39:37.153137     Puertos: [80, 161, 443, 9000, 23]
  Direccion: xx.yy.zz.60   Last Update: 2017-04-02T04:26:29.175604     Puertos: [9000]
  Direccion: xx.yy.zz.61   Last Update: 2021-11-19T22:16:35.216801     Puertos: [123, 23]
  Direccion: xx.yy.zz.62   Last Update: 2021-11-10T07:23:28.554182     Puertos: [123, 23]
```

Para completar la información anterior y hacer una comparativa de los datos más completa, habría sido interesante sacar información histórica de ZoomEye, que tiene una función a tal efecto: *history_ip()*. Sin embargo, aunque se ha implementado en la aplicación, no solo puede ser utilizada por usuarios categorizados como *privileged* y *vip*, tal y como se muestra en la Figura 78.

Figura 78: Tipo de usuarios con acceso a las funciones de búsqueda histórica



Device History Endpoints

Device History Search

Visiblity: priviledged user, vip user

Search the device history data

GET /both/search

Parameters

Name	Type	Description	Required
ip	string	IP address of the queried device	required
history	string	To query the history data	required

Fuente: Documentación de ZoomEye www.zoomeye.org/doc#host-search.

Se ha realizado la implementación de la función *history_ip()* en la aplicación en un programa Python simple (Figura 79) y también en el módulo "*zoomeye_handler.py*" de la aplicación (Figura 80). En ambos casos, la búsqueda no devuelve ningún resultado (Figura 81 y Figura 82).

Figura 79: Implementación de *history_ip()* en un programa Python

```
print( search_string )
data = zm.history_ip( search_string )
for linea in data:
    print (f"{data.index(linea) + 1}. IP: {linea['timestamp']}")
```

Figura 80: Implementación de *history_ip()* en el módulo "*zoomeye_handler.py*"

```
def run_zoomeye_his_query(self, search_text):
    results = []
    try:
        host = self.api.history_ip(search_text)
        print("Method: ZoomeyeQuery().run_zoomeye_query | Api-Key = {} ".format(self.zoomeye_api_key))
        for linea in host:
            results.append(search_text,linea['timestamp'],linea['portinfo']['port'])
    except Exception as e:
        print('Error: {}'.format(e))
```

Figura 81: Resultado de la búsqueda histórica con un programa Python independiente

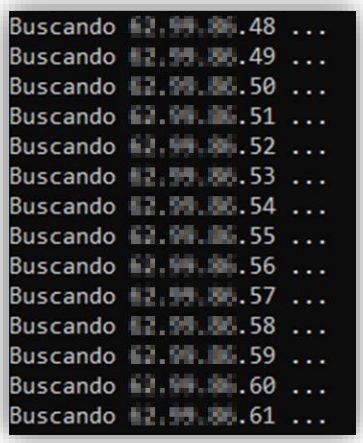
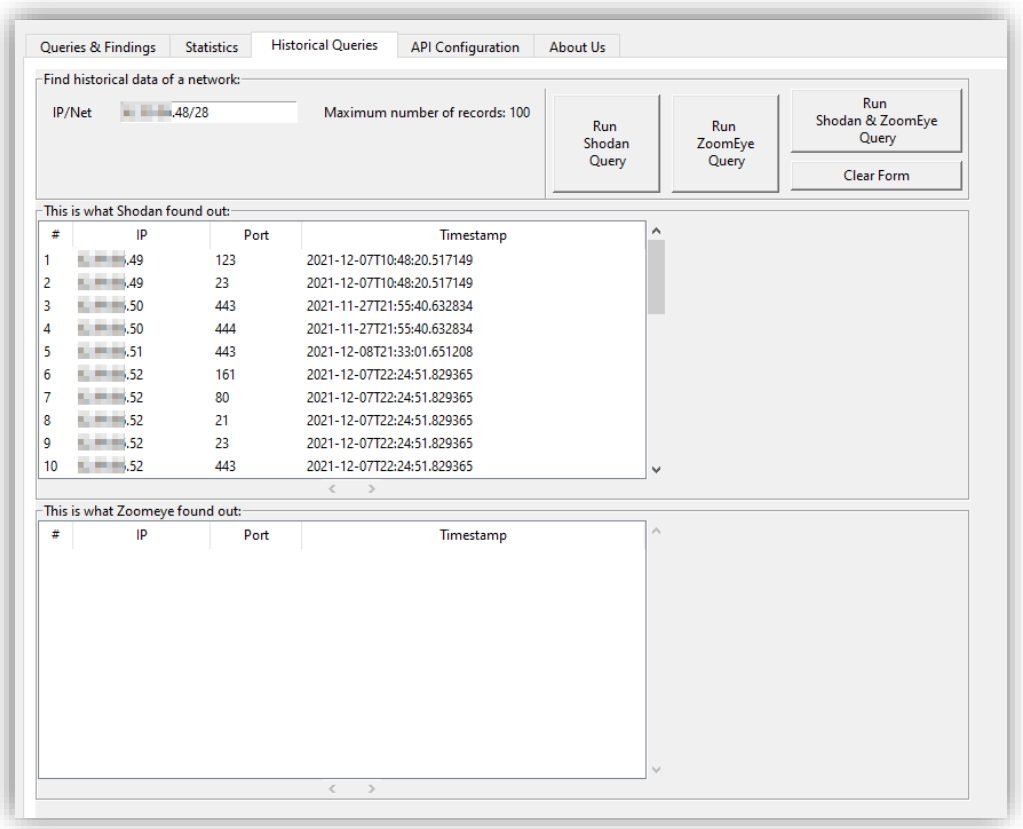


Figura 82: Resultado de la búsqueda histórica con DÉDALO comparada con Shodan



Con toda la información que se ha podido recopilar, se genera la Tabla 13, que presenta un resumen por IP y puerto de los datos históricos. La columna "*Abierto*" indica si se ha verificado manualmente que el puerto era accesible en la fecha en que se recogió la información. Las columnas "ZoomEye" y "Shodan" muestran la última fecha en que se actualizaron los datos en cada plataforma, aunque debemos tener en cuenta que en Shodan la fecha de actualización será la misma para todos los puertos de la misma IP. Los datos de actualización pueden ser posteriores a la fecha inicial de recogida de los datos porque la consulta se efectuó seis días más tarde.

Tabla 13: Comparativa datos históricos ZoomEye y Shodan

IP	portinfo/port	Abierto	ZoomEye	Shodan
62.99.86.49	22	-	23/05/2017	
62.99.86.49	23	SÍ	06/11/2021	17/11/2021
62.99.86.49	123	SÍ	04/05/2021	17/11/2021
62.99.86.50	443	SÍ	24/11/2021	27/11/2021
62.99.86.50	444	-		27/11/2021
62.99.86.50	9000	-	26/03/2017	

IP	portinfo/port	Abierto	ZoomEye	Shodan
62.99.86.51	25	-	09/10/2014	
62.99.86.51	443	SÍ	17/11/2021	25/10/2021
62.99.86.52	21	-	10/09/2019	03/11/2021
62.99.86.52	23	-	19/11/2018	03/11/2021
62.99.86.52	80	-	20/10/2021	03/11/2021
62.99.86.52	161	-	23/07/2018	03/11/2021
62.99.86.52	443	SÍ	17/11/2021	03/11/2021
62.99.86.52	444	-		03/11/2021
62.99.86.53	443	SÍ	23/10/2021	15/11/2021
62.99.86.54	21	-	10/11/2014	
62.99.86.54	80	-	09/08/2018	26/11/2021
62.99.86.54	443	SÍ	17/11/2021	26/11/2021
62.99.86.54	3389	-	30/09/2016	26/11/2021
62.99.86.55	443	-	17/11/2021	04/09/2021
62.99.86.56	21	SÍ	26/10/2021	22/11/2021
62.99.86.56	443	SÍ	14/11/2021	22/11/2021
62.99.86.56	444	-		22/11/2021
62.99.86.57	25	-	24/09/2019	
62.99.86.57	443	SÍ	23/11/2021	26/11/2021
62.99.86.58	21	-	12/06/2017	30/11/2021
62.99.86.58	23	-	09/07/2019	30/11/2021
62.99.86.58	80	-	20/10/2021	30/11/2021
62.99.86.58	161	-		30/11/2021
62.99.86.58	443	SÍ	25/11/2021	30/11/2021
62.99.86.58	444	-		30/11/2021
62.99.86.59	21	-	10/09/2019	
62.99.86.59	23	-		16/10/2021
62.99.86.59	80	-	20/10/2021	16/10/2021
62.99.86.59	161	-		16/10/2021
62.99.86.59	443	-	17/11/2021	16/10/2021
62.99.86.59	9000	-	26/03/2017	16/10/2021
62.99.86.60	9000	-	26/03/2017	02/04/2017
62.99.86.61	22	-	23/05/2017	
62.99.86.61	23	SÍ	16/11/2021	19/11/2021
62.99.86.61	123	SÍ	18/06/2021	19/11/2021
62.99.86.62	23	SÍ	23/11/2021	10/11/2021
62.99.86.62	123	SÍ	07/05/2021	10/11/2021

De un total de 43 combinaciones de IP y puerto:

- 29 han sido detectadas en algún momento por ambas plataformas

- 7 han sido detectadas solo por ZoomEye.
- 7 han sido detectadas solo por Shodan.
- Todos los puertos que están abiertos han sido detectados en algún momento por ambas plataformas.

La principal conclusión de este análisis es que no se pueden utilizar directamente los datos devueltos por las consultas a Shodan o ZoomEye. Es necesario estudiarlos en profundidad para determinar su validez y para ello puede ser necesario utilizar otras herramientas externas como por ejemplo: Nmap, Telnet u otras herramientas accesibles desde API o línea de comandos.

7.2.COMPARACIÓN DEL NÚMERO DE RESULTADOS Y TIEMPOS DE EJECUCIÓN

La aplicación es capaz de mostrar el número total de resultados y el tiempo empleado en ejecutar las consultas. Aprovechando esta característica, se han realizado búsquedas en momentos diferentes, para evaluar el desempeño general de Shodan y ZoomEye. La

Tabla 14 muestra algunas de las mediciones realizadas. Aunque el conjunto de datos que se ha obtenido no ha sido tan grande como para obtener conclusiones definitivas, sí ha servido para ratificar algunas impresiones preliminares formadas durante el desarrollo y las pruebas de la aplicación.

Tabla 14: Ejemplos de mediciones sobre la ejecución de las consultas

		Total de resultados		Tiempo empleado (segundos)	
Fecha	**Consulta**	**Shodan**	**ZoomEye**	**Shodan**	**ZoomEye**
08/12/2021	port:80	64.852.522	251.105.584	5,357	28,164
08/12/2021	port:80	64.852.812	251.105.635	4,855	20,5
23/01/2022	port:80	98.209.611	256.670.803	4,767	20,986
08/12/2021	port:80 protocol:http	19	140.268	2,465	102,313
23/01/2022	port:80 protocol:http	19	140.890	1,899	65,545
08/12/2021	net:xx.yy.zz.48/28	15	36	5,236	9,355
23/01/2022	net:xx.yy.zz.48/28	13	36	0,48	10,415
08/12/2021	port:80 city:Madrid country:ES	368.520	547.715	4,729	35,365
23/01/2022	port:80 city:Madrid country:ES	318.830	553.279	28,353	44,668
08/12/2021	city:Madrid country:ES	1.718.962	3.865.857	9,036	18,333
14/12/2021	city:Madrid country:ES	1.683.723	3.871.491	7,896	82,942
23/01/2022	city:Madrid country:ES	1.448.773	3.912.394	0,819	17,682

		Total de resultados		Tiempo empleado (segundos)	
Fecha	**Consulta**	**Shodan**	**ZoomEye**	**Shodan**	**ZoomEye**
14/12/2021	city:Madrid	1.685.587	3.871.501	3,702	202,679
23/01/2022	city:Madrid	1.453.613	3.912.475	2,075	57,119

Las conclusiones obtenidas a partir del estudio de los datos recopilados son:

- ZoomEye saca una mayor cantidad de resultados.
- En ZoomEye, por regla general, la cantidad de resultados para una misma consulta aumenta según transcurre el tiempo. En Shodan no se cumple esta afirmación y no es extraño que la cantidad de resultados devueltos por una misma consulta descienda pasados varios días.
- Las consultas en Shodan se ejecutan de media en un tiempo de 5 segundos, 10 veces más rápido que la media de la duración de las consultas en ZoomEye, unos 50 segundos.
- Las búsquedas en Shodan tienen una duración mucho más predecible que en ZoomEye. En Shodan la mayoría de las consultas terminan antes de transcurrir 10 segundos. Sin embargo, en ZoomEye hay una gran variabilidad. El tiempo empleado rara vez baja de los 10 segundos, pero puede llegar a superar los 200 segundos.
- Hay ocasiones en las que las consultas en ZoomEye no responden y terminan con un error. Con Shodan esto solo ha sucedido en muy raros casos.

Según el análisis del punto anterior, tiene sentido que ZoomEye devuelva más resultados que Shodan, ya que devuelve todas aquellas combinaciones de IP y puerto que hayan sido detectadas en alguna ocasión.

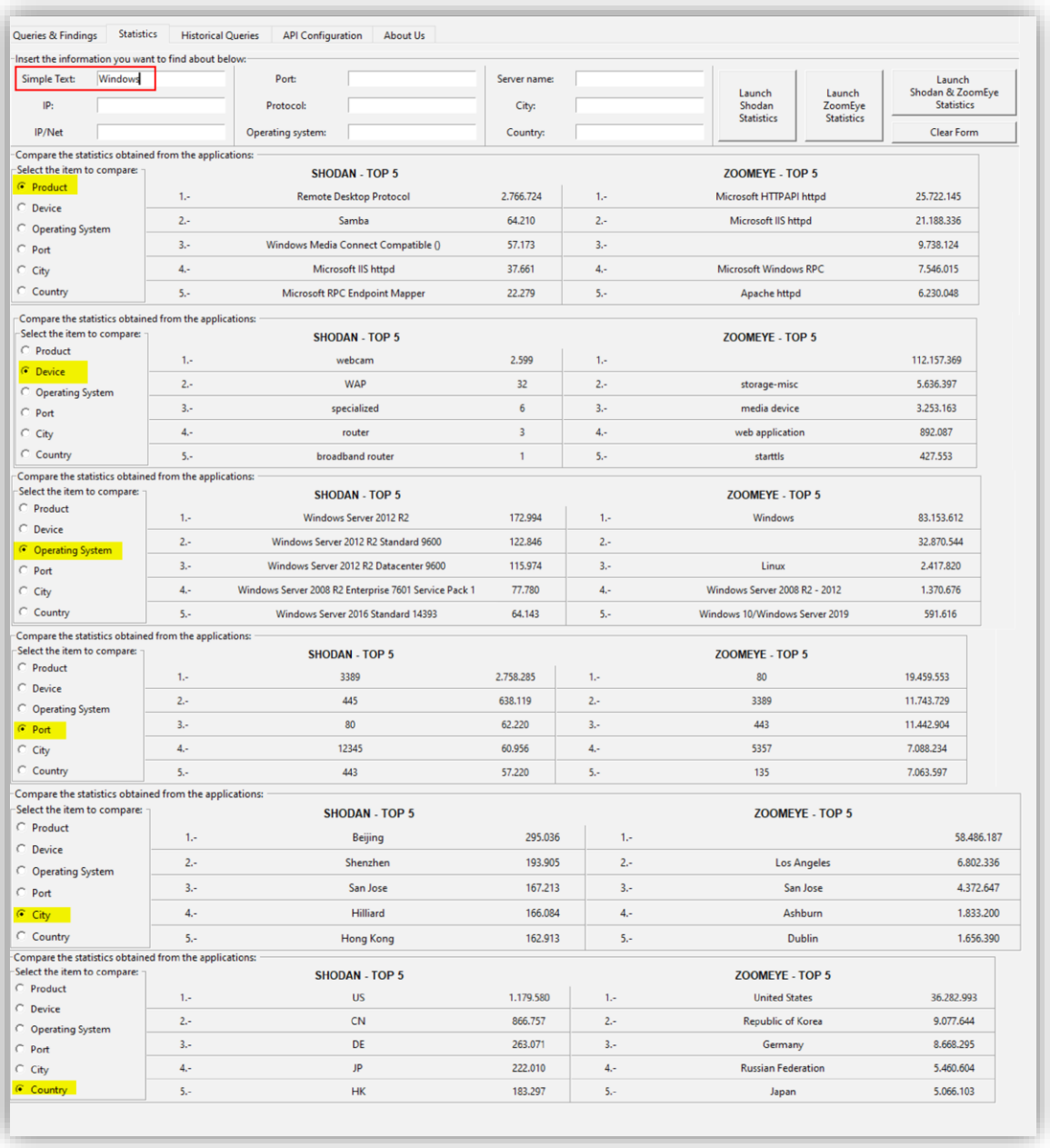
Las fluctuaciones en la cantidad de datos devueltos por Shodan merecerían un estudio en mayor profundidad, ya que resulta imposible afirmar con seguridad su origen. Una posible causa de la bajada en el número de dispositivos detectados en consultas sucesivas sería que esos dispositivos han sido retirados o protegidos contra el rastreo de Shodan. También parece posible que los sucesivos rastreos vayan corrigiendo datos incorrectos guardados en las versiones más antiguas de la base de datos de la plataforma.

En cuanto al tiempo de ejecución de las consultas, queda clara la estabilidad de la plataforma Shodan y que es más rápida que ZoomEye. Sin embargo, una vez más sería descabellado sacar conclusiones categóricas porque el tiempo de respuesta puede no depender únicamente de la plataforma. Sería interesante poder hacer estas mismas verificaciones desde un punto de la zona geográfica de China para conocer los tiempos de respuesta máximos que puede dar la plataforma.

7.3.COMPARACIÓN DE LAS CONSULTAS ESTADÍSTICAS EN SHODAN Y ZOOMEYE

La existencia de búsquedas estadísticas permite comparar los conjuntos de resultados que devuelven los dos motores de búsqueda utilizados. En la Figura 83 se han incluido los resultados arrojados por la búsqueda "Windows". Esta búsqueda servirá como ejemplo, porque el análisis realizado con otras búsquedas completamente diferentes nos ha llevado a conclusiones similares.

Figura 83: Resultados agrupados de la consulta "Windows"



En todas las consultas realizadas se destacan las siguientes peculiaridades:

- La cantidad de resultados que devuelve ZoomEye es mucho mayor, refrendando los hallazgos de los análisis anteriores.
- ZoomEye, en algunos casos, deja campos vacíos, como podemos ver en la agrupación por dispositivos, en la que no indica el tipo de dispositivo que ocupa el primer lugar de la lista. No ha sido posible descubrir si los campos vacíos son los que no tienen un valor definido o si ZoomEye oculta su valor.

- El principal hecho a destacar está en la disparidad de los valores mostrados por cada motor de búsqueda en cada una de las categorías. Por ejemplo, en las categorías producto y dispositivo, Shodan muestra primero "*Remote Desktop Protocol*" y "*webcam*", que en ZoomEye ni si quiera aparecen. La causa de esto puede ser que las dos plataformas organizan la información de forma muy diferente.

Por otra parte, ha resultado llamativo que en ninguna de las búsquedas realizadas para este análisis la clasificación geográfica haya devuelto datos localizados en China. La Figura 84 y la Figura 85 sirven de ejemplo. Esto podría indicar que puede haber un filtrado de datos que de forma sistemática no son tenidos en cuenta. Ante la importancia de esta hipótesis, en el apartado *7.4* se realiza un análisis más exhaustivo.

Figura 84: Clasificación de resultados de búsqueda de puerto en Shodan y ZoomEye

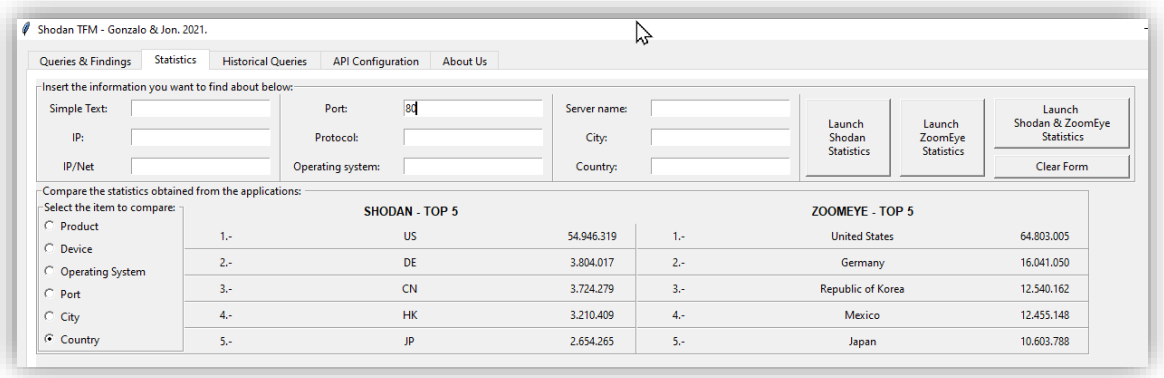
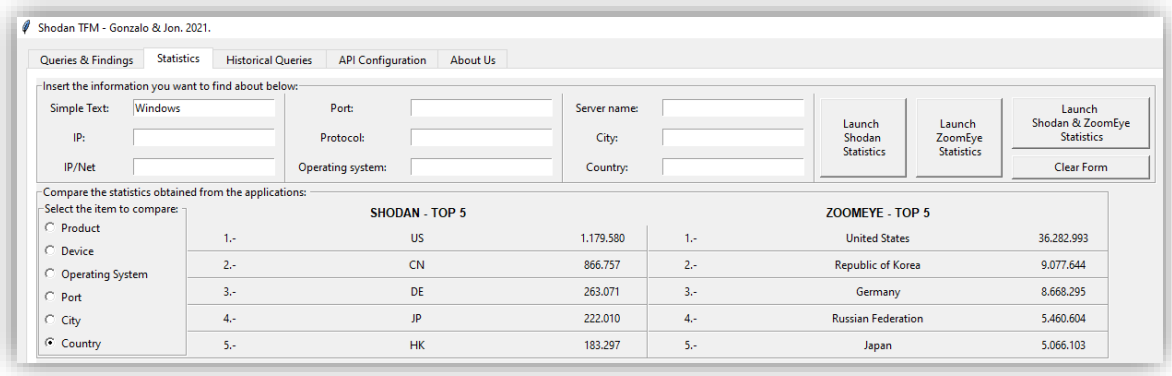


Figura 85: Clasificación de resultados de búsqueda simple en Shodan y ZoomEye

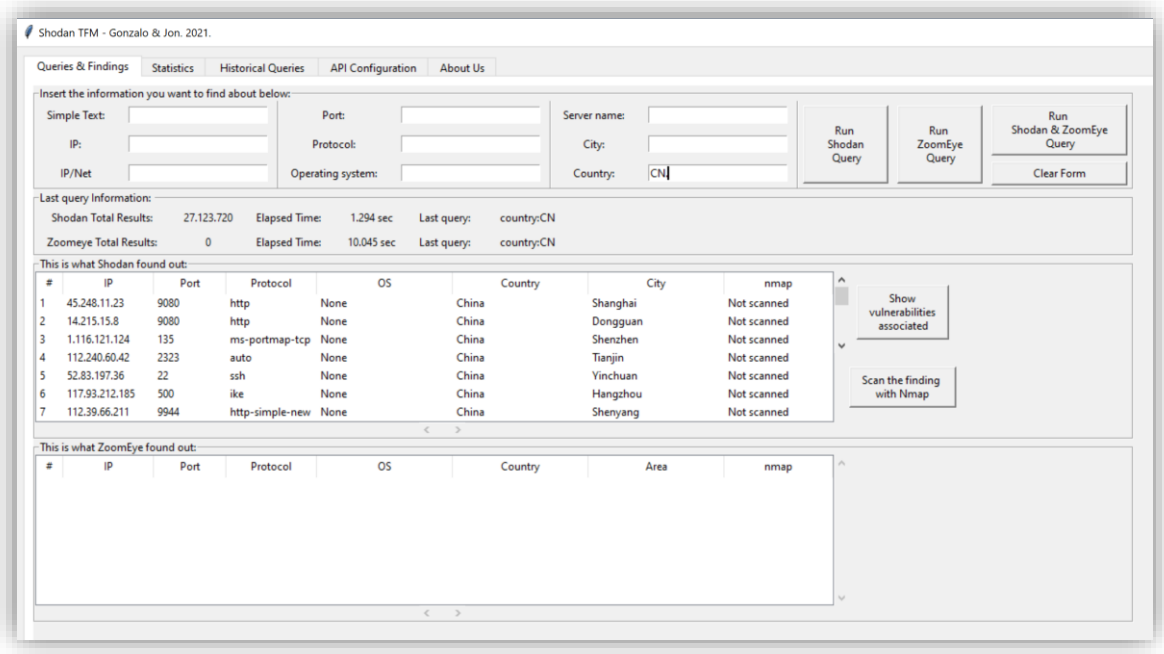


7.4. ANÁLISIS PARA VERIFICAR SI ZOOMEYE BLOQUEA RESULTADOS CON ORIGEN CHINA

A la vista de los resultados que devuelven las búsquedas estadísticas, llamaba la atención el hecho de que en el *top* 5 por países de casi todas las búsquedas, en Shodan apareciera siempre China, pero que en ZoomEye no saliera nunca este país (Figura 84 y Figura 85). Se ha aprovechado DÉDALO para estudiar si esto es una simple sospecha o hay datos que lo refrenden.

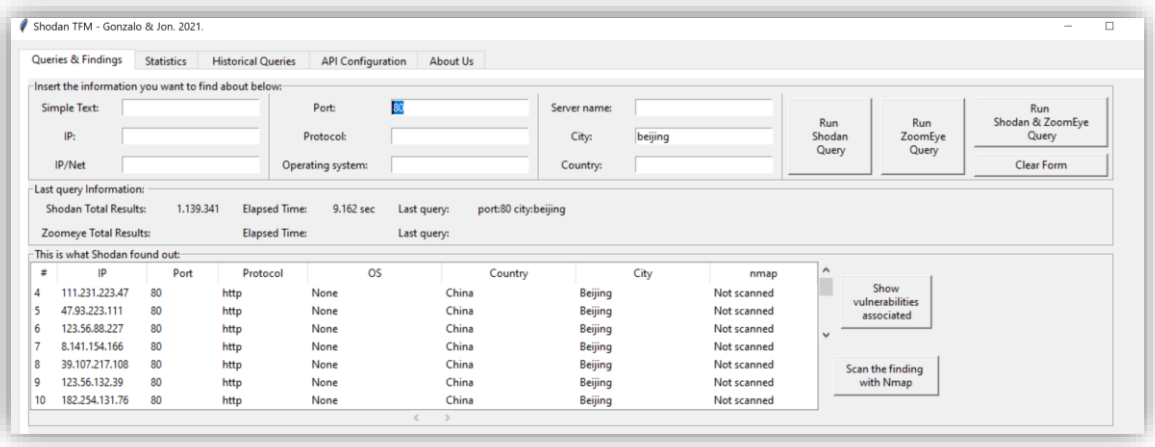
La primera comprobación consistió en buscar e IPs chinas ("*country:CN*") en las dos plataformas. Como se puede observar en la Figura 86, Shodan devuelve más de 27 millones de resultados, mientras que ZoomEye no devuelve ninguno.

Figura 86: Búsqueda en la aplicación de IPs con origen en China ("*country:CN*")



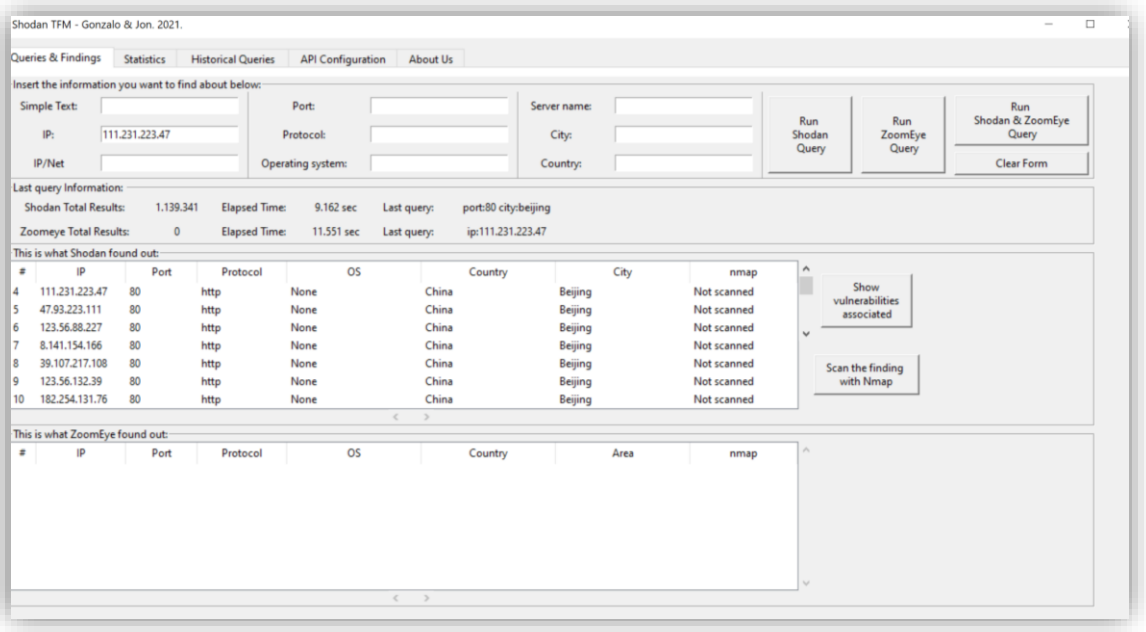
Al no tener éxito con el intento anterior, se intenta encontrar IPs chinas de una forma indirecta, en dos pasos. El primer paso será una búsqueda en Shodan que nos devuelva una lista de IPs localizadas en China, seguido de búsquedas de cada una de las IPs en ZoomEye. Sobre Shodan buscamos IPs que tengan el puerto 80 abierto y que estén localizadas en Beijing (Figura 87).

Figura 87: Búsqueda en Shodan de IPs con el puerto 80 abierto localizadas en Beijing



A continuación, se busca cada IP en la plataforma ZoomEye. En la Figura 88 se observa, a modo de ejemplo, que ZoomEye no muestra información de la primera IP devuelta por Shodan, 111.231.223.47.

Figura 88: Búsqueda en ZoomEye de puertos abiertos de la IP 111.231.223.47

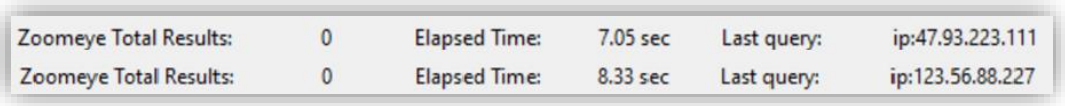


A modo de comprobación, se intenta abrir la IP estudiada en un navegador: http://111.231.223.47 La responde en el puerto 80. No llegamos a abrir la página, pero el antivirus la bloquea, lo que demuestra que esa IP tiene el puerto 80 abierto. La Figura 89 evidencia que el puerto 80 está abierto y los caracteres chinos pueden validar la procedencia china de la IP.

Figura 89: Aspecto de la página http://111.231.223.47

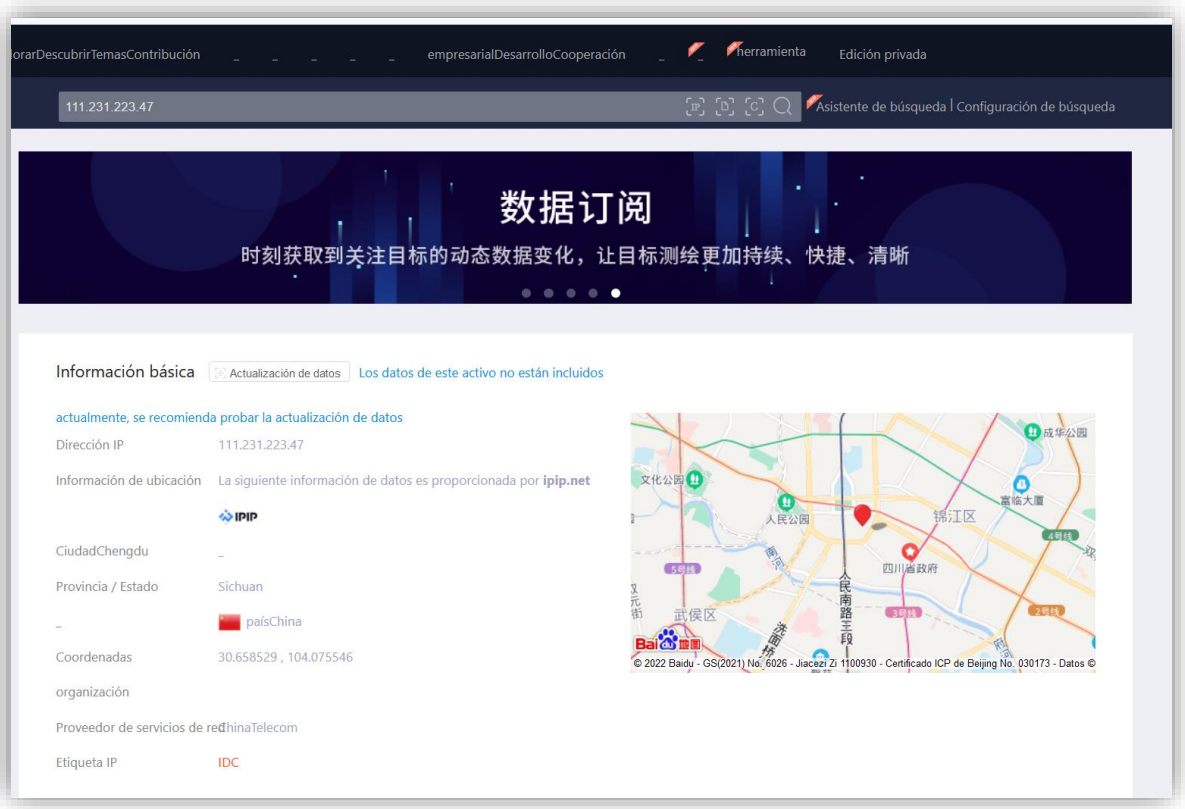
Las comprobaciones en ZoomEye con las demás IPs resultantes de la búsqueda en Shodan arrojan el mismo resultado negativo (Figura 90).

Figura 90: Búsqueda en ZoomEye de puertos abiertos de 47.93.223.111 y 123.56.88.227



Sorprendentemente, la página www.zoomeye.org sí devuelve resultados para la búsqueda de la IP 111.231.223.47 (Figura 91).

Figura 91: Búsqueda en www.zoomeye.org de puertos abiertos de la IP 111.231.223.47



Se busca "*country:CN*" en la página www.zoomeye.org. Para la misma búsqueda que en la aplicación no devolvía resultados, la página indica que hay más de 453 millones de direcciones IPv4.

Figura 92: Búsqueda en www.zoomeye.org de "*country:CN*"

A fin de validar las pruebas anteriores, se realiza una búsqueda de IPs para el país España en la aplicación y en la página web. La búsqueda "*country:ES*" en la web www.zoomeye.org devuelve 39.495.517 direcciones IPv4 (Figura 93) y en la aplicación casi la misma cantidad: 39.495.524 (Figura 94).

Figura 93: Búsqueda en www.zoomeye.org de "*country:ES*"

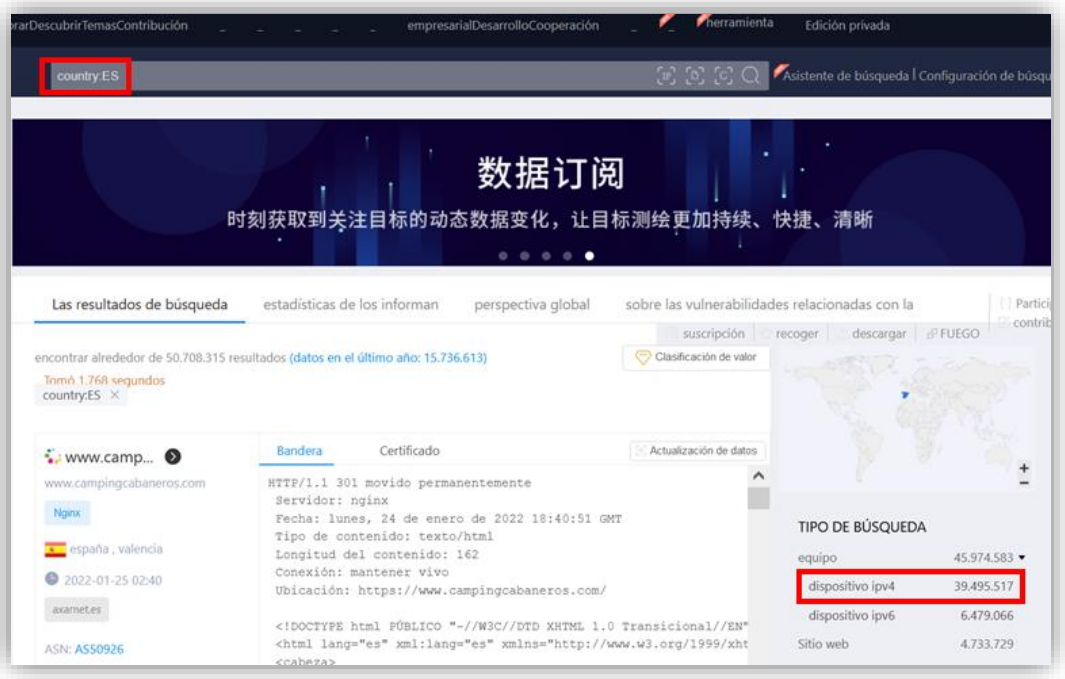
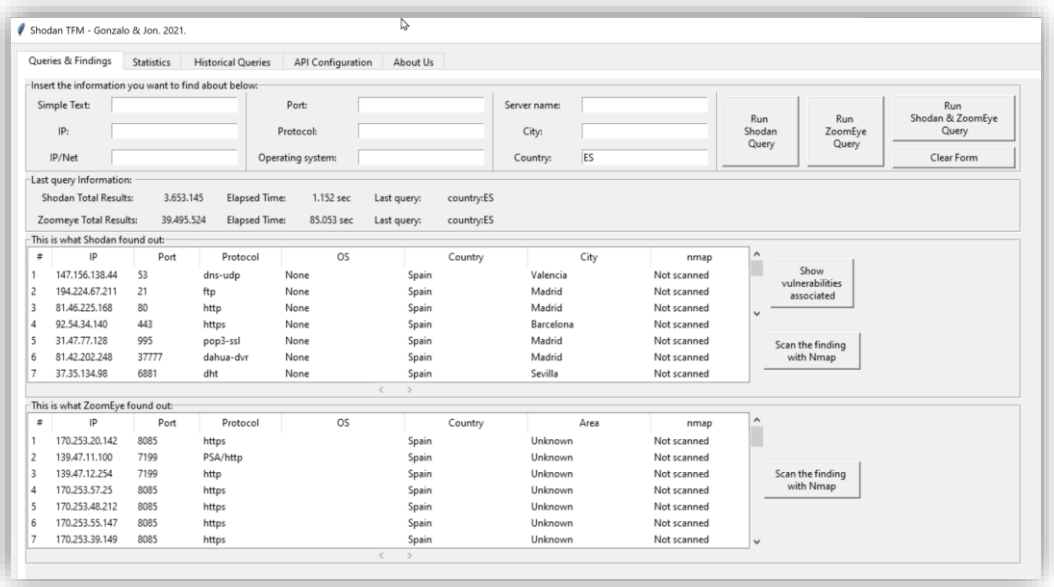


Figura 94: Búsqueda en DÉDALO de IPs con origen en España ("*country:ES*")



Después de buscar las IPs localizadas en España, parece demostrado que DÉDALO funciona correctamente. Las pruebas se han repetido en días posteriores y se han obtenido los mismos resultados. Ante las evidencias presentadas, la conclusión preliminar es que parece posible que ZoomEye esté bloqueando los datos con origen China cuando se hacen búsquedas utilizando la API pero no los bloquea cuando se buscan en su página web, www.zoomeye.org.

8. CONCLUSIONES Y TRABAJO FUTURO

En este último apartado, se describen tanto las principales conclusiones obtenidas tras la realización del proyecto como posibles líneas futuras de trabajo que se pueden llevar a cabo para evolucionar la aplicación implementada.

8.1.CONCLUSIONES PRINCIPALES

En primer lugar, se ha realizado un estudio del **estado del arte** para entender el contexto en el que se mueven las **herramientas o motores de búsqueda de dispositivos IoT**. Para ello, se han buscado los artículos y artículos académicos más relevantes. De este análisis, se ha tomado la decisión de explotar más específicamente las dos herramientas que, a priori, más dispositivos tienen capacidad de encontrar en el mercado: **Shodan y ZoomEye**. Algunos estudios apuntaban que ZoomEye tenía una mayor capacidad de búsqueda que Shodan, y este ha sido uno de los aspectos que se han ido analizando.

Además, se ha conseguido desarrollar una aplicación que hace uso de los **dos motores de búsqueda** mencionados y que explota varias funcionalidades de estas.

En cuanto a la **usabilidad** de DÉDALO desarrollada, es de **fácil uso** y no hace falta un conocimiento extenso de cada una de las herramientas para poder utilizarla. Esto hace que las búsquedas se faciliten porque el usuario no tiene la necesidad de comprender el lenguaje de etiquetas o filtros nativos de cada una de las aplicaciones, ya que la aplicación se encarga de construir las búsquedas a realizar basándose en un *input* de usuario limitado.

Adicionalmente, DÉDALO **consume pocos recursos** y es capaz de ejecutarse en entornos estándares, por lo que no es necesario disponer de una computadora con requisitos especiales para utilizarla.

Por otra parte, la aplicación es **capaz de sacar las vulnerabilidades asociadas** de tal manera que el usuario en todo momento es capaz de saber si un resultado es explotable a vulnerabilidades conocidas o no tiene dicho riesgo.

La aplicación tiene también una **verificación o chequeo externo mediante el uso de Nmap**, que permite conocer el estado de un puerto. Esto puede ayudar al usuario a saber si un resultado encontrado es correcto o incorrecto mediante un verificador externo.

DÉDALO es capaz, además, de realizar **búsquedas estadísticas y agrupar conceptos** para dar una foto general del estado de una búsqueda concreta y poder comparar los resultados obtenidos en una y otra herramienta. Adicionalmente, es capaz de **realizar búsquedas históricas** para saber la evolución de los resultados obtenidos. Esto puede ayudar a equipos IT o seguridad a hacer **seguimiento de la evolución del estado de su red** corporativa, por ejemplo.

En general, la aplicación puede **ayudar a comprender el funcionamiento y la lógica de las herramientas** Shodan y ZoomEye.

Relacionado con el desarrollo de DÉDALO, está **implementada en un lenguaje de uso muy extendido** como es Python y almacenada en un repositorio GitHub, por lo que sus **funcionalidades pueden ser ampliadas por parte de otros usuarios** que quieran evolucionar sus capacidades.

Además de las capacidades, funcionalidades y usabilidad de la aplicación, también se han **propuesto varios tipos de análisis** para los que esta puede ser utilizada. En primer lugar, DÉDALO sirve para **ayudar en el descubrimiento de dispositivos expuestos en una red conocida**. Esto ayuda a organizaciones y usuarios particulares tener controlado el parque expuesto. En segundo lugar, la aplicación permite resumir las estadísticas y resultados aparecidos en este tipo de motores de búsqueda de dispositivos, por lo que **puede ayudar a investigadores a entender el funcionamiento de las herramientas** de este tipo. También aporta datos en cuanto a tiempo de cómputo necesario para completar búsquedas. Por último, se ha realizado un análisis de un posible **ocultamiento de información** por parte de ZoomEye relacionado con dispositivos de China.

Algunos puntos reseñables acerca de la experiencia de haber utilizado ambas herramientas son los siguientes:

- Dado que Shodan y ZoomEye son herramientas de *footprinting*, su uso es completamente legal, ya que evitan el acceso directo a los sistemas estudiados. Sin embargo, para validar la información, puede ser necesario utilizar herramientas de *fingerprinter* como Nmap, que permiten tener una segunda capa de chequeo para poder aportar mayor fiabilidad en el resultado obtenido.

- Los datos que devuelven Shodan y ZoomEye deben ser interpretados. Un conocimiento profundo de cómo realizan los sondeos y almacenan la información permitirá un mejor aprovechamiento de los datos.

Por último, algunas conclusiones interesantes acerca de los análisis realizados en el apartado de *Escenarios de* aplicación prácticos de DÉDALO son las siguientes:

- Aunque en esencia la información que almacenan Shodan y ZoomEye es similar, el análisis de las búsquedas estadísticas demuestra que ambas plataformas procesan de forma distinta la información. Una vez más, es importante conocer cómo trabaja cada plataforma para aprovechar mejor los datos.
- ZoomEye devuelve información de todos los dispositivos que en algún momento han sido detectados, estén actualmente abiertos o no, mientras que Shodan solo devuelve los que en el último sondeo se hayan encontrado abiertos.
- ZoomEye siembra algunas dudas. Estas cuestiones son motivadas probablemente por la falta de información que se tiene acerca de esta herramienta. Una de las cuestiones más relevantes es que la API parece no devolver resultados localizados en China y que se ha comprobado en uno de los casos de estudio.

8.2. LÍNEAS DE TRABAJO FUTURO

En este subapartado se plantean una serie de posibles líneas de investigación futura. Uno de los objetivos de este proyecto ha sido el realizar un aporte a la comunidad en materia de descubrimiento de dispositivos IoT y la comparación de esos tipos de motores de búsqueda.

A continuación, se listan varias propuestas para continuar el trabajo comenzado y seguir aportando conocimiento a la comunidad:

- **Aumentar funcionalidades gracias a las posibilidades ofrecidas por Shodan y ZoomEye**. En el trabajo expuesto en el presente documento se han tenido en cuenta algunas capacidades que permiten explotar las herramientas, pero estas disponen de más. Se puede seguir aportando a la comunidad tratando de integrar otras capacidades ofrecidas por las APIs de las herramientas.
- **Investigar cómo puede ayudar ZoomEye a buscar *exploits* o CVEs**. DÉDALO es capaz de buscar las vulnerabilidades que Shodan asocia a los dispositivos encontrados. Sería

interesante comprobar si ZoomEye también ofrece ese tipo de información y compararla con la de Shodan.

- **Aumento de resultados obtenidos de las herramientas**. Actualmente la aplicación limita a 100 los resultados que se obtienen en las búsquedas realizadas tanto a Shodan como a ZoomEye. Esto ha sido derivado por la naturaleza de "prototipo" de la aplicación y por las necesidades de licenciamiento que se tiene.
- **Aumento de los campos o filtros a buscar**. Actualmente DÉDALO dispone de nueve campos para realizar búsquedas (texto simple, IP, red, puerto, protocolo, sistema operativo, nombre de dispositivo, ciudad y país). Se pueden implementar campos adicionales para que el usuario pueda introducir datos adicionales y completar búsquedas aún más complejas.
- **Integración de otros motores de búsquedas**. Otro evolutivo interesante puede ser añadir a DÉDALO el uso de otros motores de búsqueda de dispositivos. Por ejemplo, se podrían implementar las mismas funcionalidades existentes para Censys. De esta manera, se dispondría de una herramienta que compare o busque resultados en Shodan, ZoomEye y Censys.
- **Investigación de métodos de comprobación o verificación de los resultados**. DÉDALO utiliza Nmap para ejecutar una segunda capa de verificación acerca de los resultados obtenidos. Sin embargo, existe la posibilidad de integrar más herramientas de este tipo, por ejemplo, telnet. Se debe tener en cuenta el riesgo que conlleva el escaneo de puertos, que puede no ser legal en algunas legislaciones de no tener la aprobación del titular del servidor estudiado.
- **Adición de interfaces específicas para casos de estudios particulares**. Otro posible evolutivo puede tratarse de diseñar interfaces específicas que permitan ejecutar otros casos de estudio concretos.
- **Mejora de la detección de vulnerabilidades**. Actualmente DÉDALO dispone de una funcionalidad que permite visualizar las vulnerabilidades (en forma de código CVE y nombre de la misma) asociadas a los resultados obtenidos. Sin embargo, esta interfaz puede evolucionar para aportar más datos, calcular un CVSS en base a métricas ambientales, etc.
- **Implementar un sistema de búsquedas periódicas con avisos a usuarios**. Otro evolutivo de interés, especialmente para organizaciones, puede ser la implementación

de un sistema de búsquedas periódicas lanzando la aplicación en 24x7 de tal manera que genere avisos automáticos a usuarios (por ejemplo, mediante correo electrónico) cuando se encuentre un nuevo dispositivo expuesto bajo alcance. Esto puede ayudar a las organizaciones a monitorizar en tiempo real que su parque de dispositivos públicos está controlado y no hay, por ejemplo, aperturas de puertos que no deberían producirse.

- **Profundización en la comparativa entre Shodan y ZoomEye**. Se pueden plantear líneas de investigación que consigan encontrar más respuestas acerca de las diferencias entre las herramientas.
- **Estudio del desempeño de Shodan y ZoomEye según la zona geográfica desde la que se hace la consulta**. ZoomEye presenta mucha más variabilidad que Shodan en los tiempos de respuesta para la misma consulta lanzada en momentos diferentes. También se ha encontrado que la API de ZoomEye no devuelve registros de IPs localizadas en China. Un punto interesante sería conocer si cambiaría algo que se hicieran las pruebas desde China, físicamente o a través de una VPN.
- **Generación de documentación en ZoomEye**. Uno de los problemas al que este trabajo ha tenido que hacer frente ha sido a la falta de información relacionada con el uso de ZoomEye. Adicionalmente, no se han encontrado demasiados artículos de investigación relativos a ZoomEye. Por ello, se puede continuar generando más conocimiento relativo a esta herramienta.

Por supuesto, se pueden abrir más líneas de investigación a partir del trabajo realizado para continuar aportando a la comunidad de la ciberseguridad mayor conocimiento y capacidad de defensa.

Referencias bibliográficas

20minutos. (2021, noviembre 1). *Desalojan a casi 2.900 personas de cuatro locales de Sevilla en los que se celebraban fiestas de Halloween*. www.20minutos.es - Últimas Noticias. https://www.20minutos.es/noticia/4875105/0/desalojan-a-casi-2-900-personas-de-cuatro-locales-de-sevilla-en-los-que-se-celebraban-fiestas-de-halloween/

AlFardan, N., Bernstein, D. J., Paterson, K. G., Poettering, B., & Schuldt, J. C. N. (2013). *On the Security of {RC4} in {TLS}*. 305-320. https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/alFardan

Bada, M., & Pete, I. (2020). An exploration of the cybercrime ecosystem around Shodan. *2020 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, 1-8. https://doi.org/10.1109/IOTSMS52051.2020.9340224

Čolaković, A., & Hadžialić, M. (2018). Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues. *Computer Networks*, *144*, 17-39. https://doi.org/10.1016/j.comnet.2018.07.017

Durumeric, Z., Adrian, D., Mirian, A., Bailey, M., & Halderman, J. A. (2015). A Search Engine Backed by Internet-Wide Scanning. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 542-553. https://doi.org/10.1145/2810103.2813703

Filter Reference. (s. f.). Recuperado 30 de septiembre de 2021, de https://beta.shodan.io/search/filters

FOCA (Fingerprinting Organizations with Collected Archives). (2021). [C#]. ElevenPaths. https://github.com/ElevenPaths/FOCA (Original work published 2017)

Genge, B., & Enăchescu, C. (2016). ShoVAT: Shodan-based vulnerability assessment tool for Internet-facing services. *Security and Communication Networks*, *9*(15), 2696-2714. https://doi.org/10.1002/sec.1262

Google Hacking. (2021). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Google_Hacking&oldid=139790402

Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P., & Sikdar, B. (2019). A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures. *IEEE Access*, *7*, 82721-82743. https://doi.org/10.1109/ACCESS.2019.2924045

Internet of Things (IoT)—Definition. (s. f.). Recuperado 17 de enero de 2022, de https://www.trendmicro.com/vinfo/us/security/definition/Internet-of-things

lanmaster53. (2021). *The Recon-ng Framework* [Python]. https://github.com/lanmaster53/recon-ng (Original work published 2019)

Li, R., Shen, M., Yu, H., Li, C., Duan, P., & Zhu, L. (2020). A Survey on Cyberspace Search Engines. En W. Lu, Q. Wen, Y. Zhang, B. Lang, W. Wen, H. Yan, C. Li, L. Ding, R. Li, & Y. Zhou (Eds.), *Cyber Security* (Vol. 1299, pp. 206-214). Springer Singapore. https://doi.org/10.1007/978-981-33-4922-3_15

Li, X., Wang, Y., Shi, F., & Jia, W. (s. f.). *Crawler for Nodes in the Internet of Things*. 5.

Madakam, S., Ramaswamy, R., & Tripathi, S. (2015). Internet of Things (IoT): A Literature Review. *Journal of Computer and Communications*, *03*(05), 164-173. https://doi.org/10.4236/jcc.2015.35021

Margolis, J., Oh, T. T., Jadhav, S., Kim, Y. H., & Kim, J. N. (2017). An In-Depth Analysis of the Mirai Botnet. *2017 International Conference on Software Security and Assurance (ICSSA)*, 6-12. https://doi.org/10.1109/ICSSA.2017.12

Matherly, J. (2020, mayo 21). *What is a banner?* Shodan Blog. http://blog.shodan.io/what-is-a-banner/

Newman, P. (s. f.). *THE INTERNET OF THINGS 2020: Here's what over 400 IoT decision-makers say about the future of enterprise connectivity and how IoT companies can use it to grow revenue*. Business Insider. Recuperado 17 de enero de 2022, de https://www.businessinsider.com/internet-of-things-report

Nmap: The Network Mapper—Free Security Scanner. (s. f.). Recuperado 24 de enero de 2022, de https://nmap.org/

On-Demand Scanning—Shodan Help Center. (s. f.). Shodan Help Center. Recuperado 29 de septiembre de 2021, de https://help.shodan.io/the-basics/on-demand-scanning

OSINT - La información es poder. (2014, mayo 28). INCIBE-CERT. https://www.incibe-cert.es/blog/osint-la-informacion-es-poder

OSINT, ¿Qué es? ¿Para qué sirve? | Derecho de la Red. (s. f.). Recuperado 17 de enero de 2022, de https://derechodelared.com/osint/

Palma, A. J. P., Jiménez, A. N., & Díaz, J. A. I. (s. f.). *Complemento de VirusTotal para Maltego VirusTotal plugin for Maltego*. 75.

Punkspider. (s. f.). Recuperado 18 de enero de 2022, de https://punkspider.org/

Python. (2021). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Python&oldid=139257175

Research Access to Censys Data. (s. f.). Censys. Recuperado 3 de octubre de 2021, de https://support.censys.io/hc/en-us/articles/360038761891-Research-Access-to-Censys-Data

Schwaber, K. (1997). SCRUM Development Process. En J. Sutherland, C. Casanave, J. Miller, P. Patel, & G. Hollowell (Eds.), *Business Object Design and Implementation* (pp. 117-134). Springer London. https://doi.org/10.1007/978-1-4471-0947-1_11

Shodan Credits Explained—Shodan Help Center. (s. f.). Recuperado 2 de octubre de 2021, de https://help.shodan.io/the-basics/credit-types-explained

Shodanier—A expanded version of Shodan search engine. (s. f.). Recuperado 18 de enero de 2022, de https://sinister.ly/Thread-Shodanier-A-expanded-version-of-Shodan-search-engine?page=4

The ZMap Project. (s. f.-a). Recuperado 3 de octubre de 2021, de https://zmap.io/

The ZMap Project. (s. f.-b). Recuperado 17 de enero de 2022, de https://zmap.io/

TheHarvester. (s. f.). Recuperado 17 de enero de 2022, de http://www.edge-security.com/theharvester.php

Tkinter—Interface de Python para Tcl/Tk—Documentación de Python—3.10.0. (s. f.). Recuperado 9 de noviembre de 2021, de https://docs.python.org/es/3/library/tkinter.html

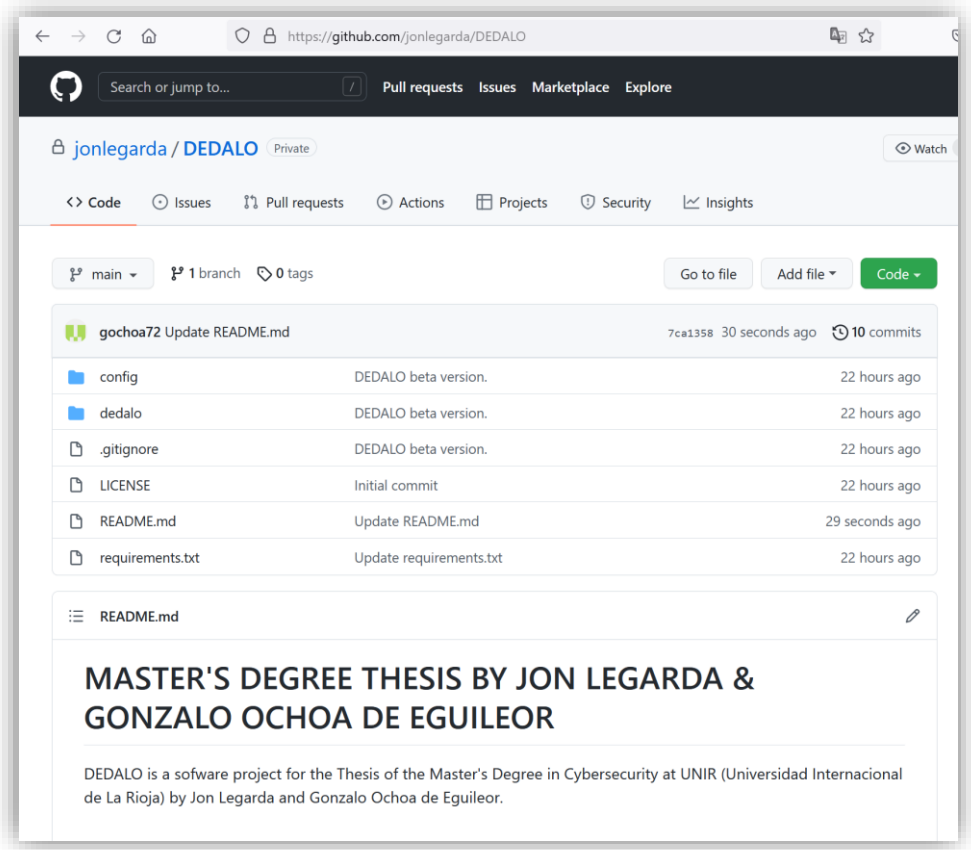
ZoomEye—Cyberspace Search Engine. (s. f.). *www.zoomeye.org*. https://www.zoomeye.org/about

ZoomEye-python. (2021). [Python]. Knownsec, Inc. https://github.com/knownsec/ZoomEye-python (Original work published 2016)

ANEXO A: Repositorio GitHub de DÉDALO

La aplicación DÉDALO está alojada la página del repositorio GitHub www.github.com/jonlegarda/DEDALO (Figura 95). Contiene todos los archivos necesarios para su ejecución y las instrucciones para descargarlo localmente y hacerlo funcionar.

Figura 95: Página del proyecto DÉDALO en GitHub



Fuente: www.github.com/jonlegarda/DEDALO

En el repositorio se encuentra el siguiente contenido:

- Carpeta *config*: contiene el archivo de configuración de la aplicación.
- Carpeta *dedalo*: en ella se encuentran los archivos con el código fuente en Python.
- Archivo *.gitignore*: se utiliza para que la sincronización del repositorio con un disco local ignore ciertos archivos que no deben ser actualizados.
- Archivo *LICENSE*: información sobre la licencia bajo la que se ha publicado la aplicación.
- Archivo *README.md*: Descripción del proyecto que se muestra en el repositorio y que GitHub muestra por defecto en la página principal del proyecto.

- Archivo *requirements.txt*: Fichero que contiene todas las librerías que deben ser instaladas y se puede utilizar como parámetro para que el comando de Python '*pip install*' instale todas las dependencias de una sola vez.

El archivo *README.md* contiene la presentación del proyecto y las instrucciones que permiten su ejecución (Figura 96).

Figura 96: Contenido del archivo *README.md*

MASTER'S DEGREE THESIS BY JON LEGARDA & GONZALO OCHOA DE EGUILEOR

DEDALO is a sofware project for the Thesis of the Master's Degree in Cybersecurity at UNIR (Universidad Internacional de La Rioja) by Jon Legarda and Gonzalo Ochoa de Eguileor.

DEDALO

DEDALO is a python application that uses Shodan's and ZoomEye's APIs in order to search IoT Devices over the Internet.

Usage

Cloning the repo.

```
git clone https://github.com/jonlegarda/DEDALO
cd dedalo
python main.py
```

Before running DEDALO you must set your own API KEYS. Please edit config/config.cfg and replace "ENTER_YOUR_SHODAN_API_KEY" and "ENTER_YOUR_ZOOMEYE_API_KEY" by the keys provided by Shodan and Zoomeye

[SHODAN] API_SHODAN = ENTER_YOUR_SHODAN_API_KEY

[ZOOMEYE] API_ZOOMEYE = ENTER_YOUR_ZOOMEYE_API_KEY

Dependencies

Besides the Shodan and Zoomeye module DEDALO makes use of other libraries that need to be installed in order to work. Please use `pip` or use the `requirements.txt` to install the dependencies.

```
pip install shodan
pip install zoomeye-SDK
pip install python-nmap
pip install configparser
pip install IPy
```

OR

```
pip install -r requirements.txt
```

Fuente: www.github.com/jonlegarda/DEDALO

DÉDALO debe ser descargado para su ejecución. En GitHub, la descarga toma el nombre de "clonado". Para clonar DÉDALO hay que lanzar el siguiente comando:

```
git clone https://github.com/jonlegarda/DEDALO
```

Una vez descargado el proyecto, simplemente hay que desplazarse a la carpeta principal y ejecutar el *main.py*:

```
cd dedalo
python main.py
```

Para que DÉDALO funcione correctamente, previamente hay que preparar su entorno de ejecución.

En primer lugar, se debe introducir en el archivo *config.cfg* las *API KEYs* que se obtienen mediante las cuentas personales en Shodan y ZoomEye. Para ello debe abrirse el archivo config/config.cfg y sustituir los textos "*ENTER_YOUR_SHODAN_API_KEY"* y "*ENTER_YOUR_ZOOMEYE_API_KEY"* por los valores de las claves.

En segundo lugar, hay que instalar las librerías no estándar de Python. Las librerías se especifican en el archivo *requirements.txt* y se podrían instalar con la instrucción: `pip install -r requirements.txt` También es posible instalarlas una por una, utilizando la siguiente cadena de comandos:

```
pip install shodan
pip install zoomeye-SDK
pip install python-nmap
pip install configparser
pip install IPy
```

En el archivo *.gitignore* se han incluido las carpetas que van guardando los resultados de las consultas en formato *JSON* (Figura 97). Estas carpetas se generan dinámicamente en la aplicación y no es necesario descargarlas localmente para ejecutar DÉDALO.

Figura 97: Contenido del archivo *.gitignore*

DÉDALO ha sido licenciado bajo **GNU General Public License v3.0**.

Para más información, consultar www.gnu.org/licenses/gpl-3.0.html.