HACKING API: DESCUBRIMIENTO, ENUMERACIÓN ACTIVA E INGENIERÍA INVERSA

Open-Sec

Toshiro Nagata

Pentester en Open-Sec. Cuenta con múltiples certificaciones de ciberseguridad, incluyendo: Certified API Security Analyst (CASA), MCRTA, eCPPTv2, eWPT, eJPT, entre otras. Tiene experiencia en pruebas de penetración en aplicaciones web e infraestructura, tanto en redes internas como externas, y en el análisis de inteligencia de amenazas.

A medida que las aplicaciones modernas continúan evolucionando, las API (Interfaces de Programación de Aplicaciones) se han convertido en componentes esenciales para la infraestructura tecnológica. Pero, el uso de estas API, también generan riesgos significativos, es importante primero analizar cuál es la función de las API y por qué es importante su evaluación de seguridad.

Las API son conjuntos de funciones y procedimientos que facilitan la comunicación y el intercambio de datos entre aplicaciones, independientemente de sus lenguajes de programación, plataformas o entornos operativos.

Facilitan la integración de sistemas y la creación de aplicaciones más funcionales.

Sin embargo, al ser un punto de acceso a sistemas y datos, las API también son un objetivo sustancial para algunos ciber criminales. Una API insegura puede llevar a la exposición de datos sensibles, pérdida en términos financieros y daño a la reputación de una organización.

Si bien en un ataque web se tiene en cuenta la profundidad de la superficie de exposición, es importante aclarar que este término se relaciona con el nivel de detalle en el que se examinan los directorios durante las evaluaciones de seguridad.

Este concepto se puede ilustrar mediante el análisis de diferentes niveles de directorios dentro de una API, como se muestra a continuación:

- /api/v1/usuario
- /api/v2/usuario
- /api/v2/admin
- /desarrollador/usuario

En este contexto, la profundidad se define de la siguiente manera:

Nivel de profundidad	Descripción	Ejemplos
0	Representa el nivel más básico, enfocándose en el directorio raíz del sitio o la aplicación.	/

1	Abarca el primer conjunto de subdirectorios directamente bajo la raíz, proporcionando una mirada inicial a las principales categorías o áreas funcionales.	/api /desarrollador
2	Profundiza un nivel más, explorando subdirectorios adicionales para obtener una visión más detallada de las áreas específicas de la aplicación.	/api/v1 /api/v2 /desarrollador/usuario
3	Alcanza una mayor especificidad, evaluando directorios más profundos que representan funcionalidades o secciones específicas dentro de las áreas identificadas en la profundidad 2.	/api/v1/usuario /api/v2/usuario /api/v2/admin

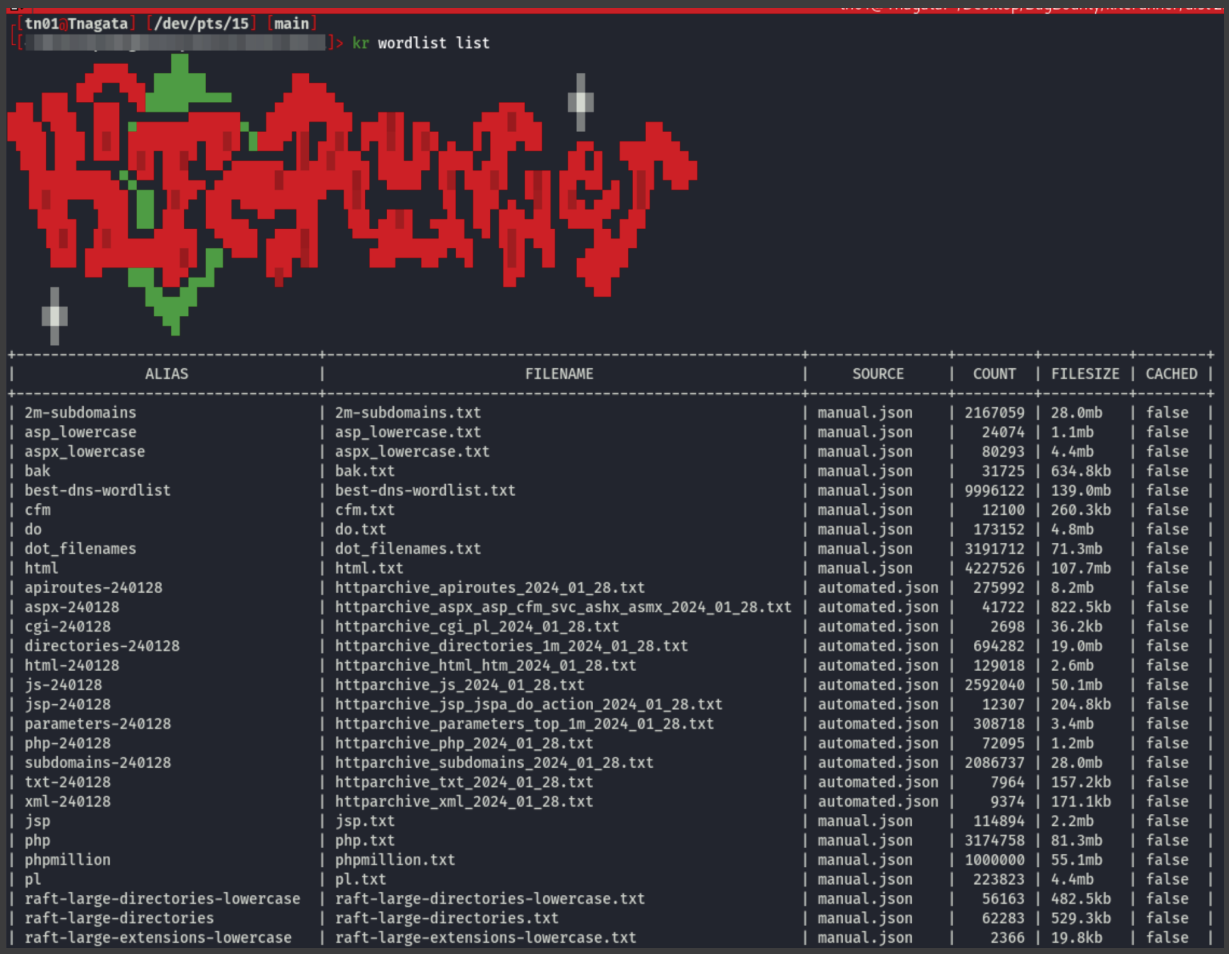
Descubrimiento y enumeración activa

Este proceso implica más que simplemente identificar los endpoints disponibles; se trata de un análisis detallado que busca revelar no solo la estructura de la API, sino también sus métodos, parámetros, y las posibles vulnerabilidades asociadas. En el escenario actual, donde las aplicaciones web se han vuelto muy dinámicas y complejas, este paso inicial es importante para establecer un entendimiento sólido de la superficie de ataque, si bien es conocido que existen distintas herramientas, algunas de las que suelen estar presentes en las evaluaciones de seguridad a las API, son:

KitRunner

Como lo describe en su repositorio oficial, el cual pude ser consultado en https://github.com/assetnote/kiterunner, esta herramienta es empleada para el descubrimiento de APIs y está diseñada para interactuar de manera inteligente con frameworks de desarrollo web actuales como Flask, Rails, Express, y Django, los cuales requieren especificaciones claras para las rutas, incluyendo métodos HTTP aceptables, encabezados requeridos, y parámetros esperados. Utilizando un enfoque basado en datos, KitRunner aprovecha un conjunto de especificaciones Swagger, recolectadas de diversas fuentes, incluidos análisis exhaustivos en internet y repositorios en GitHub. Esta base de datos le permite a KitRunner ejecutar ataques de fuerza bruta con alta precisión, enviando solicitudes que se ajustan exactamente a los requisitos específicos de cada API, revelando así puntos finales que otras herramientas podrían pasar por alto.

KitRunner permite visualizar todas las listas de palabras disponibles mediante el comando kr wordlist list, proporcionando detalles como Alias, Filename, Source, Count, Filesize, y Cached, como se observa en la siguiente imagen



ALIAS	FILENAME	SOURCE	COUNT	FILESIZE	CACHED
2m-subdomains	2m-subdomains.txt	manual.json	2167059	28.0mb	false
asp_lowercase	asp_lowercase.txt	manual.json	24074	1.1mb	false
aspx_lowercase	aspx_lowercase.txt	manual.json	80293	4.4mb	false
bak	bak.txt	manual.json	31725	634.8kb	false
best-dns-wordlist	best-dns-wordlist.txt	manual.json	9996122	139.0mb	false
cfm	cfm.txt	manual.json	12100	260.3kb	false
do	do.txt	manual.json	173152	4.8mb	false
dot_filenames	dot_filenames.txt	manual.json	3191712	71.3mb	false
html	html.txt	manual.json	4227526	107.7mb	false
apiroutes-240128	httparchive_apiroutes_2024_01_28.txt	automated.json	275992	8.2mb	false
aspx-240128	httparchive_aspx_asp_cfm_svc_ashx_asmx_2024_01_28.txt	automated.json	41722	822.5kb	false
cgi-240128	httparchive_cgi_pl_2024_01_28.txt	automated.json	2698	36.2kb	false
directories-240128	httparchive_directories_1m_2024_01_28.txt	automated.json	694282	19.0mb	false
html-240128	httparchive_html_htm_2024_01_28.txt	automated.json	129018	2.6mb	false
js-240128	httparchive_js_2024_01_28.txt	automated.json	2592040	50.1mb	false
jsp-240128	httparchive_jsp_jspa_do_action_2024_01_28.txt	automated.json	12307	204.8kb	false
parameters-240128	httparchive_parameters_top_1m_2024_01_28.txt	automated.json	308718	3.4mb	false
php-240128	httparchive_php_2024_01_28.txt	automated.json	72095	1.2mb	false
subdomains-240128	httparchive_subdomains_2024_01_28.txt	automated.json	2086737	28.0mb	false
txt-240128	httparchive_txt_2024_01_28.txt	automated.json	7964	157.2kb	false
xml-240128	httparchive_xml_2024_01_28.txt	automated.json	9374	171.1kb	false
jsp	jsp.txt	manual.json	114894	2.2mb	false
php	php.txt	manual.json	3174758	81.3mb	false
phpmillion	phpmillion.txt	manual.json	1000000	55.1mb	false
pl	pl.txt	manual.json	223823	4.4mb	false
raft-large-directories-lowercase	raft-large-directories-lowercase.txt	manual.json	56163	482.5kb	false
raft-large-directories	raft-large-directories.txt	manual.json	62283	529.3kb	false
raft-large-extensions-lowercase	raft-large-extensions-lowercase.txt	manual.json	2366	19.8kb	false

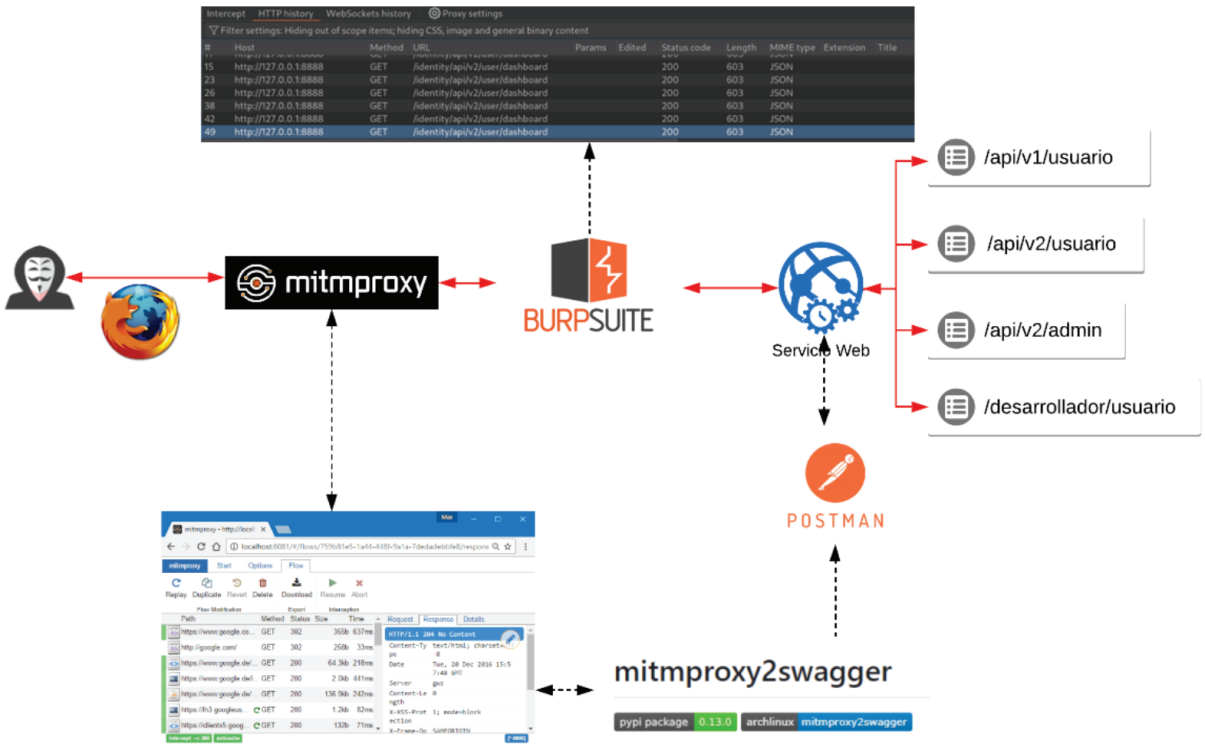
Si bien KitRunner puede ser de mucha ayuda, cabe resaltar que no se centra exclusivamente en la fuerza bruta como su principal funcionalidad; más bien, está diseñado para realizar escaneos inteligentes de APIs y web aplicando conocimientos predefinidos sobre estructuras de API comunes, lo que lo diferencia de herramientas como ffuf, wfuzz o dirb, que se enfocan más en la enumeración por fuerza bruta de archivos y directorios en servidores web.

Ingeniería inversa

Es importante destacar que las técnicas y procedimientos descritos en este documento están enfocadas en APIs de tipo RESTful. En el caso de otros tipos de APIs, como SOAP o GraphQL, las TTPs (Tácticas, Técnicas y Procedimientos) pueden variar significativamente. Por ejemplo, las APIs SOAP a menudo utilizan mensajes XML y tienen un conjunto diferente de vulnerabilidades y técnicas de ataque comparadas con las APIs RESTful. De igual manera, las APIs GraphQL presentan desafíos únicos debido a su capacidad de permitir consultas más complejas y personalizadas. Por lo tanto, es fundamental adaptar las estrategias de análisis y ataque según el tipo específico de API que se esté evaluando.

Si bien se podrían usar las herramientas mencionadas para realizar fuerza bruta, es importante resaltar que en muchas ocasiones es importante tener un mayor control sobre la manera en la que se realiza la enumeración, ya sea por seguridad del propio servicio; es decir para evitar una posible denegación de servicio (DOS), o porque se encuentre realizando un ejercicio de Red Team en el cual desea ser sigiloso. Para ello, se puede realizar el análisis del archivo Swagger o la documentación de la API, pero hay ocasiones en las que esto no es posible, por lo cual una alternativa podría ser crear un archivo Swagger (según nuestro recorrido por el flujo de las APIs).

A continuación se observa una imagen con el flujo de herramientas propuesto.



Algunas ventajas al emplear este flujo son:

Mejor Comprensión y Documentación con la Generación de Swagger: Al interceptar el tráfico con mitmproxy (y por extensión mitmweb) y luego procesarlo con herramientas como mitmproxy2swagger, puedes generar documentación de API (Swagger) a partir del tráfico observado. Esto es útil en escenarios donde la documentación de la API podría no estar disponible, facilitando un entendimiento profundo de la API para pruebas más dirigidas y efectivas.

Funcionalidades Avanzadas de Análisis y Ataque con Burp Suite: Después de pasar por mitmweb, el tráfico es analizado y manipulado adicionalmente por Burp Suite, que ofrece un conjunto de herramientas más robustas y avanzadas para el análisis de seguridad, incluyendo escaneo de vulnerabilidades, fuzzing, y otras técnicas de pentesting.

WEBGRAFÍA

- https://owasp.org/API-Security/editions/2023/en/0x00-header/
- https://portswigger.net/web-security/api-testing
- https://securityboulevard.com/2022/08/the-beginners-guide-to-api-hacking/
- https://medium.com/@windsorheightsbookfair/hacking-apis-by-cory-j-ball-book-review-baa5a9486565
- https://nordicapis.com/5-ways-to-hack-an-api-and-how-to-defend/
- https://www.wallarm.com/what/how-to-hack-api-in-60-minutes-with-open-source
- https://zerodayhacker.com/understanding-query-parameters-and-path-variables-in-postman/

Open-Sec

SIMPLIFY, SECURE, ACCELERATE

info@open-sec.com
+1 (561) 600-0818
www.open-sec.com

Open-Sec @OpenSec @OpenSec @open53c