

HERRAMIENTA PARA ENUMERAR SERVICIOS WEB DE UNA ORGANIZACIÓN



TRABAJO FIN DE MÁSTER

CURSO 2024-2025

Autor: Ferran Bernabé Puigmartí

Tutor del TFM: Pablo González

9aEd. Máster Profesional en Seguridad Ofensiva (OSCP)

Universidad Católica de Murcia

Indice

1. Estado del arte	1
2. Motivación	11
3. Hipótesis	15
4. Tesis	16
5. Planificación	18
6. Implementación	22
6.1. Backend	22
6.2. Modos de ejecución	33
6.3. Frontend	42
6.4. Instalación de la herramienta	45
6.5. Ejecución del backend	48
6.6. Uso práctico de la herramienta	53
7. Conclusiones	56
8. Bibliografía	58

1. Estado del arte

En ciberseguridad, una de las etapas críticas para los analistas es la enumeración. Esta fase es inmensa y varía en función del objetivo. Puede ir desde la identificación de dispositivos conectados a Internet, ejercicios *OSINT*, detección de puertos abiertos en los servidores, descubrimiento de *endpoints* en servidores web, enumeración de configuraciones en ejercicios sobre *Active Directory*...

Si nos centramos en ejercicios externos de caja negra, en los que no se dispone de información por anticipado sobre los objetivos y no tenemos acceso a la red interna de la organización, la enumeración de activos expuestos es un proceso muy lento y tedioso. Por ejemplo, la primera fase de cualquier programa *Bug Bounty* basado en *Open Scope* o *Wildcard Domains* [1] consiste en encontrar qué dispositivos o dominios están expuestos. Aunque es muy típico tener esta fase en los programas *Bug Bounty*, también es importante en ejercicios de *pentesting* y en ataques llevados a cabo por actores maliciosos. Esta fase, conocida en *Bug Bounty* como *go wide*, se basa en ampliar la superficie de ataque para disponer de más caminos diferentes por los que intentar explotar vulnerabilidades en los activos de la organización.

Actualmente existen dos grandes metodologías principales para descubrir servicios expuestos a Internet. La primera es un enfoque clásico, de carácter manual, en el que se utilizan plataformas como *Shodan* [3]. La segunda, y más novedosa, se basa en herramientas como las desarrolladas por *Project Discovery* [4], y otras similares, las cuales tienen un enfoque más automatizado. Para analizar los pros y los contras de ambas metodologías, he seleccionado [el programa Bug Bounty de booking.com](#) para realizar todas las pruebas de este estudio.

Antes de enumerar los dispositivos con las dos metodologías mencionadas, revisemos primero el *scope* del programa. En primer lugar, podemos observar varios dominios individuales:

Asset name ↑	Type ↑	Coverage ↑	Max. severity ↓	Bounty ↑
supplier.auth.toag.booking.com	Domain	In scope	Critical	Eligible
experiences.booking.com	Domain	In scope	Critical	Eligible
webhooks.booking.com	Domain	In scope	Critical	Eligible

Fig 1. Dominios simples en el programa de Bug Bounty de Booking

En estos casos, la fase de descubrimiento de dominios o dispositivos expuestos a Internet no es necesaria, ya que el objetivo está claramente definido, un *endpoint* concreto. Ahora bien, el mismo programa también tiene dominios *wildcard*:

Asset name ↑	Type ↑	Coverage ↑	Max. severity ↓	Bounty ↑
*.fareharbor.engineering	Wildcard	In scope	Critical	Eligible
*.rentalcars.com if there's any vulnerabilities raised on this asset that are owned by a third party we will not be accepting those reports	Wildcard	In scope	Critical	Eligible

Fig2. Dominios wildcard en el programa de Bug Bounty de Booking

En estos casos, se aceptan reportes de cualquier subdominio de los indicados en el programa. Por ejemplo, para el dominio *booking.com*, podrían existir subdominios como *test.booking.com*, *admin.booking.com* o *api.v3.booking.com*. El analista de ciberseguridad deberá enumerar el mayor número posible de subdominios para ampliar la superficie de ataque, lo que generará más oportunidades de encontrar vulnerabilidades.

Finalmente, en el *scope* del programa se incluyen los activos de la organización que están *out of scope*, es decir, que no forman parte del programa de *Bug Bounty*. Esto implica que no pueden ser evaluados y que no se dará ninguna recompensa por descubrir vulnerabilidades en estos *endpoints*. Si analizamos los elementos marcados como *out of scope*, observamos que se listan algunos dominios y *endpoints* específicos, por ejemplo:

Fig 3. Dominio out of scope en el programa de Bug Bounty de Booking

En el programa no se especifica que cualquier otro dispositivo expuesto de la organización esté marcado como *out of scope*. Es decir, podríamos suponer que este programa podría tratarse como un *open scope*. Este tipo de programas permiten analizar cualquier activo de la empresa expuesto a Internet, sin definir dominios ni *endpoints* [5] específicos a evaluar.

En el caso concreto del programa de *Booking.com*, no se establece qué ocurre si se encuentra una vulnerabilidad en un activo que no está ni dentro del *scope* ni listado como *out of scope*. Por lo general, si se encuentra alguna vulnerabilidad en estos activos, el analista suele ser recompensado, aunque la decisión final depende de la organización. Para los tests de este estudio, asumiremos que este enfoque también es válido.

Así, tras analizar el programa, queda claro que debemos llevar a cabo una búsqueda de subdominios a partir de los *wildcards* y, además, identificar dispositivos de la organización expuestos a Internet, sin importar si pertenecen a un dominio marcado como *in scope*.

En primer lugar, se estudiará cómo enumerar activos de una organización utilizando el enfoque clásico, principalmente usando *Shodan*.

Shodan es un motor de búsqueda muy usado en ciberseguridad para indexar y catalogar dispositivos conectados a Internet, incluidos servidores, *routers* y diversos sistemas *IoT*. Así, mientras que los motores de búsqueda clásicos dependen de las palabras clave que teclean los usuarios, *Shodan* es un rastreador global que escanea continuamente la red global, recuperando información relativa a puertos abiertos, servicios activos y configuraciones de dispositivos expuestos. Esta información se obtiene de forma gratuita a través de su sitio web, lo que lleva a muchos analistas a optar por este tipo de información accesible de forma manual, aunque también es posible la automatización mediante una suscripción.

Es sin duda uno de los principales servicios utilizados para encontrar dispositivos de organizaciones. Ahora mostraré un pequeño ejemplo de cómo empezaríamos a enumerar algunos dispositivos y dominios expuestos con esta herramienta.

El primer paso consiste en analizar el certificado TLS de alguna página web de la organización. Por ejemplo, si accedemos a *booking.com* a través de su puerto HTTPS, observaremos la siguiente configuración del certificado:

Nombre común (CN)	*.booking.com
Organización (O)	Booking.com BV

Fig 4. Nombre común y Organización del certificado TLS de booking.com

Esta información nos sugiere que podremos usar *Shodan* para identificar dispositivos de forma más rápida que si no tuvieramos un nombre de la organización o un *Common Name* definidos en el TLS.

Por ejemplo, podríamos buscar subdominios con el siguiente operador [6]:

```
ssl.cert.subject.CN:"booking.com"
```

Fig 5. Operador *ssl.cert.subject.CN* de Shodan

Vemos los resultados de la búsqueda:

The screenshot displays two search results from Shodan. The first result is for 'Welcome to Keycloak' on IP 51.75.5.217, showing an API endpoint for booking.com with an SSL certificate issued by Let's Encrypt. The second result is for IP 5.57.16.164, showing the gw.booking.com subdomain with an SSL certificate issued by DigiCert. Both results include details about the certificate's common name, organization, and supported versions.

Fig 6. Primeros resultados de la búsqueda realizada en Shodan

De esta forma, si nos llama la atención el subdominio *gw.booking.com*, podríamos ver qué tiene indexado *Shodan* con un simple click:

5.57.16.164 Regular View Raw Data Timeline

General Information

Hostnames	booking.cn gw.booking.cn prod.gw.booking.cn booking.com gw.booking.com prod.gw.booking.com rentalcars.com
Domains	BOOKING.CN BOOKING.COM RENTALCARS.COM
Country	United Kingdom
City	London
Organization	Booking.com GB
ISP	Booking.com BV
ASN	AS43996

Fig 7. Salida de una búsqueda por IP en Shodan – Parte 1

Open Ports

80 443

// 80 / TCP

```

HTTP/1.1 403 Forbidden
x-denied-reason: src_is_not_cloudfront
content-length: 0
  
```

// 443 / TCP

```

HTTP/1.1 403 Forbidden
x-denied-reason: src_is_not_cloudfront
content-length: 0
  
```

SSL Certificate

Certificate:

Data:

```

Version: 3 (0x2)
Serial Number:
  07:fa:b6:11:d7:04:ca:76:47:78:31:cb:ff:7f:f8:a4
Signature Algorithm: sha256WithRSAEncryption
Issuer: C=US, O=DigiCert Inc, CN=DigiCert Global G2 TLS RSA SHA256 2020 CA1
Validity
  Not Before: Oct 17 00:00:00 2024 GMT
  Not After : Oct 16 23:59:59 2025 GMT
Subject: C=NL, L=Amsterdam, O=Booking.com BV, CN=*.gw.booking.com
Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
  Public-Key: (2048 bit)
  Modulus:
  
```

Fig 8. Salida de una búsqueda por IP en Shodan – Parte 2

También nos proporciona información sobre la fecha de actualización de estos resultados, que en este caso fue hace dos días. En *Shodan*, los datos pueden estar desactualizados por

varios días o semanas, ya que el *crawler* recorre continuamente todo el rango de direcciones de Internet, un proceso que implica un alto coste computacional [7].

De este modo, podemos analizar la información almacenada sobre este *Common Name* e ir enumerando subdominios. Cabe destacar que, con el plan gratuito, los resultados están limitados a un máximo de 20 registros.

En el segundo caso, buscaremos dispositivos de la organización expuestos a Internet. Este filtrado también se basa en el análisis del certificado TLS, y *Shodan* permite buscar coincidencias dentro del apartado de organización de los certificados TLS en su base de datos.

De este modo, para identificar dispositivos pertenecientes a la empresa *Booking*, sin importar el dominio específico, podríamos utilizar el siguiente operador:

```
ssl:"Booking.com BV"
```

En los resultados podemos observar que encuentra dispositivos de la organización de varios dominios distintos:

<p>5.57.16.164 </p> <p>booking.cn booking.com gw.booking.com prod.gw.booking.cn Booking.com GB United Kingdom, London</p>	<p>SSL Certificate</p> <p>Issued By: - Common Name: DigiCert Global G2 TLS RSA SHA256 2020 CA1</p> <p> - Organization: DigiCert Inc</p> <p>Issued To: - Common Name: *.gw.booking.com</p> <p> - Organization: Booking.com BV</p> <p>Supported SSL Versions: TLSv1.2, TLSv1.3</p>	<p>HTTP/1.1 403 Forbidden x-denied-reason: src_is_not_cloudfront content-length: 0</p>
<p>403 Forbidden </p> <p>104.19.164.108 rentalcars.com Cloudflare, Inc. United States, San Francisco </p>	<p>SSL Certificate</p> <p>Issued By: - Common Name: DigiCert Global G2 TLS RSA SHA256 2020 CA1</p> <p> - Organization: DigiCert Inc</p> <p>Issued To: - Common Name: *.rentalcars.com</p> <p> - Organization: Booking.com BV</p> <p>Supported SSL Versions: TLSv1.2</p>	<p>HTTP/1.1 403 Forbidden Server: cloudflare Date: Tue, 04 Mar 2025 10:18:59 GMT Content-Type: text/html Content-Length: 553 Connection: keep-alive CF-RAY: 91b0a1fa2d24fa2e-SJC</p>

Fig 9. Salida de la búsqueda por organización en Shodan

Una de las funcionalidades más potentes de *Shodan* es la opción *Facet*, que permite filtrar y agrupar resultados de manera eficiente.

Por ejemplo, podríamos listar todos los dispositivos pertenecientes a la empresa *Booking*, excluyendo aquellos cuyo dominio sea *booking.com*, y filtrar la salida según el título de la respuesta HTTP:

```
ssl:"Booking.com BV" -Ssl.cert.subject.CN:"booking.com"
```

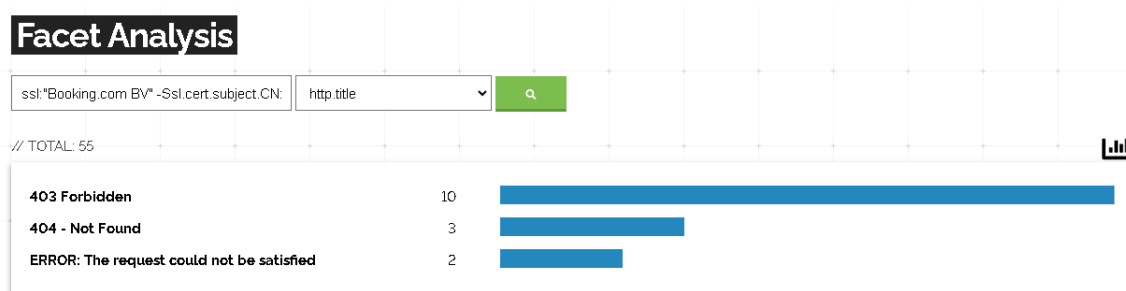


Fig 10. Facet Analysis de Shodan

De esta forma, podríamos filtrar los resultados por respuestas HTTP, países, ciudades, hashes, vulnerabilidades...

Shodan ofrece muchas más funcionalidades y operadores avanzados, lo que explica en parte su éxito en la industria. Sin embargo, no es la única opción a la hora de identificar dispositivos expuestos a Internet o de enumerar subdominios de forma manual.

Existen varias alternativas que pueden utilizarse con este propósito, como *Fofa*, *Censys*, *URLScan* o *VirusTotal*. Incluso el uso de *Google Dorking* [8] puede ser útil para la enumeración de subdominios:

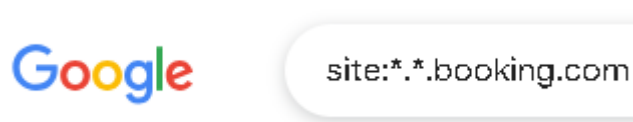


Fig 11. Google Dorking para buscar third-level-domains

Booking.com Taxi API: Getting Started

The Booking.com Taxi Supplier Public API allows our supply partners to **provide prices for journeys**, accept bookings, reject bookings and send driver events to ...

Perfil de usuario de Booking.com for Business

Para gestionar perfiles desde el menú de administración, ve a **Personas > Gestionar el equipo**. Haz clic en EDITAR en cualquiera de los perfiles que quieras ...

Fig 12. Salida de la búsqueda con el Dork de Google

La prueba podría ampliarse utilizando una combinación de diferentes métodos y herramientas, pero no es el objetivo de este trabajo. Así pues, la primera metodología nos ha permitido enumerar pasivamente varios subdominios y dispositivos sin interactuar con la organización, lo que significa que es difícil que dicha organización detecte que estamos recopilando información sobre ellos.

No obstante, esta metodología es manual, es decir, el analista deberá centrar sus esfuerzos en enumerar cualquier activo disponible. Por lo tanto, se desperdicia un tiempo y unas energías enormes que el analista podría haber empleado mejor en auditar los activos en vez de enumerarlos. Aunque *Shodan* y algunas de las otras herramientas descritas ofrecen una API para automatizar el proceso, el acceso a esa funcionalidad necesita de algún tipo de suscripción de nivel variable, en función de las capacidades que se requieran.

La segunda metodología surge precisamente como respuesta a la necesidad de reducir esta carga de trabajo para que el analista pueda enfocarse más en auditar los activos de la organización en lugar de centrarse en la fase de enumeración. Existen varias herramientas que permiten automatizar la búsqueda pasiva de subdominios, aunque son pocas las que resultan efectivas para programas de *open scope*, más allá de *Shodan* y las mencionadas anteriormente.

En este apartado, nos centraremos en algunas herramientas desarrolladas por *Project Discovery*, una empresa dedicada al desarrollo de herramientas de código abierto diseñadas para facilitar la enumeración y el descubrimiento de activos en ciberseguridad, entre otras funciones.

Entre sus herramientas más destacadas se encuentran *subfinder* [9], que recopila subdominios de forma pasiva utilizando fuentes públicas como certificados TLS y bases de datos de DNS, y *httpx* [10], que confirma la disponibilidad de estos subdominios mediante solicitudes HTTP/HTTPS. Este enfoque es ampliamente utilizado en ejercicios de *pentesting* y programas de *Bug Bounty* debido a su flexibilidad y capacidad de automatización, ofreciendo una alternativa moderna a los enfoques tradicionales.

En este apartado se emplearán únicamente dos herramientas combinadas para evaluar su potencial, *Project Discovery* ofrece muchas otras herramientas útiles para distintas fases de una auditoría de seguridad. Algunas de ellas son *katana* [11], diseñada para descubrir *endpoints* en servidores web, *nuclei* [12], un escáner masivo de vulnerabilidades con *templates* públicos y fácilmente programables, o *naabu* [13], un escáner de puertos.

Project Discovery es una empresa muy activa actualmente en el ámbito de la ciberseguridad, y, debido a su creciente popularidad, es fácil pensar que en un futuro desarrollarán nuevas herramientas útiles para este campo. De esta forma, realizaremos una prueba con sus herramientas *subfinder* y *httpx*, que son útiles para programas de *Wildcard Domain*, pero no se realizarán pruebas para programas *Open Scope*, ya que, por el momento, no disponen de soluciones específicas para la identificación de activos en redes IPv4 únicamente.

Con *subfinder* podemos listar subdominios de forma pasiva:

```
subfinder -d booking.com | tee -a rutas.txt
```

Fig 13. Comando usado para listar subdominios con *subfinder*

Se han enumerado 1686 subdominios:

```
# wc -l rutas.txt  
1686 rutas.txt
```

Fig 14. Número de subdominios enumerados con *subfinder*

Y ahora listamos qué subdominios están activos recopilando información sobre estos con *httpx*:

```
cat rutas.txt | httpx -sc -ip -tech-detect -silent -H 'X-Bug-Bounty: True' > httpx.txt
```

Fig 15. Ejecución de la herramienta *httpx* a partir del output de *subfinder*

Se han enumerado 268 dominios activos. Podemos ver la salida que se guarda en este archivo:

```
https://account.booking.com [405] [52.84.174.22] [AWS WAF Captcha,Amazon CloudFront,Amazon Web Services,HSTS]
https://about.booking.com [200] [66.33.60.129] [HSTS,Vercel]
https://about-staging.booking.com [200] [64.29.17.65] [HSTS,Vercel]
https://admin.api.booking.com [404] [18.164.52.83] [Amazon CloudFront,Amazon Web Services,HSTS,Nginx]
http://admin.booking.com [301] [13.249.9.110] [Amazon CloudFront,Amazon Web Services]
https://affiliates.booking.com [302] [52.84.174.51] [Amazon CloudFront,Amazon Web Services,HSTS,Nginx]
https://agencywise.booking.com [302] [216.198.53.1] [Cloudflare,Cloudflare Bot Management,HSTS,Zendesk]
https://affiliates.support.booking.com [301] [35.158.127.53] [HSTS]
https://api-lhr4.booking.com [404] [18.164.52.45] [Amazon CloudFront,Amazon Web Services,HSTS,Nginx]
```

Fig 16. Salida de los subdominios activos enumerados con httpx

Esta metodología es bastante rápida, pero por sí sola no permite filtrar resultados ni enumerar programas de *open scope*.

Existen numerosas herramientas que pueden utilizarse con el propósito de enumerar activos de una organización, muchas de las cuales no pertenecen a *Project Discovery*. Sin embargo, las herramientas de esta organización han ganado una gran popularidad y son muy utilizadas en programas de *Bug Bounty* en la actualidad.

En la siguiente sección, analizaré las debilidades y limitaciones de las metodologías y herramientas descritas en este estado del arte y así motivar la propuesta de la herramienta presentada en este trabajo. Este análisis justificará la necesidad de buscar nuevas soluciones para mejorar los procesos de enumeración de dispositivos y subdominios expuestos a Internet.

2. Motivación

En el apartado anterior se han presentado las metodologías disponibles para enumerar activos de una organización. Ambas metodologías presentan debilidades que se tratarán de enumerar en este apartado. En cuanto al enfoque clásico, utilizando herramientas como *Shodan* y similares, se ha visto que se requiere de una enumeración manual por parte del analista. Este hecho hace que se necesite una gran cantidad de tiempo de análisis para la enumeración, que dependerá en parte de la infraestructura de la organización. Por lo tanto, la expansión de la superficie de ataque se convierte en un cuello de botella, malgastando mucho tiempo del analista, dejando poco tiempo para cualquier análisis más profundo de posibles vulnerabilidades o configuraciones erróneas en cualquiera de los activos descubiertos. La solución a esta limitación es hacer que la enumeración se lleve a cabo automáticamente para que el analista pueda centrarse en la parte más crítica del análisis, es decir, encontrar y explotar las vulnerabilidades.

Shodan ofrece una solución a este problema, ya que permite automatizar esta fase mediante su API, pero esta funcionalidad es de pago con suscripciones mensuales para acceder a ella. Si bien puede ser adecuado para escenarios de *Bug Bounty* en organizaciones pequeñas, para organizaciones grandes sería necesario realizar múltiples solicitudes, por lo que se necesitaría una suscripción más alta, debido a la demanda de consultas. Otro problema de *Shodan* es el número limitado de resultados. Esto hace que el trabajo de enumerar todos los subdominios de una organización sea largo y difícil, dado que con una cuenta del plan gratuito se podría obtener bastante menos información de la disponible en la base de datos del servicio. Además, sería casi imposible filtrar esos resultados por organización si ésta no rellena el campo del certificado TLS en cuestión o reutiliza *favicons*, lo que limitaría aún más la eficacia de la herramienta.

Otra faceta a tener en cuenta es que, dado que *Shodan* no realiza escaneos en tiempo real, los resultados podrían estar desfasados algunos días, o incluso semanas. Esto afecta a la eficacia de la herramienta en entornos dinámicos, como los de los programas *Bug Bounty* con dominios *Wildcard*, en los que los servidores cambian rápidamente de estado, lo que hace que se necesite información actualizada.

Sin embargo, los puntos mencionados anteriormente se derivan principalmente de la filosofía de negocio de herramientas como *Shodan*, pero existe una debilidad aún mayor, la falta de especificidad empresarial. El diseño de *Shodan* está orientado a realizar un

mapeo global de Internet, pero no está enfocado en proporcionar un análisis específico para una organización en particular. Esto genera un volumen de datos excesivo y, en muchos casos, poco práctico para objetivos concretos como los de un programa de *Bug Bounty*. Este enfoque amplio también limita la capacidad de *Shodan* para enumerar *virtual hosts* en un servidor. El escaneo activo de *virtual hosts* sería inviable debido al tiempo que tomaría, y además, muchas direcciones IP están protegidas por CDN o *firewalls* [14] que bloquearían el acceso de los *crawlers*. Por otro lado, un enfoque pseudo-pasivo aumentaría significativamente el tiempo necesario para analizar cada servidor, lo que afectaría negativamente el modelo de negocio de la herramienta.

En cuanto al enfoque más moderno, por ejemplo utilizando herramientas como las desarrolladas por *Project Discovery*, busca mitigar este problema de falta de especificidad empresarial que caracteriza al modelo tradicional. Estas herramientas son eficaces en el análisis de activos digitales de una organización. Sin embargo, también tienen algunos puntos débiles cuando se comparan con el enfoque tradicional. En primer lugar, el uso de herramientas como las de *Project Discovery* requiere un mayor conocimiento técnico y un enfoque más metodológico. Mientras que con *Shodan* basta con buscar un dominio para obtener resultados y subdominios, herramientas como *subfinder* y *httpx* exige que el analista tenga clara la metodología a seguir en cada fase del proceso. Por ejemplo, una metodología común sería primero enumerar todos los subdominios de manera pasiva, luego verificar cuáles de ellos siguen activos y, posteriormente, clasificarlos según su código de estado HTTP en diferentes archivos. Esta metodología necesita un conocimiento de la función de cada herramienta y de la metodología que se quiere seguir, lo que puede resultar un reto si el analista no tiene experiencia en el uso de estas herramientas. Además, es importante tener en cuenta las restricciones que puedan imponer los programas de *Bug Bounty*, ya que algunas de estas herramientas pueden realizar escaneos activos, lo cual podría estar prohibido dependiendo de las políticas del programa..

Finalmente, aunque las herramientas de *Project Discovery* se especializan principalmente en la enumeración de subdominios y servicios web específicos, no permiten enumerar dispositivos con un enfoque *Open Scope* en donde no se proporcionen dominios. Este hecho limita su capacidad para proporcionar un panorama completo de la infraestructura expuesta a Internet. Esto contrasta con enfoques más amplios como el de *Shodan*, que

realiza un mapeo global de todos los dispositivos expuestos, con las limitaciones ya mencionadas.

El análisis de ambos enfoques, tanto el clásico como el moderno, dejó claro que había que desarrollar una herramienta propia, una que combinara lo mejor de ambos mundos. Lo que propongo es mezclar ambos enfoques y el resultado final será una herramienta que mitigue los puntos débiles de estos enfoques, y sea adecuada a las necesidades propias en los programas *Bug Bounty* y otras auditorías de ciberseguridad.

La idea principal de esta herramienta es adoptar un enfoque más tradicional, lo que significa hacer solicitudes web a servicios disponibles en Internet, pero con un enfoque más preciso, dirigido a una organización o empresa en particular. Esto permitiría obtener resultados más específicos, en lugar de hacer un escaneo global como lo hace *Shodan*, ajustándose a la infraestructura de la organización objetivo. Sin embargo, este enfoque no se limitaría solo a escaneos activos, ya que también combinaría técnicas más modernas y pasivas que permiten explorar la infraestructura de la organización sin necesidad de interactuar de forma intrusiva con los sistemas. De esta manera, se minimiza el riesgo de detección y se maximiza la eficiencia del proceso.

En cuanto a la parte más moderna, mi herramienta utilizaría varias técnicas avanzadas para enumerar subdominios y *virtual hosts* de manera pseudo-pasiva, lo que evitaría el uso de escaneos activos masivos que puedan ser más fáciles de detectar y bloquear. Al emplear técnicas como la búsqueda en fuentes públicas de datos, como certificados TLS, registros DNS, y otras fuentes pasivas de información, la herramienta podría identificar subdominios y *virtual hosts* de forma rápida y eficiente sin necesidad de realizar escaneos excesivamente intrusivos. Esta metodología permitiría a los analistas centrar su esfuerzo en el análisis de vulnerabilidades y configuraciones erróneas, dejando atrás las tareas repetitivas y lentas de enumeración.

Además, uno de los grandes beneficios de esta combinación de enfoques es que la herramienta podría adaptarse a diferentes tipos de organizaciones y contextos. Mientras que el enfoque tradicional es adecuado para obtener una visión global de los activos expuestos, la integración con técnicas pasivas permitiría obtener una visión más detallada y precisa, especialmente en programas de *Bug Bounty* donde la información debe ser recopilada de manera discreta y con el menor impacto posible sobre los sistemas objetivo. La capacidad de automatizar estos procesos también sería importante, ya que aliviaría a

los analistas de las tareas manuales y les permitiría centrarse en la investigación de posibles vulnerabilidades y no en la enumeración de activos.

En pocas palabras, el desarrollo de esta herramienta no solo busca unir lo mejor de ambos enfoques, sino que también tiene como objetivo hacer más eficiente el proceso de recopilación y enumeración de información. Esto le ofrecerá a los analistas una solución más ágil, flexible y específica. Este enfoque es especialmente valioso en el ámbito de la ciberseguridad, como en los programas de *Bug Bounty*, donde la rapidez, precisión y discreción son esenciales debido a la alta competitividad en el sector. Al fusionar ambas metodologías en una única herramienta, se mejorará el rendimiento en las etapas iniciales de un *pentest* o auditoría de seguridad, permitiendo a los profesionales enfocarse en auditar los activos de la organización.

3. Hipótesis

En este trabajo, quiero presentar una solución que permita identificar de manera rápida y eficiente los servidores web que están expuestos a Internet y que están relacionados con una organización específica o con empresas que comparten infraestructura. Para ello, utilizaré como punto de partida el nombre de la empresa o un grupo de dominios.

La herramienta combinará técnicas de enumeración pasiva y automatización avanzada para reducir la necesidad de intervención manual y minimizar la intrusión en los sistemas que se están analizando. Mi objetivo es integrar métodos tradicionales, como la búsqueda de dispositivos a través de solicitudes web, con enfoques más modernos, como la enumeración de subdominios a partir de fuentes pasivas y la detección de *virtual hosts*. Todo esto con el fin de minimizar el tiempo de trabajo manual, enumerar activos y proporcionar resultados más actualizados y específicos para cada organización.

Si esta hipótesis resulta ser cierta, la herramienta facilitará la automatización de la enumeración en programas de *Bug Bounty* y pruebas de penetración. Esto no solo reducirá el uso de recursos, sino que también aliviará la carga de trabajo de los analistas de ciberseguridad.

Como resultado, espero desarrollar una interfaz web fácil de usar que permita al analista filtrar servidores según patrones simples. El objetivo final es que esta herramienta actúe como una guía, permitiendo al analista concentrarse en auditar la seguridad de los sistemas sin perder tiempo en la enumeración de estos.

4. Tesis

El propósito principal de este trabajo es crear una herramienta que ayude a los analistas de ciberseguridad a enumerar servidores web de una organización concreta. Esta herramienta está enfocada a auditorías externas, donde, al principio, se dispone de poca información sobre la infraestructura, por lo que la fase de enumeración se convierte en un proceso clave. Mediante el *frontend*, la herramienta permite a los analistas observar los activos expuestos de la organización de forma rápida y sencilla.

De esta forma, uno de los objetivos principales es que la herramienta ofrezca la mayor información posible sobre los servidores web, usando los métodos menos agresivos posibles. Así pues, permitirá combinar tanto el enfoque clásico como el moderno, buscando una automatización de la fase de enumeración.

Además, la herramienta aporta un enfoque innovador para descubrir *virtual hosts*, empleando una estrategia pseudo-pasiva. Para hacerlo posible, se crean listas de posibles *virtual hosts* sin tener que recurrir a la tradicional fuerza bruta con diccionarios predefinidos [15]. Gracias al enfoque programado, se reduce el riesgo de sobrecargar o activar alertas en los sistemas objetivo.

En cuanto a su utilidad, esta herramienta ha sido creada para ser útil en contextos específicos, como las auditorías externas, especialmente en programas de *Bug Bounty*. Por ende, no solo se puede utilizar en empresas con una infraestructura bien definida, sino también en aquellas donde la información disponible es escasa o fragmentada. Además, la herramienta puede ser muy valiosa en situaciones donde se tienen algunos dominios listados pero la organización no tiene redes propias a enumerar.

La herramienta cuenta con varios modos de ejecución, ajustándose a las necesidades del analista. Algunos de estos modos se centran en recopilar la mayor cantidad de información posible, aunque esto puede llevar a falsos positivos. Por otro lado, hay modos que se enfocan en ofrecer resultados más precisos, aunque con un alcance más limitado. Esta variedad permite que el analista tenga una gran flexibilidad para decidir qué tipo de información le es conveniente en cada situación.

En cuanto al desarrollo, se ha basado el código en la programación orientada a objetos (OOP) [16], lo que le da una estructura ordenada y escalable. Además, se han llevado a cabo pruebas unitarias [17] para comprobar que cada componente funcione como debe y

cumpla con los requisitos establecidos. Para mejorar la velocidad de procesamiento, se ha incorporado programación concurrente [18], especialmente en tareas que requieren de mucho recurso, como las peticiones web y las resoluciones DNS [19]. Esto garantiza que la herramienta pueda manejar grandes volúmenes de datos sin sacrificar excesivamente su rendimiento.

En cuanto a la evaluación de la herramienta, se han definido indicadores de rendimientos, como son el tiempo de ejecución del *backend*, las direcciones IP encontradas, número de *hosts* enumerados, y los *virtual hosts* encontrados. Es difícil comparar métodos de ejecución ya que los resultados pueden diferir según el tipo de infraestructura, momento en el que se ejecuta la herramienta, estado de la red del auditor... Y también es difícil medir el éxito de la herramienta entre diferentes organizaciones, por ejemplo, las empresas con un mayor número de servidores web activos generarán más resultados, mientras que aquellas con infraestructuras más pequeñas o menos expuestas obtendrán resultados más limitados.

La herramienta se ha probado en diferentes programas de *Bug Bounty*. Para este TFM, se han realizado las pruebas sobre la empresa *Booking*. Además, los resultados obtenidos en estos entornos han demostrado que la herramienta es muy eficaz para identificar servidores web, *virtual hosts* y otros activos, con un esfuerzo mínimo por parte del analista.

Esta herramienta está diseñada principalmente para analistas de seguridad, conocidos como *bug hunters*, que participan en programas de *Bug Bounty* y buscan identificar activos y vulnerabilidades en servidores web. Más concretamente, a aquellos *bug hunters* que se centran en la metodología *go wide*. Aún así, también puede ser útil para *pentesters* y administradores de seguridad, quienes pueden aprovechar su capacidad para descubrir infraestructuras web. Pese a que el objetivo es promover una mayor seguridad cibernética, también puede ser usada por cibercriminales.

En resumen, la herramienta permite automatizar y acelerar el descubrimiento de servidores web, proporcionando a los analistas de ciberseguridad una forma de realizar auditorías externas combinando los enfoques clásicos y modernos, recolectando la máxima información con la menor intrusividad posible. De esta forma, permite servir de guía para que el analista decida qué servidores auditar, a su vez, reduciendo los riesgos que conllevan técnicas de enumeración más agresivas.

5. Planificación

En cuanto a la planificación, he decidido adoptar un enfoque cíclico para programar la herramienta. De esta forma, en cada iteración nos aseguramos de que no haya errores ni comportamientos no deseados, lo que nos ayuda a evitar que los problemas se acumulen al final del desarrollo, algo que sería típico de un enfoque en cascada [21]. La planificación se ha ido ajustando de manera iterativa a medida que fui avanzando en la programación de la herramienta. Por ejemplo, en la planificación inicial no había considerado la clase *x6GetVhostsPassive*, pero luego la añadí al *diagrama de Gantt* después de darme cuenta de que era necesaria. Por lo que el diagrama presentado en este apartado es el definitivo después de añadirle todas las funcionalidades necesarias. De esta forma, la planificación representada en un *diagrama de Gantt* se ve así:

	Noviembre		Diciembre		Enero		Febrero		Marzo	
Tareas	Primera mitad	Segunda mitad	Primera mitad	Segunda mitad	Primera mitad	Segunda mitad	Primera mitad	Segunda mitad	Primera mitad	Segunda mitad
Formular propuesta TFM										
Idear estrategia para el enfoque de Open Scope										
Programar clase x1GetNets										
Programar clase x2Masscan										
Programar clase x3TlsDomains										
Tests unitarios hasta esta parte										
Programar clase x4Smap										
Tests unitarios hasta esta parte										
Programar clase x5GetSubdomainsPassive										
Estudiar métodos para enumerar posibles virtual hosts sin recurrir a fuerza bruta										
Programar clase x6GetVhostsPassive										
Tests unitarios hasta esta parte										
Programar clase x7CheckWebPorts										
Testear la herramienta en programas de Bug Bounty										
Idear estrategia para el enfoque de Wildcard Domains										
Retocar las clases de Open Scope para integrarlas en el enfoque de Wildcard Domains										
Testear la herramienta en programas de Bug Bounty usando Wildcard Domains										
Programar el frontend										
Tests finales y corrección de errores										
Documentación TFM										

Fig 17. Diagrama de Gantt de la planificación de tareas del proyecto

Desglose de las tareas y entregas previstas:

- Formular propuesta TFM -> Esta será la primera tarea que se llevará a cabo. Consistirá en desarrollar la idea de la herramienta y presentarla a los tutores del TFM para que la validen antes de seguir adelante. Esto se realizará durante la primera mitad de noviembre.
- Idear estrategia para el enfoque de *Open Scope* -> Antes de empezar a programar la herramienta, necesitamos pasar por una fase de diseño donde definiremos las

clases que vamos a programar y cómo funcionará la herramienta. Primero, nos enfocaremos en crear el diseño para los programas de *Open Scope*. Esta tarea deberá ser realizada durante la primera mitad de noviembre.

- Programar clase *x1GetNets* -> Esta funcionalidad de la herramienta será la primera en implementarse. Basándose en el nombre de la empresa, la clase deberá identificar las redes IPv4 que estén asociadas a esa empresa, si es que existen. Esta funcionalidad inicial deberá estar lista para la primera mitad de noviembre.
- Programar clase *x2Masscan* -> Esta función va a ejecutar *Masscan* en el puerto HTTPS para las redes que ya hemos identificado. Deberá estar en funcionamiento durante la primera mitad de noviembre.
- Programar clase *x3TLSDomains* -> Esta función recogerá los certificados TLS de los servidores que se hayan detectado previamente con *Masscan* y guardará los *Common Names* y los *Subject Alternative Names*. Deberá estar disponible para la segunda mitad de noviembre.
- Tests unitarios -> En esta tarea, se llevarán a cabo pruebas unitarias para asegurarnos de que las funcionalidades anteriores están funcionando correctamente y no producen respuestas inesperadas o duplicadas. Es importante confirmar que todo funcione correctamente para finales de noviembre.
- Programar clase *x4Smap* -> Esta función nos ayudará a obtener información de *Shodan* que, por alguna razón, no hayamos recopilado. Así, garantizamos que la herramienta tenga al menos la misma información que *Shodan*. Deberá estar disponible en la primera mitad de diciembre.
- Tests unitarios -> Dado que la clase anterior tiene varias funciones y debe cubrir muchas situaciones diferentes, estas pruebas se enfocarán en cubrir todas las posibilidades, integrando la clase con las funcionalidades que ya están en funcionamiento. Los tests deben estar listos y ser satisfactorios para mediados de diciembre.
- Programar clase *x5GetSubdomainsPassive* -> Esta es una de esas características que no se habían contemplado al principio en la planificación, pero que se añadió durante el proceso de reestructuración iterativa. Esta clase se encargará de recopilar subdominios de forma pasiva, incorporando aquellos que estén activos y respondan a solicitudes DNS, siempre y cuando no hayan sido listados antes. Deberá estar en funcionamiento para la segunda mitad de diciembre.

- Estudiar métodos para enumerar posibles *virtual hosts* sin recurrir a fuerza bruta
-> Ahora que hemos llegado a este punto de la herramienta, se ha considerado la opción de enumerar *virtual hosts* sin tener que recurrir a la fuerza bruta utilizando diccionarios predefinidos. Para conseguirlo, será necesario llevar a cabo un pequeño estudio de métodos que podamos usar, tanto a través de la similitud de dominios como de las relaciones en los certificados. Esta fase de recopilación de información deberá estar finalizada en la segunda mitad de diciembre, unos días antes de que termine el mes.
- Programar clase *x6GetVhostsPassive* -> Esta función debe aplicar todos los métodos que identificamos en la fase anterior. Como vamos a utilizar varias técnicas, se esperará que la implementación de esta funcionalidad esté lista para la primera mitad de enero.
- Tests unitarios -> Dado que la clase anterior tiene muchas funcionalidades, es importante analizarla a fondo. Por eso, las pruebas unitarias se enfocarán en asegurarse de que todas las técnicas se hayan implementado correctamente y que la integración con las funcionalidades que ya están en funcionamiento sea la adecuada.
- Programar clase *x7CheckWebPorts* -> Esta función es la última que deberá ser programada. Su tarea será explorar los datos listados y hacer solicitudes web para recoger las respuestas de los diferentes servidores. Se espera que esté lista para la primera mitad de enero.
- Testear la herramienta en programas de *Bug Bounty* -> Esta tarea es un tipo de prueba para verificar que toda la herramienta funciona correctamente. Para ello, se seleccionan programas de *Bug Bounty* y se comprueba que la salida funcione correctamente, aún sin hacer un excesivo énfasis en los resultados finales. Se deberá confirmar que la herramienta opera adecuadamente para la primera mitad de enero.
- Idear estrategia para el enfoque de *Wildcard Domains* -> Una vez que hayamos verificado que la herramienta funciona para los programas *Open Scope*, será el momento de definir la metodología que utilizaremos para los *Wildcard Domains*. La idea principal es reestructurar las clases que se usaron en los programas *Open Scope*, ajustándolas a este nuevo enfoque. Es importante que la metodología esté bien planificada para la segunda mitad de enero.

- Modificar las clases de *Open Scope* para integrarlas en el enfoque de *Wildcard Domains* -> En esta etapa de la planificación, nos enfocaremos en identificar qué clases ya están programadas y podemos reutilizar para este nuevo enfoque, haciendo las modificaciones necesarias para adaptarlas. Es importante que todas las clases estén listas para la primera mitad de febrero.
- Testear la herramienta en programas de *Bug Bounty* usando *Wildcard Domains* -> La idea principal de esta fase es asegurarnos de que la nueva sección de la herramienta funcione correctamente para los programas de *Bug Bounty* que utilizan *Wildcard Domains*. Esta parte de la herramienta debería estar lista y operativa para la primera mitad de febrero.
- Programar el *frontend* -> Una vez que hayamos configurado ambos enfoques de la herramienta, será necesario crear un *frontend* simple que nos permita ver los datos que hemos recopilado en el *backend*. La interfaz web deberá estar lista para la segunda mitad de febrero.
- Tests finales y corrección de errores -> Esta es la última fase planificada para probar la herramienta. En ella, se verificará que la herramienta funciona correctamente en todos sus modos de ejecución, tanto para programas basados en *Open Scope* como en *Wildcard Domains*. Esta fase debe estar concluida a mediados de marzo, mientras se realiza la documentación final en paralelo.
- Documentación TFM -> Esta es la etapa final del TFM, donde se trata de documentar todo el trabajo que se ha realizado siguiendo las pautas proporcionadas por los tutores. La documentación tiene que estar entregada antes del 31 de marzo.

6. Implementación

6.1. Backend

En esta sección, voy a explicar las funcionalidades que se han implementado en cada una de las clases del *backend* [22]. La herramienta está organizada en dos módulos principales:

1. Enumeración de dispositivos expuestos en Internet en función del nombre de una organización (o proporcionando las redes a analizar).
2. Enumeración de servicios en función de los dominios proporcionados por el usuario.

En primer lugar, explicaré las funcionalidades que se han desarrollado para el primer módulo, el cual es útil en programas de *open scope*.

Clase x1GetNets:

El objetivo principal de esta clase es recopilar las redes públicas asociadas a una organización. Se busca que, a partir del nombre de la empresa proporcionado por el usuario, la herramienta sea capaz de identificar los rangos de direcciones IPv4 [23] pertenecientes a dicha organización, si es que existen.

Para conseguirlo, la clase se apoya en dos fuentes principales de información, bgpview.io y bgp.he.net. Estas páginas permiten consultar datos sobre los prefijos de red asignados a una organización y obtener detalles sobre sus sistemas autónomos (ASNs) [24].

Por ejemplo, la salida para la búsqueda de “booking” en *bgp.he.net* se ve así:












Result	Type	Description
booking	Domain Name	
.booking	TLD	
AS63762	ASN	VietNam Booking corporation 
AS43996	ASN	Booking.com BV 
AS212689	ASN	Booking.com BV 
AS203164	ASN	Booking.com Transport Limited 
AS202196	ASN	Booking.com BV 
91.206.232.0/23	Route	Booking.com BV
91.195.236.0/23	Route	Booking.com BV
5.57.22.0/24	Route	Booking.com GB 
5.57.21.0/24	Route	Booking.com NL 
5.57.20.0/24	Route	Booking.com GB 
5.57.20.0/23	Route	Booking.com BV 
5.57.19.0/24	Route	Booking.com NL 
5.57.18.0/24	Route	Booking.com 

Fig 17. Salida de la búsqueda de “booking” en *bgp.he.net*

A través de *web scraping* [25], la herramienta recogerá todas las filas de la sección "Result" y guardará la información en una variable. Luego, se utilizarán expresiones regulares (regex) [26] para filtrar solo las redes IPv4.

Una vez tengamos las redes IPv4, eliminaremos aquellas que sean subredes de otras más grandes, evitando redundancias innecesarias.

Finalmente, las redes que se enumeren se guardarán en el archivo "files/ips.txt", las cuales serán usadas en las siguientes clases de la herramienta..

Clase x2Masscan:

El propósito de esta clase es identificar los dispositivos activos en las redes que hemos encontrado anteriormente. Para asegurar la máxima velocidad y eficiencia, utilizamos la herramienta *Masscan* [27] para escanear el puerto 443 en todas las direcciones IP de las redes.

Masscan es una herramienta de código abierto que se ha creado para realizar escaneos de puertos TCP de manera masiva y a gran velocidad. Puede alcanzar velocidades de hasta 10 millones de paquetes por segundo, siendo una opción mucho más rápida que otros escáneres como Nmap, aunque hay que tener en cuenta que ofrece menos detalle en los resultados. Es especialmente útil para el reconocimiento inicial y el descubrimiento masivo de puertos abiertos, permitiendo escanear todo el espacio IPv4 de Internet en menos de seis minutos. Además, permite definir rangos específicos de direcciones IP y puertos que se desean analizar.

En este caso, se ha optado por el puerto 443 (HTTPS), porque en la próxima etapa vamos a listar los certificados TLS de los servidores que hayamos detectado anteriormente.

El resultado final de esta clase será un archivo que contendrá todas las direcciones IP activas dentro de los rangos previamente identificados, siempre que tengan el puerto 443 expuesto a Internet.

Clase x3TLSDomains:

Esta clase se dedica a reunir y procesar los certificados TLS [28] de cada dirección IP que se ha identificado en la fase anterior. Para cada IP, se extraen dos elementos fundamentales del certificado:

- Common Name (CN)
- Subject Alternative Names (SANs)

Toda esta información se guarda en un diccionario organizado por IP, lo que garantiza que cada dirección esté vinculada solo a los valores que realmente importan.

Además, se han programado funciones para eliminar duplicados, prestando atención a evitar los duplicados web. Se considera un duplicado web cuando hay dominios como *www.dominio.com* y *dominio.com*, ya que normalmente el segundo redirige al primero, lo que hace que no sea necesario mantener ambos en la estructura de datos.

El resultado de la ejecución de esta clase es un diccionario optimizado, en el que cada IP contiene su *Common Name* (CN) y *Subject Alternative Names* (SANs) enumerados y sin redundancias.

Clase x4Smap:

En la clase x2Masscan, usamos *Masscan* para identificar los dispositivos que tienen el puerto 443 expuesto en las redes de la organización. Sin embargo, este método tiene dos limitaciones principales:

1. Escaneo limitado de puertos: Solo se identifican dispositivos que tienen el puerto 443 abierto, lo que significa que otros servicios que podrían estar expuestos en otros puertos que podrían ser pasados por alto.
2. Posibles falsos negativos: *Masscan* es muy rápido, pero esa velocidad puede hacer que algunos servicios se pasen por alto o que, durante el escaneo, algún servidor no responda.

Para solucionar estos problemas, se ha programado una función que complementa los resultados de *Masscan* con datos obtenidos a través de *Smap* [29].

Smap es una herramienta de escaneo de puertos de código abierto que se apoya en la API gratuita de *Shodan* para recopilar información sobre puertos abiertos y servicios. Se ha diseñado como una alternativa más ligera a *Nmap*, generando salidas que son compatibles con sus formatos. El punto positivo de este enfoque es que utiliza datos ya existentes de *Shodan* sin necesidad de contactar directamente a los objetivos.

De esta forma, ejecutando *Smap* sobre todas las redes enumeradas nos aseguramos de que, como mínimo, el resultado final de la herramienta tenga los mismos datos que *Shodan*.

Para integrar *Smap* en nuestra herramienta sin generar datos redundantes o desactualizados, seguimos una metodología estructurada:

1. Ejecución de *Smap* sobre todas las redes IPv4 previamente identificadas.
2. Procesamiento de los resultados de *Smap*:
 - Para cada IP encontrada, se comprueba si ya existe en nuestra estructura de datos.
 - Si la IP no existe, se añade la información de *Shodan* directamente.
 - Si la IP ya existe, se verifican los *Common Names* (CN) y *Subject Alternative Names* (SANs):
 - Si alguno coincide, se añade cualquier información nueva que no estuviera previamente almacenada.
 - Si no coinciden, la información de *Smap* se considera desactualizada o irrelevante.
3. Eliminación de redundancias:
 - Se eliminan IPs distintas con información idéntica, lo cual es común en entornos con balanceadores de carga.

Con este enfoque, nos aseguramos de que el resultado final de nuestra herramienta siempre incluya al menos la misma información que *Shodan*, evitando así cualquier pérdida de datos que pudiera surgir por falsos negativos de *Masscan*. Además, al filtrar información que ya no está actualizada y eliminar resultados que se repiten, optimizamos el análisis. De esta manera, se van descartando duplicados innecesarios durante la

ejecución de la herramienta y no al final, haciendo que el volumen de datos irrelevantes sea menor, así como el ruido que se genera en la última clase de la herramienta.

Clase x5GetSubdomainsPassive:

Hasta ahora, hemos creado una estructura de datos que contiene información detallada sobre las redes IPv4 de la organización. Sin embargo, todavía puede haber dominios relacionados con la empresa que no estén incluidos en los rangos de IP que hemos identificado previamente.

Para aumentar la superficie de ataque, es importante hacer un listado de subdominios. Estos pueden dirigirnos a servidores que, aunque no estén directamente registrados en redes a nombre de la organización, siguen siendo activos y relevantes para la auditoría.

La estrategia que se usa en esta clase se centra en un enfoque pasivo:

1. Identificación de los *First-Level Domains* más repetidos
 - A partir de la estructura de datos existente, se identifican los cinco *First-Level Domains* (FLD) más frecuentes. Esto nos ayuda a enfocar el análisis en los dominios que son más relevantes para la organización, lo que nos permite optimizar mejor los recursos de enumeración, sin tener que enumerar todos los *FLD* encontrados ya que implicaría una gran pérdida de rendimiento .
2. Recopilación de subdominios
 - Se ejecuta la herramienta *Assetfinder* [31], que permite obtener subdominios de manera pasiva, sin generar tráfico hacia los servidores del objetivo.
 - Se filtran los subdominios obtenidos, eliminando aquellos que ya existen en nuestra estructura de datos para evitar redundancias.
3. Resolución DNS de los subdominios nuevos
 - Se realiza una consulta DNS sobre los subdominios recopilados, separándolos en dos listas:
 - Subdominios que responden a peticiones DNS.

- Subdominios inactivos (sin resolución DNS en el momento del escaneo).
- Para los subdominios que responden correctamente, se analiza la IP devuelta y se realizan las siguientes acciones:
 - Si la IP ya existe en nuestra estructura de datos, el subdominio se añade como *Subject Alternative Name* (SAN) o, en caso de estar vacío, como *Common Name* (CN).
 - Si la IP no existe en nuestra estructura de datos, se crea una nueva entrada para dicha IP, asignándole el subdominio como *Common Name*. También se ejecuta Smap sobre esta IP para obtener datos que *Shodan* pueda tener.

4. Eliminación de redundancias

- Se realiza una limpieza final para eliminar IPs distintas que contienen la misma información, lo que suele ocurrir en entornos con balanceadores de carga o configuraciones multi-IP.

Clase x6GetVhostsPassive:

Esta clase es la que incluye la mayor variedad de técnicas en el proceso de enumeración. En la clase anterior, se encontraron subdominios que no respondían a las solicitudes DNS, pero eso no significa que estén completamente fuera de uso. En algunos casos, estos subdominios pueden estar configurados como *virtual hosts* en un servidor, incluso sin tener una entrada DNS específica. Esto ocurre cuando un mismo servidor web alberga varios sitios bajo una única dirección IP y responde de manera diferente según el *header* “host” de la solicitud HTTP.

El objetivo de esta clase es identificar posibles *virtual hosts* utilizando diversas técnicas y heurísticas. Queremos asegurarnos de que la enumeración sea eficiente y evitar métodos poco efectivos, como los ataques de fuerza bruta con diccionarios grandes. Para conseguirlo, se empieza con un método que consiste en consultar crt.sh, un servicio público que indexa certificados SSL/TLS registrados en los *logs* de *Certificate*

Transparency. La información que se obtiene de estos certificados nos ayuda a descubrir dominios que han sido utilizados en algún momento, incluso si en la actualidad no tienen una resolución DNS activa.

Para obtener esta información, la herramienta hace una solicitud a *crt.sh* usando el dominio principal de la organización como parámetro de búsqueda. Después, se recopilan los *Common Names* (CN) y los *Subject Alternative Names* (SANs) que están relacionados con los certificados encontrados. Con estos datos, se filtran los resultados para eliminar duplicados y descartar aquellos dominios que ya se han identificado en etapas anteriores del proceso de enumeración. Al final, los nuevos subdominios que se obtienen se añaden a la estructura de datos para ser utilizados en las siguientes fases del análisis.

Este proceso facilita la detección de posibles *virtual hosts* sin tener que hacer escaneos activos, lo que evita generar ruido en la infraestructura de destino. Además, al utilizar registros históricos de certificados, se aumentan las posibilidades de descubrir dominios adicionales que podrían seguir en funcionamiento, incluso si su resolución DNS ha cambiado.

Esta página es bastante útil porque nos permite ver qué *Common Names* tienen *Matching Identities*. De esta manera, podemos descubrir subdominios que parecen estar conectados a un subdominio específico. Por ejemplo, si buscamos el subdominio *admin.booking.com*:

21391415	2016-06-08	2016-06-08	2017-06-13	admin.booking.com	admin.booking.com m.admin.booking.com	C=US,O=DigiCert Inc,CN=DigiCert SHA2 Secure Server CA
--------------------------	------------	------------	------------	-------------------	--	---

Fig 18. Ejemplo de la salida de la búsqueda “admin.booking.com” en *crt.sh*

Podríamos pensar que *m.admin.booking.com* está relacionado de cierta forma con *admin.booking.com*. A estos subdominios les llamaremos asociados.

Para seguir con la enumeración de *virtual hosts*, la herramienta cuenta con una función que analiza los subdominios que no responden a las solicitudes DNS y busca encontrar subdominios relacionados. Para lograr esto, se revisa cada uno de estos subdominios y se identifican sus subdominios asociados, creando una lista de listas donde cada grupo incluye subdominios que tienen una alta probabilidad de estar en el mismo servidor. Esto es útil porque, si al menos uno de estos subdominios ya ha sido identificado en nuestra base de datos, podemos añadir todos los subdominios del grupo como posibles *Subject Alternative Names*, ya que es probable que sean *virtual hosts* del mismo servidor.

Sin embargo, esta técnica tiene un inconveniente, y es que si se realizan demasiadas consultas, el *firewall* del sitio web puede interpretar esta actividad como un ataque de denegación de servicio (DoS) y bloquear la dirección IP del analista. Para evitar este problema, se utiliza una segunda técnica que se basa en la herramienta *Certgraph* [32]. *Certgraph* es una herramienta de línea de comandos de código abierto que está diseñada para rastrear las relaciones entre dominios, construyendo un grafo dirigido a partir de los *Subject Alternative Names* (SANs) que se encuentran en los certificados SSL/TLS. En este modelo, cada dominio se representa como un nodo, y los SANs funcionan como aristas que conectan dominios relacionados, lo que permite descubrir asociaciones que de otro modo podrían pasar desapercibidas.

Certgraph recopila esta información al establecer conexiones TCP con los servidores y extraer los datos de sus certificados. Cuando se aplica a los subdominios que ya se han identificado, se obtiene un resultado parecido al de la primera técnica, creando una lista organizada de dominios asociados. Al unir ambas funciones, se incrementa la cantidad de *virtual hosts* detectados sin tener que recurrir a métodos demasiado intrusivos, lo que mejora la efectividad del reconocimiento sin poner en riesgo la discreción del proceso.

Una vez que tenemos los resultados de ambas funcionalidades, los juntamos en una sola lista, eliminando duplicados para no tener redundancias. Luego, revisamos cada grupo de dominios asociados y, si al menos uno de esos dominios ya está en nuestra estructura de datos, añadimos todos los dominios del grupo a la IP correspondiente en la estructura. Los incorporamos como posibles *virtual hosts* en el campo de *Subject Alternative Names* (SANs). Con este proceso, logramos ampliar de manera significativa la identificación de posibles *virtual hosts*, basándonos en las relaciones establecidas a través de certificados SSL/TLS, pese a que el ruido generado se incrementa.

Las funcionalidades explicadas hasta ahora se basan en la relación entre certificados y los dominios que tienen asociados. Sin embargo, hay otros métodos para intentar deducir la posible existencia de *virtual hosts*, como observar la similitud entre los nombres de dominio. Normalmente, los subdominios dentro de una misma infraestructura siguen patrones que pueden sugerir que ciertos dominios están alojados en el mismo servidor. Por ejemplo, si encontramos *admin.gw2.booking.com*, podría ser que

admin.gwl.booking.com también estuviera en el mismo servidor, mientras que *support.chat.booking.com* no lo estuviera.

Para poder agrupar los *virtual hosts* según similitud, se utiliza una técnica de agrupamiento que se basa en la distancia de Jaro [33]. Este algoritmo está diseñado para medir cómo de similares son las cadenas de texto. Para integrarlo en nuestra estructura, se revisa cada lista de dominios asociados y, para cada dominio en la lista, se compara su distancia de Jaro con los dominios de otras listas. Si el coeficiente de similitud supera el 90%, consideraremos que los dominios podrían estar relacionados, y fusionamos sus listas, eliminando los duplicados. Esta técnica permite ampliar la lista de posibles *virtual hosts* sin necesidad de enviar peticiones a la organización, manteniendo así un enfoque pasivo.

La siguiente funcionalidad implementada se basa en visitar merklemaps.com. Esta página permite buscar dominios por distancia. Por ejemplo, para el dominio *secure-iphone-xml.booking.com* veríamos el siguiente *output*:

HOSTNAME	COMMON NAME	FIRST SEEN
secure-iphone-xml.booking.com	secure-iphone-xml.booking.com	23/4/2018
secure-iphone-xml.booking.cn	secure-iphone-xml.booking.cn	11/1/2019

Fig 19. Ejemplo de la salida de la búsqueda “*secure-iphone-xml.booking.com*” en *merklemaps.com*

De esta manera, la función añadirá los nuevos dominios que se hayan encontrado a la lista de dominios asociados y, después, eliminará cualquier duplicado para evitar redundancias.

Una vez que tenemos la nueva lista de dominios asociados, integramos los resultados en nuestra estructura de datos de la misma manera que lo hicimos para la detección de *virtual hosts* basado en relaciones de certificados. Para ello, primero verificamos si alguno de los dominios de la lista ya existe en nuestra estructura. Si es así, añadimos todos los dominios de la lista a los *Subject Alternative Names* (SAN), asegurándonos de que no haya duplicados.

Finalmente, se revisa cada lista de dominios asociados de nuevo y, si alguno ya está en la estructura de datos, se añaden todos los dominios relacionados. Esta función es la misma que se aplicó en la primera fase de la clase. Luego, se aplica la misma lógica, pero agrupando los dominios según su similitud. En este caso, si un dominio de la lista es

parecido a alguno de la lista enumerada, se incorporan todos los dominios asociados a él en los *Subject Alternative Names*.

Clase x7CheckWebPorts:

La clase x7CheckWebPorts es la última clase que se ha programado. Hasta ahora, la estructura de datos creada para cada IP enumerada incluye el *Common Name*, los *Subject Alternative Names* y los puertos abiertos. El objetivo es hacer solicitudes web para recoger información del servidor, como el código de estado, el título de la página, la URL de redirección, el puerto, el contenido de la respuesta y los *headers* que se devuelven en la respuesta.

Para realizar este proceso, se han definido tres enfoques distintos:

1. Petición directa a la IP: Se realiza una solicitud GET a la dirección IP en cada uno de los puertos identificados.
2. Petición a los dominios: Mismo procedimiento, pero en lugar de apuntar a la IP, las peticiones se hacen a los dominios asociados a esa dirección.
3. Petición con *Virtual Hosts*: En este caso se hacen múltiples solicitudes a la IP, enviando en cada petición la cabecera HTTP "Host" con el valor de cada *Subject Alternative Name*. De esta manera, se intenta identificar posibles *virtual hosts* alojados en el mismo servidor. Esto es posible puesto que en las fases anteriores se han enumerado varios dominios que podrían ser *virtual hosts*, y los hemos almacenado en la lista de *Subject Alternative Names*.

Finalmente, para evitar redundancias, se eliminan las respuestas duplicadas. Se considera que dos respuestas son duplicadas si coinciden en su código de estado, título de la página y contenido de la respuesta [34]. También se considera que dos respuestas son duplicadas si coincide la *request*, es decir, mismo protocolo, dominio y puerto, evitando así contenido duplicado alojado en distintos servidores, típico de balanceadores de carga.

Con la implementación de estas clases, hemos terminado la primera parte de la herramienta, que se enfoca en programas de *open scope*. Para la segunda parte, que se basa en dominios específicos, reutilizaremos gran parte del código que ya hemos

desarrollado, ya que la estructura de datos que manejamos es la misma en ambos casos. Sin embargo, hay diferencias importantes en el orden de ejecución de las clases.

En este enfoque, la primera clase que se ejecuta es *GetSubdomainsPassive*, ya que el punto de partida ha cambiado de redes IPv4 a dominios que el analista deberá proporcionar. Por lo tanto, el primer paso es realizar una enumeración pasiva de los subdominios que están asociados a estos dominios. La gran diferencia con la metodología de la clase que se utiliza en *open scope* es que aquí se utilizan dos herramientas en vez de una para obtener un mayor número de subdominios recopilados. Estas herramientas son *Assetfinder* y *Subfinder*. Además, esta clase incluye una función específica para crear desde cero la estructura de datos, ya que, a diferencia del enfoque basado en *open scope*, esta estructura no proviene de clases anteriores.

Una vez que tenemos los subdominios, pasamos a recopilar los certificados TLS de aquellos que están activos. En este punto, reutilizamos la misma clase que implementamos antes para el *open scope*.

En cuanto a la integración de *Smap*, para asegurarnos de que funcione igual que en el caso de los *open scope*, se ha añadido una funcionalidad que permite volcar las direcciones IP obtenidas en la estructura de datos dentro del archivo donde antes se guardaban las redes IPv4 en el contexto de *open scope*. Debido a que *Smap* maneja de la misma manera una red y una IP individual, el resto de la clase no necesita modificaciones.

Finalmente, las clases *GetVhostsPassive* y *CheckWebPorts* se mantienen sin cambios respecto a la versión utilizada en *open scope*, ya que su lógica de funcionamiento sigue siendo aplicable para este enfoque.

Con esta explicación, llegamos al final de la sección sobre la implementación del *backend*. He optado por no profundizar en los detalles del código y, en su lugar, explicar las técnicas utilizadas desde un enfoque teórico, para que se entienda mejor el propósito de cada clase. En la próxima sección, presentaré los diferentes modos de ejecución que ofrece la herramienta, las ventajas de cada uno y los resultados que generan.

6.2. Modos de ejecución

Open Scope

Modo de ejecución corto

Este modo de ejecución es bastante sencillo, ya que se enfoca en analizar las redes IPv4 sin hacer una búsqueda adicional de subdominios o *virtual hosts*, más allá de los que se identifican en las etapas ejecutadas. En primer lugar, se identifican las redes IPv4 de la organización, a menos que el usuario decida proporcionar las redes a escanear. Luego, se utiliza *Masscan* en estas redes para encontrar dispositivos con puertos abiertos y, a continuación, se recopilan los certificados TLS de los servicios web enumerados. Después, se ejecuta *Smap* para obtener información actualizada de *Shodan*, asegurando que los datos sobre puertos y servicios abiertos no estén desactualizados. Por último, se hacen solicitudes web a los servidores detectados para recopilar información adicional sobre su estado y configuración.

Este método de ejecución tiene varias ventajas. En primer lugar, es el más rápido de los tres, ya que no necesita el uso de técnicas avanzadas para descubrir *virtual hosts* o subdominios. Además, genera menor ruido en comparación con los otros enfoques, reduciendo el riesgo de ser detectado por sistemas de defensa. También, al centrarse únicamente en información directa de redes IPv4 y certificados, se minimiza la posibilidad de generar falsos positivos. Sin embargo, el análisis se limita únicamente a las redes IPv4 que se identificaron en la fase inicial, sin la opción de ampliar la superficie de ataque al identificar dominios adicionales. Además, la detección de *virtual hosts* será menor. Aunque es posible encontrar algunos *virtual hosts* a partir de la información extraída de los certificados TLS y los datos de *Smap*, estos casos serán excepcionales y no representarán la norma en este modo de ejecución.

Diagrama de flujo:

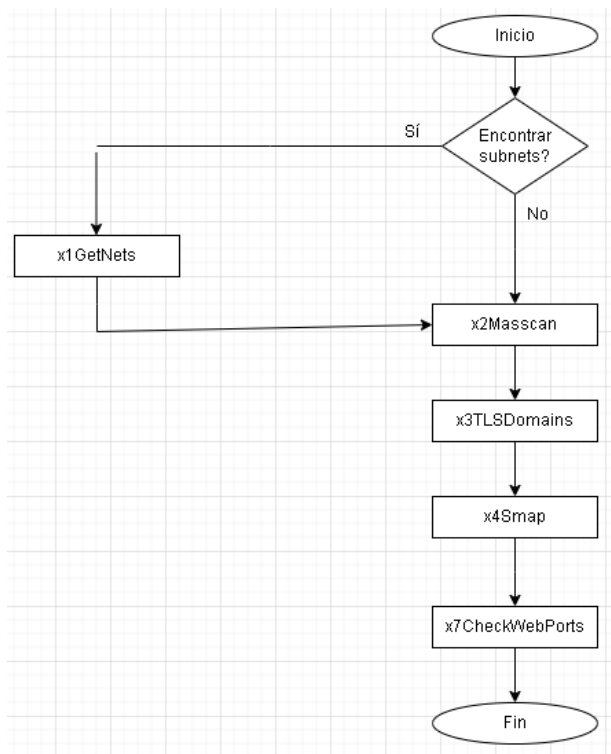


Fig 20. Diagrama de flujo para el modo de ejecución corto del enfoque para programas de Open Scope

Modo de ejecución medio

Este modo de ejecución se parece al corto, pero hay una diferencia importante ya que en este caso, se amplía la superficie de ataque al enumerar subdominios. Esto permite descubrir servidores adicionales mediante la resolución DNS de estos subdominios. Para los nuevos subdominios activos, si no se cuenta con información previa en la estructura de datos, se ejecuta *Smap* sobre ellos para obtener los datos que tenga *Shodan*. A diferencia del modo largo, este método no incluye la recopilación de posibles *virtual hosts*. Aún así, es posible que se detecten algunos *virtual hosts* a partir de los dominios que se identificaron en las etapas anteriores.

En comparación con el modo de ejecución corto, tiene la ventaja de poder descubrir un mayor número de servidores que tal vez no estén directamente alojados en las redes de la organización. También permite identificar más *virtual hosts*, aunque no es su objetivo principal. En cuanto a desventajas frente al modo corto, tenemos su tiempo de ejecución, el cual es más largo debido a la búsqueda pasiva de subdominios y a las resoluciones DNS que se generan. Además, produce un mayor nivel de ruido y aumenta la posibilidad de

falsos positivos, debido a que se hacen más peticiones tanto a dominios como a posibles *virtual hosts*.

Este modo de ejecución se sitúa en un punto intermedio entre los modos corto y largo, debido a que es el menos extremista por sus ventajas y desventajas.

Diagrama de flujo:

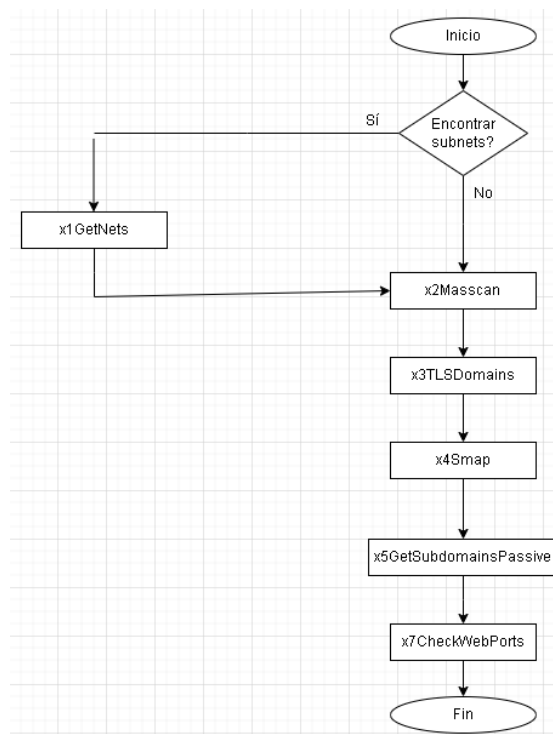


Fig 21. Diagrama de flujo para el modo de ejecución medio del enfoque para programas de Open Scope

Modo de ejecución largo

Finalmente, el modo de ejecución largo es el que se encarga de ejecutar todas las clases explicadas en la sección de implementación. A diferencia del modo de ejecución medio, este modo incluye la habilidad de detectar *virtual hosts* utilizando las técnicas que se han explicado antes.

Al incluir más pasos y técnicas, es el que más tiempo tarda en ejecutarse de media. Además, genera más ruido y aumenta la posibilidad de obtener falsos positivos. Su mayor ventaja es que produce más resultados, lo que permite descubrir más dominios y *virtual hosts*. Por lo tanto, es la opción más completa, capaz de ofrecer información más detallada

y valiosa, aunque pueda tener un rendimiento algo menor en cuanto a tiempo de ejecución.

Diagrama de flujo:

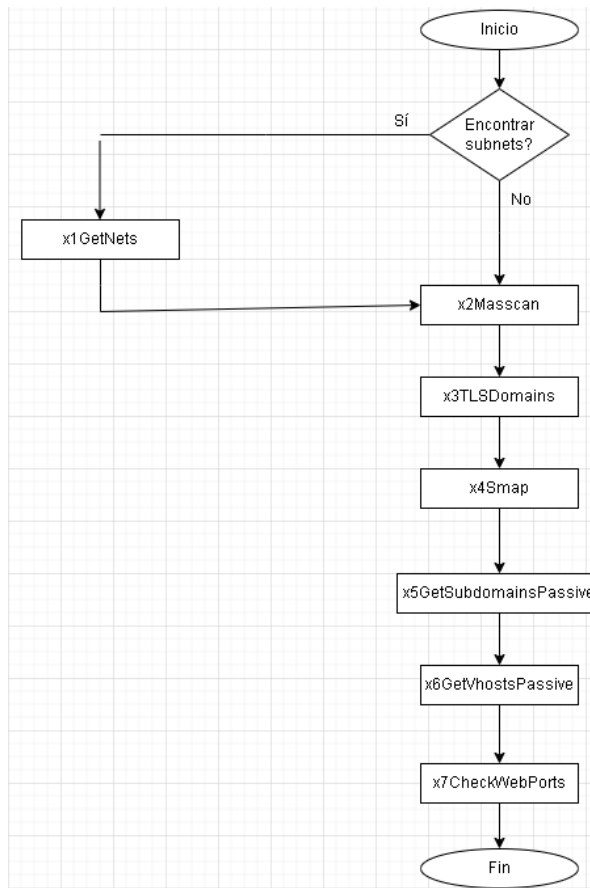


Fig 22. Diagrama de flujo para el modo de ejecución largo del enfoque para programas de Open Scope

Wildcard domains

Modo de ejecución corto

En este modo de ejecución, no se recopilan *virtual hosts* de manera pasiva, aunque se pueden enumerar algunos de forma indirecta, sin ser este su objetivo principal.

En primer lugar se leen los dominios que el usuario ha proporcionado en el archivo “files/subdomains.txt”. Luego, se enumeran subdominios sobre estos dominios, asegurando que se resuelvan mediante consultas DNS. Después, se extrae información relevante de los dominios que se han identificado y se recopilan datos pasivos de *Shodan*

utilizando *Smap*. Finalmente, se hacen peticiones web a cada servidor identificado para obtener información sobre ellos.

Este modo se centra en la velocidad y en minimizar los falsos positivos al no enumerar los *virtual hosts*. Sin embargo, esta limitación puede resultar en la pérdida de información valiosa que podría ayudar a ampliar la superficie de ataque.

Diagrama de flujo:

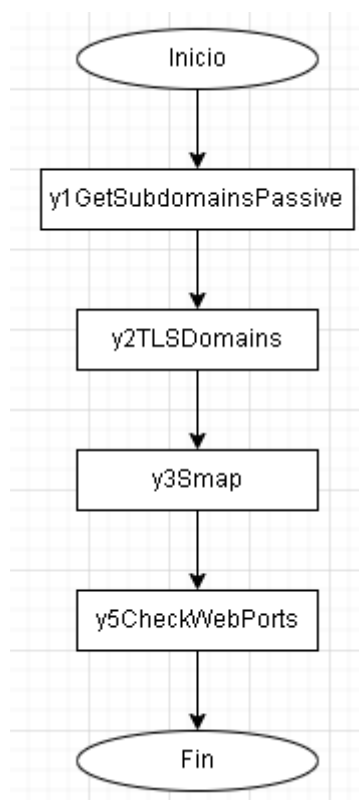


Fig 23. Diagrama de flujo para el modo de ejecución corto del enfoque para programas de Wildcard Domains

Modo de ejecución largo

A diferencia del modo de ejecución corto, este modo se centra en enumerar los *virtual hosts* de la manera más discreta posible. Para esto, se utilizan las mismas funciones que en el modo corto, pero antes de recopilar información de los servidores web, se identifican los posibles *virtual hosts* siguiendo el mismo proceso que se usa en el modo largo del enfoque basado en *open scope*.

Este modo incrementa la posibilidad de identificar *virtual hosts*, pudiendo encontrar información valiosa para el auditor. Sin embargo, este hecho también implica una menor velocidad de ejecución y un mayor riesgo de obtener falsos positivos.

Diagrama de flujo:

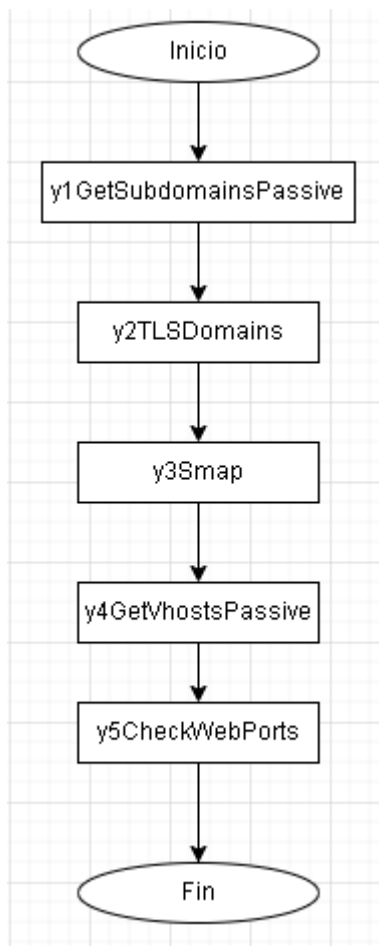


Fig 24. Diagrama de flujo para el modo de ejecución largo del enfoque para programas de Wildcard Domains

Resultados de los distintos modos de ejecución

La idea original al acabar la implementación de la herramienta era probar los diferentes modos de ejecución y comparar los resultados que se obtenían. Sin embargo, a medida que se han llevado a cabo las pruebas, me he dado cuenta de que esta comparación no es viable, ya que el contexto varía en cada ejecución. No habrá dos ejecuciones que se hagan en el mismo contexto, lo que hace que no sean comparables entre sí. Este hecho se debe a dos razones principales:

- Infraestructura dinámica de las organizaciones -> Las organizaciones están en constante cambio con sus activos digitales, lo que significa que los datos disponibles pueden variar de una ejecución a otra. Aunque gran parte de la infraestructura se mantiene estable, a veces puede suceder que un servidor no responda en un momento determinado o que alguno se haya apagado entre las ejecuciones.
- Dependencia de servicios externos -> La herramienta se basa en varios servicios y herramientas externas. Por ejemplo, servicios como *crt.sh* o *Shodan* actualizan sus bases de datos de manera impredecible y pueden bloquear la IP del analista después de varias ejecuciones, lo que puede afectar a los resultados de la herramienta. Además, herramientas como *assetfinder* también dependen de otros servicios externos, lo que significa que enfrentan el mismo problema de no tener un comportamiento determinista, lo que a su vez impacta en nuestra herramienta.

Es por estas razones que cada ejecución será única en su contexto, y los datos que se recojan pueden variar (aunque serán bastante parecidos entre diferentes ejecuciones). Además, la cantidad de datos que se recopilen dependerá del tamaño de la infraestructura de la organización, la potencia de cómputo de la máquina del analista, la velocidad de la red en la que se esté trabajando...

A pesar de todo, se han llevado a cabo pruebas para el programa de *Booking*. En el primer caso, se ha considerado el programa como un *Open Scope* de la empresa, lo que significa que se van a analizar las redes IPv4 de *Booking*. Se ha ejecutado cada modo diez veces y los resultados promedio de esta fase han sido los siguientes:

Open Scope	Modo corto	Modo medio	Modo largo
Tiempo de ejecución	48 minutos	122 minutos	161 min
IPs encontradas	37	120	121
Hosts encontrados	50	321	376
Virtual hosts encontrados	2	35	38

Como se ha comentado anteriormente, estos resultados no son demasiado significativos, puesto que variarán dependiendo del entorno del analista, la empresa objetivo, el equipo en el que se ejecute el programa... Pero sí nos pueden ser útiles para ver que los objetivos de cada modo de ejecución se cumplen de media. Es decir, el modo corto es el que menos

información da, pero el que más rápido se ejecuta. Por el contrario, el modo largo es el que más información da pero es el que más lento se ejecuta, mientras que el modo medio es el menos extremista de ambos, combinando información y velocidad de ejecución.

Para las pruebas respectivas a los Wildcard Domain, se han seleccionado estos 4 dominios que se encuentran en el scope del programa:

***.fareharbor.engineering**

***.rentalcars.com**

***.fareharbor.com**

***.booking.com**

Los resultados promedio tras cinco ejecuciones han sido los siguientes:

Wildcard Domains (4 dominios)	Modo corto	Modo largo
Tiempo de ejecución	9 minutos	25 minutos
IPs encontradas	66	60
Hosts encontrados	200	219
Virtual hosts encontrados	11	19

Como se puede ver, el resultado es justo lo que esperábamos. El modo corto se ejecuta mucho más rápido porque no recopila virtual hosts, lo que evita esa fase pasiva. Además, al recopilar información de los servidores web, hace muchas menos peticiones, ya que su lista de SAN es mucho más corta. En cuanto a las IPs encontradas, es normal que el resultado varíe un poco debido a los motivos explicados anteriormente. En cuanto a *hosts* encontrados y *virtual hosts*, es lógico que el modo largo permita enumerar más cantidad, puesto que se hace énfasis en esta parte, creando listas más grandes de *Subject Alternative Names*, pese a que este hecho pueda causar que haya más falsos positivos.

Vamos a ver cómo se ven las ejecuciones del *backend* en la terminal.

Ejemplo de ejecución por nombre:

```
└─# python server.py
¿Qué script deseas ejecutar?
1. serverApexdomains.py
2. serverSubdomains.py
Selecciona una opción (1/2): 1
Ejecutando serverApexdomains.py...
Elija método de ejecución (1: corto, 2: medio, 3: largo): 2
Encontrar subnets (0) / Usar redes definidas en files/ips.txt (1): 0
Nombre de la empresa: booking
Buscando nets...
Subnets encontradas!
Ejecutando masscan...
Recopilando certificados TLS...
Ejecutando Smap...
Recopilando subdominios de forma pasiva con assetfinder...
Resolviendo el DNS de los subdominios recopilados...
Recopilando información de los puertos web...
Borrando la base de datos anterior...
Datos guardados en la base de datos y archivo JSON.
Tiempo de ejecución: 3069.41 segundos
serverApexdomains.py se ejecutó correctamente.
```

Fig 25. Ejemplo de ejecución del backend por nombre

Ejemplo de ejecución por dominios:

```
└─# cat files/subdomains.txt
fareharbor.engineering
rentalcars.com
fareharbor.com
booking.com
```

Fig 26. Ejemplo archivo files/subdomains.txt

```
└─# python server.py
¿Qué script deseas ejecutar?
1. serverApexdomains.py
2. serverSubdomains.py
Selecciona una opción (1/2): 2
Ejecutando serverSubdomains.py...
Elija método de ejecución (1: corto, 2: largo): 1
Recopilando subdominios de forma pasiva...
Resolviendo el DNS de los subdominios recopilados...
Recopilando certificados TLS...
Ejecutando Smap...
Recopilando información de los puertos web...
Borrando la base de datos anterior...
Datos guardados en la base de datos y archivo JSON.
Tiempo de ejecución: 577.78 segundos
serverSubdomains.py se ejecutó correctamente.
```

Fig 27. Ejemplo de ejecución del backend con enfoque Wildcard Domains

6.3. Frontend

Para mostrar los resultados, se ha decidido crear un frontend en Python usando el framework Flask, el cual nos permitirá facilitar tanto la visualización como la búsqueda de la información que hemos recopilado durante el reconocimiento de servidores web. La aplicación se conecta a una base de datos SQLite, desde donde extraemos datos organizados por IP, que luego se reestructuran de forma visual. Para esto, se ha establecido un orden específico de campos, título, código de estado, solicitud, URL de redirección, puerto, contenido de respuesta y cabeceras. Este orden se aplica a todos los registros para que la información se presente de manera uniforme.

En cuanto a la interfaz, se han definido algunas rutas principales. Una ruta principal que se encarga de renderizar una plantilla HTML con todos los resultados, una ruta de búsqueda que permite a los analistas filtrar los datos mediante sus consultas y devuelve los resultados en formato JSON, y una ruta adicional que muestra todos los registros, también en formato JSON. Además, se han implementado algunas funciones para reorganizar los campos y realizar búsquedas *case sensitive*.

Hablando de las funciones de la interfaz, lo primero a destacar es una barra de búsqueda que cuenta con dos botones: "Search" y "All". Esta barra le permite al analista escribir texto para buscar coincidencias en las respuestas. Por ejemplo, si escribes el término "admin", la interfaz te mostrará todas las respuestas que incluyan esa palabra en cualquiera de los campos recopilados. Se puede ver su funcionamiento en el siguiente ejemplo.

Query en la interfaz:

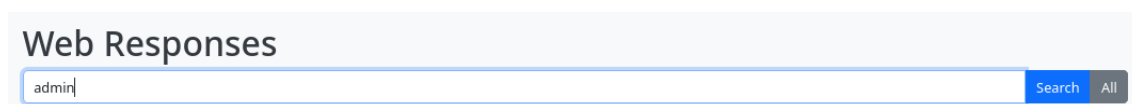


Fig 28. Barra de búsqueda en la interfaz web de la herramienta

Query interna:

```
127.0.0.1 - - [10/Mar/2025 16:46:40] "GET /search?q=admin HTTP/1.1" 200 -
```

Fig 29. Query interna de la búsqueda de una cadena en la interfaz web

Resultado:

5.57.16.230	
title:	Booking.com
status_code:	200
request:	https://admin.bookings.org:443
redirected_url:	https://account.booking.com/sign-in?op_token=EgWvYXV0aCYAqUln03Mm9IT2QzNk5uN3prM3BpcmgSCWF1dGhvcml6ZRoaaHR0cHM6Ly9hZG1pb5ib29raW5nLmNvbS8qOnsiYXV0aF9hdHRlbiBBOXB2kljoiZDYtZTE4NGEtZmRjIn00MTc3LTlkMjAtZTliNDJmZGM3MWlyIn0yOzNs2bFRMGfHrINFX0fK53ZO6b0xftUNXVUs0eU5WEIzGc3IakZQdyQ2Zs6BFMyNTZCBGNvZGUGeZCj5eik_swnOgBCAFjn35n91zl
port:	443
response_text:	Please enable JavaScript in your browser to proceed
response_headers:	{ "Content-Type": "text/html; charset=UTF-8", "Transfer-Encoding": "chunked", "Connection": "keep-alive", "Server": "envoy", "Date": "Mon, 10 Mar 2025 10:42:49 GMT", "content-security-policy": "base-uri 'none'; frame-ancestors https://*.booking.com https://*.booking.cn; object-src 'none'; report-uri https://nelliie.booking.com/csp-report-uri?type=block&e=UmFuZG9tSVYkc2Rllyh9YSWktKO5TsgOptwVTPfHZKNW3Hcrm1RLgc98bqahgr47O5fWUs4jdc2RbF6WTeHzI9a_GGqjDsm2HufnscKwNeU; script-src 'report-sample' 'nonce-IXzpULXuTKrgFlw' 'strict-dynamic' 'unsafe-eval' 'unsafe-hashes' 'sha256-

Fig 30. Salida de la herramienta de la búsqueda por cadena – parte 1

5.57.16.92	
title:	
status_code:	403
request:	https://j1.booking.com:443
redirected_url:	
port:	443
response_text:	<html><body><h1>403 Forbidden</h1> Request forbidden by administrative rules. </body></html>
response_headers:	{ "Content-Type": "text/html", "Content-Length": "93", "Connection": "keep-alive", "Date": "Mon, 10 Mar 2025 10:42:50 GMT", "Cache-Control": "no-cache", "X-Cache": "Error from cloudfront", "Via": "1.1 eabf0052502240e2b09c2e962490cab.cloudfront.net (CloudFront)", "X-Amz-Cf-Pop": "HAM50-P1", "X-Amz-Cf-Id": "Xwh14P2lLoZNMqGb12X8zYKjNWXOMCZdxEjCBkwlqWnfSwmPOv_6PQ==", "Strict-Transport-Security": "max-age=63072000; includeSubDomains; preload" }

Fig 31. Salida de la herramienta de la búsqueda por cadena – parte 2

37.10.30.138	
title:	Booking.com - Sign In
status_code:	200
request:	https://workforce-dev.voicedqs.booking.com:443
redirected_url:	https://okta.booking.com/app/bookingcom_teleoptiwfmlab_1/exk5n1ktaLt7MUwfj416/sso/saml?RelayState=http://workforce-dev.booking.com/TeleoptiWFM/
port:	443
response_text:	Connecting to Sign in with your account to access TeleOpti WFM - Lab Javascript is required Javascript is disabled on your browser. Please enable Javascript and refresh this page. Refresh Your OneDrive version is not supported Upgrade now by installing the OneDrive for Business Next Generation Sync Client to login to Okta Learn how to upgrade Cookies are required Cookies are disabled on your browser. Please enable Cookies and refresh this page. Refresh Privacy Policy The page has timed out If this page does not reload automatically, please refresh your browser.
response_headers:	{ "Date": "Mon, 10 Mar 2025 10:42:47 GMT", "Server": "nginx", "Content-Type": "text/html; charset=utf-8", "Vary": "Accept-Encoding", "x-okta-request-id": "Z87CJw6nm99oyxZZV0adAAAA58", "x-xss-protection": "0", "p3p": "CP=\"I\"HONK\"", "content-security-policy": "default-src 'self' booking.okta.com okta.booking.com *.oktacdn.com; connect-src 'self' booking.okta.com booking-admin.okta.com okta.booking.com *.oktacdn.com *.mixpanel.com *.mapbox.com *.mtls.okta.com booking.kerberos.okta.com *.authenticatorlocalprod.com:8769 http://localhost:8769 http://127.0.0.1:8769 *.authenticatorlocalprod.com:65111 http://localhost:65111 http://127.0.0.1:65111

Fig 32. Salida de la herramienta de la búsqueda por cadena – parte 3

Los tres casos que se han presentado son ejemplos de diferentes áreas donde se pueden encontrar coincidencias. Por falta de tiempo, no he podido implementar operadores que ayuden a filtrar la búsqueda en campos específicos, a diferencia de lo que ofrece *Shodan*.

Si se selecciona el botón “All”, se muestran todas las respuestas, tal y como se visualizan al acceder inicialmente a la interfaz web.

Query en el backend:

```
127.0.0.1 - - [10/Mar/2025 16:50:31] "GET /all HTTP/1.1" 200 -
```

Fig 33. Query interna cuando se selecciona el botón "All" en la interfaz web

De esta forma, se pueden ver todos los resultados desplazándose hacia abajo o filtrarlos mediante palabras clave de interés, modos de uso que se explicarán más tarde en el estudio.

6.4. Instalación de la herramienta

Para poder usar la herramienta se ha añadido un apartado de “installation” como se puede ver en el [repositorio del trabajo](#):

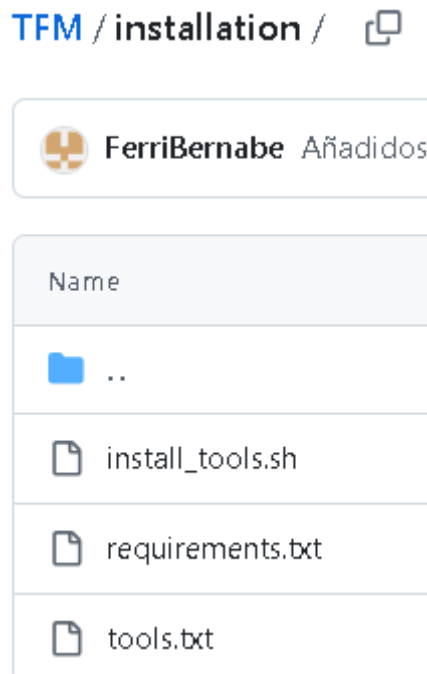


Fig 34. Carpeta “installation” del repositorio de la herramienta

Para poder instalar y ejecutar la herramienta se deben seguir los siguientes pasos:

1. Clonar el [repositorio](#):

```
git clone https://github.com/FerriBernabe/TFM.git
```

Fig 35. Comando para clonar el repositorio

2. Acceder a la carpeta “installation”:

```
osboxes@osboxes:~/TFM$ cd installation/  
osboxes@osboxes:~/TFM/installation$ ls -la  
total 20  
drwxrwxr-x 2 osboxes osboxes 4096 Mar 24 07:00 .  
drwxrwxr-x 8 osboxes osboxes 4096 Mar 24 07:00 ..  
-rw-rw-r-- 1 osboxes osboxes 1787 Mar 24 07:00 install_tools.sh  
-rw-rw-r-- 1 osboxes osboxes 60 Mar 24 07:00 requirements.txt  
-rw-rw-r-- 1 osboxes osboxes 45 Mar 24 07:00 tools.txt
```

Fig 36. Contenido de la carpeta “installation”

3. Instalar los “requirements”:

```
pip3 install -r requirements.txt
```

Fig 37. Comando para instalar los requerimientos

4. Ejecutar el *script* “install_tools.sh”:

```
chmod +x install_tools.sh
./install_tools.sh
```

```
¿Quieres instalar la última versión de Go? (s/n): s
```

Fig 38. Comandos para ejecutar el script de instalación

5. Si se desea ejecutar la herramienta por alguna interfaz que no sea la “tun0” se deberá cambiar en el código. Por ejemplo, si queremos que la herramienta se ejecute por la interfaz “enp0s3”:

```
osboxes@osboxes:~/TFM$ nano classes/apexdomains/x2Masscan.py
osboxes@osboxes:~/TFM$ head -n7 classes/apexdomains/x2Masscan.py
import os
import subprocess

class ExecMasscan:
    #1. Constructor __init__ y getters
    #Le pasamos los dos archivos por defecto que tendrá
    def __init__(self, masscan_results_file="files/masscanResults.txt", ips_file="files/ips.txt", masscan_rate=8000, interfaz_de_red
="enp0s3"):
```

Fig 39. Constructor de la clase x2Masscan para ejecutar la herramienta en la interfaz “enp0s3”

A partir de este punto ya se puede ejecutar la herramienta. Se recomienda leer el apartado 6.5 para entender cómo se ejecuta cada uno de los modos explicados en apartados anteriores. Por ejemplo, para ejecutar la herramienta con enfoque basado en *Wildcard Domains*:

```
osboxes@osboxes:~/TFM$ nano files/subdomains.txt
osboxes@osboxes:~/TFM$ cat files/subdomains.txt
fareharbor.com
```

Fig 40. Ejemplo del archivo “files/subdomains.txt” para enumerar servicios web relacionados con un dominio en particular

```
osboxes@osboxes:~/TFM$ python3 server.py
¿Qué script deseas ejecutar?
1. serverApexdomains.py
2. serverSubdomains.py
Selecciona una opción (1/2): 2
Ejecutando serverSubdomains.py...
Elija método de ejecución (1: corto, 2: largo): 1
Recopilando subdominios de forma pasiva...
Resolviendo el DNS de los subdominios recopilados...
Recopilando certificados TLS...
Ejecutando Smap...
Recopilando información de los puertos web...
Borrando la base de datos anterior...
Datos guardados en la base de datos y archivo JSON.
Tiempo de ejecución: 102.28 segundos
serverSubdomains.py se ejecutó correctamente.
```

Fig 41. Ejemplo de la ejecución del backend de la herramienta con enfoque basado en *Wildcard Domains*

Finalmente, ejecutamos el frontend para ver los resultados en una interfaz web (se recomienda leer el apartado 6.6 para entender cómo usarlo):


```

osboxes@osboxes:~/TFM/frontend$ python webserver.py
Command 'python' not found, did you mean:
  command 'python3' from deb python3
  command 'python' from deb python-is-python3
osboxes@osboxes:~/TFM/frontend$ python3 webserver.py
* Serving Flask app 'webserver'
* Debug mode: on
WARNING: This is a development server. Do not use it
* Running on http://127.0.0.1:5000

```

Fig 42. Ejemplo de la ejecución del frontend de la herramienta

Web Responses	
127.0.0.1:5000	
52.9.57.221	
title:	FareHarbor Integration Center
status_code:	200
request:	http://developer.fareharbor.com:80
redirected_url:	https://developer.fareharbor.com/api/external/v1/
port:	80
response_text:	FareHarbor Integration Center Home API Reference Webhook Webhook Best Practices Guides & FAQs F Feedback Support Change Log
response_headers:	{ "Date": "Mon, 24 Mar 2025 11:16:08 GMT", "Con "chunked", "Connection": "keep-alive", "Content-E "CP="This is not a P3P policy.", "Set-Cookie": "fh SameSite=Lax", "Strict-Transport-Security": "max- "Root=1-67e13ef8-01087dc416b2f5285f269c06", "production", "X-Frame-Options": "SAMEORIGIN", "
52.9.57.221	
title:	FareHarbor
status_code:	429
request:	https://fhbr.co:443

Fig 43. Salida de la interfaz web de la herramienta

En el siguiente apartado se explicará como ejecutar cada modo comentado en los apartados anteriores.

6.5. Ejecución del backend

En esta sección y en la siguiente, se enseñará a ejecutar la herramienta de distintas formas y a cómo utilizar el *frontend* para conseguir los resultados que busque el analista. En este apartado, se explicará cómo ejecutar el *backend* para recopilar información sobre la organización objetivo.

Open Scope – Redes descubiertas automáticamente

Esta es la primera manera de usar la herramienta. En este caso, el analista se enfoca en proporcionar el nombre de la organización, y la herramienta se encargará de encontrar automáticamente las redes asociadas a esa organización.

Para evitar falsos positivos, se aconseja visitar primero las páginas de *bgp.he.net* y *bgpview.io*. Así se pueden ver qué resultados ofrecen estas páginas y ajustar el nombre de la organización que se le proporciona a la herramienta.

Por ejemplo, si se quisiera encontrar las rutas para enumerar la organización “Booking.com BV”. Si se filtra por “booking” en *bgp.he.net*, se verá que aparecen varias rutas, algunas pertenecientes a “Booking.com BV” y otras no:

booking

Search

5.57.16.0/24	Route	Booking.com GB
5.57.16.0/22	Route	Booking.com
5.57.16.0/21	Route	Booking.com BV
37.10.60.0/24	Route	Booking.com
37.10.60.0/23	Route	Booking.com BV
37.10.56.0/23	Route	Booking.com BV
37.10.31.0/24	Route	Booking.com BV
37.10.30.0/24	Route	Booking.com BV
37.10.30.0/23	Route	Booking.com BV
37.10.29.0/24	Route	Booking.com NL
37.10.28.0/24	Route	Booking.com
37.10.27.0/24	Route	Booking.com
37.10.26.0/24	Route	Booking.com

Fig 44. Resultado de la búsqueda “booking” en *bgp.he.net*

El analista entonces deberá buscar la manera de evitar falsos positivos y filtrar solo las rutas pertenecientes a la organización objetivo. En este caso, el analista podría filtrar por “Booking.com BV”:

"Booking.com BV"	Search
------------------	--------

37.10.60.0/23	Route	Booking.com BV
37.10.56.0/23	Route	Booking.com BV
37.10.31.0/24	Route	Booking.com BV
37.10.30.0/24	Route	Booking.com BV
37.10.30.0/23	Route	Booking.com BV

Fig 45. Resultado de la búsqueda “booking.com BV” en bgp.he.net

Para *bgpview.io*, el proceso es bastante similar. Es importante mencionar que filtrando los resultados de cualquiera de las dos páginas es suficiente, y la herramienta podrá identificar las redes de la empresa. Si el analista tiene dificultades para filtrar las rutas de manera adecuada y no quiere tener falsos positivos, puede utilizar la herramienta en el modo “Open Scope – Redes proporcionadas”, que se explicará en el siguiente subapartado.

Una vez descubierta la cadena a proporcionarle a la herramienta para enumerar las redes de la organización, hay un detalle importante a tener en cuenta antes de ejecutar la herramienta. A la hora de ejecutar el modo enfocado a *Open Scope*, se ejecuta la herramienta *Masscan* con una cadencia bastante elevada. Esto puede traer algunos problemas de restricciones de *rate limits*, por lo que se da la opción al analista de cambiar los parámetros de *masscan_rate* e *interfaz_de_red*. Por defecto, la interfaz de red tiene el valor “tun0”, pero se puede cambiar en el constructor de la clase. Para ello, el analista debe abrir el archivo “classes/apexdomains/x2Masscan.py”. En este archivo se encontrará un constructor con ambos parámetros modificables:

```
class ExecMasscan:
    #1. Constructor __init__ y getters
    #Le pasamos los dos archivos por defecto que tendrá
    def __init__(self, masscan_results_file="files/masscanResults.txt", ips_file="files/ips.txt", masscan_rate=8000, interfaz_de_red="tun0"):
```

Fig 46. Código del constructor e la clase ExecMasscan

Se deberá cambiar la interfaz de red por la que el analista prefiera usar. Por lo general, se recomienda usar un servicio de VPN si se va a mantener una cadencia alta de peticiones.

Una vez realizados los pasos anteriores se puede proceder a ejecutar la herramienta. Para ello, se deberá ejecutar el archivo “server.py” y proporcionar los datos necesarios que pide la herramienta:

```
python server.py
¿Qué script deseas ejecutar?
1. serverApexdomains.py
2. serverSubdomains.py
Selecciona una opción (1/2): 1
Ejecutando serverApexdomains.py...
Elija método de ejecución (1: corto, 2: medio, 3: largo): 3
Encontrar subnets (0) / Usar redes definidas en files/ips.txt (1): 0
Nombre de la empresa: booking
```

Fig 47. Ejecución inicial de la herramienta para el enfoque basado en Open Scope

En este caso se ha seleccionado el modo “serverApexdomains.py” que es el equivalente al modo enfocado en *Open Scope*. Además, se ha elegido el método de ejecución largo, explicado en los apartados anteriores. Finalmente, se ha especificado que la herramienta encuentre las redes de la organización automáticamente a través de la cadena “booking”.

Cuando el *backend* acabe de ejecutarse se verá en la terminal de la siguiente forma:

```
└─# python server.py e/ /BugBounty/Tools/BOT/final_bot
¿Qué script deseas ejecutar?
1. serverApexdomains.py
2. serverSubdomains.py
Selecciona una opción (1/2): 1
Ejecutando serverApexdomains.py...
Elija método de ejecución (1: corto, 2: medio, 3: largo): 3
Encontrar subnets (0) / Usar redes definidas en files/ips.txt (1): 0
Nombre de la empresa: booking
Buscando nets...
Subnets encontradas!
Ejecutando masscan...
Recopilando certificados TLS...
Ejecutando Smap...
Recopilando subdominios de forma pasiva con assetfinder...
Resolviendo el DNS de los subdominios recopilados...
Enumerando posibles vhosts de manera pasiva...
Recopilando información de los puertos web...
Borrando la base de datos anterior...
Datos guardados en la base de datos y archivo JSON.
Tiempo de ejecución: 8454.08 segundos
serverApexdomains.py se ejecutó correctamente.
```

Fig 48. Ejecución final de la herramienta para el enfoque basado en *Open Scope*

Open Scope – Redes proporcionadas por el analista

Este modo es útil cuando no se quiere automatizar la búsqueda de redes, ya sea porque el analista no ha sido capaz de encontrar la cadena adecuada o porque prefiere proporcionar las redes manualmente. Para ello, el analista deberá poner las redes a analizar en el archivo “files/ips.txt”:

```

└─# cat files/ips.txt
37.10.0.0/18
5.57.16.0/21
103.75.176.0/22
185.150.60.0/22
185.28.220.0/22
103.177.38.0/23
91.206.232.0/23
103.38.238.0/23
91.195.236.0/23
94.188.197.0/24

```

Fig 49. Ejemplo del archivo “files/ips.txt” con redes proporcionadas por el analista

En cuanto al uso de la clase *Masscan* se deberá definir la interfaz de red a usar en el mismo constructor de la clase, tal y como se ha explicado en el subapartado anterior.

La ejecución del *backend* se verá de la siguiente forma:

```

└─# python server.py
¿Qué script deseas ejecutar?
1. serverApexdomains.py
2. serverSubdomains.py
Selecciona una opción (1/2): 1
Ejecutando serverApexdomains.py...
Elija método de ejecución (1: corto, 2: medio, 3: largo): 1
Encontrar subnets (0) / Usar redes definidas en files/ips.txt (1): 1
Ejecutando masscan...

```

Fig 50. Ejecución inicial de la herramienta para el enfoque basado en Open Scope proporcionando las redes directamente

Wildcard Domains

Para ejecutar este modo no se necesitará modificar el código pero sí será necesario proporcionar los dominios a enumerar. Para proporcionarlos, se deberá poner un dominio en cada línea del archivo “files/subdomains.txt”:

```

└─# cat files/subdomains.txt
fareharbor.engineering
rentalcars.com
fareharbor.com
booking.com

```

Fig 51. Ejemplo archivo “files/subdomains.txt” para el enfoque basado en Wildcard Domains

Una vez realizado este paso, ya se puede ejecutar el *backend* proporcionando los *inputs* necesarios:

```

└─# python server.py
¿Qué script deseas ejecutar?
1. serverApexdomains.py
2. serverSubdomains.py
Selecciona una opción (1/2): 2
Ejecutando serverSubdomains.py...
Elija método de ejecución (1: corto, 2: largo): 1

```

Fig 52. Ejecución inicial de la herramienta para el enfoque basado en Wildcard Domains

El resultado final cuando se acabe de ejecutar el *backend* se verá de la siguiente forma:

```

└─# python server.py
¿Qué script deseas ejecutar?
1. serverApexdomains.py - notauthorized.booking.com
2. serverSubdomains.py
Selecciona una opción (1/2): 2
Ejecutando serverSubdomains.py...
Elija método de ejecución (1: corto, 2: largo): 1
Recopilando subdominios de forma pasiva...
Resolviendo el DNS de los subdominios recopilados...
Recopilando certificados TLS...
Ejecutando Smap...
Recopilando información de los puertos web...
Borrando la base de datos anterior...
Datos guardados en la base de datos y archivo JSON.
Tiempo de ejecución: 479.44 segundos
serverSubdomains.py se ejecutó correctamente.

```

Fig 53. Ejecución final de la herramienta para el enfoque basado en Wildcard Domains

6.6. Uso práctico de la herramienta

En esta sección, se hablará sobre cómo se puede utilizar la herramienta, centrándonos en su aplicación dentro del programa de *Bug Bounty* de *Booking*. El *frontend* permite dos enfoques principales, el primero se basa en hacer un análisis detallado, respuesta por respuesta, y el segundo se basa en realizar una búsqueda más específica utilizando filtros por palabras clave. Cada método tiene sus propias ventajas, dependiendo de lo que busque el analista. Personalmente, creo que lo mejor es combinar ambos para hacer el proceso más eficiente. Por ejemplo, se podría empezar filtrando las respuestas con palabras clave relevantes y luego hacer un análisis más profundo de todas las respuestas interesantes recopiladas. De esta forma, la flexibilidad de la herramienta permite que cada analista puede ajustarla a sus propias necesidades y preferencias. A continuación, se verán ejemplos para cada enfoque explicado.

Modo respuesta por respuesta

El análisis respuesta por respuesta es el método más simple, debido a que se basa en examinar cada respuesta en el orden en el que se han recopilado. De esta forma, el analista acabará analizando cada servicio encontrado. Al ingresar en la interfaz web se presenta una lista de servidores recopilados después de haber ejecutado el *backend*. Este enfoque permite al analista auditar de forma sistemática cada resultado, inspeccionando detalles como configuraciones, posibles vulnerabilidades o cualquier anomalía que se pueda encontrar. La gran ventaja de este método es que asegura que no se pase por alto ningún servicio expuesto, al menos de los enumerados. Por ejemplo, al acceder a la interfaz web, la primera respuesta que encontramos es la siguiente:

5.57.16.196	
title:	Booking.com Official site The best hotels, flights, car rentals & accommodations
status_code:	200
request:	https://booking.com:443
redirected_url:	https://www.booking.com/
port:	443
response_text:	Skip to main content USD List your property Register Sign in Stays Flights Flight + Hotel Car rentals Cruises Attractions Airport taxis More Find deals for any season From cozy bed & breakfasts to luxury hotels Check-in date — Check-out date 2 adults · 0 children · 1 room Search I'm looking for flights Offers Promotions, deals, and special offers for you Save on stays worldwide Start your year with an adventure, saving 15% or more with Early 2025 Deals. Save 15% or more Travel more, spend less Sign in, save money Save 10% or more at participating properties - just look for the blue Genius label Sign in Register Get inspiration for your next trip More 5 of the best hotels in Los Angeles From Hollywood to Beverly Hills discover 5 of the best hotels in Los Angeles for your stay The 6 best Orlando hotels for families Discover the best Orlando hotels for families for your vacation. 5 best ski towns around the world Discover a winter wonderland in these charming ski destinations 5 vacation homes for a Thanksgiving getaway Enjoy Thanksgiving dinner at these vacation homes. 6 incredible Bangkok rooftop bars Amazing city views, cocktails, and world-class cuisine. Trending destinations Most popular choices for travelers from the United States Orlando Las Vegas New York Tokyo Atlanta Popular with travelers from the United States Domestic cities International cities Regions Countries Places to stay Las Vegas hotels Nevada New York hotels New York San Diego hotels California Orlando hotels Florida Los Angeles hotels California Myrtle Beach hotels South Carolina Washington, D.C. hotels District Of Columbia Atlanta hotels Georgia Ocean City hotels Maryland San Francisco hotels California Pigeon Forge hotels Tennessee Chicago hotels Illinois San Antonio hotels Texas Virginia Beach hotels Virginia Houston hotels Texas Boston hotels Massachusetts New Orleans
response_headers:	{ "Content-Type": "text/html; charset=UTF-8", "Content-Length": "160759", "Connection": "keep-alive", "Server": "nginx", "Date": "Tue, 11 Mar 2025 12:16:14 GMT", "Cache-Control": "private", "Vary": "User-Agent, Accept-Encoding", "Content-Encoding": "br", "nel": "{ \"max_age\": 604800, \"report_to\": \"\\\"default\\\"\" }\", \"report-to\": \"\\\"default\\\"\" }\", \"group\": \"\\\"default\\\"\", \"max_age\": 604800\", \"Set-Cookie\": \"bknng=11UmFuZG93SVYkc2Rllyh9Yaa29%2F3xUOLbXpFeYC4TUH8QaaLhV81ZEY0HgTa0gkOurxn3oWcmQUmkhyic2Kxc0OEUFAGLPopqT80x%2BbTqyV82%2BX2BAY0NSKYL1PZnSp3%2BStHNVECI9w6VbOucR4wjOWYu1ml9lnq11HADC%2BqkK%2FUBW%2Bw%2FRnYjrULuF3YXH3dR

Fig 54. Primera respuesta al acceder a la interfaz web

A partir de este momento, el auditor tomará la decisión de si auditar o no el dominio que se ha enumerado. En un programa de *Bug Bounty*, como el de *Booking*, el dominio principal suele ser el más examinado por la comunidad, por lo que puede ser difícil encontrar nuevas vulnerabilidades. En este escenario, el analista podría decidir avanzar a la siguiente respuesta en la lista, considerando si se trata de un dominio más inusual, lo que podría aumentar las posibilidades de detectar fallos. Así, el auditor tiene la flexibilidad de revisar las respuestas en el orden que se presentan y decidir cuáles le conviene auditar, basándose en su propio criterio y experiencia.

Modo filtrado por palabras clave

El segundo modo de uso implica filtrar las respuestas utilizando palabras clave. Por ejemplo, el auditor podría usar la palabra “api” para restringir los resultados a aquellos que están relacionados con *endpoints* de este tipo. Este modo permite al analista enfocarse rápidamente en áreas específicas, haciendo que el proceso de análisis sea más ágil y adaptado a sus conocimientos o prioridades. Por ejemplo, si buscamos la cadena “api”:

5.57.17.250	
title:	Booking.com: 404 Not Found
status_code:	404
request:	https://api.booking.com:443
redirected_url:	
port:	443
response_text:	404 Not Found I see no / here.
response_headers:	{ "Content-Type": "text/html; charset=utf-8", "Transfer-Encoding": "chunked", "Connection": "keep-alive", "Server": "nginx", "Date": "Tue, 11 Mar 2025 12:16:13 GMT", "x-xss-protection": "1; mode=block", "Vary": "Accept-Encoding", "strict-transport-security": "max-age=63072000; includeSubDomains; preload", "Content-Encoding": "gzip", "X-Cache": "Error from cloudfront", "Via": "1.1 3c2c38b11de7f29e091125f84ca68d28.cloudfront.net (CloudFront)", "X-Amz-CF-Pop": "MUC50-P4", "X-Amz-CF-Id": "UBw8P5RGVZBclCQkTHmSiwXFuipE1dLzO1u02_2n1tj8Z75Stp_kQ==" }

Fig 55. Salida de la búsqueda “api” en la barra de búsqueda

O por ejemplo, si se desea auditar servidores IIS:

94.188.197.78	
title:	IIS Windows Server
status_code:	200
request:	https://94.188.197.78:443
redirected_url:	
port:	443
response_text:	
response_headers:	{ "Content-Type": "text/html", "Last-Modified": "Tue, 20 Nov 2018 06:51:30 GMT", "Accept-Ranges": "bytes", "Etag": "\"13f075779d80d410\"", "Server": "Microsoft-IIS/10.0", "X-Powered-By": "ASP.NET", "Date": "Tue, 11 Mar 2025 12:16:12 GMT", "Content-Length": "703" }

Fig 56. Salida de la búsqueda “IIS” en la barra de búsqueda

En este caso, dado que la solicitud se ha hecho directamente a una IP, el auditor necesita asegurarse de que esa IP realmente pertenece a la empresa que está siendo auditada, en este caso, *Booking*. A veces, esta verificación se puede hacer revisando los *headers* de la respuesta, que podrían dar pistas sobre quién es el propietario del servidor. Pero si esa información no es suficiente o no está disponible, hay herramientas externas que pueden ayudar, como *ipinfo.io* o *Shodan*. Por ejemplo, al buscar la IP en *Shodan*, se puede obtener la siguiente información:

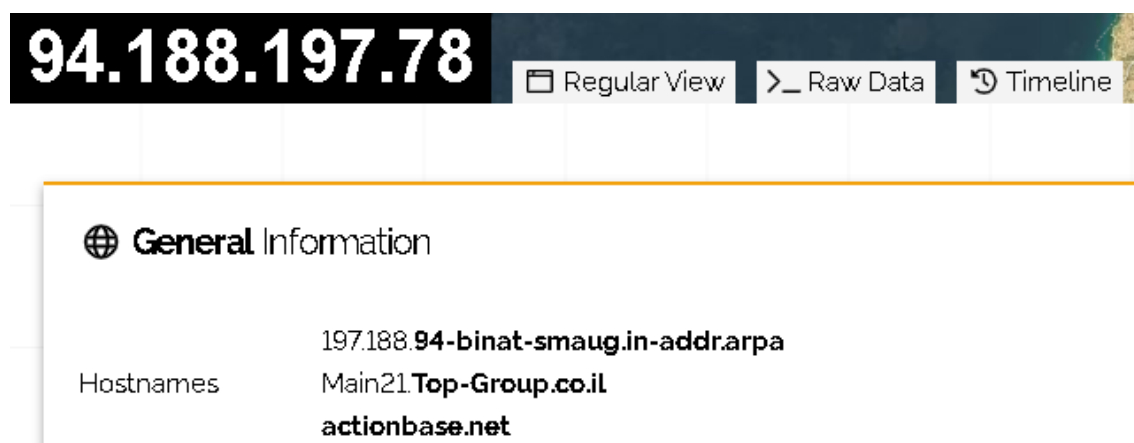


Fig 57. Salida de una búsqueda por IP en Shodan

A partir de esta información, el auditor tiene la oportunidad de decidir si se trata de un falso positivo o si el dominio *actionbase.net* tiene alguna conexión con *Booking*. Para ello, puede utilizar la interfaz web de la herramienta, revisando si el término “actionbase” aparece en otros servidores o si la misma IP muestra respuestas vinculadas a otros dominios. Este análisis ayuda al usuario a confirmar que el servidor encontrado forma parte de la organización objetivo.

Como hemos podido ver, la herramienta realmente cumple con su función principal, la cual es servir como una guía práctica para auditar los servidores web de una organización. Hace que sea más fácil identificar rápidamente los *hosts* que son más útiles para el analista, dependiendo de cada situación particular, lo que optimiza la auditoría y se adapta a las necesidades del analista.

7. Conclusiones

En este trabajo se ha creado una herramienta que ayuda a identificar los servidores web activos de una empresa, utilizando tanto el nombre de la empresa como sus dominios para lograrlo. En esta sección, se comentan las conclusiones que han surgido a lo largo del proceso.

El objetivo principal se ha alcanzado gracias a la implementación de diversas metodologías diseñadas para programas de *Bug Bounty*, como *Open Scope* y *Wildcard Domains*. De esta manera, la herramienta se ha creado para ser útil en ambos tipos de programas, logrando identificar activos de forma automatizada en cada situación y adaptándose a sus necesidades particulares.

La herramienta recopila información de puertos web y subdominios de forma eficiente, integrando servicios como *crt.sh*, *merklemap.com* y *Shodan*, además de herramientas como *subfinder*, *assetfinder* y *certgraph*. Los resultados se guardan en una base de datos *SQLite* y se muestran en una interfaz web sencilla creada con *Flask*, lo que facilita al analista revisarlos y trabajar con ellos.

Durante el desarrollo, me encontré con varios contratiempos, especialmente con algunos casos que no había previsto al principio en las diferentes clases. Estos problemas aparecieron gracias a las pruebas unitarias y a la metodología cíclica que utilicé. Por ejemplo, tuve dificultades con falsos positivos al listar algunos *virtual hosts*, dificultades con la programación concurrente y con algunos resultados duplicados al ejecutar distintas clases.

En cuanto a la contribución de este trabajo al área de investigación, pienso que la herramienta realmente se destaca al automatizar el proceso de descubrimiento de servidores dentro de una organización, presentando los resultados de una manera visual y práctica que resulta muy útil para el analista. Además, emplea técnicas pasivas para listar subdominios y *virtual hosts*, y ofrece al usuario la posibilidad de elegir entre diferentes modos de ejecución según sus preferencias, ya sea que busque más información, mayor certeza en los datos o simplemente un menor tiempo de ejecución. A su vez, la interfaz web permite revisar rápidamente las cabeceras HTTP e identificar servidores interesantes para auditar sin complicaciones.

En cuanto a las limitaciones, hay dos puntos importantes a resaltar. En primer lugar, el tiempo de ejecución. Para aprovechar al máximo la herramienta, el analista debería dejarla funcionando con suficiente antelación (por ejemplo, ejecutándola durante la noche), ya que el proceso puede tardar dependiendo del tamaño de la empresa que se esté analizando, la potencia del equipo y la velocidad de la red. Aunque puede parecer que el tiempo es excesivo, hacerlo manualmente sería mucho más complicado y podría afectar el trabajo del analista. La segunda limitación está relacionada con los falsos positivos, especialmente al listar *virtual hosts* en servicios CDN, dependiendo del modo de ejecución que se seleccione. Aun así, la interfaz web proporciona información detallada que ayuda al analista a identificar estos casos rápidamente y descartarlos.

Para el futuro, hay varias mejoras a considerar. Por ejemplo, en el *frontend* se podrían implementar búsquedas avanzadas utilizando operadores en la base de datos, algo similar a lo que hace *Shodan*. Además, sería interesante integrar nuevas herramientas y servicios que estén surgiendo en este campo.

En resumen, este TFM establece las bases para continuar la investigación en la enumeración de activos web, abriendo la puerta a herramientas más completas y accesibles que faciliten el análisis de infraestructuras digitales.

8. Bibliografía

1. HackerOne. (s.f.). What are bug bounties and how do they work? <https://www.hackerone.com/blog/what-are-bug-bounties-and-how-do-they-work>
2. Bugcrowd. (2023). Understanding bug bounty scope [Datasheet]. <https://www.bugcrowd.com/wp-content/uploads/2023/11/Understanding-Bug-Bounty-Scope-Datasheet.pdf>
3. Shodan. (s.f.). What is Shodan? <https://help.shodan.io/the-basics/what-is-shodan>
4. ProjectDiscovery. (s.f.). ProjectDiscovery [Repositorio de GitHub]. <https://github.com/projectdiscovery>
5. BizTalk360. (s.f.). Web endpoint monitoring with BizTalk360. <https://www.biztalk360.com/blog/web-endpoint-monitoring-biztalk360/>
6. Shodan. (s.f.). Search examples. <https://www.shodan.io/search/examples>
7. Shodan. (s.f.). On-demand scanning. <https://help.shodan.io/the-basics/on-demand-scanning>
8. INCIBE. (s.f.). Google Dorks: Te ayuda a encontrar información sobre ti en la red. <https://www.incibe.es/ciudadania/blog/google-dorks-te-ayuda-encontrar-informacion-sobre-ti-en-la-red>
9. ProjectDiscovery. (s.f.). Subfinder overview. <https://docs.projectdiscovery.io/tools/subfinder/overview>
10. ProjectDiscovery. (s.f.). HTTPX overview. <https://docs.projectdiscovery.io/tools/httpx/overview>
11. ProjectDiscovery. (s.f.). Katana overview. <https://docs.projectdiscovery.io/tools/katana/overview>
12. ProjectDiscovery. (s.f.). Nuclei overview. <https://docs.projectdiscovery.io/tools/nuclei/overview>
13. ProjectDiscovery. (s.f.). Naabu overview. <https://docs.projectdiscovery.io/tools/naabu/overview>

14. Cisco. (s.f.). What is a firewall? <https://www.cisco.com/site/us/en/learn/topics/security/what-is-a-firewall.html>
15. The Hacker Recipes. (s.f.). Virtual host fuzzing. <https://www.thehacker.recipes/web/recon/virtual-host-fuzzing>
16. TechTarget. (s.f.). Object-oriented programming (OOP). <https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP>
17. Wikipedia. (s.f.). Prueba unitaria. https://es.wikipedia.org/wiki/Prueba_unitaria
18. Código Facilito. (s.f.). Programación concurrente: Qué es y cómo funciona. <https://codigofacilito.com/articulos/programacion-concurrente>
19. Datadog. (s.f.). DNS resolution. <https://www.datadoghq.com/knowledge-center/dns-resolution/>
20. Secop Solution. (s.f.). Bug hunting: Safeguarding the digital frontier. <https://www.secopsolution.com/blog/bug-hunting-safeguarding-the-digital-frontier>
21. Asana. (s.f.). Metodología de gestión de proyectos en cascada. <https://asana.com/es/resources/waterfall-project-management-methodology>
22. CareerFoundry. (s.f.). What's the difference between frontend and backend? <https://careerfoundry.com/en/blog/web-development/whats-the-difference-between-frontend-and-backend/>
23. Uptrends. (s.f.). What is IPv4? <https://www.uptrends.com/what-is/ipv4>
24. Cloudflare. (s.f.). ¿Qué es un sistema autónomo? <https://www.cloudflare.com/es-es/learning/network-layer/what-is-an-autonomous-system/>
25. ParseHub. (s.f.). What is web scraping? <https://www.parsehub.com/blog/what-is-web-scraping/>
26. Wikipedia. (s.f.). Regular expression. https://en.wikipedia.org/wiki/Regular_expression
27. Graham, R. D. (s.f.). Masscan [Repositorio de GitHub]. <https://github.com/robertdavidgraham/masscan>

28. DigiCert. (s.f.). TLS/SSL certificates. <https://www.digicert.com/tls-ssl/tls-ssl-certificates>
29. S0md3v. (s.f.). Smap [Repositorio de GitHub]. <https://github.com/s0md3v/Smap>
30. Nmap Project. (s.f.). Nmap: Network exploration tool and security/port scanner. <https://nmap.org/>
31. Tomnomnom. (s.f.). Assetfinder [Repositorio de GitHub]. <https://github.com/tomnomnom/assetfinder>
32. Lanrat. (s.f.). Certgraph [Repositorio de GitHub]. <https://github.com/lanrat/certgraph>
33. GeeksforGeeks. (s.f.). Jaro and Jaro-Winkler similarity. <https://www.geeksforgeeks.org/jaro-and-jaro-winkler-similarity/>
34. IBM. (s.f.). HTTP responses. <https://www.ibm.com/docs/en/cics-ts/6.x?topic=protocol-http-responses>

Repositorio del trabajo: <https://github.com/FerriBernabe/TFM>